Forecast in the Pharmaceutical Area – Statistic Models vs Deep Learning

Raquel Ferreira[1], Martinho Braga[2], and Victor Alves[3(✉)]

[1] Department of Informatics, University of Minho, Braga, Portugal
raquel.mferreira@hotmail.com
[2] Tlantic Portugal SI, Porto, Portugal
martinho.braga@tlantic.com
[3] Algoritmi Centre, University of Minho, Braga, Portugal
valves@di.uminho.pt

Abstract. The main goal of this work was to evaluate the application of statistical and connectionist models for the problem of pharmacy sales forecasting. Since R is one of the most used software environment for statistical computation, we used the functions presented in its forecast package. These functions allowed for the construction of models that were then compared with the models developed using Deep Learning algorithms. The Deep Learning architecture was constructed using Long Short-Term Memory layers. It is very common to use statistical models in time series forecasting, namely the ARIMA model, however, with the arising of Deep Learning models our challenge was to compare the performance of these two approaches applied to pharmacy sales. The experiments studied, showed that for the used dataset, even a quickly developed LSTM model, outperformed the long used R forecasting package ARIMA model. This model will allow the optimization of stock levels, consequently the reduction of stock costs, possibly increase the sales and the optimization of human resources in a pharmacy.

Keywords: ARIMA · Deep Learning · Forecast · LSTM · Pharmacy sales

1 Introduction

In the pharmaceutical retailing industry, it's essential to ensure there is an adequate stock of medications and supplies to serve the needs of pharmacy clients, so a careful inventory management can increase its profitability [1]. The constitution of stocks aims to combat stock disruption, the eventualities related with consumption, unforeseen circumstances associated with the delivery and serve as a regulator between deliveries and uses that are made at different rates. The manager of a pharmacy has as main tasks study the stock level and its required quantity, the amount to order and what is the best time to place the order. The objective is avoid an excess of stock and, at the same time, ensure that a stock rupture does not occur [1, 2]. It's important to highlight that drug shortage can disrupt the healing processes of patients and produce negative effects on public health.

Á. Rocha et al. (Eds.): WorldCIST'18 2018, AISC 747, pp. 165–175, 2018.
https://doi.org/10.1007/978-3-319-77700-9_17

Forecasting future demands can increase operational efficiency, effectiveness, and flexibility, improving customer services, and reducing both costs and problems. [1, 2]. In this context, the concept of Business Intelligence emerges, which brings a wide range of technologies that intend to extract from raw data, patterns, and trends that can be used to improve decision making in an organization, such as forecasting. These techniques represent a shift from tactical thinking to strategic thinking [3].

Forecasting is required in many situations, and whatever the circumstances or time horizons involved it is considered an important aid to effective and efficient planning. The predictability of an event depends on how well we understand the factors that contribute to it, how much data is available and whether the forecasts can affect what we are trying to forecast [4, 5]. When satisfied these three conditions, forecasts can be highly accurate. However, a forecasting model may in some cases not add any value. Anything that is observed sequentially over time is considered a time series and it aims to estimate how the sequence of observations will continue in the future. Forecasting time series commonly extrapolate seasonal trends and patterns, and all auxiliary information such as marketing initiatives, competitor activity, changing economic conditions can subsequently be identified by analyzing the data [4, 5].

A time series can be broken down into four elements:

1. Cycles C_t: Variations in the series that although periodic do not have any known periodicity.
2. Trend T_t: Increase or decrease in the behavior of the series over time.
3. Seasonality S_t: Cycles or repetitive patterns that occur over time.
4. Noise ε_t: Variations that occur along the time series without possible explanation.

2 Methods and Related Word

The experiments for the preparation of this study were based on a statistical and a deep learning approach.

Statistical models

With statistics, we can study, analyze and collect information for making predictions and conclusions about the future. R is a free software environment for statistical computation and graphics widely used by scientists, companies, mathematicians and analysts which choose to incorporate this language into their business logic since it offers a better view of all the information they are dealing with, does not require the development of customized tools and is very performant [4, 6, 7].

It has available a forecasting package, called *forecast* that was used in this study. The chosen integrated development environment for working with this software was *Rstudio*.

Deep learning models

Deep Learning (DL) is a subfield of Machine Learning (ML) where the system can discover how to learn, generate and transform features from data on its own, exploring neural networks with more than one layer of nonlinear processing [8, 9]. It is about learning multiple levels of representation and abstraction that help to make sense of data [9].

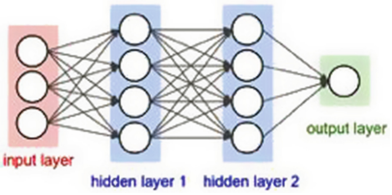

Fig. 1. Neural net connections [10].

Neural Networks (NN) consist in many processors called neurons simply connected, each producing a sequence of real-valued activations. Input neurons get activated through sensors perceiving the environment, while others get activated through weighted connections from previously active neurons [8]. There are basically three types of layers (input, hidden and output) and depending on the problem we want to model they can have more than one hidden layer (Fig. 1).

Time series usually contain temporal dependencies that make two otherwise identical time points belong to different classes or predict different behaviors. This feature usually increases the difficulty of analyzing and predicting them. Recurrent neural networks are a type of NN designed to deal with sequence dependency. A Long-Short Term Memory (LSTM) network is a recurrent neural network architecture that remembers data over arbitrary time intervals. It is a network optimized to learn from and act on temporal data, with indefinite or unknown temporal dimension. Its architecture allows persistence by giving the network a certain memory since it can be seen as multiple copies of the same network. It has been shown that LSTMs have the capability to predict various non-linear behaviors [11, 12].

In [13] the authors describe the use of a deep neural network architecture in the weather prediction. It uses multi-stacked LSTMs to map sequences of weather values of the same length. It was demonstrated that LSTMs are competitive with the traditional methods and can also be considered a better alternative to forecasting general weather conditions. Another study that has used LSTM is presented in [12] and it's about the human trajectory in crowded spaces. The authors argue that "the problem of trajectory prediction can be viewed as a sequence generation task, where we are interested in predicting the future trajectory of people based on their past positions". At the end, it was shown that LSTMs has the capability of predict various non-linear behaviors, such as a group of individuals moving together.

The programing language used for applying DL was Python as it is a general-purpose programming language, unlike R and Matlab, which are more oriented for numerical computing [14]. SciPy is a collection of mathematical algorithms and convenience functions built on the *NumPy* extension of Python, which enables the development of sophisticated programs and specialized applications [15]. S*cikit-learn* is a machine learning library, interoperating with Python numerical and scientific libraries such as *NumPy* and *SciPy*. Currently several numerical libraries can be used for developing deep learning models in Python, (e.g. *Theano, TensorFlow, Caffe*). In this work, we used *TensorFlow*, an open source software library for numerical computation using data flow graphs. It was originally developed for conducting machine learning and deep neural networks research [16]. On top of *TensorFlow* we used *Keras*, a high-level neural networks API, written in Python, that supports both convolutional networks and recurrent networks, as well as combinations of the two [17, 18].

3 Experiments

The dataset used contains 779569 records which correspond to a two years sales history of a set of pharmaceutical products and the first processing stage was the choice of the attributes that would be used to train the models, as well as the identification of the distinct categories of products present. The attributes that were selected include the date, the hour of the day and the quantity sold. All values were aggregated per day resulting in a dataset represented only by two attributes: date and quantity, resulting in a dataset with 733 records. Next, seven more attributes were added, one for each day of the week. For each row, the day of the week was identified and the correspondent attribute filled with one (1) leaving all the others filled with zeros (0). For the case oriented to statistics, if all variables were identified this would lead to a multicollinearity error, which occurs when at least two highly correlated predictors are assessed simultaneously in a regression model. So, for these models only six attributes were created since the algorithms themselves are capable of identifying the day that is missing. The identification of the months underwent a similar treatment: twelve new attributes were added to the result dataset, referring to the months of the year, and then populated with binary variables (for the statistical models, only from February to December).

Attributes for significant date events that could influence sales behavior were also added (Table 1). These events included flu season, holidays, bathing season, holiday season (weeks and days before and after), the carnival season, the Easter season, New Year's Day, Christmas Day, among others and were also identified through binary variables. It should be noted that these events are very important factors in identifying periods of seasonality and eventual peaks of sales, which would be more difficult to identify and explain without them. At this step, the dataset consisted of 733 records and 67 attributes. Outliers above 95% and below 5% for which we didn't have an explanation through the selected events were also removed.

Table 1. Attributes of the used dataset.

Attribute	Description	Characteristic
Date	Date of the sell (YYYY-MM-DD)	Linear
Quantity (**predictor**)	Quantity sold in a day	Integer
Days of the week	Monday to sunday (for R: one day less)	Binary
Months of the year	January to december (for R: one month less)	Binary
Events matrix	Days that could influence sales behavior	Binary

A subset of the currently available dataset was treated as future data to then compare forecasts with actual results. The models were developed using a set of data designated by training and then it was applied to the set designated by test, allowing to infer about the credibility and degree of confidence of the model [19]:

- x_train - training dataset (90% of the initial sample)
- y_train - training dataset variable to be predicted ($mean = 4428.13; SD = 1348.396$)

- x_test - test dataset (remaining 10% of the initial sample)
- y_test - test dataset variable to be predicted ($mean = 4106.836; SD = 1788.803$)

Model development

Both experiments followed the same model construction plan (Fig. 2).

Fig. 2. Forecast model development.

Experiment using statistical models

It was decided to use the *forecast* package that includes a large set of models capable of forecasting a univariate time series. Based on literature research, the group of models that are most commonly used and have good results are: Holt-Winters, NNETAR, TSLM and ARIMA [4]. However, since ARIMA was described as having the best results, it will be the model described and assessed in this study. Auto Regressive (AR) with Integrated Moving Average (IMA) (ARIMA) models aim to describe the autocorrelations present in the data. These models are defined by three parameters: order of the AR part (p), number of differentiations needed to obtain a stationary time series (d), order of the IMA part (q). The reason for this model to be one of the most used and the one that usually presents the best results, is due to the fact that it combines these two components, AR and IMA, which enable the creation of models much more robust and complex. The *auto.arima()* function presented in the R *forecast* package automatically chooses these three parameters, making the ARIMA model building an easy task. If the series is not stationary, the function itself differentiates the values [4, 20]. Since the vector constructed with several date events is likely to influence sales behavior it can be incorporate when building the model. This vector is represented in the code below by the variable *x_train*. To analyze the effect of this vector in the precision of the forecast, this experiment was also applied without it (model 2):

```
#Model 1 - auto.arima() function call with the events vector
M1=auto.arima(y_train,xreg=x_train[,-1],max.p=10,
max.P=10,max.q=10,max.Q=10)
#Model 2 - auto.arima() function call without the events vector
M2=auto.arima(y_train,max.p=10,max.P=10,max.q=10,
max.Q=10)
```

Experiment using Deep Learning

ML algorithms generally perform better if the time series has a consistent scale or distribution, which is achieved by normalization or standardization. Since the dataset consists mostly of binary variables and DL uses NN, this leads to choosing normalization rather than standardization because NNs assign different weights to the input values, so the dataset must be all on the same scale [21]. At the end, it was necessary to reverse this normalization so the error could be calculated on the same scale as the variable to be predicted [21]. For the structure of the LSTM networks it was necessary to determine how many layers would be the optimal, as well as the number of nodes and typology of each one. Using few neurons, the network could be unable to model complex data, resulting in *underfitting*. This situation can be recognized when the error is too great, both in the training and the test cases. If the number of neurons is too high for the dataset, it could result in *overfitting,* leading to loss of its predictive capability (the error in the training data decreases while in the test data it stagnates or starts to rise (Fig. 3)).

The final optimized architecture is presented in Fig. 4. Using *Keras* it is not necessary to define the first input layer because it is automatically created when the intermediate layers are defined. The following layers are two pairs of LSTM and Dropout layers. Finally, we included two Multi Perceptron Layers (Dense layers) with a Dropout layer in the middle. The LSTM layers appear in first place since they are the ones with memory capabilities (both old and recent).

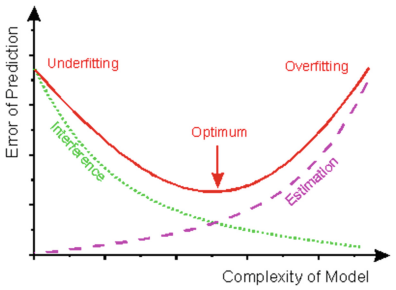

Fig. 3. Underfitting vs overfitting [22].

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 30, 128)	82944
dropout_1 (Dropout)	(None, 30, 128)	0
lstm_2 (LSTM)	(None, 64)	49408
dropout_2 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2080
dropout_3 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 1)	33

Total params: 134,465
Trainable params: 134,465
Non-trainable params: 0

Fig. 4. Architecture of the final model.

The activation function implements the decision regarding the output limits by combining the input values with their weights [10]. Since the case being studied was a regression problem of positive values, the chosen activation function for the *dense* layers was *ReLU*. It sets all negative values to zero and the rest remain as they are. In our first approaches in the design of the network architecture we verified the occurrence of overfitting. Generally, it can be reduced by adding more cases to the training set, reducing the complexity of the network architecture or adding *dropout* between the layers [10]. *Dropout* corresponds to erasing parts of the neural network allowing the network to generalize and depends on the problem and the results achieved in the training. Due to the fact that the size of the dataset was fixed, the number of nodes in the second LSTM layer was reduced and, consequently, in the first *Dense* layer also. Three *dropout* layers between the other layers were also needed.

An *epoch* consists of one full training cycle on the training cases. The *batch size* is the number of training cases that are used in each *epoch*. The number of *epochs* and *batch size* were increased until the appearance of overfitting. The best results were achieved with 1300 *epochs* and a *batch size* of 600.

4 Results

The presented results were obtained after a cycle process of parameter fine tuning with the goal of obtaining satisfactory results (Fig. 5).

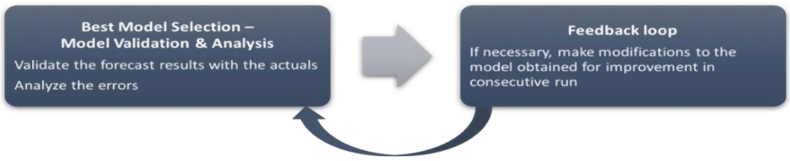

Fig. 5. Model fine-tuning cycle.

Statistical Models

The best model was selected using the Root Mean Squared Error (RMSE) analysis and the correlation between the real and predicted values. The discrepancy of the error between the two *auto.arima()* models highlights the relevance of the *x_train* vector in the predictive capability of the model. RMSE is commonly used to measure accuracy since it measures the mean magnitude of the errors in a set of predictions and the average magnitude of the error itself. This is probably the most easily interpreted error since it's presented in the units of the variable that is being predicted and depends on its range of values [23]. The correlation, as the name indicates, aims to measure the degree of relationship between variables, in this case, between *y_test* values and the predictions made.

The model that presented the best results was the *auto.arima*() using the *x_train* vector, which obtain 1352,951 for the RMSE and reached 66% in the correlation test (Table 2). The obtained values for the ***p***, ***d*** and ***q*** parameters were 3, 1 and 3, respectively, which means that the time series was not considered stationary by the model and it had to differentiate the series once.

The *noise* (residuals) relating to predicted values should be similar to a white noise, i.e., independently of the fraction of time that is observed, it should look like floating points with no meaning. If the residuals don't exhibit a white noise-like behavior it is an indication that additional improvements can be made to the forecast model. Through the AutoCorrelation function (*acf*) graph, the histogram (check normal distribution) and applying the *Box-Pierce* and *Ljung-Box* it was possible to verify if the residuals were similar to a *white noise*. The *acf* is a measure of the correlation between the observations of a time series that are separated by k (lags) units of time $(y_t \, \text{e} \, y_{t-k})$. It is possible to have 1 or 2 significant correlations in higher order lags, however they can be due to random errors [24]. In the *Box-Pierce* test, if the *p-value* is less than 0.05 indicates that the series could be non-stationary [4].

Finally, it was concluded that the model successfully captured all the information present in the data since all these tests reported that there was no correlation present in the residuals.

Deep Learning

The DL analysis was done during the construction of the model. Like the loop presented in Fig. 5, the parameters were changed several times until they were finally adjusted to the model. Loss graphs were analyzed, showing the error that occurred for the training and test data during the training periods (*epochs*).

Table 2. RMSE and correlation average measure

	ARIMA	AVG (LSTM)
RMSE	1352.95	1220.55
Correlation	0.65	0.69

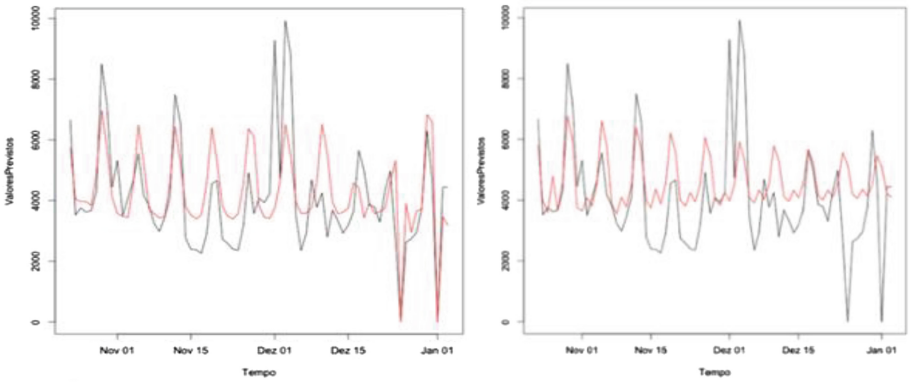

Fig. 6. Forecast graphs using the modeling function *auto.arima()* (with the events vector - left graph; without the events vector – right graph).

The validation of the model was also done using the RMSE and the correlation value. Since the initialization of the network is random it causes a variation in these results, so the model was applied to the training data seven times and then an average of its results was made. The maximum learning occurred at 750 *epochs* and 600 *batch size*.

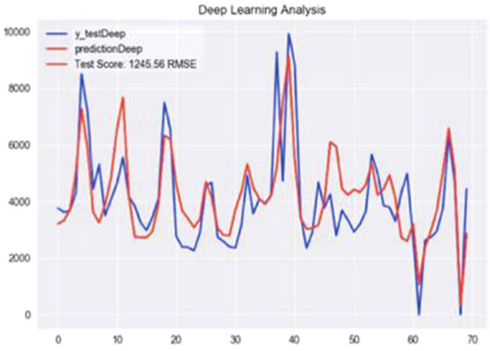

Fig. 7. Forecast graphs using the DL approach.

The results of the two approaches can be observed and compared in Table 2 and Figs. 6 and 7. The RMSE of the ARIMA model and the average of the DL model when compared with the mean of the *y_test* values (4106.836) are 32,94% and 29,72%, respectively. In the graph of the ARIMA model at the right (without the events vector), the model is doing mostly an average of the values, being the predicted values near the middle of the graph. On the other ARIMA graph, the effect of using the events vector, can be verified by the two low values in the end. The LSTM networks also could identify this "outliers" with the corresponded events, showing the capacity of learning by these networks. In general, both the models could follow the behavior of the *y_test* series.

5 Conclusions

After comparing the results obtained by the two experiments it was possible to affirm that the deep learning approach achieved better results than the usually used traditional statistical model ARIMA, both in the RMSE and the correlation value. In both models, when the obtained *noise* series were analyzed, it was concluded that they captured the maximum information present in the data, which means that the models were correct and considered accurate.

DL in general and LSTM in particularly is still a growing technique but it is one that is increasingly emphasized by its success in several areas and this study has demonstrated the potential for deep learning in the forecasting area. Each experiment must be analyzed separated and all datasets have different behaviors, translating into different accuracies of precision by the models, however when this study was realized another three very different datasets were also analyzed and DL presented better results for them also.

An interesting analysis would be to study the introduction of a new product in the market, since it does not have a set of historical data to train and construct a model. Eventually using clustering, we could define to which group/category of products it belongs, and do a study from there. Another interesting approach would be to aggregate daily sales per week to verify if this could achieve less error. First it would be necessary to find the correspondent percentage of sales for each day of the week and then divide the weekly forecast by the respective percentage.

Acknowledgments. This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

References

1. Yousefi, N., Alibabaei, A.: Information flow in the pharmaceutical supply chain. Iran. J. Pharm. Res. **14**(4), 1299–1303 (2015)
2. Microsoft: Business Intelligence for Healthcare : The new prescription for boosting Cost Management, Productivity and Medical Outcomes. Business Intelligence for Healthcare: The New Prescription for Boosting Cost Management, Productivity and Medical Outcomes. BusinessWe, February 2009
3. Ashrafi, N., Kelleher, L., Kuilboer, J.-P.: The impact of business intelligence on healthcare delivery in the USA. Interdiscip. J. Inf. **9**(9), 117–130 (2014)
4. Hyndman, R.J., Athanasopoulos, G.: Forecasting: Principles and Practice, pp. 46–51. OTexts, Australia (2014)
5. Shumway, R.H., Stoffer, D.S.: Time Series Analysis and Its Applications, 3rd edn. Springer, Heidelberg (2017)
6. Ohri, A.: Why every business analyst needs to learn R? (2012). http://analyticstraining.com/2012/why-every-business-analyst-needs-to-learn-r/. Accessed 01 June 2017
7. Verma, E.: A Quick Guide to R Programming Language for Business Analytics (2015). https://www.simplilearn.com/r-programming-language-business-analytics-quick-guide-article. Accessed 01 June 2017
8. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
9. Deng, L., Yu, D., Deep Learning : Methods and Applications (2013)
10. Russakovsky, O.: Convolutional Neural Networks for Visual Recognition (2015)
11. Olah, C.: Understanding LSTM Networks (2015)
12. Alahi, K., Goel, V., Ramanathan, A., Robicquet, Fei-Fei, L., Savarese, S.: Social LSTM : Human Trajectory Prediction in Crowded Spaces (2014)
13. Zaytar, M.A., El Amrani, C.: Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. Int. J. Comput. Appl. **143**(11), 975–8887 (2016)
14. May, M.: An Overview of Python Deep Learning Frameworks (2017)
15. Community, S.: SciPy Reference Guide (2015)
16. TensorFlow: https://www.tensorflow.org/. Accessed 01 Nov 2017
17. Keras: https://keras.io/. Accessed 01 Nov 2017
18. Brownlee, J.: Deep Learning With Python. https://machinelearningmastery.com/deep-learning-with-python/. Accessed 18 June 2017
19. Steinberg, D.: Why Data Scientists Split Data into Train and Test (2014). http://info.salford-systems.com/blog/bid/337783/Why-Data-Scientists-Split-Data-into-Train-and-Test. Accessed 11 June 2017
20. Hyndman, R.J.: New in forecast 4.0 (2012). https://robjhyndman.com/hyndsight/forecast4/. Accessed 12 June 2017
21. Brownlee, J.: How to Normalize and Standardize Time Series Data in Python (2016). http://machinelearningmastery.com/normalize-standardize-time-series-data-python/. Accessed 18 June 2017

22. Dieterle, D.F.: Overfitting, Underfitting and Model Complexity (2016)
23. Wesner, J.: MAE and RMSE — Which Metric is Better? (2016). https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d. Accessed 15 June 2017
24. Mukaka, M.M.: Statistics corner: a guide to appropriate use of correlation coefficient in medical research. Malawi Med. J. **24**(3), 69–71 (2012)