NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Time Series Forecasting for a Call Center in a Warsaw Holding Company

Dominika Leszko

Internship report presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

TIME SERIES FORECASTING FOR A CALL CENTER IN A WARSAW HOLDING COMPANY

by

Dominika Leszko

Internship report presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

Supervisor: Flavio Pinheiro
External Supervisor*:* Pawel Brach

April 2020

ABSTRACT

In recent years, artificial intelligence and cognitive technologies are actively being adopted in industries that use conversational marketing. Workforce managers face the constant challenge of balancing the priorities of service levels and related service costs. This problem is especially common when inaccurate forecasts lead to inefficient scheduling decisions and in turn result in dramatic impact on the customer engagement and experience and thus call center's profitability. The main trigger of this project development was the Company X's struggle to estimate the number of inbound phone calls expected in the upcoming 40 days. Accurate phone call volume forecast could significantly improve consultants' time management, as well as, service quality. Keeping this goal in mind, the main focus of this internship is to conduct a set of experiments with various types of predictive models and identify the best performing for the analyzed use case. After a thorough review of literature covering work related to time series analysis, the empirical part of the internship follows which describes the process of developing both, univariate and multivariate statistical models. The methods used in the report also include two types of recurrent neural networks which are commonly used for time series prediction. The exogenous variables used in multivariate models are derived from the Media Planning department of the company which stores information about the ads being published in the newspapers. The outcome of the research shows that statistical models outperformed the neural networks in this specific application. This report covers the overview of statistical and neural network models used. After that, a comparative study of all tested models is conducted and one best performing model is selected. Evidently, the experiments showed that SARIMAX model yields best predictions for the analyzed use-case and thus it is recommended for the company to be used for a better staff management driving a more pleasant customer experience of the call center.

KEYWORDS

Machine learning; Time Series Analysis; ARIMA; Forecasting; Supervised Learning; Predictive Models; Artificial Neural Networks; Recurrent Neural Networks; Gated Recurrent Unit; Long Short Term Memory;

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In recent years, call centers have been revolutionized by the data. Even though some static one-size-fits-all strategies, as well as, static call scripts still remain, technology has changed significantly the way that call centers are functioning. The power of call centers remains in the huge amounts of data that such institutions gather when interacting with customers. It is possible for call center agents to know yet before receiving the call many crucial characteristics of the customer, such as which advertisement he has seen, what age range he falls into and what is his expenditure tendency. Naturally, as the time passes by, the customer database becomes larger and finally powerful enough to enable the call center to leverage all available data and drive appropriate interaction with each customer. American Express proved that 78 % of consumers have resigned from making an intended purchase due to poor customer service experience (2017, American Express Barometer). This is a clear sign that call centers must direct their focus onto a seamless and convenient experience for customers, unless they are ready to risk losing out on a competitor. To ensure that such scenario does not happen, more and more call centers are turning to technologies, such as Artificial Intelligence, Machine Learning and Time Series Analysis in order to provide them with useful insights. Layering in AI can include call volume forecast for better resource management or customer churn to determine appropriate marketing strategies, next best action, and more. Below report will present one of many possible applications of Statistical and Machine Learning methodologies aiming at improvement of operational efficiency in the life cycle of a Call Center.

1.1. PROJECT DESCRIPTION

At the second year of the NOVA IMS master program in Data Science and Advanced Analytics, I have enrolled in the internship program at a Holding Company (Company X) in Warsaw, Poland. The below report is a summary of the analytic activities developed by me as an intern during that period. During the internship, I joined the analytical team, which was mainly in charge of providing data visualizations in the form of BI dashboards, performing ETL processes and building a reliable Data Warehouse.

The main focus of my work was to identify Machine Learning opportunities within the umbrella of organizations to improve and automate the decision making processes. After throughout analysis of possible AI implementations and a number of discussions with business decision makers within the company, it has been agreed that the biggest need for automated optimization is currently laying in the subsidiary company providing Call Center service. More

specifically, the main challenge that the Call Center is currently facing is the consultants' inefficient time management. Since the number of inbound phone calls is very unexpected, current call load balancing proves to be inefficient, often leading to either inactive at work agents as phone call frequency drops, or, on the other hand, insufficient number of calling agents at peak times. The latter often leads to dissatisfaction of customers caused by extended waiting time to be served by the consultant. This problem was previously attempted to be tackled within the company by means of a simple estimation that the phone call volume at the X following time steps would be equal to its average of X proceeding days. This solution did not result in a satisfactory accuracy, neither did it take advantage of the data stored at the Media Planning section of the company. This being said, the company's solution leaves a lot of room for improvement given the relatively poor prediction quality. Last but not least, it is worth mentioning that accurate predictions of the inbound phone calls will bring value to not only the Call Center itself, but also, to the Supply Stock Management, as well as, Delivery Management Teams, which will have a better idea on which days more products will need to be available in stock for shipping.

Finally, in the work reported in this document, we aimed to combine business know-how and data engineering practices to gain the best prediction possible of the number of inbound phone calls for each of the next upcoming 40 days. Ultimately, we aimed to generate a series of predictions of customers' responses to press-published advertisements of dietary supplements with the use of provided data that was collected by the company in the last 2 years. As aforementioned, accurate predictions would help in different areas of company's activities, such as Delivery, Supply Stock Management and foremost in the Call Center's staff time management, ultimately leading to a better Customer Experience and to a better company's image in the market. Last but not least, another objective of the internship was to make a thorough and diversified research of the most optimal model for the analyzed use-case, including univariate and multivariate models, statistical models and recurrent neural networks.

1.2. BUSINESS CONTEXTUALIZATION

The work described in this report was developed during an internship in a Holding Company (Company X), consisting of an umbrella of brands, such as marketing agency, call center agency, logistics provider, software house, financial services provider, beauty service and employment agency.

The company contains rich and diverse data sources describing entire Path-To-Purchase of the customer, as he is targeted with marketing campaigns both online and in the press all the way to the moment when the product is delivered to him. In the present report, the focus is directed to the inbound phone calls from press advertisements aiming to increase the sale volume of dietary supplements. Such predictions will further contribute to primarily better staff management within the Call Center aiming at improvement of User Experience.

The data used in the work described in this report has been anonymized. We could not track any data to specific people, nor did we have access to any personal or sensitive information of the clients.

1.3. STRUCTURE OF THE PRESENT REPORT

The next chapter (Chapter 2 – theoretical framework) contains a theoretical summary of the methodology of algorithms used in this work (ARIMA, SARIMA, SARIMAX, GRU, LSTM). Further, chapter 3 (Chapter 3 – Related work) is dedicated to review the related work previously performed in a wide range of applications which make use of time series analysis methodology. This is followed by chapter 4 (Chapter 4 – Experiments and discussion) which describes the empirical part of this work, i.e. tested forecasting approaches and its results. Finally, the last chapter (Chapter 5) includes general conclusion, reflection on the limitations and suggestions for future work towards further improvement of the model.

2. THEORETICAL FRAMEWORK

2.1. TIME SERIES ANALYSIS

2.1.1. Introduction to Time Series

The term “Time series” relates to a data format consisting of the two main components: a time unit and a value or values associated with that particular time unit. What differs a time series from a standard dataset, time does not stand for just a metric, but it serves as a primary axis. A time series can be stored in two fundamental ways. One of them is to record a time series intervals as discrete points. These points can also represent other values which were measured for that specific timestamp and might occur in a periodic manner. Such time series are known as discrete time series. Financial or economical time series is a good example of a discrete time series, as usually various attributes are recorded on particular time intervals. Storing the values continuously along the time axis stands for the alternative way of recording a time series. Some examples of such time series include sensor data streamed from various Internet Of Things devices recording the data in a continuous manner.

2.1.2. Time Series Classification Types

Time series can be classified according to different attributes, one of which is a classification based on the stationarity. A feature which exhibits a change in mean, variance and time covariance is defined as stationarity. When it comes to the time series classification based on its stationarity, we can distinguish two categories:

- Stationary Time Series – A stationary time series has the property that the mean, variance and autocorrelation structure remain approximately constant over time. Stationarity can be defined in precise mathematical terms, but for this work purpose it can be considered as a flat looking series, not exhibiting any upward nor downward trend, with a constant and autocorrelation structure over time and with no periodic fluctuations (seasonality).
- Non-stationary Time Series - contrary to the stationary time series, it exhibits some non-flat patterns containing trend, seasonality, non-constant mean, variance or time covariance. In reality, great majority of time series are mostly categorized as non-stationary. Since a big number of time series techniques assumes stationarity of the data, some data pre-processing is often required before moving onto a forecasting phase.

Another kind of classification possible for a time series data is a classification based on a dependency between a new recorded value and its past values. Such classification results in two types of values:

- Long-Term memory time series – A typical characteristic of a long-term memory time series is slowly decreasing autocorrelation at consecutive lags in the autocorrelation function. In other words, it means that current values have high and significant correlations with a relatively large set of lags in the series. This property has been observed in both, financial series, as well as, stationary meteorological and environmental series, such as temperature change in the atmosphere, where today's day temperature can be reconstructed by a large change of historic date temperatures.
- Short-Tem memory time series – Unlike long-term memory time series, these exhibit a fast, exponential decrease in the autocorrelation function, meaning that correlation between current value falls dramatically fast in the successive lags of the time series. Some typical examples of short-term memory series include econometric processes.

2.1.3. Time Series Components

One of primary goals of a time series analysis is to detect trends and other repeating patterns that occur over time. After correct identification and removal of the pattern, the remainder of the data should appear as a random, stable process with a chance variation, i.e. it should comply with the concepts of the previously described time series stationarity. The search of these patterns can be accomplished by relatively sophisticated statistical analyses, however, simple time plots are often capable of revealing the underlying patterns.

A key to analyzing a time series is to understand the form of any underlying pattern of the data ordered over time. This pattern potentially consists of several different components, all of which combine to yield the observed values of the time series. A time series analysis can isolate each component and quantify the extent to which each component influences the form of the observed data. If one or more individual components of a time series are isolated and identified, a forecast can project the underlying pattern into the future. (David Gerbing, 2016).

The first component of a time series is known as a trend and takes a form of a long-term growth or decline. A trend can be linear or nonlinear, which has been visualized in the figure below.

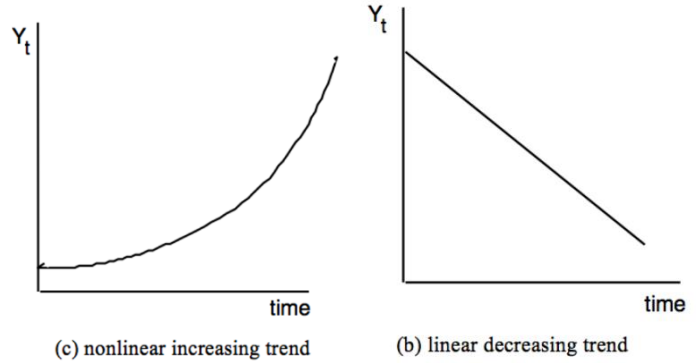

Figure 1: Trend Component of Time Series (David Gerbing, 2016)

The second component in the time series is called cyclical component. This pattern exists when data rises or falls do not happen over a fixed period. The duration of cyclical fluctuations is usually at least 2 years long. Cyclical pattern can be seen in a form of cyclical or long periodical rises and falls on a typical trend line. Due to the fact that this component is exhibited in as long time intervals as years or even decades, it is not common to see it in practical time series.

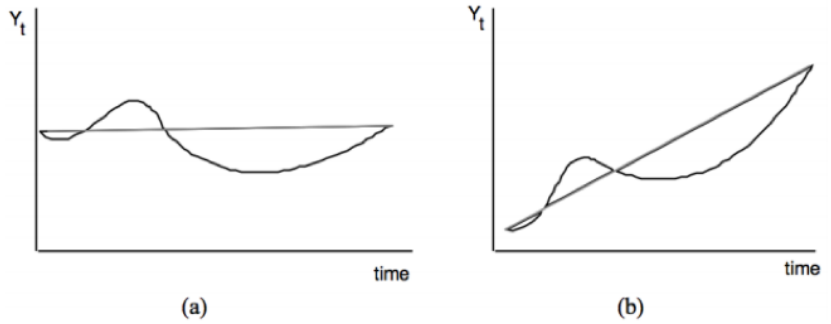

Figure 2: Cyclical Component of Time Series. (David Gerbing, 2016)

The third and last component of a time series is a seasonal component. This pattern can be seen in the form of short but repetitive fluctuations across the trend line. Seasonality is a relatively common aspect in the time series and can be often found in e.g. sales volume across time.

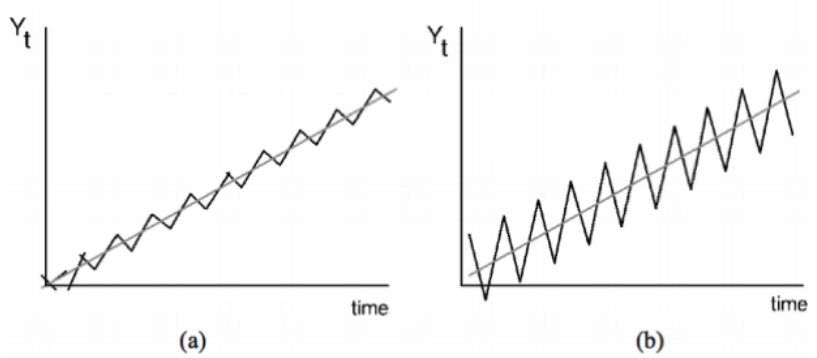

Figure 3: Seasonal Component of Time Series. (David Gerbing, 2016)

2.1.4. Types of Time Series Models

There are two commonly known types of time series models, which are differentiated according to the way that the time series components interact with each other:

- Additive models – Synthetically, these are the models, in which the effects of individual components are added together in order to model the underlying process. In other words, the behavior is linear and the values change consistently over time by the same amount, like e.g. linear trend. In such case, linear seasonality would imply same amplitude and frequency. This model can be represented by:

$$Y(t) = Trend + Seasonal\ Component + Cyclical\ Component + Error\ Component \quad (1)$$

- Multiplicative models – Intuitively, seasonal, cyclical and error components are multiplied, making one component impact another model. Contrary to additive model, the multiplicative model has either an increasing or decreasing amplitude over time. In other words, it is not linear but could be exponential or quadratic, represented by a curved line defined as:

$$Y(t) = Trend * Seasonal\ Component * Cyclical\ Component * Error\ Component \quad (2)$$

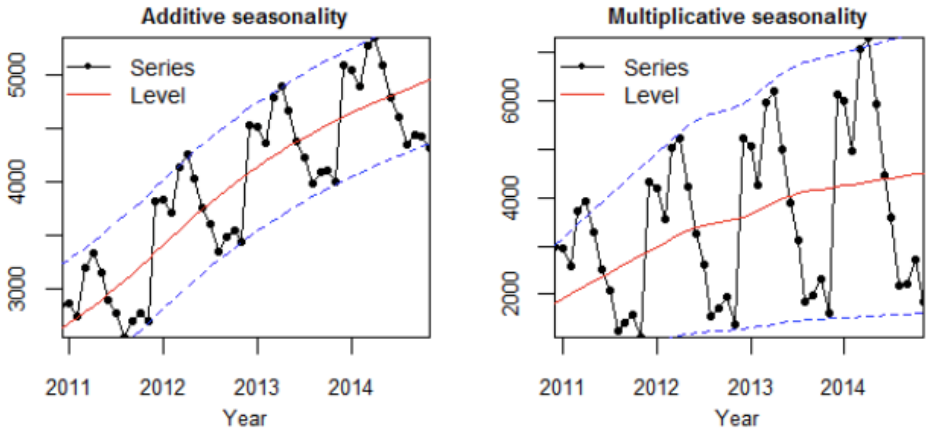

Figure 4: Additive and Multiplicative Seasonality. (Nikolaos Kourentzes, 2014)

In order to decide which type of model would better explain the underlying time series, we should analyze whether the variance of fluctuations is stable over time, indicating for an additive model, or else this variance is increasing across time, making a multiplicative model more appropriate.

2.1.5. Autocorrelation and Partial Autocorrelation

Correlation expresses the strength of a linear relationship between two quantitative variables. In case of time series analysis, which goal is to predict future values based on the past, we are interested in the correlation of current observation with the past observations at particular lags of the given time series. Value at lag k signifies a value k intervals apart from the current value. Serial correlation between current value and different lags of the time series is known as autocorrelation.

Autocorrelation plot (aka ACF) is usually used in order to identify the order of differencing needed for the data to become stationary, as well as, order of Moving Average component appropriate for modelling the underlying process. However, the nature of time series autocorrelation leads to a chaining effect, giving a false impression of strong correlation with greater lags. In order to tackle this problem, partial autocorrelation plot (aka PACF) is widely used in pair with ACF plot. Contrary to the autocorrelation plot, it identifies correlation not between present value and past lags, but rather between the residuals, which remain after removing the effects already explained by earlier lags. Looking at the PACF plot, we can see if there is any hidden information remaining in the residuals. If this is the case, it is advisable to keep that next lag as a feature while modelling, keeping in mind to limit the number of features to only the relevant ones in order to avoid the multicollinearity issues.

Below figure (*Figure 5*) represents a set of ACF and PACF plots for a sample series of data. The description of each scenario, as well as, possible interpretations of autoregressive and moving average terms has been presented below.

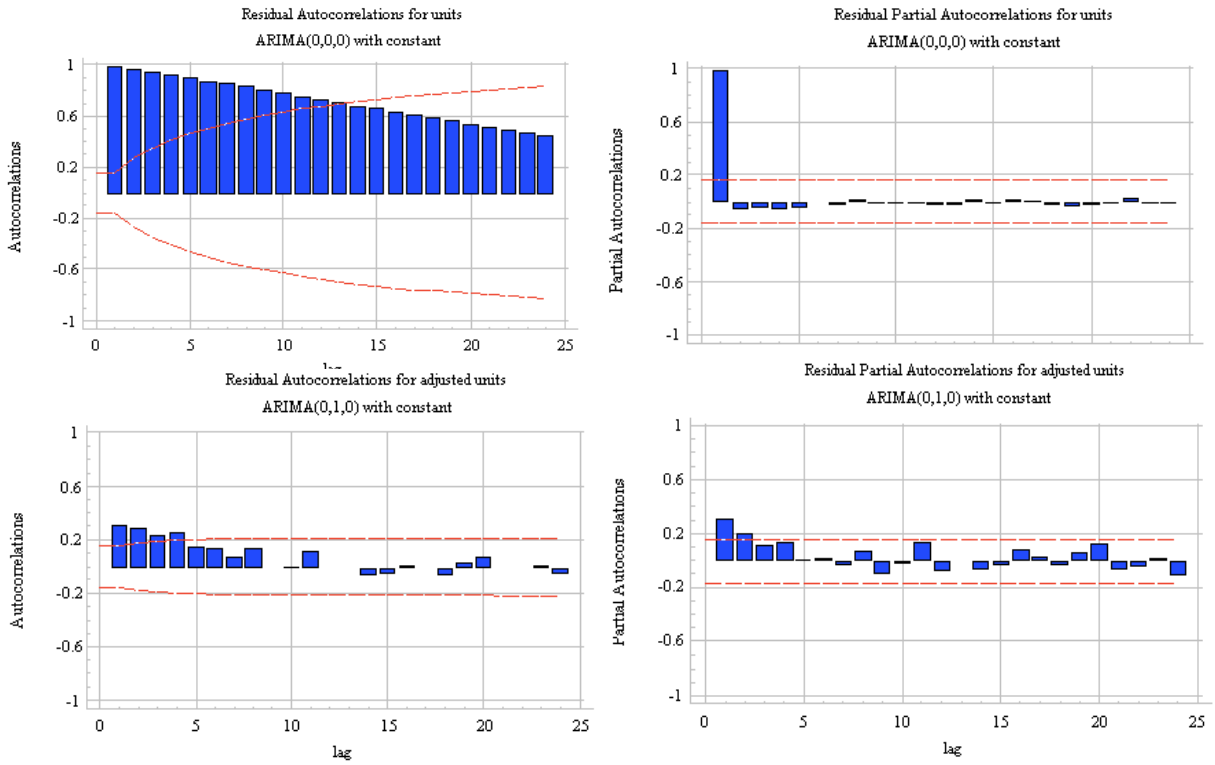

Figure 5: Examples of ACF and PACF plots and interpretations. (Robert Nau, 2019)

The visualization in the top left panel is an autocorrelation plot of the raw data. The autocorrelations look highly significant up to the 13th lag, which may be caused by the previously described propagation of autocorrelation at lag 1 (chaining effect). That is why, the top right panel of the Figure 5 represents a partial autocorrelation plot of the same series. Looking at the PACF plot, it is clear that lag-1 autocorrelation explains all remaining autocorrelations at the higher lags which are below the red line of significance level. This being said, it is safe to conclude that AR(1) model would be appropriate for the above series. Another possible approach to modelling above series is to perform a 1st order of non-seasonal differencing as a pre-processing step. The ACF and PACF plot of differenced data has been plotted in the bottom left and right panel of the *Figure 5*, respectively. Comparing the two graphs, it is clear that autocorrelations decay slower in the ACF plot, where all spikes up to 4 are significant, while in the PACF only the 2 initial spikes appear to be significant and then they shut off. This being said, the analysis of ACF and PACF plots on differenced data suggests modelling the series by means of the ARIMA(2,1,0) model, where 2 stands for the

autoregressive term, 1 indicated the order of differencing, and 0 means no moving average term used.

2.1.6. Types of Time Series Forecasting

Time series forecasting refers to predicting the future the most accurately as possible based on the information recorded at previous time steps. Based on the dimensionality of the analyzed data, Time Series Forecasting can be divided into two types:

- Univariate Modelling - Such forecasting methods use only the time and target value as inputs for modelling. An example of Univariate Modelling can be forecasting of the sales volume based on sales volumes recorded in the past.
- Multivariate Modelling – often a more efficient way of predicting a series of future values thanks to the advantage of using any additional information, such as knowledge of any future events which may impact the forecasts. Such models, where other than only time features are also used, can often make up for some fluctuations caused by some external factors, which could not be explained by regular trend, cycle or seasonal time components. An example of a multivariate modelling use-case can be a forecast of the air temperature based on not only air temperatures recorded in the past, but also, using the information about the rainfall, air humidity and the sunlight.

2.1.7. Forecast Error Metrics

The common goal of statistical predictions is to minimize the error of the predictions. The error can be defined as the difference between the forecast and its observed value. In time series, we should not understand the "error" as a mistake, but rather as a part of the observation, which is unpredictable. The formula for error calculation can be found below, where $\{y_{1,} \dots, y_T\}$ are the train data, while $\{y_{T+1}, y_{T+2}, \dots\}$ are the test data.

$$e_{T+h} = y_{T+h|T} - \hat{y}_{T+h|T} \tag{3}$$

The errors differ from previously described residuals in two aspects. First of all, while residuals are calculated on the training set, the error is calculated on the validation set. Additionally, forecast errors can involve multi-step forecasts, which is not the case for residuals, as these are based on one-step forecast only.

There is a number of forecast error metrics which vary in different kinds of information used to identify the prediction accuracy. These can be classified into 3 categories:

- Scale-dependent errors

  As the name suggests, scale-dependent errors cannot be used to compare between series involving different units, as they are scale dependent and expressed in some particular units. The most commonly used scale-dependent accuracy measures are based on absolute errors or squared errors, such as:

  1. Mean Absolute Error (MAE) – an easily interpretable metric which treats errors in absolute terms, i.e. error of 10 is treated twice as bad as error of 5.

$$MAE = \frac{1}{N}\sum_{t=1}^{N}\left|Z(t) - \hat{Z}(t)\right| \tag{4}$$

  2. Root Mean Squared Error (RMSE) – gives a higher weight to large errors, penalizing them when they are very undesirable.

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(Z(t) - \hat{Z}(t)\right)^2} \tag{5}$$

- Percentage errors

  Percentage errors solve the problem of being scale-dependent, making a comparison of various time series possible, regardless of the units expressed in the time series. On the other hand, percentage errors' downside is the fact that they are infinite or undefined if the true value that we are trying to predict equals zero. This being said, intermittent-demand data predictions should not be measured with the use of percentage errors due to zero demand values present in the series. Additionally, in case actual values oscillate around zero, the distribution of percentage errors can be very skewed.

  The two most commonly used percentage errors involve:

  1. Mean Absolute Percentage Error (MAPE) – an easily interpretable metric with a drawback of more heavy penalization on negative forecasting errors as compared to the positive forecasting errors.

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}\frac{\left|Z(t) - \hat{Z}(t)\right|}{Z(t)} \cdot 100\% \tag{6}$$

2. Symmetric Mean Absolute Percentage Error (sMAPE) – a modified version of MAPE, created in the M3-competition (Makridakis & Hibon, 2000). The motivation behind this metric is to cancel the heavy penalization on negative forecasting errors, which is present in the classic MAPE metric. The downside of sMAPE is the fact that it does not eliminate the problem of division by a number close to zero, as a result it may lead to infinite error values for certain predictions. Finally, it is negative sMAPE values are possible making the interpretation ambiguous.

$$SMAPE = \frac{1}{N}\sum_{t=1}^{N}\frac{\left|Z(t) - \hat{Z}(t)\right|}{\left(Z(t) + \hat{Z}(t)\right)} \cdot 200 \tag{7}$$

2.1.8. Cross-validation for Time Series

"It can not be emphasized enough that no claim whatsoever is being made in this paper that all algorithms are equivalent in practice in the real world. In particular no claim is being made that one should not use cross-validation in the real world." - Wolpert (1994a)

It is important to appropriately estimate the accuracy of a model not only to predict the future prediction accuracy but also for choosing the best of the analyzed models (aka model selection) (Wolpert). The goal is to create a model yielding predictions with low bias and low variance. In order to correctly assess the results of statistical analysis or model-generated predictions, we need to ensure that our model will generalize to an independent data set. One way to achieve this is by applying cross-validation methodology. The motivation to use cross-validation techniques is to verify how well the model will perform in practice, i.e. outside of a training set (in-sample data). Finally, cross-validation can also give us insights about potential underfitting (high bias, low variance) or overfitting (low bias, high variance) of the model.

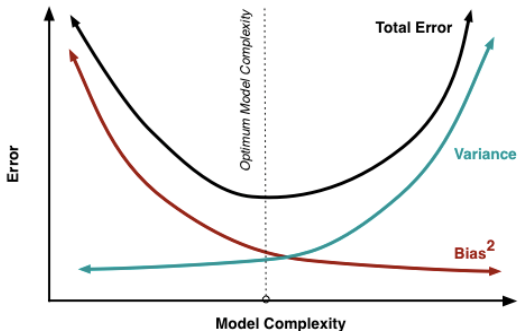

Figure 6: Model Complexity Bias-Variance Tradeoff. (Scott Fortmann-Roe, 2016)

The two most common cross-validation techniques involve leave p out and K-fold cross-validation. The former randomly selects p samples as the validation set, using the rest of the samples as the training set. The latter creates random K equal-size partitions of the data, each of which is in turn used as a validation set, while the remaining K-1 subsets constitute for the training set. Unfortunately, there is a number of problems arising from applying these methods for Time Series Predictions:

- Autocorrelation along the time axis is a common component of a time series data. An example of such autocorrelation is car traffic, which affects all drivers on the route across consecutive time points. The randomization will most likely lead to presence of strong correlation of data samples from validation set and from the training set. Such phenomena breaks the purpose of validation set, which should be previously unseen by the model. In such case, the model virtually 'knows' about the validation set apriori, leading to the artificially good prediction accuracy, which is a sign of overfitting.

- Time series is a time-ordered series of data, in which past observations are used to predict the future. Standard cross-validation techniques involve randomization, which does not preserve the time ordering. A negative side-effect of traditional cross-validation for time series would be generating predictions for some samples using a model trained on posterior data points.

In the answer to above issues, a new cross-validation approach has been created for Time Series Predictions. One such methodology is known as Walk-Forward Cross Validation. This cross-validation technique involves arranging the data from past to present and splitting it to k equal blocks of contiguous samples, deciding that first p blocks will constitute a train set. This way, the training set consists of the blocks from 1 to p, while block p+1 is the validation set. After, the splits successively shift to the right, making the following split's training set consist of blocks from 2 to p+1 and the validation set p+2, and so on. In this way, the 'walking forward' methodology leads to k-p splits. The visual representation of this methodology is presented at the Figure 7 below with the blue points representing training data and red points representing the validation data. For the final prediction accuracy, the chosen performance metric is averaged across the different folds. Of course, walk-forward cross validation allows to expand the validation period by more than one, i.e. it is possible to perform cross-validation

for weekly or monthly prediction (expanding the validation window by 7 or 30 data points, respectively and considering daily data).

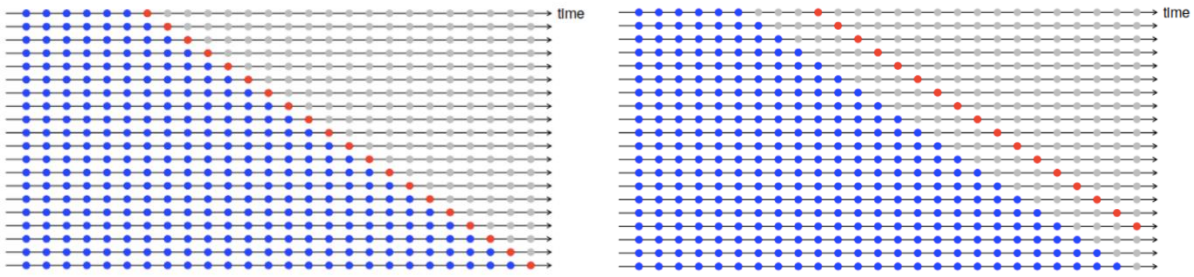

Figure 7: Walk forward cross-validation technique with window size of one and four (Hyndman, R.J., & Athanasopoulos, G., 2018)

2.1.9. Data Preprocessing

2.1.9.1. Data Transformation

The main focus of time series data preprocessing lays in the data transformation methods. The most commonly used four data transformation methods for time series predictions include:

- Power Transform – makes the distribution more similar to Gaussian by removing a shift from data. In a time series dataset, it can be seen as a stabilization of variance over time. Some common examples of power transform methods include square root, cube root and log transformations.

- Difference Transform – this transformation is commonly used in order to make the mean, variance and covariance of a time series constant across time, i.e. in other words, stationary which is a prerequisite of some popular forecasting models (e.g. ARIMA). The 1^{st} order differencing often helps to remove trend from a series of data, while k^{th} differencing on a series with seasonality of length k should help remove the seasonal fluctuations. Difference transform methods are usually applied iteratively until the time series becomes stationary.

- Standardization – makes the distribution of the data similar to Gaussian. Standardization primarily consists of subtracting the mean and then dividing the result by a standard deviation of the data sample. As a result, transformed data has a mean of 0 and a standard deviation of 1, making it a standard Gaussian distribution.

- Normalization – a very common in Machine Learning transformation method leading to a scale adjustment within some boundaries (usually <0,1>). Such transformed time series is easier to be predicted by a forecasting model and often provides a better accuracy.

2.1.9.2. Missing Values Imputation

Missing values in a time series can be a result of a natural process, e.g. in the in-store sales time series there is no sales recorded on public holidays. Introducing a dummy variable indicating whether a given day is a public holiday is a possible way to approach such problem, avoiding the sales underestimation on the first day after the public holiday and then overestimations in the following days. However, in real-life dataset, the missing values are often not the result of a natural process but appear to be rather random. In such cases, the cause of missing values may be simply a device malfunction which did not record the data when needed or a human error in case the data is entered manually. In such scenarios, we can either take advantage of the models which work flawlessly even with missing data (e.g. Naïve Forecast) or make use of missing values imputation methods.

One way the null values in a time series may be imputed is by taking the series of data after the last missing value and generate a forecast of the next missing value, in effect replacing it with that prediction. Alternatively, it is possible to use simple computation methods like mean, median or mode imputation. In case of time series, however, there is also some special imputation methods, which include: Last Observation Carried Forward (LOCF), Next Observation Carried Backward (NOCB) and Linear Interpolation. The latter imputation methods rely on the assumption that adjacent observations are similar to each other, thus they do not work well when this assumption is not satisfied, especially in the presence of strong seasonality.

2.1.9.3. Outlier Imputation

Outliers are defined as observations that are very different from the majority of time series' observations and can be seen on time series plots as sudden spikes or falls. As in the case of missing values, they can be simply erroneous data caused by e.g. a human error who entered the data manually or by a device malfunction. A common practice is to replace outliers with imputation methods described above in order to force the data to be more consistent with the majority of the series. One needs to be careful about replacing outliers without making sure that

these are in fact erroneous data, as they may often provide useful information about the underlying process which should be taken into consideration in the forecasting process.

2.2. TIME SERIES FORECASTING METHODS OVERVIEW

2.2.1. Evolution of time series analysis

In the nineteenth century early attempts to study time series were mostly based on the idea of a deterministic world until 1927, when the first major breakthrough in the area of time series forecasting took place essentially thanks to the contribution of Yule who introduced the notion of stochasticity in the time series. Simply put, Yule stated that every sequence of such data can be considered as the realization of a stochastic process, making this simple idea the base of further developed time series concepts over the years. More innovative forecasting concepts include the concept of autoregressive (AR) and moving average (MA) models formulated by Slutsky, Walker, Yaglom and Yule. Further, Kolmogorov (1941) proposed a solution to the linear forecasting problem relying on the idea of Wold's decomposition theorem. After that, a vast research about time series parameter estimation, identification, model diagnosis and forecasting has been developed until a crucial revelation in the area of time series development was included in the publication by Box & Jenkins (1970, 1976) which was a vital integration of so-far existing knowledge. The book, making an enormous impact on the theory and practice of modern time series analysis and forecasting is also known for introducing a concept of Box-Jenkins approach, which is essentially a coherent, versatile three-stage iterative cycle for time series identification, estimation and verification. Investigation of time series forecasting methodology is remaining active until today with new approaches and extensions are being tested.

2.2.2. Time series forecasting methods

Time series forecasting falls onto a category of quantitative forecasting methods and serves a purpose to predict the manner in which the sequence of observations will continue into the future, using the series of past data collected at regular intervals of time (hourly, daily, weekly etc.).

Since there is a number of time series forecasting methods and none can be objectively indicated as better than other, an appropriate model selection needs to be made. Before deciding on the model type to be used, one needs to take into consideration both, time and computation resources, the data available, the accuracy of the competing models, as well as, the manner in which the forecasting model will be used. It is important to notice that time series forecasting

methods vary significantly in complexity. Some time series forecasting methods are as simple as a Naïve Forecast, which simply put, sets the all forecasts to be equal to the value of proceeding observation. This method is appropriate for data, which follows a random walk, such as economic or financial time series data, and is also known as a Random Walk Forecast.

$$\hat{y}_{T+h|T} = y_T \tag{8}$$

Naïve Forecast can also be applied to a seasonal time series, where the forecast values are set as the last observed value of the same season of the year (e.g. the same day of previous week).

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)} \tag{9}$$

Finally, another variation of the simplest forecasting methods has been created and named as Drift Method. Drift forecast model allows the forecasts to fluctuate up and down over time and sets the drift (i.e. change over time) to the average change recorded in the historical data. These models do capture the trend component but fail to do so with the seasonal component.

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right) \tag{10}$$

In addition to the above extremely simple and surprisingly effective time series forecasting methods, more advanced and widely used nowadays methods will be described and studied in the sections below.

2.2.3. Autoregressive Models

Similarly to multiple linear regression, where variable of interest is forecasted with use of a linear combination of predictors, the autoregressive models (AR) uses a linear combination of past values of target variable in order to predict the next values in the series. Thus, the autoregression term indicating that it is a regression of the target variable against itself. This model is suitable for a time series not exhibiting any trend nor seasonal components (stationary time series). An autoregressive model of order p AR(p) can be written as:

$$\tilde{Z}_t = \phi_1\tilde{Z}_{t-1} + \phi_2\tilde{Z}_{t-2} + \cdots + \phi_p\tilde{Z}_{t-p} + a_t \tag{11}$$

, where a_t represents white noise, putting an assumption that any element in a time series can be represented as a random draw from a population distribution with constant variance and mean centered at 0.

Another possible notation for the AR(p) is following:

$$\phi(B)\tilde{Z}_t = a_t \tag{12}$$

with a backshift operator defined as:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p \tag{13}$$

The effect of backshift (aka Lag) operator is shifting the data back by one period. One single application of a backshift operator results in one backward shift of the data, while two such applications lead to shifting the data twice. In case the underlying process units are months and we wish to look at the data "same month last year", then the appropriate notation looks as following $B^{12}y_t=y_{t-12}$. Backshift operator is very useful when expressing the order of differencing applied in the time series data preprocessing phase.

Finally, the variance of a white noise process is estimated from the data. There is no one best method to determine the correct order of autoregressive term for underlying process. Obtaining the right model should be the effect of trial and error supported with the guidance of Autocorrelation and Partial Autocorrelation Plots analysis. Optionally, one can use built-in functions provided by many data analysis softwares, such as *auto.arima* in *pmdarima* Python package, which conduct a search of all possible models, returning the one with the highest accuracy according to AIC, AICc or BIC criteria. It is worth noticing that models returned by such built-in functions will not always result in the highest possible accuracy, as criteria used in the search penalize for a number of parameters in the model for a lower model complexity. Below figure (*Figure 8*) illustrates two examples of Autoregressive Models.

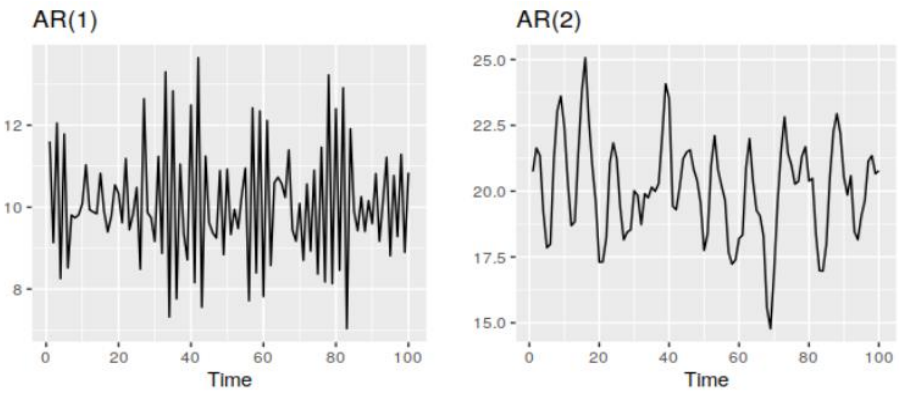

Figure 8: Examples of data from autoregressive models with different parameters. (Hyndman, R.J., & Athanasopoulos, G., 2018)

2.2.4. Moving Average Models

Moving Average Models differ from Autoregressive Models in a way that a target value is forecasted with a regression-like model using past errors in place of the past values. Similarly to the autoregressive model, it is suitable for a time series not exhibiting trend nor seasonal components (stationary time series). Moving average model (MA) of order q can be written as:

$$\tilde{Z}_t = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q} \tag{14}$$

, where a_t represents white noise. Each value of y_t can be defined as a weighted moving average error of the past few forecast errors. One should be careful not to confuse one of well-known time series decomposition methods, i.e. moving average smoothing with the moving average model, as they serve different purposes. While moving average smoothing is used to estimate the trend and cycle of past values, a moving average model is used for forecasting future values of a series. Alternative notation for a moving average MA(q) model is following:

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q \tag{15}$$

Similarly to the autoregressive models, the variance of white noise, as well as, the order of moving average term needs to be estimated from the data with analysis of Autocorrelation and Partial Autocorrelation plots, as well as, built-in statistical packages functions performing an automatic search for the best suited model. Below figure (*Figure 9*) illustrates two examples of Moving Average Models.

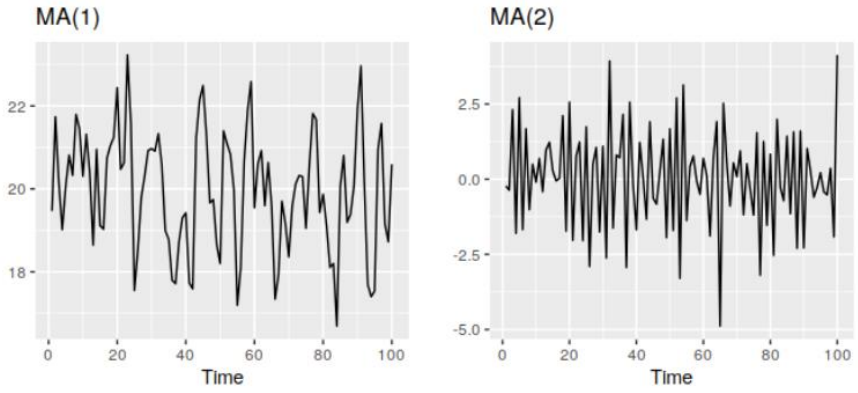

Figure 9: Examples of data from autoregressive models with different parameters(Hyndman, R.J., & Athanasopoulos, G., 2018)

2.2.5. ARIMA Models

Box and Jenkins proved in their publication of Time Series Analysis: Forecasting and Control (1970, 1976) that better prediction quality can be achieved by combining the autoregressive

with moving average models. In other words, ARIMA model uses a linear combination of the target variable's past values, as well as, past forecasting errors for a new value prediction. A common notation for such a combined model is ARMA(p,q), which consists of an autoregressive component of order *p*, as well as, moving average component of order *q*. Additionally, in case the difference transform method was applied in the data preprocessing phase in order to remove the non-stationarity, then such model is referred as ARIMA(p,d,q), *d* parameter expressing the degree of first differencing. ARIMA(p,d,q) can be written as following:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \mathcal{E}_{t-1} + \cdots + \theta_q \mathcal{E}_{t-q} + \mathcal{E}_t \quad (16)$$

, where y'_t is a differenced time series.

Backshift notation of an ARIMA(p,d,q) model is presented below:

$$\left(1 - \phi_1 B - \cdots - \phi_p B^p\right)(1 - B)^d y_t = c + \left(1 + \theta_1 B + \cdots + \theta_q B^q\right)\varepsilon_t \quad (17)$$

Last but not least, ARIMA works on the assumption of time series stationarity, thus should not be used unless this assumption is met. As in autoregressive and moving average models, appropriate values of *p*, *d* and *q* parameters need to be estimated by analyzing the autocorrelation and partial autocorrelation plots.

2.2.6. ARIMAX Models

Some processes can be best predicted when alongside the autoregressive and moving average terms, other exogenous variables are also used. ARIMAX is a modified version of the simple ARIMA, which is extended with a series of exogenous regressors with its coefficients. ARIMAX belongs to a family of multivariate models which tend to improve predictive accuracy by taking advantage of the information hidden in the valuable exogenous features. For instance, the air temperature may be predicted with use of solely time series, including autoregressive and moving average terms, however, the air temperature is also dependent on a series of other features, such as the rainfall or air humidity. Thus, this is a good example, where a series of temperature and rainfall records could be used as additional indicators to boost the model's accuracy at predicting the air temperature at future time steps. Backshift notation of ARIMAX model is presented below:

$$\phi_p(B)(1 - B)_{dy_t} = \beta_{TX_t}\theta_q(B)\epsilon_t \quad (18)$$

2.2.7. SARIMA Models

SARIMA is a special kind of ARIMA model which involves additional seasonal AR and/or MA terms necessary to capture the patterns in a wide range of seasonal data. For instance, in case of monthly time series, it is often the case to see some annual patterns that repeat every 12 months, such as, December every year is a peak shopping season. In such series, it is advisable to consider using the seasonal first order autoregressive model, which would take advantage of data at time step x_{t-12} (December a year ago) to predict x_t (next December). Apart from the seasonal AR terms, SARIMA can also include seasonal MA terms, which take advantage of errors at time with lags which are multiples of S (span of seasonality).

For ARIMA modelling of a stationary series with a seasonal component, the model notation takes a form of SARIMA(p,d,q)(P,D,Q)_m, where *p, d, q* stand for the same aspects as in case of a non-seasonal ARIMA model, i.e. non-seasonal AR terms, differencing order, MA terms. Seasonal component is then described with the use of the *P*, *D*, *Q* parameters determining the seasonal autoregressive term, seasonal differencing order and seasonal moving average term, respectively. Finally, *m* stands for the number of time steps for a single seasonal period.

Backshift notation of an SARIMA$(1,1,1)(1,1,1)_{12}$ is presented below:

$$(1-\phi_1 B)(1-\boldsymbol{\phi}_1 B^{12})(1-B)(1-B^{12})y_t = (1+\theta_1 B)(1+\boldsymbol{\theta}_1 B^{12}) \qquad (19)$$

2.2.8. ARFIMA Models

Yet another type of ARIMA model is used in the presence of long memory in the series. In such cases, the Autoregressive Fractionally Integrated Moving Average (ARFIMA) model is used. This model is essentially a generalization of ARIMA thanks to the possibility of setting the differencing parameter as non-integer. The ARFIMA (p,d,q) belongs to a class of long memory models and its main objective is to account for the long term correlations in the data. Long memory can be identified by analyzing the autocorrelation plot (ACF) and is detected when deviations from the long-run mean decay more slowly as compared to the exponential decay. In essence, autocorrelation function in long memory models decays hyperbolically, as opposed to the "short term" ARIMA models exhibiting a geometric decay.

Backshift notation of AFRIMA(p,d,q) model is presented below:

$$\phi B(1-B)^d(y_t-\mu)=\theta(B)_{\varepsilon_t}, \varepsilon_t \sim i.i.d.(0,\sigma_\epsilon^2) \quad (20)$$

, where $(1-B)^d$ is a fractional differencing operator.

2.2.9. Artificial Neural Networks

Classical time series forecasting models suffer from a few limitations, which can be handled by applying more complex yet often more effective methods, such as artificial neural networks. The main limitations of classical time series forecasting methods that can be handled by ANNs include:

- Classical time series models are not suitable for forecasting a series following non-linear pattern
- Classical time series forecasting methods can usually work on univariate data only with the exception of (S)ARIMA methods which can be extended to multivariate forecasting ((S)ARIMAX)
- Lack of support for missing values

Deep Learning can be thought of as the most powerful area of machine learning with a constantly growing interest among various industries. Deep learning solutions can be applied to a wide range of problems from classical regression and classification problems to pattern recognition and time series forecasting. The power of artificial neural networks lays in the fact that they are capable of solving non-linear problems thanks to the non-linear function(s) applied to the inputs as they are propagated across the consecutive network layers. Additionally, ANNs don't require feature selection as they perform it by themselves, setting weight values of particular neurons to nearly zero. Finally, they can solve a big variety of arbitrarily complex problems, searching for patterns in anything that can be translated into numbers, thus in pictures, sounds, and finally a series of data.

Of course, as any other method, neural networks also suffer from some drawbacks, such as a need for a big amount of data to train, long training time, as well as, it is considered a bit as a "black box", i.e. it is relatively hard to comprehend the highly complex processes happening inside of a deep neural network. Finally, fine-tuning deep neural networks is extremely time consuming as the training consumes a lot of time and there is a wide range of parameters to tune (optimizer, activation function, number of layers, number of hidden neurons, and more…). Last but not least, the data used in a training process of a neural network needs to be of a much larger volume as compared to the standard machine learning models training. This is caused by the fact that deep neural networks have a relatively larger complexity and a bigger

number of parameters to optimize, in result requiring a larger amount of training examples before it can learn the underlying patterns. Finally, a good point of reference for guidelines on deep neural networks architecture setup can be found in the extensive summary available in Zhang et al. (1998).

2.2.10. Recurrent Neural Networks

Recurrent neural networks are a special kind of neural networks, which are designed to work on supervised learning problems with the data of sequential nature. Some examples of such problems which can be solved by RNN are speech recognition, video processing or time series forecasting.

Recurrent neural networks are in fact neural networks with memory, i.e. they are capable of remembering things from the past, which is essential when dealing with time-dependent problems. To produce accurate time series forecasts, RNNs use not only the input data but also the previous forecast values. Since simple passing values forward in time is not efficient enough, recurrent neural networks make use of some memory mechanism which helps the algorithm to filter to only the useful bits of data stored in memory and pass them to make a new prediction. The mechanism of memory in a recurrent neural network is used in order to imitate how the human brain works, i.e. the forecast of the next value in the sequence of data is not made from scratch every time a new information is passed into a model, however, a set of values which were previously passed is remembered, adding some extra context to the currently transferred piece of information. A recurrent neural network is an advancement of a traditional neural network, making it possible to model sequence data not only thanks to the memory unit providing context to the information inputs, but also giving a flexibility in terms of the inputs and outputs dimensions, i.e. RNNs are able to capture sequence-to-sequence relations with inputs and outputs in the form of sequences of vectors. Thanks to this capability, RNNs can easily provide a series of predictions of any length, being fed with multivariate, sequential inputs, which in turn can feed into the algorithm additional information in the form of exogenous variables (aka multivariate modelling). Additionally, recurrent neural networks have cyclical connections over time. During RNN training, an internal, maintained throughout the network state is constantly updated with activation functions, in turn providing a memory to the network. RNNs consist of a cycle besides the general information propagation existing in a traditional NN and thanks to this cycle information is not lost but passed forward across training steps.

The functioning of Recurrent Neural Network has been presented in the figure (*Figure 10*) below.

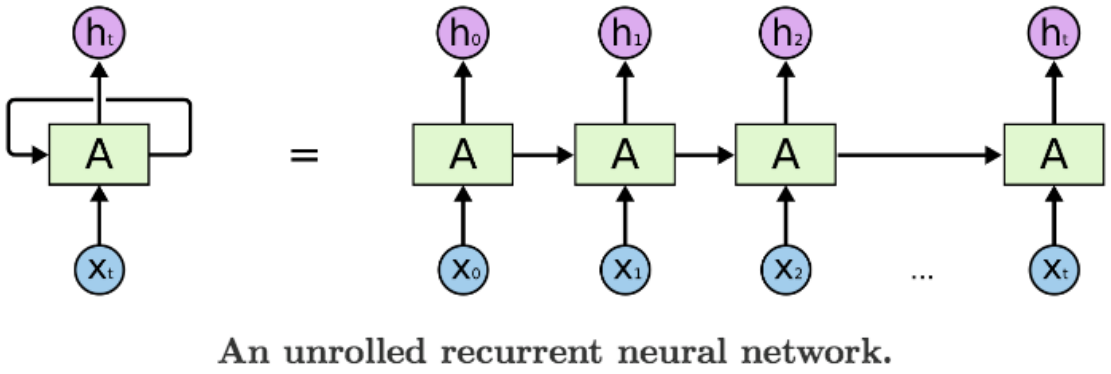

Figure 10: Recurrent Neural Network Loop (Christopher, 2015)

A traditional recurrent neural network can also be thought of as a list of copies of the same network, each of which is passing a message to a successor. Or, it can be also considered as a fully connected neural network if we spread out or unroll the time axis. (Kondratenko et al, 2003). In such way, each step of the RNN takes X as an input and returns h as an output, at the same time passing some information to the successor. It is crucial to mention that returned by each RNN's component output's content is not only impacted by the just-fed in input X but also by the whole history of inputs previously fed onto a network. The chain-like architecture of RNNs makes them a natural choice for lists and sequences of data, such as time series.

Even though traditional RNNs have been very successful in a wide range of applications requiring sequential data, the downside of classical RNNs is that they are not so good at handling long-term dependencies. For example, given a simple RNN a sequence of words "The clouds are in the…", it does not require a broader context and is pretty obvious based on only recently passed information that predicted value of the sequence will be a word sky. However, in case the input sentences are longer, such as "I grew up in Germany (…). I speak fluent…", the RNN indeed needs a broader context and needs to look further in the past to find out that predicted word should be German. This is an example of a challenge which a classical RNN will most likely fail to solve, as the gap between relevant information and currently predicted value is too large. Indeed, RNNs become unable to learn to connect relevant information as that gap grows. Theoretically speaking, RNNs should have the capability to capture such long-term dependencies, however, Hochreiter (1991) and Bengio, et al. (1994) proved that this is not the case. This problem is known as a vanishing gradients problem and occurs when long sequences of data need to be learnt. The core of vanishing gradients problem lays in the fact that the longer

the data sequence is, the further the gradients need to carry the information used in the RNN parameter update, making these updates become extremely small, in effect not learning at all. According to (Xavier et al 2011) ReLU activation function slows down the vanishing gradient problem, as compared to the sigmoid function.

2.2.11. Long Short-Term Memory Network

Long Short Term Memory networks are a special kind of recurrent neural networks, which can deal better with the gradient vanishing problem, making LSTMs capable of learning long-term dependencies. Hochreiter & Schmidhuber (1997) were the ones to introduce and popularize this algorithm. LSTMs are known for their outstanding accuracy, when its parameters are carefully set-up. Additionally, remembering long-term dependencies is something that they are cut out for.

The structure of this powerful algorithm has been presented below:

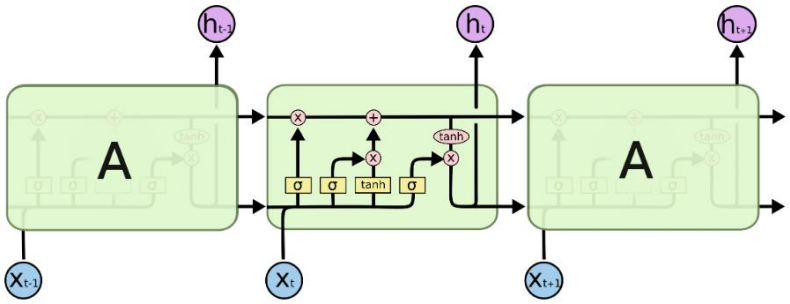

Figure 11: LSTM Memory Cell (Christopher, 2015)

LSTMs, just like standard RNNs, are in the form of a chain of repeating modules. What differs them however, is the structure of these modules, as instead of having a single neural network layer in each as classical RNNs do, they consist of four layers interacting with each other in a precise way. The horizontal line running across above part of the diagram presents the core idea behind LSTM and is known as the cell state. The main role of the cell state is to propagate the information, leaving it unchanged, adding relevant and removing irrelevant pieces of information which is structured by the gates. The forget gates are composed of a sigmoid layer and a point-wise multiplication operation. The output of the sigmoid layer is a value between zero and one, determining how much information should be let through (and forgotten) into the cell state. The next step is the input gate, which is essentially a hidden layer of sigmoid activated nodes, outputting values between 0 and 1 that are able to control the amount of input which will flow into the unit. The tanh layer creates a vector of new candidate values which can be added

to the state. Finally, the output gate determines with the use of a sigmoid function which values of the state will serve as the output from the cell.

The flexibility of LSTM is provided thanks the number of gating functions controlling what needs to be forgotten and remembered by the network. However, as Machine Learning is known to be a science of trade-offs, here is no exception made, and thus the flexibility of LSTM also comes with the cost of a big number of parameters that need to be fine-tuned for the network to perform efficiently, making it computationally expensive to train.

2.2.12. Gated Recurrent Units

A gated recurrent unit has been introduced by Cho, et al. (2014) as a newer generation of RNNs. It works similarly to LSTMs with two main differences: instead of a cell state, GRU uses a hidden state in order to propagate the information. Additionally, GRU has only two gates: a reset and update gate. Gated recurrent unit has been introduced as an alternative to LSTMs, which would be still able to avoid the vanishing gradient problem, but would not have such a big number of parameters to tune and would be less computationally complex. In addition, according to the research of Tang, et al. (2017), GRU is more robust to noise as compared to LSTM and it tends to outperform LSTM in several applications.

The GRU unit architecture has been presented in the diagram below:

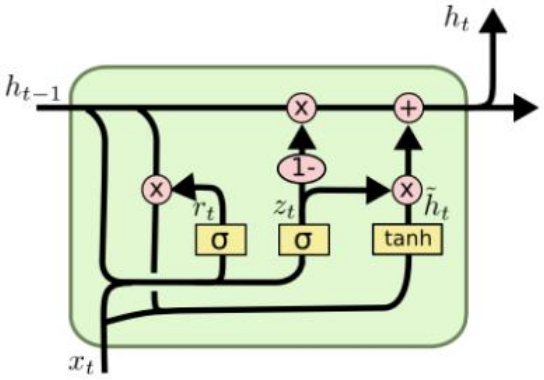

Figure 12: Gated Recurrent Unit Cell (Christopher, 2015)

The update gate in a GRU unit serves a similar purpose as the forget and input gates of an LSTM, i.e. it controls the amount of past information that needs to be passed to the next step, as well as, what new information to add. The reset gate on the other hand is used to determine the amount of past information to forget. Due to the fact that GRU unit consists of only two gates, it is faster to train as compared to LSTM.

3. RELATED WORK

3.1. POSSIBLE TIME SERIES APPLICATIONS

The analysis of experimental data recorded at different points in time requires a special approach as it leads to new and unique problems in the field of statistical modelling and inference. This is mainly due to the nature of time series breaks the assumption of many conventional statistical methods, which rely on the fact that the observations are independent and identically distributed. The below-described list of fields where time series problems may arise is a proof of the impact that time series analysis may have on diverse scientific applications. One of the most common time series applications can be found in the field of economics, where daily stock market quotations or monthly employment rates need to be analyzed and predicted. Other examples of time series analyses' applications include socio-demographic series, such as school enrollments, birth rates, gender or fertility tendencies. In medicine, on the other hand, analysis of blood pressure measurements over time might be useful for evaluating drugs used in treating hypertension. Time series in combination with computer vision in the form of a series of magnetic resonance imaging can be effective for analysis of the brain's reaction to certain stimuli under various experimental conditions. Finally, a number of sophisticated applications of time series methodologies have been practiced in the environmental and physical sciences. This being said, the monthly sunspot numbers are one of the earliest recorded series studied by Schuster (1906). Environmental sciences rely on time series also in more modern investigations, such as global temperature measurements and analysis of global warming evolution. Speech series modelling is another important area where time series analysis is used and it makes the transmission of voice recordings efficient. Rainfall and air temperature may also be predicted based on geophysical time series. Finally, seismic recordings over time can be useful to differentiate the earthquakes from nuclear explosions. Above listed examples of time series analysis applications are just a few of all that are being used nowadays.

3.2. TIME SERIES COMPARATIVE STUDIES

There is a number of studies focusing on comparison of traditional time series analysis methodology with the methods from the family of Artificial Neural Networks. With the use of a dataset from the famous "M-3 competition" Foster, et.al. (1991) proved that deep neural networks are inferior to the least squares statistical models for a yearly tie series data. The results of studies conducted by Sharda and Patil (1992) and Tang, et.al. (1991) argued that for

a large number of observations, ANN models and Box-Jenkins models deliver similarly good results. Kang (1991) observed in his study comparing 18 different neural network architectures that forecast errors tend to be smaller for the series which contain the trend and seasonal component. Additionally, according to Kang the neural networks often outperform the simple statistical time series forecasting models in the long horizon forecast periods. Another study comparing neural networks to the traditional forecasting methods have been performed by Hill, et.al. (1996) who compared his forecasts to those obtained by Makridakis, et.al. (1982) and concluded that ANNs do better for monthly and quarterly series. Another conclusion driven by their study is that fewer network parameters to be estimated is crucial to successful neural network modelling. Kohzadi, et.al. (1996) is the owner of a very controversial statement, that he has made based on his empirical investigation aiming at comparing the forecasting performance of ANNs to ARIMA model: "The neural network with only one hidden layer can precisely and satifactorily approximate any continuous function" (Kohzadi, et.al., 1996, p.179). Kohzadi, et.al. (1996) has also concluded that ANNs are more efficient at catching the turning points of the series when compared to ARIMA model. However, some of the studies conducted by Adya and Collopy (1998) based on 48 articles published between 1988 and 1994 resulted in contradicting to Kohzadi's statements, indicating that real-world data may not always be consistent with theoretical inferences.

Even though a big part of the time series literature is focused on modelling time series data and comparing the performance of various competing models, Gorr, et.al. (1994) used cross section data to predict the values of student grade point averages and compared the predictive power of a linear regression and a stepwise polynomial regression versus the ANN. Based on these studies, they have concluded that the mean errors of forecasts generated by different models were not comparable as none of the models was significantly superior compared to the others. This result has been justified by the selection ANN structure. However, it can be also argued that another reason for the unsatisfactory results delivered by the neural network is the fact that qualitative binary variables have been used in 6 of the models. In such cases, ANN learning algorithm may not work well due to the large numerical distinction in the observations between the binary and continuous variables.

The effect of a time series stationarity on the forecasting performance of the statistical and deep learning models has been tested by Lachtermacher and Fuller (1995) in their study of a series of annual river flows. Lachtermacher and Fuller followed Box-Jenkins methodology to produce an ARIMA model, as well as, the ANN model. They have concluded that the ANN

delivered better prediction results as compared to the ARIMA model. Additionally, the forecast improvement was much higher for the models trained on non-stationary data, meaning that stationary data is essential for good quality traditional forecasting models. This is consistent with the study of of Tang, et.al. (1991). Nelson, et.al. (1999) focused on emphasizing the effect of seasonal component on the time series forecasting accuracy. Comparison of ANN with Box-Jenkins methodology forecasts made in this study indicated that in case of a seasonal time series, ANN forecasts can deliver better quality forecasts when trained on de-seasonalized data as compared to the model trained on the raw, non de-seasonalized data.

A number of studies on time series predictions of macroeconomic phenomena have proved to be problematic to capture by linear models, which is caused by the non-linear nature of macroeconomic relations. Maasoumi, et.al. (1994) constructed a Neural Network to predict US macroeconomic series, such as GDP, wages, Price Index or unemployment rate. The downside of this study is lack of comparison between different models and as a result it has been only interpreted that the sample results fails to show the prediction power in the underlying study. Swanson and White (1997) delivered comparably more clear conclusions in their study where they have applied an ANN model to forecast 9 macroeconomic series, comparing its predictive power to traditional forecasting methods. The results of this study are not clear cut, however, Swanson and White do conclude that ANN models have a good forecast performance as compared to the traditional approaches even if there is no clear evidence of nonlinearity in the data. Moshiri and Cameron (2000) on the other hand proved in their comparative studies of linear and non-linear inflation rate forecasting approaches that artificial neural networks perform equally well as econometric models but they tend to outperform the latter ones in some cases.

4. EXPERIMENTS AND DISCUSSION

This section of the report is dedicated to present the empirical work, which goal is to develop time series forecasting models and compare their performance to find the most optimal modelling approach.

To start with statistical modelling, Box Jenkins methodology was used to identify an optimal ARIMA model. This approach leads to a linear prediction on a univariate time series. Secondly, the ARIMA model has been extended to capture the seasonality component leading to a development of a SARIMA model. Finally, another attempt to increase the model accuracy has been made by adding a set of exogenous variables which could boost the predictions' accuracy. This way, SARIMAX model has been produced, which is the first multivariate modelling case studied in below dissertation.

After statistical modelling, deep learning methods from the family of recurrent neural networks have been explored, such as Gated Recurrent Unit and Long-Short Term Memory models. In these methods, only multivariate modelling approach has been tested due to the fact that neural networks are able to perform an automatic feature selection by setting the appropriate weights to the connections between the neurons. Finally, different architectures of neural networks have been tested and their impact on the predictive power was presented.

According to the No Free Lunch Theorem any forecasting model might work the best for our particular problem, thus all of above forecasting methods have been compared and one model with the best predictive power has been selected as the final result.

4.1. DATASET EXPLANATION

After the research of possible opportunities for modelling within the company and ideas prioritization alongside the business analysts, prediction of call center arrivals in response to the press-published advertisements has been selected as the goal of this project. The first fundamental step was concerning the access of the data. The call center data is of incidence-based structure, i.e. each time a customer was contacting the call center, a new data point was recorded in a database stored in a SQL server (event-driven data). The data concerning the inbound phone calls is distributed across multiple tables, thus the first goal is to consolidate and export all needed information into one dataset, which would be easily fed onto forecasting models afterwards. Additionally, the data regarding the strategy details of press advertisements, including newspaper titles, where ads would be published, ad formats, the costs associated with ad placement, etc. was stored in a comma-separated values file (.csv), thus it also needed to be

aggregated with the data about incoming phone calls to limit the various database connections needed to fetch the data onto the models.

For the first, incidence-based dataset regarding phone call arrivals a set of data cleaning actions were necessary. First of all, the data has been filtered to include only the phone calls incoming as dialed to press-published phone numbers, indicating that they were related to press marketing campaigns. Each such successfully collected by an agent phone call was treated as one lead attracted by the ad. Then, the records marked as fraud phone calls, as well as phone calls with erroneous call start and end timestamps (call picked up later than ended) have been filtered out not to introduce noise onto the model. Finally, phone calls which were outbound (call-backs made by call center agents), never picked up by a call center agent or interrupted for some reason were also filtered out as a result of business decision and company's definition of a lead, which is internally considered as a unique inbound phone call, collected by an agent and correctly recorded by the system. After the data cleaning process, in order to keep the uniformity of a time index across the dataset, a stored procedure in SQL server was used to import the time series data aggregated to a daily incidence dataset on a uniform time index. This exported from SQL server data constituted the first dataset used for modelling, which was univariate and only contained the timestamp along with the number of phone call arrivals on that day which were related to press-published advertisements.

The other dataset prepared for forecasting included incidence-based data related to exogenous variables used for multivariate forecasting. More specifically, this dataset contained information about marketing strategic plans for press-published advertisements, such as the timestamp of when the ad is planned to be published, the ad format, the cost associated with the ad publishing, the name of the publisher, as well as, the newspaper name. Firstly, a business decision has been made to select the relevant features which can affect the press-published ads' efficiency, which in turn translate into a bigger number of phone calls received by a call center. Secondly, since time series forecasting can be framed as a supervised learning problem in a way that whichever series of training data is provided to the algorithm for training, it also requires to be provided with the future values of these training features for the model to predict on further time steps. This being said, in case of a monthly univariate model prediction, we only need to pass it a set of following thirty time stamps so the model knows what to predict on. In case of multivariate forecasting, however, we need to be able to provide the algorithm with a series of future values of all exogenous variables used in the model training phase. That is why, some features included in the press marketing plans .csv file needed to be excluded from a final dataset used for modelling, as they are not planned as early as 40 days in advance, which is a

prerequisite for that model taking into account the business requirement to deliver inbound press-related phone calls forecasts for the next 40 days. Finally, after taking into consideration above feature exclusions, which were mostly the results of business decisions, the multivariate dataset has been aggregated to the daily timestamp level (it was possible for more than one ad to be published on one day). Each time step is assigned with the following set of values: an average number of ads published, average cost spent on ad publishing, average ad format, the number of ads published by specific publisher and the number of ads published in a specific newspaper.

To conclude, the data extraction process led to a creation of two daily-aggregated datasets, the univariate one about call center inbound press-published ads related phone calls, and a multivariate one, with information about published ads on specific days. Both datasets are based on uniform time series indexes and cover the time period between September 2017 and October 2019, making up 810 data points in total.

4.2. DATA EXPLORATION

The first step taken in the data exploration phase is to visualize the target variable in its original form, as received from the company and try to understand the hidden insights or patterns in it. The below figure presents the time plot of collected call center arrivals regarding the press-published ads (internally considered as Press Leads).

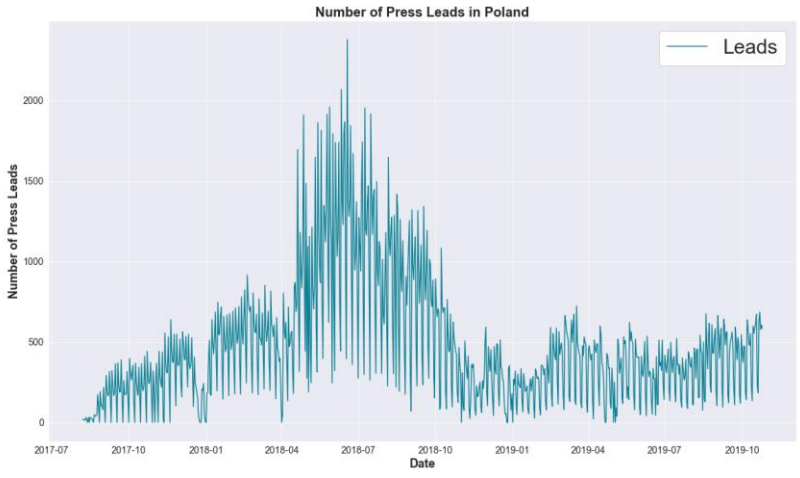

Figure 13: Univariate Dataset Distribution.

Above visualization of call center press leads across the years 2017-2019 suggests that the data is not evenly distributed along the timeline and some clear trends are visible, as well as, seasonal patterns. Due to these clear fluctuations, we can state that the underlying data is not a white noise process, as its mean and variance are not constant over time. This being said, we assume

that the time series is not stationary in its raw form and will most likely require a set of data transformations to become stationary and comply with the necessary prerequisites required by ARIMA models. Further testing and transforming of the data into its stationary form is described in the sections below.

4.2.1. Time series stationarity

Stationarity of a time series have been tested in three steps, starting with seasonal decomposition of a time series to visualize trend, seasonality and the residual components existing in the underlying process. Further, a set of data transformations were applied as an attempt to make the residuals more stationary. After that, rolling statistics of the time series were plotted to verify whether the mean and standard deviation of the series is constant over time. Finally, statistical tests were used to verify the hypothesis whether the time series satisfies the stationarity conditions.

4.2.1.1. Time series decomposition

To start with, for a more thorough picture of existing tendencies in the data, a seasonal decomposition of the time series has been plotted and presented below:

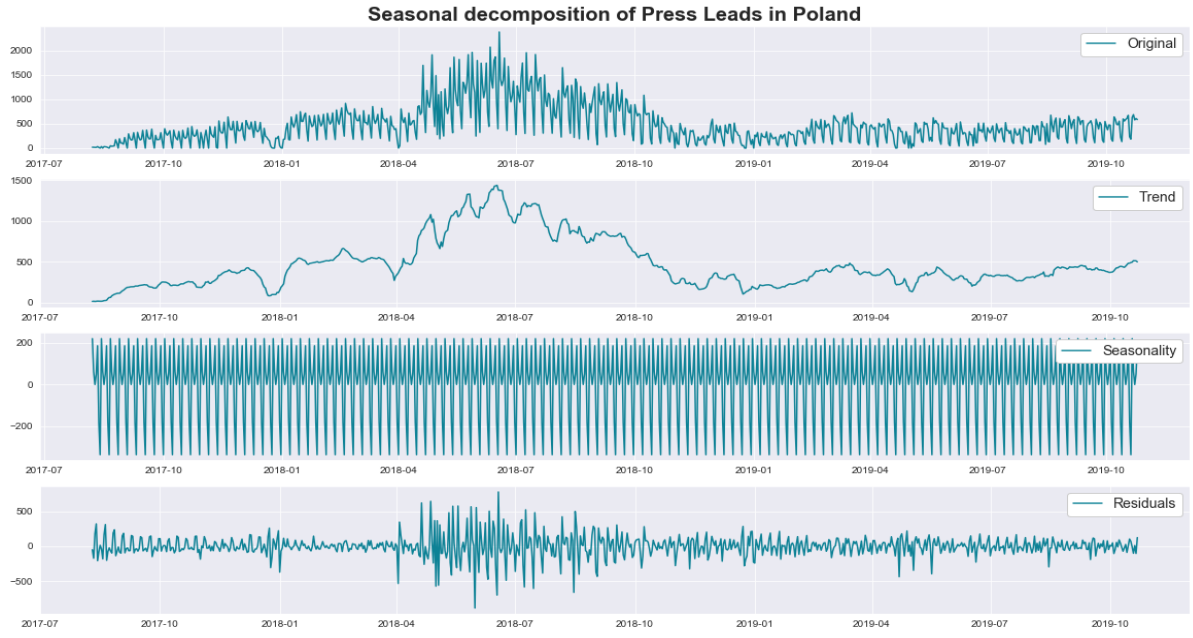

Figure 14: Time Series Component Decomposition.

Above visualization of a univariate time series data presents raw series, as well as, 3 of its components individually: trend component, seasonal component, random error component. This way, it gives a clearer view on each of them, often providing valuable insights which impact the underlying process. Looking at above-presented seasonal decomposition of call

center's press leads, we can deduct that there is a nonlinear trend component, which has an upward direction until July 2018 and then it goes downward until January 2019, staying approximately constant or slightly upward all the way until October 2019. Another insight provided by the above plot is a strong seasonal component of order 7 present in the data. Finally, the residual component should be centered around zero and have a constant variance for it to be independent of time and stochastic in nature. As long as the former condition is met, as we can see that mean of the residuals is centered at zero, ensuring unbiased forecasts, we cannot be so confident about meeting the latter condition due to the less stable variance in the period between April and September of 2018. As an attempt to reduce observed residuals' heteroscedasticity, following data transformation methods have been tried out with resulting residuals across time plots presented below:

- Residuals of original time series (Benchmark)

Figure 15: Residuals of original time series.

- Residuals of time series after Log Transformation log(x+1)

Figure 16: Residuals of log-transformed time series.

- Residuals of time series after Square Root Transformation

Figure 17: Residuals of Square Root-transformed time series.

- Residuals of time series after Box Cox Transformation (lambda = 0.416259)

Figure 18: Residuals of Box Cox-transformed time series.

Based on the above plots, as long as there is no clear winner among the tested data transformation methods yielding perfectly constant variance, log transformation looks promising enough with the residuals looking the smoothest compared to the other above-presented transformed series. This being said, the validity of log transformation for analyzed series will be verified in the next sections both, visually and statistically by plotting rolling statistics and performing statistical tests on log-transformed data.

4.2.1.2. Rolling statistics

The second approach to test the stationarity of the data involved plotting rolling statistics, i.e. rolling standard deviation and rolling mean of the series.

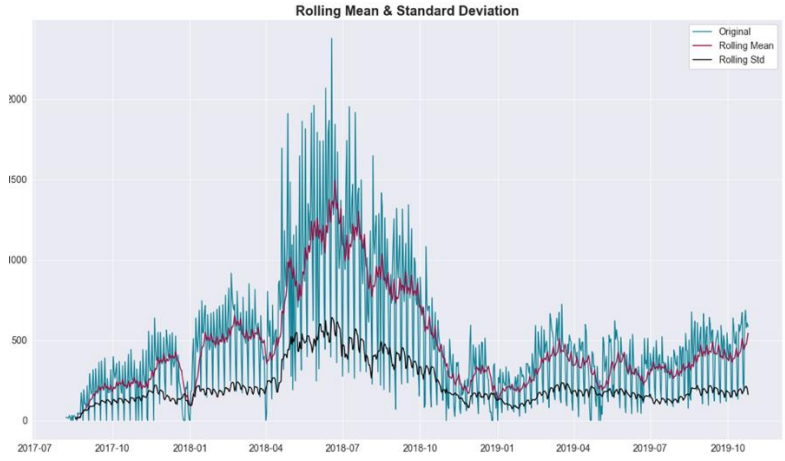

Figure 19: Rolling Mean and Standard Deviation over raw data series.

Above plot provides three proofs of the time series being unstationary: clear trend in the data, non-constant rolling mean and finally non-constant rolling standard deviation of the data. One common way to tackle the problem of non-stationarity caused by a trend present in the data is applying a difference transform, which should help stabilize the mean of a time series across time, in result 'removing' the trend from a series. Another transformation applied to the data as an attempt to make the data more stationary by stabilizing its variance is the log transformation.

Below plot presents rolling statistics of a series after the 1st order differencing and log transformation applied.

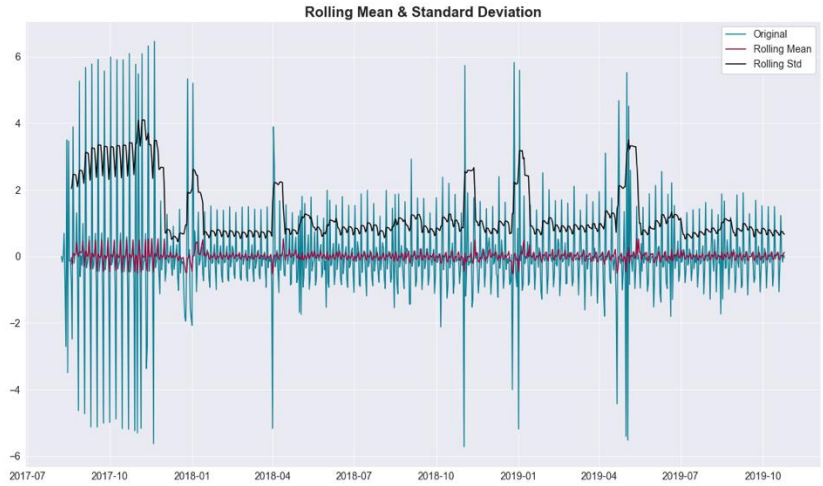

Figure 20: Rolling Mean and Standard Deviation over transformed data series.

First of all, it can be clearly seen that 1st differenced time series can be considered trend-stationary as its rolling mean is now centered at 0 across time. For variance stability, it can be observed that log transformation managed to flatten the rolling standard deviation especially in the period between April and October 2018, in which raw data's variance was the most unstable. However, the side effect of a log transformation is variance instability in the last months of the year 2017. One way to go around it would be to drop this part of the series, however, taking into account the fact that we are provided with only 2 years of data, we decided to keep this data for further analysis at the price of having a less stable variance. For our case, the unstable variance is not of a big concern in terms of validity of further data modelling, as the variance is constant across majority of the series. Last but not least, non-constant variance should not affect the accuracy of the forecasts itself which is the main goal of this project but it may only affect the interpretability of confidence intervals (Hyndman, R.J., & Athanasopoulos, G., 2018).

4.2.1.3. Statistical tests

Finally, the last approach determining whether the time series is stationary involves statistical testing of a hypothesis posed in the Dickey-Fuller Test (DF Test). The test was developed in 1979 by David Dickey and Wayne Fuller and involved testing whether a series contains unit root feature severally impacting a statistical reference. In other words, Dickey and Fuller's goal was to determine how strongly a time series can be defined by a trend. Since the DF test is the simplest method to detect unit root, we decided to use augmented version of it, being aware that

many practical datasets are usually too complex. The main difference between the augmented Dickey-Fuller test and its traditional version is the fact that ADF test is used for larger and more complex time series models. The test defines null hypothesis as that the time series can be represented by a unit root, meaning it is not stationary as it has some time-dependent structure. The alternate hypothesis states that the time series cannot be represented by a unit root, hence can be considered stationary. Finally, the ADF statistic indicates the strength of the rejection of the null hypothesis, i.e. the more negative it is, the more confident we can be admitting that the series is stationary.

Below table represents the results of Augmented Dickey-Fuller test on original time series, on 1st difference transformed data and on 1st and 7th differenced data to make up for the trend and seasonality present in the series.

Original data	1st differenced data	1st and 7th differenced data
ADF Statistic: -1.772618 p-value: 0.394138 Critical Values: 1%: -3.439 5%: -2.865 10%: -2.569	ADF Statistic: -8.562380 p-value: 0.000000 Critical Values: 1%: -3.439 5%: -2.865 10%: -2.569	ADF Statistic: -10.906058 p-value: 0.000000 Critical Values: 1%: -3.439 5%: -2.865 10%: -2.569

Table 1: Results of Dickey-Fuller test on raw data.

Finally, since the visual methods did not provide a clear answer on whether a log transformation helps to stabilize the variance in the analyzed series, we decided to perform statistical tests also for log-transformed data, which results are presented in the table below.

Log-transformed data	1st differenced log-transformed data	1st and 7th differenced and log-transformed data
ADF Statistic: -3.625567 p-value: 0.005291 Critical Values: 1%: -3.439 5%: -2.865 10%: -2.569	ADF Statistic: -9.267603 p-value: 0.000000 Critical Values: 1%: -3.439 5%: -2.865 10%: -2.569	ADF Statistic: -10.056810 p-value: 0.000000 Critical Values: 1%: -3.439 5%: -2.865 10%: -2.569

Table 2: Results of Dickey-Fuller test on log-transformed data.

Above tables indicate that p-value yielded by the test performed on the original series is higher than the threshold value (0.05), which concludes that raw time series is not stationary neither in its original form nor after log transformation. Test results for only 1st differenced and both, 1st and 7th differenced data have p-values lower than the threshold value indicating the stationarity of these time series in case of both, original and log-transformed series. ADF

Statistic, however, is the most negative for the raw time series, where both difference transform methods were applied, making them the right choices for our data. To conclude, raw data with both seasonal and first difference methods applied yields the most negative ADF statistic of -10. 906058, thus is selected as the most appropriate and complying with stationarity assumption for ARIMA models developed in the next section.

4.3. ARIMA UNIVARIATE MODELLING

Box-Jenkins methodology has been followed in the below sections of the report in order to obtain a reliable statistical model. According to the 5th edition of the Time Series Analysis: Forecasting and Control (2016), stochastic model building is an iterative approach that consists of the 3 following steps:

1. Model Identification – Analyze the data and its visual representation to select a subclass of a model which may best summarize the data.
2. Model Estimation - Use the parameter selection techniques to determine the parameters of the model.
3. Diagnostic Checking - Evaluate the fitted model and its residuals to stay reassured that overfitting is not present and no information is remaining in the residual errors.

4.3.1. Model identification

The next step in ARIMA forecasting, after ensuring about the stationarity of a time series is selection of parameters for the model. The ARIMA forecasting methods takes a form of a linear equation depending on three parameters (p, d, q):

1. Number of AR terms (p) – autoregressive terms indicating the lags of dependent variable. Lagged values of y_t serve as predictor variables, e.g. if p is 4, the predictors used for $x_{(t)}$ will include lagged values of $x_{(t-1)}$, $x_{(t-2)}…x_{(t-4)}$
2. Number of MA terms (q) – moving average terms describe the values of weighted moving average past forecast errors. If *q* equals 4, the predictors used for $x_{(t)}$ will include weighted values of $e_{(t-1)}$, $e_{(t-2)}$, …$e_{(t-4)}$ with $e_{(i)}$ being weighted forecast error at time *i*.
3. Order of differencing (d) – the number of times the series underwent 1st (non-seasonal) differencing. Differencing is an iterative process and may require more than one differencing for the series to become stationary.

For ARIMA modelling of a stationary series with a seasonal component, the model notation takes a form of SARIMA(p,d,q)(P,D,Q)_m, where p, d, q stand for the same aspects

as in case of a non-seasonal ARIMA model, i.e. non-seasonal AR terms, differencing order, MA terms. Seasonal component is then described with the use of the P, D, Q parameters determining the seasonal autoregressive term, seasonal differencing order and seasonal moving average term, respectively. Finally, m stands for the number of time steps for a single seasonal period.

To determine appropriate values of above parameters, Autocorrelation Plot needs to be plotted in pair with Partial Autocorrelation Plots, which has been done and presented in the figure below (*Figure 21*).

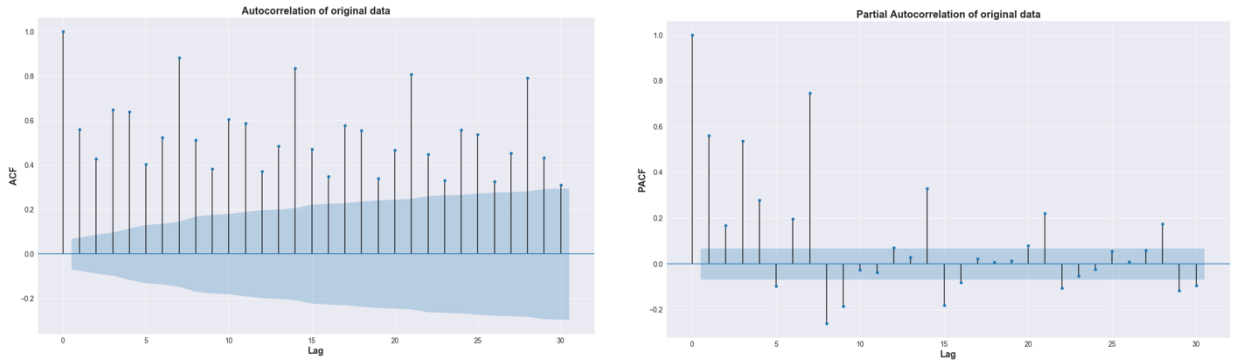

Figure 21: Autocorrelation and Partial Autocorrelation plots.

In the above ACF Plot it can be observed that the time series process exhibits a long memory. This phenomena can be proved by the fact that autocorrelation at successive lags decays much slower as compared to the exponential decay, making autocorrelation at high lags significant, revealing the existence of a long memory. Time series with long term memory require special modelling approaches, such as ARFIMA (Autoregressive Fractionally Integrated Moving Average) or ARMA(1,1) which is known for approximating a long memory feature existing in the time series. Another insight possible to be drawn from slowly decreasing lags in the ACF Plot is the need to apply 1st order differencing onto the data. Additionally, clearly visible spikes at every 7th lag on both, ACF and PACF plots indicate for a clear seasonal pattern of order 7 present in the data.

Following these insights, ACF and PACF plots have been plotted for data after taking its 1st order difference and a seasonal difference of order 7.

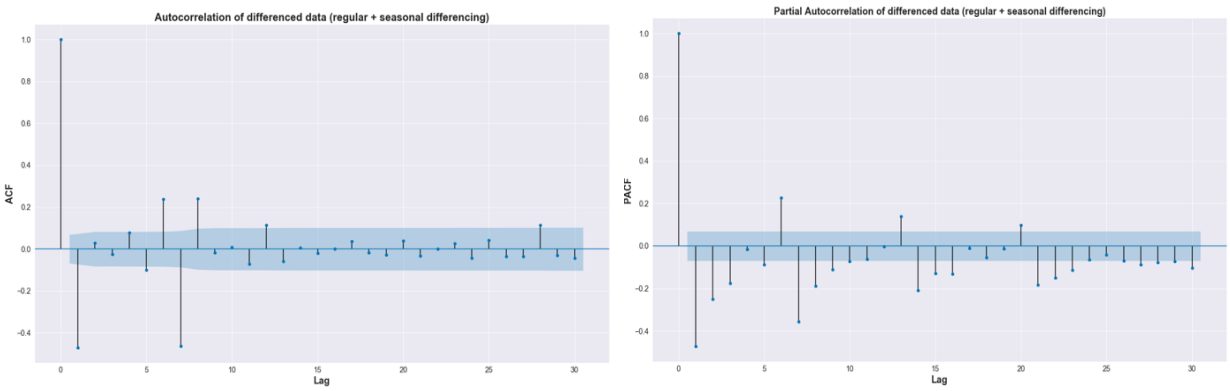

Figure 22: Autocorrelation and Partial Autocorrelation plots on 1st order differenced data.

Above ACF and PACF plots reveal that non-seasonal and seasonal differencing removed the long memory aspect from the time series. Additionally, PACF's linearly decaying lags and visible on the ACF plot's highly significant spikes at 1st and 7th lag indicate that MA(1) and seasonal MA(1) components should be included in the ARIMA model. Due to no abrupt shut-offs noted on the PACF plot, visual parameter selection method does not suggest to use neither regular nor seasonal autoregressive components for that series. To conclude, analysis of autocorrelation and partial autocorrelation plots results in a suggestion for a model of SARIMA(0,1,1)(0,1,1)_7.

4.3.2. Model evaluation

Results of presented models were generated in Python programming language and were supported by standard python libraries, such as *scikitlearn, statsmodels, pmdarima* and more. The forecast window consists of 40 days for two reasons: firstly, due to the business requirement, secondly, further work will include multivariate forecasting including variables from media planning that cover the period of following 40 days and we would like to generate forecasts for the same period length for them to be comparable.

The first tested model incorporates the parameters selected in the previous section of the report, i.e. it is SARIMA(0,1,1)(0,1,1)_7. The summary of this model can be seen on the figure below.

Statespace Model Results

Dep. Variable:	count	No. Observations:	780
Model:	SARIMAX(0, 1, 1)x(0, 1, 1, 7)	Log Likelihood	-4914.970
Date:	wto, 14 sty 2020	AIC	9835.940
Time:	14:37:07	BIC	9849.852
Sample:	08-07-2017 - 09-25-2019	HQIC	9841.297
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.7011	0.021	-33.047	0.000	-0.743	-0.660
ma.S.L7	-0.6679	0.018	-36.187	0.000	-0.704	-0.632
sigma2	2.3e+04	492.283	46.729	0.000	2.2e+04	2.4e+04

Ljung-Box (Q):	73.08	Jarque-Bera (JB):	3031.25
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	0.41	Skew:	-0.45
Prob(H) (two-sided):	0.00	Kurtosis:	12.72

Figure 23: Manually defined model summary.

Above summary indicates that association between the target variable and both, seasonal and non-seasonal moving average terms is significant due to the p-values associated with respective terms, which are lower than the threshold of 5 %. The Akaike Information Criterion (AIC) score is a measure of how well the model fits the data avoiding overfitting by penalizing overly complex models. Even though the value of AIC does not say much, it is widely used to compare competing models and selecting the one with the lowest AIC score. In above-tested model, the AIC is equal to 9835.940.

Next, we will check if *auto.arima* function from *pmdarima* Python package which performs an automatic search for the best possible model is able to find a better model with lower value of AIC.

Statespace Model Results

Dep. Variable:	y	No. Observations:	780
Model:	SARIMAX(1, 1, 2)x(1, 0, 1, 7)	Log Likelihood	-4941.976
Date:	wto, 14 sty 2020	AIC	9921.951
Time:	15:44:29	BIC	10010.454
Sample:	0 - 780	HQIC	9955.993
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-5.0052	2.229	-2.245	0.025	-9.374	-0.636
x1	-0.0006	0.002	-0.264	0.792	-0.005	0.004
x2	2.5405	20.376	0.125	0.901	-37.395	42.476
x3	1.8347	19.984	0.092	0.927	-37.333	41.002
x4	67.5487	31.119	2.171	0.030	6.556	128.542
x5	22.4295	9.269	2.420	0.016	4.263	40.596
x6	147.0905	32.380	4.543	0.000	83.627	210.554
x7	267.4216	40.054	6.677	0.000	188.918	345.925
x8	118.9391	26.116	4.554	0.000	67.753	170.125
x9	48.0306	13.469	3.566	0.000	21.632	74.430
x10	-22.3581	32.849	-0.681	0.496	-86.740	42.024
x11	-217.8960	76.465	-2.850	0.004	-367.764	-68.028
x12	-39.6245	20.177	-1.964	0.050	-79.170	-0.078
ar.L1	-0.8712	0.289	-3.013	0.003	-1.438	-0.304
ma.L1	0.1978	0.280	0.706	0.480	-0.351	0.747
ma.L2	-0.6068	0.185	-3.275	0.001	-0.970	-0.244
ar.S.L7	0.8736	0.035	24.648	0.000	0.804	0.943
ma.S.L7	-0.5140	0.053	-9.741	0.000	-0.617	-0.411
sigma2	2.407e+04	1164.788	20.669	0.000	2.18e+04	2.64e+04

Ljung-Box (Q):	63.03	Jarque-Bera (JB):	2492.23
Prob(Q):	0.01	Prob(JB):	0.00
Heteroskedasticity (H):	0.78	Skew:	-0.22
Prob(H) (two-sided):	0.04	Kurtosis:	11.75

Figure 24: Automatically defined model summary.

The *auto.arima* function returned a SARIMA(1,1,2)(1,0,1)_7 as an optimal model. However, since the yielded AIC score is higher than of the manually defined model, meaning it has a worse balance between its ability to fit the data set and avoid overfitting, we are going to stick to the simpler and better performing model, i.e. SARIMA(0,1,1)(0,1,1)_7.

4.3.3. Model diagnostics

The last part of the Box-Jenkins approach involves diagnostics of the model's residual errors. Residuals generated by an ideal model should resemble a white noise process, i.e. Gaussian distribution with a mean zero and a symmetrical variance. Additionally, residuals should be independent of time. Below figure represents a set of generated by residuals' visualizations which should help to diagnose the forecasting errors.

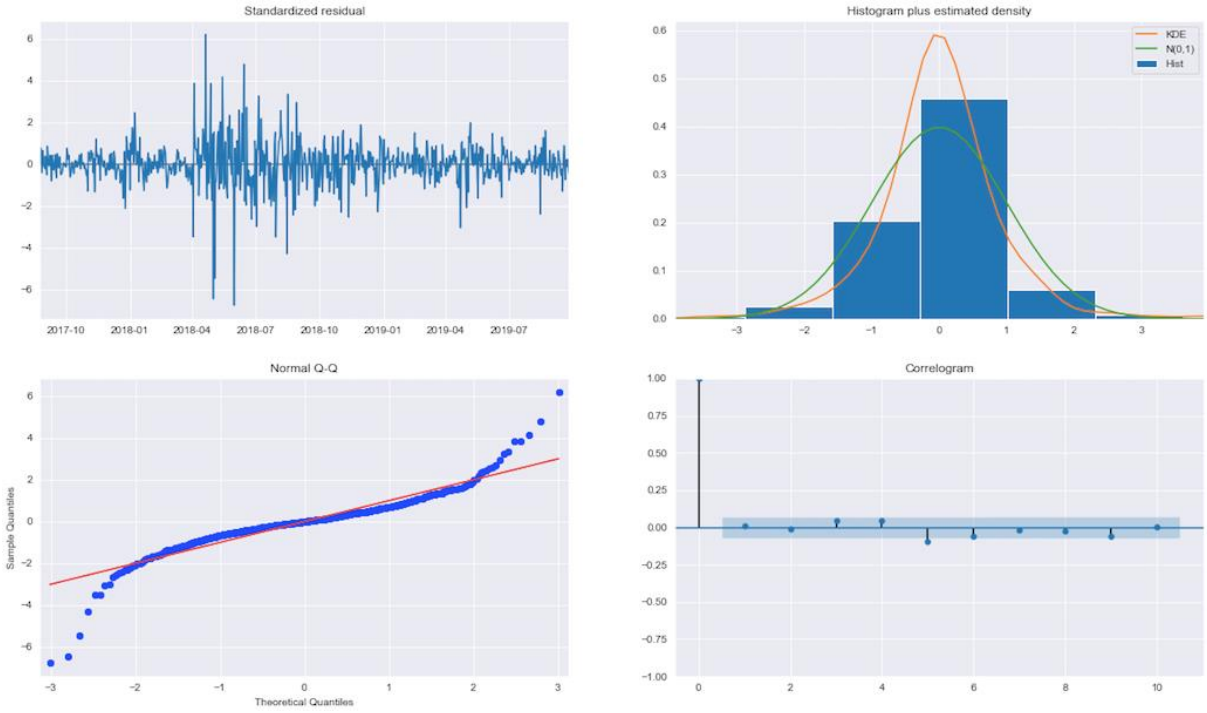

Figure 25: Model diagnostics plots.

The first plot of standardized residuals informs about the mean of residuals being independent of time and centered at 0, meaning that there is no need to correct for a bias in the forecasts provided by the model. Histogram and the density plot of residuals prove that the error distribution is similar to normal apart from the fact that the distribution is slightly skewed to the left. Moving onto a Normal QQ-plot, great majority of points lays on the red line, which is desired to conclude that the data follows a normal distribution. Finally, the last chart plotted in the bottom right part of above figure represents a correlogram with no significant

autocorrelation at successive lags. As a result, it can be concluded that residuals are independent of time.

4.4. SARIMAX MULTIVARIATE MODELLING

The other statistical model studied in this report is known as Autoregressive Integrated Moving Average with Exogenous Inputs (ARIMAX). ARIMAX works similarly to the traditional ARIMA model but it additionally takes the impact of covariates on the forecasting into account, which often leads to better accuracy of the predictions. Given that we were provided with selected media planning data regarding the advertisements published in press, we will take advantage of them and incorporate these as new set of time-dependent vector hoping for better prediction accuracy as compared to the traditional ARIMA model developed in the previous section.

4.4.1. Multivariate dataset

The dataset provided by the Media Planning Team includes information about ads being published in the press across time. This data has been transformed, aggregated and enriched with feature engineering methods to extract more information about the data. Categorical variables have been transformed into dummy variables and only the dummy variables with at least 5% of positive values were kept in order to limit the number of variables in the dataset. This was necessary for instance in the case of *Press* string variable which described the name of the newspaper where ads are published, as we had over 100 unique newspaper names, a lot of these were sparse and had very few publications. Another data preprocessing step involved imputation of null values, which was done by means of mean imputation and mode imputation for numerical and categorical variables, respectively. The dataset did not contain any erroneous values, thus no outlier imputation was needed. Finally, the pre-processed dataset used for multivariate modelling consists of following series of variables:

- *Unique_reflinkno* – Number of unique ads published in press
- *Avg_cost* – The average budget spent on ads publishing in press
- *Avg_*format – The average ad format of ads published in press
- *Pub-1* – The number of ads published at Publisher Pub-1
- *Pub-2* - The number of ads published at Publisher Pub-2
- *Pub-3* - The number of ads published at Publisher Pub-3
- *Press-1* – The number of ads published in the newspaper of Press-1

- *Press-2* – The number of ads published in the newspaper of Press-2
- *Press-3* - The number of ads published in the newspaper of Press-3
- *Press-4* – The number of ads published in the newspaper of Press-4
- *Press-5* – The number of ads published in the newspaper of Press-5
- *Press-6* - The number of ads published in the newspaper of Press-6
- *Press-7* – The number of ads published in the newspaper of Press-7
- *Press-8* – The number of ads published in the newspaper of Press-8
- *Press-9* – The number of ads published in the newspaper of Press-9
- *Dayofweek-1* – Dummy variable indicating whether the day is Sunday
- *Dayofweek-2* – Dummy variable indicating whether the day is Monday
- *Dayofweek-3* – Dummy variable indicating whether the day is Tuesday
- *Dayofweek-4* – Dummy variable indicating whether the day is Wednesday
- *Dayofweek-5* – Dummy variable indicating whether the day is Thursday
- Dayofweek-6 – Dummy variable indicating whether the day is Friday
- *Is-holiday* – Dummy variable indicating whether the day is holiday
- *Next_is_holiday* – Dummy variable indicating whether the following day is holiday
- *Count* – The number of call center collected inbound phone calls related to press-published ads

4.4.2. Feature selection

The goal of the feature selection process is to identify the variables that can best describe the target variable. Efficient feature selection can improve the predictive power of the model thanks to the elimination of irrelevant, noisy variables from the model. Additionally, feature selection is advised in order to reduce the dataset dimensionality and avoid producing computationally complex models. Some algorithms, such as Artificial Neural Networks are able to perform the feature selection work by themselves, however, this is not the case for statistical time series models. The basic type of feature selection has already been applied in the preprocessing step, where after one-hot encoding categorical variables, one of the generated dummy features has been dropped in order to avoid the phenomena known as the Curse of Dimensionality, which takes place when the dataset contains too many sparse features.

For media planning dataset, backward elimination method for feature selection was used. Gini feature importance index serves as a criterion to select the most relevant features for our multivariate modelling problem. Gini Importance is one of the impurity measures used in the Decision Tree machine learning models. It is in fact a criterion to minimize the probability

of misclassification, meaning that smaller values of Gini index mean that a given variable is a better regressor or classifier of the target variable. Gini feature importance has been widely used as a feature selection method in various machine learning applications and so has been applied in our modelling problem.

Below figure illustrates features sorted according to their importance.

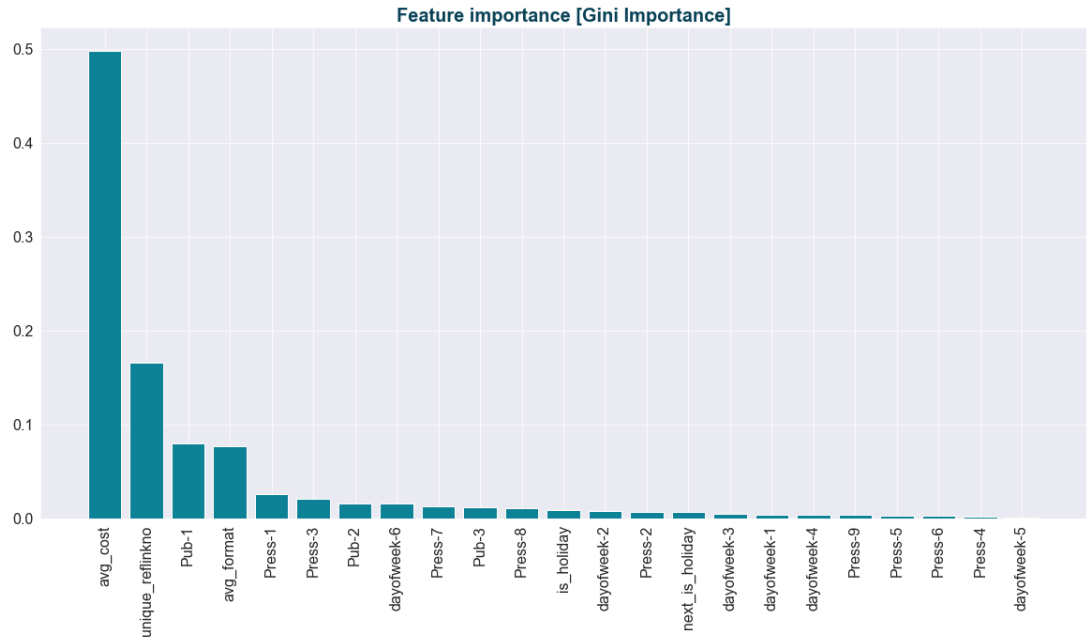

Figure 26: Gini feature importance bar plot.

Above figure indicates that the variables with the highest feature importance include the average cost spent on press advertising on a particular day (*avg_cost*), the number of ads published on that day (*unique_reflinkno*), the number of ads published by a publisher Pub-1 (*Pub-1*) and the average ad format (*avg_format*). These results make an intuitive sense, as the number of collected press-published ads phone calls incoming to the call center is usually the highest on the first day when the ad was published, as customers usually buy them right on the release day. Also, more money spent on ads publishing is also contributing to the higher number of press leads. Regarding the least relevant variables, these include the dummy variable indicating that it is Thursday (*dayofweek-5*) and some other dummy variables indicating particular newspaper names (e.g. *Press-4*, *Press-6*).

4.4.3. SARIMAX results

In this section the cross-validated results of SARIMAX modelling will be presented and compared to the model which is currently used by the company (Company Model). The goal is to take advantage of conducted feature importance analysis by using backward elimination

methodology, i.e. we will start with fitting a SARIMAX model including all exogenous variables. After that, we will be gradually excluding the least relevant variables from the model until we finally get to the univariate model, which uses solely lagged values and weighted prediction errors to predict future values. The seasonal and non-seasonal parameters will be defined with the use of *auto.arima* function which performs an automatic search of the most optimal model. Finally, the prediction accuracies achieved by each of the models will be compared to the currently used by the company model and the most accurate setup will be selected.

N most relevant exogenous variables	Optimal model	AIC	Cross – validated error measures	
			MAE	RMSE
23	**SARIMAX(1, 1, 2)x(1, 0, 1)_7**	**9845.121**	**75.494785**	**96.286265**
22	SARIMAX(4, 1, 2)x(2, 0, 2)_7	9832.958	80.704501	103.979423
21	SARIMAX(4, 1, 2)x(1, 0, 2)_7	9824.091	82.584269	107.880913
20	SARIMAX(1, 1, 2)x(1, 0, 1)_7	9888.515	80.007072	103.860881
19	SARIMAX(1, 1, 2)x(1, 0, 1)_7	9882.835	82.507558	106.638797
18	SARIMAX(2, 1, 3)x(2, 0, 1)_7	9885.122	81.209798	107.252855
17	SARIMAX(4, 1, 2)x(2, 0, 2)_7	9840.809	75.026542	94.816101
16	SARIMAX(1, 1, 2)x(2, 0, 2)_7	9901.299	82.707534	101.035266
15	SARIMAX(4, 1, 2)x(2, 0, 2)_7	9834.998	83.052508	101.217615
14	SARIMAX(4, 1, 2)x(2, 0, 2)_7	9828.445	84.081256	102.038029
13	SARIMAX(1, 1, 2)x(1, 0, 1)_7	9911.960	79.931227	98.958813
12	SARIMAX(1, 1, 2)x(1, 0, 1)_7	9888.145	84.262227	102.190261
11	SARIMAX(4, 1, 2)x(1, 0, 1)_7	9932.508	90.461700	108.738352
10	SARIMAX(4, 1, 2)x(1, 0, 1)_7	9910.672	86.130049	103.662428
9	SARIMAX(1, 1, 2)x(1, 0, 1)_7	9905.604	83.638324	99.998627
8	SARIMAX(1, 1, 2)x(1, 0, 1)_7	9943.249	83.916326	101.801737
7	SARIMAX(1, 1, 2)x(1, 0, 1)_7	10065.717	90.927261	109.477274
6	SARIMAX(0, 1, 2)x(1, 0, 1)_7	9998.221	91.297980	110.687680
5	SARIMAX(4, 1, 2)x(2, 0, 2)_7	9943.999	91.909677	111.396810
4	SARIMAX(1, 1, 2)x(2, 0, 2)_7	10065.666	96.898702	116.399103
3	SARIMAX(3, 1, 3)x(1, 0, 1)_7	10036.464	95.735298	115.717307
2	SARIMAX(2, 1, 5)x(1, 0, 1)_7	10024.631	95.786078	114.633758
1	SARIMAX(2, 1, 5)x(1, 0, 1)_7	10104.690	95.276864	114.155127
0	SARIMAX(0, 1, 1)x(0, 1, 1)_7	9835.940	82.208944	99.940257
Company Model	**Average of 40 proceeding values**	**-**	**150.033760**	**169.917893**

Table 3: Backward elimination results for SARIMAX vs Base Model.

Above table shows values of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for models with top 23, 22, 21… 0 most relevant according to the gini index exogenous values. The values of error metrics were obtained by walk-forward cross-validation method, as such each of the models was trained and tested on a 40-day prediction interval 4 times. After that, the final values of MAE and RMSE were calculated as an average of the 4 rounds of

testing, which was further recorded into the above table. Looking at above results, we can conclude that all of above tested SARIMA models outperform the currently used by the company model, which simply estimates the following 40 daily inbound phone call volume to be equal to the average calculated over 40 previous time steps. Above results suggest that it is best to use all available exogenous features, as such model generates predictions with a MAE of 75 Press Leads, resulting in a significant improvement of predictions accuracy, decreasing the prediction error by 50% as compared to the Company Model. Regarding the feature selection aspect, the conclusion possible to be drawn from above table is that not using exogenous features as model indicators at all decreases the prediction accuracy by only 7 units as compared to the multivariate model.

To visualize the ability to capture seasonal patterns by univariate vs multivariate model, true values and predictions generated by the former and the latter method have been illustrated in the figure below *(Figure 27*).

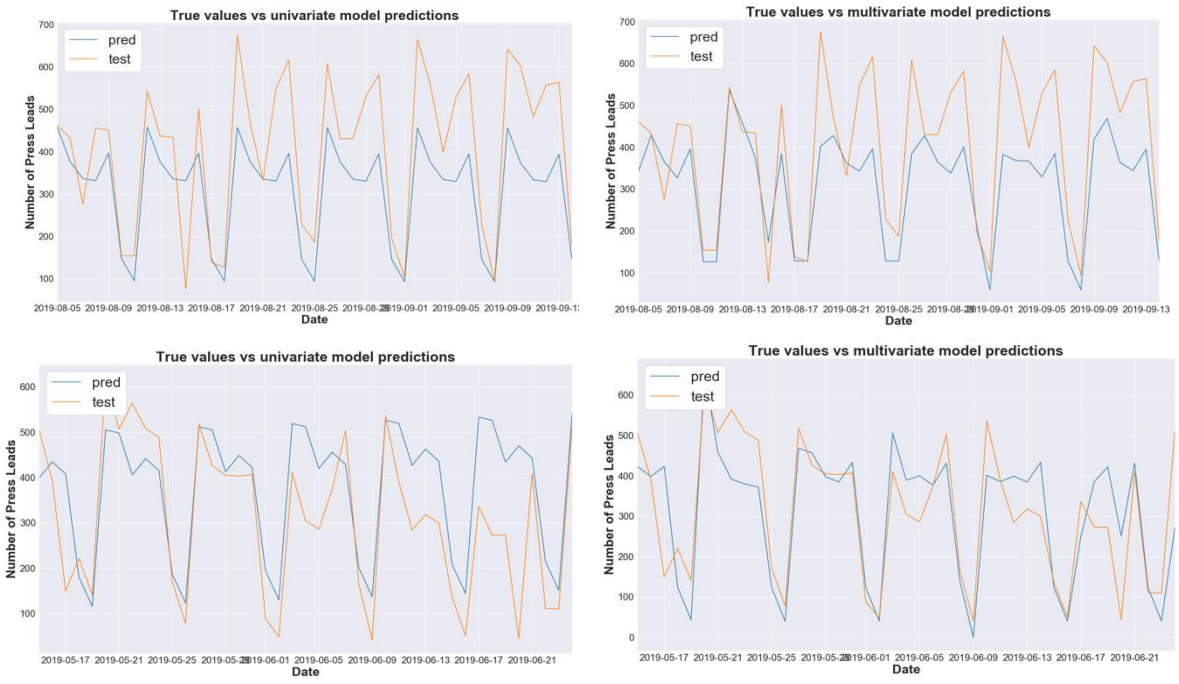

Figure 27: Predictive accuracy of univariate and multivariate SARIMA(X) models.

Above figure shows real values in orange and predictions in blue generated by univariate SARIMA (left panel) and multivariate SARIMAX (right panel). Comparing both visualizations from the left to the right panel, we can clearly see that the latter, i.e. multivariate model does a better job at capturing the turning points of the series, as the blue line corresponding to predictions is more sensitive to abrupt changes in the direction of the series. This is achieved

thanks to the advantage of information hidden in the exogenous variables used at training the exogenous model. Univariate model, on the other hand, depicted in the left panel visualizations, generates predictions of very regular patterns, being less sensitive to abrupt peaks or falls in the series. The relatively small improvement of 7 in the error metric between these models' accuracies is probably caused by the non-linear relationships between the exogenous variables and the target variable, which cannot be captured by a linear model. Additionally, it is likely that other factors, which are not included in the set of exogenous variables, affect the number of inbound press-related phone calls in the call center, which cannot be captured by the model due to missing information.

In the next section the so-far best found linear model will be attempted to be improved by means of applying models from a family of Recurrent Neural Networks, which is capable of capturing the non-linear relationships between the features.

4.5. GATED RECURRENT UNIT EXPERIMENT

Since the previous research on time series prediction strongly supports the No Free Lunch Theorem concluding that depending on a variety of factors statistical modelling can overcome the predictive power of Recurrent Neural Networks and vice versa, we decided to experiment with two deep learning approaches commonly used for time series prediction: GRU and LSTM. The first part of this section focuses on the former method which is proved to be faster at training the model. Time efficiency is especially important for our use-case taking into account the fact that the input fed onto a network is 24-dimensional including volume history of inbound phone calls, as well as, 23 exogenous variables extracted from Media Planning. The multidimensionality of the input data has a significant impact on the training time. No feature selection has been made for the RNN taking into account the nature of neural networks, which are able to perform automatic feature selection by setting particular weights connecting irrelevant features to close to 0 values. The initial training setup of the Gated Recurrent Unit model along with testing for different architectures has been described in the section below.

4.5.1. GRU Base Model Setup

We started off with a default GRU model that was further attempted to be improved by means of parameter tuning experiments. The training set-up of the initial model is covered below.

- The model has been developed in Python with use of *Keras* machine learning library on top of the open-source machine learning framework called *TensorFlow*.

- Custom Mean Squared Error (MSE) was selected as a loss function to penalize large errors more heavily. The time-shift between the input and output signals is fixed at 40 days, thus the MSE metric has been customized not to include the early time-steps in its loss calculation, to avoid the distortion in the later output of the model. Thus, a "warmup-period" of 30 time-steps has been given to the model, which is not taken into account in the loss calculation.
- Optimizer - The *RMSprop* optimizer has been used with the initial learning rate of $1e^{-3}$.
- Callbacks - *ReduceLROnPlateau* keras callback was used to decrease the value of a learning rate by a factor of 0.1 when the validation loss has not improved since the last epoch (patience=0). *EarlyStopping* callback was applied to stop the model training process unless the validation loss went down in the since 5 epochs (patience=5).
- Data preparation - Artificial Neural Networks, such as GRU differ from statistical forecasting methods by the fact that they are one of the supervised learning methods, meaning that we need to transform the series of data in a way to feed onto the network the series of inputs along with a sequence of target outputs. Neural networks can be fed with one or more attributes, as well as, they are able to output vectors of values. Since the goal of the model is to predict the number of inbound phone calls for the following 40 days, the initial model was designed to intake batches of 40-day long sequences of historical data as an input and it outputs a sequence of following 40 target values in a series. Since we also need to include the exogenous variables in the training process, the input data consists of not only the historical records of inbound phone calls but also it includes a series of 23 exogenous variables extracted from Media Planning. As a result, the training input data is of shape (64, 40, 24), where 64 constitutes the batch size, 40 is the sequence length and 24 is the number of features. The output, on the other hand, is of (64, 40, 1) shape, meaning it outputs a series of expected phone calls in the next 40 days.
- Batch size - batch size determines the number of training samples based on which the loss function is computed. In the initial model the batch size is set to 64.
- Maximum epoch – Maximum epoch in the initial model is set to 30, however, the training may finish earlier due to the *EarlyStopping* callback in use.
- Train and test sets – the data has been partitioned by 80:20 splitting rule, i.e. the earliest 80% of time series constitutes for the training set, while the remaining 20% is used to validate the generalization ability of the model.

- Data transformation – The data has been scaled to the interval of (0,1), which is known to help the neural network learn underlying patterns both, more accurately and more efficiently.

The visual presentation of the initial model setup is presented in the illustration below.

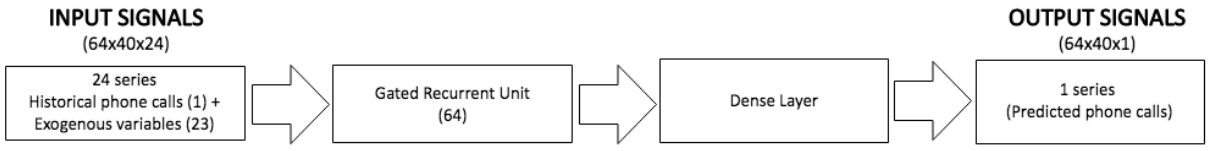

Figure 28: GRU Base model architecture visualization.

Taking into account the stochastic nature of the neural networks caused by random weights initialization, the model performance will be tested as an average error over calculated over 3 seeds. This way, the base model yields cross-validated predictions with Mean Absolute Error of 137.609351207. Meaning that the initial model is wrong on average by nearly 138 phone calls.

The predictions versus the true values have been plotted in the graphs below for both, train and test sets.

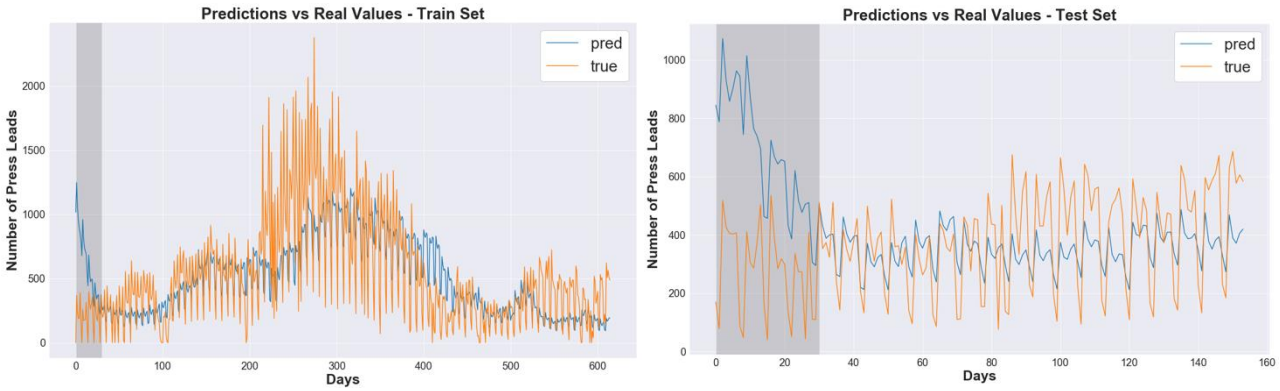

Figure 29: GRU Base Model predictions over real data for train and test sets.

It is important to highlight that above plot shows the output-signals of the model, which are 40 days into the future. The time-shift between the input and output signals is fixed at 40 days and since the model always predicts a sequence of 40 outputs, the x axis merely shows how many time-steps of the input-signals has been seen by the predictive model so far. This being said, the first 30 values presented in the above plots are very bad predictions caused by the fact that the model, generating a single time-step prediction for each time-step of the input data, it knows

very little about the history of the input-signals in the first few time steps and thus cannot make an accurate prediction. As a result, the first 30 predictions constituting for a "warmup-period" are ignored both during the training in the calculation of the prediction errors, as well as, during the evaluation of the model's performance (MAE). That is why the first 30 predictions are marked as a grey box in the plots above to highlight that these are considered as a model's warmup period.

Nevertheless, despite a relatively large MAE of over 138 phone calls, above plots show that the initial GRU network is able to capture the weekly seasonal patterns in the data both in the training, as well as, in the validation set. Both graphs indicate that the model does see the general tendencies (whether the number of phone calls would go up or down), however, the bias of the predictions remains pretty high which will be attempted to be improved by means of parameter tuning in the sections below.

4.5.2. Parameter tuning

After training the initial GRU model, a number of experiments have been conducted to verify the impact of various model parameters on its predictive power. Another goal of these experiments was to identify the most optimal GRU model for our use case. The experiments with the recurrent neural network's architecture include testing different values of the batch size, GRU units, the sequence length, the initial learning rate and the number of layers. The results of these experiments are compared to the initial model and presented in the following sections of the report.

Taking into account the computational cost related to the training of the neural networks, as well as, the importance of a big enough data volume needed for a network to capture underlying patterns in the data, the walk-forward cross-validation was applied on the test set only. This way, the reported cross-validated MAE is an average Mean Absolute Error obtained from successively shifted by one step 40 day long predictions made on the test set. Additionally, to ensure the consistency of the reported model's accuracy, the evaluation was performed three times over different seeds for each network architecture to avoid reporting seemingly good results obtained thanks to the stochastic factor. The following sections of the report present tables with average MSE and MAE of predictions calculated over 3 seeds for various model architectures. While MSE metric is printed for reporting purposes, as it was used during training as a loss function penalizing larger errors more than the small ones, the MAE metric is the one used for the optimal model selection.

4.5.2.1. GRU Units

The first modification of the initial model's setup involved the GRU units value, i.e. the dimensionality of the output space after the GRU Layer. The performance of the model tested across 3 seeds with different amounts of nodes in the GRU Layer has been presented in the table below.

GRU nodes	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
64 (Base model)	0.00524784345	137.609351207
128	0.00384287885	124.051554311
256	0.00510440347	136.195194523
512	0.00509942974	136.488891859

Table 4: GRU parameter tuning: GRU nodes.

Above table indicates that the optimal number of nodes for our use case is 128, which yields a cross-validated Mean Absolute Error of approximately 124 phone calls, resulting in predictions more accurate by over 13 phone calls as compared to the initial model. That is why, we will continue testing for other parameters keeping the GRU units set at 128.

4.5.2.2. Batch size

The second architecture aspect of the Recurrent Neural Network that has been tested and compared is the batch size which has a critical impact both, on the convergence of the training process, as well as, on the resulting accuracy of the predictions. Since our data is time dependent, we cannot use standard random batches but the batches need to be actually sequences of data. That is why, our custom batch generator provides groups (batches) of data sequences which are randomly drawn from the training set. Unfortunately, there is no one optimal batch size that fits all neural networks or data sets thus its size needs to be tested for specific use-cases. Another limitation that needs to be taken into account when setting up a batch size is the available GPU memory, as a larger amount of samples in one batch increases the required for training GPU memory. Finally, the critical consequences of setting up a small or large batch size are following:

- Generalization – A too large batch size may cause the neural network to get stuck in a local minimum, leading to a bad generalization ability and poor performance on the unseen data.

- Convergence speed – A too small batch size may lead to a slower convergence of the algorithm. Training samples belonging to a batch are randomly drawn from a training set, thus the resulting gradients might be noisy when calculated on partial (not sufficient) amount of data. The fewer data samples are included in each batch, the bigger impact each sample has on the weights updates, which leads to a more noisy and fluctuating learning process which may take a much longer time to converge.

Selecting the appropriate value of a batch size is a trade-off between accuracy and training speed and the way we can decide on it is to try out different set-ups and compare resulted performance, which has been done for the GRU model and presented in the table below.

Batch size	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
16	0.0046236442	129.875899979
32	0.00419642768	129.574339469
64 (Best so far)	0.00384287885	124.051554311
128	0.00452610268	130.294346036

Table 5: GRU parameter tuning: Batch Size.

Above table indicates that the batch size of 64 is optimal for the model. In result, further search of optimal parameters of the model will be performed with keeping the batch size set as 64.

4.5.2.3. Sequence length

In this section of parameter tuning, we will focus on finding the most optimal sequence length of series that constitute training batches of the model. The minimal considered sequence length equals 40, since the goal is to predict following 40 data points. Additionally, sequence length needs to be long enough given that during the loss computation process, the first 30 predictions of the sequence are considered as model's warmup period and thus do not contribute to the loss calculation.

Sequence length	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
40 (Best so far)	0.00384287885	124.051554311
60	0.00418972821	128.437400744
80	0.00408885061	126.038051988
100	0.00445234371	128.859144503

Table 6: GRU parameter tuning: Sequence length.

Above table indicates that the optimal sequence length for the model is 40, as it was set up in the initial model. The model fed on the 40 day long sequences of inputs delivers predictions with a mean absolute error of 124.051554311, which will be attempted to be improved by further parameter tuning in the next section.

4.5.2.4. Learning rate

The next section of GRU parameter tuning involves searching for an optimal starting learning rate. The model is set up with a *ReduceLROnPlateau* callback, which decreases the learning rate by a factor of 0.1 when the validation loss has not improved since the last epoch (patience=0). The initial learning rate in the base model has a value of 0.001, however, other initial learning rates have been tested and cross-validated on 3 seeds. The results of this experiment can be found in the table below.

Initial learning rate	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
0.001 (Best so far)	0.00384287885	124.051554311
0.003	0.00287297267	104.833812592
0.005	0.00282125974	100.225130295
0.007	0.00280603696	100.807216142
0.01	0.00484107403	130.060345202

Table 7: GRU parameter tuning: Learning Rate.

The cross validated results of above experiments indicate that the most optimal initial learning rate for our model equals 0.005 and it delivers predictions with a Mean Absolute Error of 100 phone calls. As compared to the so far most optimal found model, it gives an improvement of nearly 24 phone calls.

4.5.2.5. Layer setup

The last part of parameter tuning involves checking for the most optimal layer combination of a deep recurrent neural network. The results of different layer setups and its prediction accuracies are presented in the table below.

Layer setup	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
Layer_1(GRU=128) (Best so far)	0.00282125974	100.225130295
Layer_1(GRU=128); Layer_2(GRU=64)	0.00322681401	109.875247877
Layer_1(GRU=128); Layer_2(GRU=128)	0.00342085949	115.256874561
Layer_1(GRU=64); Layer_2(GRU=64)	0.0028749577	107.349124511
Layer_1(GRU=256); Layer_2(GRU=128)	0.00321624916	113.210154907
Layer_1(GRU=32); Layer_2(GRU=32)	0.00356565446	115.278734997
Layer_1(GRU=32); Layer_2(GRU=16)	0.00426270967	126.075860482

Table 8: GRU parameter tuning: Layer Setup.

The cross-validated results presented in the above table indicate that the model delivers most accurate predictions with 1 layer of 128 GRU units. Since increasing the number of layers did not result in a better predictive power, the final model will be only 1 layer deep.

4.5.3. Final GRU model

To conclude, the set of GRU parameter tuning experiments led to a conclusion that the most optimal parameter setup includes following criteria:

- 1 layer of 128 nodes
- Initial learning rate of 0.005 decreased by a factor of 0.1 unless improved as compared to the previous epoch
- Batch size of 64
- Sequence length of 40

The predictions versus the true values on training and data sets have been plotted in the charts below for both, train and test sets.

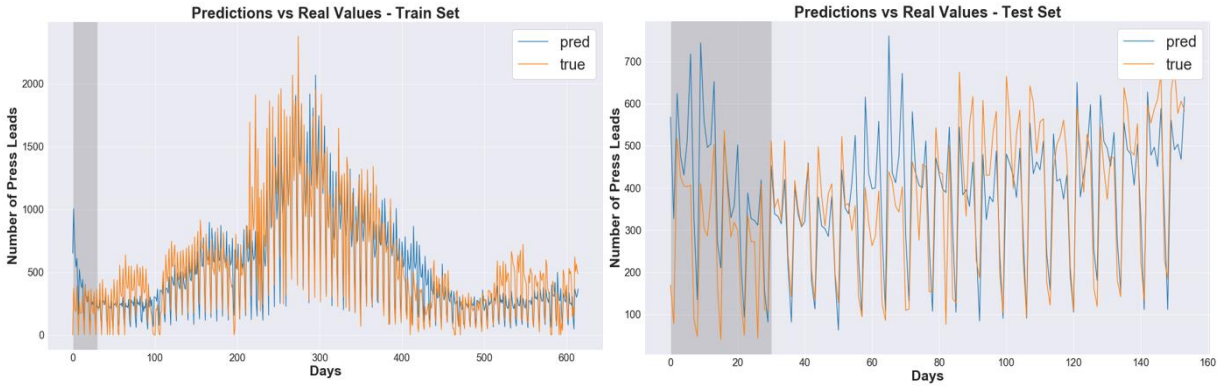

Figure 30: GRU Final Model predictions over real data for train and test sets.

Visualized predictions versus true values show that the model is able to capture the seasonality and trend much better than the initial model. The bias is relatively low across the whole sequence of the test set, however, we can see that the predictions at the time steps between 50 and 70 include a larger prediction bias. Last but not least, the model is very accurate at capturing the general tendencies, i.e. it usually predicts well the directions of the phone call spikes.

Given the fact that the GRU initial model's predictions had a Mean Absolute Error of 137.609351207 phone calls, the most optimal GRU model that was found provides predictions with a Mean Absolute Error of 100.225130295, which constitutes for an accuracy improvement of nearly 38 phone calls. This result outperforms the currently used by the company model providing accuracy which is higher by 50 units. Despite the significant improvement achieved by means of parameter tuning, the GRU final model proved to be less accurate for the underlying problem as compared to the previously developed statistical model, i.e. *SARIMAX(1,1,2)(1,0,1)_7* with a MAE of 75.494785. This being said, in the next section we will attempt to improve the accuracy delivered by the GRU model by means of another recurrent type of a recurrent neural network, i.e. Long-Short Term Memory. This model is known to be slower at training yet it tends to deliver more accurate predictions as compared to the Gated Recurrent Unit and it has also been widely used for time series predictions. The application of this model and its performance is described in the next section of the report.

4.6. LONG-SHORT TERM MEMORY EXPERIMENT

The initial LSTM model was run with the parameters that were found as the most optimal for the GRU model in the previous section. After that, a set of experiments were conducted iteratively, checking the effect of different values of particular model parameters on its predictive performance. The outcome of these experiments is reported similarly as in case of

the GRU model, as an average value obtained from tests run on 3 seeds, as well as, cross-validated with use of a walk forward cross-validation method of step 1.

4.6.1. LSTM Base Model Setup

The parameter values of the base model are taken after the most optimal GRU model. As such, the base Long-Short Term Memory model consists of 1 layer of 128 hidden neurons, starting learning rate of 0.005, batch size of 64 and a sequence length of 40. Model set up this way provides predictions that are on average wrong by 94.9113853228 phone calls. This constitutes for an improvement in the model's accuracy of 6 phone calls as compared to the GRU model run with the same parameter values.

Predictions versus true values of the initial LSTM model are plotted for both, train and test sets in the graphs below.

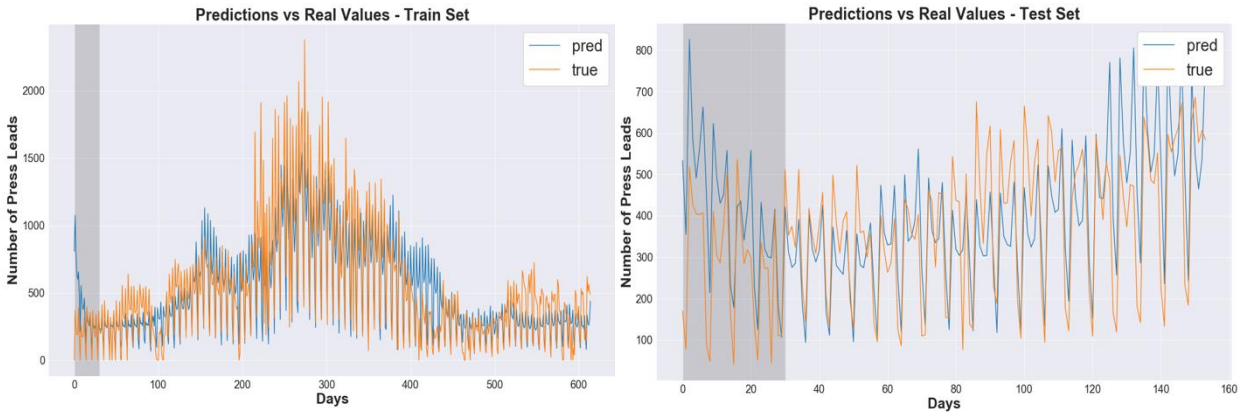

Figure 31: LSTM Base and Final Model predictions over real data for train and test sets.

Above plots prove that base LSTM model is able to capture the weekly seasonality with a slightly lower bias as compared to the GRU model. This result is already giving the LSTM an advantage over GRU model and we hope to achieve a yet better accuracy of the LSTM by fine tuning its parameters based on a set of experiments, which was described in the next sections of the report.

4.6.2. LSTM nodes

The first parameter search includes the number of LSTM nodes in the layer. The cross-validated results obtained with different parameter values of the model are presented in the table below.

LSTM nodes	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
64	0.00230794112	99.8904650703
128 (Base model)	0.00217102367	94.9113853228
256	0.00792687487	175.858612652
512	0.00772189038	171.640870962

Table 9: LSTM parameter tuning: LSTM nodes.

Looking at above table of the results, we can conclude that the number of LSTM nodes which gives the most accurate predictions is the same as in case of the most optimal GRU model and it equals 64. Therefore, the value of this parameter will be set to 64 in the next rounds of parameter tuning for the LSTM model.

4.6.3. Batch size

Similarly, as in the parameter search for GRU, the next examined parameter is the batch size. The cross-validated prediction accuracy of models with different values of this parameter are presented in the table below.

Batch size	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
16	0.00343510865	114.65276475
32	0.00252165156	101.772592588
64 (Base model)	0.00217102367	94.9113853228
128	0.00267651902	104.272608753

Table 10: LSTM parameter tuning: Batch Size.

Above table indicates that the optimal batch size yielding best LSTM predictions is the same as in case of GRU and it equals to 64. This being said, the batch size will be set to 64 in the following sections in the search of optimal values for remaining LSTM parameters.

4.6.4. Sequence length

In this section we are checking the impact of different sequence lengths in the training set on the LSTM model's predictive power. The results of these experiments have been shown in the table below.

Sequence length	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
40 (Base model)	0.00217102367	94.9113853228
60	0.00962270346	146.899543609
80	0.00619360439	118.163417619
100	0.00275136615	103.85033778

Table 11: LSTM parameter tuning: Sequence Length.

The cross-validated Mean Absolute Error proves to be the lowest for the base model, thus the most optimal sequence length for that case is the same as in the optimal version of GRU model, equal to 40.

4.6.5. Learning rate

The following part of parameter tuning involves searching for an optimal initial learning rate. This parameter has been tested across 3 seeds and its impact of LSTM model's performance is presented in the table below.

Initial learning rate	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
0.001	0.0055219874	155.264774973
0.003	0.00268421089	108.612304687
0.005 (Base model)	0.00217102367	94.9113853228
0.007	0.00252816546	98.7090978146
0.01	0.00250119001	98.1544482541

Table 12: LSTM parameter tuning: Learning Rate.

Above table indicates that the most optimal initial learning rate turns out to be the same as in the case of GRU optimal model, i.e. it equals 0.005.

4.6.6. Layer setup

The last part of parameter tuning involves checking for the most optimal layer setup of a deep LSTM network. The results of different configurations and its prediction accuracies are presented in the table below.

Layer setup	Validation Loss MSE (over 3 seeds)	Cross-validated MAE (over 3 seeds)
Layer_1(LSTM=128) (Base model)	0.00217102367	94.9113853228
Layer_1(LSTM =128); Layer_2(LSTM =64)	0.00457890657	129.247157256
Layer_1(LSTM =128); Layer_2(LSTM =128)	0.00710014572	154.427538832
Layer_1(LSTM =64); Layer_2(LSTM =64)	0.00578577397	148.10400959
Layer_1(LSTM =256); Layer_2(LSTM =128)	0.00762837209	163.478928806
Layer_1(LSTM =32); Layer_2(LSTM =32)	0.00386882231	118.216303277
Layer_1(LSTM =32); Layer_2(LSTM =16)	0.00346415823	112.144094827

Table 13: LSTM parameter tuning: Layer Setup.

The final parameter search suggests that again, the same layer configuration is optimal for both GRU and LSTM models, which is one layer consisting of 128 neurons.

4.6.7. Final LSTM Model

The final LSTM model provides predictions that are more accurate from both: the final GRU model and the currently used by the company model by 6 and 56 units, respectively. After the above-reported parameter search which was performed to find the most optimal values for the LSTM model's parameters, we can conclude that the same neural network setup works best for both tried types of recurrent neural networks, however, the predictions made with LSTM model are off by on average 94.9113853228, while GRU model's predictions Mean Absolute Error equals 100.225130295. As long as the parameter search was helpful in case of the GRU model as it increased its predictive power significantly, the same sequence of parameter search experiments applied on the LSTM network did not provide such improvement in its accuracy.

5. CONCLUSION

The main goal of this project was to deliver time series predictions of the number of inbound phone calls related to press-published ads expected to arrive in a call center in the consecutive 40 days. The first steps of analysis include data extraction, aggregation and preprocessing. After that, data visualization methods have been used in order to identify general tendencies and seasonal patterns hidden in the data. Finally, following the No Free Lunch Theorem, a number of univariate and multivariate predictive models have been conducted and compared. The result of comparative study performed on a variety of tested predictive models leads to a number of conclusions.

The first surprising insight is that statistical models, yet mathematically simpler, outperformed the tested recurrent neural networks (GRU and LSTM). One possible reason for this phenomena is the fact that statistical models require much less data as compared to the neural networks to be able to learn the underlying patterns. This reasoning is additionally supported by the multidimensionality of the data inputs, in result increasing the size of data required for training. Another interesting insight can be derived from the parameter tuning process of the recurrent neural networks, as the most optimal architecture setup turned out to be the same for both: GRU and LSTM models. Finally, it is interesting to note that increasing a number of layers for both networks didn't result in a higher model accuracy. This conclusion may be supported by 2 hypotheses. First, the underlying pattern is simple enough to be captured by a 1-layer network. Second, a deeper neural network is more complex and thus it requires larger amount of training data before it can learn the underlying patterns. The final conclusion drawn from the comparative study of the neural networks states that LSTM exhibited advantage over the GRU model providing significantly more accurate predictions. However, its performance was yet worse than the one of the statistical models.

Finally, the performance analysis of various types of predictive models proved that multivariate SARIMA(X) outperformed each of the other tested approaches and thus is recommended to be used as a final model by the company. The obtained prediction accuracy of the final model is also significantly higher as compared to the model used by the company. This being said, the objective of the internship was fulfilled and the internship was evaluated as successful.

5.1. LIMITATIONS AND FUTURE WORK

The limitations of the conducted research include the limited amount of historical data which merely covers 810 consecutive days and may not be sufficient for an effective training of recurrent neural networks. Another important limitation is the non-exhaustive set of exogenous features provided for analysis. As discussed with the specialists in the relevant business area, there is a number of factors affecting the number of press leads within the company which is not recorded and currently cannot be known beforehand by the company – such features include the competitive company advertising in the press at the same time. Another limitation of the research is the cross-validation methodology for the recurrent neural networks. In case of the statistical model the regular walk-forward cross-validation methodology was possible to be applied as the model does not require large amounts of data for the training to capture the underlying patterns effectively. In case of the neural networks, however, we only performed the walk-forward cross-validation method on the 20% of the most recent data for two reasons: first, applying regular cross-validation technique on entire dataset would mean training the neural network multiple times which is impossible for the sake of given time limitation, secondly, training the neural network on the even smaller amount of data would not be sufficient for the model to learn underlying patterns. Last but not least, given more time, more detailed parameter search could be performed and a larger number of epochs set up to give the model more time for learning. Finally, in the future work, some other time series modelling approaches, such as a novel CNN for time series prediction can be tested as an attempt to increase the predictions' accuracy.

5.2. FINAL THOUGHTS

Regarding my overall experience during the internship, I have to admit that this was an outstanding opportunity to be immersed into the professional working environment right after the first year of theoretical classes. I am very grateful for having this opportunity to work with both, business and technical professionals, who were always there sharing their expertise and advice on the development of my work. This working experience has enriched both, my technical and organizational skills, and I am now feeling much more confident moving onto another job as a more experienced data scientist. I also have to admit that the internship was not a bed of roses and I learnt how to deal with work under pressure and how make complex things understood easily by the business audience. Another set of skills that have been developed during my time spent at the Company X was adaptation and presentational skills, as I was to regularly update the decisive persons in the company about the development of my work and

adjust further actions according to the priorities, as aligned. Finally, it is important to highlight that NOVA IMS master program in Data Science and Advanced Analytics provided me with necessary fundamentals which were of a huge help during my internship. The skills obtained at the NOVA IMS and used during this project development include coding in Python, querying the SQL database, developing and assessing predictive models and many more. To conclude, this internship would not have been completed successfully if it wasn't for the valuable knowledge gained at NOVA IMS master program and the always supportive colleagues.

6. BIBLIOGRAPHY

Abraham, B. & Ledolter, J. (1986). *Forecast functions implied by autoregressive integrated moving average models and other related forecast procedures*. International Statistical Review 54(1), 51-66.

Bottcher, M., Hoppner, F., & Spiliopoulou, M. (2008). *On exploiting the power of time in data mining*.

Box, G. & Jenkins, G. (1970). *Time Series Analysis: Forecasting and Control*, Holden-Day.

Chatfield C. (2000*). Time-series forecasting*. Chapman & Hall/CRC.

Chen, Z., Yang, Y. (2004*). Assessing Forecast Accuracy Measures.* Iowa State University.

Gerbing D. (2016). *Time series components*.

Goodwin, P., & Lawton, R. (1999). *On the asymmetry of the symmetric MAPE*. International journal of forecasting, 15(4), 405-408.

Gooijer, J. G. D. & Hyndman, R. J. (2006), *25 years of time series forecasting*. International Journal of Forecasting 22(3), 443-473.

Hippert, H., Pedreira, C. & Souza, R. (2001), *Neural networks for short-term load forecasting: a review and evaluation*, IEEE Transactions on Power Systems 16(1), 44-55.

Hyndman, R. J. (2001), *It's time to move from what to why*. International Journal of Forecasting 17, 567-570.

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22, 679–688.

Hyndman, R.J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice* (2nd edition). OTexts: Melbourne, Australia. OTexts.com/fpp2.

Kourentzes N. (2014). Additive and Multiplicative Seasonality – can you identify them correctly? Forecasting research. https://kourentzes.com/forecasting/

Makridakis, S. & Hibon, M. (2000). *The M3-competition: Results, conclusions and implications*. International Journal of Forecasting 16(4), 451-476.

Mittelman R. (2015). *Time-series modeling with undecimated fully convolutional neural networks*. ArXiv preprint arXiv:1508.00317.

Nau, R. (2019). *Statistical forecasting: notes on regression and time series analysis*. [Fuqua School of Business, Duke University]

Olah, C. (27, 08, 2015). *Understanding LSTM Networks*. Olah's Blog. http://colah.github.io/posts/2015-08-Understanding-LSTMs

Ostashchuk, O. (2017*). Time Series Data Prediction and Analysis.*

Pascanu, R., Mikolov, T., Bengio, Y. (2013). *On the difficulty of training Recurrent Neural Networks*.

Schmidhuber, J. (2014). *Deep Learning in Neural Networks: An Overview*.

Staudemeyer, R., C. (2012). The importance of time: Modelling network intrusions with long short-term memory recurrent neural networs. [Doctoral dissertation, University of Applied Sciences Schmalkalden]

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). *Recurrent Neural Network Regularization*.

Zhang, G., Patuwo, B. & Hu, M. (1998). *Forecasting with artificial neural networks: The state of the art*. International Journal of Forecasting 14(1), 35-62.