# Time series Forecasting on crime data in Amsterdam for a Software Company

Prakash Singh

Internship report presented as partial requirement for obtaining the master's degree in Advanced Analytics

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

# TIME SERIES FORECASTING ON CRIME DATA IN AMSTERDAM FOR A SOFTWARE COMPANY

by

Prakash Singh

Internship report presented as partial requirement for obtaining the master's degree in Advanced Analytics

**Supervisor:** Jorge Mendes
**External Supervisor:** Vincent Hoekstra

November 2018

## DEDICATION

Dedicated to my parents for their unwavered love and support throughout my life.

# ACKNOWLEDGEMENTS

Firstly, I would like to thank the Sentient Information Systems for giving me the opportunity to work on this exciting project. I couldn't have asked for a better company where I was given clean slate to explore different paths. My special thanks to Tim Den Uyl, CEO of Sentient, for his valuable time and feedback at each stage.

I feel indebted to my external supervisor, Vincent Hoekstra, for advising and mentoring me during the internship. His expertise and experience were very important in shaping this internship towards success. Also, Thank you Pieter Bison for your presence and insightful feedback in every status review meeting.

I am also thankful to my friend Amogh Gudi for always being there in case of any technical or conceptual difficulty in my internship. It was a pleasure to work with you and thank you for your knowledge, wit and code!

Finally, I am grateful to my Nova IMS advisor, Jorge Mendes, for his time and effort in reviewing my reports and ensuring I meet the requirements for this internship report.

## ABSTRACT

In recent years, there have been many discussions of data mining technology implementation in the fight against terrorism and crime. Sentient as a software company has been supporting the police for years by applying data mining techniques in the *DataDetective* application (Sentient, 2017). Experimenting with various types of predictive model solutions, selecting the most efficient and promising solution are the objectives of this internship.

Initially, extended literatures were reviewed in the field of data mining, crime analysis and crime data mining. Sentient provided 7 years of crime data which was aggregated on daily basis to create a univariate dataset. Also, an incidence type daily aggregation was done to create a multivariate dataset. The prediction length for each solution was 7 days. The experiments were divided into two major categories: Statistical models and neural network models. Neural networks outperformed statistical models for the crime data.

This paper provides the overview of statistical models and neural network models used. A comparative study of all the models on similar dataset gives a clear picture of their performance on available data and generalization capability. Evidently, the experiments showed that Gated Recurrent units (GRU) produced better prediction in comparison to other models. In conclusion, gated recurrent unit implementation could give benefit to police in predicting crime. Hence, time series analysis using GRU could be a prospective additional feature in *DataDetective.*

## KEYWORDS

Time series analysis; Forecasting; ARIMA; Supervised learning; machine learning; predictive models; Neural networks; Recurrent Neural Networks; Convolutional neural networks

# CONTENTS

# LIST OF FIGURES

# 1. Introduction

In recent years, Information Communication and Technology (ICT) have been widely used in facilitating human tasks, to increase efficiency and reduce errors. Data mining is an ICT service that explores large amounts of data, quickly and efficiently to find patterns, relationships or trends. The identified pattern is used to build a model. This model can be used to predict the future values.

Data mining has been widely used in the criminal domain (Chen et al., 2004) for instance to detect criminal identity deception; authorship in cybercrime, and criminal network. The development of crime data mining tools has increased rapidly in recent years. IALEIA[1], COPLINK[2], ATAC[3], I2 Analyst's Notebook[4] and *DataDetective*[5] are some examples of data mining tools.

Temporal analysis has increasingly attracted research and development. In 2008, I2 launched I2 Pattern Tracer that finds patterns in telephone data calls. It tries to detect the hierarchy of a criminal organization, identify potential targets and identify relationships between individuals, by analysing phone call patterns. Criminal career analysis is another example of sequence analysis that also attracts researchers' attention. The goal is to detect career similarities between criminals.

*DataDetective* is Sentient's own data mining platform. Sentient is a software company that was founded in Amsterdam more than 20 years ago, which offers analytical tools and an associative memory engine. Its product has successfully been implemented by the police forces *Midden en West-Brabant, Brabant Noord and Amsterdam-Amstelland*. *DataDetective* offers a complete data mining solution for its customers, including fuzzy matching, profiling analysis, frequency analyses, cross table analysis, decision trees, segmentation, geographic visualization and modelling. However, *DataDetective* has not provided any advanced sequence and time series analysis feature yet.

## 1.1 Problem and objective

As discussed above, *DataDetective* does not have any advanced sequence and time series analysis yet. This acts as a significant constraint to the clients if they want to see the number of incidences that might occur in next week. Until now, the application uses different data mining technique to analyse the patterns in data, but time was not used as a feature upon which the analysis could be performed.

This internship acts as the first attempt at introducing temporal behaviour to *DataDetective,* which means that this internship will act as the common ground on which furthermore advanced spatial-temporal work will be carried out. It is conducted within Sentient ICT Company to analyse the crime incidence data and develop multiple statistical and machine learning models to predict the number of incidences that would happen in next week (7 days). The internship result was expected to confirm

1. International Association of Law Enforcement Intelligent analyst, Journal 2017. https://www.ialeia.org/intelligence_library.php
2. Forensic logic's COPLINK application. https://forensiclogic.com/
3. Automated Tactical Analysis of Crime, Bair Analytics Inc. https://risk.lexisnexis.com/law-enforcement-and-public-safety/crime-analytics-and-mapping
4. IBM i2 Analyst Notebook. https://www.ibm.com/us-en/marketplace/analysts-notebook/details
5. Sentient Information Systems. https://www.sentient.nl/datadetective-1

Sentient which model and type of model performs the best for the given data and whether adding advance temporal analysis tools into *DataDetective* would contribute a significant benefit into its clients or not. predicting crime time series was selected as the most promising challenge to be solved. Additionally, an objective of the internship is to diversify the search for most optimal model like, univariate and multivariate models, statistical models and neural networks, feedforward neural network and recurrent neural networks.

# 2. Related Work

## 2.1 Crime analysis

A brief research is conducted to understand the crime-analysis through the literature study (Elkblom, 1998). An analyst of *Midden and West* Brabant Police confirmed that criminal profiling (Oatley, Zeleznikow & Ewart, 2005) was one of the main issues in crime analysis. On the strategic level, the police try to identify persons who tend to commit more crimes, identify persons who tend to be victims and detect where crime will possibly take place. Police on the tactical level use the information to organize the police's patrol route, which will be implemented by the operational police. Profile analysis, geographical analysis and frequency analysis features in the *DataDetective* application (Sentient, 2008) support these crime analysis approaches.

## 2.2 Crime data mining

Data mining has been assisting law-enforcements and intelligence-gathering organizations to analyse crime intelligently. Entity extraction, association, prediction and pattern visualization are parts of the crime analysis approaches, according to (Chen et al., 2004). Some of the implementations are deceptive-identity detection and criminal-network analysis. Many data mining techniques such as clustering, association, sequential pattern mining, classification and string comparator; have been implemented on data mining application such as *Coplink* (Chen et al., 2004) and *DataDetective* (Sentient, 2008). However, this research is focused on temporal analysis algorithms: sequence mining, temporal mining and time series analysis. Still in temporal analysis, change mining (Bottcher, Hoppner & Spiliopoulou, 2008), which analyses how models have changed and try to predict how it will change, is also an interesting topic but not necessarily applied in this research.

## 2.3 Time series mining

Time series analysis is mostly applied in economy analysis but also found in bio surveillance, crime mapping (Gorr & McKay, 2004) and geographysical sensor network (Granat, 2009). Time series clustering (Keogh, Lin & Truppel, 2003), time series anomaly and sequential detection (Nguyen, 2006) are examples of time series analysis implementations. However, we regarded the time series forecasting (Hyndman, 2006) as the most interesting implementation in the criminal domain. There were several algorithms that have been researched to extrapolate the time series, namely structural time series (Sridharan, Vujic & Koopman, p. 18, 2003), neural network (Rutka, 2008), C-L approach

(Greenberg, 2001) and ARIMA models (Harvey, 1993; Kapinos, 2006). However, ARIMA is commonly used in modelling time series, since numbers of time series analysis literatures in the Free University library applied the ARIMA method.

## 2.4 Crime data mining implementations

Sequence mining can be applied to find frequent criminal career patterns, and sequence alignment method can be used to check the similarity between patterns. An analyst who already did research on criminal career analysis with the Amsterdam Police confirmed the obstacles of this idea. This analysis will give more benefit to the Justice Department than for the police analysts, Sentient's active customer. Criminal career analysis predicts the prospective criminals and Justice Department can arrange rehabilitation program for the potential offenders. In contrast, police can only act when a crime has been committed. In addition, there is lots of criminals' career missing value, for examples: the police do not have information when criminals stop doing crime or stay in jail.

The literatures of the spatial-temporal mining is extensive. Many researchers interest with this area including a criminologist in Amsterdam. Geographical Information System (GIS) has successfully assisted police in profiling location with high crime rates. Many clustering methods are used to map the high-crime rate location. The spatial-temporal mining's goal is not only to be more precise in predicting where the crime will take place, but also when it will possibly happen. These are some implementations of the spatial-temporal mining: to identify the mobility pattern of offenders, to predict when and how the crime will diffuse (Kim, 2008) and to predict the next crime location base on the offender social network activities (Furtado et al., 2007).

## 2.5 Crime data mining applications

To design and develop a state-of-the-art temporal application, we analysed other applications and their features, which provide data mining services to the law enforcement. These are some examples beside the examples on the introduction section: Law Enforcement Information System (Oatley, Ewart & Zelenikow, 2006) and Crimelibs (Kovachev, Reichert & Speck, 2008).

# 3. Time Series Analysis

## 3.1 Introduction to time series

The term “Time series”, signifies a data format, which consists of two main components – a time unit and value/values associated with that time unit. Unlike normal datasets, time isn’t just a metric, but a primary axis.

There are two fundamental ways we can store a time series. The first way of recording a time series is that the time intervals are treated as discrete points. These points can also represent values which are measured for just specific timestamps, which might occur in a periodic or an occasional manner. These time series are called discrete time series. A good example of discrete time series is the financial/economical time series where various attributes or indicators are recorded periodically.

Another way of recording a time series is to store values continuously along a time axis. These series are called continuous time series. There are multiple examples of continuous time series like, sensor data coming out of various Internet Of Things (IOT) devices which record the data on a continuous time interval.

## 3.2 Time Series Classification types

One way to classify the types of time series is the dependency between a new recorded value and its past/historic values. There are mainly two types of values under this classification:

- Long-term memory time series: These series exhibit slow decrease in autocorrelation function. In other words, these types of series show that the current values in series have high correlations with relatively large set of lags in series. An example is the temperature change in the atmosphere where current day temperature can be exhibited by large chain of past date temperatures.
- Short-Term memory time series: These series have fast decrease in autocorrelation function and hence the correlation between current value decrease dramatically fast when compared with different lags in series. Typical examples contain processes from econometric sectors.

Another way of classification is based on stationarity. Stationarity of time series is the feature which exhibits a change in statistical properties like mean, variance and time covariance. There are mainly two types of time series:

- Stationary Time series: stationary time series demonstrate constant stationarity over time. So, the mean, variance and time covariance are relatively constant over time.
- Non-stationary Time series: As the name suggests, the time series do not demonstrate the constant equilibrium of mean, variance and time covariance.

In practical world, time series are mostly categorised into non-stationary time series and hence when using stationary models, certain kinds of pre-processing needs to be done on these time series before we go into forecasting phase.

## 3.3 Vision of time series analysis

Time series analysis can be divided into two parts.

1. Analysing the historical time series to get insights and the patterns in those datasets. This can also be considered interpolating.
2. Forecasting the future values of time series based on current and past values. This can be considered as extrapolation.

This can be done by creation and identification of the optimized time series model. A good model can easily deliver good insights into the data like the trend and seasonality in the data. It can also predict future values by using information from previous steps.

## 3.4 Time Series Components

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

The most basic time series component in the long-term growth or decline. This component is called long term trend or just trend. A trend can be linear or nonlinear. [David Gerbing, 2016]

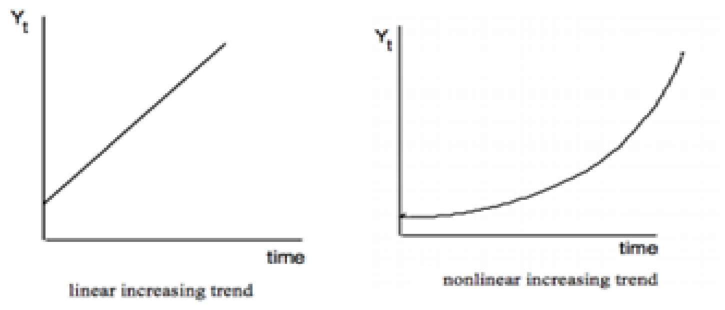


*Figure 1 Trend Component of Time series*

Second component in time series is called cyclical component. This component is analysed long time intervals like years or even decades. It shows cyclical or long periodical swings on a typical trend line. Since this component is exhibited in long time intervals, it is usually not present in the practical time series.

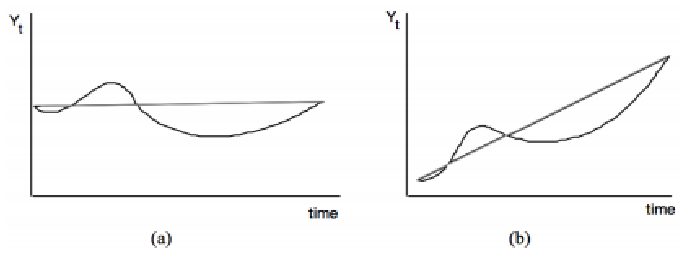


*Figure 2 Cyclical Component of Time series*

Seasonal component is the next component in time series. This component shows short and repetitive fluctuations across a trend line. Usually, seasonal components of a sales time series data are divided into four quarters or seasons.

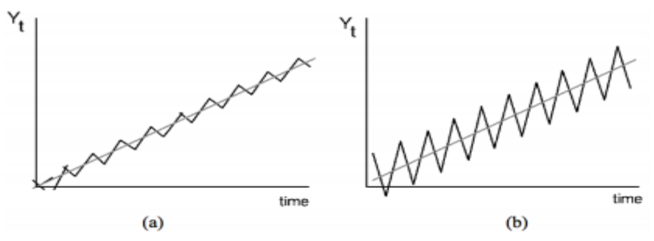


*Figure 3 Seasonal Component of Time series*

The trend, cyclical, and seasonal components combine to form the pattern underlying the time series data, the values of Y ordered over time, though not all components characterize every data set. An important purpose of time series analysis is to isolate each of these three components and demonstrate how each affects the value of Y over time, including forecasts for the future. The identification of this pattern, the predictable aspect of a time series, however, is complicated by the presence of random error.
Since, every time series contains random error, combinations of a trend, cyclical, and seasonal components are observed together with the error component.

## 3.5 Types of time series model

Type of time series model can be determined based on how the components of time series interact with each other. There are mainly two types of time series models:

- Additive models: In these models, cyclical, seasonal, and error components are added to the trend components. In other words, each component is discrete in in its effects and changes in one do not affect another component. This model is used where changes in values are measured in absolute quantity.
  **Time series = Trend + Seasonal Component + Cyclical Component + Error Component**

- Multiplicative models: As the name suggests, all the components are multiplied in these models and hence effects of one component impact another model. This model is used where change is measured in percentage changes. Practically, data is usually
  **Time series = Trend * Seasonal Component * Cyclical Component * Error Component**

## 3.6 Autocorrelation and Partial Autocorrelation

Correlation signifies the strength in relationship between two variables. We can calculate the correlation of current observation of time series with the past observations of time series. Since instead of using two different variables, we are using different lags of same series, we call this correlation as serial correlation or autocorrelation.

Now let's consider a scenario where a current observation is only correlated to its immediately prior observation. In this case, just using an autocorrelation will give a false insight of having strong correlation with greater lags which is attributed to the chaining effect of a time series. Here, partial autocorrelation is used to nullify this chaining effect of correlation. Partial autocorrelation can be thought as a correlation between two points that are separated by some number of periods, but with the effects of intervening correlation removed.

## 3.7 Time series forecasting

Time series forecasting refers to the process of predicting values in future time steps of time series based on previous time steps information. This process of extrapolation is carried out by a forecasting model which in turn is a set of algorithmic actions called forecasting method.

A time series could be just a univariate sequence of values in a variable indexed by time or it can be a multivariate sequence of values where many different external factors could be used to better explain the effects on the time series. For example, external factors of crime could be various indicators like locality, weather and population.

## 3.8 Forecast Accuracy

Forecasting accuracy as described in (Hyndmen et al, 2006) is a measure that determines performance of a forecasting model. This feature determines following points:

- Forecasting model's bias
- Forecasting error
- Best performing forecasting model.

There are many kinds of functional accuracy measures. Each measure carries different kind of information to identify the accuracy. We tried to use these measures in terms of forecast error which is difference between the forecast and the actual value in series. Let's classify these measures in two categories:

- Scale-Dependent errors: Here the forecasting errors are on same scales of the values in time series. There are mainly two types of scale dependent errors:

    1. Mean Absolute errors

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

2. Root Mean Squared error

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

Root mean squared errors penalize strong errors less than mean absolute errors.

- Percentage error: It provides information about the percentage of forecasting error against the actual observation. One of the major advantages is that since it is unit free, it can be used to measure forecasting error across different datasets as well.

  Mean Absolute Percentage error (MAPE): It is one of most used forecasting accuracy measure in time series analysis. A major drawback of MAPE is that it can only be used when the values in series are significantly greater than 0.

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

Another drawback is that it penalizes the negative forecasting errors more heavily than the positive forecasting errors. This observation led to the development and use of symmetric Mean absolute percentage error, SMAPE.

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

However, its calculation involves division by zero, making the calculation unstable. Since the value can also be negative, SMAPE is not actually “Absolute”.
Hyndman & Koehler (2006) recommended against using SMAPE.

## 3.9 Data Pre-Processing

Data pre-processing is an integral step in the time series analysis and forecasting process. Using raw data into the forecasting model will lead to several undesirable problem including very low forecasting accuracy. We are going to discuss some of the common data pre-processing methods used in time series forecasting:

### 3.9.1 Smoothing

Every time series will have random error component. Smoothing is a process of local averaging, which helps in removal of unwanted noise signal.  Smoothing helps in better understanding of the pattern and trends by removing irregular roughness in the time series. Few well known smoothing methods are moving average filter and linear filter.
Moving average filter replaces an observation as the weighted average of values in previous timestamps. Similarly, Linear filter replaces current observation with the smoothed value which is calculated as the linear combination of values at surrounding times.

### 3.9.2 Differencing

As discussed in the above topics, a time series in real lie scenario is not stationary i.e. its stationary features like mean, variance and covariance of time series are not constant but are functions of time. This non-stationary time series is not ideal since most of forecasting models work on stationary series. The most common way of removing non-stationarity is differencing the time series. Differencing is an iterative process and it is done until the time series is stationary.

$$Z(t) = Z(t) - Z(t-1)$$

### 3.9.3 Scaling

This method is very commonly used in the machine learning and it transforms the data by adjusting the scale within some boundaries. This helps the forecasting model to better predict the next steps and increase the accuracy. Below, we can see a typical <0,1> boundary scaling of dataset.

$$\hat{Z}(t) = \frac{Z(t) - minimum}{maximum - minimum}$$

### 3.9.4 Normalization

Normalization is most used pre-processing step used currently. The aim of this step is to transform the dataset so that mean=0, variance=1. Post normalization the mean of the data is centred at 0 and variance of time series dataset is centred at 1.

### 3.9.5 Outlier Detection and Removal

Usually, outliers are the data points in time series which significantly different than other data points. An outlier can be a mistakenly recorded data, or it can be a correct data but their occurrence in the dataset will affect the model accuracy in a negative way. Therefore, outliers are eliminated. The process of outlier detection is also called filtering. Outlier detection can be classified into six categories:

1. Classification based
2. Nearest neighbour based
3. Clustering based
4. Information theory based
5. Statistical based
6. Spectral theory based

One of the important thresholds to determine a data point as outlier is prediction confidence interval (PCI). It is calculated by the predicted value and the confidence threshold.

# 4.Forecasting methods

In this chapter, we are going to discuss various forecasting methods to analyse a time series and then extrapolate the series by forecasting values in future time indexes. Each forecasting method has its special use and care must be taken to select correct method for an application.

The selection of a method depends on many factors like context of forecast, the availability of historical data, degree of accuracy desired in application, number of steps ahead to be predicted and time available for the analysis.

The most basic time series forecasting method is called the naive forecast and sets the forecast to the last time series observation. Another simple method is the moving average, where the forecast is the arithmetic mean of the most recent values of the time series, discarding old and potentially inapplicable observations. Beyond these, the first more sophisticated methods date back to the 1950’s and 1960’s, with new approaches and extensions constantly being investigated until today. The choice of publications to cite has been difficult, as a vast majority of contributions are smaller case studies applied to a specific application area, which mostly provide results that contradict each other.

## 4.1 Autoregressive and moving average models

Autoregressive (AR) models are models in which the value of a variable in one period is related to its values in previous periods. In this model, the actual value of the process is expressed as the sum of finite linear combinations of previous values and the impulses, called white noise.

White noise put forth an assumption that each element in time series is a random draw from a population with zero mean and constant variance.

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + \ldots + \phi_p \tilde{z}_{t-p} + a_t$$

AR (p) is an autoregressive process of order $p$ where $\tilde{z}_t$ is the deviation from $\mu$, $\tilde{z}_t = z_t - \mu$. The autoregressive model may also be written as

$$\phi(B)\tilde{z}_t = a_t$$

With backshift operator as

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$

The order $p$ and parameters $\mu, \phi_1, \ldots \phi_p$ and $\sigma_a^2$, which is variance of the white noise process $a_t$, must be estimated from the data.

*Moving average* models are another kind of models, which represent the observed time series by making $\tilde{z}_t$ linearly dependent on a finite number $q$ of the previous $a$.

$$\tilde{z}_t = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ldots - \theta_q a_{t-q}$$

The moving average may also be written as

$$\tilde{z}_t = \theta(B) a_t$$

With moving average operator as

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q$$

The order $q$ and parameters $q, \mu, \theta_1, \ldots \theta_q$ and $\sigma_a^2$, which is the variance of the white noise process, must be estimated from the data.

## 4.2 ARIMA Models

To achieve better prediction quality, two previous models are often merged into one model, autoregressive and moving average model. Common model is denoted as ARMA(p,q) and it unites a moving average filter of order q and autoregression of filtered values of order p. If the time series shows signs of non-stationarity, then the above-mentioned differencing step can be applied to remove non-stationarity. A post differencing ARMA model is called an ARIMA(p,d,q) model. The parameter d represents the degree of differencing, which is a fundamental part of this model.

One of the most influential publications in time series forecasting is Box & Jenkins (1970), having an extraordinary impact on forecasting theory and practice until today. The authors introduced the group of autoregressive integrated moving average (ARIMA) models, which can simulate the behaviour of diverse types of time series. An ARIMA model consists of an autoregressive and a moving average part whose orders must be estimated and involves a certain degree of differencing.

The performance and benefits of ARIMA models has been fiercely discussed in the aftermath of the M3-competition, whose results have been published in Makridakis & Hibon (2000). In this publication, the organisers criticise the approach of building statistically complex models like the ARIMA model,

disregarding all empirical evidence that simpler ones predict the future just as well or even better in real life situations, for example provided by their competition.
furthermore, add that the only advantage a sophisticated model has compared to a simple one is the ability to better fit historical data, which is no guarantee for a better out-of-sample performance.

## 4.3 Artificial neural networks

In the past few years, there can be observed a great interest in machine learning, especially in artificial neural networks domain. Artificial neural networks are tools, that are being used today for solving huge number of tasks from different areas. The most frequent examples are time series forecasting, pattern recognition, data clustering and classification. Such a great success is determined by several reasons.

1. Artificial neural networks represent exclusively powerful tool, that enables to reproduce very complex nonlinear dependencies. For many years linear models played the leading role in the most areas, as there were a lot of well-designed and optimized tools, which satisfactorily coped with assigned tasks, but problem was with tasks, for which the linear approximation is unsatisfactorily.
2. Artificial neural networks are learning from examples. Artificial neural networks receive a set of representative examples and then a learning process starts, which tries to find and extract the structure of data. Certainly, proper application of artificial neural network demands specific requirements for a correct formulation of representative data set and network's architecture. However, proper construction of a such artificial neural network allows to cope with tasks, which can be solved by the traditional algorithms only with the great difficulties. For example, pattern recognition task, practically used for face recognition, solving it in traditional way would result in a very complex problem.

An extensive summary of work done in multilayer perceptrons can be found in Zhang et al. (1998), which is somewhat outdated but still frequently cited and very relevant in terms of guidelines given. Not only reviewing many related publications, the authors also make recommendations concerning network architecture (number of hidden/input and output nodes and their interconnection), activation functions, training algorithms, data normalisation, training/testing sample size and performance measures. Another comprehensive literature survey, albeit limited to the application area of electrical load forecasting, can be found in Hippert et al. (2001), who motivated more rigorous research in the area by stating that many publications present seemingly mis-specified models that have not been sufficiently tested. [Juergen, 2014]

## 4.4 Recurrent Neural networks

One of the main features of time series data is the sequential pattern of the dataset. A time series data point carries information from past series of data points and hence if we train a neural network, it should not learn every information from the scratch but rather it should have a memory to retain sequential information which is passed through a model. For e.g. While watching a movie, every scene is a continuation of previous scenes and to understand that scene we need to have context of previous storyline. This calls out to have a **persistence of thoughts** in our mind.

A traditional neural network is unable to provide that context or storyline over the period of the movie and hence we needed an advancement which would persist information from past data. The limitation of NN is because of its fixed size of input and output vectors. This does not allow the model to save the growing context by reading the input over time.

Recurrent neural networks are an advancement over the conventional neural networks which saves information from previous steps. RNN have cyclical connections over time. Also, an internal state is maintained throughout the network and after each step, that state is updated with activations of each step, which provides a memory to the network. A recurrent neural network or RNN is a neural network that contains a cycle besides the general flow of a traditional NN. This cycle allows information to be passed from one step of the network to the next. The working of this process is such that

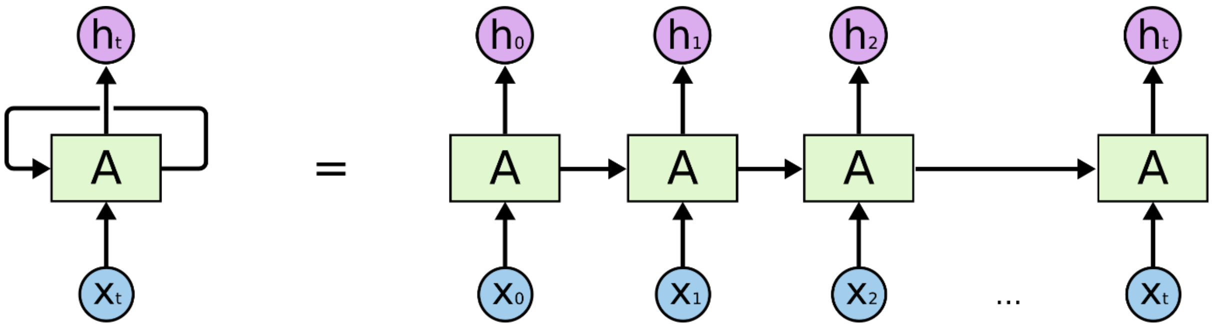


*Figure 4 Recurrent Neural Network Loop (Christopher, 2015)*

This cycling can also be considered as a chain of copies of same network with each step in the neural network(A) takes an input(X) and outputs an immediate output(h) as well as passes a message to next step. This chain like structure of neural network provides a very native or default architecture for the sequential or time series data. The simplest RNN can be considered as a fully connected neural network if we spread out or unroll the time axis. (Kondratenko et al, 2003)

One fundamental issue with RNN is that it can only handle short term memory and it renders itself ineffective if we have longer term dependency. RNN suffers from vanishing gradient problem.

According to [Razvan et al, 2013], RNN face vanishing and exploding gradient problem which makes it hard to train RNN effectively. [Xavier et al 2011] showed that ReLU activation function survives this issue longer than sigmoid function.

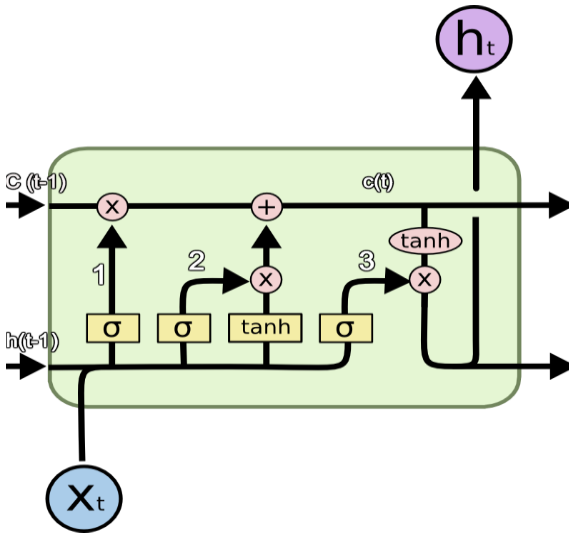
## 4.5 Long Short-Term Memory Networks

LSTM networks (Christopher, 2015) are an extension of Recurrent neural networks as they overcome the problem of vanishing gradient problem and help in remembering information for much longer period. A simple enhancement in design is that a node in RNN is replaced by a memory cell in LSTM. So rather than RNN where each node is a single node with a single activation function, an LSTM architecture replaces each node with a memory cell which stores its own information and maintains a cell state of its own. Each cell inputs the previous cell state and input, and outputs new cell state.

The working of LSTM cell can be explained in three main steps:

1) We choose the what to **remember** from previous cell state and forget the information which is not needed.
2) Control and selectively **update** the cell state based on new input that we have read.
3) We can control and selectively **output** the part of cell state as the new state.

Let’s discuss the architecture and how it achieves above mentioned steps. An LSTM cell consists of three gates: The forget gate, Input gate, and output gate.



The horizontal line going through the top of LSTM cell carries the LSTM cell state which is a common chain along the LSTM cell layer.

*Figure 5 LSTM Memory Cell contains four interacting Neural network layers demonstrated by yellow squared boxes (Christopher, 2015)*

Forget Gate: This gate gives system a chance to select the information we want to remove from cell state. In this gate, a sigmoid function decides the amount of information to retain.

Input Gate:

Input gate is basically a hidden layer of sigmoid activated nodes, which outputs values between 0 and 1 and they determine amount of input to flow into the unit.

Output Gate:

The output has a sigmoid gating function which determine which values of the state are output from the cell.

The LSTM cell is very flexible, with gating functions controlling what is input, what is “remembered” in the internal state variable, and finally what is output from the LSTM cell. Due to the large number of parameters in the LSTM, it is assumed that LSTM networks have large computational complexity.

## 4.6 Gated Recurrent Units

A gated recurrent unit (GRU) is a successful recurrent neural network architecture for time-series data. LSTM models were developed as an advancement from recurrent neural networks. However, the LSTM has the complex structures and numerous parameters with which to learn the long-term dependencies. As a way of reducing the number of parameters while avoiding the vanishing gradient problem, a gated recurrent unit (GRU) was proposed in (Cho et al, 2014); the GRU has only two gate functions that hold or update the state which summarizes the past information. In addition, (Tang et al. 2017) show that the GRU is more robust to noise than the LSTM is, and it outperforms the LSTM in several tasks.

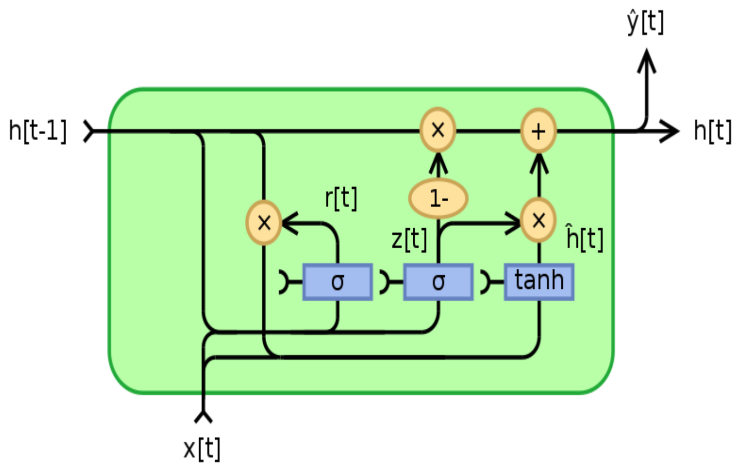


*Figure 6 Gated Recurrent Unit Cell, (Wikipedia)*

Above image shows a typical GRU unit architecture. This architecture has two gates with various operations.

<u>Update Gate</u>

Update gate for the GRU helps the unit to decide and control how much amount of past information needs to be passed in the next timestamp.

<u>Forget Gate</u>

this gate is used from the model to decide how much of the past information to forget.

## 4.7 Convolutional Neural networks

Convolutional neural networks are another form of artificial neural networks. A convolutional neural network (CNNs) is a biologically-inspired type of deep neural network (DNN) that has recently gained popularity due to its success in classification problems (e.g. image recognition or time series classification). The CNN consists of a sequence of convolutional layers, the output of which is connected only to local regions in the input. This is achieved by sliding a filter (Yu et al, 2014), or weight matrix, over the input and at each point computing the dot product between the two (i.e. a convolution between the input and filter). This structure allows the model to learn filters that can recognize specific patterns in the input data. Recent advances in CNNs for time series forecasting where the authors propose an undecimated convolutional network for time series modelling based on the undecimated wavelet transform and one in which the authors propose to use an autoregressive-type weighting system for forecasting crime time series, where the weights can be data-dependent by learning them through a CNN. In general literature on crime time series forecasting with convolutional architectures is still scarce, as these types of networks are much more commonly applied in classification problems.

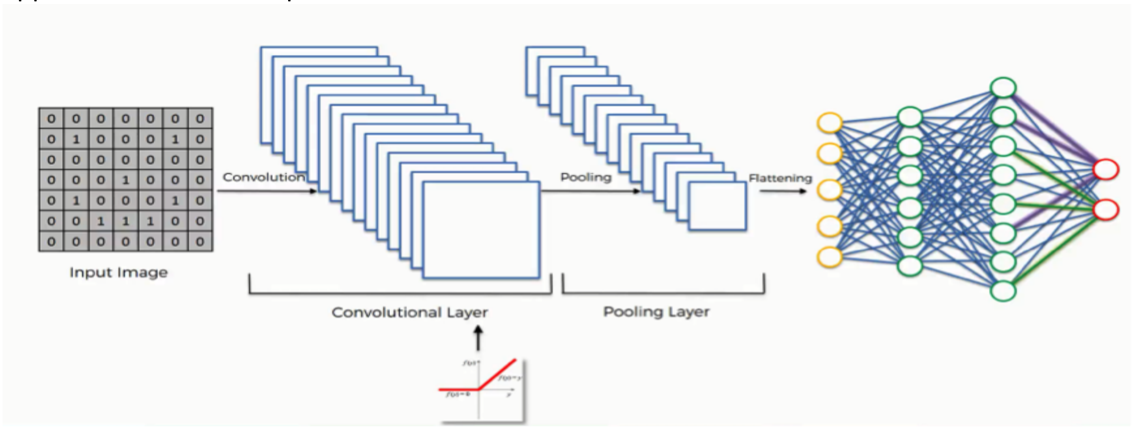


*Figure 7 Convolutional Neural Network Layers*

Intuitively, the idea of applying CNNs to time series forecasting would be to learn filters that represent certain repeating patterns in the series and use these to forecast the future values. Due to the layered structure of CNNs, they might work well on noisy series, by discarding in each subsequent layer the noise and extracting only the meaningful patterns.

One of the main reasons of using convolutional layer instead of fully connected layer is that if we use fully connected layer to flatten the input, we will end up with massive number of parameters which would require great amount of resources. We can see this issue in LSTM where large number of parameters results in large computational complexity. So unlike LSTM, 1D CNN can be used on low resource environments. Also, lower parameters in the CNN will result in avoiding overfitting.
[R. Mittelman, 2015]

## 5. Experiments and Discussion

The main aim of this dissertation is: analysis of provided time series and then develop various time series forecasting model and then compare the models to find the most optimized model among them.

The approach is mainly divided into two major segments:

1. Box Jenkins Approach: this methodology includes AR, MA and ARIMA models. This approach uses linear prediction on the given data.
2. Artificial Neural network approach: This method includes various variants of artificial neural network to determine the adaptability and scalability of the models. We will start by exploring the simple fully connected neural network. Later we will move towards recurrent neural networks which have some theoretical scalability advantages over the fully connected neural networks. Then, we will go into Long- Short term memory networks, LSTM, which are specific form of recurrent neural networks. We will also explore one dimensional convolutional neural network to explore its advantages in sequential data processing and prediction. Finally, we will try to explore Gated recurrent units, GRU, which are again a part of neural networks. This model has less computational overload than LSTM.

Also, the first segment mainly deals with univariate time series and they do not have capability to handle multivariate dataset. Hence our approach will go towards analysing all the models from both segments on univariate time series to find their efficiency on given model and later we will use second approach to analyse the multivariate dataset to determine their predictive power.

Ideally, we would have liked to determine a model which has best predictive capability on both univariate and multivariate dataset. Also, we would have ideally liked to have a model which has the best linear and nonlinear capabilities. As we know from no free lunch theorem, there is no model that can perform exceedingly well on all factors. Hence, we explore all the models and try to determine their efficiency on various parameters. We expect that using a combination of these two segments will help us in addressing both linear and non-linear properties of forecasting models.

## 5.1 Dataset Explanation

The first and fundamental step is the access of data. This data is the incidence-based and hence each row of data indicates one incident and the description of that incident. The data is stored in the SQL server atmosphere and is divided into various relational schema designed tables. The first goal is to create a dataset which would be easily consumed by the model. The initial exploration showed that data is mainly distributed among three tables where each table had following information, a crime event description, type of crime, and severity of crime. Next step included creating a stored procedure in SQL server and export the data from the above-mentioned tables into the .csv file. The aim of exportation of data to a file is done to limit the creation of various database connection to fetch data that various forecasting models need.

We have been provided the crime incident dataset from year 2001 to 2007. One of the necessities of time series dataset is the uniformity of time index across the dataset. Hence, first step of importing the dataset from the client system through the stored procedure in SQL server is the aggregation of the incident on a uniform time index. The first dataset exported from system contains a series of daily aggregated incidence dataset. Similarly, a second stored procedure exports a multivariate incidence dataset where each row of dataset is daily aggregation of 4 types of crimes namely:

1. Theft / Burglary in house
2. Nuisance
3. Pickpocketing on streets
4. Theft of mopeds or bicycles on street.

At the end of this data import phase, we have two datasets of incidence. A univariate dataset which contains daily aggregation of all the incidences reported to the police. A multivariate dataset will contain daily aggregation of four selected types of incidence based on interest. Later, we will also try to see the effect of the models on the weekly and monthly aggregation of total incidences and type of incidence.

## 5.2 Data Exploration

We will initialize the process of data exploration to understand the hidden insights from the dataset imported from the client. This process includes visualizing the dataset to catch if we can just see some patterns in the data without any transformation in the dataset. Later, we will try to smoothen the dataset to remove the noise and find some patterns in the dataset.

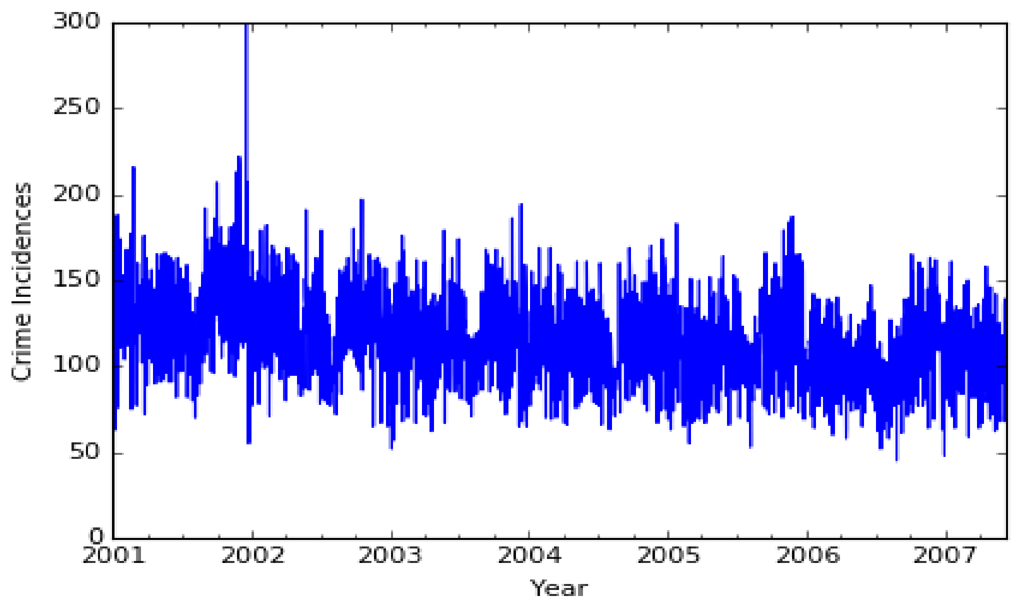


*Figure 8 Univariate Dataset distribution of crime incidences*

Above raw visualization of data shows the distribution of total crime incidences over the year period of 2001-2007. The first insight we can hypothesize from looking at raw visualization is that data is evenly distributed over the years and there is a slight trend in the dataset along with the seasonal variations. To calculate the mentioned hypothesis on data, we can calculate if the mean, variance and time covariance of the dataset remains constant.

## 5.3 Statistical Model experiment

### 5.3.1 Testing stationarity

There are two different ways to test stationarity:

1. Plotting rolling statistics: This technique is also assumed as a visual technique. We tried to plot the moving average and variance and see if the moving average and its variance changes over time.
2. Dicky- Fuller Test – This test is a mathematical approach to determine if the time series is stationary or not. This test was developed by David Dicky and Wayne fuller in 1979 to test whether a series contain unit root feature, which can severely impact a statistical inference.

The intuition behind a unit root test is that it determines how strongly a time series is defined by a trend.
Dicky fuller test is based on linear regression. Dicky fuller test is the simplest approach to detect unit root but since many practical datasets are too complex to be captured by simple autoregressive model, hence we used augmented dicky-fuller test. The null hypothesis of the test is that the time series can be represented by a unit root, that it is not stationary (has some time-dependent structure). The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.
The primary differentiator between the two tests is that the ADF is utilized for a larger and more complicated set of time series models. The augmented Dickey-Fuller statistic used in the ADF test is a negative number, and the more negative it is, the stronger the rejection of the hypothesis that there is a unit root.

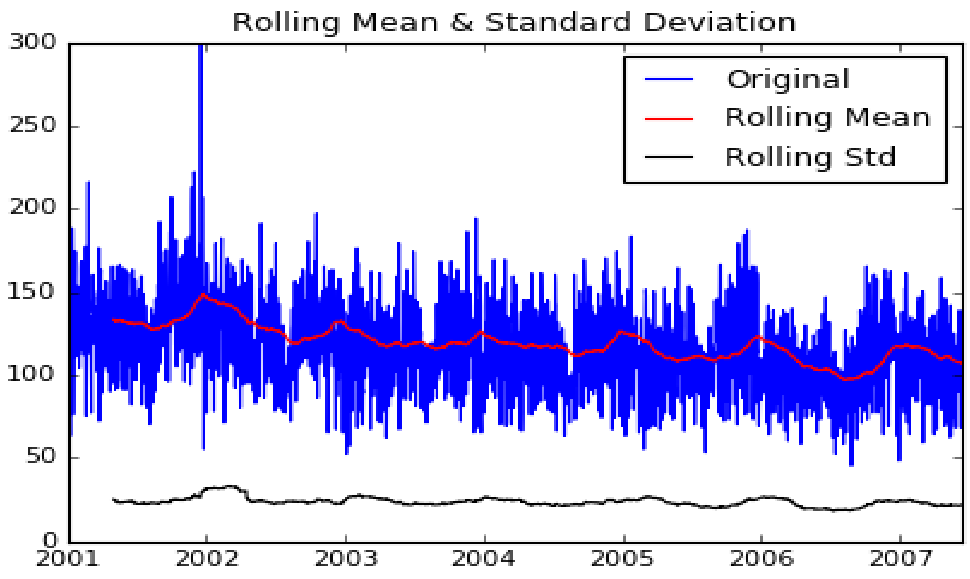
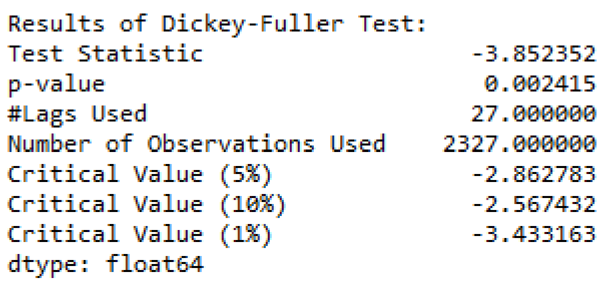


*Figure 9 Rolling Mean and Standard Deviation over the univariate dataset*

We drew a rolling mean and rolling standard deviation for the univariate dataset to explore the trend in the dataset. Here, one has used a quarterly (120 days) window for rolling mean and rolling standard deviation. We can see that though the variation in standard deviation is almost non-existent, there is a slight decreasing trend in the rolling mean line. This shows that there is a small long-term trend in the univariate dataset.



```
Results of Dickey-Fuller Test:
Test Statistic                   -3.852352
p-value                           0.002415
#Lags Used                       27.000000
Number of Observations Used    2327.000000
Critical Value (5%)              -2.862783
Critical Value (10%)             -2.567432
Critical Value (1%)              -3.433163
dtype: float64
```

Running the augmented dicky-fuller test shows that the test statistics is smaller than Critical values and the p-value is lower than threshold value (0.05) which says that the univariate dataset is stationary.

*Figure 10 Results of Dicky-Fuller Test*

Though the dataset is identified as stationary by augmented dicky-fuller test, it is still better to log transform the dataset since the statistics of dataset is not significantly different than the critical values and hence a log transform will make the distribution smoother and better meet the expectation of the statistical test.

### 5.3.2 Decomposition of time series data

A time series is composed of four components: trend component, seasonal component, cyclical component and random error component. If the dataset is visually decomposed into these four components, it can give valuable insights on the components which are mainly impacting the series. Here, we have decided to decompose our univariate dataset into its individual component.

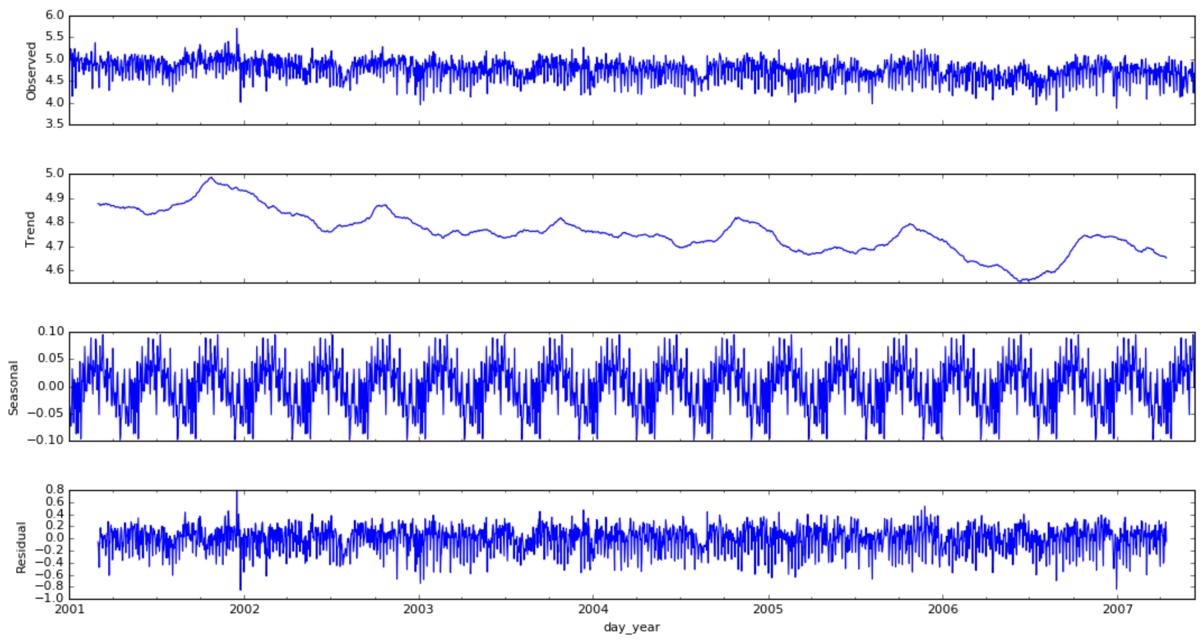
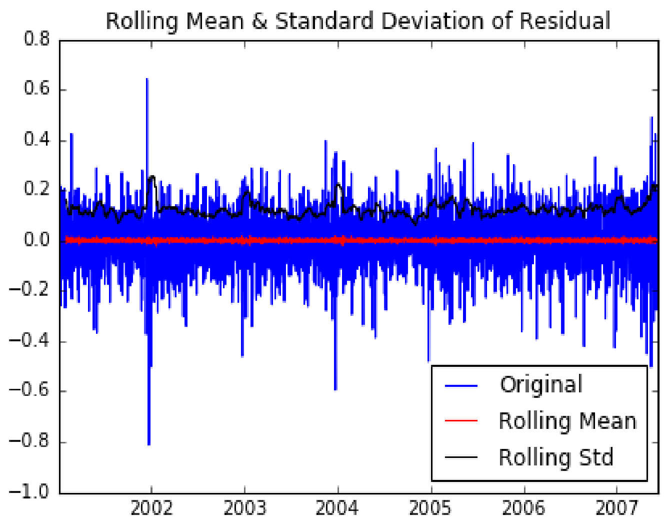


*Figure 11 Time Series Component Decomposition*

In above picture, we can easily see the small downward trend and a strong seasonal component in the dataset.



Also, the residual component is independent of time and is stochastic in nature. Desirably, the residual component is averaged around 0. This shows that error component is a random component and is not a function of time.

*Figure 12 Rolling mean and standard deviation of error component*

### 5.3.3 ACF And PACF Plot for optimal parameter selection

The ARIMA forecasting for a stationary time series is nothing but a linear (like a linear regression) equation. The predictors depend on the parameters (p,d,q) of the ARIMA model:

1. Number of AR (Auto-Regressive) terms (p): AR terms are just lags of dependent variable. For instance if p is 5, the predictors for x(t) will be x(t-1)….x(t-5).
2. Number of MA (Moving Average) terms (q): MA terms are lagged forecast errors in prediction equation. For instance if q is 5, the predictors for x(t) will be e(t-1)….e(t-5) where e(i) is the difference between the moving average at ith instant and actual value.
3. Number of Differences (d): These are the number of nonseasonal differences, i.e. in this case we took the first order difference. So, either we can pass that differenced variable and put d=0 or pass the original variable and put d=1. Both will generate same results.

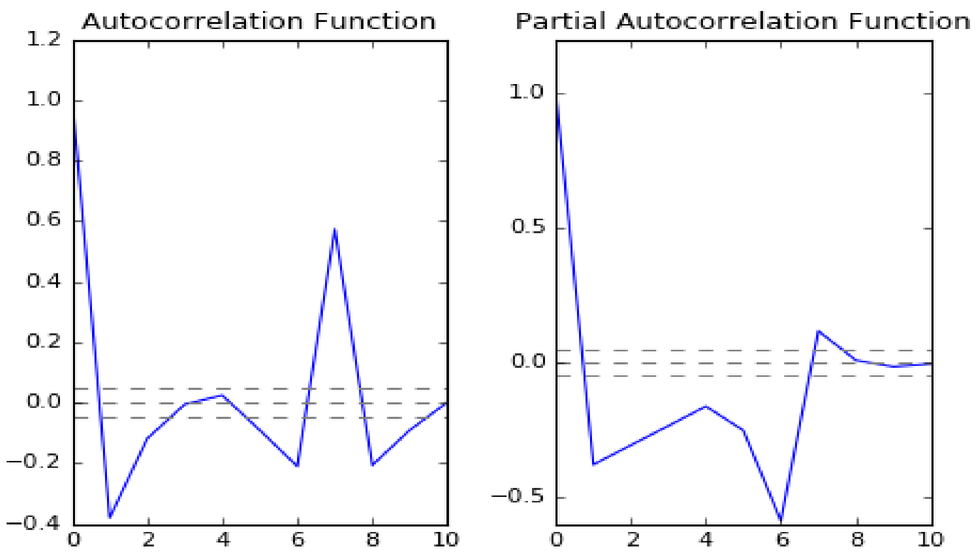


*Figure 13 Autocorrelation and Partial Autocorrelation Function plots*

As per the partial autocorrelation function, the value of p to be selected is the point when the upper threshold is crossed. This value is 7 as per the plot. Similarly, the value of q is 1. So, the Arima model used will have the parameters as (7,1,1).

### 5.3.4 statistical models Experiment & Discussion

#### 5.3.4.1 ARIMA Model

Autoregressive integrated Moving average models are the most general class of models for forecasting time series. We used standard python library to run the model. The parameters of ARIMA models (p=7,d=1,q=1)are chosen from the previous steps. Firstly, the univariate dataset is scaled to (0,1) scale. This might help model to fight non-linearity of the dataset.

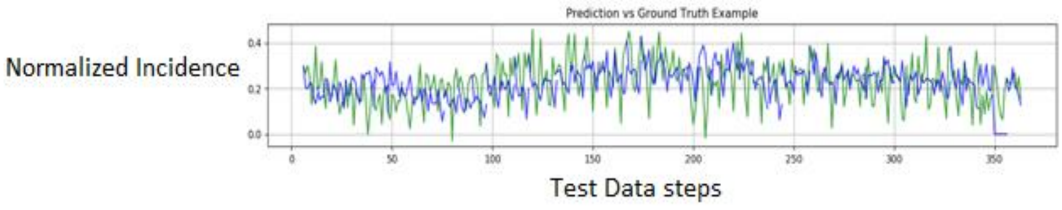


*Figure 14 Prediction plot of ARIMA model on the test data*

Because of linear property of ARIMA models, it is assumed that long term forecasting of ARIMA models will start turning insignificant quickly. Hence, we chose a 7-day prediction in one go. Again, this prediction will be used as ground truth in predicting next 7-day prediction. This iteration will help a model to predict entire test dataset.

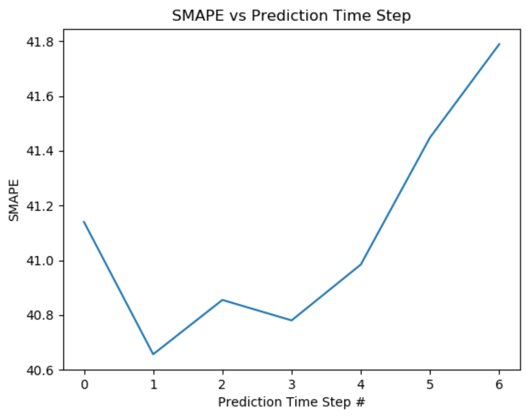


*Figure 15 SMAPE error value on different days prediction*

This plot shows that prediction error percentage is low initially but when the prediction step is increased to 7 days, the error becomes very large (~41%). This error percentage too large for the practical application. The lowest error percentage is on 2nd day prediction.

Clearly, ARIMA model's predictive capability is very low on the existing dataset since model has been used to fit seasonal data. This is done as per internship requirement to use minimal data specific parameters. Also, as the size of dataset is increased, short-term predictability of ARIMA will result in multiple iterations of test data, which would further reduce the application of ARIMA. Hence, we will use ARIMA as a benchmark model for further explorations of neural networks. ARIMA models are the univariate class of statistical models

## 5.4 Neural networks

Apart from the statistical method, this internship also included exploring some prominent neural network methodologies to address the issues of Box-Jenkins approach. The classical methods used for time series prediction like Box-Jenkins, ARMA or ARIMA assumes that there is a linear relationship between inputs and outputs. Neural Networks have the advantage that can approximate any nonlinear functions without any apriori information about the properties of the data series.

### 5.4.1 Training Setup

We have tried to keep the training parameters of different variants of neural networks as uniform so that we can have a meaningful and fair comparison between them. Few of the training parameters are:

- We used Keras machine learning library in python (https://keras.io/) on top of TensorFlow. TensorFlow is an open-source machine learning framework.
- We used mean squared error (MSE) as the loss function. MSE has the tendency to penalize large errors more heavily in the training processes.
- We used ADAM optimizer for all the neural network variations in this internship. The learning rates are determined automatically, based on a moving estimate of the per-parameter gradient, and the per-parameter squared gradient. This is in huge contrast with stock vanilla Stochastic Gradient Descent. [Kingma et al, 2017]
- **Dataset change:** One of the major characteristic difference of Artificial neural networks with statistical models is that Artificial neural networks undergo supervised learning which means that the data for training an Artificial neural network model must have attributes as well as a target. So, we must change the dataset in a way that a series of attributes will also have target values which will act as output values. An input to Artificial neural network can be a single attribute or a group of attributes entering an Artificial neural network unit and the output can be a single output or a collection of variables as an output. Since our desired efficiency will be that a model should predict a week-long forecast by training on existing data. Here, we have hypothesized that will use a 30-day historical data to predict 7 days of forecast. Hence an input into a model will be 30 days of historical incidence data and the output will 7 days of forecast.
- Batch size has been fixed on 100. A batch size determines the amount of training sample used to compute a loss function.
- Maximum epoch has been set to 1000. In case of reaching an optimum, we have set an early stopping criterion where if the SMAPE value does not change for 15 epochs, the fitting will stop.
- We have used 70-30 partition of the dataset. Hence, 70% of the dataset is used to train the model and 30% is used to validate the generalization of the model.
- For better predictability of the model, the dataset is scaled to an interval of (0,1). This helps model to learn the dataset more efficiently.

## 5.4.2 1-Layer Linear Neural network experiment

**Introduction**

The first step towards using neural network is the 1-layer linear neural networks. These networks are the most basic and simplest type of feed-forward neural network. Generally, this network mainly acts as a linear regressor where an output can be considered as linear regression of the input provided into the network. The weight of the input is updated based on the difference between the actual output and the predicted output.

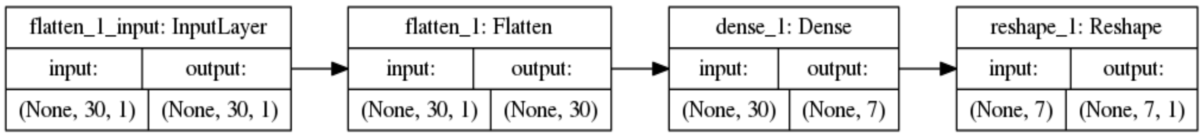


*Figure 16 Tensorboard visualization of Single layered NN implementation in Keras*

As demonstrated from above pic, we are using a sequential layered model from keras where the Dense layer acts as a simple 1-layer neural network. A flattened input from flatten layer is provided to Dense layer which outputs 7 days forecast which reshaped in last layer.

**Experiment**

By running the above-mentioned trained model on test dataset, we have plotted the prediction against the ground truth.

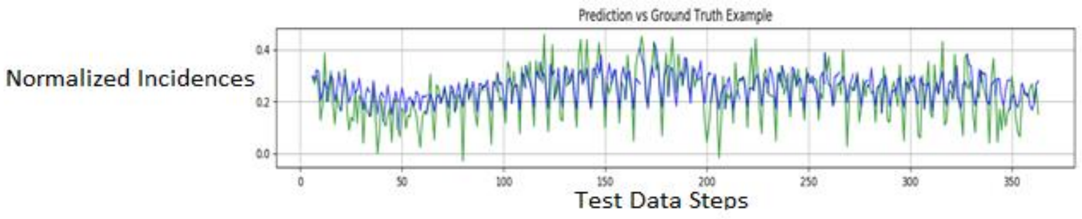


*Figure 17 Visualisation plot for Single Layered NN prediction over test data*

As we can see that the model is able to predict the low points (dips) in the in incidence relatively better than the high points (peaks) in the crime incidence. The reason is that the dips in the crime data is seasonally repetitive and hence, model has learnt the pattern and thus it is easier to predict. Also, single layered neural networks prediction for initial 100 test data points is not satisfactory and it fails to understand the dips in the data due to the linear predictive capability of the model. Since random error component in a time series are stochastic in nature and do not have a clear pattern, linear models have tough time in predicting this component. This limitation of the model makes it incompatible for the practical dataset since crime dataset will always have non-linear property.

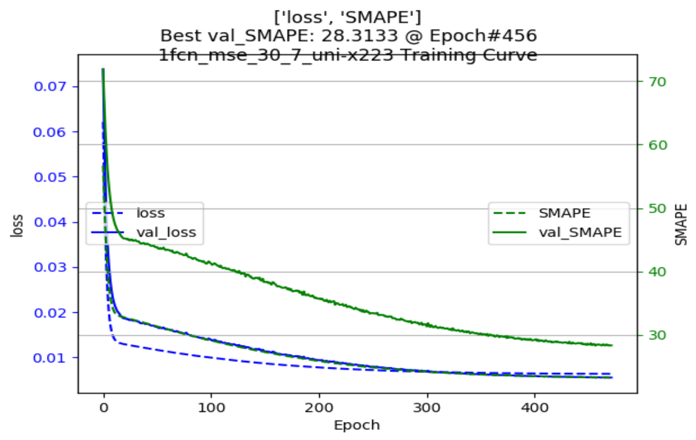


*Figure 18 Single Layer NN loss and SMAPE curve on training and test data over increasing number of epochs*

It shows that it took a lot of epochs (456) for SMAPE to converge to an optima. Also, loss curve shows very quick convergence to a value around 0.01 and there is no significant improvement in the loss value after 100 epochs. This shows that the model has reached and stuck on local optima and is unable to go beyond due to its linear property. Clearly, we need some improvement in the configuration of model to provide model enough support to come out of its local optima and reach toward global optima.

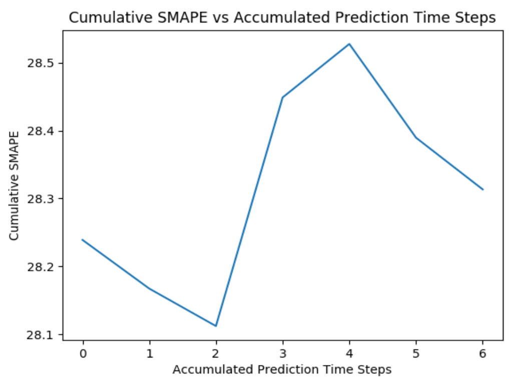


*Figure 19 Single Layered NN cumulative SMAPE over increasing prediction steps*

Above plot shows the Variation of cumulative SMAPE over the course of a week ahead forecast. For this model, Symmetric mean absolute percentage error (SMAPE) is lowest on third day prediction. It reaches its worst prediction capability on 5th day prediction and then its decreases for next two days prediction. This plot is strangely unpredictive and doesn't show an expected prediction degradation when increasing number of steps. This needs to be investigated further.

### 5.4.3 Deep fully connected neural network

**Introduction**

Deep fully connected neural network is an extension of simple 1 layered neural network. Unlike the former neural network example, this network has 3 hidden layers to provide additional scrutiny of input data. Also, this neural network variation will have non-linearity which in turn will provide additional predictive capability to the model.

Deep fully connected networks provide more complexity by incorporating 3 hidden layers. Each hidden layer is a 1 Layered non-linear neural network. One of the disadvantages of this model could be a problem of overfitting. Unlike single layered linear neural network, this network has large number of parameters and it is non-linear in nature. These properties will mean that model will try to learn the training data patterns and noise to the extent that it will impact its performance on new data.

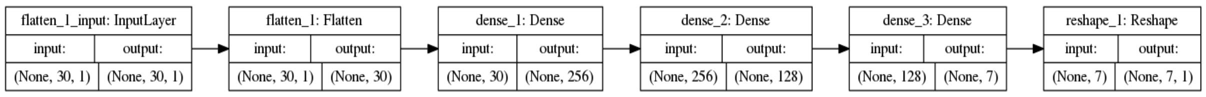


*Figure 20 Tensorboard visualisation of Fully Connected NN implementation in Keras*

Above picture shows the Keras sequential layered structure of the experiment. This architecture contains 3 dense layers. A non-linear activation function, ReLU, is attached after every layer to provide non-linearity to the model. Here, an input layer inserts the flattened data into the first dense layer and then in third layer the data is transformed into the output shape of 7 days.

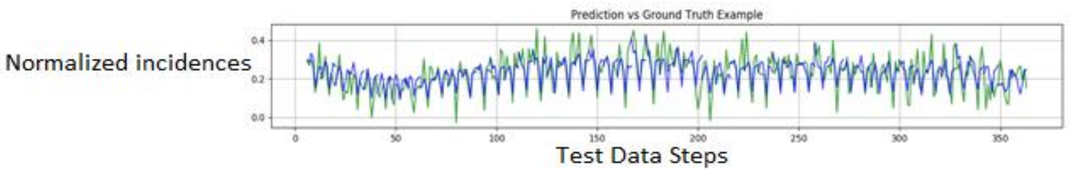


*Figure 21 Visualisation plot for Fully connected NN prediction over test data*

Comparing this plot of test data prediction with the one layered linear neural network output, we can assume that since this model is non-linear in nature, it tries to learn the patterns of the initial 100 test data better than previous experiment. Also, the peaks and dips in the incidence are better predicted by this model than the previous model.

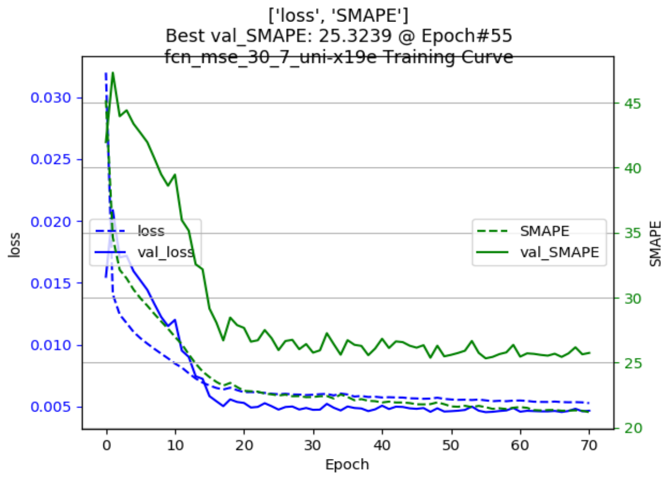


*Figure 22 Fully Connected NN loss and SMAPE curve on training and test data over increasing number of epochs*

In above pic, we can see that the unlike previous model, SMAPE of the test data start converging after 20 epochs. Also, the validation SMAPE converged after 55 epochs. It’s evident from the plot that loss value converges at much lower level (0.005) than the previous model. The noise in the validation SMAPE is a reason of non-linearity of the model and the model tries to search towards the minimum optima. The quality criteria of validation SMAPE is 25.32 which is better than linear neural network.

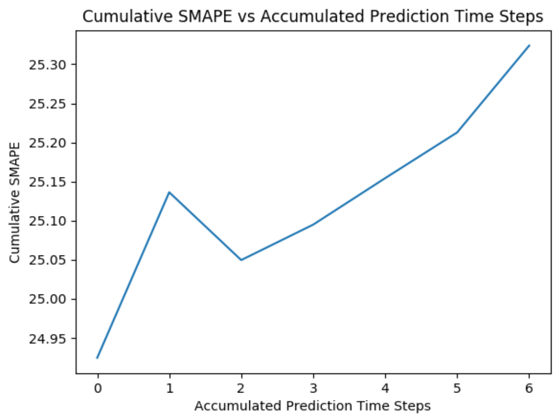


*Figure 23 Fully Connected NN cumulative SMAPE over increasing prediction steps*

From the above plot of Cumulative SMAPE verses prediction time step, we can infer that the cumulative SMAPE value decreases till 3rd day and it starts going worse with increment of following day. This helps us in defining a restriction on number of days a model can effectively predict in future.

Hence, deep learning neural networks are better at fitting even complex training data due to its higher complexity and non-linearity but there is a chance of higher overfitting. In our case, it performed better than linear neural networks.

### 5.4.4 Recurrent neural network explanation

As discussed in previous chapter, recurrent neural networks are an extension of feed forward neural networks. These networks have a loop structure for its data flow which creates a short-term memory for the network which gives RNNs to identify longer patterns in data. Also, with loop like structure, the computational complexity of RNNs are significantly higher than the conventional feed forward neural networks. We mainly used two types of recurrent neural networks in this internship:

### 5.4.5 LSTM explanation

**Introduction**

Long short-term memory networks are specific types of recurrent neural networks which solved a major drawback of recurrent neural networks, vanishing gradient problem. This problem is caused when the input is transformed repeatedly in a loop, after a while the rate of change of weight vector becomes so small that it becomes negligible and hence the learning stops. This is handled in LSTMS by a very specific architecture of LSTM memory units involving various gates. LSTM have been widely used in the time series forecasting because of its ease of handling non-linear datasets with large patterns.

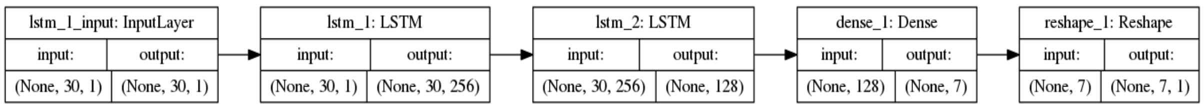


*Figure 24 Tensorboard visualisation of LSTM Networks implementation in Keras*

The model architecture has 2 LSTM layers where first has 256 LSTM units and second has 128 LSTM units. This model takes input of 30 days of data and predict the next 7 days of data.

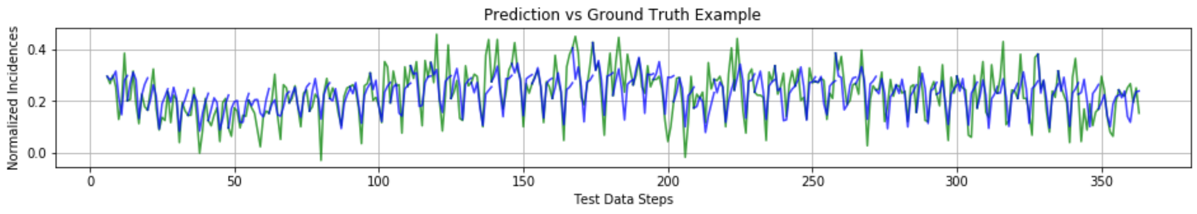


*Figure 25 Visualisation plot for LSTM Networks prediction over test data*

In above picture, we can see the fitting of 7-day predictions on the test dataset. For our current dataset, this LSTM model performs relatively equivalent to deep learning fully connected neural network. Clearly, the LSTM model can catch the highs and lows of the incidence datasets well.

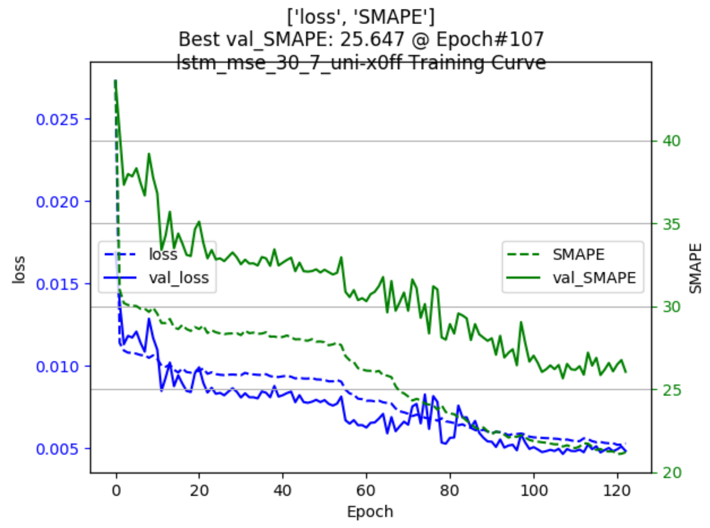


*Figure 26 LSTM Networks loss and SMAPE curve on training and test data over increasing number of epochs*

We can see from the above plot of loss and SMAPE over different epochs that though validation SMAPE decreases continuously, it doesn't converge to a minimum optimum until 107 epochs. Also, the value of validation SMAPE on test dataset is nearly same as deep learning fully connected neural network. The main reason of comparable performance of deep fully connected neural network with LSTM is the limited amount of data available for the analysis. It can be hypothesized that if the LSTM has large amount of data to learn the long-range patterns, it will outperform the conventional deep learning networks.

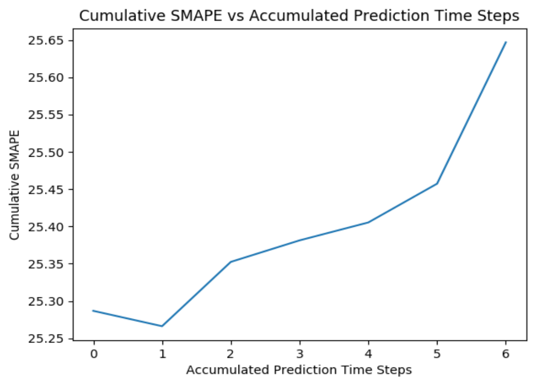


*Figure 27 LSTM Networks cumulative SMAPE over increasing prediction steps*

We can infer from the above plot that for LSTM networks on this dataset, the prediction capability decreases with increasing number of days of prediction. There is a significantly higher increased cumulation of error (SMAPE) from 6th to 7th day prediction. We can infer from this plot that increasing number of days of prediction from 7 days will result in significantly lower quality of prediction.

### 5.4.6 1-D Convolutional neural network explanation

**Introduction**

Convolutional neural networks are generally used in the image recognition/classification problems, but we can use 1-D convolutional neural networks to learn the patterns in time series data and predict the number of incidences in future. This neural network is considered to avoid overfitting and is faster than the recurrent neural network because of less parameters to train the data.

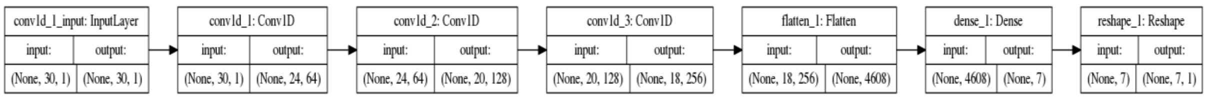


*Figure 28 Tensorboard visualisation of 1D CNN implementation in Keras*

This model architecture has three 1D CNN layers. Each CNN layer is separated by ReLU activation function. Here, an input layer receives 30-day input as a 30 x 1 matrix.

First convolutional neural network has 64 filters to learn 64 different features in the dataset, each filter has a kernel size of 7. So, a kernel of 7 days will be taken to lean each feature and create convolution for the dataset. Each column of the output matrix holds the weights of one single filter.

Similarly, second 1D CNN layer has 128 filters, each filter having kernel size = 5. Third 1D CNN has 256 filters, each filter having kernel size = 3. It’s evident that we have continuously increased the number of filters and decreased the kernel size. It means that we want to initially learn small number of long-term patterns and then gradually large number of short-term patterns under those convolutions. This approach leads to large number of computations with less resources.

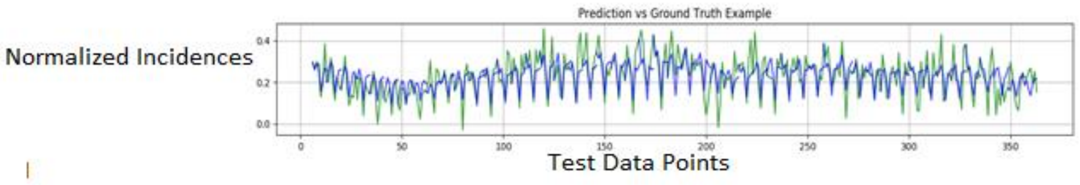


*Figure 29 Visualisation plot for 1D CNN prediction over test data*

Above picture shows prediction plot of test data. By looking at the plot, model has performed relatively well, and it has also captured most of dips in incidence data. Also, it can calculate many peaks in the complex data.

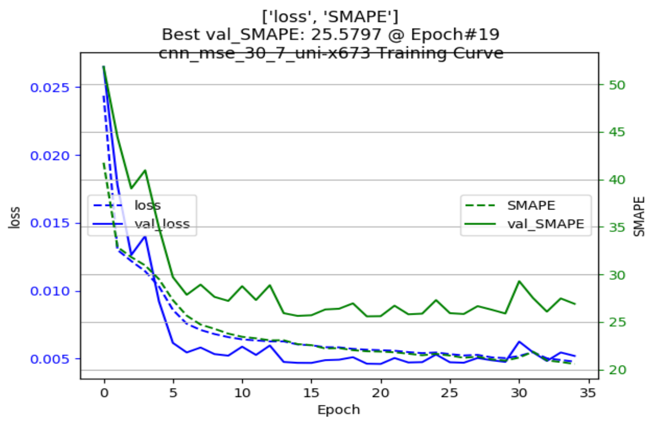


*Figure 30 1D CNN loss and SMAPE curve on training and test data over increasing number of epochs*

Above picture shows variation of loss and SMAPE over training and test datasets over different epoch runs. We can clearly see that the model reached the convergence of validation SMAPE at 19th epoch with a SMAPE value of 25.58, the lowest error percentage till now. This shows that 1D CNN was able to train the model with less overfitting and less epochs. This was possible due to less computational complexity of model due to less parameters from convolutions.

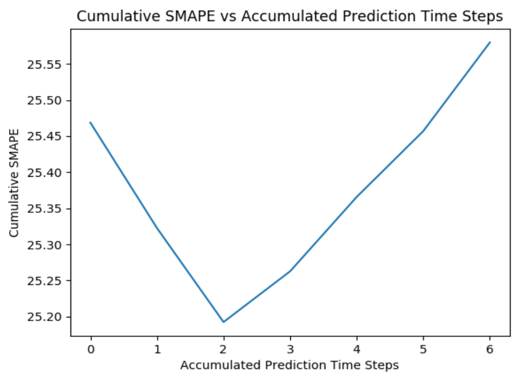


*Figure 31 1D CNN cumulative SMAPE over increasing prediction steps*

It's very interesting to note the variation of SMAPE over different step-ahead prediction for this model. We can see that the cumulative error percentage decreases for initial 3 days and then it starts increasing for later days prediction. This could be investigated further on why the model's cumulative error decreased for initial 3 days.

It's safe to assume that CNN works very effectively for the dataset since these models have less parameters. Also, since the error percentage on training and test data is not very different, it shows that model has less overfitting. This model performs very good for the structured data. Also, since this model needs less resources to train, we can use this model instead of recurrent neural network to get same level of accuracy. One of the major disadvantages of this model for time series analysis is the lack of related work in this category. Most of the work for CNN involves facial recognition (Amogh Gudi, 2016) and image classification and hence, there is not much work available in time series analysis by 1D CNN.

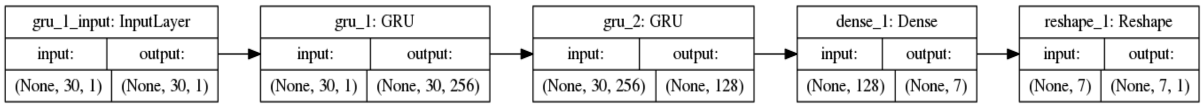
### 5.4.7 Gated Recurrent Units

**Introduction**

Gated recurrent units are a variation of recurrent neural network. It was developed with twin purpose:

- Remove the vanishing gradient problem of RNN which helps GRU to learn longer patterns.
- Solve the computational complexity of LSTM. This problem had mainly limited the use of LSTM to big resource environment.

The structure of GRU is very similar to LSTM except that GRU has just two gates to determine the amount of previous data to retain or remove. LSTM had 3 gates for the same purpose and it increases the computation for the model.



gru_1_input: InputLayer
input: (None, 30, 1) output: (None, 30, 1)
gru_1: GRU
input: (None, 30, 1) output: (None, 30, 256)
gru_2: GRU
input: (None, 30, 256) output: (None, 128)
dense_1: Dense
input: (None, 128) output: (None, 7)
reshape_1: Reshape
input: (None, 7) output: (None, 7, 1)

*Figure 32 Tensorboard visualisation of GRU Networks implementation in Keras*

Input layer receives 30-day data to predict 7-day ahead incidences. This data is passed into first GRU layer which contains 256 GRU units. Since return sequence is true for this layer, all the results will be passed on to the next GRU layer where each row of the output represents one output from a GRU unit.

Next GRU layer contains 128 units and hence it will provide 128 values as result. This result when assed through dense (single layered neural network), it will provide 7 output values for 7-day prediction.

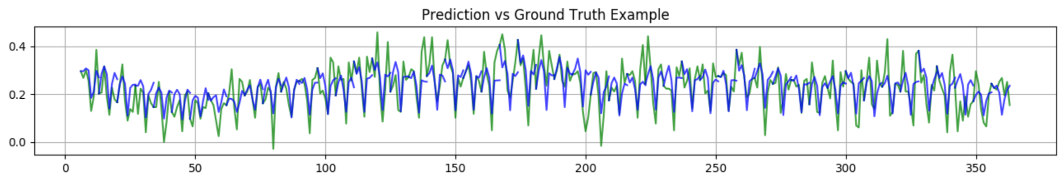


*Figure 33 Visualisation plot for GRU Networks prediction over test data*

Above picture shows the prediction plot for the test data. We can see that model is able to predict most of the incidence changes in the dataset. Also, the dis and peaks in the dataset is predicted in a satisfactory manner.

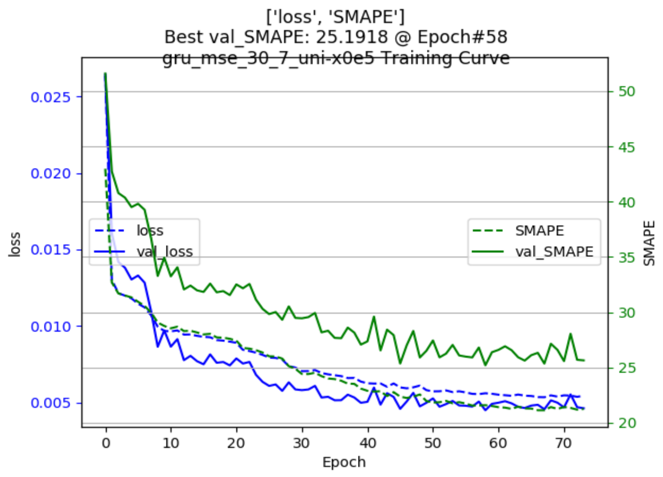


*Figure 34 GRU Networks loss and SMAPE curve on training and test data over increasing number of epochs*

Above picture shows the loss and SMAPE for train and test dataset over different epochs in the training period. We can see that values have decreased gradually over the increase of epochs and the SMAPE value for test data has converged on 25.19 SMAPE at 58th epoch.

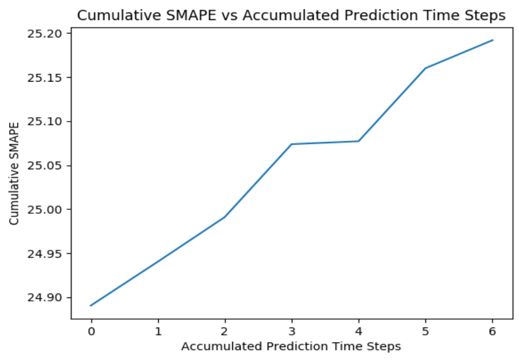


*Figure 35 GRU Networks cumulative SMAPE over increasing prediction steps*

We can say from the above picture that SMAPE value increases significantly with every increase in the prediction time steps. Hence, any further increase in the time step will result in more error and will nullify the advantages of having a prediction for the day.

Hence, GRU has performed well on the dataset and we can say that when the number of parameters of LSTM is controlled (like in GRU), it can perform relatively well. GRU have less computational complexity than LSTM and hence it is easier to deploy on low resource environments as well. Also, GRU has long short-term memory like LSTM and so, it can learn longer data patterns in time series.

#### *5.4.7.1 Date injection in input*

Until now, we have imported data from univariate dataset which are like a sequential data. The models try their level best to understand the long term and short-term patterns in those datasets. To ideally work the neural network models on time series data, it is desirable that the model have time index as the input since a time index gives additional perspective or information to the model. For example, if day of the week is provided as input, model can learn the incidence pattern on weekends and later when the model is asked to predict weekends in test data, model will have better learning and hence give better predictions. Similarly, model can learn the incidence numbers on various dates of month.

We propose the addition of 3 extra columns in the dataset:

1. Normalized day of the month: This is the actual day of month divided by average number of days in a month.
2. Normalized month: This is current month divided by average number of months in a year.
3. Normalized day of the week: This is the actual day of the week divided by the number of days in a week.

We experimented with the univariate GRU model with date injection to predict above mentioned three date features. This experiment would further help us determine if addition of date feature into model will train the model to take the date into account and its main aim is to demonstrate the date predictability. So, this model will take 4 variables as input where 3 inputs are date, month and day of the week. It will predict same 4 outputs.

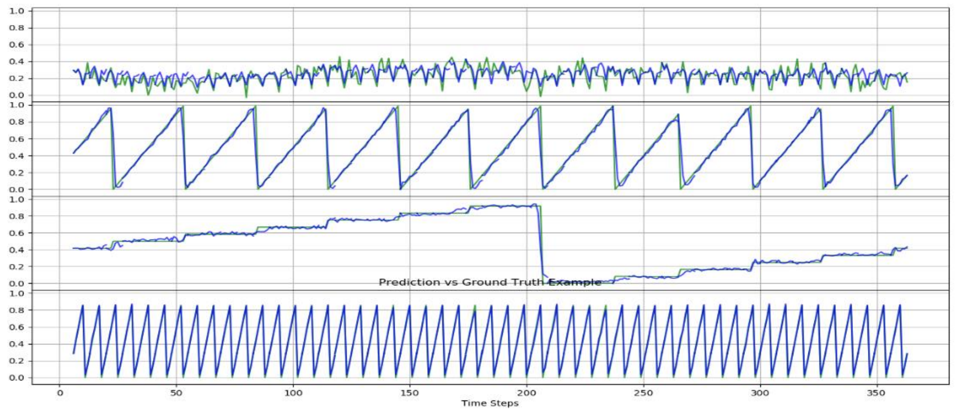


*Figure 36 Visualization plot: 1) univariate model prediction of test data, 2) date of the month prediction, 3) Month Prediction, 4) day of the week prediction*

Above picture demonstrates that the model can understand the date, month and day of the week effectively. The first plot is the timeseries data prediction. Second plot shows date prediction and it can understand the date. Third plot shows the month prediction where the model can gaze the month pattern. Also, the fourth plot shows models day of the week prediction which is very accurate.

This clearly demonstrates that model takes the date features into consideration to analyse the patterns in data and this eventually helps in overall prediction.

As per previous experiments, GRU has Performed relatively better than any other type of neural networks. Hence, we will take GRU network model as the benchmark model to implement DII(Date in input) proposal.

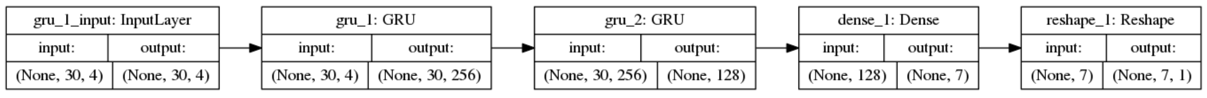


*Figure 37 Tensorboard visualisation of Univariate GRU Networks with date injection implementation in Keras*

Unlike the previous univariate GRU model, the input layer is injected with input in shape of (30 x 4) where other three columns are date, month and day of the week.

This input layer output is fed into the first GRU layer which has 256 GRU units. Each unit output is stored and passed into the next round in (30 x 256) shape where each column stores the output of one GRU unit.

This output is sent into the next GRU layer which has 128 GRU units and its outputs 128 value-long array. Later, this output is sent to a next fully connected neural network layer which outputs a 7 value-long array. This output is 7 day ahead prediction.

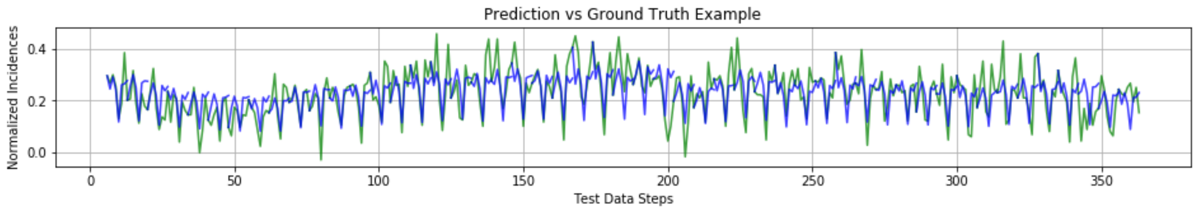


*Figure 38 Visualisation plot for GRU Networks with date injection prediction over test data*

Above plot shows the prediction quality on test data. The prediction has performed relatively well for GRU with date injection. The dips are predicted correctly and some of peaks that were not predicted in previous examples seem to be covered in this example. One reason of this improved prediction could be the models understanding that the incidences increase or decrease on certain days like weekends.

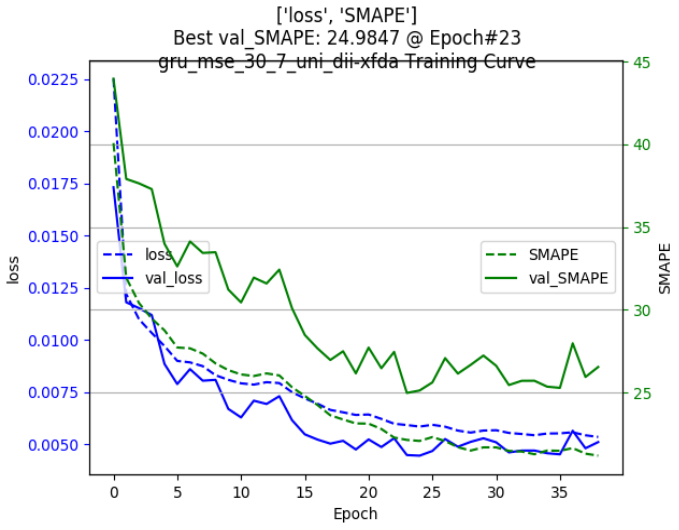


*Figure 39 GRU Networks with date injection model loss and SMAPE curve on training and test data over increasing number of epochs*

We can see that the loss function (MSE) has converged and looking at the loss variation on training and test dataset, GRU network with date injection has very good generality and it avoids overfitting on the training data. Also, SMAPE on the test data converges to a minimum at Epoch number 23 with a value of 24.98.

Test data converged SMAPE for the GRU with date injection is less than conventional GRU value (25.19 at epoch 58). This shows that date injection in GRU model has helped in better prediction it has also helped in the early convergence of SMAPE to a minimum. The only disadvantage we can think is the computational complexity by adding more columns in the datasets.

The same approach was applied on other neural network alternatives and we noticed that there is an improvement on most of the neural network variants. This shows that date injection is helpful for the model to learn the data patterns.

*Table 1 Effect of Date Injection in input of different Univariate Neural Network Alternatives*

| SMAPE | 1FCN | FCN | LSTM | 1D CNN | GRU |
|---|---|---|---|---|---|
| **For Conventional Model** | 28.31 | 25.32 | 25.65 | 25.58 | 25.1 |
| **For Model with Date Injection** | 25.63 | 25.38 | 25.41 | 25.19 | 24.98 |

### *5.4.7.2 Impact of different input size:*

Our earlier hypothesis was that a month of historical data will be enough to predict a week-long prediction. Next, we will try to experiment with different input sizes to determine the optimal number input days required to perform a week-ahead prediction.

We chose univariate GRU Model with date injection as the model since it outperformed other models in SMAPE criteria. We experimented with 7 different number of input days (1, 3, 7, 14, 30, 60 & 120).

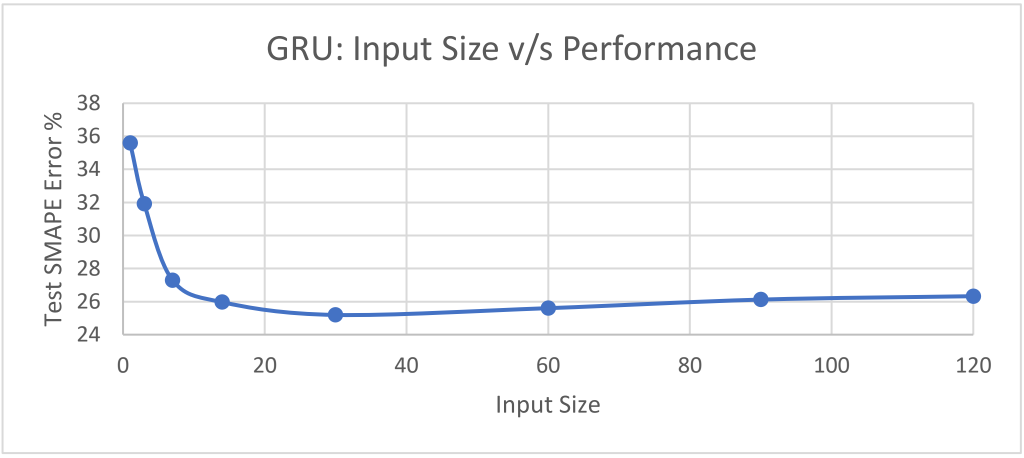


*Figure 40 Plot identifying the performance of GRU network on different number of input size*

This graph shows that the 30 days of input provides the least SMAPE error percentage on the test data. The error decreases linearly till 30 days and then it stays nearly same Hence, it proves our hypothesis that 30-day input is optimal data to predict next seven days.

### *5.4.7.3 Multivariate Analysis*

One of the major drawbacks of the Box-Jenkins approach is the univariate property of the model. This doesn't provide the model to analyse the certain effects of one variable on another variable's prediction. Each attribute in the dataset is not a discreate value and some of the effects of each attribute will affect the other. Taking an example of crime data, effect on number of incidences could be predicted correctly if we can also introduce other variables like population and climate in area. Also, multivariate prediction capability of a model could help predict different types of incidences simultaneously. Hence, Neural networks in time series analysis was introduced to also perform multivariate analysis. This analysis was performed on the multivariate incidence dataset containing 4 daily aggregated incidences.

Since GRU performed best for main experiments and since GRU gave best results with date injection, we chose GRU with date injection as the baseline model to run the multivariate time series forecasting. So, we will try to train a univariate model with each attribute (each incidence type) and then we will try to train a multivariate GRU model with date injection to understand the effect on the predictive capability. This model will take 30 days of incidence data in shape of (30 x 4) where each row will contain 4 types of incidences and 3 columns of date features. The output will contain 7 days or prediction where each prediction will contain 4 types of crime incidences.

Univariate analysis

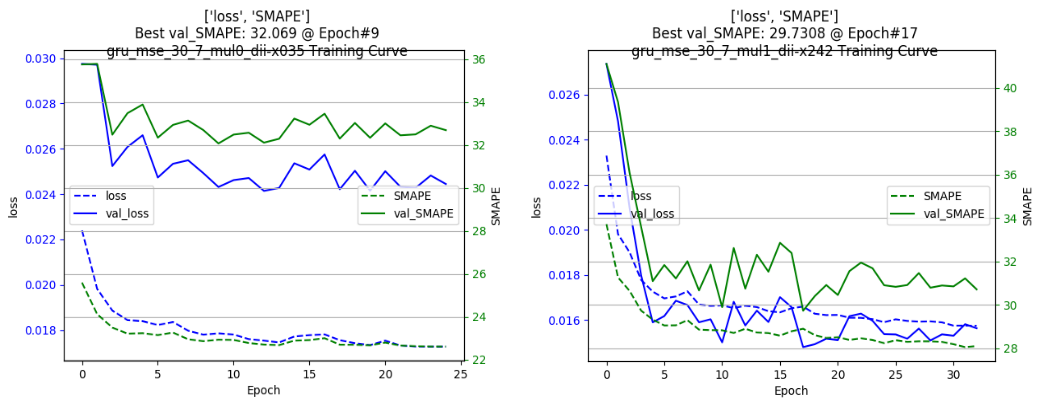


*Figure 41 Training Curve for 1) Theft & Burglary, 2) Nuisance*

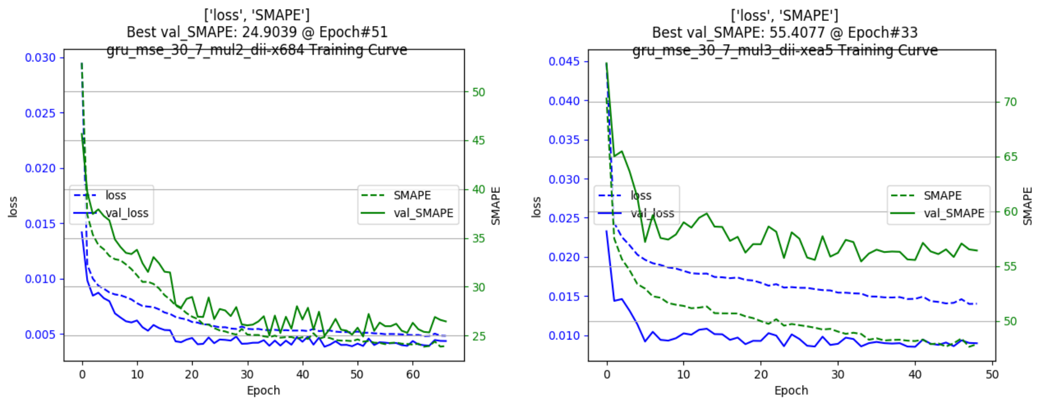


*Figure 42 Training Curve for 1) Pickpocketing on streets, 2) Theft of mopeds & bicycles*

Above images show univariate prediction analysis of all the four different types of crime incidences. First plot shows a very high level of overfitting and hence the model converges on a non-global minimum. The second plot shows an improved performance by the model on the second incidence type with satisfactory generalization and SMAPE value.  The univariate model performed best on the third model with good generalization and lowest SMAPE value of 24.9. But, the predictive capability of model takes a big hit for fourth incidence type and it has highest SMAPE value of 55.40.

Let's assume that a model can predict all four different types of incidences. We will train the model on the parameters and try to analyse the performance.

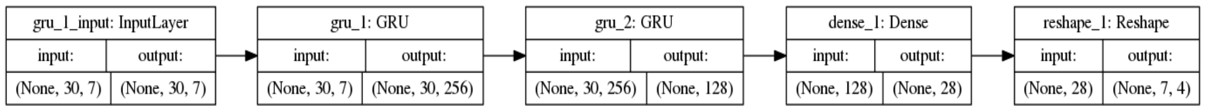


*Figure 43 Tensorboard visualization of Multivariate GRU Network with date injection*

The architecture of the model will be identical to the univariate GRU model with date injection with an addition of three extra columns of data for multivariate input. The output layer will provide an output of shape (7 x 4), where each column represents an incidence type and each row represents a day's prediction.

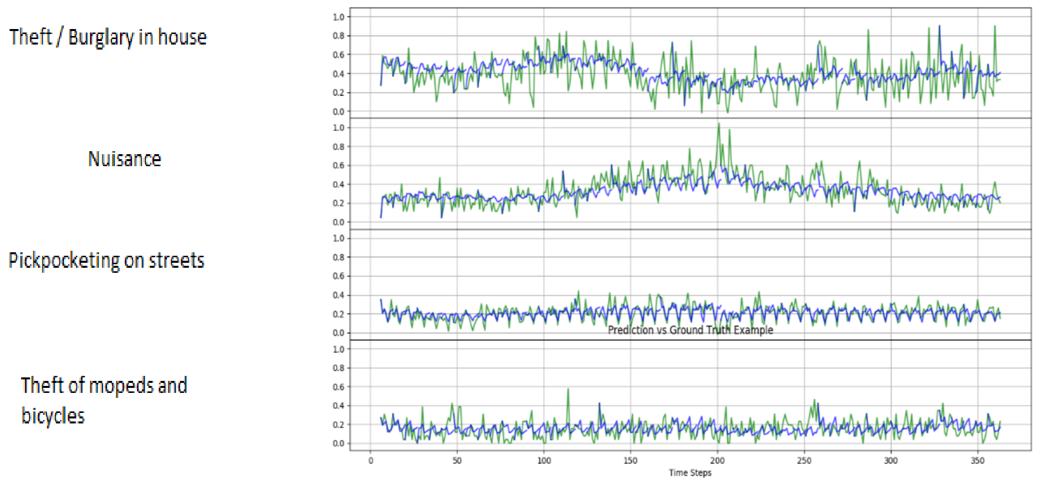


*Figure 44 Multivariate prediction plot GRU Network with date injection on test data*

Above plot explains the predictive capability of model on all the four types of crime incidences. As it is clear from the graph for the following incidence types:

First Incidence Type (Theft / Burglary in house): This attribute's test data distribution shows that the variation in the data is quite irregular and hence the prediction is unable to catch the extremities of the data. Hence, the model tries to keep the prediction in mean area to avoid going haywire.

Second Incidence Type (Nuisance): This attribute's prediction on the test data shows the irregular patterns in test data has led to the near mean (middle) prediction. Generally, models become very conservative in their learning from the data when the pattern is very noisy.

Third Incidence Type (Pickpocketing on streets): This attribute is by far the most consistent in all the four incidence types. This is also shown in the prediction where model is expecting a certain pattern in the test data and hence, all the dips are being predicted satisfactorily.

Fourth Incidence Type (Theft of mopeds or bicycles on street): This attribute has a varied pattern and the model is unable to read and predict the patterns effectively. Evidently, the model can read the direction of the prediction, the degree of prediction to highs and lows are fairly missed. One of the main reasons for this behaviour can be the amount of data used to train the model because the training data is not big enough for model to read and learn the irregular patterns and high and low extremities of incidences.

Since models are built and tested on dataset of size (2355 x 4), since the model doesn't have enough data, it is understandable that the multivariate property of model could not be utilized well.

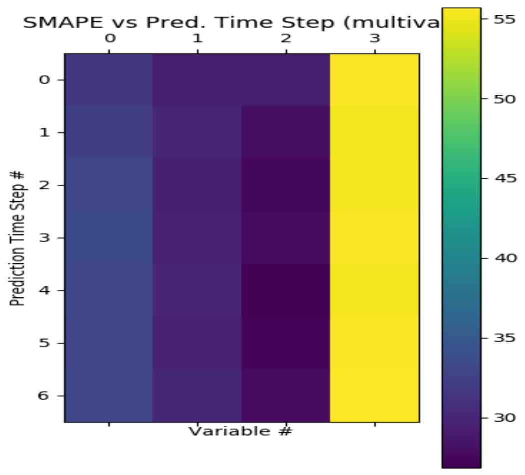


*Figure 45 Heat Map to visualize the effect on SMAPE value by increasing prediction steps on different incidence types*

The heat map can explain the variation in prediction behaviour in relation with number of variables and number of days of prediction. Clearly, prediction of Theft of mopeds or bicycles on street is the worst in the multivariate model. The best prediction is the Pickpocketing on streets incidence type.

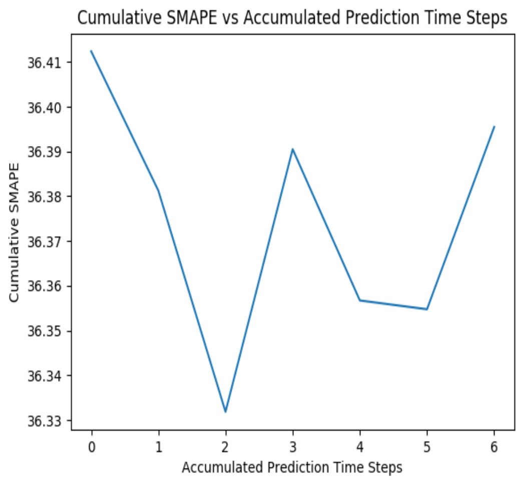


*Figure 46 Cumulative SMAPE error measure on increasing number of prediction steps*

Also, we have tried to discover the cumulative SMAPE on increasing prediction days and it is surprising to uncover that predictive power is very random across various number of days with the lowest error till 3rd day prediction. This finding needs to be explored further to understand the reason for this random behaviour in prediction.

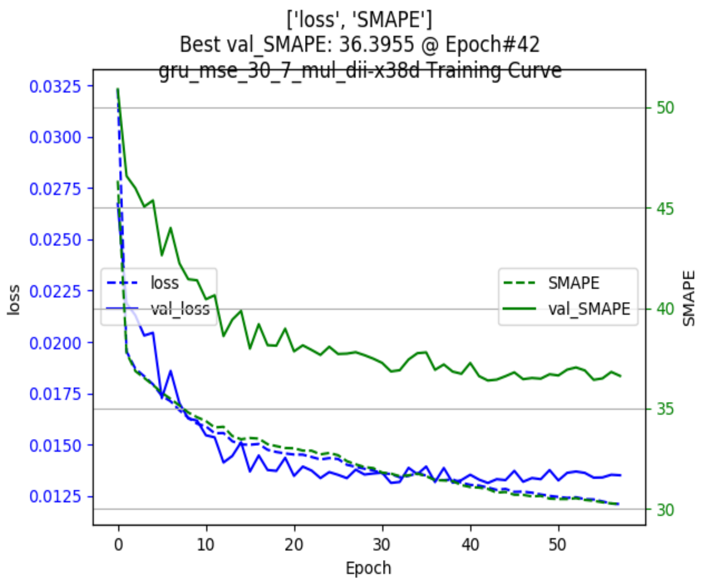


*Figure 47 Multivariate GRU Networks with date injection model loss and SMAPE curve on training and test data over increasing number of epochs*

We can say few things from the above plot of loss and SMAPE on the train and test data over different number of epochs. Firstly, the error percentage on the test data converged prematurely while the model fitted well on the training dataset. Also, the loss function

(MSE) on the test data converged very early while the model started overfitting on the training data. Though, the model reached its least SMAPE value of 36.39% and it reached its threshold value of early stopping.

*Table 2 Comparison of error percentage for univariate and multivariate GRU Network models*

| | **Theft & Burglary** | **Nuisance** | **Pickpocketing on Streets** | **Theft of mopeds & Bicycles** |
|---|---|---|---|---|
| **SMAPE for Univariate model** | 32.07 | 29.73 | 24.90 | 55.41 |
| **SMAPE for Multivariate Model** | 31.55 | 29.31 | 29.28 | 55.30 |

Above table demonstrates the predictive capability of univariate and multivariate models on the incidence types. We can see that multivariate model deliver better result than Univariate models on first two incidence types, but it fails to outperform univariate models on remaining two incidence types. One of the reasons for this result may be the number of training datapoints for the model because multivariate model will need more data for convergence of its variables. But, its interesting to note that the multivariate model was able to perform like, and sometimes outperform, separate univariate models and it was able to read the individual patterns of its variables.

We also tried to run the multivariate versions of various neural network alternatives. This experiment was aimed at finding out the effect of multivariate property on the models.

*Table 3 Comparison of Error Percentage for different Neural Network Alternatives*

| **Multivariate Models** | **1FCN** | **FCN** | **CNN** | **LSTM** | **GRU** |
|---|---|---|---|---|---|
| **Test SMAPE** | 37.51 | 36.93 | 36.47 | 37.60 | 36.39 |

Above result showed that the models behaved differently with different models and it behaved the best with the GRU model and it performed the worst with the LSTM network. Overall, the models performed worse than the univariate models. One of the reasons for this result is the lack of enough data for models to learn the patterns in all the incidence types and hence it led to non-convergence of models for all the attributes.

### 5.4.8 Comparison between different neural network models:

After completing the experiments on different types of models, we will try to analyse and visualize their performance on train and test dataset. This would give a clear picture on their predictive power, their generalization capability and their overfitting.

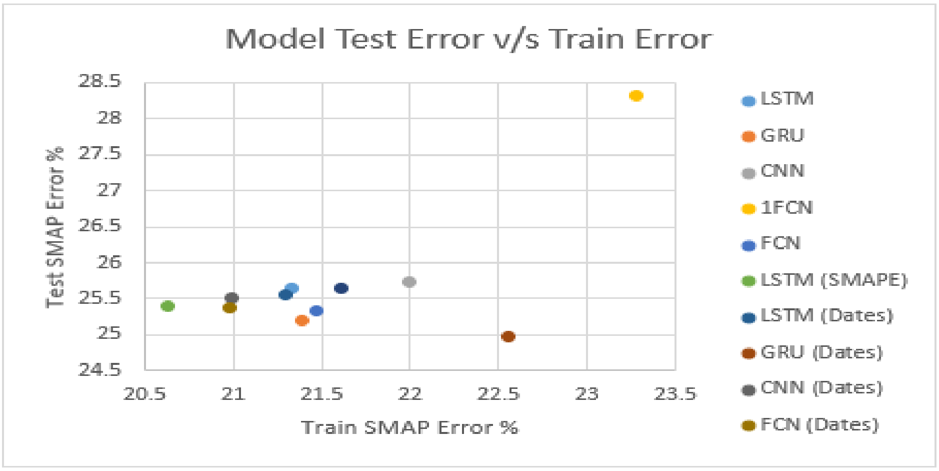


*Figure 48 Plot to demonstrate the generalization capability of model*

Above picture shows a test SMAPE vs train SMAPE plot of different models. We can see that single layer linear neural network has the worst performance with highest test and train error percentage. This can be a result of its linear property and its less complexity. Surprisingly, same model performed much better with (train, test) percentage as (21.6%, 25.6%). This improvement is because the model learns the data patterns based on date, month and day of the week, hence giving it more information.

In general, the models with date injection demonstrate better performance and better generalization capability than their conventional alternatives. The models were not able to showcase a significant improvement over other and they are in a common zone of train vs test plot. Nevertheless, GRU shows relatively high level of generalization on test data (24.9%) and it shows that GRU is the best model to use in the current scenario.

# 6.Conclusion and future work

The main objectives of this project were: to analyse a crime incidence dataset and describe various patterns in the data; and compare different predictive models for forecasting incidents on crimes into the future. Both these objectives are part of Sentient's efforts to provide temporal capability to their existing software package, DataDetective.

An exhaustive comparative study is performed on the given data with various statistical and machine learning (neural network) approaches. Through experiments and comparative analysis, we were able to demonstrate that Gated recurrent neural networks (GRUs) with date injection were the ideal models for this dataset. This model had the highest prediction accuracy and the lowest gap between train and test errors, suggesting good generalization capability. Another important insight derived from the project was that the statistical models were significantly outperformed by neural networks. This could be attributed to the linear property of the statistical models used, as they were unable to capture the non-linear patterns of the data.

Among the neural networks, recurrent neural networks like GRU and LSTM exhibited advantage in the field of time series analysis when compared to conventional feed-forward neural networks. A relatively novel exploration of 1D CNN models was also performed for time-series forecasting, and this yielded promising results. Additionally, the low computational and memory costs of CNNs make them a rather attractive choice for the task, in comparison with heavy statistical models and relatively slower recurrent models.

The performance analysis of different types of predictive models showed that Gated recurrent Units (GRU) have performed better than rest of the models for univariate and multivariate models. This shows that the objective of the internship was fulfilled, and the internship was successful.

This internship opens door to a series of future work:

- An accumulation of more data into the system could help in exploring the true potential of the recurrent neural networks and 1D convolutional neural networks. Also, this would help exhibit real difference in performance.
- We can collect some geo-spatial data like weather and region. This would help in aggregating data based on the region. The final goal of temporal analysis in *DataDetective* is to help clients in forecasting the number and types of incidences that could happen in next week in different regions in Amsterdam. This geospatial-temporal analysis will help clients in distributing their workforce to the areas of high crime prediction. Ultimately, this would help in better utilization of their workforce.
- *DataDetective* can also be extended in other spheres of business-like supermarket sales to predict the number of customers and types of customers expected in the next week.

# 7. Bibliography

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

Amogh Gudi, Recognizing Semantic Features in Faces using Deep Learning, (2016)

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

Box, G. & Jenkins, G. (1970), Time Series Analysis, Holden-Day, San Francisco.

Chen, H., Chung, W., Xu, J.J., Wang, G., Qin, Y. Chay, M. (2004). Crime data mining: A general framework and some examples

Chen, P., Yuan, H. & Shu, X. (2008). Forecasting crime using the ARIMA model.

Christiane Lemke, Combinations of Time Series Forecasts:When and Why Are They Benecial?, (2010)

Christopher Olah, Understanding LSTM Networks, (2015)

David Gerbing, Time series components, (2016)

Diederik P. Kingma, Jimmy Lei Ba. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION, (2017) Elkblom, P. (1988). *Getting the best out of crime analysis* (Heal, K, Ed.)

Everette S. Gardner, Jr. (2005) Exponential Smoothing: The State of art – Part II
Oatley, G.C., Zeleznikow, J., Ewart, B.W. (2005). Matching and predicting crimes.

Furtado, V., Melo, A., Coelho, A., & Menezes, R. (2007). A crime simulation model based on social networks and swarm intelligence.

Glorot Xavier, Bordes, Bengio. Deep sparse rectifier neural networks, (2011)

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

Granat, R. Capabilities of statistical models for geophysical sensor arrays, (2009)

Granger, C.W.J. and Ramanathan, R. (1984) Improved Methods of Combining Forecasts. Journal of Forecasting, 3, 197-204.

Greenberg, D.F. (2001). Time series analysis of crime rates. *Journal of Quantitative Criminology, 17(4),* 291-327.

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

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

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

Juergen Schmidhuber, Deep Learning in Neural Networks: An Overview, 2014

Kim. Y. (2008). Space-time measures of crime diffusion*. In Artificial Crime Analysis Systems VIII*.

Kovachev, S., Reichert, P., & Speck, H. (2008). Crimeblips: Web based framework for crime incident analysis and visualization.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder– decoder for statistical machine translation. In Proc. EMNLP, pages 1724–1734. ACL, 2014.

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

Nguyen, X.L. (2006). *Anomaly and sequential detection with time series data*

Oleg Ostashchuk, Time Series Data Prediction and Analysis, (2017)

Razvan Pascanu, Tomas Mikolov, Yoshua Bengio. On the difficulty of training Recurrent Neural Networks, (2013)

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

V. Kondratenko, Yu. A Kuperin, Using Recurrent Neural Networks To Forecasting of Forex, Condensed Matter, Statistical Finance, 2003

Yufeng Yu, Yuelong Zhu, Shijin Li, and Dingsheng Wan. Time Series Outlier Detection Based on Sliding Window Prediction. (2014)

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

Zhiyuan Tang, Ying Shi, Dong Wang, Yang Feng, and Shiyue Zhang. Memory visualization for gated recurrent neural networks in speech recognition. In Proc. ICASSP, pages 2736–2740. IEEE, 2017.