Imperial College London

BEng Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

Active Delay Bus Journey Modelling

Author:
Serene Chongtrakul

Supervisor:
Peter McBrien

2nd Marker:
Thomas Lancaster

June 16, 2020

Submitted in partial fulfillment of the requirements for the Joint Mathematics and Computing BEng of Imperial College London

Contents

1 Introduction

1.1 Motivation

For the year ending in June 2019, the number of local bus passenger journeys in London decreased by 0.9% [1]. This was the 4th consecutive year in which passenger journeys on buses in London have fallen. Transport for London (TfL) attributes this decrease in bus popularity to a downward trend in bus performance, for example, slower bus speeds or unreliable bus arrival times [2]. The latter issue is what this project will focus on.

It is important that more people use buses as a means of transportation and reduce their use of private vehicles. London's population is set to expand from 8.7 million in 2017 to 10.5 million by 2041, generating more than 5 million additional trips each day across the transport network [3]. Traffic congestion is increasing and air quality is worsening. Although Sadiq Khan, who was elected the London Mayor in 2016, has taken steps to try to encourage more use of public transportation, for example by introducing Ultra Low Emission Zone (ULEZ) charges in 2019, as of February 2020 the car is still the most popular single mode of travel for Londoners, with 29.8 % of workers opting to commute in this manner [4]. More than 50% of London's toxic air pollution is caused by vehicles and more than 2 million Londoners live in areas that exceed legal limits for NO2 [5]. Therefore, encouraging bus use through improving the experience of using a bus, will help to improve the air quality and the environment overall.

1.2 Aims

This project aims to investigate various journey time prediction models and implement the best performing model in a simple mobile app. The mobile app serves merely as a way of displaying the model's results. This project also aims to study various parameters that could affect the journey time of a bus, ranging from time of day to pre/post COVID-19 lockdown. One of the main objectives is to actively react to delays that occur in the bus network and provide predictions that reflect this.

Although there has been a fair amount of research that has gone into the topic of bus journey time modelling, many of them are based on static models [6], [7], [8]. This project will focus on dynamic models because it is hypothesised that recent information on bus journey times has more of an impact on predictions than merely using static data. Of course dynamic modelling has been researched [9], [10], [11]. However, these papers tend to use GPS co-ordinates to aid in their predictions, which is not information that is explored and used in this project.

1.3 Overview

The models that will be developed to predict the arrival times of the buses will make use of TfL's publicly provided APIs [12]. Currently TfL provides a variety of APIs including ones that describe bus routes or bus stop locations. In fact, TfL also provides an arrival prediction API *Countdown,* for both bus routes and tube lines. However, it can

be inferred that this API uses the GPS position of a bus alongside the timetabled arrival time to make its prediction. This project does not have access to GPS positions of buses in London and therefore, the models are developed using other means. Countdown's predictions are used as the baseline model for which to compare the models developed in this project against.

Historical average models, regression models, interpolation models, Kalman filter models and artificial neural network models, are explored however, only the first three models are implemented and investigated further. Historical average models were chosen because they are generally suitable for real time predictions, due to the more simplistic nature of the algorithms and smaller computation time [10]. Regression and interpolation models were chosen because they are able to work satisfactorily if traffic conditions are unstable. It was initially planned to implement artificial neural network models too since many papers demonstrated that artificial neural network models outperform historical and regression models [6]. However, neural networks are time intensive and so may not be as suitable to be implemented in a mobile app as the other models, therefore, it was not prioritised. Due to lack of time, the model was then never implemented. Furthermore, research has shown that artificial neural networks have the potential to overfit and so results obtained might not generalise well to routes not used in training [10]. Kalman models were not considered to be implemented because of time constraints and because the mathematical background required was unfamiliar.

It is hypothesised that a number of factors including traffic conditions, weather conditions and time of day, can cause a bus to deviate from its scheduled journey time. The factors that have occurred most often in the research papers are explored further in order to decide which ones are more important. The time of day and day of week are implemented in the models developed as they were found to be statistically significant [13].

If this study is successful, new insights into methods used for modelling bus journey times will have been discovered. On the more ambitious side, the end product should be a model that predicts the arrival time of any bus on any route in London more accurately than the Countdown API. Bus passengers want reliable and timely services, but if that is not possible, they want to know how long it will take them to reach their destination. Passengers want to be able to take the bus if their journey time would be shorter than normal or avoid taking the bus if the journey will be significantly longer than usual. By providing a service that supplies users with reliable bus journey times, it is intended that more people will consider the bus over the car as their primary mode of transport.

However, it is entirely plausible that the dynamic models could prove less successful than the static model used by Countdown. If that is the case, a thorough evaluation will be done to investigate why. The success of the model is measured by the error of its predictions compared to the actual arrival times and compared to Transport for London's own predictions.

2 Background

There are a number of models that can be used for bus arrival prediction, but there are four main models that are most widely used [10]. They are as follows:

- **Historical average models**: Also known as simple forecasting, these models make use of historical bus arrival and travel times to predict the arrival time of buses. On its own, performance could be fairly weak, due to the unreliability of traffic conditions and other factors that vary over time. For example, Jeong and Rilett found that their historical model was outperformed by their artificial neural network model [6].

- **Regression models**: These models are a form of predictive technique, which can be used to investigate the relationship between factors that affect bus arrival times (predictors) and bus arrival times (target) [14]. This technique will help to see which factors are more or less important. For example, Patnaik et al. used regression models to discover that "weather variables were not among the significant factors for estimating arrival times" [13].

- **Kalman Filtering models**: The Kalman filter is one of the optimal solutions for tracking and data prediction tasks. These models are a dynamic estimation technique that have been used fairly frequently in bus delay predictions. However, as Sun et al. found, these models are expensive to compute [11].

- **Artificial Neural Network models**: These models have been widely used in research to try to improve public transportation. They are suited towards more complex and noisy data and finding non linear relationships between dependent and independent variables [10].

In this project, data that is used for the modelling is collected from TfL's publicly available APIs.

2.1 Transport for London APIs

Across the entire London transport system there are over 19,000 bus stops, 700 bus routes, 8,000 buses and approximately 130,000 bus arrival predictions at any point in time [15]. However, Transport for London (TfL) does not provide an API that gives exact bus arrival times and so this information has to be inferred from the Countdown API which provides estimated arrival times [12].

2.1.1 Countdown API

The Countdown API provides real time predicted arrival times for buses in London. The expected arrival times are updated at the Countdown source every 30 seconds, so there is some room for error in regards to the inferred arrival time of the buses. This also means that there is no point in requesting information more frequently than every 30 seconds.

Depending on the request, the response can be made up of 5 different array types: `Stop`, `Prediction`, `Flexible`, `Message` and `URA Version` arrays. The sequence of these arrays in the response is undefined except that the `URA Version` array always appears first. For the purpose of this project, the arrays that are relevant and will be returned are the `URA Version` array and the `Prediction` array.

An example request where Countdown was called for bus route 9 and stop code 490010357F (commonly known as NORTH END ROAD) can be seen in Figure 1. The following parameters are set:

- stop code 2: the unique national identifier of the bus stop. Not to be confused with stop code 1 which is the public code for the bus stop displayed on the bus stop flag.

- line name: the route number displayed on the front of the bus.

```
http://countdown.api.tfl.gov.uk/interfaces/ura/instant_V1
Stopcode2=490010357F&LineName=9&
ReturnList=StopPointName,LineName,DestinationText,EstimatedTime,ExpireTime,VehicleID,DirectionID
```

Figure 1: Countdown API request

Figure 1 also demonstrates that users can choose what parameters Countdown should return in the response. For this project the following fields are set and returned in the `Prediction` array:

- `response type`: this is returned by default and is always 1 for the `Prediction` array.

- `stop name`: name of the bus stop.

- `line name`: the route number displayed on the front of the bus.

- `direction id`: 1 = outbound or 2 = inbound.

- `destination text`: the full length destination name of the trip the vehicle is on.

- `vehicle id`: the unique identifier of the vehicle.

- `estimated arrival time`: absolute time in UTC as per Unix epoch (in milliseconds).

- `expiry time`: time at which the corresponding prediction is no longer valid and should no longer be used. In this case it is the estimated arrival time + 30 seconds.

The response also contains the timestamp that the response was processed in the `URA Version` array.

The Countdown API returns a JSON of the form `[[URA array],[Prediction Array]]`. An example response where Countdown was called for bus route 9 and stop code 490010357F

6

(commonly known as NORTH END ROAD) is seen in Table 1 and Table 2 (the JSON information has been displayed in a table for ease of reading). Countdown was called at: Friday 15th May at 07:02:06 (time A).

Response Type	URA Version	Time that response was processed
4	"1.0"	May 15, 2020 07:02:06.913

Table 1: `URA Version` Array Route 9 at NORTH END ROAD at Time A

Response Type	Stop Name	Line Name	Direction ID	Destination Text	Vehicle ID	Estimated Arrival	Expiry Time
1	"North End Road"	"9"	2	"Aldwych"	14510	May 15, 2020 07:01:57	May 15, 2020 07:02:27
1	"North End Road"	"9"	2	"Aldwych"	15448	May 15, 2020 07:10:02	May 15, 2020 07:10:32

Table 2: `Prediction` Array Route 9 at NORTH END ROAD at Time A

When Countdown was called to give the expected arrival time on route 9 for NORTH END ROAD, it was also called at the same time to give the response for KENSINGTON OLYMPIA STATION. Table 3 shows the `Prediction Array` for bus route 9 and stop code 490008652A (commonly known as KENSINGTON OLYMPIA STATION). Note that KENSINGTON OLYMPIA STATION is the immediate station after NORTH END ROAD in the 9 route to "Aldwych". Comparing Table 2 and Table 3, it can be seen that the two same vehicles appear in both responses. This response can be interpreted visually as in Figure 2.

Response Type	Stop Name	Line Name	Direction ID	Destination Text	Vehicle ID	Estimated Arrival	Expiry Time
1	"Kensington Olympia Station"	"9"	2	"Aldwych"	14510	May 15, 2020 07:02:49	May 15, 2020 07:03:19
1	"Kensington Olympia Station"	"9"	2	"Aldwych"	15448	May 15, 2020 07:10:54	May 15, 2020 07:11:24

Table 3: `Prediction` Array Route 9 at KENSINGTON OLYMPIA STATION at Time A

When Countdown is called again at Friday 15th May at 07:02:43 (time B) for both NORTH END ROAD and KENSINGTON OLYMPIA STATION, the new response can be seen visually interpreted in Figure 3. It can be seen that bus 14510 has now arrived at and passed NORTH END ROAD and is on its way to KENSINGTON OLYMPIA STATION. This is indicated by the JSON response shown in Table 4 where bus 14510 is no longer returned in the `Prediction Array` for Route 9 at NORTH END ROAD.

Response Type	Stop Name	Line Name	Direction ID	Destination Text	Vehicle ID	Estimated Arrival	Expiry Time
1	"North End Road"	"9"	2	"Aldwych"	15448	May 15, 2020 07:10:02	May 15, 2020 07:10:32

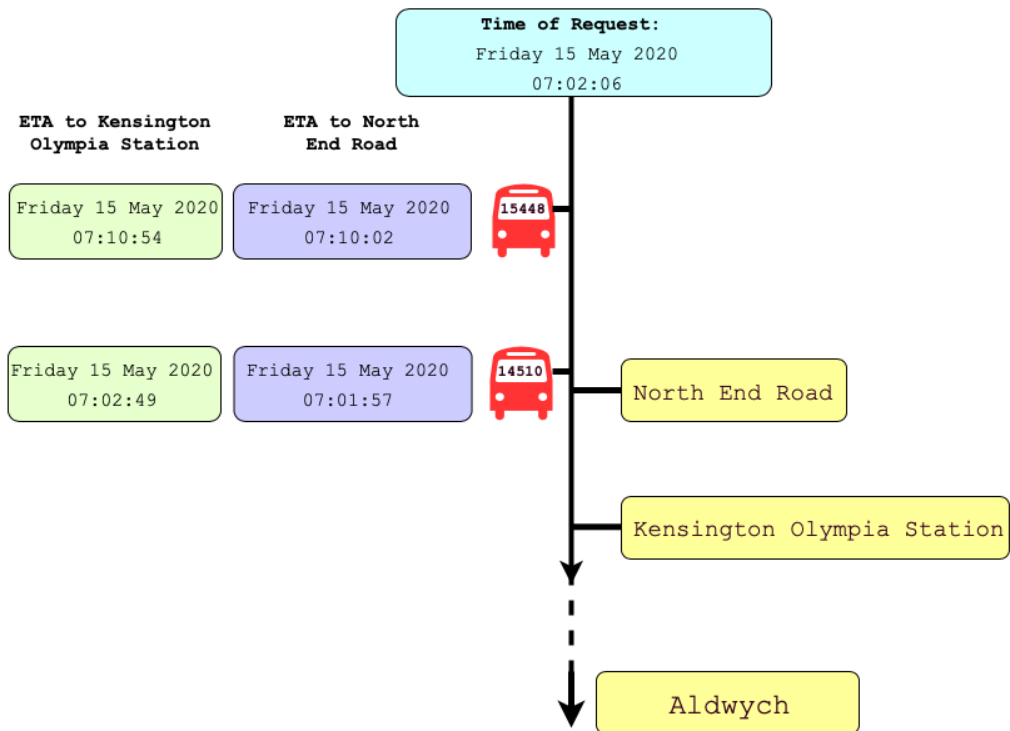Table 4: `Prediction` Array Route 9 at NORTH END ROAD at Time B
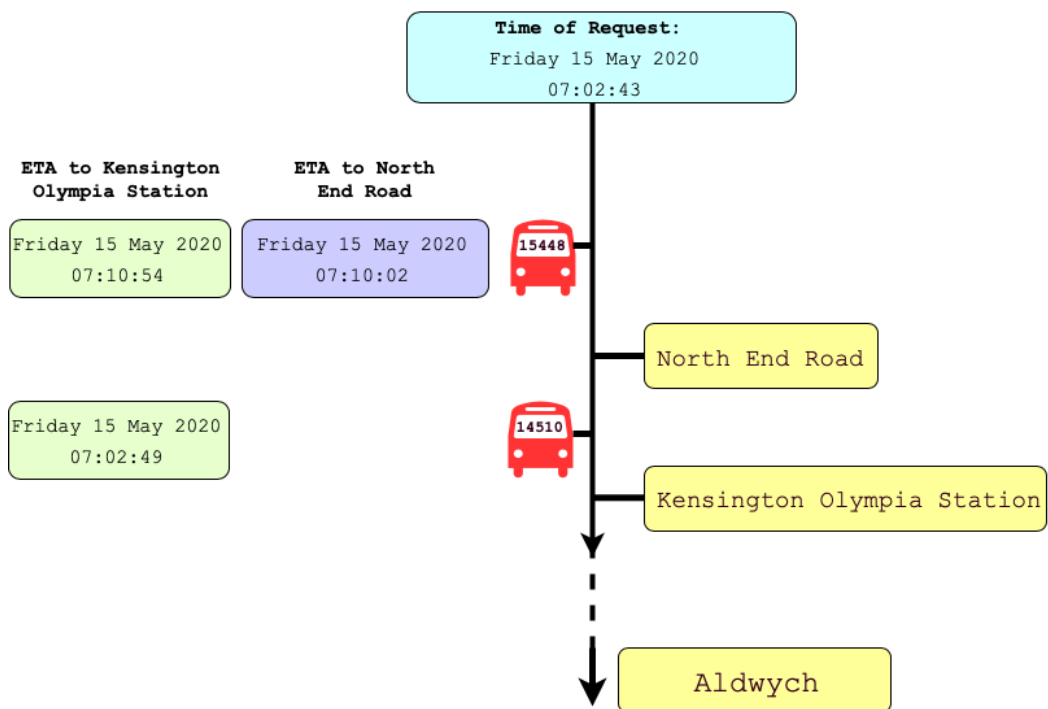
Figure 2: Countdown API response at time A

Figure 3: Countdown API response at time B

2.1.2 Stoppoints API

This API is called to give information on all bus stops for a particular route. A request takes the form:

$$https://api.tfl.gov.uk/line/bus\_route\_id/stoppoints.$$

The response returns an array of stops. For each stop, information includes bus stop name, bus stop letter, all lines/routes serving that stop, facilities at the stop, including toilets, WiFi, phone number etc. and latitude and longitude. However, this API returns more stop IDs than there actually are stops on a route. When put as parameters into the Countdown API call, some of the stops return estimated arrival times as expected, whereas others return HTTP errors. A possible explanation for this could be differences in routing or bus stop additions/deletions over the years. Therefore, it is useful to verify which stop ids are valid and which are invalid.

2.1.3 Stoppoints Sequence API

This API is called to give the list of stops in the order that they are visited for a particular route. It can be called for both bus and underground routes. A request takes the form:

$$https://api.tfl.gov.uk/Line/bus\_route\_id/Route/Sequence/outbound$$

For each of the stops returned in the list, the response also gives the interchanges with other lines and modes. The response also returns the latitude and longitude showing the path of the route so it can be plotted on a map.

2.2 Historical Average Models

Historical average models or forecasting models have some of the simplest algorithms. This tends to lead to shorter computation times, which in this case, could be very useful because of the sheer volume of data across London's bus network. However, unless the bus journey times are fairly constant in history, it is easy for the model performance to be weaker than other types of models.

2.2.1 Forecasting

Forecasting is an important aid to effective and efficient planning. Statistical forecasting typically refers to the use of statistics on historical data to project what could happen in the future [16]. The effectiveness of forecasting as a method depends on several factors including: how well we understand the factors that contribute to an event, how much data is available and whether the forecasts can affect the thing we are trying to forecast [17]. A good forecast should capture the genuine patterns and relationships which exist in the historical data, but do not replicate past events that will not occur again. In the case of historical bus data, the data collected will be gathered at regular intervals over time, and therefore, can be seen as time series data. In this way, quantitative forecasting can be applied because it is reasonable to assume that some aspects of the past patterns will continue into the future.

2.2.2 Methods

When forecasting time series data, the aim is to estimate how the sequence of observations will continue into the future. The simplest time series forecasting methods only use information on the variable to be forecast without attempting to discover the factors that could affect its behaviour. Therefore, these methods will extrapolate trends and seasonal patterns, but will ignore other information. For example, when forecasting bus arrival times, we would look at historical information on bus delays, but would ignore factors such as traffic conditions or weather conditions.

Some of the most common simple forecasting methods are listed below [17]:

- **Mean Method**: All the future forecast values are equal to the average of the historical data. This method is optimal when data has a fairly steady value and does not change much over time.

- **Naive Method**: All the future forecast values are equal to the value of the last observation. This method is optimal when data follows a random walk.

- **Seasonal Naive Method**: Each forecast value is set to be equal to the last observed value from the same season of the year, for example from the same month of the previous year or the same quarter of the previous year. This method tends to perform better than a simple naive method, especially when the historical data has a clear seasonal trend.

- **Drift Method**: This method is a variation of the naive method. The forecast values increase or decrease over time and the amount of change over time is called the drift. This drift is set to be the average change seen in the historical data. This method is equivalent to drawing a straight line between the first and last observations and extrapolating it into the future.

Figure 4 shows what the *mean*, *naive* and *seasonal naive* methods look like when applied to quarterly beer production data. It can be seen that both the *mean* and *naive* methods result in flat line forecasts, and therefore, would not be particularly suitable for bus arrival predictions because research has shown that for different time periods, the trends in journey times vary [10], [6], [11]. However, a slight variation on the mean and naive method that essentially combines the two methods, where the model takes the average of the last two or three bus travel times for each forecast, is much more likely to provide a better estimate. This is because this method will use historical results that should theoretically be in the same time period as the current time to be forecast. This method will also avoid going too far into the past and using data that may not be relevant. Figure 4 also shows that the *seasonal naive* method results in an oscillating prediction. Since there is reason to believe that historical data on bus delay times show an up and down trend, the seasonal naive method could also be an equally suitable model to use.
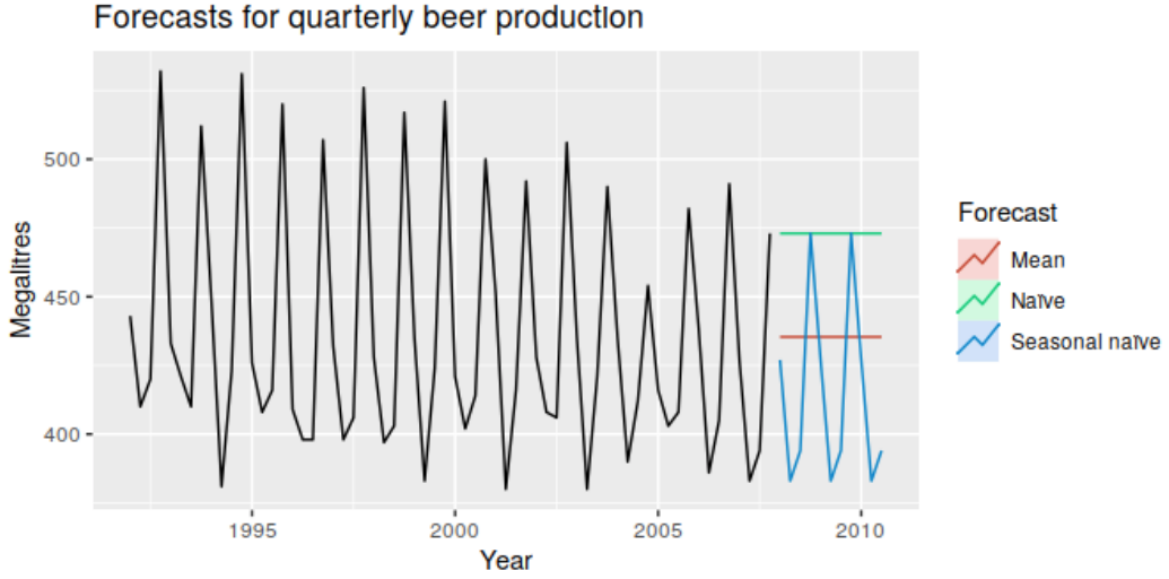
Figure 4: Example of the mean, naive and seasonal naive methods, original image from [17]

2.2.3 Measuring Success

A forecast may be greater than or less than the actual value. The difference between this predicted value and the actual value is called the forecast error. Equation 1 shows the forecast error for an individual forecast [18].

$$\text{FE}_t = A_t - F_t \tag{1}$$

where FE_t is the forecast error, A_t is the actual value and F_t is the forecasted value.

Measures to evaluate the performance of a forecasting method can be classified into those that look at direction and those that look at size of the error. The bias (Equation 2) is the most popular measure for evaluating the direction of the error, i.e. whether the predictions are over or under the actual values. The closer the bias is to zero, the better the model. If the bias is negative, this implies that the actual values are on average, smaller than the predicted value. Therefore, the forecasting model overestimates the actual values. Let n be the number of samples, then:

$$Bias = \frac{\sum (A_t - F_t)}{n} \tag{2}$$

The size of the error could mean the amount of the error or the dispersion of the error or the relative magnitude of the error. One of the most popular measures of the size of the error is the mean squared error (MSE). The MSE (Equation 3) is the average of the sum of squares of the forecast errors and measures the amount of dispersion in

11

the error. The smaller the value, the better.

$$MSE = \frac{\sum(A_t - F_t)^2}{n} \tag{3}$$

Another common measure is the root mean squared error (RMSE), which is, as its name suggests, the square root of the MSE. The RMSE measures how spread out the residuals are, where the residuals are how far away from the regression line the data points are. More intuitively, the RMSE describes how concentrated the data is around the line of best fit. However, neither the MSE nor the RMSE take into account the magnitude of the actual values.

A measure of success that does attempt to take into account the effect of the magnitude of the actual values is the mean absolute error (MAE). The MAE (Equation 4) is the average of the absolute values of the errors. As with the RMSE and MSE, the smaller the value, the better.

$$MAE = \frac{\sum|A_t - F_t|}{n} \tag{4}$$

The RMSE is bounded by the MAE:

- MAE $\leq$ RMSE: The RMSE will always be larger or equal to the MAE. If all of the errors have the same magnitude, then the two values are equal.

- RMSE $\leq$ (MAE * $\sqrt{n}$): The difference between the RMSE and the MAE is greatest when all of the prediction error comes from a single test sample. This implies that as the sampel size increases, the RMSE tends to become increasingly larger than the MAE.

2.3   Regression Models

Regression is a form of statistical modelling that investigates the relationship between a dependent (target) and independent (predictor) variable [14]. When performing statistical modelling, we observe some data y, known as the dependent variable, and interpret each data point as a realisation of some random variable Y, following some probability distribution. A statistical model is a specification of the distribution of Y up to an unknown parameter θ. Often $y = (y_1, y_2, ..., y_n) \in \mathbb{R}^n$ is a vector and $Y = (Y_1, ..., Y_n)$ is a random vector. The distribution of $Y_1, ..., Y_n$ may depend on (non random or deterministic) quantities $x_1, ..., x_n$ known as covariates or predictors.

Regression models are suitable when the variable that is trying to be predicted relies on a number of different independent factors. This is a very important point to ensure since non-independence is a reason for sub-optimal predictions in linear models. Since it is hypothesised that the journey time of a bus relies on many things such as weather conditions, time of day, traffic conditions and so on, this makes regression models a good choice to implement. Regression models also help to see which of these independent variables are statistically significant, and so can provide insight into which of the factors that are being considered are worth being explored further. However, the viability of

regression models is limited by the hypothesis that many of the factors affecting bus journey times are in fact correlated to one another and not independent after all.

2.3.1 Linear Regression

In the simplest case, the regression model allows for a linear relationship between the observable variable Y and a single predictor variable x (Equation 5):

$$Y_t = \beta_1 + x_i \beta_2 + \epsilon_t, t = 1, ..., n \tag{5}$$

where Y_t is the outcome or observable random variable, x_t is the covariate constant, β_0 and β_1 denote the intercept and slope of the line respectively and ϵ_t are iid (independent and identically distributed) errors. $E(\epsilon_t) = 0$, $Var(\epsilon_t) = \sigma^2$ for $t = 1, ...n$. $\sigma^2 > 0$ is also unknown. The errors are not observable and can be thought of as the deviation from the underlying straight line model, capturing anything that may affect Y_t other than x_t [17]. Figure 5 visualises the linear relationship as defined in Equation 5.
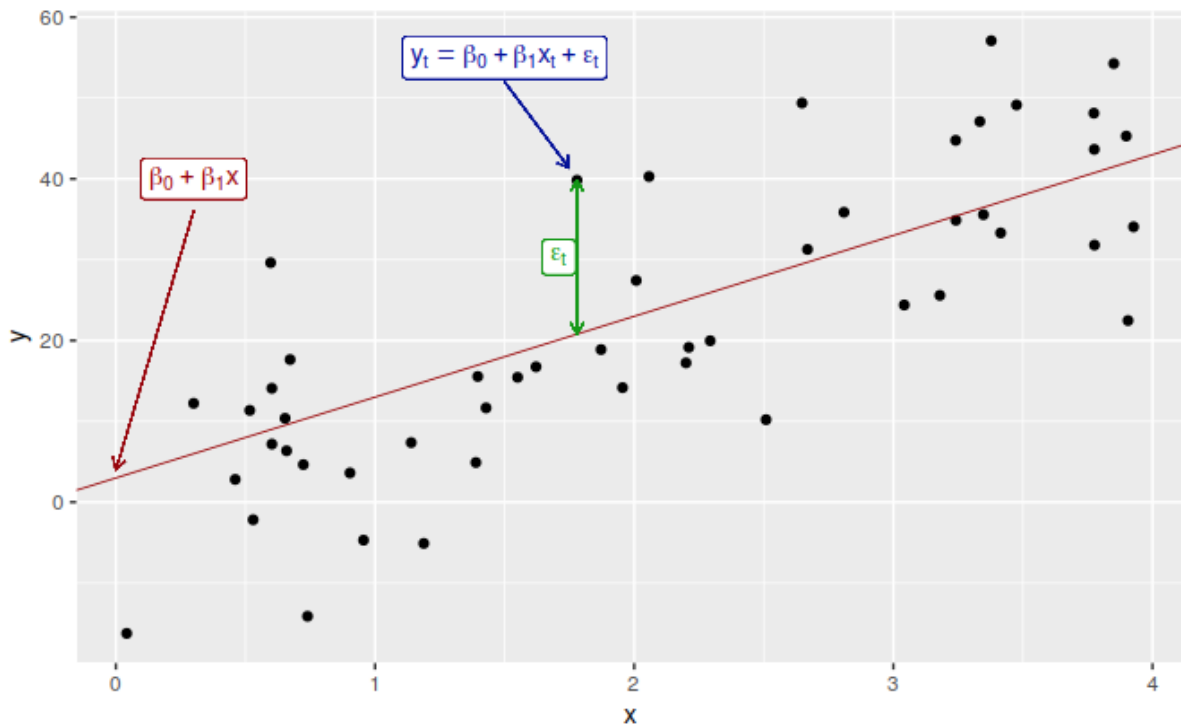

Figure 5: Linear Regression, original image from [17]

2.3.2 Multiple Linear Regression

When there are two or more predictor variables, the model is called a multiple regression model. The general formulation in a linear model is shown in Equation 6:

$$Y = \mathbf{X}\beta + \epsilon \tag{6}$$

13

where Y is an n-dimensional random vector (observable), $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a known matrix (often called the 'design matrix'), $\beta \in \mathbb{R}^p$ is an unknown parameter and ϵ is an n-variate random vector (not observable) with $E(\epsilon) = \mathbf{0}$. The β coefficients essentially measure the effect of each predictor after taking into account the effects of all the other predictors in the model.

When we use a linear or multiple regression model, we make the following assumptions.

- The model is a reasonable approximation to reality, that is, the relationship between the forecast variable and the predictor variable satisfies this linear equation.

- We make the following assumptions about the errors:
    - they have a mean of zero; otherwise the predictions will be systematically biased.
    - they are not autocorrelated; otherwise the predictions will be inefficient.
    - they are unrelated to the predictor variables; otherwise there would be more information that should be included as a predictor variable.

- Each predictor x is not a random variable.

2.3.3   Pearson Correlation Coefficient

The Pearson correlation coefficient, commonly written simply as ρ is the measure of the linear relationship between two variables [19], [20]. It is calculated as the covariance of the two variables divided by the product of the standard deviation of each data sample. In other words, it is the normalization of the covariance between the two variables. Given a pair of variables (X, Y), the Pearson correlation coefficient follows Equation 7:

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \tag{7}$$

where cov is the covariance, σ_X is the standard deviation of X and σ_Y is the standard deviation of Y.

The coefficient is bounded between -1 and 1, where values near 0 indicate no correlation. Positive values indicate a positive correlation, where the larger X gets, the larger Y also gets. Negative values indicate a negative correlation, where as X increases, Y decreases.

2.3.4   Measuring Success

The standard error is equal to the standard deviation, which would be obtained from repeatedly estimating the β coefficients on similar data sets. This gives a measure of the uncertainty in the estimated β coefficient.

The p-value is the probability of the estimated β coefficient being as large as it is if there was no real relationship between the forecast variable and the corresponding predictor. This is particularly useful when studying the effect of each predictor, and can be used as another way of seeing which parameter to use in the models and which to discard.

A common way to summarise how well a regression model fits the data is via the coefficient of determination (Equation 8:

$$R^2 = \frac{\sum(\hat{y}_t - \bar{y})^2}{\sum(y_t - \bar{y})^2} \tag{8}$$

where the summations are over all the observations, $\hat{y}_t$ are the predicted values, y_t are the observed values and R^2 is bounded between 0 and 1. Although this is a way of measuring 'goodness-of-fit', validating the performance of a model against a test set is much better than measuring the R^2 value on the training data [17].

The differences between the observed y values and the corresponding fitted $\hat{y}$ values are known as residuals. To detect problems with a model, we can plot standardised residuals against some each of the predictor variables. We expect the residuals to be randomly scattered without showing any systematic pattern because a pattern would indicate a nonlinear relationship between the variables. If the model is correct then the resulting plot should just show 'noise' with no distinct patterns.

As with the historical models, the mean absolute error (MAE) and the root mean squared error (RMSE) can also be used as measures of success.

2.4   Kalman Filtering Models

Kalman filtering is a dynamic two-step algorithm that uses measurements observed over time to estimate some unknown variables [21]. The first part is the prediction process and the second part is the measurement process. The two processes are modelled by groups of equations in the state space model. Kalman filter models generally provide good predictions because they are able to optimally estimate the system's error covariance and recursively use the prediction to improve the system measurements [22]. In this case, the journey time of a future bus would be predicted based on the system's state in past time steps. Kalman Models also have great capabilities of filtering out noise.

Kalman filter models would be suitable for the task of estimating bus arrival times because unstable traffic conditions can be accommodated by their time dependent parameters [10]. Furthermore, Kalman filter models are able to make predictions based purely on journey times, without the need for other traffic conditions. However, Kalman filter models struggle to handle abrupt changes between consecutive travel times. So, if there was a significant difference in travel times between the morning peak and late morning hours, the model is unable to filter the noise smoothly.

2.5 Artificial Neural Network Models

(Feed-forward) Artificial Neural Networks (ANNs) are a type of machine learning inspired by real world biological neurons.

ANNs have the ability to solve complex non-linear relationships and can capture subtle relationships between the data, even if the underlying relationships are unknown [10]. Therefore, ANNs are a popular choice for bus prediction models because it can learn the non-linear correlations between bus travel times. If the ANN has enough neurons in the hidden layers, then it is very capable of arbitrary input-output matching [23]. However, this also implies that ANNs have a high potential for over-fitting and therefore, could struggle to generalise to data not given in the training process. This over-fitting is also what makes ANNs so resource intensive. Furthermore, the back propagation algorithm, which is the most used algorithm for transportation problems, has a long learning process.

ANNs are also harder to interpret compared to historical models and regression models. For example, the coefficients gotten from the regression model describe the relative importance of each of the factors. However, the weights and biases gotten after training a neural network model are not as easy to understand intuitively.

2.5.1 Perceptrons

An ANN has a collection of nodes called artificial neurons, the most common type of which is called a perceptron [24], [25]. A perceptron takes $m + 1$ inputs $x_{i \in 0, \dots, m}$ with $m + 1$ weights $w_{i \in 0, \dots, m}$ and produces a single binary output y. Weights are real numbers expressing the importance of the respective inputs to the output. So a perceptron's output, 0 or 1, is determined by whether the weighted sum is less than or greater than some threshold value (Equation 9).

$$output = \begin{cases} 0 \text{ if } \sum_j w_j x_j \leqslant \text{threshold} \\ 1 \text{ if } \sum_j w_j x_j > \text{threshold} \end{cases} \tag{9}$$

By varying the weights and the threshold, we can get different models of decision making. We set the bias of a perceptron to be w_0 and set $x_0 = 1$. The bias is a measure of how easy it is to get the perceptron to output a 1. The perceptron can be 'activated' using a non linear activation function g, allowing us to approximate arbitrarily complex function.

There are a number of different activation functions, with some common ones including the *Sigmoid* function: $\sigma(x) = \frac{1}{1+e^{-x}}$ and the *ReLU* function: $max(0, x)$. Figure 6 shows a single perceptron with 3 input neurons ($x_0 = 1$ and w_0 is the bias) and output $\hat{\mathbf{y}}$.
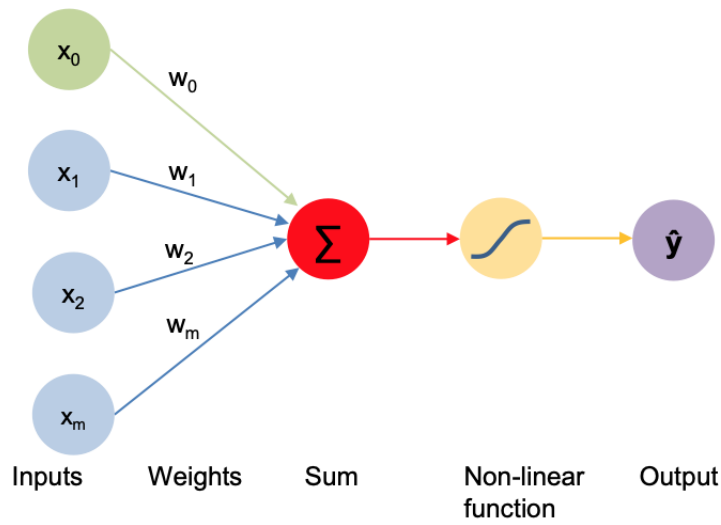
Figure 6: The Perceptron: forward propagation, original image from [24]

2.5.2 Architecture

ANNs consists of stacking neurons into layers, with different layers performing transformations on their inputs. These models are called feed-forward because information flows from input x to output y without any feedback connections in which outputs of the model are fed back into itself, i.e. information always travels forward [26]. Figure 7 shows a fully connected neural network with a single layer. The leftmost layer is called the input layer, the rightmost layer is called the output layer and the middle layers are called hidden layers. Neural networks with multiple hidden layers are sometimes called multi-layer perceptrons.
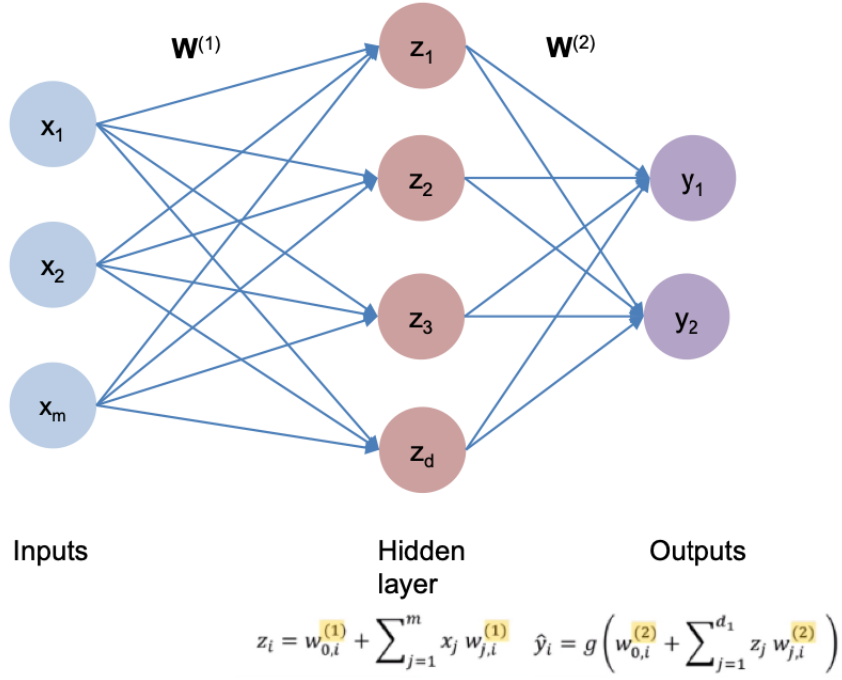
$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,i}^{(1)} \qquad \hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} z_j \, w_{j,i}^{(2)} \right)$$

Figure 7: Fully connected single layer neural network, original image from [24]

2.5.3 Back Propagation

The desired output from the output layer neurons is known. During the training process, the network uses the given examples to adjust the weights between the neurons to produce the known output. The error in the output is propagated back through the layers to help in the weight adjustment decisions [6].

2.5.4 Regularization and Overfitting

Regularization is implemented to constrain the optimization problem, ensuring that our model can generalize to unseen data and not overfit. One method of regularization is to implement dropouts. This randomly sets some of the neurons to 0, typically 50% of neurons in each layer, preventing reliance on a single node [24]. Another regularization method is early stopping. Here, we stop training the model as soon as the desired accuracy is reached or as soon as overfitting is detected. Generally, the variance or error increases in overfitted models.

2.6 Numerical Interpolation and Approximation

Numerical interpolation is the process of attempting to fit a function to given data points [27]. In interpolation, the data is fitted exactly, while in approximation this is not the case. Interpolation is not appropriate when the data points are likely to have experimental errors (for example errors that occur when the data is being collected). Therefore,

since this project requires the bus data to be collected manually and thus there is room for human error, this is an argument against the use of an interpolation model. On the other hand, combining regression analysis with interpolation allows models to take into consideration the actual, fluctuating travel conditions and thus provide predictions with more flexibility [28].

2.6.1   Linear Interpolation

Linear interpolation models search for a straight line that passes through the given end points x_A and x_B as shown in Equation 10 [29].

$$X_i = (1 - \alpha)x_B + \alpha x_A \tag{10}$$

where α is the interpolation factor, varying from 0 to 1. The interpolated data is bound between x_A and x_B.

2.6.2   Polynomial Interpolation

Polynomial interpolation models search for a polynomial that fits exactly with the given information about a real valued function f of a single real variable x [30], assuming that there is always a unique polynomial of degree at most $n - 1$ passing through n data points. The corresponding polynomial is called the Lagrange interpolation polynomial. If f is differentiable, the corresponding polynomial, called the Hermite interpolation polynomial, may include values of the derivative of f at some finite set of points.

If the function values $f(x)$ are known for all x in a closed interval of the real line, then the aim of polynomial interpolation is to approximate the function f by a polynomial over this interval. However, if the function values $f(x)$ are only known for a finite set of points $x_0, ..., x_n$, then the aim of polynomial interpolation is to construct the unknown function f by seeking a polynomial p_n. Where p_n passes through the points with co-ordinates $(x_i, f(x_i)$ for $i = 0, ..., n$ on the (x, y) plane [30].

However, fitting a single polynomial to a large number of data points tends to lead to overfitting because the function oscillates between the data points [27]. Piece-wise interpolation attempts to solve this issue.

2.6.3   Piecewise Interpolation

Piecewise interpolation models attempt to fit a number of different low-degree polynomials to a large number of data points [27]. This eliminates excessive oscillations and non-convergence.

Subdivide $[a, b]$ into J equal subintervals $[x_{j-1}, x_j]$ for $j = 1, ..., J$ of length h. Let $h = \frac{b-a}{j}$ then
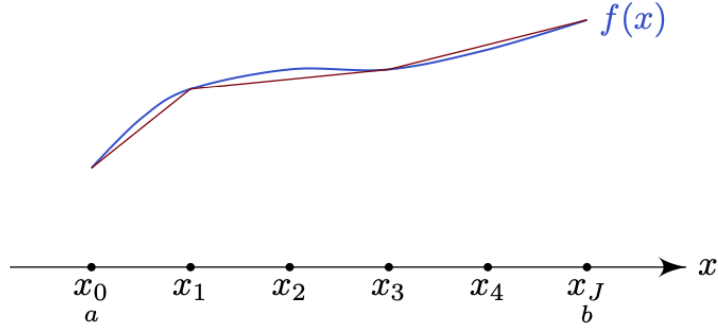
$$x_j = a + jh, j = 0, ..., J \tag{11}$$

Figure 8: Piecewise polynomial interpolation, original image from [31]

On each subinterval $[x_{j-1}, x_j]$ for $j = 1, ..., J$ linear interpolation is performed.

A problem with piecewise interpolation is that the graphs can have corners at which points the derivative is discontinuous. In other words, the interpolation is not as smooth.

2.6.4 Piecewise Polynomial Approximation

Divide the interval $[a, b]$ into a number of subintervals and for each subinterval, attempt to approximate it by a polynomial of low degree. These piece-wise polynomial approximations are called *splines* and the endpoints of the subintervals are called [30]. A spline of degree n, $n \geq 1$ is a function which is a polynomial of degree n or less in each subinterval. Splines have continuous derivatives of order up to k for $0 \leq k < n$. If the derivative of order n is required to be continuous everywhere, then the spline is just a single polynomial.

A particular example is a cubic spline. Consider the set S of all functions $s \in C^2[a, b]$ such that Equation 12 and Equation 13 are true.

$$s(x_j) = f(x_j), \ j = 0, ..., J \tag{12}$$

$$s \text{ is a cubic on } [x_{j-1}, x_j], \ j = 1, ..., J \tag{13}$$

Any element of S is referred to as an interpolating cubic spline. There is more than 1 interpolating cubic spline that satisfies the conditions. Indeed there are $4J$ coefficients of cubic polynomials and only $J + 1$ interpolating conditions and $3(J - 1)$ continuity conditions. Since s belongs to $C^2[a, b]$ this means that s, s' and s'' are continuous at the internal knots $x_1, ..., x_{J-1}$ [30].

A special case of the cubic spline is the natural cubic spline, which must also satisfy Equation 14. This condition means that the piece-wise cubic polynomial is twice continuously differentiable and has less tendency to osciltate between data points. The natural

cubic spline is linear at the boundary subintervals, resulting in less boundary bias.

$$s''(x_0) = s''(x_j) = 0 \tag{14}$$

2.6.5 Measuring Success

As with the historical models, the mean absolute error (MAE) and the root mean squared error (RMSE) can also be used as measures of success.

2.7 Exploratory Data Analysis

It is important to explore the dataset being used for a number of reasons including: extracting important variables, identifying outliers or human error and understanding the relationships (or lack of) between variables [32].

Descriptive statistic methods include looking at the mean, median, variance, interquartile range and skewness. Visualisation methods include creating histograms, density graphs and scatter plots. More complex methods include dimensionality reduction and cluster analysis.

2.7.1 Outliers

An outlier is an observation that is distant from all the other data points in a dataset. If an observation has been identified as a likely outlier, it is important to try and analyse why it is the case, as outliers can arise from incorrect data collection or can be an indication of the underlying model being unsuitable [17]. Since the data used for the bus journey modelling is manually collected, there is room for human error and therefore, it is important to ensure the outliers are detected and removed accordingly.

Observations that have a large influence on the estimated coefficients of a regression model are called influential observations. They depend on the residual (the vertical distance) and the leverage (the horizontal distance). Figure 9 is an example showing how an outlier can affect predictions. The red line is the regression line fitted to the data including the outliers, the black line is the regression line fitted to the data without the outlier. It can be seen that the gradient of the 2 lines are not the same and therefore, the models would give differing predictions.
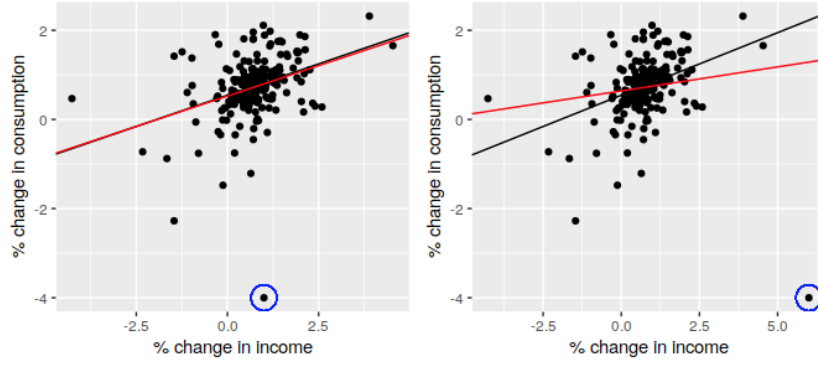
Figure 9: Example showing how an outlier can affect predictions, original image from [17]

Data points that fall outside 1.5 times of the interquartile range above the third quartile or below the first quartile can be classed as outliers. It is also possible to identify outliers using the z-score. The z-score is the signed number of standard deviations by which the value of an observation is above the mean value of what is being observed (Equation 15) [33].

$$ZScore = \frac{Observation - Mean}{StandardDeviation} \tag{15}$$

The calculation rescales and centers the data and looks for data points which are too far from zero. Convention is that data points that give a z-score of greater than 3 or -3 are classed as outliers.

3   Literature Review

Other papers researching bus journey predictions look at a wide variety of models and factors affecting the journey time. This project focuses on bus journeys in London, however, it was not possible to find papers that also looked into modelling bus journeys in London. The results and findings presented below are all for regions outside London and therefore, care must be taken to apply the conclusions drawn from these papers directly to this project. Table 5 shows a summary of the research papers that were studied.

Patnaik et al. developed a set of regression models to estimate the arrival times for buses travelling between two points. They used data collected by automatic passenger counters (APC) installed on buses in North-East United States of America [13]. For their model, they chose the following variables as independent variables: distance, number of stops, dwell times, number of boarding and alighting passengers and weather descriptors. They hypothesised that their results could have been improved if they had combined a Kalman Filter Model with their existing Regression Model.

Fan and Gurmu studied historical average models, Kalman Filter models and artificial neural networks (ANN) as the three models for predicting bus arrival times [10]. It was discovered that the Kalman Filter model gave the better prediction in a lot of the time intervals. It was hypothesised that this was because the model always uses the current measurement to predict the next step. However, the model was found to be vulnerable where there were huge differences in travel times between two consecutive time periods, perhaps because it struggled to filter out noise as smoothly and so was not as capable of handling abrupt changes between consecutive travel times. Although the ANN model did not get the best predictions in most of the time intervals, overall it provided the best prediction in both accuracy and robustness. This is possibly because the ANN could handle the differences in travel times between time periods and so gave relatively stable prediction during both peak and off-peak hours. However, it was also found that the ANN model was less effective in predicting bus travel times for very short and/or very long trips. This could be because very short trips have high percentage errors.

Sun et al. used a Kalman Filter model to analyse their data for real-time bus arrival time predictions [11]. They discovered that as the length of time from which the historical real time data is used (e.g. using the data collected from the past two hours as opposed to historical data from last week), increases from 30 minutes to 90 minutes, the model has a vast improvement on accuracy. However, the performance increase begins to plateau as the length of time increases any further, with the model's accuracy flat lining when the time window goes beyond 120 minutes. This indicates that only data within the last 120 minutes is significant for real time delay prediction. It was also discovered that the prediction performance of the models worsened as the prediction is applied further in the future.

Author(s)	Location	Model Developed	Factors Affecting Journey Times	Findings
Patnaik et al.	North East, USA	Multivariate Regression Model	distance travelled, dwell times, number of boarding and alighting passengers, weather descriptors, day of week, time of day	weather and day of week are not significant. time of day, distance and dwell times are significant.
Fan and Gurmu	Macae, Rio de Janeiro, Brazil	Historical average of all collected data	time of day, correlation of travel times of successive journeys	Outperformed by ANN and Kalman Model
		Kalman Filter Model	time of day	Good prediction with reasonable accuracy
		Artificial Neural Network (ANN)	time of day	Relatively stable prediction during peak and off-peak hours. Outperforms Kalman model.
Sun et al.	Nashville, USA	Kalman Filter Model	time of day, day of week	Looking at data in the past 90 minutes gives good results. Performance worsens when data is from more than 120 minutes ago
Jeong and Rilett	Texas, USA	Historical average	traffic congestion, dwell time at stops, time of day, day of week	Outperforms regression model.
		Multivariate Regression Model	traffic congestion, dwell time at stops, time of day, day of week, distance between stops	Hypothesised that there might be nonlinear relationships between the arrival time and some of the independent variables.
		Artificial Neural Network (ANN)	dwell time at stops, time of day, day of week	Outperforms historical and regression model.

Table 5: Summary of literature review findings

Jeong and Rilett studied historical average models, multivariate regression models and ANNs. This study found that the ANN model outperformed the other two models [6]. It was hypothesized that this was because the ANN was able to identify the complex nonlinear relationship between bus travel time and the independent variables. Jeong

and Rilett also found that the historical model provided best predictions for 'peak' hours. Their hypothesis behind this was that congestion made travel times consistently high, i.e. less variable.

3.1 Factors Affecting Bus Arrival Times

There are a variety of different factors that can affect the arrival time of a bus, be that through affecting the travel time of the bus or the dwell time of the bus. In this case, the dwell time refers to time that a bus spends at a bus stop before it continues on its route. For example, the bus' travel time could be affected by traffic conditions. Similarly, the bus' dwell time could be affected if there are a large number of people that get on at the previous stop. Due to the sheer number of factors that could be investigated, this section will take a brief look at a selection of the factors that have been considered in other papers as well as some other factors that have the potential to be important. This section will provide justification for why some of the factors listed below do not warrant further investigation and do not need to be included in the models that will be developed.

It should be noted that most of the factors mentioned below are largely dependent upon one another. For example, the time of day could affect the traffic conditions (say rush hour leading to more vehicles on the road) or the weather could affect the dwell time (if it is particularly cold or icy, there may be fewer people willing to walk or cycle, so there are more people using buses). Therefore, care will have to be taken when implementing the models, to ensure that not too much weighting is placed on any singular factor.

The majority of research into bus journey predictions make use of data on traffic stream variables, for example weather, speed, distance [10]. However, in the absence of such information and with limited resources, it is important to see if predictions made without this data can be made. Furthermore, it would be interesting to see if these predictions are comparable to those made with traffic stream information.

3.1.1 Weather

Koetse and Reitveld found that there is a substantial reduction in traffic speed when weather conditions are extreme [7]. In the presence of rain there was a decrease of up to 6% in traffic speed, up to 13% for snow and up to 12% for reduced visibility. Slower traffic speeds means that vehicles will take longer to arrive at their destinations than usual and therefore, it can be argued that weather should be taken into account when creating models that predict bus arrival times. Furthermore, when temperatures are at the extremes, there is a case for people being more likely to take a bus than walking or cycling. Not only would this increase the load on the bus, leading to a slower bus speed, but the dwell time at the bus stops would also increase as more time would have to be taken for people to get on and off the buses.

It was found by Patnaik et al. that there was little to no effect between weather and bus delay time [13]. This finding was attributed to the fact that the weather data used

in this investigation was not sufficiently detailed, or that during the study period the weather variations were not significant enough to have an impact on arrival times.

3.1.2 Time of Day and Time of Week

It is hypothesised that at different times of day and different days of the week, there are different travel patterns. For example, at weekday rush hour times, there are more likely to be more vehicles on the road and thus more congestion. On the other hand, on weekends, there is less likely to be a strong correlation between time of day and late buses as there is not a particular time when a large proportion of the population has to travel somewhere.

Fan and Gurmu clustered their travel time data by time period and explored the effects of this. The results can be seen in Figure 10, which plots the travel time index against the time of day [10]. Here, the travel time index is the ratio of the average travel time per weekday and the average travel time across all days. An observation that was made was that in the evening rush hour (5 - 6 pm), the travel times were 30% higher than the average across all time periods and days. This supports the idea of further exploring time of day as a parameter in the predicting of bus arrival times.

Fan and Gurmu also observed that the travel time distributions over the different days of the week seemed to be nearly the same. Therefore, since Figure 10 only shows data for weekdays, it could be argued that for weekdays, it will suffice to look at the effect of the time of day, without taking into account which day of the week it is. A similar concession can be made for weekends, i.e. the prediction model for Saturday 3pm should be no different to the prediction model for Sunday 3pm. However, Fan and Gurmu's findings are based on a model for a city in Brazil. It is unclear how strongly these findings would generalise to London. For example, the daily travel landscape in terms of when people leave work etc. may not be the same, or this city may not have bus lanes as in London.
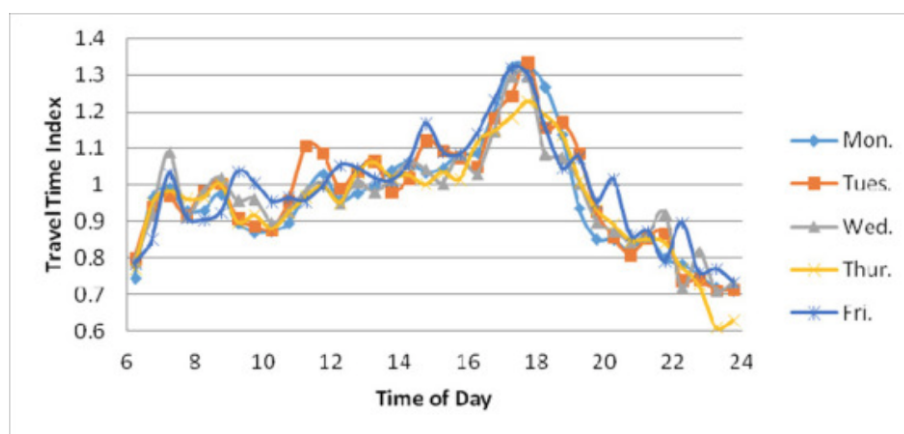

Figure 10: Effect of time of day and day of week, original image from [10]

26

3.1.3 Time of Year

Similarly to the previous parameter, it is hypothesised that at different times of year, there are different travel patterns for buses. This could be due to factors such as special occasion (e.g. New Year's Day or bank holidays) or school holidays. For example, during term time, buses are not likely to have have the same travel time as during school holidays.

Since this project only runs from March to June, it was initially believed that there would not be a significant enough event that would lead to a marked change in bus travel times. However, due to the situation with COVID-19, London was placed officially in lockdown from March 24th 2020 and the public was heavily discouraged from using public transport. From March 16th 2020 people had been told to avoid all non essential contact with others and people were asked to work from home where possible [34]. This dramatically reduced the load on the London transport network with the number of people using buses dropping by 85% in April and May 2020 [35]. Therefore, it was hypothesised that due to the lower number of vehicles on the road as well as the fewer number of people using buses, journey times would be shorter. As of May 18th 2020, TfL has begun to increase its bus services again, with the network running at 85% capacity from this date [36]. Furthermore, the government's new legislation means that from the 31st of May 2020, many of the previous restrictions are now being lifted. Therefore if data collected between March 24th 2020 and May 31st 2020 is labelled as lockdown data and data collected outside of these dates is labelled as pre/post lockdown data, then there should be enough data to adequately model this.

3.1.4 Dwell Time at Bus Stops

The amount of time a bus spends at a bus stop before continuing on its journey, also known as its dwell time, is one of the factors that have been considered to affect the arrival time of a bus. The dwell time of a bus can be affected by varying numbers of passengers at bus stops. For example, if a bus stop is outside a particularly popular destination, then the number of passengers getting off the bus would be higher than normal and thus the bus would have to wait at that particular stop for a longer amount of time.

Fan and Gurmu found that bus stop dwell times were less important and statistically not as significant as other potential factors affecting bus journey times [10]. Furthermore, with the tools currently available, there is no real way to calculate how long a bus waits a single stop for. TfL does not currently provide any APIs that would directly give a bus' dwell time nor does it provide APIs that could aid in the calculation of dwell time, e.g. the number of passengers boarding or deboarding a bus.

3.1.5 Traffic Conditions

Jeong and Rilett argued that in order to accurately predict bus travel time, it it essential to consider traffic conditions [6]. In June 2019 Google Maps introduced a new feature for forecasting bus delays by taking into account real time car traffic data and combining

this with data on bus routes and stops [8]. Both of these support the idea that traffic conditions should be considered as one of the parameters that affect bus arrival times.

However, Williams and Hoel found that daily traffic condition patterns were consistent across the weeks [37]. This implies that if bus delays are affected by traffic, then historical bus travel times should not vary across the same time periods for different weeks. Therefore, traffic conditions do not need to be taken into account as a factor, especially for historical average models.

It can also be argued that the presence of bus only lanes means that there is little point in taking general traffic into account because traffic conditions will have less of an effect on the travel time of buses. However, bus lanes in London are occasionally occupied by other vehicles or road works, resulting in less smooth traffic in bus only lanes.

3.2   Conclusions

Following in Patnaik et al.'s example [13], the effect of weather on bus arrival times will not be explored further. All the data that is used in this project has been collected manually since TfL does not provide any historical data on bus arrival times. Therefore, the collected data spans from mid March to early June. This is unlikely to be a large enough sample of different and extreme weather conditions to accurately identify and quantify the effects of differences in weather conditions. So, based off of Patnaik et al's findings, this supports the idea that weather would not be very suitable to be studied as a factor affecting bus arrival times in this particular project.

Time of day and day of week will be studied further as factors affecting bus journey times. Time of year generally is unlikely to be explored any further as a factor affecting bus arrival times because it is impossible to get a whole year's worth of data to use for modelling. However, the effect of COVID-19 lockdown on bus journey times will be considered further.

Since there is no way to get information on bus dwell times within the scope of this project, this factor will not be explored any further. However, this could be a possible extension or further work. Traffic conditions will also not be explored any further.

4 Design

4.1 Data Collection

The over arching algorithm for collecting data is shown in Figure 11. For each bus route the following steps occur.
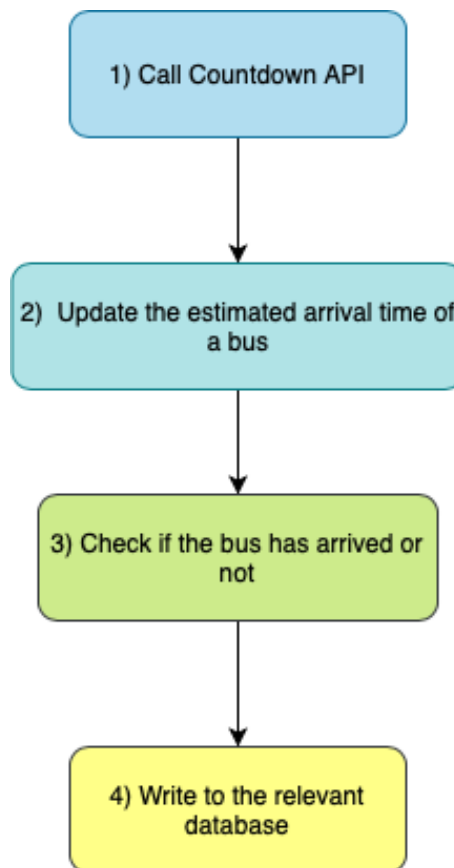

Figure 11: Overarching data collection algorithm

1) Call Countdown API: Data provided by the Countdown API is stop centric. This means that stop information is crucial to the requests made. Therefore, before the Countdown API can be called, the Stoppoints API must be called to determine the stop ids of all of the stops on a bus route.

Stoppoints could be called before each Countdown call, however, this is a waste of time since as mentioned in Section 2.1 Stoppoints returns a mix of valid and invalid stop ids. Time would have to be spent filtering the Countdown responses to see which have returned expected arrival times and which have returned HTTP errors. Furthermore, it is much faster to access a list of valid stop ids from a database than to retrieve information from an API call.

Therefore, it is more computationally efficient to call the Stoppoints API for each bus route used in the data collection process once a week and store the response in a database. Since Countdown only requires the unique national identifier (`naptanID`) of the bus stop, it is only necessary to extract this parameter from the API response for each stop. The common English name of the stop (`commonName`) is also extracted from the API for ease of data analytics later on. The `valid_stop_ids_route` table stores the list of valid stop ids and its respective plain English name for that particular route. A snippet of valid stops for bus route 9 is shown in Figure 12. The database design is shown in Figure 13.

stop_id	stop_name
490010357F	North End Road
490011198W	Prince Of Wales Gate
490004455BB	Brook Green
490000093PE	Green Park Station
490008652C	Kensington Olympia Station
490010574C	Old Park Lane / Hard Rock Cafe
490003193R	Aldwych / Somerset House
490000130KE	Knightsbridge Station / Harrods
490000110F	High Street Kensington Station
490011822W	Knightsbridge Barracks
490005528Y	The Design Museum
490013766E	Charing Cross Stn / Trafalgar Square
490006691W	Exhibition Road
490004455BA	Brook Green
490011750W	Royal Albert Hall
490G000386	Aldwych / Somerset House
490007960R	Haymarket / Jermyn Street
490019703E	Aldwych / Drury Lane
490000119T	Hyde Park Corner Station

Figure 12: Valid Stop Ids Example

valid_stop_ids_route
stop_id: varchar(20)
stop_name: varchar(50)

Figure 13: Valid Stop Ids Table

2) Update the estimated arrival time of a bus: Countdown is called continuously every 30 seconds because as mentioned in Section 2.1 the information is only updated at source every 30 seconds. If the new expected arrival time returned from the API call is the same as what the database already holds, there is no need to update the item. Otherwise, the database updates both the bus' expected arrival time and the time of the API request.

Figure 14 shows two different expected arrival times for DALGARNO GARDENS on bus route 7. The figure demonstrates how the predicted arrival time for the same vehicle changes the closer it gets to the actual arrival time. Therefore, data much be collected and updated continuously in order to get as accurate an inferred arrival time as possible.

bus_stop_name ℹ	direction	expected_arrival	time_of_req
Dalgarno Gardens	out	2020-03-11 14:32:55	2020-03-11 14:21:19
Dalgarno Gardens	out	2020-03-11 14:34:06	2020-03-11 14:25:43

Figure 14: Route 7 data for a particular vehicle X

3) Check if the bus has arrived or not: Figure 15 shows the flow of logic which takes place in order to ascertain whether a bus has arrived or not. If a bus *A* is due to arrive at time *X*, when Countdown is called at time *X* and *A* does not appear in the response, it cannot be assumed that *A* has arrived at time *X*. It is possible that there could have been a glitch in the system and *A* has just temporarily vanished. Therefore, the algorithm waits 5 minutes after time *X* and if *A* does not appear in the Countdown response again, then it can be inferred that *A* did indeed arrive at time *X*. However, if during the 5 minutes after time *X* bus *A* reappears, this implies that *A* will return with a new expected time *Y* and thus did not arrive at time *X*.
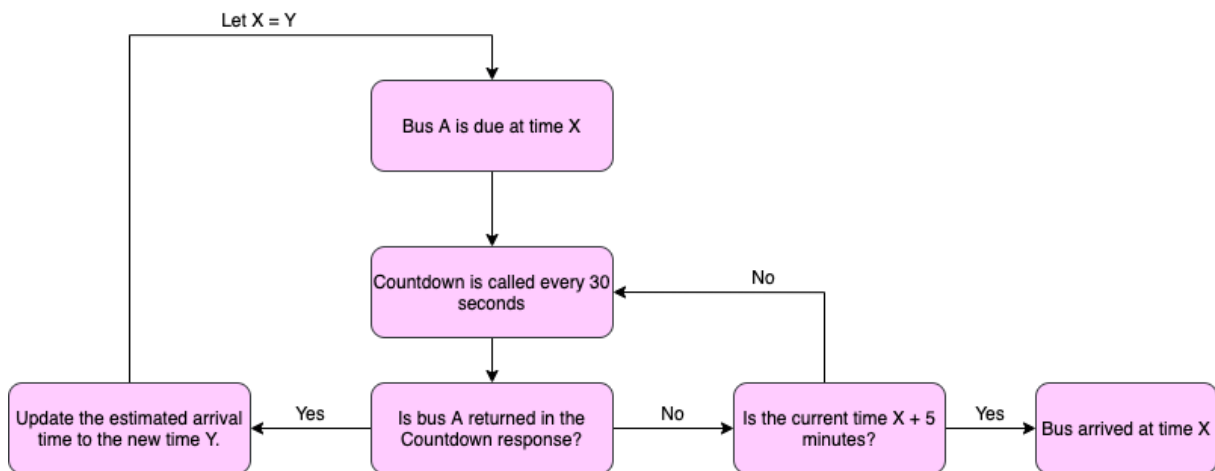

Figure 15: Check if the bus has arrived or not

4) Write to the relevant database: Figure 16 shows the designs of each of the databases described below.

- `bus_information_route` stores information on buses that haven't arrived yet for a particular route.

- `bus_arrivals_route` stores information on all buses that have arrived for this route.

Figure 16: Databases

Recall the example Countdown responses in Section 2.1 for buses arriving at NORTH END ROAD on bus route 9. Figure 17 shows how the JSON information has been converted into strings and datetime objects. Specifically, bus *14510* was predicted to arrive at *1589526117000* in UNIX epoch time. This converts to *07:01:57* as seen in the third row of Figure 17. The vehicle_id is a combination of the actual bus id, the stop id, the date of arrival, the direction and the journey number that the bus is on. This is necessary because a particular bus e.g. bus with id *14510* will complete more than one full journey on its route.

vehicle_id	bus_stop_id	time_of_arrival	time_of_req	direction
14462_490010357F_2020-05-15_inbound_0	490010357F	2020-05-15 06:52:44	2020-05-15 06:50:57.09	inbound
14511_490010357F_2020-05-15_inbound_0	490010357F	2020-05-15 06:56:24	2020-05-15 06:56:33.794	inbound
14510_490010357F_2020-05-15_inbound_0	490010357F	2020-05-15 07:01:57	2020-05-15 06:59:33.185	inbound
15448_490010357F_2020-05-15_inbound_0	490010357F	2020-05-15 07:11:51	2020-05-15 07:11:29.136	inbound
15453_490010357F_2020-05-15_inbound_0	490010357F	2020-05-15 07:24:15	2020-05-15 07:22:53.622	inbound
15449_490010357F_2020-05-15_inbound_0	490010357F	2020-05-15 07:34:29	2020-05-15 07:34:25.417	inbound

Figure 17: Buses that arrived at North End Road

4.1.1 Technology Choices

This project is quite broad in terms of implementation decisions, therefore, there was no necessity to pick any particular language. Since I had done a lot of projects in Python in my final year of university and because Python has a lot of support online, I chose to write the code that performs the API calls and logic for whether a bus has arrived in Python.

Initially, the data collection code was hosted on Amazon Web Services (AWS), using AWS Lambdas to run the data collection functionality and AWS DynamoDB to store the data. AWS Lambda allows code to be run without requiring a server to be provisioned or managed. AWS DynamoDB is a key-value NOSQL database. I chose AWS because its free tier had generous usage and its services worked cohesively together easily and smoothly. Unfortunately, approximately two months into the data collection stage - the

code having been left running while I moved on to other parts of the project - I realised that I had exceeded the free tier usage. Therefore, I had to relocate my code onto the DoC Cloud VM. This was chosen because there is less of a memory/usage capacity limit and furthermore is free.

The data collection code has now been put in a Docker container. This is because it made it much easier to move the code between environments as it already held all the dependencies needed to run the code.

The current data collection code has now changed to use a PostgreSQL database, which is a relational database. This change was done partly because by this point I had already begun writing the code to do the arrival time predictions and I found that it was much more convenient to query for values instead of having to loop through the key value pairs.

4.2   Data Exploration

It is important to explore the dataset collected before building the models in order to understand it better. This could be by discovering patterns, spotting outliers or understanding relationships between variables.

For each of the bus routes, graphs were plotted to show the total number of buses arriving at all bus stops on a route at different times of day for data collected over a three and a half week period. An example can be seen in Figure 18, which shows that over the three and a half weeks, a total of just under 1,200 buses arrived at all the stops on route 6. This helped to see the pattern of numbers of buses according to time of day. It was hypothesised that there would be more buses running during the day than during the night. This was supported by the graphs, for example, from Figure 18 it can be seen that for route 6 during the day, approximately between 0700-1700, there are many more buses on the road, compared to say 2100-0300.
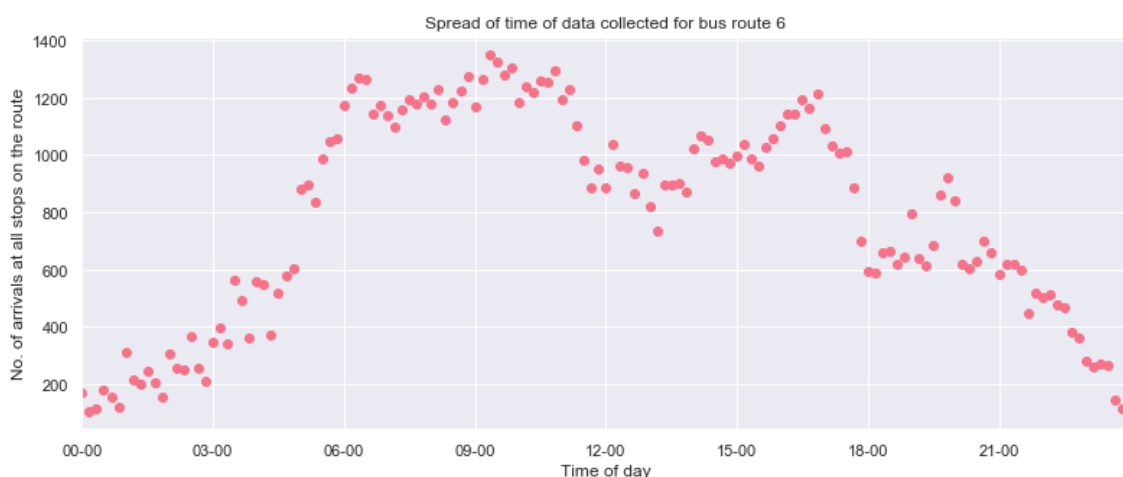

Figure 18: Number of buses during the day for route 6

33

Graphs were also plotted to see the total number of buses arriving at all bus stops on different days of the week for data collected over the same three and a half week period. An example can be seen in Figure 19, which shows that a total of approximately 20,000 buses arrived at all stops on route 6. It was hypothesised that there would be more buses running during weekdays compared to the weekend. However, it can be seen from Figure 19 that there did not seem to be a pattern indicating that the hypothesis was correct.
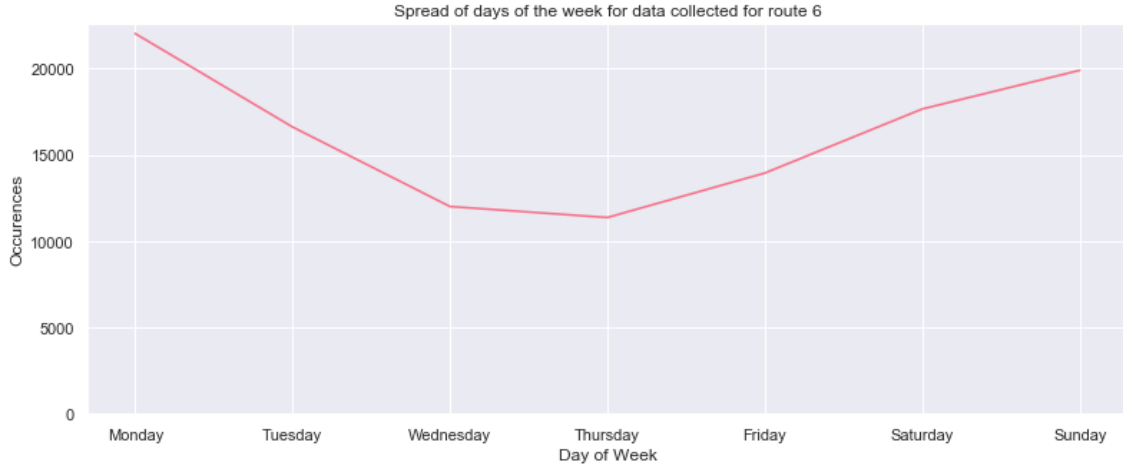

Figure 19: Number of buses during the week for route 6

For each route, two bus stops were chosen and the travel times of the bus were calculated. The z-score (Equation 16) was calculated for each journey time and used to measure whether a journey time was an outlier or not. A z-score of larger than 3 or smaller than -3 indicates an outlier [32].

$$z = \frac{x_i - \mu}{\sigma} \tag{16}$$

Where x_i is a journey time, μ is the mean and σ is the standard deviation. The number of occurrences of a travel time was plotted against the actual travel time, marking the outliers in a different colour. An example of this is seen in Figure 20 and Figure 21. Figure 20 shows the travel times of bus route 52 between ALL SOULS AVENUE and NOTTINGHILL GATE STATION, whilst Figure 21 shows the travel times of bus route 9 between NORTH END ROAD and PHILLIMORE GARDENS. In both figures the outliers are marked in green. In the figures below, 'occurrences' refers to the number of times that a particular journey time was recorded.
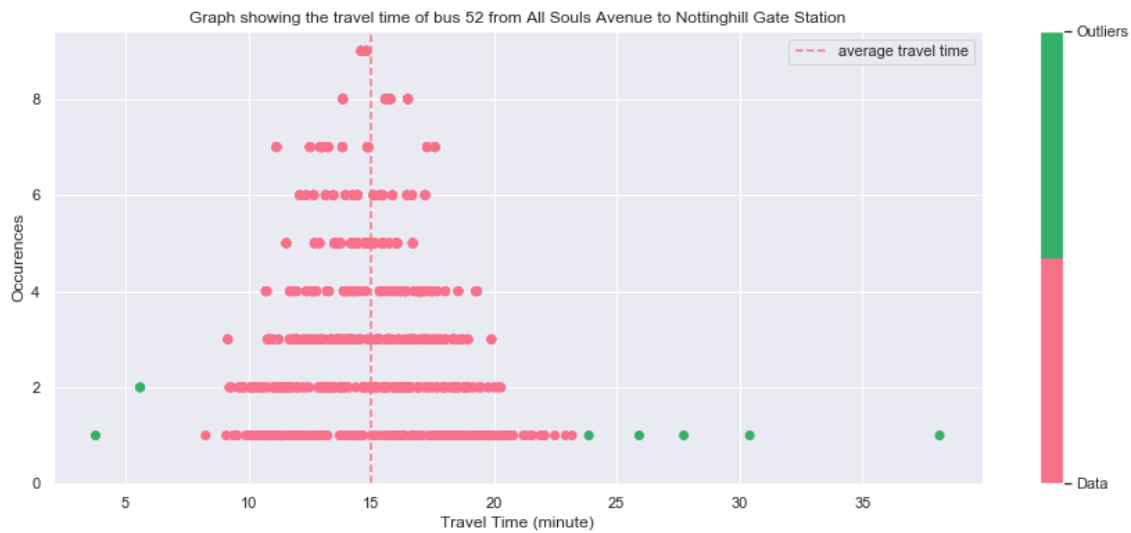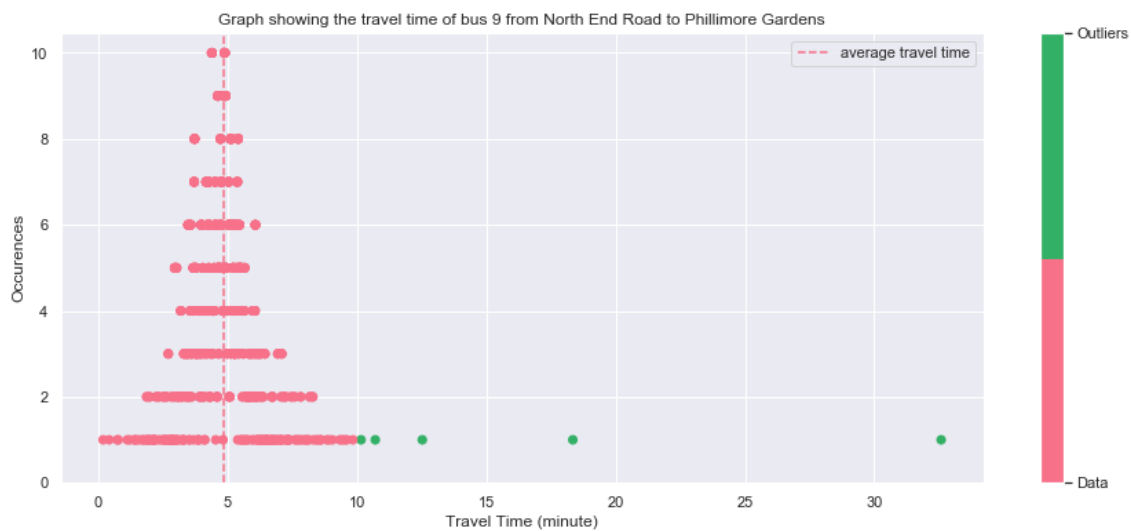
34

Figure 20: Route 52 travel time and outliers

Figure 21: Route 9 travel time and outliers

The outliers that have been identified for each route were then removed from the dataset so that the models developed using the data would not be affected by them.

The variance and standard deviation were also calculated. For the journey times for buses on route 52 between ALL SOULS AVENUE and NOTTINGHILL GATE STATION, the standard deviation was 2.761 (3sf) and the variance was 7.624 (3sf). For the journey times for buses on route 9 between NORTH END ROAD and PHILLIMORE GARDENS, the standard deviation was 1.759 (3sf) and the variance was 3.095 (3sf). ALL SOULS AVENUE and NOTTINGHILL GATE STATION are 16 stops apart whilst NORTH END ROAD and PHILLIMORE GARDENS are 5 stops apart. Therefore, it was hypothesised that the standard deviation and variance for journey times are lower for stops that are closer to each

other. In order to confirm this, ALDWYCH / SOMERSET HOUSE was chosen as the start stop and the variance and standard deviation of the journey times between the start stop and stops further away on route 6 were calculated. The results for between two stop away to twenty stops away can be seen in Figure 22.
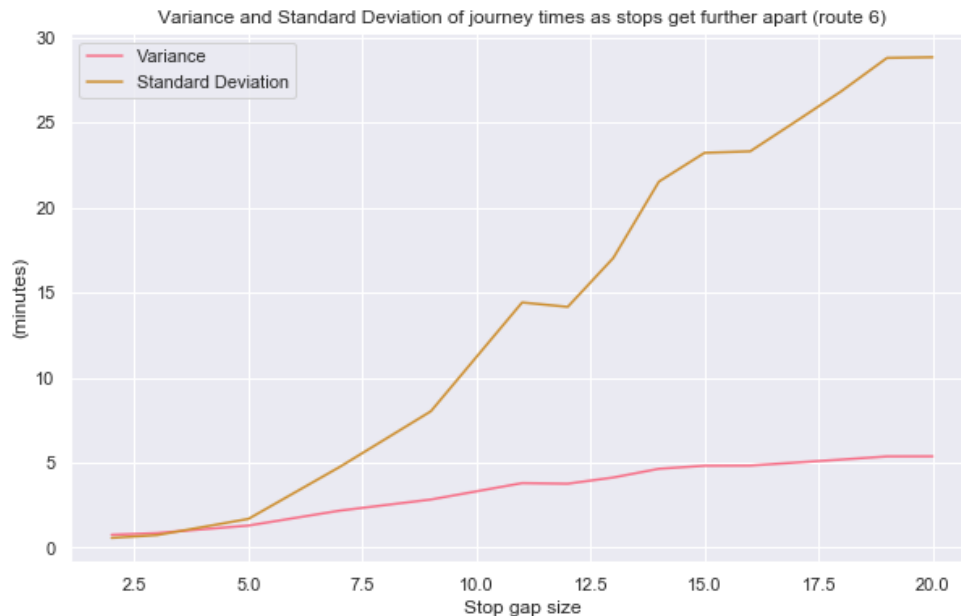

Figure 22: Route 6 Variance and Standard Deviation

Theoretically, the variance graph should follow a straight-line relationship with gap size. The deviation from this can be attributed to inconsistencies in stop gap size, meaning two stops that are five stops apart are not exactly the same distance in kilometres apart. Based on the results as seen in Figure 22, it is possible to conclude that there is indeed an upward trend in the variance and standard deviation for stops that are further apart.Theoretically, the variance graph should follow a straight-line relationship with gap size. The deviation from this can be attributed to inconsistencies in stop gap size, meaning two stops that are five stops apart are not exactly the same distance in kilometres apart. Therefore, predictions made for bus stops that are closer together are likely to be more accurate.

4.2.1   Effect of time of day and day of week

To see the effect of time of day on bus journey times, the times of day were split into eight sections: 00-03, 03-06, 06-09, 09-12, 12-15, 15-18, 18-21, 21-00. For each route two bus stops were chosen and the travel times of the bus were calculated and plotted, as in Figure 23 and Figure 24. Figure 23 shows the travel times of buses from Florence Road to Star Lane on route 69, which are five stops apart. Figure 24 shows the travel times of buses from All Souls Avenue to Notting Hill Gate Station on route 52, which are 16 stops apart. The Figures also have the average journey time across all hours of the day plotted as a black dotted line.

For Figure 23 the statistics are as follows (all units are in minutes and the numbers are to five significant figures):

- **00-03**: standard deviation = 0.97508, mean = 3.0929

- **03-06**: standard deviation = 0.85072, mean = 3.5386

- **06-09**: standard deviation = 1.0559, mean = 3.9989

- **09-12**: standard deviation = 0.91878, mean = 4.1066

- **12-15**: standard deviation = 0.91933, mean = 4.36120

- **15-18**: standard deviation = 1.1008, mean = 4.1694

- **18-21**: standard deviation = 1.2570, mean = 3.8441

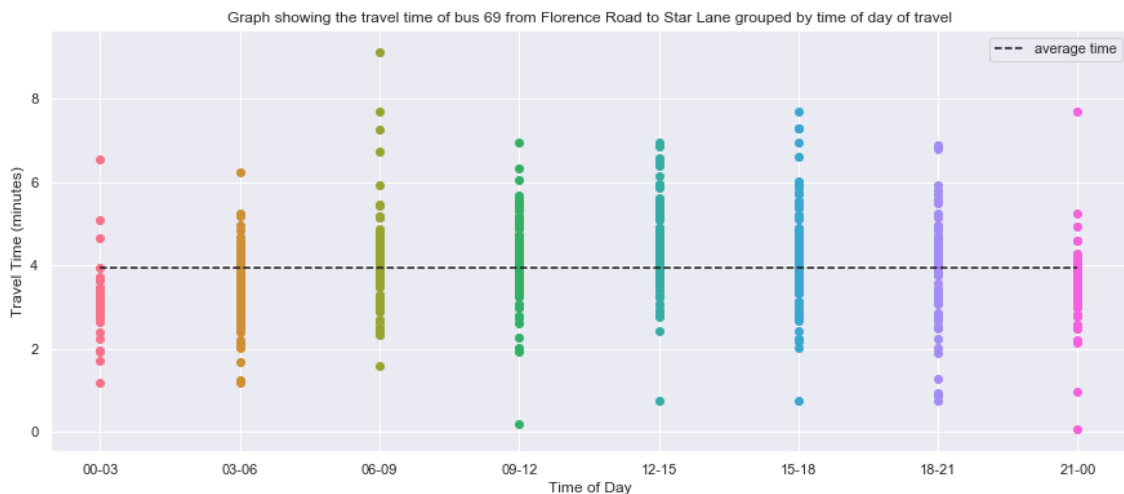- **21-00**: standard deviation = 1.0189, mean = 3.4616

Figure 23: Route 69 travel time grouped by time of day

For Figure 24 the statistics are as follows (all units are in minutes and the numbers are to five significant figures):

- **00-03**: standard deviation = 1.2467, mean = 12.567

- **03-06**: standard deviation = 2.0847, mean = 12.698

- **06-09**: standard deviation = 1.6443, mean = 14.644

- **09-12**: standard deviation = 2.1843, mean = 15.995

- **12-15**: standard deviation = 1.6768, mean = 17.386

- **15-18**: standard deviation = 2.2638, mean = 16.500

- **18-21**: standard deviation = 3.2022, mean = 14.707

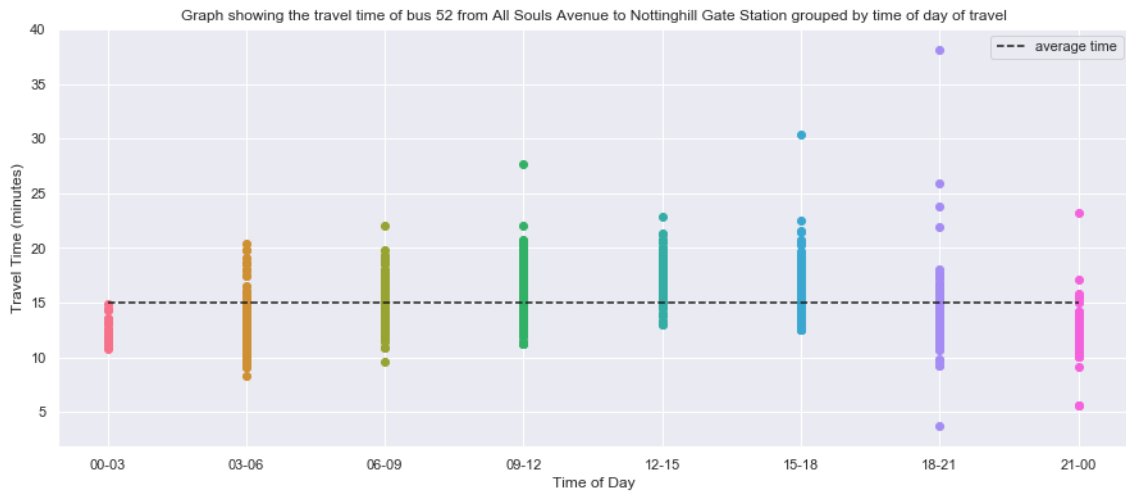- **21-00**: standard deviation = 2.0572, mean = 12.800

Figure 24: Route 52 travel time grouped by time of day

To see the effect of the day of week on bus journey times, the days of the week were split into weekdays versus weekends. As above, for each route two bus stops were chosen and the travel times calculated and plotted. An example can be seen in Figure 25, which shows the travel times for a bus on route 6 from Marble Arch to Church Street Market. The overall average, weekday average and weekend average travel times are plotted horizontally. It can be seen that there does not seem to be a marked difference between weekday versus weekend journey times, with all three lines almost on top of each other.
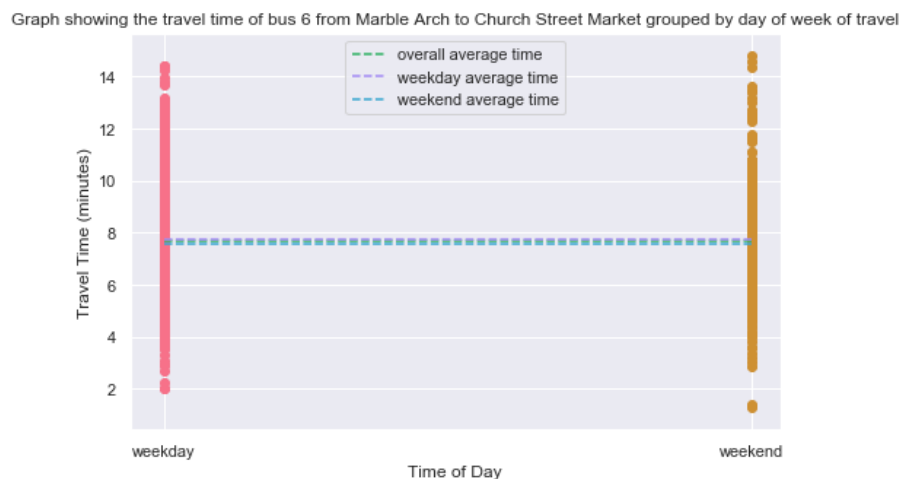

Figure 25: Route 6 travel time grouped by day of week

4.2.2 Effect of COVID-19 lockdown

To see the effect of the lockdown in London on bus journey times, the journeys were split into journeys that occurred during lockdown and journeys that occurred prior or post lockdown. In this case, lockdown was considered to have begun from 24th March 2020 up to 31st May 2020, when the government began easing restrictions on travel [38]. As before, pairs of bus stops were chosen and the travel times calculated and plotted. Figure 26 shows an example of travel times for a bus on route 6 from Marble Arch to Church Street Market. Due to the reduced number of vehicles on the road and the reduced number of passengers, it was hypothesised that journey times during the lockdown period would be less than journey times outside of this period. This is confirmed by the Figure 26, which shows three horizontal dotted lines representing the overall average, pre/post lockdown and during lockdown average travel time. It can be seen that the purple line representing average travel time before or after lockdown is higher than the other two lines.
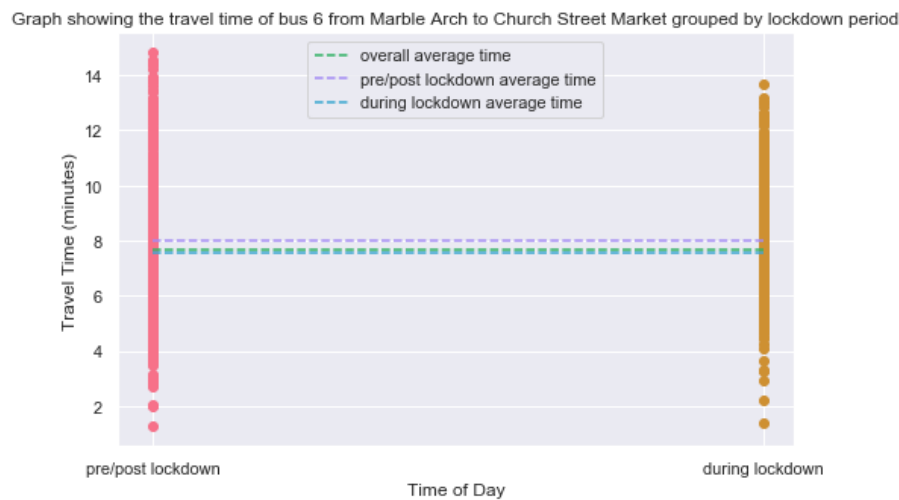

Figure 26: Route 6 travel time grouped by lockdown period

4.2.3 Conclusions

It was hypothesised that there would be a morning and afternoon peak where journey times would be longer. This is not so obvious in Figure 24 and Figure 23. This could be because the time of day was grouped into sets of three hours, which may be too sparse to show a pattern. However, it can be seen that in the early morning and late night hours (approximately 2100 - 0300), the journey times are generally shorter than other times of day. This makes sense because at these times there should be less traffic on the roads and fewer passengers boarding/disembarking. Therefore, time of day was decided to be chosen to explore as a factor affecting bus journey times.

It was also hypothesised that although overall journey times would not be vastly different between weekdays and weekends, it was likely that the morning peak or afternoon peak would be at different times for the weekend compared to weekdays. For

this reason, day of week was still chosen to be explored further as a factor affecting bus journey times.

However, whether a journey was made during lockdown or not was not chosen as a factor to be implemented in the models. This is because, data from June 2020 onward will no longer count as being in the lockdown period. Therefore, since predictions use recent data, none of the recent data will qualify as being during lockdown and so there isn't much point including it as a factor.

4.2.4 Technology Choices

As with the previous section, Python was again chosen as the language to implement the data exploration code in. Furthermore, Jupyter Notebook was chosen to write and run the code in. This is because Jupyter Notebook is a great tool for data exploration and quick prototyping, enabled by its interactive kernels.

4.3 Historical Models

The historical model is a combination of the mean and naive forecasting methods; it takes the weighted average of the journey times looking back at either a different number of buses or different amounts of time.

The hypothesis is that the journey time of a bus is dependent on the journey times of recent buses that also completed this journey. Therefore, to explore what 'recent' means in this case, the algorithm looks back at 1) different amounts of time and 2) different numbers of buses. The point of looking only at recent data is to ensure that the prediction is able to react to new changes. For example, if there has been an accident on the road, then it would make sense to assume that journeys soon after this accident would take longer. However, if this accident happened far enough into the past, for example an hour ago, it would also be reasonable to assume that this would no longer affect the journey times of vehicles now.

If the user gets on bus route X at stop A and requests the estimated arrival time at stop B, the algorithm by which this time is predicted is shown in Figure 27:
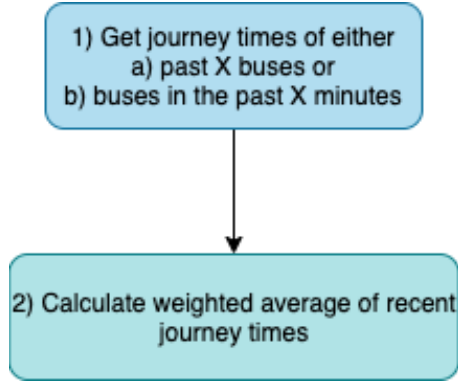
Figure 27: Overarching historical model

1) Get recent journey times: To ensure a wide enough breadth of 'recent' journey times to explore, eight submodels are explored.

(a) Looking back different amounts of time: The historical model looks back at buses from the past 15 minutes, 30 minutes, 60 minutes and 120 minutes.

(b) Looking back different numbers of buses: The historical model looks back at the last 2 buses, 5 buses, 10 buses and 15 buses.

2) Calculate weighted average of recent journey times: It is hypothesised that more recent journeys are more important than ones not as close to the request time, therefore a weighted average is most suitable. To confirm this, different weightings were applied to the same model and the weightings that provided the best RMSE and MAE was chosen.

For example, Figure 28 shows the RMSE and MAE of four sets of weights for the historical model that looks back at the last 5 buses. Let t be the amount of time that a bus arrived at stop B before the request time. Let n be the nth most recent bus to arrive at stop B before the request time. For example, $t = 10$ implies that the bus arrived at stop B 10 minutes before the request time and $n = 1$ implies that bus was the most recent to arrive at stop B before the request time. The four models used were:

- $0 < t \leq 2 : 0.5, 2 < t \leq 5 : 0.5$, to equal weightings would.

- $0 < t \leq 2 : 0.85, 2 < t \leq 5 : 0.15$, to model weightings heavily skewed towards recent journeys.

- $0 < t \leq 2 : 0.55, 2 < t \leq 5 : 0.45$, to model weights slightly more skewed towards recent journeys.

- $0 < t \leq 2 : 0.35, 2 < t \leq 5 : 0.65$, to model weightings skewed towards not as recent journeys

Figure 28: Weight tuning

The final weightings as chosen from the weight tuning are as follows:

- Weights for the past 15 minutes: $0 < t \leq 10 : 0.65, 10 < t \leq 15 : 0.35$

- Weights for the past 30 minutes: $0 < t \leq 10 : 0.65, 10 < t \leq 20 : 0.3, 20 < t \leq 30 : 0.05$

- Weights for the past 60 minutes: $0 < t \leq 10 : 0.65, 10 < t \leq 20 : 0.2, 20 < t \leq 40 : 0.1, 40 < t \leq 60 : 0.05$

- Weights for the past 120 minutes:$0 < t \leq 10 : 0.65, 10 < t \leq 20 : 0.18, 20 < t \leq 40 : 0.1, 40 < t \leq 80 : 0.05, 80 < t \leq 120 : 0.02$

- Weights for the past 2 buses: $0 < n \leq 2 : 1$

- Weights for the past 5 buses: $0 < n \leq 2 : 0.55, 2 < n \leq 5 : 0.45$

- Weights for the past 10 buses: $0 < n \leq 1 : 0.55, 2 < n \leq 5 : 0.35, 5 < n \leq 10 : 0.1$

- Weights for the past 15 buses: $0 < n \leq 1 : 0.45, 2 < n \leq 5 : 0.3, 5 < n \leq 10 : 0.2, 10 < n \leq 15 : 0.05$

4.3.1 Evaluating the model

The model was evaluated by choosing random pairs of stops and routes and then making requests at thirty minute intervals from 2020/05/25 02:00:00 to 2020/06/08 23:59:59. This constitutes new testing data outside of the training set. The root mean squared error (RMSE) and the mean absolute absolute errors (MAE) were calculated and compared against each other.

Initially, the eight sub-models were tested on pairs of stops that were five stops apart. Then, the best two of the sub-models were chosen by averaging their RMSE and MAE

scores. These were run again using pairs of stops that were further apart to see how well the models generalised to stops that were further apart. As described in Section 4.2, the variance in the journey times of the stops increase as the gap between the stops increase. Therefore, the models perform best on stops that are closer together. So, the best overall model is the one that still maintains a good MAE and RMSE for larger gaps.

4.3.2 Technology Choices

Python was chosen as the language to develop the models. This is due to the large amount of support for statistical modelling (e.g. sklearn) and because I have experience using Python in my Methods for Data Science module in the Department of Mathematics.

4.4 Regression + Interpolation Models

The regression and interpolation model is a two part model that combines two models together to make a prediction. The first model is a regression model on characterstics of the journey itself (such as time of day, day of week) and the second model is one of three sub-models based on recent journey times for those stops: weighted average model, line of best fit model and natural cubic spline model. The two sub-models combined form a linear interpolation model.

The hypothesis is that the journey time of a bus relies on both 'global' conditions and 'local' conditions. 'Global' conditions in this case refers to factors such as the time of day or the day of the week that have a constant effect throughout history. 'Local' conditions on the other hand refers to more recent journey data that could indicate an increase or decrease in journey time. For example, it is assumed that the general trend in travel conditions is more or less the same every weekend - this is a 'global' condition. However, if on a particular weekend there is a car crash in the morning that causes journey times to increase, then for this particular weekend, the expected journey time would not be the same as with other weekends - this is a 'local' condition.

This is why a two part model was chosen; so that the first part, the regression model, could calculate the 'global' prediction, and then the second part could calculate the 'local' prediction. A weighted average of the two predictions is then used to calculate the overall prediction of a journey time.

Before being able to choose the 'global' conditions, it is necessary to explore the different possible features to see if they are correlated to each other or not. If the factors are dependent upon one another, then it is sub-optimal to use them as separate covariates.

If the user gets on bus route X at stop A and requests the estimated arrival time at stop B, the algorithm by which this time is predicted is shown in Figure 29.
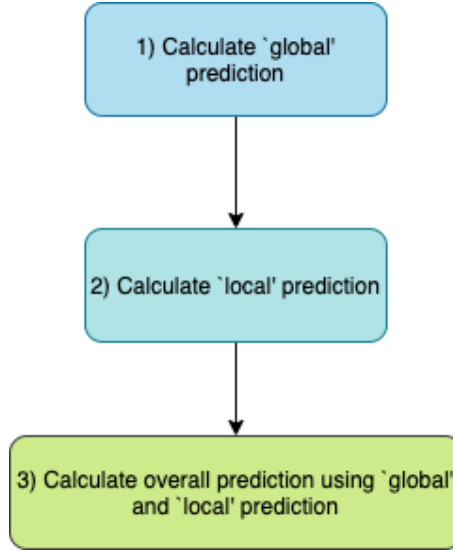
43

Figure 29: Overarching combined model

1) **Calculate 'global' prediction**: The first part of the model, which is the regression model, predicts the journey time of a bus from stop A to stop B based on the 'global' conditions. The chosen factors are the gap size between the stops, the time of day and the day of week. Equation 17 shows the linear regression model used for the prediction.

$$y_{pred1} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + ... + b_{26} x_{26} \tag{17}$$

where

- x_1 is the number of stops apart stop A and stop B are (gap size).

- x_2 is a binary value {0, 1} representing the day of week, where 0 is weekday and 1 is weekend

- $x_3...x_{26}$ is the one hot encoding of the time of day where x_i represents the (i-3)th hour of the day, for i = [3,26]. e.g. x_3 represents the 0th hour of the day i.e. between 00:00 and 00:59

- y_{pred1} is the predicted journey time of a bus from stop A to stop B based on 'global' conditions.

Figure 30 is an example of the data that is used to train the regression model. The 'Journey Time (s)' column is the y truth values to compare against the y_{pred1} predicted values. The remaining columns are the x training values fed into the model.

index	Journey Time (s)	Gap	Day of Week Encoded	0	1	2	3	4	5	...	14	15	16	17	18	19	20	21	22	23
0	414.0	5	0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	276.0	5	0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	273.0	5	0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	300.0	5	0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	300.0	5	0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 30: Part 1 Regression input example

The model is trained once on different pairs of stops of varying routes and gap sizes and then its intercept and coefficients are stored. When a request is made the algorithm need only gather the required features and put them into Equation 17 to get y_{pred1}, using the stored intercept and coefficient values. This calculated y_{pred1} is stored for further use in step 3).

2) **Calculate 'local' prediction**: The second part of the model predicts the journey time of a bus from stop A to stop B based on the 'local' conditions, which in this case is the last 10 buses to travel from stop A to stop B. It was decided to look back 10 buses because based on the historical models as described in Section 4.3, the model that looked back 10 buses performed the best. There are three different sub-models that were used:

- **Weighted Average**

$$y_{pred2} = weighted \ average \ of \ last \ 10 \ journeys \qquad (18)$$

  where the weights are the same as in Section 4.3: up to 2 buses = 0.55, between 2 to 5 buses = 0.35, between 5 to 10 buses = 0.1. This makes this sub-model essentially the same model as the historical average model. A weighted average model is good for cancelling out rivalling effects or when the data is noisy. However, it does not indicate trends in the data. For example, as seen in Figure 31 if the last 10 journeys are decreasing in journey time, it might be assumed that at the current request time, the journey time would be even less, following the downward trend. However, the weighted average model is unable to provide such a prediction.
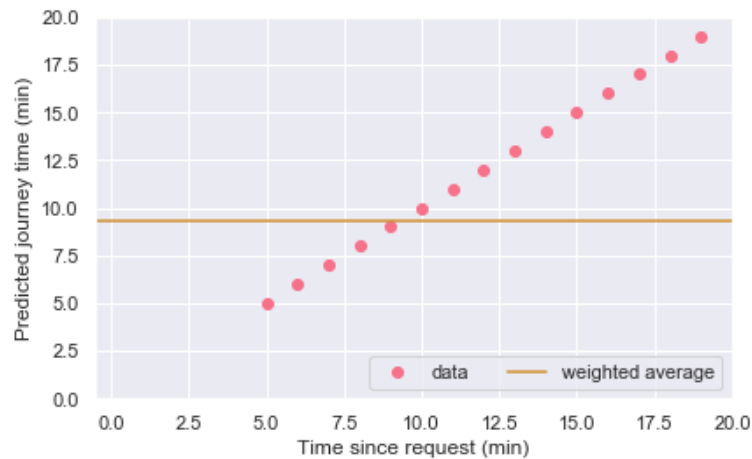
45

Figure 31: Unable to capture trend

- **Linear Regression or Line of best fit**

$$y_{pred2} = linear\ regression\ based\ on\ last\ 10\ journeys \tag{19}$$

If the y axis is the journey time and the x axis is the time since the request, then y_{pred2} is the intercept with the y axis. It is undesirable to use a regression model with degree larger than 1 because at the extremes the values become very large or very small. Thus, the intercept, would also be very large or very small and not close enough to the desired predicted value. This is demonstrated in Figure 32, which shows the journey times of the last 10 journeys alongside the regression models. It could be inferred from the last 10 journey times that the bus journey time at the request time (i.e. at 0 on the x axis) would be somewhere in between 5 and 7 minutes. This is captured best with the model with degree = 1.
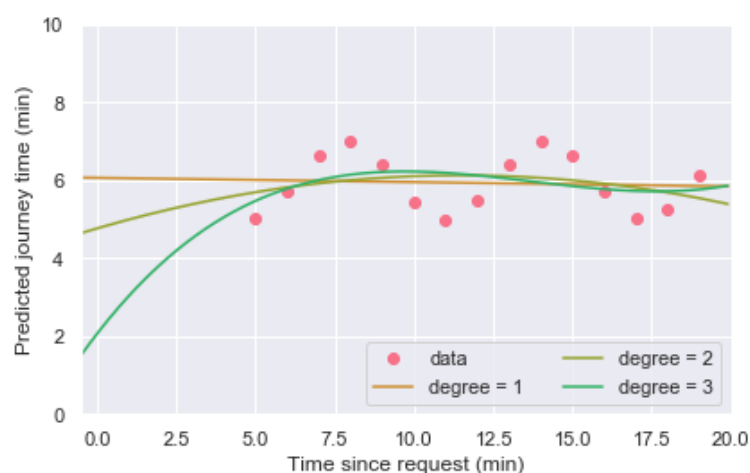

Figure 32: Intercept becomes more extreme

- **Cubic Spline Interpolation**

$$y_{pred2} = fit\ a\ cubic\ spline\ to\ the\ last\ 10\ journeys \tag{20}$$

Similarly to the linear regression case, the y axis is the journey time, the x axis is the time since the request and the intercept is y_{pred2}. Cubic splines with three different types of boundary conditions were used: 'not-a-knot', 'clamped' and 'natural'. 'not-a-knot' indicates that the first and second segment at a curve end are the same. 'clamped' indicates that the first derivative at the curves' ends are zero. 'natural' indicates that the second derivatives at the curves' ends are zero. Figure 33 demonstrates that the 'not-a-knot' and 'clamped' conditions provide more extreme intercept values than the 'natural' condition.
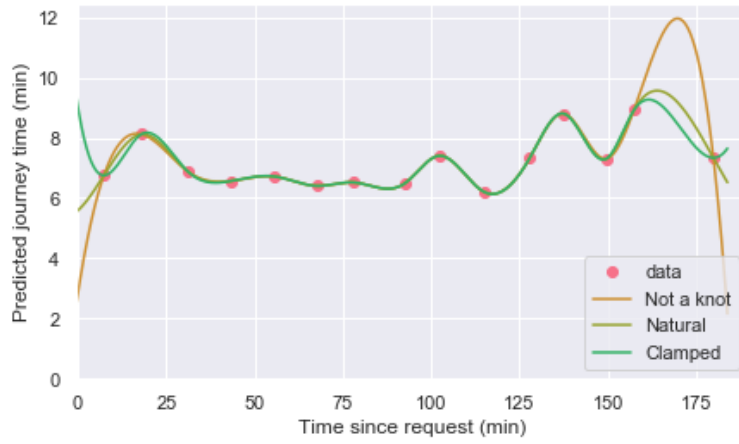

Figure 33: Three Boundary Conditions

All three 'local' models were used in the training process, but only the best performing one will be used in the mobile app. The input to all three models is simply the journey times of the most recent fifteen journeys from stop A to stop B and how long ago the bus arrived at stop B compared to the request time. Therefore, y_pred2 can be calculated in such a manner and is stored for further use in step 3).

3) Calculate overall prediction using 'global' and 'local' prediction: The last part of the model combines y_{pred1} with y_{pred2} to form Equation 21 for the overall prediction. This is essentially a linear interpolation model:

$$y_{pred} = \alpha y_{pred1} + (1 - \alpha)y_{pred2} \tag{21}$$

where

- y_{pred} is the overall journey time prediction for a bus travelling from stop A to stop B at the request time.

- α is some coefficient that we are looking to find such that it minimises the validation root mean squared error (RMSE) and mean absolute error (MAE).

Parameter tuning on α is performed on y_{pred1} and all five y_{pred2} values, with α varying from zero to one. The optimal value of α will provide insight into the relative importance of the 'global' vs 'local' conditions.

4.4.1 Model Evaluation

The overall model was evaluated in the same way part one of the model is evaluated: requests are made from 25/05/2020 02:00:00 to 08/06/20 23:59:59 at thirty minute intervals for a number of different pairs of stops of varying routes and gap sizes. The root mean squared errors (RMSE) and mean absolute errors (MAE) were calculated for the five possible models (due to the five possibilities for part two of the model) and compared against each other.

4.4.2 Technology Choices

As with the Historical Models, Python was the language used to develop the models. This was due to the large amount of built in help for regression and interpolation that comes with the `sklearn` library.

4.5 Mobile App

The purpose of the mobile app is merely to provide a more user friendly way of presenting the models' results. Therefore, the design of it is very basic. The application allows the user to enter the start and end stops along with the bus route they intend to take and then is returned the predicted journey time. The assumption is that the user wants to make a journey at the time that they enter the information into the app, i.e. the predicted journey time will be for 'now'.

There is no capability for requesting journey times in the future. This is because predicting future journey times, such as in CityMapper, is a functionality that is outside the scope of this project, requiring more specialized, context-specific forecasting methods and potentially a whole new prediction framework.

The mobile application implements the combined regression, interpolation, weighted average model. The only information required to be inputted by the user is the start stop, end stop and route. The other information required to actually perform the prediction can be found using these given values.

4.5.1 Front End

The home page of the app provides text inputs for the user to input their start stop, end stop and bus route. Once the user presses 'Go', this leads the user to another screen which displays the predicted journey time. On this second page, there is also a button that allows the user to return to the first page to request another prediction. The entered start stop, end stop and route is sent to an endpoint using a POST request. The predicted journey time is fetched using a GET request from the same endpoint.

Some error checking in the front end is provided. The start and end stops must not be left empty and the route entered must be a number. Other error checking, such as whether the given stops are valid stops or actually on the given route, is performed in the back end. This is because knowing this information is required anyway to perform the model calculations, so it makes sense to do it all in one go, rather than in multiple places.

The front end has been written using Expo, which is an open-source platform using Javascript and React to make native apps for Android, iOS and the web. Expo was chosen because I had some prior experience using React for the 2nd year Webapps group project in the Department of Computing. Therefore, it was believed that some of this experience would carry over into mobile app development.

4.5.2 Back End

The prediction model requires the best α value, the part 1 coefficients, the part 1 intercept, the part 1 pre-processing scaler and the journey times of the last 10 journeys. The first four things listed have been pre-computed or pre-trained and so can just be imported using pickles.

The journey times of the last 10 journeys needs to be gathered live, therefore, data must be constantly collected in the background. The same code used to perform the data collection for the model training and exploration is used to collect data and write it into a database. Since only the last 10 journeys ever need to be used, the database deletes any data that is older than six hours ago. This six hour leeway is in case requests are made early in the morning on a non 24 hour bus route. For example if a request is made at 05:50 and the first bus of the day is at 05:24, some of the last 10 journeys will be from around midnight.

The JSON object returned from the back end is shown below. The 'success' term tells the front end whether to navigate to the results page or not, the 'time' term contains the predicted journey time and the 'stopError' term indicates whether there was a problem with the inputted start or end stops e.g. if they are misspelled.

```
predObj = {
        "success": True,
        "time": predTime,
        "stopError": False
    }
```

Flask, which is a lightweight web framework for Python, was chosen as to implement the back end in. This was because all of the data collection and modelling had been done in Python and since there was some overlap in the code that had to be used in the mobile app (i.e. the data collection and the model prediction), it made sense to to choose a back end that was already in Python. This way, the code would not have to be rewritten. Furthermore, since pickling was required to get the pre trained values such as the part 1 intercept, and pickles are native to Python, it made sense to keep everything within the same language. A postgres database was chosen to hold the data for the mobile app. Again, this was chosen because the data collection code had already

been written such that it wrote and read from a postgres database, therefore, it didn't seem prudent to change the code unnecessarily.

5 Implementation

5.1 Data Collection

There is no historical data for bus arrival times available online, so all the data that I will use to train and test my models has to be manually collected.

The following bus routes were chosen to collect data from: 6, 7, 9, 14, 35, 37, 52, 69, 267, 277, 328, 452. Of these routes, the 6, 14, 37, 52 and 69 bus routes are twenty four hour bus routes and the 37, 69, 267 and 277 bus routes do not travel through Zone 1. This allows there to be variety in the travel routes, traffic conditions and times of travel and so the data collected is not heavily biased towards particular areas of London.

1) Call Countdown API: The JSON information returned from the Countdown call is converted into a list of dictionaries, where each dictionary represents a single vehicle. The format of each dictionary is as below:

```
vehicle_info = {
            "vehicle_id": vehicle_id,
            "bus_stop_id": bus_stop_id,
            "direction": direction,
            "expected_arrival": eta,
            "time_of_req": time_of_request
        }
```

In the dictionary, the `vehicle_id`, `bus_stop_id` and `direction` are strings and the `expected_arrival` and `time_of_req` are Python `DateTime` objects.

2) Update the estimated arrival time of a bus: For each of the dictionaries, check if the vehicle corresponds to one already in the `bus_information_route` table. If it already exists, then update its expected arrival time, otherwise, it is a new vehicle to track and so needs to be added to the table.

3) Check if the bus has arrived or not: If the current time is after the predicted arrival time of the bus, then it is classified as 'due'. If the current time is 5 minutes after the predicted arrival time of the bus, then it is classified as 'arrived'. Return the 'due' and 'arrived' vehicles in separate lists so that they can be processed and written to the correct database.

4) Write to the relevant database: The 'due' buses have updated arrival times. Therefore, this information is updated in the `bus_information_route` table. The 'arrived' buses are removed from the `bus_information_route` table and added to the `bus_arrivals_route` table. When adding to the `bus_arrivals_route` table, check to see if on this day this particular vehicle is already in the table. It is necessary to check this because each vehicle will complete the bus route more than once per day, and therefore, the current trip number has to be kept track of too. This trip number is appended to the end of the `vehicle_id`.

5.2 Historical Models

According to the eight sub-models described in Section 4.3, the journey times were calculated for random pairs of stops and routes. The weighted average was then calculated, using the pre-discovered weightings. Using `sklearn`'s inbuilt functions, the mean absolute error (MAE) and root mean square error (RMSE) were calculated, first for journeys that are five stops apart. The best two sub-models that obtained the best combined RMSE and MAE were then tested using stops that are further apart.

5.3 Regression Models

The data collected from the tables is in the following format as shown in Figure 34. It must be processed into the format as shown in Figure 35 in order for it to be able to be analysed.

	vehicle_id	bus_stop_id	time_of_arrival	time_of_req	direction
0	10623_490003256N_2020-05-17_inbound_0	490003256N	2020-05-17 17:19:04	2020-05-17 17:19:17.892	inbound
1	10623_490006108E_2020-05-17_inbound_0	490006108E	2020-05-17 17:20:01	2020-05-17 17:19:24.951	inbound
2	10636_490006471E_2020-05-17_outbound_0	490006471E	2020-05-17 17:19:30	2020-05-17 17:19:27.03	outbound
3	10285_490013046M_2020-05-17_outbound_0	490013046M	2020-05-17 17:19:58	2020-05-17 17:19:36.662	outbound
4	11485_490000186A_2020-05-17_inbound_0	490000186A	2020-05-17 17:20:09	2020-05-17 17:19:37.928	inbound

Figure 34: Data before

The data collected has the time of arrival of a bus at a stop on a route, but it does not have the journey time since this would require it to be given a start and end stop. Therefore, pairs of stops across the routes were chosen to form journey routes. The pairs of stops were chosen so that there was variety in the gap sizes between the stops. The routes were also chosen so that there would be 24-hour bus routes as well as bus routes that don't run through Zone 1. The journey times were then put into a dataframe.

For each journey collected, the code looks at the time of arrival and converts the date into a day of week using Python's `calendar` library. Then, it converts this day of week into either a weekday or weekend. The code then checks if the time of arrival was between March 24th 2020 and 31st May 2020 or not. If it was in between those dates, then it categorises the entry as being during lockdown, else as pre/post-lockdown. The code also looks at the hour at which the bus arrives at the destination stop. This hour counted as the time of day of the journey. The resulting dataframe can be seen in Figure 35.

	Journey Time	Journey Time (s)	Day of Week	Time of Day	Pre Lockdown	Gap	Arrival at Stop A	Arrival at Stop B
0	00:06:54	414.0	weekday	3	False	5	2020-04-16 03:11:10	2020-04-16 03:18:04
1	00:04:36	276.0	weekday	5	False	5	2020-04-16 05:04:07	2020-04-16 05:08:43
2	00:04:33	273.0	weekday	5	False	5	2020-04-16 05:16:17	2020-04-16 05:20:50
3	00:05:00	300.0	weekday	5	False	5	2020-04-16 05:22:17	2020-04-16 05:27:17
4	00:05:00	300.0	weekday	5	False	5	2020-04-16 05:29:17	2020-04-16 05:34:17

Figure 35: dataframe of processed data

1) Calculate 'global' prediction: Before the data can be fed into a regression or interpolation model, the values must be encoded. So, using `sklearn`'s label encoder, the day of week and pre-lockdown are converted into categorical values. Based on Equation 17, the time of day has to be one-hot encoded, so this is also done using `sklearn`. The result of this can be seen in Figure 36.

index	Journey Time (s)	Gap	Day of Week Encoded	0	1	2	3	4	5	...	14	15	16	17	18	19	20	21	22	23
0	414.0	5	0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	276.0	5	0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	273.0	5	0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	300.0	5	0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	300.0	5	0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 36: Data for part 1 regression

The features are scaled and then fed into a linear regression model for training.

2) Calculate 'local' prediction: For the three sub-models, it is necessary to get the last 10 journeys for each request time and pair of stops, as well as how long ago the journey was in relation to the journey time. An example can be seen in Figure 37, where 'Bus 0' is the most recent bus to have completed the journey and 'Bus 0 time to req' is how long ago in seconds this bus completed that journey relative to the request time. The columns go all the way down to 'Bus 9' and 'Bus 9 time to req'.

	Bus 0	Bus 0 time to req	Bus 1	Bus 1 time to req	Bus 2	Bus 2 time to req	Bus 3	Bus 3 time to req	Bus 4	Bus 4 time to req	...
1996	339.0	343.0	339.0	343.0	326.0	1285.0	326.0	1285.0	210.0	3340.0	...
1322	405.0	442.0	487.0	1080.0	412.0	1876.0	394.0	2613.0	403.0	3329.0	...
1970	330.0	435.0	330.0	435.0	425.0	1073.0	425.0	1073.0	463.0	1446.0	...

Figure 37: Data for part 2 sub-model

5.4 Mobile App

On front end, three `TextInputs` are used to allow the user to enter the start stop, end stop and route. On the press of the 'Go' button, this sends a POST request to the *predict* endpoint with the typed in information in one object stored in JSON format. The Home page is seen in Figure 38.
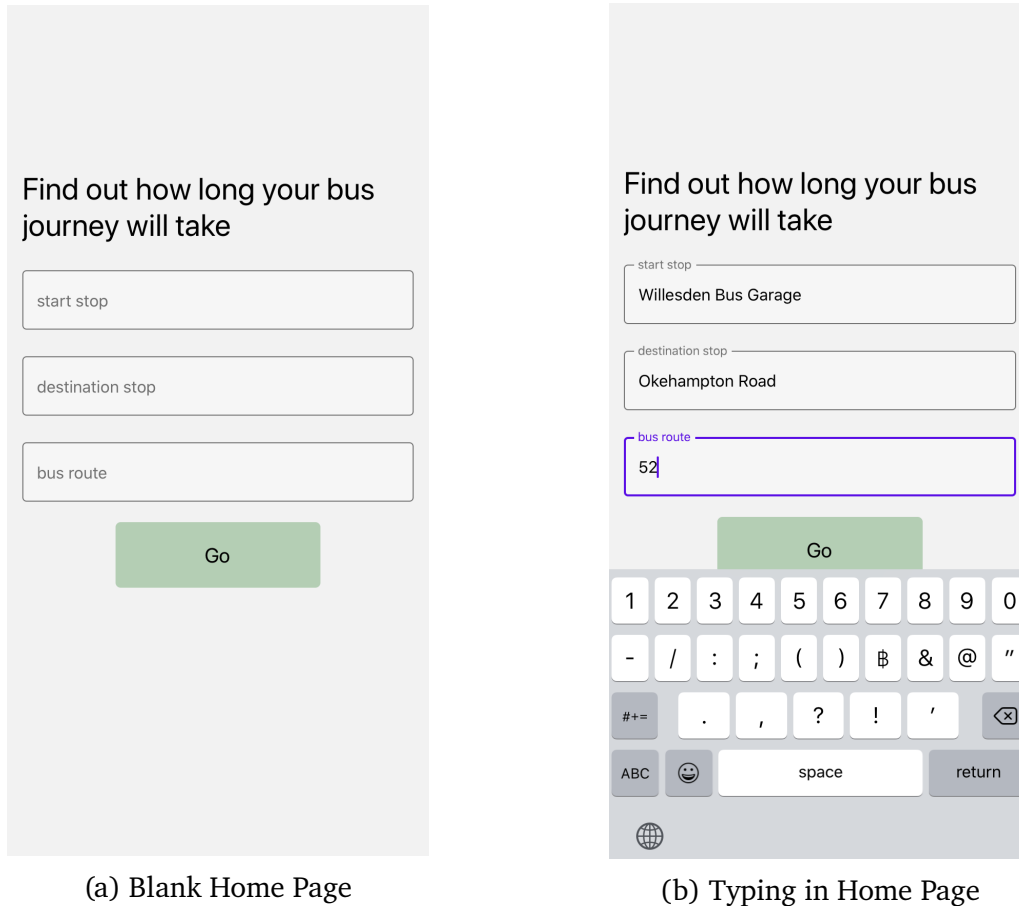

(a) Blank Home Page

(b) Typing in Home Page

Figure 38: Mobile Home Page

On the back end, if the *predict* endpoint detects a POST request, it gets the information out of the JSON object.

The parameters required to do the part 1 prediction are the gap size, the day of week as a binary encoding and the time of day as a one hot encoding. Since it is assumed that the request time is the time the user entered the information into the app, a `datetime.datetime.now()` object is used to find the day of week and time of day. The found day of week and time of day are then encoded accordingly. TfL's Stoppoints Sequence API is used to find a list of the stops in sequence for the given route. Then, the gap size is found by subtracting the indexes of the start point from the end point. If the entered start or end stop cannot be found in the list of stops in sequences, this indicates that the user has entered a stop that either has been misspelled or is not on the given route. Therefore, the 'stopError' term in the JSON object is set to `True`. Using the

encoded time of day and day of week and found gap size, the information is scaled and multiplied to the pickled part 1 coefficients. Finally, this number is added to the pickled part 1 intercept to obtained the part 1 prediction.

To calculate the part 2 prediction, the only information needed is the journey times of the last 10 buses. The database is queried for the last 10 buses to arrive at the two given stops and then their arrival times are deducted to give the journey times. The weighted average is then found, using the same weights as from the model training.

To get the overall prediction, the pickled alpha value is used as per Equation 21. The 'time' term in the JSON object is then set to this value and the object is returned to the end point. On the front end, this is received as the response and then passed as a parameter to the prediction page to be displayed. The prediction page is shown in Figure 39.
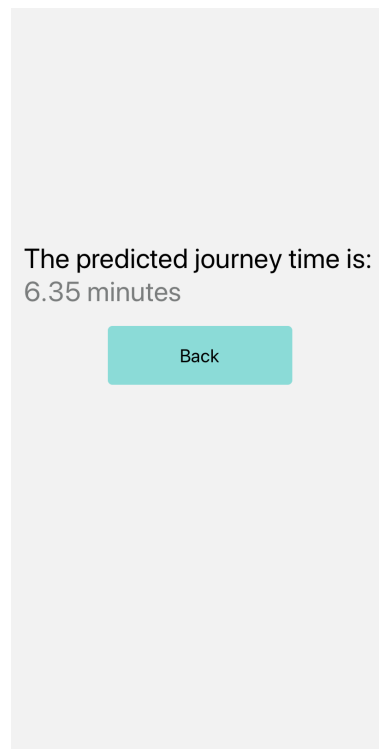

The predicted journey time is:
6.35 minutes

Back

Figure 39: Prediction Page

6 Results and Evaluation

One of the objectives of this project is to react to events, which is why recent information is crucial to all of the models developed. Consider the two contrasting examples: 1) if there has been a collision on the road, all buses that use this route will be delayed compared to 2) if a bus driver is in an argument with one of the passengers getting on the bus, then only that particular bus would be delayed, not any of the other buses on the route. In the first case, the desired outcome is indeed to react to this increase in journey time and provide predictions with longer than usual journey times. However, in the second case, the 'delay' is localised to that particular bus and so predictions should not provide predictions with longer than usual journey times. Therefore, it is possible to overreact to situations and adjust the prediction when there is not an actual cause for the model to do so.

In the context of bus journey times, the mean absolute error (MAE) and the root mean squared error (RMSE) mean different things and provide different insights. The MAE results describe how many seconds off the actual journey time a model consistently predicts. For example, a MAE of 100 seconds implies that the model consistently predicts a journey time of 100 seconds more or less than the actual journey time (although it does not provide insight into whether the model over or under predicts). On the other hand, since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This penalization of large errors could be appropriate if for example, a prediction that is four minutes off is more than twice as bad as a prediction that is two minutes off. This brings to question, what counts as a 'good' bus journey prediction?

So, is a model that predicts journeys that are consistently 3 minutes off better than a model that predicts journeys that are 1 minute off occasionally and 5 minutes off occasionally? In the long run, waiting an extra minute or two doesn't make much of a difference. However, waiting upwards of 5 minutes is significant. This suggests that the RMSE results are more important than the MAE results. On the other hand, if waiting 5 extra minutes is no more than twice as bad as waiting around 2 extra minutes, then the MAE is actually more appropriate. Therefore since this is a fairly subjective issue, for this project, both the MAE and RMSE will be presented.

6.1 Historical Model

For the training process, the models are first trained on journeys between stops of gap 5 before being trained on journeys between stops of all sizes. The sample set used for the gap of 5 initial model exploration was for all journeys between CHESTERTON ROAD and NOTTING HILL GATE STATION on route 52. The amount of training data is approximately 3,000 entries. The sample set with all sizes of gaps used journey times for seven pairs of stops on route 52, amounting to approximately 18,000 entries.

First consider the results for journeys between stops that are five apart.

Figure 40 shows the predicted journey time versus the actual journey time versus

TfL's own predicted journey time. This figure is for buses on route 52 for journeys between Chesterton Road to Notting Hill Gate Station, where the predictions have been found using the historical model that looks back at buses in the past two hours.
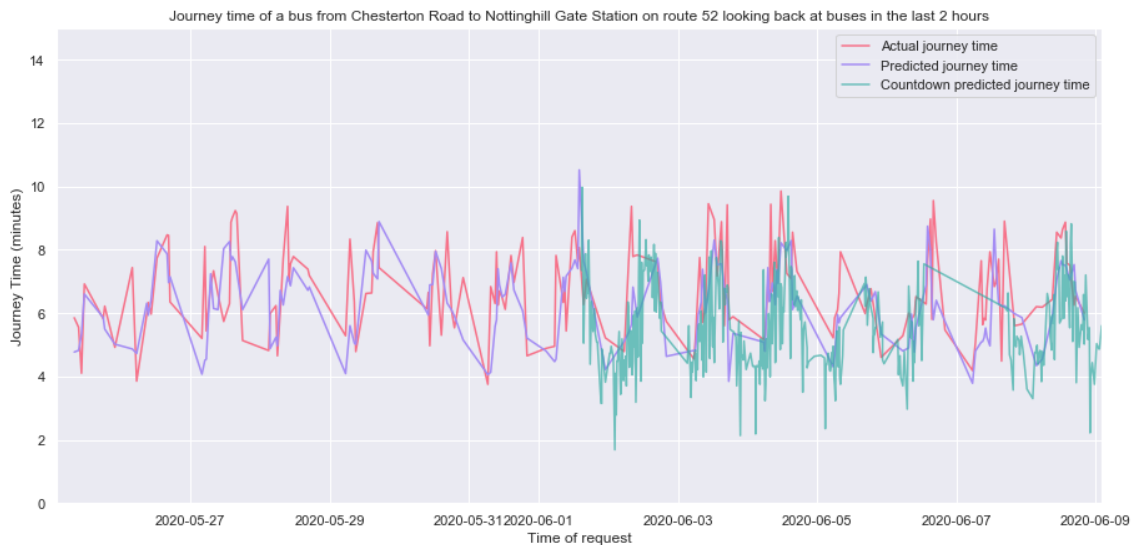

Figure 40: Historical Model looking at buses from past two hours

Graphs such as in Figure 40 were also produced for predictions that were found using the historical models that looked back at the past 15 minutes, 30 minutes and 60 minutes. The MAE and RMSE were calculated for all four models that look back X amount of time. The results can be seen in Figure 41.
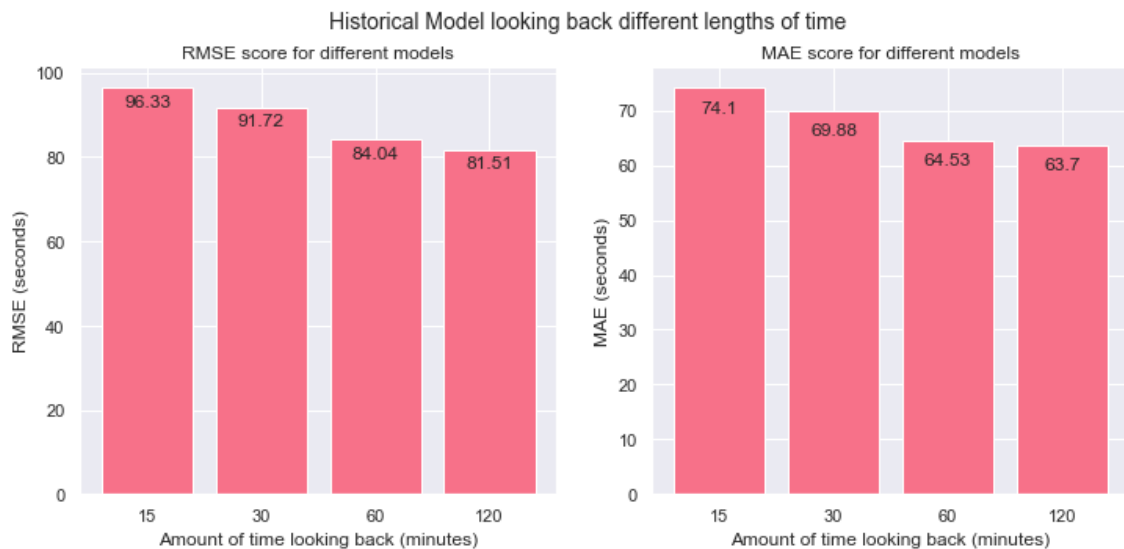

Figure 41: Historical Model looking back at x amount of time

Consider the model that looks back at journeys from the past 120 minutes. Figure 41

57

indicates that based on the MAE, the predictions are on average about a minute higher or lower than the actual journey time. This is an excellent result, however, bear in mind that this is only for stops that are five apart. It will be shown later that when these models are applied to stops that are further apart, the MAE and RMSE grow. Since the RMSE is only about 20 seconds more than the MAE, this suggests that either predictions that are significantly off don't occur very often or predictions are generally close to 1 minute off.

Intuitively, it makes sense that the model that looks back at the last 15 minutes performs the worst out of the four sub-models presented above. This is because looking at only the past 15 minutes means the probability of overreacting to an event is much higher. In particular, Figure 40 demonstrates that the actual journey time ranges between 4 minutes to 10 minutes and so when the model looks back at the last 15 minutes, it will be looking at most at the last three buses. In the worst case, it would only look back at the last bus. This means that if there is something that causes only that particular bus to delay, the model will not realise that this is an anomalous journey time and so will predict a longer journey time when it should not have.

Figure 42 shows the predicted journey time versus the actual journey time versus TfL's own predicted journey time. This figure is also for buses on route 52 for journeys between CHESTERTON ROAD to NOTTING HILL GATE STATION. However, here the predictions have been found using the historical model that looks back at the last two buses to complete the journey.
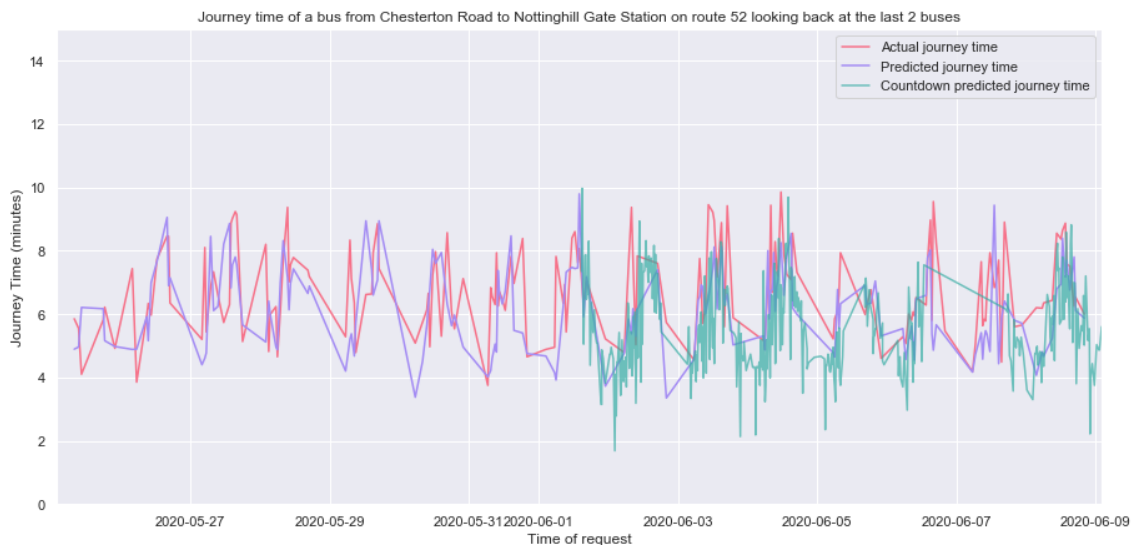

Figure 42: Historical Model looking at past two buses

Graphs such as in Figure 42 were also produced for predictions that were found using the historical models that looked back at the past 5 buses, 10 buses and 15 buses. The MAE and RMSE were calculated and can be seen in Figure 43.

Figure 43: Historical Model looking back at x number of buses

Consider the model that looks back at the last five buses. Based on the MAE, Figure 43 implies that predictions are on average about a minute more or less than the actual journey time. However, as before, these are results for bus stops that are five apart and so should not be treated as conclusive results.

In a similar argument to earlier, it makes sense for the model that looks back at the last two buses to perform the worst out of the four sub-models presented above. This is because of its high probability of overreacting to events. On the other hand, it also makes sense that the model that looks back at the last 15 buses doesn't perform as well as the one that looks back at the last 5 or 10 buses. This is because this model is more likely to take journey times that are no longer relevant into account. For example, consider the following case: a bus is not a 24 hour bus route - its last bus is at 00:30 and its first bus is at 05:25. If a prediction is requested at 05:40, the last 15 buses will include bus journeys made around midnight or before. As mentioned in Section 4.2, at different times of day the journey times are not the same. Therefore, the prediction could be less than it should be. Furthermore, if there was some disruption that caused the bus journeys around midnight to be significantly longer or shorter, this too could adversely affect the prediction. However, since its performance is not actually significantly worse than the model that looks back at the last 5 or 10 buses, this could be put down to the weights that were chosen that ensured that bus journeys long before the request time do not have as much impact on the prediction.

Figure 44 shows all eight sub-models' MAE and RMSE for stops that are five apart compared against each other. According to the RMSE the best two models are 1) looking back 10 buses 2) looking back 5 buses. According to the MAE the best two models are 1) looking back 5 buses 2) looking back 10 buses. Overall, this means that the top two sub-models, found by averaging the MAE and RMSE are 1) looking back 10 buses 2) looking back 5 buses. The third and fourth best sub-models overall are 3) looking back
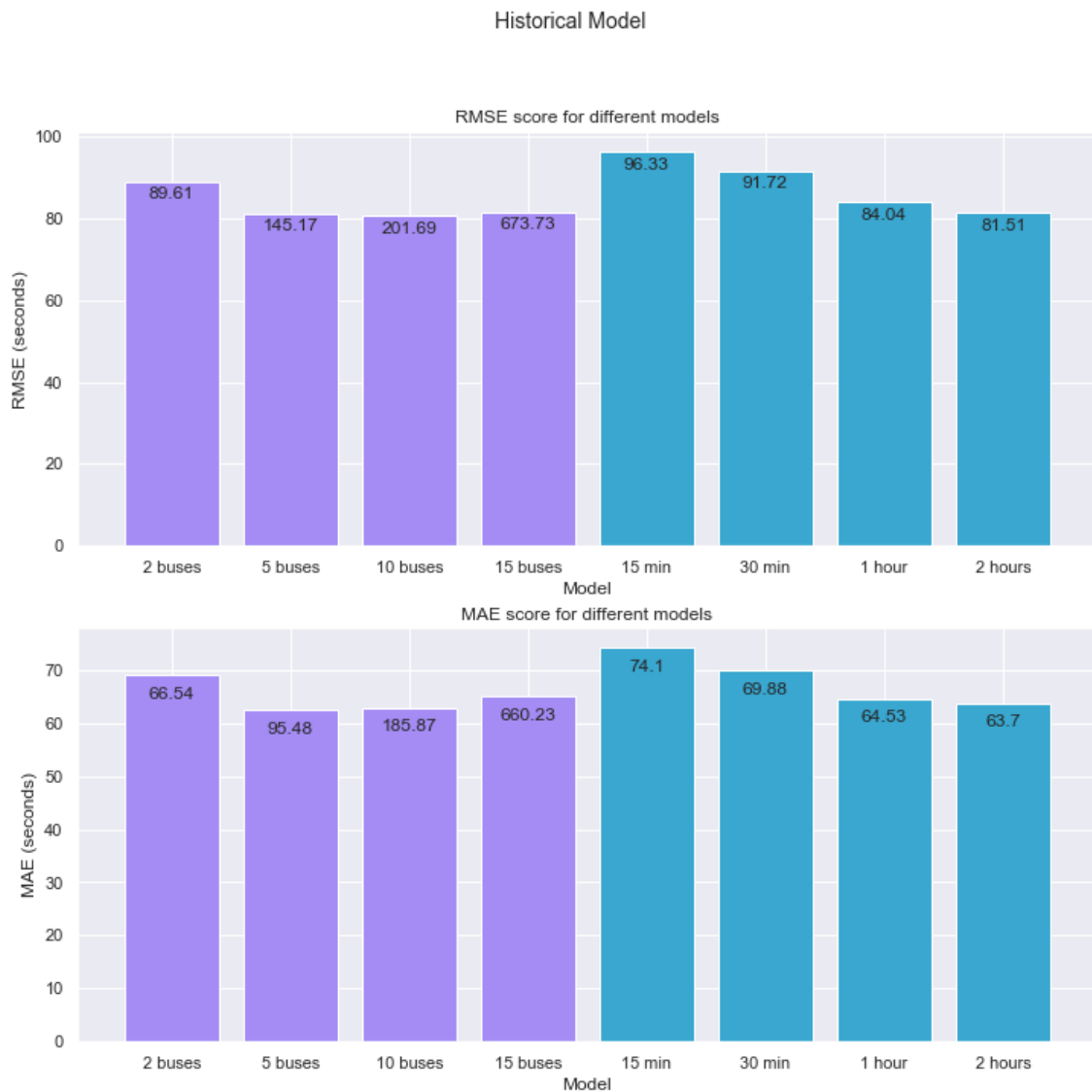
2 hours 4) looking back 15 buses.

Figure 44: Historical Model

Figure 44 shows that the models that look back a certain amount of time have a larger range than the models that look back a certain amount of time. This suggests that the models that look back a certain amount of time are more likely to produce predictions that are further away from the actual value. Therefore, it can be concluded that looking back a specified number of buses is more likely to produce better results than looking back a specified amount of time. The ranges are as follows:

- Historical model looking back different amounts of time: RMSE range = 14.82 seconds, MAE range = 10.4 seconds.

- Historical model looking back different numbers of buses: RMSE range = 8.41 seconds, MAE range = 6.67 seconds.

The best two models were applied to pairs of stops that are larger than five apart. The results for the model that looks back 10 stops are seen in Figure 45 and the results for the model that looks back 5 stops are seen in Figure 46.

Figure 45: Looking back at the last 10 journeys

Figure 46: Looking back at the last 5 journeys

It can be seen that for both models, as the distance between the stops grow, so too does the RMSE and MAE. This makes sense because as the stops grow further apart, the standard deviation and variance of the actual journey times also increase. This means that there is a lot more fluctuation in the training times given to the models to calculate a weighted average out of. It can be seen that up to fifteen stops apart, the performance

can still be deemed as good, however, it is at gaps of around twenty that the performance drops significantly.

Taking the average of the MAE and RMSE for the different gap sizes for both models gives the following results:

- Historical model looking back at the last 5 buses: RMSE = 277.55 (5sf), MAE = 252.03 (5sf)

- Historical model looking back at the last 10 buses: RMSE = 280.35 (5sf), MAE = 254.86 (5sf)

It can be seen that the two models have very similar results when looking at their performance across all gap sizes. This suggests that overall, looking back at the last five or ten buses does not make much of a difference in predicting journey times.

6.2 Regression + Interpolation Model

For the training process, all journeys between twenty-one pairs of stops over three routes were chosen. The pairs of stops were chosen such that there was variety in gap size and variety in time and area of travel, with one of the routes not going through Zone 1 and two of the routes being 24h routes. The sample set amounted to approximately 22,000 entries.

Part 1 of the regression + interpolation model calculates the 'global' prediction. The regression model was evaluated on requests made from 20/04/2020 01:00:00 to 08/06/2020 23:59:59 at thirty minute intervals for a number of different pairs of stops of varying routes and gap sizes. The root mean squared error (RMSE), R2-score and mean absolute error (MAE) were then calculated and the results are shown below:

- RMSE: 343.09 seconds (5sf)

- R2-score: 0.86865 (5sf)

- MAE: 224.58 seconds (5sf)

Since an R2-score of 1 means a perfect prediction and the R2-score is bounded between 0 and 1, a score of 0.85749 indicates that the model gives a fairly good prediction of the journey time. However, the MAE and RMSE score indicate that the predictions can be around 4 and 6 minutes off respectively, which doesn't seem too great. This supports the hypothesis that recent data, i.e. the 'local' conditions, need to also be taken into account as looking at these 'global' factors alone does not provide an accurate enough prediction. However, it should be noted that this result is for gaps of all sizes on a bus route, and so if the journey that is being made is normally around 40 minutes, an extra 4 - 6 minutes doesn't actually make much of a difference - this would be an extra 10 - 15% journey time.

- Coefficients: [[7.78320354e+02, -2.15578663e+01, 5.73810236e+12,
  6.16413662e+12, 6.90322104e+12, 9.84142661e+12, 1.12486687e+13,
  1.41533295e+13, 1.78363604e+13, 1.87106607e+13, 1.92728119e+13,
  2.02578876e+13, 2.12199016e+13, 2.17152238e+13, 2.01181975e+13,
  1.96601961e+13, 2.00780471e+13, 2.12757351e+13, 2.06500324e+13,
  1.84661903e+13, 1.53716651e+13, 1.15428232e+13, 1.10275325e+13,
  9.32177162e+12, 1.05914818e+13, 6.93639804e+12]]

- Intercept: [1078.31645828]

Part 2 of the regression + interpolation model are the five different sub-models that calculate the 'local' prediction. Figure 47 shows the pink dots as the actual journey times of the past 10 journeys at a particular stop. The prediction was requested at 05:00 on April 20th 2020. The x axis shows the time of the journeys before the request in seconds, for example, the first pink dot indicates that the most recent journey time was around 400 seconds long and occurred approximately 500 seconds or 8 minutes ago. The predictions made by the different models have been drawn out in straight lines so that they can be more easily seen relative to the past 10 journey times. It can be seen that the weighted average model provides the most intuitive prediction. There is no clear upward or down trend in the past 10 journey times so a predicted journey time that does not fall within the range of previous values, as the other four sub models gives, does not make sense.
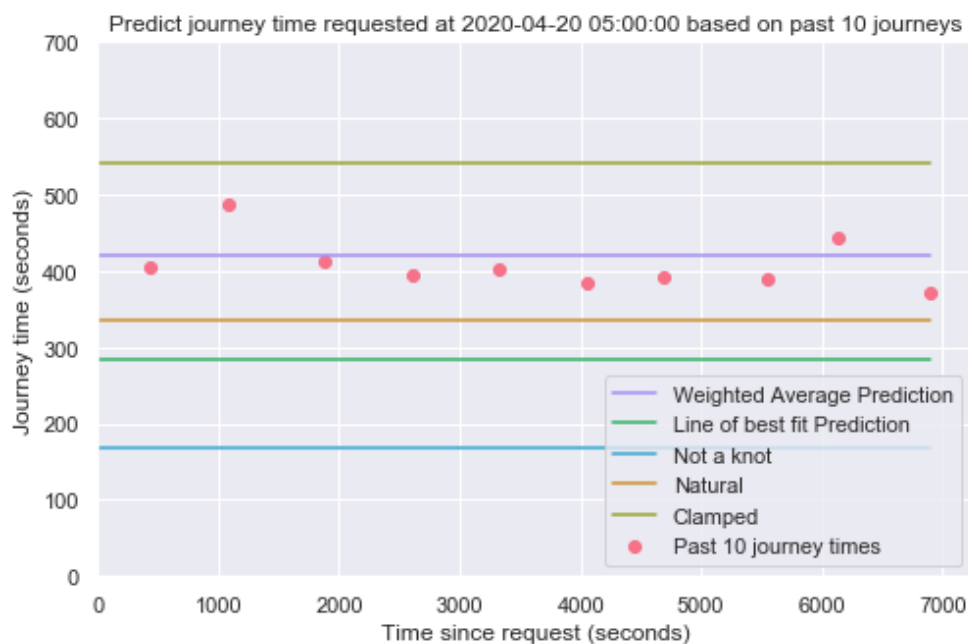

Figure 47: Part 2 prediction comparison

The combined regression + interpolation model that gives the final prediction result uses linear interpolation of y_{pred1} from the 'global' prediction and y_{pred2} from the 'local'

prediction. The α parameter needs to be tuned and the resulting RMSE and MAE scores are seen in Figures 48, 49 and 50.

Figure 48 shows how the RMSE and MAE changes for the combined model where the part 2 model is the weighted average model. A multi-stage grid search cross validation was performed to find the optimal α. The initial range was twenty steps from 0 to 1, however, results showed that the dip in the MAE and RMSE occurred closer to 0, and hence the values that it was searched over was changed to twenty steps between 0 and 0.2. This is more efficient than searching 100 steps between 0 and 1, which is the number required to get the same resolution. The optimal result comes at $\alpha = 0.094737$ (5sf), giving a MAE of 157.69 (5sf) or $\alpha = 0.031579$ (5sf), giving a RMSE of 267.87 (5sf). A value of α close to 0 implies that y_{pred1} is not as important as y_{pred2}. In other words, this result says that the predicted journey time of a bus relies mostly on the journey times of the recent buses and does not rely as much on 'global' factors such as the time of day or the day of the week.
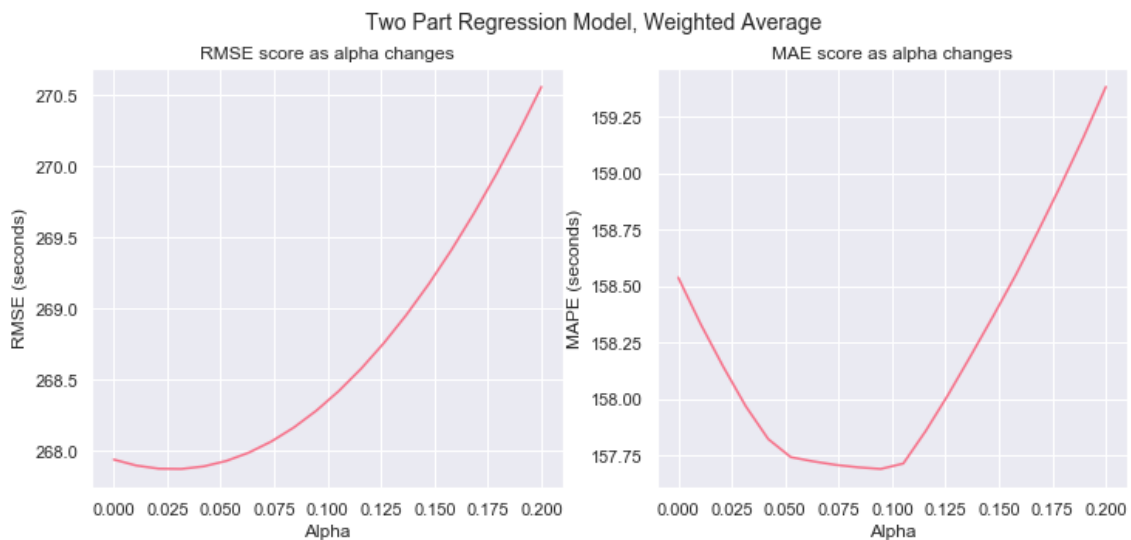

Figure 48: Part 2 = Weighted Average

Figure 49 shows how the RMSE and MAE changes for the combined model where the part 2 model is the linear regression model. α was initially varied from 0 to 1, however, results showed that the dip in the MAE and RMSE occurred closer to 1, and hence the values that it was varied over was changed to between 0.8 and 1. The optimal result comes at $\alpha = 0.98947$ (5sf), giving a MAE of 220.19 (5sf) or $\alpha = 0.97895$ (5sf), giving a RMSE of 339.23 (5sf). A value of α close to 1 implies that y_{pred2} is not as important as y_{pred1}. In other words, this result says that the predicted journey time of a bus relies mostly on 'global' factors such as the time of day or the day of week, and doesn't rely as much on recent journey times.
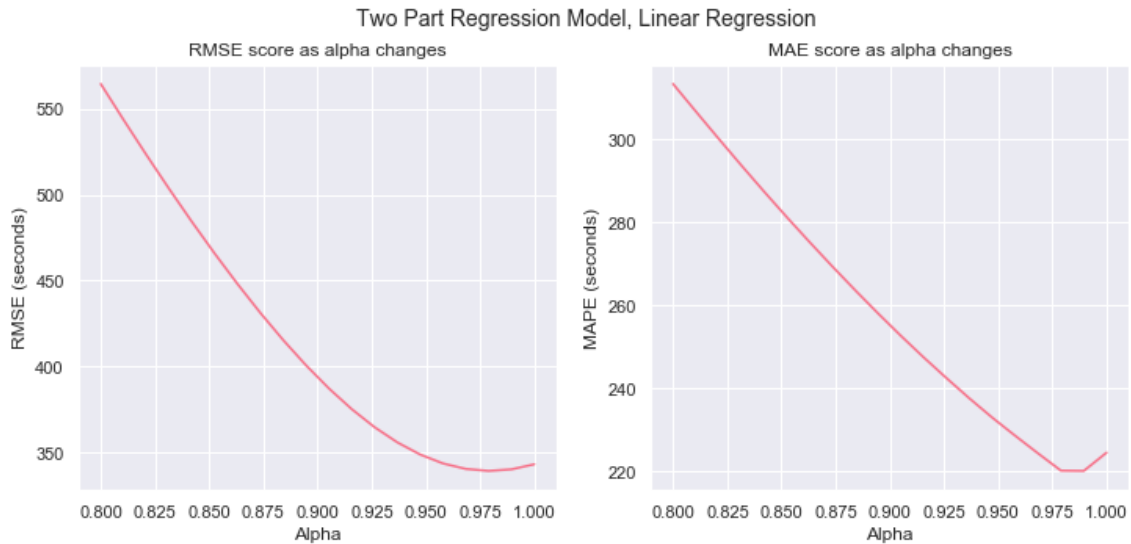
64

Figure 49: Part 2 = Linear regression

Figure 50 shows how the RMSE and MAE changes for the combined model where the part 2 model is the three different cubic spline sub-models. α was varied from 0 to 1, but it can be seen that the MAE and RMSE are significantly big (note the scale of the y axis). Therefore, cubic spline sub-models are not suitable to be used as the part 2 prediction model. Due to the nature of cubic splines and interpolations of polynomials of degree greater than two, it makes sense that they do not provide the best results. In particular, when the last 10 journeys are further away from the time of request, the intercept of the cubic spline polynomial will be at an extreme value (for example in the case where a request is made at 05:30 and the last 10 journeys are from midnight or before).
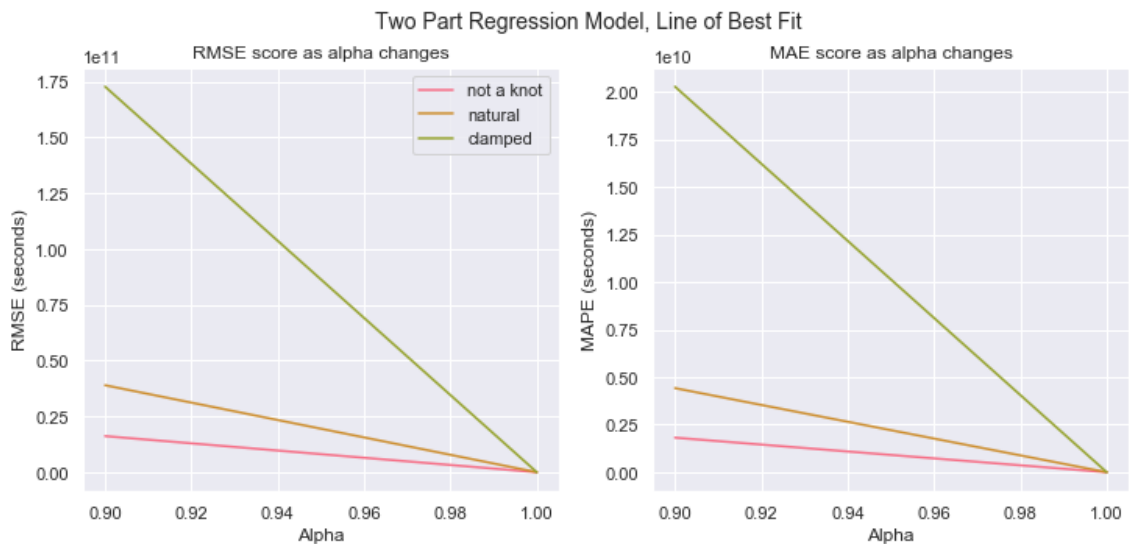

Figure 50: Part 2 = Cubic Spline

The initial hypothesis was that α would be in the region of 0.5. However, the results gotten have α at the extremes near 0 or 1. This does not support the belief that the journey time of a bus relies equally on 'global' factors and 'local' factors. In Figure 51 it can be seen that the combined model with the weighted average model as the part 2 model out performs linear regression as the part 2 model. In this case, the value of α that led to the optimal model is closer to 0, implying that bus journey prediction models should rely more on recent journey times. This supports the argument that dynamic models are more suitable than static models.

Figure 51: All Gap size comparison

Figure 52 shows the prediction made by the best combined regression model (where the part 2 model is a weighted average model and $\alpha = 0.97895$) vs TfL Countdown's predictions versus the actual journey times. This particular graph is for stops that are 5 apart, so that it is more comparable to the historical model. The figure shows that the combined model regularly under predicts the journey time.
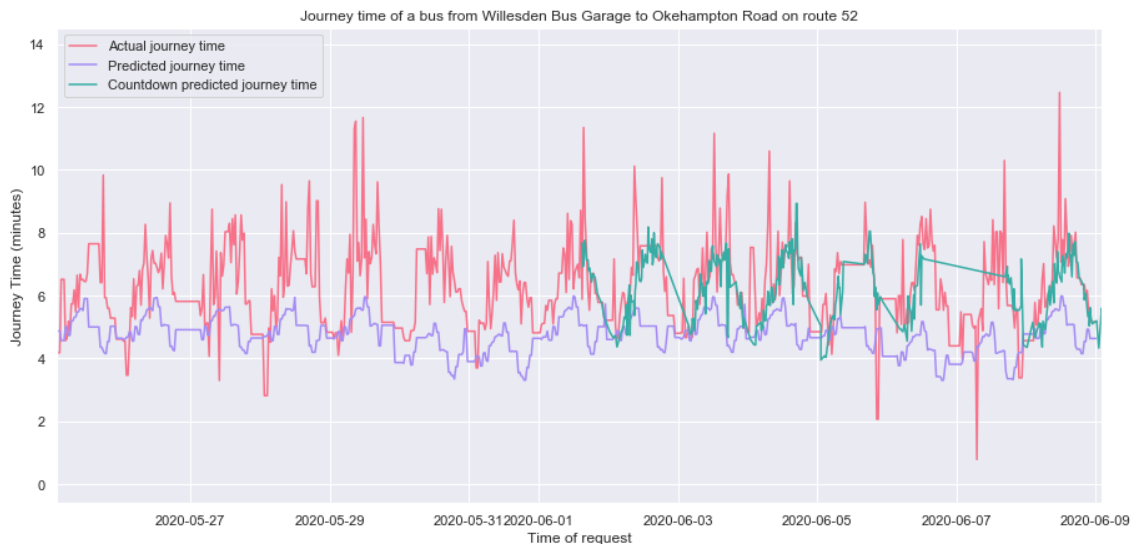
Figure 52: Regression Model

6.3 Cross Model Comparison

Figure 53 shows the RMSE and MAE of all the 8 complete models compared against TfL Countdown's scores. The scores are for stops that are 5 apart.

By RMSE, Figure 53 shows that the best model is the combined model with the part 2 model being the weighted average model. This model also outperforms TfL Countdown's model, which is one of the aims that this project set out to do. Therefore, this can be seen as a success. However, although this model is also the best model according to MAE, it does not outperform TfL Countdown's model using this measure. Outperforming on the RMSE means that this model is less likely than the TfL model to predict values that are significantly larger or smaller than the actual value. However, under performing on the MAE means this model is not as consistent in its predictions as the TfL model. It should be noted that the difference between the scores for the two models is very small, approximately 5 seconds for both MAE and RMSE. Therefore, it could make sense to conclude that the two models are equally good. Furthermore, the range of scores achieved for stops that are five apart is not large, with the scores suggesting that all of the models are predicting within two minutes of the actual time.
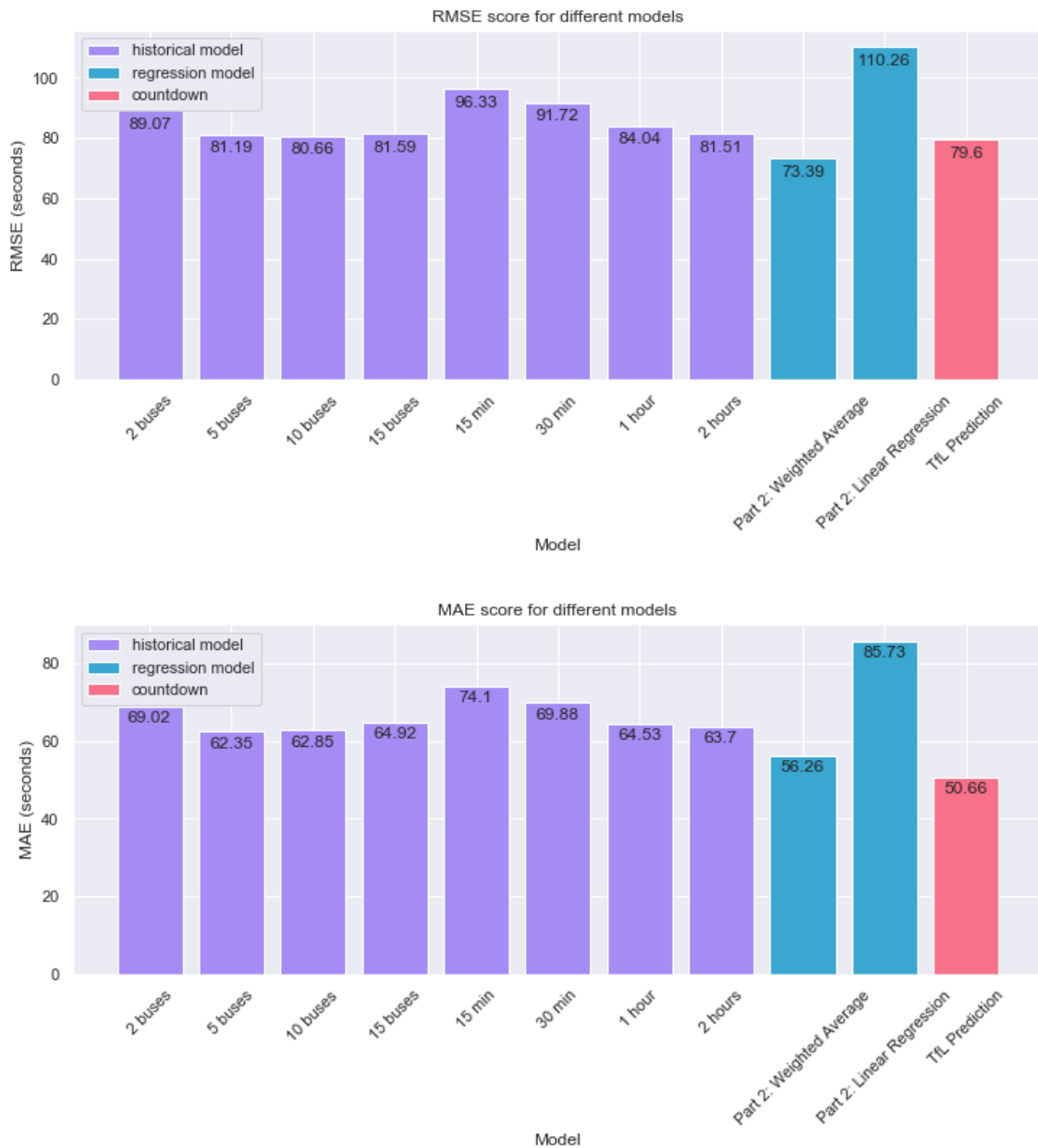
Figure 53: All Model Comparison Gap 5

Figure 54 shows the RMSE and MAE of all the top two historical models and top two regression models compared against TfL Countdown's scores. The scores are for stops of all gap sizes.

Unlike with looking at just stops that are five apart, Figure 54 shows that the four models presented all out perform TfL Countdown's model on both MAE and RMSE. This is a great achievement as this was one of the goals of the project - to be able to create a

model that is better than TfL's own. According to the MAE, the four models developed in this project predict results that are four minutes, four minutes, two and a half minutes and three and a half minutes respectively off from the actual time. Realistically, for journeys that take longer than fifteen minutes, an extra two to four minutes does not make much of a difference. TfL's predictions on the other hand are on average about six minutes off from the actual time. Since the RMSE indicates when bigger errors are present, Figure 54 suggests that the two historical models and the combined model with the weighted average model as the part 2 model, are fairly equal in not giving outlandishly larger or small predictions. However, by the MAE it is clear that the combined model with the weighted average model as the part 2 model far exceeds the predictions made with the other models.
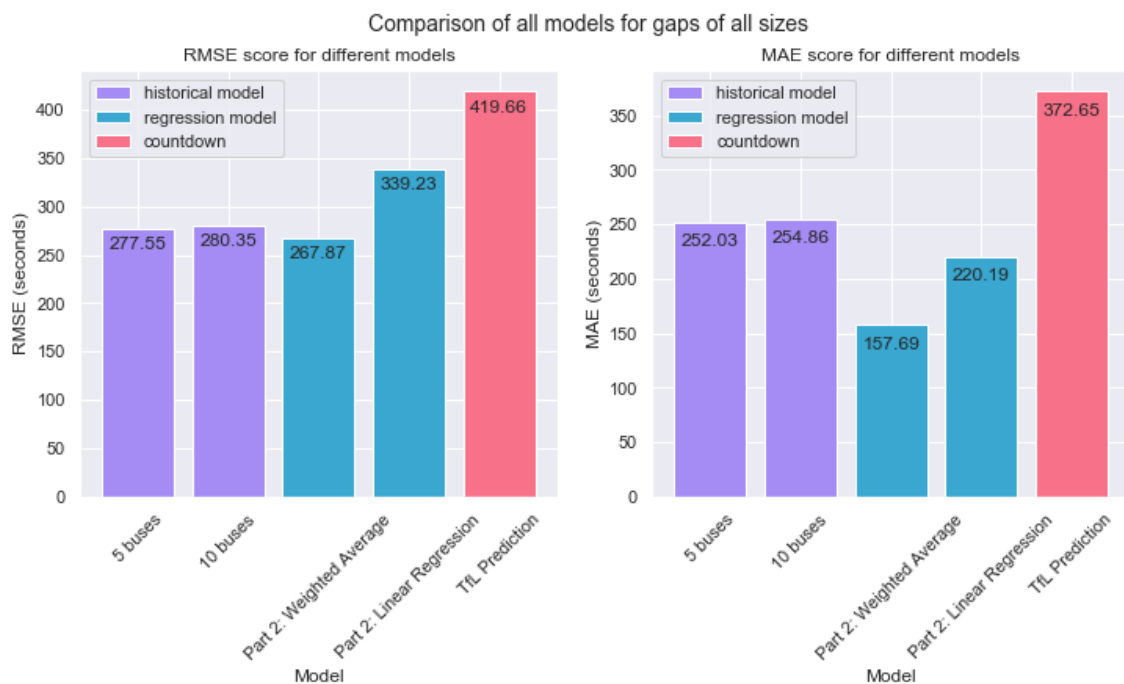

Figure 54: All Model Comparison

It is surprising that the historical models that do not take any of the 'global' factors into account still perform fairly well. This could perhaps be explained by the idea that the recent journeys already take into account the 'global' factors such as time of day or traffic conditions without these factors having to be explicitly found.

7 Conclusions

In this project, 10 fully developed models that predict bus journey times in London have been presented. The best performing model is a combined regression, interpolation and weighted average model that is able to outperform Transport for London's own predictions using the root mean squared error and mean absolute error as the measure of performance. For stops that are close to each other, the predicted journey time is usually within one minute of the actual journey time. For stops that are further away, this error increases such that on average the model predicts a journey time that is within two and a half minutes of the actual journey time. As a bus user, this is not a bad result at all - two to three minutes does not generally make a difference to travel time. Furthermore, this model fulfills the project objective of actively reacting to delays.

Currently, the models developed in this project are only compared against each other and against TfL Countdown's own predictions. None of the models made use of traffic conditions and it has been assumed that Countdown also does not look at factors such as congestion. On the other hand, other applications such as CityMapper or Google Maps provide adjustable predicted journey times based on traffic conditions. So, in the future, it would be interesting to see a direct comparison of this project's models against CityMapper and Google Map's predictions. This should provide insight into whether traffic conditions is necessary for vehicle journey predictions. Furthermore, if the phone app is extended such that predictions from different sources can be displayed side by side, this could be useful for users so that they can have multiple predictions to choose from.

It was hypothesised that the journey time of a bus would rely equally on both 'local' conditions and 'global' conditions. This was not proven correct as the best model implied that 'local' conditions are much more important than 'global' ones. However, due to limited time and resources, it was not possible to collect data on other 'global' factors such as weather or passenger counts at each bus stop. Therefore, it would be interesting to see if including these factors in the part 1 regression model would lead to an α value that was nearer to 0.5. In other words, whether including more 'global' factors would allow this hypothesis to be true.

Artificial Neural Networks were a type of model that was not prioritised for this project, although good results had been found by other papers that implemented this method. However, given more time, it would be interesting to see how an ANN model would compare to the historical average and regression + interpolation combined model.

Finally, since the data used for the modelling was all collected manually, there is a high possibility of human error in the code written to do the collecting. Therefore, it is believed that with official historical data, these models could perform even better.

References

[1] Quarterly bus statistics: April to June 2019. `https://www.gov.uk/government/statistics/quarterly-bus-statistics-april-to-june-2019`. Accessed: 2020-01-15.

[2] Annual Bus Statistics: England 2017/18. `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/774565/annual-bus-statistics-year-ending-mar-2018.pdf`, January 2019. Accessed: 2020-01-15.

[3] Mayor sets out ambitious plan to persuade Londoners to reduce car use. `https://www.london.gov.uk/press-releases/mayoral/plan-to-persuade-londoners-to-reduce-car-use`, June 2017. Accessed: 2020-01-15.

[4] Mobility - RAC Foundation. `https://www.racfoundation.org/motoring-faqs/mobility`. Accessed: 2020-01-15.

[5] Mayor launches plans for London's biggest Car Free Day celebrations. `https://www.london.gov.uk/press-releases/mayoral/londons-biggest-ever-car-free-day`, June 2019. Accessed: 2020-01-15.

[6] R. Jeong and R. Rilett. Bus arrival time prediction using artificial neural network model. In *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, pages 353–366. IEEE, 2004.

[7] Mark J.Koetse and Piet Rietveld. The impact of climate change and weather on transport: an overview of empirical findings. *Transportation Research Part D: Transport and Environment*, Volume 14 Issue 03:205–221, May 2009.

[8] Alex Fabrikant. Predicting Bus Delays with Machine Learning. `https://ai.googleblog.com/2019/06/predicting-bus-delays-with-machine.html`, June 2019. Accessed: 2020-01-15.

[9] Jithendra. H .K and Naga Ravi Kanth Devarapalli. Predicting bus arrival time based on traffic modelling and real-time delay. *International Journal of Engineering Research & Technology (IJERT)*, Volume 4 Issue 06:421–426, June 2015.

[10] Wei Fan and Zegeye Gurmu. Dynamic travel time prediction models for buses using only gps data. *International Journal of Transportation Science and Technology*, Volume 4 Issue 4:353–366, 2015.

[11] Jules White Fangzhou Sun, Yao Pan and Abhishek Dubey. Real-time and predictive analytics for smart public transportation decision support system. Technical report, Institute of Software Integrated Systems, Department of EECS, Vanderbilt University, Nashville, TN, USA, 2016.

[12] Transport for London Unified API Documentation. `https://api-portal.tfl.gov.uk/docs`. Accessed: 2020-02-08.

[13] Steven Chien Jayakrishna Patnaik and Athanassios Bladikas. Estimation of bus arrival times using apc data. *Journal of Public Transportation*, Volume 7 Issue 1:1–20, 2004.

[14] Sunil Ray. 7 Regression Techniques you should know! `https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/`, August 2015. Accessed: 2020-01-27.

[15] TfL Live Bus & River Bus Arrivals API Interface Documentation. `http://content.tfl.gov.uk/tfl-live-bus-river-bus-arrivals-api-documentation.pdf`. Accessed: 2020-04-30.

[16] Sujit Singh. What is Statistical Forecasting? A snowfall-based explanation. `https://blog.arkieva.com/statistical-forecasting-definition/`, April 2018. Accessed: 2020-01-22.

[17] Rob J Hyndman and George Athanasopoulos. Forecasting: Principles and Practice. `https://otexts.com/fpp2/`, January 2020. Accessed: 2020-01-27.

[18] Ronald Klimberg, George Sillup, Kevin Boyle, and Vinay Tavva. Forecasting performance measures - what are their practical meaning? *Advances in Business and Management Forecasting*, 7:137–147, November 2010.

[19] Student. Probable error of a correlation coefficient. *Biometrika*, Volume 6:302–310, September 1908.

[20] Laerd Statistics: Pearson's product moment correlation. Statistical tutorials and software guides. `https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php`. Accessed: 2020-06-11.

[21] Youngjoo Kim and Hyochoong Bang. Introduction to Kalman Filter and Its Applications. `https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications`, November 2018. Accessed: 2020-02-02.

[22] Lim Tien Sze Lim Chot Hun, Ong Lee Yeng and Koo Voon Chet. Kalman Filtering and Its Real-Time Applications. `https://cdn.intechopen.com/pdfs/50419.pdf`, June 2016. Accessed: 2020-02-02.

[23] C. Lim J. Houghton, J. Reiners. Intelligent transport system: how cities can improve mobility. *IBM Institute for Business Value*, 2009.

[24] Robert L. Peach. Artificial Neural Networks, Methods for Data Science, MATH96007/MATH97019/MATH97097 Imperial College Mathematics Department. `https://bb.imperial.ac.uk/bbcswebdav/pid-1701938-dt-content-rid-5490698_1/courses/JY201910/lectures_ANN_1.pdf`, November 2019. Accessed: 2020-01-24.

[25] Michael A. Nielsen. Neural Networks and Deep Learning. `http://neuralnetworksanddeeplearning.com/chap1.html`, 2015. Accessed: 2020-01-22.

[26] Yoshua Bengio Ian Goodfellow and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[27] K. Atkinson. *An Introduction to Numerical Analysis*. John Wiley and Sons, 1978.

[28] H. Chai and K. Lee. En-route arrival time prediction via locally weighted linear regression and interpolation. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019.

[29] François H.L.R. Clemens Mathieu Lepot, Jean-Baptiste Aubin. Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. *MDPI Water*, October 2017.

[30] E. Suli and D. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

[31] Robert Nurnberg. Introduction to numerical analysis m2aa3 imperial college mathematics department. `https://union.ic.ac.uk/rcsu/mathsoc/files/M2AA3-16.pdf`, 2016. Accessed: 2020-06-12.

[32] Jean leah Njoroge. Significance of exploratory data analysis (eda). `http://www.jeannjoroge.com/significance-of-exploratory-data-anaysis/`, November 2017. Accessed: 2020-06-12.

[33] Nist-sematech engineering statistics handbook. `https://www.itl.nist.gov/div898/handbook/eda/section3/eda35h.htm`, 2003. Accessed: 2020-06-12.

[34] When will lockdown en d? the uk's lockdown rules, explained. `https://www.wired.co.uk/article/uk-lockdown`, 2020. Accessed: 2020-06-12.

[35] Transport for London Press Release. Tfl introduces middle-door only boarding across the london bus network, April 2020. Accessed: 2020-06-12.

[36] Transport for London Press Release. Tube, rail and bus services stepped up for people who have to use public transport, May 2020. Accessed: 2020-06-12.

[37] B. M. Williams and L. A. Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, Volume 129:664–672, 2003.

[38] Coronavirus: Lockdown easing in England 'modest' - Jenrick. `https://www.bbc.com/news/uk-52869875`. Accessed: 2020-06-10.