Send Frequency Prediction on Email Marketing

Universidade Fernando Pessoa

Carolina Tomé de Araújo

Advisor: Professor Doutor Christophe Soares
Supervisor: Professor Doutor Ivo Pereira

Faculdade de Ciência e Tecnologia

Universidade Fernando Pessoa

A thesis submitted for the degree of

Master of Science

2021

Resumo

O E-mail Marketing é uma forma de marketing direta que utiliza o e-mail como um meio de comunicação comercial pelo que numa perspetiva mais ampla, qualquer e-mail enviado a um potencial subscritor e atuais subscritores também pode ser considerado e-mail marketing.

Assim sendo, o subscritor vai receber várias comunicações ao longo do dia, reduzindo a visibilidade dos e-mails mais antigos com a entrada de novas comunicações e consequentemente, reduzindo as taxas de aberturas.

Tendo em conta que existem subscritores que preferem abrir e ler as suas comunicações de manhã, outros de tarde e alguns durante a noite, é necessário enviar uma comunicação que proporcione uma maior visibilidade que perpetue maiores taxas de abertura e uma maior captação de interesse do subscritor com a entidade que enviou uma comunicação.

Esta tese apresenta uma solução para enviar comunicações de marketing na altura certa aos subscritores ou potenciais subscritores. A sua contribuição consiste num modelo segmentado que utiliza um algoritmo tradicional de clustering baseado na informação trocada entre as empresas e os seus subscritores. O modelo implementa posteriormente uma abordagem de ensemble paralelo utilizando técnicas como simple averaging e stacking com algoritmos de regressão treinados (RF, Linear Regression, KNN e SVR) e com um algoritmo de deep learning (RNNs) para determinar a melhor altura para enviar comunicações de e-mail. A implementação é executada utilizando um dataset fornecido pela empresa E-goi para treinar e testar a abordagem mencionada.

Os resultados obtidos nesta tese indicam que o algoritmo KNN é mais adequado para prever o melhor momento para enviar comunicações de e-mail dos algoritmos ML treinados. Das duas técnicas utilizadas para a abordagem do ensemble paralelo, o stacking é o mais adequado para prever o melhor momento para o envio das comunicações de e-mail.

Abstract

Email Marketing is a form of direct marketing that uses email as a means of commercial communication. In a broader perspective, any email sent to a potential subscriber and current subscribers can also be considered email marketing.

Therefore, the subscriber will receive several communications throughout the day, reducing the visibility of older emails with the entry of new communications and consequently reducing open rates.

Considering that there are subscribers who prefer to open and read their communications in the morning, others in the afternoon, and some at night, it is necessary to send a communication that provides the visibility that leads to higher open rates and capture the subscribers' interest with the entity that sent the communication.

This thesis presents a solution to send marketing communications at the right time to subscribers or potential subscribers. Its contribution consists of a segmented model that uses a traditional clustering algorithm based on the information exchanged between companies and subscribers. The model then implements a parallel ensemble approach using simple averaging and stacking techniques with trained regression algorithms (RF, Linear Regression, KNN, and SVR) and a deep learning algorithm (RNNs) to determine the best time to send email communications. The implementation is executed using a dataset provided by the company E-goi to train and test the mentioned approach.

The results obtained in this thesis indicate that the KNN algorithm is better suited to predict the best time to send email communications of all the trained ML algorithms. Stacking is the most suitable for predicting the best time to send email communications of the two techniques used for the parallel ensemble approach.

To my family and friends.

Acknowledgements

The elaboration of this thesis would not have been possible without the collaboration, encouragement, and goodwill of several people, whom I would like to thank:

To the Faculty of Science and Technology of Fernando Pessoa University.

To the supervisors' Professor Dr. Christophe Soares and Professor Dr. Ivo Pereira, for the constant sharing of knowledge, guidance, availability, and opportunity.

To my family, especially my parents, brother, godmother, and grandparents, for their permanent and unconditional support, encouragement, patience, and giving me the bases to overcome life's challenges.

To my friends, for always encouraging me to be a better person and always believing in me.

Finally, to E-goi for creating a friendly, educational, and adequate environment as well for believing in me and alllowing me to develop this project.

This Master Thesis was created as a result of the project *Criação de um Núcleo de I&D para a geração de novo conhecimento nas áreas de Inteligência Artificial, Machine Learning, Intelligent Marketing e One-2-One Marketing*, supported by Operational Programme for Competitiveness and Internationalisation (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

Contents

List of Figures

List of Tables

Glossary & Acronymous

Agent *Anything that views its environment and acts upon it. For example, a human agent has eyes, ears, and other organs for sensors; legs, and hands for actuators (Russell and Norving, 2020)*

AI *Artificial Intelligence is the study of agents that perceive the surrounding environment and perform actions that affect that environment (Russell and Norving, 2020)*

ANNs *Artificial Neural Network consist of a machine learning model of a network of simple information-processing units called neurons. It is possible to create different neural networks by modifying the connections between the neurons in the network. Examples of popular neural networks are Feed-Forward, Convolutional Neural Networks, and Recurrent Neural Networks (Kelleher, 2019)*

Backpropagation *is an algorithm used to train neural networks with hidden layers of neurons. During training, the weights in a network are iteratively updated to reduce the network's error. In order to update those weights, it is necessary to calculate an estimate of the contribution of the output of that neuron to the overall error of the network. This algorithm is a solution to calculate those estimates for each neuron. Once the error estimates are calculated, the weights are updated using an optimization algorithm like gradient descent. It works in two phases: a forward pass and a backward pass. In the former, the network receives the example. It calculates the overall error of the network, at the output layer, by comparing the output with each layer receiving a portion of the blame for the error in proportion to the sensitivity of the error to changes happening in the output of that neuron. The process of sharing the errors back and forward through the network is why the algorithm has its name (Kelleher, 2019).*

Bias *Gap between the value predicted by the model and the actual value of the data. Suppose an increase of the bias values happens, the predictions will likely be skewed in a particular direction away from the true values (Theobald, 2018)*

Clustering *Algorithm that organizes data into different groups with similar characteristics (Henderson, 2019)*

Cox Proportional Hazard Model *Regression model standard in medical research to investigate associations between patients' lifetime and one or more predictive variables. This model extends the methods of survival analysis to determine the effects of various risk factors on survival time (Singh et al., 2020)*

Clickthrough rates *Percentage of subscribers who clicked on one or more links in an email sent (Monitor, 2020)*

Customer *Individual who has purchased one product and has less commitment to the brand or company that is selling something (Richardson, 2020)*

LTV *Lifetime value estimates the average revenue that a customer will generate through their lifespan as a customer. This 'worth' of a customer can help determine many economic decisions for a company, including marketing budget, resources, profitability, and forecasting. It is a key metric in subscription-based business models (Optideia, 2020)*

Dataset *is a collection of instances (set of features). A dataset is an n * m matrix in the most basic form, where n is the number of instances (rows), and m is the number of features (columns) (Kelleher, 2019)*

Deep Learning *Subfield of Machine Learning that designs and evaluates training algorithms and architectures for modern network models. A deep neural network consists of a network that has multiple (usually more than 2) layers of hidden units (called neurons) (Kelleher, 2019)*

Email Tracking *is the process of tracking emails and using the data to make business decisions. Most email tracking tools capture data such as open rates, times, locations, clicks, and attachments (Hubspot, 2020)*

FNNs *Feed-Forward Networks are neural networks where all the connections in the network point forward to the neurons in the subsequent layers. In other words, there are no links backward from the output of a neuron to the input of a neuron in a previous layer (Kelleher, 2019)*

Function *Deterministic mapping from a set of input values to one or more output values. In ML, the term function is often interchangeable with the term model (Kelleher, 2019)*

KPIs *Key performance indicators are performance measures of the success of an email campaign. In the context of email marketing, they include open rates, clickthrough rates, and subscription fees (Monitor, 2020)*

Leads *Organization or a person interested in what is being sold to them. The interest is expressed by sharing information, such as email or phone (INTERNET, 2020)*

ML *Machine Learning consists of a branch of AI that allows the systems to learn and improve automatically through the experience without being explicitly programmed (Henderson, 2019)*

Model *In ML, a model is a computer program that encodes the patterns that the ML algorithms extract from a dataset. A model is created by running a ML algorithm on a dataset (Kelleher, 2019)*

Open rates *Percentage of subscribers that open a sent email (Monitor, 2020)*

Overfitting *Occurs when there is high variance and low bias. An overfitted model generates accurate predictions using training data however is less precise in making predictions using the test data. It can happen when the training and test are not randomly split, and the data is not attributed evenly (Theobald, 2018)*

RNNs *Recurrent Neural Networks consist of an artificial network that uses sequential data or time-series data for temporal problems. These networks use training data to learn and are distinguished by their "memory" since they receive information from previous inputs (outputs) to influence the current input and output (Kelleher, 2019)*

ROI *Performance measure used to evaluate the efficiency or profitability of an investment. Return on investment tries to directly measure the amount of return on a particular investment relative to the investment's cost (Fernando, 2020)*

RFM *Metrics that measure consumer response behaviors in three dimensions. The first dimension is recency, which indicates the freshness of the customer activity (e.g., how long it has been since the customer last responded). The second dimension is the frequency which measures the regularity of the customer transactions or visits (e.g., how often the customers have responded to the receiving mailings). The last dimension is monetary, which measures the amount of money, or the number of products that the customer has spent in response to the mailings (Jonker et al., 2004)*

STO *Send Time Optimization consists of an AI program or software for email campaigns that analyze the email responses and, based on that, sends the following email at the ideal time (Smartt, 2020)*

SaaS *Software as a Service is a delivery and licensing concept in which software is accessed online via subscriptions (Lexico, 2020)*

Subscriber *Person who signs up for recurring, automatic billing scenarios, which means repeat purchases and has more commitment to the brand or company selling something (Richardson, 2020)*

Subscription Rate *Price charged for a subscription (Collins, 2020)*

Survival Analysis *Statistical modeling where the primary variable of interest is the time of an event (e.g., opening an email). Historically, this event is determined by death. A predominant feature is a censorship; not all individuals experience an event in the observation window. This censorship happens because the events have not occurred during the observation window, and the corresponding individual can no longer be monitored (Singh et al., 2020)*

Underfitting *happens when there is low variance and high bias. An underfitting model is overly simple, leading to inaccurate predictions for training and test data (Theobald, 2018)*

Variance *describes how scattered the predicted values are to each other. Bias and variance contribute to the error, but it is the prediction error that we want to minimize specifically (Theobald, 2018)*

Chapter 1

Introduction

Nowadays, any company that wants to sell a product or a service sends a marketing communication to a subscriber through communication channels, such as email. Digital marketing platforms create automated solutions to enable this type of interaction.

However, it leads to an overflow of communications between both parties by sending too many emails or/and at the wrong time, which leads to subscribers not opening nor reading them. This thesis aims to prevent these behaviors by predicting the best period of time to communicate to a subscriber using machine and deep learning algorithms.

1.1 Context

In recent years, with the significant technological advances in the digital world, the demand for better interactions between subscribers and companies is growing, creating a need for more personalized and individualized experiences. Companies are at risk of *"falling behind"* or losing an advantage if they do not keep up with the trends imposed by their subscribers (Deshmukh, 2019). To guarantee this personalization, companies need to understand and know their subscribers.

One of the most important marketing channels is Email Marketing (Abakouy et al., 2019), which aims to send the best offer or communication to the right person at the right time based on the subscribers' profile. It is still challenging to send the best offer and predict the right time to send a communication to a subscriber (Bartley, 2021). In a broader perspective, any email message a company sends consists of Email Marketing (Fariborzi and Zahedifard, 2012).

Another study by Campaign Monitor (Monitor, 2020) in 2014 states that email exceeds social networks in attracting subscribers. VentureBeat (Forsey, 2020) states that email was the communication channel with the highest ROI surpassing social networks.

A study by Direct Marketing Association (van Rijn, 2015) in 2015, states that more than 90% of businesses use Email Marketing as direct and efficient communication,

strengthening the Return on Investment (ROI) rates. The same study states that one in five companies reports an ROI of more than 70 dollars for each email sent. The previous study and another one conducted by Salesforce (Salesforce, 2017) in 2017 state that one of the fastest-growing channels, in recent years, was email. It refers that the growth rate was 83%, and an email generated 38 dollars in ROI for every dollar invested.

Research led by HubSpot (Bump, 2020), between November and December of 2019, found that more than 78% of marketers saw increased email engagement in the last twelve months. It also found that more than 73% of millennials prefer business communications to be sent by email. According to a study conducted by Khoros (Netzer, 2020), it is the most expressive age group, estimating that their global expenditure should exceed 4 million dollars. Another study conducted by Hubspot (Bump, 2020), in 2020, states that 80% of marketers affirmed that the interactions with the subscribers by email has improved compared to the previous years, and on average, companies manage to earn 42 dollars for every dollar invested in the advertising sent in the email.

With the exponential growth of email over the years, the subscribers receive large flows of campaigns, which is a problem for both companies and subscribers. For companies, it leads to loss of visibility, low opening rates, and low sales rates. For subscribers, it results in the accumulation of emails leading to the elimination of emails, the non-opening, and the classification of these emails as spam (Monitor, 2020).

A study conducted by HubSpot (Bump, 2020) in 2020, concludes that subscribers only open emails if they contain any value for themselves, determining that relevance using the subject is the key to attract subscribers (Monitor, 2020). Considering this acknowledgment with the best time to send individualized and personalized communications, companies have better opportunities to conquer the subscriber.

Over the years, intelligent systems have been used in several domains, including in marketing. According to Salesforce's 2017 study (Salesforce, 2017), the two areas where AI would have the most significant impact in the next five years would be delivering the right message through the right channel at the right time, with 61% substantial impact on businesses and 59% percent impact on segmentation subscribers

In this context, ML algorithms allow subscribers to be segmented/organized by their behavior/profiles, considering open rates, clickthrough rates, frequency, or period of interactions with the companies. Based on this information, companies should be able to send their communications on time.

This thesis proposes machine and deep learning algorithms to predict the best period of time to send communications by email, using a regression approach trained with historical labeled data.

1.2 Organization

The company E-goi (Figure 1.1), created by Miguel Gonçalves, is based in Matosinhos, Portugal, and develops solutions and products to improve the relationship between companies and their clients.

The main product consists of a SaaS for multichannel marketing automation, using Email, Smart SMS, Web Push, Smart Wi-fi, Mobile Push, Ads, and transactional SMS.

The objective of the Portuguese company involved in this project is to understand the best period of time to send marketing communication to its subscribers. The company intends to offer a service that automatically determines the best time interval – the day of the week and the time of the day to send a marketing message to the individual subscriber based on their profile.

Figure 1.1: Egoi's logo

1.3 Objectives

Send Frequency Optimization significantly impacts sending marketing emails and how subscribers react to the campaigns received (iPULLRANK, 2020). Machine Learning models, and consequently, deep learning models, can answer to the following questions:

- How often should we pay attention to the individual subscriber and send a new marketing message?
- How often should we track leads?
- How often should we contact the subscriber initially?
- What is the most effective time to send a new marketing message?

This thesis aims to segment subscribers by their profile and, based on it, predict and determine the best period of time to send marketing communications using predictive regression modeling and a deep learning approach.

1.4 Document Structure

This document consists of six chapters to present the developed solution. The first chapter consists of an Introduction with a brief contextualization, description of the problem, and the objectives. The second chapter provides general concepts important to the problem and the solution, such as Artificial Intelligence and Deep Learning, Artificial Intelligence Marketing (AIM), followed by a presentation of the conducted research in the fields of the proposed problem, including techniques used to perform Feature Engineering, one of the most important phases of any ML project. The following chapter introduces important information about the proposed model denominated as Send Frequency Predictive Model (SFM), such as the description of the client from where the data was extracted, an analysis of the dataset, and the process of Feature Engineering on the former. The fourth chapter describes the approaches implemented to find the optimal solution to predict the best period of time to send marketing communication. The fifth chapter evaluates and analyzes the experiments in the previous chapter. The last chapter presents the future work and the conclusions.

Chapter 2

Predictive Models To Send Automated Communications

This chapter begins with the literature review on the following topics: ML, and Deep Learning. Firstly, it is explained what Digital Marketing is and how it expanded into the AI field. Then the explanation of AI devolves into ML, the different learning types associated, and its techniques.

Artificial Intelligence Marketing (AIM) explains deeper the bridge between Digital Marketing and AI, mostly with ML techniques. It also presents examples of companies who already implemented ML concepts in their sales processes and other parts of their businesses. Deep Learning, mostly the concept of neural networks and recurrent neural networks, is presented. It ends with an exposition of the scientific work and commercial implementations of the best time to send marketing communications and the segmentation of subscribers.

2.1 Background

A basic and traditional definition of Marketing refers to an interaction between a business/company and its consumers/subscribers. The goal is the creation of beneficial value for both.

According to the Oxford Dictionary, Digital Marketing consists of digital technologies to promote products and services (Dictionary, 2020). Digital Marketing revolutionized new market perceptions by increasing sales and promotions across the digital world. It provided new mechanisms for expanding subscribers' power of choice and influence. Additionally, it provided new opportunities for companies/brands to interact dynamically with their subscribers/consumers amplifying their experience. In this thesis, it is highlighted the use of Email Marketing as one of the main interactions between the companies and their subscribers. This communication channel allows a direct relationship between

a company and its subscribers and allows to improve an existing relationship.

With the advances in the digital world and the constant changes in the subscriber markets, valuable data related to the subscriber and their interactions accumulate daily. This amount of information provides companies with mechanisms and tools to capture new subscribers and improve relationships with the existing subscribers. Furthermore, these mechanisms help in the individualization and personalization of services and products towards the subscribers. In this context, ML as a subfield of AI appears as a new field to help automatize the processes mentioned above and assists to the automation of Digital Marketing.

2.2 Technologies

2.2.1 Artificial Intelligence

One of the AI concept pioneers was Alan Turing, in 1950, with the Turing Test, when he proposed a practical definition of theoretical concepts of intelligence (Russell and Norving, 2020). The object of the called *"Imitation Game"* consisted of an interrogator that distinguished whether an answer came from a person or a machine based on the contents of the answers. Physical contact was deliberately avoided (Russell and Norving, 2020; Gugery, 2006). He debated that a machine could be considered *"intelligent"* if it knew how to answer questions in such a way that it would not be possible to distinguish whether the answer came from a machine or a human being (Russell and Norving, 2020).

The term Artificial Intelligence was introduced in 1956 during an academic conference, however, the idea that a machine could *"think"* by itself was already circulating (Gugery, 2006). In the same year, researchers at Carnegie Technological Institute (now Carnegie Mellon University) developed the first AI program called 'Logic Theoris' (Gugery, 2006).

The term, though, was patented by John McCarthy in 1970, considered the father of AI. John McCarthy named AI as *"the science and engineering of making intelligent machines"* (McCarthy, 2020). For example, a machine that recognizes objects and can classify images, as humans do, but with better precision, could be considered AI (Anyoha, 2017).

In a more current definition, we can encompass any situation where machines perform tasks *"intelligently"*. Using Tesler's theorem, in honor of the computer scientist, *"Intelligence is what machines have not yet done"* (Hofstadter, 1979; Automation, 2020). In other words, computational objectives or tasks that the machine have yet to perform or develop.

Through the years, it was understood that tackling *"intelligence"* was a challenge due to its inherent complexity and technical limitations. Therefore, rather than solving the

concept as a whole, the focus expanded to the subproblems like language, facial/visual recognition, and many more subproblems. Consequently, the subproblems were divided into Narrow AI and General Artificial Intelligence (Valigi and Mauro, 2020). Recently, it was added another category called Artificial Super Intelligence (Bruner, 2020).

Artificial Narrow AI, also known as weak AI, is an AI program capable of solving a specific and well-defined task at a time. It ranges from identifying and recognizing objects from pictures to predict which customers who bought a specific product *X* are more likely to buy the product *Y*. Google Assistant, Google Translate, or Siri are examples of this type of AI. It is called *"weak"* because machines do not possess any *"human intelligence"*. Using the examples below, Siri or Google Assistant can only process the human language into a search engine and return the results. Any query that consists of abstract questions (e.g., the meaning of life) does not obtain an answer. However, if it is asked about the weather's condition today, the results are accurate.

Artificial General Intelligence, also known as strong AI, is an artificial intelligence program capable of tackling all sorts of challenges. It is called *"strong"* because of the intelligent ability machines have. However, despite the promising advances in this type of AI, machines cannot yet think abstractly and strategize as humans do, and to achieve human intelligence, machines must experience consciousness.

Artificial Super Intelligence was recently introduced to the mixture of AI categories. The Oxford philosopher Nick Bostrom defines superintelligence as *"any intellect that greatly exceeds the cognitive performance of humans in virtually all domains of interest"*. A lot more futuristic than General Artificial Intelligence, this type of AI describes machines capable of exhibiting human intelligence at its full spectrum.

Futurist Ray Kurzweil (Futurism, 2020) describes it as coexistence between AI machines and humanity where machines can reinforce human abilities. However, it is not possible to predict when developments in this type of AI will appear.

2.2.2 Machine Learning

Machine Learning, a field of AI, is currently revolutionizing the computing world with its digital interactions. The learning process begins with data analysis and identifies patterns with that data. Then, the models learn to adjust accordingly to the environment/data they find themselves in (Henderson, 2019).

The term Machine Learning was patented by Arthur Samuel in 1959, stating that ML is the field of study that gives computers the ability to learn without being explicitly programmed. However, Tom Mitchell, in 1997 provided a more formal and currently used definition, stating that a computer program learns from experience (E) concerning a class of tasks (T) and a measure of performance (P) if the performance is measured by a performance (P) and improved by experience (E) (Michalski et al., 2013). For computers

to learn from their environments and their respective experiences, it is necessary to use automation and analysis of the data and analytical algorithms. ML's goal is to automatically learn how to *"research"* and identify new perspectives without human intervention (Henderson, 2019).

ML is organized into three primary research focuses (Michalski et al., 2013) such as:

1. Task-Oriented Studies – focuses on developing and analyzing a learning system to improve performance in a predetermined set of tasks, also known as the *"engineering approach"*;

2. Cognitive simulation – focuses on the investigation and computer simulation of human learning processes;

3. Theoretical analysis – focuses on the theoretical exploration of the space of possible learning methods, independent of the application domain.

Of all these approaches, ML usually employ either task-oriented approaches or cognitive simulation.

Tom Mitchell et al. (Michalski et al., 2013) define the process of learning as a change in the system that allows it to perform better the second time the task is performed. In a formal definition of *task*, the process of learning is not the task, it is the ability to perform the task. For example, if one wants a robot to be able to walk, then walking is the task (Goodfellow et al., 2016).

Learning in ML can be divided into several approaches. The most common approaches are Supervised Learning and Unsupervised Learning.

2.2.2.1 Supervised Learning

In supervised learning, an agent learns a function that maps an input to an output based on the examples of input-output pairs. For example, an input can be camera images, and each image is complemented with an output describing the images as *"bus"* or *"pedestrian"*. Based on these observations, an agent learns a function that predicts the appropriate output (called a label) when given a new image (Russell and Norving, 2020).

Supervised learning originates from the fact that the target (also called output) needs an instructor or a teacher to show the system what to do (Goodfellow et al., 2016). Using the example of the camera images, the teacher is the environment.

The prediction of a house is another example of supervised learning. A set of input features (e.g., number of rooms) are analyzed in response to the labeled values across the many houses predicted to build a prediction model (Theobald, 2018). ML tasks are usually described by how the ML systems should process an example . The most common machine learning tasks include classification and regression. In classification tasks, the

output variables are categories that can be demographic data such as country or a description of food as *"good"* or *"bad"*. The most used classification algorithms are Support Vector Machine, Random Forest, or Naïve Bayes (iPULLRANK, 2020).

In regression tasks, the output variables used are real values which can be the height of a person or tomorrow's minimum and maximum temperature. A common algorithm is Linear Regression. These models can determine an independent variable(s) impact on a dependent variable(s) by finding the best *"form"* to minimize the error. Other regression models combine linear models for more complex scenarios (iPULLRANK, 2020).

A better name for this type of task can be an approximation or numeric prediction. For example, in 1886, Francis Galton wrote an article (Galton, 1886) describing the concept of regression to the mean as the children of tall parents are likely to be taller than average but not as tall as the parents. He showed plots with what he described as *"regression lines"*, which led to regression association with the statistical technique of function approximation rather than the regression to the mean (Goodfellow et al., 2016).

2.2.2.2 Unsupervised Learning

In unsupervised learning, the agent learns patterns without having an explicit feedback (Russell and Norving, 2020) which means no instructor or teacher can assist. Thus, the algorithm must learn to make sense of the data without any assistance (Goodfellow et al., 2016). The dependent variables are not known or labeled, and the model looks for patterns among the independent variables to create an output.

Unsupervised learning aims to create an output with fewer dimensions (fewer features) than the original input data. There is no true outcome observation to check and validate the model, leading to more subjective predictions(Theobald, 2018).

Unsupervised learning is helpful in situations with no single clear prediction goal, and exploratory data analysis is required to uncover new categories. The most common algorithms for unsupervised learning are clustering and association (iPULLRANK, 2020). In clustering, the data is grouped by similar data points and connections that generalize patterns, such as the grouping of suburbs with two-bedroom apartments that generate a high valuation.

In association, the output is based on rules that explore connections between the data, for example, one can create a rule that defines people who buy a product X are more likely to buy the product Y (iPULLRANK, 2020).

2.2.2.3 Ensemble Methods

An ensemble method is a ML algorithm that aims to improve predictive performance on a task by aggregating the predictions of multiple estimators or base models. The components of an ensemble are called base estimators or base learners (Kunapul, 2020)

This type of method leverages the power of the *"wisdom of the crowds"*, which relies on the principle that a group's collective opinion is more effective than any individual in the group (Kunapul, 2020).

A successful case of implementation of ensemble methods is in the Netflix movies recommendations systems (Kunapul, 2020) developed by Andreas Toscher and Michael Jahrer for the contest held by Netflix to improve its recommendation system. Both summed up their success to the quality of the individual algorithms and the general idea of ensembles (Kunapul, 2020).

The popularity of ensemble methods also grew due to their dominance in data science competitions. They succeed with different machine learning tasks in finance, medicine, healthcare, recommendation systems, and many more (Kunapul, 2020).

Ensemble methods aim to overcome overfitting and high variance by combining several low bias models to reduce their variance (through bagging) or combine several low variances models to reduce their bias (through boosting) (Kunapul, 2020).Sequential methods include one learning method called boosting.

In parallel ensemble methods, the methods generate the base estimators in parallel, and the methods take a path of democracy. Some algorithms allow the base estimator to make a prediction, and then the prediction with the highest vote (for classification) or the highest average (for regression) is picked (Nelson, 2020). Parallel ensemble methods are distinguished between homogeneous and heterogeneous learners, depending on the learning algorithm used (Kunapul, 2020). The homogeneous parallel ensemble uses the same ML algorithms generating diverse base estimators through bagging with random sampling, patches, or even through Random Forest called ExtraTrees (Kunapul, 2020). The heterogeneous ensemble methods use different learning algorithms to ensure ensemble diversity. For example, a heterogeneous ensemble can consist of three base estimators – a decision tree, a support vector machine, and an artificial neural network (Kunapul, 2020). The base estimators are trained independently of each other. An ML method called stacking generates heterogeneous results.

The earliest heterogeneous ensemble methods, such as stacking, were developed around 1992, however, these methods rose to popularity due to the Netflix Prize competitions (Kocaguneli et al., 2009).

Kocaguneli et al. (Kocaguneli et al., 2009) evaluated a heterogeneous ensemble of multiple learners by averaging them. A state-of-the-art method using heterogeneous learning for stacking regression is the Cocktail ensemble learning, developed by Yu et al. (Yu et al., 2007) in 2007. The approach used an error ambiguity decomposition to analyze the optimal linear combination of two ensembles and extended it to multiple ensembles via pairwise combinations, claiming a hybrid ensemble is superior to the base ensemble, a simple averaging.

2.2.3 Artificial Intelligence Marketing

Artificial Intelligence Marketing, abbreviated to AIM, uses AI technologies to make automated decisions based on data collection, analysis, and additional observations of audience or economic trends that may impact marketing offers (Evolution, 2020).

Any form of automation entails a fixed algorithm that receives an input and returns an output as a response to a given problem. For instance, suppose it is given a list of numbers, and it is required to sort these numbers into ascending order. The system will introduce the numbers as input, sort them by applying some rules and mathematical manipulation, and then returns the sorted list as output (Liebowitz, 2020).

This simple automation can extend to higher and more complex tasks, such as detecting fraudulent transactions. The system receives a list of possible fraudulent actions and searches, creates correlations in the data to find the actions leading to fraud, and returns the fraudulent transactions (Liebowitz, 2020).

Advanced ML techniques are applied to anticipate subscriber needs, behavior and boost prediction accuracy. It allows the automation of certain marketing decisions by detecting subtle and hidden events in the subscriber behavior, meaning it is detected hidden correlations between the subscriber and the likelihood of certain actions performed by the former. These correlations or relationships are registered as new data points to improve the system's performance (Liebowitz, 2020).

One of the main goals of marketers is to find the best way to offer the right product to the right subscriber/customer at the right time. AI allows a better way to perform and deliver those offers by providing better ways to understand the right time, the right way, and the right products for the subscriber based on their profile. Furthermore, these studies allow forecasts about future behavior, which leads to better personalization and individualization on a bigger scale. Consequently, it leads to better-planned campaigns with greater relevance/precision and allows marketing teams to send tailored and personalized communications (Valigi and Mauro, 2020).

Data-Driven AI describes a business state where the data is used to power decision making and other related activities efficiently in real-time and is currently improving subscriber experiences through personalization. The characteristics of this type of AI are well-integrated data and algorithmic automation using AI (Knight, 2021).

Companies already use AI and ML tools to boost their sales and improve their interaction with their subscribers.

Starbucks uses predictive analysis to identify the needs of its consumers/subscribers and their preferences. With ML algorithms, the company obtained an annual revenue of 21% (9% more than without ML's presence). Carrefour and Target introduced beacons to gather data to predict consumer behavior and send personalized promotions while consumers are in their stores buying products. As a result, the company states an increase in

the engagement rates by 400% and the number of subscribers in their mobile applications (Mitall, 2021).

Unilever uses AI, ML, and voice-related technologies to deliver personalized and immersive experiences to Unilever's consumer platforms. The company is building its database using online customer registrations, third-party sites that register the consumer visits, and data from the loyalty cards (Venkatesan and Lecinski, 2021). JP Morgan Chase boosts customer acquisition by using AI and ML to optimize the messaging for its digital advertising and direct mail. Checkli, a SAS provider of checklists, empowers retention by using AI-powered email optimization to increase mobile application engagement and decrease spam reports (Venkatesan and Lecinski, 2021).

According to a study by Salesforce (Salesforce, 2017), in 2017, 57% of marketers use at least some form of AI, and 27% intended to add it to their campaigns. It also states that 30% of the companies planned to use AI in at least one sale process by 2020. Another study by Smart Insights (Insights, 2018), in 2018, indicates that the personalization of marketing communications and data analysis were the dominant areas for AI marketing, with 29% and 22% of the respondents (respectively) stating that it would be the areas with the most significant potential for improvements with AI. In 2021, another study conducted by Marketing AI Institute (Institute, 2021) affirms that the COVID-19 pandemic accelerated AI-driven digital transformations across many companies. Additional McKinsey research (Institute, 2021) claims that 25% of the 2400 leaders interviewed said the AI adoption increased due to the pandemic.

2.2.4 Artificial Neural Networks

A neural network is a computational model inspired by the structure of the human brain. The human brain comprises several nerve cells, called neurons. Those have a three-part structure consisting of a cell body, a set of fibbers called dendrites, and a single long fiber called the axon. The dendrites and the axon stem from the cell body, and the dendrites of one neuron are connected to the axons of other neurons. The dendrites act as input channels to the neuron and receive signals from other neurons along their axons (Kelleher, 2019).

The axons act as the output channel of a neuron, and so other neurons whose dendrites are connected to the axon receive the signals sent as input. If the incoming stimuli is strong enough, the neuron transmits an electric pulse called an action potential along its axon to the connected neurons. In a way, the neuron acts as an *"all-none switch"* that takes in a set of inputs and either outputs an action potential or does not. This brief explanation explains the analogy between the structure of the human brain and the models called neural networks (Kelleher, 2019).

An artificial neural network contains simple information-processing units called neu-

rons organized into layers. Usually, deep learning networks have several hidden neurons, where the minimum number to consider deep is two (Kelleher, 2019). A neural network also consists of an input and an output layer, with hidden layers in between. Each neuron takes numeric values as inputs (the circles presented in Figure 2.1) and maps them into a single output value. Each input to a processing neuron is either the output of another sensing neuron or the output of another processing neuron. Each connection in a network is directional, and the connections flow in one direction. Also, each connection has a weight associated, affecting how a neuron processes information it receives. Essentially, training an ANN involves searching for the best (or optimal) set of weights (Kelleher, 2019). A neuron also implements a two-way process to map inputs to outputs, where the first step involves calculating the weighted sum of the inputs to the neurons. Then, the weighted sum calculation result is fed through a second function that maps the results to the neuron's final output value. This function is known as the activation function, and the most used is the rectifier activation function (Kelleher, 2019).

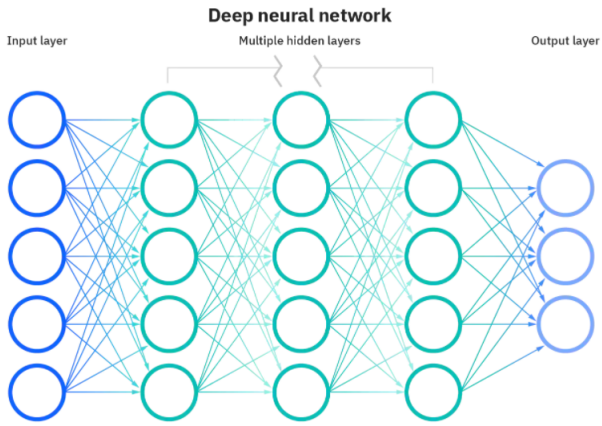

Figure 2.1: Artificial Neural Network's Architecture (IBM, 2020)

2.2.5 Deep Learning

Deep Learning is one of the most advanced fields of ML. LeCun et al. (LeCun et al., 2015) define Deep learning as a subfield of AI that focuses on developing models of neural networks capable of making decisions based on the data. The models, composed of multiple layers of processing, can learn several representations with multiple levels of abstraction, which allows the discovery of intrinsic structures when using backpropagation. This algorithm indicates how a machine should change its internal patterns in each layer (LeCun et al., 2015).

Deep learning models are suitable for contexts where the data is complex and for large datasets. The models automatically learn features from low-level data and complex

nonlinear mappings from inputs to outputs. This ability is better applied to extensive datasets where the performance usually outperforms the ML approaches (Kelleher, 2019).

Deep learning models learn subscriber behavioral patterns that allow a more personalized and conversational experience between the subscriber and the companies, enhancing the *"hyper-personalization"* of the experiences. It can also help make decisions customized for each segmented subscriber (Walden, 2020). In addition, the models are helpful in marketing since they enable companies to look for nonlinearities in the data leading to more comprehensive views of the subscribers and their interaction with the company (Urban et al., 2019). This capacity turns deep learning models more interesting since they assume patterns that traditional ML models cannot identify (Hay, 2019).

The concept of deep learning underwent different phases, names, and stages of development and research.

The first phase, called cybernetics, happened between 1940 and 1960, and models were based on biological learning, demonstrating the learning that happened in the brain. Artificial Neural Networks appeared in this phase. McCulloch-Pitts created the neurons, which consisted of a linear model that recognized two different categories (positive and negative) and a human operator assigned the weights. In 1950, perceptron was created by Rosenblatt, which became the first model to learn weights (Goodfellow et al., 2016). The second phase, connectionism or distributed parallel processing, occurred between 1980 and 1990 and expanded the cognitive sciences (interdisciplinary approach to understand the mind combining different levels of analysis). The models were neuronal implementations, and the main idea consisted of having many straightforward computational units achieving intelligent behavior when interconnected in a network. One of the greatest successes of this phase was developing the backpropagation algorithm to train neural networks, created by Rumelhart and LeCun (Goodfellow et al., 2016). Finally, the current phase, called Deep learning, was driven to focus on unsupervised learning, Big Data, and deep models' ability to generalize well from small datasets (Goodfellow et al., 2016).

2.3 Data Preprocessing

Most of the programming work consists of data preparation in data analysis and modeling, which includes processing, transforming, or rearranging data (Press, 2020). A survey conducted by Forbes (Press, 2020) found that data preparation accounts for about 80% of the work of data scientists, with 60% of the data scientists spending their time cleaning and organizing data (Figure 2.2).

This step is necessary since the data stored in the original files or provided by the databases cannot be trained or modeled instantly, making it unsuitable for modeling. The process dedicated to data preparation and preprocessing is called Feature Engineering. Tomasz Malisisichz, a research scientist at Amazon Robotics AI, describes it as a method

to transform new raw data into features that describe and represent the problem to be solved, leading to better and improved accuracy and performance (Brownlee, 2020b).

Pedro Domingos states that some ML projects can succeed or fail. The difference lies in the features used (Domingos, 2020).

Scikit-learn (scikit learn, 2020), also known as sklearn is a python library to implement ML models and statistical modelling. Through this library, it is also provided functionality for feature extraction, feature selection, and among other techniques (Ashish, 2021)

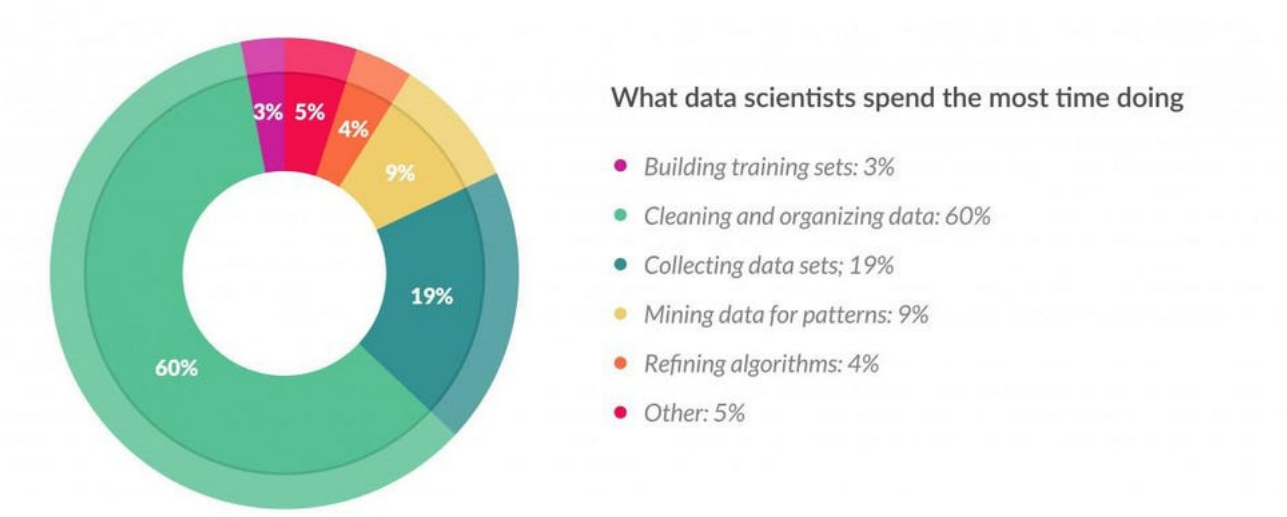

Figure 2.2: A descriptive graph of the data scientists' work according to Forbes survey (Press, 2020)

2.3.1 Missing Data

Real-world data usually has many missing values. The absence of values can affect the performance of ML models. The handling of missing data is significant during this preprocessing phase, as many ML algorithms do not support missing values. Since ML algorithms cannot work with missing feature, it is crucial to convert them to numerical values. It is possible to convert them by imputing missing values with the mean or median, or by filling the missing values with the value zero or eliminate them, depending on the context of the problem. Before any operation is performed, missing values need to be quantified (Brownlee, 2020c).

2.3.2 Date Features

During this preprocessing phase, it was also essential to understand how to approach and handle features with date formats and their properties.

Date and time variables are features that contain information that require the concepts of feature engineering to transform these features from raw data to insightful information that can be applied to the ML models.

Typically, these variables are seen as *DateTime* variables and contain many different labels corresponding to a specific combination of date and time. Feeding the ML models the *DateTime* variables in their raw format is not recommended, so converting these features into multiple features is necessary (Galli, 2020).

2.3.3 Categorical Features

Categorical predictors have categorical variables with *"good"* and *"bad"* or other qualitative types of values as output values. These values are not quantifiable, and most ML models require predictors/variables to be numeric. This conversion is executed using a technique called categorical encoding that involves several approaches (Galli, 2020).

There are several approach, such as *Label Encoding* and *One Hot Encoder* as the most used. However, it is possible to use other approaches such as *Ordinal Encoding* and *Binary Custom Encoding*. This approach is a combination of hash encoding and one-hot encoding. The categorical feature is first converted into numerical using an ordinal encoder. Then the numbers are transformed in the binary number. After that binary value is split into different columns (Saxena, 2020).

Label Encoding has the disadvantage that numeric values can be *"misinterpreted"* by the algorithm. *One hot encoding* is useful, although it can cause columns to expand if many unique values are present in the column significantly. *One hot encoding* can lead to many features in many categories, slowing down training and degrading performance (Géron, 2017).

In some scenarios, depending on the dataset, it might be needed to use label encoding and one-hot encoding to create a binary column that meets the developer's needs. This approach is called custom binary encoding (Galli, 2020). The *OrdinalEncoder* assigns an integer value to each unique category. The estimator transforms categorical features into one new integer feature. Scikit-learn supports this feature by using *OrdinalEncoder* (Géron, 2017).

2.3.4 Train-Test Split

This technique splits the dataset into two subsets called train and test set to improve the computation times since the dataset used contained too many observations that required computational resources (e.g., RAM or CPU usage) and had slower computation times.

The idea of a *"sufficiently large"* dataset is specific to each predictive modeling problem; however, it can be seen as enough data that allows splitting the dataset into train and

test sets. Each of the train and test sets are suitable representations of the problem domain (Brownlee, 2020c).

This procedure splits the original dataset with a random selection technique. The train set is used to fit the model, and the test set is used to evaluate the trained data's performance by comparing the predictions' values to the actual values. The procedure has one main configuration parameter: the size of the train and the test sets. The most common split percentage include (Brownlee, 2020c):

- Train: 80% and Test: 20%
- Train: 50% and Test: 50%

There is no optimal split percentage, so one can choose a split percentage that believes it meets the project's objectives.

The Scikit-learn (scikit learn, 2020) provides an implementation of the train-test split evaluation by calling the *train-test split()* function.

It is done to ensure that the datasets represent the original data and a representative sample of observations from the problem domain. It is achieved by fixing the seed for the pseudo-random number generator used when splitting the dataset.

2.3.5 Cross-Validation

The next step of developing the solution is to estimate the model's skill on the new data. Cross-Validation is the process primarily used to estimate the unseen data using a limited sample to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model (Brownlee, 2020a).The Scikit-learn library provides an implementation that allows the split of the given data sample by calling the *KFold()* class.

In addition, the process of Cross-Validation can be embedded in a few estimators present in the Scikit-learn library, those are the estimators where their name end with CV. *GridSearchCV* and *RandomizedSearchCV* are examples.

2.3.6 Time-Series Split

When working with time-related data and data that can change through time, it can be helpful to use a time-based splitting approach, in addition to the traditional approach *train-test split()*. This approach uses time-based cross-validation, creating a *"sliding window"* training approach (Herman-Shaffar, 2020). Figure 2.3 explains the time-series cross-validation approach.

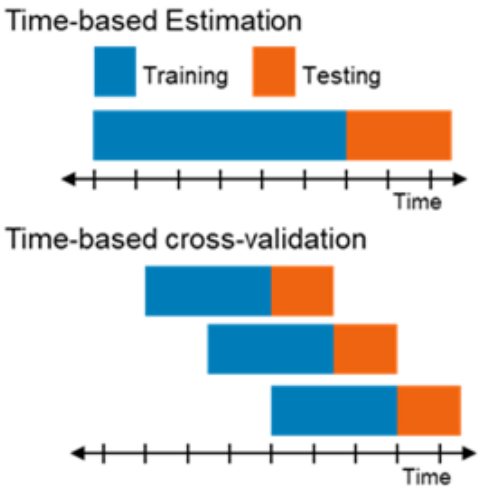

Figure 2.3: Time-series Cross-Validation approach

2.4 Regression Evaluation

A common question to regression predictive modeling is how one can calculate the accuracy of the regression models. The answer to the question is that it is impossible to calculate the accuracy since it is a metric only implemented in classifications tasks.

The performance of regression tasks must be seen as an error in the predictions. For example, if one wants to predict a numeric value (e.g., height), one needs to know how close the predictions are to the actual values (Brownlee, 2020d).

Error addresses this and summarizes, on average, how close the predictions are to the actual values. The error metrics more commonly used for evaluation of the regression task performance is (Brownlee, 2020d):

- Mean Absolute Error (MAE) - an average of the absolute differences between predictions and the actual values. It gives an idea of the magnitude of the error but has no idea of the direction. In other words, it allows a perception of how *"wrong"* the predictions are. The perfect MAE value is 0, which means that all the predictions match the expected values. However, these values are rare, and a *"good MAE value"* is relative to the problem's domain and its context in the real world.

- Mean Squared Error (MSE) – mean absolute error that provides a total idea of the magnitude of the error. It minimizes the mean squared error between the predictions and the expected values.

- Root Mean Squared Error (RMSE) – square root of the average root of squared error in a set of predicted values, without considering the directions. Lower values mean a better model, and it is always greater in magnitude than MAE. If the RMSE values

are loosely much higher than the MAE, the error variance is high, and outliers in the data can exist.

- R squared - metric that indicates the goodness of fit of a set of predictions to the actual values. In statistical literature, this metric is called the coefficient of determination with values between 0 and 1. Statistically, it represents the proportion of the variance for the dependent variables explained by the independent variables in a regression model.

2.5 Predictive Models Related

This section presents the scientific development and commercial implementations in predicting the best time to send marketing communications in different communication channels such as Email or SMS and segmentation of subscribers. The predictions are made using different user behavior such as open rates and clickthrough based on the subscribers' profiles.

2.5.1 Predictive Models for Sending Marketing Communications

Deligiannis et al. (Deligiannis et al., 2020a) describe models to send marketing communications at the ideal time for a subscriber to repurchase a product. The communications are sent by SMS through a Rabbit MQ Protocol (Naik, 2017) that allows sending and receiving mobile communications. The first model (Figure 2.4) consists of regression algorithms to identify the number of days between purchasing the final product and the subscriber's next purchase. This model uses as input: i) the open messages rates, ii) the purchase transactions, iii) the participation frequency, and iv) clickthrough rates. The second proposed model (Figure 2.5) is based on the date predicted by the first model to establish an approximate date for the automatic reminder of the repurchase. XGBoost (Chen and Guestrin, 2016) has the best performance of all the implemented algorithms with 95% of confidence. A limitation regarding this project is the small size of the dataset they worked on.

Deligiannis et al. (Deligiannis et al., 2020b) intend to predict the impact of business campaigns by estimating the percentage of subscribers who interact with the communications received. The model (Figure 2.6) considers as input: i) the email hyperlinks, ii) the clickthrough rates, iii) the time between an email sent, and iv) a subscriber's action time. It is used regression algorithms to estimate clickthrough rates. Natural Language Processing (Goldberg, 2015) processes text from messages to understand the message's impact and content. Subscribers are segmented by the service provider of each client company with a clustering algorithm.

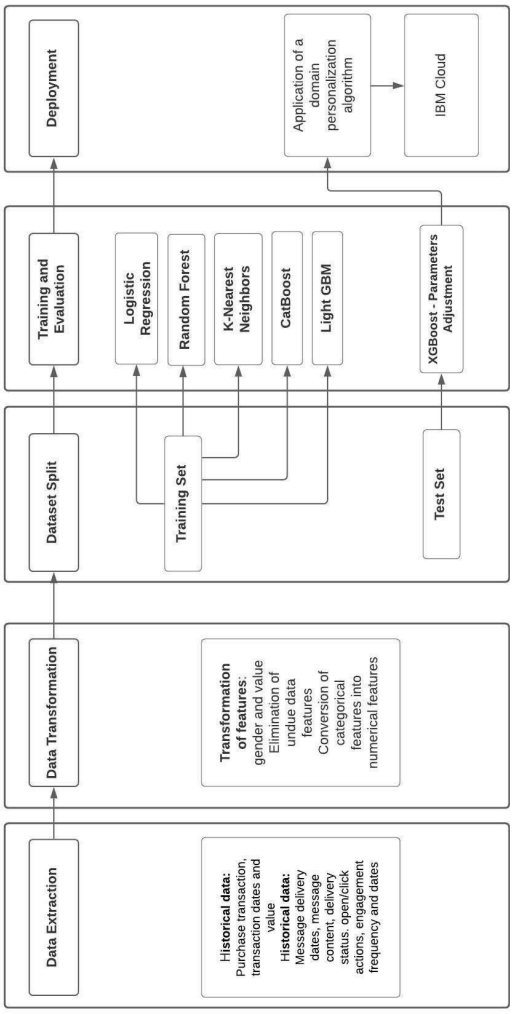

Figure 2.4: Architecture of the first model proposed by (Deligiannis et al., 2020a)

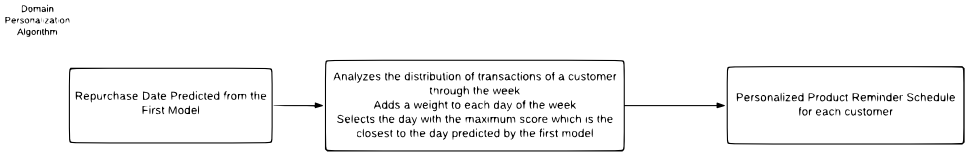

Figure 2.5: Implementation of the second model proposed by (Deligiannis et al., 2020a)

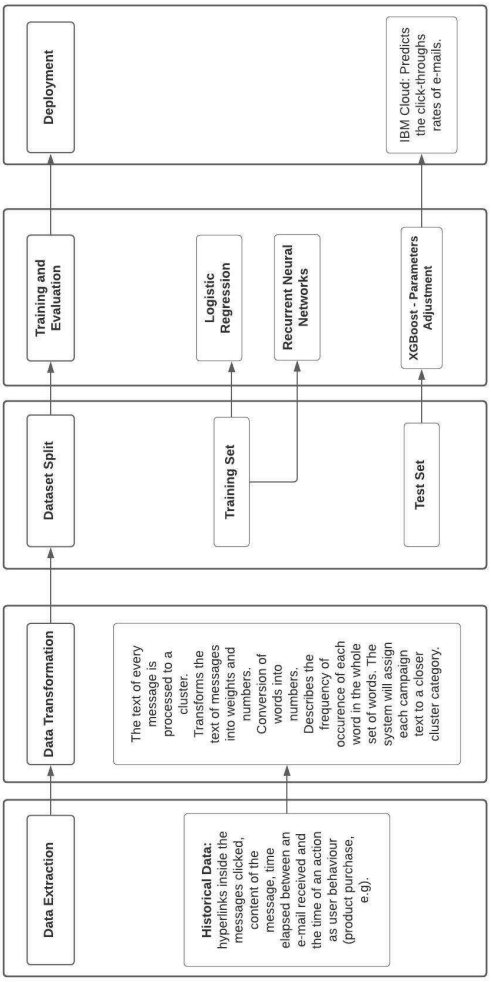

Figure 2.6: Architecture of the model proposed by (Deligiannis et al., 2020b)

Paralic et al. (Paralic et al., 2020) determine the best time to send communications based on the information collected from email communications between companies and their subscribers. The prediction of the best time is made through segmentation by subscriber types/profiles. It is modeled three classification models (Figure 2.7) for the user's status (if the email was opened) and the time of that opening (hour and day). The classification algorithms are Decision trees (Quinlan, 1986), Random forest (Breiman, 2001), and Naïve Bayes (Barnard and Bayes, 1958), where decision trees obtained the best performance with 93% for the first model (open/non-open) with 80.54% for the second model and 88.63% for the last model (opening day).

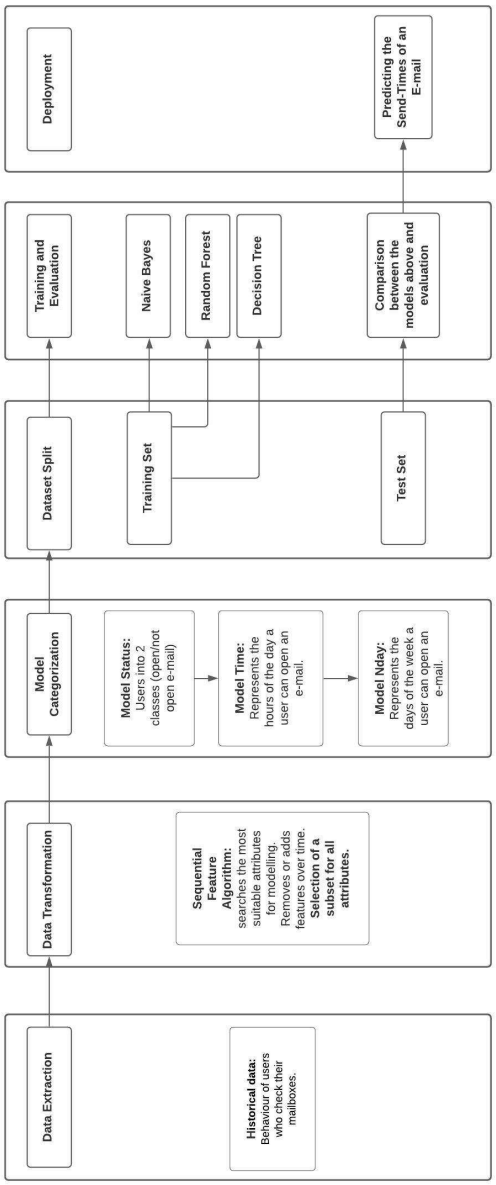

Figure 2.7: Architecture of the model proposed by (Paralic et al., 2020)

Conceição et al. (Conceição and Gama, 2019) analyze which factors might influence

the opening rates of marketing emails with a financial nature. Based on the open rates, two classification models (Figure 2.8) were modeled that determine whether a campaign was successful. The classification algorithms used are Decision trees (Quinlan, 1986), Random forest (Breiman, 2001) and Gradient tree boosting (Friedman, 2001) to improve and fine-tune the parameters. Random forest obtained the best performance for campaigns labeled as *"success"* with an F1-score of 71% and campaigns labeled as *"failures"* with an F1-score of 93%. F1-score combines the precision and recall metrics into a single metric, consisting on the average of precision and recall (Korstanje, 2021). The classification models are based on: i) the names of campaigns, ii) recipients and senders, iii) the content of the message, iv) the number of emails sent, v) the number of emails delivered, and vi) the number of open emails.

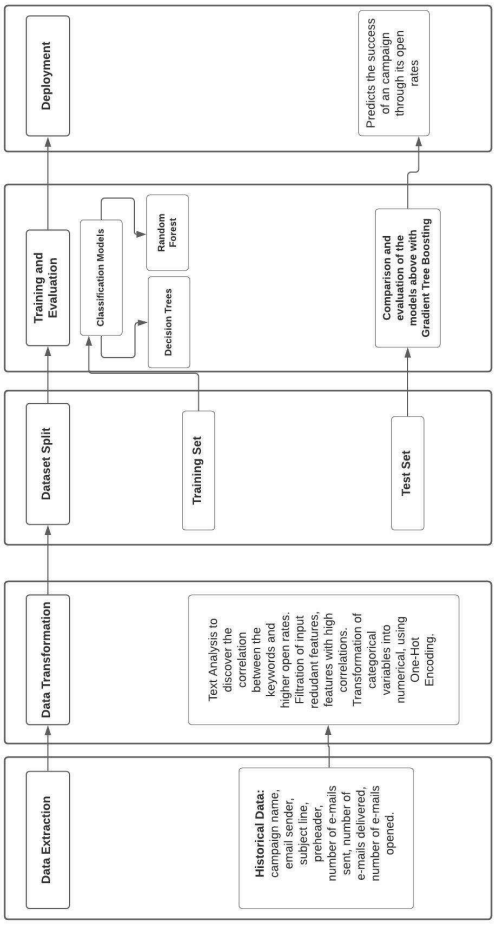

Figure 2.8: Architecture of the model proposed by (Conceição and Gama, 2019)

Luo et al. (Luo et al., 2015) describe two classification models (Figure 2.9) to predict the opening rates of emails based on the characteristics extracted from the email and user profiles. The classification algorithms used are Decision trees (Quinlan, 1986) and Support vector machines (Cortes and Vapnik, 1995). Decision trees performed better with an F1-score of 80% in the opening rates. Support vector machines achieved an F1-score

of 74% in the opening rates. The second model assesses the domains' impact on the performance, so the domain is filtered. Decision trees obtained an F1-score of 72% and support vector machines an F1-score of 70%. Both models are based on: i) the emails sent, ii) the user's action, iii) the content of the email sent, iv) the day and the time when the email was opened, v) the location, and vi) the users' email domain.

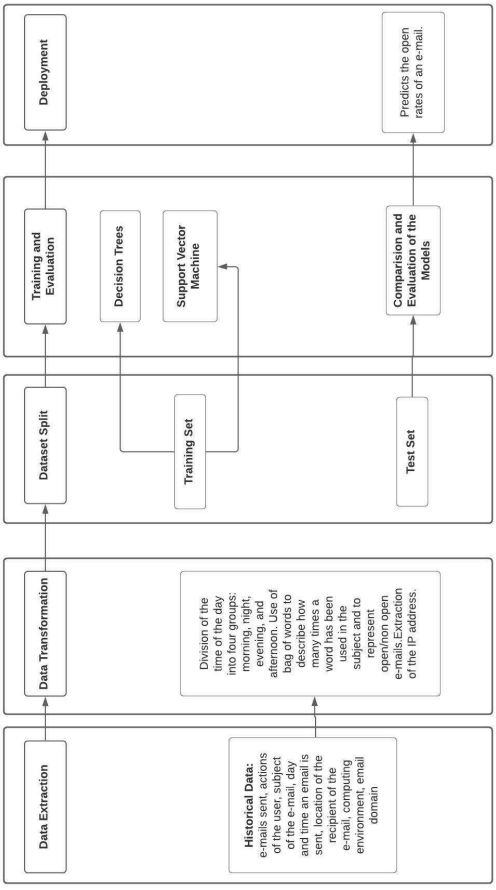

Figure 2.9: Architecture of the model proposed by (Luo et al., 2015)

Sinha et al. (Sinha et al., 2019) determine email opening times based on the subscriber interest and engagement with the communications received. The model (Figure 2.10) is based on: i) the number of open messages, ii) clickthrough rates, and iii) the last message sent. In addition, the model uses classification algorithms to identify the opening event and regression algorithms such as Cox Proportional Hazard Regression (R., 1972) to determine the time most likely for subscribers to open their communications. A survival analysis (Glazier, 2019) approach is used to join the opening event and the time of email opening. It is concluded that 43% of email openings happen between 6 am and 12 pm, with 57% and 74%. It is also stated that after midnight, openings become rarer and that 90% of emails are not opened since the subscribers ignore them.

Singh et al. (Singh et al., 2020) determine the times for sending email communications

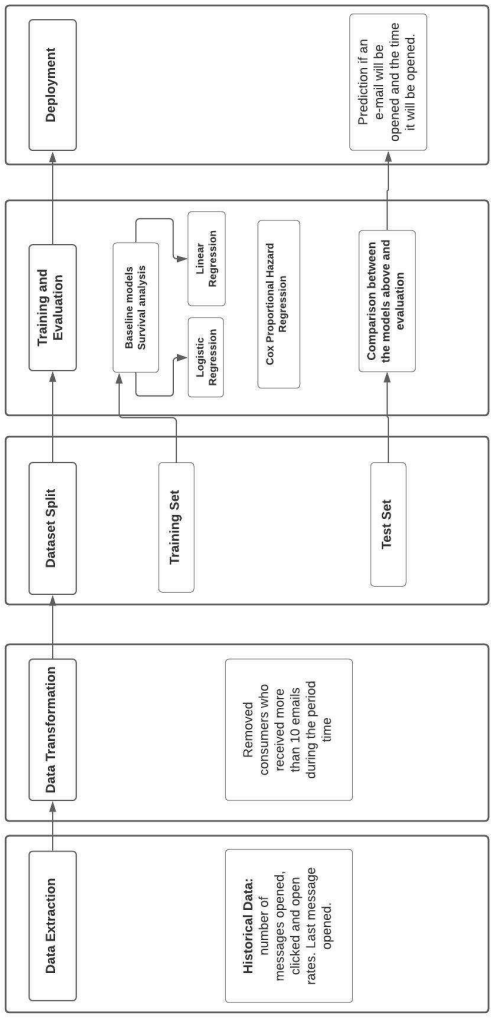

Figure 2.10: Architecture of the model proposed by (Sinha et al., 2019)

for each user. The model (Figure 2.11) is based on: i) the last open email, ii) the number of days between the last open email, iii) the number of days that occurs since the email was sent and the time was opened, iv) the time intervals of the current email, v) clickthrough rates and vi) the dates and the frequency of purchase. The authors highlight the duration of the opening times to identify the time to send an email. A deep learning approach (Kvamme et al., 2019) uses survival analysis to model sequential information dependency and a regression model to determine the opening times. The regression algorithm used is the Cox Proportional Hazard Regression (R., 1972).

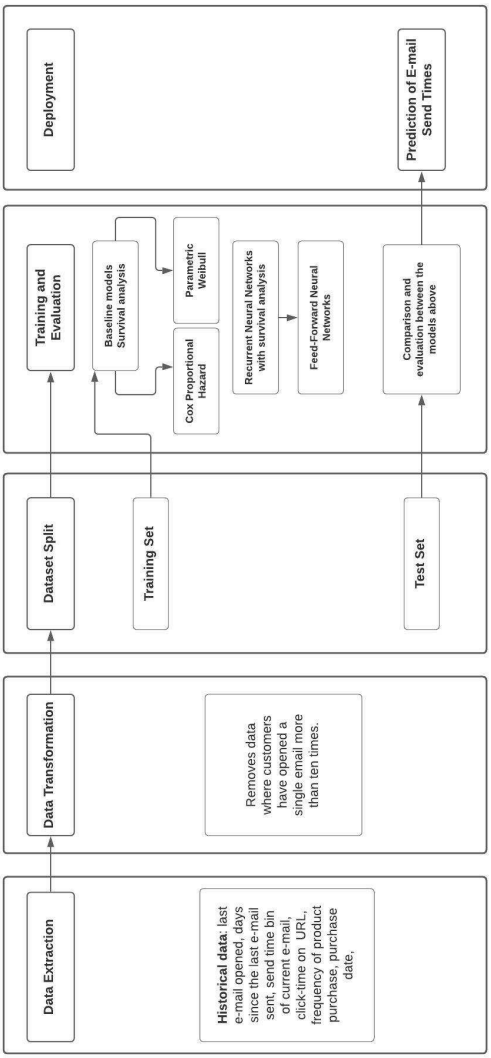

Figure 2.11: Architecture of the model proposed by (Singh et al., 2020)

Singh et al. (Singh et al., 2019) use a management system to determine transmission times for digital content. The authors model deep learning models (Figure 2.12) such as recurrent neural networks to predict transmission times and are based on: i) the last open message, ii) the number of days between the message sent and the purchase, iii) binary indicators to determine whether the purchase was made online and iv) the opening rates.

The neural network model uses a survival analysis approach and a classification algorithm to determine the opening event. The authors use a Feed-Forward Network (Zell, 2000) to compare and evaluate the best transmission times than the RNN model. The system is also able to determine the number of messages and the frequency of transmission times. The model with logistic regression obtained an F1-score of 0.602%, and the model with FNN obtained an F1-score of 0.627%, and the model with RNN an F1-score of 0.624%.

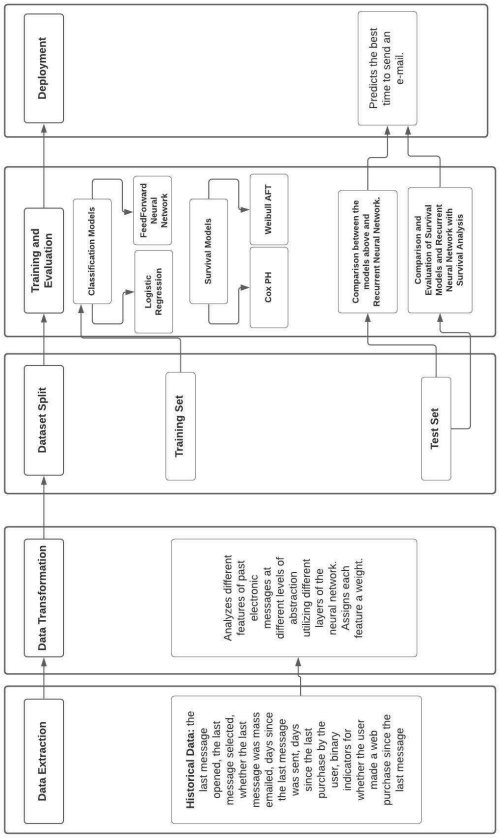

Figure 2.12: Architecture of the model proposed by (Singh et al., 2019)

Piersma et al. (Piersma and Jonker, 2004) determine the frequency in which an email should be sent to the individual subscriber to establish a long-term relationship. The frequency of sending is optimized, with restrictions on the number of emails sent in a time interval. A model (Figure 2.13) is developed to predict the sending frequency using a Hidden Markov Model Classifier (Baum, 1972). This allows the model to determine the number of emails to be sent in a period of time. The model uses as input i) the user's personal information, that is, the user's status, how the user was approached, ii) the date of their inactivity, iii) the date of each email sent, iv) the date of each answer and v) the size of each answer.

Table 2.1 describes the summarized features implemented in the models proposed by each paper mentioned previously.

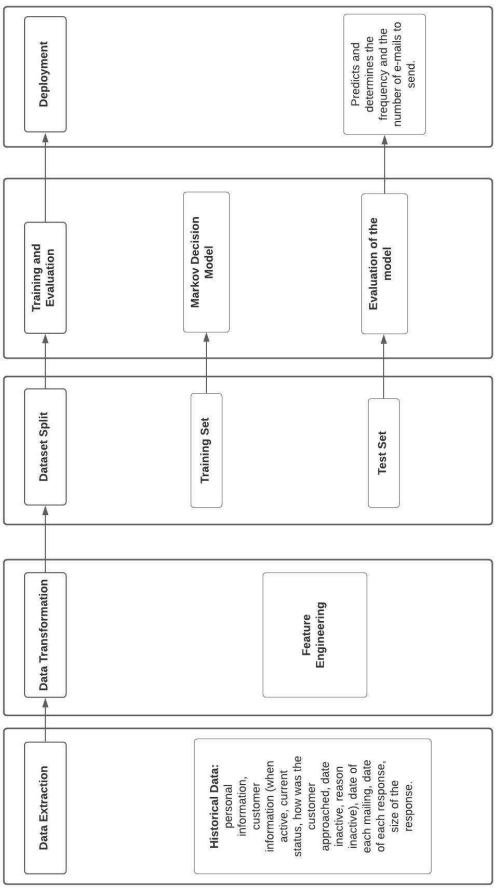

Figure 2.13: Architecture of the model proposed by (Piersma and Jonker, 2004)

Table 2.1: Description of the features implemented in the models proposed by the papers listed in Section 2.5.1.

Paper	**Features implemented by the models**
Deligiannis et al. (Deligiannis et al., 2020a)	Open rates, participation frequency, and purchase transactions
Deligiannis et al. (Deligiannis et al., 2020b)	Email hyperlinks, clickthrough rates, the time between an email sent and the subscriber action time
Paralic et al. (Paralic et al., 2020)	User status and open rates of the email
Conceição et al. (Conceição and Gama, 2019)	Names of the campaigns, recipients and senders, content of the message, and the number of emails sent, delivered, and opened
Luo et al. (Luo et al., 2015)	Emails sent, the content of the email sent, user's action and email domain and their location and the day and time the email was open
Sinha et al. (Sinha et al., 2019)	Number of open messages, the last message sent, and clickthrough rates
Singh et al. (Singh et al., 2020)	Last opened email, the number of days between the last opened email, the number of days that occur since the email was sent, open rates and time intervals of the current email, clickthrough rates, and the dates and frequency of the purchase
Singh et al. (Singh et al., 2019)	Last opened message, the number of days between the message sent and the purchase, binary indicators to determine whether the purchase was made online;
Piersma et al. (Piersma and Jonker, 2004)	User's status, personal information, how the user was approached, date of their inactivity, date of each email sent, date of each answer, and the size of each answer

Salesforce Einstein uses Send Time Optimization to determine the best time to send a communication. ML algorithms and 90-day data are used to determine the best time, within 24 hours, to send a message. It also collects information on twenty factors, such as the number of emails opened and the day of the week that the email is opened (Smartt, 2020). Bluecore also uses Send Time Optimization, and it tries to estimate the probability of opening per hour for each user, using a function that indicates the optimal value for sending. The value of the function is obtained by regression mechanisms and historical data (Tunnell, 2020).

Seventh Sense sends individual marketing emails at an optimal time and frequency, identifies the needs of its consumers, and sends an email when it is most likely to be opened and answered (Sense, 2020).

Listrak uses the user's behavior to send specific emails and messages based on the consumers' timezone and activity time, using Send Time Optimization (Brophy, 2020). Dynamic Yield registers and records user behavior and neural networks to identify that

behavior patterns (Yield, 2020).

Velocidi uses predictive audience and ML algorithms to predict which consumers will most likely interact with the emails. ML model that detects patterns in the users' online behavior is created, and additionally, segments are created to isolate potential buyers and an (Velocidi, 2020).

Otto, a German e-commerce company, uses AI algorithms to identify and learn from significant data patterns such as past transactions, social media trends, shopping patterns, online history, and seasonal shopping patterns (Economist, 2020).

2.5.2 Predictive Models for Subscriber Segmentation

Jonker et al. (Jonker et al., 2004) describe an optimization approach to segment consumers into homogeneous groups of customers and determine an optimal approach policy (e.g., what action to take from a set of available actions) towards each segment to maximize long-term profits (Figure 2.14). The authors implement the optimization framework in a direct email setting for a charitable organization and use an RFM approach to segment the customers. The model takes as input socio-demographic variables or variables that describe past response behavior. When using the RFM approach, the number of mailings without response since the last contact of the customer is used as a recency variable (R), the response percentage over the last two years defined by the total number of responses and divided by the number of mailing and the response percentage over the period of the registration is used as the frequency (F) variable, and it is used the average purchase of the previous two years and the whole period of the registration as the monetary (M) variable. A Markov decision model is used to achieve a marketing strategy policy with the best long-term performance. In order to find a new candidate for the segmentation, it is implemented a local search method with genetic algorithms where it searches for an alternative segmentation in the neighborhood of the current segmentation. Figure 2.15 illustrates the proposed procedure from segmentation to an optimal marketing strategy.

Figure 2.14: The general procedure implemented by the authors in (Jonker et al., 2004)

Chan et al. (Chan, 2008) describe a novel approach that combines customer targeting and customer segmentation for campaign strategies. The authors identify customer behavior using the RFM approach and transform data into binary strings using genetic algorithms (GA) to select customers. The model considers input variables that connect customer behavior and campaign programs, such as customer transaction data and demo-

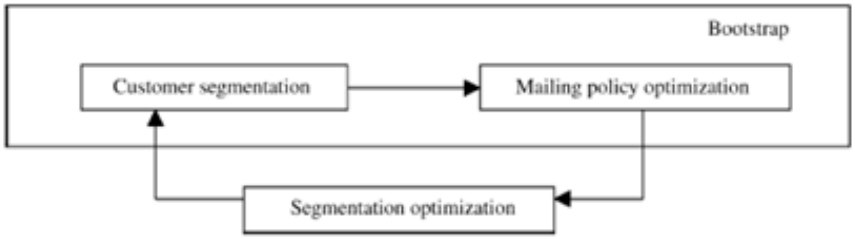

Figure 2.15: The proposed procedure from segmentation to an optimal marketing strategy proposed by (Jonker et al., 2004)

graphic data. In order to perform the segmentation by RFM, it is introduced the latest purchase amount as the recency (R) variable, the total number of purchases during a specific period as the frequency (F) variable, and the monetary value spent during one specific period as the monetary (M) variable. It is also implemented a LTV model that considers the fitness values of the genetic algorithms and evaluates the segments created and proposed by the RFM segmentation. This model considers the current customer value as input and predicts a potential customer value, considering the correlation between a campaign strategy and the customer value. In addition, the authors assume promotions increase customer value, so it is segmented the campaign programs into four major marketing strategies – acquisition, growing, retention, and relationship management strategies. The acquisition strategy aims to attract new customers, while the growing strategy is designed to increase customer values delivered by a campaign promotion program. The retention strategy aims to retain customers as long as possible and raise customer loyalty. Meanwhile, the relationship management strategy focuses on targeting profitable customers or those with good potential. Figure 2.16 illustrates the framework created for the proposed customer segmentation.

Christy et al. (Christy et al., 2021) describe an efficient segmentation of customers categorized into groups based on similar behavior using the RFM approach. Initially, is performed RFM analysis on the customer's transactional data, and then, it is implemented a clustering segmentation using traditional algorithms such as K-Means. The variables used in the RFM approach are the number of days a customer takes between two purchases as the recency (R) variable. In a specific period, the number of purchases a customer makes is used as the frequency (F) variable. The smaller value of this variable indicates that the customer visits the company repeatedly in a short period, and the greater value implies the customer is less likely to visit the company. The higher the value of the frequency, the more loyal the customer is. Lastly, the amount of money spent by the customer is used as the monetary variable. The amount of money earned by each segment of customers segmented by the RFM approach is calculated to find the segment of customers that gives more revenue to the company. When implementing the clustering segmentation

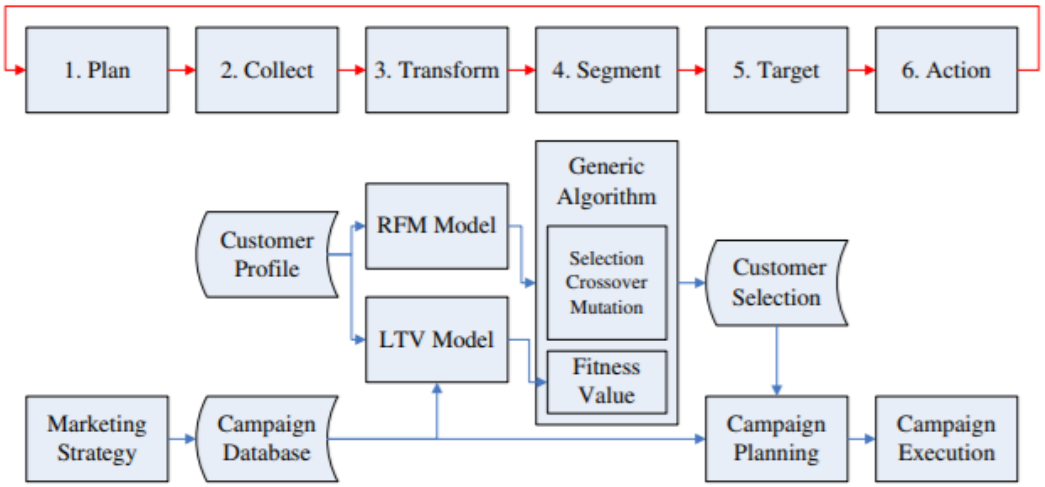

Figure 2.16: Framework of the customer segmentation proposed in (Chan, 2008)

using the K-Means algorithm, the authors limit the number of clusters to ten. Additionally, a Fuzzy-C algorithm is applied, an algorithm deriving from the K-Means but with better results when applied to larger datasets. The authors also propose a novel idea of choosing the initial centroids based on the repetitive median to reduce the iterations and times of each segmentation performed by the K-Means algorithm. Figure 2.17 illustrates the framework created for the proposed customer segmentation.

Figure 2.17: Framework of the RFM K-Means analysis implementation in (Christy et al., 2021)

Carmein et al. (Carnein and Trautmann, 2019) propose a stream clustering algorithm applicable to customer segmentation where the algorithm can track segments over time

without expensive recalculations. The authors propose this solution considering that the biggest challenge in customer segmentation is that segmentation often relies on the customer's purchase history or their profile. This data often changes over time (e.g., when the user purchases or changes their address in the profile). These changes require updates on the existing observations and adjustments on the segments' clustering, even when the observation has already been incorporated into the model. The algorithm Stream clustering (Carnein and Trautmann, 2019) extends traditional clustering, which handles a continuous stream of new observations. It updates the underlying clustering over time without the need to recompute the entire model. In order to make this work, the authors remove the existing observations from the clustered segments before re-adding them with the updated values.

Yan et al. (Yan and Li, 2006) describe an approach of customer segmentation considering customer purchase behavior based on a neural network and a clustering technique (Figure 2.18). The authors cluster customers based on their dynamic attributes, which is, data about the customer that will change over time, such as customer transaction data. In order to cluster customers by purchase history, the authors include trend attributes that describe an upward, downward, or balance trend of the customer's transaction amount. In other words, it is used to reflect the long-term trend of the customer. In addition to the segmentation, the authors use the mean value of the customer's purchase amount, which reflects the customer's contributions to the overall turnover of the enterprise and the purchase frequency to reflect the customer's purchase habit. The dynamic attributes are then combined with the static attributes, that is, attributes that do not change and reflect some basic states of the customers, such as the customer's name, gender, and many more. It is developed a hybrid model of segmentation with clustering algorithm and ANNs procedures. The clustering algorithm identifies customers' purchase history, and ANN is used to segment customers based on their credit classification.

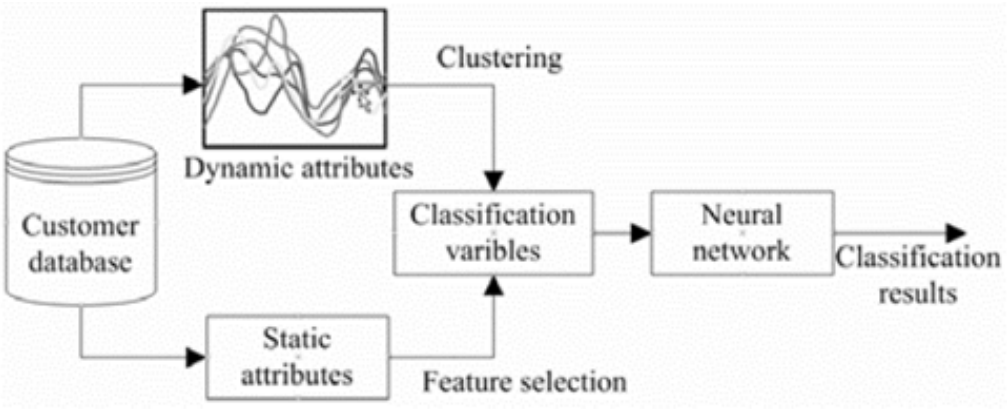

Figure 2.18: General approach of customer segmentation proposed by (Yan and Li, 2006)

Table 2.2 describes the summarized features implemented in the models proposed by

Table 2.2: Description of the features implemented in the models proposed by the papers listed in Section 2.5.2.

Paper	**Features implemented by the models**
Jonker et al. (Jonker et al., 2004)	Socio-demographic data, the purchase amount of a specific period, number of mailings without response since the last response, the response percentage over the last two years, and over the period of registration.
Chan et al. (Chan, 2008)	Demographic data, campaign programs, customer transaction data: the latest purchase amount, the total number of purchases over a specific period, and the monetary value spent
Christy et al. (Christy et al., 2021)	The number of days a customer takes between two purchases, the number of purchases, and the amount of money spent
Carmein et al. (Carnein and Trautmann, 2019)	Customer's purchase history.
Yan et al. (Yan and Li, 2006)	Customer transaction data: customer purchase amount, the purchase frequency, and customer profile information

each paper mentioned previously.

Nivea Sun uses demographic segmentation to aggregate their customers and analyze their behavior to create a good product range. As a result, it allows the brand to provide more value to its customers (Sharma, 2020)

Porsche offers different kinds of product mixes based on the geographic region they cater to. For example, the brand offers a higher percentage of convertibles in their product mix to consumers from the warmer south or southwest of the United States than the north (Sharma, 2020)

Amazon targets their customers based on their recent purchases and recently viewed products, this constitutes behavioral segmentation (Sharma, 2020).

2.6 Summary

This chapter reviewed the literature on predicting the best time to send marketing communications to subscribers, and in subscriber segmentation around four main areas: AI, ML, AIM, and Deep Learning. What emerged from the evaluation of the literature is that the automation technology in AI, driven by the ML algorithms in Marketing aids companies to deliver personalized communications and estimate the best period of time to send them. Deep learning algorithms extend even further those personalized experiences.

The predictive models presented in section 2.5.1 are based on different features, however, some are common to each predictive model. Those features include the open rates, the number of emails delivered (and sent), the clickthrough rates, and the profile informa-

tion of the subscribers.

Some predictive models approached predicting the time to send marketing communications as a classification task, implementing Random Forest, Decision Tree, and Support Vector Machine algorithms. Other predictive models approached it as a regression task, using algorithms such as Cox Proportional Hazard Regression. Some predictive models were approached with deep learning models using Hidden Markov, Recurrent Neural Networks (RNNs), and Feed-Forward Networks (FNNs), mixed with survival analysis methods and the Cox Proportional Hazard Regression algorithm.

Two predictive models implemented sequential ensemble methods with the boosting technique. The algorithms used were XGBoost and Gradient Tree Boosting.

Additionally, a different communication channel was presented, mobile messaging (SMS) recurring to Rabbit MQ Protocol to send and receive communications from the subscribers/consumers.

The predictive models presented in section 2.5.2 are based on RFM and LTV segmentation techniques. The subscribers' profile information and data associated with purchases are used in the segmentation techniques. It was presented alternatives to the traditional clustering algorithm used for the segmentation (the K-Means algorithm), with derivated clustering algorithms that improve segmentation performance, demanding less computational resources and less training.

Chapter 3

SFM: Send Frequency Predictive Model

The first section of this chapter describes the applicability (or the purpose) of the Send Frequency Predictive Model (SFM) for the company and the business model that the client who provided the dataset applies daily in its sale processes.

The second section introduces the SFM architecture and pipeline created to meet the objectives listed in section 1.3.

The last section describes the Feature Engineering phase proposed in the architecture of the SFM model, where preprocessing techniques and methods are applied to the dataset.

3.1 Applicability

This thesis was tailored to assist the company's need to have a system that predicts the optimal frequency of multichannel communications and campaigns suited to the individual profile of each subscriber.

The company-client that provides the dataset follows a business-to-business (B2B) model. It is a business model where the end customer is another company rather than an individual consumer. In other words, it is a business model where the businesses are conducted between companies. This business stands in contrast to B2C business model (process of selling products and services directly between a business and a consumer who is the end-user of the products and services).

The SFM model intends to be applied to scenarios where companies need to determine the time to send their marketing communications to their subscribers and apply a B2B business model.

3.2 Architecture

The data exchanged between the subscribers and companies is usually stored in a database. A database is a collection of related fields that offer the advantage of powerful search facilities which can be used to locate and retrieve information many times faster than by manual methods. The databases used in a company are usually accessed by many different users across a network system (Hardcastle, 2020).

The first phase of the architectural pipeline (Figure 3.1) starts with extracting the data from one of the company's databases. The second phase of the architecture, denominated *"Data Transformation"*, follows with a preprocessing of the data to suit the problem's context. Section 3.4 explains the methods and techniques of Feature Engineering explained in section 2.3 applied to the dataset.

The *Dataset Split* phase splits the dataset into two subsets using the approach *train-test split*. During this phase, subscriber segmentation is implemented using a traditional clustering algorithm. The segmentation of the subscribers is explained in detail in section 4.2.

The clustering algorithm results are combined in a parallel ensemble approach (Figure 3.1). The training of the regression algorithms is implemented through parallel ensemble approaches (Figure 3.2) such as simple averaging and stacking techniques.

Instead of using classification algorithms proposed in several predictive models in section 2.5.1, it is proposed to use regression algorithms to determine and predict the best period of time to send marketing communications. It is not intended to classify an event as opening neither to associate it with an hour and day, but to determine the best period of time of the event.

A deep learning algorithm is also trained independently, using recurrent neural networks (RNNs). Similar to the article (Singh et al., 2020), the use of the RNN is proposed. With this approach, it is possible to store all the historical and temporal information of the emails and their timing, thus, performing temporal modeling of the information for better results, performance and processing. The proposed regression algorithms consist of the following:

- Random Forest Regressor (RF);
- Multiple Linear Regression;
- K-Neighbors Regressor (KNN);
- Support Vector Regressor (SVR);
- Recurrent Neural Networks (RNNs).

Given that organizations define and use KPIs to evaluate the success of marketing campaigns, it is necessary to categorize which KPIs are relevant to the proposed problem's

context. Therefore, in the context of email marketing, the success is evaluated by open rates, clickthrough rates and subscription fees (Monitor, 2020). The implemented system uses the two first metrics (open rates and clickthrough) as subscriber action.

Figure 3.3 illustrates the simple averaging method on the trained regression algorithms.

The regression algorithms training phase implementation is explained in section 4.3. The test phase and the evaluation is explained in section 5.2 and in section 5.3.

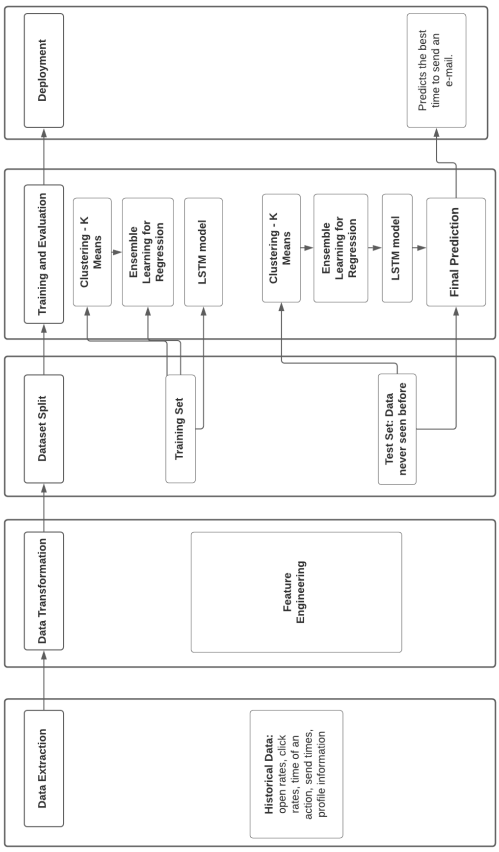

Figure 3.1: The architecture proposed for the SFM

3.3 Dataset Analysis

The provided dataset has twenty-three features, where each feature has different relevance to the domain of the problem.

Features are a specific representation of raw data, individual and measurable attributes typically described by a column in a dataset. A generic dataset consists of rows with each observation and columns with each feature containing a specific value for the observations.

The twenty-three features mentioned before are the following:

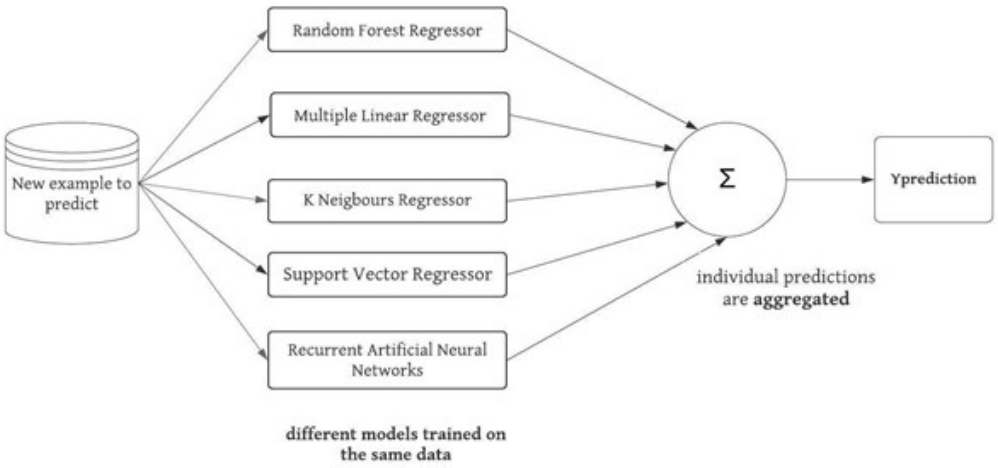

Figure 3.2: Parallel ensemble approach for the SFM

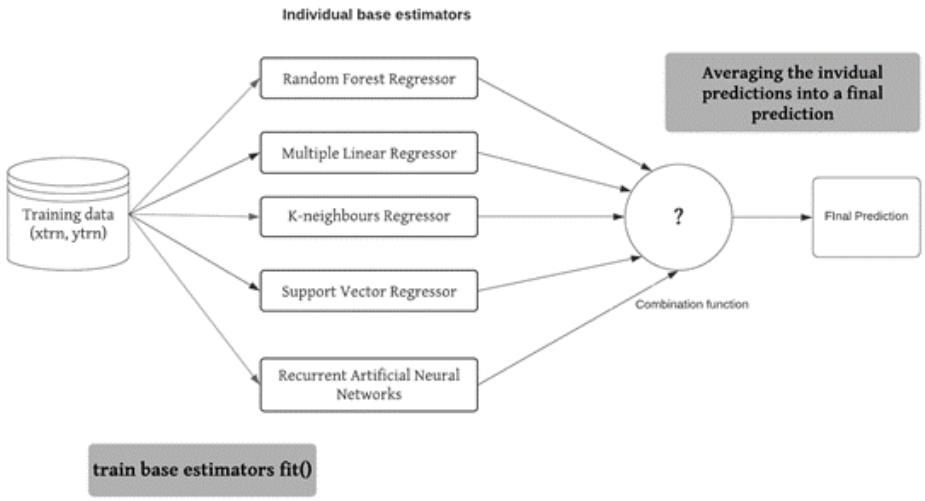

Figure 3.3: Parallel ensemble approach for the training phase

- **uid**: subscriber identification
- **action**: an action the subscriber will take (e.g., opening or click on an email)
- **campaign**: campaign identification
- **time**: timestamp of an action (by default, sending an email, opening or clicking on an email)
- **sendTimes**: time of when an email communication is sent to a subscriber (measured as timestamp)
- **timeAction**: time at which an action occurs (measured as timestamp)

- **emailDomain**: domain of the email (e.g., hotmail, gmail)
- **city**: city associated with the subscriber's location
- **region**: region associated with the subscriber's location
- **country**: country associated with the subscriber's location
- **ops**: operating system used by the subscriber (e.g., Windows, MacOs, etc.)
- **equip**: equipment used by the subscriber (e.g., smartphone, computer, tablet)
- **weekDaySendTimes**: day of the week an email campaign is sent to a subscriber
- **weekDayTimeAction**: day of the week an action occurred
- **yearSendTimes**: year an email campaign is sent to a subscriber
- **yearTimeAction**: year an action of a subscriber occurred
- **monthSendTimes**: month an email campaign is sent to a subscriber
- **monthTimeAction**: month an action of a subscriber occurred
- **hourSendTimes**: hour an email campaign is sent to a subscriber
- **hourTimeAction**: hour an action of a subscriber occurred
- **minuteSendTimes**: minute an email campaign is sent to a subscriber
- **minuteTimeAction** minute an action of a subscriber occurred

The dataset has observations gathered during one year 2019 to 2020. Figure 3.4 demonstrates the distribution of the data through that year. The year 2020 demonstrates more email delivery than the previous year 2019.

Based on the observations of the Figure 3.5, it is possible to visualize that the highest email delivery occurs during Thursdays (day=3) followed by Fridays (day=4). Monday (day=0) has the least count of email delivery, and Sunday has no email delivery. These observations conclude that Thursdays and Fridays are the days where it is most likely to send marketing campaigns. Figure 3.6 shows that the subscriber action follows the tendency of the email delivery, that is, the subscriber open or click on emails as soon as they receive it.

Based on the observations of Figure 3.7, the email delivery is higher during the mornings (interval between 8h and 9h). Email delivery is also notable during the lunchtime (close to 12h), and afternoons (interval between 14h to 17h). There is little email delivery

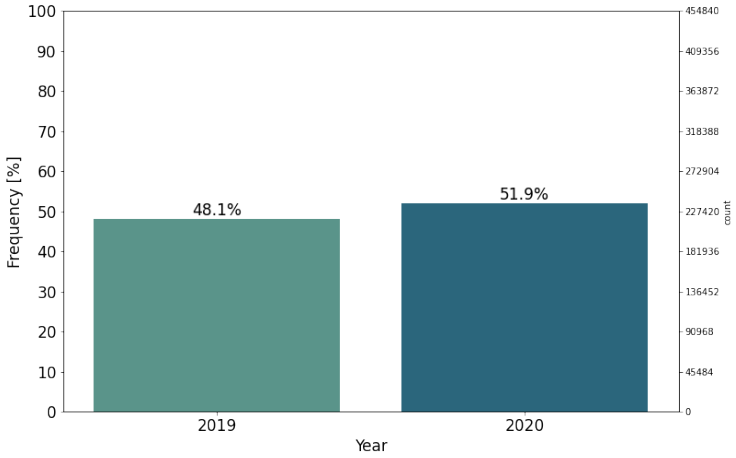

Figure 3.4: The distribution of the dataset through 2019 to 2020

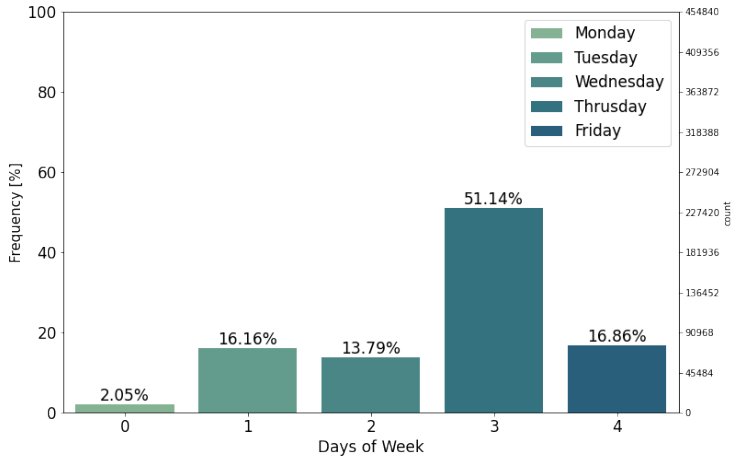

Figure 3.5: The days of the week with the highest email campaign delivery

during 10h to 11h, and around the night (18h to 21). These observations mean that the emails are more likely to be sent during the mornings.

Referring to the observations of Figure 3.8, most subscribers opened or clicked on emails during the mornings (mostly around 9h and 8h). There are subscribers that opened or clicked on emails during noon and others during the afternoon (interval of 14h to 17h). There are a few subscribers that opened around 13h, and during the morning period between 10h and 11h. A few subscribers opened and clicked on emails during the night (18h to 23h) and exceptionally, some subscribers opened and clicked on emails during the dawn (interval between 5 and 6h).

All observations on the figures presented, it is possible to conclude that most of the subscribers opened and clicked on emails as soon as they received the email campaigns,

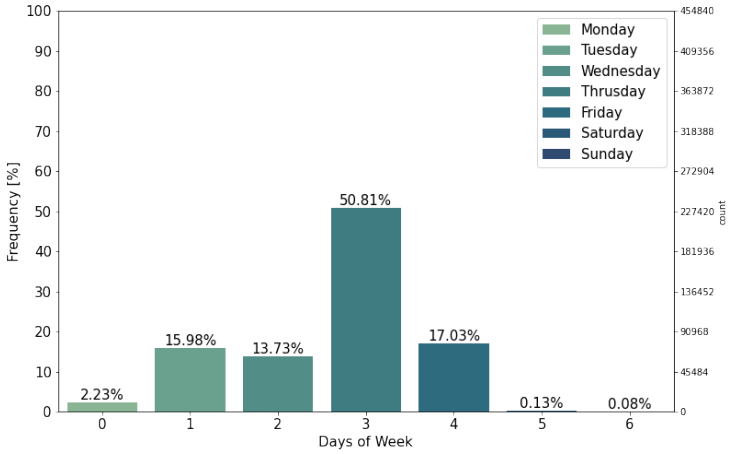

Figure 3.6: The days of the week with the highest subscriber action

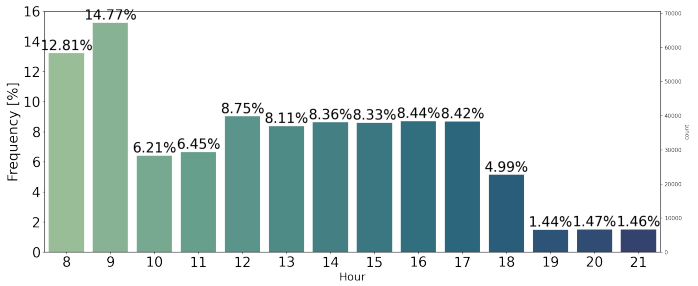

Figure 3.7: The hours of the day with the highest email campaign delivery

or one hour later. These behaviors can be explained by the fact that subscribers are more active during these periods of time (e.g., starting to work, waking up). Additionally, one can deduce that the pandemic changed some subscribers' behavior, leading them to be more active online during certain periods that previously were not.

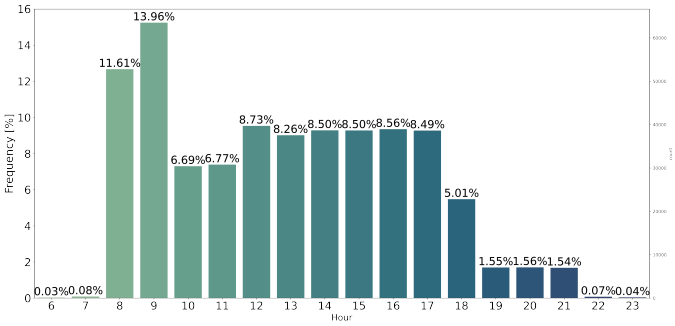

Figure 3.8: The hours of the day with the highest subscriber action

3.4 Data Transformation

This section explains the methods and the techniques of Feature Engineering explained in section 2.3 applied to the dataset used in the context of the problem.

3.4.1 Feature Selection

The first step of preprocessing involved the visualization of the features present in the dataset and the comprehension of their importance and relevance to the domain of the problem.

A correlation coefficient between the features was computed to understand the relevance and the linearity between each feature.

Additionally, a few observations of the remaining features with less than ten occurrences did not contribute to the predictive performance since the probability of those subscribers subscribing to the email campaigns is small.

3.4.2 Missing Data

In the preprocessing of the dataset, a function was computed to verify the absence of missing values in each feature. Then, it was computed the percentage of the missing values (Table 3.1). There were five features where 99% of the observations contained missing values.

One of the most common techniques to handle missing data is to use the function *fillna()* provided by the library Pandas. The missing values in the feature *campaign* and feature *destination* were filled with zero.

New features based on the existing features were subsequently created after this treatment, such as date features, explained in the next section

Table 3.1: The percentage of missing values in the identified features

Feature	Percentage
campaign	21.21%
destination	60.32 %
os	99.10%
country	99.13%
region	99.47%
city	99.54%

3.4.3 Date Features

Typically, these variables are seen as *DateTime* variables and contain many different labels corresponding to a specific combination of date and time. Feeding the ML models

these variables in their raw format is not recommended, so converting these features into multiple features is necessary.

Based on the information, the dates associated with the sending and the action (e.g., opening or clicking on an email) were converted to the *DateTime* format, and consequently diverged into several columns representing each property of the data such as the day of the week and hour.

In addition, a few outliers that were created regarding the dates were verified, with the dates associated with zero handled (by dropping them).

3.4.4 Categorical Features

In order to transform the categorical features in the dataset to numerical features, it was implemented the class *OrdinalEncoding()* from the *scikit-library*.

Figure 3.9 demonstrates the process of converting the categorical values associated with the feature *country* into numerical values.

```
ord_enc1 = OrdinalEncoder()
df['country'] = ord_enc1.fit_transform(df[['pais']].astype(str))
df[['pais', 'country']].head(11)
```

Figure 3.9: The conversion from categorical values in feature *country* to numerical values

3.4.5 Train-test Split

The implementation of the *train-test split* approach through the function *train-test split* of the *scikit-learn library* was implemented. In this approach, 80% of the dataset consisted was to train data and 20% was to evaluate and test the data.

The dataset is transformed into a multi-regression approach that includes features that determine the day of the week and the hour a marketing communication should be sent to a subscriber.

3.5 Summary

This chapter explains the applicability of the SFM system and the architecture designed to assist in the prediction of the best period of time to send marketing communications.

The architecture comprises of six phases (or *"pipelines"*), where the first phase consists of data extraction. The second phase consists of the transformation of the data extracted from the previous phase. The third phase allows the split of the dataset into two more phases called training and testing phase after the data transformation is executed.

The final phase consists of the deployment of the ML models. The first phase consists of the extraction of the data from one of the databases from the company.

This chapter describes the second and third phase in detail, explaining the feature engineering concepts and methods applied on the extracted dataset regarding the preprocessing and transformation of the data, and subsequently the method applied to split the dataset.

This chapter, additionally, provides an analysis of the dataset used in the context of the problem.

The next chapter explains, in detail, the practical implementation of the models during the training phase, explaining the approaches designed to find the solution to the proposed problem.

Chapter 4

SFM: Implementation

This chapter describes the implementation of several approaches proposed in the architectural pipeline in section 3.2 for subscriber segmentation and in predicting the best period of time to send marketing communications.

The first section of the chapter describes the experimental environment and the machine's physical specifications where the models were trained. The second section of the chapter describes the implementation of subscriber segmentation. The third section of the chapter describes the different approaches associated with predicting the optimal time to send marketing communications.

4.1 Environment Specifications

The environment where the experiments are trained consists of a server-client application called Jupyter notebook (Jupyter, 2020). It consists of an open-source web application that allows creating and sharing documents with code, equations, visualizations, and narrative text.

One of the most important factors of the environment is that it has enough resources to run the processes needed to apply the algorithms. Table 4.1 explains the specifications of the physical machine where the models were trained.

Table 4.1: Specifications of the physical machine used to run the experiments in the SFM

Specifications	**Values**
CPU	Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz
Cores	4
OS	Windows 10
Total Memory	16.0 GB

4.2 Subscriber Segmentation

The implementation of the subscriber segmentation is executed using the K-Means algorithm, which allows the subscribers to be segmented based on specific characteristics.

Based on the features provided in the dataset, the features used to segment subscribers (represented in Table 4.2) were based on the email domain of the subscriber and the operating system, which allowed to create the feature equipment (it allows to understand the type of device the subscribers use). In addition, demographic data associated with the subscriber was incorporated into the segmentation process.

Table 4.2: The input features for the subscriber segmentation

Input attributes	**Description**
Equipment (*equip*)	Smartphone, Computer, Tablet
Email Domain (*emailDomain*)	Sapo, Hotmail, Outlook, Gmail, etc...
Operating System (*ops*)	Linux, Unbuntu, Android, macOS, Windows, iOS
Demographic data	country, city, region

The visualization of the optimal number of clusters to segment becomes complicated as soon as the number of features increases. In order to resolve this situation, it is used a mathematical approach called the Elbow Method.

Elbow Method graphs the relationship between the number of clusters and the within-cluster sum of squares (WCSS). The WCSS consists on the sum of squared distance between each point and the centroid in the cluster.

The optimal number of clusters to segment the data is chosen where/when the WCSS begins to level off. Figure 4.1 demonstrates the optimal number of segments, determining two segments. The optimal number of segments means subscribers are segmented into subscribers with no associated profile information (cluster 0) and subscribers with little or more profile information (cluster 1). A new and last feature called *label* is created based on the number of segments and therefore added to the predictive models.

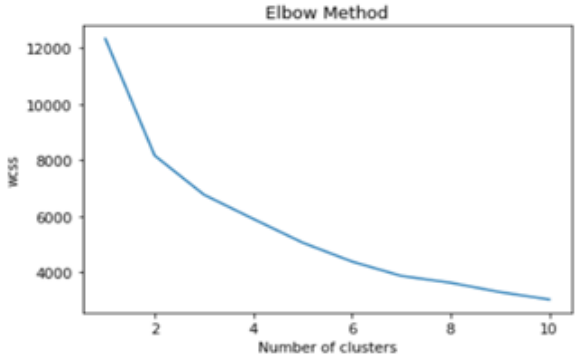

Figure 4.1: Elbow Method applied to the SFM segmentation process

To avoid randomness in the algorithm, K-Means initializes with a better optimized and smarter algorithm embedded in the K-Means pipeline called K-Means++ (Kumar, 2020). It uses the probabilistic distribution to pick the initial centroids, which increases the model's clustering performance.

4.3 Predicting the Best Time to Send Marketing Communications

The first approach involves creating a baseline hybrid ensemble of ML algorithms – Random Forest Regressor (RF), Linear Regression, K-Neighbours Regressor (KNN), and Support Vector Regressor (SVR) as base estimators.

4.3.1 Approach I: Hybrid ML Ensemble

Before ensembling the base estimators, it is essential to perform experiments individually on each model since the effectiveness of the ensemble relies on the performance and the predictions of each model.

One technique can be applied when dealing with the heterogeneous ensemble. The first technique consists of weighting methods that assign individual estimator predictions a weight corresponding to their strength. The weighting method used is simple averaging (Figure 4.2 demonstrates the training process), and the results of this method constitute the benchmark for other experiments executed.

The base estimators are trained using the provided dataset adapted to the context of the problem. After the training process, the ensemble returns the final predictions by weighting the base estimators' predictions. The ensemble becomes a better choice since it generalizes better when merging more base estimators. It also reduces the variance since it *"smoothes"* mistakes made by the base estimators and replaces them with the "wisdom of the ensemble".

Another technique can be implemented to resolve the situation where each base estimator contributes equally to the ensemble.

The second technique implements meta-learning methods with the most common meta-learning method called stacking (Figure 4.3 illustrates the stacking training). Stacking is a meta-learning algorithm with two levels of learning. The first level of learning consists of models that fit the data, and the second level consists of the meta-learner model, which takes the previous models' predictions and applies its own. The meta-learner algorithm used in the context of the proposed problem is Linear Regression.

```
base_estimators = [("rf", base_model_rf),("ml", base_model_lr), ("knn", base_model_knn), ("svr", SVR())]
model_base = MultiOutputRegressor(VotingRegressSor(estimators=base_estimators)).fit(X_train, y_train)
```

Figure 4.2: Simple Averaging Training Phase

```
base_stack = [("rf", base_model_rf), ("ml", base_model_lr), ("knn", base_model_knn), ("svr", SVR())]
model_stack_base = MultiOutputRegressor(StackingRegressor(estimators=base_stack, final_estimator=LinearRegression()))
model_stack_base.fit(X_train, y_train)
```

Figure 4.3: Stacking Training Phase

4.3.2 Approach II: Hybrid ML and DL Ensemble

The second approach follows the methods mentioned in the previous approach, however, the difference is that a fifth estimator is added to the ensemble. The fifth estimator is a deep learning model called RNN.

In terms of practical implementations, it is possible to implement Artificial Neural Networks (ANNs) or any model related to the neural networks (e.g., perceptron) as a base estimator in a ML heterogeneous ensemble, as long as the model requires the same dimensional input data to train models.

RNNs are examples of networks with tailored architectures, which means their architectures are specific to the domain they are inserted. Due to this uniqueness, the data is processed differently from the other *"simpler"* neural networks models.

RNNs require three-dimensional input data and follow the three-dimensional matrix rule [sample, timestep, feature]. Each row represents the number of samples, the number of time steps, and the number of features (Figure 4.4 demonstrates the conversion from two-dimensional input to three-dimensional input).

```
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
```

```
train_X.shape, train_y.shape, test_X.shape, test_y.shape
```

```
((21979, 1, 16), (21979, 2), (5495, 1, 16), (5495, 2))
```

Figure 4.4: The three-dimensional data required to train RNN

This data handling means that the dataset of the previous experiment needs to be trained and processed by the RNNs, and, subsequently, the long-short-term memory (LSTMs) models.

Unfortunately, it generates incompatibility with the ML models since they require two-dimensional arrays to process the input data.

Due to this incompatibility, it is impossible to aggregate an ensemble where any RNN model acts as a base estimator. In contrast, the result of the base estimators requires different dimensional arrays.

As a result, it is also impossible to apply any heterogeneous ensemble method associated with the library scikit-learn or the library *mlxtend* (Python library that contains ML extension, it is a derivation of the scikit-learn library).

Despite this reality, it is possible to average each base estimators' predictions by calling the function *mean()* from the *NumPy* library, a mathematical library, or manually summing all the predictions and dividing them by the number of base estimators.

4.3.3 Approach III: RNN model

The third and last approach implements more complex Recurrent Neural Networks (RNNs) called Long-Term-Short-Term Networks (LSTMs) (Figure 4.5).

RNNs suffer from the vanishing gradient problem since the gradient decreases as it backpropagates through time. It means that if a gradient value becomes extremely small, it cannot contribute to the network learning process, and as the depth of the RNNs increases, the gradient can either vanish or explode.

The vanishing gradient is the process that enables the ability to recall information from the past. To overcome the vanishing gradient and exploding gradient problem, networks capable of learning and avoiding long-term dependencies were embedded in the RNNs.

LSTMs (Hochreiter and Schmidhuber, 1997) are the networks capable of avoiding long-term dependencies by remembering information for longer intervals.

```
model = Sequential()

model.add(LSTM(10, activation='relu', input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dropout(0.2))

model.add(Dense(2, activation='linear'))
opt = Adam(learning_rate=0.001)
model.compile(loss='mean_squared_error', optimizer=opt, metrics='mae')

early_stop = EarlyStopping(monitor='val_loss',patience=10)
history = model.fit(train_X, train_y, epochs=30, batch_size=72, shuffle=True, validation_data=(test_X, test_y), verbose=2,
```

Figure 4.5: The Long-Term-Short Memory (LSTM) training phase

Their architectures are similar to the simple RNNs, however, their repeating modules have different components. The main difference between the two networks is the gated cells and their states included. Figure 4.6 illustrates the difference between their architectures.

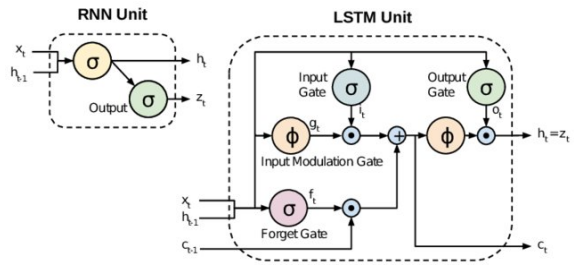

Figure 4.6: Simple RNN's architecture and LSTM's architecture (Rassem et al., 2017)

4.4 Summary

This chapter explains the practical implementations developed in the architecture's fourth pipeline, the training phase. The first step in the training pipeline involves subscriber segmentation. Through a clustering algorithm (K-Means), it is possible to segment the subscribers based on their profile information. The subscribers were segmented based on the email domain, the operating system, the equipment (which allows understanding the type of device the subscriber uses) and demographic data. The subscribers are then divided into subscribers with two clusters – subscribers with no profile information and subscribers with little or more profile information.

The second step in the training pipeline involves predicting the best period of time to send marketing communications, divided into three approaches. The first approach starts with the training of each of the ML models before they are ensembled. Successively, parallel ensemble methods such as simple averaging and stacking are trained. The second approach follows the same principle of the previous approach, however, it is added a deep learning model to the ensemble. The third and last approach consists of the training of the LSTM model.

In addition, this chapter presents the experimental environment and the machine's physical specifications where the models were trained.

The next chapter describes the evaluation phase to assess the behavior of the models and determine the optimal model(s) to predict the best period of time.

Chapter 5

SFM: System Evaluation and Analysis

This chapter describes the evaluation and analysis of several models implemented in the previous chapter to predict the best period of time to send marketing communications.

The first section of the chapter describes the evaluation system used to measure the performance and determine the SFM system's prediction accuracy. The second section analyzes and evaluates the results of the first experience that implements the *train-test split*. The last section examines and evaluates the second experience results from testing unseen data on the models in order to understand how they behave.

5.1 Temporal Classification System

A temporal classification system was implemented in order to evaluate the accuracy of the values of the predictions. The system associates interval of times to several phases of the day such as i) morning, ii) lunchtime, iii) afternoon, iv) evening, and v) dawn. Table 5.1 details the hours associated with each daily phase.

Table 5.1: The temporal classification system designed for the predicted time (hour) to send marketing communications to subscribers

Period	**Interval of Hours**
Morning	8h to 12h
Lunchtime	12h to 14h
Afternoon	14h to 18h
Evening	18h to 23h
Dawn	23h to 8h

5.2 First Experiment: Dataset Split

In the first phase of the project, there was only one dataset to test and validate the models; therefore, there was a need to split the former into smaller parts of data that could be

trained, tested, and validated.

The validation set in this project was implemented using cross-validation with hyper-parameter tuning methods (saranya Mandava, 2018). This set allows to validate the value of certain parameters of the training algorithms to choose the correct parameters that best fit the models.

5.2.1 Approach I: Hybrid ML Ensemble

The individual models trained in section 4.3.1 need to be evaluated and tested individually first before being grouped together into an ensemble since the performance of each model contributes to the final performance of the ensemble. This section evaluates each individual ML model before they are ensembled.

5.2.1.1 Approach I: Baseline Individual ML Models

Based on the temporal classification system (illustrated by Table 5.1), and on the results from the predictions by the RF model (presented by Table 5.2), a marketer can send communications to a specific subscriber (e.g., the first subscriber of the Table 5.2) on a Tuesday during the morning. The second subscriber can receive communications on a Tuesday during the afternoon (the value 0.71, is rounded leading to 1), and so on.

The values of the predictions for the first model (RF) are obtained by implementing the function *predict* from the *scikit-learn* library (Figure 5.1).

Table 5.2: Comparison of the predicted values by the RF model to the expected values

Actual Values		**Expected Values**	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	1.00	8.00
0	16	0.71	15.03
3	12	3.00	12.00
1	9	1.00	9.00
1	10	1.00	10.70
3	12	3.00	12.01

```
preds = base_model_rf.predict(X_test)
```

Figure 5.1: The implementation of the predictions for the RF model

Table 5.3 demonstrates the first five values predicted by the linear regression model compared to the actual values. Figure 5.2 demonstrates the implementation of the predictions for the second model.

Table 5.3: Comparison of the predicted values by the linear regression model to the expected values

Actual Values		Expected Values	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	1.31	10.95
0	16	2.13	11.57
3	12	3.01	12.49
1	9	1.33	10.76
1	10	2.21	11.02
3	12	3.06	12.87

```
preds_lr = base_model_lr.predict(X_test)
```

Figure 5.2: The implementation of the predictions for the linear regression model

Table 5.4 demonstrates the first five values predicted by the KNN model compared to the actual values. Figure 5.3 demonstrates the implementation of the predictions for the third model.

Table 5.4: Comparison of the predicted values by the KNN model to the expected values

Actual Values		Expected Values	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	1.00	8.00
0	16	0.00	15.60
3	12	3.00	12.00
1	9	1.00	9.00
1	10	1.80	15.80
3	12	3.00	12.00

```
preds_knn = base_model_knn.predict(X_test)
```

Figure 5.3: The implementation of the predictions for the KNN model

Table 5.5 demonstrates the first predicted values by the SVR model compared to the expected values. Figure 5.4 illustrates the implementation of the predictions for the fourth model.

Based on the values of the metrics summarized in Table 5.6, the model RF explains 83.9% of the variance of the target variable or the linear dependency between the independent variable and the target variable. The model RF has an average absolute difference

Table 5.5: Comparison of the predicted values by the SVR model to the expected values

Actual Values		Expected Values	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	2.85	10.73
0	16	2.78	10.58
3	12	3.02	11.13
1	9	2.85	10.73
1	10	2.77	10.55
3	12	2.86	10.77

```
preds_svr = base_model_svr.predict(X_test)
preds_svr
```

Figure 5.4: The implementation of the predictions for the SVR model

(MAE) between the predictions and the actual values of 0.330, and the mean squared error (MSE) of 1.579. The predictions are wrong by 7% and 4%, respectively, in predicting the day of the week and the hour.

The model linear regression explains 31.2% of the variance of the target variable, with an average absolute difference between the predictions and the actual values is 1.621 and the average squared error of 6.748. The predictions are wrong by 26% and 23% in predicting the day and the hour, respectively.

The model KNN explains 86.5% of the variance of the target variable. The model obtains an average absolute difference between the predictions and the expected (actual) values of 0.278 and the mean squared error of 1.482. The predictions are wrong by 23% in predicting the day of the week. The *inf* value means that there are present zero values, and the division with zero values leads to infinity. The values present in the dataset represent the first day of the week, which is Monday, hence the result.

The model SVR explains negatively 4.8% of the variance of the target variable. The model obtains an average absolute difference between the predictions and the expected values of 2.052 while the mean squared error is 8.786. The predictions are wrong by 37% and 28%, respectively, in predicting the day of the week and the hour.

5.2.1.2 Approach I: Baseline Ensemble Method

This section evaluates the results of the implementation of the parallel ensemble methods such as simple averaging and stacking from the approach in section 4.3.1. Table 5.7 presents the regression metrics applied to evaluate the first parallel ensemble method.

The simple averaging model explains 71.1% of the variance of the target variable. The average absolute difference (MAE) between the predictions and the actual values is 1.025.

Table 5.6: Regression metrics applied to evaluate the performance of each ML model

Metrics	RF	Linear Regression	KNN	SVR
RMSE	1.256	2.598	1.217	2.964
MSE	1.579	6.748	1.482	8.786
MAE	0.330	1.621	0.278	2.052
MAPE	0.07 & 0.04	0.26 & 0.23	inf & 0.04	0.37 & 0.28
R^2	0.839	0.312	0.865	-0.048

Table 5.7: Regression metrics applied to evaluate the simple averaging model

Metrics	Values
RMSE	1.697
MSE	2.881
MAE	1.025
MAPE	0.20 & 0.14
R^2	0.711

```
preds_sav_base = model_sav_base.predict(X_test)
```

Figure 5.5: The implementation of the predictions for the simple averaging model

The mean absolute error (MSE) is 2.881, and the predictions are wrong by 20% and 14%, respectively, in predicting the day and hour.

Figure 5.5 illustrates the implementation of the prediction for the simple averaging method.

Table 5.8 compares the first five predicted values by the implementation of the simple averaging to the actual values.

Table 5.8: Comparison of the predicted values by the linear regression model to the expected values

Actual Values		Expected Values	
Day of Week	Hour	Day of Week	Hour
1	8	1.540	9.421
0	16	1.232	13.190
3	12	3.009	11.905
1	9	1.546	9.875
1	10	2.198	12.067
3	12	2.981	11.984

In order to understand, in depth, the contribution of the model SVR to the ensemble

taking into account its bad prediction score, the error values, and the equal contribution of each base estimator to the ensemble, the model SVR was removed.

Table 5.9 evaluates the performance of the ensemble using the simple averaging method without the SVR model. Table 5.10 compares the five predicted values to the expected values.

Table 5.9: Regression metrics applied to evaluate the second simple averaging model

Metrics	**Values**
RMSE	1.413
MSE	1.997
MAE	0.713
MAPE	0.13 & 0.10
R^2	0.814

Table 5.10: Comparison of the predicted values by the second simple averaging model to the expected values

Actual Values		**Expected Values**	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	1.10	8.98
0	16	0.71	14.17
3	12	3.00	12.16
1	9	1.11	9.58
1	10	1.96	12.65
3	12	3.02	12.30

The second simple averaging model explains 81.4% of the variance of the target variable, showing an increase of 10.3% of the variance compared to the first simple averaging model. The average absolute difference (MAE) between the predictions and the actual values is 0.713, revealing a decrease in the difference of 31.2% compared to the first model. The predictions are wrong by 13% and 10%, showing an increase of 7% and 4% in the accuracy of the predictions compared to the first model. The mean absolute error (MSE) shows a decrease of 88.4% compared to the first model.

According to the results of both implementations, it is possible to conclude that the SVR model contributes negatively to the ensemble's predictions and the error scores.

The meta-learning approach appears to resolve this situation where each base estimator contributes equally to the ensemble.

Table 5.11 evaluates the performance of the stacking model. Table 5.12 compares the first predicted values to the expected values. Figure 5.6 illustrates the implementation of the prediction for the stacking method.

Table 5.11: Regression metrics applied to evaluate the stacking model

Metrics	Values
RMSE	1.165
MSE	1.357
MAE	0.292
MAPE	0.08 & 0.04
R^2	0.883

```
preds_stack_base = model_stack_base.predict(X_test)
```

Figure 5.6: The implementation of the predictions for the stacking model

The stacking model explains 88.3% of the variance of the target variable. The average absolute difference (MAE) between the predictions and the actual values is 0.292, and the mean absolute error (MSE) is 1.357. The predictions are wrong by 8% and 4%, respectively, in predicting the day and the hour.

Table 5.12: Comparison of the predicted values by the stacking model to the expected values

Actual Values		Expected Values	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	0.998	7.957
0	16	0.019	15.383
3	12	3.005	11.965
1	9	0.998	8.970
1	10	1.685	13.892
3	12	3.002	12.004

Comparing the previous results to the second simple averaging model, there was an increase of 6.9% of the variance. There was a decrease in the difference between the predictions and the actual values of 42.1%, meaning less than 42.1% of the error in the predictions. Additionally, the accuracy of the predictions increased 5% and 6%.

Based on the previous results, the stacking model is the optimal ensemble model to predict the best period of time to send marketing communications.

5.2.1.3 Approach I: Best Individual ML Models

This section describes the searching for the best parameters through hyperparameter optimization methods for the models in section 5.2.1.1.

Table 5.13 evaluates the performance of the RF model when RandomizedSearchCV is applied. Table 5.14 evaluates the performance of the RF model when GridSearchCV is applied. Figure 5.7 demonstrate the implementation for the RandomizedSearchCV. Figure 5.8 demonstrate the implementation for the GridSearchCV.

Table 5.13: Regression metrics applied to evaluate the RF model with the RandomizedSearchCV method

Metrics	**Values**
RMSE	1.254
MSE	1.572
MAE	0.328
MAPE	0.08 & 0.04
R^2	0.840

Table 5.14: Regression metrics applied to evaluate the RF model with the GridSearchCV method

Metrics	**Values**
RMSE	1.404
MSE	1.972
MAE	0.472
MAPE	0.11 & 0.06
R^2	0.797

GridSearchCV focuses on the promising hyperparameters (mentioned in Figure 5.8) range found in the method RandomizedSearchCV to obtain better results than the previous method.

According to the results demonstrated in Table 5.14, the GridSearchCV results are not better than the results illustrated in Table 5.13, therefore, the best model RF can be achieved by the implementation of RandomizedSearchCV.

Regularization shrinkage methods are applied when dealing with linear models to diminish coefficients and produce simple and effective methods. The two shrinkage methods applied are Ridge and Lasso. Their difference lies in the penalty term.

Ridge performs L2 regularization, i.e., adds penalty equivalent to the square of the magnitude of coefficients. In contrast,

Lasso performs L1 regularization, i.e., adds penalty equivalent to the absolute value of the magnitude of the coefficients.

Table 5.15 evaluates the performance of the linear regression model with the ridge implementation. Table 5.16 evaluates the performance of the linear regression model with the lasso implementation.

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
```

```
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

```
rf = RandomForestRegressor()
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 100, cv = 5, verbose=2)
rf_random.fit(X_train, y_train)
```

```
rf_random.best_params_
```

```
preds_rf_rand = rf_random.predict(X_test)
```

Figure 5.7: The RandomizedSearchCV implementation on RF model (Koehrsen, 2018)

```
param_grid = {
 'n_estimators': [1600],
 'min_samples_split': [5],
 'min_samples_leaf': [1],
 'max_features': ['auto'],
 'max_depth': [10],
 'bootstrap': [True]}
```

```
grid_rf = RandomForestRegressor()
rf_grid = GridSearchCV(estimator = grid_rf, param_grid = param_grid, cv = 5, verbose = 2)
rf_grid.fit(X_train, y_train)
```

```
preds_rf_grid = rf_grid.predict(X_test)
```

Figure 5.8: The GridSearchCV implementation on RF model

Table 5.15: Regression metrics applied to evaluate the linear regression model with the ridge implementation

Metrics	**Values**
RMSE	2.598
MSE	6.748
MAE	1.621
MAPE	0.26 & 0.23
R^2	0.312

Table 5.16: Regression metrics applied to evaluate the linear regression model with the lasso implementation

Metrics	Values
RMSE	2.689
MSE	7.231
MAE	1.934
MAPE	0.42 & 0.24
R^2	0.086

Figure 5.9 illustrates the implementation of the ridge method. Figure 5.10 illustrates the implementation of the lasso method.

```
lambdas=np.linspace(1,100,100)
param_grid={'alpha':lambdas}
```

```
model_ridge = Ridge(fit_intercept=True)
ridge = GridSearchCV(estimator = model_ridge, param_grid = param_grid, cv = 5, verbose = 2)
ridge.fit(X_train, y_train)
```

```
preds_ridge = ridge.predict(X_test)
```

Figure 5.9: The Ridge implementation on the linear regression model

```
lambdas=np.linspace(1,10,100)
model_lasso = Lasso(fit_intercept=True)
params={'alpha':lambdas}
```

```
lasso = GridSearchCV(estimator = model_lasso, param_grid = param_grid, cv = 5, verbose = 2)
lasso.fit(X_train, y_train)
```

```
preds_lasso = lasso.predict(X_test)
```

Figure 5.10: The Lasso implementation on the linear regression model

According to the results illustrated in table 5.15, the results from the Ridge model are better than the results from the Lasso model. Therefore, the ridge model can lead to an optimal regression model.

The Ridge model achieves 31.2% of the variance of the target variable, which is 22.6% more variance than the one obtained in the Lasso model. Additionally, the Ridge model has an average absolute difference between the predictions and the actual values of 1.621. It is 31.3% less in prediction error compared to the Lasso model. The predictions are wrong by 26% and 23% in both of the models.

Figure 5.11 illustrates the implementation of the GridSearchCV method on the KNN model.

Table 5.17: Regression metrics applied to evaluate the optimal KNN model

Metrics	Values
RMSE	1.082
MSE	1.171
MAE	0.166
MAPE	inf & inf
R^2	0.898

```
param_grid = {
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'leaf_size': [1, 50],
    'metric': ['minkowski', 'euclidean', 'manhattan'],
    'n_neighbors': [1, 50],
    'p': [1, 2],
    'weights':['uniform', 'distance']
}
```

```
knn = KNeighborsRegressor()
knn_random = RandomizedSearchCV(estimator = knn, param_distributions = param_grid, n_iter = 100, cv = 5, verbose=2,
knn_random.fit(X_train, y_train)
```

```
preds_knn_grid = knn_grid.predict(X_test)
```

Figure 5.11: The GridSearchCV implementation on the KNN model

The optimal KNN model explains 89.8% of the variance of the target variable. The average absolute difference between the predictions and the expected values is 0.166 while the mean absolute error is 1.171.

The values represented in the table 5.17 are the lowest values obtained from all the ML models evaluated and tested, concluding that the model KNN is the ML model with the best results to predict the best period of time to send marketing communications.

Due to the high demand for computational resources for the SVR model when training and testing, it was only implemented GridSearchCV on the SVR model.

Table 5.18 evaluates the performance of the SVR model with the GridSearchCV implementation. The results obtained demonstrate that the SVR model is the model with the worst results to predict the best period of time send communications.

Figure 5.12 illustrates the implementation of the GridSearchCV on the SVR model.

```
random_grid = {'C':[0.1,1],'kernel':['rbf']}
```

```
svr = SVR()
svr_grid = MultiOutputRegressor(GridSearchCV(estimator = svr, param_grid = random_grid, verbose = 2))
svr_grid.fit(X_train, y_train)
```

```
preds_svr_grid = svr_grid.predict(X_test)
```

Figure 5.12: The GridSearchCV implementation on the SVR model

Table 5.18: Regression metrics applied to evaluate the optimal SVR model

Metrics	Values
RMSE	2.950
MSE	8.700
MAE	2.056
MAPE	0.37 & 0.28
R^2	-0.042

5.2.1.4 Approach I: Best Ensemble Model

This section describes the searching for the best parameters through hyperparameter optimization methods for the models in section 5.2.1.2.

Table 5.19 evaluates the performance of the hyper-tuned models of the ensemble through the simple averaging. Table 5.20 compares the predicted values of the simple averaging model to the expected values.

Based on the results of table 5.19, the optimal simple averaging model explains 84.0% of the variance of the model. The average absolute difference between the predictions and the actual values is 0.675, and the mean absolute error is 1.719. The predictions are wrong by 13% and 10%, respectively, in predicting the day and the hour.

Table 5.19: Regression metrics applied to evaluate the optimal simple averaging model

Metrics	Values
RMSE	1.311
MSE	1.719
MAE	0.675
MAPE	0.13 & 0.10
R^2	0.840

Table 5.21 evaluates the performance of the hyper-tuned models of the ensemble through the stacking method. Table 5.22 compares the predicted values by the optimal model to the expected values.

The optimal stacking model explains 91.0% of the variance of the target variable.

The average absolute error difference between the predictions and the actual values is 0.204, and the mean absolute error is 1.051. This value is the lowest of all the ensemble methods applied and only 3.8% more than the KNN algorithm.

Table 5.20: Comparison of the predicted values by the optimal simple averaging model to the expected values

Actual Values		**Expected Values**	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	1.103	8.982
0	16	0.716	14.202
3	12	3.003	12.166
1	9	1.110	9.587
1	10	1.674	11.304
3	12	3.020	12.355

Table 5.21: Regression metrics applied to evaluate the optimal stacking model

Metrics	**Values**
RMSE	1.025
MSE	1.051
MAE	0.204
MAPE	0.08 & 0.03
R^2	0.91

The predictions are wrong by 8% and 4%, respectively, in predicting the best and the hour.

Based on the results, it is possible to conclude that the stacking model even with the hyperparameter optimization methods still remains the best optimal model to predict the best period of time to send marketing communications.

Table 5.22: Comparison of the predicted values by the optimal stacking model to the expected values

Actual Values		**Expected Values**	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	0.998	7.957
0	16	0.01	15.383
3	12	3.005	11.965
1	9	0.998	8.970
1	10	1.686	13.892
3	12	3.00	12.004

5.2.2 Approach II: Hybrid ML and DL Ensemble

The models trained in section 4.3.2 are evaluated and tested. This section presents the results from that evaluation and analysis.

5.2.2.1 Approach II: Baseline Hybrid ML and DL Ensemble Model

Table 5.23 compares the results of implementing the simple averaging model of the previous experiments to the results obtained by adding the LSTM model to the ensemble. Table 5.24 demonstrates the predicted values obtained with the proposed ML and DL ensemble in this section.

Table 5.23: Comparison between the results of the baseline hybrid ML ensemble to the results of the baseline hybrid ML & DL ensemble

Metrics	**ML Ensemble**	**ML & DL Ensemble**	**Observations**
RMSE	1.413	1.852	Increase of 43.9%
MSE	1.997	3.430	Increase of 143.3%
MAE	0.713	1.159	Increase of 46.4%
MAPE	0.13 & 0.10	0.22 & 0.15	Increase of 9 and 5%
R^2	0.814	0.640	Decrease of 17.4%

The baseline simple averaging model of the previous ensemble achieves 71.1% of the variance of the target variable. The baseline hybrid ML and DL ensemble achieves 64.4%

of the variance. The previous result shows a decrease of 17.4% of the variance of the target variable compared to the ML ensemble.

The simple averaging model of the ML ensemble has an average absolute difference (MAE) between the predictions and the actual values of 0.713. In contrast, the hybrid ML and DL ensemble has 1.159 as the average absolute difference. It means that the second ensemble has an increase of 44.6% in the difference between the predictions and the expected values, compared to the first ensemble. MSE is 1.997 for the first ensemble and 3.430 for the second, revealing an increase of 143.3% of the mean absolute error for the second ensemble.

The predictions are wrong 22% and 15%, respectively, for the day and the hour. These results mean that the accuracy of the predictions decreased by 9% and 5% compared to the first ensemble, due to the fact the LSTM model contributes little to the ML & DL ensemble.

Table 5.24: Comparison of the predicted values by the hybrid ML & DL ensemble model to the expected values

Actual Values		**Expected Values**	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	1.548	9.711
0	16	1.571	12.827
3	12	3.03	11.979
1	9	1.563	11.738
1	10	2.944	14.302
3	12	1.845	14.030

5.2.2.2 Approach II: Best Hybrid ML and DL Ensemble model

This section describes the searching for the best parameters through the hyper parameter optimization methods for the models in section 4.3.2.

Table 5.25 compares the performance of the hybrid hyper tuned ML ensemble using the best simple averaging implementation to the performance of the hybrid ML and DL hyper tuned ensemble. Table 5.26 demonstrates the predicted values for the best implementation of the simple averaging on the ensemble.

The best simple averaging model in the ML ensemble achieves 84.0% of the variance of the target variable while the implementation of the hyperarameter optimization in the ML and DL ensemble leads to 64.4% in variance. The results show a decrease of 17.4% of the variance for the second ensemble compared to the first ensemble.

The best simple averaging model in the ML ensemble achieves an average absolute difference between the predictions and the actual values of 0.675 while the ensemble ML

Table 5.25: Comparison between the results of the best hybrid ML ensemble to the results of the best hybrid ML & DL ensemble

Metrics	ML Ensemble	ML & DL Ensemble	Observations
RMSE	1.311	1.852	Increase of 54.1%
MSE	1.719	2.510	Increase of 79.1%
MAE	0.675	1.159	Increase of 48.4%
MAPE	0.13 & 0.10	0.22 & 0.15	Increase of 9 and 5%
R^2	0.840	0.64	Decrease of 17.4%

and DL obtains 1.159. This results mean that the second ensemble has an increase of 48.4% difference between the predictions and the expected values compared to the first ensemble.

The mean absolute error (MSE) is 1.719 for the first ensemble and 2.510 for the second, revealing an increase of 143.3% of the mean absolute error for the second ensemble.The predictions are wrong 22% and 15%, respectively, for the day and the hour. These results mean that the accuracy of the predictions decreased by 9% and 5% compared to the first ensemble.

Table 5.26: Comparison of the predicted values by the best hybrid ML and DL ensemble model to the expected values

Actual Values		Expected Values	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	1.603	9.660
0	16	1.575	12.929
3	12	2.957	11.939
1	9	1.596	9.997
1	10	1.903	11.089
3	12	2.931	12.131

5.2.3 Approach III: RNN Model

Time-series data consists of data that measures how things change over time, and in this approach, time is considered a primary axis. In addition, the additional dimensional features allow additional information to the predictions.

Due to this uniqueness (as shown in Figure 4.3 of Chapter 4), it is necessary specialized data manipulation in the datasets used. A univariate time series or single input forecasting is a series of data with one dependent variable. For example, when predicting the next energy consumption in a specific location, the prediction will be the time values as input and the historical energy as the output.

A multivariate series or multiple input forecasting has more than a one-time dependent variable where each variable depends on past values and has a dependency on other variables. For example, when predicting pollution levels for the next hour, the predictions will rely on the weather conditions (e.g., temperature, pressure, wind direction, wind speed) as input and the pollution levels from previous hours as output. Real-world time series forecasting is challenging for several reasons, and one of them is the presence of multiple input variables.

5.2.3.1 Approach III: Baseline RNN model

This section evaluates and analyzes the results from the section 4.3.3. Figure 5.13 illustrates the plot of the model loss through the training and test phase.

Table 5.27 evaluates the performance of the model when applying a baseline LSTM model. Table 5.28 compares the predicted values by the LSTM model to the expected values.

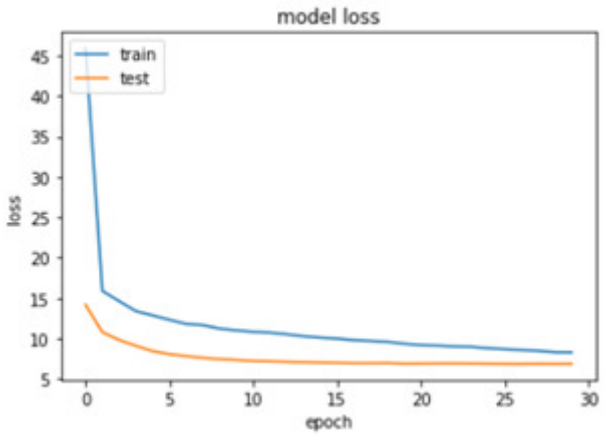

Figure 5.13: Behavior plot of the baseline LSTM model through the training and test phase

Table 5.27: Regression metrics applied to evaluate the baseline LSTM model

Metrics	Values
RMSE	2.614
MSE	6.831
MAE	1.702
MAPE	0.31 & 0.23
R^2	0.271

LSTM model achieves 27.1% of the variance of the target variable. The model has an average absolute difference (MAE) between the predictions and the actual values of

1.702. The mean absolute error (MSE) is 6.831. The predictions are wrong by 31% and 23%, respectively, in predicting the day of the week and the hour of the day.

Table 5.28: Comparison of the predicted values by the baseline LSTM model to the expected values

Actual Values		**Expected Values**	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	1.698	10.613
0	16	2.207	11.269
3	12	2.811	12.403
1	9	1.719	10.395
1	10	2.334	10.501
3	12	2.765	12.918

5.2.3.2 Approach III: Best RNN model

This section concludes the previous experiment by applying hyperparameter optimization.

In this approach, it was necessary to perform even more additional transformations than the initial ones when performing cross-validation. Cross-validation in deep learning models involves converting the Keras model from the Keras package to a *scikit-learn* model through the Keras package's wrappers.

Figure 5.14 demonstrates the Python code that reproduces the conversion from the Keras model to a *scikit-learn* model. It is also important to define the type of task the model is performing when dealing with the cross-validation transformation.

```
model = tensorflow.keras.wrappers.scikit_learn.KerasRegressor(build_fn=create_model,epochs=30,verbose=2
model._estimator_type = "regressor"
```

Figure 5.14: The conversion from the Keras model to a *scikit-learn* model

Figure 5.15 illustrates the plot of the LSTM model through the training and testing phase after being applied the hyperparameter optimization.

Table 5.29 evaluates the performance of the LSTM model after being applied the hyperparameter optimization. Table 5.30 demonstrates the first predicted values for the LSTM model.

The LSTM model with hyperparameter optimization achieves 25.5% of the variance of the target variable. The model has an average absolute difference (MAE) between the predictions and the actual values of 1.734 and the mean absolute error (MSE). The predictions are wrong by 33% and 23%, respectively, in predicting the day and the hour.

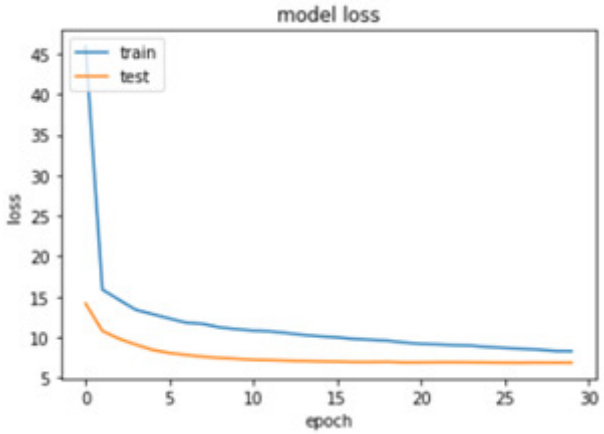

Figure 5.15: Behavior plot of the best LSTM model loss through the training and test phase

Table 5.29: Regression metrics applied to evaluate the optimal LSTM model

Metrics	**Values**
RMSE	2.624
MSE	6.887
MAE	1.732
MAPE	0.33 & 0.23
R^2	0.255

Table 5.30: The predicted by the hyper tuned LSTM model compared to the expected values

Actual Values		**Expected Values**	
Day of Week	**Hour**	**Day of Week**	**Hour**
1	8	1.854	10.463
0	16	2.384	11.172
3	12	2.749	12.270
1	9	1.795	10.333
1	10	2.285	10.492
3	12	2.734	12.885

5.3 Second Experience: Evaluation on New Data

This section explains the implementation of the previous approaches in unseen data. The dataset also has twenty-two features, with the same features mentioned in section 3.3.

The dataset has observations gathered during one year. Figure 5.16 demonstrates the distribution of the data through one year.

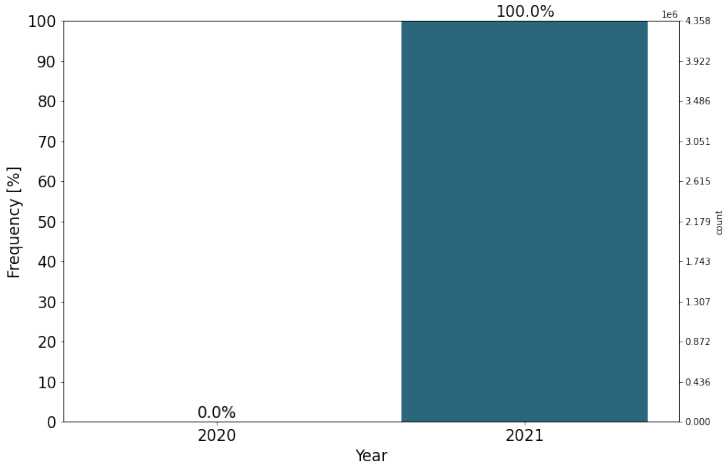

Figure 5.16: Distribution of the data through one year

Based on the observations of the Figure 5.17, the highest email delivery occurs during Tuesdays (day=1), followed by Mondays (day=0). Friday (day=4) has the least count of email delivery and Sunday has no email delivery. These observations conclude that Tuesdays and Mondays are the days more likely to send marketing campaigns.

The observations listed in Figure 5.18, show that the subscriber action follow the tendency, opening or clicking on the emails as soon as they receive it.

As shown in Figure 5.19 the email delivery is higher during the mornings (interval between 8h and 9h). Email delivery is also notable during the lunchtime and afternoons (interval of 14h to 17h). There is little email delivery during the interval of 10h to 11h and around the night (18h to 20h). These observations mean that the emails are more likely to be sent during the mornings.

Figure 5.20 show that some subscribers opened or clicked on emails during the mornings (mostly around 9h and 8h). There are subscribers that opened or clicked on emails during noon and others during the afternoon (interval of 14h to 17h). There are also some subscribers that opened or clicked on emails during the night (18h to 23h). Exceptionally, some subscribers opened or clicked on emails during the dawn (6 to 7h).

Based on the observations of all the figures presented, it is possible to conclude that most subscribers opened as soon as they received the email or one hour later. These behaviors can be explained by the fact that subscribers are more active during these periods

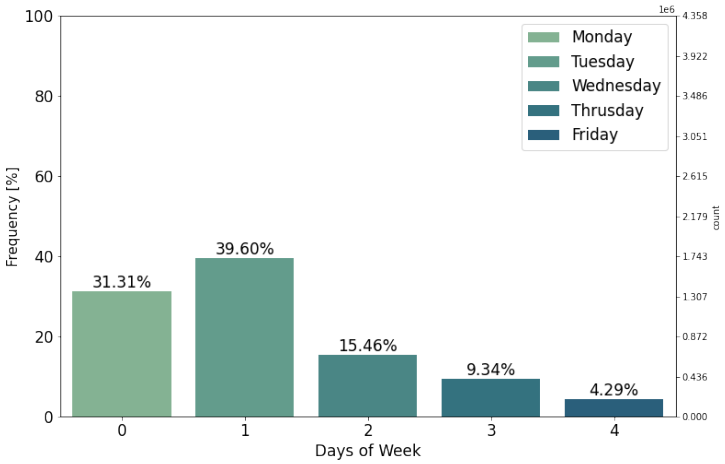

Figure 5.17: The day of the week with the highest email campaign delivery

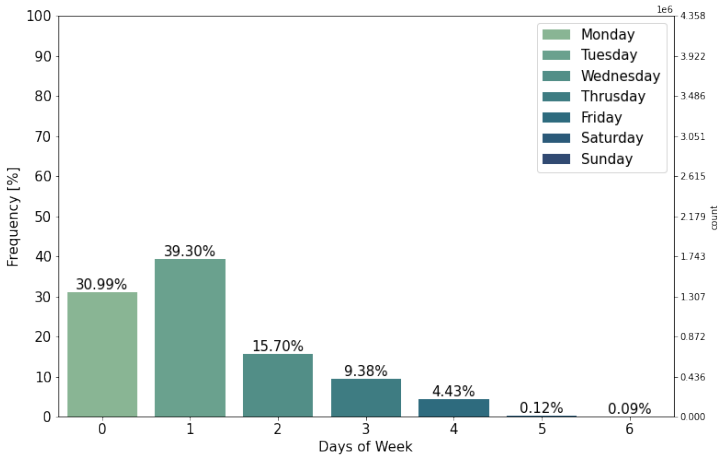

Figure 5.18: The day of the week with the highest subscriber action

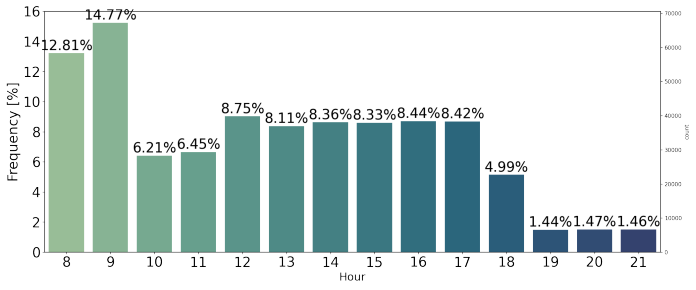

Figure 5.19: The hours of the day with the highest email campaign delivery

of time (e.g., starting to work, waking up). Additionally, one can deduce that the pan-

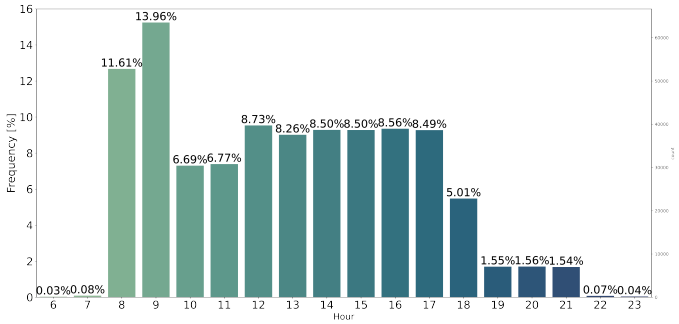

Figure 5.20: The hours of the day with the highest subscriber action

Table 5.31: Comparison of the predicted values obtained by the RF model when testing new data to the expected values

Actual Values		**Expected Values**	
Day of Week	**Hour**	**Day of Week**	**Hour**
3	17	2.30	16.02
0	10	1.88	15.82
0	12	1.73	15.77
0	13	2.17	16.70
0	13	2.41	14.22
0	8	2.38	15.42
3	15	4.38	20.10
0	15	2.64	14.49
1	14	3.31	16.91
1	19	2.36	15.36

demic changed some subscribers' behavior, leading them to be more active online during certain periods that previously were not.

5.3.1 Experiment I: Hybrid ML Ensemble

This section evaluates the models of the approaches explained in section 5.2.1.1 on unseen data. The approaches evaluated with the unseen data were implemented based on the premise that the models associated with the approach were the optimal models to predict the best period of time to send communications.

5.3.1.1 Experiment I: Baseline Hybrid ML Ensemble

Table 5.31 demonstrates the first predicted values for the first ML model of the ensemble compared to the expected values.

Table 5.32: Comparison of the predicted values obtained by the linear regression model when testing new data to the expected values

Actual Values		Expected Values	
Day of Week	**Hour**	**Day of Week**	**Hour**
3	17	1.42	12.59
0	10	0.66	11.65
0	12	0.78	12.35
0	13	0.78	13.16
0	13	0.82	13.15
0	8	0.49	9.80
3	15	3.03	12.66
0	15	0.64	11.27
1	14	0.76	14.82
1	19	1.40	12.42

Table 5.33: Comparison of the predicted values obtained by the KNN model when testing new data to the expected values

Actual Values		Expected Values	
Day of Week	**Hour**	**Day of Week**	**Hour**
3	17	5.0	21.4
0	10	5.0	21.4
0	12	5.0	21.4
0	13	5.0	21.4
0	13	5.0	21.4
0	8	5.0	21.4
3	15	5.0	21.4
0	15	5.0	21.4
1	14	5.0	21.4
1	19	5.0	21.4

Table 5.32 compares the first predicted values by the second model, the linear regression model, to the expected values.

Table 5.33 demonstrates the first predicted values by the third model, the KNN model, and the comparison to the actual values.

Table 5.34 demonstrates the first predicted values by the fourth model, the SVR model, and the comparison to the first actual values.

Based on the results listed in table 5.35, the best suitable model using the new data to predict the best period of time to send communication is the linear regression model, which obtains less error in the predictions (the lowest value of MAE) and the highest

Table 5.34: Comparison of the predicted values obtained by the SVR model when testing new data to the expected values

Actual Values		**Expected Values**	
Day of Week	**Hour**	**Day of Week**	**Hour**
3	17	2.67	10.31
0	10	2.63	10.22
0	12	2.67	10.33
0	13	2.67	10.33
0	13	2.66	10.31
0	8	2.63	10.22
3	15	2.68	10.35
0	15	2.68	10.22
1	14	2.67	10.32
1	19	2.66	10.30

Table 5.35: Regression metrics applied to evaluate the performance of each ML model with the new data

Metrics	**RF**	**Linear Regression**	**KNN**	**SVR**
RMSE	4.238	3.137	7.282	3.788
MSE	17.965	9.841	53.032	14.346
MAE	3.102	1.952	6.060	2.678
MAPE	0.58 & 0.29	0.59 & 0.27	0.70 & 0.40	0.58 & 0.37
R^2	-0.708	0.249	-5.231	-0.524

variance compared to the rest of the models. The SVR model follows next with the less value in the predictions error and afterwards the RF model. The worst model with this new data is the KNN with the highest prediction error value and the lowest variance.

Table 5.36 evaluates the performance of the baseline simple averaging implementation applied to the new data. Table 5.37 demonstrates the predicted values by the simple averaging implementation compared to the first ten actual values.

Implementing the simple averaging technique on the ensemble negatively achieves 73.3% of the variance of the target variable. On average, the difference between the predictions and the actual values is 3.163. The predictions are wrong by 59% and 29%, respectively, in predicting the day and the hour.

Table 5.38 evaluates the performance of the stacking implementation on the ensemble of the new data. Table 5.39 compares the first predicted values by the stacking implementation compared to the first ten values.

The stacking implementation on the new data negatively achieves 270.9% of the variance of the target variable. The average absolute difference between the predictions and

Table 5.36: Regression metrics applied to evaluate the simple averaging model

Metrics	Values
RMSE	4.040
MSE	16.324
MAE	3.163
MAPE	0.59 — 0.29
R^2	-0.733

Table 5.37: Comparison of the predicted values by the simple averaging model on the new data to the expected values

Actual Values		Expected Values	
Day of Week	**Hour**	**Day of Week**	**Hour**
3	17	3.16	16.36
0	10	2.46	16.11
0	12	2.50	14.66
0	13	2.81	15.75
0	13	2.56	15.20
0	8	2.43	15.76
3	15	3.92	15.74
0	15	2.79	16.17
1	14	2.62	16.15
1	19	3.04	16.35

Table 5.38: Regression metrics applied to evaluate the stacking model

Metrics	Values
RMSE	6.802
MSE	46.261
MAE	5.175
MAPE	0.62 — 0.39
R^2	-2.709

the actual values is 5.175. The predictions are wrong by 62% and 39%, respectively, in predicting the best day and hour. Comparing the results from tables 43 and 45, simple averaging achieves better predictions and fewer predictions error (lowest value of MAE) and

Table 5.39: Comparison of the predicted values by the stacking model to the expected values

Actual Values		Expected Values	
Day of Week	**Hour**	**Day of Week**	**Hour**
3	17	3.85	21.28
0	10	2.57	21.27
0	12	2.50	18.87
0	13	3.65	20.10
0	13	2.99	19.29
0	8	2.74	21.55
3	15	5.00	20.10
0	15	3.68	21.69
1	14	3.11	20.17
1	19	3.67	21.34

has a higher variance than the stacking implementation. Therefore, the simple averaging implementation using the new data is best suitable to predict the best day and hour.

5.4 Comparison Between models

Several methodologies were explored and assessed in this chapter in order to validate the potential of the SFM system suggested in this thesis.

To begin, a temporal categorization system was created to make sense of the predicted values in each technique that was later analyzed. The system was evaluated in two stages: the first using the *train-test split* approach and the second approach using previously unknown data (with an unseen dataset) for the models.

The first approach of the first experience was broken into four sub-approaches to aid predicting the optimal time to send marketing communications. This approach involved evaluating each ML model as a base estimator for the ensemble. The KNN model outperformed the other three models, becoming the ML model with best results for predicting the best period of time to send marketing communications.

The second approach of the first experience comprised analyzing the ensemble's performance using the ensemble techniques such as simple averaging and stacking. The stacking model outperformed the other ensemble model. Underneath this approach, two other sub-approaches were implemented, removing the SVR from the ensemble in order to understand its predictive performance and behavior and the other consisted in joining a deep learning algorithm estimator called LSTM (trained and evaluated independently) to the ensemble and compare the performance to the ML ensemble.

Finally, it was applied techniques of hyperparameter tuning to all the approaches men-

tioned previously. The KNN model obtained the best results from all the ML models and the stacking model obtained the best results of all the ensemble.

The second experience involved testing unseen data with a new dataset on the best models determined in the first experience, which was in the first approach. Within the ML models individually tested with the unseen data, the Linear Regression model obtained the best performance of the four models. Within the ensemble methods applied, the simple averaging model obtained the best performance.

The data is an important factor in how the models will behave and how the observations are fed into the models. The second dataset showed no linear dependency between the independent variables and the target variable (resulting in low values of r2 squared), which could influence the predicted and associated error values compared to the results from the previous dataset.

Due to the fact that the previous models of the first experience are based on past behaviors (behaviors that could have been changed drastically due to the 2020 pandemic situation) and the models degrade over time, it is necessary to re-train the models with the new data.

Chapter 6

Conclusion

Email Marketing is a form of direct marketing that uses email as a means of commercial communication. In a broader sense, any email sent to a potential subscriber consists of a marketing email.

One main concern in sending any communication to a subscriber is knowing if it will succeed and if it has any interest to the former. The success is reflected in higher open rates, clickthrough rates, or sales rates.

With the constant overflow of communications exchanged between companies and their subscribers, a subscriber is subjected to receive daily communications at any time of the day. This leads to lower visibility of older emails that have not been read before and lower open rates and clickthrough rates.

A solution to this problem is implementing ML algorithms that allow communication automation and ensure each subscriber's personalization.

The automation technology in AI-driven by ML algorithms in Marketing, over the recent years, allows companies to deliver personalized communications and estimate the best period of time to send them. Algorithms of deep learning, an advanced field of ML, enhance even more than personalization.

The research regarding sending a marketing communication at the right time to a subscriber has increased, in recent years, in the commercial and scientific fields. Companies are investing more in the ML algorithms' implementation in their services and products – companies such as Salesforce, Adobe, Starbucks, Netflix, and many more. In the scientific field, the most recent approaches use deep learning algorithms.

This thesis proposes a solution to sending marketing communications at the right period of time to individual subscribers by proposing segmented models based on the historical profile information and a parallel ensemble approach to trained ML algorithms and a individual trained deep learning algorithm. The experiments' environment was simulated by using a dataset containing profile information of subscribers and their KPIs.

This thesis focuses on the segmentation of the subscribers based on the profile information provided by the dataset used. In the future, it is proposed that the subscribers'

segmentation be done as well with the approaches RFM and LTV, or any segmentation that allows more personalized and tailored experiences to the subscribers.

The segmentation was implemented using traditional clustering algorithms such as K-Means. In the future, the work can extend to the derivated clustering algorithms (e.g., Fuzzy-C) that improve segmentation performance, demand less computational resources, and are less time-consuming. In addition, the segmentation can also be implemented with the algorithm Stream Clustering proposed in (Carnein and Trautmann, 2019), which tracks segments over time without expensive recalculations and handles continuous streams of new observations without recomputing the entire model(s).

A deep learning model was also proposed based on time-series data, the recurrent neural networks (RNNs), which extend to long-term short memory models (LSTMs). Due to time constraints and the time-consuming of both models training, it was not possible to test another deep learning model such as FNNs and compare the performance to the RNNs. FNNs could be a good approach to the dataset since the networks do not require special handling and continuous data.

It is also proposed, in the future, that the final model optimizes the sending of the future dates based on the score calculated from the subscriber's KPIs. In other words, based on the number of openings and clickthroughs, calculate the probability score of that subscriber opening and clicking on the email, and based on that probability score, associate the highest probability of the time of the day. It is also proposed that this work extends to mobile messaging, determining the best period of time to communicate to a subscriber via SMS.

The classic approach of the train-test split can create a limitation regarding the deployment phase. Some subscribers might be selected to be trained, and others tested, which means only a few chosen subscribers will have predictions associated. In order to avoid this situation, it is recommended, in the future, to split the dataset into train and test sets based on the date a communication was sent, using the TimeSeriesSplit approach.

The results obtained from all the experiments indicated that the algorithm KNN performed better for the baseline model and best model, and the algorithm SVR performed worst. In terms of the performance of the parallel ensemble, the stacking method obtained the best performance of all the models implemented, obtaining less error in the predictions and the best result in terms of variance of the target variable, which was 91%. The predictions were wrong by 8% and 3%, respectively, in predicting the best day of the week and the hour of the day. The LSTM model, a derivation of the RNNs model, performed worse than the other experiments, obtaining higher prediction errors.

Bibliography

Redouan Abakouy, El Mokhtar En-naimi, Anass El Haddadi, and Elaachak Lotfi. Data-driven marketing: how machine learning will improve decision-making for marketers. In *Proceedings of the 4th International Conference on Smart City Applications*, pages 1–5, 2019. 1

Rockwell Anyoha. The history of artificial intelligence. `http://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/`, 2017. [Online; accessed 12-December-2020]. 6

Ashish. 15 Most Important Features of Scikit-Learn! `https://www.analyticsvidhya.com/blog/2021/07/15-most-important-features-of-scikit-learn/`, 2021. [Online; accessed 27-January-2022]. 15

Think Automation. Tesler's theorem and the problem of defining ai. `https://www.thinkautomation.com/bots-and-ai/teslers-theorem-and-the-problem-of-defining-ai/`, 2020. [Online; accessed 16-December-2020]. 6

George A Barnard and Thomas Bayes. Studies in the history of probability and statistics: Ix. thomas bayes's essay towards solving a problem in the doctrine of chances. *Biometrika*, 45(3/4):293–315, 1958. 22

Melinda Bartley. The best time to send emails to boost opens, clicks and sales. `https://optinmonster.com/the-best-time-to-send-emails-heres-what-studies-show/`, 2021. [Online; accessed on 27-July-2021]. 1

Leonard E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In Oved Shisha, editor, *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8, University of California, Los Angeles, 1972. Academic Press. 27

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A%3A1010933404324. 22, 23

Brophy. About send time optimization. https://help.listrak.com/en/articles/1522912-scheduled-and-recurring-messages, 2020. [Online; accessed on 20-November-2020]. 29

Jason Brownlee. A gentle introduction to k-fold cross-validation. https://machinelearningmastery.com/k-fold-cross-validation/, 2020a. [Online; accessed on 26-June-2021]. 17

Jason Brownlee. Discover feature engineering, how to engineer features and how to get good at it. https://machinelearningmastery.com/discover-feature-engineering-how\-to-engineer-features-and-how-to-get-good-at-it/, 2020b. [Online; accessed on 22-March-2021]. 15

Jason Brownlee. How to handle missing data with python. https://machinelearningmastery.com/handle-missing-data-python/, 2020c. [Online; accessed on 15-April-2021]. 15, 17

Jason Brownlee. Metrics to evaluate machine learning algorithms in python. https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/, 2020d. [Online; accessed on 20-June-2021]. 18

Giovanni Bruner. No, artificial intelligence doesn't exist (yet). https://towardsdatascience.com/no-artificial-intelligence-doesnt-exist-yet-3318d83fdfe8, 2020. [Online; accessed 28-July-2021]. 7

Pamela Bump. Why consumers subscribe and unsubscribe from email [new data]. blog.hubspot.com/marketing/why-consumers-subscribe-to-email, 2020. [Online; accessed 17-December-2020]. 2

Matthias Carnein and Heike Trautmann. Customer segmentation based on transactional data using stream clustering. *Advances in Knowledge Discovery and Data Mining*, pages 280–292, 03 2019. doi: 10.1007/978-3-030-16148-4_22. 32, 33, 34, 80

Chu Chai Chan. Intelligent value-based customer segmentation method for campaign management: A case study of automobile retailer. *Expert Systems with Applications*, 34:2754–2762, 05 2008. doi: 10.1016/j.eswa.2007.05.043. ix, 30, 32, 34

Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016. doi: 10.1145/2939672.2939785. 19

A. Joy Christy, A. Umamakeswari, L. Priyatharsini, and A. Neyaa. Rfm ranking – an effective approach to customer segmentation. *Journal of King Saud University - Computer and Information Sciences*, 33(10):1251–1257, 2021. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2018.09.004. URL https://www.sciencedirect.com/science/article/pii/S1319157818304178. ix, 31, 32, 34

Collins. Definition of subscription rate. https://www.collinsdictionary.com/us/dictionary/english/subscription-rate, 2020. [Online; accessed on February 6-February-2021]. xvii

Andreia Conceição and João Gama. Main factors driving the open rate of email marketing campaigns. In *Discovery Science*, pages 145–154, Cham, 2019. Springer International Publishing. ISBN 978-3-030-33777-3. doi: 10.1007/978-3-030-33778-0_12. ix, 22, 23, 29

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995. 23

Alexandros Deligiannis, Charalampos Argyriou, and Dimitrios Kourtesis. Predicting the optimal date and time to send personalized marketing messages to repeat buyers. *International Journal of Advanced Computer Science and Applications*, 11:90–99, 05 2020a. doi: 10.14569/IJACSA.2020.0110413. ix, 19, 20, 29

Alexandros Deligiannis, Charalampos Argyriou, and Dimitrios Kourtesis. Building a cloud-based regression model to predict click-through rate in business messaging campaigns. *International Journal of Modeling and Optimization*, 10:26–31, 02 2020b. doi: 10.7763/IJMO.2020.V10.742. ix, 19, 21, 29

Sachin Deshmukh. Study on artificial intelligence in marketing. *International Journal of Advance and Innovative Research*, pages 112–116, 03 2019. ISSN 2394-7780. 1

Oxford Dictionary. Digital marketing. https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095718214, 2020. [Online; accessed 12-December-2020]. 5

Pedro Domingos. A few useful things to know about machine learning. https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf, 2020. [Online; accessed on 25-May-2021]. 15

Economist. How germany's otto uses artificial intelligence? https://www.economist.com/business/2017/04/12/how-germanys-otto-uses-artificial-intelligence, 2020. [Online; accessed on 20-November-2020]. 30

Marketing Evolution. What is ai marketing? https://www.marketingevolution.com/marketing-essentials/ai-markeitng, 2020. [Online; accessed on 28-November-2020]. 11

E. Fariborzi and M. Zahedifard. E-mail marketing: Advantages, disadvantages and improving techniques. *International Journal of e-Education, e-Business, e-Management and e-Learning*, 2012. 1

Jason Fernando. Return on investment (roi. https://www.investopedia.com/terms/r/returnoninvestment.asp, 2020. [Online; accessed 20-May-2021]. xvi

Caroline Forsey. The Ultimate List of Email Marketing Stats for 2020. blog.hubspot.com/marketing/email-marketing-stats, 2020. [Online; accessed 17-December-2020]. 1

Jerome H. Friedman. Greedy function approximation: A gradient boostingmachine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi: 10.1214/aos/1013203451. 23

Futurism. Ray kurzweil: "ai will not displace humans, it's going to enhance us. https://futurism.com/ray-kurzweil-ai-displace-humans-going-enhance, 2020. [Online; accessed on 11-June-2021]. 7

Soledad. Galli. *Python Feature Engineering Cookbook: Over 70 recipes for creating, engineering, and transforming features to build machine learning models*. Packt Publishing, 2020. 16

Francis Galton. Regression Towards Mediocrity in Hereditary Stature. https://www.jstor.org/stable/2841583, 1886. [Online; accessed 28-September-2021]. 9

Seth William Glazier. Sequential survival analysis with deep learning. *Theses and Dissertations*, 7528, 2019. 24

Yoav Goldberg. A primer on neural network models for natural language processing, 2015. 19

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618. 8, 9, 14

Leo Gugery. Newell and simon's logic theorist: Historical background and impact on cognitive modeling. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50, 2006. 6

Aurélien. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2 edition, 2017. 16

Elizabeth Hardcastle. Business information systems. https://paginas.fe.up.pt/~apm/ESIN/docs/bis.pdf, 2020. [Online; accessed on 11-June-2021]. 37

Simon Hay. How ai and deep learning enhances market research. https://www.researchworld.com/how-ai-and-deep-learning-enhances-market-research/, 2019. [Online; accessed on 26-February-2021]. 14

Matt Henderson. *Machine Learning for Beginners 2019 : The Ultimate Guide to Artificial Intelligence, Neural Networks, and Predictive Modelling (Data Mining Algorithms & Applications for Finance, Business & Marketing)*. Independently Published, 2019. xiv, xvi, 7, 8

Or Herman-Shaffar. Time based cross-validation. https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8, 2020. [Online; accessed on 27-July-2021]. 17

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735. 50

Dan Hofstadter. *Godël, Eschcer, Bach: An Eternal Golden Braid*. Basic Books, 1979. 6

Hubspot. The ultimate guide to email tracking. https://www.hubspot.com/sales/email-tracking-guide, 2020. [Online; accessed on 16th December 2020]. xv

IBM. Neural networks. https://www.ibm.com/cloud/learn/neural-networks, 2020. [Online; accessed 27-January-2022]. ix, 13

Smart Insights. How ai is changing the role of the marketer in 2018. https://www.smartinsights.com/managing-digital-marketing/managing-marketing-technology/how-ai-is-changing-the-role-of-the-marketer-in-2018/, 2018. [Online; accessed on 16-June-2021]. 12

Marketing AI Institute. The 2021 state of marketing ai report. "https://www.marketingaiinstitute.com/2021-state-of-marketing-ai-report?hsCtaTracking=5da6ecf5-d127-48d0-a850-67bf1ff9c9e4%7C2e72b1d2-7062-4522-87ff-, 2021. [Online; accessed on 3-March-2021]. 12

AT INTERNET. Lead. www.atinternet.com/en/glossary/lead/, 2020. [Online; accessed 08-April-2021]. xvi

iPULLRANK. Machine Learning for Marketers – A Comprehensive Guide to Machine Learning. assets.ctfassets.net/j5zy0n17n2ql/2D4mX8PjV6iC6i8cIuSCwk/23a4ebb99a6e9d5a82b2f03e1262f39d/ml-whitepaper.pdf, 2020. [Online; accessed 18-December-2020]. 3, 9

Jedid-Jah Jonker, Nanda Piersma, and Dirk Van den Poel. Joint optimization of customer segmentation and marketing policy to maximize long-term profitability. *Expert Systems with Applications*, 27:159–168, 08 2004. doi: 10.1016/j.eswa.2004.01.010. ix, xvi, 30, 31, 34

Jupyter. Jupyter project documentation. https://docs.jupyter.org/en/latest/, 2020. [Online; accessed on 31-January-2022]. 46

John Kelleher. *Deep Learning*. The Mit Press, 2019. xiv, xv, xvi, 12, 13, 14

Michelle Knight. What is data-driven? https://www.dataversity.net/what-is-data-driven/, 2021. [Online; accessed on 05-May-2021]. 11

Y. E. Kocaguneli, Kultur, and A. Bener. Combining multiple learners induced on multiple datasets for software effort prediction. *International Symposium on Software Reliability Engineering (ISSRE)*, 2009. 10

Will Koehrsen. Hyperparameter tuning the random forest in python. https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74 2018. [Online; accessed on 31-January-2022]. x, 60

Joos Korstanje. The f1 score. https://towardsdatascience.com/the-f1-score-bec2bbc38aa6, 2021. [Online; accessed on 27-January-2022]. 23

Satyam Kumar. Understanding k-means, k-means++ and, k-medoids clustering algorithms. https://towardsdatascience.com/understanding-k-means-k-means-and-k-medoids-clustering-algorithms-ad9c9fbf47ca, 2020. [Online; accessed on 31-January-2022]. 48

Gautam. Kunapul. Ensemble methods for machine learning. https://www.manning.com/books/ensemble-methods-for-machine-learning, 2020. [Online; accessed 12-January-2021]. 9, 10

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and cox regression, 2019. 26

Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. doi: 10.1038/nature14539. 13

Lexico. Saas. https://www.lexico.com/definition/saas, 2020. [Online; accessed on 10-February-2021]. xvi

Jay. Liebowitz. *Data Analytics and AI (Data Analytics Applications).* Auerbach Publication, 2020. 11

Xiao Luo, Revanth Nadanasabapathy, A. Nur Zincir-Heywood, Keith Gallant, and Janit Peduruge. Predictive analysis on tracking emails for targeted marketing. In Nathalie Japkowicz and Stan Matwin, editors, *Discovery Science*, pages 116–130, Cham, 2015. Springer International Publishing. ISBN 9783319242828. ix, 23, 24, 29

John McCarthy. What is ai? basic questions. http://jmc.stanford.edu/artificial-intelligence/what-is-ai/index.html, 2020. [Online; accessed 27-January-2022]. 6

Ryszard Stanislaw Michalski, Jaime Guillermo Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013. 7, 8

Shubhi Mitall. Proximity marketing examples: 28 retail companies nailing it with their campaigns. https://blog.beaconstac.com/2016/02/25-retailers-nailing-it-with-their-proximity-marketing-campaigns/, 2021. [Online; accessed on 28-February-2021]. 12

Campaign Monitor. The New Rules of Email Marketing. www.campaignmonitor.com/resources/guides/email-marketing-new-rules, 2020. [Online; accessed 17-December-2020]. xv, xvi, 1, 2, 38

Nitin Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE international systems engineering symposium (ISSE)*, pages 1–7. IEEE, 2017. 19

Dan Nelson. Ensemble/Voting Classification in Python with Scikit-Learn. https://stackabuse.com/

ensemble-voting-classification-in-python-with-scikit-learn/, 2020. [Online; accessed on 15-May-2021]. 10

Jaime Netzer. The top millennial buying habits and insights for 2021. https://khoros.com/blog/millennial-buying-habits, 2020. [Online; accessed on 12-February-2021]. 2

Optideia. Lifetime value. https://www.optimizely.com/optimization-glossary/lifetime-value/, 2020. [Online; accessed on 11-June-2021]. xv

Jan Paralic, Tomáš Kaszoni, and Jakub Mačina. Predicting suitable time for sending marketing emails. In *Information Systems Architecture and Technology: Proceedings of 40th Anniversary International Conference on Information Systems Architecture and Technology – ISAT 2019*, pages 189–196, Cham, 2020. Springer International Publishing. ISBN 9783030306045. ix, 22, 29

Nanda Piersma and Jedid-Jah Jonker. Determining the optimal direct mailing frequency. *European Journal of Operational Research*, 158:173–182, 02 2004. doi: 10.1016/S0377-2217(03)00349-7. ix, 27, 28, 29

Gil Press. Cleaning big data: Most time-consuming, least enjoyable data science task, survey say. https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation/-most-time-consuming-least-enjoyable-data-science-task-survey-says/#1594bda36f633, 2020. [Online; accessed on 20-March-2021]. ix, 14, 15

J. R. Quinlan. Induction of decision trees. *MACH. LEARN*, 1:81–106, 1986. 22, 23

Cox D. R. Regression models and life tables. *Journal of the Royal Statistic Society*, B (34):187–202, 1972. 24, 26

Aliaa Rassem, Mohammed El-Beltagy, and Mohamed Saleh. Cross-country skiing gears classification using deep learning, 2017. x, 51

Jesse Richardson. What's the difference between a subscriber and a customer? https://www.business2community.com/online-marketing/whats-the-difference-between-a-subscriber-and-a-customer-01268403, 2020. [Online; accessed on 11-June-2021]. xv, xvii

Stuart Russell and Peter Norving. *Artificial Intelligence: A Modern Approach*. Pearson, 2020. xiv, 6, 8, 9

Salesforce. Fourth annual state of marketing: Insights and trends from 3.500 global marketing leaders. www.salesforce.com/content/dam/web/en_us/www/assets/pdf/

datasheets/salesforce-research-fourth-annual-state-of-marketing.pdf, 2017. [Online; accessed 17-December-2020]. 2, 12

saranya Mandava. Cross validation and hyperparameter tuning in python. https://medium.com/@mandava807/cross-validation-and-hyperparameter-tuning-in-python-65cfb80ee485, 2018. [Online; accessed on 27-January-2022]. 53

Shipra Saxena. Here's All you Need to Know About Encoding Categorical Data. https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/, 2020. [Online; accessed 27-January-2022]. 16

scikit learn. Scikit-learn Machine Learning in Python. https://scikit-learn.org/stable/, 2020. [Online; accessed 27-January-2022]. 15, 17

Seventh Sense. Using send time optimization (sto) with hubspot multistep automated workflow campaigns. https://theseventhsense.zendesk.com/hc/en-us/articles/360046757731\-Using-Send-Time-Optimization-STO-with-HubSpot-Automated-Workflow-Campaigns, 2020. [Online; accessed on 20-November-2020]. 29

Gaurav Sharma. 4 types of customer segmentation all marketers should know. https://www.business2community.com/customer-experience/4-types-of-customer-segmentation-all-marketers-should-know-02120397, 2020. [Online; accessed on 22-December-2020]. 34

Harvineet Singh, Moumita Sinha, Atanu Sinha, Sahil Garg, and Neha Banerjee. *Determining strategic digital content transmission time utilizing recurrent neural networks and survival analysis*. Adobe Inc, 2019. ix, 26, 27, 29

Harvineet Singh, Moumita Sinha, Atanu R. Sinha, Sahil Garg, and Neha Banerjee. An rnn-survival model to decide email send times, 2020. ix, xv, xvii, 24, 26, 29, 37

Moumita Sinha, Vishwa Vinay, and Harvineet Singh. Modeling time to open of emails with a latent state for user engagement level. *CoRR*, abs/1908.06512, 2019. ix, 24, 25, 29

Smartt. Einstein send time optimization for journey builder. https://smaartt.com/knowledge/einstein-send-time-optimization-for-journey-builder/, 2020. [Online; accessed on 26-February-2021]. xvi, 29

Oliver Theobald. *Machine Learning for Absolute Beginners: A Plain English Introduction*. Independently Published, 2018. xiv, xvi, xvii, 8, 9

Amber Tunnell. Send time optimization. https://help.bluecore.com/en/articles/3616204-send-time-optimization, 2020. [Online; accessed on 20-November-2020]. 29

Glen Urban, Artem Timoshenko, Paramveer Dhillon, and John R. Hauser. Is deep learning a game changer for marketing analytics? https://sloanreview.mit.edu/article/is-deep-learning-a-game-changer-for-marketing-analytics/, 2019. [Online; accessed on 18-May-2021]. 14

Nicolo Valigi and Gianluca Mauro. *"Zero to AI"*. Manning Publication, 2020. 7, 11

Jordie van Rijn. DMA National Client Email Report 2015. www.emailmonday.com/dma-national-client-email-report-2015/, 2015. [Online; accessed 16-December-2020]. 1

Velocidi. Velocidi. https://www.velocidi.com/blog/audience-segmentation-on-autopilot/, 2020. [Online; accessed on 20-November-2020]. 30

Ray Venkatesan and Jim Lecinski. *A Five-Stage Road Map to Implementing Artificial Intelligence in Marketing*. Stanford University Press, 2021. 12

Haley Walden. Deep learning for marketing: What you need to know. https://www.elegantthemes.com/blog/marketing/deep-learning-for-marketing, 2020. [Online; accessed on 18-May-2021]. 14

Xiang-Bin Yan and Yi-Jun Li. Customer segmentation based on neural network with clustering technique. In *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, AIKED'06, page 265–268, Stevens Point, Wisconsin, USA, 2006. World Scientific and Engineering Academy and Society (WSEAS). ISBN 1112222339. ix, 33, 34

Dynamic Yield. Meet dynamic yield. https://www.dynamicyield.com/product-demo/, 2020. [Online; accessed on 20-November-2020]. 30

Y. Yu, Zhou Z., and K. M. Ting. Cocktail ensemble for regression. *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 721–726, 2007. doi: 10.1109/ICDM.2007.60. 10

Andreas Zell. *Simulation neuronaler Netze*. Oldenbourg, 2000. ISBN 978-3-486-24350-5. 27