Imperial College London

BEng Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

Dynamic Bus Delay Predictions

Author:
Serene Chongtrakul

Supervisor:
Peter McBrien

2nd Marker:
Thomas Lancaster

February 6, 2020

Submitted in partial fulfillment of the requirements for the BEng of Imperial College London

Contents

1 Introduction

For the year 2017/18, there were 2.23 billion passenger journeys made by local bus in London. This is the 4th consecutive year in which passenger journeys on buses in London have fallen [1]. For the year ending in June 2019, the number of local bus passenger journeys in London decreased by 0.9% [2]. 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. The latter issue is what this project will focus on. There is a large number of applications already available that provide real time information on London bus services such as City Mapper or Google Maps. However, these applications do not currently gather information on when and why delays are occurring and actively warn users of this. In other words, these applications are static in their predictions. This project aims to study various parameters that could affect the journey time of a bus, ranging from time of day to weather conditions to congestion, and implement a number of models based on these factors to predict bus arrival times. The best performing model will then be used to create a simple mobile or web application that allows users to see the dynamically predicted arrival time from any bus stop for the bus that they intend to take.

The models that will be developed to predict the arrival times of the buses will make use of TfL's publicly provided APIs. Currently TfL provides a variety of APIs including ones that describe bus routes or bus stop locations, as well as ones that provide the live times for when a bus arrives at a specific stop. In fact, TfL also provides an arrival prediction API *Countdown*, for both bus routes and tube lines. This perhaps could be seen as negating the point of the project as described previously. However, it can be inferred that this API is static in its implementation, meaning it does not use new/current information to aid in its prediction. For a particular bus, say 452, and a particular stop, say 'York House Place', Countdown makes use of the timetabled arrival time and also looks at the GPS position of 452 relative to 'York House Place', in order to make a prediction for its arrival time there. I will be exploring both static and dynamic models, but Countdown will be used as the baseline model for which to compare my models against.

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. 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. The algorithms tend to be fairly simple and are not dynamic in nature. On its own, performance could be fairly weak, due to the unreliability of traffic conditions and other factors that vary over time. (See Section 2.2)

- **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) [3]. 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" [4]. (See Section 2.3)

- **Kalman Filtering models**: The Kalman filter is one of the optimal solutions for

tracking and data prediction tasks [5]. These models are a dynamic estimation technique that have been used fairly frequently in bus delay predictions. (See Section 2.4)

- **Machine Learning models**: These models have been widely used in research to try to improve public transportation. This project will look particularly at Artificial Neural Networks (ANNs), which are suited towards more complex and noisy data and finding non linear relationships between dependent and independent variables [6]. (See Section 2.5)

I will only explore historical average models, regression models and machine learning models. The following parameters will also be investigated to see how relevant they are in predicting bus arrival times: weather, time of day and time of week, time of year (for example special holidays and school holidays), dwell time at bus stops and traffic conditions (See Section 2.6).

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 [7]. Traffic congestion is increasing and air quality is worsening. Although Sadiq Khan, the current London Mayor since 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, the car is still the most popular single mode of travel for Londoners, with 29.8 % of workers opting to commute in this manner [8]. 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 [9]. Therefore, encouraging bus use through improving the experience of using a bus, will help to improve the air quality and the environment overall.

If this study is successful, the end product should be an application that predicts more accurately than the Countdown API, the arrival time of any bus on any route in London. Bus passengers want reliable and timely services, but if that is not possible, they want to know when a bus will actually turn up so that they are not left waiting at a bus stop for a bus that the board says is due, but has been saying it is due for the past 3 minutes. By providing a service that supplies users with reliable bus arrival times, I hope 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.

2 Background

2.1 Clustering Analysis

Clustering is one of the most common techniques used to explore a set of data. It allows us to get an intuition about the structure of the data by trying to group 'similar' data together into subgroups or clusters. There is a variety of different similarity measures such as euclidean-based distances or correlation-based distance [10]. Clustering is an unsupervised learning method because we don't have any ground truth labels for which to compare the output of the clustering algorithm to.

2.1.1 K-means

K-Means is one of the most popular clustering algorithms. The k-means algorithms divides a set of N samples X into K disjoint clusters C, each described by the mean μ_j of the samples in the cluster [11]. The k groups are defined by the means, commonly known as a centroid, and a point is considered to be in a particular cluster if it is closer to that cluster's centroid than any other centroid [12]. K-Means aims to choose the centroids that minimise the inertia or within-cluster sum-of-squares criterion (a measure of how internally coherent the clusters are):

$$\sum_{i=0}^{n} min_{\mu_j \in C}(||x_i - \mu_j||)^2) \tag{1}$$

The algorithm has three steps:

1. Choose the initial k centroids (generally randomly chosen).
2. Assign each sample to its nearest centroid.
3. Create new centroids by taking the mean value of all of the samples assigned to each previous centroid. Repeat step 2 and 3 until the centroids do not move significantly.

An example of the algorithm in action is shown in Figure 1. Training samples are shown as dots. Cluster centroids are shown as crosses. a) shows the original dataset. b) shows the random initial cluster centroids. c) - f) shows two iterations of k-means. In each iteration, the training samples are assigned to the closest cluster centroid. The cluster centroid is then recalculated to the mean of the points assigned to it.

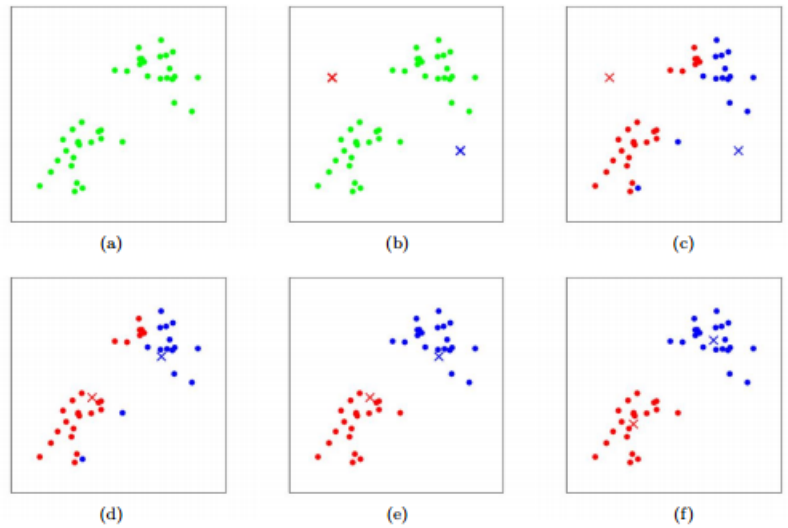

Figure 1: K-Means in action, original image from [12]

After a sufficient number of iterations, the results of the k-means algorithm will eventually converge. However, the algorithm will attempt to cluster data regardless of if it can be clustered or not. Therefore, K-means tends to perform more poorly on data that doesn't have a spherical shape. See Figure 2 for examples of k-means not performing well on complicated geometric shapes.

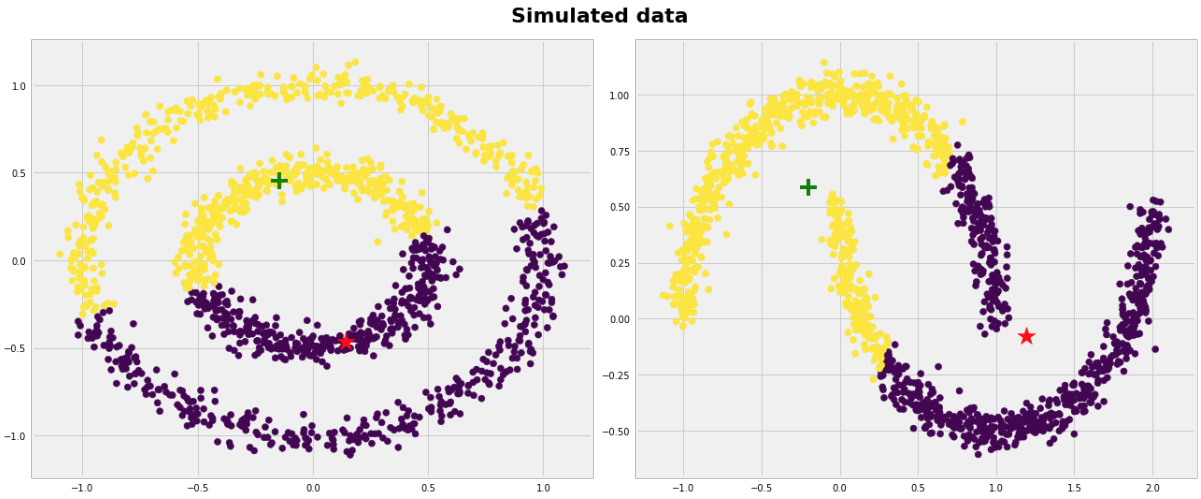

Figure 2: K-Means performed on complicated geometric shapes, original image from [10]

The k-means clustering algorithm uses a randomly generated seed to determine the starting centroids of the clusters. This inherent random nature of k-means can lead to the minimisation function getting stuck in a local minima. The 'k-means++' initialisation scheme helps to mitigate this by initialising the centroids to be generally distant from each other, leading to provably better results than random initialisation.

2.1.2 Cluster Quality Measures

The two main methods of measuring the quality of a clustering are the *Calinski-Harabasz* (CH) score and the *Silhouette Coefficient* [13].

The CH score [14], also known as the Variance Ratio Criterion, is defined as the ratio between the within-cluster variance and the between-cluster variance. For a set of data of size N, which has been clustered into k clusters:

$$CH = \frac{SS_B}{SS_W} \times \frac{N-k}{k-1} \tag{2}$$

Where SS_B is the overall between cluster variance, SS_W is the overall within cluster variance, k is the number of clusters and N is the number of observations.

$$SS_B = \sum_{i=1}^{k} n_i ||m_i - m||^2 \tag{3}$$

Where k is the number of clusters, n_i is the number of observations in cluster i, m_i is the centroid of cluster i, m is the overall mean of the sample data, and $||m_i - m||$ is the L^2 norm (Euclidean distance) between the two vectors.

$$SS_W = \sum_{i=1}^{k} \sum_{x \in c_i} ||x - m_i||^2 \tag{4}$$

where k is the number of clusters, x is a data point, c_i is the i^{th} cluster, m_i is the centroid of cluster i, and $||x - m_i||$ is the L^2 norm (Euclidean distance) between the two vectors. Well defined clusters have a large between cluster variance and a small within cluster variance. The higher the CH score, the better defined clusters the model has.

The Silhouette Coefficient [15] measures how similar that point is to other points in its cluster. It is bound between -1 and 1, where a higher score indicates that point i is well matched to its cluster. If many points have a low or negative silhouette value, then the clustering solution might have too many or too few clusters. The silhouette coefficient for a single sample i is composed of two scores whose mean then gives the overall silhouette coefficient score for that sample.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{5}$$

where $a(i)$ is the average distance between data point i and all other points in the same cluster and $b(i)$ is the average distance between data point i and all other points in the next nearest cluster. Silhouette values can be used as a clustering evaluation criterion with any distance metric.

2.1.3 Clustering Analysis in Bus Arrival Time Predictions

Clustering methods have been used to investigate typical bus delay information. F. Sun, Y. Pan, J. White and A. Dube clustered the travel times of buses by different times of day and discovered that there were different travel patterns over different time periods.

Sun et al. applied the K-means algorithm for each of day of the week to cluster the bus delay data according to delay and time in the day, varying k from 2 to 5 [16]. The silhouette score was then calculated on each of the different clusters obtained and the k value that had the best silhouette score was chosen to be the optimal number of clusters. Figure 3 shows the results of the clustering for the historical delay data of a bus at a particular bus stop on an outbound journey over a 24 hour period on every Tuesday between 01/08/2015 and 31/08/2015. It can be seen that most of the points are roughly clustered into the green and blue groups, suggesting two different delay patterns: one before 1230 and one after.

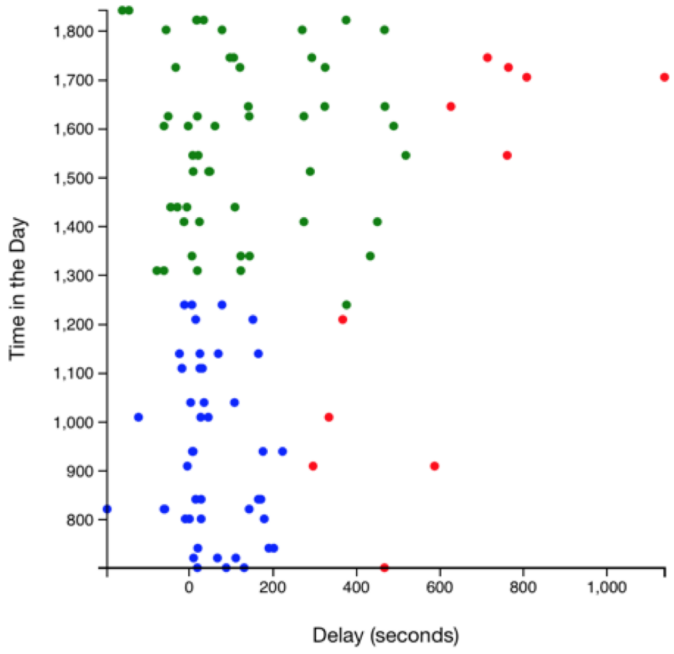

Figure 3: Clustered historical delay data, original image from [16]

F. Sun et al.'s results support the hypothesis that time of day does affect the delay, and thus arrival time of a bus. Therefore, this parameter is worth investigating further in this project. See Section 2.6.2 for further discussion on time of day and week as a parameter affecting bus arrival times.

We can see that this method of clustering data acts as a good way of deciding which of the parameters that could affect bus arrival times are worth pursuing further.

2.2 Historical Average 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 [17]. 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 [18]. 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 Simple 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:

- **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.

Both the *mean* and *naive* methods result in flat line forecasts, and therefore, would not be particularly suitable for bus arrival prediction because research has shown that there are different trends in delay times for different time periods [6], [19], [16]. 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 (or clustering) 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. Because there is reason to believe that historical data on bus delay times show an up and down trend, out of the

four simple methods above, the seasonal naive method seems to be the one that is most suited.

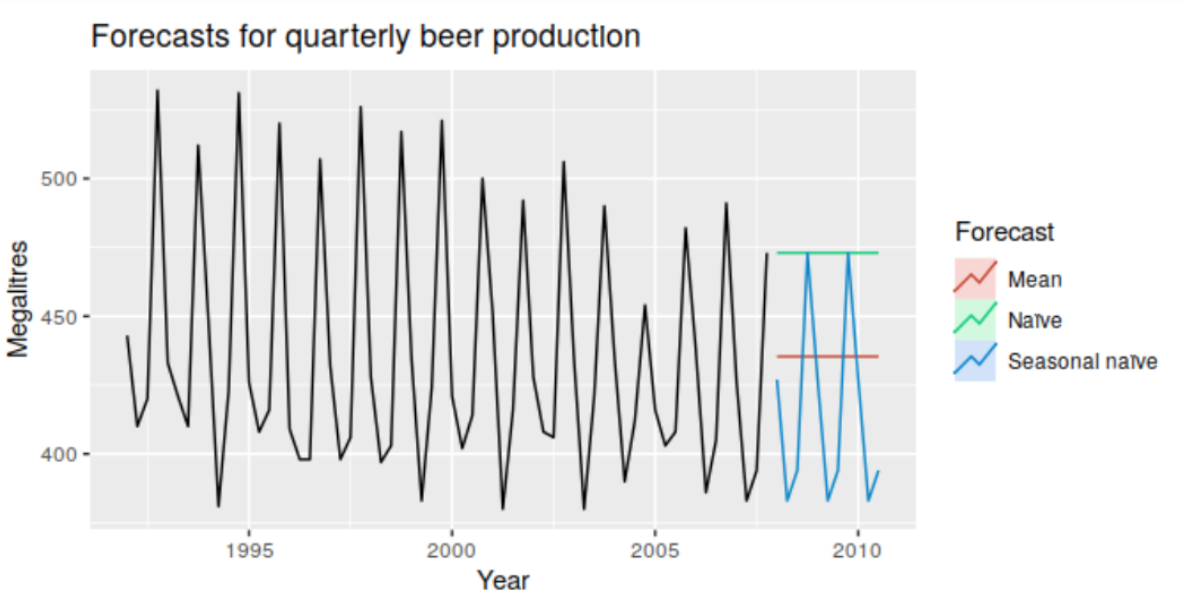

Figure 4: Example of the first 3 methods listed above applied to quarterly beer production data, original image from [18]

2.3 Regression Models

Regression is a form of statistical modelling that investigates the relationship between a dependent (target) and independent (predictor) variable [3]. 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 [20]. 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.

2.3.1 Simple Linear Regression and Multiple Regression

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

$$Y_t = \beta_1 + x_i\beta_2 + \epsilon_t, t = 1, ..., n \tag{6}$$

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 [18].

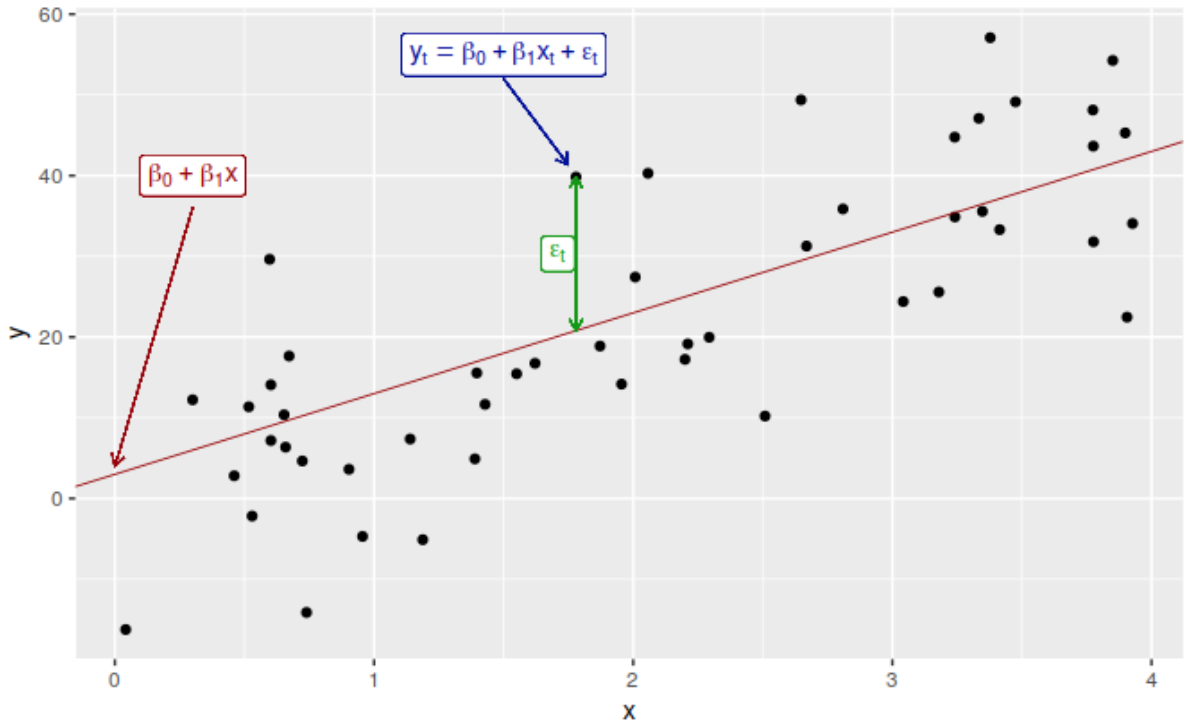

Figure 5: Linear Regression, original image from [18]

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

$$Y = \mathbf{X}\beta + \epsilon \tag{7}$$

where Y is an n-dimensional random vector (observable), $\mathbf{X} \in \mathbf{R}^{n \times p}$ is a known matrix (often called the 'design matrix'), $\beta \in \mathbf{R}^p$ 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 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.

2.3.2 Least Squares Estimation

Least squares estimation is a method of estimating parameters by minimising the squared discrepancies between observed data and their expected values [21]. Choose β to minimise:

$$S(\beta) = \sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 = (\mathbf{Y} - X\beta)^T(\mathbf{Y} - X\beta) \tag{8}$$

where as before, Y is an n-dimensional random vector, $\mathbf{X} \in \mathbf{R}^{n\times p}$ is a known matrix and $\beta \in \mathbf{R}^p$ is an unknown parameter. We denote the the estimated value that gives the best fit to the data of β by $\hat{\beta}$.

2.3.3 Measuring Success

The standard error is the 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 linear regression model fits the data is via the coefficient of determination:

$$R^2 = \frac{\sum(\hat{y}_t - \bar{y})^2}{\sum(y_t - \bar{y})^2} \tag{9}$$

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 [18].

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.

2.3.4 Outliers

An outlier is an observation that does not conform to the general pattern of the rest of the data. Observations that have a large influence on the estimated coefficients of a regression model are called influential observations. 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 [18], [20]. Figure 6 is an example showing that an outlier can have a great effect on predictions in regression models.

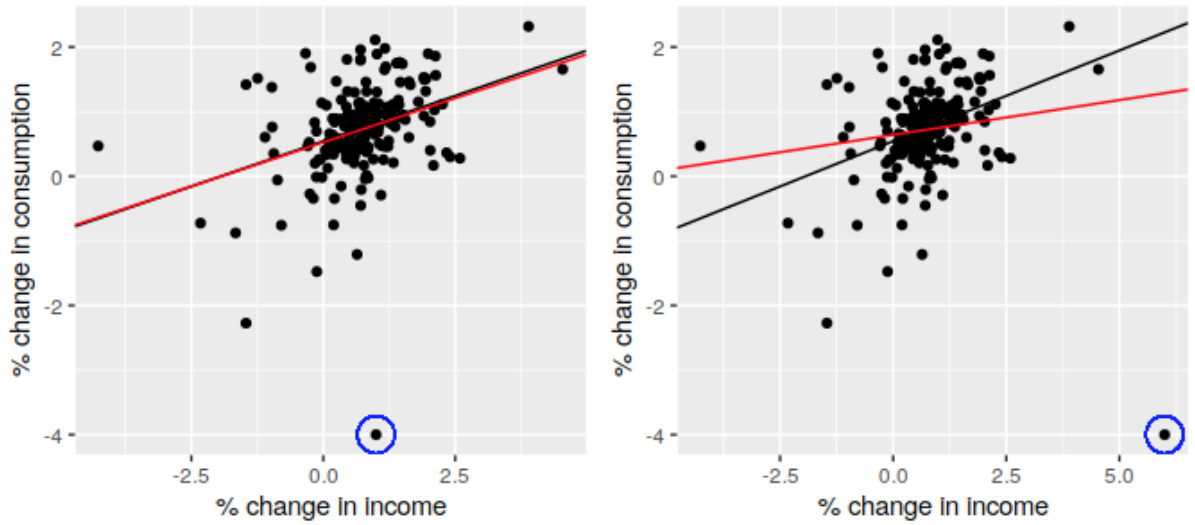

Figure 6: Example showing how an outlier can affect predictions. The red line is the regression line fitted to the data including the outliers, the black line is the regression line fitted to the data without the outlier, original image from [18]

2.3.5 Regression and Bus Arrival Predictions

Univariate and multivariate regression models have been used to predict bus travel times. These models are able to work satisfactorily even if traffic conditions are not stable and therefore, would be a good choice to try out as one of the models to explore. However, regression models are generally outperformed by other types of models [6].

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 [4]. 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. To decide whether a model was reasonable, Patnaik et al. looked at the values of the R^2 and the correlation between the variables and analysed the residuals. The results of these investigations indicated that weather was not a significant factor for estimating bus arrival times (see Section 2.6.1 for a further discussion on this). Other findings from this investigation were that trips taking place on different days of the week (excluding weekends) did not contribute any measurable difference to the travel, whereas time of day appeared to affect travel time significantly. The effect of time of day and time of week is discussed further in Section 2.6.2.

2.4 Kalman Filtering Models

Kalman filtering is an algorithm that provides estimates of some unknown variables, given the measurements observed over time [22]. The algorithm is unique in that it consists of two processes, the prediction process and the measurement process, that work recursively [23].

The Kalman filtering algorithm starts in the prediction process and estimates the prediction state based on the derived state space equation (Equation 10). This stage covers the prediction of the a priori state and a priori error covariance. The process model defines the evolution of the state from time $k-1$ to time k as:

$$x_k = Fx_{k-1} + Bu_{k-1} + w_{k-1} \tag{10}$$

where F is the state transition matrix applied to the previous state vector x_{k-1}, B is the control input matrix applied to the optional control vector u_{k-1}, and w_{k-1} is the process noise vector that is assumed to be zero-mean Gaussian with covariance Q, i.e. $w_{k-1} \sim N(0, Q)$. The next stage is the measurement process (Equation 11), which covers the calculation of the optimal Kalman gain, updating the a posteriori estimation state and the a posteriori error covariance. The measurement model describes the relationship between the state and the measurement at the current time step k as:

$$z_k = Hx_k + v_k \tag{11}$$

where z_k is the measurement vector, H is the measurement matrix and v_k is the measurement noise vector that is assumed to be zero-mean Gaussian with covariance R, i.e. $v_k \sim N(0, R)$

The Kalman Filter provides estimates of x_k given the initial estimate x_0, the series of measurements z_1, ..., z_k and the information described by F, B, H, Q and R.

2.4.1 Kalman Filters and Bus Arrival Predictions

Kalman filtering has been a popular choice of model for those researching bus delay predictions because of its capacity of filtering noise and its dynamic nature.

Fan and Gurmu studied Kalman Filters as one of three models for predicting bus arrival times [6]. They used the mean absolute percentage error (MAPE) to evaluate the performances of their models (See Section 5 for more details) and discovered that the Kalman Filtering model gave a better prediction in most 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.

Sun et al. also used a Kalman Filter model to analyse their data for real-time bus arrival time predictions [16]. To evaluate their results, the root mean square deviation (RMSD) of delay predictions was used:

$$RMSD = \sqrt{\frac{\sum(t_{arr}^{act} - t_{arr}^{pred})^2}{n}} \tag{12}$$

where t_{arr}^{act} is the actual arrival time, t_{arr}^{pred} is the predicted arrival time and n is the number of bus trips in the data set. 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 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.

I am not planning on exploring Kalman models any further as a model for predicting bus arrival times. This is because I think that if I include Kalman models, I will have insufficient time to properly refine all the models. I would rather have a smaller number of highly optimised models, rather than a larger number of very simple or unoptimised models.

2.5 Machine Learning Models

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

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 $x_{i\in 0,\ldots,m}$ with $m+1$ weights $w_{i\in 0,\ldots,m}$ 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.

$$output = \begin{cases} 0 if \sum_j w_j x_j \leqslant threshold \\ 1 if \sum_j w_j x_j > threshold \end{cases} \tag{13}$$

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 Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$ and ReLU: $max(0, x)$.

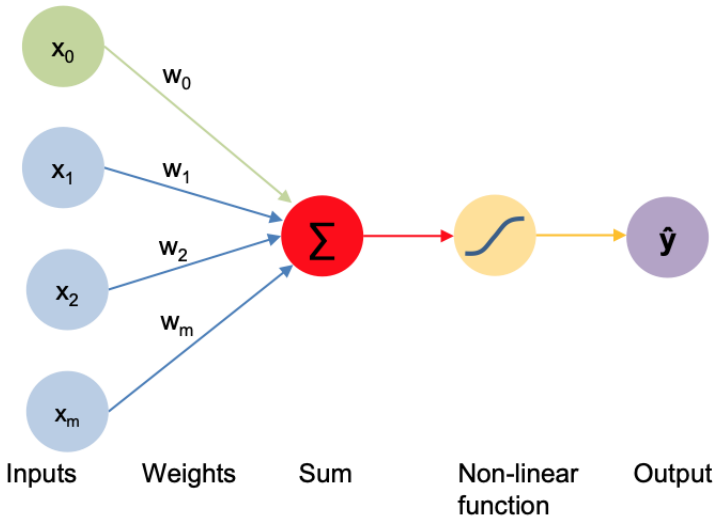

Figure 7: The Perceptron: forward propagation, original image from [24]

2.5.2 Neural Network 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 8 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.

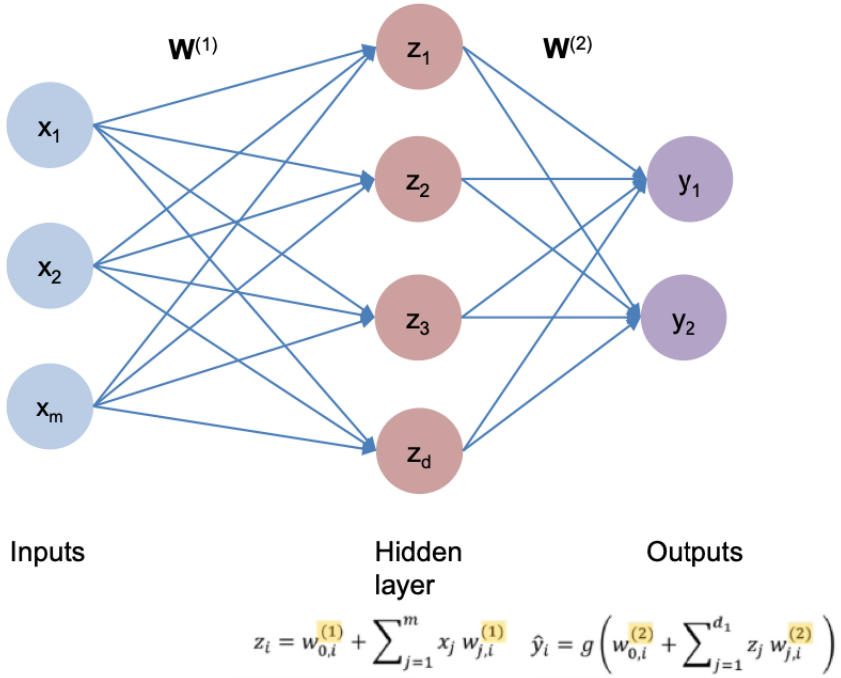

Figure 8: Fully connected single layer neural network, original image from [24]

2.5.3 Optimisation

Loss functions, also known as objective functions cost functions empirical risk, are used to train the neural network: $L(f(x^{(i)}; W), y^{(i)})$ where $f(x^{(i)}; W)$ is the predicted value, $y^{(i)}$ is the actual value and $W = W^{(0)}, W^{(1)}, \ldots$ [24]. There are a number of different loss functions, including:

- Empirical Loss: the mean loss across all samples.
- Binary Cross Entropy Loss: comparing models that output a probability between 0 and 1.
- Mean Square Error Loss: instead of 0 or 1, we might have a regression model for continuous output values.

The goal is to find the weights that achieve the lowest loss. We do this using gradient descent for example stochastic gradient descent, which is easy to compute, but noisy, or batch gradient descent, which is fast to compute and provides a much better estimate than stochastic gradient descent.

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.

2.5.5 Neural Networks and Bus Arrival Predictions

In more recent studies, neural networks have been used more often as a dynamic model for studying bus arrival time predictions.

Fan and Gurmu used artificial neural networks (ANN) as one of their models, along side a historical average model and a Kalman filtering model, when predicting bus arrival times [6] For their ANN, a fully connected multi-layer perceptron with a single hidden layer was used, along with $tanh$ as the nonlinear activation function. It was discovered that the ANN outperformed the other two models in both overall prediction accuracy and robustness. 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.

Another study that used ANNs was done by Jeong and Rilett. This study also found that the ANN model outperformed the historical average model and regression model that was investigated in the same study [19]. This ANN was also a fully connected multi-layer perceptron with a single hidden layer, and also used $tanh$ as the nonlinear activation function. 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.

2.6 Factors Affecting Bus Arrival Times

There are a variety of different factors that can affect the arrival time of a bus, be that by affecting the bus' travel time, for example traffic conditions, or affecting the bus' dwell time (the time that it spends at a bus stop before it can continue on its route), for example 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 also attempt to argue 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 traffic conditions (if it is particularly cold and icy, there may be fewer people willing to drive to work, so there are fewer cars on the road). Therefore, care will have to be taken when implementing the models, to ensure that not too much weighting is placed on any singular factor.

2.6.1 Weather

Koetse and Reitveld found that there is a substantial reduction in traffic speed when weather conditions are extreme [27]. 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.

As mentioned in Section 2.3, it was found by Patnaik et al. that there was little to no effect between weather and bus delay time [4]. 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. Therefore, it is likely that I will not be exploring any further the effect of weather on bus arrival times. All the data that will be used in this project has to be collected manually since TfL does not provide any historical data on bus arrival times, and so most of the collected data would be from the periods spanning mid January to March/April time. This would not be a large enough sample of different and extreme weather conditions and 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.

2.6.2 Time of Day and Time of Week

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.

W. Fan and Z. Gurmu clustered their travel time data by time period and explored the effects of this. The results can be seen in Figure 9, which plots the travel time index (i.e. the ratio of the average travel time per weekday and the average travel time across all days) against the time of day for the LT11 line in the Northbound direction in Macaé, Rio de Janeiro, Brazil [6]. 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. Another observation that was made was that the travel time distributions over the different days of the week seemed to be nearly the same. Therefore, 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.

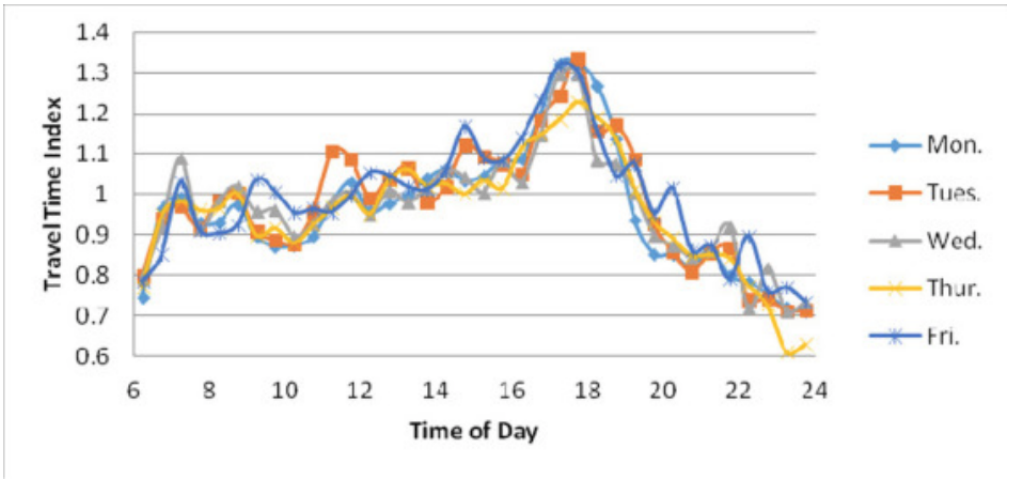

Figure 9: How time and day of week (not including weekends) affects travel time, original image from [6]

Google studied the delay patterns of buses for a new feature introduced into Google Maps and some of the results can be seen in Figure 10, which shows the predicted travel time for a bus ride with traffic held constant [28]. This suggests that travel patterns during the week versus the weekend are different and further supports the idea that the same models can be used for all weekdays whilst a different model should be used for weekends.

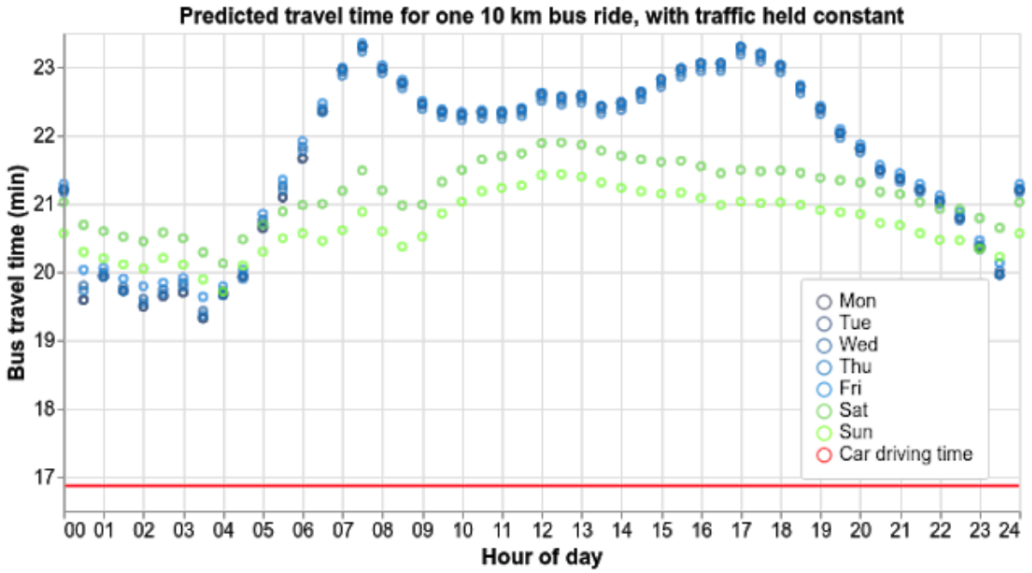

Figure 10: How time and day of week affects travel time, original image from [28]

Based on the above, the effect of time of day and time of week on bus arrival times will be investigated further in my models.

2.6.3 Time of Year

Similarly to the previous parameter, 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 likely to have have the same travel time as during school holidays. According to National Travel Survey statistics, 19% of children aged 5 to 16 travelled to school by bus and 21% travelled by bus or van in the year 2017/18 [29]. Therefore, during term time, there are not only more vehicles on the road, thus increasing the likelihood of congestion, but buses will also have longer dwell times because they will have to wait for children to board and deboard. Therefore, during school holidays when a large number of these children will no longer be making daily regular trips on the bus and there are fewer vehicles on the road, the travel time of a bus may be faster and thus it may be less likely to be delayed. However, in this particular example of school holidays playing a part in predicting bus arrival times, it could be argued that because children travel to school around the same time as rush hour, the effect of this could be explored in the previous parameter: time of day and time of week.

Time of year is unlikely to be explored any further as a factor affecting bus arrival times. As mentioned previously, the data used in this project will have to be collected manually and due to time constraints and lack of historical data, it is impossible to get a whole year's worth of data to use for modelling.

2.6.4 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.

In a study that used regression models to look at parameters affecting bus arrival times, it was found that bus stop dwell times were less important and statistically not as significant [6]. 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. Therefore, dwell times of buses will not be taken into account in the models that I will develop.

2.6.5 Traffic Conditions

Jeong and Rilett argued that in order to accurately predict bus travel time, it it essential to consider traffic conditions [19]. Google Maps recently 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 [28]. 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 [30]. 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 and therefore, traffic conditions do not need to be taken into account as a factor, especially for historical average models. This is supported by Fan and Gurmu, who argue that if traffic patterns are relatively stable over time, then historical average models will perform well [6].

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. Therefore, taking all of the above into account, traffic conditions will not only be considered further as a factor affecting bus arrival times if I have time.

3 Implementation

3.1 Data Collection

There is no historical data for bus arrival times available online and so all the data that I will use to train and test my models has to be manually collected. Furthermore, TfL does not provide an API that gives exact bus arrival times and so this information has to be inferred from other data that is available.

I am currently in the process of writing code to collect information about bus arrival times using TfL's Countdown APIs. I have chosen to write this code in Python because I have some experience performing statistical clustering, creating neural networks and other data science methods using Python. Furthermore, there are a lot of libraries in Python, for example sklearn, that support the kind of analytics and modelling that I will need to do. Hopefully, this will minimise the amount of work that I would have to do to get started in the first place and also allow me not to have to create functions from scratch when performing tasks such as k-means clustering.

Using the APIs that are available, I can get the predicted arrival times of any bus at any bus stop, but not the actual arrival time. In order to collect information on bus arrival times, I call the Countdown API every 30 seconds and store the information about the bus ID and its estimated arrival time in a dictionary. If the bus ID already exists in the dictionary, I update the estimated arrival time and the timestamp (the time when Countdown last updates this estimate). A particular vehicle will continue to show up in the Countdown call until it arrives or there is an error in the system and it 'vanishes'. At this point, I have to decide whether to mark that vehicle down as having arrived at that time or not. Once a vehicle has been marked down as arrived, it gets added to a separate dictionary that only contains information on buses that have arrived and the time of arrival. This is necessary because the same buses are reused during the day on the same route and so, if the arrival information for a bus is not moved to another dictionary, its arrival time will be overwritten when the bus next comes back into circulation at this particular stop.

```
vehicle_id,direction,timestamp,expected_arrival,arrived
LTZ1072,outbound,2020-02-06T14:08:56.6911542Z,2020-02-06T14:19:52Z,False
LTZ1148,outbound,2020-02-06T14:04:35.533452Z,2020-02-06T14:07:10Z,False
LTZ1665,outbound,2020-02-06T14:08:56.6911542Z,2020-02-06T14:23:42Z,False
LTZ1668,outbound,2020-02-06T13:43:08.8404252Z,2020-02-06T14:03:49Z,True
LTZ1677,outbound,2020-02-06T13:43:08.8404252Z,2020-02-06T13:55:36Z,True
LTZ1084,outbound,2020-02-06T13:43:08.8404252Z,2020-02-06T13:49:49Z,True
LTZ1678,outbound,2020-02-06T13:43:08.8404252Z,2020-02-06T13:46:06Z,True
LTZ1078,outbound,2020-02-06T14:08:56.6911542Z,2020-02-06T14:33:23Z,False
```

Figure 11: Data collected from Bus Route 9 at the stop 'High Street Kensington'. Bus 'LTZ1678' has been marked down as having arrived at 13:46:06. When the same bus is soon to arrive at High Street Kensington again, its expected arrival time will be updated and the arrived field will become False once more.

Let us consider the following scenario: we have been tracking the 452 bus with ID 'LTZ1076', with predicted arrival time at 'York House Place' at 17:30:12 and the current time is 17:26:00. Every 30 seconds, Countdown is called and the predicted arrival time of 'LTZ1076' will be updated along with the timestamp, which is the time when Countdown made the prediction. For the sake of this example, let us assume that the predicted arrival time does not change. If at 17:30:30, the API call is made and 'LTZ1076' is no longer returned on the list of arrival predictions, we cannot yet presume that it did indeed arrive at the predicted time 17:30:12. This is because there could have been a glitch in the system and 'LTZ1076' is still travelling to its destination. The API continues to be called in case 'LTZ1076' shows back up on data returned from the API calls. If the estimated arrival time has surpassed 3 minutes past the current time of calling the API and 'LTZ1076' does appear again, then it will also appear with a new predicted arrival time that is later than 17:30:12, because the time of calling the API will also be later than 17:30:12. In this case, I don't have to consider whether 'LTZ1076' has arrived or not until the new time predicted arrival time comes around. However, if after the 3 minutes has passed and no new information on 'LTZ1076' appears from the API call, then I can conclude that 'LTZ1076' did indeed arrive at the first predicted arrival time 17:30:12. If the bus has indeed arrived, this information is copied into another dictionary that only contains information on buses that have arrived. In an hour or so when 'LTZ1076' makes its way back to 'York House Place', it will be able to modify the `expected_arrival`, `timestamp` and `arrived` fields without losing historical data.

Currently this information is stored in different CSV files, a different file for a different bus route and stop. A specific bus route id and bus stop id is passed in and the code attempts to locate a file that contains information. If it does not exist, then it creates an empty file, but if it does exist, it loads the information from that file back into a dictionary to be processed. After the desired number of API calls are made, the information is compiled back into the same CSV file, thus updating the information in the file as necessary. I am using CSV files at the moment because this data is going to be used for training and testing and therefore, there is no real need for quick information retrieval. It is possible that I may need to use a database in the future, but at the moment, there does not seem to be any impact on the performance of the code that requires me to use a database.

4 Project Plan

This project will involve investigating various models for predicting arrival times and then implementing the best performing model in a simple web or mobile app. I have weekly meetings with my supervisor on a Tuesday and so I intend to structure my plan around these meetings - aiming to complete milestones by the next Tuesday or every other Tuesday, depending on the size of the task.

The plan is to have the bulk of the interim report completed by 28/01/2020. I aim to send this version of the interim report to a couple of people who do not have computer science backgrounds and using their feedback, spend the week leading up to the interim report deadline fine tuning the report.

The sections that do not have to be completed by 28/01/2020 are ones that will talk about any coding implementation that I have done. Before the interim report deadline on 06/02/2020, I want to have written some code to collect data from the TfL APIs. In particular, I want to get around a week's worth of data for the bus 452 and the stop 'York House Place'. This data should be gathered every 30 seconds for the time periods 0600 - 0900, 1100-1400, 1600-1900 and 2100-2400 (Unless 452 is not a night bus, and then the time periods may shift). Once this code and data collation has been completed or at least have had significant progress, then I can include this in the interim report. If this has not been completed before the deadline for the interim report, I should still include what I have done so far of the data collation.

After the interim report has been handed in, then I can spend the majority of my time focused on data collection. I need to have a large enough data set to perform my models on and therefore, I should aim to collect at least 2-3 weeks worth of data for the bus 452 and at least 4-5 of the bus stops that 452 serves. Throughout the rest of this term, data should continue to be collected in the background in order for me to have a variety of test sets and a large enough training set. I will aim to include 2 more buses and for each of the other buses, at least 3 or 4 of their bus stops (although the more data the better). It will be hard to try to explore the effects of time of year on data because I do not have a year's worth of data to collect, but I might be able to collect data during the February half term and count that towards effect of school holidays on bus arrival times.

Once enough data has been collected, then I can begin basic data exploration. This will include performing clustering analysis on the data to see if there are any clear travel patterns. The clustering will be based on a variety of factors that could affect bus travel time as listed in Section 2.6. After clustering analysis has been performed on the training data, this should give me a clear indication of which factors to stop considering and which to continue to take into account when creating the models. An evaluation of the clustering methods that I have used should be done at this point.

The first model I should try to implement once all the data exploration has been completed is a simple historical approach. For example, I may take the last 2 journey times and use its average as the predicted journey time for the current bus or a seasonal naive model. This can be used as a base model for which to compare my other more complex models against.

Once I have achieved satisfactory results from the historical data approach, the next models I should try to implement are more complex historical models, followed by the

regression models, and finally the artificial neural network models. These models should all be evaluated as I go along, but a final more thorough evaluation comparing all the models against each other, will be performed at the end.

By 11/02/2020, I want to have arranged a date and time with the second marker for a project review meeting. After this meeting has taken place, I can begin to make changes or new inclusions to the project, taking into account the 2nd marker's feedback.

Over the Easter holidays I have revision to do for my 1 computing and 1 maths exam. However, I plan to continue working on the different models that I want to test out. By the start of the summer term, I want to have finished with all the modelling and start evaluating the models against each other.

By the end of the first 2 weeks of term I want to have decided on a 'best' model and started developing the web app that will display the relevant data. The rest of the time should be dedicated towards the web app - no more work on modelling should be done after this point. If I end up having more time than planned, then the web app can be more complex or I can try to refine the models to make them faster.

5 Evaluation Plan

Because bus arrival prediction already exists, the evaluation of this project will be less focused on the novelty aspect or how much new ground is being covered. Instead my evaluation will look more at how the results of the models that I have implemented compare to what already exists in practice i.e. looking at how viable my best models are compared to the ones used in popular applications.

The end product will be a web app that allows a user to input a bus number and bus station and returns a time for when the bus is predicted to arrive. One way that I will measure the success of this is by comparing the predicted arrival times for bus routes that I used in training versus the predicted arrival times for bus routes that I did not use (since there are so many bus routes in London, it would be impossible to test my models on all of them). If the predicted arrival times for the bus routes that I used in training are significantly better than the predicted arrival times for the bus routes that I did not use in training, then this will indicate that I chose a model that was not general enough and I may have overfitted my model. Therefore, this would be a measure of some failure on my part to thoroughly test each of the models that I develop enough.

I intend to explore a variety of different models and therefore, by comparing the performance of the models against each other, this will provide another benchmark for the degree of success that my project has achieved. The results of the comparison will be visualised through a variety of graph plots. However, there is more than one way of evaluating how successful a prediction model is. Is the 'best' model the one that is consistently 3 minutes off the actual bus arrival time? Or is the 'best' model the one that accurately predicts the bus arrival time 80% of the time, but then the other 20% of the time, gives a predicted arrival time that is 6 minutes incorrect? This perhaps is more subjective, and so I may make a poll or ask some people to fill in forms to indicate which of the above they think makes a better model. I will then use the most popular answer as the main method of evaluating success.

For example, Fan and Gurmu [6] measured the success of their model by using the mean absolute percentage error (MAPE). This represented the average percentage difference between the observed value and the predicted value:

$$MAPE = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{Y_t^P - Y_t^{Obs}}{Y_t^{Obs}} \right| \tag{14}$$

where Y_t^P is the predicted bus travel time from the most recent bus stop to the target bus stop, Y_t^{Obs} is the observed bus travel time and n is the number of test sets. Figure 12 shows a comparison of the MAPE score for 3 different models that they tested out: an ANN model, a Kalman Filter model and a historical average model.

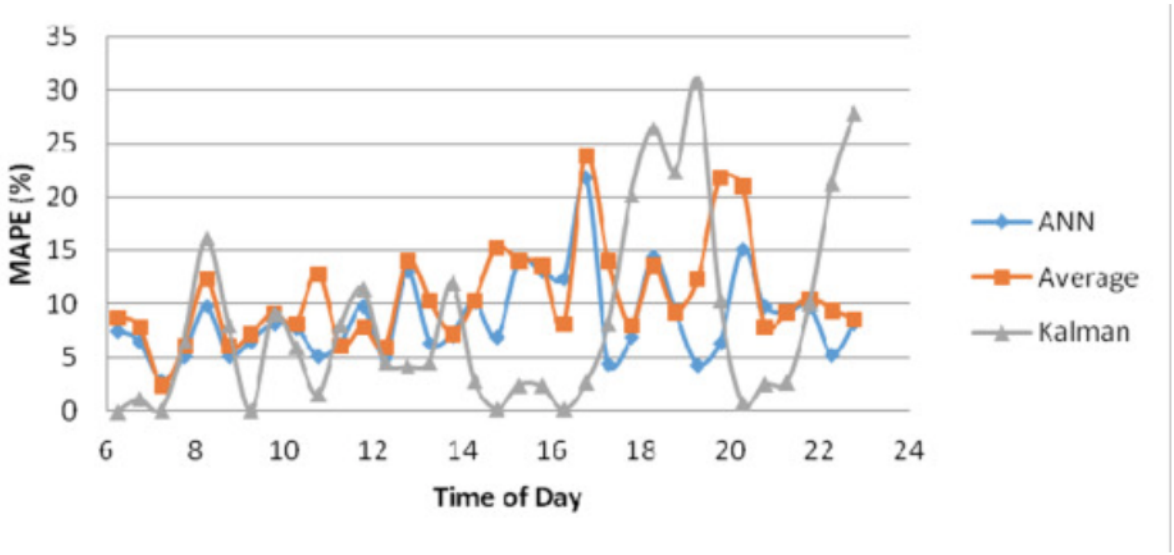

Figure 12: Comparison of different prediction models by MAPE, original image from [6]

As mentioned previously, TfL already has an API that predicts bus arrival times. Therefore, another measure of success will be to compare the predictions made by my 'best' model against the predictions made by the Countdown API. If my results are better, then this indicates that what I have created could act as a better middleman for current travel planning applications e.g. City Mapper or Google Maps. If I were to turn my model into a callable API, then these applications could make requests to my API instead of using the currently available Countdown API. Similarly to above, when comparing the performances of my model against the Countdown API, what makes one of the models 'better' will be based upon the results of the survey that I will put out towards the end of this term. However, it is completely plausible that my model may not perform as well as Countdown. If this is the case, more time should be spent to evaluate why a dynamic approach does not outperform a static one and full justification should attempt to be made.

As the application part of the project is not the priority, less focus will be spent on evaluating the usability of the web app. Of course it is important that the user can easily input their desired bus stop and bus number and see the results clearly, but the main part of the project is the actual investigative and modelling parts. The only 'usability' evaluation that will be taken into account will be the speed at which a user gets the results of a predicted arrival time. A user should not have to wait around for minutes for the results when minutes could be how long it takes for the user to miss a particular bus. Therefore, to measure the success here, I will compare the time taken for my application to return a predicted arrival time, after the user has already entered all the required details, with the time taken for the TfL to return a predicted bus arrival time on their website (which is pretty much instantaneous, so that is what my application should aspire to reach).

References

[1] 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.

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

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

[4] 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.

[5] Tony Lacey. Tutorial: The Kalman Filter. https://www.notion.so/Kalman-Filtering-Models-98ea83239def416083617327b1dcaf5f#9a2efce5c29b423396cc2097508b0dfb. Accessed: 2020-02-02.

[6] 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.

[7] 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.

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

[9] 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.

[10] Imad Dabbura. K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a, September 2018. Accessed: 2020-01-22.

[11] K Means, Clustering, scikit learn User Guide. https://scikit-learn.org/stable/modules/clustering.html#k-means. Accessed: 2020-01-22.

[12] Chris Piech. K Means, Stanford CS221. https://stanford.edu/~cpiech/cs221/handouts/kmeans.html, 2013. Accessed: 2020-01-22.

[13] Clustering performance evaluation, Clustering, scikit learn User Guide. https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation. Accessed: 2020-01-24.

[14] Tadeusz Caliński and J.A. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, Volume 3 Issue 1:1–27, January 1974.

[15] Peter Rousseeuw. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, Volume 20 Issue 1:53–65, November 1987.

[16] 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.

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

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

[19] 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.

[20] Ben Calderhead. Statistical Modelling 1 M2S2 Lecture Notes. https://union.ic.ac.uk/rcsu/mathsoc/files/M2S2-16.pdf, 2019. Accessed: 2020-01-27.

[21] Sara A. Van de Geer. Least squares estimation. *Encyclopaedia of Statistics in Behavioural Science*, Volume 2:1041–1045, 2005.

[22] 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.

[23] 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.

[24] Robert L. Peach. Artificial Neural Networks, Methods for Data Science, MATH96007/MATH97019/MATH97097. 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] 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.

[28] 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.

[29] Department for Transport statistics, National Travel Survey 2018: Average distance travelled by mode, region and Rural-Urban Classification: England . https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/821445/nts9904.ods, 2018. Accessed: 2020-01-20.

[30] 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.

[31] 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.

[32] Marek Rei and Josiah Wang. Introduction to Machine Learning CO395 Lecture 1. https://materials.doc.ic.ac.uk/view/1920/395/lecture_notes/0, January 2020. Accessed: 2020-01-15.