



---

Machine and deep learning applications for improving  
the measurement of key indicators for financial  
institutions: stock market volatility and general  
insurance reserving risk

---

PHD. THESIS

April 2022

Author: Eduardo Ramos-Pérez  
Thesis Tutor I: Pablo J. Alonso-González  
Thesis Tutor II: José Javier Núñez-Velázquez



*“e quindi uscimmo a riveder le stelle”*  
*Inferno, Dante Alighieri*

*To my girlfriend, parents and grandmother*



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Economic and statistical framework</b>	<b>3</b>
2.1	Aftermath of recent financial crises and thesis objectives . . . . .	3
2.2	Machine and deep learning: Background and methods applied . . . . .	7
2.3	Stock market volatility and machine learning . . . . .	23
2.4	Reserving in general insurance and machine learning . . . . .	27
<b>3</b>	<b>Forecasting volatility with a stacked model based on a hybridized Artificial Neural Network</b>	<b>31</b>
3.1	Introduction . . . . .	32
3.2	Benchmark models, risk measurements and statistical tests . . . . .	35
3.3	Stacked model . . . . .	37
3.4	Results . . . . .	41
3.5	Conclusions . . . . .	46
<b>4</b>	<b>Stochastic reserving with a stacked model based on a hybridized Artificial Neural Network</b>	<b>49</b>
4.1	Introduction . . . . .	50
4.2	Benchmark models and validation . . . . .	54
4.3	Stochastic reserving model based on the stacking algorithm approach .	56
4.4	Results . . . . .	63
4.5	Conclusions . . . . .	70
<b>5</b>	<b>Multi-Transformer: A new neural network-based architecture for forecasting S&amp;P volatility</b>	<b>72</b>
5.1	Introduction . . . . .	73
5.2	Materials and Methods . . . . .	76
5.3	Results . . . . .	86
5.4	Discussion . . . . .	90
5.5	Conclusion . . . . .	91
<b>6</b>	<b>Mack-Net model: Blending Mack's model with Recurrent Neural Networks</b>	<b>93</b>
6.1	Introduction . . . . .	94
6.2	Benchmark model and validation metrics . . . . .	97
6.3	Data and Mack-Net architecture . . . . .	99
6.4	Model fitting and results . . . . .	106
6.5	Conclusions . . . . .	113
<b>7</b>	<b>Conclusions</b>	<b>115</b>
7.1	Main Findings . . . . .	115
7.2	Further research . . . . .	120

<b>8</b>	<b>Annexes</b>	<b>122</b>
8.1	Annex I. Published Paper. Forecasting volatility with a stacked model based on a hybridized Artificial Neural Network . . . . .	123
8.2	Annex II. Published Paper. Stochastic reserving with a stacked model based on a hybridized Artificial Neural Network . . . . .	133
8.3	Annex III. Published Paper. Multi-Transformer: A new neural network- based architecture for forecasting S&P volatility . . . . .	146
8.4	Annex IV. Published Paper. Mack-Net model: Blending Mack's model with Recurrent Neural Networks . . . . .	165
	<b>References</b>	<b>175</b>

## Preface

It is sometimes hard for me to believe how much this thesis has changed over the last years. I did not know anything regarding the peculiarities of the academic or research field when I started this dissertation. The tutors of this thesis encouraged me to produce papers for some of the most relevant scientific journals. They also taught me how to prepare the information for the journals. At the time of writing, three papers have been published in D1 journals and one additional article is in the reviewing phase of a D1 journal. I would like to thank the tutors of this thesis for believing that this research deserves to be published and teaching me how to make it possible.

Producing this dissertation did not only increase my knowledge of the academic field, deep learning techniques and financial risk models, but it also made me progress in many other fields that are not directly related to this research. The scientific method and the critical thinking have had a huge impact on how I see the world and they have deeply changed my understanding process. I really think this dissertation made me grow in so many fields over the last years.

The process was enriching and beautiful but also tough for me and, above all, my beloved ones. Working in the private sector for long hours and producing this dissertation at the same time was not easy. During the last years, most of my free time on holidays and weekends was spent on this research. Although this was not my intention when I started the doctoral program, the burden of working in the private sector and researching at the same time was not only carried by me but also by my beloved ones. I greatly apologize for that. I would like to thank my girlfriend and parents for all the selfless support and love you gave me during this period. Without you this would not be possible. Giving this selfless love has more merit than producing any dissertation.

The present PhD thesis, ‘Machine and deep learning applications for improving the measurement of key indicators for financial institutions: stock market volatility and general insurance reserving risk’, has been conducted with the collaboration of the Universidad de Alcalá de Henares and, particularly, with the support of the professors related to the doctorate program in Economics and Business Management. This thesis has been supervised by the professors Pablo J. Alonso-González and José Javier Núñez-Velázquez.

# 1 Introduction

Over the last years, this thesis has evolved from a research focused on applying existing machine learning models for stock valuation purposes, to the creation of deep learning architectures and risk models for predicting some of the key magnitudes for financial institutions, such as the best estimate of liabilities, reserving risk, stock market volatility or equity risk. After the Financial Crisis of 2007-2008, these variables have become remarkably relevant. Since then, regulators have enhanced the risk management framework of financial institutions with laws such as Basel III, Solvency II or Swiss Solvency Test. In addition, companies have embedded a stronger risk management framework in their regular management process and they have integrated risk assessment and monitoring activities in their internal control systems thanks to lessons learned during this crisis. Nowadays, indicators such as Solvency Ratio or Return on Risk Capital have a significant impact on dividends and market valuation of financial institutions. Therefore, risk models are specially relevant in the current environment due to their importance in the decision making process of these companies. As demonstrated during the Financial Crisis of 2007-2008, an appropriate risk management strategy and accurate risk models can lead to significant competitive advantages in the financial sector.

Section 2 presents the aftermath of recent financial crises and the thesis objectives. The risk models introduced by this dissertation have the aim of improving the performance of traditional methodologies thanks to the predictive power of machine and deep learning techniques. Therefore, the background and theoretical framework of the main machine and deep learning methods used are also explained. Finally, the main existing reserving and stock volatility forecasting models are presented at the end of this section.

This thesis has the aim of applying deep and machine learning to improve the assessment of risks related to insurance and financial markets. The models introduced by this research are focused on the estimation of S&P 500 volatility, equity risk and general insurance reserving risk. Four different papers collect the risk models and deep learning architectures introduced by this thesis. As articles are organized by publication date and not by topic, Section 3 and 5 present the papers focused on stock volatility forecasting and equity risk, while Section 4 and 6 show the research related to general insurance reserving and reserving risk.

The stock volatility forecasting model of Section 3 is based on stacking. This methodology is applied to some of the most widely used machine learning algorithms to analyse structured data. The first level of the model architecture is composed of Random Forest (RF) (Breiman 2001), Gradient Boosting with regression trees (GB) (Friedman 2000) and Support Vector Machine (Cortes and Vapnik 1995). A feed forward ANN combines the previous algorithms to obtain the final volatility forecast.

Section 5 presents a novel neural network-based architecture called Multi-Transformer. This layer is a variant of Transformer models (Vaswani et al. 2017), which have al-

ready been successfully applied in the field of natural language processing. Indeed, the most popular and accurate models in this field such as BERT (Devlin et al. 2018) or GPT-3 (Brown et al. 2020) are based on this type of layers. In this paper, Artificial Neural Networks (ANNs) composed of Multi-Transformer and Transformer layers are combined with traditional auto-regressive models in order to forecast the volatility and assess the equity risk.

As previously stated, Section 4 and 6 presents two different models for calculating general insurance reserves and reserving risk. The reserving model proposed in Section 4 uses a feed forward ANN to merge the claim cost predictions made by the Chain Ladder methodology, Changing Settlement Rate model (CSR), GB, RF and ANN. It is worth mentioning that CSR is a Bayesian Markov Chain Monte-Carlo reserving model introduced by Meyers (2015). The approach proposed in this paper is combined with a log-normal distribution to obtain the full reserve distribution. On the other hand, the reserving model presented in Section 6 blends Mack (1993) model with Long-Short Term Memory (LSTM) cells in order to increase the predictive power of traditional Mack's methodology. In this case, no assumption with regard to the underlying distribution of payments is taken by the proposed model. Traditional Mack approach and the methodology presented in Section 6 use only the first two moments to produce the full reserve distribution.

## 2 Economic and statistical framework

This section is divided in four different subsections. The aftermath of the recent financial crises and the thesis objectives are presented in Section 2.1. The evolution of machine learning and the methods used in the volatility and reserving models proposed by this thesis are explained in Section 2.2. The last two subsections are focused on summarizing the existing methodologies in the field of stock volatility forecasting and general insurance reserving.

### 2.1 Aftermath of recent financial crises and thesis objectives

The Financial Crisis of 2007-2008 is probably one of the most relevant shocks for financial institutions in the last years. In fact, the causes and consequences have been deeply discussed in the literature (Williams 2010, Singh 2010, Temin 2010 and Duffie 2019, among others). This recession began with the bursting of the housing bubble in the United States. The major causes of this bubble and the following financial crisis were the excessive private debt levels, the shadow banking system, the U.S. housing policies and the rise of some financial products such as credit default swaps, mortgage backed securities and collateralized debt obligations.

Federal Reserve reduced interest rates and eased credit availability after the market crash of the dot-com bubble in 2000. This policy had the aim of reducing the negative impacts of the economic slowdown provoked by this market crash. Private and household debt increased sharply thanks to the low interest rate environment. As a significant amount of this debt was used to buy houses, this policy had a positive impact on housing prices. It is worth mentioning that a high level of private debt has a negative impact on the default probabilities of every economic agent. The higher the overall private debt in the economy, the more probable that one default leads to other defaults. During the Financial Crisis of 2007-2008, homeowners stopped paying their mortgages and, thus, the market value of mortgage backed securities and collateralized debt obligations decreased sharply. This led to the bankrupt of Lehman Brothers. Bear Stearns and Merrill Lynch were sold at sale price and Goldman Sachs and Morgan Stanley became commercial banks in order to have access to the credit issued by the Federal Reserve.

The U.S. shadow banking system also played a key role in the Financial Crisis of 2007-2008. The term shadow banking refers to non-financial intermediaries that are outside the banking regulations, but nevertheless they provide services similar to commercial banks. During the years previous to the bursting of the housing bubble, these entities conducted a significant amount of operations with financial derivatives such as mortgage backed securities, credit default swaps or collateralized debt obligations. The market size of these products and the weight of the shadow banking system increased significantly between the dot-com bubble and the Financial Crisis of 2007-2008. The high level of leverage in financial markets and the rise of the dependencies between the traditional financial institutions and the shadow banking

system aggravated the consequences of mortgages defaults.

Another relevant cause of this crisis was the government policies related to the housing market. The United States Department of Housing and Urban Development eased the conditions of mortgages. Freddie Mac (Federal Home Loan Mortgage Corporation) and Fannie Mae (Federal National Mortgage Association) received funds to purchase risky mortgages and loans. This government policy provoked an increase of the housing price and homeowners debt. As the balance sheets of Fannie Mae and Freddie Mac had a significant amount of subprime mortgages or financial products related to this type of debt, the Federal Housing Finance Agency announced on 2008 that these two institutions were placed into the conservatorship of this Agency.

The Financial Crisis of 2007-2008 had a significant impact on the U.S. economy. Although the recession lasted from the end of 2007 until June 2009, the pre-crisis levels of many key economic indicators were not reached until 2016. For example, the real GDP, the household net worth and the unemployment rate did not regained the pre-recession levels until 2011, 2012 and 2016 respectively. As individuals and businesses needed more time to pay-back the high levels of debt, the recovery was slower than in other crisis where the debt level was lower.

The U.S. government took several actions to reduce the impact of this recession. The Emergency Economic Stabilization Act was passed in 2008. This law included the Troubled Asset Relief Program (TARP), whose aim was to purchase illiquid and toxic assets from banks and other financial institutions. TARP increased the liquidity of the derivatives products related to the housing market such as mortgaged backed securities or collateralized debt obligations. On February 2009, the American Recovery and Reinvestment Act was approved. The stimulus package of this law included both, spending and tax cuts. It is also worth mentioning that the Federal Reserve reduced interest rates and expanded the money supply to minimize, as much as possible, the impact of the recession.

European governments also took several measures to overcome this crisis. The regulatory framework of the financial institutions was strengthened with Directives such as Solvency for insurance entities or Basel for the banking sector. These regulations have the aim of enhancing the risk management function of financial institutions and assessing their risk profile. In addition, banking bailouts and stimulus measures were applied by these governments in order to mitigate the liquidity and solvency problems of financial institutions, the decrease on the gross domestic product and the increase of the unemployment rate. As these last measures provoked an increase of the public debt, the crisis in Europe progressed from the Financial Crisis of 2007-2008 to the European sovereign debt crisis.

In addition to the burst of the housing bubble and the increase of the government expenditure, countries dependent on the foreign lending stopped receiving capital due to the Financial Crisis of 2007-2008. Most of these countries had government

structural deficits due to all the measures linked to this crisis. The high Greek public debt in 2009 caused an increasing fear in the investors about the possibility of sovereign defaults, not only in Greece but also in other European countries such as Ireland, Spain, Portugal or Italy. Consequently, the credit quality of some European countries was rapidly downgraded and their interest rates soared.

The European Financial Stability Facility (EFSF) and European Financial Stabilisation Mechanism (EFSM) were created with the aim of providing financial assistance to European Union states and overcoming the European sovereign debt crisis. The first legal instrument, EFSF, raised funds in order to buy sovereign bonds or giving loans to countries with difficulties in accessing to international finance. In order to do so, the debt instruments issued by EFSF were backed by the European Union states in proportion to their weight in the European Central Bank. On the other hand, the EFSM raised funds from the financial markets using the European Union budget as collateral. As with EFSF, the financial stability of the countries in difficulties was the main objective of EFSM.

These temporal legal instruments were replaced by the European Stability Mechanism (ESM), which was created in 2012. The ESM members can ask for a bailout but two main conditions need to be met. First, the Memorandum of Understanding has to be accepted by the rest of the ESM countries. This document does not only contain the bailout proposal, but it also outlines the reforms to be applied in order to bring back the fiscal balance. Second, the European Fiscal Compact has to be fully ratified by the country asking for the bailout.

In addition to the previous legal instruments, the European Central Bank (ECB) put in place other measures to mitigate the impact of the sovereign debt crisis. In 2010, the ECB launched the Securities Market Programme. This measure had the aim of purchasing sovereign bonds in the secondary markets in order to mitigate the severe tensions in the bond market of countries such as Spain or Italy. ECB also took actions to address the difficulties suffered by banks. In 2011, the Long-Term Refinancing Operations (LTRO) programme launched a 1% interest loans for financial institutions. Although this measure did not have any direct impact on the governments, part of the liquidity obtained by banks thanks to LTROs was used to purchase sovereign debt at a higher interest rate.

The low interest rate environment and the quantitative easing policy are still in force since the Financial Crisis of 2007-2008 and the European sovereign debt crisis. The ECB decreased interest rate from 4% in 2008 to 0% in 2016. It remains at the same level since then. On the other hand, the quantitative easing policy has been enhanced by launching programmes such as the Covered Bonds Purchasing Programme and the Asset-Backed Securities Programme. The ECB also extended the quantitative easing policy to corporate debt with the Corporate Sector Purchase Programme.

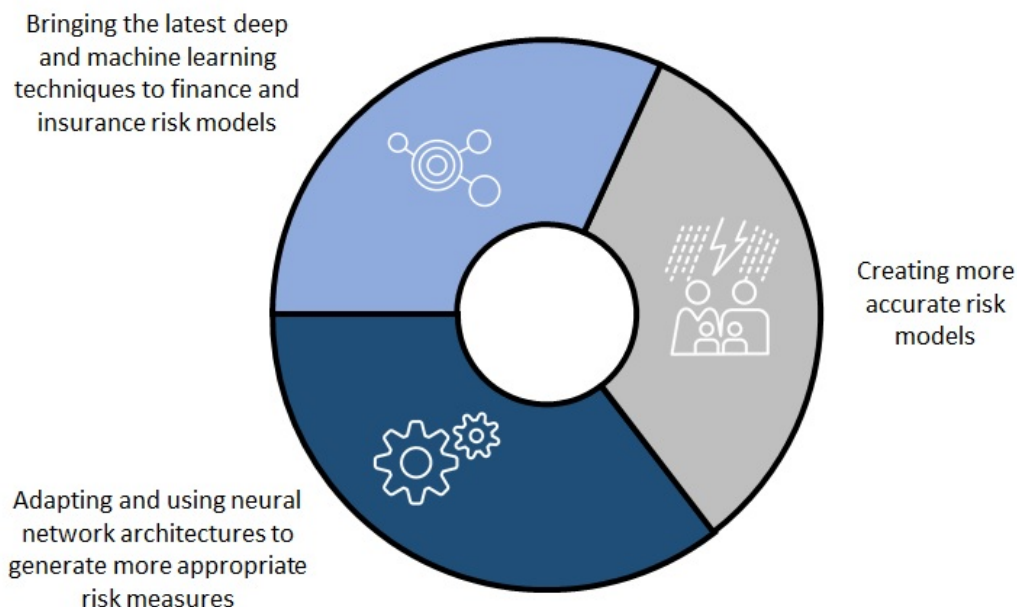
Restrictions due to Covid-19 pandemic are also having a significant impact on the global economy. In this context, ECB launched the Pandemic Emergency Purchase

Programme in order to lower borrowing costs. LTROs were also re-launched by EBC to enhance the liquidity position of financial institutions. Quantitative easing measures were also applied in most of the countries. For example, the Federal Reserve launched several stimulus packages in the United States to provide financial assistance to families, companies and financial institutions.

Because of the large exposure to financial markets and their correlation with the economical activity, crises are specially relevant for financial institutions. Indeed, events such as the Financial Crisis of 2007-2008 and the European sovereign debt crisis changed substantially the insurance and banking sector. Although crises are unpredictable, financial entities with an appropriate risk management strategy deal with market shocks more successfully. Thus, banks and insurance entities can generate a strong competitive advantage thanks to their risk management function.

According to the economic background explained in the previous paragraphs, the most relevant disruptions have conditioned the subsequent actions taken by the different economic agents. These shocks can be provoked by mismanagement or unexpected events. The Financial Crisis of 2007-2008 and the European sovereign debt can be included in the first group, whereas COVID-19 crisis was provoked by an unexpected event. Regardless the origin of the recession, they have a significant impact on the risks associated with the economic activity. The assessment of these shocks is quite relevant for businesses. This thesis has the aim of making progress in this field and it has the following objectives (Figure 1):

Figure 1: Thesis objectives



Source: Own elaboration

- *Creating more accurate risk models.* As previously stated, investors and regulators are specially interested on the risk profile of the financial institutions since the Financial Crisis of 2007-2008 and the European sovereign debt crisis. In addition, banks and insurance companies have enhanced their risk management strategy in order to create competitive advantages. Thus, metrics that take into consideration profits and risk such as ‘Return on Risk Capital’ (Braun et al. 2018) have become very relevant for top management and investors and, consequently, for the decision making process and market value of financial institutions. Taking advantage of the accuracy demonstrated by deep and machine techniques in many other fields, the different stock volatility and general insurance reserving models developed in this thesis have the aim of generating a more appropriate risk assessment.
- *Bringing the latest deep and machine learning techniques to finance and insurance risk models.* Computer vision, natural language processing or autonomous driving are the fields where most of the machine and deep learning developments are taking place. The aggressive competitiveness in the technological sector leads to the necessity of developing new algorithms and applications in order to keep the market share. In fact, financial technology (fintech) companies and decentralized finance based on cryptocurrencies are increasing more and more the competitiveness of the financial sector in the last years. Companies are taking advantage of new application programming interfaces (APIs), deep learning and cryptography in order to provide an end-to-end service via the internet. It is also worth mentioning the increasing importance of robo-advisers. Even though most of them are based on a reduced number of portfolio management theories, the number of robo-advisers based on machine learning algorithms is increasing sharply in the last years. Thus, traditional banking and insurance companies will need to invest in developing deep and machine learning techniques in areas such as risk management or marketing in order to create competitive advantages and compete with by fintechs and decentralized finance.
- *Adapting and using neural network architectures to generate more appropriate risk measures.* The problems related to computer vision, natural language processing or autonomous driving have different characteristics than the ones faced by financial institutions. Thus, this thesis has the aim of adapting machine and deep learning algorithms in order to address the risk management problems of banks and insurance companies. In addition, the risk models proposed by this thesis demonstrate that merging neural network architectures with other algorithms or risk models lead to more accurate assessment of the uncertainty faced by financial institutions.

## 2.2 Machine and deep learning: Background and methods applied

As the risk models presented in this thesis are based on deep and machine learning techniques, this subsection gives an introduction to this field. First, the different types of learning problems are explained. Second, the history and evolution of ma-

chine learning and neural network architectures are discussed. Nowadays, this algorithm is the state of the art in fields such as computer vision, natural language processing or autonomous driving. Indeed, insurance companies are starting to apply neural network architectures for fraud detection and general insurance pricing (Caldeira et al. 2015, Johnson and Khoshgoftaar 2019 and Blier Wong et al. 2021, among others). Finally, the theoretical background of the machine and deep learning methods used in this thesis is presented.

## Types of learning problems in machine and deep learning

The learning process is usually divided in three different classes: supervised, unsupervised and reinforcement learning. The characteristics of the problem and the data available for fitting the algorithms are the key factors taken into consideration for dividing the machine and deep learning problems in these classes.

Supervised learning is the most common approach. It consists of learning a mapping between the input vectors and their corresponding target variables (Bishop 2006). Thus, algorithms are fitted on training data in order to create a model able to predict accurately new observations (Russell and Norvig 2009). Depending on the characteristics of the target variable, the supervised problem is defined as:

- Regression, when the target variable is numerical.
- Classification, in case the target variable is categorical.

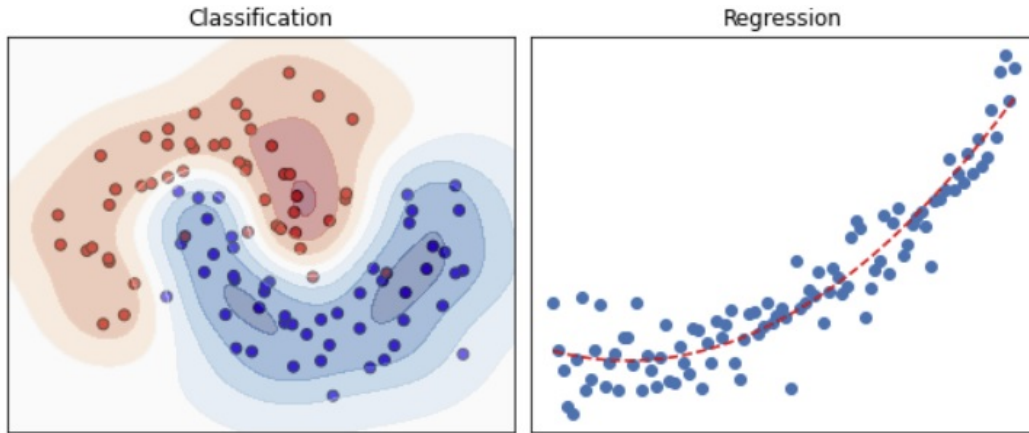
Figure 2 shows a supervised regression and the boundaries of a two classes classification. Regardless of the type of supervised problem, explanatory variables can be numerical or categorical. Supervised learning has applications in several fields such as finance (Dixon et al. 2020), insurance (Hastie et al. 2009), autonomous driving (Howard et al. 2019), natural language processing (Devlin et al. 2018 and Brown et al. 2020) or medical diagnosis (Wu et al. 2018), among others.

When algorithms are fitted in order to describe the different relationships present in the data, the problem is defined as unsupervised learning. In contrast to the previous approach, unsupervised learning does not have a target variable. Thus, the algorithms used within this approach are able to work without a response variable. The main types of unsupervised learning problems are:

- Clustering. Observations are grouped in different clusters depending on the input variables given to the algorithm. This is the most common problem in the field of unsupervised learning.
- Density estimation. The distribution function of one variable is estimated taking into consideration the observations given to the algorithm.

Finally, reinforcement learning is used when the algorithm needs to take actions or decisions in an environment. In this approach, the learner takes an action and it

Figure 2: Examples of supervised learning



*Source:* Own elaboration

receives a reward depending on the appropriateness of the action taken. In contrast to the previous approaches, there is not a fixed database in reinforcement learning. As stated by Sutton and Barto (2018), algorithms are fitted thanks to the definition of a set of goals (called policy). Thus, in a first step the model makes a decision or action and, then, it obtains a reward depending on the defined policy. Algorithms trained with this approach have overcome human performance in specific tasks such as playing chess (Silver et al. 2017) or 'go' (Silver et al. 2016).

## The rise of neural networks and machine learning

Although neural networks and tree-based models such as random forests and gradient boosting were developed years before the rise of the machine learning, they played a key role in it. These algorithms were not widely used until the late 2000s due to the computational power required to fit them. Thus, less computationally expensive algorithms like autoregressive (Engle 1982) and generalised linear models (Nelder and Wedderburn 1972) were widely used during the 1990s. The revolution in this field starts in 2008 with the possibility to fit algorithms in the Graphic Processing Unit (GPU). Since then, artificial neural networks and tree-based models have shown that they are able to solve new problems and overcome the performance of traditional algorithms.

First neural network architectures were described by McCulloch and Pitts (1943). As previously stated, this algorithm was hard to train due to the lack of computational power. Thus, during the following years Rosenblatt (1958) and Widrow and Hoff (1962) developed and trained simple artificial neural networks. Figure 3 illustrates the main milestones of this algorithm since its creation.

Since 1960s, several researches focused their efforts on making more efficient the optimization of neural networks. Gradient descent methods based on Euler-LaGrange equations were introduced by Kelley (1960), Bryson (1961), Bryson and Denham (1961) and Amari (1967), among others. Even though it was not focused on training neural networks, the first efficient back-propagation and general automatic differentiation algorithm was developed by Linnainmaa (1970). Dreyfus (1973) used the previous approach to minimize cost functions and Werbos (1974) suggested the possibility of applying this method for optimizing neural networks. Rumelhart et al. (1986a) and Rumelhart and Zipser (1986) popularized and improved the implementation of back-propagation in the field of neural networks. Since then, so many variations such as R-prop (Riedmiller and Braun 1993), iRprop (Igel and Hüsken 2003), RMSprop (Zeiler 2012) or ADAM (Kingma and Ba 2014) were developed. In addition, convolutional neural networks (Fukushima 1979 and LeCun et al. 1990), long-short term memory cells (Hochreiter and Schmidhuber 1997a) and gated recurrent units (Cho et al. 2014) were introduced in order to deal with problems such as image recognition or time series forecasting.

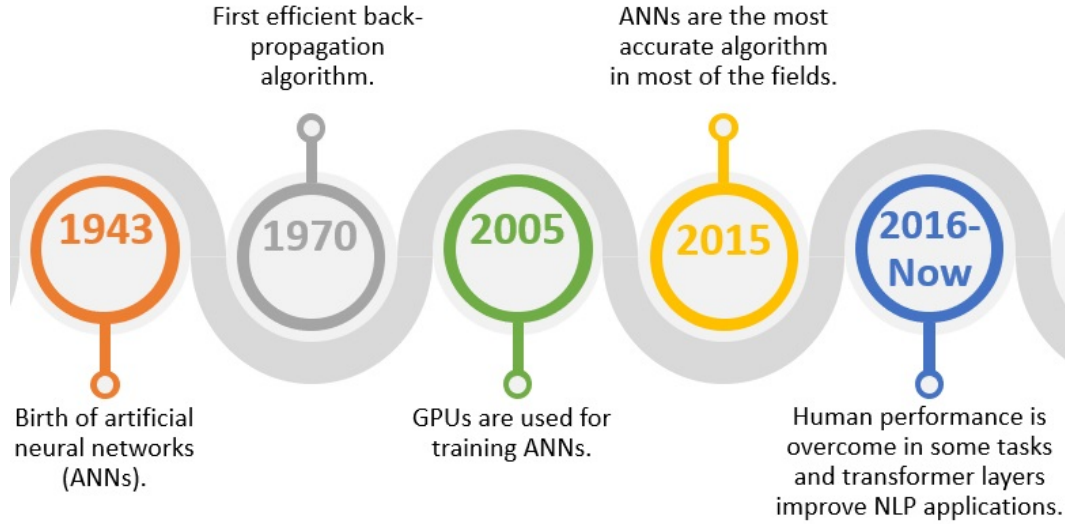
Although the previous contributions, algorithms such as support vector machines (Cortes and Vapnik 1995) dominated pattern recognition problems during 2000s. As previously stated, the implementation of GPU-based neural networks played a key role in the rise of this algorithm. Oh and Jung (2004) and Chellapilla et al. (2006) were some of the first researchers using GPUs to reduce the training time of neural networks. Thanks to the increase of computational power, neural networks started to outperform the rest of machine learning algorithms in so many fields such as speech recognition (Graves et al. 2013, Gonzalez-Dominguez et al. 2014, Geiger et al. 2014, Fernandez et al. 2014 and Sak et al. 2014), audio detection (Marchi et al. 2014, Fan et al. 2014 and Brueckner and Schuler 2014), language translation (Sutskever et al. 2014), computer vision (Szegedy et al. 2014, Sermanet et al. 2013, Goodfellow et al. 2014), object detection (Szegedy et al. 2013), medical diagnosis (Li et al. 2014) or video classification (Karpathy et al. 2014).

The recent inclusion of transformer layers in artificial neural networks (Vaswani et al. 2017) improved the performance of the already existing natural language processing (NLP) applications (Devlin et al. 2018 and Brown et al. 2020). In addition, it is worth mentioning that this algorithm have also overcome the human performance in some tasks such as playing chess (Silver et al. 2017) or ‘go’ (Silver et al. 2016).

Although neural networks are considered the state of art in so many fields, tree-based algorithms have demonstrated that they can be more accurate in some structured data problems. Thanks to the increase of the computational power, tree-based algorithms have evolved from individual decision or regression trees to more complex structures such as random forest (Breiman 2001), gradient boosting with trees (Friedman 2000) or XGBoost (Chen and Guestrin 2016).

Classification and decision trees are easy to interpret. Nevertheless, they have not been widely used because of the two following major disadvantages. First, small

Figure 3: Timeline of artificial neural networks



Source: Own elaboration

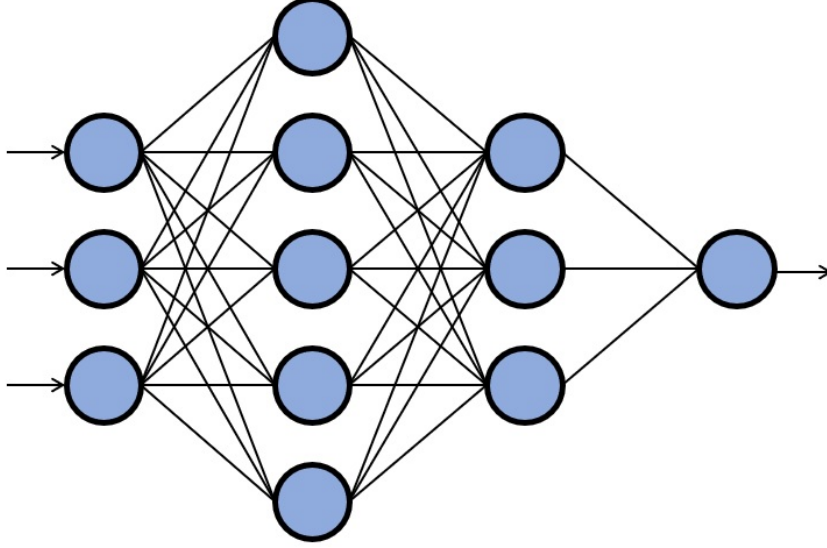
changes in the dataset can lead to significant modifications in the optimum configuration of the tree. Second, overfitting is usually a problem due to the characteristics and architecture of trees. Random forest addresses the previous problems by averaging  $n$  trees and randomizing the input data. The explanatory variables considered for splitting each single tree within the model architecture are also selected randomly. In contrast to random forest, gradient boosting and XGBoost apply a sequential approach to solve the disadvantages of individual trees. Therefore, the predictions made by the  $i^{th}$  tree are calculated taking into consideration the estimates made by the trees already added to the model.

## Main machine and deep learning methods applied

This subsection presents the main deep and machine learning methods applied for developing the stock volatility and reserving risk models presented in sections 3, 4, 5 and 6. Thus, several variants of neural networks, random forest and gradient boosting with trees will be explained in the following paragraphs. It is worth mentioning that some of these algorithms have been modified or merged to develop the models introduced by this thesis. As the procedures of merging or modifying the algorithms are explained in sections 3, 4, 5 and 6, this subsection is focused on presenting the algorithms without the modifications applied for developing the stock volatility and reserving risk models.

Artificial neural network is an algorithm inspired by the biology of human brains. The main components of this algorithm are the following:

Figure 4: Architecture of a feed forward neural network



*Source:* Own elaboration

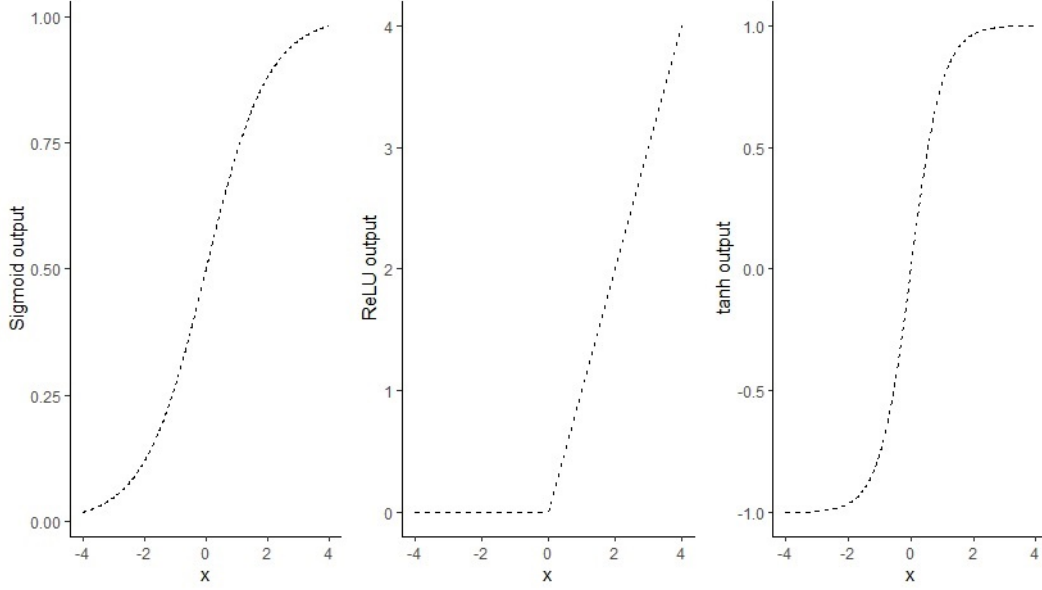
- **Neurons.** As represented by the blue circles of Figure 4, artificial neural networks are composed of individual neurons. The neurons with no predecessor are named input neurons, while those ones without successor are called output neurons.
- **Weights and connections.** Connections are represented by the lines connecting each neuron in Figure 4. They transfer the output of one neuron to the next one. In the feed forward neural networks, the inputs received by each neuron are linearly transformed as follows:

$$\sum_{i=1}^I w_i x_i + w_0 \quad (1)$$

where  $w_i$  is the weight and  $x_i$  the output value of the predecessor neuron  $i$ . The bias term is  $w_0$ . For the sake of simplicity, the bias term and connection are not represented by Figure 4.

- **Activation function.** The outputs of the neurons are modified with the so-called activation function. There are numerous functions such as sigmoid, tanh, softmax, ReLU, Leaky ReLU or linear. As they have different ranges, the characteristics of the problem faced determine the activation function to be used in the neural network. Tanh, sigmoid and ReLU (Figure 5) are the main activation functions used within the different stock volatility and reserving models proposed by this thesis. The sigmoid activation function is:

Figure 5: Example of activation functions



Source: Own elaboration

$$h(x) = \frac{1}{1 + \exp(-x)} \quad (2)$$

ReLU is defined as follows:

$$h(x) = \max(0, x) \quad (3)$$

Finally, tanh is calculated as shown below:

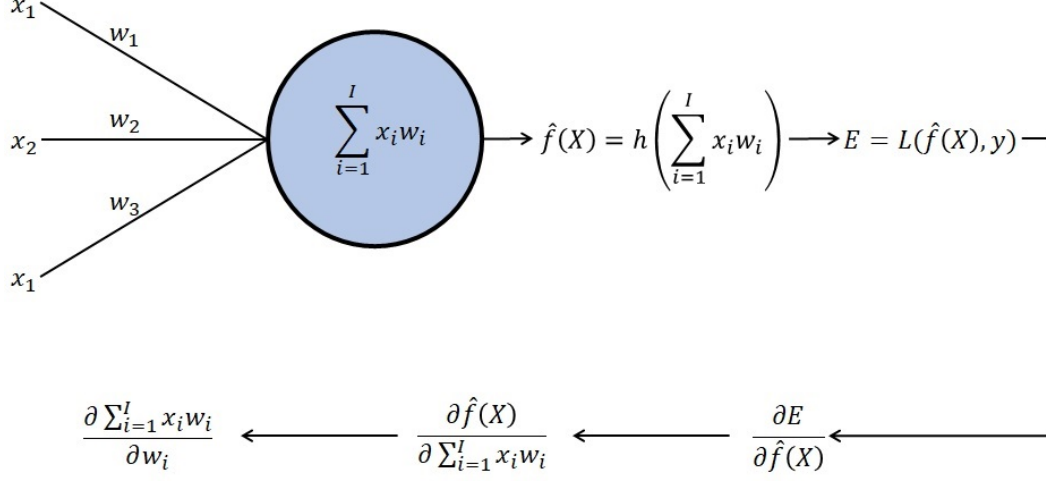
$$h(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \quad (4)$$

Considering all the elements of neural networks and following the previous notation, the output of a neuron can be defined as follows:  $h(\sum_{i=1}^I w_i x_i + w_0)$ . As neural networks are normally composed of several layers and neurons, a more complex notation is required to represent their architecture. Taking as a reference the notation given by Bishop (2006), the expression of a feed forward neural network with two hidden layers and one output neuron can be defined as follows:

$$\hat{f}(X) = h^{(3)} \left( \sum_{k=1}^T w_{p,k}^{(3)} h^{(2)} \left( \sum_{j=1}^M w_{k,j}^{(2)} h^{(1)} \left( \sum_{i=1}^D w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right) + w_{p,0}^{(3)} \right) \quad (5)$$

where  $h^{(n)}$  is the activation function of layer  $n$ ,  $w_{z,v}^{(n)}$  is the  $v$ -th weight of neuron  $z$  inside the layer  $n$  and  $x_i$  refers to the  $i$  input variable. As the previous formula has

Figure 6: Forward and backward phases in a one layer network



Source: Own elaboration

the aim of representing a feed forward neural network with one output neuron,  $p = 1$ .

Back-propagation algorithm is used iteratively for training the weights of neural networks. This method back-propagates the gradient of the error function to every weight of the algorithm. The steps of back-propagation are illustrated by Figure 6 and explained below:

1. The forward phase is calculated taking into consideration the explanatory variables and the initial weights. This calculation or phase gives the prediction of the network,  $\hat{f}(X)$ .
2. The error is computed taking into consideration the loss function ( $L$ ), the target output ( $y$ ) and the prediction made in the previous step:  $E_d = L(\hat{f}(X)_d, y_d)$ , where  $d$  represents the observation number within the database. Binary cross entropy and root mean squared error are the most popular loss functions for classification and regression respectively.
3. Calculation of the backward phase. In this step, the gradient of each observation with respect to the weights is calculated. The calculation of the partial derivatives and the chain rule are especially relevant in this phase. For example, the three following partial derivatives are needed to compute the backward pass of the weights of Figure 6:
  - $\partial E_d / \partial \hat{f}(X)_d$ . The expression of this derivative depends on the loss function selected.
  - $\partial \hat{f}(X)_d / \partial \sum_{i=1}^I w_i x_{i,d}$ . As  $\hat{f}(X) = h(\sum_{i=1}^I w_i x_i)$ , this partial derivative is determined by the activation function selected,  $h$ .

- $\partial \sum_{i=1}^I w_i x_{i,d} / \partial w_i$ . This is the derivative of the linear combination of weights and inputs.

The chain rule is applied to obtain the gradient with respect to the weight:

$$\frac{\partial E_d}{\partial w_i} = \frac{\partial E_d}{\partial \hat{f}(X)_d} \frac{\partial \hat{f}(X)_d}{\partial \sum_{i=1}^I w_i x_{i,d}} \frac{\partial \sum_{i=1}^I w_i x_{i,d}}{\partial w_i} \quad (6)$$

4. The individual gradients are combined as follows  $g_i = (\sum_{d=1}^D \partial E_d / \partial w_i) / D$ . Where  $D$  is the total number of observations taken into consideration within the back-propagation algorithm.
5. Finally, the rule to update the weights of the neural network is applied. As the gradient of neural networks tends to be plenty of local minimums, several rules to update the weights (such as iRprop, RMSprop or ADAM) have been developed. The rule used to update the weights of the neural networks trained in this thesis is Adaptive Moment Estimation (ADAM). The following equations define this updating rule developed by Kingma and Ba (2014):

$$\omega_{i,t} = \omega_{i,t-1} - \delta \frac{\hat{m}_{i,t}}{\sqrt{\hat{v}_{i,t} + \epsilon}} \quad (7)$$

$$\hat{m}_{i,t} = \frac{\beta_1 m_{i,t-1} + (1 - \beta_1) g_{i,t}}{1 - \beta_1^t} \quad (8)$$

$$\hat{v}_{i,t} = \frac{\beta_2 v_{i,t-1} + (1 - \beta_2) g_{i,t}^2}{1 - \beta_2^t} \quad (9)$$

where  $g_{i,t}$  is the gradient assigned to the weight  $i$  in the iteration  $t$  and  $\delta$  the initial learning rate. The default calibration proposed by the authors for  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 10^{-8}$  is applied in this thesis. Weighted decay can be added to ADAM to penalise large weights and, thus, reduce the overfitting. The update of the weights has to be modified as follows to implement this regularization method:

$$\omega_{i,t} = \omega_{i,t-1} - \delta \left( \frac{\hat{m}_{i,t}}{\sqrt{\hat{v}_{i,t} + \epsilon}} + \lambda \omega_{i,t-1} \right) \quad (10)$$

where  $\lambda$  is the parameter responsible of penalizing large weights.

The previous steps of the back-propagation algorithm are repeated until the weights are optimized. Epochs and batch size are the neural network hyper-parameters determining the total number of iterations of the back-propagation algorithm during the training phase. On one hand, the number of complete evaluations of the training set is determined by the epochs. On the other hand, the batch size refers to the number of observations taken into consideration in each step of the back-propagation algorithm. For instance, the total number of back-propagation iterations will be equal to 4 if the train set has 64 observations and epochs and batch size are equal to 2 and 32 respectively. As this example assumes a database of 64 observations and a batch

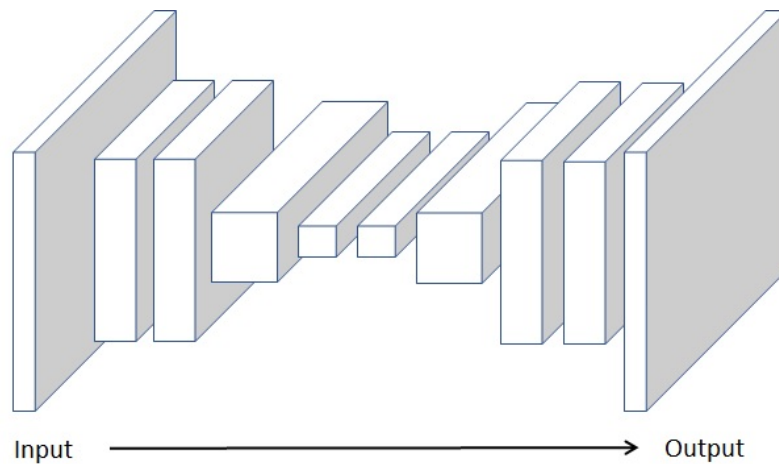
size of 32, two back-propagation iterations are required to complete on single epoch.

With regard to the regularization approaches available in neural networks, it is also worth mentioning the dropout method. In contrast to the inclusion of a weighted decay within ADAM, this approach does not modify the formula determining how weights are updated. In this method, some weights are randomly ignored during the optimization process. Thus, the non-omitted weights are required to correct mistakes from the omitted weights, making the model more robust and less prone to suffer overfitting.

One of the main advantages of artificial neural networks is their flexibility. The architecture of this algorithm can be adapted to the nature of the problem faced. Thanks to this, more specific approaches can be developed to solve certain problems such as image recognition or time series forecasting. The main families or architectures of artificial neural networks are the following:

- Feed forward neural networks. This family is the most commonly used. In contrast to other types of neural networks, this architecture can be used in a wide variety of problems thanks to its flexibility. In fact, feed forward layers are usually applied in combination with the other families of networks. They are specially appropriate for structured data regression and classification.

Figure 7: Example of convolutional neural network architecture



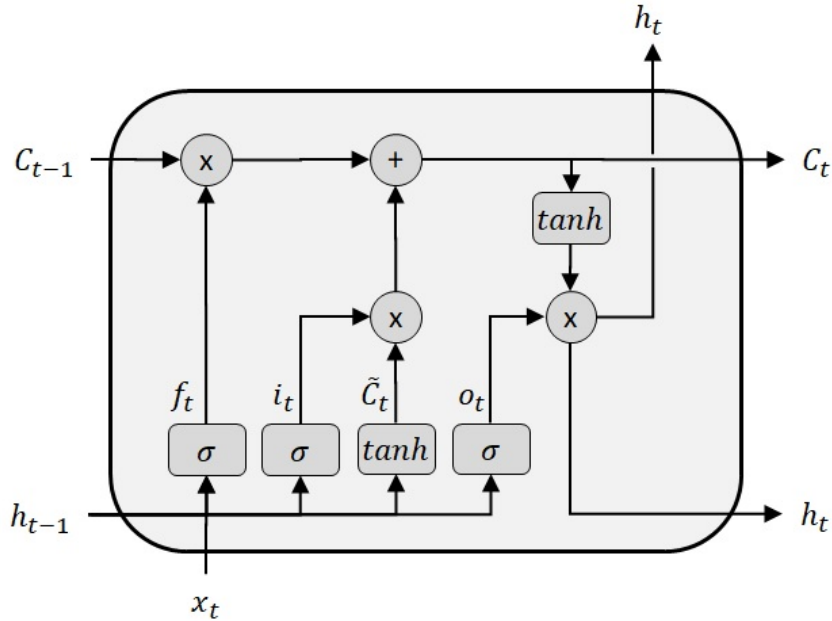
*Source: Own elaboration*

- Convolutional neural networks are commonly used in the field of computer vision. Images can be decomposed in a set of three dimensional matrices by using the RGB (Red-Green-Blue) code of each pixel. Convolutional filters (which are used in this class of networks) are able to deal with 3D data and recognise the correlation between pixels. Thus, they are specially appropriate for image recognition and computer vision problems. As previously stated, they can

be combined with feed forward layers for applications such as image classification. Figure 7 illustrates the architecture of a specific type of convolutional neural network called encoder-decoder. This structure is widely used for image translation or modification purposes, where the output is also another image. Although this family is not applied in this thesis, it is worth including it in this list because it has a remarkable importance on some of the main problems addressed by machine and deep learning such as autonomous driving or image recognition.

- Recurrent neural networks. This class of networks is especially tuned to recognise the temporal correlation of the input data. Thus, they are mainly used in problems such as speech recognition, audio detection or language translation. They are composed of memory cells such as gated recurrent units (Cho et al. 2014) or long-short term memory architectures (Hochreiter and Schmidhuber 1997a). The recurrent neural networks used in this thesis are based on long-short term memory (LSTM) units, which are more complex than gated recurrent units (GRU). Although the training time of LSTMs is longer than GRUs, the higher level of complexity gives the possibility of recognizing more complex temporal patterns. Figure 8 and the following equations define the LSTMs architecture:

Figure 8: LSTM structure



Source: Own elaboration

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (11)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (12)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (13)$$

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \quad (14)$$

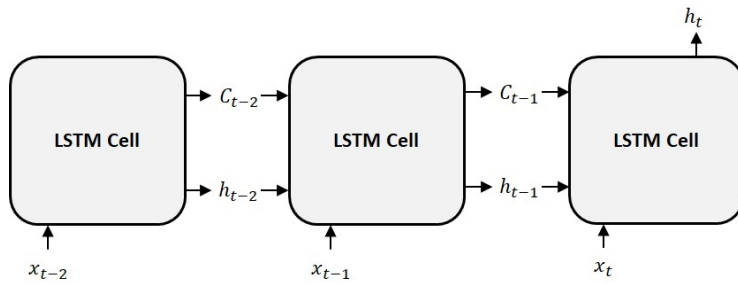
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (15)$$

$$h_t = o_t \tanh(C_t) \quad (16)$$

Where  $W_f$ ,  $W_i$ ,  $W_c$ ,  $W_o$ ,  $b_f$ ,  $b_i$ ,  $b_c$  and  $b_o$  represent the weights of the RNN and  $\sigma(x)$  the sigmoid function. The previous expressions are combined with a loop in order to consider any number of previous steps as input data for the LSTM cell. Figure 9 shows the architecture of a LSTM with several input steps and just one output. As with the convolutional neural networks, this class tends to be combined with feed forward layers. First, time series are managed by several LSTM or GRU units and, then, the outputs of these units are transformed by feed forward layers to obtain the final prediction.

Time series and structured data are the main types of information in financial problems. Thus, feed forward layers and recurrent neural networks are the main types of structures applied by this thesis. In addition to these architectures, Transformer layers (Vaswani et al. 2017) are used in one of the stock volatility models presented by this thesis. This novel architecture based on attention mechanisms and feed forward layers is specially prepared for dealing with time series. Indeed, Transformer layers have overcome the performance of LSTMs and GRUs in the field of natural language processing (Devlin et al. 2018 and Brown et al. 2020).

Figure 9: Many-to-one LSTM (One unit)

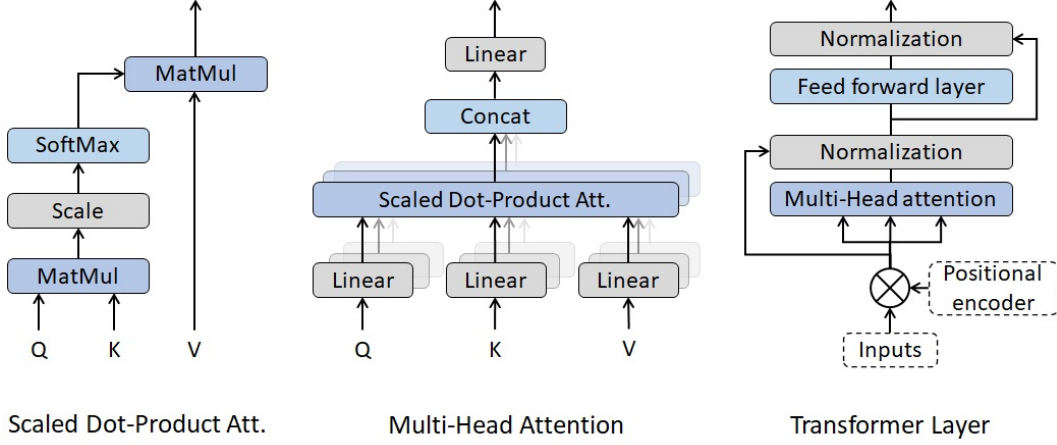


Source: Own elaboration

In contrast to LSTM or GRU cells, Transformer layers have no recurrence. They rely on a combination of positional encoders, feed forward layers and attention mechanisms to deal with time series. As no recurrence is present in Transformer layers, the different time lapses of the input data can be managed in parallel by the computer, reducing the time required to fit the algorithm.

The original architecture of Transformer layers (Vaswani et al. 2017) is shown in Figure 10. A description of the main components is given below:

Figure 10: Multi-Head attention and Transformer architecture



Source: Own elaboration

- Positional encoder. As previously stated, Transformer layers are not a recurrent algorithm such as LSTM or GRU cells. Thus, these layers need a positional encoder to recognize the different time lapses present in the input data. The positional encoder modifies the input data depending on its relative position. Vaswani et al. (2017) proposed the following wave functions as positional encoders for NLP purposes:

$$PE_{(pos, 2i)} = \sin(pos/1000^{2i/dim}) \quad (17)$$

$$PE_{(pos, 2i+1)} = \cos(pos/1000^{2i/dim}) \quad (18)$$

where  $dim$  is the total number of time series (or word embedding dimension in NLP) given as input to the model,  $pos$  is the position of the observation within the time series and  $i = (1, 2, \dots, dim - 1)$ .

- Multi-Head attention. It can be considered the key component of the transformer architecture. As shown in Figure 10, Multi-Head attention mechanism consists of several scaled dot-product attention units running in parallel. The scaled dot-product attention proposed by Vaswani et al. (2017) is calculated as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) V \quad (19)$$

where  $Q$ ,  $K$  and  $V$  are the input matrices and  $d_k$  the number of input variables taken into consideration within the dot-product attention mechanism.

Multi-head attention mechanism splits the explicative variables in different groups or ‘heads’ and, then,  $h$  attention mechanisms are computed in parallel. To do so, the input matrices and their weights are split in different heads before the computation of the scaled dot-product attention:  $Q_i$ ,  $K_i$  and  $V_i$  are the input data of the head  $i$  and  $W_i^Q$ ,  $W_i^K$  and  $W_i^V$  their respective weights. Then, the outputs of the different heads or attention mechanisms are concatenated. Thus, the expression of the Multi-Head attention mechanism is

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h) W^O \quad (20)$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (21)$$

where  $h$  is the total number of heads. It is worth mentioning that all the matrices of weights ( $W_i^Q$ ,  $W_i^K$ ,  $W_i^V$  and  $W^O$ ) are trained using feed forward layers with linear activations.

- Normalization. The outputs from the Multi-Head attention mechanism are normalized in order to facilitate the training of the Transformer layer.
- Feed forward layer. The normalized output from the attention mechanism is modified by a feed forward layer with ReLU activation function.

Transformer layers also include two residual connections (He et al. 2016). Thanks to these connections the model will have the possibility of skipping the training of some parts of the architecture. In addition, these connections have a positive impact when the vanishing gradients problem is present because the model can use the activation of previous layers until the skipped part of the model starts to learn. The Python implementation of transformer layers and the volatility models proposed by this thesis is available in <https://github.com/EduardoRamosP/MultiTransformer>.

Apart from neural networks, tree-based algorithms such as random forest or gradient boosting with trees are also applied in the risk models developed in this thesis. Decision or classification trees define a set of rules to split the database in different nodes. The final prediction of this algorithm is obtained by calculating the average value of the response variable (regression) in every terminal node. The architecture of decision and classification trees leads to two major disadvantages. First, overfitting tends to be an issue with this algorithm. Second, small changes in the data can provoke remarkable changes on the optimum tree configuration.

Random forests address the disadvantages of trees by randomizing the explanatory variables taken into consideration in the splitting process and averaging  $n$  different trees. The input data given to each tree of the random forest architecture is also selected randomly. The steps of the algorithm are the following:

1.  $n_{tree}$  bootstrap samples are drawn from the original dataset.
2. For each sample drawn in the previous step, an unpruned classification or regression tree is grown. Instead of taking into consideration all the explicative

variables to split the trees, only  $m_{try}$  variables randomly selected are considered. Thus, bagging can be considered a particular of random forests case where  $m_{try}$  is equal to the number of explicative variables.

3. New data is predicted by selecting the majority class (classification) or averaging (regression) the predictions of the  $n_{tree}$  trees.

In contrast to random forest, gradient boosting applies a sequential approach to solve the disadvantages of individual decision or regression trees. The steps of this algorithm are the following:

1. Initialize  $\hat{f}_0(x)$  with a constant value that meets the following:

$$\hat{f}_0(x) = \underset{\rho}{\operatorname{argmax}} \sum_{i=1}^n L(y_i, \rho) \quad (22)$$

where  $L(y_i, \rho)$  is the loss function selected and  $n$  the number of observations.

2. For  $t = 1$  to  $T$ :

- 2.1. Compute the negative gradient or pseudo-residual:

$$z_{it} = - \frac{\partial L(y_i, \hat{f}_{t-1}(x_i))}{\partial \hat{f}_{t-1}(x_i)} \quad (23)$$

where  $i = 1, \dots, n$

- 2.2.  $Z$  observations are selected randomly from the database.
- 2.3. Fit a tree,  $\hat{h}_t(x)$ , to the negative gradient calculated in step 2.1. The learner is trained with the subset of data selected in step 2.2.
- 2.4. Update the model as follows:

$$\hat{f}_t(x) = \hat{f}_{t-1}(x) + \delta \hat{h}_t(x) \quad (24)$$

where  $\delta$  is the learning rate.

3. Output the final model  $\hat{f}_T(x)$ .

As it might be understood from the previous steps, this algorithm includes the recurrence of the boosting methodology and the randomness of bagging. Boosting produces strong learners by combining sequentially weak learners, while bagging helps to reduce the variance and overfitting of the learner.

There are several implementations and extensions of this algorithm. Nevertheless, extreme gradient boosting (Chen and Guestrin 2016), commonly known as XGBoost, might be the most relevant one. Apart from being less computationally intensive than traditional gradient boosting, XGBoost brings in the possibility of penalizing the model through both L1 and L2 regularization. L1 regularization adds a penalty equal to the absolute magnitude of the coefficients, while L2 brings in a penalty based

on the square of their magnitude. Thus, the objective function of XGBoost includes two parts: the loss function and the regularization term.

$$Obj_{(t)} = \sum_{i=1}^n L(y_i, \hat{y}_{it}) + \sum_{i=1}^t \Omega_i \quad (25)$$

where  $\Omega$  is the regularization term and  $\hat{y}_i^{(t)}$  the predictions of the algorithm in the iteration  $t$ . As stated before,  $T$  refers to the total number of trees and  $n$  is the number of observations. Notice that, in some implementations of this algorithm, only the observations not selected in step 2.2. are used for calculating the loss. This approach produces a fairer error measure because the observations used for fitting the last tree are not considered in the error calculation. Chen and Guestrin (2016) defines classification or regression trees as follows:

$$h_t(x) = w_{q(x)}, w \in \mathbb{R}^J, q : \mathbb{R}^d \rightarrow \{1, 2, \dots, J\} \quad (26)$$

where  $w$  is the vector of scores on leaves,  $q$  is a function assigning each data point to the corresponding leaf, and  $J$  is the number of leaves. Then, Chen and Guestrin (2016) suggest the following definition for the regularization term:

$$\Omega_t = \gamma J + \frac{1}{2} \lambda \sum_{j=1}^J w_j^2 \quad (27)$$

where  $\gamma$  and  $\lambda$  are the parameters fixing the level of L1 and L2 regularization respectively. As XGBoost is a sequential algorithm, the objective function can be decomposed as follows:

$$Obj_{(t)} = \sum_{i=1}^n L(y_i, \hat{y}_{i,t-1} + \hat{h}_t(x_i)) + \Omega_t + \text{constant} \quad (28)$$

Chen and Guestrin (2016) take the Taylor expansion of the objective function up to the second order:

$$Obj_{(t)} \simeq \sum_{i=1}^n [L(y_i, \hat{y}_{i,t-1}) + g_i \hat{h}_t(x_i) + \frac{1}{2} e_i \hat{h}_t^2(x_i)] + \Omega_t + \text{constant} \quad (29)$$

where  $g_i = \partial_{\hat{y}_{i,t-1}} L(y_i, \hat{y}_{i,t-1})$  and  $e_i = \partial_{\hat{y}_{i,t-1}}^2 L(y_i, \hat{y}_{i,t-1})$  are the first and second gradient of the loss function respectively. Removing the constant terms, the objective function at step  $t$  can be defined as follows:

$$Obj_{(t)} \simeq \sum_{i=1}^n [g_i \hat{h}_t(x_i) + \frac{1}{2} e_i \hat{h}_t^2(x_i)] + \Omega_t \quad (30)$$

Defining  $I_d = \{i \mid q(x_i) = d\}$  as the set of data points assigned to the leaf  $d$  and expanding  $\Omega_t$ , the objective function of XGBoost at step  $t$  can be rewritten as follows:

$$Obj_{(t)} \simeq \sum_{i=1}^n [g_i \hat{h}_t(x_i) + \frac{1}{2} e_i \hat{h}_t^2(x_i)] + \gamma J + \frac{1}{2} \lambda \sum_{j=1}^J w_j^2 \quad (31)$$

$$= \sum_{j=1}^J [(\sum_{i \in I_d} g_i) w_j + \frac{1}{2} (\sum_{i \in I_d} e_i + \lambda) w_j^2] + \gamma J \quad (32)$$

Notice that the summation index of the second line can be modified because all the observations on the same leaf of the tree get the same score. Thus, the optimal  $w$  of leaf  $j$  for a fixed structure  $q(x)$  is:

$$w_j^* = -\frac{\sum_{i \in I_d} g_i}{\sum_{i \in I_d} e_i + \lambda} \quad (33)$$

Therefore, the best objective reduction is determined by the following expression:

$$Obj_{(t)}(q) = -\frac{1}{2} \sum_{j=1}^J \frac{(\sum_{i \in I_d} g_i)^2}{\sum_{i \in I_d} e_i + \lambda} + \gamma J \quad (34)$$

Hence, this function can be used to assess the appropriateness of a specific tree structure  $q$ . Calculating the quality of all the possible final tree structures is computational expensive due to all the potential combinations given by the data. With the aim of solving this problem, an algorithm is iteratively applied in order to calculate the best branches. Thus, the optimal tree structure is calculated step by step, calculating sequentially the optimal splits. Because of this, Chen and Guestrin (2016) suggest the following objective function to evaluate the tree splits in XGBoost:

$$Obj_{(t)}^{Split} = \frac{1}{2} \left[ \frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} e_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} e_i + \lambda} + \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} e_i + \lambda} \right] - \gamma \quad (35)$$

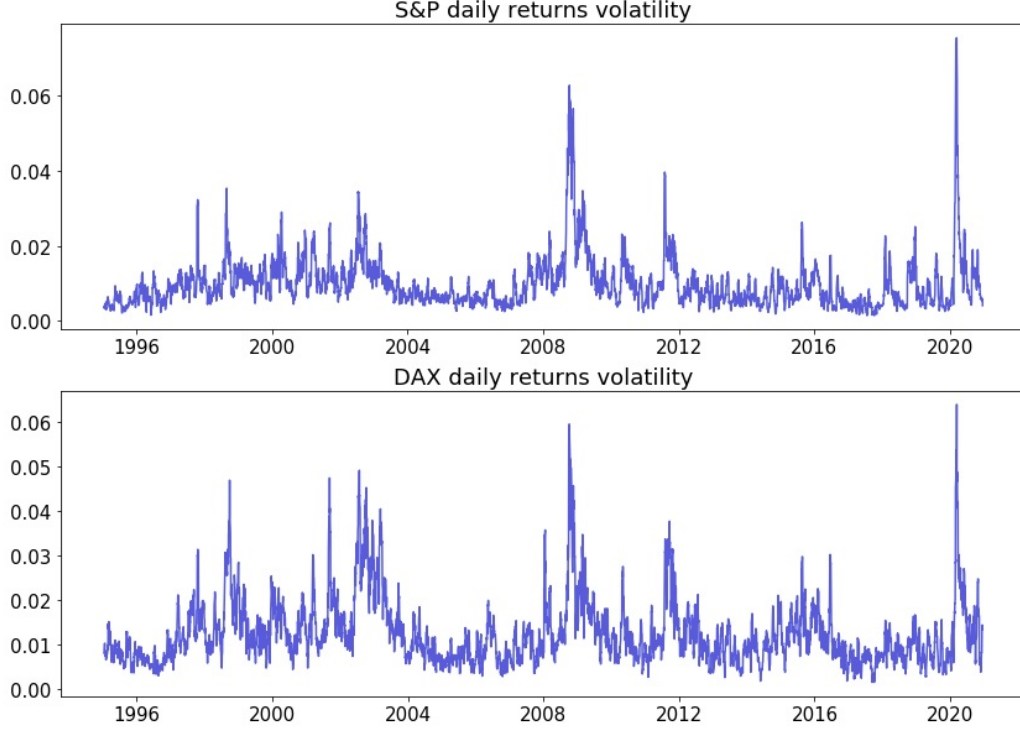
Where  $I_L$  and  $I_R$  are the set of data points assigned to the left and right nodes after the split and  $I = I_L \cup I_R$ . XGboost applies  $Obj_{(t)}^{Split}$  iteratively to calculate the optimal splits of every single tree included in the algorithm. Notice that a branch would not be added to the tree, if the left part of the expression is lower than  $\gamma$ .

### 2.3 Stock market volatility and machine learning

This subsection is focused on presenting the main existing methodologies in the field of stock volatility forecasting. Sections 3 and 5 explain the stock volatility forecasting models based on the machine and deep learning methods presented in the previous subsection.

Since the Financial Crisis of 2007-2008, investors and regulators are interested not only on the profit of financial institutions but also on the risk assumed by the entities. Regulations such as Solvency II, Basel III or Swiss Solvency Test have enhanced the risk management framework of financial institutions. In addition, measures that relate risk, profitability and own funds such as return on risk capital or the solvency ratio play a key role in acquisitions, portfolio management and dividend payments, among others. Thus, methods and models for the valuation of assets, liabilities and their variability play an essential role in the market value of financial institutions. Figure 11 shows the 10 days volatility of the S&P500 and DAX daily returns, which are the main stock indices of the United States and Germany receptively. As stock markets volatility changes sharply, financial institutions need accurate volatility forecasting models to set appropriate risk management strategies.

Figure 11: Historical volatility of S&P and DAX



Source: Own elaboration

GARCH-based algorithms are considered the main family of stock volatility models because they have the ability of fitting the volatility clustering observed in financial time series (Mandelbrot 1963). The generalization of autoregressive conditional heteroskedasticity models (GARCH) developed by Engle (1982) and Bollerslev (1986) are widely used in the stock volatility forecasting literature (Mapa 2003, Frimpong and Oteng-Abayie 2006, Floros 2008, Ugurlu et al. 2014, Maqsood et al. 2017, Rastogi et al. 2018 and Gulay and Emeç 2019, among others) and it has the following expression:

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^q \alpha_i r_{t-i}^2 + \sum_{i=1}^p \beta_i \sigma_{t-i}^2 \quad / \quad \hat{r}_t = \hat{\sigma}_t \epsilon_t \quad (36)$$

where  $\omega$ ,  $\alpha_i$  and  $\beta_i$  are the parameters to be estimated,  $r_t$  the return and  $\sigma_t^2$  the volatility. The shape of the returns sampled by this algorithm is determined by the distribution selected for the model innovations ( $\epsilon_t$ ). Normal and Student's t-distribution are the most common options used for generating these innovations.

Volatility behaves differently depending on the market tendency, thus EGARCH and GJR-GARCH were developed by Nelson (1991) and Glosten et al. (1993) respectively

in order to fit this behaviour. EGARCH model is defined as follows:

$$\log \hat{\sigma}_t^2 = \omega + \sum_{i=1}^p \alpha_i \log \hat{\sigma}_{t-i}^2 + \sum_{i=1}^q (\beta_i e_{t-i} + \gamma_i (|e_{t-i}| - E|e_{t-i}|)) \quad (37)$$

where  $\omega$ ,  $\alpha_i$ ,  $\beta_i$  and  $\gamma_i$  are the parameters to be estimated and  $e_t = r_t/\sigma_t$ . On the other hand, GJR-GARCH has the following expression:

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^q \alpha_i r_{t-i}^2 + \sum_{i=1}^o \gamma_i r_{t-i}^2 I_{[r_{t-1} < 0]} + \sum_{i=1}^p \beta_i \sigma_{t-i}^2 \quad (38)$$

As with EGARCH model,  $\omega$ ,  $\alpha_i$ ,  $\beta_i$  and  $\gamma_i$  are the parameters to be estimated.  $I$  is an indicator function that takes the value of 1 when the condition is met. Threshold GARCH (TrGARCH) model also generates a different volatility depending on the market momentum:

$$\hat{\sigma}_t = \omega + \sum_{i=1}^q \alpha_i |r_{t-i}| + \sum_{i=1}^o \gamma_i |r_{t-i}| I_{[r_{t-i} < 0]} + \sum_{i=1}^p \beta_i \sigma_{t-i} \quad (39)$$

where  $\omega_i$ ,  $\alpha_i$ ,  $\beta_i$  and  $\gamma_i$  are the model parameters.

In addition to the previous algorithms, there are many other variants of autoregressive conditional heteroskedasticity models. Bollerslev et al. (1988) applied multivariate GARCH algorithms to financial time series. Engle (2002) developed the dynamic conditional correlation GARCH, whereas Engle and Kroner (1995) and Engle et al. (1990) introduced the BEKK-GARCH and Factor-GARCH models respectively. As conditional variance is close to zero over a long time span (Bauwens et al. 2012), several extensions of the traditional GARCH model were proposed by Engle and Lee (1999), Haas et al. (2004a), Haas et al. (2004b) and Haas and Paolella (2012) in order to overcome this problem. It is also worth mentioning that Zhang et al. (2018) proposed a zero-drift model. This last variant of the algorithm studies heteroskedasticity and conditional heteroskedasticity together because it is non-stationary regardless of the sign of Lyapunov exponent.

Stock volatility can be also modelled with stochastic volatility models. This family assumes that volatility follows its own stochastic process. Heston Model is the most popular approach within this family. This model assumes that stock prices follow the following Brownian process:

$$dX_t = \mu X_t dt + \sqrt{\sigma_t^2} X_t dB_t \quad (40)$$

Where  $B_t \sim \mathcal{N}(0, \sigma_t^2 t)$ ,  $X_t$  is the stock price and  $\sigma_t^2$  the volatility. Heston (1993) also assumes that volatility is driven by an Ornstein-Uhlenbeck process and, thus, changes in volatility are determined by the following expression:

$$d\sigma_t^2 = \theta(v - \sigma_t^2)dt + \delta \sigma_t dB_t^* \quad (41)$$

As it can be derived from the previous formula, this model assumes that volatility has a long term mean ( $v$ ). The rate at which the variance reverts to this mean is

determined by  $\theta$ ,  $\delta$  is the volatility of  $\sigma_t^2$  and, finally,  $B_t^*$  is a Wiener process. Notice that  $B_t^*$  can be correlated with  $B_t$ , being  $\rho$  the level of correlation between them.

Other relevant stochastic volatility processes are the constant elasticity (Linetsky and Mendoza 2009) and SABR models (Hagan et al. 2002). The first process has the aim of modelling the leverage effect (volatility increases when the price falls). The second tries to capture the volatility smile in derivative markets.

It is also worth mentioning that stock volatility forecasting models based on machine and deep learning have increased significantly their relevance in the last years. Notice that GARCH models are not included in this last family even though they are part of the machine learning tool-kit. GARCH models have been presented as a complete different family due to its relevance in the field of stock volatility. Therefore, this family includes the stock volatility models based on algorithms such as neural networks, random forests, gradient boosting with trees or support vector machines. It also includes the so-called hybrid models (a merge of GARCH with other algorithms such as neural networks).

As previously stated, the literature related to this family of models have increased remarkably in the last years. Dias et al. (2019) applied support vector machines and hidden Markov models to forecast financial time series. Hamid and Iqbid (2002) demonstrated that neural networks can forecast implied volatility more accurately than Barone-Adesi and Whaley models. Roh (2006), Hajizadeh et al. (2012), Lu et al. (2016) Monfared and Enke (2014) and Kristjanpoller et al. (2014) merged the output of GARCH models with feed forward artificial neural networks to obtain more accurate volatility forecasts. Hybrid models have been also applied to forecast the volatility of oil (Kristjanpoller and Minutolo 2016) or metals (Kristjanpoller and Minutolo 2015, Kristjanpoller and Hernández 2017 and Vidal and Kristjanpoller 2020). The stock volatility forecasting models proposed by this thesis are based on hybrid algorithms. Therefore, they belong to this last family. The differences between the approaches introduced by this thesis and the existing models of this family are presented in Sections 3 and 5.

From an economical and management perspective, stock volatility forecasting and equity risk models have become more and more relevant in the last years. The main reasons are the following:

- Investors are not only interested in profits but also on the risk assumed by financial institutions. Since the Financial Crisis of 2007-2008, the risk-profit KPIs have become more and more relevant for investors and, thus, the market cap of these entities.
- Stimulus packages and banking bailouts were required in the latest financial crises, provoking an increase of the public debt. Therefore, the interest of governments on the risk management strategy of financial institutions have increased sharply, leading to a more robust regulation.

- During financial crises, some financial institutions were sold at sale price. Thus, banks and/or insurance entities can exploit the competitive advantage of having an appropriate risk management strategy. For example, during the Financial Crisis of 2007-2008 some of the most relevant US investment banks were sold at sale price to other financial institutions.

These models have become quite relevant for the decision-making process of financial institutions. Thus, more accurate equity risk models can lead also to more accurate management actions. This thesis has the aim of improving the current stock volatility forecasting and equity models by adapting and applying the latest machine and deep learning techniques.

## 2.4 Reserving in general insurance and machine learning

This section is focused on presenting the main existing models in the field of general insurance reserving. Sections 4 and 6 present the stochastic reserving models based on the methods and algorithms described in Section 2.2.

In contrast to most of the economical sectors, the cost of the product sold by insurance companies is unknown. These entities does not only have to estimate the cost of new clients but they also need to forecast the ultimate cost of the outstanding claims. This last estimation is key for understanding the profitability of general insurance long tail portfolios such as motor third party liability, engineering, credit and industrial property, among others. Indeed, the main source of volatility in the technical result of general insurance entities is the deviation between the actual and the expected cost. Given the importance of estimating an accurate outstanding claim and new client cost, pricing models were introduced to forecast an accurate premium while reserving models were developed to forecast the expected ultimate cost of outstanding claims and its variability.

Reserving models have evolved from deterministic methodologies toward stochastic models able to assess the variability of general insurance reserves. Chain Ladder is considered the main deterministic model. This technique takes into consideration the aggregated historical claim payments by development and accident or underwriting year to estimate the expected ultimate cost. The previous aggregation of data is usually called loss triangle. Table 1 shows an example of a cumulative loss triangle.

Table 1: Example of cumulative loss triangle in millions

	Development Year							
	1	2	3	4	5	6	7	8
Accident Year								
2013	1.90	2.15	2.28	2.35	2.42	2.45	2.45	2.45
2014	1.15	1.95	2.08	2.12	2.15	2.16	2.16	
2015	1.26	2.05	2.20	2.23	2.25	2.25		
2016	1.38	2.26	2.35	2.40	2.42			
2017	1.34	2.13	2.22	2.29				
2018	1.52	2.20	2.22					
2019	1.40	2.21						
2020	1.45							

Source: own elaboration

Chain Ladder methodology assumes that future cumulative payments,  $D_{ij}$ , or incurred cost have the following expected value:

$$E[D_{ij}] = \hat{f}_j D_{i,j-1} \quad (42)$$

where  $i = (1, 2, \dots, I)$  indicates the accident or underwriting year and  $j = (1, 2, \dots, I)$  the development year. The development factor  $\hat{f}_j$  is calculated as follows:

$$\hat{f}_j = \frac{\sum_{i=1}^{I-j+1} D_{ij}}{\sum_{i=1}^{I-j+1} D_{i,j-1}} \quad (43)$$

where  $\{\hat{f}_j : j = (2, 3, \dots, I)\}$ .

Stochastic reserving models have been fostered by regulations such as Solvency II or Swiss Solvency Test and the increasing importance of the risk management in financial institutions. Indeed, dividends of insurance entities are normally conditioned by the solvency ratio. The risk strategy of most of these companies requires a certain level of solvency ratio to distribute dividends.

The most popular stochastic reserving models are also based on the Chain Ladder technique and loss triangles. Mack (1993) developed a free-distribution model by limiting the reserve distribution analysis to the first two moments. This stochastic model based on Chain Ladder is specially interesting for the insurance industry because the analyst does not need to take any assumption regarding the theoretical distribution of the claim cost. Mack's model assumes that cumulative payments,  $D_{ij}$ , have the following expected value and variance:

$$E[D_{ij}] = \hat{f}_j D_{i,j-1} \quad \text{Var}[D_{ij}] = \hat{\sigma}_j^2 D_{i,j-1} \quad (44)$$

Chain Ladder and Mack's model expect the same cumulative payments. Therefore, the expected reserve of the stochastic distribution produced by the Mack's model converges to the value obtained by the deterministic Chain Ladder technique. The

variance of the stochastic model depends on  $\hat{\sigma}_j^2$ . As explained by Mack (1993), this parameter has the following expression:

$$\hat{\sigma}_j^2 = \frac{1}{I-j-1} \sum_{i=1}^{I-j+1} D_{i,j-1} \left( \frac{D_{ij}}{D_{i,j-1}} - \hat{f}_j \right)^2 \quad (45)$$

where  $\{\hat{\sigma}_j^2 : j = (2, 3, \dots, I)\}$ . Bootstrapping (England and Verrall 2006) can be applied to sample the reserve distribution. The following residuals need to be calculated to apply this method:

$$\hat{r}_{ij} = \frac{\sqrt{D_{i,j-1}} * \left( \frac{D_{ij}}{D_{i,j-1}} - \hat{f}_j \right)}{\hat{\sigma}_j} \quad (46)$$

As suggested by England and Verrall (2006), a bias adjustment is added to the previous residuals:

$$\hat{r}_{ij} = \sqrt{\frac{N}{N-p}} * \frac{\sqrt{D_{i,j-1}} * \left( \frac{D_{ij}}{D_{i,j-1}} - \hat{f}_j \right)}{\hat{\sigma}_j} \quad (47)$$

where  $N$  is the total number of residuals and  $p$  the number of parameters. Thus, the resampled link ratios are computed as follows:

$$f_{ij}^B = \hat{f}_j + r_{ij}^B \frac{\hat{\sigma}_j}{\sqrt{D_{i,j-1}}} \quad (48)$$

where  $B$  refers to the number of upper triangles to be simulated and  $r_{ij}^B$  to the residual resampled in the position  $(i, j)$  of the  $B^{th}$  triangle. The resampled development factors ( $\tilde{f}_j^B$ ) are calculated applying the Chain Ladder technique to  $D_{i,j}$  and  $f_{ij}^B$ .

The lower triangle ( $D_{i,j}$  where  $i + j > I + 1$ ) without process variance is predicted by combining  $f_j^B$  and the upper triangle ( $D_{i,j}$  where  $i + j \leq I + 1$ ). Then, the process variance is incorporated by adding the following to the lower loss triangle:  $\hat{\sigma}_j r_{ij}^B \sqrt{D_{i,j-1}}$ .

It is also worth mentioning that there are other stochastic models based on Chain Ladder and theoretical distributions such as overdispersed Poisson (Renshaw and Verrall 1998), log-normal (Kremer 1982), gamma (Mack 1991) and negative binomial (Verrall 2000). As in Mack's model, bootstrapping can be applied to obtain the reserve distribution from these models. If the bootstrapping procedure is to be avoided, England and Verrall (2006) introduced a stochastic Bayesian implementation of the overdispersed Poisson, negative binomial and Mack's model.

More recently, Meyers (2015) developed the Changing Settlement Rate (CSR) model, which is based on Markov Chain Monte-Carlo. In contrast to other stochastic models such as Mack or ODP, this approach gives the possibility of recognising a change in the claim settlement rate over the years. The default calibration and prior distributions suggested by the author are the following:

- $\alpha_i \sim N(\ln P_i + \log elr, \sqrt{10})$ , where  $\log elr \sim U(-1, 0.5)$  and  $P_i$  are the premiums by accident year.
- $\beta_j \sim U(-5, 5)$  for  $j = 1, \dots, J - 1$ . In the last development year,  $\beta_J = 0$ .
- $\mu_{i,j} = \alpha_i + \beta_j(1 - \gamma)^{i-1}$ , where  $\gamma \sim N(0, 0.025)$ .
- Each  $\sigma_j = \sum_{i=j}^J a_i$ , where  $a_i \sim U(0, 1)$ .

The cumulative payments simulated by this model follow a log-normal distribution,  $D_{i,j} \sim LN(\mu_{i,j}, \sigma_j)$ , subject to the constraint  $\sigma_1 > \sigma_2 > \dots > \sigma_J$ .

In addition to the models based on Chain Ladder and Markov Chain Monte-Carlo, there are general insurance reserving models based on machine and deep learning. Algorithms such as artificial neural networks (Gabrielli et al. 2018, Gabrielli and Wüthrich 2018 and Wüthrich 2018b), regression trees (Wüthrich 2018a), recurrent neural networks (Kuo 2018) or tree-based algorithms (Baudry and Robert 2019 and Lopez et al. 2019) have been used to predict claim reserves.

The differences between the reserving models introduced by this thesis and the methodologies explained above are presented in Sections 4 and 6.

From an economic and management point of view, reserving models are specially relevant for insurance entities due to the following main reasons:

- Differences between the reserve estimated by the models and the actual cost of outstanding claims can provoke significant fluctuations on the P&L. Obviously, this can also have an impact on dividends and, thus, on the market cap of financial institutions.
- Insurance companies need to forecast the ultimate cost of outstanding claims to estimate the profitability of the current portfolio/clients. As management decisions on the current portfolio are also based on the estimations from reserving models, they are key on the decision-making process of insurance entities.

This thesis has the aim of improving the current reserving models by adapting and applying the latest machine and deep learning techniques.

### **3 Forecasting volatility with a stacked model based on a hybridized Artificial Neural Network**

**Authors:** Eduardo Ramos-Pérez, Pablo J. Alonso-González and José Javier Núñez-Velázquez.

**Journal:** Expert Systems With Applications, vol: 129 1-9. ISSN 0957-4174.

**DOI:** 10.1016/j.eswa.2019.03.046

**Journal Impact Factor in 2019:** 5.452

**Rank by Journal Impact Factor in 2019:** 2/83 in Operations Research and Management Business (Q1-D1).

**Journal Citation Indicator in 2019:** 1.58

**Rank by Journal Citation Indicator in 2019:** 7/99 in Operations Research and Management Business (Q1-D1).

**Published:** September 2019

**Citations:** 19 (Web of Science), 26 (Google Scholar).

**Arxiv Repository:** <https://arxiv.org/abs/2006.16383>

This manuscript version is made available under the CC-BY-NC-ND 4.0 license  
<http://creativecommons.org/licenses/by-nc-nd/4.0/>

## Abstract

An appropriate calibration and forecasting of volatility and market risk are some of the main challenges faced by companies that have to manage the uncertainty inherent to their investments or funding operations such as banks, pension funds or insurance companies. This has become even more evident after the 2007-2008 Financial Crisis, when the forecasting models assessing the market risk and volatility failed. Since then, a significant number of theoretical developments and methodologies have appeared to improve the accuracy of the volatility forecasts and market risk assessments. Following this line of thinking, this paper introduces a model based on using a set of Machine Learning techniques, such as Gradient Descent Boosting, Random Forest, Support Vector Machine and Artificial Neural Network, where those algorithms are stacked to predict S&P500 volatility. The results suggest that our construction outperforms other habitual models on the ability to forecast the level of volatility, leading to a more accurate assessment of the market risk.

**Keywords:** Machine learning, Stacking algorithms, Risk assessment, Volatility forecasting, Hybrid models

**AMS Subject Classification:** 62-07, 62P05, 65C60, 90-08.

## 3.1 Introduction

During the Financial Crisis of 2007-2008, unexpected falls in stock prices resulted in significant losses for individual investors and financial institutions. Since then, new regulations have entered in force in order to ensure the correctness of the market risk assessment provided by financial institutions and to allow individual market participants to be aware of the risk linked to financial products. As volatility is an indicator of the uncertainty associated with the asset profitability (Hull 2015 and Rajashree and Ranjeeeta 2015), this variable tends to play a key role within the risk models. In fact, events like the bankruptcy of LTCM in 1998 (Lowenstein 2000), the dotcom crash in 2001 (Aharon et al. 2010) or, more recently, the aforementioned Financial Crisis of 2007-2008 were not foreseen by most of the risk models due to inaccurate estimates produced by the volatility forecasting models. It is worth mentioning that, as volatility is not directly observed, before estimating any statistical model it is necessary to select a volatility proxy (Poon and Granger 2003). In the following paragraphs, the proposed methodology and main families of volatility forecasting models (GARCH, Stochastic and Machine Learning) are presented.

First of all, GARCH models are introduced as this family of models is probably the most widely used in the literature due to its ability to fit the volatility clustering (Mandelbrot 1963) empirically observed in financial time series. This auto-regressive approach and its generalization were developed by Engle (1982) and Bollerslev (1986) respectively. Classical GARCH models were discovered to be too rigid for fitting returns series, especially over a long time span, because the estimated persistence of conditional variances is close to one (Bauwens et al. 2012). Therefore, more flexible GARCH models were developed in order to overcome this problem. Engle and Lee (1999) suggested a two equation model where each of them represents long-run

and short-run components of volatility, respectively. Mixed-normal GARCH (Haas et al. 2004a) is a second way to deal with this problem. This kind of model allows to choose amongst several regimes in each instant of time  $t$ . The drawback of this methodology is that it assumes that the variables used to decide amongst regimes are all independent over time. To overcome this problem, Haas et al. (2004b) proposed a Markov-switching model where the parameters of a GARCH model change according to a Markov process. An extension of this kind of model can be found in Haas and Paoletta (2012). Before concluding with the GARCH models, it is important to mention that volatility can behave differently depending on the trend of the market: bullish or bearish. To fit this behaviour, Nelson (1991) developed the EGARCH model that allows the sign and the volume of previous values to have separate impacts on the volatility forecasts. In addition to the EGARCH model, Glosten et al. (1993) proposed the GJR-GARCH to replicate the aforementioned behaviour. Other developments within this family can be found in Engle and Kroner (1995) with their BEKK model, the factor model (Engle et al. 1990), the Constant Conditional Correlation model (Bollerslev 1990), the time-varying correlation model (Tse and Tsui 2002), the dynamic correlation model (Engle 2002) or the multivariate GARCH approach proposed by Kraft and Engle (1982) and Engle et al. (1984) and its financial implementation by Bollerslev et al. (1988). More recently, Zhang et al. (2018) have proposed a first order zero drift GARCH (ZD-GARCH) to study heteroscedasticity and conditional heteroscedasticity together.

The second family is composed of those models which assume that the volatility is driven by its own stochastic process. This approach was introduced by Taylor (1982) as an Euler approximation of the underlying diffusion model. Assuming that stock prices follow a Brownian motion, Heston (1993) derived a model where the volatility follows an Ornstein-Uhlenbeck process. To derive the parameters of the Heston Model, two different strategies have been adopted in the literature: moment or simulation. For the first one, the Generalized Method of Moments was proposed by Melino and Turnbull (1990) and Andersen and Sorensen (1999), while the simulation approach has been used by Danielsson (2004), Durbin and Koopman (1997), Broto and Ruiz (2004) or Andersen (2009), amongst others.

The last family presented is Machine Learning, which comprises a set of techniques used to analyse the future evolution of stock prices and volatility. These algorithms try to learn automatically and recognize patterns in a large amount of data (Krollner et al. 2010). It is worth mentioning that the fitting of these algorithms is quite sensitive to the forecasting time-frame and the selected input variables. Armano et al. (2005) and de Faria et al. (2009) suggest using one day as a time-frame and lagged or technical indicators as input variables for the Machine Learning algorithms. Stock prices, volatilities and portfolio selection have been analysed using different methodologies based on Machine Learning, such as Support Vector Machine (Gestel et al. 2001), hidden Markov models (Gupta and Dhingra 2012 and Dias et al. 2019) or Artificial Neural Networks (ANN) (Hamid and Iqbid 2002). These last authors showed that volatility forecasts made by an ANN outperform the implied volatility derived from Barone-Adesi and Whaley options models. Additionally, ANNs have been ap-

plied successfully to other financial series different from volatility and stock prices: bond rates (Surkan and Xingren 2001) and bank failures (Hutchinson et al. 1994). Deep learning (LeCun et al. 2015a) is a framework closely related with ANN which has been employed for predicting the evolution of Korean stock market index (Chang et al. 2017).

Despite the high performance of ANN, predictions derived from the use of this algorithm could be inaccurate when stock prices move sharply (Patel and Yalamalle 2014). To overcome this problem, ANN were combined with other statistical models (Kristjanpoller et al. 2014) creating the so called hybrid models. Hybridization can be defined as an approach in which several models are merged to form a new enhanced model in order to produce better forecasting results. Therefore, a hybrid model is a combination of the artificial intelligence techniques with some components of the traditional forecasting models (like the ones presented within the GARCH family). Examples of this approach are discussed in Roh (2006), Hajizadeh et al. (2012), Lu et al. (2016) Monfared and Enke (2014) or Kristjanpoller et al. (2014), where different outputs from a GARCH-based model are used as inputs in an ANN. A more general picture of this type of hybrid models is provided by Bildirici and Ersin (2009), since they compared and combined an ANN with different types of GARCH models (GARCH, EGARCH, GJR-GARCH, TGARCH, NGARCH, SAGARCH, PGARCH, APGARCH and NPGARCH). In addition to the above-mentioned researches, this type of hybrid models has been broadly used in other papers. Bildirici and Ersin (2014) proposed a MS-GARCH with an ANN to improve the forecasting accuracy, Bektipratiwi and Irawan (2011) combined a radial basis function with an EGARCH to model stocks returns of an Indonesian bank and Arneric and Poklepovic (2016) developed an ANN model as an extension of a GJR-GARCH to forecast the market returns of six European emerging markets. GARCH-based models have been also combined with ANNs to predict the volatility in commodity markets, such as gold (Kristjanpoller and Minutolo 2015) or oil (Kristjanpoller and Minutolo 2016). In this last case, the hybrid model included financial variables to improve the forecasts. This strategy can also be found in Kristjanpoller and Hernández (2017). Kim and Won (2018) propose a hybrid model that combines a LSTM with various GARCH-type models to forecast the volatility of KOSPI index. A refinement of this model can be found in Back and Kim (2018). It should be mentioned that these models can be generated in both directions: some outputs of a GARCH model can be used as input of an ANN and vice versa (Lu et al. 2016). Finally, it should be noted that hybridization can not only be made with ANN. Peng et al. (2018) proposed a structure combining traditional GARCH-models with Support Vector Machine (SVM) (Cortes and Vapnik 1995).

The research carried out along this paper develops a volatility forecasting model that consists of two different levels which is based on stacking algorithms methodology (Hastie et al. 2009) and statistical models of the Machine Learning family. Random Forest (RF) (Breiman 2001), Gradient Boosting (GB) with regression trees (Friedman 2000) and Support Vector Machine (SVM) (Cortes and Vapnik 1995) are used in the first level, while an ANN (Mcculloch and Pitts 1943) is incorporated within the

second level of the stacked model (Stacked-ANN) in order to generate the volatility forecasts. A different two-level approach can be found in Kristjanpoller and Minutolo (2018). They use an ANN-GARCH model with a pre-processing based on principal components analysis to reduce the number of inputs employed in their network. In contrast to the hybrid models defined previously, the proposed model is merging the results arising from other machine learning algorithms which are free of some theoretical assumptions like the use of a predefined distribution for the underlying asset returns or the constant level of unconditional variance. Because of this and with the aim to build a more flexible model, the GARCH-based models are not present in the Stacked-ANN architecture. The proposed model relies completely on the predictions made by machine learning algorithms and market data. Additionally, in the case of the Stacked-ANN the final forecasts made by the first level algorithms are directly used as inputs within the ANN while, in most of the hybrid models discussed in the previous paragraphs, sections of the GARCH-based models are inserted separately in the ANN.

The rest of the paper proceeds as follows: Section 3.2 presents the set of volatility forecasting models used for comparison purposes. Furthermore, the risk measures and tests used to validate the results are discussed. In Section 3.3 the theoretical background and architecture of the volatility forecasting model based on stacking algorithms (Stacked-ANN) are explained. The empirical results of the different forecasting models are shown in Section 3.4, where the accuracy and the risk measures arising from the proposed model are compared with results obtained by the methodologies explained in Section 3.2. Finally, Section 3.5 presents the main conclusions of the results and comparisons carried out along Section 3.4.

### 3.2 Benchmark models, risk measurements and statistical tests

As stated above, this section is focused on explaining the benchmark models and the tests used to back-test the risk measurements. Thus, the first paragraphs are dedicated to ANN, ANN-GARCH, ANN-EGARCH and Heston Model, while the end of this section is focused on the risk measurements and tests performed to validate and compare the results of the benchmark models with the one proposed in Section 3.3.

The first benchmark model is a feed-forward ANN. Following the notation provided by Bishop (2006) and assuming that the algorithm has two hidden layers, the model would be defined by the following expression:

$$\hat{\sigma}_{t+1} = h^{(3)} \left( \sum_{k=1}^T w_{p,k}^{(3)} h^{(2)} \left( \sum_{j=1}^M w_{k,j}^{(2)} h^{(1)} \left( \sum_{i=1}^D w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right) + w_{p,0}^{(3)} \right) \quad (49)$$

Where  $h^{(n)}$  is the activation function associated with the layer  $n$ ,  $w_{z,v}^{(n)}$  is the  $v$ -th weight associated with the neuron  $z$  inside the layer  $n$  and  $x_i$  refers to the  $i$  input variable of database comprised by the explicative variables selected by the analyst.

The second benchmark model is an ANN-GARCH( $p, q$ ). As briefly introduced in Section 3.1, the aim of this hybrid model is to combine the GARCH( $p, q$ ) estimates with other input variables by using an ANN, which is a more flexible model than GARCH( $p, q$ ). Therefore, before starting with the fitting of the ANN, the parameters of the GARCH( $p, q$ ) model need to be estimated:

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^q \alpha_i r_{t-i}^2 + \sum_{i=1}^p \beta_i \sigma_{t-i}^2 \quad / \quad \hat{r}_t = \hat{\sigma}_t \epsilon_t \quad (50)$$

In this formulation  $\omega$ ,  $\alpha_i$  and  $\beta_i$  are the parameters to be estimated, while  $r_t$  and  $\sigma_t^2$  refer to the return and volatility respectively. The returns distribution is determined by the distribution selected for  $\epsilon_t$ . If a standardize normal or standardize Student's t-distribution is selected, then the returns generated by the model follow a conditional normal (CND) or conditional t-distribution (CTD) respectively (Bauwens et al. 2012). Once the GARCH( $p, q$ ) parameters are estimated,  $\sum_{i=1}^q \alpha_i r_{t-1}^2$  and  $\sum_{i=1}^p \beta_i \sigma_{t-1}^2$  can be computed and used as input (together with the rest of explicative variables) within the ANN.

The third benchmark model is an ANN-EGARCH. The architecture of this model and the previous one can be considered the same with the unique difference that the first step consists of fitting an EGARCH( $p, q$ ) instead of a GARCH( $p, q$ ) model. The EGARCH( $p, q$ ) can be defined as follows (Nelson 1991):

$$\log \hat{\sigma}_t^2 = \omega + \sum_{i=1}^p \alpha_i \log \hat{\sigma}_{t-i}^2 + \sum_{i=1}^q (\beta_i e_{t-i} + \gamma_i (|e_{t-i}| - E |e_{t-i}|)) \quad (51)$$

Once the EGARCH is fitted, the following terms can be calculated and used as input within the ANN together with the rest of the explicative variables selected by the analyst:

$$\sum_{i=1}^p \alpha_i \log \hat{\sigma}_{t-i}^2 \quad \sum_{i=1}^q \beta_i e_{t-i} \quad \sum_{i=1}^q \gamma_i (|e_{t-i}| - E |e_{t-i}|) \quad (52)$$

The last benchmark is the Heston (1993) Model. Even though this approach belongs to the stochastic family and the proposed one to the Machine Learning one, this model is going to be used as benchmark during this paper as this process is the most widely used within the family of the stochastic volatility models. It assumes that changes in stock prices through the time ( $dX_t$ ) follow a Brownian diffusion process:

$$dX_t = \mu X_t dt + \sqrt{\sigma_t^2} X_t dB_t \quad (53)$$

Where  $B_t \sim \mathcal{N}(0, \sigma_t^2 t)$ . Therefore, if volatility follows an Ornstein-Uhlenbeck process, the changes in this variable are defined by the following expression:

$$d\sigma_t^2 = \theta(v - \sigma_t^2)dt + \delta \sigma_t dB_t^* \quad (54)$$

where  $v$  is the long term volatility,  $\theta$  is the rate of return to  $v$ ,  $\delta$  is the volatility of  $\sigma_t^2$  and  $B_t^*$  is a Wiener process that has a correlation of  $\rho$  with  $B_t$ .

Once the four benchmark models have been explained, the section focuses on the risk measurements. As stated before, volatility plays a key role in market risk assessment. Therefore, the models will not be only compared in terms of accuracy, but the risk measurements arising from every volatility model are going to be tested. For this purpose, VaR and CVaR have been selected as risk measures. Even though VaR is probably the most used metric due to its simplicity and easy interpretation, CVaR has been also included as it is considered to be a coherent risk measure (Artzner et al. 1999). Consequently, for every volatility model the aforementioned risk measures are going to be computed and validated by means of the following tests:

- Kupiec (1995) introduced a test in order to check if the number of VaR excesses are align with the level of confidence selected.
- An extension of the previous test was developed by Christoffersen et al. (1997). The aim of this test is to validate that VaR excesses are independent, identically distributed and in line with the selected level of confidence.
- Acerbi and Szekely (2014) developed a test (AS1) to assess the appropriateness of the CVaR based on the assumption that VaR has been already tested and considered to be correct from a statistical point of view. The test is inspired by the following equation:

$$E \left[ \frac{r_t}{CVaR_{\alpha,t}} + 1 \middle| r_t + VaR_{\alpha,t} < 0 \right] = 0 \quad (55)$$

As VaR needs to be previously validated, the result of this test has to be assessed together with the two aforementioned tests.

- In addition to the previous test, Acerbi and Szekely (2014) introduced another method (AS2) to validate the CVaR without making any assumption about the appropriateness of the VaR. To do so, this test tries to check a CVaR expression that is not conditioned by the correctness of a previous VaR estimate.

Before beginning with the Stacked-ANN architecture, it is worth noticing that the two first tests are parametric while the two last are non-parametric so, for further details about how to compute the statistics and their distributions please refer to aforementioned papers.

### 3.3 Stacked model

This section has been divided in several sub-sections in order to explain sequentially the proposed volatility forecasting model. As the Stacked-ANN model is composed by two different levels, the two first sub-sections are dedicated to the input data and the algorithms within the first level of the Stacked-ANN model, while the third and forth sub-sections are focused on the data required to generate the stacking procedure and the details of the ANN fitted with the aforementioned information. (Figure 12 explains briefly the process followed to estimate and test the Stacked-ANN model)

### First level: Input data

The first step is concerned with the creation of the database containing the volatility proxy to be used as a response and the explanatory variables selected to fit the algorithms. As the aim of the study is to predict future volatilities, the True Realized Volatility (hereinafter TRV) is going to be used as response variable (Roh 2006):

$$TRV_t = \sqrt{\frac{1}{n} \sum_{i=1}^n (r_{t+i-1} - \hat{r}_t)^2} \quad (56)$$

Where  $\hat{r}_t = \sum_{i=1}^n (r_{t+i-1})/n$  and  $n = 5$ . The window has been selected to be large enough to compute a stable TRV and small enough to avoid, as much as possible, mixing different volatility regimes.

The variables given to the first level algorithms to forecast the TRV are the last 30 volatilities computed with returns already observed in the market:

$$V_t = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} (r_{t-n+i} - \hat{r}_t)^2} \quad (57)$$

Where  $\hat{r}_t = \sum_{i=0}^{n-1} (r_{t-n+i})/n$  and  $n = 5$ . Only the last 30 volatilities have been selected because the correlations between previous volatilities and the TRV are residual and therefore their explanatory power is considered to be non-significant. The historical data to compute all the aforementioned variables is obtained by using the `quantmod` (Ryan and Ulrich 2017) package from the R project (R Core Team 2017) and, as suggested by Hastie et al. (2009), they will be scaled to the range  $[0, 1]$  to improve the training of the algorithms.

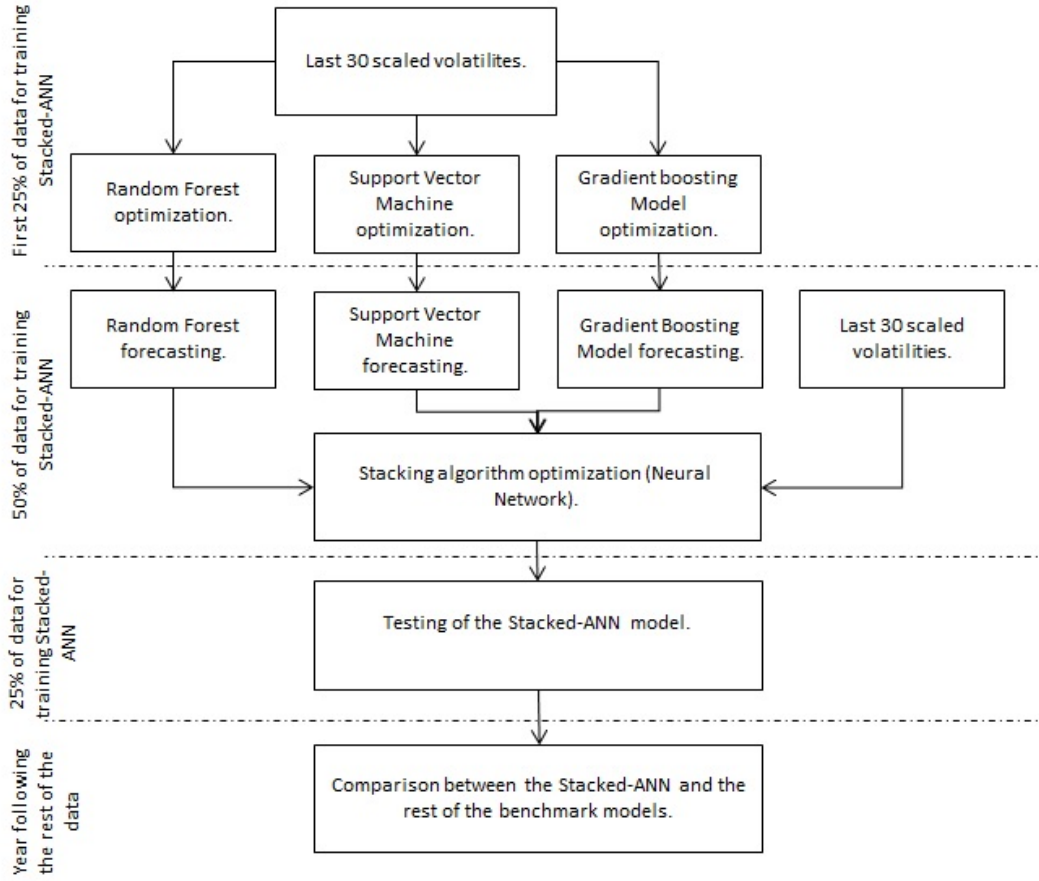
Before beginning with the section related with the algorithms included within the first level, it is important to mention that the first 25% of the data is used to fit the first level algorithms, the next 50% is dedicated to the ANN estimation and the last 25% is the test set. The comparison of the benchmark models with the proposed one in terms of accuracy and risk measurement will be made with a different set of data containing the information of the following year (e.g. if data from 2000 to 2007 is used to train and test the Stacked-ANN model, the out of sample data selected for comparison purposes would be market movements happened during 2008).

### First level: Individual algorithms

The methods applied to optimize the hyper-parameters of the algorithms within the first level of the Stacked-ANN architecture are introduced below:

- Minimization of the Mean Square Error (hereinafter, MMSE) for the whole database to train the first level algorithms.

Figure 12: Stacked-ANN model structure



- Circular Block Bootstrap (CBB). This method (Politis and Romano 1991) generates new samples by selecting random blocks from the original database. The length of these blocks is fixed and the procedure to calculate it was introduced by Politis and White (2004) and Patton et al. (2009). CBB can only be applied to stationary time series.
- Stationary Bootstrap (hereinafter, SB) (Politis and Romano 1994). Similar to the case of CBB, this method can only be used with stationary time series. However, the difference with the former method is that the length of the blocks instead of being fixed, it is randomly selected with a certain average that can be calculated using different approaches (see Politis and White 2004 and Patton et al. 2009).
- Maximum Entropy Bootstrap (hereinafter, MEB) (Vinod 2006 and Vinod and de Lacalle 2009). Unlike the two previous approaches, stationarity is not required as the new samples are obtained from the maximum entropy distribution of the original time series.
- H Cross-Validation (HCV). This method introduced by Chu and Marron (1991)

tries to avoid the effect of the correlation that can exist between the response and the explanatory variables while dealing with time series by eliminating  $h$  data points between them. The bandwidth selection is obtained minimizing the absolute autocorrelation between the response and explanatory variables, with a maximum width of 100 days.

The optimum hyper-parameters combination of each one of the five previous methods is obtained by applying grid search. Then, these combinations are tested against data out of sample (the following 50% of the database) to choose the most accurate option for fitting the algorithm.

As stated before, the first level of the stacked model architecture is composed by three algorithms: Random Forest (RF) (Breiman 2001), Gradient Boosting with regression trees (GB) (Friedman 2000) and Support Vector Machine (SVM) (Cortes and Vapnik 1995).

## Second level: Input data

As explained in Section 3.3, the first 25% percent of data is dedicated to fit the first level algorithms while the following 50% and 25% are used for fitting the ANN and testing the results respectively. The explanatory variables given to the ANN are:

- As with the first level algorithms, the last 30 volatilities ( $V_t, V_{t-1}, \dots, V_{t-29}$ ) scaled to the range  $[0, 1]$ .
- The True Realized Volatility forecasts made by the first level algorithms: Random forest ( $\widehat{TRV}_{t,RF}$ ), Gradient boosting ( $\widehat{TRV}_{t,GB}$ ) and Support Vector Machine ( $\widehat{TRV}_{t,SVM}$ ).

The response variable is the  $TRV_t$  as defined in Section 3.3.

## Second level: Stacking algorithm

As stated previously, the last step of the Stacked-ANN model is the fitting of the ANN, which is the algorithm stacking the forecasts made by the RF, GB and SVM. Before starting with the details of the ANN architecture, notice that the methods and procedures related to the hyper-parameters optimization are the same as the first level algorithms: Grid search in combination with the methods explained in Section 3.3 and final hyper-parameters decision based on the out of sample error (last 25% of the database).

Below, the main characteristics and details of the stacking algorithm are presented:

- The feed-forward ANN has two hidden layers with 20 and 10 neurons respectively. The sigmoid activation function has been selected for all the neurons within the hidden layers while the linear activation function has been used in the output layer, which is comprised by one neuron.

- The optimization algorithm selected is Adaptive Moment Estimation (ADAM), which was created by Kingma and Ba (2014). This method consists in a progressive adaptation of the initial learning rate, taking into consideration current and previous gradients. The default calibration proposed by the authors is applied:  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ .
- The number of epochs are 10,000 and the batch size is equal to the length of the data used for training the ANN.
- The backward pass calculations are done according to the selection of root mean squared error as a loss function.
- As indicated in Section 3.3, the 50% of the information is selected for training the ANN while the following 25% of the data is the test set. Note that the first 25% of the data is used to fit the first level algorithms.
- The parameter adjusting the level of L2 regularization ( $\phi$ ) and the initial learning rate  $\lambda$  used within ADAM are the hyper-parameters to be optimized during the estimation process.

Taking into consideration all the above-mentioned details, the  $TRV_t$  forecasted by the Stacked-ANN model (S-ANN) is obtained by means of the following expression:

$$\begin{aligned} \widehat{TRV}_{t,S-ANN} &= \widehat{f}(\widehat{TRV}_{t,RF}, \widehat{TRV}_{t,GB}, \widehat{TRV}_{t,SVM}, V_t, V_{t-1}, \dots, V_{t-29}) = \\ &= h^{(3)} \left( \sum_{k=1}^{10} w_{1,k}^{(3)} h^{(2)} \left( \sum_{j=1}^{20} w_{k,j}^{(2)} h^{(1)} \left( \sum_{i=1}^{33} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right) + w_{1,0}^{(3)} \right) \end{aligned} \quad (58)$$

As explained in Section 3.3,  $x_i$  are the last 30 volatilities scaled to the range  $[0, 1]$  and the forecasts made by the first level algorithms.

### 3.4 Results

During this section, the data used in the empirical analysis, the fitting process and the final comparison between the Stacked-ANN and the benchmark models are shown.

#### Data

In order to analyse the models under different market conditions, the algorithms have been trained and tested five different times with the S&P 500 volatilities observed in the following periods: 2000-2007, 2001-2008, 2002-2009, 2009-2016 and 2010-2017. As stated in Section 3.3, during the training and testing of the models the first 25% of the periods selected is dedicated to fit the first level algorithms, the next 50% is used to optimize the ANN while the last 25% is reserved for testing purposes. The year after the aforementioned periods (2008, 2009, 2010, 2017 and 2018 respectively for each period) has been used to compare the out of sample results of the Stacked-ANN with the benchmark models. The first three data-sets have been selected in order to analyse the performance of the models during the years after the financial

crisis, when the markets were dominated by a high volatile regime. Although the years influenced by the financial crisis are valuable to test the accuracy of the volatility forecasting models, the two last data-sets have been selected in order to analyse the models performance with the most recent data. Additionally, the lower level of volatility during the last periods, especially in 2017, allows to assess the robustness of the models by analysing them in different market conditions. In order to support the explanations given during this paragraph, Table 2 summarizes the moments of the TRV during the different periods selected to compare the models:

Table 2: True Realised Volatility statistics

Period	Mean	STD	Skewness	Kurtosis
Year 2008	0.022	0.016	1.510	4.519
Year 2009	0.015	0.008	0.853	3.248
Year 2010	0.010	0.006	0.854	3.736
Year 2017	0.004	0.002	0.911	3.369
Year 2018	0.009	0.006	1.406	4.702

*Source:* own elaboration

In addition, the Kolmogorov-Smirnov test has been applied sequentially to the TRV in order to assess statistically if the behaviour of the volatility changes over the different periods. As 2008 is the year when the most extreme events related with crisis happened and the market changed from a low to a high volatile regime, the skewness and mean of that year volatility is higher than the one related with 2009. Because of that, the aforementioned test reveals that the volatility of 2008 and 2009 do not belong to the same distribution ( $KS_{p-value} = 0.001$ ). However, when comparing the volatility of 2009 with the 2010 one, the test indicates that they come from the same distribution ( $KS_{p-value} = 0.690$ ). Even though the volatility follows a downward trend, both years are heavily conditioned by the events occurred during 2008 and therefore the test accepts the hypothesis that volatilities belong to the same distribution. Finally, the pair comprised by the volatilities of 2017 and 2018 shows an upward trend. Nevertheless, this increase is not big enough to reject the hypothesis that they come from the same distribution ( $KS_{p-value} = 0.167$ ).

The use of some of the methods proposed in Section 3.3 requires the time series to be stationary. Therefore, before using block bootstrap it has been checked if historical volatility satisfies this property by applying the Augmented Dickey-Fuller test (Dickey and Fuller (1979)) to the different data-sets dedicated to fit the algorithms within the first and second level. The results are shown in Table 3:

As the critical values are  $-2.63$  and  $-3.43$  with a probability of 5% and 1% respectively, it can be concluded that the data meet the requirements imposed by CBB and SB methods.

Previously to the fitting of the algorithms, the parameters needed for the different

Table 3: Augmented Dickey-Fuller Test

Period	ADF statistic: Data for training 1st level	ADF statistic: Data for training 2nd level
(2000-2007)	-6.61	-6.41
(2000-2008)	-6.13	-7.91
(2000-2009)	-5.25	-7.57
(2009-2016)	-4.72	-7.27
(2010-2017)	-4.58	-8.82

*Source:* own elaboration

bootstrap and cross validation methods are obtained by means of the methodologies presented in Section 3.3. As the Stacked-ANN architecture is comprised by two different levels, the length of blocks for CBB, the average of the blocks for SB and the distance,  $h$ , to be used within the HCV method are obtained for both, the data-set to fit first level algorithms and the one dedicated to the second level. Table 4 summarizes the former parameters and it shows non-significant changes over time for the different periods and levels:

Table 4: Calibration of the elements for bootstrap and CV

Method	Period	Data for training 1st level algorithms	Data for training 2nd level algorithm
CBB Block	(2000-2007)	28	63
CBB Block	(2001-2008)	36	58
CBB Block	(2002-2009)	40	56
CBB Block	(2009-2016)	39	58
CBB Block	(2010-2017)	38	30
SB Block average	(2000-2007)	25	55
SB Block average	(2001-2008)	32	51
SB Block average	(2002-2009)	35	49
SB Block average	(2009-2016)	34	51
SB Block average	(2010-2017)	33	27
HCV length	(2000-2007)	26	31
HCV length	(2001-2008)	31	51
HCV length	(2002-2009)	31	40
HCV length	(2009-2016)	32	55
HCV length	(2010-2017)	35	27

*Source:* own elaboration

## Fitting of the Stacked-ANN model

As explained in Section 3.3, different approaches have been followed to find the optimum hyper-parameter combination. Table 5 shows the methods that minimize the out of sample error per each algorithm and period:

Table 5: Methods optimizing OOS error

Period	Stacking Algorithm (ANN)	Random Forest	Gradient Boosting	Support Vector Machine
(2000-2007)	ME	SB	CBB	SB
(2001-2008)	CBB	CBB	CBB	SB
(2002-2009)	CBB	CBB	CBB	CBB
(2009-2016)	HCV	HCV	HCV	SB
(2010-2017)	SB	CBB	SB	SB

*Source:* own elaboration

Regardless of the period, the empirical results suggest that CBB and SB outperform the rest of the methods. These outcomes are expected as these two methods based on re-sampling blocks from the original database are specifically prepared to work with stationary time series. Table 6 summarizes the hyper-parameters suggested by the methods shown in Table 5:

Table 6: Final hyper-parameters

Period	Stacking Algorithm (ANN)	Random Forest	Gradient Boosting	Support Vector Machine
(2000-2007)	$\phi = 0$	$N = 10$	$B = 1479$	$\gamma = 0.0001$
	$\lambda = 0.0033$	$Obs = 24$	$\lambda = 0.003$	$\epsilon = 0.45$
(2001-2008)	$\phi = 0.01$	$N = 10$	$B = 3000$	$\gamma = 0.0001$
	$\lambda = 0.0059$	$Obs = 107$	$\lambda = 0.001$	$\epsilon = 0.55$
(2002-2009)	$\phi = 0$	$N = 1$	$B = 3583$	$\gamma = 0.0004$
	$\lambda = 0.0136$	$Obs = 37$	$\lambda = 0.001$	$\epsilon = 0.17$
(2009-2016)	$\phi = 0.02$	$N = 30$	$B = 1000$	$\gamma = 0.0002$
	$\lambda = 0.085$	$Obs = 118$	$\lambda = 0.009$	$\epsilon = 0.13$
(2010-2017)	$\phi = 0.01$	$N = 7$	$B = 1000$	$\gamma = 0.0001$
	$\lambda = 0.011$	$Obs = 175$	$\lambda = 0.003$	$\epsilon = 0.54$

*Source:* own elaboration

Where  $\lambda$  is the learning rate of the ANN and GB,  $\phi$  the parameter adjusting the level of L2 regularization of the ANN,  $B$  the number of iterations performed while fitting the GB,  $N$  the number of variables randomly selected by the RF and  $Obs$  the minimum number of observations to be kept in the terminal nodes of every fitted tree within the RF architecture. Finally,  $\gamma$  refers to the parameter included within the radial basis function kernel (the lower the parameter, the higher the non-linearity) and  $\epsilon$  defines the threshold where the error begins to be penalized by the SVM.

## Comparison against benchmark models

Once the Stacked-ANN is fitted, its performance is compared with the benchmark models explained in Section 3.2 (ANN, ANN-GARCH(1,1), ANN-EGARCH(1,1) and Heston Model). Before beginning with the comparisons, the three following remarks about the benchmark models have to be done:

- Due to the nature of the Heston Model, 20,000 simulations per each day have been computed and the daily average of them has been taken to assess its accuracy.
- The GARCH(1,1) and EGARCH(1,1) (included in the ANN-GARCH(1,1) and ANN-EGARCH(1,1) architecture respectively) have been estimated assuming Student-t innovations.
- The fitting procedure and architecture of the ANNs included within ANN-GARCH(1,1), ANN-EGARCH(1,1) and ANN models are the same as the ones explained for the Stacked-ANN (see Section 3.3).

Table 7: Accuracy analysis

Model	RMSE: 2008	RMSE: 2009	RMSE: 2010	RMSE: 2017	RMSE: 2018
Stacked-ANN	0.01192	0.00534	0.00494	0.00254	0.00544
ANN-EGARCH	0.01332	0.00588	0.00537	0.00276	0.00571
ANN-GARCH	0.01335	0.00584	0.00539	0.00263	0.00575
Heston	0.02066	0.00714	0.00547	0.00359	0.00610
ANN	0.01526	0.00615	0.00541	0.00274	0.00590

*Source:* own elaboration

Table 7 shows the out of sample error of the different periods selected to compare the performance and robustness of the Stacked-ANN with the benchmark models. The results shown in this table suggest the following conclusions:

- Regardless of the period, the Stacked-ANN outperforms other hybrid models based on auto-regressive methodologies like ANN-GARCH and ANN-EGARCH. In relative terms, minor deviations are observed between the different periods.
- All the hybridized models tend to outperform the pure ANN model.

- As expected due to the extremely high volatilities observed during the financial crisis, the results show that, regardless of the model, 2008 forecasts are less accurate. All the models minimize their error rate in the year with the lowest level volatility, 2017.
- The forecasts made by the Heston Model tend to be the less accurate due to the non-predictive nature of this model.

In addition to the above-mentioned analysis, the risk measures obtained by using each one of the volatility models are tested. In order to do so, a returns distribution is selected for each one of the forecasting volatility methods. As described in Section 3.2, Heston Model requires the changes in stock prices to follow a Brownian diffusion process. Nevertheless, for the rest of the benchmark models and the Stacked-ANN (which are free of assumptions about the returns) a Student t-distribution has been combined with the different volatility forecasts. This assumption about Student t-distribution has been selected when possible as returns tend to be leptokurtic and heavier-tailed than Normal distribution (McNeil et al. 2015).

Before analysing the results of the tests presented in Section 3.2, it is worth mentioning that the level of confidence (99%) and number of days (10) selected are based on the ones set by Basel Directive, whose aim is to monitor, amongst others, the market risk. Table 8 shows the p-value of the tests dedicated to VaR (Kupiec and Christoffersen) and CVaR (AS1 and AS2). If a 95% is set as confidence level, Stacked-ANN in combination with Student t-distribution is the only model that produces an appropriate p-value for Kupiec, AS1 and AS2 tests in every period under analysis. All the models show difficulties to produce a p-value higher or equal than 0.05 for the Christoffersen test because VaR exceedances tend to happen in a short period of time instead of spread over the period analysed. It is worth mentioning that the hybrid models taken as benchmark (ANN-EGARCH and ANN-GARCH) also fail in producing an appropriate value for the Kupiec test in several periods while, as stated before, the proposed hybrid model (Stacked-ANN) pass the test for every period. Finally, Heston Model tends to produce less appropriate risk measures due to the distribution constrain mentioned previously.

### 3.5 Conclusions

This paper introduces a Stacked-ANN model based only on Machine Learning techniques with the aim to improve the accuracy of the volatility forecasts made by other hybrid models based on a combination of GARCH or EGARCH with ANNs. Its predictive power and performance has been tested in terms of RMSE, VaR and CVaR.

Two main results have to be pointed out. Firstly, the Stacked-ANN has been able to generate more accurate volatility forecasts than other models in a high volatile regime period like the one occurred after the Financial Crisis of 2007-2008. The models outperformed by the Stacked-ANN during that time lapse are other hybrid models like ANN-GARCH and ANN-EGARCH, the most widely used stochastic volatility theory (Heston Model) and a feed-forward ANN without any combination with other

Table 8: P-value of the VaR and CVaR tests

Model	Test	Period: 2008	Period: 2009	Period: 2010	Period: 2017	Period: 2018
Stacked-ANN	Kupiec	0.85	0.84	0.65	0.85	0.85
	Christ.	0.01	0.79	0.02	0.01	0.01
	AS1	0.66	0.85	0.61	0.90	0.91
	AS2	0.56	0.63	0.36	0.67	0.69
ANN-EGARCH	Kupiec	0.12	0.12	0.84	0.03	0.03
	Christ.	0.00	0.00	0.01	0.03	0.03
	AS1	0.52	0.85	0.61	1.00	1.00
	AS2	0.07	0.19	0.62	0.91	0.91
ANN-GARCH	Kupiec	0.12	0.03	0.01	0.03	0.03
	Christ.	0.00	0.03	0.00	0.03	0.03
	AS1	0.51	1.00	0.77	1.00	1.00
	AS2	0.08	0.92	0.05	0.85	0.89
Heston Model	Kupiec	0.00	0.00	0.65	0.03	0.00
	Christ.	0.00	0.00	0.59	0.03	0.00
	AS1	0.00	0.01	0.83	1.00	0.06
	AS2	0.00	0.00	0.36	0.92	0.00
ANN	Kupiec	0.65	0.04	0.65	0.30	0.29
	Christ.	0.02	0.00	0.00	0.00	0.00
	AS1	0.24	0.86	0.59	0.81	0.00
	AS2	0.29	0.11	0.35	0.24	0.00

*Source:* own elaboration

algorithms or statistical models. Notwithstanding the Stacked-ANN performance, it is observed for every model that the higher the volatility the lower the accuracy. In addition to this analysis, the Stacked-ANN has been tested with the most recent data (2017 and 2018) in order to check its performance in the current market conditions. As it occurred with the tests carried out during the financial crisis, the proposed architecture outperforms the benchmark models in terms of accuracy. The superior performance shown by the Stacked-ANN in periods with different levels of volatility are due to the model flexibility. In contrast with ANN-GARCH or ANN-EGARCH, the inputs introduced in the ANN stacked model do not follow any theoretical assumption about the returns distribution or volatility. As explained throughout Section 3.3, the architecture proposed uses previous volatilities and forecasts made by a random forest, gradient boosting with regression trees and support vector machine as inputs. Before beginning with the second point of the conclusion, it is worth mentioning that it has been empirically demonstrated that block bootstrap methods are of special interest when fitting algorithms to volatility as these procedures are especially prepared to work with stationary time series.

Secondly, the forecasts made by the volatility models have been combined with a certain distribution in order to compute the VaR and CVaR for all the different periods

analysed. The distribution selected has been the Student's t-distribution for every model with the exception of the Heston Model which requires changes in asset prices to follow a Brownian diffusion process. The empirical results demonstrated that only the Stacked-ANN model is able to produce an appropriate p-value for Kupiec, AS1 and AS2 tests in every period under analysis, including those ones related with the financial crisis.

The aforementioned flexibility and predictive power of the Stacked-ANN compared with other volatility models suggest to develop further investigations about the implications of using this model for derivative valuation purposes. As the price of these instruments is closely related to the volatility of the underlying assets, further researches should be done in order to compare the implied volatilities observed in the market with the ones arising from the proposed model. If the volatility measured by the Stacked-ANN is more accurate than market expectations, it would be possible to identify under and overvalued derivatives.

## 4 Stochastic reserving with a stacked model based on a hybridized Artificial Neural Network

**Authors:** Eduardo Ramos-Pérez, Pablo J. Alonso-González and José Javier Núñez-Velázquez.

**Journal:** Expert Systems with Applications, vol: 163 1-12. ISSN 0957-4174.

**DOI:** 10.1016/j.eswa.2020.113782

**Journal Impact Factor in 2020 (last available):** 6.954

**Rank by Journal Impact Factor in 2020 (last available):** 8/84 in Operations Research and Management Business (Q1-D1).

**Journal Citation Indicator in 2020 (last available):** 1.68

**Rank by Journal Citation Indicator in 2020 (last available):** 6/99 in Operations Research and Management Business (Q1-D1).

**Published:** January 2021

**Citations:** 1 (Web of Science), 4 (Google Scholar).

**Arxiv Repository:** <https://arxiv.org/abs/2008.07564>

This manuscript version is made available under the CC-BY-NC-ND 4.0 license  
<http://creativecommons.org/licenses/by-nc-nd/4.0/>

## Abstract

Currently, legal requirements demand that insurance companies increase their emphasis on monitoring the risks linked to the underwriting and asset management activities. Regarding underwriting risks, the main uncertainties that insurers must manage are related to the premium sufficiency to cover future claims and the adequacy of the current reserves to pay outstanding claims. Both risks are calibrated using stochastic models due to their nature. This paper introduces a reserving model based on a set of machine learning techniques such as Gradient Boosting, Random Forest and Artificial Neural Networks. These algorithms and other widely used reserving models are stacked to predict the shape of the runoff. To compute the deviation around a former prediction, a log-normal approach is combined with the suggested model. The empirical results demonstrate that the proposed methodology can be used to improve the performance of the traditional reserving techniques based on Bayesian statistics and a Chain Ladder, leading to a more accurate assessment of the reserving risk.

**Keywords:** Stochastic reserving, Reserving Risk, Machine Learning, General insurance, Run-off prediction

**AMS Subject Classification:** 62-07, 62P05, 65C60, 90-08.

## 4.1 Introduction

As with any other company, the survival of an insurance firm depends on its ability to obtain a sustainable profit over the years. These entities have to offer their services at an adequate and competitive premium, while the ultimate cost of the claims is subject to uncertainty. Thus, reserving models were developed in order to estimate and monitor the expected ultimate cost of outstanding claims. Although life insurance contracts manifest uncertainty about the claims cost, reserving takes a special relevance in general insurance as that uncertainty tends to be higher, at least in the short term.

Methods of estimating the level of reserves in non-life insurance have evolved from classical and deterministic methods toward others that take into account the loss reserve uncertainty. The aim of the first type is to estimate the expected level of reserves by taking the historical information into consideration. Chain Ladder is the most frequently used method of this family. When historical data are not stable enough to use the Chain Ladder technique, the Bornhuetter and Ferguson (1972) model tends to be the preferred option to obtain an adequate estimate of the expected ultimate cost.

The increasing interest of investors in the risk profile of financial institutions since the Financial Crisis of 2007-2008 and the implementation of the Solvency II Directive in the European market have fostered the use of stochastic reserving models. As in the case of deterministic approaches, stochastic models based on the Chain Ladder technique are the most commonly used. One of the main techniques within this family is the Overdispersed Poisson (ODP) model developed by Renshaw and Verrall (1998) and its bootstrap implementation suggested by England and Verrall

(1999) and England (2002) which assumes that incremental claims follow an ODP distribution where the variance is proportional to the mean.

In this model, incremental claims must be positive, but this limitation can be overcome by using the quasi-likelihood approach introduced by McCullagh and Nelder (1989). In cases where the ODP assumption does not properly fit the data, Kremer (1982), Mack (1991) and Verrall (2000) developed other models assuming log-normal, gamma and negative binomial distributions respectively. In contrast to the methods within this family, Mack (1993) developed a free-distribution model by focusing and limiting the claims reserve distribution analysis to the first two moments.

Thus, the bootstrap implementation of Mack's model allows the analyst to obtain a reserve distribution without the necessity of defining a theoretical distribution for the cumulative or incremental claim cost. If the bootstrapping procedure is to be avoided, England and Verrall (2006) introduced a stochastic Bayesian implementation of the ODP, Negative Binomial and this last free-distribution model. This approach was recently expanded by Meyers (2015), who developed some Bayesian Markov Chain Monte-Carlo (MCMC) models (Levelled Chain-Ladder, Correlated Chain-Ladder, Levelled Incremental Trend, Correlated Incremental Trend and Changing Settlement Rate) for incurred and paid data. Their aim is to improve the performance of ODP and Mack models by using different approaches such as recognizing the correlation between accident years, including a skewed distribution to model negative incremental payments, introducing a trend over the development years and allowing changes in the claim settlement rate.

Another set of models is focused on using several triangles simultaneously in order to take into consideration different characteristics of incurred and paid data. The main models within this family are the Munich Chain Ladder (MCL) method and Double Chain Ladder (DCL) model developed by Quarg and Mack (2004) and Martínez-Miranda et al. (2012), respectively. By modifying this last method, Margraf et al. (2018) addressed the problem of calculating general insurance reserves when the portfolio is covered by an excess-of-loss reinsurance. In addition to MCL and DCL, Merz and Wüthrich (2010) introduced a Bayesian implementation of the paid-incurred chain (PIC) reserving method (Posthuma et al. 2008) based on using both incurred and paid data. Happ et al. (2012) and Happ and Wüthrich (2013) also investigated and developed models related to the PIC method, while Halliwell (2009) and Venter (2008) introduced regression approaches based on using both data sources. Pigeon et al. (2014), Antonio and Plat (2014), and Martínez-Miranda et al. (2013b) also proposed models by taking into consideration different data sources to estimate the expected ultimate claim cost.

In addition to the different approaches exposed above, it is possible to find models where the information is not organized in an aggregated way, as in the classical triangles, but rather in individual claims data (see Taylor et al. 2008, Jessen et al. 2011, Pigeon et al. 2013, Antonio and Plat 2014, Martínez-Miranda et al. 2015, Charpentier and Pigeon 2016, or Wüthrich 2018b).

Thanks to the increase in computational power, machine learning techniques have turned into an adequate tool for reserving purposes. Artificial Neural Networks (Gabrielli and Wüthrich 2018 and Wüthrich 2018b), regression trees (Wüthrich 2018a), Recurrent Neural Networks (Kuo 2018) or tree-based algorithms (Lopez et al. 2019) have been used to predict claim reserves. Gabrielli et al. (2018) embedded the ODP model into a neural network framework, and Baudry and Robert (2019) introduced a nonparametric reserving model based on extremely randomized trees (Geurts et al. 2006) and individual claims data. In addition to the aforementioned algorithms, other machine learning techniques were used by Martínez-Miranda et al. (2013a) for reserving purposes, and a support vector machine was applied to classify risks prior to the reserve calculation (Duma et al. 2011).

The research carried out in this paper develops a nonparametric reserving model based on the stacking algorithm methodology. The proposed architecture consists of two different levels. Random Forest (RF) (Breiman 2001), Gradient Boosting (GB) with regression trees (Friedman 2000), Artificial Neural Network (ANN) (McCulloch and Pitts 1943), Changing Settlement Rate (CSR) reserving model and the Chain Ladder assumptions are incorporated within the first level, while an ANN is included in the second level of the stacked model (Stacked-ANN) architecture in order to generate the final predictions. Therefore, the aim of this hybrid model is to improve the performance of the individual components by creating an architecture that can learn from the different algorithms and the reserving models included within the first level.

Although the overall methodology is based on that proposed by Ramos-Pérez et al. (2019) for stock volatility forecasting purposes, the model architecture proposed in this study is different. In this research, machine learning algorithms and reserving models are present in the first level, while in the architecture developed by Ramos-Pérez et al. (2019), only machine learning algorithms were included. Therefore, the most popular models for forecasting volatility such as GARCH or EGARCH were not integrated within the model architecture, while in this case, Chain Ladder and CSR are incorporated. It is also worth mentioning that in contrast to the hybrid model proposed for forecasting volatility purposes, in this research, the second level only receives information already processed by the models within the first level. In addition to the main differences explained above, it should be pointed out that the stacking algorithm methodology has not appeared previously in the actuarial literature related to the valuation of loss reserves. Apart from that, a log-normal approach is combined with the suggested reserving model based on machine learning in order to compute the reserve variability.

As all the different algorithms and reserving models of the first level are incorporated in the ANN of the second level, some of the most important research studies carried out in the context of selecting the optimal ANN architecture will be discussed. There is a significant amount of literature supporting the use of ANNs with just one hidden layer because under mild assumptions on the activation functions, the universal

approximation theorem states that a feedforward ANN with a single hidden layer and a finite number of neurons can approximate any continuous function on compact subsets of the Euclidean space.

Based on regularization techniques and using just one hidden layer network, Poggio and Girosi (1990) developed a theoretical framework to approximate nonlinear mappings named regularization networks. These authors demonstrated that their architecture can approximate any continuous function on a compact domain if the number of units is high enough. Cybenko (1989) and Hornik et al. (1989) also proved that one hidden layer networks with sigmoidal activation functions can approximate continuous functions on any compact Euclidean space. It was also shown that, under certain conditions, an arbitrarily small error between a single hidden layer ANN and any other continuous function can be obtained by increasing the number of neurons (Barron 1994, Funahashi 1989 and Hornik 1993). Nakama (2011) showed that the range of effective learning rates is wider in the case of ANN with one hidden layer than in architectures with multiple hidden layers.

On the other hand, Hornik (1991) and Leshno et al. (1993) demonstrated that ANNs have the potential of being universal approximators not only due to the choice of a specific activation function but also because of the possibility of using several hidden layers. Limitations of the approximation capabilities of one hidden layer networks were demonstrated by Chui et al. (1994) and Chui et al. (1996). In recent years, multi-hidden layer architectures have improved the state of the art in machine learning.

For example, in the context of natural language processing, the models and architectures created by Devlin et al. (2018) (BERT), Brown et al. (2020) (GPT3) and Vaswani et al. (2017) (Transformer) overcome the performance of other less complex models. In addition, it is worth mentioning that agents trained with multi-hidden layer ANNs have been able to overcome the human performance in specific tasks such as playing chess (Silver et al. 2017) or ‘go’ (Silver et al. 2016). With respect to the optimal number of neurons, Celikoglu (2007) analysed this issue in the context of solving the dynamic network loading problem, while Sheela and Deepa (2013) proposed a list of principles to select this number.

Results from recently published papers in the actuarial field support the idea of applying ANNs with multiple hidden layers. Indeed, Richman and Wüthrich (2018) and Nigri et al. (2019) applied this structure to model human mortality, while Castellani et al. (2018) used it for estimating the economic capital of insurance companies under the Solvency II framework. Thus, the ANNs included within the architecture of the Stacked-ANN model have several hidden layers.

The rest of the paper proceeds as follows: Section 4.2 presents the set of models used for comparison purposes. Additionally, the error and risk measures taken to validate the stochastic reserves, payments and ultimate losses are discussed. In Section 4.3, the theoretical background and architecture of the reserving model based on stacking

algorithms (Stacked-ANN) are explained. Details about the log-normal approach proposed for obtaining a stochastic distribution are also given in this section. The empirical results, error and risk measures of the different reserving models are shown in Section 4.4. Finally, Section 4.5 presents the main conclusions derived from the results and comparisons presented in Section 4.4.

## 4.2 Benchmark models and validation

As previously stated, this section explains the benchmark models and the different measures used to assess their performance. Thus, the first paragraphs are dedicated to ODP, Mack's model, CSR and a nonparametric approach based on ANNs, while the end of this section presents the indicators used to compare and validate the reserve distribution functions estimated by the benchmark models with those simulated by the model presented in Section 4.3.

The first benchmark model is ODP (Renshaw and Verrall 1998 and England and Verrall 1999). Denoting the origin year as  $i$  and the development year as  $j$ , this reserving model based on the Chain Ladder technique assumes that incremental payments,  $C_{ij}$ , follow an overdispersed Poisson distribution with a variance proportional to the mean:

$$E[C_{ij}] = \mu_{ij} \qquad \qquad \qquad Var[C_{ij}] = \phi \mu_{ij} \qquad (59)$$

where  $\phi$  is the parameter that determines the level of overdispersion. Even though this model assumes  $C_{ij}$  to be a positive integer, the quasi-likelihood (McCullagh and Nelder 1989) approach allows fits the model to non-integer data, which can be either positive or negative. The bootstrapping procedure used in this study to compute a reserve distribution function with the ODP model was introduced by England and Verrall (1999) and England (2002).

Mack (1993) model, which is also based on the Chain Ladder technique, is the second benchmark. The main characteristic of this reserving model is the lack of assumptions about the underlying distribution of the payments. This is achieved by using only the first two moments:

$$E[D_{ij}] = \lambda_j D_{i,j-1} \qquad \qquad \qquad Var[D_{ij}] = \sigma_j^2 D_{i,j-1} \qquad (60)$$

where  $\lambda_j$  and  $\sigma_j^2$  refer to the parameters to be estimated, and  $D_{ij}$  is the cumulative payment. As with the ODP model, a bootstrapping procedure is used to calculate the reserve distribution function with Mack's model.

The third benchmark model is CSR, a Bayesian approach introduced by Meyers (2015). The default calibration and prior distributions suggested by this author will be used in this study:

- $\alpha_i \sim N(\ln P_i + \text{logelr}, \sqrt{10})$ , where  $\text{logelr} \sim U(-1, 0.5)$  and  $P_i$  are the premiums by accident year.

- $\beta_j \sim U(-5, 5)$  for  $j = 1, \dots, J - 1$ . In the last development year,  $\beta_J = 0$ .
- $\mu_{i,j} = \alpha_i + \beta_j(1 - \gamma)^{i-1}$ , where  $\gamma \sim N(0, 0.025)$ .
- Each  $\sigma_j = \sum_{i=j}^J a_i$ , where  $a_i \sim U(0, 1)$ .

Taking into consideration the aforementioned distributions and parameters, the cumulative payments simulated by the CSR model follow a log-normal distribution,  $D_{i,j} \sim LN(\mu_{i,j}, \sigma_j)$ , subject to the constraint  $\sigma_1 > \sigma_2 > \dots > \sigma_J$ .

To analyse the improvement in the performance due to the stacking procedure that is presented in Section 4.3, the last benchmark model to be introduced is an individual ANN. The inputs and characteristics (hidden layers, activation functions, etc.) of this algorithm will be the same as those used for the ANN included within the first level of the Stacked-ANN. Additionally, the log-normal procedure to obtain the reserve variability is the same as that for the Stacked-ANN model. To avoid repeating content, refer to Section 4.3 for further details about the characteristics of the ANN used as a benchmark.

Once the four benchmark models are explained, the different measures selected to compare the performance of the Stacked-ANN with the aforementioned reserving models are presented. Insurance regulations such as the Solvency II Directive and Swiss Solvency Test ask the general insurance companies to evaluate their expected reserves and potential deviations from these central scenarios. Thus, the error of the estimated reserves will be computed in order to compare the performance of the different models. As several triangles with different levels of payments are used during this study, the measure for evaluating the reserves is

$$\%RMSE(R^t) = \frac{\sqrt{\sum_{k=1}^K (\hat{R}_{k,\mu}^t - R_k^t)^2 / K}}{\sum_{k=1}^K R_k^t} * 100 = \frac{RMSE(R^t)}{\sum_{k=1}^K R_k^t} * 100 \quad (61)$$

where  $K$  is the total number of triangles,  $t$  is the calendar year when the reserves are evaluated,  $\hat{R}_{k,\mu}^t$  is the reserve predicted by the reserving model using the triangle  $k$  and  $R_k^t$  the reserves that were actually observed for that triangle. As it can be derived from the former expression, the aim of this error measure is the evaluation of the weight of the root mean squared error over the total reserves. To understand the model's performance, this error measure will also be calculated for the next year's payments ( $\%RMSE(P^{t+1})$ ) and the ultimate loss cost ( $\%RMSE(U^t)$ ).

In addition to the aforementioned error measures, the reserving risk ( $RR$ ) per unit of reserve derived from the use of the different stochastic reserving models will be analysed. As previously stated, the models are going to be fitted to several triangles, so the average of the former ratio is taken as a risk measure:

$$Ratio(RR_{1-\alpha}^t) = \frac{\sum_{k=1}^K (\hat{R}_{k,1-\alpha}^t - \hat{R}_{k,\mu}^t) / \hat{R}_{k,\mu}^t}{K} = \frac{\sum_{k=1}^K RR_{1-\alpha}^t / \hat{R}_{k,\mu}^t}{K} \quad (62)$$

where  $\hat{R}_{k,\mu}^t$  is the mean and  $\hat{R}_{k,1-\alpha}^t$  is the percentile  $1 - \alpha$  of the estimated reserve distribution function of the company  $k$ . A deeper evaluation of the variation estimated by the different stochastic models is carried out by calculating the standard deviation per unit of reserve:

$$Ratio(\sigma) = \frac{\sum_{k=1}^K \sigma(\hat{R}_k^t) / \hat{R}_{k,\mu}^t}{K} \quad (63)$$

Finally, in order to check the adequacy of the reserving risk calculated for the different companies, the Kupiec (1995) test is applied in order to verify if the number of excesses is aligned with the selected confidence level. The empirical results of the test and measures are collected in Section 4.4.

### 4.3 Stochastic reserving model based on the stacking algorithm approach

This section is divided into several subsections in order to sequentially explain the proposed reserving model. In addition, Figure 24 presents the model architecture in order to support the explanation.

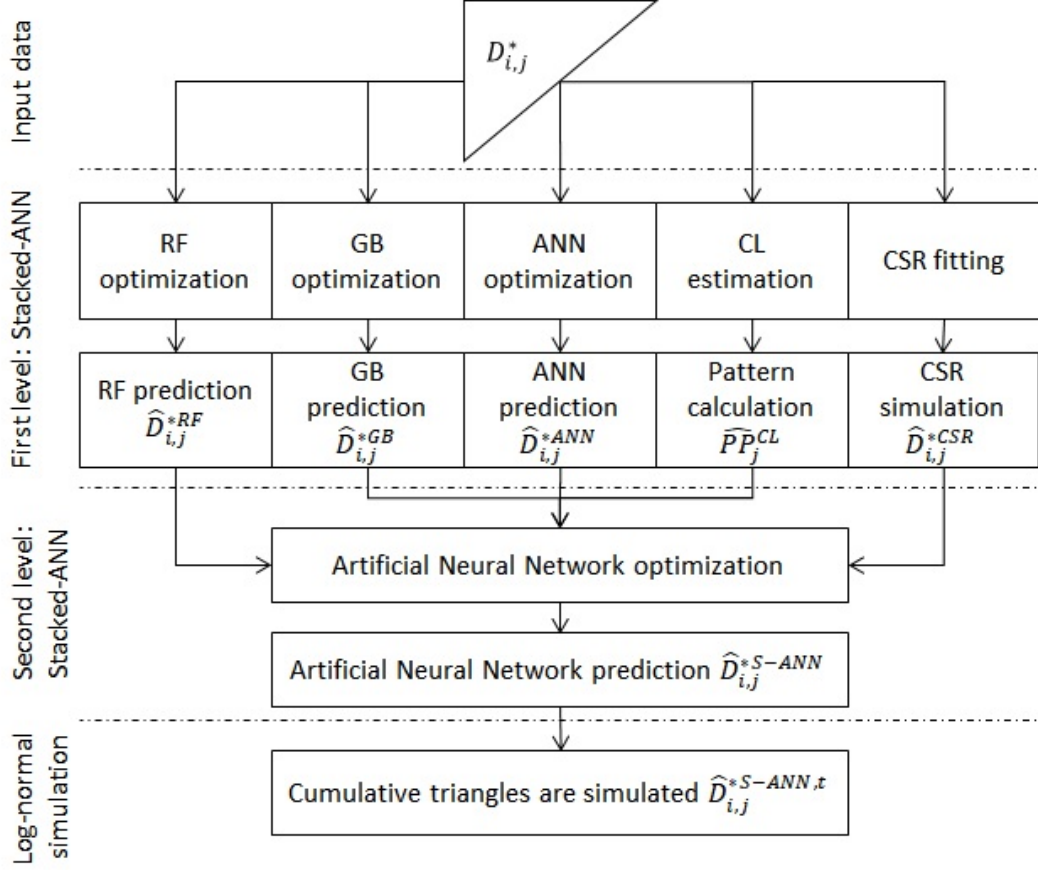
#### Model inputs

Before estimating the different reserving models within the first level of the Stacked-ANN model, the database used, as well as the response and explicative variables for fitting the algorithms within this level, need to be defined.

The lower and upper triangles needed to fit and validate the models are obtained from Schedule P of the NAIC Annual Statement. This database (available on the CAS website) was collected from property and casualty insurers that underwrite business in the US, and it contains both paid and incurred losses (net of reinsurance) of the accident years from 1988 to 1997. Ten development years are available for every accident year. In addition to loss data, gross and net premiums by accident year are also reported in the database.

In this paper, the different reserving models will be fitted to 200 loss triangles from NAIC Schedule P, 50 from each of the following lines of business: Commercial Auto (CA), Private Passenger Auto Liability (PA), Workers' Compensation (WC) and Other Liability (OL). As pointed out by Meyers (2015), selecting triangles from insurers who made significant changes in business operations is one of the main mistakes that could be made with NAIC Schedule P data. The coefficient of variation of the net premiums and the net/gross premium ratio should be appropriate indicators of changes in business operations, so this author selected insurers that minimize the aforementioned metrics. The triangles selected by Meyers (2015) are used in this research in order to avoid the former issue and ensure comparability with other studies.

Figure 13: Stacked-ANN model structure



With regard to the explanatory variables, as with other nonparametric reserving models based on Generalized Additive Models (Hastie and Tibshirani 1986 and England and Verrall 2002) or RNN (Kuo 2018), accident  $i$  and development  $j$  years were selected to be the inputs of the first-level algorithms. Both were initialized as one and then scaled to range  $[0, 1]$  (hereinafter  $AY_i^*$  and  $DY_j^*$ ) in order to facilitate the fitting of the algorithms (Hastie et al. 2009).

The response variable of these algorithms is the scaled cumulative payments  $D_{ij}^*$ . Depending on the data availability and the characteristics of the portfolio to be modelled, different exposure measures can be selected to scale  $D_{ij}$ . In this paper, net premiums  $P_i$  play the role of exposure measure, as this is the most relevant option between the variables available in the database.

Loss triangles are a representation of payments over time by accident or underwriting year. Thus, the training and optimization of the deep learning algorithms within the Stacked-ANN model architecture need to take into consideration that loss triangles are composed of temporal series. Accordingly, the last diagonal is selected as a test

Figure 14: Train and test sets



set because it contains the most updated information, while the rest of the triangle is used for fitting the algorithms (Figure 26).

During the optimization process, different configurations of the algorithms are fitted with the training data. To obtain the best configuration, the test set is predicted, and the root mean squared error of every option is computed. Finally, the configuration that minimizes the former test error is selected.

### First level: Individual models

The first level of the Stacked-ANN model consists of a Chain Ladder, CSR, and three algorithms whose inputs were described in Section 4.3. It is worth mentioning that as ODP and Mack's model are based on the Chain Ladder technique, the Stacked ANN model incorporates the core rationale behind these stochastic reserving models. The machine learning algorithms (RF, GB and ANN) fitted at this step are explained in the following paragraphs and will be optimized by applying a grid search to some hyperparameters and by measuring the test error. Additionally, at the end of this subsection, the Chain Ladder and CSR hypothesis are integrated within the Stacked-ANN model architecture.

The Random Forest (RF) algorithm introduced by Breiman (2001) averages  $B$  different regression trees. In every fitted tree, the explanatory variables and data points used during the training are randomly selected. Therefore, the formal expression to predict the scaled cumulative payments is:

$$\hat{D}_{ij}^{*RF} = \frac{\hat{D}_{ij}^{RF}}{P_i} = \frac{\sum_{b=1}^B T_b(X)}{B} \quad (64)$$

$T_b$  represents the  $b$ -th regression tree fitted and  $X$  the selected subset of  $AY_i^*$  and  $DY_j^*$  to fit  $T_b$ . During the estimation process, the hyper-parameters optimized are the number of variables randomly selected,  $N$ , and the minimum number of observations to be kept in the terminal nodes of every fitted tree,  $Obs_{RF}$ .

The second algorithm within the first level is Gradient Boosting (GB) with regression trees (Friedman 2000). In this case, the gradient is minimized by sequentially fitting  $B$  regression trees. The subset of data to be used during the estimation process of

every tree is also randomly selected. The expression to obtain the predicted scaled cumulative payments is

$$\hat{D}_{ij}^{*GB} = \frac{\hat{D}_{ij}^{GB}}{P_i} = \hat{f}_{B-1}(X) + \delta_{GB} T_B(X) \quad (65)$$

$\hat{f}_{B-1}(X)$  represents the function obtained after adding sequentially  $B - 1$  regression tree models and,  $\delta_{GB}$  is the learning rate. The hyperparameter selected to be optimized during the training process is the minimum number of observations to be kept in the terminal nodes of every fitted tree,  $Obs_{GB}$ . Regarding the hyperparameters, it is worth mentioning that the learning rate,  $\delta_{GB}$ , is set to 0.01.

The last algorithm of the first layer is an Artificial Neural Network (ANN) (Mcculloch and Pitts 1943). Following the notation provided by Bishop (2006) and taking into consideration that the feed-forward ANN used in this paper is composed of 2 hidden layers with 5 neurons each, the formal expression to obtain the predictions can be defined as follows:

$$\begin{aligned} \hat{D}_{ij}^{*ANN} &= \hat{D}_{ij}^{ANN} / P_i = \\ &= h^{(3)} \left( \sum_{k=1}^5 w_{1,k}^{(3)} h^{(2)} \left( \sum_{j=1}^5 w_{k,j}^{(2)} h^{(1)} \left( \sum_{i=1}^2 w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right) + w_{1,0}^{(3)} \right) \end{aligned} \quad (66)$$

where  $h^{(n)}$  is the activation function associated with layer  $n$ ,  $w_{z,v}^{(n)}$  is the  $v$ -th weight associated with the neuron  $z$  inside layer  $n$ , and  $x_i$  refers to the  $i$ -th input variable of the database composed of two explanatory variables, the scaled accident ( $AY_i^*$ ) and development year ( $DY_j^*$ ). The percentage of dropout regularization  $\theta$  is the hyperparameter to be optimized by applying a grid search and measuring the test error. As with the other algorithms, upper triangle predictions will be used as input within the second level of the architecture.

In addition to the three aforementioned algorithms, Chain Ladder assumptions are incorporated in the model architecture. To do so, the development factors of the Chain Ladder technique are used as an input in the second level of the Stacked-ANN model:

$$\hat{\lambda}_j^{*CL} = \frac{\sum_{i=1}^{n-j-1} D_{ij}^*}{\sum_{i=1}^{n-j-1} D_{i,j-1}^*} \quad (67)$$

where  $\{\hat{\lambda}_j^{*CL} : j = (2, 3, \dots, J)\}$ . Although the Chain Ladder methodology does not produce any parameters for  $j = 1$ , the second-level algorithm needs a value for  $j = 1$ . Thus, within the Stacked-ANN methodology, it is assumed that  $\hat{\lambda}_1^{*CL} = 1$ .

Finally, CSR methodology (Meyers 2015) is integrated. To achieve this, 10,000 MCMC simulations are produced within the first level of the Stacked-ANN model. Then, the expected scaled cumulative payments of the upper triangle arising from

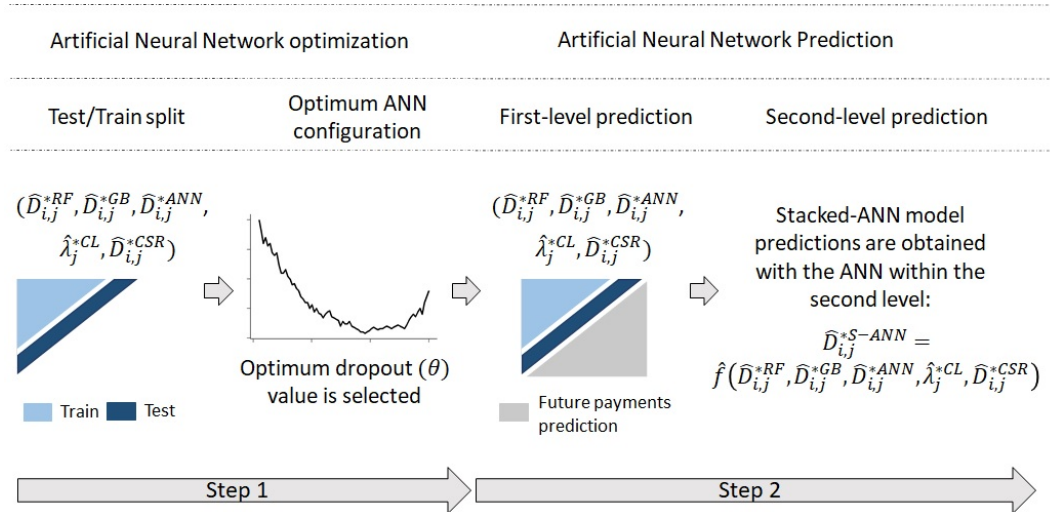
the aforementioned simulations are used as input in the algorithm within the second level of the Stacked-ANN model:

$$\hat{D}_{ij}^{*CSR} = \frac{\sum_{k=1}^{10,000} \hat{D}_{ijk}^{CSR} / P_i}{10,000} \quad (68)$$

## Second level: Stacking algorithm

As previously stated, the inputs of this level are the scaled cumulative payments predicted by the algorithms named in Section 4.3 (RF, GB and ANN), the development factors based on the Chain Ladder technique and the expected scaled cumulative payments simulated by the CSR model. On the other hand, the output of the ANN within the second level and the Stacked-ANN are the cumulative payments  $\hat{D}_{ij}^{*S-ANN}$  by accident and development year.

Figure 15: Second-level structure



Similar to the first-level algorithms, the training and optimization processes of the ANN within this level need to recognize that loss triangles are composed of a set of time series. The most recent information of the loss triangles is the last diagonal; thus, the explicative and response variables of this diagonal are selected as a test set, while the rest of the upper triangle data is used as a training set.

Once the test and training sets are defined, the optimum configuration of the ANN needs to be obtained. To do so, the training data are used to fit ANNs with different levels of dropout regularization  $\theta$ . Then, the root mean squared error is computed by taking into consideration the predictions made by every ANN configuration. The  $\theta$  that minimizes the test error is selected.

Due to the Stacked-ANN architecture, two substeps need to be carried out in order to make the final predictions. First, the lower triangle of the first-level models

need to be predicted. Second, the data predicted in the previous step are used as input of the ANN within the second layer to make the final predictions. Thus, the Stacked-ANN model tries to obtain more accurate predictions by combining different reserving models and algorithms.

Figure 24 shows the overall Stacked-ANN architecture, and Figure 27 provides a detailed summary of the process defined in the previous paragraphs. Technical details about the feedforward ANN fitted within this level of the Stacked-ANN model are presented below:

- It contains two hidden layers with 5 neurons each. The sigmoid activation function was selected for all neurons within the hidden layers while the linear activation function was used in the output layer, which is composed of one neuron.
- The selected optimization algorithm is Adaptive Moment Estimation (ADAM), which was created by Kingma and Ba (2014). This method consists of a progressive adaptation of the initial learning rate, taking into consideration current and previous gradients. The default calibration proposed by the authors for the ADAM parameters is applied as  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . Thus, the ANN parameters are updated as follows:

$$\omega_t = \omega_{t-1} - \delta_{ANN} \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (69)$$

$$\hat{m}_t = \frac{\beta_1 m_{t-1} + (1 - \beta_1) g_t}{1 - \beta_1^t} \quad (70)$$

$$\hat{v}_t = \frac{\beta_2 v_{t-1} + (1 - \beta_2) g_t^2}{1 - \beta_2^t} \quad (71)$$

where  $\omega$  is the parameter to be updated and  $g_t$  the gradient in the epoch  $t$ . The initial learning rate is set to  $\delta_{ANN} = 0.01$ .

- The number of epochs is 10,000, and the batch size is equal to the length of the data used for training the ANN.
- The backward pass calculations are done according to the selection of the root mean squared error as a loss function.
- As previously stated, the percentage of dropout regularization  $\theta$  is the hyperparameter to be optimized by applying a grid search and measuring the test error.

Taking the abovementioned details into consideration, the scaled cumulative payments predicted by the Stacked-ANN model are obtained by means of the following

expression:

$$\begin{aligned}\hat{D}_{ij}^{*S-ANN} &= \frac{\hat{D}_{ij}^{*S-ANN}}{P_i} = \hat{f}(\hat{D}_{ij}^{*RF}, \hat{D}_{ij}^{*GB}, \hat{D}_{ij}^{*ANN}, \hat{\lambda}_j^{*CL}, \hat{D}_{ij}^{*CSR}) = \\ &= h^{(3)} \left( \sum_{k=1}^5 w_{1,k}^{(3)} h^{(2)} \left( \sum_{j=1}^5 w_{k,j}^{(2)} h^{(1)} \left( \sum_{i=1}^5 w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right) + w_{1,0}^{(3)} \right)\end{aligned}\quad (72)$$

## Log-normal simulation

To compute the Kupiec test and the measures related to reserve variability (Section 4.2), the deviation around the central scenario predicted by the Stacked-ANN model needs to be obtained. Due to its right skewness and long tail, log-normal distribution is widely used within reserving models to derive the variability of the claims cost. Many papers used the lognormal distribution to compute this variability (see, among others, Kremer (1982), Antonio et al. (2006), Rehman and Klugman (2009), Weke and Ratemo (2013), Meyers (2015) or more recently, Omari et al. (2018)).

In this study, a log-normal distribution is used to compute the reserve variability around the central scenario predicted by the Stacked-ANN. To do so, the parameters of this distribution are obtained using the aforementioned predictions and the moments method. Therefore, regardless of the distribution selected, the central scenario is that predicted by the Stacked-ANN, and thus, changing the distribution has no effect on the error measures described in Section 4.2. Nevertheless, changes to the log-normal hypothesis will modify the variability and, consequently, the risk measures ( $Ratio(RR_{1-\alpha}^t)$  and  $Ratio(\sigma)$ ) and the results of the Kupiec test. Below, the steps of the procedure are described:

1. Starting with the scaled cumulative payments predicted by the Stacked-ANN ( $\hat{D}_{ij}^{*S-ANN}$ ), the variance by development year is computed as follows:

$$Var[\hat{D}_j^{*S-ANN}] = \frac{\sum_{i=1}^n \left( \hat{D}_{ij}^{*S-ANN} - E[\hat{D}_j^{*S-ANN}] \right)^2}{n-1} \quad (73)$$

where  $n$  refers to the total number of accident years and  $E[\hat{D}_j^{*S-ANN}]$  is the mean of the scaled cumulative payments by development year.

2. By using the method of the moments and values calculated in the previous step, the parameters of the log-normal distribution are obtained:

$$\hat{\mu}_{ij}[\hat{D}_j^{*S-ANN}] = \ln \left( \frac{E[\hat{D}_j^{*S-ANN}]^2}{\sqrt{Var[\hat{D}_j^{*S-ANN}] + E[\hat{D}_j^{*S-ANN}]^2}} \right) \quad (74)$$

$$\hat{\sigma}_j^2[\hat{D}_j^{*S-ANN}] = \ln \left( 1 + \frac{Var[\hat{D}_j^{*S-ANN}]}{E[\hat{D}_j^{*S-ANN}]^2} \right) \quad (75)$$

3. For  $t = (1, 2, \dots, T)$ :

- (a) A triangle is generated by sampling random values from the following distribution function:  $\hat{C}_{ij}^{*S-ANN,k} \sim LN(\hat{\mu}_{ij}[\hat{D}^{*S-ANN}], \hat{\sigma}_j^2[\hat{D}^{*S-ANN}])$ .
- (b) The final simulated values,  $\hat{C}_{ij}^{S-ANN,k}$ , are obtained by removing the scaling. Hence, the scaled payments obtained in the previous step are multiplied by  $P_i$ .

## 4.4 Results

In this section, the data used, the fitting process and a final comparison between the Stacked-ANN and the benchmark models are shown.

### Data and fitting of the Stacked-ANN

As stated in Section 4.3, the upper and lower triangles required to fit and validate the models are obtained from Schedule P of the NAIC Annual Statement. This database contains the losses, reserves and premiums from 1988 until 1997 of different property and casualty insurers that underwrite business in the United States.

Meyers (2015) indicated that one of the main mistakes with the NAIC Schedule P data is selecting triangles from insurers that made significant changes in their businesses. Meyers used the coefficient of variation of the net premiums and the net-on-gross ratio to select 50 triangles of each of the following lines of business: Commercial Auto (CA), Private Passenger Auto Liability (PA), Workers' Compensation (WC) and Other Liability (OL). This triangle selection was also used in this paper in order to ensure comparability with other studies that take as a reference the selection made by Meyers (2015). For further details about the data used to fit the Stacked-ANN, refer to Section 4.3.

Once the data have been presented, the subsection focuses on the fitting of the Stacked-ANN. The first level of the proposed model is composed of three individual algorithms (RF, GB and ANN), the CSR reserving model and the development factors derived from the use of the Chain Ladder technique. The second level is composed of an ANN. As pointed out in Sections 4.3 and 4.3, the optimum hyperparameters of the algorithms within the first and second levels are obtained for each triangle using a grid search. Table 9 lists the minor differences across the lines of business in the means of the 50 optimum hyperparameters obtained for each algorithm.

Table 9: Mean of the optimum hyperparameters by line of business

Line of Business	RF first level	GB first level	ANN first level	ANN second level
CA	$Obs_{RF} = 2.04; N = 1.94$	$Obs_{GB} = 4.38$	$\theta = 0.10$	$\theta = 0.09$
PA	$Obs_{RF} = 2.66; N = 1.86$	$Obs_{GB} = 4.22$	$\theta = 0.07$	$\theta = 0.06$
WC	$Obs_{RF} = 1.78; N = 1.68$	$Obs_{GB} = 3.66$	$\theta = 0.13$	$\theta = 0.12$
OL	$Obs_{RF} = 1.50; N = 1.82$	$Obs_{GB} = 4.24$	$\theta = 0.12$	$\theta = 0.12$

*Source:* own elaboration

As previously stated, the development factors ( $\hat{\lambda}_j^{*CL}$ ) obtained by applying the Chain Ladder technique to  $D_{ij}^*$  are used as input for the ANN included within the second level of the Stacked-ANN model. These values are calculated for each triangle. Table 10 presents the means of the development factors by line of business.

With regard to the three algorithms of the first layer and the Chain Ladder technique, the CSR model is also incorporated in the Stacked-ANN architecture by means of inputting  $\hat{D}_{ij}^{*CSR}$  in the second-level algorithm. This Bayesian reserving model is fitted to every single triangle. Tables 11 and 12 list the means of the CSR parameters by line of business.

Table 10: Mean of the development factors by line of business

Development factors	CA	PA	WC	OL
$\hat{\lambda}_1^{*CL}$	1.89	1.77	2.21	6.66
$\hat{\lambda}_2^{*CL}$	1.35	1.22	1.29	1.90
$\hat{\lambda}_3^{*CL}$	1.16	1.10	1.13	1.33
$\hat{\lambda}_4^{*CL}$	1.08	1.06	1.07	1.18
$\hat{\lambda}_5^{*CL}$	1.04	1.03	1.04	1.10
$\hat{\lambda}_6^{*CL}$	1.02	1.01	1.02	1.04
$\hat{\lambda}_7^{*CL}$	1.00	1.01	1.02	1.02
$\hat{\lambda}_8^{*CL}$	1.01	1.00	1.01	1.02
$\hat{\lambda}_9^{*CL}$	1.00	1.00	1.01	1.01
$\hat{\lambda}_{10}^{*CL}$	1.00	1.00	1.00	1.00

*Source:* own elaboration

Table 11, which is focused on the parameters needed to calculate the mean of the cumulative payments, presents positive  $\gamma$  and negative  $\beta_j$  for every line of business with the unique exception of CA, where  $\beta_6, \beta_7, \beta_8$  and  $\beta_9$  are positive. According to the model definition, the claims settlement speed increases when  $\beta_j < 0$  and  $\gamma > 0$ . This common trend across the different lines of business about the claim settlement rate of the NAIC Schedule P data was already observed by Meyers (2015).

Table 11: CSR parameters by line of business:  $D_{i,j}$  mean

CSR parameter	CA	PA	WC	OL	CSR parameter	CA	PA	WC	OL
$\alpha_1$	7.094	8.959	8.423	6.162	$\beta_1$	-1.235	-0.987	-1.447	-2.446
$\alpha_2$	7.166	9.047	8.612	6.269	$\beta_2$	-0.514	-0.400	-0.626	-1.332
$\alpha_3$	7.171	9.148	8.779	6.330	$\beta_3$	-0.229	-0.198	-0.322	-0.709
$\alpha_4$	7.280	9.143	8.666	6.309	$\beta_4$	-0.085	-0.097	-0.178	-0.363
$\alpha_5$	7.348	9.201	8.644	6.334	$\beta_5$	-0.003	-0.042	-0.089	-0.173
$\alpha_6$	7.347	9.282	8.537	6.515	$\beta_6$	0.039	-0.017	-0.057	-0.079
$\alpha_7$	7.554	9.374	8.595	6.480	$\beta_7$	0.060	-0.008	-0.041	-0.045
$\alpha_8$	7.540	9.389	8.514	6.327	$\beta_8$	0.028	-0.001	-0.029	-0.030
$\alpha_9$	7.494	9.464	8.543	6.543	$\beta_9$	0.012	-0.001	-0.013	-0.014
$\alpha_{10}$	7.556	9.492	8.500	6.327	$\gamma$	0.021	0.008	0.016	0.028

*Source:* own elaboration

The comparison of the CSR deviation by development year presented in Table 12 reveals that OL is the most volatile portfolio, while PA has the most stable reserves. For CA and WC, the reserve variability estimated by this Bayesian reserving model is quite similar, and it is located at an intermediate point between the OL and PA lines of business.

Table 12: CSR parameters by line of business:  $D_{i,j}$  STD

CSR Parameter	CA	PA	WC	OL
$\sigma_1$	0.303	0.028	0.236	0.771
$\sigma_2$	0.176	0.011	0.164	0.488
$\sigma_3$	0.109	0.007	0.117	0.327
$\sigma_4$	0.079	0.004	0.090	0.229
$\sigma_5$	0.063	0.003	0.069	0.164
$\sigma_6$	0.052	0.002	0.052	0.120
$\sigma_7$	0.043	0.002	0.037	0.087
$\sigma_8$	0.035	0.001	0.025	0.061
$\sigma_9$	0.026	0.001	0.016	0.038
$\sigma_{10}$	0.014	0.001	0.008	0.019

*Source:* own elaboration

## Comparison against benchmark models

Once the Stacked-ANN reserving model is fitted, its performance is compared with the benchmark models explained in Section 4.2 (ODP, Mack, CSR and an individual ANN).

Table 13 lists the %RMSEs associated with reserves  $R^t$ , next year payments  $P^{t+1}$  and ultimate losses  $U^t$  by line of business and reserving model. For further details

about the measures presented in the table, refer to Section 4.2.

Table 13: %RMSE by line of business and reserving model

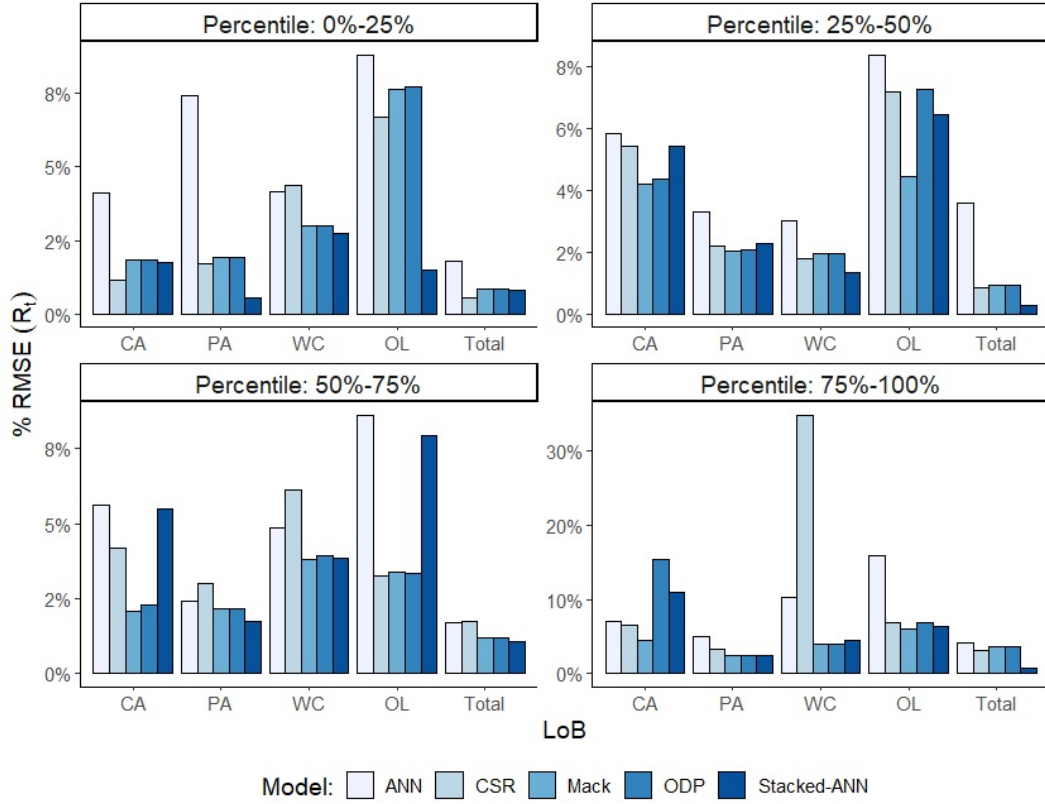
Error Measure	Line of Business	ODP	Mack's Model	CSR	ANN	Stacked ANN
$\%RMSE(R^t)$	CA	0.896%	0.896%	0.534%	1.768%	0.739%
$\%RMSE(P^{t+1})$	CA	0.668%	0.669%	0.573%	1.775%	0.876%
$\%RMSE(U^t)$	CA	0.170%	0.171%	0.102%	0.337%	0.141%
$\%RMSE(R^t)$	PA	1.012%	1.004%	0.823%	5.006%	0.254%
$\%RMSE(P^{t+1})$	PA	1.290%	1.286%	0.258%	1.900%	0.320%
$\%RMSE(U^t)$	PA	0.131%	0.131%	0.107%	0.651%	0.033%
$\%RMSE(R^t)$	WC	1.295%	1.286%	1.751%	1.943%	1.058%
$\%RMSE(P^{t+1})$	WC	0.887%	0.880%	1.531%	1.525%	0.676%
$\%RMSE(U^t)$	WC	0.222%	0.221%	0.301%	0.333%	0.182%
$\%RMSE(R^t)$	OL	5.274%	5.086%	3.153%	5.725%	0.722%
$\%RMSE(P^{t+1})$	OL	2.216%	2.102%	5.528%	0.268%	1.095%
$\%RMSE(U^t)$	OL	1.760%	1.709%	1.056%	1.918%	0.242%

Source: own elaboration

The results obtained by using the different reserving models are summarized as follows:

- The Stacked-ANN model outperforms the individual ANN. The proposed architecture is empirically more accurate because it can learn from the reserving models (Chain Ladder and CSR) and machine learning algorithms (RF, GB and ANN) included within the first level of the Stacked-ANN, while the individual ANN must base its training only on the origin data ( $AY_i^*$  and  $DY_j^*$ ) without taking advantage of other models that are able to capture different patterns and characteristics.
- As they are based on Chain Ladder assumptions, the mean of the distributions generated by ODP and Mack's model should converge to the values obtained by applying the deterministic approach of the Chain Ladder technique. Consequently, the error measures observed in Table 13 for these two stochastic reserving models are almost the same. The table also reveals that ODP and Mack are less accurate than the Stacked-ANN model in most cases.  $\%RMSE(P^{t+1})$  of CA is a unique category in which the benchmark models based on the Chain Ladder technique are more accurate than the proposed methodology.
- Regarding the comparison between Stacked-ANN and CSR,  $R^t$  and  $U^t$  of PA, WC and OL estimated by the proposed model are significantly more accurate than those obtained when using the Bayesian model. Additionally,  $\%RMSE(P^{t+1})$  of the Stacked-ANN model is lower in WC and OL. Thus, in the majority of cases, the CSR model is outperformed by the proposed methodology.

Figure 16:  $\%RMSE(R^t)$  by line of business and volume of reserves.



To enhance the analysis of the results presented in Table 13, Figure 25 shows the  $\%RMSE(R^t)$  by line of business and volume of reserves. First, the companies were classified in four different groups taking into consideration the volume of reserves and the quartiles associated with the distribution. Then, the error rate of each reserving model was computed by line of business. The former calculation was carried out without making any distinction between lines of business.

The results of the aforementioned figure reveal that when no distinction between lines of business is made, the Stacked-ANN architecture outperforms the rest of the reserving models regardless of the company size. This difference is especially relevant for those companies with a higher level of reserves, whose results are collected in the graph labelled 'Percentile: 75%-100%'. As expected, some fluctuations in the performance of the models are observed when the results are analysed by line of business and volume of reserves. Nonetheless, the error rate of the Stacked-ANN tends to be lower than the rest of the benchmark models.

In accordance with the reasons explained within the former paragraphs, it can be concluded that the Stacked-ANN model takes advantage of the different characteristics of several reserving models and machine learning algorithms, leading to a more

flexible and precise architecture in most of the cases.

In addition to the error analysis, the risk measures ( $Ratio(RR_{1-\alpha}^t)$  and  $Ratio(\sigma)$ ) and the p-values of the Kupiec test obtained by using each reserving model are compared. Before examining the results, it is important to point out that Mack's model does not make any assumptions about the payment distribution, ODP assumes that incremental payments follow an overdispersed Poisson distribution, and CSR, ANN and Stacked-ANN presume that cumulative payments are log-normally distributed. The hypothesis taken regarding the payments impact the distribution shape and consequently the risk measures. Therefore, in this case, ODP and Mack's model are not going to converge like they did in the central scenario.

Table 14: Risk measures by line of business and reserving model

Risk measures	Line of Business	ODP	Mack's Model	CSR	ANN	Stacked ANN
$Ratio(RR_{0.995}^t)$	CA	1.936	1.460	2.776	1.387	1.884
$Ratio(\sigma)$	CA	2.561	0.461	0.681	0.456	0.642
Kupiec p-value	CA	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$< 0.05$	$\geq 0.05$
$Ratio(RR_{0.995}^t)$	PA	0.544	0.373	0.918	0.783	0.888
$Ratio(\sigma)$	PA	0.277	0.135	0.270	0.279	0.332
Kupiec p-value	PA	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$
$Ratio(RR_{0.995}^t)$	WC	2.525	0.691	1.797	2.194	2.149
$Ratio(\sigma)$	WC	1.273	0.245	0.474	0.682	0.717
Kupiec p-value	WC	$< 0.05$	$< 0.05$	$< 0.05$	$< 0.05$	$\geq 0.05$
$Ratio(RR_{0.995}^t)$	OL	7.506	3.287	4.843	2.315	3.522
$Ratio(\sigma)$	OL	6.275	1.217	1.119	0.690	1.099
Kupiec p-value	OL	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$

Source: own elaboration

The  $Ratio(RR_{1-\alpha}^t)$  and the p-values collected in Table 29 evaluate the 99.5 percentile ( $\alpha = 0.005$ ) of the reserve distribution function, which is the confidence level set up by Solvency II to calculate the risk of the insurance companies. The results of this table are summarized below:

- According to the results of the Kupiec test, the Stacked-ANN generates an adequate risk assessment for every line of business. It is worth mentioning that when compared with the individual ANN, the empirical results show that the stacking process not only improves the error rate but also allows for the generation of more appropriate distribution functions using the same simulation approach (presented in Section 4.3). With regard to the comparison between the Stacked-ANN and the rest of benchmark models, the Kupiec test reveals that CSR, ODP and Mack's model do not produce appropriate risk measures for WC, while the proposed methodology passes the test.
- Intuitively, the duration of the liabilities should have a close relation with

$Ratio(RR_{0.995}^t)$  and  $Ratio(\sigma)$ : the longer the duration, the higher is the uncertainty around each economic unit of reserve. The development factors based on the Chain Ladder technique measure the claim settlement speed. Therefore, they can be considered a good indicator of the duration of liabilities. A development factor at year  $t$ ,  $\lambda_t$ , means that the  $t + 1$  cumulative payment is  $\lambda_t$  times the cumulative claims settled at  $t$ . Consequently, high development factors indicate a long duration, while low values reflect a high settlement speed. According to Table 10, OL is the line of business with the highest duration, while PA has the lowest. CA and WC, whose durations are in a similar range, are located at an intermediate point between PA and OL. As can be observed in Table 29, this intuition about the relation between the duration and reserve uncertainty is followed by the Stacked-ANN and benchmark models.

- In general, the  $Ratio(RR_{0.995}^t)$  and  $Ratio(\sigma)$  by line of business are similar across the different reserving models. The two main exceptions are the risk measures of ODP for OL and Mack for WC. The high values observed in the ODP estimations for OL are due to two companies whose  $RR_{1-\alpha}^t/\hat{R}_{k,\mu}^t$  ratios are higher than 60, while in the second case, Mack's model systematically underestimates the variability of the payments, leading to lower values compared with the rest of the models and an inadequate risk assessment according to the results of the Kupiec test. The  $Ratio(RR_{0.995}^t)$  and  $Ratio(\sigma)$  of the Stacked-ANN are in line with the majority of the benchmark models, and no extremely high/low risk measures are observed in Table 29.

## Sensitivity analysis of the number of hidden layers

As explained in Section 4.3, the ANNs included within the proposed Stacked-ANN architecture are composed of two hidden layers, each with five neurons. To analyse the impact of the ANN complexity (Cybenko 1989, Hornik et al. 1989, Hornik 1991 and Leshno et al. 1993, among others, introduced the theoretical framework to analyse the approximation capabilities of neural networks) on the predictive power of the Stacked-ANN model, a sensitivity analysis of the number of hidden layers was carried out. Thus, Table 15 compares the configuration selected for the Stacked-ANN model in this paper with two alternative configurations: ANNs composed of one and three hidden layers with five neurons each.

Two main conclusions can be drawn from the results obtained. First, the high level of error of the one hidden layer model demonstrates that more complexity is needed in order to properly predict general insurance reserves. The structure proposed during this study for the Stacked-ANN model (two hidden layers) performs significantly better than this first alternative in every single line of business.

Second, the performance of the three hidden layers alternative is similar to that of the suggested architecture. As no significant differences are observed, the two hidden layer structure is considered more appropriate because the three hidden layer structure adds complexity to the model without a significant improvement in the error rate.

Table 15: Sensitivity analysis of the number of hidden layers

Hidden layers	Line of Business	$\%RMSE(R^t)$	$\%RMSE(P^{t+1})$	$\%RMSE(U^t)$
1	CA	0.840%	0.766%	0.160%
2	CA	0.739%	0.876%	0.141%
3	CA	0.780%	0.643%	0.149%
1	PA	2.512%	3.507%	0.326%
2	PA	0.254%	0.320%	0.033%
3	PA	0.231%	0.115%	0.030%
1	WC	1.398%	1.296%	0.240%
2	WC	1.058%	0.676%	0.182%
3	WC	1.140%	1.030%	0.196%
1	OL	1.419%	1.145%	0.475%
2	OL	0.722%	1.095%	0.242%
3	OL	0.613%	0.794%	0.205%

*Source:* own elaboration

## 4.5 Conclusions

This paper introduced a stochastic reserving model based on stacking different machine learning algorithms (RF, GB and ANN) and reserving models (Chain Ladder and CSR). The predictive power and reserve volatility of the proposed approach, named Stacked-ANN, were compared with stochastic reserving models based on the Chain Ladder technique (ODP and Mack’s model), an individual ANN and CSR, which is a Bayesian loss reserving model.

Three main conclusions were drawn. First, a comparison of the Stacked-ANN with the individual ANN revealed that the predictions of the reserves  $R^t$ , next year payments  $P^{t+1}$  and ultimate losses  $U^t$  made by machine learning algorithms were improved by applying the proposed stacking procedure. The hybrid architecture learns patterns and characteristics from several algorithms and reserving models, resulting in a more flexible and accurate model than an individual ANN, whose inputs for training are limited to the original data.

Second, the empirical results indicated that the Stacked-ANN model is more precise than CSR and the most widely used stochastic reserving models based on the Chain Ladder technique (ODP and Mack’s model). In particular, the  $R^t$  and  $U^t$  predictions made by the Stacked-ANN were more precise than those of ODP and Mack’s model in all the lines of business analysed, while the Bayesian model (CSR) was outperformed by the proposed architecture in three out of four lines of business. It is important to remark that in Other Liability (OL), which is a line of business with a longer duration and therefore a portfolio where the importance of an accurate reserves estimation is especially relevant, the error of the models based on Chain Ladder or Bayesian statistics was more than four times the error of the Stacked-ANN. Therefore, it can be

concluded that machine or deep learning techniques can be used to improve the performance of the traditional reserving techniques based on Bayesian statistics or the Chain Ladder.

With regard to accuracy, it is worth mentioning that the proposed structure of the ANNs (two hidden layers) within the Stacked-ANN model seems to be the optimal configuration according to the empirical results. On the one hand, the error increased significantly when the number of hidden layers is reduced to one. On the other hand, the results demonstrated that increasing the number of hidden layers does not have an impact on the accuracy. Thus, increasing the complexity of the ANNs by up to three hidden layers will extend the training phase without making any significant improvement in the error.

Third, the results of a Kupiec test revealed that the risk estimation made by the Stacked-ANN can be considered as appropriate in all lines of business analysed, while the rest of the benchmark models failed the test at least once. In particular, CSR, ODP and Mack's model were unable to produce an appropriate p-value for the Kupiec test in the Workers' Compensation (WC) business, while the individual ANN failed the test in Commercial Auto (CA) and, as with the previous models, in Workers' Compensation. Taking into consideration that the same log-normal approach was used to obtain the reserves variability of the individual ANN and the Stacked-ANN, it must be mentioned that the stacking procedure not only increases the accuracy but also allows for the simulation of more adequate distribution functions.

The aforementioned robustness and predictive power of the Stacked-ANN compared with other reserving models suggest that further investigation should be conducted about the possible application of this model within the *actuary in the box* approach. The generation of outliers is one of the main problems when using the former methodology with Chain Ladder models. Therefore, the robustness of the Stacked-ANN can be exploited in order to improve the *actuary in the box* methodology, which is widely used to assess the fact that reserves can be insufficient to cover their runoff over a 12-month time horizon.

## 5 Multi-Transformer: A new neural network-based architecture for forecasting S&P volatility

**Authors:** Eduardo Ramos-Pérez, Pablo J. Alonso-González and José Javier Núñez-Velázquez.

**Journal:** Mathematics. 9/15 1-18. ISSN 2227-7390.

**DOI:** 10.3390/math9151794

**Journal Impact Factor in 2020 (last available):** 2.258

**Rank by Journal Impact Factor in 2020 (last available):** 24/330 in Mathematics (Q1-D1).

**Journal Citation Indicator in 2020 (last available):** 2.10

**Rank by Journal Citation Indicator in 2020 (last available):** 18/471 in Mathematics (Q1-D1).

**Published:** July 2021

**Citations:** 2 (Google Scholar).

**Arxiv Repository:** <https://arxiv.org/abs/2109.12621>

This manuscript version is made available under the CC-BY-NC-ND 4.0 license  
<http://creativecommons.org/licenses/by-nc-nd/4.0/>

## Abstract

Events such as the Financial Crisis of 2007-2008 or the Covid-19 pandemic caused significant losses to banks and insurance entities. They also demonstrated the importance of using accurate equity risk models and having a risk management function able to implement effective hedging strategies. Stock volatility forecasts play a key role in the estimation of equity risk and, thus, in the management actions carried out by financial institutions. Therefore, this paper has the aim of proposing more accurate stock volatility models based on novel machine and deep learning techniques. This paper introduces a neural network-based architecture, called Multi-Transformer. Multi-Transformer is a variant of Transformer models, which have already been successfully applied in the field of natural language processing. Indeed, this paper also adapts traditional Transformer layers in order to be used in volatility forecasting models. The empirical results obtained in this paper suggest that the hybrid models based on Multi-Transformer and Transformer layers are more accurate and, hence, they lead to more appropriate risk measures than other autoregressive algorithms or hybrid models based on feed forward layers or long short term memory cells.

**Keywords:** Deep Learning, Neural Networks, Risk Management, Stock Volatility, Transformer.

## 5.1 Introduction

Since the Financial Crisis of 2007-2008, financial institutions have enhanced their risk management framework in order to meet the new regulatory requirements set by Solvency II or Basel III. These regulations have the aim of measuring the risk profile of financial institutions and minimizing losses from unexpected events such as the European sovereign debt crisis or Covid-19 pandemic. Even though banks and insurance entities have reduced their losses thanks to the efforts made in the last years, unexpected events still cause remarkable losses to financial institutions. Thus, efforts are still required to further enhance market and equity risk models in which stock volatility forecasts play a fundamental role. Volatility, understood as a measure of an asset uncertainty (Hull 2015 and Rajashree and Ranjeeeta 2015), is not directly observed in stock markets. Thus, taking into consideration the stock market movements, a statistical model is applied in order to compute the volatility of a security.

GARCH-based models (Engle 1982 and Bollerslev 1986) are widely used for volatility forecasting purposes. This family of models is especially relevant because it takes into consideration the volatility clustering observed by Mandelbrot (1963). Nevertheless, as the persistence of conditional variance tends to be close to zero, Engle and Lee (1999), Haas et al. (2004a), Haas et al. (2004b) and Haas and Paoletta (2012) developed more flexible variations of the traditional GARCH models. In addition, the models introduced by Nelson (1991) (EGARCH) and Glosten et al. (1993) (GJR-GARCH) take into consideration that stocks volatility behaves differently depending on the market trend, bearish or bullish. Multivariate GARCH models were developed by Kraft and Engle (1982) and Engle et al. (1984). Bollerslev et al. (1988) applied the previous model to financial time series, while Tse and Tsui (2002)

introduced a time-varying multivariate GARCH. Dynamic conditional correlation GARCH, BEKK-GARCH and Factor-GARCH were other variants of this family that were developed by Engle (2002), Engle and Kroner (1995) and Engle et al. (1990) respectively. Finally, it is worth mentioning that, in contrast to classical GARCH, the first-order zero-drift GARCH model (ZD-GARCH) proposed by Zhang et al. (2018) is non-stationary regardless of the sign of Lyapunov exponent and, thus, it can be used for studying heteroscedasticity and conditional heteroscedasticity together.

Another relevant family is composed by stochastic volatility models. As they assume that volatility follows its own stochastic process, these models are widely used in combination with Black-Scholes formula to assess derivatives price. The most popular process of this family is the Heston (1993) model which assumes that volatility follows an Cox-Ingersoll-Ross (Cox, Ingersoll, and Ross 1985) process and stock returns a Brownian motion. The main challenge of the Heston model is the estimation of its parameters. Melino and Turnbull (1990) and Andersen and Sorensen (1999) proposed a generalized method of moments to obtain the parameters of the stochastic process, while Durbin and Koopman (1997), Broto and Ruiz (2004), Danielsson (2004) and Andersen (2009) used a simulation approach to estimate them. Other relevant stochastic volatility processes are Hull-White (Hull and White 1987) and SABR (Hagan et al. 2002) models.

The last relevant family is composed of those models based on machine and deep learning techniques. Even though GARCH models are considered part of the machine learning tool-kit, these models are considered another different family due to the significant importance that they have in the field of stock volatility. Thus, this family takes into consideration the models based on the rest of the machine and deep learning algorithms such as artificial neural networks (McCulloch and Pitts 1943), gradient boosting with regression trees (Friedman 2000), random forests (Breiman 2001) or support vector machines (Cortes and Vapnik 1995). Gestel et al. (2001), Gupta and Dhingra (2012), Dias et al. (2019) applied machine learning techniques such as Support Vector Machines or hidden Markov models to forecast financial time series. Hamid and Iqbal (2002) applied Artificial Neural Networks (ANNs) to demonstrate that the implied volatility forecasted by this algorithm is more accurate than Barone-Adesi and Whaley models.

ANNs have been also combined with other statistical models with the aim of improving the forecasting power of individual ANNs. The most common approach applied in the field of stocks volatility is merging GARCH-based models with ANNs. Roh (2006), Hajizadeh et al. (2012), Kristjanpoller et al. (2014), Monfared and Enke (2014), Lu et al. (2016), Kim and Won (2018) and Back and Kim (2018) developed different architectures based in the previous approach for stock volatility forecasting purposes. All these authors demonstrated that hybrid models overcome the performance of traditional GARCH models in the field of stock volatility forecasting. It is also worth mentioning the contribution of Bildirici and Ersin (2009), who combined different GARCH models with ANNs in order to compare their predictive power. ANN-GARCH models have been also applied to forecast other financial time series

such as metals (Kristjanpoller and Minutolo 2015 and Kristjanpoller and Hernández 2017) or oil (Kristjanpoller and Minutolo 2016 and Verma 2021) volatility. Apart from the combination with GARCH-based models, ANNs have been merged with other models for volatility forecasting purposes. Ramos-Pérez et al. (2019) merged ANNs, random forests, support vector machines (SVM) and gradient boosting with regression trees in order to forecast S&P500 volatility. This model overcame the performance of a hybrid model based on feed forward layers and GARCH. Vidal and Kristjanpoller (2020) proposed an architecture based on convolutional neural networks (CNNs) and long-short term memory (LSTM) units to forecast gold volatility. LSTMs were also used by Jung and Choi (2021) to forecast currency exchange rates volatility. It is also worth mentioning that GARCH models have not been only merged with ANNs, Peng et al. (2018) combined SVM with GARCH-based models in order to predict cryptocurrencies volatility.

The aim of this paper is to introduce a more accurate stock volatility model based on an innovative machine and deep learning technique. For this purpose, hybrid models based on merging Transformer and Multi-Transformer layers with other approaches such as GARCH-based algorithms or LSTM units are introduced by this paper. Multi-Transformer layers, which are also introduced in this paper, are based on the Transformer architecture developed by Vaswani et al. (2017). Transformer layers have been successfully implemented in the field of natural language processing (NLP). Indeed, the models developed by Devlin et al. (2018) and Brown et al. (2020) demonstrated that Transformer layers are able to overcome the performance of traditional NLP models. Thus, this recently developed architecture is currently considered the state-of-the-art in the field of NLP. In contrast to LSTM, Transformer layers do not incorporate recurrence in their structure. This novel structure relies on a multi-head attention mechanism and positional embeddings in order to forecast time series. As Vaswani et al. (2017) developed Transformer for NLP purposes, positional embeddings are used in combination with word embeddings. The problem faced in this paper is the forecasting of stock volatility and, thus, the word embedding is not needed and the positional embedding has been modified as it is explained in Section 5.2.

In contrast to Transformer, Multi-Transformer randomly selects different subsets of training data and merges several multi-head attention mechanisms to produce the final output. Following the intuition of bagging, the aim of this architecture is to improve the stability and accurateness of the attention mechanism. It is worth mentioning that the GARCH-based algorithms used in combination with Transformer and Multi-Transformer layers are GARCH, EGARCH, GJR-GARCH, TrGARCH, FIGARCH and AVGARCH.

Therefore, three main contributions are provided by this study. First, Transformer layers are adapted in order to forecast stocks volatility. In addition, an extension of the previous structure is presented (Multi-Transformer). Second, this paper demonstrates that merging Transformer and Multi-Transformer layers with other models lead to more accurate volatility forecasting models. Third, the proposed stock volatil-

ity models generate appropriate risk measures in low and high volatility regimes. The Python implementation of the volatility models proposed in this paper is available in <https://github.com/EduardoRamosP/MultiTransformer>.

As it is shown by the extensive literature included in this section, stock volatility forecasting has been a relevant topic not only for financial institutions and regulators but also for the academia. As financial markets can suffer drastic sudden drops, it is highly desirable to use models that can adequately forecast volatility. It is also useful to have indicators that can accurately measure risk. This paper makes use of recent deep and machine learning techniques to create more accurate stock volatility models and appropriate equity risk measures.

The rest of the paper is organized as follows: Section 5.2 describes the dataset, the measures used for validating the volatility forecasts and provides a look at the volatility models used as benchmark. Then, this section presents the volatility forecasting models proposed in this paper, which are based on Transformer and Multi-Transformer layers. As NLP Transformers need to be adapted in order to be used for volatility forecasting purposes and Multi-Transformer layers are introduced by this paper, explanations about the theoretical background of these structures are also given. The analysis of empirical results is presented in section 5.3. Finally, the results are discussed in Section 5.4, followed by concluding remarks in Section 5.5.

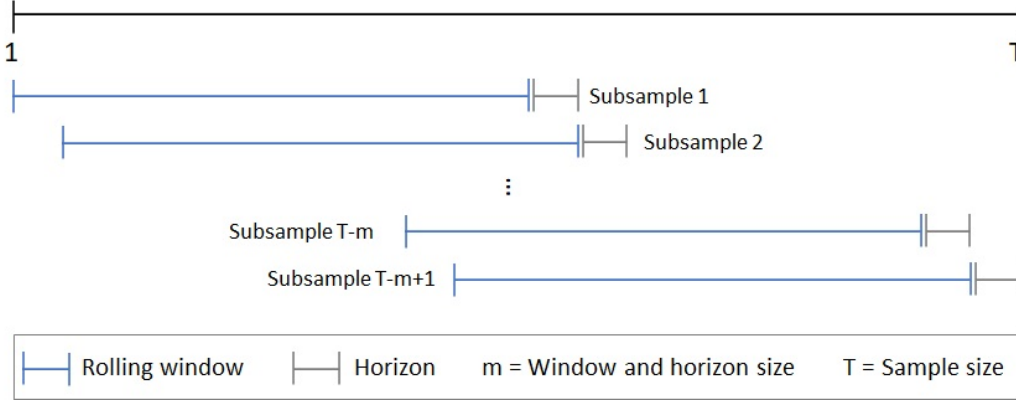
## 5.2 Materials and Methods

This section is divided in five different subsections. The first one (Section 5.2) describes the data for fitting the models. The measures for validating the accuracy and value at risk (VaR) of each stock volatility model are explained in Section 5.2. Section 5.2 presents the stock volatility models and algorithms used for benchmarking purposes. Section 5.2 explains the adaptation of Transformer layers in order to be used for volatility forecasting purposes and, finally, the Multi-Transformer layers and the models based on them are presented in Section 5.2.

### Data and model inputs

The proposed architectures and benchmark models are fitted using the rolling window approach (see Figure 17). This widely used methodology has been applied in finance, among others, by Swanson (1998), Goyal and Welch (2002), Zivot and Wang (2006) and Molodtsova and Papell (2012). Rolling window uses a fixed sample length for fitting the model and, then, the following step is forecasted. As in this paper the window size is set to 650 and the forecast horizon to 1, the proposed and benchmark models are fitted using the last 650 S&P trading days and, then, the next day volatility is forecasted. This process is repeated until the whole period under analysis is forecasted. The periods used as training and testing set will be defined at the end of this subsection.

Figure 17: Rolling window methodology



The input variables of the models proposed are the daily logarithmic returns ( $r_{t-i}$ ) and the standard deviation of the last five daily logarithmic returns:

$$\sigma_{t-1} = \sqrt{\frac{\sum_{i=1}^n (r_{t-i} - E[r])^2}{n-1}} \quad (76)$$

As Multi-Transformer, Transformer and LSTM layers are able to manage time series, a lag of the last 10 observations of the previous variables are taken into consideration for fitting these layers. Thus, the input variables are:

$$X_1 = (\sigma_{t-1}, \sigma_{t-2}, \dots, \sigma_{t-10}) \quad (77)$$

$$X_2 = (r_{t-1}, r_{t-2}, \dots, r_{t-10}) \quad (78)$$

In accordance with other studies such as Roh (2006) or Ramos-Pérez et al. (2019), the realized volatility is used as response variable for the models based on ANNs;

$$Y = \hat{\sigma}_{i,t} = \sqrt{\frac{\sum_{n=0}^{i-1} (r_{t+n} - E[r_f])^2}{i-1}} \quad (79)$$

where  $E[r_f] = \sum_{n=0}^{i-1} r_{t+n}/i$  and  $i = 5$ . As shown in the previous formula, the realized volatility can be defined as the standard deviation of future logarithmic returns.

The dataset for fitting and evaluating the volatility forecasting models contains market data of S&P from 01/01/2008 to 31/12/2020. The optimum configuration of the models is obtained by applying the rolling window approach and selecting the configuration which minimizes the error (RMSE) in the period going from 01/01/2008 to 31/12/2015. The optimum configuration in combination with the rolling window methodology is applied in order to forecast the volatility contained in the testing set (from 01/01/2016 to 31/12/2020). The empirical results presented in Section 5.3 are based on the forecasts of the testing set.

## Models validation

This subsection presents the measures selected for validating and comparing the performance of the benchmark models with the algorithms proposed in this paper.

The mean absolute value (*MAE*) and the root mean squared error (*RMSE*) have been selected for validating the forecasting power of the different stock volatility models:

$$MAE = \sum_{t=1}^N \frac{|\sigma_{i,t} - \hat{\sigma}_{i,t}|}{N} \quad / \quad RMSE = \sum_{t=1}^N \frac{(\sigma_{i,t} - \hat{\sigma}_{i,t})^2}{N} \quad (80)$$

where  $N$  is the total number of observations.

The validation carried out by this study is not only interested on the accuracy, but also on the appropriateness of the risk measures generated by the different stock volatility forecasting models. In accordance with Solvency II Directive, 99.5% VaR has been selected as risk measure. Although Solvency II has the aim of obtaining the yearly VaR, the calculations carried out in this paper will be based on a daily VaR in order to have more data points and, thus, more robust conclusions on the performance of the different models. The parametric approach developed by Kupiec (1995) is used for validating the different VaR estimations. The aim of this test is accepting (or rejecting) the hypothesis that the number of VaR exceedances are aligned with the confidence level selected for calculating the risk measure. In addition to the previous test, the approach suggested by Christoffersen et al. (1997) is also applied in order to validate the appropriateness of VaR.

## Benchmark models

This subsection introduces the benchmark models used in this paper: GARCH, EGARCH, AVGARCH, GJR-GARCH, TrARCH, FIGARCH and two architectures that combine GARCH-based algorithms with ANN and LSTM respectively. The GARCH-based algorithms will be fitted assuming that innovations,  $\epsilon_t$ , follow a Student's t-distribution. Thus, the returns generated by these models follow a conditional t-distribution (Bauwens et al. 2012).

The generalized autoregressive conditional heteroskedasticity (GARCH) model developed by Bollerslev (1986) has been widely used for stock volatility forecasting purposes. GARCH(p,q) has the following expression:

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^q \alpha_i r_{t-i}^2 + \sum_{i=1}^p \beta_i \sigma_{t-i}^2 \quad / \quad \hat{r}_t = \hat{\sigma}_t \epsilon_t \quad (81)$$

where  $\omega_i$ ,  $\alpha_i$  and  $\beta_i$  are the parameters to be estimated,  $r_{t-i}$  the previous returns and  $\sigma_{t-i}^2$  the last observed volatility. As previously stated, innovations ( $\epsilon_t$ ) follow a Student's t-distribution.

The absolute value GARCH (Taylor 1986), AVGARCH(p,q), is similar to the traditional GARCH model. In this case, the absolute value of previous return and volatility is taken into consideration to forecast volatility:

$$\hat{\sigma}_t = \omega + \sum_{i=1}^q \alpha_i |r_{t-i}| + \sum_{i=1}^p \beta_i \sigma_{t-i} \quad (82)$$

As volatility behaves differently depending on the market tendency, models such as EGARCH, GJR-GARCH or TrGARCH were developed. EGARCH(p,q) (Nelson 1991) has the following expression for the logarithm of stocks volatility:

$$\log \hat{\sigma}_t^2 = \omega + \sum_{i=1}^p \alpha_i \log \hat{\sigma}_{t-i}^2 + \sum_{i=1}^q (\beta_i e_{t-i} + \gamma_i (|e_{t-i}| - E(|e_{t-i}|))) \quad (83)$$

where  $\omega_i$ ,  $\alpha_i$ ,  $\beta_i$  and  $\gamma_i$  are the parameters to be estimated and  $e_t = r_t/\sigma_t$ . The GJR-GARCH(p,o,q) developed by Glosten et al. (1993) has the following expression:

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^q \alpha_i r_{t-i}^2 + \sum_{i=1}^o \gamma_i r_{t-i}^2 I_{[r_{t-1} < 0]} + \sum_{i=1}^p \beta_i \sigma_{t-i}^2 \quad (84)$$

As with EGARCH model,  $\omega_i$ ,  $\alpha_i$ ,  $\beta_i$  and  $\gamma_i$  are the parameters to be estimated.  $I_{[r_{t-1} < 0]}$  takes the value of 1 when the subscript condition is met. Otherwise  $I_{[r_{t-1} < 0]} = 0$ . The volatility of the Threshold GARCH(p,o,q) (TrGARCH) model is obtained as follows:

$$\hat{\sigma}_t = \omega + \sum_{i=1}^q \alpha_i |r_{t-i}| + \sum_{i=1}^o \gamma_i |r_{t-i}| I_{[r_{t-i} < 0]} + \sum_{i=1}^p \beta_i \sigma_{t-i} \quad (85)$$

As with the previous two architectures,  $\omega_i$ ,  $\alpha_i$ ,  $\beta_i$  and  $\gamma_i$  are the model parameters. The last GARCH-based algorithm used in this paper is the fractionally integrated GARCH (FIGARCH) model developed by Baillie et al. (1996). The conditional variance dynamic is

$$\hat{\sigma}_t = \omega + \left[1 - \beta L - \phi L(1 - L)^d\right] \epsilon_t^2 + \sigma h_{t-1} \quad (86)$$

where  $L$  is the lag operator and  $d$  the fractional differencing parameter.

In addition to the previous approaches, two other hybrid models based on merging autoregressive algorithms with ANNs and LSTMs are also used as benchmark. Figure 18 shows the architecture of ANN-GARCH and LSTM-GARCH. The inputs of the algorithms are the following:

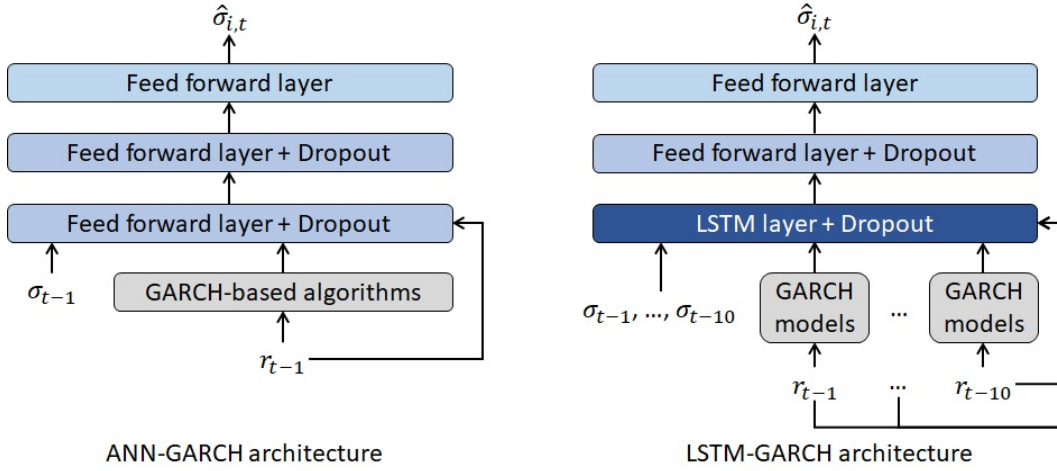
- The last daily logarithmic return,  $r_{t-1}$ , for the ANN-GARCH and the last ten in the case of the LSTM-GARCH (as explained in Section 5.2).
- The standard deviation of the last five daily logarithmic returns:

$$\sigma_{t-1} = \sqrt{\frac{\sum_{i=1}^n (r_{t-i} - E[r])^2}{n - 1}} \quad (87)$$

where  $E[r] = \sum_{i=1}^n r_{t-i}/n$  and  $n = 5$ . As with the previous input variable, the last standard deviation is considered in the ANN-GARCH, whereas the last ten are taken into consideration by the LSTM-GARCH architecture.

The GARCH-based algorithms included within the ANN-GARCH and LSTM-GARCH models are the six algorithms previously presented in this same subsection (GARCH, EGARCH, AVGARCH, GJR-GARCH, TrARCH, FIGARCH).

Figure 18: ANN-GARCH and LSTM-GARCH architectures



As explained in Section 5.2, the true implied volatility,  $\sigma_{i,t}$ , is used as response variable to train the models. This variable is the standard deviation of the future logarithmic returns:

$$\hat{\sigma}_{i,t} = \sqrt{\frac{\sum_{n=0}^{i-1} (r_{t+n} - E[r_f])^2}{i-1}} \quad (88)$$

where  $E[r_f] = \sum_{n=0}^{i-1} r_{t+n}/i$ . In this paper,  $i = 5$ .

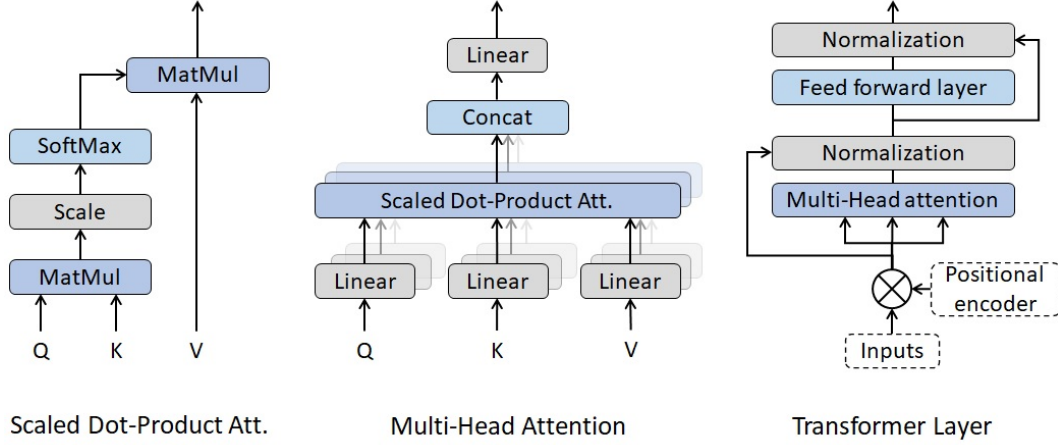
As it is shown in Figure 18, the input of the ANN-GARCH model is processed by two feed forward layers with dropout regularization. These layers have 16 and 8 neurons respectively. The final output is produced by a feed forward layer with one neuron. In the case of the LSTM-GARCH, inputs are processed by a LSTM layer with 32 units and two feed forward layers with 8 and 1 neurons respectively in order to produce the final forecast.

## Transformer-based models

Before explaining the volatility models based on Transformer layers (see Figure 19), all the modifications applied to their architecture are presented in this subsection.

As previously stated, Transformer layers (Vaswani et al. 2017) were developed for NLP purposes. Thus, some modifications are needed in order to apply this layer for volatility forecasting purposes.

Figure 19: Transformer and Multi-Head attention mechanism



In contrast to LSTM, recurrence is not present in the architecture of Transformer layers. The two main components used by these layers in order to deal with time series are the following:

- **Positional encoder.** As previously stated, Transformer layers have no recurrence structure. Thus, the information about the relative position of the observations within the time series needs to be included in the model. To do so, a positional encoding is added to the input data. In the context of NLP, Vaswani et al. (2017) suggested the following wave functions as positional encoders:

$$PE_{(pos, 2i)} = \sin(pos/1000^{2i/dim}) \quad (89)$$

$$PE_{(pos, 2i+1)} = \cos(pos/1000^{2i/dim}) \quad (90)$$

where  $dim$  is the total number of explanatory variables (or word embedding dimension in NLP) used as input in the model,  $pos$  is the position of the observation within the time series and  $i = (1, 2, \dots, dim - 1)$ . This positional encoder modifies the input data depending on the lag of the time series and the embedding dimension used for the words.

As volatility models do not use words as inputs, the positional encoder is modified in order to avoid any variation of the inputs depending on the number of time series used as input. Thus, the positional encoder suggested in this paper changes depending on the lag, but it remains the same across the different explanatory variables introduced in the model. As in the previous case, a wave

function plays the role of positional encoder:

$$PE_{pos} = \cos\left(\pi \frac{pos}{N_{pos} - 1}\right) = \sin\left(\frac{\pi}{2} + \pi \frac{pos}{N_{pos} - 1}\right) \quad (91)$$

where  $pos = (0, 1, \dots, N_{pos} - 1)$  is the position of the observation within the time series and  $N_{pos}$  maximum lag.

- Multi-Head attention. It can be considered the key component of the Transformer layers proposed by Vaswani et al. (2017). As shown in Figure 19, Multi-Head attention is composed of several scaled dot-product attention units running in parallel. Scaled dot-product attention is computed as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (92)$$

where  $Q$ ,  $K$  and  $V$  are input matrices and  $d_k$  the number of input variables taken into consideration within the dot-product attention mechanism. Multi-Head attention splits the explicative variables in different groups or ‘heads’ in order to run the different scaled dot-product attention units in parallel. Once the different heads are calculated, the outputs are concatenated (*Concat* operator) and connected to a feed forward layer with linear activation. Thus, the Multi-Head attention mechanism has the following expression:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O \quad (93)$$

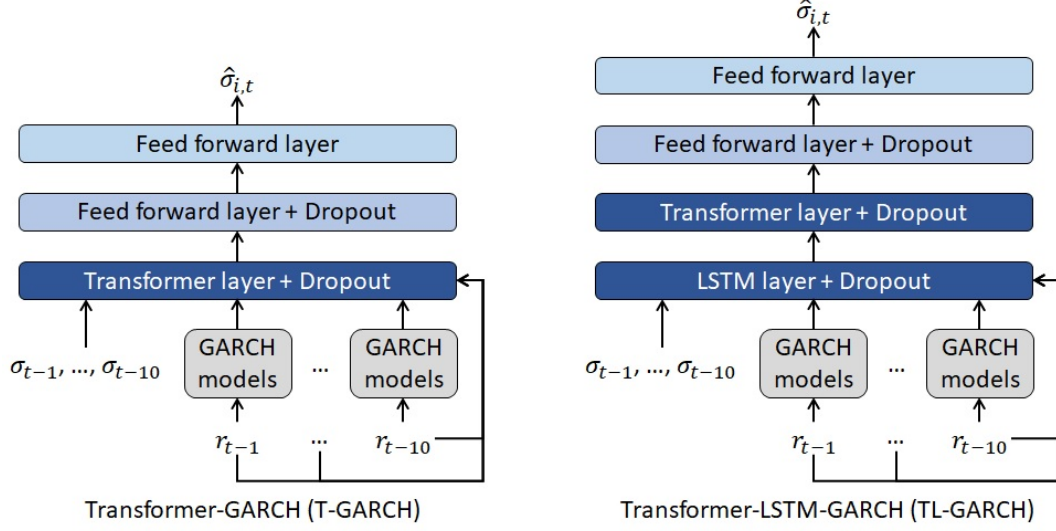
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (94)$$

where  $h$  is the number of heads. It is also worth mentioning that all the matrices of parameters ( $W_i^Q$ ,  $W_i^K$ ,  $W_i^V$  and  $W^O$ ) are trained using feed forward layers with linear activations.

In addition to the scaled dot-product and the Multi-Head attention mechanisms, Figure 19 shows the Transformer layers used in this paper. As suggested by Vaswani et al. (2017), the Multi-Head attention is followed by a normalization, a feed forward layer with ReLU activation and, again, a normalization layer. Transformer layers also include two residual connections (He et al. 2016). Thanks to these connections, the model will decide by itself if the training of some layers needs to be skipped during some phases of the fitting process.

The modified version of Transformer layers explained in the previous paragraphs are used in the volatility models presented in Figure 20. The T-GARCH architecture proposed in this paper merges the six GARCH algorithms presented in Section 5.2 with Transformer and feed forward layers in order to forecast  $\hat{\sigma}_{i,t}$ . In addition to the previous algorithms and layers, TL-GARCH includes a LSTM with 32 units. In this last model, the temporal structure of the data is recognized and modelled by the LSTM layer and, thus, no positional encoder is needed in the Transformer layer. Both models have the following characteristics:

Figure 20: T-GARCH and TL-GARCH volatility models



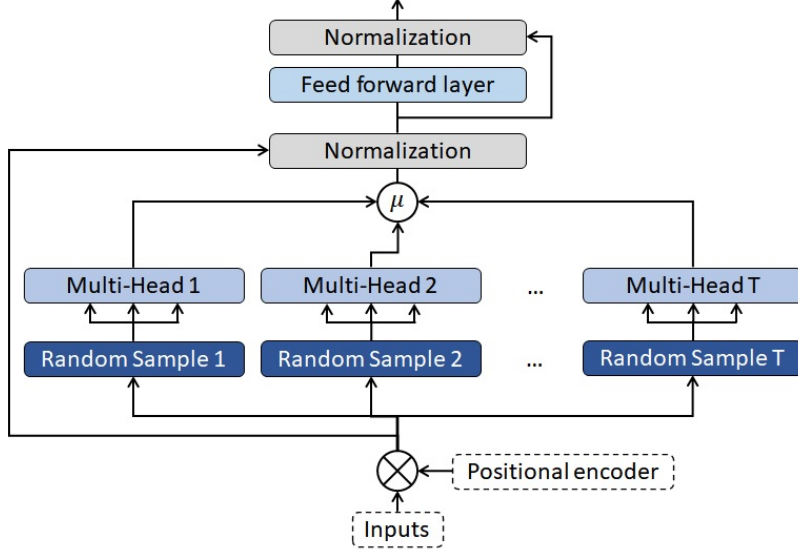
- Adaptive Moment Estimator (ADAM) is the algorithm used for updating the weights of the feed forward, LSTM and Transformer layers. This algorithm takes into consideration current and previous gradients in order to implement a progressive adaptation of the initial learning rate. The values suggested by Kingma and Ba (2014) for the ADAM parameters are used in this paper and the initial learning rate is set to  $\delta = 0.01$ .
- The feed forward layers with dropout present in both models have 8 neurons, while the output layer has just one.
- The level of dropout regularization  $\theta$  (Srivastava et al. 2014) is optimized with the training set mentioned in Section 5.2.
- The loss function used for weights optimization and back propagation purposes is the mean squared error.
- Batch size is equal to 64 and the models are trained during 5,000 epochs in order to obtain the final weights.

### Multi-Transformer-based models

This subsection presents the Multi-Transformer layers and the volatility models based on them. The Multi-Transformer architecture proposed in this paper is a variant of the Transformer layers proposed by Vaswani et al. (2017). The main differences between both architectures are the following:

- As shown in Figure 21, Multi-Transformer layers generate  $T$  different random samples of the input data. In the volatility models proposed in this paper, 90%

Figure 21: Multi-Transformer architecture



of the observations of the database are randomly selected in order to compute the different samples.

- Multi-Transformer architecture is composed of  $T$  Multi-Head attention units (in this paper  $T = 5$ ), one per each random sample of the input data. Then, the average of the different units is computed in order to obtain the final attention matrix. Thus, the Average Multi-Head (AMH) mechanism present in Multi-Transformer can be defined as follows:

$$AMH(Q, K, V) = \frac{\sum_{t=1}^T \text{Concat}(\text{head}_{1,t}, \dots, \text{head}_{h,t}) W_t^O}{T} \quad (95)$$

$$\text{head}_{i,t} = \text{Attention}(Q_t W_{i,t}^Q, K_t W_{i,t}^K, V_t W_{i,t}^V) \quad (96)$$

As with the Transformer architecture applied in this paper, the positional encoder used is  $PE_{pos}$  instead of  $PE_{(pos, 2i)}$  and  $PE_{(pos, 2i+1)}$ .

The aim of the Multi-Transformer layers presented in the paper is to improve the stability and accuracy by applying bagging (Breiman 1996) to the attention mechanism. This technique is usually applied to algorithms such as linear regression, neural networks or decision trees. Instead of applying the procedure to all the data received by the model, the proposed methodology applies bagging only to the key component of the layer architecture.

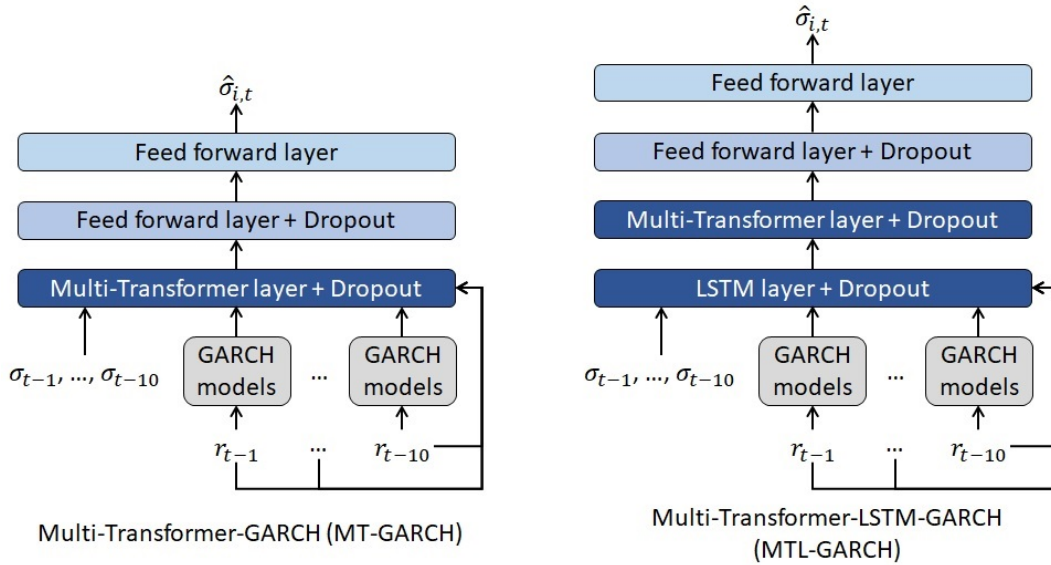
The computational power required by bagging is one of the main limitations of this technique. As Multi-Transformer applies bagging to the attention mechanisms, their weights are trained several times in each epoch. Nevertheless, bagging is not applied to the rest of the layer weights and, thus, this offsets partially the previous limitation.

It is also worth mentioning that bagging preserves the bias and this may result in underfitting.

On the other hand, this technique should bring two main advantages to the Multi-Transformer layer. First, bagging reduces significantly the error variance. Second, the aggregation of learners using this technique leads to a higher accuracy and reduces the risk of overfitting.

The structure of the volatility models based on Multi-Transformer layers (Figure 22) is similar to the architectures presented in Section 5.2. The MT-GARCH merges Multi-Transformer and feed forward layers with the six GARCH models presented in Section 5.2. In addition to the previous algorithms and layers, MTL-GARCH adds a LSTM with 32 units. The rest of the characteristics such as the optimizer, the number of neurons of the feed forward layers or the level of dropout regularization are the same than those presented in the previous section for T-GARCH and TL-GARCH.

Figure 22: MT-GARCH and MTL-GARCH volatility models



The risk measures of ANN-GARCH, LSTM-GARCH and all the models introduced by this paper (sections 5.2 and 5.2) are calculated assuming that daily log-returns follow a non-standardize Student's t-distribution with standard deviation equal to the forecasts made by the volatility models. It is worth mentioning that Student's t-distribution generates more appropriate risk measures than normal distribution due to the shape of its tail (Jeremic and Terzić 2014 and McNeil et al. 2015). In addition, this assumption is in line with the GARCH-based models used as benchmark and the inputs of the hybrid models presented in this paper.

### 5.3 Results

In this section, the forecasts and the risk measures of the volatility models presented in previous sections are compared with the ones obtained from the benchmark models. In addition, the following subsection shows the optimum hyperparameters of the benchmark and proposed hybrid volatility models.

#### Fitting of models based on neural networks

As explained in Section 5.2, rolling window approach (Swanson 1998, Goyal and Welch 2002, Zivot and Wang 2006 and Molodtsova and Papell 2012 among others) is applied for fitting the algorithms. The training set used for optimizing the level of dropout regularization contains S&P returns and observed volatilities from 01/01/2008 to 31/12/2015. Table 16 presents the error by model and level of  $\theta$ .

Table 16: RMSE by level of  $\theta$

Model	$\theta = 0$	$\theta = 0.05$	$\theta = 0.10$	$\theta = 0.15$
ANN-GARCH	0.0351	0.0092	0.0085	0.0082
LSTM-GARCH	0.0065	0.0057	0.0056	0.0054
T-GARCH	0.0089	0.0076	0.0072	0.0074
TL-GARCH	0.0050	0.0045	0.0044	0.0045
MT-GARCH	0.0068	0.0062	0.0064	0.0064
MTL-GARCH	0.0047	0.0045	0.0042	0.0044

*Source:* own elaboration

The results of the optimization process reveals that  $\theta = 0$  generates higher error rates than the rest of the possible values regardless of the model. This means that models based on architectures such as Transformer, LSTM or feed forward layers need an appropriate level of regularization in order to avoid overfitting. According to the results, this is especially relevant for ANN-GARCH, where the error strongly depends on the level of regularization. The dropout level that minimizes the error of each model is selected.

#### Comparison against benchmark models

Once the optimum dropout level of each of the proposed volatility forecasting models based on Transformer and Multi-Transformer is selected, their performance is compared with the benchmark models (traditional GARCH processes, ANN-GARCH and LSTM-GARCH) presented in Section 5.2.

Tables 17 and 18 present the validation error (RMSE and MAE) by year and model. The column ‘Total’ shows the error of the whole test period (from 01/01/2016 to 31/12/2020). The main conclusions drawn from the these tables are the following:

- Traditional GARCH processes are outperformed by models based on merging artificial neural network architectures such as feed forward, LSTM or Transformer layers with the outcomes of autoregressive algorithms (also named hybrid models).
- The comparison between ANN-GARCH and the rest of the volatility forecasting models based on artificial neural networks (LSTM-GARCH, T-GARCH, TL-GARCH, MT-GARCH and MTL-GARCH) reveals that feed forward layers lead to less accurate forecasts than other architectures. Multi-Transformer, Transformer and LSTM were specially created to forecast time series and, thus, the volatility models based on these layers are more accurate than ANN-GARCH.
- Merging Multi-Transformer and Transformer layers with LSTMs leads to more accurate predictions than traditional LSTM-based architectures. Indeed, TL-GARCH achieves better results than LSTM-GARCH, even though the number of weights of TL-GARCH is significantly lower. Thus, the novel Transformer and Multi-Transformer layers introduced for NLPs purposes can be adapted as described in section 5.2 and 5.2 in order to generate more accurate volatility forecasting models. It is also worth mentioning that Multi-Transformer layers, which were also introduced in this paper, lead to more accurate forecasts thanks to their ability to average several attention mechanisms. In fact, the model that achieves the lower MAE and RMSE is a mixture of Multi-Transformer and LSTM layers (MTL-GARCH).

Table 17: RMSE by volatility model and year

Model	2016	2017	2018	2019	2020	Total
GARCH(1,1)	0.0058	0.0026	0.0095	0.0073	0.1026	0.0464
AVGARCH(1,1)	0.0053	0.0027	0.0076	0.0056	0.0847	0.0383
EGARCH(1,1)	0.0056	0.0028	0.0093	0.0078	0.0880	0.0399
GJR-GARCH(1,1,1)	0.0090	0.0028	0.0126	0.0068	0.1248	0.0565
TrGARCH(1,1,1)	0.0074	0.0027	0.0115	0.0058	0.1153	0.0521
FIGARCH(1,1)	0.0062	0.0029	0.0095	0.0066	0.1011	0.0457
ANN-GARCH	0.0042	0.0023	0.0060	0.0044	0.0171	0.0086
LSTM-GARCH	0.0032	0.0021	0.0043	0.0030	0.0101	0.0054
T-GARCH	0.0048	0.0029	0.0058	0.0044	0.0117	0.0067
TL-GARCH	0.0030	0.0019	0.0033	0.0026	0.0070	0.0040
MT-GARCH	0.0036	0.0021	0.0046	0.0033	0.0096	0.0054
MTL-GARCH	0.0030	0.0016	0.0033	0.0026	0.0066	0.0038

*Source:* own elaboration

Table 18: MAE by volatility model and year

Model	2016	2017	2018	2019	2020	Total
GARCH(1,1)	0.0037	0.0019	0.0058	0.0044	0.0363	0.0105
AVGARCH(1,1)	0.0034	0.0019	0.0049	0.0037	0.0296	0.0087
EGARCH(1,1)	0.0035	0.0020	0.0060	0.0048	0.0333	0.0100
GJR-GARCH(1,1,1)	0.0048	0.0020	0.0074	0.0042	0.0404	0.0118
TrGARCH(1,1,1)	0.0042	0.0020	0.0069	0.0038	0.0365	0.0107
FIGARCH(1,1)	0.0038	0.0021	0.0055	0.0041	0.0361	0.0104
ANN-GARCH	0.0029	0.0019	0.0038	0.0029	0.0095	0.0042
LSTM-GARCH	0.0022	0.0015	0.0027	0.0021	0.0060	0.0029
T-GARCH	0.0035	0.0021	0.0041	0.0031	0.0070	0.0040
TL-GARCH	0.0020	0.0014	0.0021	0.0018	0.0044	0.0023
MT-GARCH	0.0024	0.0016	0.0031	0.0023	0.0057	0.0030
MTL-GARCH	0.0019	0.0012	0.0021	0.0018	0.0041	0.0022

*Source:* own elaboration

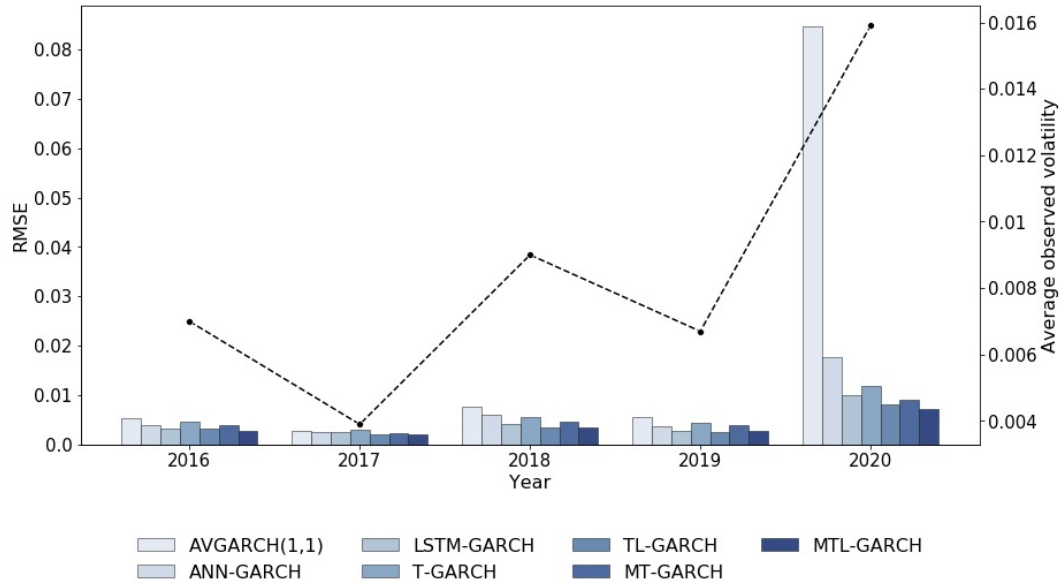
To enhance the analysis of the results shown in tables 17 and 18, Figure 23 collects the RMSE and the observed volatility by year. Notice that only the most accurate GARCH-based model is shown in order to improve the visualization of the graph. The black dashed line shows that the observed volatility of 2020 was significantly higher than the rest of the years due to the turmoil caused by Covid-19 outbreak. As expected, the error of every model is also higher in 2020 because the market volatility was more unpredictable than the rest of the years. Nevertheless, it has to be mentioned that the 2020 forecasts of traditional autoregressive algorithms are significantly less accurate than hybrid models based on architectures such as LSTM, Transformer or Multi-Transformer layers.

Although the observed volatility is lower in years before 2020, autoregressive models are also outperformed by hybrid models. Nevertheless, the difference between both sets of models is remarkably lower.

The p-values of the Kupiec and Christoffersen tests by volatility model and year are shown in tables 19 and 20 respectively. In contrast to the approach suggested by Kupiec, Christoffersen test is not only focused on the total number of exceedances, but it also takes into consideration the number of consecutive VaR exceedances. As stated in Section 5.2, the risk measure and confidence level (99.5% VaR) selected are in line with Solvency II Directive. This regulation sets the principles for calculating the capital requirements and assessing the risk profile of the insurance companies based in the European Union. This law covers not only the underwriting risks but also financial risks such as the potential losses due to variations on the interest rate curves or the equity prices.

The column ‘Total’ of tables 19 and 20 reveal that only TL-GARCH, MT-GARCH and MTL-GARCH produce appropriate risk measures (p-value higher than 0.05 in

Figure 23: Observed volatility and RMSE by year



both tests) for the period 2016-2020. The rest of the models fail both tests and, thus, their risk measures can not be considered to be appropriate for that period.

As with any other statistical test, the higher the number of data points the more relevant are the outcomes obtained from the test. That is the reason why the previous paragraph focuses on the ‘Total’ column and not on the specific results obtained by year. The results by year show that most of the models fail the test in 2020 due to the high level of volatility produced by Covid-19 pandemic.

Table 19: Kupiec test (p-values) by volatility model and year

Model	2016	2017	2018	2019	2020	Total
GARCH(1,1)	0.543	0.540	0.051	0.543	0.052	0.008
AVGARCH(1,1)	0.543	0.540	0.051	0.543	0.052	0.008
EGARCH(1,1)	0.543	0.540	0.051	0.543	0.052	0.008
GJR-GARCH(1,1,1)	0.543	0.540	0.011	0.543	0.190	0.008
TrGARCH(1,1,1)	0.543	0.540	0.051	0.810	0.190	0.042
FIGARCH(1,1)	0.543	0.540	0.051	0.543	0.052	0.008
ANN-GARCH	0.543	0.540	0.001	0.002	0.012	0.001
LSTM-GARCH	0.810	0.186	0.540	0.188	0.190	0.042
T-GARCH	0.188	0.540	0.002	0.543	0.052	0.001
TL-GARCH	0.543	0.540	0.813	0.810	0.810	0.782
MT-GARCH	0.112	0.540	0.540	0.188	0.052	0.089
MTL-GARCH	0.543	0.113	0.113	0.810	0.190	0.910

Source: own elaboration

Table 20: Christoffersen test (p-values) by volatility model and year

Model	2016	2017	2018	2019	2020	Total
GARCH(1,1)	0.522	0.520	0.004	0.523	0.048	0.002
AVGARCH(1,1)	0.522	0.520	0.004	0.523	0.048	0.002
EGARCH(1,1)	0.522	0.520	0.004	0.523	0.048	0.002
GJR-GARCH(1,1,1)	0.522	0.520	0.002	0.523	0.179	0.002
TrGARCH(1,1,1)	0.522	0.520	0.004	0.800	0.179	0.009
FIGARCH(1,1)	0.522	0.520	0.004	0.523	0.048	0.002
ANN-GARCH	0.522	0.520	0.001	0.002	0.002	0.001
LSTM-GARCH	0.800	0.180	0.520	0.177	0.179	0.037
T-GARCH	0.176	0.520	0.001	0.523	0.048	0.001
TL-GARCH	0.522	0.520	0.803	0.800	0.797	0.693
MT-GARCH	0.113	0.520	0.520	0.177	0.048	0.079
MTL-GARCH	0.522	0.113	0.113	0.800	0.179	0.790

*Source:* own elaboration

According to these results, the stock volatility models introduced in this paper (T-GARCH, TL-GARCH, MT-GARCH and MTL-GARCH) produce more accurate estimations and appropriate risk measures in most of the cases. Regarding the models accuracy, it is specially remarkable the difference observed in 2020, where Covid-19 caused a significant turmoil in the stock market. Concerning the appropriateness of equity risk measures, three out of four models based on Transformer and Multi-Transformer pass Kupiec and Christoffersen test for the period 2016-2020, while all the benchmark models fail at least one of them. Notice that the proposed models are compared with other approaches belonging to its own family (ANN-GARCH and LSTM-GARCH) and autoregressive models belonging to the GARCH family.

## 5.4 Discussion

This paper introduced a set of volatility forecasting models based on Transformer and Multi-Transformer layers. As Transformer layers were developed for NLP purposes (Vaswani et al. 2017), their architecture is adapted in order to generate stock volatility forecasting models. Multi-Transformer layers, which are introduced by this paper, have the aim of improving the stability and accuracy of Transformer layers by applying bagging to the attention mechanism. The predictive power and risk measures generated by the proposed volatility forecasting models (T-GARCH, TL-GARCH, MT-GARCH and MTL-GARCH) are compared with traditional GARCH processes and other hybrid models based on LSTM and feed forward layers.

Three main outcomes were drawn from the empirical results. First, hybrid models based on LSTM, Transformer or Multi-Transformer layers outperform traditional autoregressive algorithms and hybrid models based on feed forward layers. The validation error by year shows that this difference is more relevant in 2020, when the volatility of S&P500 was significantly higher than in the previous years due to Covid-19 pandemic. Volatility forecasting models are mainly used for pricing derivatives

and assessing the risk profile of financial institutions. As the more relevant shocks on the solvency position of financial institutions and derivatives prices are observed in high volatility regimes, the accurateness of these models is particularly important in years such as 2020.

The higher performance of hybrid models have been also demonstrated by Roh (2006), Hajizadeh et al. (2012), Kristjanpoller et al. (2014), Monfared and Enke (2014), Lu et al. (2016), Kim and Won (2018) and Back and Kim (2018). These papers merged traditional GARCH models with feed forward layers to predict stock market volatility. This type of models have shown also a superior performance in other financial fields such as oil market volatility (Kristjanpoller and Minutolo 2016 and Verma 2021) and metals price volatility (Kristjanpoller and Minutolo 2015 and Kristjanpoller and Hernández 2017). Notice that this paper does not only present a comparison with traditional autoregressive models, but it also shows that Transformer and Multi-Transformer can lead to more accurate volatility estimations than other hybrid models.

Second, Multi-Transformer layers lead to more accurate volatility forecasting models than Transformer layers. As expected, applying bagging to the attention mechanism has a positive impact on the performance of the models presented in this paper. It is also remarkable that empirical results demonstrate that merging LSTM with Transformer or Multi-Transformer layers has also a positive impact on the models performance. On one hand, the volatility forecasting model based on Multi-Transformer and LSTM (named MTL-GARCH) achieves the best results in the period 2016-2020. On the other hand, the merging of Transformer with LSTM (TL-GARCH) leads to a lower error rate than the hybrid model based only on LSTM layers (LSTM-GARCH) even though the number of weights of the first model is significantly lower. Thus, the use of Transformer layers can lead to simpler and more accurate volatility forecasting models. Notice that Transformer layers are already considered the state of art thanks to BERT (Devlin et al. 2018) and GPT-3 (Brown et al. 2020). These models have been successfully used for sentence prediction, conversational response generation, sentiment classification, coding and writing fiction, among others.

Third, the results of Kupiec and Christoffersen tests revealed that only the risk estimations made by MTL-GARCH, TL-GARCH and MT-GARCH can be considered as appropriate for the period 2016-2020, whereas traditional autoregressive algorithms and hybrid models based on feed forward and LSTM layers failed, at least, one of the tests. As previously stated, volatility does not play only a key role in risk management but also in derivative valuation models. Thus, using a volatility model that generates appropriate risk measures can lead to more accurate derivatives valuation.

## 5.5 Conclusion

Transformer layers are the state of the art in natural language processing. Indeed, the performance of this layer have overcome the performance of any other previous

model in this field (Brown et al. 2020). As Transformer layers were specially created for natural language processing, they need to be modified in order to be used for other purposes. Probably, this is one of the main reasons why this layer have not been already extended to other fields. This paper provides the modifications needed to apply this layer for stock volatility forecasting purposes. The results shown in this paper demonstrates that Transformer layers can overcome also the performance of the main stock volatility models.

Following the intuition of bagging (Breiman 1996), this paper introduces Multi-Transformer layers. This novel architecture has the aim of improving the stability and accuracy of the attention mechanism, which is the core of Transformer layers. According to the results, it can be concluded that this procedure improves the accuracy of stock volatility models based on Transformer layers.

Leaving aside the comparisons between Transformer and Multi-Transformer layers, the hybrid models based on them have overcome the performance of autoregressive algorithms and other models based on feed forward layers and LSTMs. The architecture of these hybrid models (T-GARCH, TL-GARCH, MT-GARCH and MTL-GARCH) based on Transformer and Multi-Transformer layers is also provided in this paper.

According to the results, it is also worth noticing that the risk estimations based on the previous models are specially appropriate. The VaR of most of these models can be considered accurate even in years such as 2020, when Covid-19 pandemic caused a remarkable turmoil in the stock market.

Consequently, the empirical results obtained with the hybrid models based on Transformer and Multi-Transformer layers suggest that further investigation should be conducted about the possible application of them for derivative valuation purposes. Notice that volatility plays a key role in the financial derivatives valuation. In addition, the models can be extended by merging Transformer or Multi-Transformer layers with other algorithms (such as gradient boosting with trees or random forest) or modifying some key assumptions of the attention mechanism.

## 6 Mack-Net model: Blending Mack's model with Recurrent Neural Networks

**Authors:** Eduardo Ramos-Pérez, Pablo J. Alonso-González and José Javier Núñez-Velázquez.

**Journal:** Expert Systems With Applications. ISSN 0957-4174.

**DOI:** 10.1016/j.eswa.2022.117146

**Journal Impact Factor in 2020 (last available):** 6.954

**Rank by Journal Impact Factor in 2020 (last available):** 8/84 in Operations Research and Management Business (Q1-D1).

**Published:** March 2022

**Journal Citation Indicator in 2020 (last available):** 1.68

**Rank by Journal Citation Indicator in 2020 (last available):** 6/99 in Operations Research and Management Business (Q1-D1).

This manuscript version is made available under the CC-BY-NC-ND 4.0 license  
<http://creativecommons.org/licenses/by-nc-nd/4.0/>

## Abstract

In general insurance companies, a correct estimation of liabilities plays a key role due to its impact on management and investing decisions. Since the Financial Crisis of 2007-2008 and the strengthening of regulation, the focus is not only on the total reserve but also on its variability, which is an indicator of the risk assumed by the company. Thus, measures that relate profitability with risk are crucial in order to understand the financial position of insurance firms. Taking advantage of the increasing computational power, this paper introduces a stochastic reserving model whose aim is to improve the performance of the traditional Mack's reserving model by applying an ensemble of Recurrent Neural Networks. The results demonstrate that blending traditional reserving models with deep and machine learning techniques leads to a more accurate assessment of general insurance liabilities.

**Keywords:** Deep Learning, Mack's model, Recurrent Neural Networks, Reserving Risk, Stochastic Reserving.

**AMS Subject Classification:** 62-07, 62P05, 65C60, 90-08.

## 6.1 Introduction

As an accurate estimation of future payments and its volatility allows the management to take correct underwriting and reinsurance decisions, reserving, understood as the calculation of the amount of required reserves for insurance policies, plays a fundamental role in general insurance firms. The interest of investors and regulators in analysing the volatility of financial institutions has increased significantly since the 2007-2008 Financial Crisis. Regulatory requirements have been enhanced with laws such as Solvency II Directive and Solvency Swiss Test in order to assess the risk profile of insurance companies. Since then, investors are not only focused on the profit but also on the level of risk assumed by the insurance firm to obtain it. Thus, indicators that relate profitability with risk such as the Return on Risk Adjusted Capital (Braun et al. 2018) have increased remarkably their influence on the stock prices of financial institutions.

Taking into consideration the historical information, the first reserving methods for estimating the ultimate cost in non-life insurance were only focused on obtaining the most likely scenario. Therefore, these deterministic models were unable to estimate the loss reserve uncertainty. Chain Ladder is the most widely used methodology within this family of reserving models. Nevertheless, Bornhuetter and Ferguson (1972) model tends to perform better when historical information is not stable enough to apply the Chain Ladder methodology.

As stated previously, general insurance firms are not only interested in the expected ultimate cost but also in its volatility. Consequently, different stochastic methodologies linked to the Chain Ladder procedure were developed. One of the most popular models for estimating the loss reserve variability was introduced by Mack (1993). This methodology, commonly known as Mack's model in the literature, derives the reserve variability by focusing on the first two moments. England and Verrall (2006)

developed a bootstrap method that allows the analyst to obtain a complete reserve distribution by applying the free-distribution model of Mack.

Another widely used method to calculate reserve variability is the Overdispersed Poisson (ODP) model, which was developed by Renshaw and Verrall (1998). This method assumes that incremental payments follow an ODP distribution, where their variance is proportional to their mean. In this model, incremental payments must be positive, but this limitation can be overcome by using the quasi-likelihood approach developed by McCullagh and Nelder (1989). As in the case of Mack's model, a complete reserve distribution can be obtained by applying the bootstrap procedure suggested by England and Verrall (1999) and England (2002).

For those cases where data does not follow an ODP distribution, there are other methods based on Chain Ladder such as the log-normal model of Kremer (1982), the gamma procedure of Mack (1991) and the negative binomial methodology developed by Verrall (2000). The distribution function of some of the former models can be obtained by using Bayesian inference. England and Verrall (2006) introduced the procedure for implementing a Bayesian ODP, Mack and Negative Binomial models. The computation of the loss reserve distribution by means of Bayesian inference was recently expanded by Meyers (2015), who introduced several Bayesian Markov Chain Monte-Carlo (MCMC) models for incurred and paid data. These models (Levelled Chain-Ladder, Correlated Chain-Ladder, Levelled Incremental Trend, Correlated Incremental Trend and Changing Settlement Rate) have the aim of improving the performance of the traditional models based on Chain Ladder. This is achieved by including effects such as recognizing the correlation between accident years, applying a skewed distribution to negative incremental payments, introducing a trend over the different development years and implementing the possibility of applying changes in the claim settlement rate.

As incurred and paid data can have different patterns and characteristics, there is a set of loss reserving models, based on the Chain Ladder methodology, which were developed in order to take into consideration both data sources. The most relevant methods within this family are Munich Chain Ladder (Quarg and Mack 2004), Double Chain Ladder Martínez-Miranda et al. (2012) and Paid-Incurred Chain Posthuma et al. (2008). With regard to this last model, it is worth mentioning that Merz and Wüthrich (2010) developed a Bayesian implementation of it, while Happ, Merz, and Wüthrich (2012) and Happ and Wüthrich (2013) introduced methods strongly related to this approach. Finally, Halliwell (2009) and Venter (2008) used incurred and paid data to develop regression-based reserving models, while Pigeon et al. (2014), Antonio and Plat (2014) and Martínez-Miranda et al. (2013b) used both sources of information to estimate the expected ultimate cost.

The increase of the computational power and the success of machine and deep learning in many fields LeCun et al. (2015b), Silver et al. (2016), Silver et al. (2017), Brown et al. (2020) and Ramos-Pérez et al. (2021a) have facilitated the formation of a new family of reserving models based on these techniques. Gabrielli and Wüthrich (2018)

and Wüthrich (2018b) applied Artificial Neural Networks (ANN) to predict claim reserves, while Lopez et al. (2019), Baudry and Robert (2019) and Wüthrich (2018a) used a tree-based algorithm, extremely randomized trees Geurts et al. (2006) and regression trees respectively for that purpose. Gabrielli et al. (2018) and Gabrielli (2019) demonstrated that it is possible to embed traditional Chain Ladder techniques (ODP model) into a neural network framework. This algorithm was also used by Kuo (2018) in order to predict the expected future payments. Ramos-Pérez et al. (2021b) combined ANNs with Random Forests (Breiman 2001) and Gradient Boosting with regression trees (Friedman 2000) in order to predict general insurance reserves. In addition to the previous reserving models, Duma et al. (2011) applied support vector machines to classify data in homogeneous groups of risks before the reserve calculation.

The stochastic reserving model presented in this paper (Mack-Net) combines Recurrent Neural Networks Rumelhart et al. (1986b) with Mack’s model in order to produce more accurate reserve predictions and risk measures. For each individual triangle, an ensemble of Recurrent Neural Networks (RNNs) is fitted in order to forecast both the future payments and the Mack’s model parameters. In a second stage, a bootstrap method based on Mack’s model is combined with the former predictions in order to compute a full reserve distribution. Consequently, the proposed model has the aim of improving the performance of Mack’s model by applying RNNs, which can learn more features than the Chain Ladder technique. Mack-Net model differs from other methods in many ways but the two main differences are explained. First of all, most of the existing reserving models based on machine and deep learning does not produce an estimate of the reserves variability. The few of them that can produce this estimation need to assume a pre-defined theoretical distribution for the payments or incurred cost. Nevertheless, the suggested methodology produces a full reserve distribution without considering any assumption about the payments or incurred cost distribution. Second, information from the same portfolios of several entities tend to be used for fitting reserving models based deep or machine learning techniques. As suggested by regulations like Solvency II Directive, actuaries must aggregate data in homogeneous risk groups, leading to a situation where individual companies do not have available several triangles with similar characteristics to fit the former models. Besides regulatory requirements, if individual companies split triangles with similar characteristics into different pieces, the resulting triangles will not be enough robust in most of the cases. As only one triangle is needed to fit the Mack-Net model, this problem is not present in the suggested methodology. The implementation of the MackNet model in R, the database used for fitting the models and code examples are available in <https://github.com/EduardoRamosP/MackNet>

The proposed model has the aim of producing a more appropriate risk estimation and reserve distribution than other stochastic reserving approaches. Regulations such as Solvency II, Swiss Solvency Test or IFRS promote the use of stochastic models. For example, Risk Adjustment of IFRS 17 has to be based on a certain percentile of the reserve distribution and Reserving Risk of Solvency II Directive can be derived from an stochastic reserving model. Therefore, the calculation of an accurate reserve

distribution can lead to lower solvency requirements and less liabilities (Risk Adjustment). It is also worth mentioning that an accurate estimation of the risk profile can lead to a more efficient risk strategy, better portfolio management actions and, therefore, an optimization of the profit-risk indicators. Since the Financial Crisis of 2007-2008, the valuation of financial institutions is not only based on the future profit but also on its volatility or uncertainty. Nowadays, profit-risk indicators are particularly relevant for the valuation of financial institutions.

The rest of the paper proceeds as follows: Section 6.2 defines the validations metrics and models used as benchmark to assess the performance of the suggested method. In Section 6.3, the theoretical background and architecture of the Mack-Net model are explained. Empirical results of the benchmark and proposed model are shown in Section 6.4. Finally, Section 6.5 presents the main conclusions drawn from the results shown in Section 6.4

## 6.2 Benchmark model and validation metrics

### Benchmark model

This paper presents an extension, based on RNNs, of the traditional Mack's model. Thus, this approach Mack (1993) and its bootstrap implementation England and Verrall (2006) will be used as benchmark for validating the proposed model.

The main characteristic of this model compared to others based on Chain Ladder is the lack of assumptions about the underlying distribution of the payments. Mack's model assumes that cumulative payments,  $D_{ij}$ , or incurred cost have the following variance and expected value:

$$E[D_{ij}] = \hat{f}_j D_{i,j-1} \quad \text{Var}[D_{ij}] = \hat{\sigma}_j^2 D_{i,j-1} \quad (97)$$

where  $i = (1, 2, \dots, I)$  indicates the accident or underwriting year and  $j = (1, 2, \dots, I)$  the development year. As explained by Mack (1993), the parameters of the previous expressions are calculated as follows:

$$\hat{f}_j = \frac{\sum_{i=1}^{I-j+1} D_{ij}}{\sum_{i=1}^{I-j+1} D_{i,j-1}} \quad \hat{\sigma}_j^2 = \frac{1}{I-j-1} \sum_{i=1}^{I-j+1} D_{i,j-1} \left( \frac{D_{ij}}{D_{i,j-1}} - \hat{f}_j \right)^2 \quad (98)$$

where  $\{\hat{f}_j : j = (2, 3, \dots, I)\}$  and  $\{\hat{\sigma}_j^2 : j = (2, 3, \dots, I)\}$ . The residuals needed for the bootstrap method England and Verrall (2006) are calculated as defined below:

$$\hat{r}_{ij} = \frac{\sqrt{D_{i,j-1}} * \left( \frac{D_{ij}}{D_{i,j-1}} - \hat{f}_j \right)}{\hat{\sigma}_j} \quad (99)$$

To obtain the final residuals, the bias adjustment is added accordingly to the expression suggested by England and Verrall (2006):

$$\hat{r}_{ij} = \sqrt{\frac{N}{N-p}} * \frac{\sqrt{D_{i,j-1}} * \left( \frac{D_{ij}}{D_{i,j-1}} - \hat{f}_j \right)}{\hat{\sigma}_j} \quad (100)$$

where  $N$  is the total number of residuals and  $p$  the number of parameters. Hence, the resampled link ratios are obtained as follows:

$$f_{ij}^B = \hat{f}_j + r_{ij}^B \frac{\hat{\sigma}_j}{\sqrt{D_{i,j-1}}} \quad (101)$$

where  $B$  refers to the number of upper triangles to be simulated and  $r_{ij}^B$  to the residual resampled in the position  $(i, j)$  of the  $B^{th}$  triangle. Taking into consideration  $D_{i,j}$  and the resampled link ratios, a new set of development factors,  $\tilde{f}_j^B$ , is computed. Typically, a zero mean adjustment is applied to the residuals in order to ensure that the mean of the stochastic process is the same as the deterministic Chain Ladder method, which is fully dependent on  $\hat{f}_j$ .

The lower triangle ( $D_{i,j}$  where  $i + j > I + 1$ ) is predicted by combining  $\tilde{f}_j^B$  and the upper triangle ( $D_{i,j}$  where  $i + j \leq I + 1$ ). Then, the process variance is incorporated to the lower triangle by adding the following expression:  $\hat{\sigma}_j r_{ij}^B \sqrt{D_{i,j-1}}$ . In case further details about the bootstrap method are needed, refer to England and Verrall (2006) and Joseph Lo (2011).

Although the methodology proposed in this paper merges RNNs with Mack's model, other two approaches have been selected as benchmark: Staked-ANN Ramos-Pérez et al. (2021b) and Changing Settlement Rate. The first model combines ANNs with Random Forests and Gradient Boosting with regression trees Friedman (2000) to predict general insurance reserves. The stochastic procedure of Stacked-ANN assumes that payments follow a log-normal distribution. On the other hand, Changing Settlement Rate (CSR) is a Bayesian Markov Chain Monte-Carlo model. The prior distributions and the simulation approach are proposed by Meyers (2015).

## Validation metrics

General insurance companies are asked by insurance regulations like Solvency II Directive and Solvency Swiss Test to evaluate their reserve variability. Thus, the accuracy and variability of Mack-Net model (Section 6.3) will be validated and compared with the benchmark model defined in Section 6.2.

To assess the accuracy of the reserve predicted by the models, the following error measure will be computed for every line of business:

$$\%RMSE(U^t) = \sqrt{\frac{1}{K} \sum_{n=1}^K \left( \frac{\hat{U}_n^t - U_n^t}{U_n^t} \right)^2} * 100 \quad (102)$$

$$\%MAE(U^t) = \frac{100}{K} \sum_{n=1}^K \left| \frac{\hat{U}_n^t - U_n^t}{U_n^t} \right| \quad (103)$$

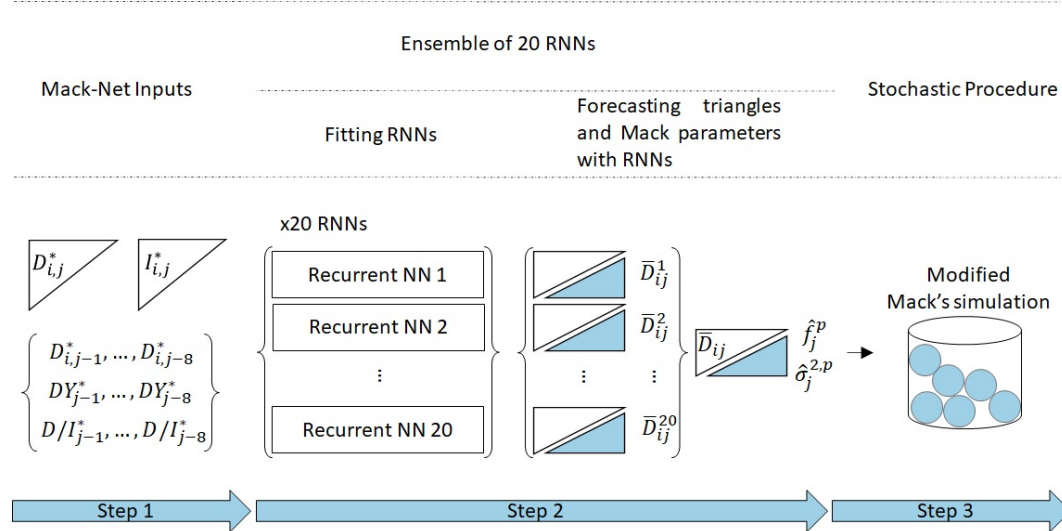
where  $K$  is the total number of companies analysed,  $\hat{U}_n^t$  the ultimate cost predicted by the reserving model for the  $n^{th}$  company and  $U_n^t$  the ultimate cost that was actually observed. The Model Confidence Test Hansen et al. (2011) will be also applied to produce a more robust comparison of models accuracy. This procedure consists on a sequence of tests which permit the identification of the best models at a certain confidence level.

In addition to the previous error measures, the reserve variability produced by the different stochastic reserving models will be assessed by applying the statistical test introduced by Kupiec (1995). The aim of this test is to validate the Value-at-Risk (VaR) by comparing the number of VaR breaches with the percentile selected for calculating the VaR. In this paper, the percentile selected for evaluating the reserve variability is  $\alpha = 0.995$ , which is the level set up by Solvency II to calculate the risk of insurance companies. The empirical results of the test and  $\%RMSE(R^t)$  are collected in Section 6.4.

### 6.3 Data and Mack-Net architecture

The aim of this section is to explain the architecture of the Mack-Net model. To do so, this section has been divided into three different subsections. The inputs of the model are described in the first one, the ensemble of RNNs is explained in the second subsection and, in the last one, the bootstrap method to obtain a reserve distribution is presented. To support the explanation, Figure 24 shows the architecture of the proposed stochastic reserving model.

Figure 24: Mack-Net model architecture



## Data source and model inputs

The starting point of the Mack-Net model is the definition of inputs used within the ensemble of RNNs. Hence, the goal of this subsection is to define the database used, as well as the response and explicative variables for fitting the RNNs ensemble.

The database of Schedule P of the NAIC Annual Statement (available on CAS website) is selected for fitting and validating the Mack-Net model. The paid data, incurred cost and premiums available in the previous database were collected from general insurance companies from the US. The range of accident years contained in the Schedule P database is 1988-1997. As the upper and lower triangles are included, ten development years are available for each accident year.

The benchmark and the proposed model will be fitted to the 200 loss triangles selected by Meyers (2015) from the Schedule P database. This author pointed out that one of the main mistakes that could be made with NAIC Schedule P data is selecting triangles from insurance companies that have experienced significant changes in business operations. Therefore, the net-on-gross premiums ratio and the coefficient of variation of the net premiums were used by Meyers to identify those companies that made changes in their business operations or reinsurance structure. Taking into consideration these indicators, Meyers selected 50 loss triangles from each of the following lines of businesses: Commercial Auto (CA), Private Passenger Auto Liability (PA), Workers' Compensation (WC) and Other Liability (OL). The codes of the companies selected can be found in Meyers (2015).

It is worth mentioning that loss triangles are considered the primary method to organize the observed incurred cost or payments for general insurance reserving purposes. Loss triangles show the total losses of different underwriting or accident years at various valuation dates. This data shows the claim settlement speed, ultimate cost and policyholders' behaviour. Thus, this data is normally not available because insurance entities do not make public this information. Schedule P database is used in the academic field by several authors (such as Leong et al. 2014, Meyers (2015), Kuo (2018) or Ramos-Pérez et al. 2021b) because it offers the possibility of testing triangles from numerous companies and different lines of business.

Before starting with the definition of the explanatory and response variables, it is worth mentioning that the last diagonal of the triangle is selected as a test set for fitting the ensemble of RNNs. Thus, the remaining triangle (9 development years) is used to train the algorithms. The sequences used as explanatory variables,  $X_i$ , and

the response variable,  $Y$ , are the following:

$$Y = C_{ij}^* = \frac{C_{ij}}{P_i} \quad (104)$$

$$X_1 = (C_{ij-1}^*, C_{ij-2}^*, \dots, C_{ij-8}^*) = \left( \frac{C_{ij-1}}{P_i}, \frac{C_{ij-2}}{P_i}, \dots, \frac{C_{ij-8}}{P_i} \right) \quad (105)$$

$$X_2 = (DY_{j-1}^*, DY_{j-2}^*, \dots, DY_{j-8}^*) = \left( \frac{DY_{j-1}}{I}, \frac{DY_{j-2}}{I}, \dots, \frac{DY_{j-8}}{I} \right) \quad (106)$$

$$X_3 = (R_{j-1}^*, \dots, R_{j-8}^*) = \left( \frac{\sum_{i=1}^{I-j+2} D_{ij-1}^*}{\sum_{i=1}^{I-j+2} \frac{IC_{ij-1}}{P_i}}, \dots, \frac{\sum_{i=1}^{I-j+9} D_{ij-8}^*}{\sum_{i=1}^{I-j+9} \frac{IC_{ij-8}}{P_i}} \right) \quad (107)$$

where  $P_i$  is the premium,  $C_{ij}$  is the incremental payment,  $D_{ij}^*$  is equal to  $D_{ij}/P_i$ ,  $I$  the total number of accident years and  $IC_{ij}$  represents the incurred cost of the accident year  $i$  and development  $j$ .  $R_j^*$  is the scaled cumulative payments between the scaled incurred cost. Thus,  $X_1$ ,  $X_2$  and  $X_3$  are the time series used as input in the RNNs to predict the next year payment. Two remarks about the variables need to be made:

1. Payments and development years were scaled.  $DY_j$  were initialized as one and then scaled to the range  $[0, 1]$ . To do so,  $DY_j$  were divided between the total number of accident years,  $I$ . In the case of payments, premiums ( $P_i$ ) play the role of exposure measure. The use of exposure measures is widely used in reserving models, otherwise the changes in the claims settlement speed will be mixed with variations in the business volume.
2. As it can be observed in the formal definition of the explanatory variables, the last 8 observations of each variable are selected. This number was chosen due to the size of the triangles. Ten development years are available but only 9 of them are used during the training of the RNNs. The other development year is used as a test set. Thus, as the aim of the RNNs is to predict the value in  $t$ , the maximum lag available for training the algorithms is  $t - 8$ . Each RNN forecasts the next payment by taking into consideration the information available in the last 8 periods.

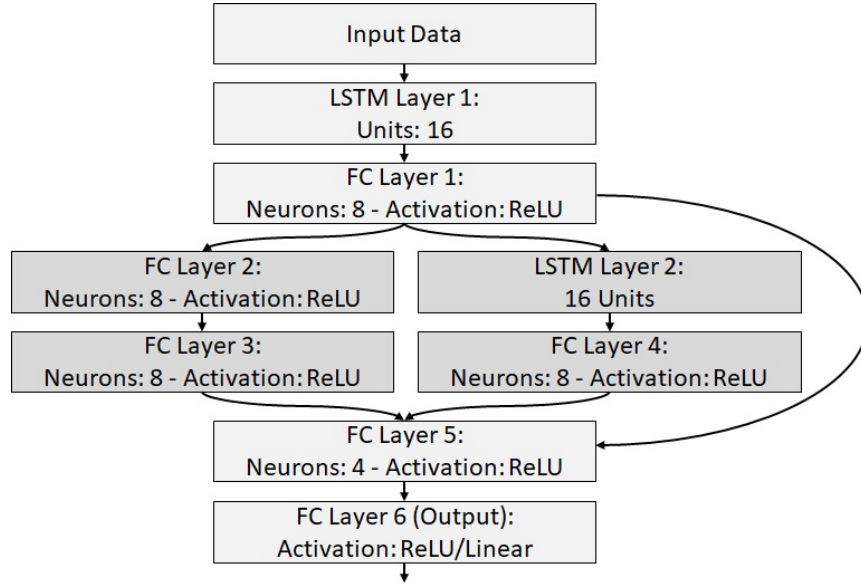
Before concluding this subsection it is worth mentioning that the model can be applied to predict the incurred cost. To do so,  $X_1$  and  $Y$  should be substituted by the following expressions:

$$Y' = i_{ij}^* = \frac{i_{ij}}{P_i} \quad (108)$$

$$X_1' = (IC_{ij-1}^*, IC_{ij-2}^*, \dots, IC_{ij-8}^*) = \left( \frac{IC_{ij-1}}{P_i}, \frac{IC_{ij-2}}{P_i}, \dots, \frac{IC_{ij-8}}{P_i} \right) \quad (109)$$

where  $i_{ij}$  is the incremental incurred cost. The method and calculations that will be explained in Section 6.3 and 6.3 are the same regardless of the variable to be predicted (payments or incurred cost).

Figure 25: Recurrent Neural Network Architecture



## Ensemble of RNNs

As shown in Figure 24, once the model inputs are prepared, 20 Recurrent Neural Networks composed of several Fully Connected (FC) and Long-Short Term Memory (LSTM) layers (Figure 25) are fitted. The number of Recurrent Neural Networks fitted is high enough to obtain the average model prediction regardless their initial weights. This strategy was also applied by Kuo (2018).

It has to be pointed out that a skip connection between ‘FC Layer 1’ and ‘FC Layer 5’ has been included. This type of connection, introduced by He et al. (2016) in the field of image recognition with Convolutional Neural Networks, gives the possibility to skip the training of a part of the Neural Network architecture. During the learning process, the RNN will decide by itself if ‘FC Layer 3’ and/or ‘FC Layer 4’ send information to ‘FC Layer 5’. Apart from giving to the network the possibility to simplify the structure by skipping layers, this kind of connection helps to avoid the problem of vanishing gradients by using the activation of a previous layer until the skipped one learns its weights.

To take temporal dependencies into consideration, the first layer of every RNN is a LSTM cell. This structure was introduced by Hochreiter and Schmidhuber (1997b) for managing time series. Figure 26 and the following expressions define the LSTM

architecture:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (110)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (111)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (112)$$

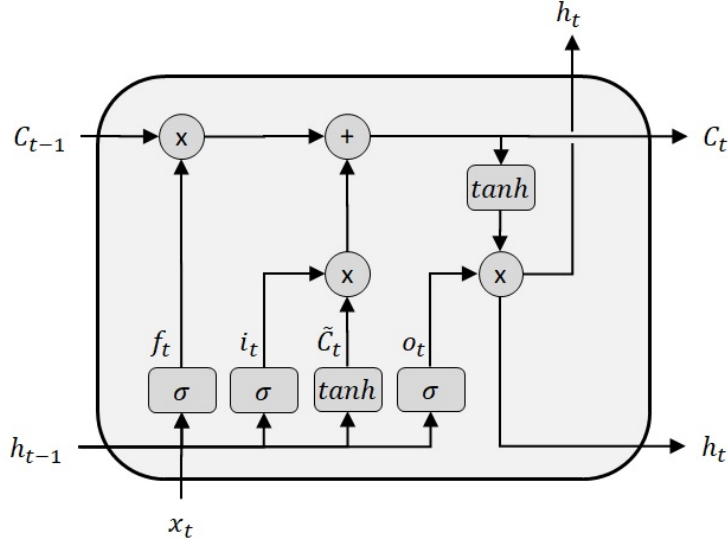
$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \quad (113)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (114)$$

$$h_t = o_t \tanh(C_t) \quad (115)$$

Where  $W_f$ ,  $W_i$ ,  $W_c$ ,  $W_o$ ,  $b_f$ ,  $b_i$ ,  $b_c$  and  $b_o$  represent the weights and bias of the RNNs and  $\sigma(x)$  the logistic sigmoid function.

Figure 26: LSTM structure



The main characteristics of the RNNs are defined below:

- The algorithm used to optimize the weights is Adaptive Moment Estimation (ADAM), which was developed by Kingma and Ba (2014). Considering current and previous gradients, this procedure allows to implement a progressive adaptation of the initial learning rate. These authors suggested the following default values for the ADAM parameters:  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . In this paper, the initial learning rate is set to  $\delta = 0.01$  and the default ADAM parameters are used during the training process.
- The batch size is equal to the number of observations of the training set
- The backward pass calculations are done taking the mean squared error as loss function.

- Each individual algorithm within the ensemble is randomly initialized. Glorot initializer (Glorot and Bengio (2010)) is used for the LSTM weights responsible of transforming linearly the inputs, while the LSTM weights for the linear transformations within the recurrent states are initialized with the orthogonal approach suggested by Saxe et al. (2013).
- In order to avoid overfitting, the level of dropout regularization  $\theta$  Srivastava et al. (2014) is set to 5%.

The training of the RNNs has been implemented using the Keras Chollet et al. (2015) and Tensorflow Abadi et al. (2015). As previously stated, the initial weights of the RNNs are randomly initialized. Thus, the lower triangle predicted by every single algorithm is going to be different. Once the lower triangles are predicted with each RNN, the Mack-Net parameters are computed with the predicted cumulative payments or incurred cost as follows:

$$\hat{f}_j^p = \frac{\sum_{i=I-j+2}^I \bar{D}_{ij}}{\sum_{i=I-j+2}^I \bar{D}_{i,j-1}} \quad (116)$$

$$\hat{\sigma}_j^{2,p} = \frac{1}{I-j-1} \sum_{i=1}^I \bar{D}_{i,j-1} \left( \frac{\bar{D}_{ij}}{\bar{D}_{i,j-1}} - \bar{f}_j \right)^2 \quad (117)$$

where  $\bar{f}_j$  is equal to  $\sum_{i=1}^I \bar{D}_{ij} / \sum_{i=1}^I \bar{D}_{i,j-1}$  and  $\bar{D}_{ij} = \sum_{k=1}^K \bar{D}_{ij}^k / K$ . In addition,  $\bar{D}_{ij}^k$  represents the cumulative payments of the lower triangle ( $i + j > I + 1$ ) predicted by  $k^{th}$  RNN and the observed values of the upper triangle ( $i + j \leq I + 1$ ).  $K$  is the number of RNNs included in the ensemble. Similar to the notation used in Section 6.2,  $\{\hat{f}_j^p : j = (2, 3, \dots, I)\}$  and  $\{\hat{\sigma}_j^{2,p} : j = (2, 3, \dots, I)\}$ .

As it can be derived from the model definition, rather than using the traditional Mack parameters described in Section 6.2, Mack-Net model parameters are estimated by taking into consideration the predictions made by the ensemble of RNNs. Therefore, the central scenario of the stochastic Mack-Net model is equal to the reserve predicted by the ensemble of RNNs, while the mean of the traditional Mack's model converge to the reserve estimated with the deterministic Chain Ladder method. As any other general insurance reserving methodology such as Mack's model, CSR or Stacked-ANN, the approach suggested by this paper is valid until a new diagonal of the loss triangle is available.

## Stochastic procedure

As previously stated, the bootstrap method of Mack-Net is based on the traditional Mack's model. This last methodology has been selected as reference for producing the full reserve distribution due to the two following reasons. First, Mack's model derives the distribution by focusing on the two first moments. In contrast to most of the stochastic reserving models, this approach does not make any assumption about the theoretical distribution followed by incurred cost or cumulative payments. Changes in policyholders' behaviour, reinsurance structures, regulation and number of policy

holders can modify significantly the payments or incurred cost collected by loss triangles. Therefore, a free-distribution model is especially appropriate for insurance companies and regulators because the previous portfolio changes happen quite often in the sector. Second, Mack's model is already applied by insurance companies to comply with regulations such as Solvency II Directive, Swiss Solvency Test or IFRS. Thus, the bootstrap method of Mack-Net is familiar and aligned with the procedures already used by the insurance market.

As no assumption about the underlying distribution of cumulative payments or incurred cost is taken, the expected value and variance of Mack-Net model is defined as follows:

$$E[D_{ij}] = \hat{f}_j^p \bar{D}_{i,j-1} \quad \text{Var}[D_{ij}] = \hat{\sigma}_j^{2,p} \bar{D}_{i,j-1} \quad (118)$$

The Mack-Net model uses the predictions made by the ensemble of RNNs to determine the mean and variance of the reserve distribution. By doing so, the forecasting power of deep and machine learning techniques are taken into consideration. The calculation of residuals needed for the bootstrap method is based on the expression provided by England and Verrall (2006) for Mack's model:

$$\hat{r}_{ij}^p = \frac{\sqrt{\bar{D}_{i,j-1}} * \left( \frac{\bar{D}_{ij}}{\bar{D}_{i,j-1}} - \bar{f}_j \right)}{\hat{\sigma}_j^p} \quad (119)$$

where  $\{\hat{r}_{ij}^p : j = (2, \dots, I); i = (1, \dots, I)\}$ . As in Mack's model, Mack-Net model is bias adjusted in accordance with the procedure suggested by England and Verrall (2006):

$$\hat{r}_{ij}^p = \sqrt{\frac{N}{N-p}} \frac{\sqrt{\bar{D}_{i,j-1}} * \left( \frac{\bar{D}_{ij}}{\bar{D}_{i,j-1}} - \bar{f}_j \right)}{\hat{\sigma}_j^p} \quad (120)$$

As with Mack's model,  $p$  is equal to the number of development factors. Once the set of residuals has been calculated, the resampled upper triangles of link ratios are calculated as follows:

$$f_{ij}^{B,p} = \hat{f}_j^p + r_{ij}^{B,p} \frac{\hat{\sigma}_j^p}{\sqrt{\bar{D}_{i,j-1}}} \quad (121)$$

where  $B$  refers to the number of upper triangles to be simulated and  $r_{ij}^{B,p}$  to the residual resampled in the position  $(i, j)$  of the  $b^{th}$  triangle. Similar to the Mack's model presented in Section 6.2, a zero mean adjustment should be applied to the residuals in order to avoid deviations between the simulated and theoretical mean. The resampled development factors ( $\tilde{f}_j^{B,p}$ ) are

$$\tilde{f}_j^{B,p} = \frac{\sum_{i=1}^{I-j+1} \bar{D}_{i,j-1} f_{ij}^{B,p}}{\sum_{i=1}^{I-j+1} \bar{D}_{i,j-1}} \quad (122)$$

It is worth mentioning that the previous calculation is carried out with the upper triangle ( $i + j \leq I + 1$ ). Then,  $\bar{D}_{i,j-1}$  can be substituted by  $D_{i,j-1}$ . The resampled development factors,  $\tilde{f}_j^{B,p}$ , are used to calculate the lower triangle ( $i + j > I + 1$ ) by applying the Chain Ladder methodology:

$$\hat{D}_{ij} = \bar{D}_{i,j-1} \tilde{f}_j^{B,p} \quad (123)$$

Then, as well as in the case of Mack's model, the process variance is included in the Mack-Net model in order to take into consideration the randomness in future outcomes:

$$\hat{D}_{ij} = \hat{D}_{ij} + \hat{\sigma}_j^p r_{ij}^{B,p} \sqrt{\bar{D}_{i,j-1}} \quad (124)$$

Notice that, as the procedure is applied recursively,  $\bar{D}_{i,j-1}$  is used in the previous equation only when the simulation refers to year after the last diagonal observed. In the rest of the cases, the recurrence is applied and, thus,  $\bar{D}_{i,j-1}$  substituted by  $\hat{D}_{ij-1}$ . As explained in Section 6.2 for Mack's model, this procedure England and Verrall (2006) allows the recognition of the process variance, which is the variability in the forecasts of future payments.

## 6.4 Model fitting and results

In this section, Mack-Net parameters by line of business and the comparison between the performance of the benchmark and the Mack-Net model are presented.

### Fitting of Mack-Net model

This subsection presents Mack-Net parameters by line of business. As stated before, the model has been fitted individually to each of the 200 triangles selected by Meyers (2015) from the Schedule P of the NAIC Annual Statement. For further details about the database refer to Section 6.3.

As explained in Section 6.3, once the ensemble of RNNs has been fitted, the Mack-Net architecture proceeds with the calculation of  $\hat{f}_j^p$  and  $\hat{\sigma}_j^{2,p}$ . Similar to  $\hat{f}_j$  and  $\hat{\sigma}_j^2$  in Mack's model, Mack-Net parameters define the variance and expected value of the cumulative payments or incurred cost. Thus, Tables 21, 22, 23 and 24 present a comparison between the average Mack and Mack-Net parameters by line of business.

Table 21: Average  $\hat{f}_j^p$  and  $\hat{f}_j$  by line of business (Paid data).

	CA		PA		WC		OL	
Dev	Mack		Mack		Mack		Mack	
Year	Mack	Net	Mack	Net	Mack	Net	Mack	Net
1	1.90	1.88	1.77	1.75	2.21	2.59	3.12	3.10
2	1.35	1.35	1.22	1.22	1.29	1.38	1.72	1.67
3	1.16	1.15	1.10	1.10	1.13	1.15	1.34	1.30
4	1.08	1.06	1.06	1.05	1.07	1.08	1.18	1.16
5	1.04	1.03	1.03	1.02	1.04	1.04	1.11	1.07
6	1.02	1.01	1.01	1.01	1.02	1.02	1.04	1.03
7	1.00	1.01	1.01	1.00	1.02	1.02	1.02	1.01
8	1.01	1.00	1.00	1.00	1.01	1.01	1.02	1.01
9	1.00	1.00	1.00	1.00	1.01	1.01	1.01	1.00

*Source:* own elaboration

Table 22: Average  $\hat{f}_j^p$  and  $\hat{f}_j$  by line of business (Incurred data).

	CA		PA		WC		OL	
Dev	Mack		Mack		Mack		Mack	
Year	Mack	Net	Mack	Net	Mack	Net	Mack	Net
1	1.27	1.33	1.14	1.15	1.31	1.40	1.45	1.56
2	1.09	1.09	1.03	1.03	1.07	1.09	1.21	1.20
3	1.03	1.03	1.01	1.01	1.02	1.03	1.07	1.07
4	1.01	1.01	1.00	1.00	1.01	1.01	1.03	1.03
5	1.00	1.01	1.00	1.00	1.00	1.01	1.04	1.01
6	0.99	1.01	1.00	1.00	1.00	1.01	1.01	1.02
7	1.00	1.01	1.00	1.00	1.00	1.01	1.00	1.01
8	1.00	1.01	1.00	1.00	1.00	1.01	1.01	1.01
9	1.00	1.01	1.00	1.00	1.00	1.01	1.00	1.01

*Source:* own elaboration

The averages of the development factors of the payment models (Table 21) present non-significant differences in most of the cases. The only remarkable differences are the second development factor of OL, and the first and second parameters of WC. Mack-Net development factors are lower than Mack's parameters in all the lines of business with the only exception of Workers' Compensation. This can be proved by computing the multiplicative development factors (product of development factors by line of business and model).

With regard to the incurred cost development factors (Table 22), the differences between both models are minor in the case of CA and PA. However, they become more relevant in the case of WC and OL, especially in the case of the first development year. It is also worth mentioning that Mack-Net development factors are higher than

Mack parameters regardless of the line of business.

Table 23: Average  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$  by line of business (Paid data).

Dev Year	CA		PA		WC		OL	
	Mack	Net	Mack	Net	Mack	Net	Mack	Net
1	13.25	12.43	12.96	11.96	14.98	13.94	26.94	25.47
2	6.82	5.96	5.73	5.09	6.06	5.43	10.35	10.21
3	4.43	3.76	3.61	3.29	4.07	3.70	6.67	5.58
4	2.97	2.31	2.59	2.55	3.07	2.78	4.96	4.29
5	1.76	1.39	1.40	1.47	1.55	1.76	3.21	2.30
6	0.98	0.74	0.87	0.79	1.46	1.19	1.95	1.23
7	0.65	0.44	0.68	0.50	1.10	0.86	0.97	0.54
8	0.42	0.26	0.19	0.24	0.71	0.55	0.78	0.42
9	0.32	0.17	0.15	0.15	0.53	0.36	0.44	0.18

*Source:* own elaboration

Table 24: Average  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$  by line of business (Incurred data).

Dev Year	CA		PA		WC		OL	
	Mack	Net	Mack	Net	Mack	Net	Mack	Net
1	7.95	7.29	10.11	9.23	15.13	13.85	13.69	12.67
2	4.67	4.11	4.79	4.35	9.48	8.23	9.00	8.56
3	3.19	2.71	3.06	3.04	4.09	3.50	5.24	4.40
4	2.29	1.96	1.62	1.57	2.69	2.19	4.01	3.58
5	1.54	1.27	1.18	1.10	2.12	1.64	3.66	2.53
6	1.13	0.97	0.78	0.71	1.79	1.34	1.50	1.18
7	0.77	0.68	0.31	0.38	1.30	0.94	1.15	0.78
8	0.41	0.46	0.18	0.26	0.80	0.64	0.52	0.48
9	0.33	0.38	0.15	0.18	0.49	0.41	0.43	0.34

*Source:* own elaboration

Before analysing the differences between  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$ , it is worth mentioning that previous cumulative payments or incurred cost play a key role in the model variance, defined in equations 2 and 20 for Mack and Mack-Net model respectively. Thus,  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$  have to be analysed by taking into consideration the analysis of the development factors explained in the previous paragraphs.

With regard to the models for paid loss data,  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$  show non-material differences. Nevertheless, it has to be pointed out that Mack parameters are higher than those of Mack-Net in every line of business. As the paid development factors of the

Mack model are also higher, the pattern shown in Table 23 reveals that Mack model generates a higher volatility than the proposed methodology.

Table 24 shows that Mack's parameters are higher than those of the Mack-Net model fitted with incurred cost data. In contrast to the models for paid loss data, this effect is partially offset by the Mack-Net development factors that, as shown in Table 22, are higher than those of the Mack's model.

As incurred cost includes the payments and the reserve set up by claim adjusters, this variable should be closer to the ultimate claim cost than the payments. Thus, development factors and reserve volatility should be lower in the case of the models fitted with incurred cost data. The comparison of the average parameters of the models for incurred (Table 22 and 24) and paid loss data (Table 21 and 23) reveals that this trend is followed by both models.

## Comparison against benchmark models

This subsection compares the performance of the Mack-Net model with the original methodology proposed by Mack. The variability and accuracy will be compared with the metrics and tests shown in Section 6.2.

As previously explained, the aim of the Mack-Net model is to improve the accuracy of the traditional Mack's methodology by using machine and deep learning algorithms and techniques such as RNNs. Table 25 and 26 show the empirical results of the metrics selected for comparing the models accuracy.

Table 25:  $\%RMSE(U^t)$  by model and line of business

Line of business	Mack Paid	Mack-Net Paid	Mack Incurred	Mack-Net Incurred	CSR	Stacked ANN
CA	7.98%	6.80%	8.18%	8.03%	9.29%	8.92%
PA	6.06%	5.01%	2.62%	4.26%	5.46%	7.78%
WC	7.86%	6.77%	8.15%	6.99%	13.29%	7.36%
OL	20.20%	17.31%	17.38%	13.48%	27.78%	19.80%

*Source:* own elaboration

Table 26:  $\%MAE(U^t)$  by model and line of business

Line of business	Mack Paid	Mack-Net Paid	Mack Incurred	Mack-Net Incurred	CSR	Stacked ANN
CA	5.96%	4.78%	5.46%	5.40%	6.35%	6.86%
PA	3.81%	3.44%	1.90%	2.86%	3.68%	3.76%
WC	5.32%	4.60%	5.27%	4.51%	6.01%	4.65%
OL	13.41%	12.16%	11.34%	9.88%	18.29%	13.47%

*Source:* own elaboration

With regard to the models for paid loss data, Mack-Net methodology improves the accuracy of the Mack's model in every line of business.  $\%RMSE(U^t)$  decreases by 14% in WC and OL, 15% in CA, and 17% in the case of PA. Similar improvements are also observed in terms of  $\%MAE(U^t)$ .

The comparison of the  $\%RMSE(U^t)$  and  $\%MAE(U^t)$  obtained from the models for incurred loss data shows that Mack-Net model outperforms the Mack's procedure in all the lines of business with the only exception of PA. It is worth mentioning that the accuracy of the Mack-Net model is especially higher in OL, which is the line of business with the longer duration of liabilities. Thus, an appropriate estimation of reserves is particularly relevant in this case. Empirical results demonstrate that Mack-Net model also outperforms general insurance reserving approaches based on Markov Chain Monte-Carlo or machine learning such as CSR or Stacked-ANN.

Tables 27 and 28 show the ranking proposed by Model Confidence Set (MCS) considering  $\%RMSE(U^t)$  and  $\%MAE(U^t)$  as loss functions. This approach confirms the outcomes presented in the previous paragraphs. In fact, Mack-Net Incurred and Paid are ranked as the best and second best model respectively when all the lines of business are considered together (row 'Total').

Table 27: Ranking of models according to MSC ( $\%RMSE(U^t)$  and  $\alpha = 0.05$ ).

Line of business	Mack Paid	Mack-Net Paid	Mack Incurred	Mack-Net Incurred	CSR	Stacked ANN
CA	2 <sup>nd</sup>	1 <sup>st</sup>	4 <sup>th</sup>	3 <sup>rd</sup>	5 <sup>th</sup>	6 <sup>th</sup>
PA	6 <sup>th</sup>	3 <sup>rd</sup>	1 <sup>st</sup>	2 <sup>nd</sup>	4 <sup>th</sup>	5 <sup>th</sup>
WC	6 <sup>th</sup>	1 <sup>st</sup>	5 <sup>th</sup>	2 <sup>nd</sup>	4 <sup>th</sup>	3 <sup>rd</sup>
OL	5 <sup>th</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	1 <sup>st</sup>	6 <sup>th</sup>	4 <sup>th</sup>
Total	5 <sup>th</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	1 <sup>st</sup>	6 <sup>th</sup>	4 <sup>th</sup>

*Source:* own elaboration

Table 28: Ranking of models according to MSC ( $\%MAE(U^t)$  and  $\alpha = 0.05$ ).

Line of business	Mack Paid	Mack-Net Paid	Mack Incurred	Mack-Net Incurred	CSR	Stacked ANN
CA	5 <sup>th</sup>	1 <sup>st</sup>	3 <sup>rd</sup>	2 <sup>nd</sup>	4 <sup>th</sup>	6 <sup>th</sup>
PA	6 <sup>th</sup>	3 <sup>rd</sup>	1 <sup>st</sup>	2 <sup>nd</sup>	5 <sup>th</sup>	4 <sup>th</sup>
WC	6 <sup>th</sup>	3 <sup>rd</sup>	5 <sup>th</sup>	1 <sup>st</sup>	4 <sup>th</sup>	2 <sup>nd</sup>
OL	5 <sup>th</sup>	3 <sup>rd</sup>	2 <sup>rd</sup>	1 <sup>st</sup>	6 <sup>th</sup>	4 <sup>th</sup>
Total	5 <sup>th</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	1 <sup>st</sup>	6 <sup>th</sup>	4 <sup>th</sup>

*Source:* own elaboration

With regard to the validation of the reserves variability, Kupiec test Kupiec (1995) is applied to assess the appropriateness of the reserve distribution generated by the stochastic process. Table 29 collects the p-values of the Kupiec test assuming a VaR percentile of  $\alpha = 0.995$ , which is the value for evaluating the risk profile of insurance companies under Solvency II Directive.

Companies included in each line of business have different volumes. This fact was taken into consideration within the Kupiec test by giving different weights to each company. The higher the standard deviation generated by the company, the higher the weight given to compute the Kupiec test. Table 29 collects the p-values by model and line of business.

Table 29: Kupiec test (p-values) by model and line of business

Line of business	Mack Paid	Mack-Net Paid	Mack Incurred	Mack-Net Incurred
CA	$\geq 0.05$	$\geq 0.05$	$< 0.05$	$\geq 0.05$
PA	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$
WC	$< 0.05$	$< 0.05$	$< 0.05$	$\geq 0.05$
OL	$\geq 0.05$	$\geq 0.05$	$< 0.05$	$\geq 0.05$

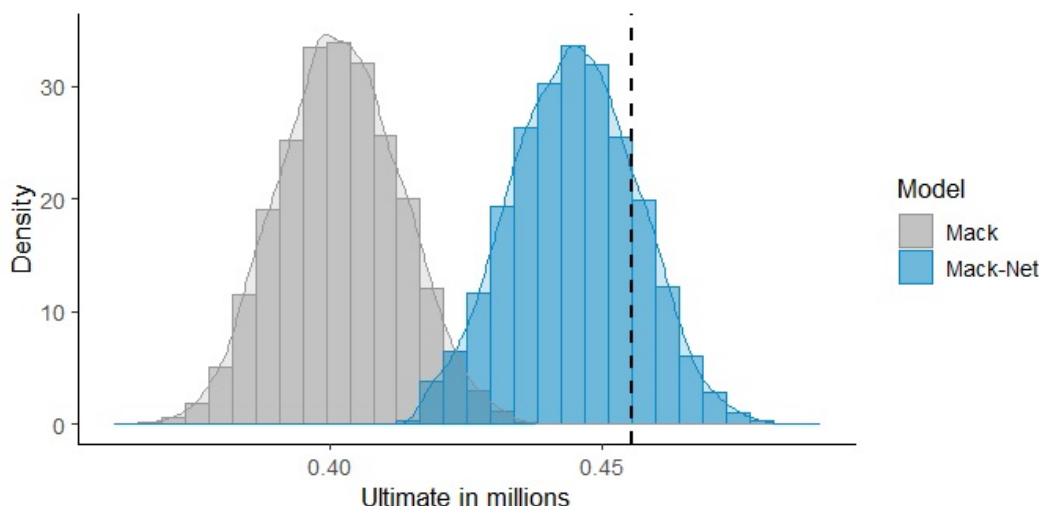
*Source:* own elaboration

According to the results of the models for paid loss data, Mack and Mack-Net methodologies are unable to produce an appropriate Value at Risk (VaR) for Workers Compensation. In the rest of lines of business, the excesses of the VaR estimated by both models are aligned with the confidence level selected ( $\alpha = 0.995$ ). It is worth mentioning that, as discussed in Section 6.2, Mack-Net parameters reveal a lower level of variance than those of the Mack's model. Thus, the higher accuracy of the Mack-Net model for paid loss data (Table 25) allows to generate appropriate risk measures with a lower level of variability.

In the case of the models for incurred cost, Mack-Net model passes the test in all the lines of business, while Mack's model fails the test in three out of four lines of

business. As it will be presented in Table 30, the coefficients of variation generated by the Mack-Net model are lower than those of the traditional Mack's methodology. Nevertheless, Mack-Net model passes the test because the accuracy of the mean of the stochastic process is higher (Table 25).

Figure 27: Company code 620. Other Liability.



Mack's model fails the test in most of the lines of business due to two reasons. First, the lower level of accuracy leads to higher differences between the actual reserve and the distribution function generated by the model. Second, the model is unable to generate a higher variance in order to offset the lack of accuracy. To illustrate this, Figure 27 shows one company of the OL segment where Mack's model produces an inappropriate VaR. The observed ultimate is represented by a black dashed line.

With the goal of comparing the volatility generated by the different stochastic reserving models, Table 30 collects the % of companies where the coefficient of variation,  $CV(U^t)$ , of Mack-Net is lower than in the case of Mack's model:

Table 30: % of companies where Mack-Net  $CV(U^t) < \text{Mack } CV(U^t)$

Line of business business	Paid loss data	Incurred loss data
CA	56%	72%
PA	46%	66%
WC	56%	54%
OL	58%	68%

*Source:* own elaboration

The results shown in tables 29 and 30 demonstrate that Mack-Net model does not need to produce higher coefficients of variation in order to generate more appropriate risk measures than Mack’s model. Thus, the proposed methodology produces more efficient risk measures thanks to its predictive power.

Finally, on the trade-off between time and accuracy, Table 31 presents how training time and error change when the input size increases. The database used for fitting the algorithms (Schedule P of the NAIC Annual Statement) contains a maximum range of 10 years per each general insurance company. This analysis sets 5 years of data as the initial scenario and, then, the input size is increased until the maximum available data is reached. The results show that the error decreases in a higher extent than training time in relative terms. This is true even when the number of years is close the to maximum (10 years). It is worth noticing that benchmark models accuracy will also decrease if the input size is reduced. Therefore, Table 31 shows the trade-off between time and accuracy of Mack-Net model in a standalone basis.

Table 31: Mack-Net: Training time and error versus input size

	5 Years	6 Years	7 Years	8 Years	9 Years	10 Years
Time Index	100.00	102.14	105.48	113.86	126.00	140.55
Error Index	100.00	70.73	50.05	33.73	19.02	7.08

*Source:* own elaboration

## 6.5 Conclusions

The Mack-Net model introduced in this paper has the aim of blending the traditional Mack’s reserving model with deep and machine learning techniques. To do so, an ensemble of RNNs is fitted to the loss triangle. Then, the predictions of this ensemble are used for calculating Mack’s model parameters. In this paper, the predictive power and reserve variability of the proposed architecture and the traditional Mack’s methodology are compared. Models were fitted to 200 incurred cost and paid loss triangles from NAIC Schedule P database (available on CAS website) in order to generate a robust comparison.

Three main conclusions are drawn from the results presented in Section 6.4. First, the comparison of the accuracy reveals that adding deep learning techniques to Mack’s model improves the predictive power. With regard to the models fitted with paid data, this paper demonstrates that Mack-Net model outperforms Mack’s model in every line of business. In the case of the models for incurred cost, Mack-Net is also more accurate than Mack’s model in all the lines of business with the only exception of Personal Auto (PA). The accuracy of reserving models is particularly relevant for long-tail lines of business such as Other Liability (OL). In the case of this last portfolio, Mack-Net methodology reduces the RMSE by 14% and 22% when using paid and incurred cost data respectively. Empirical results demonstrate that Mack-Net model

also outperforms other reserving approaches based on Markov Chain Monte-Carlo or machine learning such as Changing Settlement Rate or Stacked-ANN respectively.

Second, Kupiec test demonstrated that Mack-Net model generates more appropriate risk measures (Value at Risk) than the traditional Mack's methodology. With regard to the paid data, both models fail the test for Workers Compensation (WC). However, in the case of the incurred data, Mack-Net model passes the test in every line of business, while Mack's model fails the test in three out of four lines. Thus, Mack-Net model is not only more accurate but also generates a more appropriate Value at Risk (VaR). The confidence level selected for the VaR is  $\alpha = 0.995$ , which is the level required by Solvency II to evaluate the reserving risk in insurance companies.

Third, the blending of traditional approaches with deep learning techniques generates more efficient models for evaluating the reserving risk, which is the potential cost of deviations from the expected reserve. In other risks such as changes in equity price, the mean of the distribution does not play a relevant role (the mean of the returns are almost always close to zero). As the expected reserve can not be easily predicted, this is not the case for reserving risk, where the appropriateness of the risk measures strongly depends on both the mean and the variance of the reserving model.

Due to the reasons explained in the previous paragraph, Mack-Net model is able to generate more appropriate risk measures with a lower variance. Thus, empirical results suggest that the proposed method is more efficient (in terms of risk assessment) than Mack's model because it generates a more reliable VaR with a lower variability.

Taking into consideration the previous conclusions, the blending of deep learning techniques with reserving models can be extended to improve the accuracy and risk measures derived from the use of other bootstrapping and Bayesian approaches. In the specific case of the Bayesian reserving models, deep learning algorithms could be applied in order to estimate the parameters of the distributions.

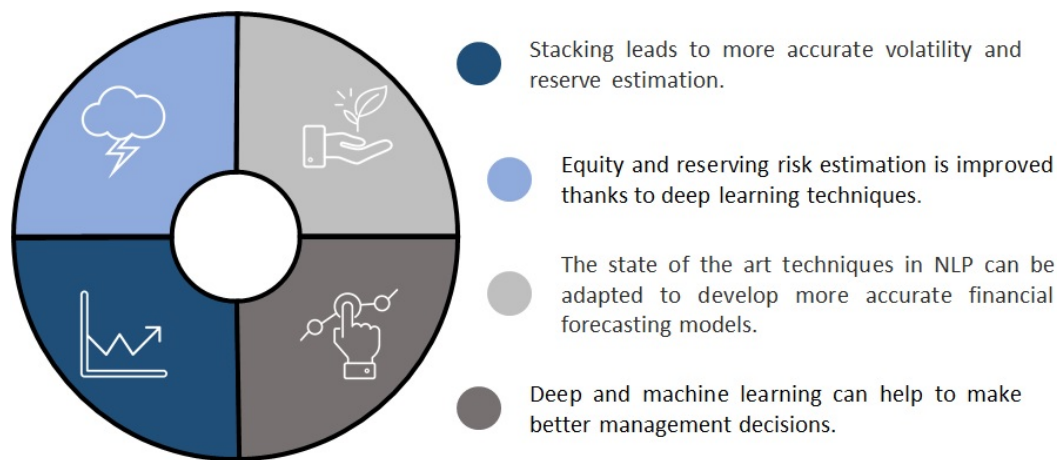
## 7 Conclusions

In this section, the main findings drawn from the proposed stock volatility and general insurance reserving models and the future lines of research are presented.

### 7.1 Main Findings

Figure 28 shows the main conclusions of this thesis. They will be discussed in the following paragraphs.

Figure 28: Thesis conclusions.



*Source:* Own elaboration

#### **Stacking leads to more accurate volatility and reserve estimation.**

As demonstrated in sections 3 and 5, stacking provides more accurate volatility forecasts than traditional stock volatility models such as GARCH, EGARCH or Heston model. This applies equally to the general insurance reserve estimation. The empirical results provided in sections 4 and 6 show that stacking also leads to more accurate estimations than general insurance reserving models based on Chain Ladder (Mack and Overdispersed Poisson) and Markov Chain Monte-Carlo (Changing Settlement Rate).

Stacking is applied in all the different models proposed by this thesis. This technique consists in training an algorithm to combine the predictions of other algorithms and/or models fitted with the training data. In the stock volatility and general insurance reserving models proposed in this thesis, Artificial Neural Network (ANN) is the algorithm responsible of combining the rest of the models. This algorithm is composed of layers and they can be modified to deal with different types of problems such as computer vision, natural language processing, financial time series forecasting, fraud detection or autonomous driving, among others. In addition, the structure

of an ANNs can be easily adapted by changing the activation function, number of layers, types of connection, regularization approach, etc. The flexibility of ANNs make them especially appropriate for playing the role of combiner. Notice that models based on stacking are usually called ‘hybrid’ in the field of stock volatility forecasting.

Section 5 introduces hybrid models based on Transformer and Multi-Transformer layers. As Transformer layers were developed for natural language processing purposes, their architecture is modified in order to be applied in the field of stock volatility. Multi-Transformer layers are an extension of Transformer and they are also introduced by this thesis. As explained in this section, this layer has the aim of improving the stability of Transformer layers by applying bagging to its attention mechanism.

Empirical results demonstrated that hybrid models based of Transformer and Multi-Transformer layers outperform traditional autoregressive algorithms and hybrid models based on feed-forward layers. The difference in the performance is more relevant in 2020, when volatility was specially higher than previous years due to Covid-19 pandemic. The higher shocks on the solvency position of financial institutions take place in high volatility regimes. Thus, the accuracy of stock volatility models is particularly important in years such as 2007, 2008 and 2020.

The existing hybrid models in the field of stock volatility forecasting merge the output of GARCH-based algorithms with feed forward layers or LSTM units. In contrast to these approaches, the hybrid model introduced in Section 3 is only composed of machine learning algorithms that take no assumption about the theoretical distribution of stock returns. In this case, the outputs of random forest, support vector machine and gradient boosting are merged with a feed forward neural network. Most of the GARCH-based algorithms assume that returns follow a conditional normal or Student-t distribution. The empirical results shown in Section 3 demonstrate that the proposed methodology outperforms other existing hybrid approaches, traditional GARCH algorithms and Heston model in high and low volatility regimes.

With regard to general insurance reserving, the model introduced in Section 4 combines the outputs from ANN, random forest, gradient boosting, Chain Ladder and Changing Settlement Rate (CSR) with feed forward layers, whereas the model proposed in Section 6 applies LSTM cells to improve the performance of traditional Mack’s model. As with the stock volatility models, empirical results show that stacking can improve the performance of models based on Chain Ladder (Mack and ODP) or Markov Chain Monte-Carlo (CSR).

The expectations about market volatility and general insurance reserve play a key role in the management decisions taken by financial institutions. For instance, the asset mix of the company can be rebalanced if a high volatility regime is foreseen by the models. Thus, more precise models can lead to more accurate management decisions. The Financial Crisis of 2007-2008 demonstrated that an appropriate risk management strategy also leads to significant competitive advantages in the financial

sector.

It is also worth mentioning that models can estimate precisely a certain variable but they might also ignore significant changes in the financial environment. Therefore, the human supervision plays a key role to update and adapt the models to new economic environments.

### **Equity and reserving risk estimation is improved thanks to deep learning techniques.**

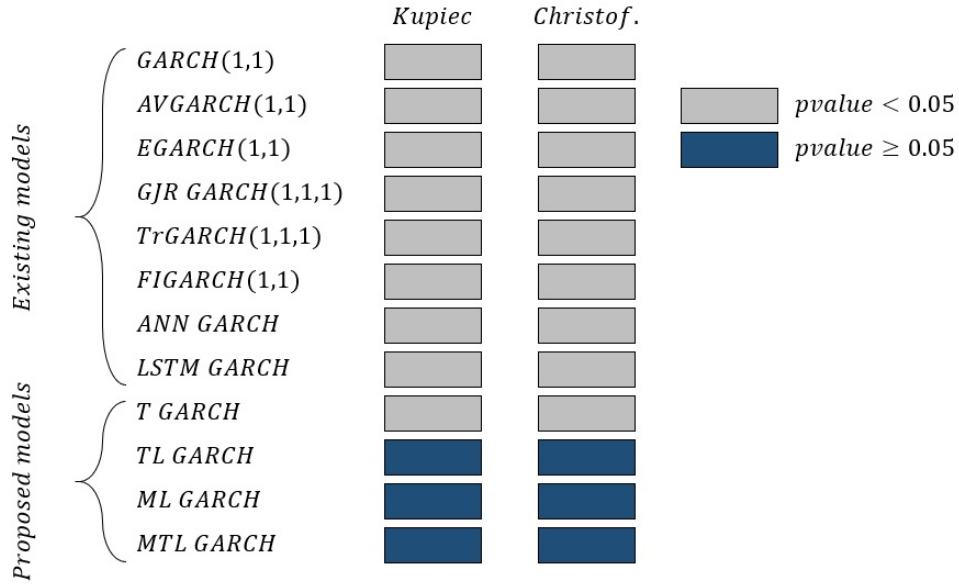
As shown in sections 3, 4, 5 and 6, the stock volatility and general insurance reserving models introduced by this thesis lead to more appropriate equity and reserving risk assessment than the existing models in these fields. After the Financial Crisis of 2007-2008, metrics that relate risk and profit have a significant impact on the investor decisions, portfolio management and dividend payments, among others. Therefore, the relevance of having accurate risk models have increased remarkably in the last years.

The empirical results shown in sections 3 and 5 demonstrate that deep and machine learning techniques can be applied to obtain more appropriate equity risk measures. Figure 29 presents the validation of the equity risk measures produced by the proposed and benchmark models of Section 5. As it can be observed, the benchmark models do not produce appropriate risk measures for the period 2016-2020. Nevertheless, three out of four models based on Transformer and Multi-Transformer layers generate appropriate equity risk measures. Notice that the period of analysis includes the considerable turmoil provoked by Covid-19 pandemic in the stock markets.

With regard to general insurance reserving, Kupiec test also demonstrated that the proposed models generate more appropriate reserving risk estimations than other approaches based on Chain Ladder and Markov Chain Monte-Carlo. As liability duration and product characteristics play a key role in general insurance reserving, four different lines of business from 50 companies from the United States have been analysed: Workers Compensation, Commercial Auto, Other Liability and Personal Auto. The reserving models introduced by this thesis shown a better performance regardless the portfolio characteristics and duration.

As previously stated, measures that relate profit with risk have a significant impact on the market value of financial institutions. An accurate estimation of the risk profile can lead to a more efficient risk strategy, better portfolio management actions and, therefore, an optimization of the profit-risk ratio and other risk related measures such as Solvency Ratio. Considering the increasing importance given by investors to these ratios in the financial sector, non-appropriate risk models can lead to a significant competitive disadvantage as it was demonstrated during the Financial Crisis of 2007-2008, when numerous financial institutions went bankrupt or were sold at a sale price.

Figure 29: Test results. Period: 2016-2020. S&P Index



Source: Own elaboration

**The state of the art techniques in NLP can be adapted to develop more accurate financial forecasting models.**

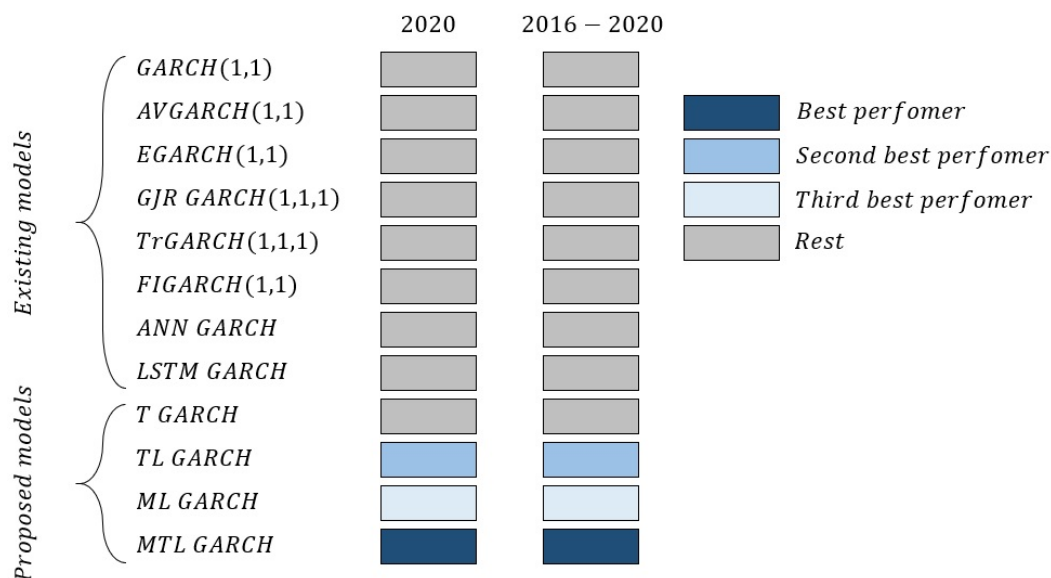
Machine and deep learning have provoked major developments in the field of speech recognition, autonomous driving, computer vision, on-line marketing and natural language processing (NLP), among others. Nevertheless, most of financial forecasting models are based on traditional algorithms or methodologies. As stated in the thesis objectives, the proposed models have the aim of applying the latest deep and machine learning techniques to financial models. Notice that the models introduced by this thesis do not only apply novel machine learning techniques, but they also include in their architectures other traditional approaches in the field of stock volatility forecasting and general insurance reserving.

Transformer layers have overcome the performance of any other algorithm in the field of NLP. Models based on these layers such as BERT, GPT-3, XLNet or Megatron-LM have brought a remarkable progress in this area. As stock volatility forecasting and NLP have substantive differences, Section 5 modifies Transformer layers in order to be applied in the field of stock volatility. In addition, an extension of this layer called Multi-Transformer is also introduced by this section. This novel layer has the aim of improving the accuracy and robustness of Transformer layers by applying bagging to the attention mechanism.

The empirical results demonstrated that bringing the latest deep learning techniques can add value to the current stock volatility forecasting models. Hybrid models

based on Transformer and Multi-Transformer were more accurate than other traditional stock volatility models such as GARCH or other hybrid models based on feed forward layers. The difference in the performance is especially relevant in 2020, when Covid-19 pandemic provoked a remarkable shock in the stock markets.

Figure 30: Performance by model (RMSE). S&P Index



Source: Own ellaboration

As shown in Figure 30, models based on Transformer and Multi-Transformer layers are the top-3 performers during the period 2016-2020. As the higher shocks on the solvency position of financial institutions take place in high volatility regimes, Figure 30 also shows 2020, when Covid-19 pandemic provoked a remarkable shock in the financial markets. It is also worth mentioning that the risk measures based on these top-3 models also pass the Kupiec and Christoffersen tests (see Figure 29).

### Deep and machine learning can help to make better management decisions.

As it was mentioned in the previous points, a more accurate estimation of the stock volatility, general insurance reserve, equity risk and reserving risk can lead to a more appropriate risk management strategy. The empirical results shown in this thesis demonstrate that the proposed models generate more accurate estimations thanks to deep and machine learning.

Models do not take decisions, they just provide useful data for the decision making process. Therefore, financial institutions should not be only focused on having accurate risk models, but they should also take into consideration the risk modelling

results in the decision making process.

After the Financial Crisis of 2007-2008, regulators have enhance the risk management framework of financial institutions with laws such as Basel III, Solvency II or Swiss Solvency Test. In addition, companies have embedded a stronger risk management framework in their regular management process and they have integrated risk assessment and monitoring activities in their internal control systems thanks to lessons learned during this crisis.

Therefore, risk models are specially relevant in the current context due to their importance in the decision making process of financial institutions. Indeed, measures such as Solvency Ratio or Return on Risk Capital have a significant impact on dividends and companies valuation. Thus, the models introduced by this thesis can lead to competitive advantages thanks to the accuracy provided by deep and machine learning algorithms.

It is also worth mentioning that big tech companies such as Alibaba, Alphabet, Amazon or Tesla are starting to sell insurance and financial products. Deep and machine learning algorithms are deeply embedded in the decision making process of these companies. Therefore, financial institutions need to adopt these techniques in order to cancel out this potential competitive advantage of big techs.

## **7.2 Further research**

The empirical results obtained in this thesis suggest three additional research areas. First, volatility plays a key role in financial derivatives pricing models. Thus, non-accurate stock volatility estimations lead to wrong valuations from these models. The architectures introduced by this thesis for stock volatility forecasting purposes can be used together with derivative pricing models in order to analyse if a more accurate derivative valuation can be obtained. Thanks to the flexibility given by derivatives, the size of their market has sharply increased in the last 25 years. Therefore, a correct valuation of this instrument is specially relevant due to the potential impact that they can have on the profit and solvency position of financial institutions.

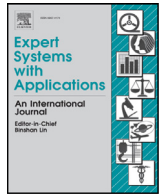
Second, the state of the art of other areas different from natural language processing can be adapted in order to generate novel financial models. This thesis brings Transformer layers and staking to the field of stock volatility forecasting and general insurance reserving. Nevertheless, other deep learning techniques such as generative adversarial networks or convolutional neural networks can be adapted in order to be applied in the field of quantitative finance. As Transformer layers in natural language processing, generative adversarial networks and convolutional neural networks are the state of the art in the field of computer vision and image generation. Their ability to recognise interdependencies can be exploited for financial purposes, where the correlation between different features play a key role. In addition to these algorithms, extreme gradient boosting can also be applied in the field of quantitative

finance. This algorithm has demonstrated a great performance in classification and regression problems with structured data. Notice that most of the finance data is structured.

Third, the robustness and predictive power of the architectures proposed for general insurance reserving purposes suggest that further investigation should be conducted to blend deep and machine learning techniques with Bayesian reserving models based on Markov Chain Montecarlo. The general insurance models introduced by this thesis blend Chain Ladder or theoretical distributions with deep and machine learning algorithms.

## 8 Annexes

## **8.1 Annex I. Published Paper. Forecasting volatility with a stacked model based on a hybridized Artificial Neural Network**



# Forecasting volatility with a stacked model based on a hybridized Artificial Neural Network

Eduardo Ramos-Pérez<sup>1</sup>, Pablo J. Alonso-González\*, José Javier Núñez-Velázquez

Economics Department, Universidad de Alcalá, Plaza de la Victoria 2, Alcalá de Henares 28802, Spain

## ARTICLE INFO

### Article history:

Received 30 October 2018

Revised 26 March 2019

Accepted 27 March 2019

Available online 27 March 2019

### MSC:

62-07

62P05

65C60

90-08

### Keywords:

Machine learning

Stacking algorithms

Risk assessment

Volatility forecasting

Hybrid models

## ABSTRACT

An appropriate calibration and forecasting of volatility and market risk are some of the main challenges faced by companies that have to manage the uncertainty inherent to their investments or funding operations such as banks, pension funds or insurance companies. This has become even more evident after the 2007–2008 Financial Crisis, when the forecasting models assessing the market risk and volatility failed. Since then, a significant number of theoretical developments and methodologies have appeared to improve the accuracy of the volatility forecasts and market risk assessments. Following this line of thinking, this paper introduces a model based on using a set of Machine Learning techniques, such as Gradient Descent Boosting, Random Forest, Support Vector Machine and Artificial Neural Network, where those algorithms are stacked to predict S&P500 volatility. The results suggest that our construction outperforms other habitual models on the ability to forecast the level of volatility, leading to a more accurate assessment of the market risk.

© 2019 Elsevier Ltd. All rights reserved.

## 1. Introduction

During the Financial Crisis of 2007–2008, unexpected falls in stock prices resulted in significant losses for individual investors and financial institutions. Since then, new regulations have entered in force in order to ensure the correctness of the market risk assessment provided by financial institutions and to allow individual market participants to be aware of the risk linked to financial products. As volatility is an indicator of the uncertainty associated with the asset profitability (Hull, 2015; Rajashree & Ranjeeeta, 2015), this variable tends to play a key role within the risk models. In fact, events like the bankruptcy of LTCM in 1998 (Lowenstein, 2000), the dotcom crash in 2001 (Aharon, Gaviols, & Yosef, 2010) or, more recently, the aforementioned Financial Crisis of 2007–2008 were not foreseen by most of the risk models due to inaccurate estimates produced by the volatility forecasting models. It is worth mentioning that, as volatility is not directly observed, before estimating any statistical model it is necessary to select a volatility proxy (Poon & Granger, 2003). In the following

paragraphs, the proposed methodology and main families of volatility forecasting models (GARCH, Stochastic and Machine Learning) are presented.

First of all, GARCH models are introduced as this family of models is probably the most widely used in the literature due to its ability to fit the volatility clustering (Mandelbrot, 1963) empirically observed in financial time series. This auto-regressive approach and its generalization were developed by Engle (1982) and Bollerslev (1986) respectively. Classical GARCH models were discovered to be too rigid for fitting returns series, especially over a long time span, because the estimated persistence of conditional variances is close to one (Bauwens, Hafner, & Laurent, 2012). Therefore, more flexible GARCH models were developed in order to overcome this problem. Engle and Lee (1999) suggested a two equation model where each of them represents long-run and short-run components of volatility, respectively. Mixed-normal GARCH (Haas, Mittnik, & Paolella, 2004a) is a second way to deal with this problem. This kind of model allows to choose amongst several regimes in each instant of time  $t$ . The drawback of this methodology is that it assumes that the variables used to decide amongst regimes are all independent over time. To overcome this problem, Haas, Mittnik, and Paolella (2004b) proposed a Markov-switching model where the parameters of a GARCH model change according to a Markov process. An extension of this kind

\* Corresponding author.

E-mail addresses: [ramos.perez.e@gmail.com](mailto:ramos.perez.e@gmail.com) (E. Ramos-Pérez), [pablo.alonsog@uah.es](mailto:pablo.alonsog@uah.es) (P.J. Alonso-González), [josej.nunez@uah.es](mailto:josej.nunez@uah.es) (J.J. Núñez-Velázquez).

<sup>1</sup> Ph.D. Student (Economics and Management Program).

of model can be found in Haas and Paoletta (2012). Before concluding with the GARCH models, it is important to mention that volatility can behave differently depending on the trend of the market: bullish or bearish. To fit this behaviour, Nelson (1991) developed the EGARCH model that allows the sign and the volume of previous values to have separate impacts on the volatility forecasts. In addition to the EGARCH model, Glosten, Jagannathan, and Runkle (1993) proposed the GJR-GARCH to replicate the aforementioned behaviour. Other developments within this family can be found in Engle and Kroner (1995) with their BEKK model, the factor model (Engle, Ng, & Rotschild, 1990), the Constant Conditional Correlation model (Bollerslev, 1990), the time-varying correlation model (Tse & Tsui, 2002), the dynamic correlation model (Engle, 2002) or the multivariate GARCH approach proposed by Kraft and Engle (1982) and Engle, Granger, and Kraft (1984) and its financial implementation by Bollerslev, Engle, and Wooldridge (1988). More recently, Zhang, Zhu, and Ling (2018) have proposed a first order zero drift GARCH (ZD-GARCH) to study heteroscedasticity and conditional heteroscedasticity together.

The second family is composed of those models which assume that the volatility is driven by its own stochastic process. This approach was introduced by Taylor (1982) as an Euler approximation of the underlying diffusion model. Assuming that stock prices follow a Brownian motion, Heston (1993) derived a model where the volatility follows an Ornstein–Uhlenbeck process. To derive the parameters of the Heston Model, two different strategies have been adopted in the literature: moment or simulation. For the first one, the Generalized Method of Moments was proposed by Melino and Turnbull (1990) and Andersen and Sørensen (1999), while the simulation approach has been used by Danielsson (2004), Durbin and Koopman (1997), Broto and Ruiz (2004) or Andersen (2009), amongst others.

The last family presented is Machine Learning, which comprises a set of techniques used to analyse the future evolution of stock prices and volatility. These algorithms try to learn automatically and recognize patterns in a large amount of data (Krollner, Vanstone, & Finnie, 2010). It is worth mentioning that the fitting of these algorithms is quite sensitive to the forecasting time-frame and the selected input variables. Armano, Marchesi, and Murru (2005) and de Faria, Albuquerque, González, Cavalcante, and Albuquerque (2009) suggest using one day as a time-frame and lagged or technical indicators as input variables for the Machine Learning algorithms. Stock prices, volatilities and portfolio selection have been analysed using different methodologies based on Machine Learning, such as Support Vector Machine (Gestel, Suykens, Baestens, Lambrechts, & Laneknet, 2001), hidden Markov models (Dias, Nogueira, Peixoto, & Moreira, 2019; Gupta & Dzinga, 2012) or Artificial Neural Networks (ANN) (Hamid & Iqbal, 2002). These last authors showed that volatility forecasts made by an ANN outperform the implied volatility derived from Barone-Adesi and Whaley options models. Additionally, ANNs have been applied successfully to other financial series different from volatility and stock prices: bond rates (Surkan & Xingren, 2001) and bank failures (Hutchinson, Lo, & Poggio, 1994). Deep learning (LeCun, Bengio, & Hinton, 2015) is a framework closely related with ANN which has been employed for predicting the evolution of Korean stock market index (Chang, Han, & Park, 2017).

Despite the high performance of ANN, predictions derived from the use of this algorithm could be inaccurate when stock prices move sharply (Patel & Yalamalle, 2014). To overcome this problem, ANN were combined with other statistical models (Kristjanpoller, Fadic, & Minutolo, 2014) creating the so called hybrid models. Hybridization can be defined as an approach in which several models are merged to form a new enhanced model in order to produce better forecasting results. Therefore, a hybrid model is

a combination of the artificial intelligence techniques with some components of the traditional forecasting models (like the ones presented within the GARCH family). Examples of this approach are discussed in Roh (2006), Hajizadeh, Seifi, Zarandi, and Turkmen (2012), Lu, Que, and Cao (2016) Monfared and Enke (2014) or Kristjanpoller et al. (2014), where different outputs from a GARCH-based model are used as inputs in an ANN. A more general picture of this type of hybrid models is provided by Bildirici and Ersin (2009), since they compared and combined an ANN with different types of GARCH models (GARCH, EGARCH, GJR-GARCH, TGARCH, NGARCH, SAGARCH, PGARCH, APGARCH and NPGARCH). In addition to the above-mentioned researches, this type of hybrid models has been broadly used in other papers. Bildirici and Ersin (2014) proposed a MS-GARCH with an ANN to improve the forecasting accuracy, Bektipratiwi and Irawan (2011) combined a radial basis function with an EGARCH to model stocks returns of an Indonesian bank and Arneric and Poklepovic (2016) developed an ANN model as an extension of a GJR-GARCH to forecast the market returns of six European emerging markets. GARCH-based models have been also combined with ANNs to predict the volatility in commodity markets, such as gold (Kristjanpoller & Minutolo, 2015) or oil (Kristjanpoller & Minutolo, 2016). In this last case, the hybrid model included financial variables to improve the forecasts. This strategy can also be found in Kristjanpoller and Hernández (2017). Kim and Won (2018) propose a hybrid model that combines a LSTM with various GARCH-type models to forecast the volatility of KOSPI index. A refinement of this model can be found in Back and Kim (2018). It should be mentioned that these models can be generated in both directions: some outputs of a GARCH model can be used as input of an ANN and vice versa (Lu et al., 2016). Finally, it should be noted that hybridisation can not only be made with ANN. Peng, Melo, Camboim de Sá, Akaishi, and Montenegro (2018) proposed a structure combining traditional GARCH-models with Support Vector Machine (SVM) (Cortes & Vapnik, 1995).

The research carried out along this paper develops a volatility forecasting model that consists of two different levels which is based on stacking algorithms methodology (Hastie, Tibshirani, & Friedman, 2009) and statistical models of the Machine Learning family. Random Forest (RF) (Breiman, 2001), Gradient Boosting (GB) with regression trees (Friedman, 2000) and Support Vector Machine (SVM) (Cortes & Vapnik, 1995) are used in the first level, while an ANN (McCulloch & Pitts, 1943) is incorporated within the second level of the stacked model (Stacked-ANN) in order to generate the volatility forecasts. A different two-level approach can be found in Kristjanpoller and Minutolo (2018). They use an ANN-GARCH model with a pre-processing based on principal components analysis to reduce the number of inputs employed in their network. In contrast to the hybrid models defined previously, the proposed model is merging the results arising from other machine learning algorithms which are free of some theoretical assumptions like the use of a predefined distribution for the underlying asset returns or the constant level of unconditional variance. Because of this and with the aim to build a more flexible model, the GARCH-based models are not present in the Stacked-ANN architecture. The proposed model relies completely on the predictions made by machine learning algorithms and market data. Additionally, in the case of the Stacked-ANN the final forecasts made by the first level algorithms are directly used as inputs within the ANN while, in most of the hybrid models discussed in the previous paragraphs, sections of the GARCH-based models are inserted separately in the ANN.

The rest of the paper proceeds as follows: Section 2 presents the set of volatility forecasting models used for comparison purposes. Furthermore, the risk measures and tests used to validate the results are discussed. In Section 3 the theoretical background

and architecture of the volatility forecasting model based on stacking algorithms (Stacked-ANN) are explained. The empirical results of the different forecasting models are shown in Section 4, where the accuracy and the risk measures arising from the proposed model are compared with results obtained by the methodologies explained in Section 2. Finally, Section 5 presents the main conclusions of the results and comparisons carried out along Section 4.

## 2. Benchmark models, risk measurements and statistical tests

As stated above, this section is focused on explaining the benchmark models and the tests used to back-test the risk measurements. Thus, the first paragraphs are dedicated to ANN, ANN-GARCH, ANN-EGARCH and Heston Model, while the end of this section is focused on the risk measurements and tests performed to validate and compare the results of the benchmark models with the one proposed in Section 3.

The first benchmark model is a feed-forward ANN. Following the notation provided by Bishop (2006) and assuming that the algorithm has two hidden layers, the model would be defined by the following expression:

$$\hat{\sigma}_{t+1} = h^{(3)} \left( \sum_{k=1}^T w_{p,k}^{(3)} h^{(2)} \left( \sum_{j=1}^M w_{k,j}^{(2)} h^{(1)} \left( \sum_{i=1}^D w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right) + w_{p,0}^{(3)} \right) \quad (1)$$

Where  $h^{(n)}$  is the activation function associated with the layer  $n$ ,  $w_{z,v}^{(n)}$  is the  $v$ th weight associated with the neuron  $z$  inside the layer  $n$  and  $x_i$  refers to the  $i$  input variable of database comprised by the explicative variables selected by the analyst.

The second benchmark model is an ANN-GARCH( $p, q$ ). As briefly introduced in Section 1, the aim of this hybrid model is to combine the GARCH( $p, q$ ) estimates with other input variables by using an ANN, which is a more flexible model than GARCH( $p, q$ ). Therefore, before starting with the fitting of the ANN, the parameters of the GARCH( $p, q$ ) model need to be estimated:

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^q \alpha_i r_{t-i}^2 + \sum_{i=1}^p \beta_i \sigma_{t-i}^2 \quad / \quad \hat{r}_t = \hat{\sigma}_t \epsilon_t \quad (2)$$

In this formulation  $\omega$ ,  $\alpha_i$  and  $\beta_i$  are the parameters to be estimated, while  $r_t$  and  $\sigma_t^2$  refer to the return and volatility respectively. The returns distribution is determined by the distribution selected for  $\epsilon_t$ . If a standardize normal or standardize Student's  $t$ -distribution is selected, then the returns generated by the model follow a conditional normal (CND) or conditional  $t$ -distribution (CTD) respectively (Bauwens et al., 2012). Once the GARCH( $p, q$ ) parameters are estimated,  $\sum_{i=1}^q \alpha_i r_{t-i}^2$  and  $\sum_{i=1}^p \beta_i \sigma_{t-i}^2$  can be computed and used as input (together with the rest of explicative variables) within the ANN.

The third benchmark model is an ANN-EGARCH. The architecture of this model and the previous one can be considered the same with the unique difference that the first step consists of fitting an EGARCH( $p, q$ ) instead of a GARCH( $p, q$ ) model. The EGARCH( $p, q$ ) can be defined as follows (Nelson, 1991):

$$\log \hat{\sigma}_t^2 = \omega + \sum_{i=1}^p \alpha_i \log \hat{\sigma}_{t-i}^2 + \sum_{i=1}^q (\beta_i \epsilon_{t-i} + \gamma_i (|\epsilon_{t-i}| - E|\epsilon_{t-i}|)) \quad (3)$$

Once the EGARCH is fitted, the following terms can be calculated and used as input within the ANN together with the rest of the explicative variables selected by the analyst:

$$\sum_{i=1}^p \alpha_i \log \hat{\sigma}_{t-i}^2 \quad \sum_{i=1}^q \beta_i \epsilon_{t-i} \quad \sum_{i=1}^q \gamma_i (|\epsilon_{t-i}| - E|\epsilon_{t-i}|) \quad (4)$$

The last benchmark is the Heston (1993) Model. Even though this approach belongs to the stochastic family and the proposed one to the Machine Learning one, this model is going to be used as benchmark during this paper as this process is the most widely used within the family of the stochastic volatility models. It assumes that changes in stock prices through the time ( $dX_t$ ) follow a Brownian diffusion process:

$$dX_t = \mu X_t dt + \sqrt{\sigma_t^2} X_t dB_t \quad (5)$$

Where  $B_t \sim \mathcal{N}(0, \sigma_t^2 t)$ . Therefore, if volatility follows an Ornstein–Uhlenbeck process, the changes in this variable are defined by the following expression:

$$d\sigma_t^2 = \theta (\nu - \sigma_t^2) dt + \delta \sigma_t dB_t^* \quad (6)$$

where  $\nu$  is the long term volatility,  $\theta$  is the rate of return to  $\nu$ ,  $\delta$  is the volatility of  $\sigma_t^2$  and  $B_t^*$  is a Wiener process that has a correlation of  $\rho$  with  $B_t$ .

Once the four benchmark models have been explained, the section focuses on the risk measurements. As stated before, volatility plays a key role in market risk assessment. Therefore, the models will not be only compared in terms of accuracy, but the risk measurements arising from every volatility model are going to be tested. For this purpose, VaR and CVaR have been selected as risk measures. Even though VaR is probably the most used metric due to its simplicity and easy interpretation, CVaR has been also included as it is considered to be a coherent risk measure (Artzner, Delbaen, Eber, & Heath, 1999). Consequently, for every volatility model the aforementioned risk measures are going to be computed and validated by means of the following tests:

- Kupiec (1995) introduced a test in order to check if the number of VaR excesses align with the level of confidence selected.
- An extension of the previous test was developed by Christoffersen et al. (1997). The aim of this test is to validate that VaR excesses are independent, identically distributed and in line with the selected level of confidence.
- Acerbi and Szekely (2014) developed a test (AS1) to assess the appropriateness of the CVaR based on the assumption that VaR has been already tested and considered to be correct from a statistical point of view. The test is inspired by the following equation:

$$E \left[ \frac{r_t}{CVaR_{\alpha,t}} + 1 \mid r_t + VaR_{\alpha,t} < 0 \right] = 0 \quad (7)$$

As VaR needs to be previously validated, the result of this test has to be assessed together with the two aforementioned tests.

- In addition to the previous test, Acerbi and Szekely (2014) introduced another method (AS2) to validate the CVaR without making any assumption about the appropriateness of the VaR. To do so, this test tries to check a CVaR expression that is not conditioned by the correctness of a previous VaR estimate.

Before beginning with the Stacked-ANN architecture, it is worth noticing that the two first tests are parametric while the two last are non-parametric so, for further details about how to compute the statistics and their distributions please refer to aforementioned papers.

## 3. Stacked model

This section has been divided in several sub-sections in order to explain sequentially the proposed volatility forecasting model. As the Stacked-ANN model is composed by two different levels,

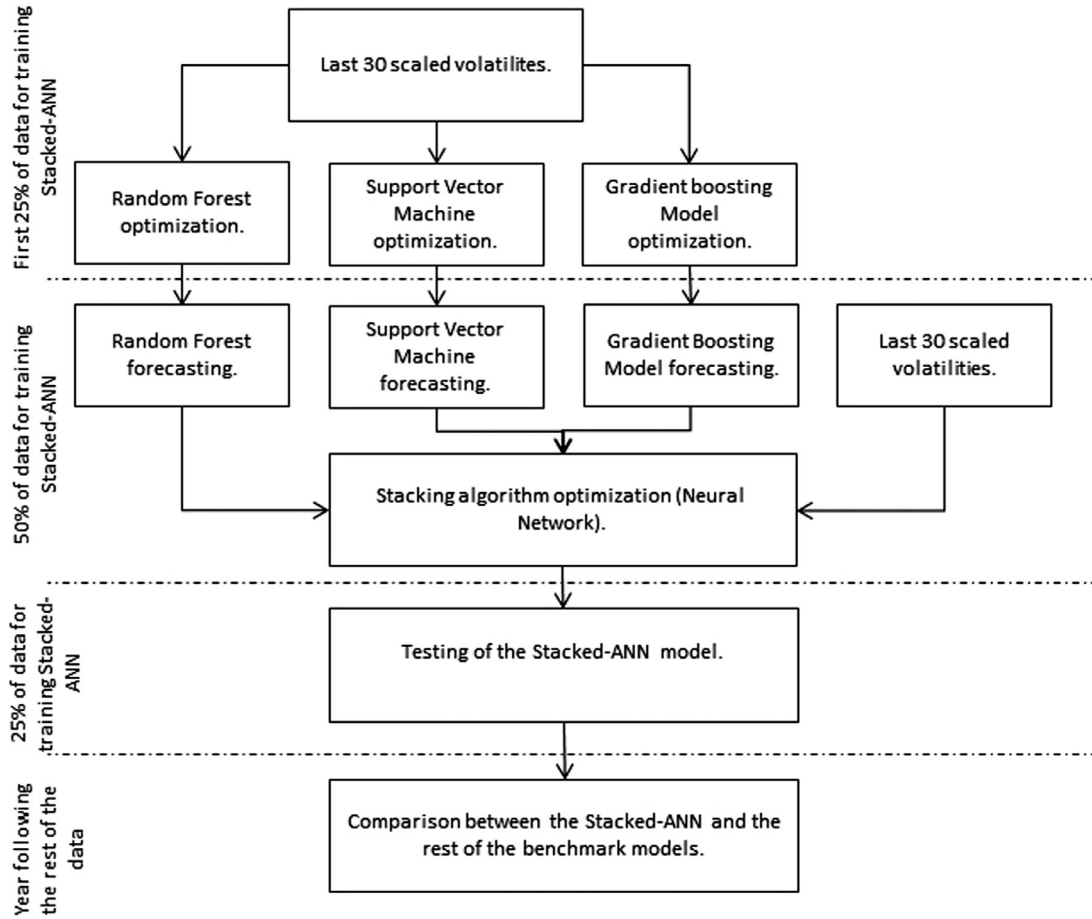


Fig. 1. Stacked-ANN model structure.

the two first sub-sections are dedicated to the input data and the algorithms within the first level of the Stacked-ANN model, while the third and forth sub-sections are focused on the data required to generate the stacking procedure and the details of the ANN fitted with the aforementioned information. (Fig. 1 explains briefly the process followed to estimate and test the Stacked-ANN model).

### 3.1. First level: Input data

The first step is concerned with the creation of the database containing the volatility proxy to be used as a response and the explanatory variables selected to fit the algorithms. As the aim of the study is to predict future volatilities, the True Realized Volatility (hereinafter TRV) is going to be used as response variable (Roh, 2006):

$$TRV_t = \sqrt{\frac{1}{n} \sum_{i=1}^n (r_{t+i-1} - \hat{r}_t)^2} \quad (8)$$

Where  $\hat{r}_t = \sum_{i=n}^n (r_{t+i-1})/n$  and  $n = 5$ . The window has been selected to be large enough to compute a stable TRV and small enough to avoid, as much as possible, mixing different volatility regimes.

The variables given to the first level algorithms to forecast the TRV are the last 30 volatilities computed with returns already observed in the market:

$$V_t = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} (r_{t-n+i} - \hat{r}_t)^2} \quad (9)$$

Where  $\hat{r}_t = \sum_{i=0}^{n-1} (r_{t-n+i})/n$  and  $n = 5$ . Only the last 30 volatilities have been selected because the correlations between previous volatilities and the TRV are residual and therefore their explanatory power is considered to be non-significant. The historical data to compute all the aforementioned variables is obtained by using the quantmod (Ryan & Ulrich, 2017) package from the R project (R Core Team, 2017) and, as suggested by Hastie et al. (2009), they will be scaled to the range [0, 1] to improve the training of the algorithms.

Before beginning with the section related with the algorithms included within the first level, it is important to mention that the first 25% of the data is used to fit the first level algorithms, the next 50% is dedicated to the ANN estimation and the last 25% is the test set. The comparison of the benchmark models with the proposed one in terms of accuracy and risk measurement will be made with a different set of data containing the information of the following year (e.g. if data from 2000 to 2007 is used to train and test the Stacked-ANN model, the out of sample data selected for comparison purposes would be market movements happened during 2008).

### 3.2. First level: Individual algorithms

The methods applied to optimize the hyper-parameters of the algorithms within the first level of the Stacked-ANN architecture are introduced below:

- Minimization of the Mean Square Error (hereinafter, MMSE) for the whole database to train the first level algorithms.

- Circular Block Bootstrap (CBB). This method (Politis & Romano, 1991) generates new samples by selecting random blocks from the original database. The length of these blocks is fixed and the procedure to calculate it was introduced by Politis and White (2004) and Patton, Politis, and White (2009). CBB can only be applied to stationary time series.
- Stationary Bootstrap (hereinafter, SB) (Politis & Romano, 1994). Similar to the case of CBB, this method can only be used with stationary time series. However, the difference with the former method is that the length of the blocks instead of being fixed, it is randomly selected with a certain average that can be calculated using different approaches (see Patton et al. (2009); Politis and White (2004)).
- Maximum Entropy Bootstrap (hereinafter, MEB) (Vinod, 2006; Vinod & de Lacalle, 2009). Unlike the two previous approaches, stationarity is not required as the new samples are obtained from the maximum entropy distribution of the original time series.
- H Cross-Validation (HCV). This method introduced by Chu and Marron (1991) tries to avoid the effect of the correlation that can exist between the response and the explanatory variables while dealing with time series by eliminating  $h$  data points between them. The bandwidth selection is obtained minimizing the absolute autocorrelation between the response and explanatory variables, with a maximum width of 100 days.

The optimum hyper-parameters combination of each one of the five previous methods is obtained by applying grid search. Then, these combinations are tested against data out of sample (the following 50% of the database) to choose the most accurate option for fitting the algorithm.

As stated before, the first level of the stacked model architecture is composed by three algorithms: Random Forest (RF) (Breiman, 2001), Gradient Boosting with regression trees (GB) (Friedman, 2000) and Support Vector Machine (SVM) (Cortes & Vapnik, 1995).

### 3.3. Second level: Input data

As explained in Section 3.1, the first 25% percent of data is dedicated to fit the first level algorithms while the following 50% and 25% are used for fitting the ANN and testing the results respectively. The explanatory variables given to the ANN are:

- As with the first level algorithms, the last 30 volatilities ( $V_t, V_{t-1}, \dots, V_{t-29}$ ) scaled to the range  $[0, 1]$ .
- The True Realized Volatility forecasts made by the first level algorithms: Random forest ( $\widehat{TRV}_{t,RF}$ ), Gradient boosting ( $\widehat{TRV}_{t,GB}$ ) and Support Vector Machine ( $\widehat{TRV}_{t,SVM}$ ).

The response variable is the  $TRV_t$  as defined in Section 3.1.

### 3.4. Second level: Stacking algorithm

As stated previously, the last step of the Stacked-ANN model is the fitting of the ANN, which is the algorithm stacking the forecasts made by the RF, GB and SVM. Before starting with the details of the ANN architecture, notice that the methods and procedures related to the hyper-parameters optimization are the same as the first level algorithms: Grid search in combination with the methods explained in Section 3.2 and final hyper-parameters decision based on the out of sample error (last 25% of the database).

Below, the main characteristics and details of the stacking algorithm are presented:

- The feed-forward ANN has two hidden layers with 20 and 10 neurons respectively. The sigmoid activation function has been selected for all the neurons within the hidden layers while the linear activation function has been used in the output layer, which is comprised by one neuron.
- The optimization algorithm selected is Adaptive Moment Estimation (ADAM), which was created by Kingma and Ba (2014). This method consists in a progressive adaptation of the initial learning rate, taking into consideration current and previous gradients. The default calibration proposed by the authors is applied:  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ .
- The number of epochs are 10,000 and the batch size is equal to the length of the data used for training the ANN.
- The backward pass calculations are done according to the selection of root mean squared error as a loss function.
- As indicated in Section 3.1, the 50% of the information is selected for training the ANN while the following 25% of the data is the test set. Note that the first 25% of the data is used to fit the first level algorithms.
- The parameter adjusting the level of L2 regularization ( $\phi$ ) and the initial learning rate  $\lambda$  used within ADAM are the hyper-parameters to be optimized during the estimation process.

Taking into consideration all the above-mentioned details, the  $TRV_t$  forecasted by the Stacked-ANN model (S-ANN) is obtained by means of the following expression:

$$\begin{aligned} \widehat{TRV}_{t,S-ANN} &= \widehat{f}(\widehat{TRV}_{t,RF}, \widehat{TRV}_{t,GB}, \widehat{TRV}_{t,SVM}, V_t, V_{t-1}, \dots, V_{t-29}) \\ &= h^{(3)} \left( \sum_{k=1}^{10} w_{1,k}^{(3)} h^{(2)} \left( \sum_{j=1}^{20} w_{k,j}^{(2)} h^{(1)} \left( \sum_{i=1}^{33} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) \right. \right. \\ &\quad \left. \left. + w_{k,0}^{(2)} \right) + w_{1,0}^{(3)} \right) \end{aligned} \quad (10)$$

As explained in Section 3.3,  $x_i$  are the last 30 volatilities scaled to the range  $[0, 1]$  and the forecasts made by the first level algorithms.

## 4. Results

During this section, the data used in the empirical analysis, the fitting process and the final comparison between the Stacked-ANN and the benchmark models are shown.

### 4.1. Data

In order to analyse the models under different market conditions, the algorithms have been trained and tested five different times with the S&P 500 volatilities observed in the following periods: 2000–2007, 2001–2008, 2002–2009, 2009–2016 and 2010–2017. As stated in Section 3.1, during the training and testing of the models the first 25% of the periods selected is dedicated to fit the first level algorithms, the next 50% is used to optimize the ANN while the last 25% is reserved for testing purposes. The year after the aforementioned periods (2008, 2009, 2010, 2017 and 2018 respectively for each period) has been used to compare the out of sample results of the Stacked-ANN with the benchmark models. The first three data-sets have been selected in order to analyse the performance of the models during the years after the financial crisis, when the markets were dominated by a high volatile regime. Although the years influenced by the financial crisis are valuable to test the accuracy of the volatility forecasting models, the two last data-sets have been selected in order to analyse the models performance with the most recent data. Additionally, the lower level

**Table 1**  
True realised volatility statistics.

Period	Mean	STD	Skewness	Kurtosis
Year 2008	0.022	0.016	1.510	4.519
Year 2009	0.015	0.008	0.853	3.248
Year 2010	0.010	0.006	0.854	3.736
Year 2017	0.004	0.002	0.911	3.369
Year 2018	0.009	0.006	1.406	4.702

Source: own elaboration.

**Table 2**  
Augmented Dickey–Fuller test.

Period	ADF statistic: Data for training 1st level	ADF statistic: Data for training 2nd level
(2000–2007)	–6.61	–6.41
(2000–2008)	–6.13	–7.91
(2000–2009)	–5.25	–7.57
(2009–2016)	–4.72	–7.27
(2010–2017)	–4.58	–8.82

Source: own elaboration.

of volatility during the last periods, especially in 2017, allows to assess the robustness of the models by analysing them in different market conditions. In order to support the explanations given during this paragraph, Table 1 summarizes the moments of the TRV during the different periods selected to compare the models:

In addition, the Kolmogorov–Smirnov test has been applied sequentially to the TRV in order to assess statistically if the behaviour of the volatility changes over the different periods. As 2008 is the year when the most extreme events related with crisis happened and the market changed from a low to a high volatile regime, the skewness and mean of that year volatility is higher than the one related with 2009. Because of that, the aforementioned test reveals that the volatility of 2008 and 2009 do not belong to the same distribution ( $KS_{p-value} = 0.001$ ). However, when comparing the volatility of 2009 with the 2010 one, the test indicates that they come from the same distribution ( $KS_{p-value} = 0.690$ ). Even though the volatility follows a downward trend, both years are heavily conditioned by the events occurred during 2008 and therefore the test accepts the hypothesis that volatilities belong to the same distribution. Finally, the pair comprised by the volatilities of 2017 and 2018 shows an upward trend. Nevertheless, this increase is not big enough to reject the hypothesis that they come from the same distribution ( $KS_{p-value} = 0.167$ ).

The use of some of the methods proposed in Section 3.2 requires the time series to be stationary. Therefore, before using block bootstrap it has been checked if historical volatility satisfies this property by applying the Augmented Dickey–Fuller test (Dickey & Fuller, 1979) to the different data-sets dedicated to fit the algorithms within the first and second level. The results are shown in Table 2:

As the critical values are –2.63 and –3.43 with a probability of 5% and 1% respectively, it can be concluded that the data meet the requirements imposed by CBB and SB methods.

Previously to the fitting of the algorithms, the parameters needed for the different bootstrap and cross validation methods are obtained by means of the methodologies presented in Section 3.2. As the Stacked-ANN architecture is comprised by two different levels, the length of blocks for CBB, the average of the blocks for SB and the distance,  $h$ , to be used within the HCV method are obtained for both, the data-set to fit first level algorithms and the one dedicated to the second level. Table 3 summarizes the former parameters and it shows non-significant changes over time for the different periods and levels:

**Table 3**  
Calibration of the elements for bootstrap and CV.

Method	Period	Data for training 1st level algorithms	Data for training 2nd level algorithm
CBB Block	(2000–2007)	28	63
CBB Block	(2001–2008)	36	58
CBB Block	(2002–2009)	40	56
CBB Block	(2009–2016)	39	58
CBB Block	(2010–2017)	38	30
SB Block average	(2000–2007)	25	55
SB Block average	(2001–2008)	32	51
SB Block average	(2002–2009)	35	49
SB Block average	(2009–2016)	34	51
SB Block average	(2010–2017)	33	27
HCV length	(2000–2007)	26	31
HCV length	(2001–2008)	31	51
HCV length	(2002–2009)	31	40
HCV length	(2009–2016)	32	55
HCV length	(2010–2017)	35	27

Source: own elaboration.

**Table 4**  
Methods optimizing OOS error.

Period	Stacking Algorithm (ANN)	Random Forest	Gradient Boosting	Support Vector Machine
(2000–2007)	ME	SB	CBB	SB
(2001–2008)	CBB	CBB	CBB	SB
(2002–2009)	CBB	CBB	CBB	CBB
(2009–2016)	HCV	HCV	HCV	SB
(2010–2017)	SB	CBB	SB	SB

Source: own elaboration.

**Table 5**  
Final hyper-parameters.

Period	Stacking Algorithm (ANN)	Random Forest	Gradient Boosting	Support Vector Machine
(2000–2007)	$\phi = 0$ $\lambda = 0.0033$	$N = 10$ $Obs = 24$	$B = 1479$ $\lambda = 0.003$	$\gamma = 0.0001$ $\epsilon = 0.45$
(2001–2008)	$\phi = 0.01$ $\lambda = 0.0059$	$N = 10$ $Obs = 107$	$B = 3000$ $\lambda = 0.001$	$\gamma = 0.0001$ $\epsilon = 0.55$
(2002–2009)	$\phi = 0$ $\lambda = 0.0136$	$N = 1$ $Obs = 37$	$B = 3583$ $\lambda = 0.001$	$\gamma = 0.0004$ $\epsilon = 0.17$
(2009–2016)	$\phi = 0.02$ $\lambda = 0.085$	$N = 30$ $Obs = 118$	$B = 1000$ $\lambda = 0.009$	$\gamma = 0.0002$ $\epsilon = 0.13$
(2010–2017)	$\phi = 0.01$ $\lambda = 0.011$	$N = 7$ $Obs = 175$	$B = 1000$ $\lambda = 0.003$	$\gamma = 0.0001$ $\epsilon = 0.54$

Source: own elaboration.

#### 4.2. Fitting of the Stacked-ANN model

As explained in Section 3.2, different approaches have been followed to find the optimum hyper-parameter combination. Table 4 shows the methods that minimize the out of sample error per each algorithm and period:

Regardless of the period, the empirical results suggest that CBB and SB outperform the rest of the methods. These outcomes are expected as these two methods based on re-sampling blocks from the original database are specifically prepared to work with stationary time series. Table 5 summarizes the hyper-parameters suggested by the methods shown in Table 4:

Where  $\lambda$  is the learning rate of the ANN and GB,  $\phi$  the parameter adjusting the level of L2 regularization of the ANN,  $B$  the number of iterations performed while fitting the GB,  $N$  the number of variables randomly selected by the RF and  $Obs$  the minimum number of observations to be kept in the terminal nodes of every fitted tree within the RF architecture. Finally,  $\gamma$  refers to the parameter included within the radial basis function kernel (the lower the parameter, the higher the non-linearity) and  $\epsilon$  defines the threshold where the error begins to be penalized by the SVM.

**Table 6**  
Accuracy analysis.

Model	RMSE: 2008	RMSE: 2009	RMSE: 2010	RMSE: 2017	RMSE: 2018
Stacked-ANN	0.01192	0.00534	0.00494	0.00254	0.00544
ANN-EGARCH	0.01332	0.00588	0.00537	0.00276	0.00571
ANN-GARCH	0.01335	0.00584	0.00539	0.00263	0.00575
Heston	0.02066	0.00714	0.00547	0.00359	0.00610
ANN	0.01526	0.00615	0.00541	0.00274	0.00590

Source: own elaboration.

#### 4.3. Comparison against benchmark models

Once the Stacked-ANN is fitted, its performance is compared with the benchmark models explained in Section 2 (ANN, ANN-GARCH(1, 1), ANN-EGARCH(1, 1) and Heston Model). Before beginning with the comparisons, the three following remarks about the benchmark models have to be done:

- Due to the nature of the Heston Model, 20,000 simulations per each day have been computed and the daily average of them has been taken to assess its accuracy.
- The GARCH(1, 1) and EGARCH(1, 1) (included in the ANN-GARCH(1, 1) and ANN-EGARCH(1, 1) architecture respectively) have been estimated assuming Student-t innovations.
- The fitting procedure and architecture of the ANNs included within ANN-GARCH(1, 1), ANN-EGARCH(1, 1) and ANN models are the same as the ones explained for the Stacked-ANN (see Section 3.4).

Table 6 shows the out of sample error of the different periods selected to compare the performance and robustness of the Stacked-ANN with the benchmark models. The results shown in this table suggest the following conclusions:

- Regardless of the period, the Stacked-ANN outperforms other hybrid models based on auto-regressive methodologies like ANN-GARCH and ANN-EGARCH. In relative terms, minor deviations are observed between the different periods.
- All the hybridized models tend to outperform the pure ANN model.
- As expected due to the extremely high volatilities observed during the financial crisis, the results show that, regardless of the model, 2008 forecasts are less accurate. All the models minimize their error rate in the year with the lowest level volatility, 2017.
- The forecasts made by the Heston Model tend to be the less accurate due to the non-predictive nature of this model.

In addition to the above-mentioned analysis, the risk measures obtained by using each one of the volatility models are tested. In order to do so, a returns distribution is selected for each one of the forecasting volatility methods. As described in Section 2, Heston Model requires the changes in stock prices to follow a Brownian diffusion process. Nevertheless, for the rest of the benchmark models and the Stacked-ANN (which are free of assumptions about the returns) a Student t-distribution has been combined with the different volatility forecasts. This assumption about Student t-distribution has been selected when possible as returns tend to be leptokurtic and heavier-tailed than Normal distribution (McNeil, Frey, & Embrechts, 2015).

Before analysing the results of the tests presented in Section 2, it is worth mentioning that the level of confidence (99%) and number of days (10) selected are based on the ones set by Basel Directive, whose aim is to monitor, amongst others, the market risk. Table 7 shows the p-value of the tests dedicated to VaR (Kupiec and Christoffersen) and CVaR (AS1 and AS2). If a 95%

**Table 7**  
P-value of the VaR and CVaR tests.

Model	Test	Period: 2008	Period: 2009	Period: 2010	Period: 2017	Period: 2018
Stacked-ANN	Kupiec	0.85	0.84	0.65	0.85	0.85
	Christ.	0.01	0.79	0.02	0.01	0.01
ANN-EGARCH	AS1	0.66	0.85	0.61	0.90	0.91
	AS2	0.56	0.63	0.36	0.67	0.69
	Kupiec	0.12	0.12	0.84	0.03	0.03
	Christ.	0.00	0.00	0.01	0.03	0.03
ANN-GARCH	AS1	0.52	0.85	0.61	1.00	1.00
	AS2	0.07	0.19	0.62	0.91	0.91
	Kupiec	0.12	0.03	0.01	0.03	0.03
	Christ.	0.00	0.03	0.00	0.03	0.03
Heston Model	AS1	0.51	1.00	0.77	1.00	1.00
	AS2	0.08	0.92	0.05	0.85	0.89
	Kupiec	0.00	0.00	0.65	0.03	0.00
	Christ.	0.00	0.00	0.59	0.03	0.00
ANN	AS1	0.00	0.01	0.83	1.00	0.06
	AS2	0.00	0.00	0.36	0.92	0.00
	Kupiec	0.65	0.04	0.65	0.30	0.29
	Christ.	0.02	0.00	0.00	0.00	0.00
	AS1	0.24	0.86	0.59	0.81	0.00
	AS2	0.29	0.11	0.35	0.24	0.00

Source: own elaboration.

is set as confidence level, Stacked-ANN in combination with Student t-distribution is the only model that produces an appropriate p-value for Kupiec, AS1 and AS2 tests in every period under analysis. All the models show difficulties to produce a p-value higher or equal than 0.05 for the Christoffersen test because VaR exceedances tend to happen in a short period of time instead of spread over the period analysed. It is worth mentioning that the hybrid models taken as benchmark (ANN-EGARCH and ANN-GARCH) also fail in producing an appropriate value for the Kupiec test in several periods while, as stated before, the proposed hybrid model (Stacked-ANN) pass the test for every period. Finally, Heston Model tends to produce less appropriate risk measures due to the distribution constrain mentioned previously.

## 5. Conclusions

This paper introduces a Stacked-ANN model based only on Machine Learning techniques with the aim to improve the accuracy of the volatility forecasts made by other hybrid models based on a combination of GARCH or EGARCH with ANNs. Its predictive power and performance has been tested in terms of RMSE, VaR and CVaR.

Two main results have to be pointed out. Firstly, the Stacked-ANN has been able to generate more accurate volatility forecasts than other models in a high volatile regime period like the one occurred after the Financial Crisis of 2007–2008. The models outperformed by the Stacked-ANN during that time lapse are other hybrid models like ANN-GARCH and ANN-EGARCH, the most widely used stochastic volatility theory (Heston Model) and a feed-forward ANN without any combination with other algorithms or statistical models. Notwithstanding the Stacked-ANN performance, it is observed for every model that the higher the volatility the lower the accuracy. In addition to this analysis, the Stacked-ANN has been tested with the most recent data (2017 and 2018) in order to check its performance in the current market conditions. As it occurred with the tests carried out during the financial crisis, the proposed architecture outperforms the benchmark models in terms of accuracy. The superior performance shown by the Stacked-ANN in periods with different levels of volatility are due to the model flexibility. In contrast with ANN-GARCH or ANN-EGARCH, the inputs introduced in the ANN stacked model do not follow any theoretical assumption about the returns distribution or volatility. As explained throughout Section 3, the architecture proposed uses

previous volatilities and forecasts made by a random forest, gradient boosting with regression trees and support vector machine as inputs. Before beginning with the second point of the conclusion, it is worth mentioning that it has been empirically demonstrated that block bootstrap methods are of special interest when fitting algorithms to volatility as these procedures are especially prepared to work with stationary time series.

Secondly, the forecasts made by the volatility models have been combined with a certain distribution in order to compute the VaR and CVaR for all the different periods analysed. The distribution selected has been the Student's t-distribution for every model with the exception of the Heston Model which requires changes in asset prices to follow a Brownian diffusion process. The empirical results demonstrated that only the Stacked-ANN model is able to produce an appropriate p-value for Kupiec, AS1 and AS2 tests in every period under analysis, including those ones related with the financial crisis.

The aforementioned flexibility and predictive power of the Stacked-ANN compared with other volatility models suggest to develop further investigations about the implications of using this model for derivative valuation purposes. As the price of these instruments is closely related to the volatility of the underlying assets, further researches should be done in order to compare the implied volatilities observed in the market with the ones arising from the proposed model. If the volatility measured by the Stacked-ANN is more accurate than market expectations, it would be possible to identify under and overvalued derivatives.

#### Credit authorship contribution statement

**Eduardo Ramos-Pérez:** Conceptualization, Methodology, Software, Formal Analysis, Writing - Original Draft, Writing - Review & Editing. **Pablo J. Alonso-González:** Methodology, Validation, Investigation, Writing - Original Draft, Writing - Review & Editing, Supervision, Project Administration. **José Javier Núñez-Velázquez:** Methodology, Validation, Investigation, Writing - Original Draft, Writing - Review & Editing, Supervision, Project Administration.

#### References

- Acerbi, C., & Szekely, B. (2014). Backtesting expected shortfall. *Risk*, 1–14.
- Aharon, D., Gavious, I., & Yosef, R. (2010). Stock markets bubble effects on mergers and acquisitions. *The Quarterly Review of Economics and Finance*, 50(4), 456–470.
- Andersen, T. (2009). *Encyclopedia of complexity and system sciences*. Springer Verlag.
- Andersen, T., & Sorensen, B. (1999). GMM estimation of a stochastic volatility model: A Monte Carlo study. *Journal of Business and Economic Statistics*, 14, 329–352.
- Armano, G., Marchesi, M., & Murru, A. (2005). A hybrid genetic-neural architecture for stock indexes forecasting. *Information Sciences*, 170(1), 3–83.
- Arneric, J., & Poklepovic, T. (2016). *Nonlinear Extensions of Asymmetric GARCH Model within Neural Network Framework*. AIRCC Publishing Corporation, Chennai, India.
- Artzner, P., Delbaen, F., Eber, J.-M., & Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, 9(3), 203–228.
- Back, Y., & Kim, H. (2018). ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications*, 113, 457–480.
- Bauwens, L., Hafner, C., & Laurent, S. (2012). *Handbook of Volatility Models and Their Applications*. Wiley Handbooks in Financial E. Wiley.
- Bektipratiwi, A., & Irawan, M. (2011). A RBF-EGARCH neural network model for time series forecasting. (pp. 1–8).
- Bildirici, M., & Ersin, O. (2009). Improving forecasts of GARCH family models with the artificial neural networks: An application to the daily returns in Istanbul Stock Exchange. *Expert Systems with Applications*, 36(4), 7355–7362.
- Bildirici, M., & Ersin, O. (2014). Modelling Markov Switching ARMA-GARCH Neural Networks Models and an Application to Forecasting Stock Returns. *Hindawi Publishing Corporation: The Scientific World Journal*, 2014.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag.
- Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3), 307–327.
- Bollerslev, T. (1990). Modeling the coherence in short-run nominal exchange rates: A Multivariate Generalized ARCH Model. *Review of Economics and Statistics*, 72, 498–505.
- Bollerslev, T., Engle, R., & Wooldridge, J. (1988). A Capital Asset Pricing Model with time-varying covariances. *Journal of Political Economy*, 96, 116–131.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. doi:10.1023/A:1010933404324.
- Broto, C., & Ruiz, E. (2004). Estimation methods for stochastic volatility models: A survey. *Journal of Economic Surveys*, 18, 613–649.
- Chang, E., Han, C., & Park, F. (2017). Deep learning networks for stock markets analysis and prediction: Methodology, data representations and case studies. *Expert System with Applications*, 83, 187–205.
- Christoffersen, P. F., Bera, A., Berkowitz, J., Bollerslev, T., Diebold, F., Giorgianni, L., et al. (1997). Evaluating interval forecasts. *International Economic Review*, 39, 841–862.
- Chu, C.-K., & Marron, J. S. (1991). Comparison of two bandwidth selectors with dependent errors. *Annals of Statistics*, 19(4), 1906–1918. doi:10.1214/aos/1176348377.
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297. doi:10.1023/A:1022627411411.
- Danielsson, J. (2004). Stochastic volatility in asset prices: Estimation by simulated maximum likelihood. *Journal of Econometrics*, 64, 375–400.
- Dias, F., Nogueira, R., Peixoto, G., & Moreira, W. (2019). Decision-making for financial trading: A fusion approach of machine learning and portfolio selection. *Expert Systems with Applications*, 115, 635–655.
- Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366a), 427–431. doi:10.1080/01621459.1979.10482531.
- Durbin, J., & Koopman, S. (1997). Monte Carlo maximum likelihood estimation for non-Gaussian state space models. *Biometrika*, 84, 669–684.
- Engle, R. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50, 987–1007.
- Engle, R. (2002). Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business and Economic Statistics*, 20, 339–350.
- Engle, R., Granger, C., & Kraft, D. (1984). Combining competing forecasts of inflation with a bivariate ARCH model. *Journal of Economic Dynamics and Control*, 8, 151–165.
- Engle, R., & Kroner, F. (1995). Multivariate simultaneous generalized ARCH. *Econometric Theory*, 11, 122–150.
- Engle, R., & Lee, G. (1999). In R. Engle, & H. White (Eds.), *A permanent and transitory component model of stock return volatility* (pp. 475–497). Oxford: Oxford University Press.
- Engle, R., Ng, V., & Rotschild, M. (1990). Asset pricing with a factor-ARCH covariance structure: Empirical estimates for Treasury Bills. *Journal of Econometrics*, 45, 213–238.
- de Faria, E., Albuquerque, M., González, J., Cavalcante, J., & Albuquerque, M. (2009). Predicting the Brazilian stock market through neural networks and adaptive exponential smoothing methods. *Expert Systems with Applications*, 36(10), 12506–12509.
- Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29, 1189–1232.
- Gestel, T., Suykens, J., Baetens, D., Lambrechts, A., & Laneknet, G. (2001). Financial time series prediction using least squares Support Vector Machines within the evidence framework. *IEEE Transactions on Neural Networks*, 12(4), 8009–8821.
- Glosten, L., Jagannathan, R., & Runkle, D. (1993). On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *The Journal of Finance*, 48(5), 1779–1801.
- Gupta, A., & Dhingra, B. (2012). Stock markets prediction using hidden Markov models. 2012 Students conference on engineering and systems, (pp. 1–4).
- Haas, M., Mittnik, S., & Paoletta, M. (2004a). Mixed normal conditional heteroskedasticity. *Journal of Financial Econometrics*, 2, 211–250.
- Haas, M., Mittnik, S., & Paoletta, M. (2004b). A new approach to Markov-switching GARCH models. *Journal of Financial Econometrics*, 2, 493–530.
- Haas, M., & Paoletta, M. (2012). In L. Bauwens, C. Hafner, & S. Laurent (Eds.), *Mixture and regime-switching GARCH models* (pp. 71–102). John Wiley and Sons.
- Hajizadeh, E., Seifi, A., Zarandi, F., & Turksen, I. (2012). A hybrid modeling approach for forecasting the volatility of S&P 500 index return. *Expert Systems with Applications*, 39(1), 531–536.
- Hamid, S., & Iqbal, Z. (2002). Using neural networks for forecasting volatility of S&P 500 index futures prices. *Journal of Business Research*, 57(10), 1116–1125.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer Series in Statistics (Second Edition). Springer New York.
- Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6, 327–343.
- Hull, J. (2015). *Risk management and financial institutions* (4th edition). Wiley and Sons, London.
- Hutchinson, J., Lo, A., & Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, 49, 851–859.
- Kim, H., & Won, C. (2018). Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications*, 103, 25–37.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. CoRR: abs/1412.6980.
- Kraft, D., & Engle, R. (1982). *Autoregressive conditional heteroskedasticity in multiple time series*. Department of Economics, UCSD.
- Kristjanpoller, W., Fadic, A., & Minutolo, M. (2014). Volatility forecast using hybrid neural network models. *Expert Systems with Applications*, 41(5), 2437–2442.

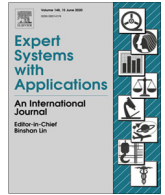
- Kristjanpoller, W., & Hernández, E. (2017). Volatility of main metals forecasted by a hybrid ANN-GARCH model with regressors. *Expert Systems with Applications*, 84, 290–300.
- Kristjanpoller, W., & Minutolo, M. (2015). Gold price volatility: A forecasting approach using the Artificial Neural Network-GARCH model. *Expert Systems with Applications*, 42(20), 7245–7251.
- Kristjanpoller, W., & Minutolo, M. (2016). Forecasting volatility of oil price using an Artificial Neural Network-GARCH model. *Expert Systems with Applications*, 65(15), 233–241.
- Kristjanpoller, W., & Minutolo, M. (2018). A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. *Expert Systems with Applications*, 109, 1–11.
- Krollner, B., Vanstone, B., & Finnie, G. (2010). Financial time series forecasting with machine learning techniques: A survey. *European symposium on artificial neural networks: Computational and machine learning*.
- Kupiec, P. H. (1995). Techniques for verifying the accuracy of risk measurement models. *The Journal of Derivatives*, 3(2), 73–84. doi:10.3905/jod.1995.407942.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Lowenstein, R. (2000). *When genius failed: The rise and fall of long-term credit management*. Random House.
- Lu, X., Que, D., & Cao, G. (2016). Volatility forecast based on the hybrid artificial neural network and garch-type models. *Procedia Computer Science*, 91, 1044–1049.
- Mandelbrot, B. (1963). The variation of certain speculative prices. *Journal of Business*, 36, 394–419.
- Mcculloch, W., & Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 127–147.
- McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton, NJ, USA: Princeton University Press.
- Melino, A., & Turnbull, S. (1990). Pricing foreign currency options with stochastic volatility. *Journal of Econometrics*, 45, 239–265.
- Monfared, S. A., & Enke, D. (2014). Volatility forecasting using a hybrid gjr-garch neural network model. *Procedia Computer Science*, 36, 246–253.
- Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: A new approach. *Econometrica*, 59(2), 347–370.
- Patel, M., & Yalamalle, S. (2014). Stock price prediction using artificial neural network. *International Journal of Innovative Research in Science, Engineering and Technology*, 3(June 2014), 13755–13762.
- Patton, A., Politis, D. N., & White, H. (2009). Correction to “automatic block-length selection for the dependent bootstrap” by d. politis and h. white. *Econometric Reviews*, 28(4), 372–375. doi:10.1080/07474930802459016.
- Peng, Y., Melo, P., Camboim de Sá, J., Akaishi, A., & Montenegro, M. (2018). The best of two worlds: Forecasting high frequency volatility for cryptocurrencies and traditional currencies with support vector regression. *Expert Systems with Applications*, 97, 177–192.
- Politis, D., & Romano, J. (1991). A Circular Block-resampling Procedure for Stationary Data. Purdue University. Department of Statistics.
- Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870.
- Politis, D. N., & White, H. (2004). Automatic block-length selection for the dependent bootstrap. *Econometric Reviews*, 23(1), 53–70. doi:10.1081/ETC-120028836.
- Poon, S., & Granger, C. (2003). Forecasting volatility in financial markets. a review. *Journal of Economic literature*, 41(2), 478–539.
- R Core Team (2017). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.
- Rajashree, P., & Ranjeeeta, B. (2015). A differential harmony search based hybrid internal type2 fuzzy EGARCH model for stock market volatility prediction. *International Journal of Approximate Reasoning*, 59, 81–104.
- Roh, T. (2006). Forecasting the volatility of stock price index. *Expert Systems with Applications*, 33(4), 916–922.
- Ryan, J. A., & Ulrich, J. M. (2017). quantmod: Quantitative financial modelling framework. R package version 0.4–12.
- Surkan, A., & Xingren, Y. (2001). Bond rating formulas derived through simplifying a trained neural network. *Proceedings of the IEEE International conference on neural network*, 2, 1028–1031.
- Taylor, S. (1982). In D. Anderson (Ed.), *Financial returns modelled by the product of two stochastic processes, a study of daily sugar prices 196179: (vol.1 (pp. 223–226))*. North-Holland.
- Tse, Y., & Tsui, K. (2002). A multivariate GARCH model with time-varying correlations. *Journal of Business and Economic Statistics*, 20, 351–362.
- Vinod, H. (2006). Maximum entropy ensembles for time series inference in economics. *Journal of Asian Economics*, 17(6), 955–978.
- Vinod, H. D., & de Lacalle, J. L. (2009). Maximum entropy bootstrap for time series: The meboot R package. *Journal of Statistical Software*, 29(5), 1–19.
- Zhang, L., Zhu, K., & Ling, S. (2018). The ZD-GARCH model: A new way to study heteroscedasticity. *Journal of Econometrics*, 202(1), 1–17.

## **8.2 Annex II. Published Paper. Stochastic reserving with a stacked model based on a hybridized Artificial Neural Network**



Contents lists available at ScienceDirect

## Expert Systems with Applications

journal homepage: [www.elsevier.com/locate/eswa](http://www.elsevier.com/locate/eswa)

## Review

## Stochastic reserving with a stacked model based on a hybridized Artificial Neural Network

Eduardo Ramos-Pérez<sup>a</sup>, Pablo J. Alonso-González<sup>b,\*</sup>, José Javier Núñez-Velázquez<sup>b</sup><sup>a</sup> Ph.D. Student (Economics and Management Program), Universidad de Alcalá, Spain<sup>b</sup> Economics Department, Universidad de Alcalá, Plaza de la Victoria 2, 28802 Alcalá de Henares, Spain

## ARTICLE INFO

## Article history:

Received 18 September 2019

Revised 17 July 2020

Accepted 18 July 2020

Available online 3 August 2020

## AMS Subject Classification:

62-07

62P05

65C60

90-08

## Keywords:

Stochastic reserving

Reserving risk

Machine learning

General insurance

Run-off prediction

## ABSTRACT

Currently, legal requirements demand that insurance companies increase their emphasis on monitoring the risks linked to the underwriting and asset management activities. Regarding underwriting risks, the main uncertainties that insurers must manage are related to the premium sufficiency to cover future claims and the adequacy of the current reserves to pay outstanding claims. Both risks are calibrated using stochastic models due to their nature. This paper introduces a reserving model based on a set of machine learning techniques such as Gradient Boosting, Random Forest and Artificial Neural Networks. These algorithms and other widely used reserving models are stacked to predict the shape of the runoff. To compute the deviation around a former prediction, a log-normal approach is combined with the suggested model. The empirical results demonstrate that the proposed methodology can be used to improve the performance of the traditional reserving techniques based on Bayesian statistics and a Chain Ladder, leading to a more accurate assessment of the reserving risk.

© 2020 Elsevier Ltd. All rights reserved.

## 1. Introduction

As with any other company, the survival of an insurance firm depends on its ability to obtain a sustainable profit over the years. These entities have to offer their services at an adequate and competitive premium, while the ultimate cost of the claims is subject to uncertainty. Thus, reserving models were developed in order to estimate and monitor the expected ultimate cost of outstanding claims. Although life insurance contracts manifest uncertainty about the claims cost, reserving takes a special relevance in general insurance as that uncertainty tends to be higher, at least in the short term.

Methods of estimating the level of reserves in non-life insurance have evolved from classical and deterministic methods toward others that take into account the loss reserve uncertainty. The aim of the first type is to estimate the expected level of reserves by taking the historical information into consideration. Chain Ladder is the most frequently used method of this family.

When historical data are not stable enough to use the Chain Ladder technique, the [Bornhuetter and Ferguson \(1972\)](#) model tends to be the preferred option to obtain an adequate estimate of the expected ultimate cost.

The increasing interest of investors in the risk profile of financial institutions since the Financial Crisis of 2007–2008 and the implementation of the Solvency II Directive in the European market have fostered the use of stochastic reserving models. As in the case of deterministic approaches, stochastic models based on the Chain Ladder technique are the most commonly used. One of the main techniques within this family is the Overdispersed Poisson (ODP) model developed by [Renshaw and Verrall \(1998\)](#) and its bootstrap implementation suggested by [England and Verrall \(1999\)](#) and [England \(2002\)](#) which assumes that incremental claims follow an ODP distribution where the variance is proportional to the mean.

In this model, incremental claims must be positive, but this limitation can be overcome by using the quasi-likelihood approach introduced by [McCullagh and Nelder \(1989\)](#). In cases where the ODP assumption does not properly fit the data, [Kremer \(1982\)](#), [Mack \(1991\)](#) and [Verrall \(2000\)](#) developed other models assuming

\* Corresponding author.

E-mail addresses: [ramos.perez.e@gmail.com](mailto:ramos.perez.e@gmail.com) (E. Ramos-Pérez), [pablo.alonso-uah.es](mailto:pablo.alonso-uah.es) (P.J. Alonso-González), [josej.nunez@uah.es](mailto:josej.nunez@uah.es) (J.J. Núñez-Velázquez).

log-normal, gamma and negative binomial distributions respectively. In contrast to the methods within this family, Mack (1993) developed a free-distribution model by focusing and limiting the claims reserve distribution analysis to the first two moments.

Thus, the bootstrap implementation of Mack's model allows the analyst to obtain a reserve distribution without the necessity of defining a theoretical distribution for the cumulative or incremental claim cost. If the bootstrapping procedure is to be avoided, England and Verrall (2006) introduced a stochastic Bayesian implementation of the ODP, Negative Binomial and this last free-distribution model. This approach was recently expanded by Meyers (2015), who developed some Bayesian Markov Chain Monte-Carlo (MCMC) models (Levelled Chain-Ladder, Correlated Chain-Ladder, Levelled Incremental Trend, Correlated Incremental Trend and Changing Settlement Rate) for incurred and paid data. Their aim is to improve the performance of ODP and Mack models by using different approaches such as recognizing the correlation between accident years, including a skewed distribution to model negative incremental payments, introducing a trend over the development years and allowing changes in the claim settlement rate.

Another set of models is focused on using several triangles simultaneously in order to take into consideration different characteristics of incurred and paid data. The main models within this family are the Munich Chain Ladder (MCL) method and Double Chain Ladder (DCL) model developed by Quarg and Mack (2004) and Martínez-Miranda, Nielsen, and Verrall (2012), respectively. By modifying this last method, Margraf, Elpidorou, and Verrall (2018) addressed the problem of calculating general insurance reserves when the portfolio is covered by an excess-of-loss reinsurance. In addition to MCL and DCL, Merz and Wüthrich (2010) introduced a Bayesian implementation of the paid-incurred chain (PIC) reserving method (Posthuma, Cator, Veerkamp, & Van Zwet, 2008) based on using both incurred and paid data. Happ, Merz, and Wüthrich (2012) and Happ and Wüthrich (2013) also investigated and developed models related to the PIC method, while Halliwell (2009) and Venter (2008) introduced regression approaches based on using both data sources. Pigeon, Antonio, and Denuit (2014), Antonio and Plat (2014), and Martínez-Miranda, Nielsen, and Verrall (2013b) also proposed models by taking into consideration different data sources to estimate the expected ultimate claim cost.

In addition to the different approaches exposed above, it is possible to find models where the information is not organized in an aggregated way, as in the classical triangles, but rather in individual claims data (see Taylor, McGuire, & Sullivan, 2008; Jessen, Mikosch, & Samorodnitsky, 2011; Pigeon, Antonio, & Denuit, 2013; Antonio & Plat, 2014; Martínez-Miranda, Nielsen, Verrall, & Wüthrich, 2015; Charpentier & Pigeon, 2016, or Wüthrich, 2018b).

Thanks to the increase in computational power, machine learning techniques have turned into an adequate tool for reserving purposes. Artificial Neural Networks (Gabrielli & Wüthrich, 2018; Wüthrich, 2018b), regression trees (Wüthrich, 2018a), Recurrent Neural Networks (Kuo, 2018) or tree-based algorithms (Lopez, Milhaud, & Théron, 2019) have been used to predict claim reserves. Gabrielli, Richman, and Wüthrich (2018) embedded the ODP model into a neural network framework, and Baudry and Robert (2019) introduced a nonparametric reserving model based on extremely randomized trees (Geurts, Ernst, & Wehenkel, 2006) and individual claims data. In addition to the aforementioned algorithms, other machine learning techniques were used by Martínez-Miranda, Nielsen, and Verrall (2013a) for reserving purposes, and a support vector machine was applied to classify risks prior to the reserve calculation (Duma, Twala, Marwala, & Nelwamondo, 2011).

The research carried out in this paper develops a nonparametric reserving model based on the stacking algorithm methodology. The proposed architecture consists of two different levels. Random Forest (RF) (Breiman, 2001), Gradient Boosting (GB) with regression trees (Friedman, 2000), Artificial Neural Network (ANN) (McCulloch & Pitts, 1943), Changing Settlement Rate (CSR) reserving model and the Chain Ladder assumptions are incorporated within the first level, while an ANN is included in the second level of the stacked model (Stacked-ANN) architecture in order to generate the final predictions. Therefore, the aim of this hybrid model is to improve the performance of the individual components by creating an architecture that can learn from the different algorithms and the reserving models included within the first level.

Although the overall methodology is based on that proposed by Ramos-Pérez, Alonso-González, and Núñez-Velázquez (2019) for stock volatility forecasting purposes, the model architecture proposed in this study is different. In this research, machine learning algorithms and reserving models are present in the first level, while in the architecture developed by Ramos-Pérez et al. (2019), only machine learning algorithms were included. Therefore, the most popular models for forecasting volatility such as GARCH or EGARCH were not integrated within the model architecture, while in this case, Chain Ladder and CSR are incorporated. It is also worth mentioning that in contrast to the hybrid model proposed for forecasting volatility purposes, in this research, the second level only receives information already processed by the models within the first level. In addition to the main differences explained above, it should be pointed out that the stacking algorithm methodology has not appeared previously in the actuarial literature related to the valuation of loss reserves. Apart from that, a log-normal approach is combined with the suggested reserving model based on machine learning in order to compute the reserve variability.

As all the different algorithms and reserving models of the first level are incorporated in the ANN of the second level, some of the most important research studies carried out in the context of selecting the optimal ANN architecture will be discussed. There is a significant amount of literature supporting the use of ANNs with just one hidden layer because under mild assumptions on the activation functions, the universal approximation theorem states that a feedforward ANN with a single hidden layer and a finite number of neurons can approximate any continuous function on compact subsets of the Euclidean space.

Based on regularization techniques and using just one hidden layer network, Poggio and Girosi (1990) developed a theoretical framework to approximate nonlinear mappings named regularization networks. These authors demonstrated that their architecture can approximate any continuous function on a compact domain if the number of units is high enough. Cybenko (1989) and Hornik, Stinchcombe, and White (1989) also proved that one hidden layer networks with sigmoidal activation functions can approximate continuous functions on any compact Euclidean space. It was also shown that, under certain conditions, an arbitrarily small error between a single hidden layer ANN and any other continuous function can be obtained by increasing the number of neurons (Barron, 1994; Funahashi, 1989; Hornik, 1993). Nakama (2011) showed that the range of effective learning rates is wider in the case of ANN with one hidden layer than in architectures with multiple hidden layers.

On the other hand, Hornik (1991) and Leshno, Lin, Pinkus, and Schocken (1993) demonstrated that ANNs have the potential of being universal approximators not only due to the choice of a specific activation function but also because of the possibility of using several hidden layers. Limitations of the approximation capabilities of one hidden layer networks were demonstrated by Chui, Li, and Mhaskar (1994) and Chui, Li, and Mhaskar (1996).

In recent years, multi-hidden layer architectures have improved the state of the art in machine learning.

For example, in the context of natural language processing, the models and architectures created by Devlin, Chang, Lee, and Toutanova (2018) (BERT), Brown et al. (2020) (GPT3) and Vaswani et al. (2017) (Transformer) overcome the performance of other less complex models. In addition, it is worth mentioning that agents trained with multi-hidden layer ANNs have been able to overcome the human performance in specific tasks such as playing chess (Silver et al., 2017) or 'go' (Silver et al., 2016). With respect to the optimal number of neurons, Celikoglu (2007) analysed this issue in the context of solving the dynamic network loading problem, while Sheela and Deepa (2013) proposed a list of principles to select this number.

Results from recently published papers in the actuarial field support the idea of applying ANNs with multiple hidden layers. Indeed, Richman and Wüthrich (2018) and Nigri, Levantesi, Marino, Scognamiglio, and Perla (2019) applied this structure to model human mortality, while Castellani et al. (2018) used it for estimating the economic capital of insurance companies under the Solvency II framework. Thus, the ANNs included within the architecture of the Stacked-ANN model have several hidden layers.

The rest of the paper proceeds as follows: Section 2 presents the set of models used for comparison purposes. Additionally, the error and risk measures taken to validate the stochastic reserves, payments and ultimate losses are discussed. In Section 3, the theoretical background and architecture of the reserving model based on stacking algorithms (Stacked-ANN) are explained. Details about the log-normal approach proposed for obtaining a stochastic distribution are also given in this section. The empirical results, error and risk measures of the different reserving models are shown in Section 4. Finally, Section 5 presents the main conclusions derived from the results and comparisons presented in Section 4.

## 2. Benchmark models and validation

As previously stated, this section explains the benchmark models and the different measures used to assess their performance. Thus, the first paragraphs are dedicated to ODP, Mack's model, CSR and a nonparametric approach based on ANNs, while the end of this section presents the indicators used to compare and validate the reserve distribution functions estimated by the benchmark models with those simulated by the model presented in Section 3.

The first benchmark model is ODP (Renshaw & Verrall, 1998; England & Verrall, 1999). Denoting the origin year as  $i$  and the development year as  $j$ , this reserving model based on the Chain Ladder technique assumes that incremental payments,  $C_{ij}$ , follow an overdispersed Poisson distribution with a variance proportional to the mean:

$$E[C_{ij}] = \mu_{ij} \quad \text{Var}[C_{ij}] = \phi \mu_{ij} \quad (1)$$

where  $\phi$  is the parameter that determines the level of overdispersion. Even though this model assumes  $C_{ij}$  to be a positive integer, the quasi-likelihood (McCullagh & Nelder, 1989) approach allows fits the model to non-integer data, which can be either positive or negative. The bootstrapping procedure used in this study to compute a reserve distribution function with the ODP model was introduced by England and Verrall (1999) and England (2002).

Mack (1993) model, which is also based on the Chain Ladder technique, is the second benchmark. The main characteristic of this reserving model is the lack of assumptions about the underlying distribution of the payments. This is achieved by using only the first two moments:

$$E[D_{ij}] = \lambda_j D_{ij-1} \quad \text{Var}[D_{ij}] = \sigma_j^2 D_{ij-1} \quad (2)$$

where  $\lambda_j$  and  $\sigma_j^2$  refer to the parameters to be estimated, and  $D_{ij}$  is the cumulative payment. As with the ODP model, a bootstrapping procedure is used to calculate the reserve distribution function with Mack's model.

The third benchmark model is CSR, a Bayesian approach introduced by Meyers (2015). The default calibration and prior distributions suggested by this author will be used in this study:

- $\alpha_i \sim N(\ln P_i + \log \text{elr}, \sqrt{10})$ , where  $\log \text{elr} \sim U(-1, 0.5)$  and  $P_i$  are the premiums by accident year.
- $\beta_j \sim U(-5, 5)$  for  $j = 1, \dots, J-1$ . In the last development year,  $\beta_J = 0$ .
- $\mu_{ij} = \alpha_i + \beta_j(1 - \gamma)^{i-1}$ , where  $\gamma \sim N(0, 0.025)$ .
- Each  $\sigma_j = \sum_{i=j}^J a_i$ , where  $a_i \sim U(0, 1)$ .

Taking into consideration the aforementioned distributions and parameters, the cumulative payments simulated by the CSR model follow a log-normal distribution,  $D_{ij} \sim LN(\mu_{ij}, \sigma_j)$ , subject to the constraint  $\sigma_1 > \sigma_2 > \dots > \sigma_J$ .

To analyse the improvement in the performance due to the stacking procedure that is presented in Section 3, the last benchmark model to be introduced is an individual ANN. The inputs and characteristics (hidden layers, activation functions, etc.) of this algorithm will be the same as those used for the ANN included within the first level of the Stacked-ANN. Additionally, the log-normal procedure to obtain the reserve variability is the same as that for the Stacked-ANN model. To avoid repeating content, refer to Section 3 for further details about the characteristics of the ANN used as a benchmark.

Once the four benchmark models are explained, the different measures selected to compare the performance of the Stacked-ANN with the aforementioned reserving models are presented. Insurance regulations such as the Solvency II Directive and Swiss Solvency Test ask the general insurance companies to evaluate their expected reserves and potential deviations from these central scenarios. Thus, the error of the estimated reserves will be computed in order to compare the performance of the different models. As several triangles with different levels of payments are used during this study, the measure for evaluating the reserves is

$$\%RMSE(R^t) = \frac{\sqrt{\sum_{k=1}^K (\hat{R}_{k,\mu}^t - R_k^t)^2 / K}}{\sum_{k=1}^K R_k^t} * 100 = \frac{RMSE(R^t)}{\sum_{k=1}^K R_k^t} * 100 \quad (3)$$

where  $K$  is the total number of triangles,  $t$  is the calendar year when the reserves are evaluated,  $\hat{R}_{k,\mu}^t$  is the reserve predicted by the reserving model using the triangle  $k$  and  $R_k^t$  the reserves that were actually observed for that triangle. As it can be derived from the former expression, the aim of this error measure is the evaluation of the weight of the root mean squared error over the total reserves. To understand the model's performance, this error measure will also be calculated for the next year's payments ( $\%RMSE(P^{t+1})$ ) and the ultimate loss cost ( $\%RMSE(U^t)$ ).

In addition to the aforementioned error measures, the reserving risk (RR) per unit of reserve derived from the use of the different stochastic reserving models will be analysed. As previously stated, the models are going to be fitted to several triangles, so the average of the former ratio is taken as a risk measure:

$$\text{Ratio}(RR_{1-\alpha}^t) = \frac{\sum_{k=1}^K (\hat{R}_{k,1-\alpha}^t - \hat{R}_{k,\mu}^t) / \hat{R}_{k,\mu}^t}{K} = \frac{\sum_{k=1}^K RR_{1-\alpha}^t / \hat{R}_{k,\mu}^t}{K} \quad (4)$$

where  $\hat{R}_{k,\mu}^t$  is the mean and  $\hat{R}_{k,1-\alpha}^t$  is the percentile  $1 - \alpha$  of the estimated reserve distribution function of the company  $k$ . A deeper evaluation of the variation estimated by the different stochastic models is carried out by calculating the standard deviation per unit of reserve:

$$\text{Ratio}(\sigma) = \frac{\sum_{k=1}^K \sigma(\hat{R}_k^t) / \hat{R}_{k,\mu}^t}{K} \quad (5)$$

Finally, in order to check the adequacy of the reserving risk calculated for the different companies, the Kupiec (1995) test is applied in order to verify if the number of excesses is aligned with the selected confidence level. The empirical results of the test and measures are collected in Section 4.

### 3. Stochastic reserving model based on the stacking algorithm approach

This section is divided into several subsections in order to sequentially explain the proposed reserving model. In addition, Fig. 1 presents the model architecture in order to support the explanation.

#### 3.1. Model inputs

Before estimating the different reserving models within the first level of the Stacked-ANN model, the database used, as well as the response and explicative variables for fitting the algorithms within this level, need to be defined.

The lower and upper triangles needed to fit and validate the models are obtained from Schedule P of the NAIC Annual Statement. This database (available on the CAS website) was collected from property and casualty insurers that underwrite business in the US, and it contains both paid and incurred losses (net of rein-

surance) of the accident years from 1988 to 1997. Ten development years are available for every accident year. In addition to loss data, gross and net premiums by accident year are also reported in the database.

In this paper, the different reserving models will be fitted to 200 loss triangles from NAIC Schedule P, 50 from each of the following lines of business: Commercial Auto (CA), Private Passenger Auto Liability (PA), Workers' Compensation (WC) and Other Liability (OL). As pointed out by Meyers (2015), selecting triangles from insurers who made significant changes in business operations is one of the main mistakes that could be made with NAIC Schedule P data. The coefficient of variation of the net premiums and the net/gross premium ratio should be appropriate indicators of changes in business operations, so this author selected insurers that minimize the aforementioned metrics. The triangles selected by Meyers (2015) are used in this research in order to avoid the former issue and ensure comparability with other studies.

With regard to the explanatory variables, as with other non-parametric reserving models based on Generalized Additive Models (Hastie & Tibshirani, 1986; England & Verrall, 2002) or RNN (Kuo, 2018), accident  $i$  and development  $j$  years were selected to be the inputs of the first-level algorithms. Both were initialized as one and then scaled to range  $[0, 1]$  (hereinafter  $AY_i^*$  and  $DY_j^*$ ) in order to facilitate the fitting of the algorithms (Hastie & Tibshirani, 2009).

The response variable of these algorithms is the scaled cumulative payments  $D_{ij}^*$ . Depending on the data availability and the characteristics of the portfolio to be modelled, different exposure measures can be selected to scale  $D_{ij}$ . In this paper, net premiums  $P_i$  play the role of exposure measure, as this is the most relevant option between the variables available in the database.

Loss triangles are a representation of payments over time by accident or underwriting year. Thus, the training and optimization of the deep learning algorithms within the Stacked-ANN model

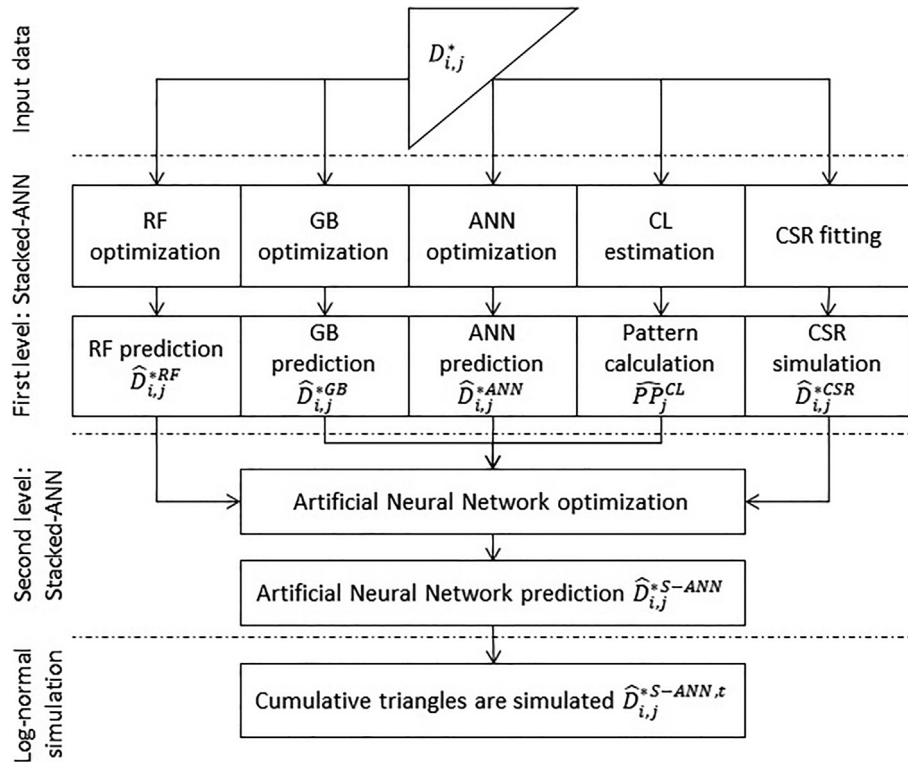


Fig. 1. Stacked-ANN model structure.

architecture need to take into consideration that loss triangles are composed of temporal series. Accordingly, the last diagonal is selected as a test set because it contains the most updated information, while the rest of the triangle is used for fitting the algorithms (see Fig. 2).

During the optimization process, different configurations of the algorithms are fitted with the training data. To obtain the best configuration, the test set is predicted, and the root mean squared error of every option is computed. Finally, the configuration that minimizes the former test error is selected.

### 3.2. First level: Individual models

The first level of the Stacked-ANN model consists of a Chain Ladder, CSR, and three algorithms whose inputs were described in Section 3.1. It is worth mentioning that as ODP and Mack's model are based on the Chain Ladder technique, the Stacked ANN model incorporates the core rationale behind these stochastic reserving models. The machine learning algorithms (RF, GB and ANN) fitted at this step are explained in the following paragraphs and will be optimized by applying a grid search to some hyperparameters and by measuring the test error. Additionally, at the end of this subsection, the Chain Ladder and CSR hypothesis are integrated within the Stacked-ANN model architecture.

The Random Forest (RF) algorithm introduced by Breiman (2001) averages  $B$  different regression trees. In every fitted tree, the explanatory variables and data points used during the training are randomly selected. Therefore, the formal expression to predict the scaled cumulative payments is:

$$\hat{D}_{ij}^{*RF} = \frac{\hat{D}_{ij}^{RF}}{P_i} = \frac{\sum_{b=1}^B T_b(X)}{B} \quad (6)$$

$T_b$  represents the  $b$ -th regression tree fitted and  $X$  the selected subset of  $AY_i^*$  and  $DY_j^*$  to fit  $T_b$ . During the estimation process, the hyper-parameters optimized are the number of variables randomly selected,  $N$ , and the minimum number of observations to be kept in the terminal nodes of every fitted tree,  $Obs_{RF}$ .

The second algorithm within the first level is Gradient Boosting (GB) with regression trees (Friedman, 2000). In this case, the gradient is minimized by sequentially fitting  $B$  regression trees. The subset of data to be used during the estimation process of every tree is also randomly selected. The expression to obtain the predicted scaled cumulative payments is

$$\hat{D}_{ij}^{*GB} = \frac{\hat{D}_{ij}^{GB}}{P_i} = \hat{f}_{B-1}(X) + \delta_{GB} T_B(X) \quad (7)$$

$\hat{f}_{B-1}(X)$  represents the function obtained after adding sequentially  $B - 1$  regression tree models and,  $\delta_{GB}$  is the learning rate. The hyperparameter selected to be optimized during the training process is the minimum number of observations to be kept in the terminal nodes of every fitted tree,  $Obs_{GB}$ . Regarding the hyperparameters, it is worth mentioning that the learning rate,  $\delta_{GB}$ , is set to 0.01.

The last algorithm of the first layer is an Artificial Neural Network (ANN) (McCulloch & Pitts, 1943). Following the notation pro-

vided by Bishop (2006) and taking into consideration that the feed-forward ANN used in this paper is composed of 2 hidden layers with 5 neurons each, the formal expression to obtain the predictions can be defined as follows:

$$\begin{aligned} \hat{D}_{ij}^{*ANN} &= \hat{D}_{ij}^{ANN} / P_i \\ &= h^{(3)} \left( \sum_{k=1}^5 w_{1,k}^{(3)} h^{(2)} \left( \sum_{j=1}^5 w_{k,j}^{(2)} h^{(1)} \left( \sum_{i=1}^2 w_{ji}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right) + w_{1,0}^{(3)} \right) \end{aligned} \quad (8)$$

where  $h^{(n)}$  is the activation function associated with layer  $n$ ,  $w_{z,v}^{(n)}$  is the  $v$ -th weight associated with the neuron  $z$  inside layer  $n$ , and  $x_i$  refers to the  $i$ -th input variable of the database composed of two explanatory variables, the scaled accident ( $AY_i^*$ ) and development year ( $DY_j^*$ ). The percentage of dropout regularization  $\theta$  is the hyperparameter to be optimized by applying a grid search and measuring the test error. As with the other algorithms, upper triangle predictions will be used as input within the second level of the architecture.

In addition to the three aforementioned algorithms, Chain Ladder assumptions are incorporated in the model architecture. To do so, the development factors of the Chain Ladder technique are used as an input in the second level of the Stacked-ANN model:

$$\hat{\lambda}_j^{*CL} = \frac{\sum_{i=1}^{n-j-1} D_{ij}^*}{\sum_{i=1}^{n-j-1} D_{ij-1}^*} \quad (9)$$

where  $\{\hat{\lambda}_j^{*CL} : j = (2, 3, \dots, J)\}$ . Although the Chain Ladder methodology does not produce any parameters for  $j = 1$ , the second-level algorithm needs a value for  $j = 1$ . Thus, within the Stacked-ANN methodology, it is assumed that  $\hat{\lambda}_1^{*CL} = 1$ .

Finally, CSR methodology (Meyers, 2015) is integrated. To achieve this, 10,000 MCMC simulations are produced within the first level of the Stacked-ANN model. Then, the expected scaled cumulative payments of the upper triangle arising from the aforementioned simulations are used as input in the algorithm within the second level of the Stacked-ANN model:

$$\hat{D}_{ij}^{*CSR} = \frac{\sum_{k=1}^{10,000} \hat{D}_{ijk}^{CSR} / P_i}{10,000} \quad (10)$$

### 3.3. Second level: Stacking algorithm

As previously stated, the inputs of this level are the scaled cumulative payments predicted by the algorithms named in Section 3.2 (RF, GB and ANN), the development factors based on the Chain Ladder technique and the expected scaled cumulative payments simulated by the CSR model. On the other hand, the output of the ANN within the second level and the Stacked-ANN are the cumulative payments  $\hat{D}_{ij}^{*S-ANN}$  by accident and development year.

Similar to the first-level algorithms, the training and optimization processes of the ANN within this level need to recognize that loss triangles are composed of a set of time series. The most recent information of the loss triangles is the last diagonal; thus, the



Fig. 2. Train and test sets.

explicative and response variables of this diagonal are selected as a test set, while the rest of the upper triangle data is used as a training set.

Once the test and training sets are defined, the optimum configuration of the ANN needs to be obtained. To do so, the training data are used to fit ANNs with different levels of dropout regularization  $\theta$ . Then, the root mean squared error is computed by taking into consideration the predictions made by every ANN configuration. The  $\theta$  that minimizes the test error is selected.

Due to the Stacked-ANN architecture, two substeps need to be carried out in order to make the final predictions. First, the lower triangle of the first-level models need to be predicted. Second, the data predicted in the previous step are used as input of the ANN within the second layer to make the final predictions. Thus, the Stacked-ANN model tries to obtain more accurate predictions by combining different reserving models and algorithms.

Fig. 1 shows the overall Stacked-ANN architecture, and Fig. 3 provides a detailed summary of the process defined in the previous paragraphs. Technical details about the feedforward ANN fitted within this level of the Stacked-ANN model are presented below:

- It contains two hidden layers with 5 neurons each. The sigmoid activation function was selected for all neurons within the hidden layers while the linear activation function was used in the output layer, which is composed of one neuron.
- The selected optimization algorithm is Adaptive Moment Estimation (ADAM), which was created by Kingma and Ba (2014). This method consists of a progressive adaptation of the initial learning rate, taking into consideration current and previous gradients. The default calibration proposed by the authors for the ADAM parameters is applied as  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . Thus, the ANN parameters are updated as follows:

$$\omega_t = \omega_{t-1} - \delta_{ANN} \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (11)$$

$$\hat{m}_t = \frac{\beta_1 m_{t-1} + (1 - \beta_1) g_t}{1 - \beta_1^t} \quad (12)$$

$$\hat{v}_t = \frac{\beta_2 v_{t-1} + (1 - \beta_2) g_t^2}{1 - \beta_2^t} \quad (13)$$

where  $\omega$  is the parameter to be updated and  $g_t$  the gradient in the epoch  $t$ . The initial learning rate is set to  $\delta_{ANN} = 0.01$ .

- The number of epochs is 10,000, and the batch size is equal to the length of the data used for training the ANN.
- The backward pass calculations are done according to the selection of the root mean squared error as a loss function.
- As previously stated, the percentage of dropout regularization  $\theta$  is the hyperparameter to be optimized by applying a grid search and measuring the test error.

Taking the abovementioned details into consideration, the scaled cumulative payments predicted by the Stacked-ANN model are obtained by means of the following expression:

$$\begin{aligned} \hat{D}_{ij}^{S-ANN} &= \frac{\hat{D}_{ij}^{S-ANN}}{P_i} = \hat{f}(\hat{D}_{ij}^{*RF}, \hat{D}_{ij}^{*GB}, \hat{D}_{ij}^{*ANN}, \hat{\lambda}_j^{*CL}, \hat{D}_{ij}^{*CSR}) \\ &= h^{(3)} \left( \sum_{k=1}^5 w_{1,k}^{(3)} h^{(2)} \left( \sum_{j=1}^5 w_{k,j}^{(2)} h^{(1)} \left( \sum_{i=1}^5 w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right) + w_{1,0}^{(3)} \right) \end{aligned} \quad (14)$$

### 3.4. Log-normal simulation

To compute the Kupiec test and the measures related to reserve variability (Section 2), the deviation around the central scenario predicted by the Stacked-ANN model needs to be obtained. Due to its right skewness and long tail, log-normal distribution is widely used within reserving models to derive the variability of the claims cost. Many papers used the lognormal distribution to compute this variability (see, among others, Kremer (1982), Antonio, Beirlant, Holdemakers, & Verlaak (2006), Rehman & Klugman (2009), Weke & Ratemo (2013), Meyers (2015) or more recently, Omari, Nyambura, & Wairimu (2018)).

In this study, a log-normal distribution is used to compute the reserve variability around the central scenario predicted by the Stacked-ANN. To do so, the parameters of this distribution are obtained using the aforementioned predictions and the moments method. Therefore, regardless of the distribution selected, the central scenario is that predicted by the Stacked-ANN, and thus, changing the distribution has no effect on the error measures described in Section 2. Nevertheless, changes to the log-normal hypothesis will modify the variability and, consequently, the risk measures ( $Ratio(RR_{1-\alpha}^t)$  and  $Ratio(\sigma)$ ) and the results of the Kupiec test. Below, the steps of the procedure are described:

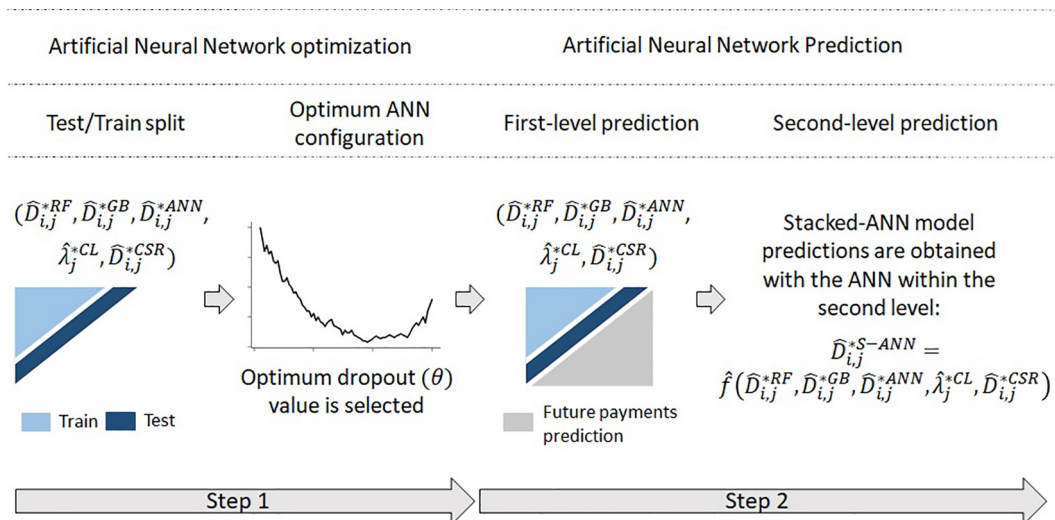


Fig. 3. Second-level structure.

- Starting with the scaled cumulative payments predicted by the Stacked-ANN ( $\hat{D}_{ij}^{S-ANN}$ ), the variance by development year is computed as follows:

$$\text{Var}[\hat{D}_{ij}^{S-ANN}] = \frac{\sum_{i=1}^n (\hat{D}_{ij}^{S-ANN} - E[\hat{D}_{ij}^{S-ANN}])^2}{n-1} \quad (15)$$

where  $n$  refers to the total number of accident years and  $E[\hat{D}_{ij}^{S-ANN}]$  is the mean of the scaled cumulative payments by development year.

- By using the method of the moments and values calculated in the previous step, the parameters of the log-normal distribution are obtained:

$$\hat{\mu}_{ij}[\hat{D}_{ij}^{S-ANN}] = \ln \left( \frac{E[\hat{D}_{ij}^{S-ANN}]^2}{\sqrt{\text{Var}[\hat{D}_{ij}^{S-ANN}] + E[\hat{D}_{ij}^{S-ANN}]^2}} \right) \quad (16)$$

$$\hat{\sigma}_{ij}^2[\hat{D}_{ij}^{S-ANN}] = \ln \left( 1 + \frac{\text{Var}[\hat{D}_{ij}^{S-ANN}]}{E[\hat{D}_{ij}^{S-ANN}]^2} \right) \quad (17)$$

- For  $t = (1, 2, \dots, T)$ :

- A triangle is generated by sampling random values from the following distribution function:

$$\hat{C}_{ij}^{S-ANN,k} \sim \text{LN}(\hat{\mu}_{ij}[\hat{D}_{ij}^{S-ANN}], \hat{\sigma}_{ij}^2[\hat{D}_{ij}^{S-ANN}]).$$

- The final simulated values,  $\hat{C}_{ij}^{S-ANN,k}$ , are obtained by removing the scaling. Hence, the scaled payments obtained in the previous step are multiplied by  $P_i$ .

## 4. Results

In this section, the data used, the fitting process and a final comparison between the Stacked-ANN and the benchmark models are shown.

### 4.1. Data and fitting of the Stacked-ANN

As stated in Section 3.1, the upper and lower triangles required to fit and validate the models are obtained from Schedule P of the NAIC Annual Statement. This database contains the losses, reserves and premiums from 1988 until 1997 of different property and casualty insurers that underwrite business in the United States.

Meyers (2015) indicated that one of the main mistakes with the NAIC Schedule P data is selecting triangles from insurers that made significant changes in their businesses. Meyers used the coefficient of variation of the net premiums and the net-on-gross ratio to select 50 triangles of each of the following lines of business: Commercial Auto (CA), Private Passenger Auto Liability (PA), Workers' Compensation (WC) and Other Liability (OL). This triangle selection was also used in this paper in order to ensure comparability with other studies that take as a reference the selection made by

Meyers (2015). For further details about the data used to fit the Stacked-ANN, refer to Section 3.1.

Once the data have been presented, the subsection focuses on the fitting of the Stacked-ANN. The first level of the proposed model is composed of three individual algorithms (RF, GB and ANN), the CSR reserving model and the development factors derived from the use of the Chain Ladder technique. The second level is composed of an ANN. As pointed out in Sections 3.2 and 3.3, the optimum hyperparameters of the algorithms within the first and second levels are obtained for each triangle using a grid search. Table 1 lists the minor differences across the lines of business in the means of the 50 optimum hyperparameters obtained for each algorithm.

As previously stated, the development factors ( $\hat{\lambda}_j^{CL}$ ) obtained by applying the Chain Ladder technique to  $D_{ij}^*$  are used as input for the ANN included within the second level of the Stacked-ANN model. These values are calculated for each triangle. Table 2 presents the means of the development factors by line of business.

With regard to the three algorithms of the first layer and the Chain Ladder technique, the CSR model is also incorporated in the Stacked-ANN architecture by means of inputting  $\hat{D}_{ij}^{CSR}$  in the second-level algorithm. This Bayesian reserving model is fitted to every single triangle. Tables 3 and 4 list the means of the CSR parameters by line of business.

Table 3, which is focused on the parameters needed to calculate the mean of the cumulative payments, presents positive  $\gamma$  and negative  $\beta_j$  for every line of business with the unique exception of CA, where  $\beta_6$ ,  $\beta_7$ ,  $\beta_8$  and  $\beta_9$  are positive. According to the model definition, the claims settlement speed increases when  $\beta_j < 0$  and  $\gamma > 0$ . This common trend across the different lines of business about the claim settlement rate of the NAIC Schedule P data was already observed by Meyers (2015).

The comparison of the CSR deviation by development year presented in Table 4 reveals that OL is the most volatile portfolio, while PA has the most stable reserves. For CA and WC, the reserve variability estimated by this Bayesian reserving model is quite

**Table 2**  
Mean of the development factors by line of business.

Development factors	CA	PA	WC	OL
$\hat{\lambda}_1^{CL}$	1.89	1.77	2.21	6.66
$\hat{\lambda}_2^{CL}$	1.35	1.22	1.29	1.90
$\hat{\lambda}_3^{CL}$	1.16	1.10	1.13	1.33
$\hat{\lambda}_4^{CL}$	1.08	1.06	1.07	1.18
$\hat{\lambda}_5^{CL}$	1.04	1.03	1.04	1.10
$\hat{\lambda}_6^{CL}$	1.02	1.01	1.02	1.04
$\hat{\lambda}_7^{CL}$	1.00	1.01	1.02	1.02
$\hat{\lambda}_8^{CL}$	1.01	1.00	1.01	1.02
$\hat{\lambda}_9^{CL}$	1.00	1.00	1.01	1.01
$\hat{\lambda}_{10}^{CL}$	1.00	1.00	1.00	1.00

Source: own elaboration.

**Table 1**  
Mean of the optimum hyperparameters by line of business.

Line of business	RF first level	GB first level	ANN first level	ANN second level
CA	$Obs_{RF} = 2.04; N = 1.94$	$Obs_{GB} = 4.38$	$\theta = 0.10$	$\theta = 0.09$
PA	$Obs_{RF} = 2.66; N = 1.86$	$Obs_{GB} = 4.22$	$\theta = 0.07$	$\theta = 0.06$
WC	$Obs_{RF} = 1.78; N = 1.68$	$Obs_{GB} = 3.66$	$\theta = 0.13$	$\theta = 0.12$
OL	$Obs_{RF} = 1.50; N = 1.82$	$Obs_{GB} = 4.24$	$\theta = 0.12$	$\theta = 0.12$

Source: own elaboration.

**Table 3**CSR parameters by line of business:  $D_{ij}$  mean.

CSR parameter	CA	PA	WC	OL	CSR parameter	CA	PA	WC	OL
$\alpha_1$	7.094	8.959	8.423	6.162	$\beta_1$	-1.235	-0.987	-1.447	-2.446
$\alpha_2$	7.166	9.047	8.612	6.269	$\beta_2$	-0.514	-0.400	-0.626	-1.332
$\alpha_3$	7.171	9.148	8.779	6.330	$\beta_3$	-0.229	-0.198	-0.322	-0.709
$\alpha_4$	7.280	9.143	8.666	6.309	$\beta_4$	-0.085	-0.097	-0.178	-0.363
$\alpha_5$	7.348	9.201	8.644	6.334	$\beta_5$	-0.003	-0.042	-0.089	-0.173
$\alpha_6$	7.347	9.282	8.537	6.515	$\beta_6$	0.039	-0.017	-0.057	-0.079
$\alpha_7$	7.554	9.374	8.595	6.480	$\beta_7$	0.060	-0.008	-0.041	-0.045
$\alpha_8$	7.540	9.389	8.514	6.327	$\beta_8$	0.028	-0.001	-0.029	-0.030
$\alpha_9$	7.494	9.464	8.543	6.543	$\beta_9$	0.012	-0.001	-0.013	-0.014
$\alpha_{10}$	7.556	9.492	8.500	6.327	$\gamma$	0.021	0.008	0.016	0.028

Source: own elaboration.

**Table 4**CSR parameters by line of business:  $D_{ij}$  STD.

CSR Parameter	CA	PA	WC	OL
$\sigma_1$	0.303	0.028	0.236	0.771
$\sigma_2$	0.176	0.011	0.164	0.488
$\sigma_3$	0.109	0.007	0.117	0.327
$\sigma_4$	0.079	0.004	0.090	0.229
$\sigma_5$	0.063	0.003	0.069	0.164
$\sigma_6$	0.052	0.002	0.052	0.120
$\sigma_7$	0.043	0.002	0.037	0.087
$\sigma_8$	0.035	0.001	0.025	0.061
$\sigma_9$	0.026	0.001	0.016	0.038
$\sigma_{10}$	0.014	0.001	0.008	0.019

Source: own elaboration.

similar, and it is located at an intermediate point between the OL and PA lines of business.

#### 4.2. Comparison against benchmark models

Once the Stacked-ANN reserving model is fitted, its performance is compared with the benchmark models explained in Section 2 (ODP, Mack, CSR and an individual ANN).

Table 5 lists the %RMSEs associated with reserves  $R^t$ , next year payments  $P^{t+1}$  and ultimate losses  $U^t$  by line of business and reserving model. For further details about the measures presented in the table, refer to Section 2.

The results obtained by using the different reserving models are summarized as follows:

- The Stacked-ANN model outperforms the individual ANN. The proposed architecture is empirically more accurate because it can learn from the reserving models (Chain Ladder and CSR)

**Table 5**

%RMSE by line of business and reserving model.

Error measure	Line of business	ODP	Mack's model	CSR	ANN	Stacked ANN
%RMSE( $R^t$ )	CA	0.896%	0.896%	0.534%	1.768%	0.739%
%RMSE( $P^{t+1}$ )	CA	0.668%	0.669%	0.573%	1.775%	0.876%
%RMSE( $U^t$ )	CA	0.170%	0.171%	0.102%	0.337%	0.141%
%RMSE( $R^t$ )	PA	1.012%	1.004%	0.823%	5.006%	0.254%
%RMSE( $P^{t+1}$ )	PA	1.290%	1.286%	0.258%	1.900%	0.320%
%RMSE( $U^t$ )	PA	0.131%	0.131%	0.107%	0.651%	0.033%
%RMSE( $R^t$ )	WC	1.295%	1.286%	1.751%	1.943%	1.058%
%RMSE( $P^{t+1}$ )	WC	0.887%	0.880%	1.531%	1.525%	0.676%
%RMSE( $U^t$ )	WC	0.222%	0.221%	0.301%	0.333%	0.182%
%RMSE( $R^t$ )	OL	5.274%	5.086%	3.153%	5.725%	0.722%
%RMSE( $P^{t+1}$ )	OL	2.216%	2.102%	5.528%	0.268%	1.095%
%RMSE( $U^t$ )	OL	1.760%	1.709%	1.056%	1.918%	0.242%

Source: own elaboration.

and machine learning algorithms (RF, GB and ANN) included within the first level of the Stacked-ANN, while the individual ANN must base its training only on the origin data ( $AY_i^*$  and  $DY_j^*$ ) without taking advantage of other models that are able to capture different patterns and characteristics.

- As they are based on Chain Ladder assumptions, the mean of the distributions generated by ODP and Mack's model should converge to the values obtained by applying the deterministic approach of the Chain Ladder technique. Consequently, the error measures observed in Table 5 for these two stochastic reserving models are almost the same. The table also reveals that ODP and Mack are less accurate than the Stacked-ANN model in most cases. %RMSE( $P^{t+1}$ ) of CA is a unique category in which the benchmark models based on the Chain Ladder technique are more accurate than the proposed methodology.
- Regarding the comparison between Stacked-ANN and CSR,  $R^t$  and  $U^t$  of PA, WC and OL estimated by the proposed model are significantly more accurate than those obtained when using the Bayesian model. Additionally, %RMSE( $P^{t+1}$ ) of the Stacked-ANN model is lower in WC and OL. Thus, in the majority of cases, the CSR model is outperformed by the proposed methodology.

To enhance the analysis of the results presented in Table 5, Fig. 4 shows the %RMSE( $R^t$ ) by line of business and volume of reserves. First, the companies were classified in four different groups taking into consideration the volume of reserves and the quartiles associated with the distribution. Then, the error rate of each reserving model was computed by line of business. The former calculation was carried out without making any distinction between lines of business.

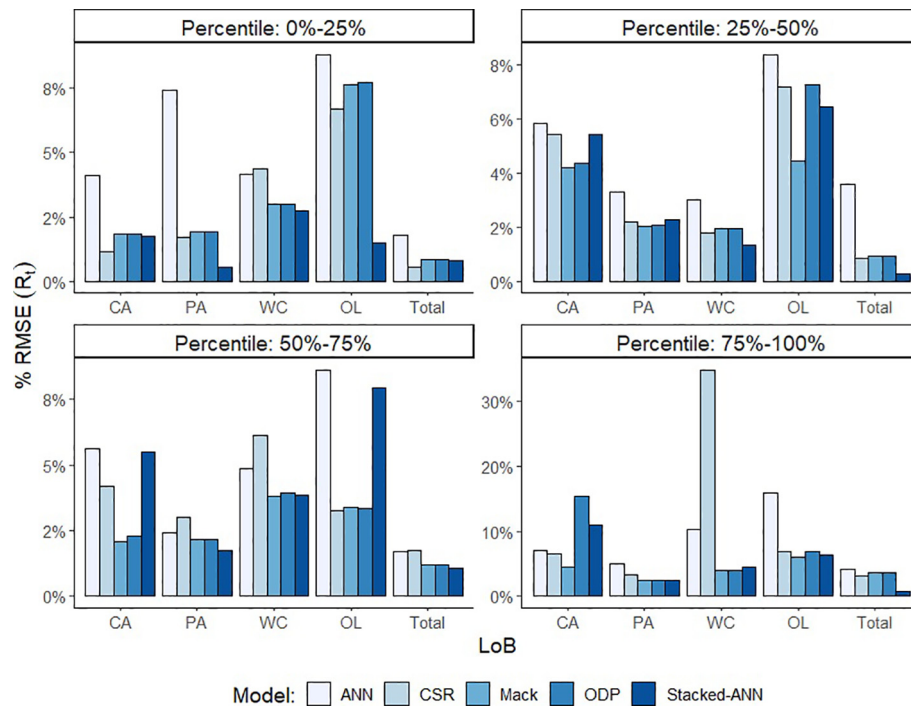


Fig. 4. %RMSE( $R_t^i$ ) by line of business and volume of reserves.

The results of the aforementioned figure reveal that when no distinction between lines of business is made, the Stacked-ANN architecture outperforms the rest of the reserving models regardless of the company size. This difference is especially relevant for those companies with a higher level of reserves, whose results are collected in the graph labelled 'Percentile: 75%–100%'. As expected, some fluctuations in the performance of the models are observed when the results are analysed by line of business and volume of reserves. Nonetheless, the error rate of the Stacked-ANN tends to be lower than the rest of the benchmark models.

In accordance with the reasons explained within the former paragraphs, it can be concluded that the Stacked-ANN model takes advantage of the different characteristics of several reserving models and machine learning algorithms, leading to a more flexible and precise architecture in most of the cases.

In addition to the error analysis, the risk measures ( $Ratio(RR_{1-\alpha}^t)$  and  $Ratio(\sigma)$ ) and the p-values of the Kupiec test obtained by using each reserving model are compared. Before examining the results, it is important to point out that Mack's model does not make any assumptions about the payment distribution, ODP assumes that

incremental payments follow an overdispersed Poisson distribution, and CSR, ANN and Stacked-ANN presume that cumulative payments are log-normally distributed. The hypothesis taken regarding the payments impact the distribution shape and consequently the risk measures. Therefore, in this case, ODP and Mack's model are not going to converge like they did in the central scenario.

The  $Ratio(RR_{1-\alpha}^t)$  and the p-values collected in Table 6 evaluate the 99.5 percentile ( $\alpha = 0.005$ ) of the reserve distribution function, which is the confidence level set up by Solvency II to calculate the risk of the insurance companies. The results of this table are summarized below:

- According to the results of the Kupiec test, the Stacked-ANN generates an adequate risk assessment for every line of business. It is worth mentioning that when compared with the individual ANN, the empirical results show that the stacking process not only improves the error rate but also allows for the generation of more appropriate distribution functions using the same simulation approach (presented in Section 3.4). With

Table 6  
Risk measures by line of business and reserving model.

Risk measures	Line of business	ODP	Mack's model model	CSR	ANN	Stacked ANN
$Ratio(RR_{0.995}^t)$	CA	1.936	1.460	2.776	1.387	1.884
$Ratio(\sigma)$	CA	2.561	0.461	0.681	0.456	0.642
Kupiec p-value	CA	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$< 0.05$	$\geq 0.05$
$Ratio(RR_{0.995}^t)$	PA	0.544	0.373	0.918	0.783	0.888
$Ratio(\sigma)$	PA	0.277	0.135	0.270	0.279	0.332
Kupiec p-value	PA	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$
$Ratio(RR_{0.995}^t)$	WC	2.525	0.691	1.797	2.194	2.149
$Ratio(\sigma)$	WC	1.273	0.245	0.474	0.682	0.717
Kupiec p-value	WC	$< 0.05$	$< 0.05$	$< 0.05$	$< 0.05$	$\geq 0.05$
$Ratio(RR_{0.995}^t)$	OL	7.506	3.287	4.843	2.315	3.522
$Ratio(\sigma)$	OL	6.275	1.217	1.119	0.690	1.099
Kupiec p-value	OL	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$

Source: own elaboration.

regard to the comparison between the Stacked-ANN and the rest of benchmark models, the Kupiec test reveals that CSR, ODP and Mack's model do not produce appropriate risk measures for WC, while the proposed methodology passes the test.

- Intuitively, the duration of the liabilities should have a close relation with  $Ratio(RR_{0.995}^t)$  and  $Ratio(\sigma)$ : the longer the duration, the higher is the uncertainty around each economic unit of reserve. The development factors based on the Chain Ladder technique measure the claim settlement speed. Therefore, they can be considered a good indicator of the duration of liabilities. A development factor at year  $t$ ,  $\lambda_t$ , means that the  $t + 1$  cumulative payment is  $\lambda_t$  times the cumulative claims settled at  $t$ . Consequently, high development factors indicate a long duration, while low values reflect a high settlement speed. According to Table 2, OL is the line of business with the highest duration, while PA has the lowest. CA and WC, whose durations are in a similar range, are located at an intermediate point between PA and OL. As can be observed in Table 6, this intuition about the relation between the duration and reserve uncertainty is followed by the Stacked-ANN and benchmark models.
- In general, the  $Ratio(RR_{0.995}^t)$  and  $Ratio(\sigma)$  by line of business are similar across the different reserving models. The two main exceptions are the risk measures of ODP for OL and Mack for WC. The high values observed in the ODP estimations for OL are due to two companies whose  $RR_{1-\alpha}^t/\hat{R}_{k,\mu}^t$  ratios are higher than 60, while in the second case, Mack's model systematically underestimates the variability of the payments, leading to lower values compared with the rest of the models and an inadequate risk assessment according to the results of the Kupiec test. The  $Ratio(RR_{0.995}^t)$  and  $Ratio(\sigma)$  of the Stacked-ANN are in line with the majority of the benchmark models, and no extremely high/low risk measures are observed in Table 6.

### 4.3. Sensitivity analysis of the number of hidden layers

As explained in Section 3, the ANNs included within the proposed Stacked-ANN architecture are composed of two hidden layers, each with five neurons. To analyse the impact of the ANN complexity (Cybenko, 1989; Hornik et al., 1989; Hornik, 1991; Leshno et al., 1993, among others, introduced the theoretical framework to analyse the approximation capabilities of neural networks) on the predictive power of the Stacked-ANN model, a sensitivity analysis of the number of hidden layers was carried out. Thus, Table 7 compares the configuration selected for the Stacked-ANN model in this paper with two alternative configurations: ANNs composed of one and three hidden layers with five neurons each.

**Table 7**  
Sensitivity analysis of the number of hidden layers.

Hidden layers	Line of business	%RMSE( $R^t$ )	%RMSE( $P^{t+1}$ )	%RMSE( $U^t$ )
1	CA	0.840%	0.766%	0.160%
2	CA	0.739%	0.876%	0.141%
3	CA	0.780%	0.643%	0.149%
1	PA	2.512%	3.507%	0.326%
2	PA	0.254%	0.320%	0.033%
3	PA	0.231%	0.115%	0.030%
1	WC	1.398%	1.296%	0.240%
2	WC	1.058%	0.676%	0.182%
3	WC	1.140%	1.030%	0.196%
1	OL	1.419%	1.145%	0.475%
2	OL	0.722%	1.095%	0.242%
3	OL	0.613%	0.794%	0.205%

Source: own elaboration.

Two main conclusions can be drawn from the results obtained. First, the high level of error of the one hidden layer model demonstrates that more complexity is needed in order to properly predict general insurance reserves. The structure proposed during this study for the Stacked-ANN model (two hidden layers) performs significantly better than this first alternative in every single line of business.

Second, the performance of the three hidden layers alternative is similar to that of the suggested architecture. As no significant differences are observed, the two hidden layer structure is considered more appropriate because the three hidden layer structure adds complexity to the model without a significant improvement in the error rate.

## 5. Conclusions

This paper introduced a stochastic reserving model based on stacking different machine learning algorithms (RF, GB and ANN) and reserving models (Chain Ladder and CSR). The predictive power and reserve volatility of the proposed approach, named Stacked-ANN, were compared with stochastic reserving models based on the Chain Ladder technique (ODP and Mack's model), an individual ANN and CSR, which is a Bayesian loss reserving model.

Three main conclusions were drawn. First, a comparison of the Stacked-ANN with the individual ANN revealed that the predictions of the reserves  $R^t$ , next year payments  $P^{t+1}$  and ultimate losses  $U^t$  made by machine learning algorithms were improved by applying the proposed stacking procedure. The hybrid architecture learns patterns and characteristics from several algorithms and reserving models, resulting in a more flexible and accurate model than an individual ANN, whose inputs for training are limited to the original data.

Second, the empirical results indicated that the Stacked-ANN model is more precise than CSR and the most widely used stochastic reserving models based on the Chain Ladder technique (ODP and Mack's model). In particular, the  $R^t$  and  $U^t$  predictions made by the Stacked-ANN were more precise than those of ODP and Mack's model in all the lines of business analysed, while the Bayesian model (CSR) was outperformed by the proposed architecture in three out of four lines of business. It is important to remark that in Other Liability (OL), which is a line of business with a longer duration and therefore a portfolio where the importance of an accurate reserves estimation is especially relevant, the error of the models based on Chain Ladder or Bayesian statistics was more than four times the error of the Stacked-ANN. Therefore, it can be concluded that machine or deep learning techniques can be used to improve the performance of the traditional reserving techniques based on Bayesian statistics or the Chain Ladder.

With regard to accuracy, it is worth mentioning that the proposed structure of the ANNs (two hidden layers) within the Stacked-ANN model seems to be the optimal configuration according to the empirical results. On the one hand, the error increased significantly when the number of hidden layers is reduced to one. On the other hand, the results demonstrated that increasing the number of hidden layers does not have an impact on the accuracy. Thus, increasing the complexity of the ANNs by up to three hidden layers will extend the training phase without making any significant improvement in the error.

Third, the results of a Kupiec test revealed that the risk estimation made by the Stacked-ANN can be considered as appropriate in all lines of business analysed, while the rest of the benchmark models failed the test at least once. In particular, CSR, ODP and Mack's model were unable to produce an appropriate p-value for the Kupiec test in the Workers' Compensation (WC) business, while the individual ANN failed the test in Commercial Auto (CA) and, as with the previous models, in Workers' Compensation. Taking into consideration that the same log-normal approach was used to obtain the reserves variability of the individual ANN and the Stacked-ANN, it must be mentioned that the stacking procedure not only increases the accuracy but also allows for the simulation of more adequate distribution functions.

The aforementioned robustness and predictive power of the Stacked-ANN compared with other reserving models suggest that further investigation should be conducted about the possible application of this model within the *actuary in the box* approach. The generation of outliers is one of the main problems when using the former methodology with Chain Ladder models. Therefore, the robustness of the Stacked-ANN can be exploited in order to improve the *actuary in the box* methodology, which is widely used to assess the fact that reserves can be insufficient to cover their runoff over a 12-month time horizon.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References




- Antonio, K., Beirlant, J., Holdemakers, T., & Verlaak, R. (2006). Log-normal mixed models for reported claims reserves. *North American Actuarial Journal*, 7, 1223–1237.
- Antonio, K., & Plat, R. (2014). Micro-level stochastic loss reserving in general insurance. *Scandinavian Actuarial Journal*, 2014, 649–669.
- Barron, A. (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 115–133.
- Baudry, M., & Robert, C. (2019). A Machine Learning approach for individual claims reserving in insurance. *Applied Stochastic Models in Business and Industry*, 1–29.
- Bishop, C. M. (2006). *Pattern recognition and machine learning (Information science and statistics)*. Berlin, Heidelberg: Springer-Verlag.
- Bornhuetter, R. L., & Ferguson, R. E. (1972). The actuary and IBNR. In *Proceedings of the Casualty Actuarial Society* (pp. 181–195).
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al. (2020). *Language models are few-shot learners*.
- Castellani, G., Fiore, U., Marino, Z., Passalacqua, L., Perla, F., Scognamiglio, S., & Zanetti, P. (2018). *An investigation of machine learning approaches in the Solvency II valuation framework*. Available at SSRN: <https://ssrn.com/abstract=3303296>.
- Celikoglu, H. (2007). A dynamic network loading process with explicit delay modelling. *Transportation Research Part C: Emerging Technologies*, 15, 279–299.
- Charpentier, A., & Pigeon, M. (2016). Macro vs micro methods in non-life claims reserving: An econometric perspective. *Risks*, 4, 12.
- Chui, C., Li, X., & Mhaskar, H. (1994). Neural networks for localized approximation. *Mathematics of Computation*.
- Chui, C., Li, X., & Mhaskar, H. (1996). Limitations of the approximation capabilities of neural networks with one hidden layer. *Advances in Computational Mathematics*.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2, 303–314.
- Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. CoRR abs/1810.04805.
- Duma, M., Twala, B., Marwala, T., Nelwamondo, F. (2011). Improving the performance of the support vector machine in insurance risk classification: A comparative study. In *Proceedings of the international conference on neural computation theory and applications (NCTA-2011)* (pp. 340–346).
- England, P. D. (2002). Addendum to analytic and bootstrap estimates of prediction errors in claims reserving. *Insurance: Mathematics and Economics*, 31.
- England, P. D., & Verrall, R. J. (1999). Analytic and bootstrap estimates of prediction errors in claims reserving. *Insurance: Mathematics and Economics*, 25(3), 281–293.
- England, P. D., & Verrall, R. J. (2002). Stochastic claims reserving in general insurance. *British Actuarial Journal*, 8, 443–544.
- England, P. D., & Verrall, R. J. (2006). Predictive distributions of outstanding liabilities in general insurance. *Annals of Actuarial Science*, 221–270.
- Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29, 1189–1232.
- Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2.
- Gabrielli, A., Richman, R., & Wüthrich, M. (2018). *Neural network embedding of the over-dispersed poisson reserving model*. Available at SSRN: <https://ssrn.com/abstract=3288454>.
- Gabrielli, A., & Wüthrich, M. (2018). An individual claims history simulation machine. *Risks*, 6, 29.
- Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63, 3–42.
- Hallinwell, L. (2009). Modeling paid and incurred losses together. *CAS E-Forum*, (Spring), 1–40.
- Happ, S., Merz, M., & Wüthrich, M. (2012). Claims development result in the paid-incurred chain reserving method. *Insurance: Mathematics and Economics*, 51, 66–72.
- Happ, S., & Wüthrich, M. (2013). Paid-incurred chain reserving method with dependence modeling. *ASTIN Bulletin*, 43, 1–20.
- Hastie, T., & Tibshirani, R. (1986). Generalized additive models. *Statistical Science*, 1(3), 297–310.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction (2nd ed.)*. Springer series in statistics. New York: Springer.
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4, 251–257.
- Hornik, K. (1993). Some new results on neural network approximation. *Neural Networks*, 6, 1069–1072.
- Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366.
- Jessen, A., Mikosch, T., & Samorodnitsky, G. (2011). Prediction of outstanding payments in a Poisson cluster model. *Scandinavian Actuarial Journal*, 2011(3), 214–237.
- Kingma, D. P., Ba, J. (2014). Adam: A method for stochastic optimization. CoRR.
- Kremer, E. (1982). IBNR claims and the two-way model of ANOVA. *Scandinavian Actuarial Journal*, 1982.
- Kuo, K. (2018). DeepTriangle: A deep learning approach to loss reserving. CoRR abs/1804.09253.
- Kupiec, P. H. (1995). Techniques for verifying the accuracy of risk measurement models. *The Journal of Derivatives*, 3(2), 73–84.
- Leshno, M., Lin, V., Pinkus, A., & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6, 861–867.
- Lopez, O., Milhaud, X., & Thérond, P. (2019). A Tree-Based algorithm adapted to microlevel reserving and long development claims. *ASTIN Bulletin*.
- Mack, T. (1991). A simple parametric model for rating automobile insurance or estimating IBNR claims reserves. *ASTIN Bulletin*, 21(1), 93–109.
- Mack, T. (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bulletin: The Journal of the International Actuarial Association*, 23(02), 213–225.
- Margraf, C., Elpidorou, V., & Verrall, R. (2018). Claims reserving in the presence of excess-of-loss reinsurance using micro models based on aggregate data. *Insurance: Mathematics and Economics*, 80, 54–65.
- Martínez-Miranda, M., Nielsen, B., & Verrall, R. (2012). Double Chain Ladder. *ASTIN Bulletin*, 42(1), 59–76.
- Martínez-Miranda, M., Nielsen, B., & Verrall, R. (2013a). Continuous Chain Ladder: Reformulating and generalizing a classical insurance problem. *Expert Systems with Applications*, 40, 5588–5603.
- Martínez-Miranda, M., Nielsen, B., & Verrall, R. (2013b). Double Chain Ladder and Bornhuetter-Ferguson. *North American Actuarial Journal*, 17, 101–113.
- Martínez-Miranda, M., Nielsen, B., Verrall, R., & Wüthrich, M. (2015). The link between classical reserving and granular reserving through double chain ladder and its extensions. *Scandinavian Actuarial Journal*, 2015, 383–405.
- McCullagh, P., & Nelder, J. (1989). *Generalized linear models (2nd ed.)*. Chapman and Hall/CRC monographs on statistics and applied probability series. Chapman & Hall.
- McCulloch, W., & Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 127–147.

- Merz, M., & Wüthrich, M. V. (2010). Paid-incurred chain claims reserving method. *Insurance: Mathematics and Economics*, 46, 568–579.
- Meyers, G. (2015). Stochastic loss reserving using Bayesian MCMC models. CAS Monograph Series, number 1. Casualty Actuarial Society..
- Nakama, T. (2011). Comparisons of single and multiple hidden layer neural networks (pp. 270–279)..
- Nigri, A., Levantesi, S., Marino, S., Scognamiglio, M., & Perla, F. (2019). A deep learning integrated lee–carter model. *Risk*, 7, 33.
- Omari, C., Nyambura, S., & Wairimu, J. (2018). Modeling the frequency and severity of auto insurance claims using statistical distributions. *Journal of Mathematical Finance*, 8, 137–160.
- Pigeon, M., Antonio, K., & Denuit, M. (2013). Individual loss reserving with the multivariate skew normal framework. *ASTIN Bulletin*, 43, 399–428.
- Pigeon, M., Antonio, K., & Denuit, M. (2014). Individual loss reserving using paid-incurred data. *Insurance: Mathematics and Economics*, 58, 121–131.
- Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78, 1481–1497.
- Posthuma, B., Cator, E., Veerkamp, W., & Van Zwet, E. (2008). Combined analysis of paid and incurred losses. CAS E-Forum Fall, 272–293..
- Quarg, G., & Mack, T. (2004). Munich Chain Ladder. *Blätter der Deutschen Gesellschaft für Versicherungs und Finanzmathematik XXVI*, 597–630..
- Ramos-Pérez, E., Alonso-González, P., & Núñez-Velázquez, J. (2019). Forecasting volatility with a stacked model based on a hybridized Artificial Neural Network. *Expert Systems with Applications*, 129, 1–9.
- Rehman, Z., & Klugman, S. (2009). Quantifying uncertainty in reserve estimates. *Variance Journal*, 4, 30–46.
- Renshaw, A. E., & Verrall, R. J. (1998). A stochastic model underlying the chain-ladder technique. *British Actuarial Journal*.
- Richman, R., & Wüthrich, M. (2018). A neural network extension of the Lee-Carter model to multiple populations. SSRN. Available at SSRN, <https://ssrn.com/abstract=3270877>..
- Sheela, D., & Deepa, S. (2013). Review on methods to fix number of hidden neurons in neural networks. *Mathematical Problems in Engineering*, 1–11.
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. CoRR abs/1712.01815..
- Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529, 484–489.
- Taylor, G., McGuire, G., & Sullivan, J. (2008). Individual claim loss reserving conditioned by case estimates. *Annals of Actuarial Science*, 3(1–2), 215–256.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. CoRR abs/1706.03762..
- Venter, G. (2008). Distribution and value of reserves using paid and incurred triangles. *CAS E-Forum*, (Fall), 348–375..
- Verrall, R. J. (2000). An investigation into stochastic claims reserving models and the chain-ladder technique. *Insurance: Mathematics and Economics*, 26(1), 91–99.
- Weke, P., & Ratemo, C. (2013). Estimating IBNR claims reserves for general insurance using archimedean copulas. *Applied Mathematical Sciences*, 7, 1223–1237.
- Wüthrich, M. (2018a). Machine learning in individual claims reserving. *Scandinavian Actuarial Journal*, 2018, 465–480.
- Wüthrich, M. (2018b). Neural networks applied to chain-ladder reserving. *European Actuarial Journal*, 8, 407–436.

**8.3 Annex III. Published Paper. Multi-Transformer: A new neural network-based architecture for forecasting S&P volatility**

## Article

# Multi-Transformer: A New Neural Network-Based Architecture for Forecasting S&P Volatility

Eduardo Ramos-Pérez <sup>1</sup> , Pablo J. Alonso-González <sup>2,\*</sup>  and José Javier Núñez-Velázquez <sup>2</sup> <sup>1</sup> Faculty of Economics, Universidad de Alcalá, Plaza de la Victoria 2, 28802 Alcalá de Henares, Madrid, Spain; ramos.perez.e@gmail.com<sup>2</sup> Economics Department, Universidad de Alcalá, Plaza de la Victoria 2, 28802 Alcalá de Henares, Madrid, Spain; josej.nunez@uah.es

\* Correspondence: pablo.alonsog@uah.es

**Abstract:** Events such as the Financial Crisis of 2007–2008 or the COVID-19 pandemic caused significant losses to banks and insurance entities. They also demonstrated the importance of using accurate equity risk models and having a risk management function able to implement effective hedging strategies. Stock volatility forecasts play a key role in the estimation of equity risk and, thus, in the management actions carried out by financial institutions. Therefore, this paper has the aim of proposing more accurate stock volatility models based on novel machine and deep learning techniques. This paper introduces a neural network-based architecture, called Multi-Transformer. Multi-Transformer is a variant of Transformer models, which have already been successfully applied in the field of natural language processing. Indeed, this paper also adapts traditional Transformer layers in order to be used in volatility forecasting models. The empirical results obtained in this paper suggest that the hybrid models based on Multi-Transformer and Transformer layers are more accurate and, hence, they lead to more appropriate risk measures than other autoregressive algorithms or hybrid models based on feed forward layers or long short term memory cells.

**Keywords:** deep learning; neural networks; risk management; stock volatility; transformer

**Citation:** Ramos-Pérez, E.;

Alonso-González, P.J.;

Núñez-Velázquez, J.J.

Multi-Transformer: A New Neural Network-Based Architecture for Forecasting S&amp;P Volatility.

*Mathematics* **2021**, *9*, 1794. <https://doi.org/10.3390/math9151794>

Academic Editor: Vicente Coll-Serrano

Received: 26 June 2021

Accepted: 26 July 2021

Published: 28 July 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Since the Financial Crisis of 2007–2008, financial institutions have enhanced their risk management framework in order to meet the new regulatory requirements set by Solvency II or Basel III. These regulations have the aim of measuring the risk profile of financial institutions and minimizing losses from unexpected events such as the European sovereign debt crisis or COVID-19 pandemic. Even though banks and insurance entities have reduced their losses thanks to the efforts made in the last years, unexpected events still cause remarkable losses to financial institutions. Thus, efforts are still required to further enhance market and equity risk models in which stock volatility forecasts play a fundamental role. Volatility, understood as a measure of an asset uncertainty [1,2], is not directly observed in stock markets. Thus, taking into consideration the stock market movements, a statistical model is applied in order to compute the volatility of a security.

GARCH-based models [3,4] are widely used for volatility forecasting purposes. This family of models is especially relevant because it takes into consideration the volatility clustering observed by [5]. Nevertheless, as the persistence of conditional variance tends to be close to zero, Refs. [6–9] developed more flexible variations of the traditional GARCH models. In addition, the models introduced by [10] (EGARCH) and [11] (GJR-GARCH) take into consideration that stocks volatility behaves differently depending on the market trend, bearish or bullish. Multivariate GARCH models were developed by [12,13]. Bollerslev et al. [14] applied the previous model to financial time series, while [15] introduced a time-varying multivariate GARCH. Dynamic conditional correlation GARCH, BEKK-GARCH and Factor-GARCH were other variants of this family that were developed by [16–18],

respectively. Finally, it is worth mentioning that, in contrast to classical GARCH, the first-order zero-drift GARCH model (ZD-GARCH) proposed by [19] is non-stationary regardless of the sign of Lyapunov exponent and, thus, it can be used for studying heteroscedasticity and conditional heteroscedasticity together.

Another relevant family is composed by stochastic volatility models. As they assume that volatility follows its own stochastic process, these models are widely used in combination with Black–Scholes formula to assess derivatives price. The most popular process of this family is the [20] model which assumes that volatility follows an Cox–Ingersoll–Ross [21] process and stock returns a Brownian motion. The main challenge of the Heston model is the estimation of its parameters. Refs. [22,23] proposed a generalized method of moments to obtain the parameters of the stochastic process, while [24–27] used a simulation approach to estimate them. Other relevant stochastic volatility processes are Hull–White [28] and SABR [29] models.

The last relevant family is composed of those models based on machine and deep learning techniques. Even though GARCH models are considered part of the machine learning tool-kit, these models are considered another different family due to the significant importance that they have in the field of stock volatility. Thus, this family takes into consideration the models based on the rest of the machine and deep learning algorithms such as artificial neural networks [30], gradient boosting with regression trees [31], random forests [32] or support vector machines [33]. Refs. [34–36] applied machine learning techniques such as Support Vector Machines or hidden Markov models to forecast financial time series. Hamid and Iqbid [37] applied Artificial Neural Networks (ANNs) to demonstrate that the implied volatility forecasted by this algorithm is more accurate than Barone–Adesi and Whaley models.

ANNs have been also combined with other statistical models with the aim of improving the forecasting power of individual ANNs. The most common approach applied in the field of stocks volatility is merging GARCH-based models with ANNs. Refs. [38–44] developed different architectures based in the previous approach for stock volatility forecasting purposes. All these authors demonstrated that hybrid models overcome the performance of traditional GARCH models in the field of stock volatility forecasting. It is also worth mentioning the contribution of [45], who combined different GARCH models with ANNs in order to compare their predictive power. ANN-GARCH models have been also applied to forecast other financial time series such as metals [46,47] or oil [48,49] volatility. Apart from the combination with GARCH-based models, ANNs have been merged with other models for volatility forecasting purposes. Ramos–Pérez et al. [50] merged ANNs, random forests, support vector machines (SVM) and gradient boosting with regression trees in order to forecast S&P500 volatility. This model overcame the performance of a hybrid model based on feed forward layers and GARCH. Vidal and Kristjanpoller [51] proposed an architecture based on convolutional neural networks (CNNs) and long-short term memory (LSTM) units to forecast gold volatility. LSTMs were also used by [52] to forecast currency exchange rates volatility. It is also worth mentioning that GARCH models have not been only merged with ANNs, Peng et al. [53] combined SVM with GARCH-based models in order to predict cryptocurrencies volatility.

The aim of this paper is to introduce a more accurate stock volatility model based on an innovative machine and deep learning technique. For this purpose, hybrid models based on merging Transformer and Multi-Transformer layers with other approaches such as GARCH-based algorithms or LSTM units are introduced by this paper. Multi-Transformer layers, which are also introduced in this paper, are based on the Transformer architecture developed by [54]. Transformer layers have been successfully implemented in the field of natural language processing (NLP). Indeed, the models developed by [55,56] demonstrated that Transformer layers are able to overcome the performance of traditional NLP models. Thus, this recently developed architecture is currently considered the state-of-the-art in the field of NLP. In contrast to LSTM, Transformer layers do not incorporate recurrence in their structure. This novel structure relies on a multi-head attention mechanism and

positional embeddings in order to forecast time series. As [54] developed Transformer for NLP purposes, positional embeddings are used in combination with word embeddings. The problem faced in this paper is the forecasting of stock volatility and, thus, the word embedding is not needed and the positional embedding has been modified as it is explained in Section 2.4.

In contrast to Transformer, Multi-Transformer randomly selects different subsets of training data and merges several multi-head attention mechanisms to produce the final output. Following the intuition of bagging, the aim of this architecture is to improve the stability and accurateness of the attention mechanism. It is worth mentioning that the GARCH-based algorithms used in combination with Transformer and Multi-Transformer layers are GARCH, EGARCH, GJR-GARCH, TrGARCH, FIGARCH and AVGARCH.

Therefore, three main contributions are provided by this study. First, Transformer layers are adapted in order to forecast stocks volatility. In addition, an extension of the previous structure is presented (Multi-Transformer). Second, this paper demonstrates that merging Transformer and Multi-Transformer layers with other models lead to more accurate volatility forecasting models. Third, the proposed stock volatility models generate appropriate risk measures in low and high volatility regimes. The Python implementation of the volatility models proposed in this paper is available in this [repository](#).

As it is shown by the extensive literature included in this section, stock volatility forecasting has been a relevant topic not only for financial institutions and regulators but also for the academia. As financial markets can suffer drastic sudden drops, it is highly desirable to use models that can adequately forecast volatility. It is also useful to have indicators that can accurately measure risk. This paper makes use of recent deep and machine learning techniques to create more accurate stock volatility models and appropriate equity risk measures.

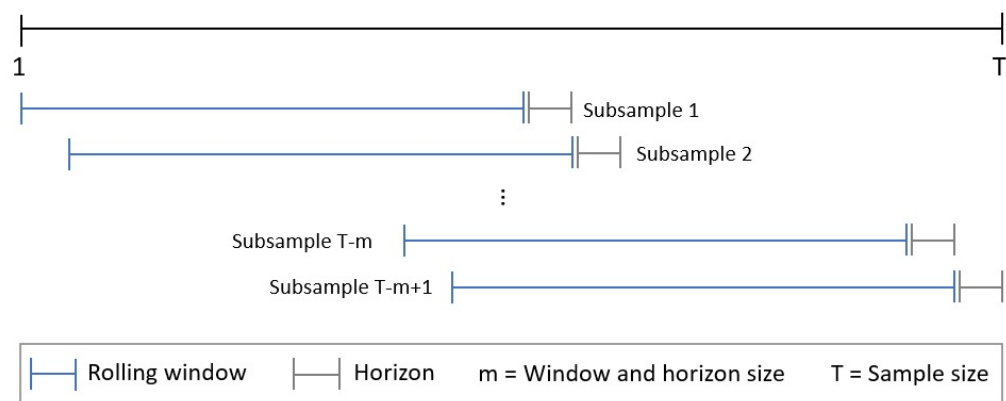
The rest of the paper is organized as follows: Section 2 describes the dataset, the measures used for validating the volatility forecasts and provides a look at the volatility models used as benchmark. Then, this section presents the volatility forecasting models proposed in this paper, which are based on Transformer and Multi-Transformer layers. As NLP Transformers need to be adapted in order to be used for volatility forecasting purposes and Multi-Transformer layers are introduced by this paper, explanations about the theoretical background of these structures are also given. The analysis of empirical results is presented in Section 3. Finally, the results are discussed in Section 4, followed by concluding remarks in Section 5.

## 2. Materials and Methods

This section is divided in five different subsections. The first one (Section 2.1) describes the data for fitting the models. The measures for validating the accuracy and value at risk (VaR) of each stock volatility model are explained in Section 2.2. Section 2.3 presents the stock volatility models and algorithms used for benchmarking purposes. Section 2.4 explains the adaptation of Transformer layers in order to be used for volatility forecasting purposes and, finally, the Multi-Transformer layers and the models based on them are presented in Section 2.5.

### 2.1. Data and Model Inputs

The proposed architectures and benchmark models are fitted using the rolling window approach (see Figure 1). This widely used methodology has been applied in finance, among others, by [57–60]. Rolling window uses a fixed sample length for fitting the model and, then, the following step is forecasted. As in this paper the window size is set to 650 and the forecast horizon to 1, the proposed and benchmark models are fitted using the last 650 S&P trading days and, then, the next day volatility is forecasted. This process is repeated until the whole period under analysis is forecasted. The periods used as training and testing set will be defined at the end of this subsection.



**Figure 1.** Rolling window methodology.

The input variables of the models proposed are the daily logarithmic returns ( $r_{t-i}$ ) and the standard deviation of the last five daily logarithmic returns:

$$\sigma_{t-1} = \sqrt{\frac{\sum_{i=1}^n (r_{t-i} - E[r])^2}{n-1}} \quad (1)$$

As Multi-Transformer, Transformer and LSTM layers are able to manage time series, a lag of the last 10 observations of the previous variables are taken into consideration for fitting these layers. Thus, the input variables are:

$$X_1 = (\sigma_{t-1}, \sigma_{t-2}, \dots, \sigma_{t-10}) \quad (2)$$

$$X_2 = (r_{t-1}, r_{t-2}, \dots, r_{t-10}) \quad (3)$$

In accordance with other studies such as [38] or [50], the realized volatility is used as response variable for the models based on ANNs;

$$Y = \hat{\sigma}_{i,t} = \sqrt{\frac{\sum_{n=0}^{i-1} (r_{t+n} - E[r_f])^2}{i-1}} \quad (4)$$

where  $E[r_f] = \sum_{n=0}^{i-1} r_{t+n} / i$  and  $i = 5$ . As shown in the previous formula, the realized volatility can be defined as the standard deviation of future logarithmic returns.

The dataset for fitting and evaluating the volatility forecasting models contains market data of S&P from 1 January 2008 to 31 December 2020. The optimum configuration of the models is obtained by applying the rolling window approach and selecting the configuration which minimizes the error (RMSE) in the period going from 1 January 2008 to 31 December 2015. The optimum configuration in combination with the rolling window methodology is applied in order to forecast the volatility contained in the testing set (from 1 January 2016 to 31 December 2020). The empirical results presented in Section 3.2 are based on the forecasts of the testing set.

## 2.2. Models Validation

This subsection presents the measures selected for validating and comparing the performance of the benchmark models with the algorithms proposed in this paper.

The mean absolute value (MAE) and the root mean squared error (RMSE) have been selected for validating the forecasting power of the different stock volatility models:

$$MAE = \sum_{t=1}^N \frac{|\sigma_{i,t} - \hat{\sigma}_{i,t}|}{N} \quad / \quad RMSE = \sum_{t=1}^N \frac{(\sigma_{i,t} - \hat{\sigma}_{i,t})^2}{N} \quad (5)$$

where  $N$  is the total number of observations.

The validation carried out by this study is not only interested on the accuracy, but also on the appropriateness of the risk measures generated by the different stock volatility forecasting models. In accordance with Solvency II Directive, 99.5% VaR has been selected as risk measure. Although Solvency II has the aim of obtaining the yearly VaR, the calculations carried out in this paper will be based on a daily VaR in order to have more data points and, thus, more robust conclusions on the performance of the different models. The parametric approach developed by [61] is used for validating the different VaR estimations. The aim of this test is accepting (or rejecting) the hypothesis that the number of VaR exceedances are aligned with the confidence level selected for calculating the risk measure. In addition to the previous test, the approach suggested by [62] is also applied in order to validate the appropriateness of VaR.

### 2.3. Benchmark Models

This subsection introduces the benchmark models used in this paper: GARCH, EGARCH, AVGARCH, GJR-GARCH, TrARCH, FIGARCH and two architectures that combine GARCH-based algorithms with ANN and LSTM, respectively. The GARCH-based algorithms will be fitted assuming that innovations,  $\epsilon_t$ , follow a Student's t-distribution. Thus, the returns generated by these models follow a conditional t-distribution [63].

The generalized autoregressive conditional heteroskedasticity (GARCH) model developed by [4] has been widely used for stock volatility forecasting purposes. GARCH(p,q) has the following expression:

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^q \alpha_i r_{t-i}^2 + \sum_{i=1}^p \beta_i \sigma_{t-i}^2 \quad / \quad \hat{r}_t = \hat{\sigma}_t \epsilon_t \quad (6)$$

where  $\omega_i$ ,  $\alpha_i$  and  $\beta_i$  are the parameters to be estimated,  $r_{t-i}$  the previous returns and  $\sigma_{t-i}^2$  the last observed volatility. As previously stated, innovations ( $\epsilon_t$ ) follow a Student's t-distribution.

The absolute value GARCH [64], AVGARCH(p,q), is similar to the traditional GARCH model. In this case, the absolute value of previous return and volatility is taken into consideration to forecast volatility:

$$\hat{\sigma}_t = \omega + \sum_{i=1}^q \alpha_i |r_{t-i}| + \sum_{i=1}^p \beta_i \sigma_{t-i} \quad (7)$$

As volatility behaves differently depending on the market tendency, models such as EGARCH, GJR-GARCH or TrGARCH were developed. EGARCH(p,q) [10] has the following expression for the logarithm of stocks volatility:

$$\log \hat{\sigma}_t^2 = \omega + \sum_{i=1}^p \alpha_i \log \hat{\sigma}_{t-i}^2 + \sum_{i=1}^q (\beta_i e_{t-i} + \gamma_i (|e_{t-i}| - E(|e_{t-i}|))) \quad (8)$$

where  $\omega_i$ ,  $\alpha_i$ ,  $\beta_i$  and  $\gamma_i$  are the parameters to be estimated and  $e_t = r_t / \sigma_t$ . The GJR-GARCH(p,o,q) developed by [11] has the following expression:

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^q \alpha_i r_{t-i}^2 + \sum_{i=1}^o \gamma_i r_{t-i}^2 I_{[r_{t-i} < 0]} + \sum_{i=1}^p \beta_i \sigma_{t-i}^2 \quad (9)$$

As with EGARCH model,  $\omega_i$ ,  $\alpha_i$ ,  $\beta_i$  and  $\gamma_i$  are the parameters to be estimated.  $I_{[r_{t-i} < 0]}$  takes the value of 1 when the subscript condition is met. Otherwise  $I_{[r_{t-i} < 0]} = 0$ . The volatility of the Threshold GARCH(p,o,q) (TrGARCH) model is obtained as follows:

$$\hat{\sigma}_t = \omega + \sum_{i=1}^q \alpha_i |r_{t-i}| + \sum_{i=1}^o \gamma_i |r_{t-i}| I_{[r_{t-i} < 0]} + \sum_{i=1}^p \beta_i \sigma_{t-i} \quad (10)$$

As with the previous two architectures,  $\omega_i$ ,  $\alpha_i$ ,  $\beta_i$  and  $\gamma_i$  are the model parameters. The last GARCH-based algorithm used in this paper is the fractionally integrated GARCH (FIGARCH) model developed by [65]. The conditional variance dynamic is

$$\hat{\sigma}_t = \omega + [1 - \beta L - \phi L(1 - L)^d] \epsilon_t^2 + \sigma h_{t-1} \quad (11)$$

where  $L$  is the lag operator and  $d$  the fractional differencing parameter.

In addition to the previous approaches, two other hybrid models based on merging autoregressive algorithms with ANNs and LSTMs are also used as benchmark. Figure 2 shows the architecture of ANN-GARCH and LSTM-GARCH. The inputs of the algorithms are the following:

- The last daily logarithmic return,  $r_{t-1}$ , for the ANN-GARCH and the last ten in the case of the LSTM-GARCH (as explained in Section 2.1).
- The standard deviation of the last five daily logarithmic returns:

$$\sigma_{t-1} = \sqrt{\frac{\sum_{i=1}^n (r_{t-i} - E[r])^2}{n-1}} \quad (12)$$

where  $E[r] = \sum_{i=1}^n r_{t-i}/n$  and  $n = 5$ . As with the previous input variable, the last standard deviation is considered in the ANN-GARCH, whereas the last ten are taken into consideration by the LSTM-GARCH architecture.

The GARCH-based algorithms included within the ANN-GARCH and LSTM-GARCH models are the six algorithms previously presented in this same subsection (GARCH, EGARCH, AVGARCH, GJR-GARCH, TrARCH, FIGARCH).

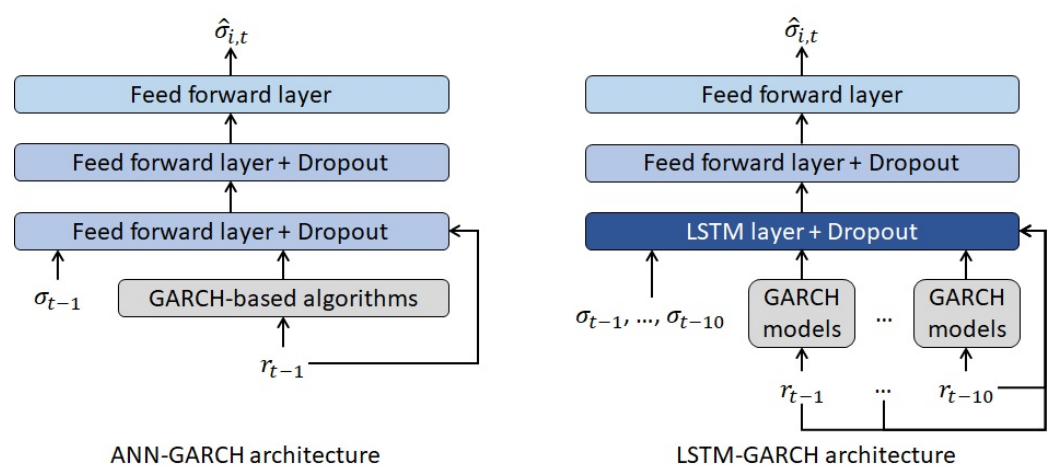


Figure 2. ANN-GARCH and LSTM-GARCH architectures.

As explained in Section 2.1, the true implied volatility,  $\sigma_{i,t}$ , is used as response variable to train the models. This variable is the standard deviation of the future logarithmic returns:

$$\hat{\sigma}_{i,t} = \sqrt{\frac{\sum_{n=0}^{i-1} (r_{t+n} - E[r_f])^2}{i-1}} \quad (13)$$

where  $E[r_f] = \sum_{n=0}^{i-1} r_{t+n}/i$ . In this paper,  $i = 5$ .

As it is shown in Figure 2, the input of the ANN-GARCH model is processed by two feed forward layers with dropout regularization. These layers have 16 and 8 neurons, respectively. The final output is produced by a feed forward layer with one neuron. In the case of the LSTM-GARCH, inputs are processed by a LSTM layer with 32 units and two feed forward layers with 8 and 1 neurons, respectively, in order to produce the final forecast.

## 2.4. Transformer-Based Models

Before explaining the volatility models based on Transformer layers (see Figure 3), all the modifications applied to their architecture are presented in this subsection. As previously stated, Transformer layers [54] were developed for NLP purposes. Thus, some modifications are needed in order to apply this layer for volatility forecasting purposes.

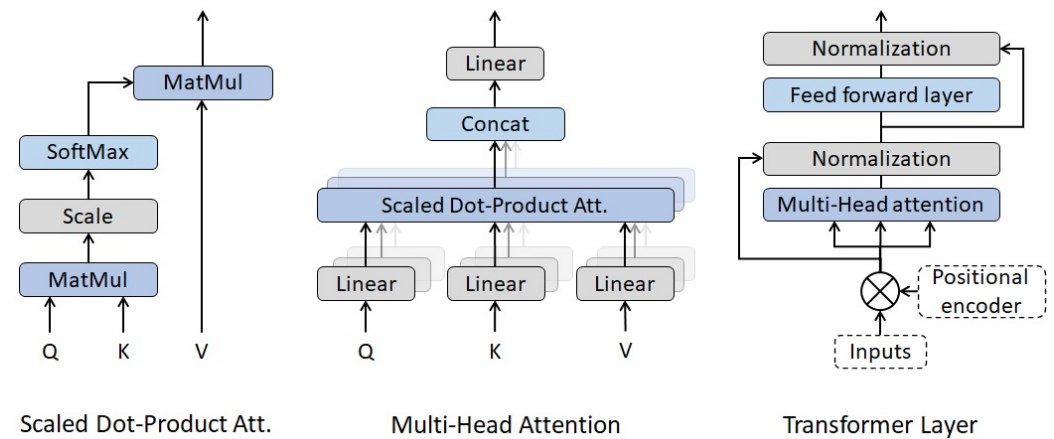


Figure 3. Transformer and Multi-Head attention mechanism.

In contrast to LSTM, recurrence is not present in the architecture of Transformer layers. The two main components used by these layers in order to deal with time series are the following:

- **Positional encoder.** As previously stated, Transformer layers have no recurrence structure. Thus, the information about the relative position of the observations within the time series needs to be included in the model. To do so, a positional encoding is added to the input data. In the context of NLP, Vaswani et al. [54] suggested the following wave functions as positional encoders:

$$PE_{(pos, 2i)} = \sin(pos/1000^{2i/dim}) \quad (14)$$

$$PE_{(pos, 2i+1)} = \cos(pos/1000^{2i/dim}) \quad (15)$$

where  $dim$  is the total number of explanatory variables (or word embedding dimension in NLP) used as input in the model,  $pos$  is the position of the observation within the time series and  $i = (1, 2, \dots, dim - 1)$ . This positional encoder modifies the input data depending on the lag of the time series and the embedding dimension used for the words.

As volatility models do not use words as inputs, the positional encoder is modified in order to avoid any variation of the inputs depending on the number of time series used as input. Thus, the positional encoder suggested in this paper changes depending on the lag, but it remains the same across the different explanatory variables introduced in the model. As in the previous case, a wave function plays the role of positional encoder:

$$PE_{pos} = \cos\left(\pi \frac{pos}{N_{pos} - 1}\right) = \sin\left(\frac{\pi}{2} + \pi \frac{pos}{N_{pos} - 1}\right) \quad (16)$$

where  $pos = (0, 1, \dots, N_{pos} - 1)$  is the position of the observation within the time series and  $N_{pos}$  maximum lag.

- **Multi-Head attention.** It can be considered the key component of the Transformer layers proposed by [54]. As shown in Figure 3, Multi-Head attention is composed

of several scaled dot-product attention units running in parallel. Scaled dot-product attention is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (17)$$

where  $Q$ ,  $K$  and  $V$  are input matrices and  $d_k$  the number of input variables taken into consideration within the dot-product attention mechanism. Multi-Head attention splits the explicative variables in different groups or 'heads' in order to run the different scaled dot-product attention units in parallel. Once the different heads are calculated, the outputs are concatenated (*Concat* operator) and connected to a feed forward layer with linear activation. Thus, the Multi-Head attention mechanism has the following expression:

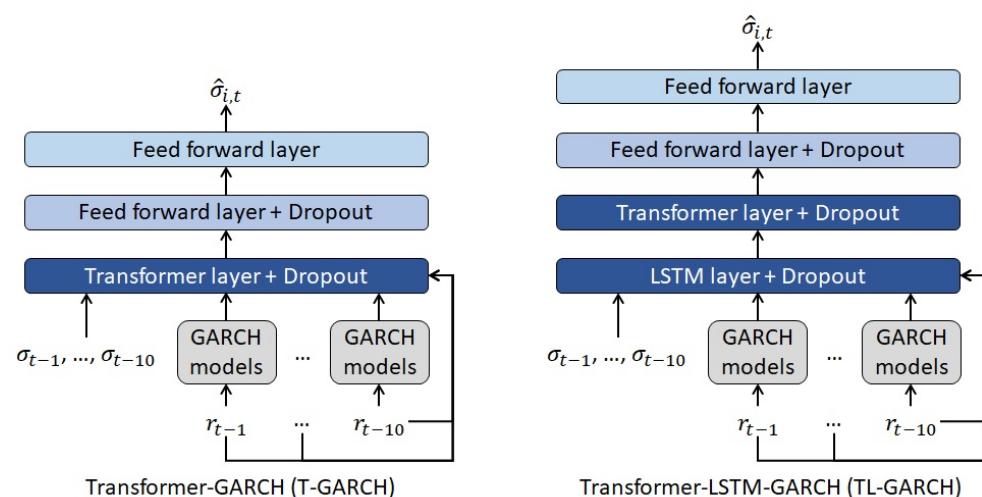
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (18)$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (19)$$

where  $h$  is the number of heads. It is also worth mentioning that all the matrices of parameters ( $W_i^Q$ ,  $W_i^K$ ,  $W_i^V$  and  $W^O$ ) are trained using feed forward layers with linear activations.

In addition to the scaled dot-product and the Multi-Head attention mechanisms, Figure 3 shows the Transformer layers used in this paper. As suggested by [54], the Multi-Head attention is followed by a normalization, a feed forward layer with ReLU activation and, again, a normalization layer. Transformer layers also include two residual connections [66]. Thanks to these connections, the model will decide by itself if the training of some layers needs to be skipped during some phases of the fitting process.

The modified version of Transformer layers explained in the previous paragraphs are used in the volatility models presented in Figure 4. The T-GARCH architecture proposed in this paper merges the six GARCH algorithms presented in Section 2.3 with Transformer and feed forward layers in order to forecast  $\hat{\sigma}_{i,t}$ . In addition to the previous algorithms and layers, TL-GARCH includes a LSTM with 32 units. In this last model, the temporal structure of the data is recognized and modelled by the LSTM layer and, thus, no positional encoder is needed in the Transformer layer. Both models have the following characteristics:



**Figure 4.** T-GARCH and TL-GARCH volatility models.

- Adaptive Moment Estimator (ADAM) is the algorithm used for updating the weights of the feed forward, LSTM and Transformer layers. This algorithm takes into consideration current and previous gradients in order to implement a progressive adaptation

of the initial learning rate. The values suggested by [67] for the ADAM parameters are used in this paper and the initial learning rate is set to  $\delta = 0.01$ .

- The feed forward layers with dropout present in both models have 8 neurons, while the output layer has just one.
- The level of dropout regularization  $\theta$  [68] is optimized with the training set mentioned in Section 2.1.
- The loss function used for weights optimization and back propagation purposes is the mean squared error.
- Batch size is equal to 64 and the models are trained during 5000 epochs in order to obtain the final weights.

## 2.5. Multi-Transformer-Based Models

This subsection presents the Multi-Transformer layers and the volatility models based on them. The Multi-Transformer architecture proposed in this paper is a variant of the Transformer layers proposed by [54]. The main differences between both architectures are the following:

- As shown in Figure 5, Multi-Transformer layers generate  $T$  different random samples of the input data. In the volatility models proposed in this paper, 90% of the observations of the database are randomly selected in order to compute the different samples.
- Multi-Transformer architecture is composed of  $T$  Multi-Head attention units (in this paper  $T = 5$ ), one per each random sample of the input data. Then, the average of the different units is computed in order to obtain the final attention matrix. Thus, the Average Multi-Head (AMH) mechanism present in Multi-Transformer can be defined as follows:

$$AMH(Q, K, V) = \frac{\sum_{t=1}^T \text{Concat}(\text{head}_{1,t}, \dots, \text{head}_{h,t}) W_t^O}{T} \quad (20)$$

$$\text{head}_{i,t} = \text{Attention}(Q_t W_{i,t}^Q, K_t W_{i,t}^K, V_t W_{i,t}^V) \quad (21)$$

As with the Transformer architecture applied in this paper, the positional encoder used is  $PE_{pos}$  instead of  $PE_{(pos, 2i)}$  and  $PE_{(pos, 2i+1)}$ .

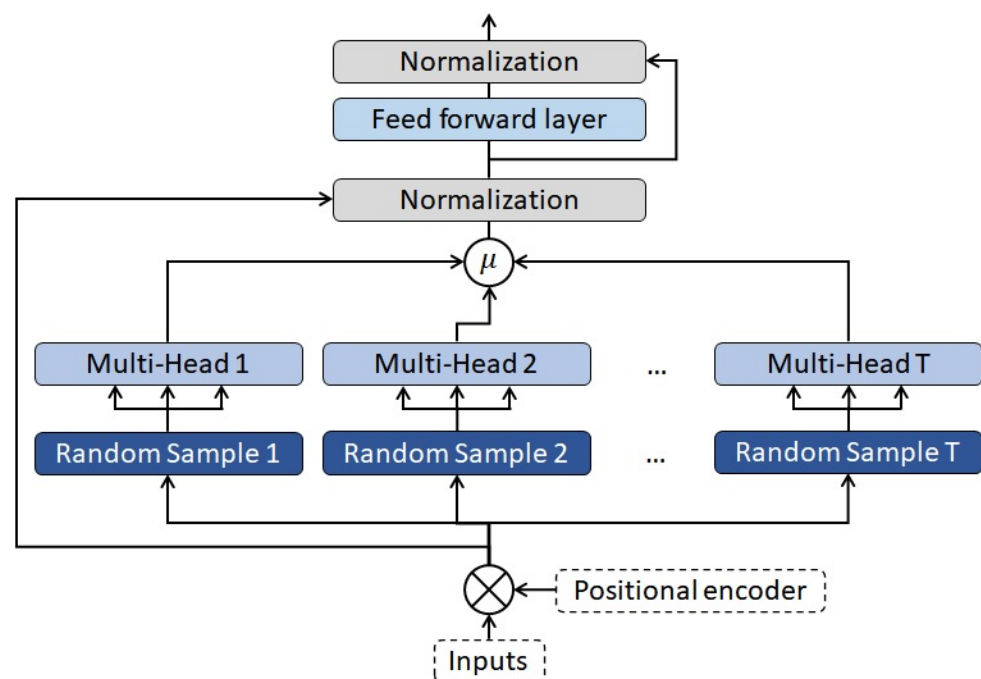


Figure 5. Multi-Transformer architecture.

The aim of the Multi-Transformer layers introduced in the paper is to improve the stability and accuracy by applying bagging [69] to the attention mechanism. This technique is usually applied to algorithms such as linear regression, neural networks or decision trees. Instead of applying the procedure on all the data that are input into the model, the proposed methodology uses bagging only to the attention mechanism of the layer architecture.

The computational power required by bagging is one of the main limitations of this technique. As Multi-Transformer applies bagging to the attention mechanisms, their weights are trained several times in each epoch. Nevertheless, bagging is not applied to the rest of the layer weights and, thus, this offsets partially the previous limitation. It is also worth mentioning that bagging preserves the bias and this may result in underfitting.

On the other hand, this technique should bring two main advantages to the Multi-Transformer layer. First, bagging reduces significantly the error variance. Second, the aggregation of learners using this technique leads to a higher accuracy and reduces the risk of overfitting.

The structure of the volatility models based on Multi-Transformer layers (Figure 6) is similar to the architectures presented in Section 2.4. The MT-GARCH merges Multi-Transformer and feed forward layers with the six GARCH models presented in Section 2.3. In addition to the previous algorithms and layers, MTL-GARCH adds a LSTM with 32 units. The rest of the characteristics such as the optimizer, the number of neurons of the feed forward layers or the level of dropout regularization are the same than those presented in the previous section for T-GARCH and TL-GARCH.

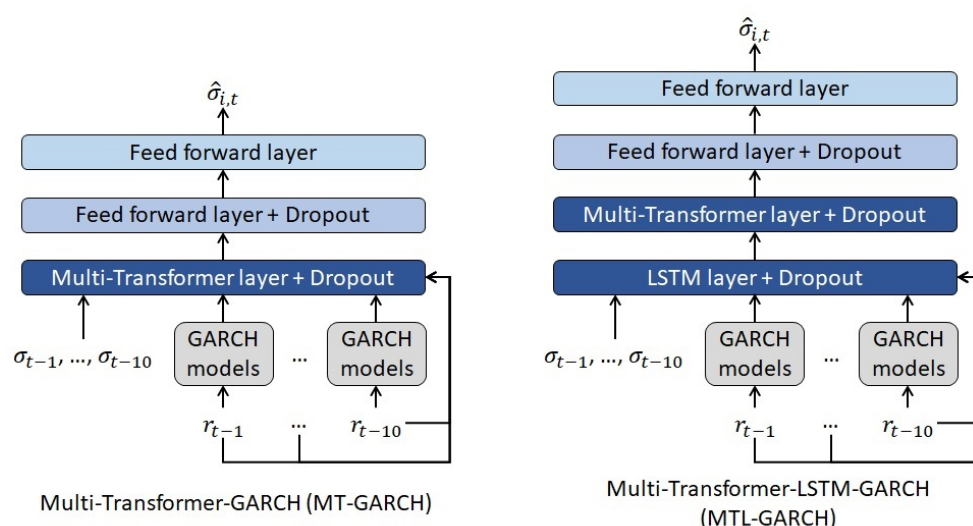


Figure 6. MT-GARCH and MTL-GARCH volatility models.

The risk measures of ANN-GARCH, LSTM-GARCH and all the models introduced by this paper (Sections 2.4 and 2.5) are calculated assuming that daily log-returns follow a non-standardize Student's t-distribution with standard deviation equal to the forecasts made by the volatility models. It is worth mentioning that Student's t-distribution generates more appropriate risk measures than normal distribution due to the shape of its tail [70,71]. In addition, this assumption is in line with the GARCH-based models used as benchmark and the inputs of the hybrid models presented in this paper.

### 3. Results

In this section, the forecasts and the risk measures of the volatility models presented in previous sections are compared with the ones obtained from the benchmark models. In addition, the following subsection shows the optimum hyperparameters of the benchmark and proposed hybrid volatility models.

### 3.1. Fitting of Models Based on Neural Networks

As explained in Section 2.1, rolling window approach ([57–60] among others) is applied for fitting the algorithms. The training set used for optimizing the level of dropout regularization contains S&P returns and observed volatilities from 1 January 2008 to 31 December 2015. Table 1 presents the error by model and level of  $\theta$ .

**Table 1.** RMSE by level of  $\theta$ .

Model	$\theta = 0$	$\theta = 0.05$	$\theta = 0.10$	$\theta = 0.15$
ANN-GARCH	0.0351	0.0092	0.0085	0.0082
LSTM-GARCH	0.0065	0.0057	0.0056	0.0054
T-GARCH	0.0089	0.0076	0.0072	0.0074
TL-GARCH	0.0050	0.0045	0.0044	0.0045
MT-GARCH	0.0068	0.0062	0.0064	0.0064
MTL-GARCH	0.0047	0.0045	0.0042	0.0044

Source: own elaboration.

The results of the optimization process reveals that  $\theta = 0$  generates higher error rates than the rest of the possible values regardless of the model. This means that models based on architectures such as Transformer, LSTM or feed forward layers need an appropriate level of regularization in order to avoid overfitting. According to the results, this is especially relevant for ANN-GARCH, where the error strongly depends on the level of regularization. The dropout level that minimizes the error of each model is selected.

### 3.2. Comparison against Benchmark Models

Once the optimum dropout level of each of the proposed volatility forecasting models based on Transformer and Multi-Transformer is selected, their performance is compared with the benchmark models (traditional GARCH processes, ANN-GARCH and LSTM-GARCH) presented in Section 2.3.

Tables 2 and 3 present the validation error (RMSE and MAE) by year and model. The column ‘Total’ shows the error of the whole test period (from 1 January 2016 to 31 December 2020). The main conclusions drawn from these tables are the following:

- Traditional GARCH processes are outperformed by models based on merging artificial neural network architectures such as feed forward, LSTM or Transformer layers with the outcomes of autoregressive algorithms (also named hybrid models).
- The comparison between ANN-GARCH and the rest of the volatility forecasting models based on artificial neural networks (LSTM-GARCH, T-GARCH, TL-GARCH, MT-GARCH and MTL-GARCH) reveals that feed forward layers lead to less accurate forecasts than other architectures. Multi-Transformer, Transformer and LSTM were specially created to forecast time series and, thus, the volatility models based on these layers are more accurate than ANN-GARCH.
- Merging Multi-Transformer and Transformer layers with LSTMs leads to more accurate predictions than traditional LSTM-based architectures. Indeed, TL-GARCH achieves better results than LSTM-GARCH, even though the number of weights of TL-GARCH is significantly lower. Thus, the novel Transformer and Multi-Transformer layers introduced for NLPs purposes can be adapted as described in Sections 2.4 and 2.5 in order to generate more accurate volatility forecasting models. It is also worth mentioning that Multi-Transformer layers, which were also introduced in this paper, lead to more accurate forecasts thanks to their ability to average several attention mechanisms. In fact, the model that achieves the lower MAE and RMSE is a mixture of Multi-Transformer and LSTM layers (MTL-GARCH).

**Table 2.** RMSE by volatility model and year.

Model	2016	2017	2018	2019	2020	Total
GARCH(1,1)	0.0058	0.0026	0.0095	0.0073	0.1026	0.0464
AVGARCH(1,1)	0.0053	0.0027	0.0076	0.0056	0.0847	0.0383
EGARCH(1,1)	0.0056	0.0028	0.0093	0.0078	0.0880	0.0399
GJR-GARCH(1,1,1)	0.0090	0.0028	0.0126	0.0068	0.1248	0.0565
TrGARCH(1,1,1)	0.0074	0.0027	0.0115	0.0058	0.1153	0.0521
FIGARCH(1,1)	0.0062	0.0029	0.0095	0.0066	0.1011	0.0457
ANN-GARCH	0.0042	0.0023	0.0060	0.0044	0.0171	0.0086
LSTM-GARCH	0.0032	0.0021	0.0043	0.0030	0.0101	0.0054
T-GARCH	0.0048	0.0029	0.0058	0.0044	0.0117	0.0067
TL-GARCH	0.0030	0.0019	0.0033	0.0026	0.0070	0.0040
MT-GARCH	0.0036	0.0021	0.0046	0.0033	0.0096	0.0054
MTL-GARCH	0.0030	0.0016	0.0033	0.0026	0.0066	0.0038

Source: own elaboration.

**Table 3.** MAE by volatility model and year.

Model	2016	2017	2018	2019	2020	Total
GARCH(1,1)	0.0037	0.0019	0.0058	0.0044	0.0363	0.0105
AVGARCH(1,1)	0.0034	0.0019	0.0049	0.0037	0.0296	0.0087
EGARCH(1,1)	0.0035	0.0020	0.0060	0.0048	0.0333	0.0100
GJR-GARCH(1,1,1)	0.0048	0.0020	0.0074	0.0042	0.0404	0.0118
TrGARCH(1,1,1)	0.0042	0.0020	0.0069	0.0038	0.0365	0.0107
FIGARCH(1,1)	0.0038	0.0021	0.0055	0.0041	0.0361	0.0104
ANN-GARCH	0.0029	0.0019	0.0038	0.0029	0.0095	0.0042
LSTM-GARCH	0.0022	0.0015	0.0027	0.0021	0.0060	0.0029
T-GARCH	0.0035	0.0021	0.0041	0.0031	0.0070	0.0040
TL-GARCH	0.0020	0.0014	0.0021	0.0018	0.0044	0.0023
MT-GARCH	0.0024	0.0016	0.0031	0.0023	0.0057	0.0030
MTL-GARCH	0.0019	0.0012	0.0021	0.0018	0.0041	0.0022

Source: own elaboration.

To enhance the analysis of the results shown in Tables 2 and 3, Figure 7 collects the RMSE and the observed volatility by year. Notice that only the most accurate GARCH-based model is shown in order to improve the visualization of the graph. The black dashed line shows that the observed volatility of 2020 was significantly higher than the rest of the years due to the turmoil caused by COVID-19 outbreak. As expected, the error of every model is also higher in 2020 because the market volatility was more unpredictable than the rest of the years. Nevertheless, it has to be mentioned that the 2020 forecasts of traditional autoregressive algorithms are significantly less accurate than hybrid models based on architectures such as LSTM, Transformer or Multi-Transformer layers.

Although the observed volatility is lower in years before 2020, autoregressive models are also outperformed by hybrid models. Nevertheless, the difference between both sets of models is remarkably lower.

The  $p$ -values of the Kupiec and Christoffersen tests by volatility model and year are shown in Tables 4 and 5, respectively. In contrast to the approach suggested by Kupiec, Christoffersen test is not only focused on the total number of exceedances, but it also takes into consideration the number of consecutive VaR exceedances. As stated in Section 2.2, the risk measure and confidence level (99.5% VaR) selected are in line with Solvency II Directive. This regulation sets the principles for calculating the capital requirements and assessing the risk profile of the insurance companies based in the European Union. This law covers not only the underwriting risks but also financial risks such as the potential losses due to variations on the interest rate curves or the equity prices.

The column ‘Total’ of Tables 4 and 5 reveal that only TL-GARCH, MT-GARCH and MTL-GARCH produce appropriate risk measures ( $p$ -value higher than 0.05 in both tests)

for the period 2016–2020. The rest of the models fail both tests and, thus, their risk measures can not be considered to be appropriate for that period.

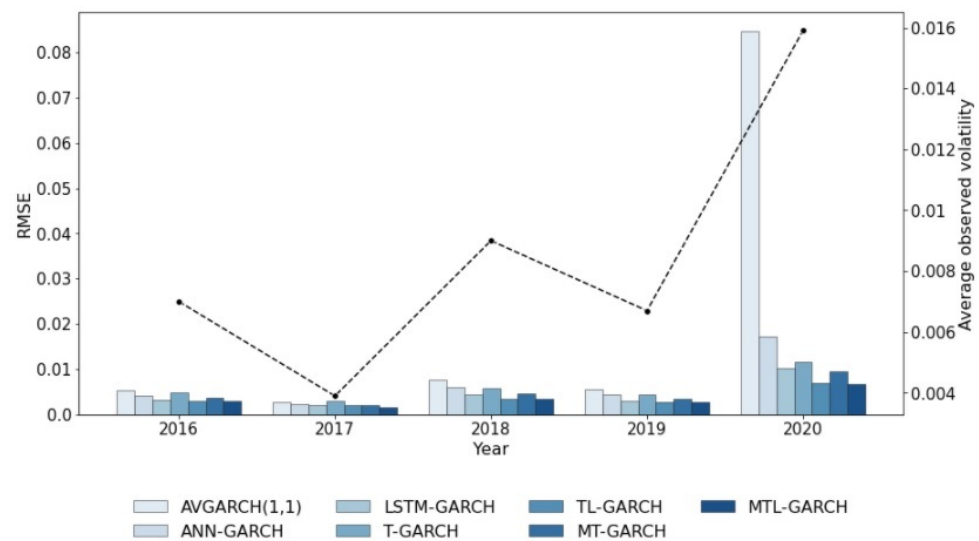


Figure 7. Observed volatility and RMSE by year.

As with any other statistical test, the higher the number of data points the more relevant are the outcomes obtained from the test. That is the reason why the previous paragraph focuses on the ‘Total’ column and not on the specific results obtained by year. The results by year show that most of the models fail the test in 2020 due to the high level of volatility produced by COVID-19 pandemic.

According to these results, the stock volatility models introduced in this paper (T-GARCH, TL-GARCH, MT-GARCH and MTL-GARCH) produce more accurate estimations and appropriate risk measures in most of the cases. Regarding the models accuracy, it is specially remarkable the difference observed in 2020, where COVID-19 caused a significant turmoil in the stock market. Concerning the appropriateness of equity risk measures, three out of four models based on Transformer and Multi-Transformer pass Kupiec and Christoffersen test for the period 2016–2020, while all the benchmark models fail at least one of them. Notice that the proposed models are compared with other approaches belonging to its own family (ANN-GARCH and LSTM-GARCH) and autoregressive models belonging to the GARCH family.

Table 4. Kupiec test ( $p$ -values) by volatility model and year.

Model	2016	2017	2018	2019	2020	Total
GARCH(1,1)	0.543	0.540	0.051	0.543	0.052	0.008
AVGARCH(1,1)	0.543	0.540	0.051	0.543	0.052	0.008
EGARCH(1,1)	0.543	0.540	0.051	0.543	0.052	0.008
GJR-GARCH(1,1,1)	0.543	0.540	0.011	0.543	0.190	0.008
TrGARCH(1,1,1)	0.543	0.540	0.051	0.810	0.190	0.042
FIGARCH(1,1)	0.543	0.540	0.051	0.543	0.052	0.008
ANN-GARCH	0.543	0.540	0.001	0.002	0.012	0.001
LSTM-GARCH	0.810	0.186	0.540	0.188	0.190	0.042
T-GARCH	0.188	0.540	0.002	0.543	0.052	0.001
TL-GARCH	0.543	0.540	0.813	0.810	0.810	0.782
MT-GARCH	0.112	0.540	0.540	0.188	0.052	0.089
MTL-GARCH	0.543	0.113	0.113	0.810	0.190	0.910

Source: own elaboration.

**Table 5.** Christoffersen test (*p*-values) by volatility model and year.

Model	2016	2017	2018	2019	2020	Total
GARCH(1,1)	0.522	0.520	0.004	0.523	0.048	0.002
AVGARCH(1,1)	0.522	0.520	0.004	0.523	0.048	0.002
EGARCH(1,1)	0.522	0.520	0.004	0.523	0.048	0.002
GJR-GARCH(1,1,1)	0.522	0.520	0.002	0.523	0.179	0.002
TrGARCH(1,1,1)	0.522	0.520	0.004	0.800	0.179	0.009
FIGARCH(1,1)	0.522	0.520	0.004	0.523	0.048	0.002
ANN-GARCH	0.522	0.520	0.001	0.002	0.002	0.001
LSTM-GARCH	0.800	0.180	0.520	0.177	0.179	0.037
T-GARCH	0.176	0.520	0.001	0.523	0.048	0.001
TL-GARCH	0.522	0.520	0.803	0.800	0.797	0.693
MT-GARCH	0.113	0.520	0.520	0.177	0.048	0.079
MTL-GARCH	0.522	0.113	0.113	0.800	0.179	0.790

Source: own elaboration.

#### 4. Discussion

This paper introduced a set of volatility forecasting models based on Transformer and Multi-Transformer layers. As Transformer layers were developed for NLP purposes [54], their architecture is adapted in order to generate stock volatility forecasting models. Multi-Transformer layers, which are introduced by this paper, have the aim of improving the stability and accuracy of Transformer layers by applying bagging to the attention mechanism. The predictive power and risk measures generated by the proposed volatility forecasting models (T-GARCH, TL-GARCH, MT-GARCH and MTL-GARCH) are compared with traditional GARCH processes and other hybrid models based on LSTM and feed forward layers.

Three main outcomes were drawn from the empirical results. First, hybrid models based on LSTM, Transformer or Multi-Transformer layers outperform traditional autoregressive algorithms and hybrid models based on feed forward layers. The validation error by year shows that this difference is more relevant in 2020, when the volatility of S&P500 was significantly higher than in the previous years due to COVID-19 pandemic. Volatility forecasting models are mainly used for pricing derivatives and assessing the risk profile of financial institutions. As the more relevant shocks on the solvency position of financial institutions and derivatives prices are observed in high volatility regimes, the accurateness of these models is particularly important in years such as 2020.

The higher performance of hybrid models have also been demonstrated by [38–44]. These papers merged traditional GARCH models with feed forward layers to predict stock market volatility. This type of models have shown also a superior performance in other financial fields such as oil market volatility [48,49] and metals price volatility [46,47]. Notice that this paper does not only present a comparison with traditional autoregressive models, but it also shows that Transformer and Multi-Transformer can lead to more accurate volatility estimations than other hybrid models.

Second, Multi-Transformer layers lead to more accurate volatility forecasting models than Transformer layers. As expected, applying bagging to the attention mechanism has a positive impact on the performance of the models presented in this paper. It is also remarkable that empirical results demonstrate that merging LSTM with Transformer or Multi-Transformer layers has also a positive impact on the models performance. On one hand, the volatility forecasting model based on Multi-Transformer and LSTM (named MTL-GARCH) achieves the best results in the period 2016–2020. On the other hand, the merging of Transformer with LSTM (TL-GARCH) leads to a lower error rate than the hybrid model based only on LSTM layers (LSTM-GARCH) even though the number of weights of the first model is significantly lower. Thus, the use of Transformer layers can lead to simpler and more accurate volatility forecasting models. Notice that Transformer layers are already considered the state of art thanks to BERT [55] and GPT-3 [56]. These models have been

successfully used for sentence prediction, conversational response generation, sentiment classification, coding and writing fiction, among others.

Third, the results of Kupiec and Christoffersen tests revealed that only the risk estimations made by MTL-GARCH, TL-GARCH and MT-GARCH can be considered as appropriate for the period 2016–2020, whereas traditional autoregressive algorithms and hybrid models based on feed forward and LSTM layers failed, at least, one of the tests. As previously stated, volatility does not play only a key role in risk management but also in derivative valuation models. Thus, using a volatility model that generates appropriate risk measures can lead to more accurate derivatives valuation.

## 5. Conclusions

Transformer layers are the state of the art in natural language processing. Indeed, the performance of this layer have overcome the performance of any other previous model in this field [56]. As Transformer layers were specially created for natural language processing, they need to be modified in order to be used for other purposes. Probably, this is one of the main reasons why this layer have not been already extended to other fields. This paper provides the modifications needed to apply this layer for stock volatility forecasting purposes. The results shown in this paper demonstrates that Transformer layers can overcome also the performance of the main stock volatility models.

Following the intuition of bagging [69], this paper introduces Multi-Transformer layers. This novel architecture has the aim of improving the stability and accuracy of the attention mechanism, which is the core of Transformer layers. According to the results, it can be concluded that this procedure improves the accuracy of stock volatility models based on Transformer layers.

Leaving aside the comparisons between Transformer and Multi-Transformer layers, the hybrid models based on them have overcome the performance of autoregressive algorithms and other models based on feed forward layers and LSTMs. The architecture of these hybrid models (T-GARCH, TL-GARCH, MT-GARCH and MTL-GARCH) based on Transformer and Multi-Transformer layers is also provided in this paper.

According to the results, it is also worth noticing that the risk estimations based on the previous models are specially appropriate. The VaR of most of these models can be considered accurate even in years such as 2020, when the COVID-19 pandemic caused a remarkable turmoil in the stock market.

Consequently, the empirical results obtained with the hybrid models based on Transformer and Multi-Transformer layers suggest that further investigation should be conducted about the possible application of them for derivative valuation purposes. Notice that volatility plays a key role in the financial derivatives valuation. In addition, the models can be extended by merging Transformer or Multi-Transformer layers with other algorithms (such as gradient boosting with trees or random forest) or modifying some key assumptions of the attention mechanism.

**Author Contributions:** Conceptualization, E.R.-P.; methodology, E.R.-P., P.J.A.-G. and J.J.N.-V.; software, E.R.-P.; validation, P.J.A.-G. and J.J.N.-V.; formal analysis, E.R.-P.; investigation, E.R.-P., P.J.A.-G. and J.J.N.-V.; writing—both original draft preparation, review and editing, E.R.-P., P.J.A.-G. and J.J.N.-V.; supervision, P.J.A.-G. and J.J.N.-V.; project administration, P.J.A.-G. and J.J.N.-V.; funding acquisition, P.J.A.-G. and J.J.N.-V. All authors have read and agreed to the published version of the manuscript.

**Funding:** The APC was funded by Economics Department of Universidad de Alcalá.

**Data Availability Statement:** The Python implementation of the volatility models proposed in this paper is available in <https://github.com/EduardoRamosP/MultiTransformer> (accessed on 26 June 2021).

**Conflicts of Interest:** The authors declare that they have no conflict of interest regarding the publication of the research article.

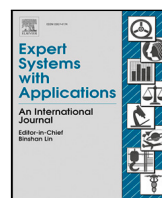
## References

- Hull, J. *Risk Management and Financial Institutions*, 4th ed.; Wiley and Sons: London, UK, 2015.
- Rajashree, P.; Ranjeeta, B. A differential harmony search based hybrid internal type2 fuzzy EGARCH model for stock market volatility prediction. *Int. J. Approx. Reason.* **2015**, *59*, 81–104.
- Engle, R. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* **1982**, *50*, 987–1007. [[CrossRef](#)]
- Bollerslev, T. Generalized autoregressive conditional heteroskedasticity. *J. Econom.* **1986**, *31*, 307–327. [[CrossRef](#)]
- Mandelbrot, B. The variation of certain speculative prices. *J. Bus.* **1963**, *36*, 394–419. [[CrossRef](#)]
- Engle, R.; Lee, G. A permanent and transitory component model of stock return volatility. In *Cointegration, Causality, and Forecasting: A Festschrift in Honor of Clive W. J. Granger*; Engle, R., White, H., Eds.; Oxford University Press: Oxford, UK, 1999; pp. 475–497.
- Haas, M.; Mittnik, S.; Paoletta, M. Mixed normal conditional heteroskedasticity. *J. Financ. Econom.* **2004**, *2*, 211–250. [[CrossRef](#)]
- Haas, M.; Mittnik, S.; Paoletta, M. A new approach to Markov-switching GARCH models. *J. Financ. Econom.* **2004**, *2*, 493–530. [[CrossRef](#)]
- Haas, M.; Paoletta, M. Mixture and regime-switching GARCH models. In *Handbook of Volatility Models and Their Applications*; Bauwens, L., Hafner, C., Laurent, S., Eds.; John Wiley and Sons: Hoboken, NJ, USA, 2012; pp. 71–102.
- Nelson, D.B. Conditional Heteroskedasticity in Asset Returns: A New Approach. *Econometrica* **1991**, *59*, 347–70. [[CrossRef](#)]
- Glosten, L.; Jagannathan, R.; Runkle, D. On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *J. Financ.* **1993**, *48*, 1779–1801. [[CrossRef](#)]
- Kraft, D.; Engle, R. *Autoregressive Conditional Heteroskedasticity in Multiple Time Series*; Department of Economics, UCSD: San Diego, CA, USA, 1982.
- Engle, R.; Granger, C.; Kraft, D. Combining competing forecasts of inflation with a bivariate ARCH model. *J. Econ. Dyn. Control.* **1984**, *8*, 151–165. [[CrossRef](#)]
- Bollerslev, T.; Engle, R.; Wooldridge, J. A Capital Asset Pricing Model with time-varying covariances. *J. Political Econ.* **1988**, *96*, 116–131. [[CrossRef](#)]
- Tse, Y.; Tsui, K. A multivariate GARCH model with time-varying correlations. *J. Bus. Econ. Stat.* **2002**, *20*, 351–362. [[CrossRef](#)]
- Engle, R. Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *J. Bus. Econ. Stat.* **2002**, *20*, 339–350. [[CrossRef](#)]
- Engle, R.; Kroner, F. Multivariate simultaneous generalized ARCH. *Econom. Theory* **1995**, *11*, 122–150. [[CrossRef](#)]
- Engle, R.; Ng, V.; Rotschild, M. Asset pricing with a factor-ARCH covariance structure: Empirical estimates for Treasury Bills. *J. Econom.* **1990**, *45*, 213–238. [[CrossRef](#)]
- Zhang, L.; Zhu, K.; Ling, S. The ZD-GARCH model: A new way to study heteroscedasticity. *J. Econom.* **2018**, *202*, 1–17. [[CrossRef](#)]
- Heston, S.L. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Rev. Financ. Stud.* **1993**, *6*, 327–343. [[CrossRef](#)]
- Cox, J.; Ingersoll, J.; Ross, S. A Theory of the Term Structure of Interest Rates. *Econometrica* **1985**, *53*, 385–407. [[CrossRef](#)]
- Melino, A.; Turnbull, S. Pricing foreign currency options with stochastic volatility. *J. Econom.* **1990**, *45*, 239–265. [[CrossRef](#)]
- Andersen, T.; Sorensen, B. GMM estimation of a stochastic volatility model: A Monte Carlo study. *J. Bus. Econ. Stat.* **1999**, *14*, 329–352.
- Durbin, J.; Koopman, S. Monte Carlo maximum likelihood estimation for non-Gaussian state space models. *Biometrika* **1997**, *84*, 669–684. [[CrossRef](#)]
- Broto, C.; Ruiz, E. Estimation methods for stochastic volatility models: A survey. *J. Econ. Surv.* **2004**, *18*, 613–649. [[CrossRef](#)]
- Danielsson, J. Stochastic volatility in asset prices: Estimation by simulated maximum likelihood. *J. Econom.* **2004**, *64*, 375–400. [[CrossRef](#)]
- Andersen, T. *Encyclopedia of Complexity and System Sciences*; Chapter Stochastic Volatility; Springer: Berlin/Heidelberg, Germany, 2009.
- Hull, J.C.; White, A. The Pricing of Options on Assets with Stochastic Volatilities. *J. Financ.* **1987**, *42*, 281–300. [[CrossRef](#)]
- Hagan, P.; Kumar, D.; Lesniewski, A.; Woodward, D. Managing Smile Risk. *Wilmott Mag.* **2002**, *1*, 84–108.
- Mcculloch, W.; Pitts, W. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bull. Math. Biophys.* **1943**, *5*, 127–147. [[CrossRef](#)]
- Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2000**, *29*, 1189–1232.
- Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [[CrossRef](#)]
- Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [[CrossRef](#)]
- Gestel, T.; Suykens, J.; Baestens, D.; Lambrechts, A.; Laneknet, G. Financial time series prediction using least squares Support Vector Machines within the evidence framework. *IEEE Trans. Neural Netw.* **2001**, *12*, 8009–821. [[CrossRef](#)] [[PubMed](#)]
- Gupta, A.; Dhingra, B. Stock markets prediction using hidden Markov models. In Proceedings of the 2012 Students Conference on Engineering and Systems, Allahabad, India, 16–18 March 2012; pp. 1–4.
- Dias, F.; Nogueira, R.; Peixoto, G.; Moreira, W. Decision-making for financial trading: A fusion approach of Machine Learning and Portfolio Selection. *Expert Syst. Appl.* **2019**, *115*, 635–655.
- Hamid, S.; Iqbid, Z. Using neural networks for forecasting volatility of S&P 500 Index futures prices. *J. Bus. Res.* **2002**, *57*, 1116–1125.

38. Roh, T. Forecasting the Volatility of Stock Price Index. *Expert Syst. Appl.* **2006**, *33*, 916–922.
39. Hajizadeh, E.; Seifi, A.; Zarandi, F.; Turksen, I. A hybrid modeling approach for forecasting the volatility of S&P 500 Index return. *Expert Syst. Appl.* **2012**, *39*, 531–536.
40. Kristjanpoller, W.; Fadic, A.; Minutolo, M. Volatility forecast using hybrid neural network models. *Expert Syst. Appl.* **2014**, *41*, 2437–2442. [[CrossRef](#)]
41. Monfared, S.A.; Enke, D. Volatility Forecasting Using a Hybrid GJR-GARCH Neural Network Model. *Procedia Comput. Sci.* **2014**, *36*, 246–253. [[CrossRef](#)]
42. Lu, X.; Que, D.; Cao, G. Volatility Forecast Based on the Hybrid Artificial Neural Network and GARCH-type Models. *Procedia Comput. Sci.* **2016**, *91*, 1044–1049. [[CrossRef](#)]
43. Kim, H.; Won, C. Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. *Expert Syst. Appl.* **2018**, *103*, 25–37. [[CrossRef](#)]
44. Back, Y.; Kim, H. ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Syst. Appl.* **2018**, *113*, 457–480. [[CrossRef](#)]
45. Bildirici, M.; Ersin, O. Improving forecasts of GARCH family models with the artificial neural networks: An applicaiton to the daily returns in Istanbul Stock Exchange. *Expert Syst. Appl.* **2009**, *36*, 7355–7362. [[CrossRef](#)]
46. Kristjanpoller, W.; Minutolo, M. Gold price volatility: A Forecasting approach using the Artificial Neural Network-GARCH model. *Expert Syst. Appl.* **2015**, *42*, 7245–7251. [[CrossRef](#)]
47. Kristjanpoller, W.; Hernández, E. Volatility of main metals forecasted by a hybrid ANN-GARCH model with regressors. *Expert Syst. Appl.* **2017**, *84*, 290–300. [[CrossRef](#)]
48. Kristjanpoller, W.; Minutolo, M. Forecasting volatility of oil price using an Artificial Neural Network-GARCH model. *Expert Syst. Appl.* **2016**, *65*, 233–241. [[CrossRef](#)]
49. Verma, S. Forecasting volatility of crude oil futures using a GARCH–RNN hybrid approach. *Intell. Syst. Accounting, Financ. Manag.* **2021**. [[CrossRef](#)]
50. Ramos-Pérez, E.; Alonso-González, P.; Núñez-Velázquez, J. Forecasting volatility with a stacked model based on a hybridized Artificial Neural Network. *Expert Syst. Appl.* **2019**, *129*, 1–9. [[CrossRef](#)]
51. Vidal, A.; Kristjanpoller, W. Gold volatility prediction using a CNN-LSTM approach. *Expert Syst. Appl.* **2020**, *157*. [[CrossRef](#)]
52. Jung, G.; Choi, S.Y. Forecasting Foreign Exchange Volatility Using Deep Learning Autoencoder-LSTM Techniques. *Complexity* **2021**, *2021*, 1–16. [[CrossRef](#)]
53. Peng, Y.; Melo, P.; Camboim de Sá, J.; Akaishi, A.; Montenegro, M. The best of two worlds: Forecasting high frequency volatility for cryptocurrencies and traditional currencies with Support Vector Regression. *Expert Syst. Appl.* **2018**, *97*, 177–192. [[CrossRef](#)]
54. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**, *2017*, 5998–6008.
55. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
56. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H., Eds.; Curran Associates, Inc.: New York, NY, USA, 2020; Volume 33, pp. 1877–1901.
57. Swanson, N.R. Money and output viewed through a rolling window. *J. Monet. Econ.* **1998**, *41*, 455–474. [[CrossRef](#)]
58. Goyal, A.; Welch, I. *Predicting the Equity Premium With Dividend Ratios*; NBER Working Papers 8788; National Bureau of Economic Research, Inc.: Cambridge, MA, USA, 2002.
59. Zivot, E.; Wang, J. *Modeling Financial Time Series with S-PLUS®*; Springer: Berlin/Heidelberg, Germany, 2006.
60. Molodtsova, T.; Papell, D. Taylor Rule Exchange Rate Forecasting during the Financial Crisis. *NBER Int. Semin. Macroecon.* **2012**, *9*, 55–97. [[CrossRef](#)]
61. Kupiec, P.H. Techniques for Verifying the Accuracy of Risk Measurement Models. *J. Deriv.* **1995**, *3*, 73–84. [[CrossRef](#)]
62. Christoffersen, P.F.; Bera, A.; Berkowitz, J.; Bollerslev, T.; Diebold, F.; Giorgianni, L.; Hahn, J.; Lopez, J.; Mariano, R. Evaluating Interval Forecasts. *Int. Econ. Rev.* **1997**, *39*, 841–862. [[CrossRef](#)]
63. Bauwens, L.; Hafner, C.; Laurent, S. *Handbook of Volatility Models and Their Applications*; Wiley Handbooks in Financial E; Wiley: Hoboken, NJ, USA, 2012.
64. Taylor, S.J. *Modelling Financial Time Series*; Wiley: Hoboken, NJ, USA, 1986.
65. Baillie, R.T.; Bollerslev, T.; Mikkelsen, H.O. Fractionally integrated generalized autoregressive conditional heteroskedasticity. *J. Econom.* **1996**, *74*, 3–30. [[CrossRef](#)]
66. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [[CrossRef](#)]
67. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
68. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
69. Breiman, L. Bagging Predictors. *Mach. Learn.* **1996**, *24*, 123–140. [[CrossRef](#)]

- 
70. Jeremic, Z.; Terzić, I. Empirical estimation and comparison of Normal and Student T linear VaR on the Belgrade Stock Exchange. In Proceedings of the Sinteza 2014—Impact of the Internet on Business Activities in Serbia and Worldwide, Belgrade, Serbia, 25–26 April 2014; pp. 298–302. [\[CrossRef\]](#)
  71. McNeil, A.J.; Frey, R.; Embrechts, P. *Quantitative Risk Management: Concepts, Techniques and Tools*; Princeton University Press: Princeton, NJ, USA, 2015.

#### **8.4 Annex IV. Published Paper. Mack-Net model: Blending Mack's model with Recurrent Neural Networks**



# Mack-Net model: Blending Mack's model with Recurrent Neural Networks

Eduardo Ramos-Pérez<sup>a</sup>, Pablo J. Alonso-González<sup>b,\*</sup>, José Javier Núñez-Velázquez<sup>b</sup>

<sup>a</sup> Faculty of Economics. Universidad de Alcalá, Plaza de la Victoria 2, 28802 Alcalá de Henares, Spain

<sup>b</sup> Economics Department. Universidad de Alcalá, Plaza de la Victoria 2, 28802 Alcalá de Henares, Spain

## ARTICLE INFO

MSC:

62-07

62P05

65C60

90-08

Keywords:

Deep learning

Mack's model

Recurrent Neural Networks

Reserving risk

Stochastic reserving

## ABSTRACT

In general insurance companies, a correct estimation of liabilities plays a key role due to its impact on management and investing decisions. Since the Financial Crisis of 2007–2008 and the strengthening of regulation, the focus is not only on the total reserve but also on its variability, which is an indicator of the risk assumed by the company. Thus, measures that relate profitability with risk are crucial in order to understand the financial position of insurance firms. Taking advantage of the increasing computational power, this paper introduces a stochastic reserving model whose aim is to improve the performance of the traditional Mack's reserving model by applying an ensemble of Recurrent Neural Networks. The results demonstrate that blending traditional reserving models with deep and machine learning techniques leads to a more accurate assessment of general insurance liabilities.

## 1. Introduction

As an accurate estimation of future payments and its volatility allows the management to take correct underwriting and reinsurance decisions, reserving, understood as the calculation of the amount of required reserves for insurance policies, plays a fundamental role in general insurance firms. The interest of investors and regulators in analysing the volatility of financial institutions has increased significantly since the 2007–2008 Financial Crisis. Regulatory requirements have been enhanced with laws such as Solvency II Directive and Solvency Swiss Test in order to assess the risk profile of insurance companies. Since then, investors are not only focused on the profit but also on the level of risk assumed by the insurance firm to obtain it. Thus, indicators that relate profitability with risk such as the Return on Risk Adjusted Capital (Braun, Schmeiser, & Schreiber, 2018) have increased remarkably their influence on the stock prices of financial institutions.

Taking into consideration the historical information, the first reserving methods for estimating the ultimate cost in non-life insurance were only focused on obtaining the most likely scenario. Therefore, these deterministic models were unable to estimate the loss reserve uncertainty. Chain Ladder is the most widely used methodology within this family of reserving models. Nevertheless, Bornhuetter and Ferguson (1972) model tends to perform better when historical information is not stable enough to apply the Chain Ladder methodology.

As stated previously, general insurance firms are not only interested in the expected ultimate cost but also in its volatility. Consequently, different stochastic methodologies linked to the Chain Ladder procedure were developed. One of the most popular models for estimating the loss reserve variability was introduced by Mack (1993). This methodology, commonly known as Mack's model in the literature, derives the reserve variability by focusing on the first two moments. England and Verrall (2006) developed a bootstrap method that allows the analyst to obtain a complete reserve distribution by applying the free-distribution model of Mack.

Another widely used method to calculate reserve variability is the Overdispersed Poisson (ODP) model, which was developed by Renshaw and Verrall (1998). This method assumes that incremental payments follow an ODP distribution, where their variance is proportional to their mean. In this model, incremental payments must be positive, but this limitation can be overcome by using the quasi-likelihood approach developed by McCullagh and Nelder (1989). As in the case of Mack's model, a complete reserve distribution can be obtained by applying the bootstrap procedure suggested by England (2002) and England and Verrall (1999).

For those cases where data does not follow an ODP distribution, there are other methods based on Chain Ladder such as the log-normal model of Kremer (1982), the gamma procedure of Mack (1991) and the negative binomial methodology developed by Verrall (2000). The distribution function of some of the former models can be obtained

\* Corresponding author.

E-mail addresses: [eduardo.ramos@edu.uah.es](mailto:eduardo.ramos@edu.uah.es) (E. Ramos-Pérez), [pablo.alonsog@uah.es](mailto:pablo.alonsog@uah.es) (P.J. Alonso-González), [josej.nunez@uah.es](mailto:josej.nunez@uah.es) (J.J. Núñez-Velázquez).

by using Bayesian inference. England and Verrall (2006) introduced the procedure for implementing a Bayesian ODP, Mack and Negative Binomial models. The computation of the loss reserve distribution by means of Bayesian inference was recently expanded by Meyers (2015), who introduced several Bayesian Markov Chain Monte-Carlo (MCMC) models for incurred and paid data. These models (Levelled Chain-Ladder, Correlated Chain-Ladder, Levelled Incremental Trend, Correlated Incremental Trend and Changing Settlement Rate) have the aim of improving the performance of the traditional models based on Chain Ladder. This is achieved by including effects such as recognizing the correlation between accident years, applying a skewed distribution to negative incremental payments, introducing a trend over the different development years and implementing the possibility of applying changes in the claim settlement rate.

As incurred and paid data can have different patterns and characteristics, there is a set of loss reserving models, based on the Chain Ladder methodology, which were developed in order to take into consideration both data sources. The most relevant methods within this family are Munich Chain Ladder (Quarg & Mack, 2004), Double Chain Ladder (Martínez-Miranda, Nielsen, & Verrall, 2012) and Paid-Incurred Chain (Posthuma, Cator, Veerkamp, & Van Zwet, 2008). With regard to this last model, it is worth mentioning that Merz and Wüthrich (2010) developed a Bayesian implementation of it, while Happ, Merz, and Wüthrich (2012) and Happ and Wüthrich (2013) introduced methods strongly related to this approach. Finally, Halliwell (2009) and Venter (2008) used incurred and paid data to develop regression-based reserving models, while Antonio and Plat (2014), Martínez-Miranda, Nielsen, and Verrall (2013) and Pigeon, Antonio, and Denuit (2014) used both sources of information to estimate the expected ultimate cost.

The increase of the computational power and the success of machine and deep learning in many fields (Brown et al., 2020; LeCun, Bengio, & Hinton, 2015; Ramos-Pérez, Alonso-González, & Núñez Velázquez, 2021a; Silver et al., 2016, 2017) have facilitated the formation of a new family of reserving models based on these techniques. Gabrielli and Wüthrich (2018) and Wüthrich (2018b) applied Artificial Neural Networks (ANN) to predict claim reserves, while Baudry and Robert (2019), Lopez, Milhaud, and Théron (2019) and Wüthrich (2018a) used a tree-based algorithm, extremely randomized trees (Geurts, Ernst, & Wehenkel, 2006) and regression trees respectively for that purpose. Gabrielli (2019) and Gabrielli, Richman, and Wüthrich (2018) demonstrated that it is possible to embed traditional Chain Ladder techniques (ODP model) into a neural network framework. This algorithm was also used by Kuo (2018) in order to predict the expected future payments. Ramos-Pérez, Alonso-González, and Núñez Velázquez (2021b) combined ANNs with Random Forests (Breiman, 2001) and Gradient Boosting with regression trees (Friedman, 2000) in order to predict general insurance reserves. In addition to the previous reserving models, Duma, Twala, Marwala, and Nelwamondo (2011) applied support vector machines to classify data in homogeneous groups of risks before the reserve calculation.

The stochastic reserving model presented in this paper (Mack-Net) combines Recurrent Neural Networks (Rumelhart, Hinton, & Williams, 1986) with Mack's model in order to produce more accurate reserve predictions and risk measures. For each individual triangle, an ensemble of Recurrent Neural Networks (RNNs) is fitted in order to forecast both the future payments and the Mack's model parameters. In a second stage, a bootstrap method based on Mack's model is combined with the former predictions in order to compute a full reserve distribution. Consequently, the proposed model has the aim of improving the performance of Mack's model by applying RNNs, which can learn more features than the Chain Ladder technique. Mack-Net model differs from other methods in many ways but the two main differences are explained. First of all, most of the existing reserving models based on machine and deep learning does not produce an estimate of the reserves variability. The few of them that can produce this estimation need to assume a pre-defined theoretical distribution for the payments

or incurred cost. Nevertheless, the suggested methodology produces a full reserve distribution without considering any assumption about the payments or incurred cost distribution. Second, information from the same portfolios of several entities tend to be used for fitting reserving models based deep or machine learning techniques. As suggested by regulations like Solvency II Directive, actuaries must aggregate data in homogeneous risk groups, leading to a situation where individual companies do not have available several triangles with similar characteristics to fit the former models. Besides regulatory requirements, if individual companies split triangles with similar characteristics into different pieces, the resulting triangles will not be enough robust in most of the cases. As only one triangle is needed to fit the Mack-Net model, this problem is not present in the suggested methodology. The implementation of the MackNet model in R, the database used for fitting the models and code examples are available in <https://github.com/EduardoRamosP/MackNet>

The proposed model has the aim of producing a more appropriate risk estimation and reserve distribution than other stochastic reserving approaches. Regulations such as Solvency II, Swiss Solvency Test or IFRS promote the use of stochastic models. For example, Risk Adjustment of IFRS 17 has to be based on a certain percentile of the reserve distribution and Reserving Risk of Solvency II Directive can be derived from an stochastic reserving model. Therefore, the calculation of an accurate reserve distribution can lead to lower solvency requirements and less liabilities (Risk Adjustment). It is also worth mentioning that an accurate estimation of the risk profile can lead to a more efficient risk strategy, better portfolio management actions and, therefore, an optimization of the profit-risk indicators. Since the Financial Crisis of 2007–2008, the valuation of financial institutions is not only based on the future profit but also on its volatility or uncertainty. Nowadays, profit-risk indicators are particularly relevant for the valuation of financial institutions.

The rest of the paper proceeds as follows: Section 2 defines the validations metrics and models used as benchmark to assess the performance of the suggested method. In Section 3, the theoretical background and architecture of the Mack-Net model are explained. Empirical results of the benchmark and proposed model are shown in Section 4. Finally, Section 5 presents the main conclusions drawn from the results shown in Section 4.

## 2. Benchmark model and validation metrics

### 2.1. Benchmark model

This paper presents an extension, based on RNNs, of the traditional Mack's model. Thus, this approach (Mack, 1993) and its bootstrap implementation (England & Verrall, 2006) will be used as benchmark for validating the proposed model.

The main characteristic of this model compared to others based on Chain Ladder is the lack of assumptions about the underlying distribution of the payments. Mack's model assumes that cumulative payments,  $D_{ij}$ , or incurred cost have the following variance and expected value:

$$E[D_{ij}] = \hat{f}_j D_{i,j-1} \quad Var[D_{ij}] = \hat{\sigma}_j^2 D_{i,j-1} \quad (1)$$

where  $i = (1, 2, \dots, I)$  indicates the accident or underwriting year and  $j = (1, 2, \dots, I)$  the development year. As explained by Mack (1993), the parameters of the previous expressions are calculated as follows:

$$\hat{f}_j = \frac{\sum_{i=1}^{I-j+1} D_{ij}}{\sum_{i=1}^{I-j+1} D_{i,j-1}} \quad \hat{\sigma}_j^2 = \frac{1}{I-j-1} \sum_{i=1}^{I-j+1} D_{i,j-1} \left( \frac{D_{ij}}{D_{i,j-1}} - \hat{f}_j \right)^2 \quad (2)$$

where  $\{\hat{f}_j : j = (2, 3, \dots, I)\}$  and  $\{\hat{\sigma}_j^2 : j = (2, 3, \dots, I)\}$ . The residuals needed for the bootstrap method (England & Verrall, 2006) are calculated as defined below:

$$\hat{r}_{ij} = \frac{\sqrt{D_{i,j-1}} * \left( \frac{D_{ij}}{D_{i,j-1}} - \hat{f}_j \right)}{\hat{\sigma}_j} \quad (3)$$

To obtain the final residuals, the bias adjustment is added accordingly to the expression suggested by (England & Verrall, 2006):

$$\hat{r}_{ij} = \sqrt{\frac{N}{N-p}} * \frac{\sqrt{D_{i,j-1}} * \left( \frac{D_{ij}}{D_{i,j-1}} - \hat{f}_j \right)}{\hat{\sigma}_j} \quad (4)$$

where  $N$  is the total number of residuals and  $p$  the number of parameters. Hence, the resampled link ratios are obtained as follows:

$$f_{ij}^B = \hat{f}_j + r_{ij}^B \frac{\hat{\sigma}_j}{\sqrt{D_{i,j-1}}} \quad (5)$$

where  $B$  refers to the number of upper triangles to be simulated and  $r_{ij}^B$  to the residual resampled in the position  $(i, j)$  of the  $B$ th triangle. Taking into consideration  $D_{i,j}$  and the resampled link ratios, a new set of development factors,  $\hat{f}_j^B$ , is computed. Typically, a zero mean adjustment is applied to the residuals in order to ensure that the mean of the stochastic process is the same as the deterministic Chain Ladder method, which is fully dependent on  $\hat{f}_j$ .

The lower triangle ( $D_{i,j}$  where  $i+j > I+1$ ) is predicted by combining  $\hat{f}_j^B$  and the upper triangle ( $D_{i,j}$  where  $i+j \leq I+1$ ). Then, the process variance is incorporated to the lower triangle by adding the following expression:  $\hat{\sigma}_j r_{ij}^B \sqrt{D_{i,j-1}}$ . In case further details about the bootstrap method are needed, refer to England and Verrall (2006) and Joseph Lo (2011).

Although the methodology proposed in this paper merges RNNs with Mack's model, other two approaches have been selected as benchmark: Stacked-ANN (Ramos-Pérez et al., 2021b) and Changing Settlement Rate. The first model combines ANNs with Random Forests and Gradient Boosting with regression trees (Friedman, 2000) to predict general insurance reserves. The stochastic procedure of Stacked-ANN assumes that payments follow a log-normal distribution. On the other hand, Changing Settlement Rate (CSR) is a Bayesian Markov Chain Monte-Carlo model. The prior distributions and the simulation approach are proposed by Meyers (2015).

## 2.2. Validation metrics

General insurance companies are asked by insurance regulations like Solvency II Directive and Solvency Swiss Test to evaluate their reserve variability. Thus, the accuracy and variability of Mack-Net model (Section 3) will be validated and compared with the benchmark model defined in Section 2.1.

To assess the accuracy of the reserve predicted by the models, the following error measure will be computed for every line of business:

$$\%RMSE(U^t) = \sqrt{\frac{1}{K} \sum_{n=1}^K \left( \frac{\hat{U}_n^t - U_n^t}{U_n^t} \right)^2} * 100 \quad (6)$$

$$\%MAE(U^t) = \frac{100}{K} \sum_{n=1}^K \left| \frac{\hat{U}_n^t - U_n^t}{U_n^t} \right| \quad (7)$$

where  $K$  is the total number of companies analysed,  $\hat{U}_n^t$  the ultimate cost predicted by the reserving model for the  $n$ th company and  $U_n^t$  the ultimate cost that was actually observed. The Model Confidence Test (Hansen, Lunde, & Nason, 2011) will be also applied to produce a more robust comparison of models accuracy. This procedure consists on a sequence of tests which permit the identification of the best models at a certain confidence level.

In addition to the previous error measures, the reserve variability produced by the different stochastic reserving models will be assessed by applying the statistical test introduced by Kupiec (1995). The aim of this test is to validate the Value-at-Risk (VaR) by comparing the number of VaR breaches with the percentile selected for calculating the VaR. In this paper, the percentile selected for evaluating the reserve variability is  $\alpha = 0.995$ , which is the level set up by Solvency II to calculate the risk of insurance companies. The empirical results of the test and  $\%RMSE(R')$  are collected in Section 4.2.

## 3. Data and Mack-Net architecture

The aim of this section is to explain the architecture of the Mack-Net model. To do so, this section has been divided into three different subsections. The inputs of the model are described in the first one, the ensemble of RNNs is explained in the second subsection and, in the last one, the bootstrap method to obtain a reserve distribution is presented. To support the explanation, Fig. 1 shows the architecture of the proposed stochastic reserving model.

### 3.1. Data source and model inputs

The starting point of the Mack-Net model is the definition of inputs used within the ensemble of RNNs. Hence, the goal of this subsection is to define the database used, as well as the response and explicative variables for fitting the RNNs ensemble.

The database of Schedule P of the NAIC Annual Statement (available on CAS website) is selected for fitting and validating the Mack-Net model. The paid data, incurred cost and premiums available in the previous database were collected from general insurance companies from the US. The range of accident years contained in the Schedule P database is 1988–1997. As the upper and lower triangles are included, ten development years are available for each accident year.

The benchmark and the proposed model will be fitted to the 200 loss triangles selected by Meyers (2015) from the Schedule P database. This author pointed out that one of the main mistakes that could be made with NAIC Schedule P data is selecting triangles from insurance companies that have experienced significant changes in business operations. Therefore, the net-on-gross premiums ratio and the coefficient of variation of the net premiums were used by Meyers to identify those companies that made changes in their business operations or reinsurance structure. Taking into consideration these indicators, Meyers selected 50 loss triangles from each of the following lines of businesses: Commercial Auto (CA), Private Passenger Auto Liability (PA), Workers' Compensation (WC) and Other Liability (OL). The codes of the companies selected can be found in Meyers (2015).

It is worth mentioning that loss triangles are considered the primary method to organize the observed incurred cost or payments for general insurance reserving purposes. Loss triangles show the total losses of different underwriting or accident years at various valuation dates. This data shows the claim settlement speed, ultimate cost and policyholders' behaviour. Thus, this data is normally not available because insurance entities do not make public this information. Schedule P database is used in the academic field by several authors (such as Kuo, 2018; Leong, Wang, & Chen, 2014; Meyers, 2015, or Ramos-Pérez et al., 2021b) because it offers the possibility of testing triangles from numerous companies and different lines of business.

Before starting with the definition of the explanatory and response variables, it is worth mentioning that the last diagonal of the triangle is selected as a test set for fitting the ensemble of RNNs. Thus, the remaining triangle (9 development years) is used to train the algorithms. The sequences used as explanatory variables,  $X_i$ , and the response variable,  $Y$ , are the following:

$$Y = C_{ij}^* = \frac{C_{ij}}{P_i} \quad (8)$$

$$X_1 = (C_{ij-1}^*, C_{ij-2}^*, \dots, C_{ij-8}^*) = \left( \frac{C_{ij-1}}{P_i}, \frac{C_{ij-2}}{P_i}, \dots, \frac{C_{ij-8}}{P_i} \right) \quad (9)$$

$$X_2 = (DY_{j-1}^*, DY_{j-2}^*, \dots, DY_{j-8}^*) = \left( \frac{DY_{j-1}}{I}, \frac{DY_{j-2}}{I}, \dots, \frac{DY_{j-8}}{I} \right) \quad (10)$$

$$X_3 = (R_{j-1}^*, \dots, R_{j-8}^*) = \left( \frac{\sum_{i=1}^{I-j+2} D_{ij-1}^*}{\sum_{i=1}^{I-j+2} \frac{IC_{ij-1}}{P_i}}, \dots, \frac{\sum_{i=1}^{I-j+9} D_{ij-8}^*}{\sum_{i=1}^{I-j+9} \frac{IC_{ij-8}}{P_i}} \right) \quad (11)$$

where  $P_i$  is the premium,  $C_{ij}$  is the incremental payment,  $D_{ij}^*$  is equal to  $D_{ij}/P_i$ ,  $I$  the total number of accident years and  $IC_{ij}$  represents the

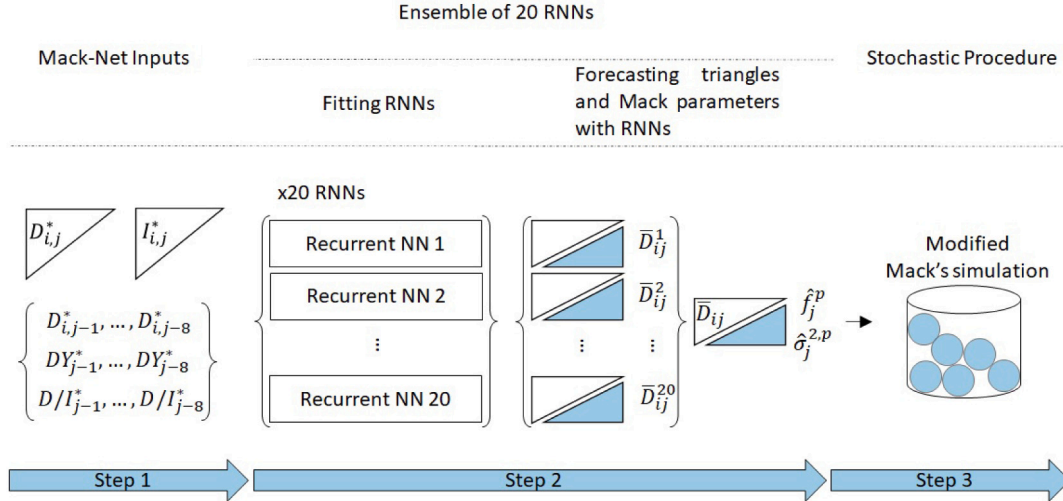


Fig. 1. Mack-Net model architecture.

incurred cost of the accident year  $i$  and development  $j$ .  $R_j^*$  is the scaled cumulative payments between the scaled incurred cost. Thus,  $X_1$ ,  $X_2$  and  $X_3$  are the time series used as input in the RNNs to predict the next year payment. Two remarks about the variables need to be made:

1. Payments and development years were scaled.  $DY_j$  were initialized as one and then scaled to the range  $[0, 1]$ . To do so,  $DY_j$  were divided between the total number of accident years,  $I$ . In the case of payments, premiums ( $P_i$ ) play the role of exposure measure. The use of exposure measures is widely used in reserving models, otherwise the changes in the claims settlement speed will be mixed with variations in the business volume.
2. As it can be observed in the formal definition of the explanatory variables, the last 8 observations of each variable are selected. This number was chosen due to the size of the triangles. Ten development years are available but only 9 of them are used during the training of the RNNs. The other development year is used as a test set. Thus, as the aim of the RNNs is to predict the value in  $t$ , the maximum lag available for training the algorithms is  $t - 8$ . Each RNN forecasts the next payment by taking into consideration the information available in the last 8 periods.

Before concluding this subsection it is worth mentioning that the model can be applied to predict the incurred cost. To do so,  $X_1$  and  $Y$  should be substituted by the following expressions:

$$Y' = i_{ij}^* = \frac{i_{ij}}{P_i} \quad (12)$$

$$X'_1 = \left( IC_{ij-1}^*, IC_{ij-2}^*, \dots, IC_{ij-8}^* \right) = \left( \frac{IC_{ij-1}}{P_i}, \frac{IC_{ij-2}}{P_i}, \dots, \frac{IC_{ij-8}}{P_i} \right) \quad (13)$$

where  $i_{ij}$  is the incremental incurred cost. The method and calculations that will be explained in Sections 3.2 and 3.3 are the same regardless of the variable to be predicted (payments or incurred cost).

### 3.2. Ensemble of RNNs

As shown in Fig. 1, once the model inputs are prepared, 20 Recurrent Neural Networks composed of several Fully Connected (FC) and Long-Short Term Memory (LSTM) layers (Fig. 2) are fitted. The number of Recurrent Neural Networks fitted is high enough to obtain the average model prediction regardless their initial weights. This strategy was also applied by Kuo (2018).

It has to be pointed out that a skip connection between 'FC Layer 1' and 'FC Layer 5' has been included. This type of connection, introduced

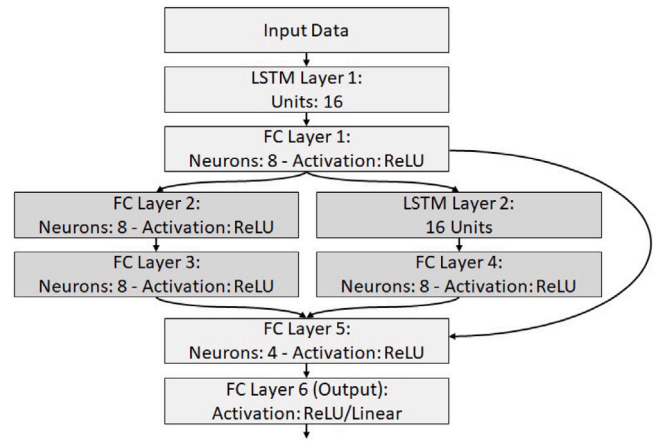


Fig. 2. Recurrent neural network architecture.

by He, Zhang, Ren, and Sun (2016) in the field of image recognition with Convolutional Neural Networks, gives the possibility to skip the training of a part of the Neural Network architecture. During the learning process, the RNN will decide by itself if 'FC Layer 3' and/or 'FC Layer 4' send information to 'FC Layer 5'. Apart from giving to the network the possibility to simplify the structure by skipping layers, this kind of connection helps to avoid the problem of vanishing gradients by using the activation of a previous layer until the skipped one learns its weights.

To take temporal dependencies into consideration, the first layer of every RNN is a LSTM cell. This structure was introduced by Hochreiter and Schmidhuber (1997) for managing time series. Fig. 3 and the following expressions define the LSTM architecture:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (14)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (15)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (16)$$

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \quad (17)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (18)$$

$$h_t = o_t \tanh(C_t) \quad (19)$$

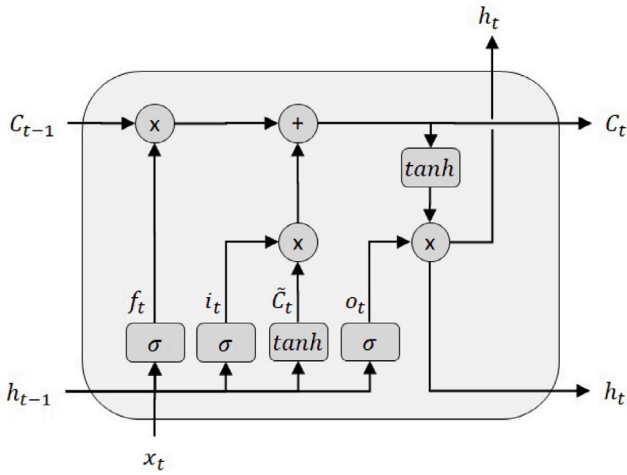


Fig. 3. LSTM structure.

where  $W_f, W_i, W_c, W_o, b_f, b_i, b_c$  and  $b_o$  represent the weights and bias of the RNNs and  $\sigma(x)$  the logistic sigmoid function.

The main characteristics of the RNNs are defined below:

- The algorithm used to optimize the weights is Adaptive Moment Estimation (ADAM), which was developed by Kingma and Ba (2014). Considering current and previous gradients, this procedure allows to implement a progressive adaptation of the initial learning rate. These authors suggested the following default values for the ADAM parameters:  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . In this paper, the initial learning rate is set to  $\delta = 0.01$  and the default ADAM parameters are used during the training process.
- The batch size is equal to the number of observations of the training set
- The backward pass calculations are done taking the mean squared error as loss function.
- Each individual algorithm within the ensemble is randomly initialized. Glorot initializer (Glorot & Bengio, 2010) is used for the LSTM weights responsible of transforming linearly the inputs, while the LSTM weights for the linear transformations within the recurrent states are initialized with the orthogonal approach suggested by Saxe, McClelland, and Ganguli (2013).
- In order to avoid overfitting, the level of dropout regularization  $\theta$  (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is set to 5%.

The training of the RNNs has been implemented using the Keras (Chollet et al., 2015) and Tensorflow (Abadi et al., 2015). As previously stated, the initial weights of the RNNs are randomly initialized. Thus, the lower triangle predicted by every single algorithm is going to be different. Once the lower triangles are predicted with each RNN, the Mack-Net parameters are computed with the predicted cumulative payments or incurred cost as follows:

$$\hat{f}_j^p = \frac{\sum_{i=j+2}^I \bar{D}_{ij}}{\sum_{i=j+2}^I \bar{D}_{i,j-1}} \quad (20)$$

$$\hat{\sigma}_j^{2,p} = \frac{1}{I-j-1} \sum_{i=1}^I \bar{D}_{i,j-1} \left( \frac{\bar{D}_{ij}}{\bar{D}_{i,j-1}} - \bar{f}_j \right)^2 \quad (21)$$

where  $\bar{f}_j$  is equal to  $\sum_{i=1}^I \bar{D}_{ij} / \sum_{i=1}^I \bar{D}_{i,j-1}$  and  $\bar{D}_{ij} = \sum_{k=1}^K \bar{D}_{ij}^k / K$ . In addition,  $\bar{D}_{ij}^k$  represents the cumulative payments of the lower triangle ( $i+j > I+1$ ) predicted by  $k$ th RNN and the observed values of the upper triangle ( $i+j \leq I+1$ ).  $K$  is the number of RNNs included in the ensemble. Similar to the notation used in Section 2.1,  $\{\hat{f}_j^p : j = (2, 3, \dots, I)\}$  and  $\{\hat{\sigma}_j^{2,p} : j = (2, 3, \dots, I)\}$ .

As it can be derived from the model definition, rather than using the traditional Mack parameters described in Section 2.1, Mack-Net model parameters are estimated by taking into consideration the predictions made by the ensemble of RNNs. Therefore, the central scenario of the stochastic Mack-Net model is equal to the reserve predicted by the ensemble of RNNs, while the mean of the traditional Mack's model converge to the reserve estimated with the deterministic Chain Ladder method. As any other general insurance reserving methodology such as Mack's model, CSR or Stacked-ANN, the approach suggested by this paper is valid until a new diagonal of the loss triangle is available.

### 3.3. Stochastic procedure

As previously stated, the bootstrap method of Mack-Net is based on the traditional Mack's model. This last methodology has been selected as reference for producing the full reserve distribution due to the two following reasons. First, Mack's model derives the distribution by focusing on the two first moments. In contrast to most of the stochastic reserving models, this approach does not make any assumption about the theoretical distribution followed by incurred cost or cumulative payments. Changes in policyholders' behaviour, reinsurance structures, regulation and number of policy holders can modify significantly the payments or incurred cost collected by loss triangles. Therefore, a free-distribution model is especially appropriate for insurance companies and regulators because the previous portfolio changes happen quite often in the sector. Second, Mack's model is already applied by insurance companies to comply with regulations such as Solvency II Directive, Swiss Solvency Test or IFRS. Thus, the bootstrap method of Mack-Net is familiar and aligned with the procedures already used by the insurance market.

As no assumption about the underlying distribution of cumulative payments or incurred cost is taken, the expected value and variance of Mack-Net model is defined as follows:

$$E[D_{ij}] = \hat{f}_j^p \bar{D}_{i,j-1} \quad Var[D_{ij}] = \hat{\sigma}_j^{2,p} \bar{D}_{i,j-1} \quad (22)$$

The Mack-Net model uses the predictions made by the ensemble of RNNs to determine the mean and variance of the reserve distribution. By doing so, the forecasting power of deep and machine learning techniques are taken into consideration. The calculation of residuals needed for the bootstrap method is based on the expression provided by England and Verrall (2006) for Mack's model:

$$\hat{r}_{ij}^p = \frac{\sqrt{\bar{D}_{i,j-1}} * \left( \frac{\bar{D}_{ij}}{\bar{D}_{i,j-1}} - \bar{f}_j \right)}{\hat{\sigma}_j^p} \quad (23)$$

where  $\{\hat{r}_{ij}^p : j = (2, \dots, I); i = (1, \dots, I)\}$ . As in Mack's model, Mack-Net model is bias adjusted in accordance with the procedure suggested by (England & Verrall, 2006):

$$\hat{r}_{ij}^p = \sqrt{\frac{N}{N-p}} \frac{\sqrt{\bar{D}_{i,j-1}} * \left( \frac{\bar{D}_{ij}}{\bar{D}_{i,j-1}} - \bar{f}_j \right)}{\hat{\sigma}_j^p} \quad (24)$$

As with Mack's model,  $p$  is equal to the number of development factors. Once the set of residuals has been calculated, the resampled upper triangles of link ratios are calculated as follows:

$$f_{ij}^{B,p} = \hat{f}_j^p + r_{ij}^{B,p} \frac{\hat{\sigma}_j^p}{\sqrt{\bar{D}_{i,j-1}}} \quad (25)$$

where  $B$  refers to the number of upper triangles to be simulated and  $r_{ij}^{B,p}$  to the residual resampled in the position  $(i, j)$  of the  $b$ th triangle. Similar to the Mack's model presented in Section 2.1, a zero mean adjustment should be applied to the residuals in order to avoid

**Table 1**

Average  $\hat{f}_j^p$  and  $\hat{f}_j$  by line of business (Paid data).  
Source: Own elaboration.

Dev year	CA		PA		WC		OL	
	Mack	Mack Net	Mack	Mack Net	Mack	Mack Net	Mack	Mack Net
1	1.90	1.88	1.77	1.75	2.21	2.59	3.12	3.10
2	1.35	1.35	1.22	1.22	1.29	1.38	1.72	1.67
3	1.16	1.15	1.10	1.10	1.13	1.15	1.34	1.30
4	1.08	1.06	1.06	1.05	1.07	1.08	1.18	1.16
5	1.04	1.03	1.03	1.02	1.04	1.04	1.11	1.07
6	1.02	1.01	1.01	1.01	1.02	1.02	1.04	1.03
7	1.00	1.01	1.01	1.00	1.02	1.02	1.02	1.01
8	1.01	1.00	1.00	1.00	1.01	1.01	1.02	1.01
9	1.00	1.00	1.00	1.00	1.01	1.01	1.01	1.00

deviations between the simulated and theoretical mean. The resampled development factors ( $\tilde{f}_j^{B,p}$ ) are

$$\tilde{f}_j^{B,p} = \frac{\sum_{i=1}^{I-j+1} \bar{D}_{i,j-1} \hat{f}_{ij}^{B,p}}{\sum_{i=1}^{I-j+1} \bar{D}_{i,j-1}} \quad (26)$$

It is worth mentioning that the previous calculation is carried out with the upper triangle ( $i + j \leq I + 1$ ). Then,  $\bar{D}_{i,j-1}$  can be substituted by  $D_{i,j-1}$ . The resampled development factors,  $\tilde{f}_j^{B,p}$ , are used to calculate the lower triangle ( $i + j > I + 1$ ) by applying the Chain Ladder methodology:

$$\hat{D}_{ij} = \bar{D}_{i,j-1} \tilde{f}_j^{B,p} \quad (27)$$

Then, as well as in the case of Mack's model, the process variance is included in the Mack-Net model in order to take into consideration the randomness in future outcomes:

$$\hat{D}_{ij} = \hat{D}_{ij} + \hat{\sigma}_{ij}^{B,p} \sqrt{\bar{D}_{i,j-1}} \quad (28)$$

Notice that, as the procedure is applied recursively,  $\bar{D}_{i,j-1}$  is used in the previous equation only when the simulation refers to year after the last diagonal observed. In the rest of the cases, the recurrence is applied and, thus,  $\bar{D}_{i,j-1}$  substituted by  $\hat{D}_{i,j-1}$ . As explained in Section 2.1 for Mack's model, this procedure (England & Verrall, 2006) allows the recognition of the process variance, which is the variability in the forecasts of future payments.

#### 4. Model fitting and results

In this section, Mack-Net parameters by line of business and the comparison between the performance of the benchmark and the Mack-Net model are presented.

##### 4.1. Fitting of Mack-Net model

This subsection presents Mack-Net parameters by line of business. As stated before, the model has been fitted individually to each of the 200 triangles selected by Meyers (2015) from the Schedule P of the NAIC Annual Statement. For further details about the database refer to Section 3.1.

As explained in Section 3.2, once the ensemble of RNNs has been fitted, the Mack-Net architecture proceeds with the calculation of  $\hat{f}_j^p$  and  $\hat{\sigma}_j^{2,p}$ . Similar to  $\hat{f}_j$  and  $\hat{\sigma}_j^2$  in Mack's model, Mack-Net parameters define the variance and expected value of the cumulative payments or incurred cost. Thus, Tables 1–4 present a comparison between the average Mack and Mack-Net parameters by line of business.

The averages of the development factors of the payment models (Table 1) present non-significant differences in most of the cases. The only remarkable differences are the second development factor of OL, and the first and second parameters of WC. Mack-Net development

**Table 2**

Average  $\hat{f}_j^p$  and  $\hat{f}_j$  by line of business (Incurred data).  
Source: Own elaboration.

Dev year	CA		PA		WC		OL	
	Mack	Mack Net	Mack	Mack Net	Mack	Mack Net	Mack	Mack Net
1	1.27	1.33	1.14	1.15	1.31	1.40	1.45	1.56
2	1.09	1.09	1.03	1.03	1.07	1.09	1.21	1.20
3	1.03	1.03	1.01	1.01	1.02	1.03	1.07	1.07
4	1.01	1.01	1.00	1.00	1.01	1.01	1.03	1.03
5	1.00	1.01	1.00	1.00	1.00	1.01	1.04	1.01
6	0.99	1.01	1.00	1.00	1.00	1.01	1.01	1.02
7	1.00	1.01	1.00	1.00	1.00	1.01	1.00	1.01
8	1.00	1.01	1.00	1.00	1.00	1.01	1.01	1.01
9	1.00	1.01	1.00	1.00	1.00	1.01	1.00	1.01

**Table 3**

Average  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$  by line of business (Paid data).  
Source: Own elaboration.

Dev year	CA		PA		WC		OL	
	Mack	Mack Net	Mack	Mack Net	Mack	Mack Net	Mack	Mack Net
1	13.25	12.43	12.96	11.96	14.98	13.94	26.94	25.47
2	6.82	5.96	5.73	5.09	6.06	5.43	10.35	10.21
3	4.43	3.76	3.61	3.29	4.07	3.70	6.67	5.58
4	2.97	2.31	2.59	2.55	3.07	2.78	4.96	4.29
5	1.76	1.39	1.40	1.47	1.55	1.76	3.21	2.30
6	0.98	0.74	0.87	0.79	1.46	1.19	1.95	1.23
7	0.65	0.44	0.68	0.50	1.10	0.86	0.97	0.54
8	0.42	0.26	0.19	0.24	0.71	0.55	0.78	0.42
9	0.32	0.17	0.15	0.15	0.53	0.36	0.44	0.18

**Table 4**

Average  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$  by line of business (Incurred data).  
Source: Own elaboration.

Dev year	CA		PA		WC		OL	
	Mack	Mack Net	Mack	Mack Net	Mack	Mack Net	Mack	Mack Net
1	7.95	7.29	10.11	9.23	15.13	13.85	13.69	12.67
2	4.67	4.11	4.79	4.35	9.48	8.23	9.00	8.56
3	3.19	2.71	3.06	3.04	4.09	3.50	5.24	4.40
4	2.29	1.96	1.62	1.57	2.69	2.19	4.01	3.58
5	1.54	1.27	1.18	1.10	2.12	1.64	3.66	2.53
6	1.13	0.97	0.78	0.71	1.79	1.34	1.50	1.18
7	0.77	0.68	0.31	0.38	1.30	0.94	1.15	0.78
8	0.41	0.46	0.18	0.26	0.80	0.64	0.52	0.48
9	0.33	0.38	0.15	0.18	0.49	0.41	0.43	0.34

factors are lower than Mack's parameters in all the lines of business with the only exception of Workers' Compensation. This can be proved by computing the multiplicative development factors (product of development factors by line of business and model).

With regard to the incurred cost development factors (Table 2), the differences between both models are minor in the case of CA and PA. However, they become more relevant in the case of WC and OL, especially in the case of the first development year. It is also worth mentioning that Mack-Net development factors are higher than Mack parameters regardless of the line of business.

Before analysing the differences between  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$ , it is worth mentioning that previous cumulative payments or incurred cost play a key role in the model variance, defined in equations 2 and 20 for Mack and Mack-Net model respectively. Thus,  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$  have to be analysed by taking into consideration the analysis of the development factors explained in the previous paragraphs.

With regard to the models for paid loss data,  $\hat{\sigma}_j^{2,p}$  and  $\hat{\sigma}_j^2$  show non-material differences. Nevertheless, it has to be pointed out that Mack parameters are higher than those of Mack-Net in every line of business. As the paid development factors of the Mack model are also higher, the pattern shown in Table 3 reveals that Mack model generates a higher volatility than the proposed methodology.

**Table 5** $\%RMSE(U')$  by model and line of business.

Source: Own elaboration.

Line of business	Mack paid	Mack-Net paid	Mack incurred	Mack-Net incurred	CSR	Stacked ANN
CA	7.98%	6.80%	8.18%	8.03%	9.29%	8.92%
PA	6.06%	5.01%	2.62%	4.26%	5.46%	7.78%
WC	7.86%	6.77%	8.15%	6.99%	13.29%	7.36%
OL	20.20%	17.31%	17.38%	13.48%	27.78%	19.80%

**Table 6** $\%MAE(U')$  by model and line of business.

Source: Own elaboration.

Line of business	Mack paid	Mack-Net paid	Mack incurred	Mack-Net incurred	CSR	Stacked ANN
CA	5.96%	4.78%	5.46%	5.40%	6.35%	6.86%
PA	3.81%	3.44%	1.90%	2.86%	3.68%	3.76%
WC	5.32%	4.60%	5.27%	4.51%	6.01%	4.65%
OL	13.41%	12.16%	11.34%	9.88%	18.29%	13.47%

Table 4 shows that Mack's parameters are higher than those of the Mack-Net model fitted with incurred cost data. In contrast to the models for paid loss data, this effect is partially offset by the Mack-Net development factors that, as shown in Table 2, are higher than those of the Mack's model.

As incurred cost includes the payments and the reserve set up by claim adjusters, this variable should be closer to the ultimate claim cost than the payments. Thus, development factors and reserve volatility should be lower in the case of the models fitted with incurred cost data. The comparison of the average parameters of the models for incurred (Tables 2 and 4) and paid loss data (Tables 1 and 3) reveals that this trend is followed by both models.

#### 4.2. Comparison against benchmark models

This subsection compares the performance of the Mack-Net model with the original methodology proposed by Mack. The variability and accuracy will be compared with the metrics and tests shown in Section 2.2.

As previously explained, the aim of the Mack-Net model is to improve the accuracy of the traditional Mack's methodology by using machine and deep learning algorithms and techniques such as RNNs. Tables 5 and 6 show the empirical results of the metrics selected for comparing the models accuracy.

With regard to the models for paid loss data, Mack-Net methodology improves the accuracy of the Mack's model in every line of business.  $\%RMSE(U')$  decreases by 14% in WC and OL, 15% in CA, and 17% in the case of PA. Similar improvements are also observed in terms of  $\%MAE(U')$ .

The comparison of the  $\%RMSE(U')$  and  $\%MAE(U')$  obtained from the models for incurred loss data shows that Mack-Net model outperforms the Mack's procedure in all the lines of business with the only exception of PA. It is worth mentioning that the accuracy of the Mack-Net model is especially higher in OL, which is the line of business with the longer duration of liabilities. Thus, an appropriate estimation of reserves is particularly relevant in this case. Empirical results demonstrate that Mack-Net model also outperforms general insurance reserving approaches based on Markov Chain Monte-Carlo or machine learning such as CSR or Stacked-ANN.

Tables 7 and 8 show the ranking proposed by Model Confidence Set (MCS) considering  $\%RMSE(U')$  and  $\%MAE(U')$  as loss functions. This approach confirms the outcomes presented in the previous paragraphs. In fact, Mack-Net Incurred and Paid are ranked as the best and second best model respectively when all the lines of business are considered together (row 'Total').

**Table 7**Ranking of models according to MSC ( $\%RMSE(U')$ ) and  $\alpha = 0.05$ .

Source: Own elaboration.

Line of business	Mack paid	Mack-Net paid	Mack incurred	Mack-Net incurred	CSR	Stacked ANN
CA	2nd	1st	4th	3rd	5th	6th
PA	6th	3rd	1st	2nd	4th	5th
WC	6th	1st	5th	2nd	4th	3rd
OL	5th	2nd	3rd	1st	6th	4th
Total	5th	2nd	3rd	1st	6th	4th

**Table 8**Ranking of models according to MSC ( $\%MAE(U')$ ) and  $\alpha = 0.05$ .

Source: Own elaboration.

Line of business	Mack paid	Mack-Net paid	Mack incurred	Mack-Net incurred	CSR	Stacked ANN
CA	5th	1st	3rd	2nd	4th	6th
PA	6th	3rd	1st	2nd	5th	4th
WC	6th	3rd	5th	1st	4th	2nd
OL	5th	3rd	2nd	1st	6th	4th
Total	5th	2nd	3rd	1st	6th	4th

**Table 9**

Kupiec test (p-values) by model and line of business.

Source: Own elaboration.

Line of business	Mack paid	Mack-Net paid	Mack incurred	Mack-Net incurred
CA	$\geq 0.05$	$\geq 0.05$	$< 0.05$	$\geq 0.05$
PA	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$	$\geq 0.05$
WC	$< 0.05$	$< 0.05$	$< 0.05$	$\geq 0.05$
OL	$\geq 0.05$	$\geq 0.05$	$< 0.05$	$\geq 0.05$

With regard to the validation of the reserves variability, Kupiec test (Kupiec, 1995) is applied to assess the appropriateness of the reserve distribution generated by the stochastic process. Table 9 collects the p-values of the Kupiec test assuming a VaR percentile of  $\alpha = 0.995$ , which is the value for evaluating the risk profile of insurance companies under Solvency II Directive.

Companies included in each line of business have different volumes. This fact was taken into consideration within the Kupiec test by giving different weights to each company. The higher the standard deviation generated by the company, the higher the weight given to compute the Kupiec test. Table 9 collects the p-values by model and line of business.

According to the results of the models for paid loss data, Mack and Mack-Net methodologies are unable to produce an appropriate Value at Risk (VaR) for Workers Compensation. In the rest of lines of business, the excesses of the VaR estimated by both models are aligned with the confidence level selected ( $\alpha = 0.995$ ). It is worth mentioning that, as discussed in Section 4.1, Mack-Net parameters reveal a lower level of variance than those of the Mack's model. Thus, the higher accuracy of the Mack-Net model for paid loss data (Table 5) allows to generate appropriate risk measures with a lower level of variability.

In the case of the models for incurred cost, Mack-Net model passes the test in all the lines of business, while Mack's model fails the test in three out of four lines of business. As it will be presented in Table 10, the coefficients of variation generated by the Mack-Net model are lower than those of the traditional Mack's methodology. Nevertheless, Mack-Net model passes the test because the accuracy of the mean of the stochastic process is higher (Table 5).

Mack's model fails the test in most of the lines of business due to two reasons. First, the lower level of accuracy leads to higher differences between the actual reserve and the distribution function generated by the model. Second, the model is unable to generate a higher variance in order to offset the lack of accuracy. To illustrate this, Fig. 4 shows one company of the OL segment where Mack's model produces an inappropriate VaR. The observed ultimate is represented by a black dashed line.

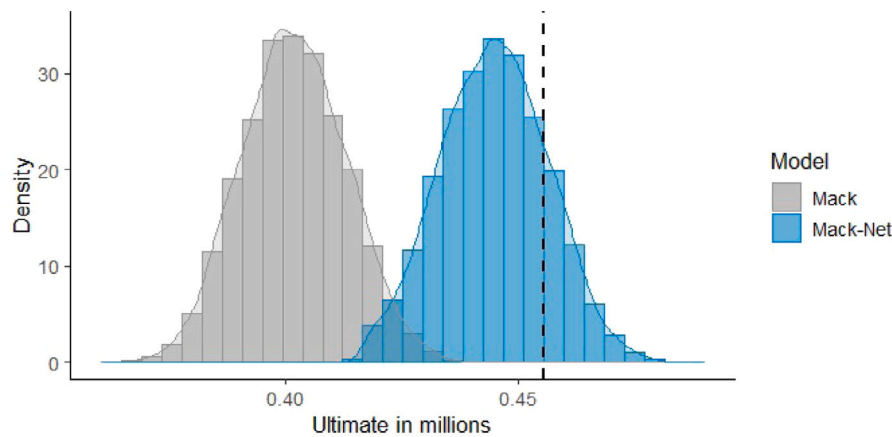


Fig. 4. Company code 620. Other liability.

Table 10

% of companies where Mack-Net  $CV(U^i) < \text{Mack } CV(U^i)$ .

Source: Own elaboration.

Line of business	Paid loss data	Incurred loss data
CA	56%	72%
PA	46%	66%
WC	56%	54%
OL	58%	68%

Table 11

Mack-Net: Training time and error versus input size.

Source: Own elaboration.

	5 Years	6 Years	7 Years	8 Years	9 Years	10 Years
Time index	100.00	102.14	105.48	113.86	126.00	140.55
Error index	100.00	70.73	50.05	33.73	19.02	7.08

With the goal of comparing the volatility generated by the different stochastic reserving models, Table 10 collects the % of companies where the coefficient of variation,  $CV(U^i)$ , of Mack-Net is lower than in the case of Mack's model:

The results shown in Tables 9 and 10 demonstrate that Mack-Net model does not need to produce higher coefficients of variation in order to generate more appropriate risk measures than Mack's model. Thus, the proposed methodology produces more efficient risk measures thanks to its predictive power.

Finally, on the trade-off between time and accuracy, Table 11 presents how training time and error change when the input size increases. The database used for fitting the algorithms (Schedule P of the NAIC Annual Statement) contains a maximum range of 10 years per each general insurance company. This analysis sets 5 years of data as the initial scenario and, then, the input size is increased until the maximum available data is reached. The results show that the error decreases in a higher extent than training time in relative terms. This is true even when the number of years is close to maximum (10 years). It is worth noticing that benchmark models accuracy will also decrease if the input size is reduced. Therefore, Table 11 shows the trade-off between time and accuracy of Mack-Net model in a standalone basis.

## 5. Conclusions

The Mack-Net model introduced in this paper has the aim of blending the traditional Mack's reserving model with deep and machine learning techniques. To do so, an ensemble of RNNs is fitted to the loss triangle. Then, the predictions of this ensemble are used for calculating Mack's model parameters. In this paper, the predictive power and reserve variability of the proposed architecture and the traditional

Mack's methodology are compared. Models were fitted to 200 incurred cost and paid loss triangles from NAIC Schedule P database (available on [CAS website](#)) in order to generate a robust comparison.

Three main conclusions are drawn from the results presented in Section 4. First, the comparison of the accuracy reveals that adding deep learning techniques to Mack's model improves the predictive power. With regard to the models fitted with paid data, this paper demonstrates that Mack-Net model outperforms Mack's model in every line of business. In the case of the models for incurred cost, Mack-Net is also more accurate than Mack's model in all the lines of business with the only exception of Personal Auto (PA). The accuracy of reserving models is particularly relevant for long-tail lines of business such as Other Liability (OL). In the case of this last portfolio, Mack-Net methodology reduces the RMSE by 14% and 22% when using paid and incurred cost data respectively. Empirical results demonstrate that Mack-Net model also outperforms other reserving approaches based on Markov Chain Monte-Carlo or machine learning such as Changing Settlement Rate or Stacked-ANN respectively.

Second, Kupiec test demonstrated that Mack-Net model generates more appropriate risk measures (Value at Risk) than the traditional Mack's methodology. With regard to the paid data, both models fail the test for Workers Compensation (WC). However, in the case of the incurred data, Mack-Net model passes the test in every line of business, while Mack's model fails the test in three out of four lines. Thus, Mack-Net model is not only more accurate but also generates a more appropriate Value at Risk (VaR). The confidence level selected for the VaR is  $\alpha = 0.995$ , which is the level required by Solvency II to evaluate the reserving risk in insurance companies.

Third, the blending of traditional approaches with deep learning techniques generates more efficient models for evaluating the reserving risk, which is the potential cost of deviations from the expected reserve. In other risks such as changes in equity price, the mean of the distribution does not play a relevant role (the mean of the returns are almost always close to zero). As the expected reserve cannot be easily predicted, this is not the case for reserving risk, where the appropriateness of the risk measures strongly depends on both the mean and the variance of the reserving model.

Due to the reasons explained in the previous paragraph, Mack-Net model is able to generate more appropriate risk measures with a lower variance. Thus, empirical results suggest that the proposed method is more efficient (in terms of risk assessment) than Mack's model because it generates a more reliable VaR with a lower variability.

Taking into consideration the previous conclusions, the blending of deep learning techniques with reserving models can be extended to improve the accuracy and risk measures derived from the use of other bootstrapping and Bayesian approaches. In the specific case of the Bayesian reserving models, deep learning algorithms could be applied in order to estimate the parameters of the distributions.

## CRedit authorship contribution statement

**Eduardo Ramos-Pérez:** Conceptualization, Methodology, Formal Analysis, Writing – original draft, Writing – review & editing. **Pablo J. Alonso-González:** Methodology, Validation, Investigation, Writing – original draft, Supervision, Project administration, Writing – review & editing. **José Javier Núñez-Velázquez:** Methodology, Validation, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project administration.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

The APC has been funded by Universidad de Alcalá.

## References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from <http://tensorflow.org/> (Software available from tensorflow.org).
- Antonio, K., & Plat, R. (2014). Micro-level stochastic loss reserving in general insurance. *Scandinavian Actuarial Journal*, 2014, 649–669.
- Baudry, M., & Robert, C. (2019). A Machine Learning approach for individual claims reserving in insurance. *Applied Stochastic Models in Business and Industry*, 1–29.
- Bornhuetter, R. L., & Ferguson, R. E. (1972). The actuary and IBNR. *Proceedings of the Casualty Actuarial Society*, 181–195.
- Braun, A., Schmeiser, H., & Schreiber, F. (2018). Return on risk-adjusted capital under solvency II: Implications for the asset management of insurance companies. *The Geneva Papers on Risk and Insurance - Issues and Practice*, 43(3), 456–472. <http://dx.doi.org/10.1057/s41288-017-0076-x>, Retrieved from <https://doi.org/10.1057/s41288-017-0076-x>.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <http://dx.doi.org/10.1023/A:1010933404324>.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners.
- Chollet, F. (2015). *Keras*. GitHub.
- Duma, M., Twala, B., Marwala, T., & Nelwamondo, F. (2011). Improving the performance of the support vector machine in insurance risk classification: A comparative study. In *Proceedings of the international conference on neural computation theory and applications (NCTA-2011)* (pp. 340–346).
- England, P. D. (2002). Addendum to Analytic and bootstrap estimates of prediction errors in claims reserving. *Insurance: Mathematics & Economics*, 31.
- England, P. D., & Verrall, R. J. (1999). Analytic and bootstrap estimates of prediction errors in claims reserving. *Insurance: Mathematics & Economics*, 25(3), 281–293.
- England, P. D., & Verrall, R. J. (2006). Predictive distributions of outstanding liabilities in general insurance. *Annals of Actuarial Science*, 221–270.
- Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 1189–1232.
- Gabrielli, A. (2019). A neural network boosted double over-dispersed poisson claims reserving model. *Astin Bulletin*, 1–36. <http://dx.doi.org/10.1017/asb.2019.33>.
- Gabrielli, A., Richman, R., & Wüthrich, M. (2018). Neural network embedding of the over-dispersed Poisson reserving model. <http://dx.doi.org/10.2139/ssrn.3288454>, (Available at SSRN, <https://ssrn.com/abstract=3288454>).
- Gabrielli, A., & Wüthrich, M. (2018). An individual claims history simulation machine. *Risks*, 6, 29.
- Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63, 3–42.
- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the international conference on artificial intelligence and statistics (AISTATS'10)*. Society for Artificial Intelligence and Statistics.
- Halliwel, L. (2009). *Modeling paid and incurred losses together* (pp. 1–40). Spring: CAS E-Forum.
- Hansen, P. R., Lunde, A., & Nason, J. M. (2011). The model confidence set. *Econometrica*, 79(2), 453–497.
- Happ, S., Merz, M., & Wüthrich, M. (2012). Claims development result in the paid-incurred chain reserving method. *Insurance: Mathematics & Economics*, 51, 66–72.
- Happ, S., & Wüthrich, M. (2013). Paid-incurred chain reserving method with dependence modeling. *Astin Bulletin*, 43, 1–20.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. (pp. 770–778). <http://dx.doi.org/10.1109/CVPR.2016.90>.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9, 1735–1780. <http://dx.doi.org/10.1162/neco.1997.9.8.1735>.
- Joseph Lo, A. (2011). Extending the mack bootstrap: Hypothesis testing and resampling techniques. *The Actuarial Profession*, 29–79.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. CoRR.
- Kremer, E. (1982). IBNR claims and the two-way model of ANOVA. *Scandinavian Actuarial Journal*, 1982.
- Kuo, K. (2018). DeepTriangle: A deep learning approach to loss reserving. CoRR, [abs/1804.09253](https://arxiv.org/abs/1804.09253).
- Kupiec, P. H. (1995). Techniques for verifying the accuracy of risk measurement models. *The Journal of Derivatives*, 3(2), 73–84. <http://dx.doi.org/10.3905/jod.1995.407942>.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444. <http://dx.doi.org/10.1038/nature14539>.
- Leong, W. K., Wang, S., & Chen, H. (2014). Back-testing the ODP bootstrap of the paid chain-ladder model with actual historical claims data. *Variance*.
- Lopez, O., Milhaud, X., & Thérond, P. (2019). A tree-based algorithm adapted to microlevel reserving and long development claims. *Astin Bulletin*, <http://dx.doi.org/10.1017/asb.2019.12>.
- Mack, T. (1991). A simple parametric model for rating automobile insurance or estimating IBNR claims reserves. *Astin Bulletin*, 21(1), 93–109. <http://dx.doi.org/10.2143/AST.21.1.2005403>.
- Mack, T. (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. *Astin Bulletin. The Journal of the International Actuarial Association*, 23(02), 213–225.
- Martínez-Miranda, M., Nielsen, B., & Verrall, R. (2012). Double chain ladder. *Astin Bulletin*, 42(1), 59–76.
- Martínez-Miranda, M., Nielsen, B., & Verrall, R. (2013). Double chain ladder and Bornhuetter-Ferguson. *North American Actuarial Journal*, 17, 101–113.
- McCullagh, P., & Nelder, J. (1989). *Chapman and Hall/CRC monographs on statistics and applied probability series, Generalized linear models* (2nd ed.). Chapman & Hall.
- Merz, M., & Wüthrich, M. V. (2010). Paid-incurred chain claims reserving method. *Insurance: Mathematics & Economics*, 46, 568–579.
- Meyers, G. (2015). *CAS monograph series: vol. 1, Stochastic loss reserving using Bayesian MCMC models*. CAS Ed., Casualty Actuarial Society.
- Pigeon, M., Antonio, K., & Denuit, M. (2014). Individual loss reserving using paid-incurred data. *Insurance: Mathematics & Economics*, 58, 121–131.
- Posthuma, B., Cator, E., Veerkamp, W., & Van Zwet, E. (2008). *Combined analysis of paid and incurred losses* (pp. 272–293). CAS E-Forum Fall.
- Quarg, G., & Mack, T. (2004). Munich chain ladder. *Blätter der Deutschen Gesellschaft für Versicherungs- und Finanzmathematik*, XXVI, 597–630.
- Ramos-Pérez, E., Alonso-González, P. J., & Núñez Velázquez, J. J. (2021a). Multi-transformer: A new neural network-based architecture for forecasting S&P volatility. *Mathematics*, 9(15), <http://dx.doi.org/10.3390/math9151794>.
- Ramos-Pérez, E., Alonso-González, P. J., & Núñez Velázquez, J. J. (2021b). Stochastic reserving with a stacked model based on a hybridized Artificial Neural Network. *Expert Systems with Applications*, 163, <http://dx.doi.org/10.1016/j.eswa.2020.113782>.
- Renshaw, A. E., & Verrall, R. J. (1998). A stochastic model underlying the chain-ladder technique. *British Actuarial Journal*, 4(4), 903–923. <http://dx.doi.org/10.1017/S1357321700000222>.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition, Vol. 1* (pp. 318–362). MIT Press.
- Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks.
- Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529, 484–489. <http://dx.doi.org/10.1038/nature16961>.
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., & Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. CoRR, [abs/1712.01815](https://arxiv.org/abs/1712.01815).
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1929–1958.
- Venter, G. (2008). *Distribution and Value of Reserves Using Paid and Incurred Triangles* (pp. 348–375). CAS E-Forum, Fall.
- Verrall, R. J. (2000). An investigation into stochastic claims reserving models and the chain-ladder technique. *Insurance: Mathematics & Economics*, 26(1), 91–99.
- Wüthrich, M. (2018a). Machine learning in individual claims reserving. *Scandinavian Actuarial Journal*, 2018, 465–480.
- Wüthrich, M. (2018b). Neural networks applied to chain-ladder reserving. *European Actuarial Journal*, 8, 407–436.

## References

- Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
- Acerbi, C. and B. Szekely (2014). Backtesting expected shortfall. *Risk*, 1–14.
- Aharon, D., I. Gavious, and R. Yosef (2010). Stock markets bubble effects on mergers and acquisitions. *The Quarterly Review of Economics and Finance* 50((4)), 456–70.
- Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Trans. EC* 16(3), 299–307.
- Andersen, T. (2009). *Encyclopedia of Complexity and System Sciences*, Chapter Stochastic volatility. Springer Verlag.
- Andersen, T. and B. Sorensen (1999). GMM estimation of a stochastic volatility model: A Monte Carlo study. *Journal of Business and Economic Statistics* 14, 329–352.
- Antonio, K., J. Beirlant, T. Holdemakers, and R. Verlaak (2006). Log-normal mixed models for reported claims reserves. *North American Actuarial Journal* 7, 1223–1237.
- Antonio, K. and R. Plat (2014). Micro-level stochastic loss reserving in general insurance. *Scandinavian Actuarial Journal* 2014, 649–669.
- Armano, G., M. Marchesi, and A. Murru (2005). A hybrid genetic-neural architecture for stock indexes forecasting. *Information Sciences* 170(1), 3–83.
- Arneric, J. and T. Poklepovic (2016). *Nonlinear Extensions of Asymmetric GARCH Model within Neural Network Framework*. AIRCC Publishing Corporation, Chennai, India.
- Artzner, P., F. Delbaen, J.-M. Eber, and D. Heath (1999). Coherent measures of risk. *Mathematical Finance* 9(3), 203–228.
- Back, Y. and H. Kim (2018). ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications* 113, 457–480.
- Baillie, R. T., T. Bollerslev, and H. O. Mikkelsen (1996). Fractionally integrated generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* 74(1), 3 – 30.

- Barron, A. (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 115–133.
- Baudry, M. and C. Robert (2019). A Machine Learning approach for individual claims reserving in insurance. *Applied Stochastic Models in Business and Industry*, 1–29.
- Bauwens, L., C. Hafner, and S. Laurent (2012). *Handbook of Volatility Models and Their Applications*. Wiley Handbooks in Financial E. Wiley.
- Bektiwati, A. and M. Irawan (2011). A RBF-EGARCH neural network model for time series forecasting. pp. 1–8.
- Bildirici, M. and O. Ersin (2009). Improving forecasts of GARCH family models with the artificial neural networks: An applicaiton to the daily returns in Istanbul Stock Exchange. *Expert Systems with Applications* 36(4), 7355–7362.
- Bildirici, M. and O. Ersin (2014). Modelling Markov Switching ARMA-GARCH Neural Networks Models and an Application to Forecasting Stock Returns. *Hindawi Publishing Corporation: The Scientific World Journal* 2014.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag.
- Blier Wong, C., H. Cossette, L. Lamontagne, and E. Marceau (2021). Machine learning in pc insurance: A review for pricing and reserving. *Risks* 9(1).
- Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* 31(3), 307–327.
- Bollerslev, T. (1990). Modeling the coherence in short-run nominal exchange rates: A Multivariate Generalized ARCH Model. *Review of Economics and Statistics* 72, 498–505.
- Bollerslev, T., R. Engle, and J. Wooldridge (1988). A Capital Asset Pricing Model with time-varying covariances. *Journal of Political Economy* 96, 116–131.
- Bornhuetter, R. L. and R. E. Ferguson (1972). The actuary and ibnr. *Proceedings of the Casualty Actuarial Society*, 181–195.
- Braun, A., H. Schmeiser, and F. Schreiber (2018, Jul). Return on risk-adjusted capital under solvency ii: Implications for the asset management of insurance companies. *The Geneva Papers on Risk and Insurance - Issues and Practice* 43(3), 456–472.
- Breiman, L. (1996). Bagging predictors. *Machine Learning* 24(2), 123–140.
- Breiman, L. (2001, Oct). Random forests. *Machine Learning* 45(1), 5–32.
- Broto, C. and E. Ruiz (2004). Estimation methods for stochastic volatility models: A survey. *Journal of Economic Surveys* 18, 613–649.

- Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Nee-lakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. Mc-Candlish, A. Radford, I. Sutskever, and D. Amodei (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), *Advances in Neural Information Processing Systems*, Volume 33, pp. 1877–1901. Curran Associates, Inc.
- Brueckner, R. and B. Schulter (2014). Social signal classification using deep BLSTM recurrent neural networks. pp. 4856–4860.
- Bryson, A. E. (1961). A gradient method for optimizing multi-stage allocation processes.
- Bryson, Jr., A. E. and W. F. Denham (1961). A steepest-ascent method for solving optimum programming problems. (BR-1303).
- Caldeira, A. M., W. Gassenferth, M. A. S. Machado, and D. J. Santos (2015). Auditing vehicles claims using neural networks. *Procedia Computer Science* 55, 62–71.
- Castellani, G., U. Fiore, Z. Marino, L. Passalacqua, F. Perla, S. Scognamiglio, and P. Zanetti (2018). An Investigation of Machine Learning Approaches in the Solvency II Valuation Framework. Available at SSRN: <https://ssrn.com/abstract=3303296>.
- Celikoglu, H. (2007, 10). A dynamic network loading process with explicit delay modelling. *Transportation Research Part C: Emerging Technologies* 15, 279–299.
- Chang, E., C. Han, and F. Park (2017). Deep learning networks for stock markets analysis and prediction: Methodology, data representations and case studies. *Expert System with Applications* 83, 187–205.
- Charpentier, A. and M. Pigeon (2016). Macro vs Micro Methods in Non-Life Claims Reserving: An Econometric Perspective. *Risks* 4, 12.
- Chellapilla, K., S. Puri, and P. Simard (2006). High performance convolutional neural networks for document processing.
- Chen, T. and C. Guestrin (2016). XGBoost: A scalable tree boosting system. pp. 785–794.
- Cho, K., B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014, October). Learning phrase representations using RNN encoder–decoder for statistical machine translation. pp. 1724–1734.
- Chollet, F. et al. (2015). Keras. <https://github.com/fchollet/keras>.
- Christoffersen, P. F., A. Bera, J. Berkowitz, T. Bollerslev, F. Diebold, L. Giorgianni, J. Hahn, J. Lopez, and R. Mariano (1997). Evaluating interval forecasts. *International Economic Review* 39, 841–862.

- Chu, C.-K. and J. S. Marron (1991, 12). Comparison of two bandwidth selectors with dependent errors. *Ann. Statist.* 19(4), 1906–1918.
- Chui, C., X. Li, and H. Mhaskar (1994). Neural networks for localized approximation. *Mathematics of computation*.
- Chui, C., X. Li, and H. Mhaskar (1996). Limitations of the approximation capabilities of neural networks with one hidden layer. *Advances in Computational Mathematics*.
- Cortes, C. and V. Vapnik (1995, Sep). Support-vector networks. *Machine Learning* 20(3), 273–297.
- Cox, J., J. Ingersoll, and S. Ross (1985, 02). A theory of the term structure of interest rates. *Econometrica* 53, 385–407.
- Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems* 2, 303–314.
- Danielsson, J. (2004). Stochastic volatility in asset prices: Estimation by simulated maximum likelihood. *Journal of Econometrics* 64, 375–400.
- de Faria, E., M. Albuquerque, J. González, J. Cavalcante, and M. Albuquerque (2009). Predicting the Brazilian stock market through neural networks and adaptive exponential smoothing methods. *Expert Systems with Applications* 36(10), 12506–12509.
- Devlin, J., M. Chang, K. Lee, and K. Toutanova (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805*.
- Dias, F., R. Nogueira, G. Peixoto, and W. Moreira (2019). Decision-making for financial trading: A fusion approach of machine learning and portfolio selection. *Expert Systems with Applications* 115, 635–655.
- Dickey, D. A. and W. A. Fuller (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association* 74(366a), 427–431.
- Dixon, M., I. Halperin, and P. Bilokon (2020, 05). *Machine Learning in Finance: From Theory to Practice*.
- Dreyfus, S. E. (1973). The computational solution of optimal control problems with time lag. *IEEE Transactions on Automatic Control* 18(4), 383–385.
- Duffie, D. (2019). Prone to fail: The pre-crisis financial system. *Journal of Economic Perspectives*.
- Duma, M., B. Twala, T. Marwala, and F. Nelwamondo (2011). Improving the Performance of the Support Vector Machine in Insurance Risk Classification: A Comparative Study. *Proceedings of the International Conference on Neural Computation Theory and Applications (NCTA-2011)*, 340–346.

- Durbin, J. and S. Koopman (1997). Monte Carlo maximum likelihood estimation for non-Gaussian state space models. *Biometrika* 84, 669–684.
- England, P. D. (2002). Addendum to analytic and bootstrap estimates of prediction errors in claims reserving. *Insurance: Mathematics and Economics* 31.
- England, P. D. and R. J. Verrall (1999). Analytic and bootstrap estimates of prediction errors in claims reserving. *Insurance: Mathematics and Economics* 25(3), 281–293.
- England, P. D. and R. J. Verrall (2002, 01). Stochastic Claims Reserving in General Insurance. *Br. Actuar. J.* 8, 443–544.
- England, P. D. and R. J. Verrall (2006). Predictive Distributions of Outstanding Liabilities in General Insurance. *Annals of Actuarial Science*, 221–270.
- Engle, R. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* 50, 987–1007.
- Engle, R. (2002). Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business and Economic Statistics* 20, 339–350.
- Engle, R., C. Granger, and D. Kraft (1984). Combining competing forecasts of inflation with a bivariate ARCH model. *Journal of Economic Dynamics and Control* 8, 151–165.
- Engle, R. and F. Kroner (1995). Multivariate simultaneous generalized ARCH. *Econometric Theory* 11, 122–150.
- Engle, R. and G. Lee (1999). *A permanent and transitory component model of stock return volatility*, pp. 475–497. Oxford: Oxford University Press.
- Engle, R., V. Ng, and M. Rothschild (1990). Asset pricing with a factor-ARCH covariance structure: Empirical estimates for Treasury Bills. *Journal of Econometrics* 45, 213–238.
- Fan, Y., Y. Qian, F. Xie, and F. K. Soong (2014). TTS synthesis with bidirectional LSTM based recurrent neural networks.
- Fernandez, R., A. Rendel, B. Ramabhadran, and R. Hoory (2014). Prosody contour prediction with Long Short-Term Memory, bi-directional, deep recurrent neural networks.
- Floros, C. (2008, 01). Modelling volatility using garch models: Evidence from egypt and israel. *Middle Eastern Finance and Economics* 2.
- Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29, 1189–1232.
- Frimpong, J. and E. F. Oteng-Abayie (2006, 01). Modelling and forecasting volatility of returns on the ghana stock exchange using garch models. *University Library of Munich, Germany, MPRA Paper*.

- Fukushima, K. (1979). Neural network model for a mechanism of pattern recognition unaffected by shift in position - Neocognitron. *Trans. IECE J62-A(10)*, 658–665.
- Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks 2*.
- Gabrielli, A. (2019, 12). A neural network boosted double Over-Dispersed Ppoisson claims reserving model. *ASTIN Bulletin*, 1–36.
- Gabrielli, A., R. Richman, and M. Wüthrich (2018, 11). Neural Network Embedding of the Over-Dispersed Poisson Reserving Model. Available at SSRN, <https://ssrn.com/abstract=3288454>.
- Gabrielli, A. and M. Wüthrich (2018). An Individual Claims History Simulation Machine. *Risks 6*, 29.
- Geiger, J. T., Z. Zhang, F. Weninger, B. Schuller, and G. Rigoll (2014). Robust speech recognition using long short-term memory recurrent neural networks for hybrid acoustic modelling.
- Gestel, T., J. Suykens, D. Baestens, A. Lambrechts, and G. Lanckriet (2001). Financial time series prediction using least squares Support Vector Machines within the evidence framework. *IEEE Transactions on Neural Networks 12(4)*, 8009–821.
- Geurts, P., D. Ernst, and L. Wehenkel (2006). Extremely Randomized Trees. *Machine Learning 63*, 3–42.
- Glorot, X. and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks.
- Glosten, L., R. Jagannathan, and D. Runkle (1993). On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *The Journal of Finance 48(5)*, 1779–1801.
- Gonzalez-Dominguez, J., I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. J. Moreno (2014). Automatic language identification using Long Short-Term Memory recurrent neural networks.
- Goodfellow, I. J., Y. Bulatov, J. Ibarz, S. Arnaud, and V. Shet (2014). Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082 v4*.
- Goyal, A. and I. Welch (2002, February). Predicting the Equity Premium With Dividend Ratios. (8788).
- Graves, A., A.-R. Mohamed, and G. E. Hinton (2013). Speech recognition with deep recurrent neural networks. pp. 6645–6649.
- Gulay, E. and H. Emeç (2019, 01). The stock returns volatility based on the garch (1,1) model: The superiority of the truncated standard normal distribution in forecasting volatility. *Iranian Economic Review 23*, 87–108.

- Gupta, A. and B. Dhingra (2012). Stock markets prediction using hidden Markov models. *2012 Students Conference on Engineering and Systems*, 1–4.
- Haas, M., S. Mittnik, and M. Paolella (2004a). Mixed normal conditional heteroskedasticity. *Journal of Financial Econometrics* 2, 211–250.
- Haas, M., S. Mittnik, and M. Paolella (2004b). A new approach to Markov-switching GARCH models. *Journal of Financial Econometrics* 2, 493–530.
- Haas, M. and M. Paolella (2012). *Mixture and regime-switching GARCH models*, pp. 71–102. John Wiley and Sons.
- Hagan, P., D. Kumar, A. Lesniewski, and D. Woodward (2002, 01). Managing smile risk. *Wilmott Magazine* 1, 84–108.
- Hajizadeh, E., A. Seifi, F. Zarandi, and I. Turksen (2012). A hybrid modeling approach for forecasting the volatility of S&P 500 index return. *Expert Systems with Applications* 39(1), 531–536.
- Halliwell, L. (2009). Modeling Paid and Incurred Losses Together. *CAS E-Forum, Spring*, 1–40.
- Hamid, S. and Z. Iqbid (2002). Using neural networks for forecasting volatility of S&P 500 index futures prices. *Journal of Business Research* 57(10), 1116–1125.
- Hansen, P. R., A. Lunde, and J. M. Nason (2011). The model confidence set. *Econometrica* 79(2), 453–497.
- Happ, S., M. Merz, and M. Wüthrich (2012). Claims development result in the paid-incurred chain reserving method. *Insurance: Mathematics and Economics* 51, 66–72.
- Happ, S. and M. Wüthrich (2013). Paid-incurred chain reserving method with dependence modeling. *ASTIN Bulletin* 43, 1–20.
- Hastie, T. and R. Tibshirani (1986, 08). Generalized Additive Models. *Statist. Sci.* 1(3), 297–310.
- Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer New York.
- He, K., X. Zhang, S. Ren, and J. Sun (2016, 06). Deep residual learning for image recognition. pp. 770–778.
- Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies* 6, 327–343.
- Hochreiter, S. and J. Schmidhuber (1997a). Long short-term memory. *Neural computation* 9(8), 1735–1780.

- Hochreiter, S. and J. Schmidhuber (1997b, 12). Long short-term memory. *Neural computation* 9, 1735–80.
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4, 251–257.
- Hornik, K. (1993). Some new results on neural network approximation. *Neural networks* 6, 1069–1072.
- Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural Networks* 2(5), 359 – 366.
- Howard, A., M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le (2019). Searching for mobilenetv3. pp. 1314–1324.
- Hull, J. (2015). *Risk management and Financial Institutions, 4th edition*. Wiley and Sons, London.
- Hull, J. C. and A. White (1987). The pricing of options on assets with stochastic volatilities. *Journal of Finance* 42(2), 281–300.
- Hutchinson, J., A. Lo, and T. Poggio (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance* 49, 851–859.
- Igel, C. and M. Hüsken (2003). Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing* 50(C), 105–123.
- Jeremic, Z. and I. Terzić (2014, 04). Empirical estimation and comparison of normal and student t linear var on the belgrade stock exchange. pp. 298–302.
- Jessen, A., T. Mikosch, and G. Samorodnitsky (2011). Prediction of outstanding payments in a Poisson cluster model. *Scandinavian Actuarial Journal* 2011(3), 214–237.
- Johnson, J. and T. Khoshgoftaar (2019, 07). Medicare fraud detection using neural networks. *Journal of Big Data* 6.
- Joseph Lo, A. (2011, August). Extending the Mack Bootstrap: Hypothesis Testing and Resampling Techniques. *The Actuarial Profession*, 29–79.
- Jung, G. and S.-Y. Choi (2021, 03). Forecasting foreign exchange volatility using deep learning autoencoder-lstm techniques. *Complexity* 2021, 1–16.
- Karpathy, A., G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei (2014). Large-scale video classification with convolutional neural networks.
- Kelley, H. (1960). Gradient theory of optimal flight paths. *ARS Journal* 30, 947–954.
- Kim, H. and C. Won (2018). Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with applications* 103, 25–37.

- Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *CoRR abs/1412.6980*.
- Kraft, D. and R. Engle (1982). Autoregressive conditional heteroskedasticity in multiple time series. Department of Economics, UCSD.
- Kremer, E. (1982, 01). IBNR claims and the two-way model of ANOVA. *Scandinavian Actuarial Journal* 1982.
- Kristjanpoller, W., A. Fadic, and M. Minutolo (2014). Volatility forecast using hybrid neural network models. *Expert Systems with Applications* 41(5), 2437–2442.
- Kristjanpoller, W. and E. Hernández (2017). Volatility of main metals forecasted by a hybrid ANN-GARCH model with regressors. *Expert Systems with Applications* 84, 290–300.
- Kristjanpoller, W. and M. Minutolo (2015). Gold price volatility: A Forecasting approach using the Artificial Neural Network-GARCH model. *Expert Systems with Applications* 42(20), 7245–7251.
- Kristjanpoller, W. and M. Minutolo (2016). Forecasting volatility of oil price using an Artificial Neural Network-GARCH model. *Expert Systems with Applications* 65(15), 233–241.
- Kristjanpoller, W. and M. Minutolo (2018). A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. *Expert Systems with Applications* 109, 1–11.
- Krollner, B., B. Vanstone, and G. Finnie (2010). Financial time series forecasting with machine learning techniques: A survey. *European Symposium on Artificial Neural Networks: Computational and Machine Learning*.
- Kuo, K. (2018). DeepTriangle: A Deep Learning Approach to Loss Reserving. *CoRR abs/1804.09253*.
- Kupiec, P. H. (1995, January). Techniques for verifying the accuracy of risk measurement models. *The Journal of Derivatives* 3(2), 73–84.
- LeCun, Y., Y. Bengio, and G. Hinton (2015a). Deep Learning. *Nature* 521(7553), 436–444.
- LeCun, Y., Y. Bengio, and G. Hinton (2015b, 05). Deep learning. *Nature* 521, 436–44.
- LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1990). Handwritten digit recognition with a back-propagation network. pp. 396–404.
- Leong, W. K., S. Wang, and H. Chen (2014). Back-testing the odp bootstrap of the paid chain-ladder model with actual historical claims data. *Variance*.

- Leshno, M., V. Lin, A. Pinkus, and S. Schocken (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks* 6, 861–867.
- Li, R., W. Zhang, H.-I. Suk, L. Wang, J. Li, D. Shen, and S. Ji (2014). Deep learning based imaging data completion for improved brain disease diagnosis.
- Linetsky, V. and R. Mendoza (2009). The constant elasticity of variance model.
- Linnainmaa, S. (1970). The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master’s thesis, Univ. Helsinki.
- Lopez, O., X. Milhaud, and P. Thérond (2019). A Tree-Based algorithm adapted to microlevel reserving and long development claims. *ASTIN Bulletin*.
- Lowenstein, R. (2000). *When genius failed: the Rise and Fall of Long-Term Credit Management*. Random House.
- Lu, X., D. Que, and G. Cao (2016). Volatility forecast based on the hybrid artificial neural network and garch-type models. *Procedia Computer Science* 91, 1044 – 1049.
- Mack, T. (1991). A Simple Parametric Model for Rating Automobile Insurance or Estimating IBNR Claims Reserves. *ASTIN Bulletin* 21(1), 93–109.
- Mack, T. (1993). Distribution-free Calculation of the Standard Error of Chain Ladder Reserve Estimates. *ASTIN Bulletin: The Journal of the International Actuarial Association* 23(02), 213–225.
- Mandelbrot, B. (1963). The variation of certain speculative prices. *Journal of Business* 36, 394–419.
- Mapa, D. (2003, 12). A range-based garch model for forecasting volatility. *Philippine Review of Economics* 40, 73–90.
- Maqsood, A., S. Safdar, R. Shafi, and N. Lelit (2017, 01). Modeling stock market volatility using garch models: A case study of nairobi securities exchange (nse). *Open Journal of Statistics* 07, 369–381.
- Marchi, E., G. Ferroni, F. Eyben, L. Gabrielli, S. Squartini, and B. Schuller (2014). Multi-resolution linear prediction based features for audio onset detection with bidirectional LSTM neural networks. pp. 2183–2187.
- Margraf, C., V. Elpidorou, and R. Verrall (2018). Claims reserving in the presence of excess-of-loss reinsurance using micro models based on aggregate data. *Insurance: Mathematics and Economics* 80, 54–65.
- Martínez-Miranda, M., B. Nielsen, and R. Verrall (2012). Double Chain Ladder. *ASTIN Bulletin* 42(1), 59–76.
- Martínez-Miranda, M., B. Nielsen, and R. Verrall (2013a). Continuous Chain Ladder: Reformulating and generalizing a classical insurance problem. *Expert Systems with Applications* 40, 5588–5603.

- Martínez-Miranda, M., B. Nielsen, and R. Verrall (2013b). Double Chain Ladder and Bornhuetter-Ferguson. *North American Actuarial Journal* 17, 101–113.
- Martínez-Miranda, M., B. Nielsen, R. Verrall, and Wüthrich (2015). The Link Between Classical Reserving and Granular Reserving Through Double Chain Ladder and its Extensions. *Scandinavian Actuarial Journal* 2015, 383–405.
- McCullagh, P. and J. Nelder (1989). *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall.
- McCulloch, W. and W. Pitts (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 127–147.
- McNeil, A. J., R. Frey, and P. Embrechts (2015). *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton, NJ, USA: Princeton University Press.
- Melino, A. and S. Turnbull (1990). Pricing foreign currency options with stochastic volatility. *Journal of Econometrics* 45, 239–265.
- Merz, M. and M. V. Wüthrich (2010). Paid-incurred chain claims reserving method. *Insurance: Mathematics and Economics* 46, 568–579.
- Meyers, G. (2015). *Stochastic Loss Reserving Using Bayesian MCMC Models*. CAS Monograph Series, number 1. Casualty Actuarial Society.
- Molodtsova, T. and D. Papell (2012, 06). Taylor rule exchange rate forecasting during the financial crisis. *NBER International Seminar on Macroeconomics* 9, 55–97.
- Monfared, S. A. and D. Enke (2014). Volatility forecasting using a hybrid gjr-garch neural network model. *Procedia Computer Science* 36, 246 – 253.
- Nakama, T. (2011). Comparisons of Single and Multiple Hidden Layer Neural Networks. pp. 270–279.
- Nelder, J. A. and R. W. M. Wedderburn (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)* 135(3), 370–384.
- Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: A new approach. *Econometrica* 59(2), 347–70.
- Nigri, A., S. Levantesi, S. Marino, M. Scognamiglio, and F. Perla (2019). A deep learning integrated lee-carter model. *Risk* 7, 33.
- Oh, K.-S. and K. Jung (2004). GPU implementation of neural networks. *Pattern Recognition* 37(6), 1311–1314.
- Omari, C., S. Nyambura, and J. Wairimu (2018). Modeling the Frequency and Severity of Auto Insurance Claims Using Statistical Distributions. *Journal of Mathematical Finance* 8, 137–160.

- Patel, M. and S. Yalamalle (2014). Stock price prediction using artificial neural network. *International Journal of Innovative Research in Science, Engineering and Technology* 3(June 2014), 13755 – 13762.
- Patton, A., D. N. Politis, and H. White (2009). Correction to “automatic block-length selection for the dependent bootstrap” by d. politis and h. white. *Econometric Reviews* 28(4), 372–375.
- Peng, Y., P. Melo, J. Camboim de Sá, A. Akaishi, and M. Montenegro (2018). The best of two worlds: Forecasting high frequency volatility for cryptocurrencies and traditional currencies with support vector regression. *Expert Systems with Applications* 97, 177–192.
- Pigeon, M., K. Antonio, and M. Denuit (2013). Individual loss reserving with the Multivariate Skew Normal framework. *ASTIN Bulletin* 43, 399–428.
- Pigeon, M., K. Antonio, and M. Denuit (2014). Individual loss reserving using paid-incurred data. *Insurance: Mathematics and Economics* 58, 121–131.
- Poggio, T. and F. Girosi (1990). Networks for approximation and learning. *Proceedings of the IEEE* 78, 1481–1497.
- Politis, D. and J. Romano (1991). *A Circular Block-resampling Procedure for Stationary Data*. Purdue University. Department of Statistics.
- Politis, D. N. and J. P. Romano (1994). The stationary bootstrap. *Journal of the American Statistical Association* 89(428), 1303–1313.
- Politis, D. N. and H. White (2004). Automatic block-length selection for the dependent bootstrap. *Econometric Reviews* 23(1), 53–70.
- Poon, S. and C. Granger (2003). Forecasting volatility in financial markets. a review. *Journal of Economic literature* 41(2), 478–539.
- Posthuma, B., E. Cator, W. Veerkamp, and E. Van Zwet (2008). Combined Analysis of Paid and Incurred Losses. *CAS E-Forum Fall*, 272–293.
- Quarg, G. and T. Mack (2004). Munich Chain Ladder. *Blätter der Deutschen Gesellschaft für Versicherungs und Finanzmathematik XXVI*, 597–630.
- R Core Team (2017). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.
- Rajashree, P. and B. Ranjeeeta (2015). A differential harmony search based hybrid internal type2 fuzzy EGARCH model for stock market volatility prediction. *International Journal of Approximate Reasoning* 59, 81–104.
- Ramos-Pérez, E., P. Alonso-González, and J. Núñez-Velázquez (2019). Forecasting volatility with a stacked model based on a hybridized Artificial Neural Network. *Expert Systems with Applications* 129, 1–9.

- Ramos-Pérez, E., P. J. Alonso-González, and J. J. Núñez Velázquez (2021a). Multi-transformer: A new neural network-based architecture for forecasting s&p volatility. *Mathematics* 9(15).
- Ramos-Pérez, E., P. J. Alonso-González, and J. J. Núñez Velázquez (2021b). Stochastic reserving with a stacked model based on a hybridized artificial neural network. *Expert Systems with Applications* 163.
- Rastogi, S., J. Don, and N. N (2018, 02). Volatility estimation using garch family of models: Comparison with option pricing. *Pacific Business Review International* 10, 54–60.
- Rehman, Z. and S. Klugman (2009). Quantifying Uncertainty in Reserve Estimates. *Variance Journal* 4, 30–46.
- Renshaw, A. E. and R. J. Verrall (1998). A Stochastic Model Underlying the Chain-Ladder Technique. *British Actuarial Journal*.
- Richman, R. and M. Wüthrich (2018). A Neural Network Extension of the Lee-Carter Model to Multiple Populations. *SSRN*. Available at SSRN, <https://ssrn.com/abstract=3270877>.
- Riedmiller, M. and H. Braun (1993). A direct adaptive method for faster backpropagation learning: The Rprop algorithm. pp. 586–591.
- Roh, T. (2006). Forecasting the volatility of stock price index. *Expert Systems with Applications* 33(4), 916–922.
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* 65 6, 386–408.
- Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986a). Learning internal representations by error propagation. 1, 318–362.
- Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986b). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. pp. 318–362.
- Rumelhart, D. E. and D. Zipser (1986). Feature discovery by competitive learning. pp. 151–193.
- Russell, S. and P. Norvig (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). USA: Prentice Hall Press.
- Ryan, J. A. and J. M. Ulrich (2017). *quantmod: Quantitative Financial Modelling Framework*. R package version 0.4-12.
- Sak, H., O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao (2014). Sequence discriminative distributed training of Long Short-Term Memory recurrent neural networks.
- Saxe, A. M., J. L. McClelland, and S. Ganguli (2013, December). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks.

- Sermanet, P., D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun (2013). OverFeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.
- Sheela, D. and S. Deepa (2013). Review on Methods to Fix Number of Hidden Neurons in Neural Networks. *Mathematical Problems in Engineering*, 1–11.
- Silver, D., A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis (2016, 01). Mastering the game of go with deep neural networks and tree search. *Nature* 529, 484–489.
- Silver, D., T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR abs/1712.01815*.
- Singh, K. (2010). Fixing global finance: A developing country perspective on global financial reforms.
- Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014, 06). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1929–1958.
- Surkan, A. and Y. Xingren (2001). Bond rating formulas derived through simplifying a trained neural network. *Proceedings of the IEEE International conference on neural network* 2, 1028–1031.
- Sutskever, I., O. Vinyals, and Q. V. Le (2014). Sequence to sequence learning with neural networks. (arXiv:1409.3215 [cs.CL]). NIPS’2014.
- Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An Introduction* (Second ed.). The MIT Press.
- Swanson, N. R. (1998). Money and output viewed through a rolling window. *Journal of Monetary Economics* 41(3), 455 – 474.
- Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2014). Going deeper with convolutions. (arXiv:1409.4842 [cs.CV]).
- Szegedy, C., A. Toshev, and D. Erhan (2013). Deep neural networks for object detection. pp. 2553–2561.
- Taylor, G., G. McGuire, and J. Sullivan (2008). Individual Claim Loss Reserving Conditioned by Case Estimates. *Annals of Actuarial Science* 3(1-2), 215–256.
- Taylor, S. (1982). *Financial returns modelled by the product of two stochastic processes, A study of daily sugar prices 1961–79*, Volume 1, pp. 223–226. North-Holland.

- Taylor, S. J. (1986). *Modelling Financial Time Series*. Wiley.
- Temin, P. (2010). The great recession and the great depression. *JSTOR*.
- Tse, Y. and K. Tsui (2002). A multivariate GARCH model with time-varying correlations. *Journal of Business and Economic Statistics* 20, 351–362.
- Ugurlu, E., E. Thalassinou, Y. Muratoglu, E. Thalassinou, and Y. Muratoglu (2014, 01). Modeling volatility in the stock markets using garch models: European emerging economies and turkey.
- Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). Attention is all you need. *CoRR abs/1706.03762*.
- Venter, G. (2008). Distribution and Value of Reserves Using Paid and Incurred Triangles. *CAS E-Forum, Fall*, 348–375.
- Verma, S. (2021, 03). Forecasting volatility of crude oil futures using a garch-rnn hybrid approach. *Intelligent Systems in Accounting, Finance and Management*.
- Verrall, R. J. (2000). An investigation into stochastic claims reserving models and the chain-ladder technique. *Insurance: Mathematics and Economics* 26(1), 91–99.
- Vidal, A. and W. Kristjanpoller (2020). Gold volatility prediction using a cnn-lstm approach. *Expert Systems with Applications* 157.
- Vinod, H. (2006). Maximum entropy ensembles for time series inference in economics. *Journal of Asian Economics* 17(6), 955–978.
- Vinod, H. D. and J. L. de Lacalle (2009). Maximum entropy bootstrap for time series: The meboot R package. *Journal of Statistical Software* 29(5), 1–19.
- Weke, P. and C. Ratemo (2013). Estimating IBNR Claims Reserves for General Insurance Using Archimedean Copulas. *Applied Mathematical Sciences* 7, 1223–1237.
- Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph. D. thesis, Harvard University.
- Widrow, B. and M. E. Hoff (1962). Associative storage and retrieval of digital information in networks of adaptive neurons. pp. 160–160.
- Williams, M. (2010). Uncontrolled risk. *McGraw-Hill Education*.
- Wu, J., B. Zhou, D. Peck, S. Hsieh, V. Dialani, L. Mackey, and G. Patterson (2018, 05). Deepminer: Discovering interpretable representations for mammogram classification and explanation.
- Wüthrich, M. (2018a). Machine learning in individual claims reserving. *Scandinavian Actuarial Journal* 2018, 465–480.
- Wüthrich, M. (2018b). Neural networks applied to chain-ladder reserving. *European Actuarial Journal* 8, 407–436.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *ArXiv abs/1212.5701*.

Zhang, L., K. Zhu, and S. Ling (2018). The ZD-GARCH model: A new way to study heteroscedasticity. *Journal of Econometrics* 202(1), 1–17.

Zivot, E. and J. Wang (2006). *Modeling Financial Time Series with S-PLUS®*. Berlin, Heidelberg: Springer-Verlag.