CEGEG076: Spatio-Temporal Analysis and Data Mining

SVM: Jack Philpott

ARIMA: Katie McNamara

STARIMA: Cory Williams

KNN: Alex Bridger

Word Count: 7409

Traffic Forecasting

Table of Contents

1 - Introduction and data description

1.1 - Introduction

Intelligent Transport Systems (ITS) are integral in the development of smart cities. There are huge volumes of data associated with transportation networks in urban environments which can be utilised to forecast future traffic conditions. This is vital for alleviating congestion, benefitting urban planners and commuters alike. Many predictive models exist, but this report will focus on a ARIMA, STARIMA, Support Vector Machines (SVMs) and k-Nearest Neighbours (KNN).

Following a brief literature and data description, an exploratory Spatio-temporal data analysis will be carried out. Next, four individual forecasting methods will be outlined, and their respective results presented. The performance of each model will be compared, and the most appropriate method outlined in the succeeding sections.

1.2 - Literature Review

Traffic forecasting and ITS have been a focus of research since the 1980's. The empirical forecasting methods employed in these papers can be classified as parametric and non-parametric (Zhang and Liu, 2009)

Auto-Regressive Integrated Moving Average (ARIMA), first developed by Box and Jenkins (1976) has become the dominant family of models in the Parametric traffic forecasting literature (Yu et al., 2015). Many extensions of ARIMA were later developed, such as Space-time ARIMA (STARIMA) developed by Pfeifer and Deutsch (1979). STARIMA differs from ARIMA as it accounts for spatial influences by using a spatial weight matrix (Haworth, 2014). Other extensions include Seasonal ARIMA (SARIMA), as used in Kumar and Vanajakshi's 2015 study of traffic flow in India.

Similarly, multiple data driven, non-parametric approaches have been developed. Examples include Support Vector Machine's (Liu et al., 2010), k-Nearest Neighbour (Zheng and Su, 2014), Artificial Neural Networks (ANN's) (Habtie, Abraham and Midesko, 2015), Baynesian networks (Castillo et al., 2008), and clustering (Xia et al., 2012). Furthermore, missing data can occur frequently during traffic monitoring (Bae et al., 2018), significantly decreasing the accuracy of popular forecasting methods such as ARIMA (Haworth and Cheng, 2012). To combat this, Howarth and Cheng developed a non-parametric Spatio-temporal kernel regression approach to dealing with this based on a dataset similar to that described in the forthcoming paragraphs.

Recently, several reviews have been published scrutinising the literature associated with traffic forecasting. For more extensive literature reviews, see Ermagun and Levinson (2016), Poonia, Jain and Kumar (2018) and Vlahogiann, Karlaftis and Golias (2018).

1.3 - Data Description

This report utilises travel time (TT) data supplied by Transport for London (TfL). As part of the London Congestion Analysis Project (LCAP), the data was collected across London's road network using Automatic Number Plate Recognition (ANPR) cameras. As of 2014, TfL were operating 1,517 cameras. Primarily, the cameras were installed to implement methods of reducing traffic flow in central London, with 646 and 342 cameras operating to enforce the London Congestion Charging Zone (CCZ) and Low Emission Zone (LEZ) respectively. Furthermore, 498 cameras are used for monitoring purposes, whilst the remaining 31 cameras are used to enforce speed limits. (Hurt, 2014; Haworth, 2014).

The TT data is captured by ANPR cameras operating in pairs. For every car which passes a camera, the registration and time are recorded. Once a car passes both cameras, the registrations match and the TT can be determined by subtracting the time *(t_1)* at which the car passed the first camera *(l_1)* from the time *(t2)* the car passed the second camera *(l_2)*. This can be visualised in figure 1.1.

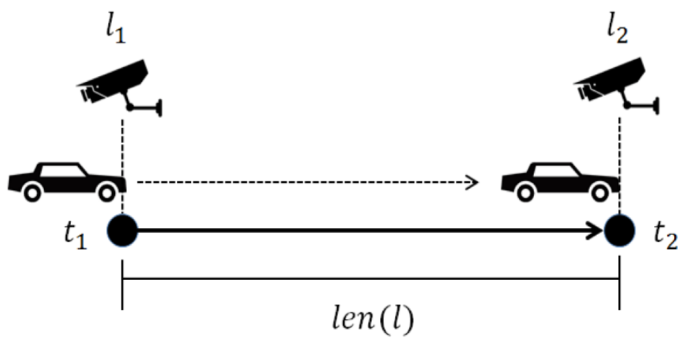


Figure 1.1 – How travel time data is collected using ANPR cameras.

Source: Haworth (2014).

The TT data has undergone some noteworthy pre-processing. Initially, the TT data would have been largely influenced by the length of each individual road link. To exclude such influence, the TT data was standardised into Unit Journey Time (UJT) which is in seconds per meter. This was calculated by dividing the TT *(t_2-t_1)* by *len(l)*, the length of the road link (Halim, 2015).

The dataset used in this study consists of 256 road links as defined by their respective ANPR camera pairs. Individual TT data are aggregated at 5-minute intervals between 6am and 9pm. Therefore, this study will analyse 180 observations per day for thirty consecutive days (01/01/2011 – 30/01/2011). Figure 1.2 presents the subset of 19 links chosen for this analysis.

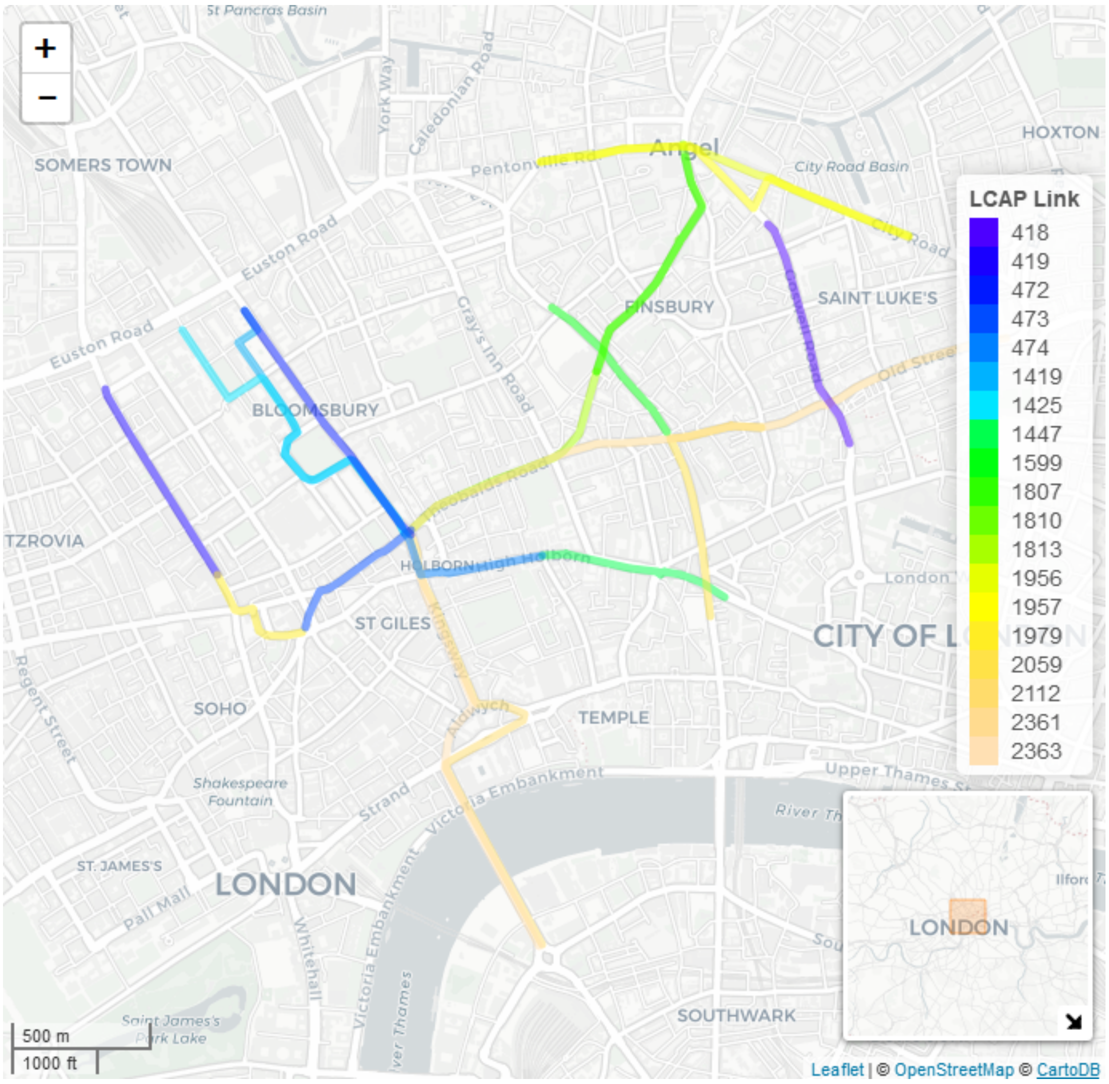


Figure 1.2 – Subset of 19 LCAP links chosen for this analysis.

2 - Exploratory Data Analysis

Exploratory data analysis through visualisation was initially carried out on the dataset to give an indication of perceived trends or cyclic patterns throughout the timeseries. This method is particularly effective when applied to traffic analysis (Cheng et al, 2013).

The most simplistic visualisation of the data can be seen in figure 2.1. This plot displays the extent the divergence of the data distribution away from the theoretical norm (red line), which occurs most significantly towards the tails of the data. The findings of this simplistic spatial analysis warranted further investigation.

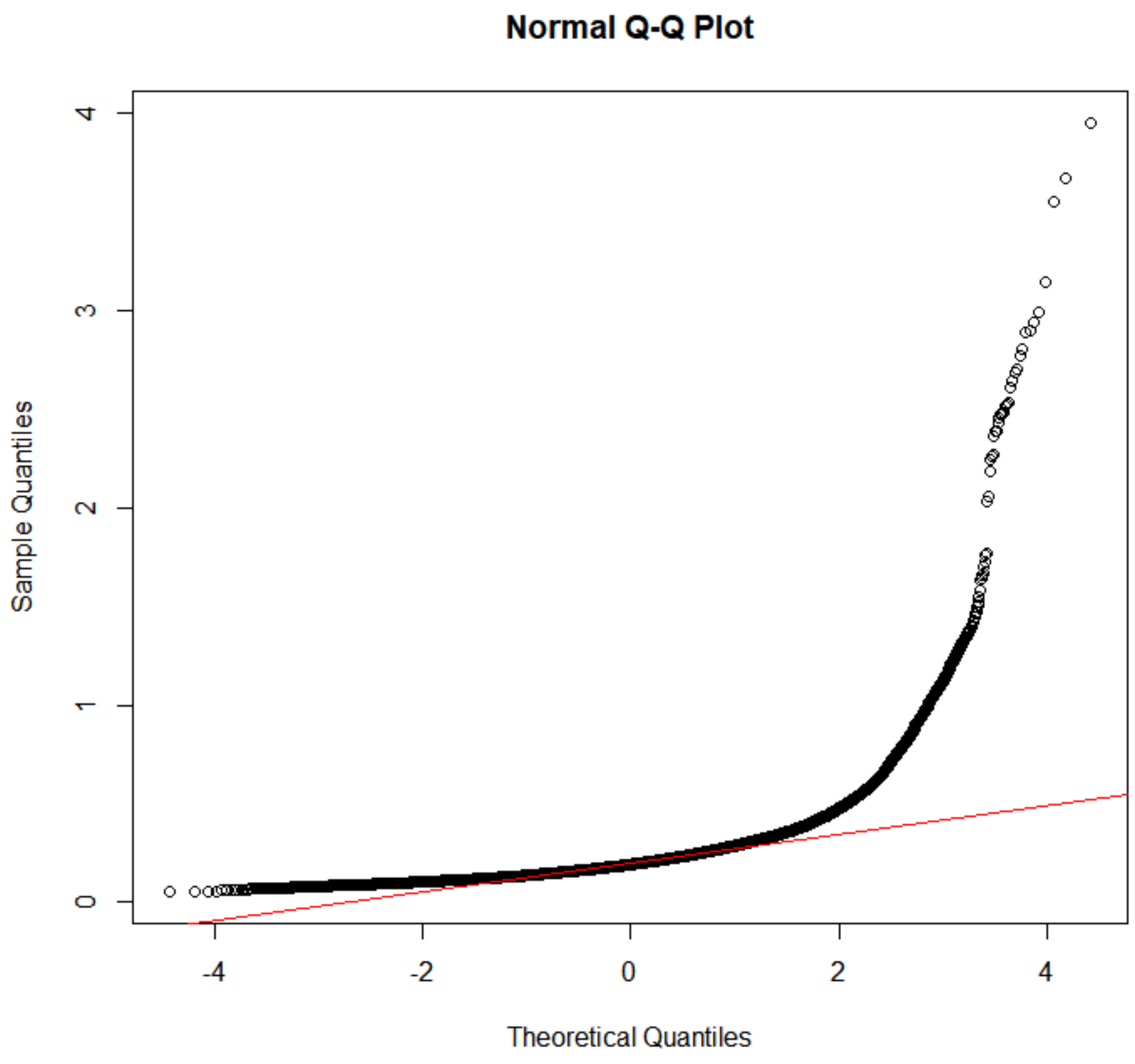


Figure 2.1 – QQ plot of the UJT data.

A more in-depth analysis of the time series can be seen in Figure 2.2, which presents the mean traffic speed across the month. The increasing gradient of the red line confirms a slight increase in traffic speed throughout the month. Furthermore, it is also very clear the dataset exhibits cyclic patterns. However, the inconsistency of the temporal variance throughout the dataset confirms heteroskedasticity, which is commonly associated with traffic data due to the increased variance during peak times (Haworth, 2014; Fosgerau and Fukuda, 2012). Particularly of interest is the spike in speed to 0.67 seconds/metre occurring at timestep 2924, which equates to 9.35am,17-01-2011.

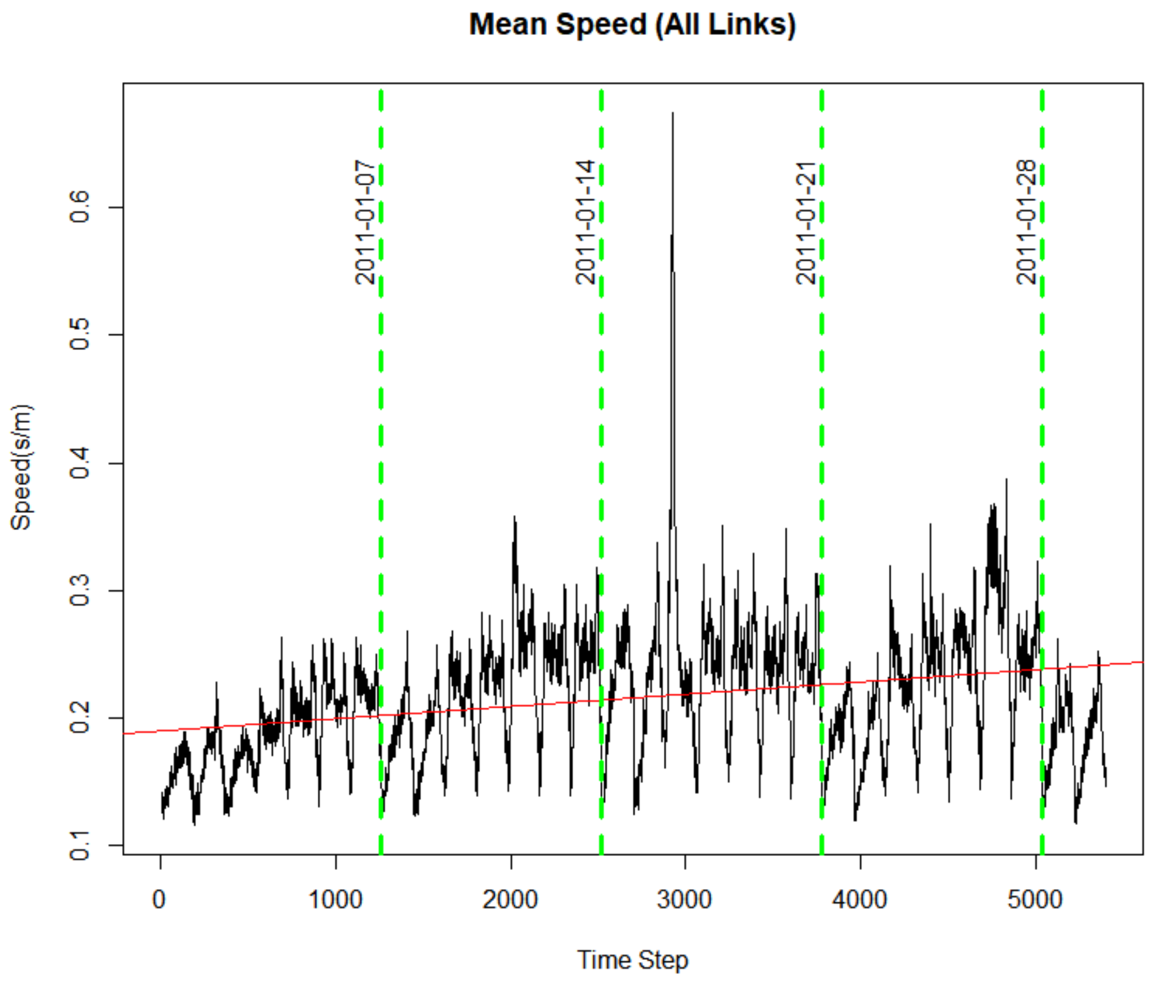


Figure 2.2 – Mean traffic speeds for all LCAP links with weekly intervals marked.

Further analysis was carried out to determine how the dataset varies throughout the day. Figure 2.3 displays how mean traffic speed varies for each LCAP link throughout the day. Although this is a relative coarse visualisation tactic, it is evident that mean speeds tend to decrease or plateau during peak hours of 7-10am and 4-7pm (TfL, 2015). It should be noted that there some links exhibit large variation during these peak times also. This may be due to the influence of the weekend traffic data, as the lighter levels of traffic will generate less congestion during these weekday peak times. Interestingly, almost every link shows a decrease in mean speed from 7-9pm, which was not expected due to lighter traffic and the resultant decrease in congestion. (TfL, 2009).

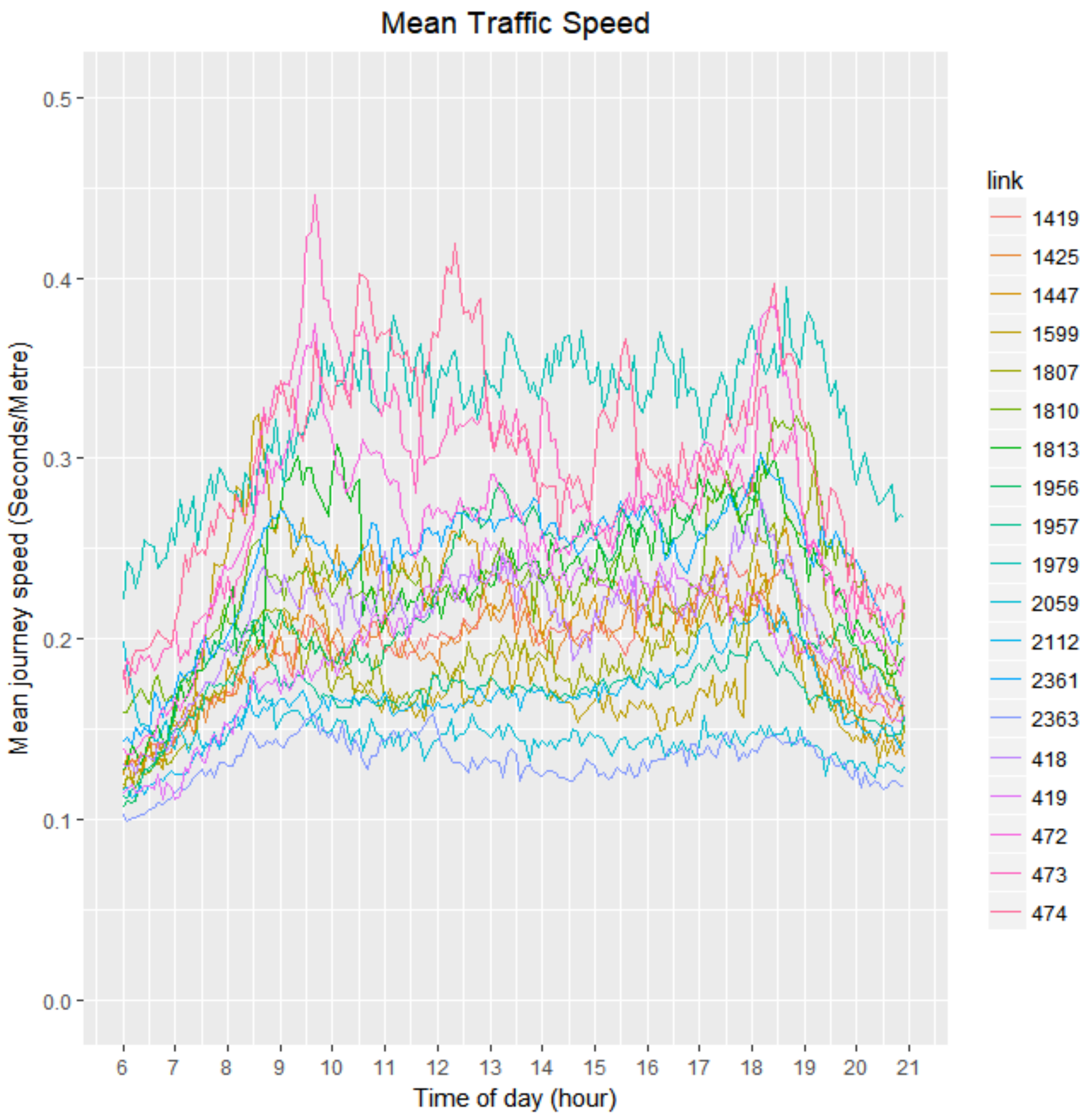


Figure 2.3 – Mean traffic speeds for all LCAP links across the day.

The Space-Time Autocorrelation Function (STACF), developed in the STARIMA package at University College London, calculates the correlation between observations separated in both time and space. Figure 2.4 presents the STACF for 1260 time steps (5 days). A seasonal component is evident within the dataset using STACF, yet less than considering only temporal data alone. The cyclic patterns occur at lags of daily intervals (~180).

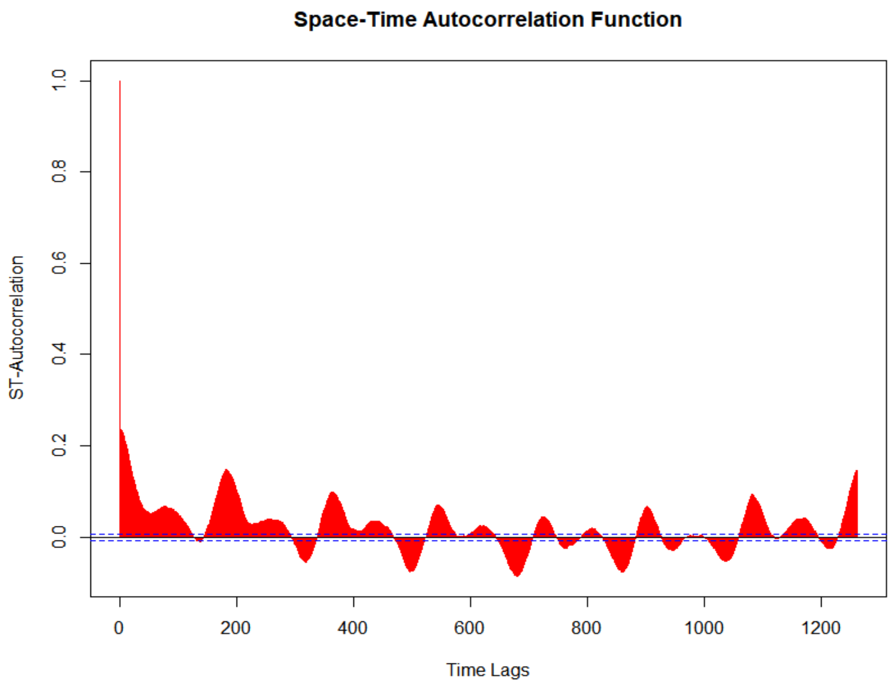


Figure 2.4 – STACF plot for 5 days (1260 time lags).

For a Spatio-temporal analysis across the full dataset of 30 days (5400 time steps), figure 2.5 is presented below. Reaffirming the observations made with figure 2.4, positive STACF values for at lags separated by a period of ~180. Furthermore, the 5400 time steps also reveal positive lags of 1260, 2520, 3780 and 5040 which correspond to the weekly intervals labelled in figure 2.2.

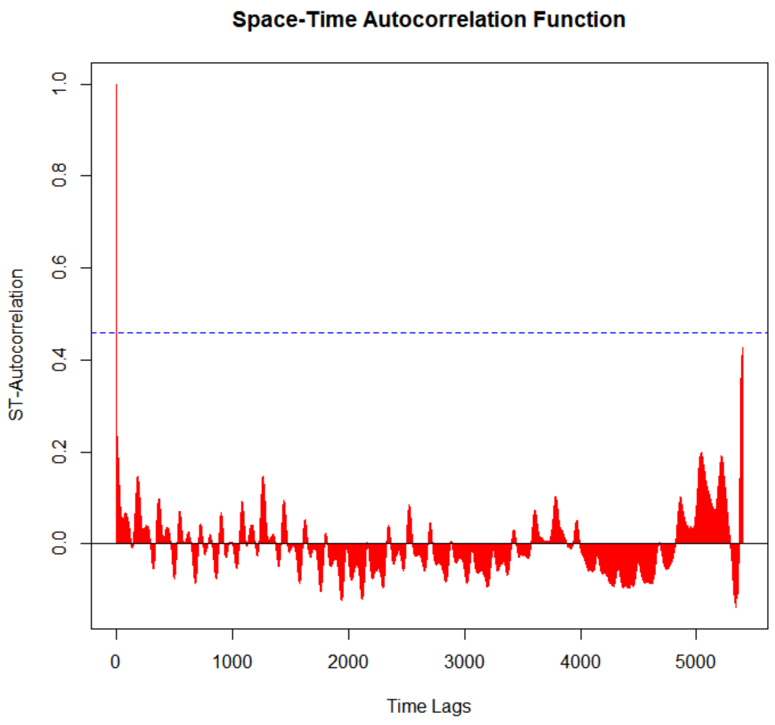


Figure 2.5– STACF plot for 5 days (1260 time lags).

3 – Individual model methodology and results

3.1 - ARIMA

A commonly used parametric statistical model is the Autoregressive Integrated Moving Average (ARIMA) model. This model has been used for traffic forecasting since the late 1970s and is still used in recent studies (Ahmed and Cook, 1979; Guo and Smith, 2007). ARIMA is a linear statistical time series model that is made up of three components: the autoregressive order (AR or p), the integration order (I or d), which is the number of differences required to transform the time series into a stationary dataset and the moving average order (MA or q). When a model has non-zero p, d and q values and has a non-seasonal component it is known as ARIMA (p,d,q). However, time series can also have an additional seasonal component and this is denoted as ARIMA (p,d,q) (P,D,Q)S. ARIMA has been widely used as a model in time series forecasting and it is often used in analysis to compare against other time series forecasting models (Haworth, 2014), therefore, it was deemed appropriate for this study.

The following 6 step methodology, shown in Figure 3.1.1, was used to implement the ARIMA model onto the time series traffic data:

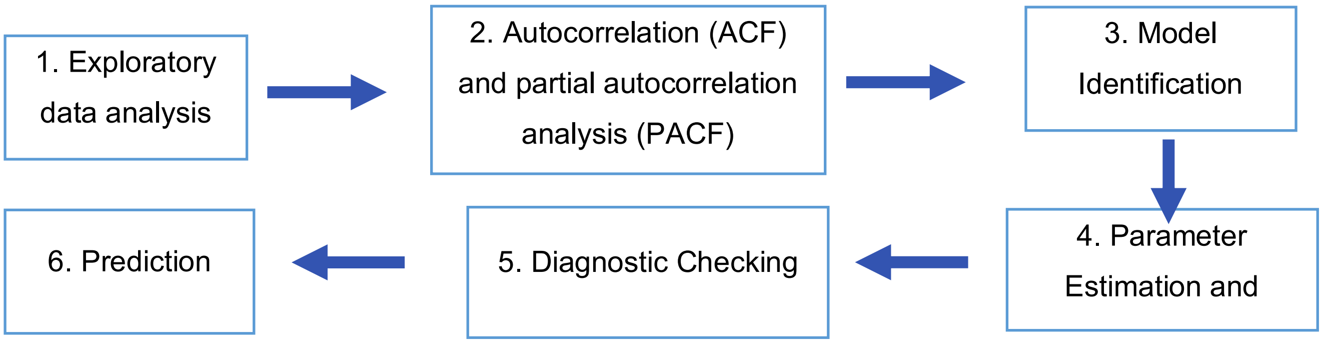

1. Exploratory data analysis
2. Autocorrelation (ACF) and partial autocorrelation analysis (PACF)
3. Model Identification
4. Parameter Estimation and
5. Diagnostic Checking
6. Prediction

Figure 3.1.1 – The 6 stages of the ARMIA model methodology

Exploratory data analysis

Stage 1 is necessary to help explore the time series and determine the types of patterns that exist in the data, notably any trends in the data and evident cyclic patterns. If the time series contains either of these types of patterns it is necessary to preprocess the data to remove them as the ARIMA model assumes time series stationarity. Plotting the test data (Figure 3.1.2) revealed evidence of a potential cyclic pattern.

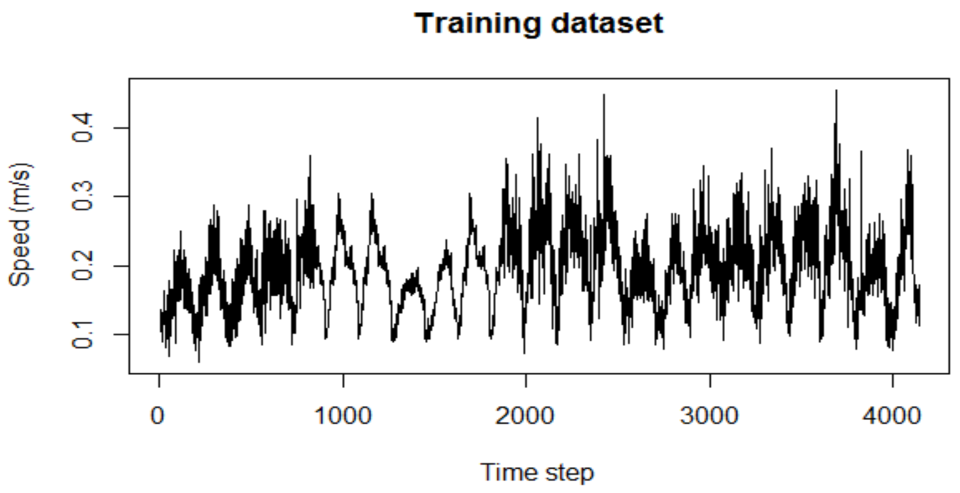


Figure 3.1.2 – A plot showing the training dataset of the time series that will be used to fit the ARMIA model

To explore this further, the first three lag plots were plotted to provide a visual impression of the series temporal dependency. As shown in Figure 3.1.3 all three scatter plots, which represent each lag, show that the points are about a straight line in a positive direction, meaning that the dependence between consecutive observations are linear.

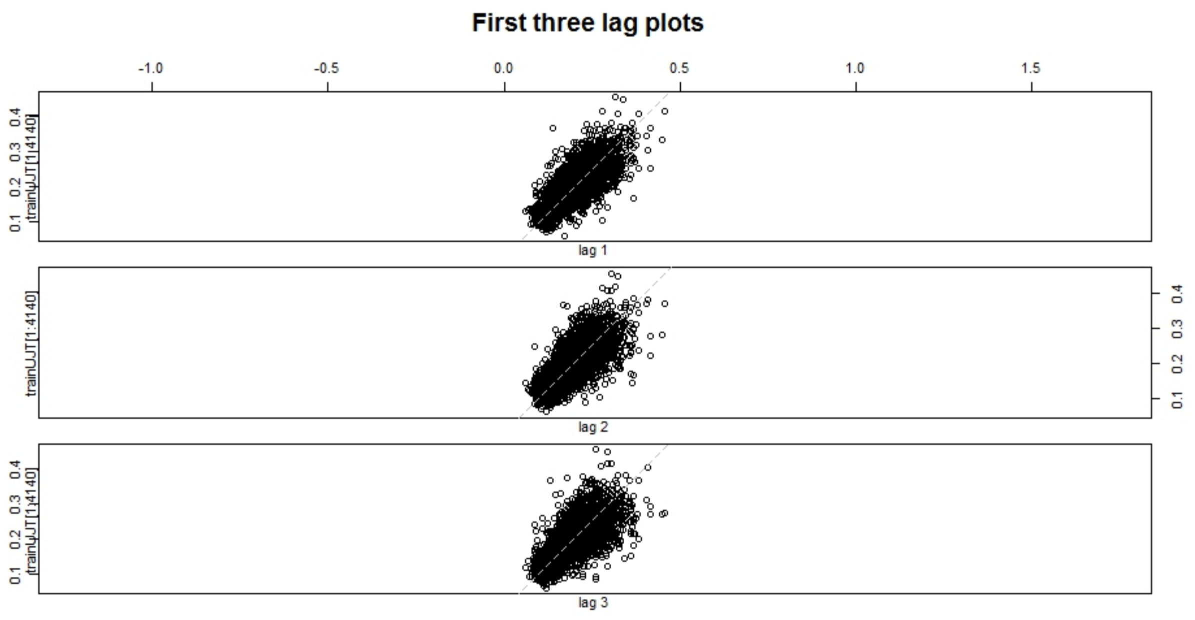


Figure 3.1.3 – A plot showing the first three lags of the training dataset

This shows that the time series is strongly non-random and non-stationary.

ACF and PACF

The second step of the methodology involves using temporal autocorrelation and partial autocorrelation analysis. This step is an integral part of the methodology as is helps decide what parameters or orders to use for the ARIMA model. For example, the ACF is used to give the autocorrelations at all the possible lags this can help determine the MA order of the ARMIA model. The AFC result without the differencing can be seen in Figure 3.1.4 and further supports the observation that the time series is non-stationary.

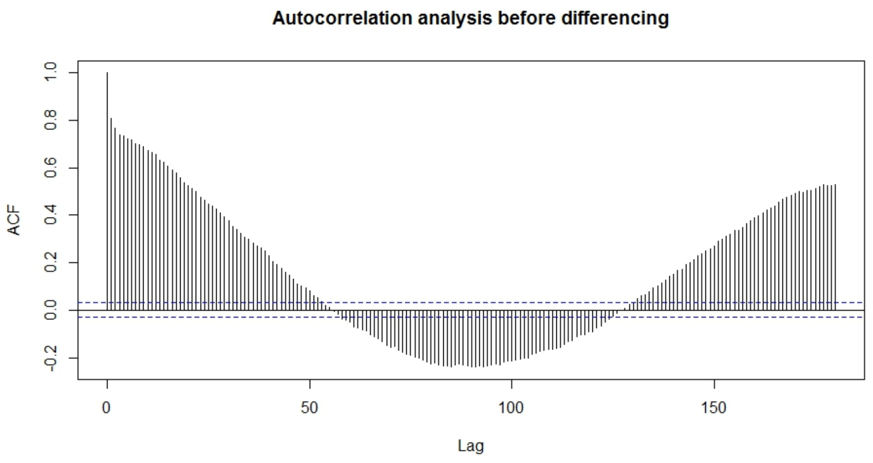


Figure 3.1.4 – The result of using the ACF without differencing

From this observation it was concluded that seasonality differencing should be used as a method to transform the time series from non-stationary into stationary. Therefore, the seasonality differencing orders of 1, 180 and 1260 were tested. The values of 180 and 1260 did not transform the time series to stationary however, having a seasonal differencing order of 1 did effectively transform the time series and consequently was selected. As shown in figure 3.1.5 when the seasonality differencing order of 1 is applied to the ACF a rapid decline to zero is experienced, which is a typical characteristic of a stationary time series.

Furthermore, it can be observed that there is a very significant value at lag 1 and a slight significant value at lag 3 meaning the MA order could be 1 or 3 for the ARMIA model.

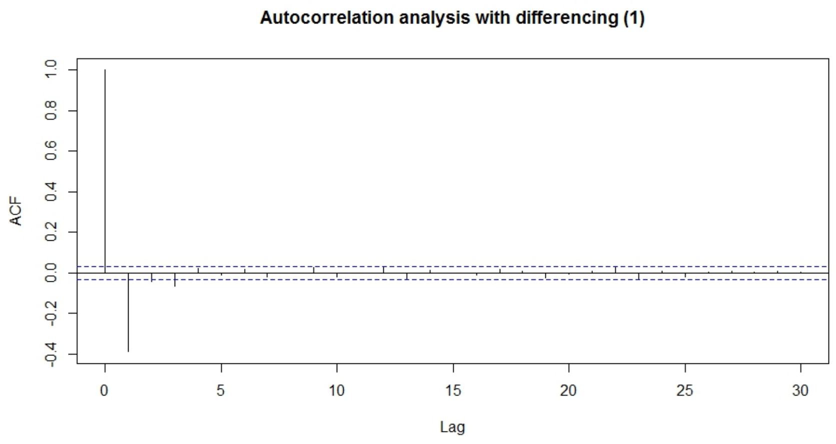


Figure 3.1.5 – The result of using the ACF with differing

Additionally, the partial autocorrelation function (PACF) measures the autocorrelation at specified lags to determine how much extra information each lag provides to a model. Therefore, this technique can help determine the AR order of the model by visualising the number of significant partial autocorrelations. Figure 3.1.6 shows the PAFC with the differencing applied and it is evident that there are significant values at lags 1, 2, and 3 therefore these values will all be trialled as the AR order the ARMIA model.

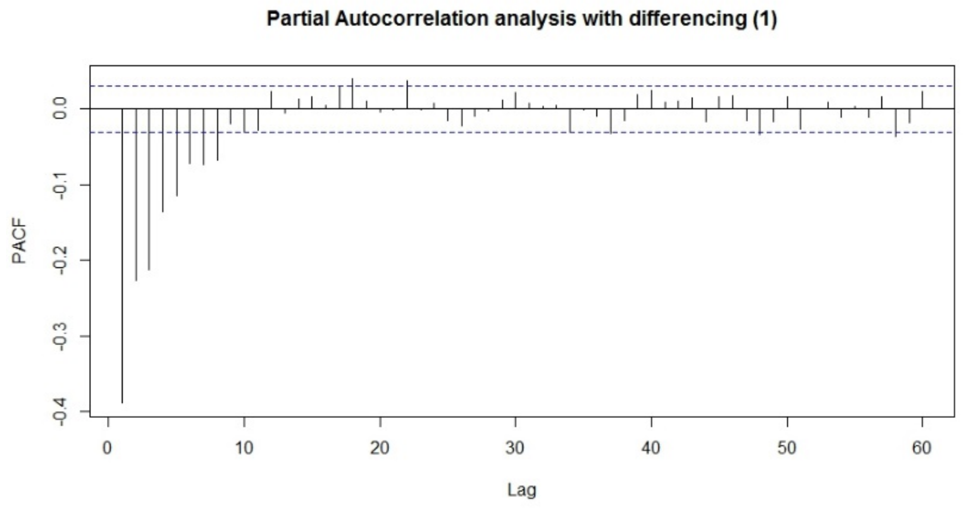


Figure 3.1.6 – The result of using the PACF with differencing

Model Identification

Using the integration order (d) and the estimate AR (p) and MA (q) orders established in steps 1 and 2 these values are put into the non-seasonal ARMIA (p,d,q) model. Therefore, step 3 is necessary to identify each candidate model for this analysis. For this study six candidate ARMIA models were trialled to find the best fit as shown in Table 3.1.1

Parameter estimation and fitting

Once the models order had been decided the function arima() from R is used to estimate the parameters. To fully test the capability of these models on an unseen dataset the time series was separated into a training and test dataset. The training set contained the first 23 days of the time series (4140 observations) and is used to fit the model, whereas the test dataset contained the remaining 7 days (1260 observations) and is used to test the model's predictive capabilities. To evaluate the model's performance the normalized root mean squared error (NRMSE) is used which is a widely used index of model performance and can be used to compare ARMIA to various other time series models. Table 3.1.1 shows the result for each model and their associated NRMSE values.

Table 3.1.1 – Each candidate ARMIA model and their associated NRMSE values and train time

Candidate model	AR order (p)	I order (d)	MA order (q)	NRMSE value	Train time (seconds)
1	1	1	1	0.5299442	1
2	2	1	1	0.5299367	2
3	3	1	1	0.5292949	2
4	1	1	3	0.5294266	1
6	2	1	3	0.5294292	1
7	3	1	3	0.5284914	2

The lower the NRMSE value the better the fit of the ARMIA model. Therefore, out of the candidate models it is evident that number 7 is the most suitable order of ARIMA model (ARIMA(3,1,3). However, it was decided to select the third candidate model, ARIMA(3,1,1), for this study due to further evaluation using the function auto.arima from the R package 'forecast'. This function automatically decides on the orders for the ARMIA model based on the time series that is input (Hyndman and Khandakar, 2007). This function to test all the possible parameters and selects the lowest Akaike Information Criterion (AIC), the small-size corrected version of Akaike Information Criterion (AICc) and Bayesian Information Criterion (BIC) values for the ARMIA model. The results from using the auto.arima function concluded that ARIMA(3,1,1) model was the most appropriate for the time series used in this study, which is the second lowest candidate model from the NRMSE results.

As well as the training and testing datasets described above two other training dataset were also selected to evaluate the influence of training duration on the selected candidate model and their resulting NERMSE values. Table 3.1.2 shows the result of the NRMSE values for ARIMA (3,1,1) using different lengths of training datasets, specifically 7 days (1260 observations) and 14 days (2520 observations).

Table 3.1.2 – Comparison of NRMSE values between ARIMA(3,1,1) and additional training datasets with the time taken to run each model

Number of days in training dataset	NRMSE values	Train time (seconds)
23 days	0.5292949	2
14 days	0.5299577	1
7 days	0.5412342	1

It is evident that as the training size increases so does the NRMSE values therefore the 23 day dataset was used as the training dataset to fit the model.

Diagnostic checking

Step 5 relates to the diagnostic checking which is also essential for model selection. The function tsdiag() was used for this step which produces three plots to test the normality of the residuals of the time series model. If the residuals are not random it is indicative that there is more information in the errors that has not been accounted for in the model, therefore it potentially should not be trusted. As showed in Figure 3.1.7, the ACF of the residuals is zero which means they are independently distributed and the p-values in the Ljung-Box statistic are all above 0.5 indicating non-significance.

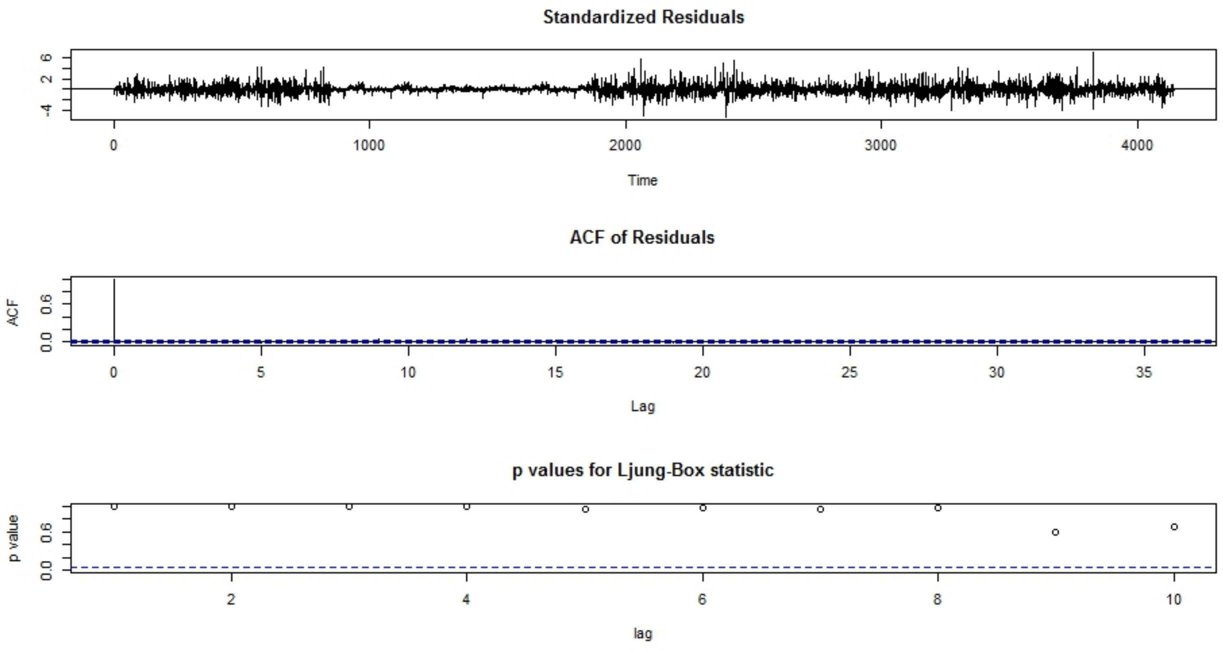


Figure 3.1.7 – A diagnostic plot of the fitted time series

Prediction

Once it has been established that the residuals in the time series model are random the last step of the methodology can occur. This involves using the predict() function (or the forecast() function from the ‘forecast’ library) to generate the models prediction of the future values of the time series. Due to previously splitting the data into a training and test dataset, the test dataset can be used to compare and evaluate the chosen ARIMA(3,1,1) candidate models prediction. Figure 3.1.8 shows that the predict function for the candidate ARMIA(3,1,1) model generates a prediction of 0.15 (red dotted line) when plotted against the test dataset.

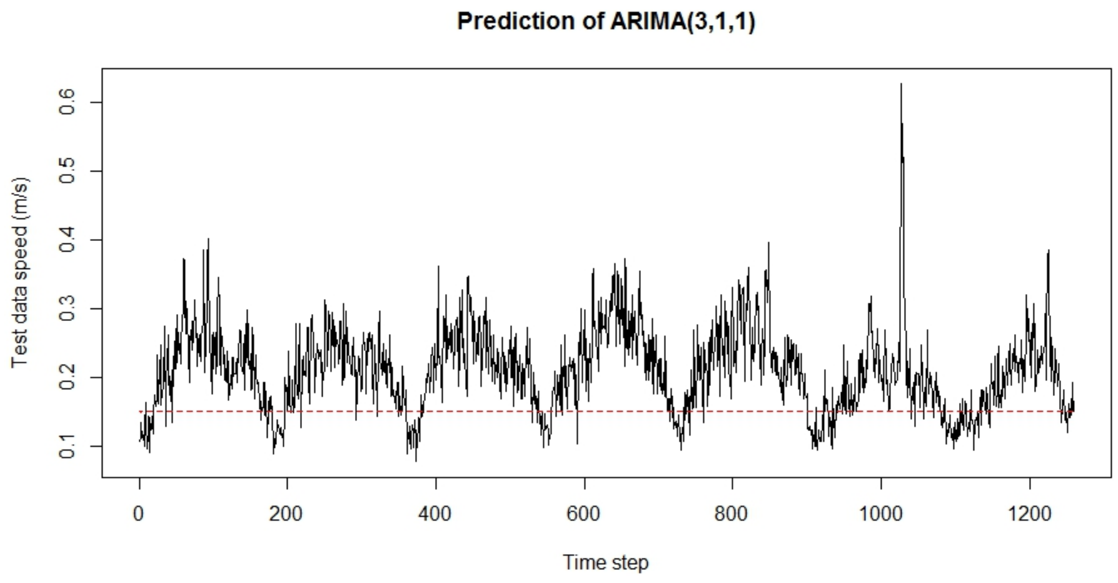


Figure 3.1.8 – shows the predicted values for the next 1260 observations (red dotted line) against the actual observations from the dataset.

The calculated NRMSE for this prediction is 1.319. This flat prediction which is not represented in the test dataset occurs because the time series has limited seasonality and therefore the ARIMA model has no seasonality component, so this is not accounted for in the prediction. Similarly, Figure 3.1.9 shows that the auto.arima function version has also predicted a relatively flat prediction which is identical to Figure 3.1.8 (approximately 0.15) for the same 1260 observations.

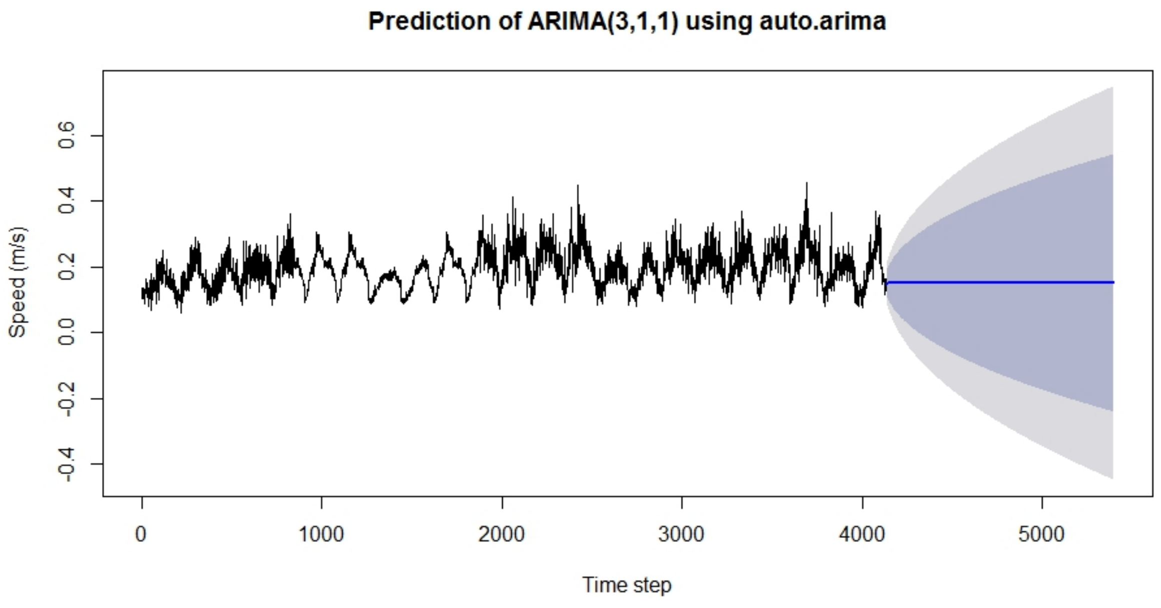


Figure 3.1.9 – Plot of the auto.arima prediction using the forecast() function

3.2 – STARIMA

Space-Time Autoregressive Moving Average (STARIMA) is a parametric space-time prediction model. It is an extension to the ARIMA model, including not only a time series but also a spatial weight matrix.

As described in section 1, the originally supplied dataset consisting of 256 LCAP links was subdivided into the 19 links being studied (see figure 1.2). Following this, Unit Journey Time (UJT) was subdivided into the first 23 days (1:4140 timesteps) and last 7 days (4141:5400 timesteps) for STARIMA model training and testing respectively.

Following this, the procedure required for STARIMA prediction was carried out. As detailed above, the method is similar to ARIMA forecasting, with two additional steps which are unique to STARIMA. Figure 3.2.1 presents an overview of this method with the additional steps highlighted in green and precedes a more detailed explanation.

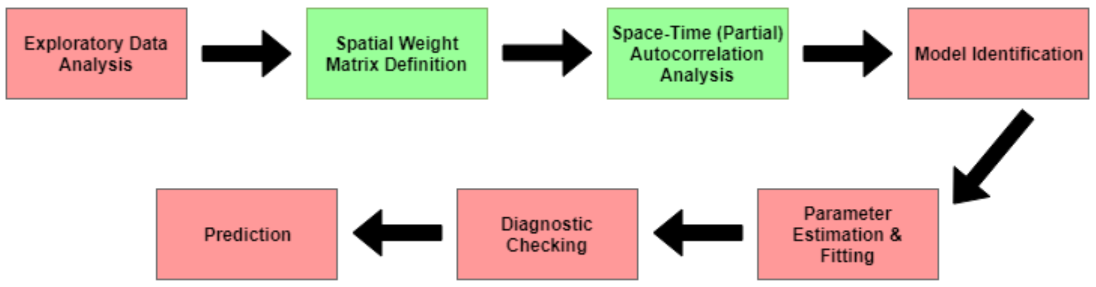

Exploratory Data Analysis
Spatial Weight Matrix Definition
Space-Time (Partial) Autocorrelation Analysis
Model Identification
Parameter Estimation & Fitting
Diagnostic Checking
Prediction

Figure 3.2.1 – The seven steps associated with space-time series analysis using STARIMA.

Exploratory Data Analysis

The results of this analysis have been presented in section 2 of this report. The main results indicate strong temporal cyclic patterns of a heteroskedastic nature. By utilising the STACF function of the STARIMA package, figures 2.4 and 2.5 confirm space-time seasonality through the presence of cyclical spikes occurring at lags ~180 and ~1260 and reoccurring with the same respective periods (e.g. 180, 360, 540…). The results from this section were used in the initial step of the STARIMA model calibration.

Spatial Weight Matrix Definition

The spatial weight matrix was part of the original data supplied following pre-processing to ensure compatibility with the STARIMA package. As with the UJT data, the adjacency matrix had to be subdivided to only include the relevant links. The spatial weight matrix is required to measure spatial autocorrelation in the network data, however, it could also be applied to grid or areal data (Haworth, 2014). The matrix encodes the spatial adjacency structure in a binary format, where a value of one indicates that two LCAP links are adjacent (neighbours) and a zero indicates they are not.

Space-Time (Partial) Autocorrelation Analysis

For this step in the STARIMA procedure, it is important to note several relevant plots have previously been presented in section 2 (exploratory analysis). Specifically, figure 2.4 and 2.5 which present the STACF of the data series at 1260 and 5400 lags respectively. The combination of both figures reveals a relatively strong seasonal component to the series, most visible at lags 180 and 1260 due to the cyclic pattern in unit journey times measured by LCAP. The same seasonality is visible in the figure 3.2.2, which presents the autocorrelations for the first 900 time lags (five days).

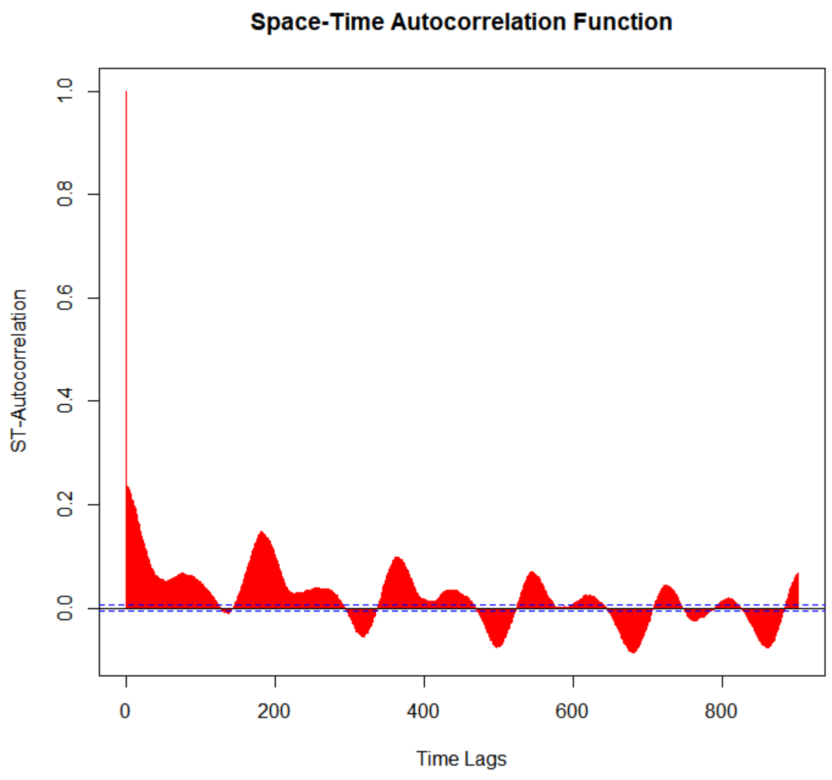


Figure 3.2.2 – STACF plot for five days (900 time lags).

Furthermore, the previously mentioned STACF plots can be used to determine whether the data series is stationary. As the series does not decay to zero following the first lag and the STARIMA package works on the assumption of space-time stationarity, the data needs to be transformed to maximise the accuracy of the prediction (Cheng, Haworth and Wang, 2011). If the data was stationary, a rapid decay would occur following the first lag, which is always one as default as it measures the space-time autocorrelation between the series and itself.

To achieve stationarity, seasonal differencing will be used as it is the most commonly used technique to transform a data series to stationary. The technique requires the subtraction of a series value from a previous value at a specific temporal lag, which results in the creation of a new time series, indicting the influence of the seasonality. Haworth (2014). The analysis of the STACF plots led to the conclusion that seasonality differencing to the order of 180 and 1260 time lags may be appropriate. However, as figure 3.2.3 shows, this did not transform the series to stationarity. Therefore, the commonly used approach of differencing to the order of one lag was used in accordance with Haworth (2013). The bottom plot of figure 3.2.3 shows this produces a stationary time series. This result infers the seasonality within the series is relatively week.

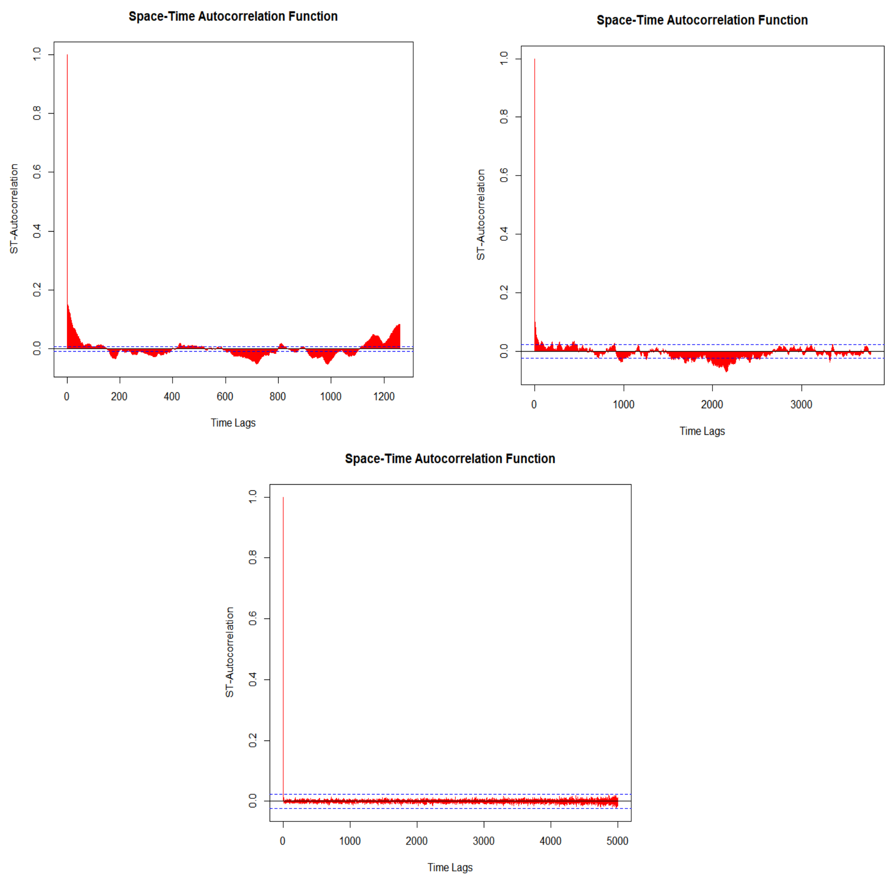


Figure 3.2.3 – STACF plots post differencing.

Top left: Differencing of 180

Top right: Differencing of 1260

Bottom: Differencing of 1

Following this, the Space-Time Partial Autocorrelation Function (STPACF) was used to measure the space-time autocorrelation present at nonzero lags. Figure 3.2.4 below only presents the STPACF following the differencing to the order of one.

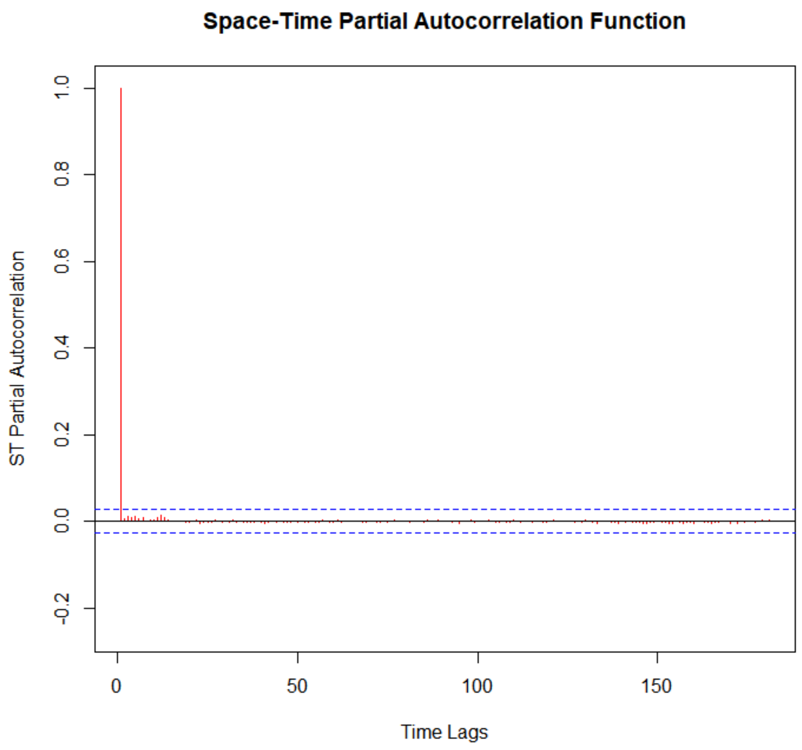


Figure 3.2.4 – STPACF plots post differencing.

Model Identification

The aim of this section is to identify the model represented as:

$$\mathbf{STARIMA}(\boldsymbol{p}, \boldsymbol{d}, \boldsymbol{q})$$

Where:

- *p* is the space-time autoregressive order
- *d* is the differencing order
- *q* is the space-time moving average order

The bottom plot of 3.2.3 determines *d* to be one. Further investigation into the same STACF plot and STPACF plot (figure 3.2.4) show that both cut off after the first time lag, suggesting a value of one to be applied respectively to the autoregressive order *(p)* and moving average order *(q)*. It should be noted that this is not definitively the best combination of prediction parameters, and alternative solutions will be explored in the subsequent section of the procedure. However, the preliminary candidate model is:

$$\mathbf{STARIMA}(\mathbf{1}, \mathbf{1}, \mathbf{1})$$

Parameter Estimation & Fitting

As the proposed model is not definitive, the prediction performance was compared with a variety of alternatives by substituting the *p* and *q* values in the above model. It should be noted *d* has to remain the same. To measure performance and assess which parameter variation is most appropriate, the Normalised Root Mean Square Error (NRMSE) was computed and presented in table 3.2.1.

Table 3.2.1 – Mean NRMSE and train time for a range of model parameters.

Model	Mean NRMSE	Train Time (Seconds)
(0,1,1)	0.61140	18
(1,1,1)	0.59102	19
(1,1,0)	0.59114	1
(2,1,0)	0.58327	1
(2,1,1)	0.58318	19
(2,1,2)	0.58299	20
(1,1,2)	0.59079	20
(0,1,2)	0.61113	18

The NRMSE value of the preliminary STARIMA(1,1,1) model positions it as the fifth best performing of the eight in table 3.2.1, whilst STARIMA(2,1,2) displayed the lowest NRMSE and is therefore the most appropriate forecasting model. For a more in-depth insight, table 3.2.2 presents the NRMSE values for each link.

Table 3.2.2 – NRMSE for each LCAP using STARIMA (2, 1, 2).

STARIMA (2, 1, 2)	
LCAP LINK	**NRMSE**
419	0.65774
1599	0.56299
1979	0.80656
1813	0.69137
1447	0.49329
1419	0.55820
1425	0.60571
2112	0.54088
2059	0.87183
2361	0.46155
2363	0.49843
472	0.34332
473	0.48481
474	0.38232
1810	0.71869
418	0.74626
1956	0.31886
1957	0.55722
1807	0.77670
Mean NRMSE	**0.58299**

The influence of training data length was then investigated. Table 3.2.3 presents these results. As expected, as training data length increased, train time and mean NRMSE increased and decreased respectively. However, NRMSE values for training lengths of 7 and 14 days only differed by 0.005, whilst train time increased more proportionately, differing by 14 seconds. The decision to use 21 days of training data is justified as it produced the most accurate results, whilst maintaining a negligible training time.

Table 3.2.3 – Influence of training data length on STARIMA (2, 1, 2).

Model (2, 1, 2)		
Training Length (Days)	**Mean NRMSE**	**Train Time (Seconds)**
7	0.58801	6
14	0.58376	12
21	0.58299	20

Diagnostic Checking

Diagnostic Checking of fitted models is required to determine if the residuals are randomly distributed. The results for all links have been presented in figure 3.2.5, which is a STACF plot of the residuals. The plot shows autocorrelations cut off after the first lag which suggests a random distribution of residuals. This can be confirmed by figure 3.2.6 which is an alternative presentation method of plotting histograms for each LCAP link. A sample of three road links have been displayed, but analysis of all 19 histograms produced confirms the distribution of residuals is random.

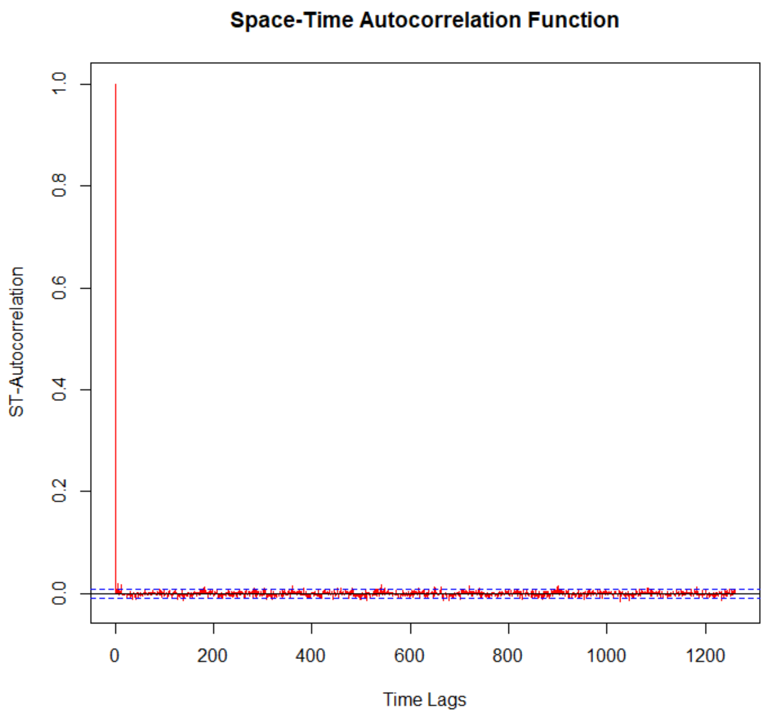


Figure 3.2.5 – STACF plot of the residuals from STARIMA (2, 1, 2).

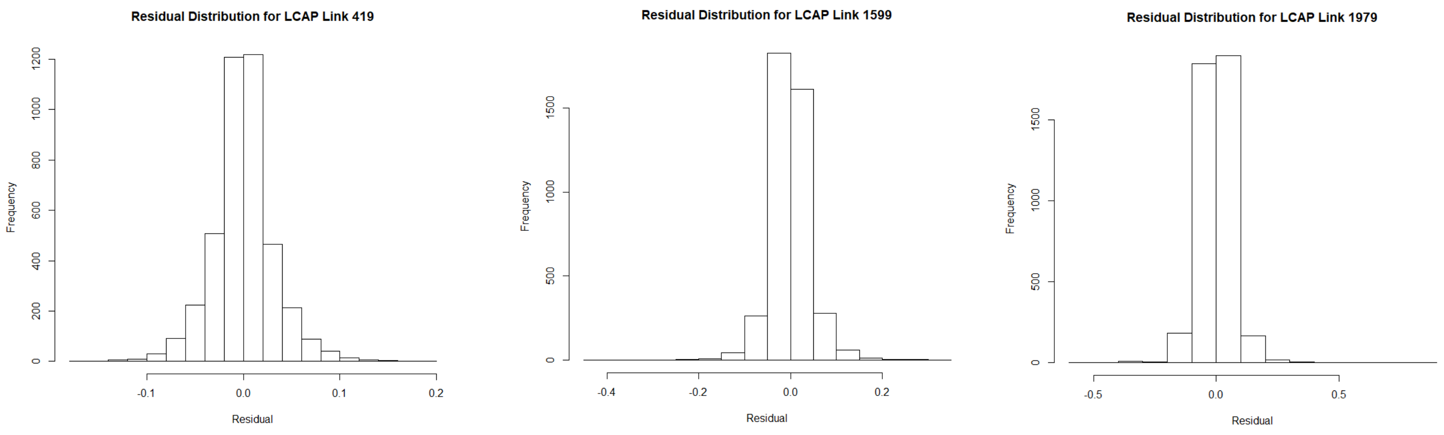
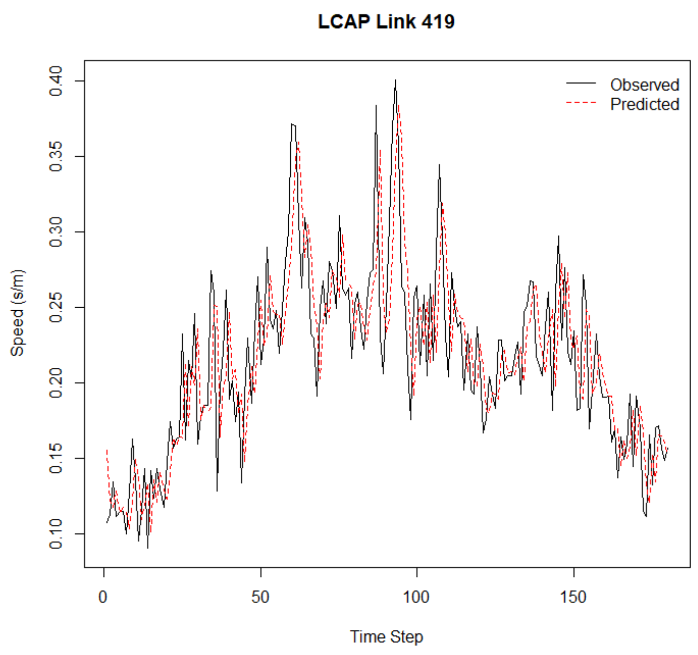
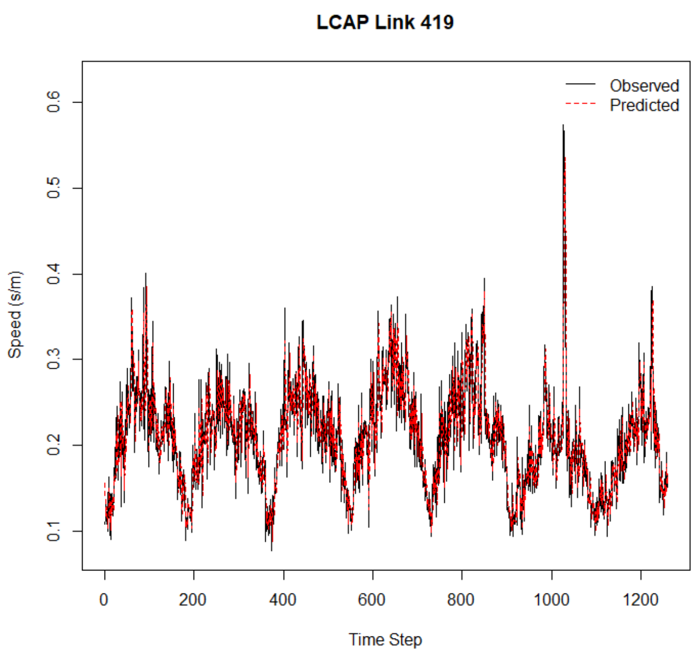


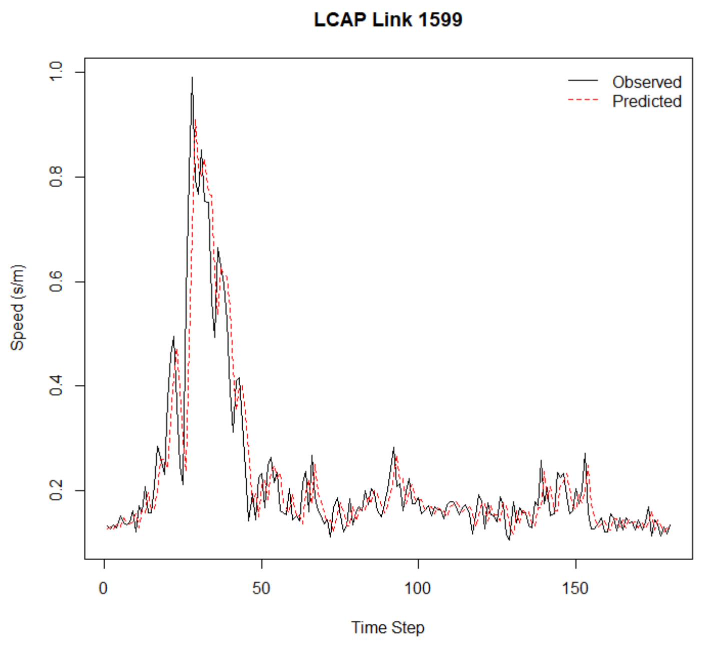
Figure 3.2.6 – Histogram residuals for LCAP link 419 (left), 1599 (middle) and 1979 (right).

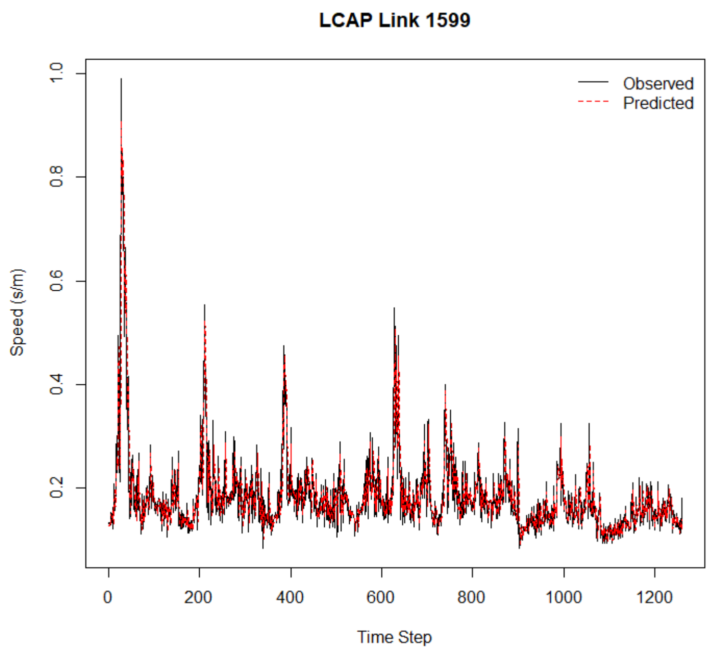
Prediction

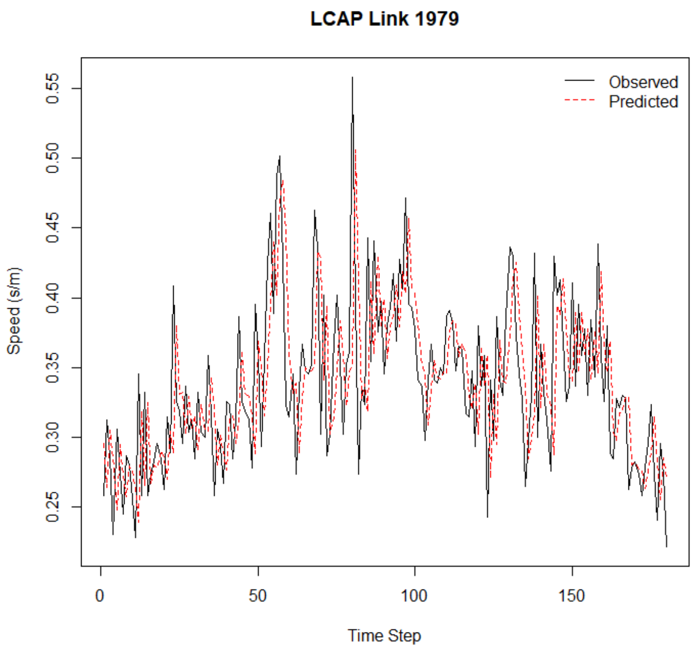
The STARIMA(2,1,2) model was trained using the first 23 days of the data series, with forecasting carried out on the reamining 7 days. The below plots display the predicted and observed values for three example links for one day and one week. This, combined with the mean NRMSE of 0.58299 can be used to determine that the prediction curve is relatively accurate, possessing the ability to reflect the variance associated with UJTs.











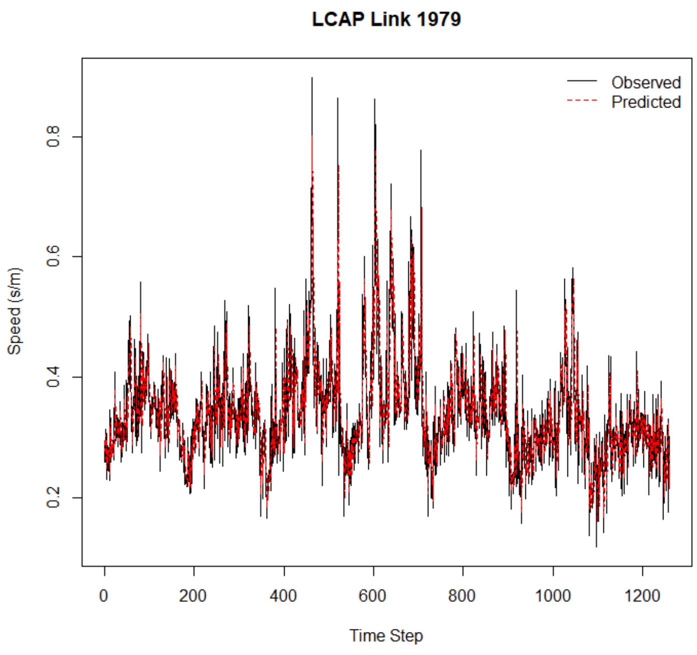


Figure 3.2.7 – One-step-ahead prediction using STARIMA (2, 1, 2). Both one day and one week predictions have been presented for LCAP link 419 (top), 1599 (middle) and 1979 (bottom) respectively.

Figure 3.2.7 presents the result of one-step-ahead prediction using the STARIMA(2,1,2) model for selected links. The visual analysis of predicting one day and one week time steps was used as they display the two main characteristics observed from these plots.

Using the one-day plots on the left of fig 3.2.7, it is evident that the predicted values exhibit a slight lag behind the observed values. The second characteristic is better visualised using the plots on the right of figure 3.2.7, displaying one-week prediction. The predictions (red dashed line) closely resemble the observed UJT values (black line) at the same time of the previous days. This is particular evident in the troughs of the data series. This is due to the STARIMA prediction calculation incorporating both previous values and previous errors.

3.3 – K Nearest Neighbour Regression (KNN)

Introduction

K-Nearest Neighbour (KNN) is a machine learning technique which can be used for forecasting spatiotemporal datasets such as the London traffic data we are exploring. KNN is a supervised learning algorithm which can be for both classification and regression (Mitchell, 1997). Due to our data set being continuous we will be performing KNN regression. Machine learning such as KNN is at the forefront of modern data science and can provide strong time series forecasting in the correct situations. There are however situations where machine learning is inappropriate or improperly used. Implementing machine learning on simple, small data sets can be time consuming and complicated and show little improvement in comparison to simpler methods (Blum, 1997). For this traffic data there are a large enough number of data points to warrant using a more complex procedure such as machine learning. Non-parametric tests, such as KNN, have adjustable parameters meaning that they are for flexible and can be fit to wider range of data sets (Cleophas, 2011). They also make no assumptions of the input data, which could skew results. This can lead to overall higher performing models as long as enough training data is provided, and the correct machine learning algorithm is used, as many algorithms are specialised for use in specific environments.

KNN has been used effectively for similar space-time forecasting with excellent results. Smith and Demetsky (1997) compared a number of forecasting methods on a freeway section in North Virginia. KNN outperformed both the parametric algorithm ARIMA and the more complex machine learning algorithms of neural networks. In a more recent study by Haworth and Cheng (2012) multiple different KKN algorithms were compared to each other and to kernel regression. They all performed similarly but KNNdist was the best performing by a small margin. KNNdist also only has one adjustable parameter, unlike KNNKernel, which makes it easier to optimise.

Methodology

The KNNdist algorithm performed well in case studies so will be the specific algorithm I am using for my analysis. KNN, like all machine learning algorithms, requires a training set to learn from and a test set to determine the quality of the analysis. From the 30 days of traffic data used, the training and test data is divided into the first 23 days and the last 7 days respectively. This provides a large training data set while testing the strength of the predictions over a week-long period.

The KNNdist first uses a distance function to identify the distance to all other points in the data set. A Euclidean distance function is a commonly used method for continuous data sets such as our London traffic data (Goldberger, 2005).

The next step is to order the data points by closest proximity at each time increment. The algorithm will then use a defined 'k' value parameter to extract the closest desired number of data points. The 'k' value is the specified number of nearest neighbours the algorithm will take into account. Increasing this value up to a point should mean that the prediction model fits the results better, but if set over a certain threshold the model will adjust towards the mean (Hassanat, 2014). From these points the inverse distance weight value is calculated. Inverse distance weighting extrapolates a value from another set of values depending on proximity (Maltamo, 1998).

KNN is a 'Lazy Learning' algorithm which means that there is no model fitting or training step. This reduces time for an individual prediction but means that repeated predictions can be computationally expensive (Zhang, 2007). To improve this, two additional data structures are implemented in my methodology; a KD tree and a cover tree. A KD tree find the median of the data, divides it and repeats to sort through data quicker (Friedman, 1977). The cover tree is another metric tree specifically designed for nearest neighbour searches. The tree can be constructed in O ($c^6 n \log n$) time where n is the 'k' value and c is the expansion constant of the dataset (Beygelzimer, 2006).

Cross-Validation

Cross-validation is a technique used to evaluate predictive models with an independent parameter which can be adjusted (Kohavi, 1995). For KNN the independent parameter used is 'k', the number of nearest neighbours registered for each predicted point. The cross-validation process will test the RMSE values of the training set against the test set at a range of values of k. The root-mean-square error (RMSE) is a method for measuring the accuracy of a predicted set of values against observed values. The value is found by calculating the value of the square root of the mean of squared errors. The use of squared errors in the formula mean that outliers in the model are magnified in the RMSE. This means that the test is particularly sensitive to large outliers.

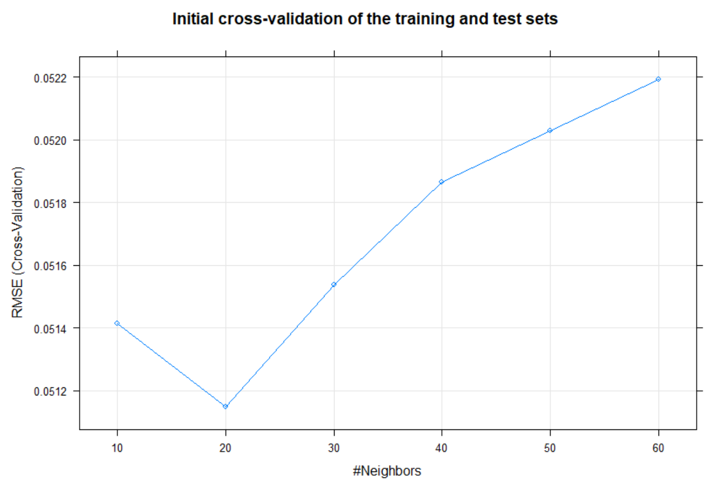


Fig 3.3.1: Initial cross-validation of the traffic data

Fig 3.3.1 shows plots the observed RMSE values at different values of ‘k’. We can see that the optimal RMSE lies somewhere around 20, but the optimal k value may be either side of it. After one set of RMSE values is calculated, the strongest RMSE value can be taken and the process repeated at a more precise level.

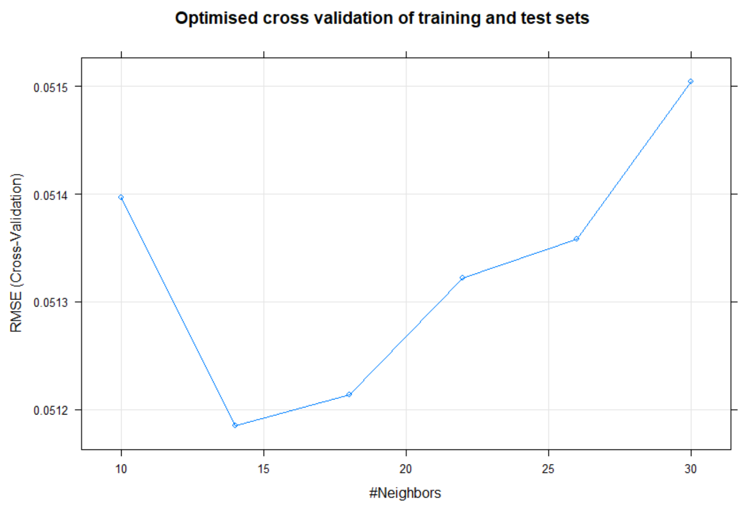


Fig 3.3.2: Optimised cross-validation of the traffic data

Fig 3.3.2 repeats the cross validation using smaller increments of ‘k’ with the strongest RMSE value as the centre point. We can see that the highest performing value for ‘k’ in the data set is 14, which will be the ‘k’ value set for the KNN algorithm.

Prediction

For the initial analysis of the predictions I will be looking at 3 specific LCAP links; 419, 1599 and 1979. The Normalized RMSE (NRMSE) adjusts the RMSE by dividing it by the standard deviation of the measured data. The conversion to NRMSE allows for comparison between different models using different scales.

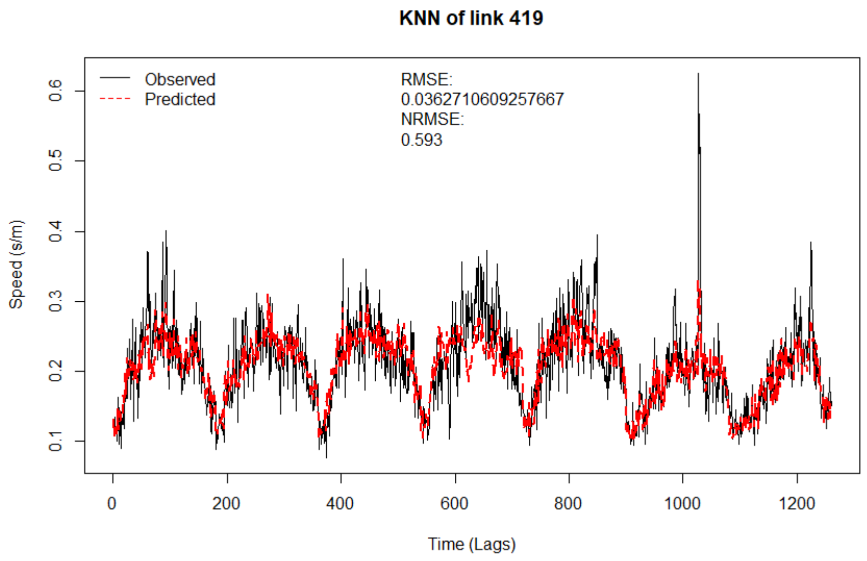


Fig 3.3.3: KNN prediction vs observed for link 499

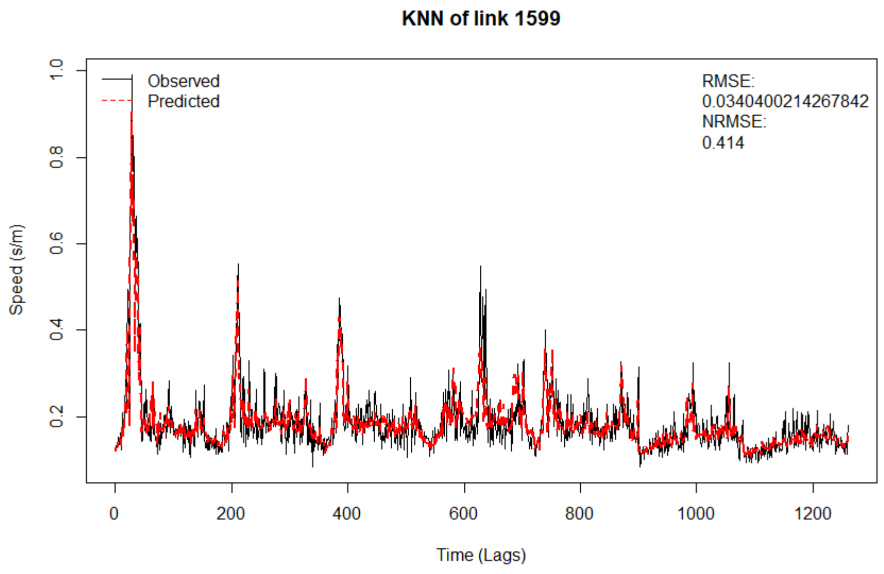


Fig 3.3.4: KNN prediction vs observed for link 1599

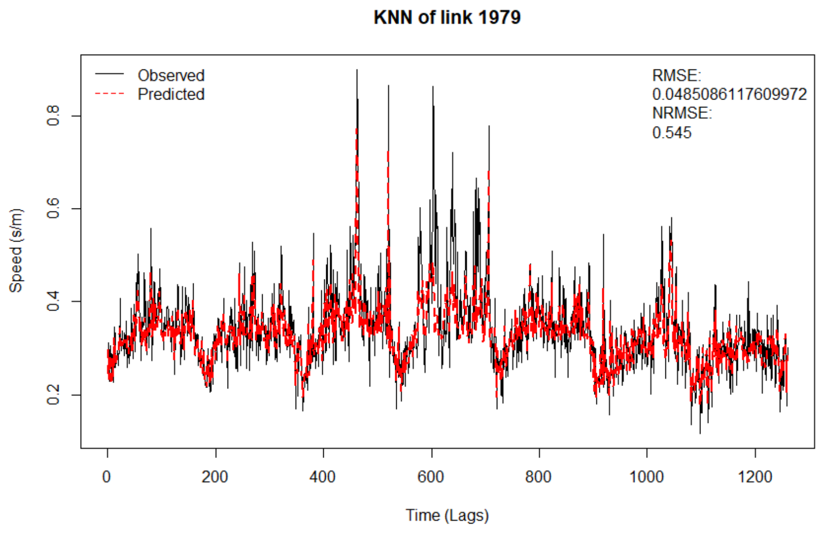


Fig 3.3.5: KNN prediction vs observed for link 1979

From these graphs we can see that the model predicts the general trend of the test data extremely well. In all cases the algorithm tends to underestimate the height of peaks and troughs that occur over very short periods of time. In some cases, larger peaks and troughs are not predicted, such as Fig 3.3.5 at 600 lags and 3.3.3 at 1000 lags. Errors such as this can increase the RMSE dramatically, such as in link 1979 (Fig 3.3.5).

In addition to comparing the visualisation of individual link predictions, the NRMSE for all links at multiple training set lengths has been collected.

Table 3.3.1: NRMSE values for all LCAP links in study

link	**23 day train**	**14 day train**	**7 day train**
449	0.593	0.613	0.724
1599	0.414	0.477	0.612
1979	0.545	0.592	0.866
1813	0.575	0.589	0.712
1447	0.519	0.64	0.744
1419	0.604	0.659	0.856
1425	0.623	0.669	0.907
2112	0.685	0.72	0.822
2059	0.725	0.735	0.77
2361	0.458	0.444	0.671
2363	0.595	0.899	0.902
472	0.35	0.486	0.653
473	0.286	0.326	0.754
474	0.386	0.43	0.575
1810	0.514	0.548	0.614
418	0.714	0.933	0.599
1956	0.412	0.41	0.463
1957	0.694	0.713	0.774
1807	0.571	0.586	0.646
Mean	**0.540157895**	**0.60363158**	**0.7191579**

We can see that NRMSE values between links can be very different (Table 3.3.1). For the full 23 day trained predictions they range from 0.286 to 0.725. This could probably be reduced if cross-validation was performed for each individual link instead of for the data set as a whole. The difference in NRMSE between training set sizes is to be expected. The mean NRMSE of the links increases as training set size goes down. However, there are cases where this does not happen for individual links, such as link 418. It could be that the specific week used for training was very similar to the prediction by chance. This will only happen occasionally and in most cases a larger training set is better (Shalev-Shwartz, 2008).

Limited Training Data

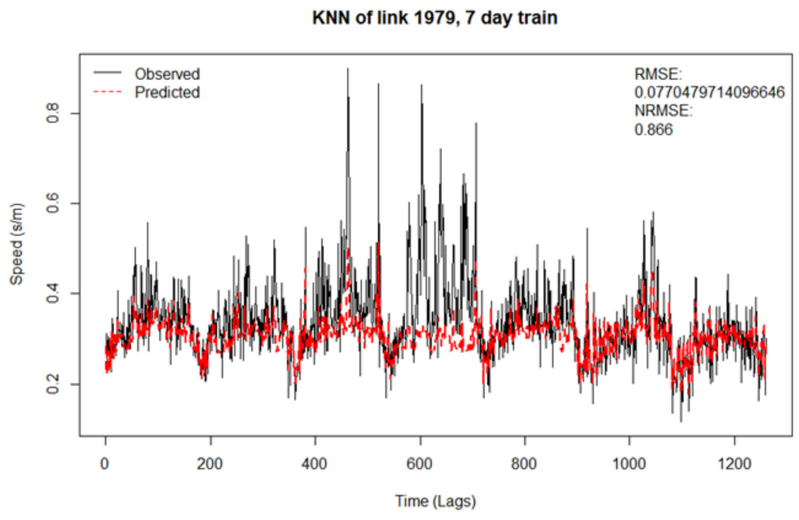


Fig 3.3.6: KNN prediction vs observation for link 1979 with 7-day train

The length of the training set was reduced from 23 days to 7 to test how the algorithm performs with limited training data. Fig 3.3.6 shows how the prediction for link 1979 performs under these conditions. The prediction underestimates the peaks and troughs to an even higher degree when compared to the 23-day training set. While most of the troughs are predicted reasonably well, most of the peaks are greatly underestimated or completely missed. The large peaks from 600-700 lags are not registered at all. This could mean that a training set larger than 23 days could improve the model further. The limitation of computation time is not visible with data this small and with KD and cover trees implemented. All times I ran the algorithm the completion time was less than 1 second, but this will increase when using larger sets of data.

Differencing

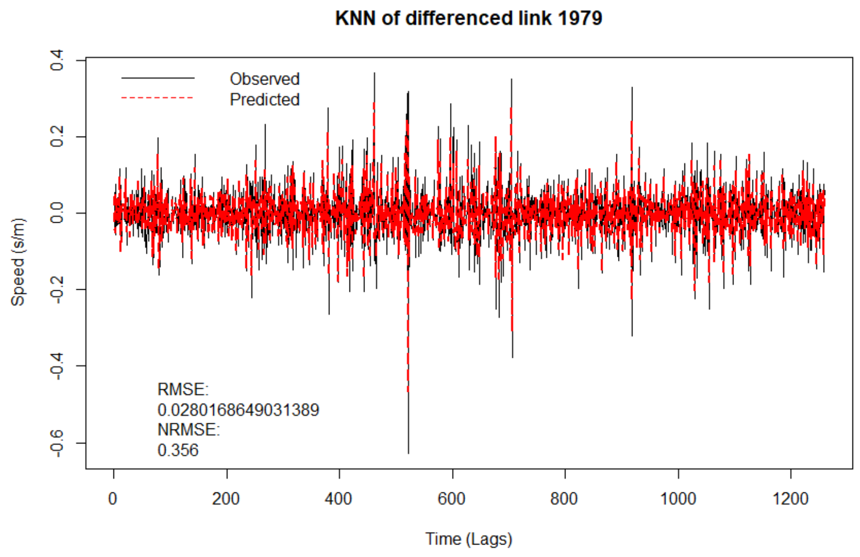


Fig 3.3.8: Differenced prediction vs observation for link 1979.

.

Table 3.3.2: RMSE values of differenced results.

link	RMSE	NRMSE
449	0.3054	0.705
1599	0.0294	0.613
1979	0.2801	0.356

Differencing is a process which makes a data set stationary. This removes trends and seasonality trends within the data. Removing this can improve the effectiveness of the algorithm as it is easier to predict series with no seasonality (Hosking, 1984). Table 3.3.2 shows that a prediction of the differenced data has an improved RMSE in all situations, but the NRMSE value was better in some and worse in others. The differenced results are however harder to interpret visually (Fig. 3.3.8).

3.4 – Support Vector Machines (SVMs)

Overview of Methodology

Support Vector Machines (SVM's), are statistical learning machines based on Vipnik (1995) theory to implement a minimized risk principle to achieve good generalization on a limited number of learning patterns (Basak, 2007). SVMs can be divided in to two classes: Support Vector Regression (SVR) and Support Vector Classification (SVC). SVRs are the predominant application due to its inherit ability as an accurate forecasting method. SVR procedures is the process deriving a function *f(x)*, with the least deviation from the obtained targets for the training data, whilst remaining as flat as possible (Smola and Scholkopf, 2004). A significant advantage of using SVRMs is due to minimization of generalization, opposed to traditional statistical regression techniques often seeking to minimize training error. (Hsu et al., 2003)

Kernel functions allow the application of regression algorithms to both linear and non-linear machine learning problems by mapping the original input space into higher-dimensional feature space. Weighting is applied on the similarity between objects, for example it will represent a spatial-temporal autocorrelation, where distance and temporal decay in relation to similarity between the observed training and predicted model. Kernel regression is a non-parametric technique applied to forecasting time-series data due to its application to multidimensional data (Leng et al., 2013, Han et al., 2010; Lu et al., 2009). (Howarth, 2014)

The following report will investigate the implementation and suitability of Kernel-SVM techniques as a forecasting method for predicting spatial-temporal variations. The consequent analysis aims to derive the most efficient KSVM model, whilst also delineating which parameters have the greatest influence. The study aims to represent the effect of tuning by comparing results of an 'Initial' and 'Optimised' model. Comparisons will be performed on three links as to reduce potential outliers affecting results.

Build an Initial Test Model

Firstly, an initial, basic space-time SVM model was constructed. Using an adaptation of the *embed* function, *st_embed,* selected a particular location of the selected UJT Data along with its adjacent neighbours using the aforementioned spatial adjacency matrix as described previously in the report. The consequent data was divided to train and test datasets, consisting of the first 23 days (1:4140 timesteps) and last 7 days (4141:5400 timesteps) respectively. This allowed the initial KSVM model to be created through Caret's '*train'* function.

For the purpose of this study, the Gaussian Radial Basis Function (RBF) was considered most suitable kernel method to obtain optimal solution. There are many advantages for using the RBF (Lin et al., 2008), and consequently is widely observed to obtain the best performance throughout multiple applications (Howarth, 2014; Han et al., 2010). The initial model has the following parameters (all other parameters were automatically computed by the KSVM model):

- *Training data length = 23 days*
- *Kernel = Radial*
- *M = 3 (m+1)*

Figure 3.4.1 - The formula for a Gaussian Radial Basis Kernel Function. Where σ (> 0) is the kernel width. (Ojemakinde, 2006)

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\frac{\left(x_i - x_j\right)^2}{\sigma^2}\right),$$

Table 3.4.1 displays weak RMSE value, the most frequent unit of measure for a regression model. The RMSE values represent the measure of average squared deviation of forecasted values, where a low RMSE value is desired. The NRMSE measure will be the predominant unit of measured inference, to allow comparative analysis between different forecasting models used throughout the study (ARIMA, STARIMA, KNN). (Adhikari and Agrawal, 2013)

The NRMSE (Normalised Root Mean Square Error) was obtained through '*nrmse'* function from the "*hydroGOF*" package. The mathematical function behind this algorithm can be found below:

$$nrmse = 100 \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(S_i - O_i)^2}}{nval}$$

$$nval = \begin{cases} sd(O_i) & \text{, norm="sd"} \\ O_{max} - O_{min} & \text{, norm="maxmin"} \end{cases}$$

Figure 3.4.2 – nrmse function from "hydroGOF". https://www.rdocumentation.org/packages/hydroGOF/versions/0.3-10/topics/nrmse.

Figure 3.4.3 displays the predicted values exhibiting a lag behind the observed values. It can be seen the predicted (red-dashed) values don't closely follow the observed (black), this however is in conjunction with the NRMSE values in *Table 3.4.1*. For example, Link-1979 has the highest and therefore worst NRMSE result, as well as showing the greatest variance in plotted lines from *figure 3.4.3*.

Table 3.4.1 – Table displays the statistical results for the 'Initial' model.

LCAP Link	*NRMSE*	*RMSE*	*R-squared*	*MAE*	*Time (seconds)*
419	0.694	0.0424642	0.52837966	0.03031629	63.686
1599	0.789	0.06495227	0.4176563	0.03529315	65.593
1979	0.854	0.07599966	0.3073217	0.05133949	70.773
Mean	**0.779**	**0.06113871**	**0.41778589**	**0.03898298**	**66.684**

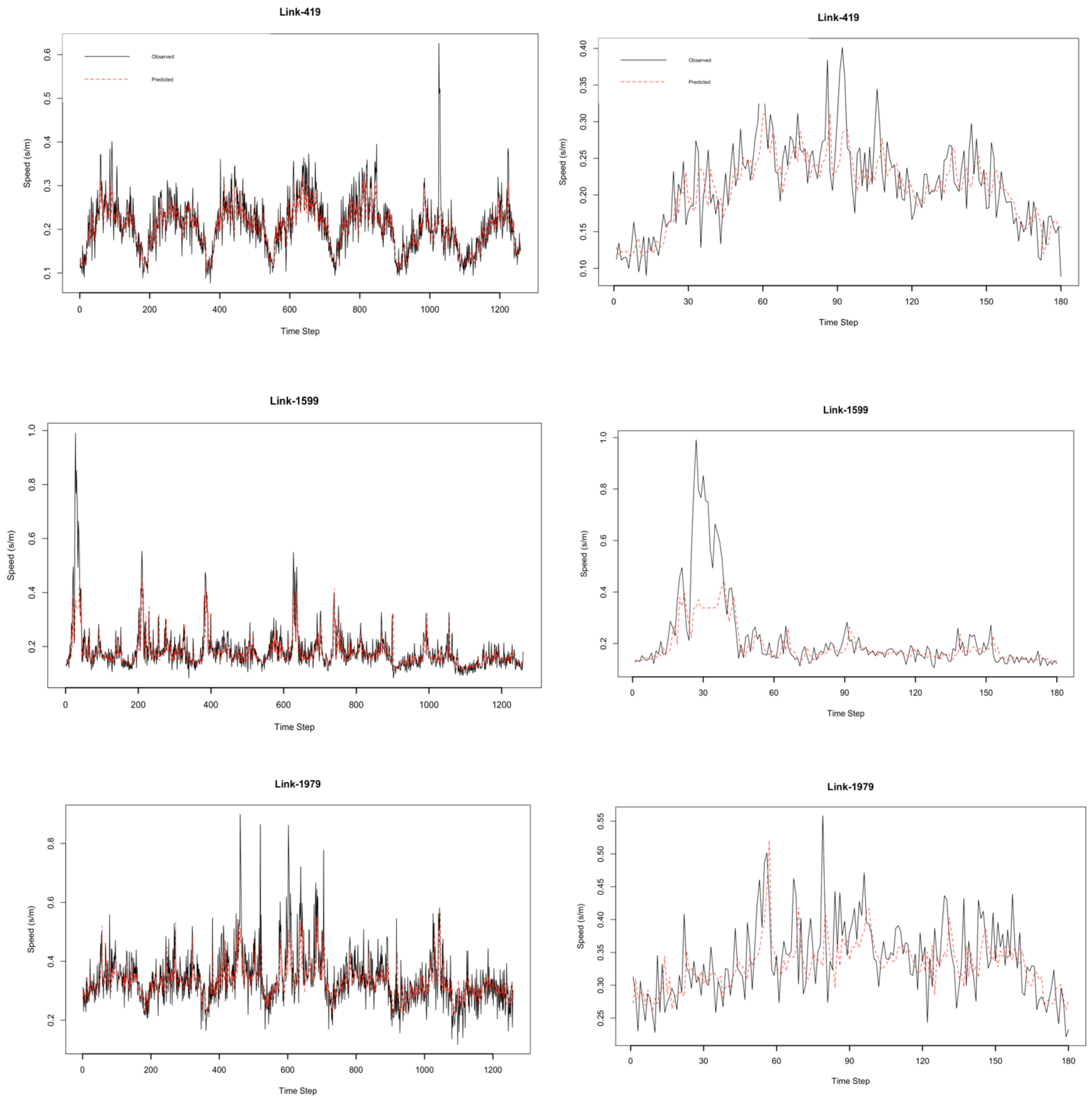


Figure 3.4.3 – One-step-ahead prediction of the 'observed' vs 'predicted' values for the initial SVR model. Both one-week (Time Step 0-1200) and one day (Time Step 0-180) predictions have presented for LCAP links; 419, 1599, and 1979.

Based on results ranging between the other two LCAP Links, Link-1599 was deemed to most likely be a true representative of the entire dataset when deciding the training parameters, and therefore is used to refine the model further.

Tuning Parameters

When performing KSVM forecasting methods, it is critical to ‘tune’ the model parameters within the training procedure. Individual parameters have been observed to greatly influence the accuracy and reliability of the model (Cherkassky and Ma, 2004).

The two most significant RBF parameters within SVMs, are Cost (C), and Gamma (Sigma). If C value is too great, the training phase accuracy is very high whilst the testing phase is very low. Conversely, if C value is too low, the accuracy becomes inadequate, rendering the model unsuitable. The Gamma parameter is the foremost influential parameter, where disproportionately large Gamma values score over-fitting, whereas, excessively large values result in under-fitting (Pardo and Sberveglieri, 2005). (Lin et al., 2008; Zeng et al., 2008)

A k-fold Cross-Validation (CV) method was implemented to tune and refine the SVM model to fully optimise the model’s potential of accurately forecasting the UJT dataset. CV (Figure 3.4.4) was used to find the optimised parameters using the grid function, where initially the model used values C=1 and Sigma=2.63, the CV resultant model obtained values C=5000 and Sigma=0.001. The new, tuned model displayed exceedingly greater statistical results.

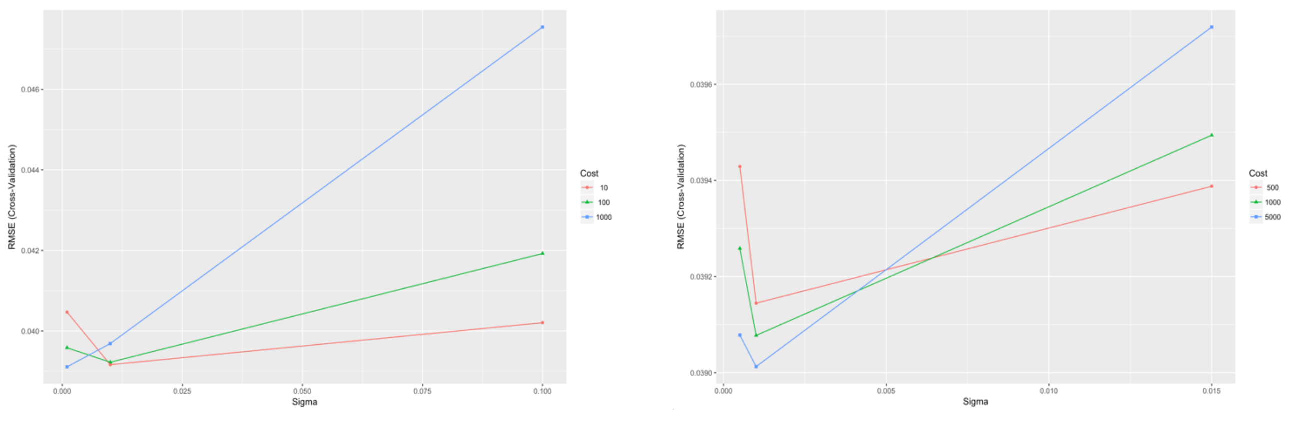
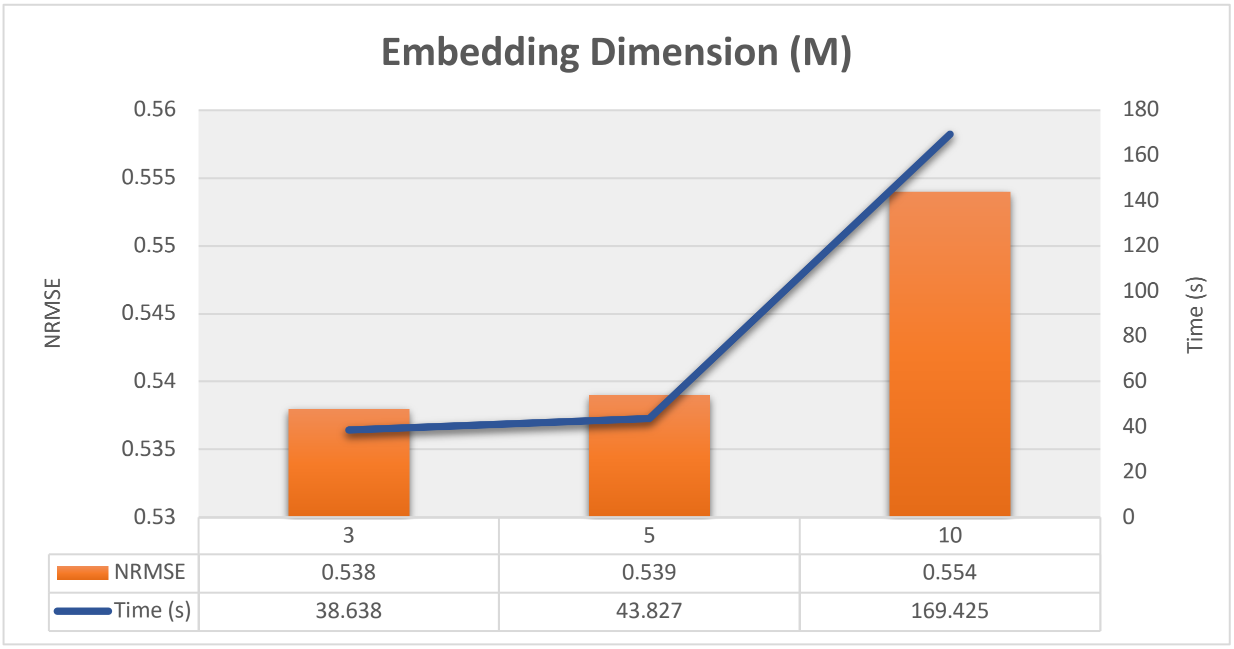


Figure 3.4.4: The two plots display the refinement of the Cost and Sigma values through the use of tuning the training grid within the model train function.

Image on the left shows the initial Cost and Sigma parameters.

The image on the right displays the final best-fit Cost and Sigma values.



	3	5	10
🟧 NRMSE	0.538	0.539	0.554
▬ Time (s)	38.638	43.827	169.425

Figure 3.4.5 – Displays graph showing the results of varying levels of M values in relation to both time and NRMSE.

The embedding dimension has been observed as another influential parameters to time-series modelling (Harikrishnan, 2017; Coa, 1997). Figure 3.4.5 displays three representative M values (3, 5, 10). The results support the observation, as the M value has strong negative correlation with NRMSE and Time (represents the computational time taken to perform model). From these results, an embedding dimension of 3 was chosen in the optimised model.

Other parameter and pre-processing techniques such as *scale* and *center* were tested but no significance influence upon accuracy or efficiency of the model, therefore these variables were dismissed.

Testing Data size

This section of the report will investigate the influence of the training dataset in relation to size, using three different size TrainUJT datasets, of 7, 14, and 23 days.

```
Call:
summary.resamples(object = resamps)

Models: SevenDays, FourteenDays, TwentyThreeDays
Number of resamples: 10

MAE
                      Min.    1st Qu.     Median       Mean    3rd Qu.       Max. NA's
SevenDays       0.01990492 0.02130889 0.02190487 0.02239403 0.02324798 0.02615521    0
FourteenDays    0.02449986 0.02487528 0.02620269 0.02607209 0.02673939 0.02837326    0
TwentyThreeDays 0.02617672 0.02682629 0.02733982 0.02782744 0.02836207 0.03202024    0

RMSE
                      Min.    1st Qu.     Median       Mean    3rd Qu.       Max. NA's
SevenDays       0.02841438 0.02961552 0.03245462 0.03338825 0.03456904 0.04417147    0
FourteenDays    0.03450916 0.03739961 0.03967189 0.03906097 0.04038196 0.04306490    0
TwentyThreeDays 0.03755908 0.03956021 0.04137363 0.04278580 0.04542247 0.05119615    0

Rsquared
                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
SevenDays       0.3994389 0.5092358 0.5827370 0.5722592 0.6426498 0.6751896    0
FourteenDays    0.5440098 0.6739988 0.6971232 0.6808935 0.7194295 0.7960818    0
TwentyThreeDays 0.6968311 0.7721002 0.8245472 0.8036247 0.8458903 0.8640000    0
```

Figure 3.4.6 – A statistical summation of results associated with the three size datasets (7, 14, 23 days) using the 'resamples' function from Caret.

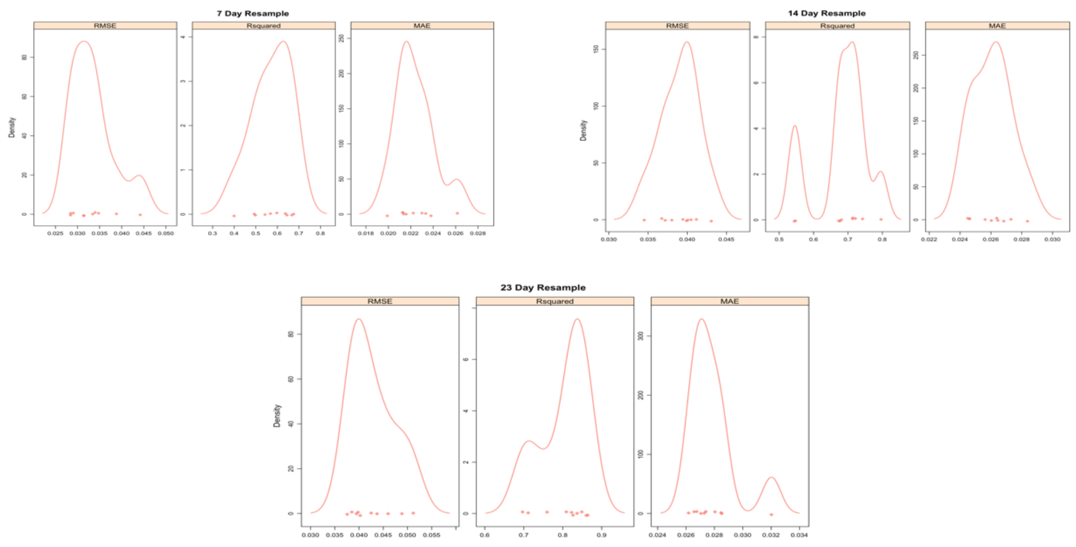


Figure 3.4.7 – Density plots of the three training datasets using the 'resamples' function in Caret. This shows the distribution of the results across their respective data training length.

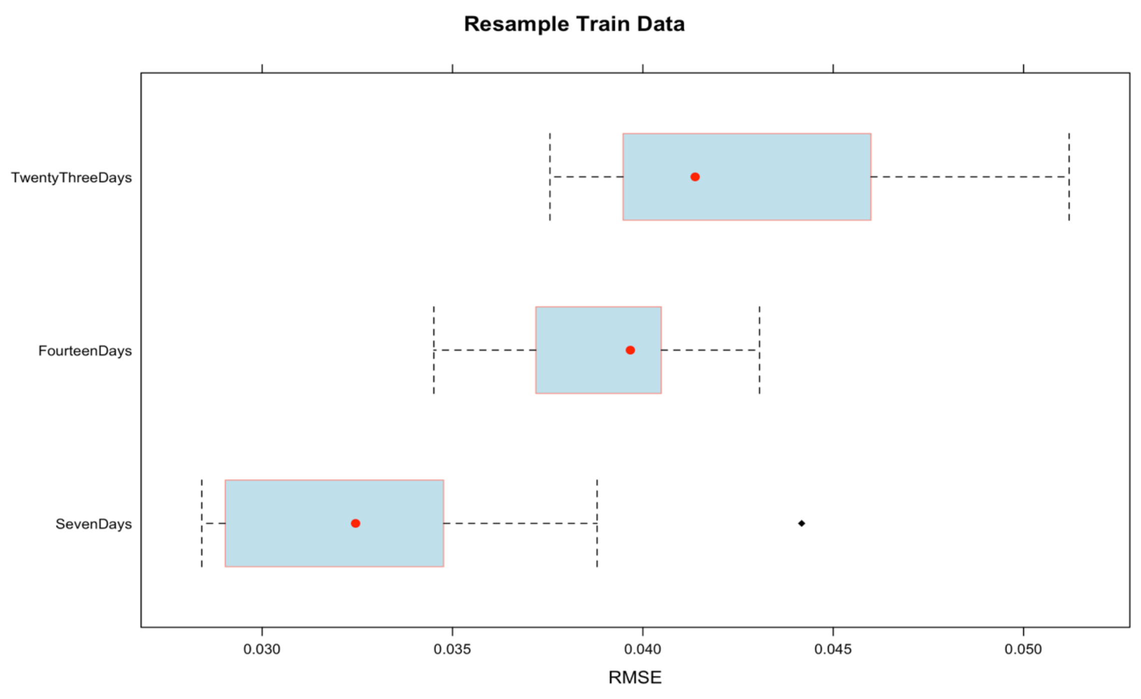


Figure 3.4.8 - Box-Whisker plot of the RMSE results, derived using the 'resamples' function from Caret.

An outlier can be observed in the Seven-Days dataset, corresponding to Figure 3.4.3 where both 1-day and 7-day test data in Link-1599 have extreme anomaly data values. Anomaly data has greater influence in smaller datasets (SevenDays), as less values to normalize and difference any outlier data. Figure's 3.4.6 and 3.4.8 both RMSE and R-square values are greatest for 23-days, consequently remaining the optimum train-data size. Figure 3.4.7 further supports the use of the 23-day training length in final model where fluctuations in distribution of RMSE and R-squared values in 7- and 14-day are far less normalised than 23-days.

Final model results

Previous procedures were performed aiming to build an 'optimised' model based on supplied data. The selected parameter values used to create the optimal model are:

- *C = 5000*
- *Sigma = 0.01*
- *Training data length = 23 days*
- *Kernel = Radial*
- *M = 3 (m+1)*

Table 3.4.2 - Table displays the statistical results for the 'Optimal' model.

LCAP Link	NRMSE	RMSE	R-Squared	MAE	Time (s)
419	0.636	0.03885595	0.59876413	0.02881658	86.364
1599	0.54	0.04442554	0.71579863	0.03021741	94.419
1979	0.674	0.04082253	0.55655852	0.03072632	76.372
Mean	**0.61666667**	**0.04136801**	**0.62370709**	**0.0299201**	**85.7183333**

Comparing results from both the 'initial' and developed 'optimal' radial KSVM models, a simple observation that tuning the parameters of an SVM model has substantial influence of the performance of the respected model. The optimal model has a significantly improved NRMSE result, initial mean value of 0.779, decreased to 0.617, a far more suitable result.

Figure 3.4.9 also enables comparisons between NRMSE results before and after the optimal tuning procedure. It's clear that each optimal Link result is notably fewer than its respective equal. The exaggeration of this correlation in Link-1599 is expected as the study procedure used Link-1599 to determine its optimal parameter values. Additionally, it must be noted that the model computation time was observed to vary dramatically when repeating identical tests throughout the experiment. This may be attributed to multiple varying controlled and uncontrolled conditions when performing each computational assessment. Consequently, within this study, time cannot be regarded as an accurate measure of assessing model efficiencies due to the uncontrolled nature of the experiments and ambiguity of timed results.

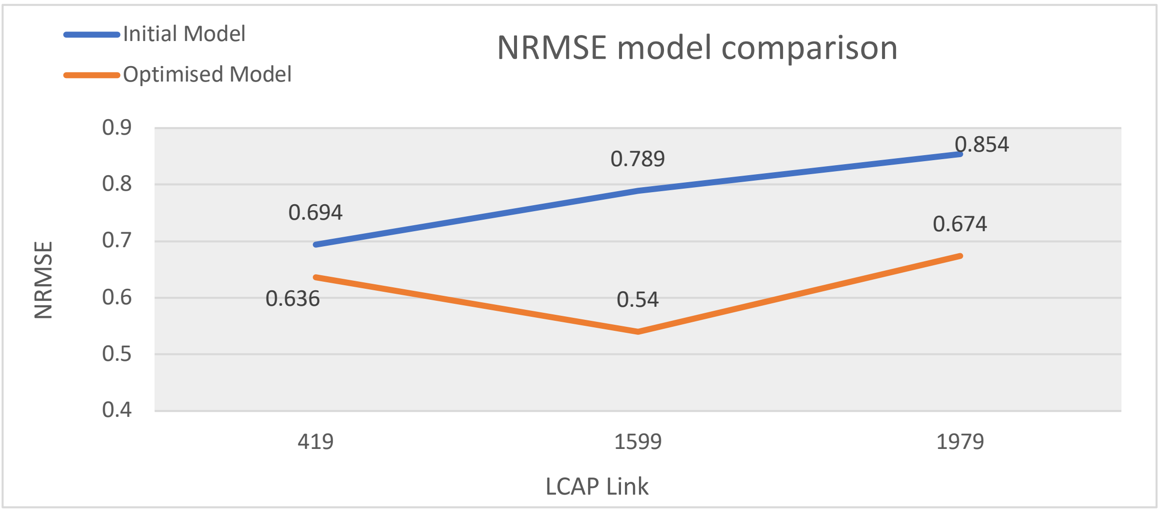


Figure 3.4.9 – NRMSE results over the three study LCAP Links, in relation to both the initial model created before improving parameter values, and optimal model

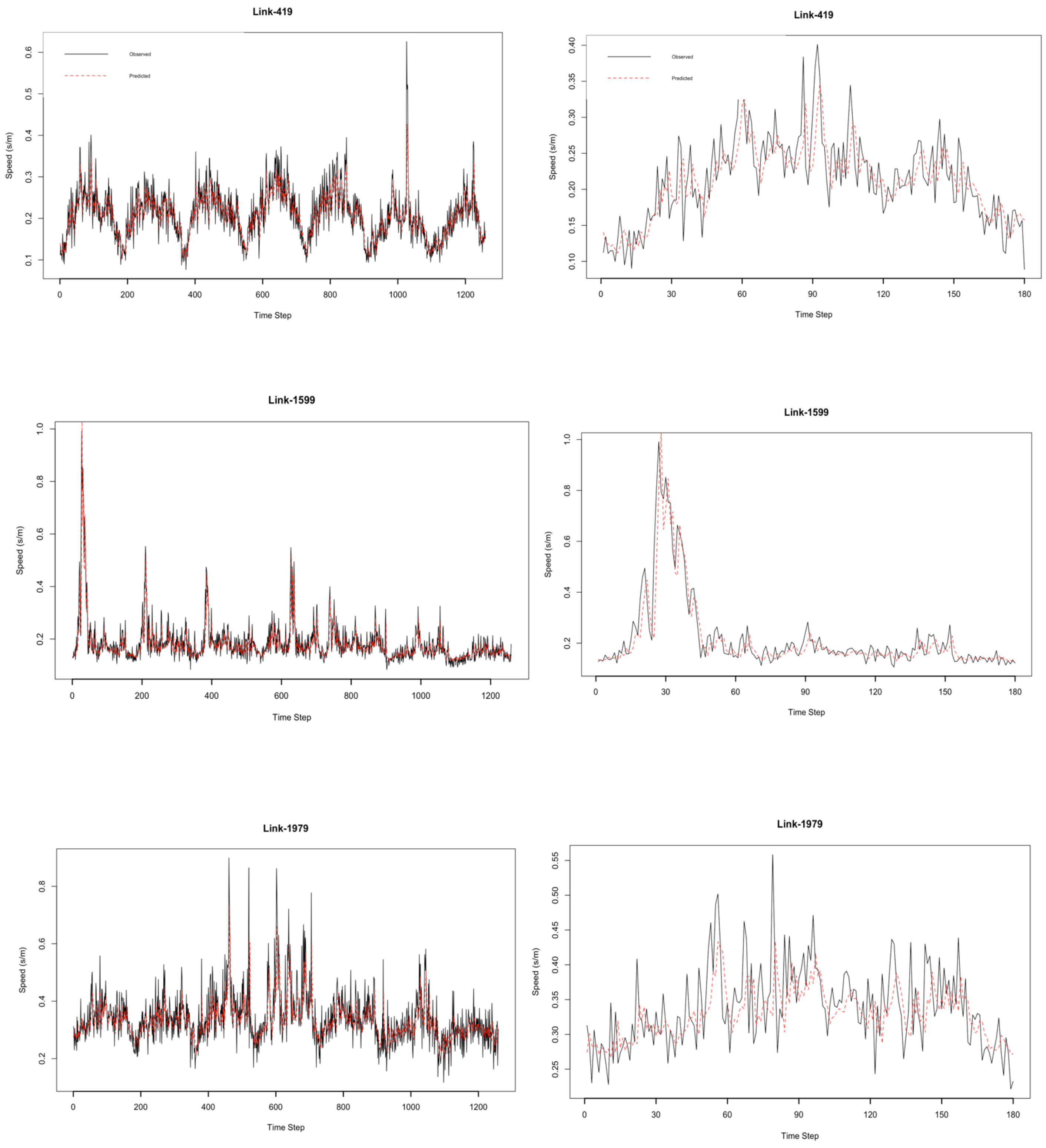


Figure 3.4.10 – One-step-ahead (OSA) prediction of the 'observed' vs 'predicted' values for the final SVR model. Both one week (Time Step 0-1200) and one day (Time Step 0.180) predictions have presented for LCAP links; 419, 1599, and 1979. One week and one day time steps were chosen to display the two main temporal characteristics from these plots.

The results from the OSA predictions further support the significance of tuning parameters. Visual analysis between Figure 3.4.3 and Figure 3.4.10 illustrates clearly a much closer fit of predicted to observed values in the optimised model. Due to this, temporal patterns in the one-week plots display daily variant patterns in the dataset.

Table 3.4.3 – NRMSE results for each LCAP Link using 'optimised' model.

LCAP Link	NRMSE
419	0.636
1599	0.54
1979	0.674
1813	0.672
1447	0.486
1419	0.546
1425	0.595
2112	0.54
2059	0.822
2361	0.646
2363	0.335
472	0.335
473	0.484
474	0.369
1810	0.688
418	0.732
1956	0.313
1957	0.544
1807	0.742
Mean	0.56310526

The methods illustrated throughout this individual study have been with the purpose of building, tuning, and analysing an 'optimised' KSVM used to forecast UJT times in relation to spatial-temporal functions. The results from this study are evident of the influence of tuning model parameters, where the 'optimised' model showed a far greater level of forecast accuracy than the 'initial' basic KSVM model.

4 – Discussion

4.1 - Evaluation of model performance

The NRMSE index was the predominant measure used for assessing each model's performance, generating values of 1.32, 0.58, 0.54 and 0.56 for ARIMA, STARIMA, KNN and SVMs respectively. Normalising the RMSE index allows the different modules to be directly compared. Therefore, the KNN model produced the most accurate forecasting model. Conversely, ARIMA was significantly outperformed by all other models. It is proposed that the absence of strong seasonality contributed to the poor performance of the ARIMA model, aforementioned in section 3.1.

4.2 - The relative merits of each model

The methods described in section 3 vary in complexity, and interpretability as a result. Although each method requires a degree of knowledge and interpretation to create an effective model, the non-parametric models, SVM and KNN are more difficult to interpret. This is in accordance with literature, as a highly documented disadvantage of both models is the lack of interpretability (Papernot and McDaniel, 2018; Nguyen and Le, 2014). Furthermore, the parametric models are more parsimonious, with substantially less parameters involved in model building which results in greater interpretability and ease of implementation (McLeod, 1993).

The training times of the models implemented in section 3 varied quite significantly. The most time-consuming model, SVM, measured values exceeding 90 seconds, notably greater than the second-highest model, STARIMA (2, 1, 2), taking only 20 seconds which is relatively negligible, reducing the appropriateness of the measure. However, it is important to note that in a future analysis covering an increased number of road links and a larger data series covering several months/years, running time will become more important. Moreover, if strong seasonality was exhibited in the data series and considered in the models, it is expected that the training time would increase, with the STARIMA model seeing the greatest effect.

The importance of hardware specification should be emphasised in regard to computation time. Each of the models in section 3 were run on different machines, with different specifications, which yield largely incomparable training results. A more reliable comparison would be made if each model was run on the same machine.

4.3 Variation of model performance across the study area

In terms of variation of model performance across the study area, there is no eminent trend or pattern. To further evaluate this, the three model performances of STARIMA, KNN and SVM were plotted in a radar chart (Figure 4.3.1) which provides an easily interpretable visualisation. ARIMA was not included in this analysis due to its unsuitability as a forecasting technique for this specific study in comparison to the other models.

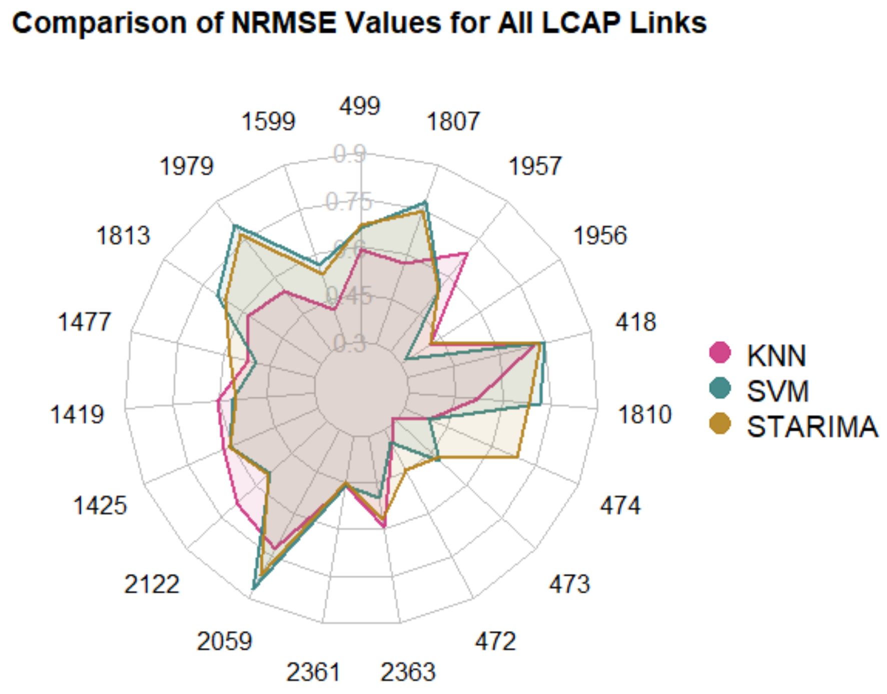


Figure 4.3.1 – Comparison of model performance for all LCAP Links.

It can be seen from Figure 4.3.1 that the performance of STARIMA and SVM on each LCAP Link was much more comparable than KNN. Each model displays great variation in forecasting accuracy for each road link. For example, KNN is much more accurate than the other models for links 1599, 473, 1979 and 1813. However, was significantly outperformed on links 2122, 1425, 1419, 1957.

Furthermore, the variation in model performance for STARIMA has been displayed geographically in figure 4.3.2. As it is evident from figure 4.3.1 that there is no trend present in the performance of the models, the geographical variation is only presented for one model as an alternative visualisation. The NRMSE values of STARIMA (2, 1, 2) were split into percentiles and plotted as a choropleth map. The result does not display any grouping of geographical errors, with each percentile being distributed across the study area, reaffirming the findings displayed in figure 4.3.1.

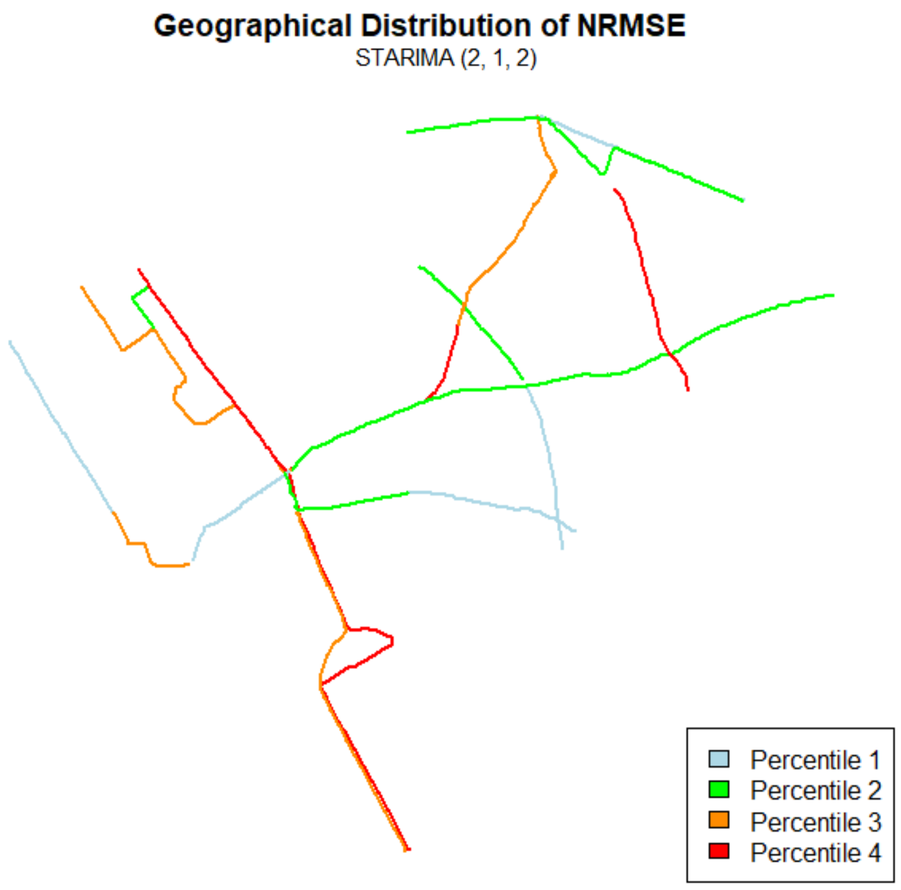


Figure 4.3.2 – Geographical distribution of STARIMA forecasting performance.

4.4 - Method improvement and further research

Greater insight and perhaps more accurate forecasting might have been achieved if the data series had been further subdivided into weekdays and weekends. This may benefit model training as weekdays and weekends exhibit different trends. For example, the cyclical evening and morning peak traffic patterns are much more evident in weekdays.

The absence of seasonality in the series was deemed to have a negative impact on model forecasting. This was specifically the case for ARIMA. Additionally, each model would have increased in accuracy if exposed to increased training data length.

Future research could aim at improving existing forecasting models by incorporating additional variables. Social media data is receiving a lot of interest as many consider it an important source for traffic information (Lv et al., 2017). Fusing of traffic and social media data sources can be seen in Lin et al., (2017), as the authors integrate tweet sensors into their prediction models. However, a recent review by Lv et al., (2017) concludes associated research is being dominated by the USA and China, which leads the authors to suggest future research determines how the incorporation of social media data affects traffic forecasting models on London's road network.

5 – References

Adhikari, R. and Agrawal, R.K., 2013. An introductory study on time series modelling and forecasting. *arXiv preprint arXiv:1302.6613*.

Ahmed, M.S. and Cook, A.R., 1979. *Analysis of freeway traffic time-series data by using Box-Jenkins techniques* (No. 722).

Bae, B., Kim, H., Lim, H., Liu, Y., Han, L. and Freeze, P., 2018. Missing data imputation for traffic flow speed using spatio-temporal cokriging. *Transportation Research Part C: Emerging Technologies*, 88, pp.124-139.

Basak, D., Pal, S. and Patranabis, D.C., 2007. Support vector regression. *Neural Information Processing-Letters and Reviews*, *11*(10), pp.203-224.

Beygelzimer, A., Kakade, S. and Langford, J., 2006, June. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning* (pp. 97-104). ACM.

Blum, A.L. and Langley, P., 1997. Selection of relevant features and examples in machine learning. Artificial intelligence, 97(1-2), pp.245-271.

Cao, L., 1997. Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D: Nonlinear Phenomena*, *110*(1-2), pp.43-50.

Castillo, E., Menéndez, J. and Sánchez-Cambronero, S., 2008. Predicting traffic flow using Bayesian networks. *Transportation Research Part B: Methodological*, 42(5), pp.482-509.

Cheng, T., Haworth, J. and Wang, J., 2011. Spatio-temporal autocorrelation of road network data. *Journal of Geographical Systems*, 14(4), pp.389-413.a

Cherkassky, V. and Ma, Y., 2004. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural networks*, *17*(1), pp.113-126.

Cleophas, T.J. and Zwinderman, A.H., 2011. Non-parametric tests. In Statistical Analysis of Clinical Data on a Pocket Calculator (pp. 9-13). Springer Netherlands.

Ermagun, A. and Levinson, D., 2016. Spatiotemporal Traffic Forecasting: Review and Proposed Directions. *Transport Reviews*, pp.1-29.

Friedman, J.H., Bentley, J.L. and Finkel, R.A., 1977. An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software (TOMS), 3(3), pp.209-226.

Goldberger, J., Hinton, G.E., Roweis, S.T. and Salakhutdinov, R.R., 2005. Neighbourhood components analysis. In Advances in neural information processing systems (pp. 513-520).

Guo, J., Smith, B.L., 2007. *Short Term Speed Variance Forecasting Using Linear Stochastic Modeling of Univariate Traffic Speed Series (Final Report No. UVACTS-15-17-10).* University of Virginia Center for Transportation Studies, Charlottesville, US.

Halim, M., 2018. *Improving Traffic Forecasting During Congestion (By Predicting the Presence of Missing Data)*. MSc. University College London.

Han, L., Shuai, M., Xie, K., Song, G. and Ma, X., 2010, June. Locally kernel regression adapting with data distribution in prediction of traffic flow. In *Geoinformatics, 2010 18th International Conference on* (pp. 1-6). IEEE.

Harikrishnan, K.P., Jacob, R., Misra, R. and Ambika, G., 2017. Determining the minimum embedding dimension for state space reconstruction through recurrence networks. *arXiv preprint arXiv:1704.08585*.

Hassanat, A.B., Abbadi, M.A., Altarawneh, G.A. and Alhasanat, A.A., 2014. Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach. arXiv preprint arXiv:1409.0919.

Haworth, J., 2014. *Spatio-temporal forecasting of network data*. PhD. University College London.

Haworth, J. and Cheng, T., 2012. Non-parametric regression for space–time forecasting under missing data. *Computers, Environment and Urban Systems*, 36(6), pp.538-550.

Haworth, J., Cheng, T., Tsapakis, I. and Bolbol, A., 2012. Inferring hybrid transportation modes from sparse GPS data using a moving window SVM classification. Computers, Environment and Urban Systems, 36(6), pp.526-537.

Hosking, J.R., 1984. Modeling persistence in hydrological time series using fractional differencing. Water resources research, 20(12), pp.1898-1908.

Hsu, C.W., Chang, C.C. and Lin, C.J., 2003. A practical guide to support vector classification.

Hurt, G., 2014. *TfL ANPR Cameras and Data - a Freedom of Information request to Transport for London*. [online] WhatdoTheyKnow.com. Available at: https://www.whatdotheyknow.com/request/tfl_anpr_cameras_and_data [Accessed 29 Mar. 2018].

Hyndman, R.J., Khandakar, Y., 2007. *Automatic time series for forecasting: the forecast package for R.* Available from: https://www.jstatsoft.org/article/view/v027i03/ [Accessed: 20/04/18].

Kohavi, R., 1995, August. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Ijcai (Vol. 14, No. 2, pp. 1137-1145).

Kumar, S. and Vanajakshi, L., 2015. Short-term traffic flow prediction using seasonal ARIMA model with limited input data. *European Transport Research Review*, 7(3).

Leng, Z., Gao, J., Qin, Y., Liu, X. and Yin, J., 2013, May. Short-term forecasting model of traffic flow based on GRNN. In *Control and Decision Conference (CCDC), 2013 25th Chinese*(pp. 3816-3820).

Lin, S.W., Ying, K.C., Chen, S.C. and Lee, Z.J., 2008. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert systems with applications*, *35*(4), pp.1817-1824.

Lin, L., Li, J., Chen, F., Ye, J. and Huai, J., 2017. Road Traffic Speed Prediction: A Probabilistic Model Fusing Multi-Source Data. *IEEE Transactions on Knowledge and Data Engineering*, pp.1-1.

Liu, X., Fang, X., Qin, Z., Ye, C. and Xie, M., 2010. A Short-Term Forecasting Algorithm for Network Traffic Based on Chaos Theory and SVM. *Journal of Network and Systems Management*, 19(4), pp.427-447.

Lu, C.J., Lee, T.S. and Chiu, C.C., 2009. Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems*, *47*(2), pp.115-125.

Lv, Y., Chen, Y., Zhang, X., Duan, Y. and Li, N., 2017. Social media based transportation research: the state of the work and the networking. *IEEE/CAA Journal of Automatica Sinica*, 4(1), pp.19-26.

Maltamo, M. and Kangas, A., 1998. Methods based on k-nearest neighbor regression in the prediction of basal area diameter distribution. Canadian Journal of Forest Research, 28(8), pp.1107-1115.

McLeod, A.I., 1993. Parsimony, model adequacy and periodic correlation in forecasting time series, International Statistical Review 61/3, 387-393.

Mitchell, T.M., 1997. Machine learning. 1997. Burr Ridge, IL: McGraw Hill, 45(37), pp.870-877.

Nguyen, D.H. and Le, M.T., 2014. Improving the Interpretability of Support Vector Machines-based Fuzzy Rules. *arXiv preprint arXiv:1408.5246*.

Ojemakinde, B.T., 2006. Support Vector Regression for Non-Stationary Time Series.

Pardo, M. and Sberveglieri, G., 2005. Classification of electronic nose data with support vector machines. *Sensors and Actuators B: Chemical*, *107*(2), pp.730-737.

Papernot, N. and McDaniel, P., 2018. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. *arXiv preprint arXiv:1803.04765*.

Pfeifer, P. and Deutsch, S., 1980. A STARIMA Model-Building Procedure with Application to Description and Regional Forecasting. *Transactions of the Institute of British Geographers*, 5(3), p.330.

Poonia, P., Jain, V. and Kumar, A., 2018. Short Term Traffic Flow Prediction Methodologies: A Review. *Mody University International Journal of Computing and Engineering Research*, 2(1), pp.37-39.

Shalev-Shwartz, S. and Srebro, N., 2008, July. SVM optimization: inverse dependence on training set size. In Proceedings of the 25th international conference on Machine learning (pp. 928-935). ACM.

Smola, A.J. and Schölkopf, B., 2004. A tutorial on support vector regression. *Statistics and computing*, *14*(3), pp.199-222.

Smith, B.L. and Demetsky, M.J., 1997. Traffic flow forecasting: comparison of modeling approaches. Journal of transportation engineering, 123(4), pp.261-266.

TfL., 2009. Transport for London Congestion Charge Factsheet. [ebook] Available at: http://content.tfl.gov.uk/congestion-charge-factsheet.pdf [Accessed 13 Apr. 2018].

Transport for London., 2015. Roads Task Force - Technical Note 14. *Thematic Analysis.*

V. N. Vapnik, The Nature of Statistical Learning Theory. Springer, 1995.
Vlahogianni, E., Karlaftis, M. and Golias, J., 2014. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, 43, pp.3-19.

Yu, Z., Sun, T., Sun, H. and Yang, F., 2015. Research on Combinational Forecast Models for the Traffic Flow. *Mathematical Problems in Engineering*, 2015, pp.1-10.

Zhang, M.L. and Zhou, Z.H., 2007. ML-KNN: A lazy learning approach to multi-label learning. Pattern recognition, 40(7), pp.2038-2048.

Zhang, Y. and Liu, Y., 2009. Comparison of parametric and nonparametric techniques for non-peak traffic forecasting. *World Academy of Science, Engineering and Technology*, *51*(27), pp.8-14.

Zeng, Z.Q., Yu, H.B., Xu, H.R., Xie, Y.Q. and Gao, J., 2008, November. Fast training support vector machines using parallel sequential minimal optimization. In *Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on* (Vol. 1, pp. 997-1001). IEEE.

Zheng, Z. and Su, D., 2014. Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. *Transportation Research Part C: Emerging Technologies*, *43*, pp.143-157.