Examensarbete

HÖGSKOLAN I SKÖVDE

A COMPARATIVE STUDY BETWEEN ALGORITHMS FOR TIME SERIES FORECASTING ON CUSTOMER PREDICTION

An investigation into the performance of ARIMA, RNN, LSTM, TCN and HMM.

Examensarbete inom huvudområdet informationsteknologi
Grundnivå 30 Högskolepoäng
Vårtermin 2019

Olof Almqvist

Handledare: Jonas Gamalielsson
Examinator: Mikael Berndtsson

Abstract

Time series prediction is one of the main areas of statistics and machine learning. In 2018 the two new algorithms higher order hidden Markov model and temporal convolutional network were proposed and emerged as challengers to the more traditional recurrent neural network and long-short term memory network as well as the autoregressive integrated moving average (ARIMA).

In this study most major algorithms together with recent innovations for time series forecasting is trained and evaluated on two datasets from the theme park industry with the aim of predicting future number of visitors. To develop models, Python libraries *Keras* and *Statsmodels* were used.

Results from this thesis show that the neural network models are slightly better than ARIMA and the hidden Markov model, and that the temporal convolutional network do not perform significantly better than the recurrent or long-short term memory networks although having the lowest prediction error on one of the datasets. Interestingly, the Markov model performed worse than all neural network models even when using no independent variables.

Acknowledgements

I would like to extend my thanks to my supervisor, assistant professor Jonas Gamalielsson at the University of Skövde, for the opportunity to work together and for valuable suggestions that improved the quality of this work.

CONTENTS

1 Introduction

Time series forecasting is an important tool in modern science for studying relationships between a dependent variable and time, possibly together with other independent variables. The goal of time series prediction is to collect historical data that can be used to create a quantitative model that explains the characteristics of the explained variable. Areas of use include econometrics (Zhang, Yin, Zhang & Li 2016; Wang 2016), biology (Huang et al. 2016), psychology (Jebb, Tay, Wang & Huang 2015) and climatology (Duchon & Hale 2012).

In addition to scientific research, time series forecasting is frequently utilized in the business sector to forecast areas like product demand as well as the need for materials and personnel. When utilized, it can be developed and used by individual analysts using code or form a module within a system such as in Amazon Forecast (Amazon 2019), the ERP system SAP (Dadouche 2018) and most interface-based analytics software such as SAS (SAS n.d), NCSS (NCSS 2019), and Tableau (Tableau 2019).

Time series can have four characteristics as described by Jebb & Tay (2017). These are trends, seasonality, cycles and noise. Algorithms for time series forecasting are appropriate for data that has a time dimension and exhibit one or more of these properties.

In the beginning of the ninetieth century deterministic models were used and the involvement of stochastic properties became prominent with the invention of the autoregressive moving average (ARIMA) model developed by major contributions from Box & Jenkins (1970). This statistical approach dominated the development of models for time series forecasting for thirty years (De Gooijer & Hyndman 2006).

Quantitative models for time series analysis based on deep learning has become more and more important in recent years (Makridakis, Spiliotis & Assimakopoulos 2018). The development of machine learning and deep learning has led to a myriad of solutions that compete with traditional statistical methods with important examples being recurrent neural networks (RNN) (Russel & Norvig 2010, pp. 729) and long-short term memory networks (LSTM) (Hochreiter & Schmidhuber 1997) which revolutionized the ability to create models for sequential problems like speech recognition (Xiangang & Wu 2014) and machine translation (Sutskever, Vinyals & Le 2014).

A novel implementation by Bai, Kolter & Koltun (2018) use feed forward networks and convolutional operations utilized in computer vision to construct a neural network called “temporal convolutional network” (TCN) that can learn faster and attain higher accuracy on some sequential datasets compared to LSTMs.

A different approach to time series modelling is using Markov chains. While originating in 1906 more recent developments by Ky & Tuyen (2018) using higher order chains and intervals of the time series as states suggest promising results on its ability to predict stock prices, a traditionally difficult problem because of its high level of stochasticity.

Research into the accuracy and reliability of different algorithms for time series forecasting can lead to improvements in the ability of the scientific community to carry out time-based analysis. Furthermore, better knowledge in which algorithm to choose for a problem could help businesses create better forecasts thereby improving their strategical and tactical planning as well as optimizing operations.

In this study, data is collected from two Swedish theme parks containing visitors per day over time. This data is fused with datasets containing weather parameters and search keyword frequency from Google Trends. With this, the three traditional algorithms ARIMA, RNN and LSTM are compared to the two novel implementations: temporal convolutional networks and higher order Markov chains.

The remainder of the report is organized as follows. In section 2, the technical and mathematical theory behind the algorithms are explained. Section 3 describes the problem area and the contribution of this study. Section 4 describes and motivates the choice of scientific methods. Acquired data and relevant tools are presented in section 5 along with an exploratory data analysis Results are presented in section 6. There is an analysis of concerning the meaning of the acquired results in section 7. Lastly, a discussion concerning the performed study and the subsequent results are presented in section 8.

2 Background

The background section examines the theoretical foundations for the investigated time series models and the general characteristics of time series datasets.

2.1 Terminology

Network Architecture – describing the morphology of a network by referring to the structure of nodes, layers, and the way their edges are connected.

Machine Learning Algorithm/algorithm – a theoretical description of a machine learning based quantitative system that has the capacity to learn from input data and create a prediction based on one or more independent variables.

Statistical Model – a general description of a mathematical equation that describe relationships between parameters and variables.

Machine Learning Model/model – an implementation of a machine learning algorithm where hyperparameters have been selected and the algorithm has been trained on a dataset.

2.2 Characteristics of Time Series Data and Forecasting

A time series is a chronological sequence of observations of a predictor variable behaving like a stochastic process; that is, individual values are impossible to predict exactly but there can be an overall pattern. This type of data can display patterns that can be used to create models to predict future behavior. The series can display a trend which can be upward, downward or stationary. Furthermore, the pattern can be of a cyclical nature changing from lower and higher values in an interval around an average point, an example of this is a heartbeat over time. Typical of cyclical pattern is that they have no set repetition and doesn’t repeat certain periods of the year. Lastly, time series can have seasonality which is a pattern where values change in a certain manner on specific points in time, for example during winter and summer (Montgomery, Jennings & Kulahci 2015, pp. 6-12).

Another important attribute that can be found in time series is called white noise. A dependent variable that changes randomly over time with constant variance and no autocorrelation can be said to be white noise. The scatter plot of such a series across time will indicate no pattern and hence forecasting the future values of such a series is not possible. It is not possible to do time series analysis on such data (Montgomery, Jennings & Kulahci 2015, p. 71).

Forecasting time series data can be classified as short-term, medium-term and long-term. Short forecasts range between a few days to a few months. Medium time forecasts can go 1-2 years into the future and long-term forecasts can be several years into the future. Statistical models are useful for short- and medium-term forecasts (Montgomery, Jennings & Kulahci 2015, p. 2).

There are two categories of forecasting techniques. *Qualitative techniques* are relatively rare and can consist of a panel of experts giving individual estimates which are then pooled together, such as with the Delphi Method. In *quantitative models* there are statistical and machine learning models for forecasting. These are considered more stable and are most commonly used. Types of quantitative forecasting models are regression models (chapter 2.6.1), smoothing models (chapter 2.5), general time series models (chapter 2.6) (Montgomery, Jennings & Kulahci 2015, pp. 4-6).

2.3 Statistical Models

Statistical models differ from machine learning (ML) models in that they usually require a set of presuppositions for the model to work. *E.g*. linear regression requires normally distributed data, homoscedasticity and lack of autocorrelation (Casson 2014). ML models on the other hand typically learn and adapt from the data and don't require as thorough preprocessing.

2.4 ETS Models

ETS stands for "Error-Trend-Seasonality" and includes models such as exponential smoothing, trend methods and ETS decomposition. ETS decomposition is a way to break down a time series into components of trends, seasonality and cycles (Jofipasi, Miftahuddin & Hizir 2017). This can be a useful tool for diagnosing time series to determine whether a seasonal ARIMA should be used and if the data must undergo transformations to become stationary. In this study, ETS decomposition will be used as a diagnostic tool in the development of the ARIMA model.

2.5 EMWA Models

EMWA is one of the more advanced smoothing models that can be used for time series forecasting. Other examples include Simple Moving Average (SMA) and Simple Moving Median. These models are also a common tool in descriptive statistics and only EMWA is usually used for producing detailed predictions.

A regular moving average can be improved by using an exponentially weighted moving average (EMWA). SMA has some weaknesses, smaller windows will lead to more noise rather than signal. Its lacks starting values and it will never reach the full peak or valley of the data due to averaging. Furthermore, it does not inform about future behavior all it really does is describe trends in the historical data. Lastly, an issue with SMA can be the presence of extreme historical values that skew the rolling mean (Montgomery, Jennings & Kulahci 2015, pp. 223-257).

EWMA reduce the lag effect of SMA and can put more weight on values that occurred more recently by applying a higher weight to the more recent values. The amount of weight given to the most recent values depend on the actual parameters used in the EWMA and the number of periods given a window size (Montgomery, Jennings & Kulahci 2015, pp. 223-257).

Although ETS- and EMWA models have been historically important in time series forecasting, they could be considered older and somewhat more simple models that have already been thoroughly studied and will therefore not be investigated in detail in this study.

2.6 ARIMA Models

ARIMA stands for Auto Regressive Integrated Moving Average. There are thus three independent components making up the model and they can be used together or with the exclusion of one or more.

ARIMA can be divided into three categories (Durka & Pastorekova 2012)

- Non-seasonal ARIMA (ARIMA)
- Seasonal ARIMA (SARIMA)
- Multivariate ARIMA (ARIMAX)

Regular ARIMA-models is a univariate time series model because it works on data that consists of single observations of a dependent variable over regular time intervals and no external predictor variables.

ARIMA models are usually applied where data show evidence of non-stationarity, where an initial differencing step – corresponding to the integrated part of the model – can be applied one or more times to eliminate the non-stationarity. A stationary time series is one whose statistical properties such as mean, variance and covariance are constant over time. Most models assume that the time series is or can be rendered approximately stationary through mathematical transformations. Making a time series stationary through differencing where needed is an important part of the process of fitting an ARIMA model (Montgomery, Jennings & Kulahci 2015, pp. 48-50).

There are several ways to make a non-stationary dataset stationary but the simplest is differencing. Subtract each value according to $Y_t - Y_{t-1}$. This can be done several times if needed and every step of differencing costs one row of data.

Differencing can also be done by season if there are long-term seasonal patterns that cause the non-stationarity. For example, if the time series is recorded monthly and there is an annual change in values that cause non-stationarity the dependent variable can be transformed according to $Y_t - Y_{t-12}$. It is also common with seasonal ARIMA to combine both methods, taking the seasonal difference of the first difference (Montgomery, Jennings & Kulahci 2015, p. 52).

Furthermore, there are hypothesis tests that can be used to indicate mathematically if a series can be considered stationary or not, one of these if the Dickey-Fuller test (Bernal & Sanso 2001).

Major components of ARIMA are the autoregressive portion, the integrated portion and the moving average portion. Non-seasonal ARIMA models are generally denoted ARIMA(p, d, q) where parameters p, d, q are non-negative integers (Montgomery, Jennings & Kulahci 2015, pp. 327-367).

2.6.1 The Autoregressive AR(p) Part of ARIMA

A regression mode that utilizes the dependent relationship between a current observation and observations over a previous period. An auto regressive model or AR model is one in which Y_t depends only on its own past values $Y_t = f(Y_{t-1}, Y_{t-2}, \ldots, Y_{t-n})$.

A representation of an autoregressive model where it depends on n of its past values (p = n) called "AR(p)" model can be mathematically represented as:

(1) $Y_t = B_0 + B_1 * Y_{t-1} + B_2 * Y_{t-2} + B_n * Y_{t-n} + E_t$

An important question is how many past values to use. AR(p) means p past values. B are coefficients like those used in linear regression models, and E_t is an error term representing random behavior (white noise) in the series (Montgomery, Jennings & Kulahci 2015, pp. 338-348).

2.6.2 The Integrated I(d) Part of ARIMA

If the time series was shown to be non-stationary there are two ways to make it stationary differencing (subchapter 2.2.3) and mathematical transformations using logarithms (Montgomery, Jennings & Kulahci 2015, p. 363) can be employed. A series which is stationary after being differentiated d times is said to be integrated of order d denoted I(d). Therefore, a series which is stationary without differencing is said to be I(0) and integrated of order 0.

2.6.3 The Moving Average MA(q) Part of ARIMA

The MA part of the model uses the dependency between an observation and a residual error from a moving average model applied to lagged observations. A moving average model is one when Y_t depends only on the random error terms which follow a white noise process (Montgomery, Jennings & Kulahci 2015, pp. 333-337).

A common representation of a moving average model where it depends on q of is past values is called MA(q).

(2) $Y_t = Q_0 + E_t + Q_1 * E_{t-1} + Q_2 * E_{t-2} +, \dots, +Q_q * E_{t-q}$

Where the error terms E_t are assumed to be white noise processes with mean zero and variance σ^2. That is average(E_t) = 0 and var(E_t) = 1.

2.6.4 Combinations of Parts

There are times when the time-series may be represented as a mix of both AR and MA models referred as ARMA(p, q). The general form of such a time series model which depends on p of its past values and q past values of white noise disturbances takes the following form (Montgomery, Jennings & Kulahci 2015, pp. 354-355). How to develop an ARIMA model will be explained in subsequent chapters.

(3) $Y_t = B_0 \ + \ B_1 \ * \ Y_{t-1} \ + \ B_2 \ * \ Y_{t-2} \ + \ B_n \ * \ Y_{t-p} \ + \ Q_1 \ * \ E_{t-1} \ + \ Q_q \ * \ E_{t-q}$

2.6.5 Autocorrelation Function (ACF)

Once the data is stationary, model selection is the next step. An autocorrelation plot – also known as a correlogram – shows the correlation of the series with itself lagged by n time units. The y-axis is the correlation and the x-axis are the number of time units of lag. This can be done several times for different times of lags.

(4) $Corr(Y_t, Y_{t-p}) = \frac{Cov(Y_t, Y_{t-p})}{\sigma Y_t * \sigma Y_{t-p}}$

The results from the ACF plot should show whether the (AR) or the (MA) part of the ARIMA model should be used, or both (Yaffee & McGee 2000, pp. 122-126).

- If the autocorrelation plot shows positive autocorrelation at the first lag (lag-1) then it suggests using the AR terms in relation to the lag.
- If the autocorrelation plot shows negative autocorrelation at the first lag, then it suggests using MA terms.

2.6.6 Partial Autocorrelation Function (PACF)

A partial correlation is a conditional correlation. It is a correlation between two variables under the assumption that some other set of variables is known and considered.

For example, in a regression context where y is the response variable and x1, x2, x3 are the predictor variables. The partial correlation between y and x3 is the correlation between the variables determined considering how both y and x3 are related to x1 and x2.

Typically, a sharp drop after lag “k” suggests an AR-k model should be used. If there is a gradual decline it suggests an MA model. Identification of an AR model is often best done with the PACF. Identification of an MA model is often best done with the ACF rather than the PACF (Yaffee & McGee 2000, pp. 122-126).

2.6.7 The Box-Jenkins Methodology

The Box-Jenkins methodology is an approach on how to build a univariate time series models in an orderly manner as the minimize the risk of faulty assumptions. It is an iterative three step approach (Akpanta & Okorie 2014).

1. Model identification and model selection.

2. Parameter estimation.

3. Model checking

In the first step, diagnostic tests are used to investigate whether the data is stationary or not. This can be done by visually exploring charts using rolling averages for standard deviation and averages and breaking the data down with ETS decomposition. How to make data conform to stationarity was discussed in (chapter 2.6). Furthermore, the Dickey-Fuller hypothesis test (chapter 2.6) can be used to verify that the time series has become stationary after relevant treatment has been implemented. In this stage, it should also be verified whether the dataset has attributes of seasonality, and if it does a special type of seasonal ARIMA should be considered.

Once the type of ARIMA has been decided, a series of tests must be done to find the AR(p), I(d) and MA(q) terms that should be used for the data. In the Box-Jenkins methodology, autocorrelation plots are used to find the MA terms and partial autocorrelation plots to find the AR terms.

In the second stage, the parameter estimation stage, computational algorithms are used to arrive at coefficients (chapters 2.6.1, 2.6.2 & 2.6.3) that best fit the selected ARIMA model. The two most common methods are maximum likelihood estimation and non-linear least-squares estimation.

The third and last stage, model checking, is done by testing whether the estimated model conforms to the specifications of a stationary univariate process. For example, the residuals should be independent of each other and constant in mean and variance over time. To verify this, a Ljung-Box test can be used to test the autocorrelation within the dataset. The Ljung-Box test tests the hypothesis that the correlation between two points with lag k are zero and can also be used to evaluate an ARIMA model by saying whether the residuals are independent or not (Ljung & Box 1978).

2.6.8 Summary of ARIMA Development

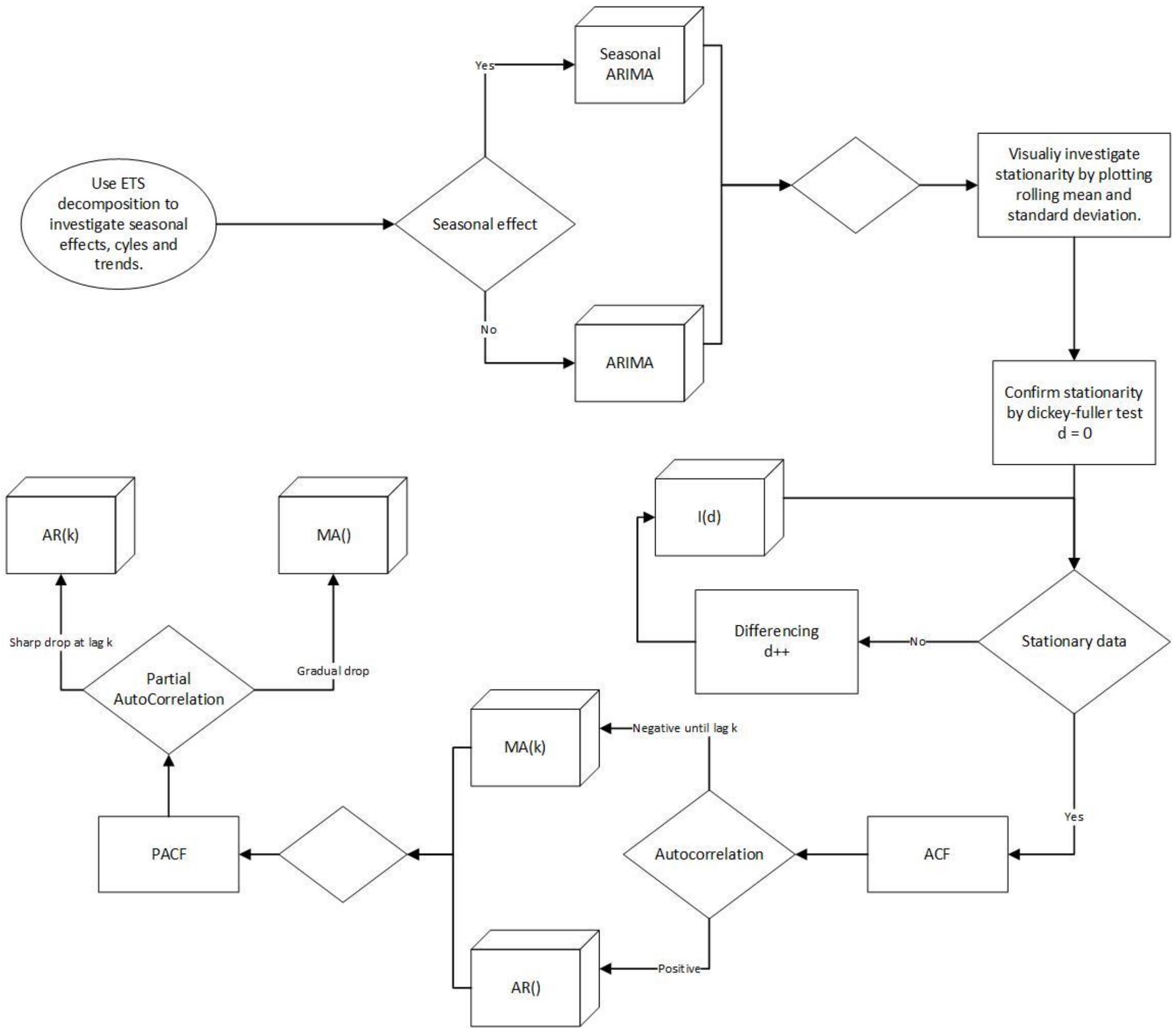

Use ETS decomposition to investigate seasonal effects, cyles and trends.
Seasonal effect
Yes
No
Seasonal ARIMA
ARIMA
Visualiy investigate stationarity by plotting rolling mean and standard deviation.
Confirm stationarity by dickey-fuller test d = 0
I(d)
Differencing d++
Stationary data
No
Yes
ACF
Autocorrelation
Negative until lag k
Positive
MA(k)
AR()
PACF
Partial AutoCorrelation
Sharp drop at lag k
Gradual drop
AR(k)
MA()

Figure 1. Overview of the development of an ARIMA model according to the Box-Jenkins methodology.

2.7 Feed Forward Networks

A basic feed forward neural network (FNN) got its name because it has architectural similarities to metazoan brain cells with the components of dendrites and axons. An FNN can be described as a weighted acyclic bipartite graph (figure 2). It consists of three distinct types of nodes: input, hidden and output. Input nodes receive the input variables which must go through unity-based normalization to transform the set of inputs into the range [0, 1]. Each input is subsequently multiplied by a weight and sent to a connected hidden node (Schmidhuber 2014).

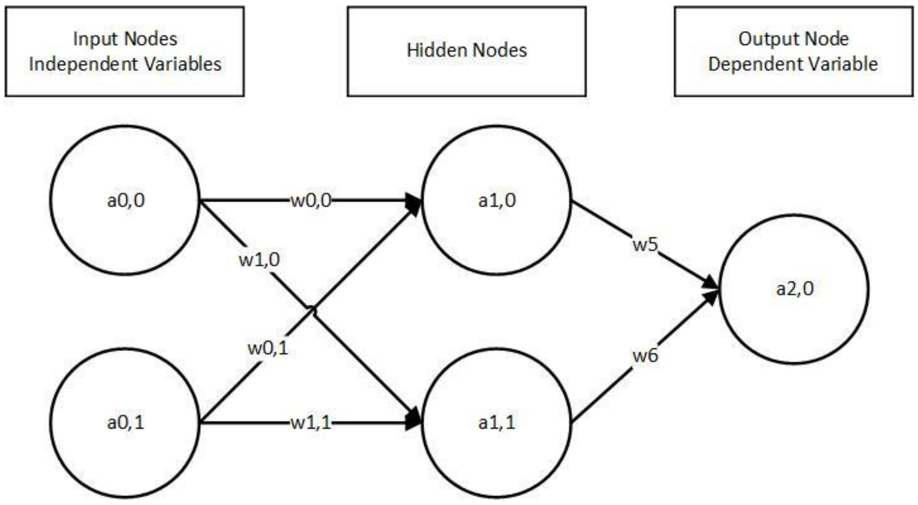


Figure 2. Graphical description of an FNN. The fist index number i specifies layer, the second signifies the node number, j. Weights $w_{i,j}$ are labeled according to node numbers with sink node as i and source node as j.

Hidden nodes are special in that they create the ability of non-linearity and being able to handle complex interactions between different input variables. They do this by using activation functions which transform the incoming value $a_{0,0} * w_{0,0} + a_{0,1} * w_{0,} + \ldots + a_{i,j} * w_{i,j}$ (a weighted sum) by a function where common examples include the rectifier function max(0, x) or a logistic function $f(x) = 1/(1 + e^{-x})$ (Glorot, Bordes & Bengio 2011). By using activation functions, each incoming weighted sum is transformed back into the original [0, 1] range before going through further calculations in the next layer.

The output of a node being $Y_n = W_n * X_{n-1}$. Where $W_n * X_{n-1} = (w_{i,j} + b_i) *$ ai,j. The b is a bias term that can be added to the weighted sum. X_{n-1} is all previously connected nodes multiplied by their related weights with added biases the generated output of that node is then transformed by the activation function $X_n = F(Y_n)$.

Every transmission of data from one layer to the next (figure 2) can also be interpreted in matrix-vector form.

(5) $F(\begin{bmatrix} w_{0,0} & w_{0,1} \\ w_{1,0} & w_{1,1} \end{bmatrix} \begin{bmatrix} a_{0,0} \\ a_{0,1} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix})$

This process can be carried out through any number of hidden layers until the signals are finally pooled together into an output layer which also usually has a special activation function in order to transform the output into a reasonable scale such as the sigmoid function for probabilities. Between each cycle, a cost function is used to calculate the error between the derived prediction value and the actual value in the training set.

The calculations between the input nodes, the weights and the activation functions are described by LeCun, Bottou, Orr & Muller (1998) as a function $M(Z^p, W)$ where Z is the input variables and W the adjustable parameters (*i.e.* weights and bias terms). Each cycle of $M(Z^p, W)$ has a related desired output value $D^0, D^1, \ldots, D^p$. These are used in a cost function E^P to calculate the training error. Commonly used cost functions are mean absolute error, root mean square error or mean square error which is $½ * (D^P – M(Z^P, W))^2$.

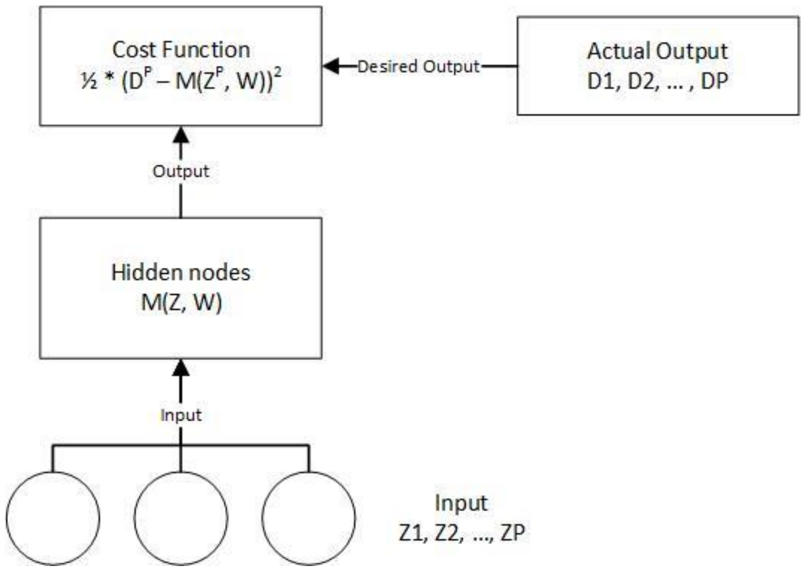


Figure 3. The FNN as a learning machine. Adapted and redrawn from Muller et al (1998).

The actual learning in the network is done by tuning the weights and the bias terms on all the edges between nodes to minimize the resulting error and an important task in the science of FNNs have thus been to develop methods to adjust W both regarding computational efficiency and accuracy in order to minimize the cost function. Two concepts become central: backpropagation and gradient descent.

The gradient ∇F of an n-dimensional space is the change in variables that cause the function to increase in value most rapidly. The concept of the gradient is the same as that of the derivative of a univariate function but in a multidimensional setting (Adams & Essex 2013, pp. 716-720).

(6) $F(x, y, z) = \nabla F(x, y, z)$

(7) $\nabla F(x, y, z) = (\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z})$

Calculating the gradient for a one-dimensional function $\nabla F(x) = \frac{\partial F}{\partial x}$ would be the same as a regular derivative. By using the current coordinates of the function and plugging them into the gradient the direction of greatest ascent is obtained. By continuously doing that, eventually a local maximum is attained and while on that point the gradient becomes zero.

In neural networks, the values derived from the cost function is used as an output variable and the vector space has as many dimensions as there are weights and bias terms in the network Muller et al. (1998). Hence, gradient descent is the opposite of the gradient $-\Delta F(W)$ (Adams & Essex 2013, ss. 720). By changing the parameters in W in order to minimize the cost function (which is an aggregate representation of all the tunable parameters) the network can improve its performance or learn.

Gradient descent is the most widely used way to optimize tunable parameters and there are three variants of this method: batch gradient descent, stochastic gradient descent and mini batch gradient descent. Batch gradient descent updates the tunable parameters of the cost function in respect to the entire dataset, making it slow and requiring that all training data fit into the RAM. Stochastic gradient descent calculates the negative gradient for every training example, making it fluctuate heavily but working well with a progressive reduction in learning rate. Mini-batch gradient descent is a combination of the two previous methods, updating parameters for a batch of n examples (Ruder 2017).

Muller et al. (1998) describes backpropagation as a process where the output of each node can be written mathematically as $X_n = F(W_n, X_{n-1})$. Here, X_n is a vector containing the outputs of the node, W_n is a vector containing the set of tunable parameters relating to the node and X_{n-1} is the input vector into the node.

The partial derivative of E^P with respect to X_n is known and because of that the partial derivative of E^P with respect to W_n and X_{n-1} can be computed.

$$(8)\ \frac{\partial E^P}{\partial W_n} = \frac{\partial F}{\partial W}(W_n, X_{n-1}) * \frac{\partial E^P}{\partial X_n}$$

$$(9)\ \frac{\partial E^P}{\partial X_{n-1}} = \frac{\partial F}{\partial X}(W_n, X_{n-1}) * \frac{\partial E^P}{\partial X_n}$$

The Jacobian or Jacobi matrix is a matrix containing partial derivatives of variables that make up a multivariable vector function. The expression $dF/dW(W_n, X_{n-1})$ is the Jacobian of F with respect to W evaluated at the point (W_n, X_{n-1}), and $dF/dX(W_n, X_{n-1})$ is the Jacobian of F with respect to X. From this, the multidimensional derivative of all values in the set of W_n and X_{n-1} can be modeled and the result of their corresponding changes on the cost function, E, can be determined (Muller et al. 1998).

These equations are applied to all nodes in reverse order from the last layer to the first layer, hence all the partial derivatives of the cost function with respect to all the parameters can be computed. This process is called backpropagation.

2.8 Recurrent Neural Networks

In FNNs it is assumed that all inputs and outputs are independent of each other, in Recurrent Neural Networks (RNNs) on the other hand, there is a dependence between an output Y_t and all previous outputs $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-n}$.

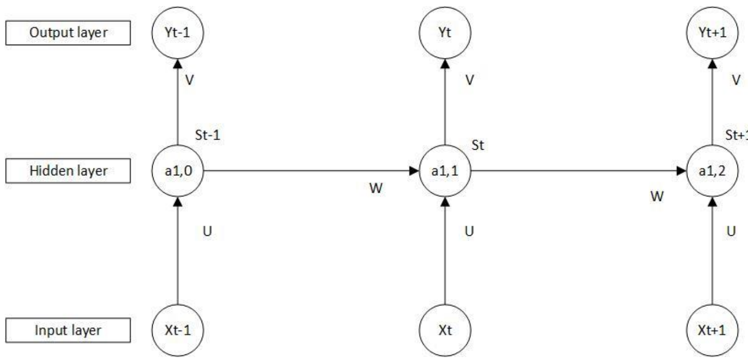


Figure 4. Schematic figure of RNN architecture. Adapted and redrawn from LeCun, Bengio, & Hinton (2015).

In figure (3), X_t is the input vector into the network at step t. S_t is the hidden state at step t and constitutes the memory of the network. These are the hidden nodes containing activation functions (LeCun, Bengio & Hinton 2015). The hidden state at S_t can be expressed as a function of the sum of the input weight multiplied by the input vector and the recurrent weight matrix W times the previous hidden state (equation 10).

(10) $S_t = f(Ux_t + WS_{t-1})$

(11) $Y_t = f(US_t)$

(12) $E_t = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$

The input X_t is multiplied by the associated weight to the edge that connects the input node and the hidden node. This is added to WS_{t-1} which is the long-term memory of the network. Finally, the weighted sum is applied on an activation function.

The error between the actual value Y_i and the predicted value Y-hat can is calculated with a loss function like mean squared error (equation 12).

An important difference between RNNs and FNNs is that unlike FNNs that use different tunable parameters between all edges and nodes, RNNs use the same parameters *i.e.* U, V, W (figure 4). This is because outputs are dependent on each other.

Two issues with RNNs are the vanishing- and exploding gradient problems (Pascanu, Mikolov & Bengio 2013). This arises from the fact that RNNs do not only have edges between nodes from the input, to the hidden and onward to the output layer like FNNs do. Here, there are recurrent weights W that connect hidden layers from $S_{t-1}, S_{t-2}, \ldots, S_{t-n}$. Since the process of calculating the negative gradient and using backpropagation to adjust the tunable parameters involves all weights and biases that contributed to the error E_t (equation 12) there is a chain of multiplication (equation 10) that can lead to unreasonably small (vanishing gradient) or large (exploding gradient) numbers. This prevents RNNs from utilizing layers too many steps back in time.

If the recurrent weights become too small there is a vanishing gradient problem and it prevents the network from learning properly, if it becomes too large there is a risk of an exploding gradient causing the weights to change too much from every training sample.

Hochreiter & Schmidhuber (1997) propose a solution to the vanishing gradient problem in the form of a modified architecture of RNNs, Long-Short Term Memory networks (LSTMs).

2.9 Long-Short Term Memory Networks

LSTMs is a special type of RNNs capable of handling long term dependencies being resistant to the vanishing gradient problem. It has a more advanced architecture than RNNs with several additions.

In LSTMs there is a cell state S_t that convey a flow of information from one module to the next. The transmission from S_t to S_{t+1} is regulated by components called gates.

The data that is removed from the cell state is regulated by a forget gate layer. This layer uses Y_{t-1} and the input into the network x_t concatenated together into a matrix. This is multiplied by associated weights (by the dot product) and a bias term is added. This value is then inserted into a sigmoid activation function (σ) to attain a value between [0, 1] and multiplied with S_{t-1}. A value of zero means that no information should be transmitted within the cell state and a value of one means that all information should be transmitted (Olah 2015).

(13) $f_t = \sigma(W_f \cdot [Y_{t-1}, x_t] + b_f)$

The second type of gate is an input gate (i_t) together with a tanh layer (G_t) which transform values to the interval [-1, 1]. This section of the module determines what information that should be stored within the cell state.

(14) $\tanh(x) = \frac{2}{1+e^{-2x}} - 1$

(15) $i_t = \sigma(W_i \cdot [Y_{t-1}, x_t] + b_i)$

(16) $G_t = \tanh(W_c \cdot [Y_{t-1}, x_t] + b_c)$

The input gates decide which values that should be updated, and the tanh layer determines candidate values that could be added to the cell state.

The new cell state at S_t is then calculated using equations (13, 15, 16).

(17) $S_t = f_t * S_{t-1} + i_t * G_t$

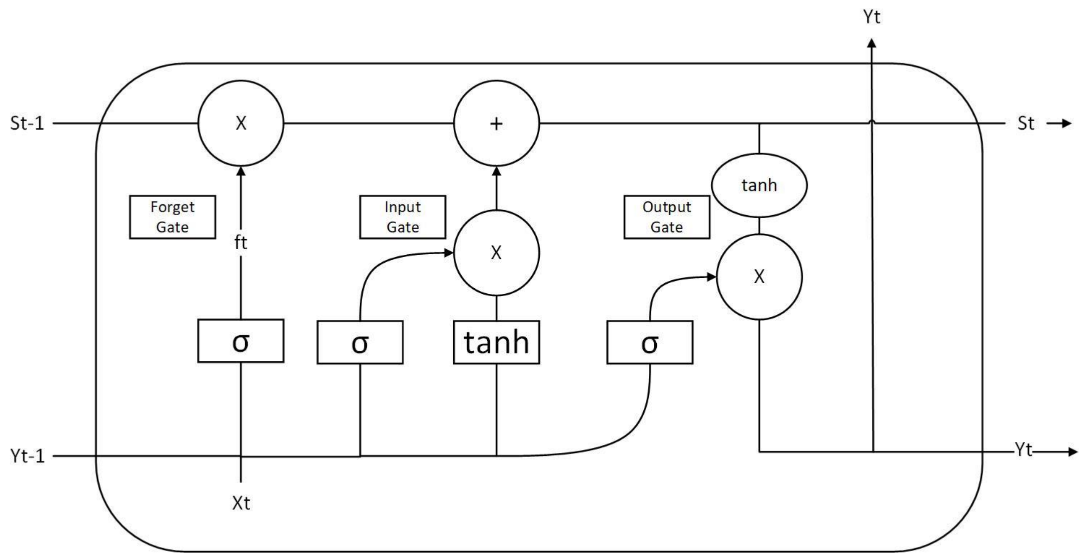


Figure 5. A module in an LSTM network. Adapted and redrawn from Olah (2015).

Equation (17) determines the final input into the module. The second part of the LSTM architecture deals with the final output. First the previous output together with the input is multiplied with associated weights and a bias term is added (equation 18). This weighted sum is put through another sigmoid activation function to determine what parts of the input is going to affect the output. In the next step the final output is the product of (equation 19) and a tanh layer that uses the cell state calculated in (equation 17). The complete architecture of a basic LSTM is displayed in figure (4).

(18) $O_t = \sigma(W_0 \cdot [Y_{t-1}, x_t] + b_0)$

(19) $Y_t = O_t * \tanh(S_t)$

2.10 Temporal Convolutional Networks

Temporal Convolutional Networks (TCNs) build on the more familiar Convolutional Neural Networks (CNNs) that has been a dominating architecture for developing deep computer vision models. Famous architectures include You Only Look Once (Redmon, Divvala, Girshick & Farhadi 2015) and Single Shot MultiBox Detector (Liu et al. 2016) that can perform real time object detection and LeNet (LeCun, Bottou, Bengio & Haffner 1998) that was one of the first deep models that could learn to recognize handwritten digits.

TCNs could be implemented in a variety of ways with different tweaks but the version that will be discussed in this section is the one used by Bai, Kolter & Koltun (2018). In this architecture, instead of using a cell state to preserve information from previous outputs as in LSTMs, TCNs use connection between previous hidden layers configured with two hyperparameters: dilation factor and filter size.

The dilation factor (d) decides how many steps back in a layer that connections should be made between the output column and previous hidden nodes. As can be seen in figure (5), a d = 1 means that there should be no interval between previous nodes while d = 2 creates a connection between every second node to the output column. Filter size (k) decides how many connections there should be in total between a certain layer and the output layer. With k = 3 (figure 6) there are three connection between every layer and the output layer (Bai, Kolter Koltun 2018).

Because d = 1 is used in the input layer, all information from the entire network is stored in later hidden layers generating a “receptive field”. This is an effective way of reducing the number of tunable parameters that must go through optimization thus increasing the training speed. The process of creating this is together with kernels for feature extraction that generate filter maps is called dilated convolution. The advantage of dilated convolutions is the ability to increase the size of the receptive field exponentially while the tunable parameters grow linearly, and the technique was first introduced as a tool in semantic segmentation (pixel-wise image classification) (Yu & Koltun 2015).

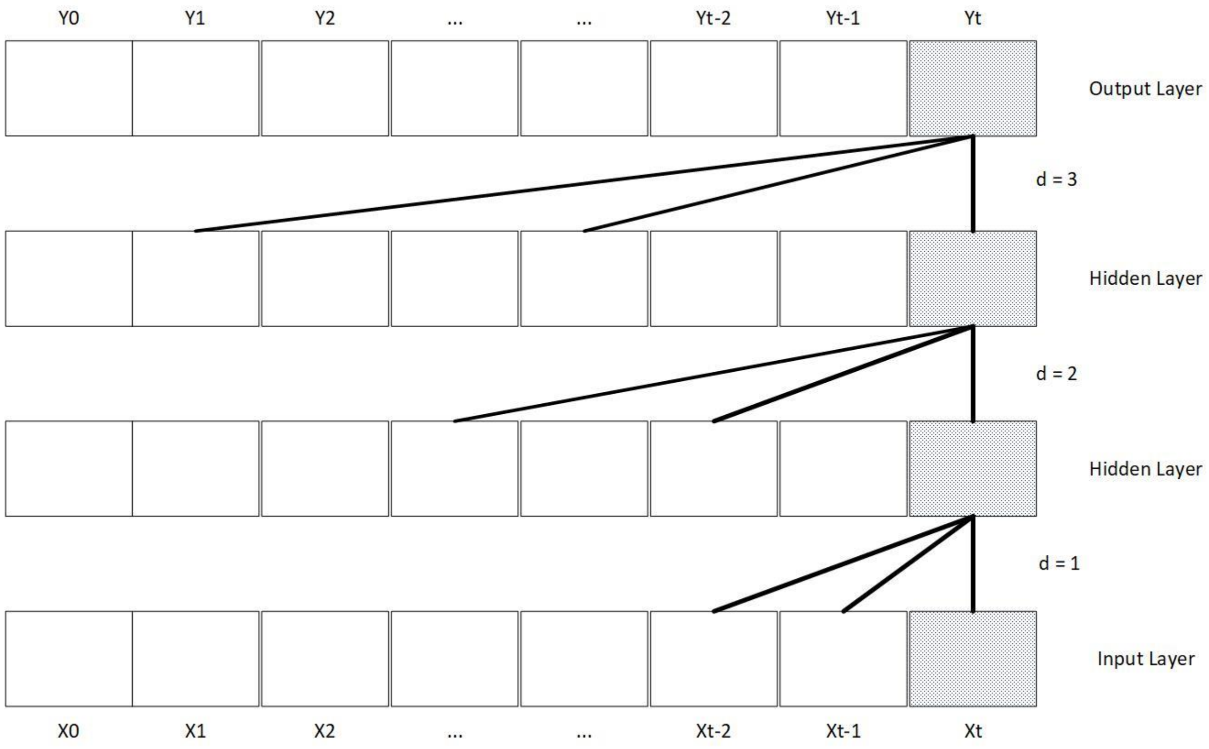


Figure 6. Schematic image of the TCN architecture with dilation factors 1, 2, and 3 and filter size = 3. Connections only shown for the output column Yt, but these can be extrapolated to every node in the output layer. Adapted and redrawn form Bai, Kolter & Koltun (2018).

Another important concept in TCNs are residual blocks (Bai, Kolter & Koltun 2018). These pool together n nodes (decided by parameters k, d) and the result is added to the input to create the final output of the block. In figure (6) the shaded column apart from the input nodes represents a residual block (figure 7).

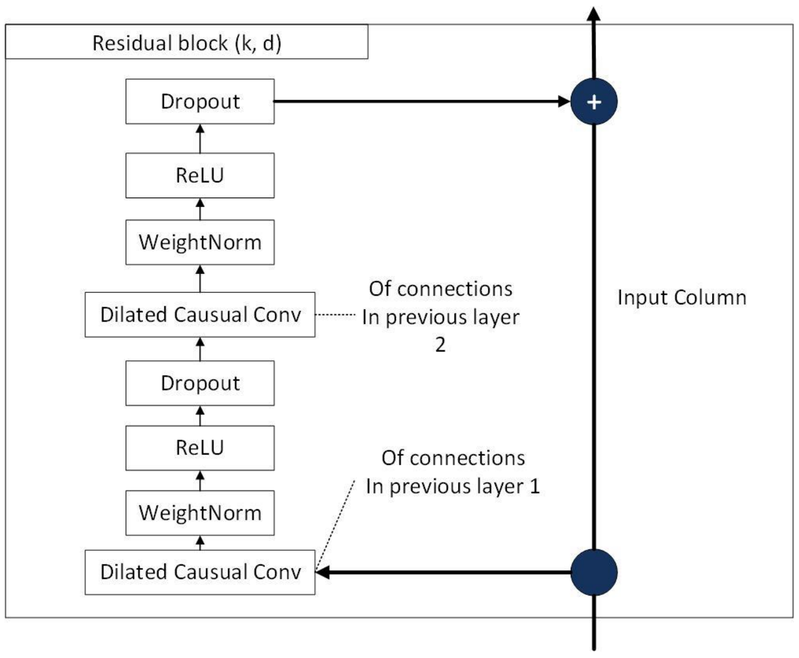


Figure 7. A residual block unit in TCNs. Adapted and redrawn form Bai, Kolter & Koltun (2018).

Inside the residual blocks there is a sequence of data transformations carried out. Weight normalization or WeightNorm (equation 21) is a technique that normalizes the weight vectors of each node and instead of tuning the weights w and bias term b (equation 20) using gradient descent, a parameter vector v and scalar parameter g is optimized (equation 21). Salimans & Kingma (2016) propose that this technique makes gradient descent converge to a local minimum faster.

(20) $Y = \sigma(w \cdot x + b)$

(21) $w = \frac{g}{|v|} v$

The next step in the residual block is a unit with the activation function rectified linear unit (ReLU) which is $f(x) = max(x, 0)$. LeCun, Bengio & Hinton (2015) describes this function as particularly useful in networks with many layers leading to a faster learning.

Finally, dropout is applied. This is a technique where random neurons are ignored during training in order to minimize the risk of overfitting. That is, their incoming information is not part of the cost function and their tunable parameters are not changed through backpropagation for that training iteration (Srivastava, Hinton, Krizhevsky & Salakhutdinov 2014).

The residual block described in figure (7) can also be viewed in the context of the network as a graph with edges and nodes (figure 8). The flow of information from the input layers, to consequent hidden layers and the final output layer and the transformations that happens in between constitutes a residual block.

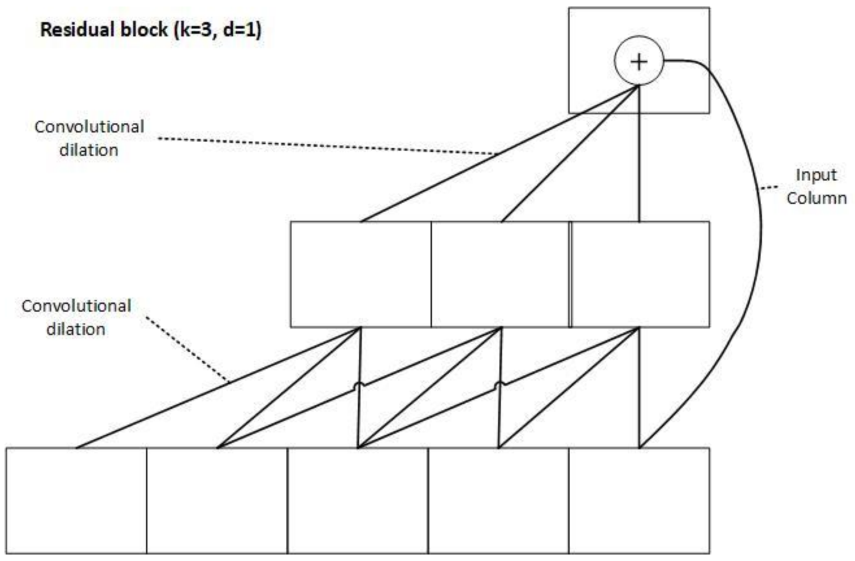


Figure 8. A residual block viewed in the context of network architecture with nodes and edges. Adapted and redrawn from Bai, Kolter & Koltun (2018).

2.11 Hidden Markov Models for Time Series

There are two basic types of Markov models, Markov chains for discrete states and Markov processes for continuous states. A Markov chain is a mathematical system that changes between different states. The set of all possible states are called the state space. In figure (8) there is a Markov Chain with state space = {A, B}. If the system is in state A, there is a 90% probability that it will remain in state A in the next time step and 10% probability that it will change from state A to state B. When modelling a Markov chain, it is presented as an $n \times n$ transition matrix where n is the cardinality of the state space and the cells contain transition probabilities. Markov chains are a common technique for creating dynamic mathematical systems and one example is Googles page rank algorithm (Rai 2016).

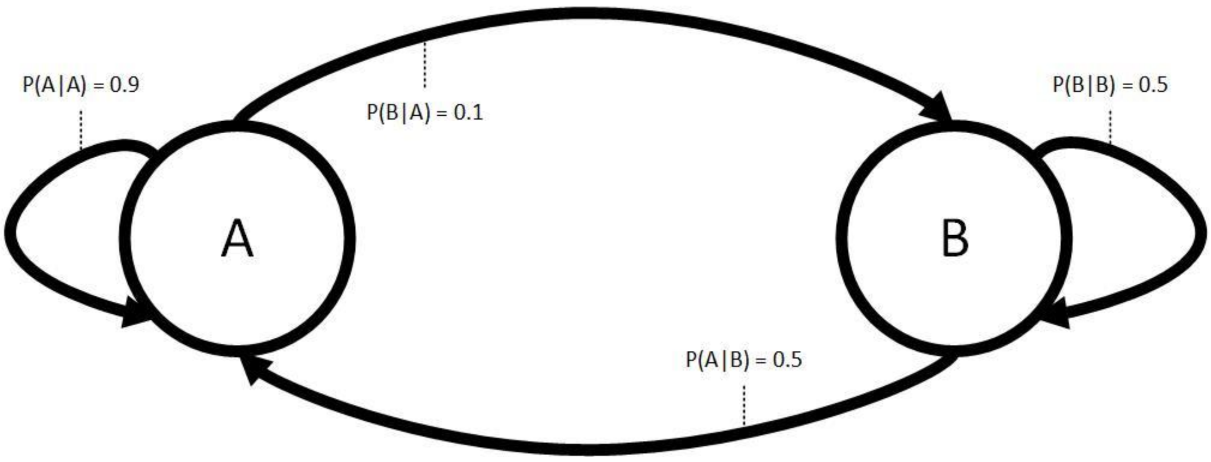


Figure 9. Illustration of a Markov chain for discrete states.

Hidden Markov Models (HMMs) differ from regular Markov models in that the probabilities of changing state (or remaining in the same state) are determined by derived probability distributions by using a training set, as opposed to using fixed probabilities.

Furthermore, the Markov property says that the state at C_{t+1} can depend solely on C_t. (Montgomery, Jennings & Kulahci 2015, p. 502). This is called a first order Markov model but there are adaptations where the current state can depend on several successive states $Pr(C_{t+1} \mid C_t, C_{t-1}, \dots, C_{t-n})$ and these models are called a Markov model of order n where n is the number of previous states that affect the current state (Ky & Tuyen 2018). These Markov models are particularly suited for time series forecasting as they acquire the ability to model patterns like trends and cycles.

3 Problem area

This chapter provides information regarding other research that has been done related to the present study. Furthermore, it also provides the research questions, aim and motivation for the study and the objectives that will be achieved.

3.1 Related Research

A performance comparison between LSTM and ARIMA conducted by Siami-Namin & Siami-Namin (2018) showed that the LSTM algorithm had six times lower RMSE than ARIMA on six different economic time series datasets. Although promising, the authors did not state in detail how the models were developed and as ARIMA must be configured for each dataset more data is required to be able to understand their respective strengths and weaknesses. Flores et al. (2016) investigated the robustness for noise in training data for ARIMA and FNNs and found similar increase in prediction error as the level of noise increased. Furthermore, Chan, Xu & Qi (2018) compared several versions of ARIMA to FNNs in their ability to forecast throughput of containers in a harbor area and found that ARIMA had significantly better performance.

Ky &Tuyen (2018) developed an HMM for time series forecasting by using historic stock price data as a training set and divided the time series into intervals which represented different states in the Markov model. The author found that an HMM with number of states adapted to the dataset and an order higher than one (adapted to the dataset) can outperform several neural network models. This makes it an interesting candidate with a different approach to time series forecasting. HMMs are heavily stochastic models because of their completely probabilistic architecture of states and movements between states which might fit particularly well for stochastic data like stock prices. In this study, they will be evaluated on more traditional time series data with trends and seasonality.

Results from Bai, Kolter & Koltun (2018) show that a modified version of CNNs specialized for sequence analysis can perform better and learn faster than traditional recurrent networks. Two of the reported experiments were predicting sequences of digits using the MNIST database (LeCun & Cortes, n.d); and predicting sequences of words using the Word Wiki-103 dataset (Merity 2016). The authors compared the performance of TCNs to LSTMs and found that TCNs have better ability to retain information from many steps back in time being able to process inputs from 250 steps back. Furthermore, they reached higher accuracy than LSTMs when being fully trained as well as learning faster when applied on sequential datasets. With these outcomes it is a relevant scientific inquiry as to whether they can also perform better on time series forecasting of continuous data, a question which has not yet been investigated.

Table 1. The algorithms that will be evaluated in the study.

Algorithm	Pros	Cons
Recurrent Neural Network	Simpler than LSTM fewer hyperparameters that can be tuned.	Unable to use more than ~10 previous steps in time because of vanishing- and exploding gradient problems.
Long-Short Term Memory Network	Does not risk vanishing gradient. Can use any number of independent variables and any number of previous time steps.	Needs a lot of training data. There is no deterministic developmental process, must be developed individually for each dataset. There is loss of data between each module.
Temporal Convolutional Network	Number of weights increase linearly while receptive field increase exponentially.	Has only been studies for sequential problems never for time series. Takes more time to train than RNN and LSTM.
ARIMA	Standardized developmental process. Good insight into how the algorithm works. No need for hyperparameter tuning.	Relatively simple model. Data must be stationary.
Hidden Markov Model	Good at modelling stochastic behavior.	Cannot use independent variables. Difficult to develop due to weak library support in R and Python.

3.2 Aim

The aim of this study is to see if there is a significant difference between the novel algorithms temporal convolutional network and higher order Markov model as compared to the more classic recurrent neural network, long-short term memory network and ARIMA for forecasting on a continuous time series dataset. This will be investigated by collecting data from two Swedish theme parks and use this to predict future number of visitors, which constitutes a regular business use (forecasting product demand/resources).

3.3 Motivation

Time series prediction is an important technique for both scientific research and optimizing business processes. With many different algorithms to choose from, it can be difficult and time consuming to find the best solution. The motivation for this study is to present a clear performance evaluation for state-of-the-art algorithms for time series forecasting and deliver a detailed description of data exploration and model development as to allow reproducibility. Furthermore, if the result is a model able to reliably predict the future number of visitors, it is possible to optimize both staffing and deliveries of food and drinks. Thereby increasing employee satisfaction as well as efficiency while at the same time limiting the companies' environmental impact.

3.4 Research Questions

This study aims to answer whether there is a significant difference between the newly publicized algorithms temporal convolutional network (Koltun et al. 2018) and A higher order Markov model (Ky & Tuyen 2018) as compared to the classical alternatives (RNN, LSTM and ARIMA).

1. Which one out of the five independently developed (using optimal variables and parameters) algorithms have the best performance as measured by RMSE?
2. Which of the algorithms have the best performance as measured by RMSE when using no independent variables?

3.5 Delimitations

Based on a pre-study of the literature, five of the most promising algorithms were selected (table 1). ETS- and EMWA models will not be investigated. The data that will be used will be based on number of visitors per day between 2010-2018 in Skara Sommarland AB and 2011-2018 in AB Furuviksparken. The Skara Sommarland AB will also provide the number of booked days for a camping. This data will be fused with Lantmet and SMHI for weather data and Google Trends for keyword search frequency. Other data sources will not be considered.

3.6 Hypothesis

The hypothesis is that the independent variables in the complete datasets will explain a significant amount of the variance in number of visitors. Thus, all algorithms apart from HMMs are presumed to either reach or be close to <20% MAPE. Also, the literature suggests that LSTMs are superior to RNNs regarding the number of previous time steps that can be taken into consideration. Furthermore Siami-Namin & Siami-Namin (2018) showed that LSTMs can predict time series much more accurately than ARIMA in some settings. The conclusion is thereby that LSTMs are a likely top performer.

The performance of TCNs on this dataset is highly uncertain. Bai, Kolter & Koltun (2018) showed that TCNs can overperform LSTMs significantly on sequential problems but they are still untested on continuous time series data. Although theoretically, with their ability to cover much more tunable parameters than LSTMs in their outputs they should have the best ability to forecast visitors and get a larger advantage as there are more independent variables and previous time steps considered.

HMMs (chapter 2.11) are assumed to have the best performance when compared to other algorithms with no independent variables used but the worst performance when all algorithms are compared with optimal variable inputs and hyperparameter configurations.

3.7 Objectives

This study encompasses four objectives that must be completed in order to answer the research questions and reach the aim.

1. Acquire data describing number of visitors and sold tickets to the Skara Sommarland camping through representatives of the two companies. Complement this with weather data from Lantmet and SMHI as well and Google Trends.

2. Handle problems with data cleaning and remove outliers. Thoroughly investigate collected variables to understand their importance as future input variables and if it is possible to construct derived variables.

3. Use the prepared datasets to develop LSTM, RNN, TCN and ARIMA using Python and HMM using R. The initial architectures will be based on Bai, Kolter & Koltun (2018) for TCN, Siami-Namin & Siami-Namin (2018) for LSTM and Ky & Tuyen (2018) for HMM. Arima will be developed according to the Box-Jenkins methodology (chapter 2.6.7 & 2.6.8) and RNN will have the same number of nodes and layers as the LSTM network. The neural networks models will go through hyperparameter tuning using sequential grid search.

4. Evaluate algorithmic performance both optimally trained with independent variables and without independent variables, using hypothesis testing on acquired RMSE values.

4 Research Methods

In this section, the reasoning behind the choice of scientific methods and their implementation are described. Furthermore, possible alternatives are explored.

4.1 Exploratory Data Analysis

John W. Tukey wrote about the need for this method in his book Exploratory Data Analysis from 1977. In the book he suggests that confirmatory statistics with the advent of hypothesis testing had become too important and that the preliminary descriptive statistics had been neglected. By exploring data and finding patterns and limitations, it is possible to get a broader and more deep understanding about a certain sample. The author writes that before hypothesis testing became dominating, descriptive statistics was the only analysis researchers did. Later, it became common to only carry out a minimal amount of exploration in order to ensure that the correct hypothesis test was selected.

When developing machine learning- and statistical models it is paramount to create the best possible dataset in order to achieve good results. By using Exploratory Data Analysis, it will provide a process for creating a detailed data understanding and document the developmental process, allowing subsequent researchers to replicate and improve upon this work. The use of this method will also be informed by the Cross-Industry Standard for Data Mining (CRISP-DM) (Shearer 2000). This is a process model that has become an industry standard for carrying out data mining projects while maximizing the chances of a high-quality result.

The employment of this method is aimed at maximizing the control in preparing the data and allowing future researchers to understand how the algorithmic performance was generated, from start to end.

4.2 Experiment

The experimental method is characterized by setting up experiments aimed at disproving a certain hypothesis, regularly with the support of statistical hypothesis testing (Berndtsson, Hansson, Olsson & Lundell 2008 p. 65). This is a good way to answer two of the three formulated research questions in this study (question 2 & 3).

There are three basic principles of experimental design (Toutenburg, Shalabh, Fienberg & Olkin 2009, pp. 4-5). The first is *Fischer's Principle of Replication*. An experiment must be done on several units in order to determine the sampling error. In this study, a unit will be one day with associated variables throughout the years 2010-2018.

The second principle is that of *Randomization*. This means that units must be assigned randomly to treatment and control groups. Also, the conditions under which treatment is delivered should be as similar as possible. In the present case, time series models will be developed on the same training- and test data and used to predict on the same validation data. The process of developing these models will serve as treatment and this will not affect the units within the groups. Because of this it is not necessary to randomize observations into different groups. However, care will be taken to make the model development as comparable as possible. To do this, the Box-Jenkins methodology will be used to develop ARIMA, network architectures from previous studies will be used together with evolutionary search to develop neural network models and the HMM will be the same as that published by Ky & Tuyen (2018). Furthermore, the cleaning, outlier removal and creation of derived variables will be carefully examined using the method described in chapter 4.2.

The last and third principle is that of *Control of Variance*. The variance can be controlled by dividing a set of data into smaller blocks that have similar characteristics such as age and sex. An advantage of the experiment in this study is that the treatment does not affect the units of study, hence there can be a complete control of variance.

4.2.1 Evaluation Metrics

Two metrics will be used to answer the two research questions. Root Mean Squared Error (RMSE) will be used to assess the difference between the chosen algorithms with respect to independent variables and without independent variables because it gives larger weight to outliers than mean average error (MAE) which is more affected by many small errors, as suggested by Chai (2014). This makes it more sensitive to few large errors and a comparatively low RMSE thus indicate predictive stability more than MAE would.

RMSE will be calculated on validation data from 2018 by weekly intervals where every week will represent one sample thus generating 12 samples per algorithm in total.

(22) RMSE = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2}$

Where n is the number of samples, y_i is the actual value and y-hat is the predicted value.

The second metric Mean Average Precision Error (MAPE) shows average percentual error (Fiores et al. 2016). For a model to be a useful tool in practice, the organizations have communicated that MAPE should be <20%. It is easy to understand and interpret and is thus a reasonable tool to answer the feasibility of a possible implementation of one of the algorithms.

(23) MAPE = $\frac{1}{n}\sum_{i=1}^{n} | \frac{y_i - \hat{y}}{y_i} |$

4.3 Implementation of Research Methods

In the initial phase of this study discussions with the CEOs of Skara Sommarland AB and Furuviksparken AB will be had in order to understand their view of relevant explanatory variables and to acquire historic datasets. This data will be fused with sets of weather variables using SMHI and Lantmet. Lastly, Google Trends will be used to create two sets containing search frequency on the keywords “Skara Sommarland” and “Furuviksparken”.

The second stage will employ the exploratory data analysis-method tailored for this study by using a workflow informed by the data preparation stage in CRISP-DM. Here, data will be analyzed, the quality of the raw data will be examined, and relevant variables will be selected. Going further, the selected data will be cleaned by using techniques for outlier removal and imputing missing values. With these preliminary datasets it will be inspected to see if it is possible to derive new variables that have relevant explanatory power. Finally, the different datasets (Skara Sommarland/Furuviksparken, Google Trends, SMHI, Lantmet) will be fused into one concluding material for each of the two organizations (appendix 1).

With the completion of the data collection and preparation, the experimental phase begins. The selected algorithms will be developed using programming languages R and Python. Using data between 2010-2017 these will be trained and tested. In addition, selection of hyperparameters will be done using evolutionary search as to minimize the risk of performance differences because of uncontrolled choices of model configurations. Data from the year 2018 will be saved so it can be completely unused for obtaining model predictions and through this, the two metrics RMSE and MAPE complemented by mean average error (MAE).

The MAPE results will be used to answer whether one or several out of the models could be used to automate visitor forecasting in the respective organizations. RMSE on the other hand will be used to do hypothesis testing to see if there is a statistical significance in the predictive power between the selected algorithms, both with and without independent variables as two different experiments.

To confirm whether there is a significant difference between the investigated algorithms, a Kruskal-Wallis H-test will be used to answer if there is a difference in the daily prediction errors.

4.4 Alternative Methods

An alternative method for collecting relevant explanatory data for this task could be to use interviews with company representatives (Berndtsson et al. 2008 pp. 60-62). This method has the advantage of generating an in-depth understanding of respondents view of the problem. It is likely that people that have worked for years in an organization could have experience to contribute to the development of the time series models. However, the focus of this study is to compare the performance of the algorithms by developing them in a controlled manner and limiting uncontrolled influences that could raise questions as to whether the predictive power depends on things like hyperparameter tuning or other configurations. The delimitations raised in chapter 3.6 should be enough for this.

Another method that has some interesting features is the case study (Berndtsson et al. 2008 pp. 62-63). This method is characterized by conducting a deep investigation into one or a few objects of study. The focus of this study could be to investigate the algorithmic performance in the specific organizations from where the data is collected. Nevertheless, regardless of whether the final dataset display trends, cycles or noise – the final evaluation will still provide a generalizable answer to their ability to accurately predict time series data in that setting.

Lastly, implementation as a choice of method was considered. This method is about implementing a solution and study how well it works in a certain setting. This is a reasonable choice for this study where one or more of the algorithms could have been developed, put into production and the impact on the organizations could have been analyzed. Another interesting research question could have been to investigate how to best implement a decision support system, which could pose a topic for a future study based on this work.

5 Data Preparation

In this section the data collection procedures are described along with an exploratory analysis (section 4.1). As the same variables were collected for both theme parks only one figure per park per variable is shown unless the data explored displayed a relevant difference between the two parks.

5.1 Data Collection

5.1.1 Organizational Data

Organizational data were provided by Skara Sommarland in the form of number of visitors per day between the years 2010-2018 and the months May to August when the park had been open. They also had the largest camping in Sweden were number of tickets sold had been recorded. Lastly, they also recorded whether there was an event during a specific day or not. Through dialogue with company representatives, it became clear that an event could mean a performing artist or a collaboration with another organization in some way.

Representatives of the Furuvik theme park were not able to provide organizational data other than number of visitors per day for the years 2011-2018.

5.1.2 Weather Data

Weather data was collected from two databases, Lantmet and SMHI. The choice of provider depended on the proximity of the measurement stations to the respective organizations and the types of measurements that were done at the different stations.

For Skara Sommarland AB, Götala weather station was used to collect solar irradiance (w/m^2), min-, max- and average daily temperatures (degrees Celsius), min-, max- and average relative humidity (%), average rainfall (mm), degree days and wind speed (m/s). The choice of this station was suitable because it was only about 6 km away from the park. This station was provided by the Lantmet database.

Further variables were collected from SMHI's Hällum weather station 36 km away from the park. This was the closest measurement station that recorded air pressure (kPa), cloud base (km) and cloud cover (%).These datasets had one measurement per hour. Since the dependent variable is observed as one value per day these cloud values were summarized. This was done by taking the average between 09:00 and 17:00 and using that as the value for each day. Cloud Base was summarized by averaging an entire day because of fewer and more irregular observations.

The closest measurement station for Furuvik was the Gävle weather station 14.2 km away from the park. Solar Irradiance was not measured by any stations within reasonable distance from the park. For air pressure, Örskär was the closest station 105.9 km way.

Table 2. Summary of weather variables collected.

Variable	Unit	Comment	Provider: Station
Solar Irradiance	Watt/m^2	Describes the amount of solar energy that reaches the ground.	**Sommarland** Lantmet: Götala **Furuvik** N/A
Min-, Max-, Average Temperature	Degrees Celsius	The lowest-, highest-, and average recorded temperature of a day.	**Sommarland** Lantmet: Götala **Furuvik** SMHI: Gävle
Min-, Max- and Average Humidity	Percentages	The lowest-, highest, and average recorded relative humidity. With 100% the air is saturated with water vapor.	**Sommarland** Lantmet: Götala **Furuvik** SMHI: Gävle
Average Rainfall	Millimeters	Rain in millimeters collected throughout a day.	**Sommarland** Lantmet: Götala **Furuvik** SMHI: Gävle
Wind Speed	Meters/Second	Daily average.	**Sommarland** Lantmet: Götala **Furuvik** SMHI: Gävle
Air Pressure	Kilopascal	Daily air pressure recorded once per hour.	**Sommarland** SMHI: Hällum **Furuvik** SMHI: Örskär
Cloud Base	Kilometers	The distance between the ground and the lowest cloud layer. Recorded once per hour.	**Sommarland** SMHI: Hällum **Furuvik** SMHI: Gävle
Cloud Cover	Percentages	The percentage of the visible sky covered with clouds. Measured by laser. Recorded once per hour.	**Sommarland** SMHI: Hällum **Furuvik** SMHI: Gävle

5.1.3 Google Trends

Google (Alphabet, Inc) provides a functionality where it is possible to view number of keyword searches per date unit. This comes in the form of a relative index which mean values can range from 1-100 based on the selected interval. For Skara Sommarland the keyword “Skara Sommarland” was used. Data was collected individually for each year by selecting the first day of the previous year when the park was open and ending in the last day of the actual year. This was downloaded in the form of a csv file and the index values for the actual year was stored. This procedure was repeated once for every year between 2010-2018. For Furuvik the keyword “Furuviksparken” was used and collected in the same manner.

5.1.4 Temporal Data

Temporal data was extracted from the dates provided by the companies where there were recorded observations with visitors in the parks. By using the date, weekday, week number and month was extracted and stored as individual variables.

5.2 Data Exploration

A major technique for exploring correlations and linear relationships was using regression analysis between the dependent variable and all predictors.

5.2.1 Organizational Variables

The linear relationship between sold camping tickets and visitors display a strong correlation with an R^2 of 65% and p-value of $5.6*10^{-150}$. The encircled column of observations in figure (10) where sold tickets were around zero although number of visitors were > 2000 and < 6000. This was investigated further, and it was revealed that these values were days on the last opening day of the season when the camping had been emptied.

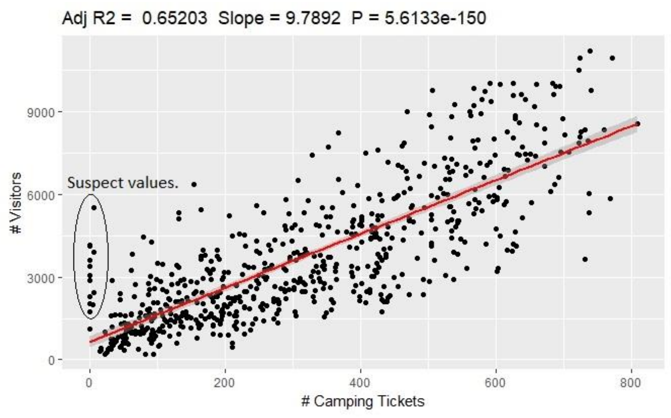


Figure 10. Number of camping tickets sold and number of visitors.

5.2.2 Weather Variables

Solar irradiance captures how much solar energy that hits the ground and could theoretically be an interesting predictor. Figure (11) show a significant correlation between dependent and independent variables although with a smaller amount of variance explained (9.3%). This variable was only available in proximity to the Skara Sommarland park.

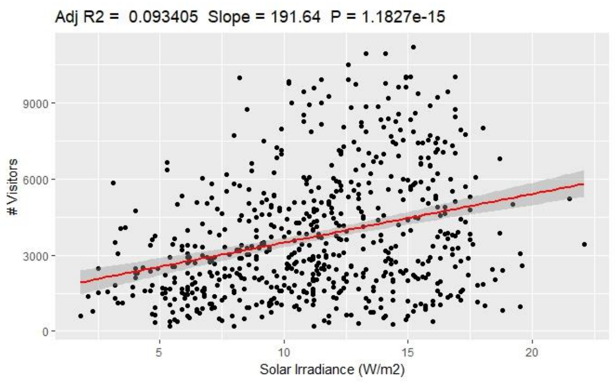


Figure 11. Number of visitors and solar irradiance.

There are three variables relating to daily temperature. The highest- and lowest daily temperatures as well as the average daily temperature (figure 12). The strongest relationship between number of visitors and temperature is found in the highest daily temperature with an R2 of 38.6% and a p-value of 1.88*10^{-70}. Speculatively, this could be because the highest daily temperature is during day time when the park is open. The weakest relationship is with the lowest daily temperature, and this value is most likely recorded during night time when the park is closed, and thus, it affects the number of visitors much less. The average temperature reflects both day- and night time and hence it falls between the two.

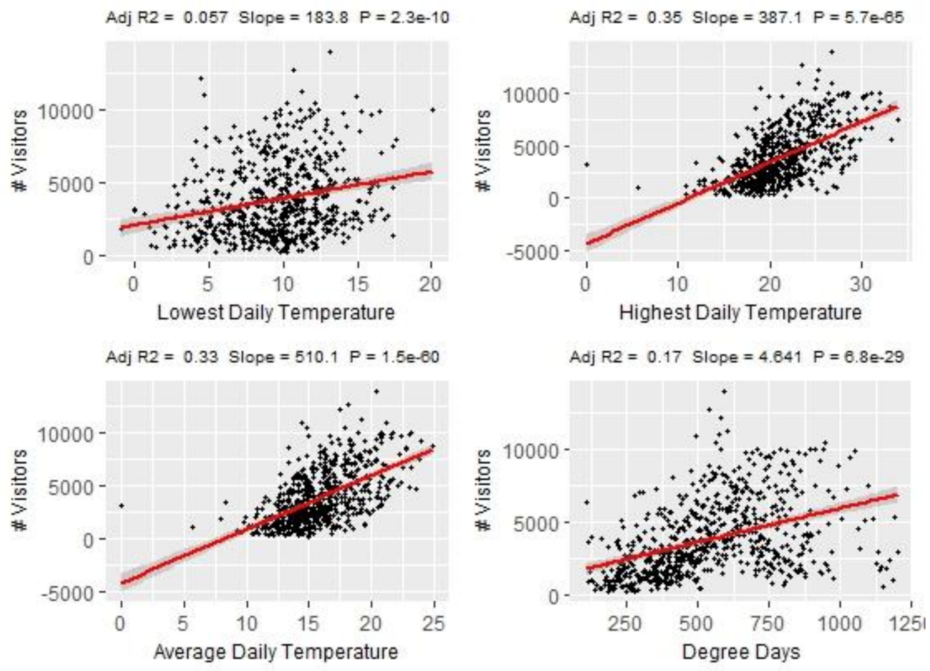


Figure 12. Number of visitors and temperature variables.

Degree days is a value that always increase throughout a year. It is a way to accumulate the occurred degrees through time (figure 12). This value reflects trends more than regular temperature measurements. If an observation has high degree days, it means that it has been warm for a long period. This value starts and the beginning of the year, making it an interesting target for a derived variable that starts at the beginning of the season.

Air pressure is generally associated with good weather and this variable could explain 7% in the variance of number of visitors with a significant linear relationship (figure 13).

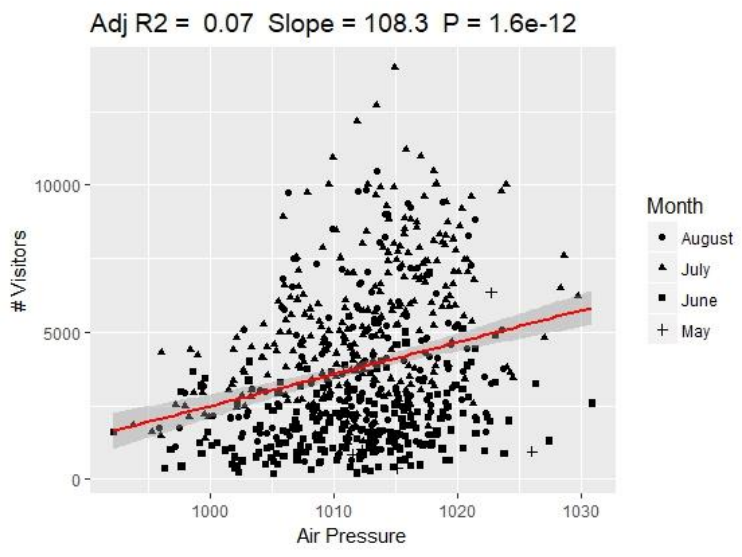


Figure 13. Air pressure and number of visitors.

The distance between the ground and the lowest layer of clouds had a significant linear relationship although weak explanatory potential (R^2 = 4.5%) (figure 14). The amount of the sky covered by clouds, average values between 9-17 had a negative linear relationship with number of visitors. For every percentage of the sky covered with clouds, there tended to be 31.38 less visitors in the park.

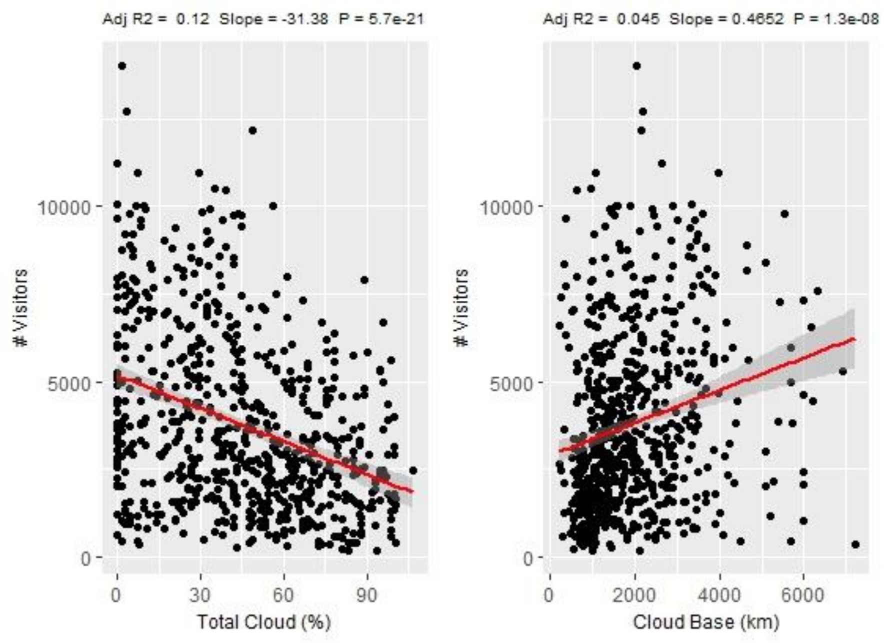


Figure 14. The two cloud variables: total cloud (amount of sky covered with clouds) and cloud base (the distance from the ground to the lowest layer of cloud).

As with temperature, there are three variables relating to humidity. There is a similar pattern here. The highest daily humidity (figure 15) does not have a statistically significant relationship (p = 0.45).

The strongest relationship is between number of visitors and the lowest daily humidity (figure 16) and this is most likely because that value is recorded during day time. The average humidity is significant and explains 7.8% of the variance is number of visitors.

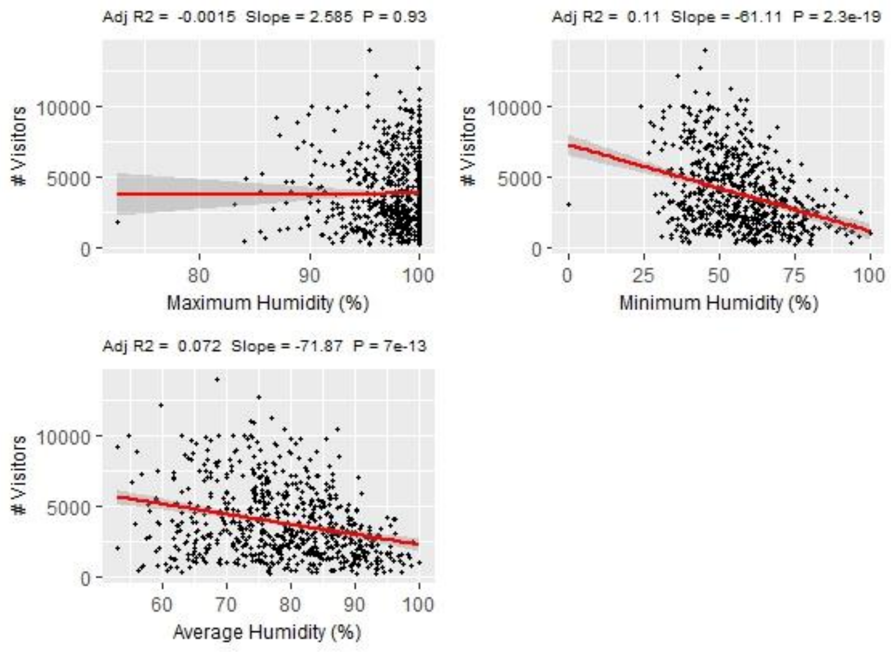


Figure 15. Number of visitors and air humidity.

Average daily rainfall (figure 16) is surprisingly weak as an explanatory variable although significant (R2 = 3.3% and p = 0.001). The reason for this could be that it is not a linear relationship. Less people do not come the more it rains, instead, it might work better a binary or categorical variable.

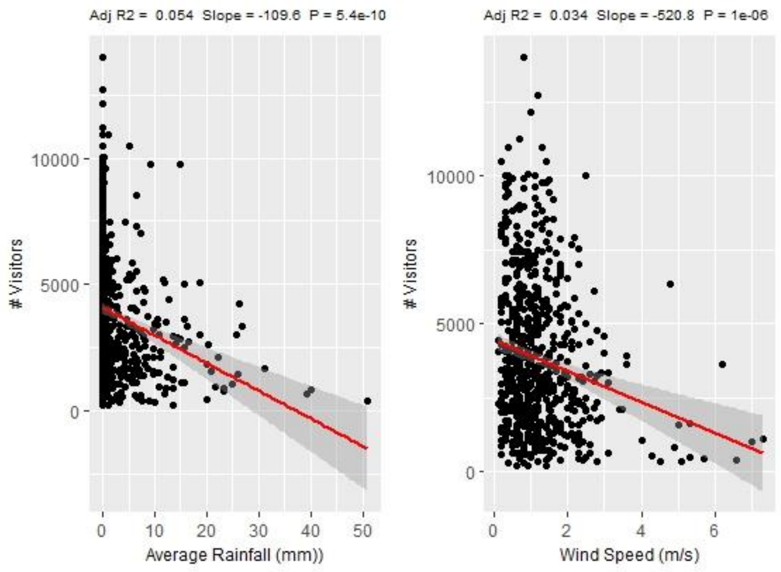


Figure 16. Number of visitors in relation to rainfall and wind speed.

Wind speed display a similar behavior like rainfall. It explains a small amount of variance in the dependent variable (R^2 = 3.6%) with a significant linear relationship (p = $7.3*10^{-7}$). By visually analyzing figure (16) it appears like higher wind speeds above ~3 m/s translate into incredibly low number of visitors while smaller values do not have a linear relationship to number of visitors.

5.2.3 Google Trends

The Google Trends variable reflects search frequency on the keyword "Skara Sommarland". It explains the most variance in number of visitors (R^2 = 46%) out of all variables apart from number of camping tickets sold.

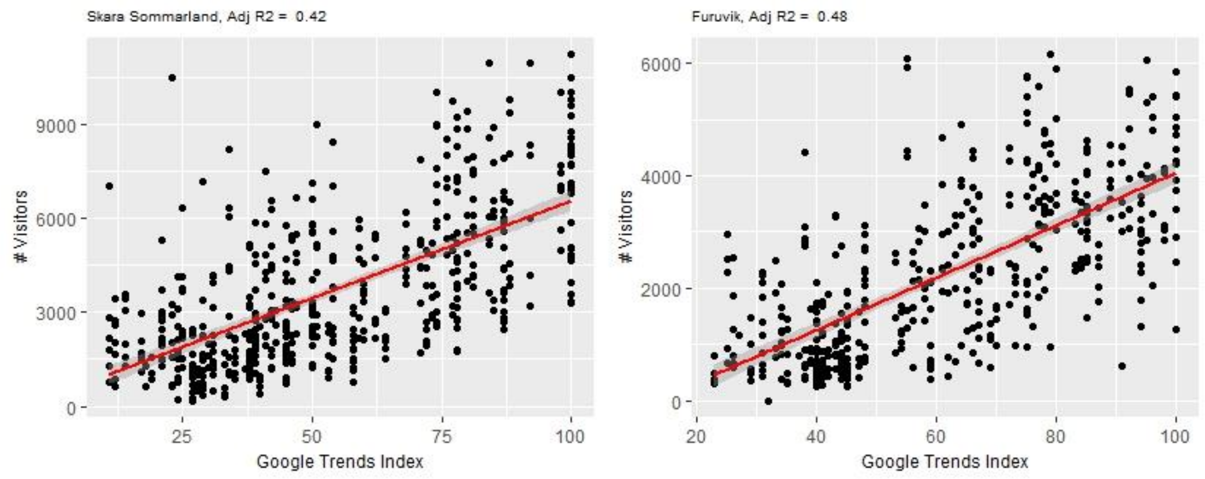


Figure 17. Relationship between Google Trends index weekly values and number of visitors.

5.2.4 Temporal Data

Boxplots was used to visualize the number of visitors per month (figure 18). July tend to have the most visitors. This month also has the highest variance while June has the lowest variance.

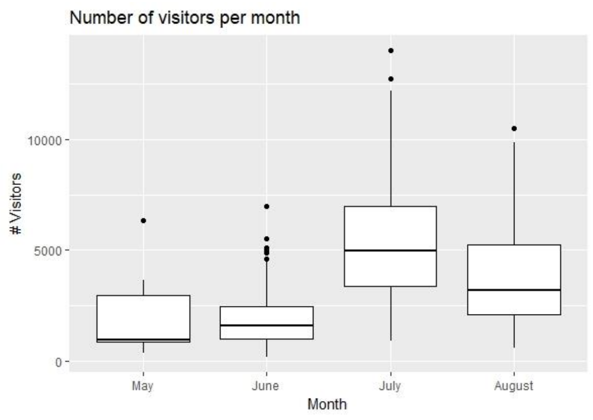


Figure 18. Number of visitors per month.

By drilling deeper into the relationship between visitors and time, a boxplot was used with week number as a category (figure 19). Early weeks in May (weeks 20 and 21) display irregular pattern with very few observations. It is interesting that visitors keep increasing until the last week of July (week 30) and then drops in august. Also, there are many outliers during the last week of august (week 33).

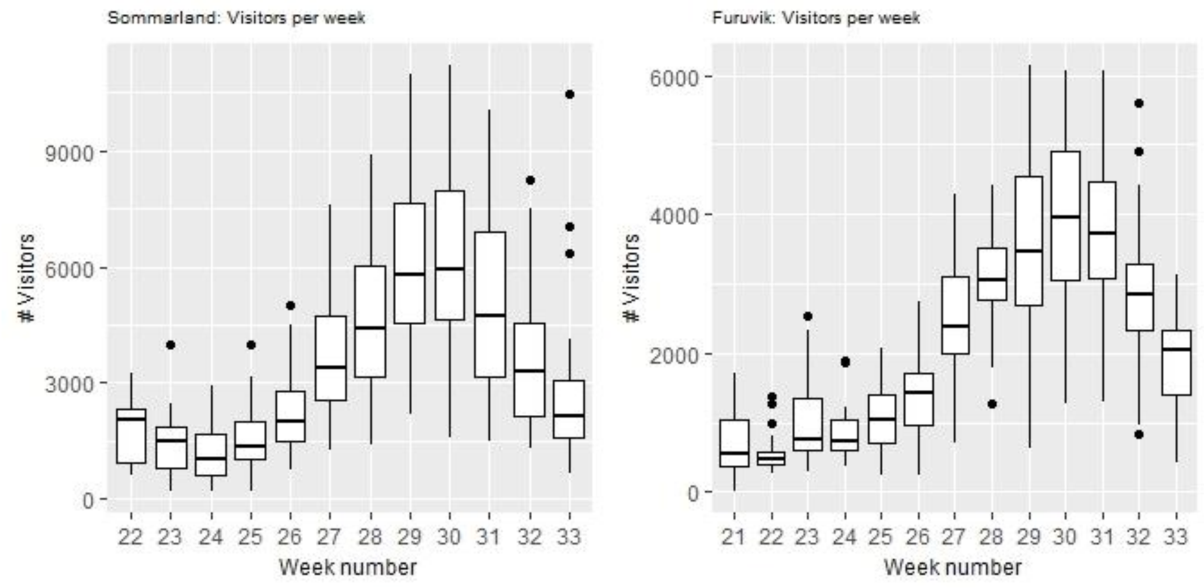


Figure 19. Visitors per week number.

Looking at weekdays as categories, it can be visually concluded that during June, Friday, Saturday and Sunday tend to have the most visitors. In July however, Sunday and Friday are the worst days while Tuesday and Wednesday are the best performers. August favors Saturday as the best day and Monday as the worst (figure 20).

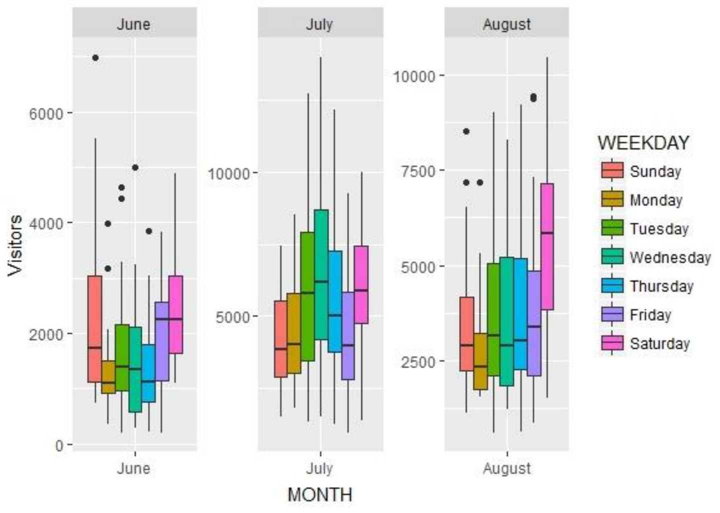


Figure 20. Visitors per weekday per month. First box is Sunday, last is Saturday otherwise in week order.

5.2.5 Variable Contributions to Explanatory Potential

Principal Component Analysis was used to investigate how different continuous variables contribute to the explanatory ability of the dataset and to see if there were variable that explained the same amount of variance. By using two components it can be seen in figure (21) that solar irradiance, air pressure and cloud base have similar influence on the dataset. Total cloud, rainfall and the humidity variables have comparable properties in the variance they explain. Furthermore, the temperature variables together with camping tickets and google trends move in the same direction from the center. The daily minimum temperature is sufficiently different from mean and max temperature to keep it in the dataset as an individual variable. Interestingly, wind speed acts as explanatory for a different subset of variance than all other variables.

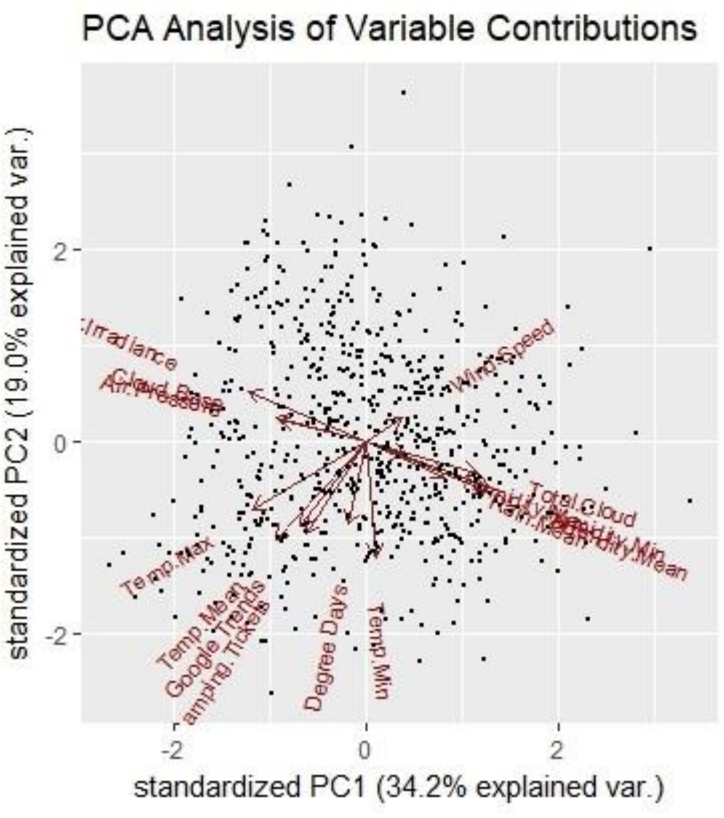


Figure 21. PCA analysis of variable contributions.

5.3 Select Data

The two organizational variables (event and camping tickets) were selected. From the weather variables the following variables were kept solar irradiance, highest daily temperature, average daily temperature, lowest daily humidity, average daily humidity, degree days, average rainfall, wind speed. This was done because they had significant linear relationships.

Highest daily humidity was discarded because it did not have a significant relationship to the dependent variable.

5.4 Data Cleaning

Taking daily averages between specific time intervals led to missing values in Cloud Base (70), Total Cloud (35) and Air Pressure (14). This was because there were no recorded measurements these days or because of irregular measurement hours.

Furthermore, the Götala station had either 25 or 26 missing values in all variables because of certain days when there had been no recordings.

Handling missing values was done by the missForest imputation method as proposed by Stekhoven & Buhlmann (2011) and Waljee et al. (2013). This has been shown to overperform alternative imputation methods like k-Nearest Neighbor and the simpler mean imputation. The random forest methods are particularly strong in complex datasets with non-linear relationships and a mixture of different types of variables (Shah et al. 2014).

5.4.1 Outliers

After the dataset was fused together, the number of observations per week was investigated with a histogram. As can be seen in figure 22, there is an even number of recorded instances for week numbers between 24 and 32. For Skara Sommarland, week 20 and 21 are noteworthy because they have a substantially smaller sample size than other weeks. Because of this, they were removed from the dataset. In the Furuvik dataset, week 20 and all weeks after and including 35 were removed.

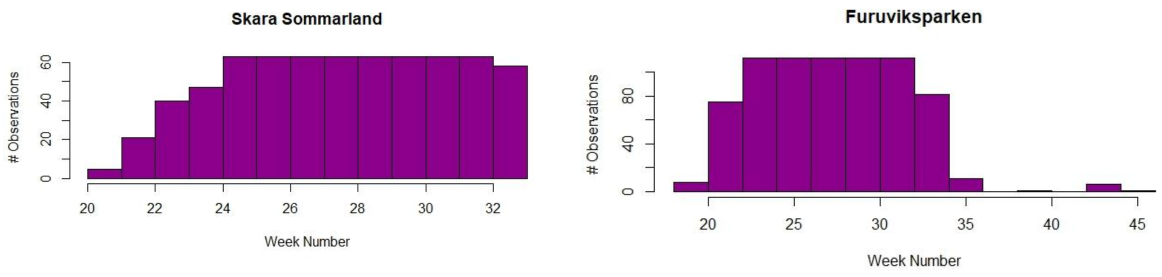


Figure 22. Observations per week number for the two datasets.

In the next step, Mahalanobis Distance (MD) was used to find outliers in both the dependent variable and independent variables (figure 23). In this way, 13 observations were removed because of MD above 40 and three were removed because number of visitors was higher than 12 000 in the Sommarland dataset. The Furuvik dataset had 5 values removed because of MD above 45. The outlier removal techniques were done on observations within the train- and test sets. Not the evaluation set.

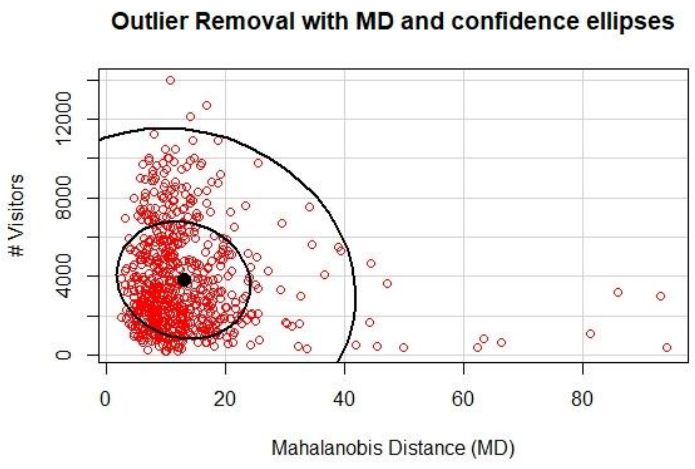


Figure 23. Mahalanobis Distance on x-axis and number of visitors on the y-axis together with 95% and 99% confidence ellipses.

In the next step a boxplot was made with visitors and week number as a category (figure 19). For Sommarland, this plot showed one outlier in each of the weeks 23, 25 and 26. Lastly, week 33 had 7 outliers. In total, the boxplot led to the removal of 6 observations. In the Furuvik dataset, there were 9 outliers removed in this way.

After all outlier removal techniques, the Sommarland dataset had 647 observations and the Furuvik dataset had 462 observations.

5.5 Derived Variables

By investigating correlations between independent variables (figure 24), average temperatures and maximum temperatures have a strong positive correlation (r = 0.91). The same is true for mean humidity and minimum humidity (r = 0.9). This makes them good candidates for using Principal Component Analysis (PCA) to pool their two-dimensional effects into one dimension in order to reduce noise (Lever, Krzywinski & Altman 2017).

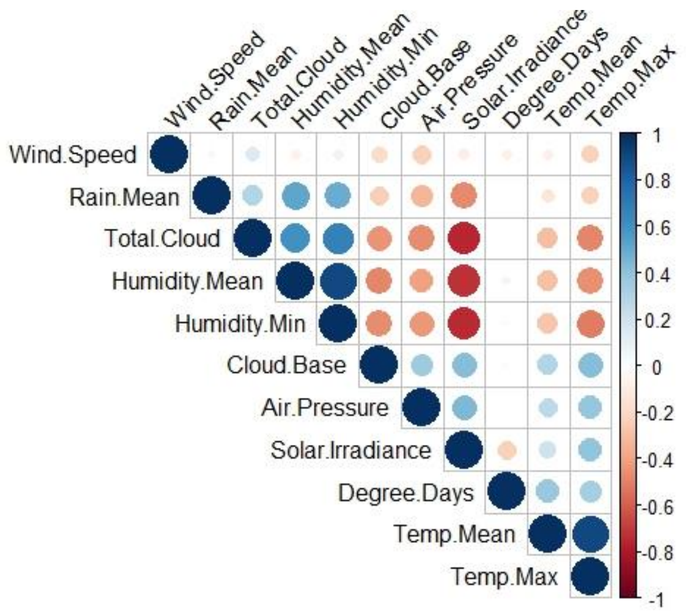


Figure 24. Correlation matrix for independent continuous variables in the Sommarland dataset. Camping tickets and Google searches were omitted.

The strongest negative correlations were found between solar irradiance and total cloud (r = -0.75), minimum humidity (r = -0.75), average humidity (r = - 0.72), and total cloud (r = -0.75). Also, it had a positive correlation with air pressure (r = 0.45).

This stage in the data preparation led to average humidity and minimum humidity being pooled together into one principal component as well as average temperature and maximum temperature being pooled into a second principal component, for both datasets.

Rainfall had a statistically significant linear relationship with number of visitors but a low coefficient of determination (figure 16). A categorical variable was created out of this continuous variable for the Sommarland dataset by constructing three levels based on the quartiles in average rain. Rainfall did not have a large enough variance to be meaningful to be categorized in the Furuvik dataset. The first and second quartile had 0 mm of rainfall. This data was put into the group “No.Rain”. The third quartile had > 0 and < 2.3 mm of rainfall and became the group “Light.Rain”. The fourth quartile had >= 2.3 mm and up to 26.8 mm of rain and became the level “Heavy.Rain” (figure 25).

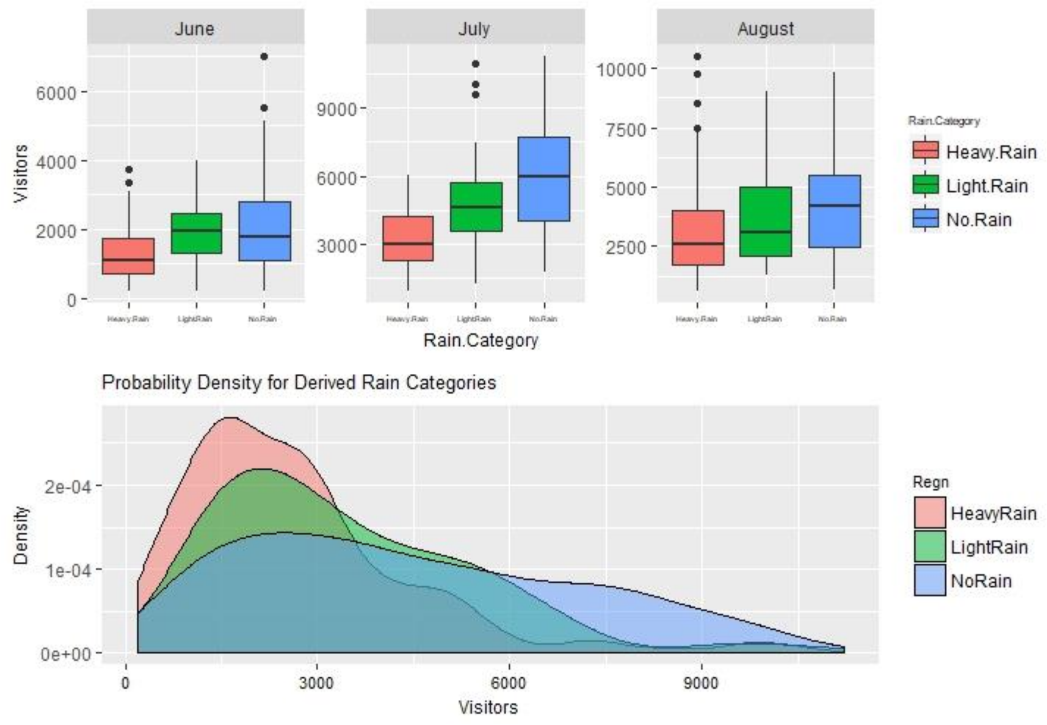


Figure 25. The derived rain categorical variable.

5.6 Model Development

The deep learning library Keras was used to implement neural networks in Python using models “LSTM” and “SimpleRNN”. For the TCN architecture an implementation by philipperemy based on the original locuslab code and Keras (Koltun et al. 2018) was used (https://github.com/philipperemy/keras-tcn).

ARIMA was implemented with the StatsModels library in Python (Perktold, Seabold & Taylor 2017).

Models were developed sequentially by training on data from the years 2010-2016 (2011-2016 for Furuvik) and testing performance on the year 2017 with a certain set of parameters. Sampling techniques like 10-fold cross validation was not used because of the need to maintain temporal integrity i.e. training and testing on a continuous time period. Each model was trained several times this way for every hyperparameter. The neural network models went through hyperparameter search in the following order: epochs (1-150), batch size (1-200), number of hidden layers (0-9), number of timesteps (1-20), dropout (0% - 20%), units (10-100) and input variables. Adam was used as optimization function for all networks (Kingma & Ba 2014).

Furthermore, TCN also went through tuning for kernel size (1-4) and number of residual blocks per layer (1-2) before the final tuning of feature selection.

ARIMA were tuned by the R function “auto.arima” in order to find optimal settings for autoregression, integration and moving average. It went through parameter search only for choice of input variables.

This sequence of parameter search was done once for models using input variables and once without. Predictions was made one day forward in time and to acquire a baseline performance, average number of visitors for each day from the training set and test set was used to generate predictions on the validation set.

6 Results

The final evaluation was done with the year 2018 on both datasets (table 7, 8) and the hyperparameter search results was acquired from the 2017 test sets (table 3, 5).

Table 3. Resulting model configurations from the hyperparameter search in the Skara Sommarland dataset.

Model	Epochs	Batch Size	Hidden Layers	Timesteps	Dropout (%)
TCN	10	130	3	3	0 in the input layer, 10 in the hidden layers.
LSTM	40	130	0	2	5 in the input layer.
RNN	40	200	7	3	5 in all layers except 20 in the last hidden.
HMM	N/A	N/A	N/A	nOrder = 4, NStates = 4	N/A
ARIMA	N/A	N/A	N/A	AR = 2, MA = 1	N/A

Excluded variables in table 4 and table 6 was arrived upon through the search technique (section 5.6) whereby each model was trained once without each variable (and once with all variables), assessed on the test set, and the model with the lowest MAE was selected and put through the same procedure again (unless the model with all variables had the lowest error).

Table 4. Excluded variables from the feature selection at the Skara Sommarland dataset.

Model	Excluded input variables
TCN	-
LSTM	Degree.Days, Rain.Category, Temp.PCA, Humidity.PCA, Google.Trends, Solar.Irradiance
RNN	Humidity.Min, Rain.Mean, Wind.Speed, Air.Pressure, Month, Temp.Min, Weeknum, Weekday, Humidity.Mean, Solar.Irradiance, Camping.Tickets
ARIMA	Google.Trends, Solar.Irradiance, Rain.Mean, Month, Air.Pressure, Temp.Min, Humidity.Mean, Total.Cloud

In addition to hyperparameters presented in table 5, the TCN network went through tuning for kernel size (4, 1, 3) and number of residual stacks (1, 2, 2) in order from the first to the last hidden layer.

Table 5. Resulting model configurations from the hyperparameter search in the Furuvik dataset.

Model	Epochs	Batch Size	Hidden Layers	Timesteps	Dropout (%)
TCN	15	200	3	1	0, 0, 5
LSTM	110	130	6	1	0 in the input layer, 1 in hidden layers, and 10 in the output
RNN	140	130	4	1	5 in the input layer, 1 in the two hidden, and 1 in the output
HMM	N/A	N/A	N/A	nOrder = 1, nStates = 19	N/A
ARIMA	N/A	N/A	N/A	AR = 1, MA = 1	N/A

Table 6. Excluded variables from the feature selection at the Furuvik dataset.

Model	Excluded input variables
TCN	Total.Cloud, Humidity.Min, Cloud.Base
LSTM	Total.Cloud, Cloud.Base, Wind.Speed, Temp.Mean, Temp.Max, Air.Pressure, Temp.Min, Temp.PCA, Rain.Mean
RNN	Weeknum, Weekday, Month, Rain.Mean, Air.Pressure, Temp.PCA, Humidity.PCA
ARIMA	Weekday, Total.Cloud, Humidity.Mean, Rain.Mean, Air.Pressure, Humidity.PCA, Temp.Min, Temp.Mean, Temp.PCA, Wind.Speed

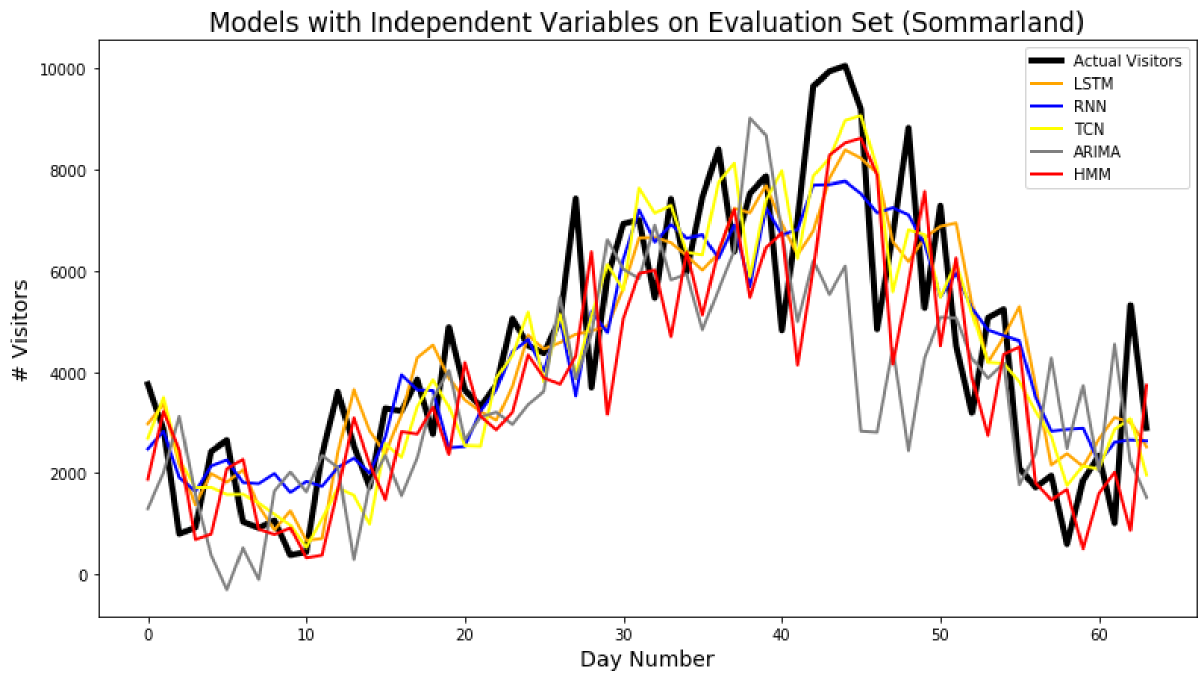


Figure 26. Results on the evaluation set for all five models on the Skara Sommarland dataset.

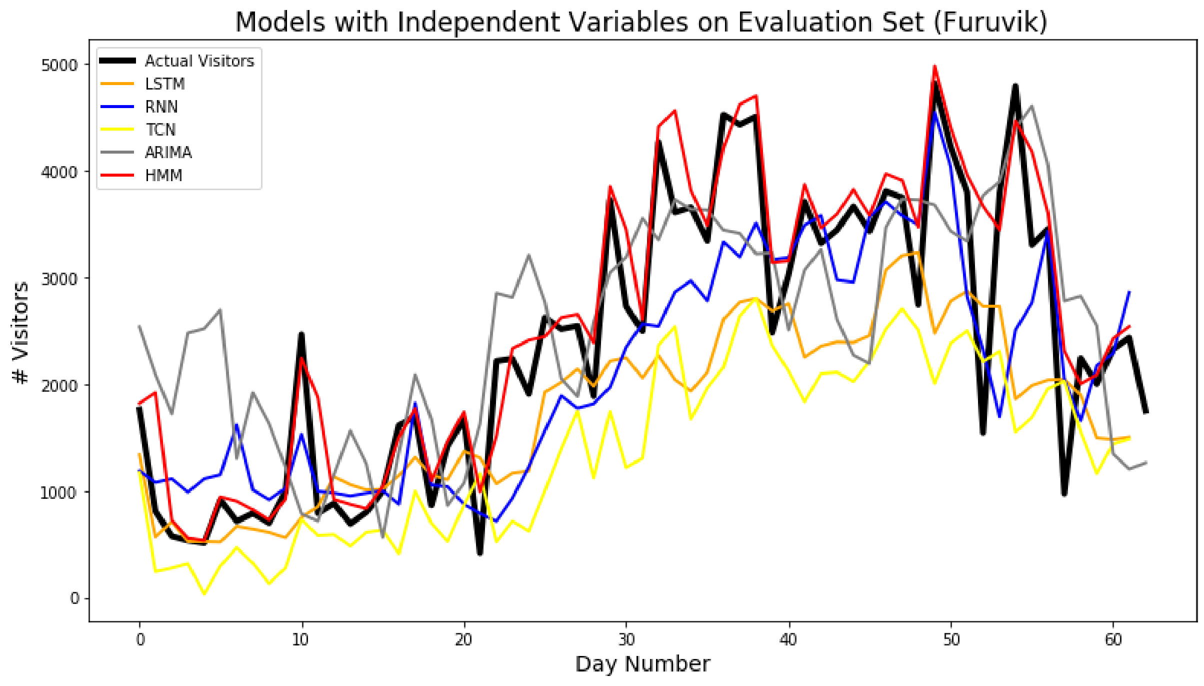


Figure 27. Results on the evaluation set for the five models on the Furuvik dataset.

Using historical average values per date as prediction for future days resulted in a MAE of 1402 and RMSE of 1718 for the Sommarland dataset.

Table 7. Performance metrics for predicting future visitors with the Skara Sommarland dataset.

Dataset	Model	MAE	RMSE	MAPE
With independent variables				
	TCN	1104	1325	38%
	ARIMA	1646	2106	60%
	RNN	1087	1358	46%
	LSTM	1090	1368	41%
	HMM	1451	1765	42%
Without independent variables				
	TCN	1355	1654	46%
	ARIMA	1587	1949	59%
	RNN	1350	1663	47%
	LSTM	1402	1721	60%
	HMM	1451	1765	42%

Table 8. Performance metrics for predicting future visitors with the Furuvik dataset.

Dataset	Model	MAE	RMSE	MAPE
With independent variables				
	TCN	948	1181	38%
	ARIMA	798	949	33%
	RNN	577	722	29%
	LSTM	804	1028	34%
	HMM	670	910	32%
Without independent variables				
	TCN	650	874	43%
	ARIMA	611	777	36%
	RNN	669	876	46%
	LSTM	649	873	44%
	HMM	670	910	46%

Evaluating algorithmic performance on the Sommarland dataset (with independent variables) by using prediction error per day as measurement and the Kruskal-Wallis H-test resulted in p=0.02 with all algorithms and p=0.07 without ARIMA. For the Furuvik dataset this testing resulted in p=0.489 with all algorithms and p=0.566 without ARIMA.

7 Analysis

All algorithms performed worse than the hypothesized MAPE of around 20%, both on the Furuvik and Sommarland datasets. The neural networks had similar values in their error metrics both when evaluated with and without independent variables. Comparing the prediction errors for all algorithms on the Sommarland dataset showed significant difference (p=0.02) because of the relatively high errors of the ARIMA algorithm. The significance disappeared when removing ARIMA from the analysis (p=0.07). The removal of ARIMA had no large effect when comparing prediction errors on the Furuvik dataset (p=0.556) and the difference was also not significant when comparing all algorithms together (p=0.489).

The hidden Markov model was hypothesized to be the best performer when comparing models developed without independent variables. This was not the case; it was the worst performing model on the Furuvik dataset and the second worst on the Sommarland dataset (after ARIMA).

The recurrent neural network showed a difference from other models on the Furuvik dataset with a MAE of 577 and MAPE of 29%. It was also the fastest network to train and develop. This suggests that it can be a good alternative when selecting neural time series models as it was either best or highly like other models while at the same time faster to train and requiring less computer memory.

A benchmark was used in the form of average historical values to make future predictions. This resulted in a MAE of 1402 on the Sommarland dataset. This was worse than all models except ARIMA and HMM. Also, a multiple linear regression (MLR) was used as comparison, resulting in a MAE of 700 for Sommarland and 474 on the Furuvik data. The hyperparameter search resulted in timesteps between 1-3 which could be considered relatively low. That the regression model strongly outperformed the time series model together with the fact that all models achieved optimized settings with few timesteps indicate that the data exhibits more regression than time series characteristics. Furthermore, the autocorrelation in the dataset is low and the variance is high ranging from around 1000 visitors up to 10 000 visitors on the Sommarland dataset and 500-5000 on the Furuvik dataset.

However, it is important to note how relatively little the decrease in prediction metrics were when excluding all input variables. This means that the time series models could, in contrast to a regression model, predict future behavior of a dependent variable without the use of any other input variables than the predicted variable itself. Thus, constituting a major advantage in settings where there is a lack of data that is relevant for predicting future values.

8 Discussion and Conclusion

The three neural networks for time series regression TCN, LSTM and RNN had highly similar performance. The LSTM network is a development from RNN to solve the exploding- and vanishing gradient problems not with the explicit purpose of improving prediction performance. In this study, timesteps no more than three steps back in time were required and thus these gradient problems were not a factor. The RNN is the most lightweight network with the fewest hyperparameters and quickest training time. TCN on the other hand, derived from computer vision networks such as convolutional neural networks, has a substantial architecture relative to both RNN and LSTM and took about ten times longer to train. Furthermore, it has more hyperparameters which results in a more protracted hyperparameter search. The TCN network did not perform better than the other network models on any of the datasets, with or without independent variables. However, it does theoretically have its strength in being able to use its receptive field to incorporate huge amounts of data in the form of previous timesteps together with large number of input variables. This is reflected in the fact that it was the only model that kept all input variables or only removed few of them. It is possible that it could outperform other neural network models in settings with more data and more timesteps. However, in this study the hyperparameter search resulted in both few previous time steps, and relatively few independent variables in most models.

The hidden Markov model could not use independent variables and performed slightly worse than the neural networks on the Sommarland dataset and similarly to the networks on the Furuvik dataset (worse than RNN, better than TCN and LSTM). Because it is not a deep learning model its much faster to train and does not require hyperparameter search, which are advantages.

ARIMA was the worst performer in all evaluations except the test without independent variables on the Furuvik data. In general, it was good at capturing the seasonal trend but reacting more slowly to the sometimes-large day to day changes in visitor numbers.

Comparisons with multiple linear regression display that the datasets investigated are more suited for regression analysis by which each timestep is treated as an independent event. It would be interesting to study the behavior of the different algorithms on data with less variance such as stock data where other studies have reported using 20 timesteps (Fang, Chen & Xue 2019; Li, Shen & Zhu 2018).

8.1 Validity

These findings do have some uncertainties, certainly relating to the hyperparameter search procedure. It is possible that other search strategies would have resulted in a different performance. In this study a sequential grid search approach was employed whereby one parameter was searched within a heuristic interval. An exhaustive grid search or a statistical Bayesian search could have resulted in different network architectures, parameters and input variables that could have generated different metrics (MAE, RMSE, MAPE).

Furthermore, the years 2010-2016 and 2011-2016 were used as training sets, 2017 was the test set and data from 2018 were saved for utilization in the final evaluation on which the prediction metrics and hypothesis test is based. In the spirit of Fischer’s Principle of Replication, it is possible to reuse the exact same division of data for further experiments. In general, units of study within groups should be randomized. In this case, though, the exact same units could be used for all algorithms rendering randomization unnecessary, this property also controls the variance between the units of data assigned to the different algorithms studied.

Lastly, the preparation of the data could be a major source of uncontrolled influence. In this stage, many decisions were made regarding which days to exclude, how to impute missing data (missforest), outlier removal (mahalanobis distance), and whether to derive new variables. It is possible that different data preparation procedures could have yielder better or worse results.

8.2 Societal Aspects

The results of this study show that RNN can be the most suitable algorithm for a time series model on a dataset with a dependent variable with dependence on few previous timesteps. Time series prediction is a major area in science and business and extended knowledge concerning which algorithm to use when can save time and improve results (Zhang, Yin, Zhang & Li 2016; Huang et al. 2016; Juang et al. 2017).

The experiments were conducted on data from the theme park industry and the results regarding the different models' performance are most fine-tuned for this sector. This knowledge will be donated to the related companies – helping them in their analytical capabilities which can lead to enhanced operational and tactical efficiency.

8.3 Scientific Aspects

The algorithmic toolbox for time series has been increasing rapidly throughout the last decades with the emergence of machine learning. Older well-established statistical models like ARIMA have been challenged by neural networks and novel Markov based models. RNNs and LSTMs have been shown to have efficient prediction capabilities in sequential problems (Siami-Namin & Siami-Namin 2018; Xiangang & Wu 2014; Sutskever, Vinyals & Le 2014). Recently, two new algorithms for time series predictions were proposed in the form of TCN, a neural network (Bai, Kolter & Koltun 2018) and HMM, a hidden Markov model (Ky & Tuyen 2018). The present study provides a comprehensive experiment on a real-life problem where the two new innovations from 2018 are compared to ARIMA as well as already established neural networks. The results from this analysis can help clarify the differences and relative strengths and weaknesses of the different algorithms.

8.4 Ethical Aspects

The data used in this study came from two privately owned companies. Care has been taken to not disclose information that could be damaging to their respective businesses while at the same time maintaining maximal transparency for academic purposes. Furthermore, the results of this study have been prepared with diligent analysis and theoretical studies in order to minimize the risk of producing erroneous results that could harm future work by presenting misguided performance metrics thereby risking clouding rather than helping to enlighten the field of time series analysis.

8.5 Future Work

It would be interesting to conduct further experiments with a more controlled hyperparameter search procedure like a full grid search or a Bayesian search. This would decrease the probability that it is the configuration of nodes, layers and parameters that produce unfair prediction metrics.

As the results differ in the best and worst algorithms for the two datasets studied, it would be valuable to perform a study where model configurations are completely controlled together with a large sample of different datasets. From this a meta prediction metric could be calculated by using aggregated values for RMSE and MAPE from the different datasets. A study like this would result in a higher confidence in the conclusion as to which algorithm really has the strongest ability to do forecasting. Coupled with this could be a thorough sub investigation into the abilities on datasets with different characteristics such as high and low variance, frequent cycles, seasonality with and without cycles and dominating stochastic behavior.

8.6 Conclusion

Temporal convolutional networks did not outperform recurrent neural networks or long-short term memory networks with statistical significance on the datasets investigated. Neural network models performed better than ARIMA and the Markov model used. The RNN model generally had the best performance on the Furuvik dataset and was the least complex and quick model to develop among the deep learning models. In contrast, the TCN model was the best performer on the Skara Sommarland dataset with independent variables. The Markov model performed worse than all neural network models even when using no independent variables, a setting where it is supposed to be particularly strong.

Even though the dataset had a clear time dimension, all the time series models had substantially higher prediction errors than a regression model (multiple linear regression). With the best MAPE values being 29% (RNN) and 38% (TCN) time series prediction might not be a satisfactorily reliable way to forecast future visitors in the theme park industry.

This thesis presents a practical analytics scenario and show that when conducting a forecasting study, it might be best to choose one of the simpler deep learning models which take less time to develop and can perform as well or better than more complex algorithms.

References

Adams, R.A., Essex, C. (2018). *Calculus*. Eight ed., Ontario: Pearson Canada Inc.

Akpanta, A.C., Okorie, I.E. (2014). Application of Box-Jenkins Techniques in Modelling and Forecasting Nigeria Crude Oil Prices. *International Journal of Statistics and Applications*, 4(6), pp. 283-291.

Amazon Web Services, Inc (2019). *Amazon Forecast.* https://aws.amazon.com/forecast/ [2019-03-24]

Bahdanau, D., Cho, K.H., Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1409.0473.

Bai, S., Kolter, J.Z., Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1409.0473.

Bernal A.M., Sanso, A. (2001). The Dickey-Fuller Test Family and Changes in the Seasonal Pattern. *Annales d'economie et de statistique*, 61(61), pp. 73-90.

Berndtsson, M., Hansson, J., Olsson, B., Lundell, B. (2008). *Thesis Projects*. Second Ed., London: Springer.

Casson, R.J. (2014). Understanding and checking the assumptions of linear regression: a primer for medical researchers. *Clinical & Experimental Ophthalmology*, 42(6), pp. 590-596.

Chai, T., Draxler, R.R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7, pp. 1247-1250. doi:10.5194/gmd-7-1247-2014

Chan, H.K., Xu, S., Qi, X. (2018). A comparison of time series methods for forecasting container throughput. *International Journal of Logistics Research and Applications*, DOI: 10.1080/13675567.2018.1525342

Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y. (2018). Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Nature* 8(6085).

Dadouche, A. (2018). Time Series Forecasting with SAP HANA APL (Automated Predictive Library). https://developers.sap.com/tutorials/teched-2016-11.html [2019-03-24]

De Gooijer, J.G, Hyndman, R.J. (2006). 25 years of time series forecasting. *International Journal of Forecasting*, 22(3), pp. 443-473. doi: https://doi.org/10.1016/j.ijforecast.2006.01.001

Duchon, C., Hale, R. (2012). Time Series Analysis in Meteorology and Climatology: An Introduction. First Ed., New Jersey: John Wiley & Sons.

Durka, P., Pastorekova, S. ARIMA vs ARIMAX – which approach is better to analyze and forecast macroeconomic time series? *Proceedings of 30th International Conference Mathematical Methods in Economics*. Karviná, Czech Republic.

Fang, Y., Chen, J., Xue, Z. (2019). Research on Quantitative Investment Strategies Based on Deep Learning. *Algorithms*, 12(2), pp. 1-34.

Flores, J.J., Calderon, F., Gonzales, J.R.C., Ortiz, J. (2016). Comparison of Time Series Forecasting with respect to Tolerance. *IEEE International Autumn Meeting on Power, Electronics and Computing.* Ixtapa, Mexico. doi: 10.1109/ROPEC.2016.7830618

Gamboa, J. (2017). Deep Learning for Time-Series Analysis. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1701.01887.

Glorot, X., Bordes, A., Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. Fort Lauderdale, USA, 15, pp. 315-323.

Hochreiter, S., Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), ss. 1735-1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735

Huang, X., Zeng, J., Zhou, L., Hu, C., Yin, P., Lin, X. (2016). A New Strategy for Analyzing Time-Series Data Using Dynamic Networks: Identifying Prospective Biomarkers of Hepatocellular Carcinoma. Scientific Reports, 6(32448). doi: 10.1038/srep32448

Jebb, A.T., Tay, L. (2017). Introduction to Time Series Analysis for Organizational Research: Methods for Longitudinal Analyses. *Organizational Research Methods*, 20(1), pp. 61-94.

Jebb, A.T., Tay, L., Wang, W., Huang, Q. (2015). Time series analysis for psychological research: examining and forecasting change. *Frontiers in Psychology*, 6(727). doi: 10.3389/fpsyg.2015.00727

Jofipasi, C.A., Miftahuddin., Hizir. (2017). Selection for the best ETS (error, trend, seasonal) model to forecast weather in the Aceh Besar District. *Materials Science and Engineering*, 352. doi: 10.1088/1757-899X/352/1/012055

Juang, W-C., Huang, S-J., Huang, F-D., Cheng, P-W., Wann, S-R. (2017). Application of time series analysis in modelling and forecasting emergency department visits in a medical centre in Southern Taiwan. *BMJ Open*, 7(11).

Kingma, D.P., Ba, J. (2014). Adam: A Method for Stochastic Optimization. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1412.6980.

Kumar, U., Jain, V.K. (2010). Time series models (Grey-Markov, Grey Model with rolling mechanism and singular spectrum analysis) to forecast energy consumption in India. *Energy*, 35(4), pp. 1709-1716. https://doi.org/10.1016/j.energy.2009.12.021

Ky, D.X., Tuyen, L.T. (2018). A Higher order Markov model for time series forecasting. International Journal of Applied Mathematics and Statistics, 57(3).

LeCun, Y., Bengio Y., Hinton, G. (2015). Deep learning. *Nature*, 521, pp. 436-444.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), pp. 2278-2324.

LeCun, Y., Cortes, C (nd). *THE MNIST DATABASE of handwritten digits.* http://yann.lecun.com/exdb/mnist/ [2019-02-06]

Lever, J., Krzywinski, M., Altman, N. (2017). Principal component analysis. *Nature Methods*, 14, pp. 641-642.

Li, H., Shen, Y., Zhu, Y. (2018). Stock Price Prediction Using Attention-based Multi-Input LSTM. *Proceedings of Machine Learning Research*, 95, pp. 454-469.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C-Y., Berg, A.C. (2016). SSD: Single Shot Multibox Detector. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1512.02325v5.

Ljung G.M., Box, G.E.P. (1978). On a measure o lack of fit in time series models. *Biometrika*, 65(2), pp. 297-303.

Makridakis, S., Spiliotis, E., Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: concerns and ways forward. *PLoS ONE*, 13(3). doi: https://doi.org/10.1371/journal.pone.0194889

Merity, S. (2016). The wikitext long term dependency language modeling dataset. Salesforce. https://www.salesforce.com/products/einstein/ai-research/the-wikitext-dependency-language-modeling-dataset/ [2018-02-06]

Montgomery, D.C., Jennings, C.L., Kulahci, M. (2015). Introduction to time series analysis and forecasting. Second Ed., New York: John Wiley & Sons, Inc.

NCSS (2019). *Time Series and Forecasting Methods in NCSS.* https://www.ncss.com/software/ncss/time-series-and-forecasting-in-ncss/ [2019-03-24]

Olah, C. (2015). Understanding LSTM Networks. *Colah's blog* [blog], 27 August. http://colah.github.io/posts/2015-08-Understanding-LSTMs/ [2019-02-20].

Pascanu, R., Mikolov, T., Bengio, Y. (2013). On the difficulty of training Recurrent Neural Networks. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1211.5063.

Perktold, J., Seabold, S., Taylor, J. (2017). *statsmodels.tsa.arima_model.ARIMA*. https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima_model.ARIMA.html [2019-03-20]

Rai, P. (2016). Google PageRank Algorithm: Markov Chain Model and Hidden Markov Model. *International Journal of Computer Applications*, 138(9). doi: 10.5120/ijca2016908942

Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1506.02640.

Ruder, S. (2017). An overview of gradient descent optimization algorithms. *Cornell University Library.* arXiv:1609.04747

Russell, S.J., Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Third ed., New Jersey: Pearson Education, Inc.

Sak, H., Senior, A., Beaufays, F. (2014). Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition. *Cornell University Library*. arXiv:1402.1128

Salimans, T., Kingma, D.P. (2016). Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. *Cornell University Library*. arXiv:1602.07868v3SAS (nd). *Getting Started with Time Series Forecasting.* http://support.sas.com/documentation/cdl/en/etsug/60372/HTML/default/viewer.htm#tfstart_toc.htm [2019-03-24]

Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *Neural Networks*, 61(2015), pp. 85-117.

Shah, A.D., Bartlett, J.W., Carpenter, J., Nichlas, O., Hemingway, H. (2014). Comparison of Random Forest and Parametric Imputation Models for Imputing Missing Data Using MICE: A CALIBER Study. *American Journal of Epidemiology*, 179(6), pp. 764-774.

Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, 5(4), pp. 13-22.

Siami-Namini, S., Namin, A.S. (2018). Forecasting Economics and Financial Time Series: ARIMA vs. LSTM. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1803.06386.

Srivatava, N., Hinton, G., Krizhevsky, A., Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15, pp. 1929-1958.

Stekhoven, D.J., Buhlmann, P. (2011). MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 20(1), pp. 112-118.

Sutskever, I., Vinyals, O., Le, Q.V. (2014). Sequence to Sequence Learning with Neural Networks. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1409.3215.

Tableau (2019). *How Forecasting Works in Tableau.* https://onlinehelp.tableau.com/current/pro/desktop/en-us/forecast_how_it_works.htm [2019-03-24]

Tarwani, K.M., Edem, S. (2017). Survey on Recurrent Neural Network in Natural Language Processing. *International Journal of Engineering Trends and Technology*, 48(6), pp. 301-304. doi: 10.14445/22315381/IJETT-V48P253

Toutenburg, H., Shalabh., Fienberg, S., Olkin, I. (2009). *Statistical Analysis of Designed Experiments*. Third ed., New York: Springer.

Waljee, A., Mukherjee, A., Singal, A., Zhang, Y., Warren, J., Balis, U., Marrero, J., Zhu, J., Higgins, P.D.R. (2013). Comparison of imputation methods for missing laboratory data in medicine. *Gastroenterology and hepatology Research*, 3(8), pp. 1-7.

Wang, T. (2016). Forecast of Economic Growth by Time Series and Scenario Planning Method – A Case Study of Shenzhen. *Modern Economy*, 7, 212-222. DOI: 10.4236/me.2016.72023

Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q. ,Macherey, K., Klinger, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J. (2016). Google's Neural Machine Translation System:

Bridging the Gap between Human and Machine Translation. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1609.08144v2.

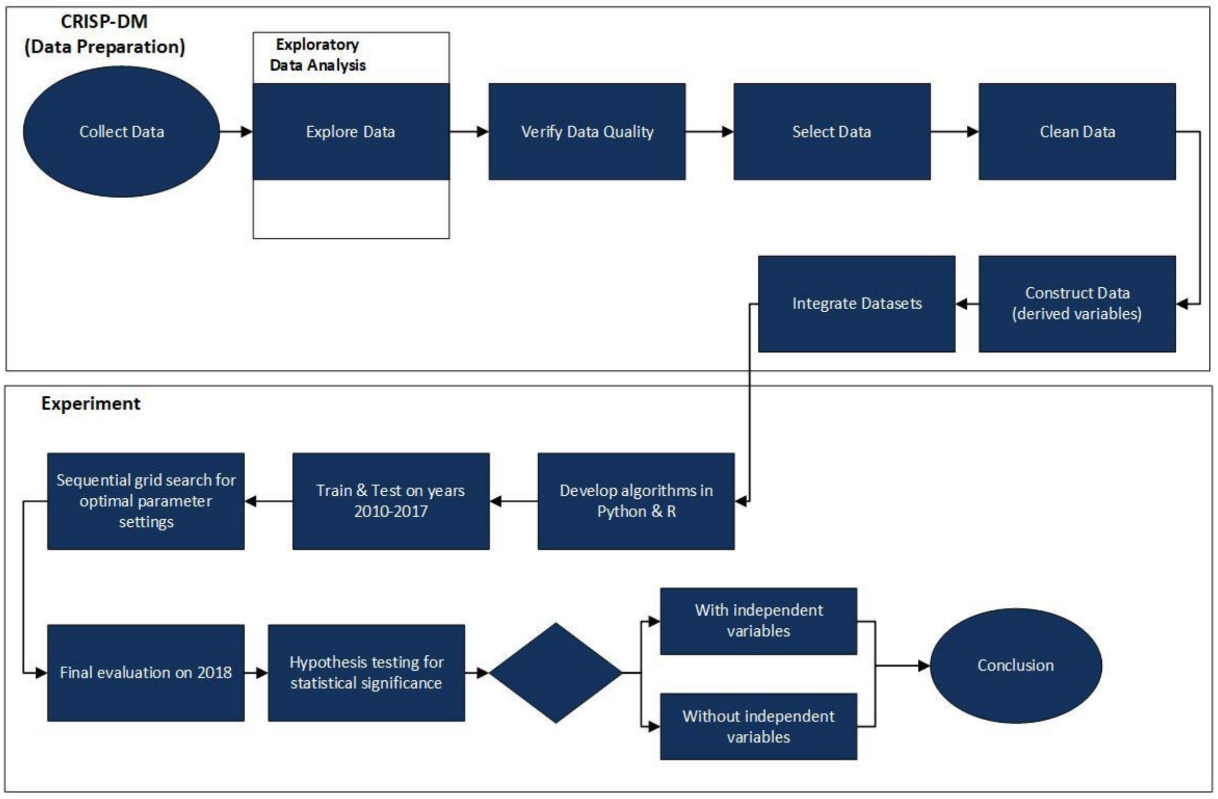
Yaffee, R., McGee, M. (2000). *An Introduction to Time Series and Forecasting*. First ed., San Diego: Academic Press.

Xiangang, L., Wu, X. (2014). Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition. *Cornell University*. arXiv:1410.4281

Yu, F., Koltun, V. (2015). Multi-Scale Context Aggregation by Dilated Convolutions. *Cornell University Library, arXiv.org,* issn: 2331-8422, arXiv:1511.07122.

Zhang, T., Yin, F., Zhou, T., Zhang, X-Y., Li, X-S. (2016). Multivariate time series analysis on the dynamic relationship between Class B notifiable diseases and gross domestic product (GDP) in China. *Scientific Reports*, 6(29). doi: https://doi.org/10.1038/s41598-016-0020-5

9 Appendix



CRISP-DM
(Data Preparation)
Collect Data
Exploratory Data Analysis
Explore Data
Verify Data Quality
Select Data
Clean Data
Construct Data (derived variables)
Integrate Datasets
Experiment
Develop algorithms in Python & R
Train & Test on years 2010-2017
Sequential grid search for optimal parameter settings
Final evaluation on 2018
Hypothesis testing for statistical significance
With independent variables
Without independent variables
Conclusion

Appendix 1. Overview of the use of research methods as a process model.