Bus Journey and Arrival Time Prediction based on Archived AVL/GPS data using Machine Learning

Ankit Taparia

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Dr. Michael Brady

September 2020

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Ankit Taparia

September 4, 2020

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Michael Brady for his attention, support and valuable guidance through each phase of this dissertation. I am indebted to his untiring perseverance, which helped me to present this work in the right perspective.

I would also like to thank my second reader Dr. Hitesh Tewari for his time, suggestions and attention to detail.

There are no words to express my gratitude to my parents and my sister for always standing by me. Their endless love and encouragement have been the major spiritual support in my life.

Ankit Taparia

University of Dublin, Trinity College
September 2020

Bus Journey and Arrival Time Prediction based on Archived AVL/GPS data using Machine Learning

Ankit Taparia, Master of Science in Computer Science

University of Dublin, Trinity College, 2020

Supervisor: Dr. Michael Brady

With a surge in the number of vehicles, traffic congestion has increased at an alarming rate. This has led to increase in travel times and decreased accessibility and mobility. One viable solution to mitigate this issue is to promote the use of public transport. Buses are considered one of the important means of public transport. However, to encourage the use of buses, there is a need to provide reliable bus travel time and arrival information to the commuters. In this study, we propose and develop predictive models to predict bus journey and arrival times based on historical AVL/GPS data and prior bus routes and stops information. There were two parts to this study. The first was to predict overall journey times and the second was to predict bus arrival times at bus stops.

To estimate total bus journey times, three models are developed using Linear Regression, Artificial Neural Network (ANN) and Long Short Term Memory Network (LSTM). Evaluation results on ground-truth dataset show that LSTM outperformed the Linear Regression model and its performance was comparable to that of ANN.

To predict bus arrival times at bus stops, three different models, namely Historical Averaging, Linear Regression and Gradient Boosting are proposed. Experimental results show that the Gradient Boosting outperformed the other models and is more robust in predicting arrival times.

Our study also reveals that it is possible to predict bus journey time with reasonable accuracy by using only GPS observations and bus routes information. This can be particularly useful in situations where data regarding other external features affecting travel time are not available.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Over recent years, traffic congestion has increased at an alarming rate and has become a global phenomenon (1). This surge is attributed to increases in motorization, urbanisation and population growth. Congestion creates burdens on transportation infrastructure, increases travel time, fuel consumption and pollution, and reduces accessibility and mobility. A way to mitigate this issue is by increasing the capacity of transport infrastructure by building more roads, highways, separate lanes etc. This option is not viable because of its own limitations. A more economical option is to encourage the use of public transport by the public and efficiently manage the existing resources using Intelligent Transportation Systems (ITS).

Buses are considered one of the important means of public transport owing to their coverage and accessibility by mass people. Buses are available all year long, are more economical and eco-friendly (2) than private vehicles. Moreover, in some cities buses have their own dedicated lanes which makes them faster than cars and help reduce travel times in heavy traffic conditions during peak hours. But to motivate and encourage usage of buses, providing commuters with reliable bus travel time and arrival information is essential. Advanced Public Transportation System (APTS) is an integral component of Intelligent transportation systems. With the advent of intelligent transport systems in cities, buses are fitted with GPS enabled in-vehicle navigation systems as part of urban planing. This tracking system integrated in buses generates automatic vehicle location (AVL) data which can be used to provide accurate bus arrival information to passengers waiting at the bus stops, leading to a decrease in waiting times. It increases the satisfaction of commuters by enabling them to plan their travel ahead of time and also attracts additional ridership.

Predicting accurate bus travel and arrival time is not an easy task because it depends on many external factors such as passenger load, passenger boarding/alighting time, number of signalised intersections, traffic congestion and weather. Moreover, it is not guaranteed to have information about all the above factors to predict bus journey times. Hence there is a need to develop intelligent models which can estimate reliable journey and arrival times using minimal features for situations where all features are not available.

1.2 Research Objectives

As discussed earlier, bus transit planning has an important role to play as part of urban transportation planning and providing passengers with accurate and reliable travel information is very important to increase additional bus ridership. In cities, bus transit times are difficult to estimate because travel times on links, dwell times at stops, and delays at intersections fluctuate spatially and temporally (3). It becomes even more challenging when we do not have much information/data about the external factors which influence the travel time.

The main objective of the thesis is to develop and compare models to predict bus journey and arrival times using archived/historical GPS-based AVL data along with prior bus routes information and stops information.

1.3 Thesis Structure

This thesis contains nine chapters including the introduction chapter. The chapters are described as follows:

- Chapter 2 presents literature review of research studies conducted for bus travel time prediction.

- Chapter 3 explains the dataset used for this study and the processes of data cleaning and pre-processing.
- Chapter 4 defines the evaluation metrics used for assessing the performance of the predictive models developed.
- Chapter 5 presents an analysis of bus journey times from the data.
- Chapter 6 discusses and presents models to predict total bus journey times and evaluation of their performances.
- Chapter 7 outlines concepts about a proposed model – LSTM and how it can predict bus journey times and its future trends.
- Chapter 8 presents and discusses models to predict bus arrival times and provides their in depth evaluation and comparison.
- Chapter 9 concludes the whole research with a reflection of the results obtained and provides future work recommendations.

Chapter 2

Related Works

Bus travel and arrival time prediction has been an active area of research for past decades. Researchers have explored and applied various approaches and techniques for accurate bus arrival time prediction. In general, those approaches/techniques can be grouped into four broad categories – historical data based approaches, statistical methods, machine learning techniques (neural networks, support vector machines, LSTM) and model based approaches (Kalman Filtering).

The history-based approach predicts the travel time of a future trip from observed historical bus travel time data of past journeys completed in the same daily time period over different days. It requires less computation and works well if the traffic conditions are stable over time. However, the model gives large prediction errors for any unexpected delay or congestion.

Statistical methods such as regression models are conventional approaches to predict a dependent variable based on a function formed by the independent variables. They model linear relationships between travel time and factors affecting travel time such as dwell times at stops, passenger load, number of traffic signal, etc. They simultaneously measure how much each independent factor affects the journey time. Since variables in transportation systems are correlated, accuracy depends on identifying truly independent variables and incorporating them in the model.

Machine learning techniques, especially Artificial Neural Networks (ANN) have been popular among researchers in the recent past and have been commonly used in many studies because of their ability to capture complex non linear relationships between dependent and independent variables and deal with noisy data. Other machine learning techniques which have been used in this domain are Support Vector Machines (SVM), K-Nearest Neighbors (KNN) and Long Short Term Memory (LSTM).

Kalman filtering is a model based approach used to predict the future state of a dependent variable. Kalman filtering models have elegant mathematical representations (e.g. linear statespace equation) which can adequately accommodate traffic fluctuations with their time-dependent parameters. There are many previous studies which have utilised Kalman filtering-based algorithms in travel time prediction models and have found the models' performance robust and promising.

Some of the research studies using the above techniques are summarized below.

In a paper presented at ITSC 04', Jeong et al. (4) presented an ANN model to estimate bus arrival times using automatic vehicle location. In addition to the past journey data, the proposed ANN model incorporated bus dwell times at stops and traffic information. In their study, the authors also developed a historic data based model and multiple linear regression models and compared their performance with the ANN model. The models were then tested on a transit route in Houston, Texas. The results showed that the ANN performed considerably better than the historical data based model and regression models in terms of mean absolute percentage error (MAPE).

In a journal paper published in IJTST 15', Fan et al. (5) developed and compared Historical Average, Kalman Filtering and ANN models for predicting bus travel times using GPS data. The models were evaluated for a bus route in Macae city, Brazil. Experimental results suggested that the ANN outperformed the other models in prediction accuracy. The proposed ANN performed best when observed travel times were in a range 20 to 50 minutes but gave large prediction errors for very short and long trips. The study also revealed that it is possible to predict bus travel times using only arrival and departure time information at stops even in the absence of traffic related data.

In another study, Chen et al. (6) used automatic passenger counters (APC) to predict bus arrival times for a specific bus route in New Jersey. The authors developed an ANN based model for prediction which takes APC data features as input in addition to weather information during the same time period considered for analysis. It was observed that precipitation had a strong impact on bus delays. The authors also performed sensitivity analysis to determine which input features had greater influence in estimating bus arrival times (BAT).

On the other hand, Liu et al. (7) presented a modified KNN method using principal component analysis to predict bus arrival times for a bus route in Beijing. The idea was to identify the most similar past sequences of trips to predict the BAT of the current trip. GPS data points were pre-processed to obtain arrival times, dwell times and departure time at bus stops. To evaluate the performance of the KNN model, the authors did an ANN model analysis and found that the proposed KNN performed better than the ANN.

Shalaby and Farhan (8) used AVL and APC data from bus route 5 of Toronto to predict bus arrival times taking into account the effect of dwell times. In their study, bus dwell time was not included in the link travel time. Instead, the authors proposed a model based on two Kalman filtering algorithms – one for modelling link travel time and other for predicting bus dwell times. Bus dwell time was calculated as a function of number of passengers boarding and alighting at a bus stop, which was obtained using APC data. The authors claim that, as the proposed model takes the effect of dwell times on bus arrival times into account, it outperformed the regression and neural network models and is more robust. The proposed model also showed promising results when tested on simulated scenarios representing a passenger surge and lane closure.

In another study by Chien et al. (9), two artificial neural networks were trained using link-based and stop-based data to estimate transit arrival times for bus route 39 of New Jersey. The data used for training the model comprised bus journeys of route 39 completed in the morning peak time (7:30–9:30 AM). The authors used the microscopic simulation model – CORSIM (10) to simulate bus operations and generate real time AVLS data for the specific route in the absence of GPS based AVL data. To further enhance the prediction accuracy, the authors introduced an adaptive algorithm and integrated it with the ANN models to take care of the prediction errors in real time.

He et al. (11) investigated predicting total travel times for passengers using multiple bus trips. LSTM is used to predict bus riding times for multiple trips whereas waiting times at transfer points were estimated using an interval based historical averaging method.

Reddy et al. (12) developed a support vector machine (SVM) model with a linear kernel function to predict bus arrival time/travel time under Indian traffic conditions, which are said to be prone to high variability due to lack of lane discipline and have heterogeneous vehicle profiles. The proposed SVM model was compared with a Kalman filtering based model. It was observed that the models' performance were comparable for trips completed during non-peak hours. However, for peak trips SVM showed lower level of prediction errors and was better able to capture the travel time variations.

Several studies have used hybrid models – combining two or more models to predict bus arrival times. In one such study Zaki et al. (13) proposed a model in which neural network is used along with Kalman filtering. The ANN predicts the time based on historical trips data and the Kalman filtering adjusts the predictions made based on real time GPS information of the bus. In another study, Yu et al. (14) presented a hybrid model in which SVM is used to predict the baseline travel times on the basic of historical trips whereas Kalman filtering based dynamic algorithm uses the latest bus arrival information to compute unexpected delays. Both combined together predicts the bus predict arrival times for the next stops. To evaluate the performance of the hybrid model, the authors built a ANN-based model using the same dataset. Test results showed that the hybrid model outperformed the ANN-based model. A similar approach is used by Seng et al. (15) who used a combination of static algorithm (SVM) and dynamic algorithm (Kalman filtering) for bus arrival time prediction. SVM provided a temporary prediction based on historical data and then Kalman filtering made a dynamic correction to it and predicted the final travel time. The authors claim that the proposed model is more stable and can make accurate predictions even for unexpected situations. However, a more comprehensive analysis and comparison with other models such as ANN could have been performed to strengthen the claim.

Chapter 3

Data

3.1 Dataset Used and its Features

We used a dataset provided by Dublin City Council which was acquired from Smart Dublin (16). It is a collection of global positioning system (GPS) points for buses in Dublin, Ireland from 6th November 2012 to 30th November 2012.
It contains $\approx$ 35 million rows and 15 features as explained below. Each row corresponds to a GPS observation and includes the following feature variables:

- Timestamp - UNIX time at which the observation was made
- Line ID - bus route number
- Direction - whether the bus is going upstream or downstream
- Journey Pattern ID - unique identifier for the route to which the observation was logged
- Production TimeFrame - the start date of the production time table
- Vehicle Journey ID – a given run on the journey pattern
- Operator - bus operator name
- Congestion - road congestion at time of observation
- Long - longitude of bus at a particular observation

- Lat - latitude of bus at a particular observation
- Delay - delay in seconds
- Block ID - a section ID of the journey pattern
- Vehicle ID - ID of the bus whose observation is made
- Stop ID - ID of the nearest stop at time of observation
- At Stop - tells whether the bus is located at a stop at time of observation

3.2 Cleaning and Pre-processing of Raw Data

Data is cleaned and pre-processed before building the models in the following manner:

1. **Removing null values and changing data-types**
   Rows containing null values for any of the columns – 'Line ID','Lat','Long','Stop ID' are dropped.
   The 'Stop ID' column datatype is changed from object type to integer whereas the 'Line ID' is changed to string type to accommodate alphanumeric characters.

2. **Converting Unix Timestamp to Date-Time format**
   The Timestamp in the dataset is UNIX time based. Therefore to extract exact date and time (hours, minutes) information it is converted to YYYY-MM-DD hh:mm:ss date-time format.

3. **Eliminating uninformative features**
   The features – 'Direction' and 'Congestion' have values of 0 throughout the dataset and are removed from the data to be used for building the models.

4. **Deriving new features**
   New features — 'day of week', 'hour' are extracted from the converted Date-Time format for each of the observations.
   A feature – 'distance from city centre' which denotes the distance of a bus from the Dublin City Centre for an observation is created. For this purpose, a point on O'Connell Street was designated the centre of the city. The Haversine formula

(17) is used to calculate the distance between the designated centre point and the GPS coordinates of the bus.

3.3 Creating a Dictionary of Bus Stops using Dublin Bus GTFS Information

Each observation in the dataset is associated with a Stop ID closest to the actual position of the bus at that particular instant. The ‘At Stop’ feature of the dataset indicates whether the bus is standing at the stop at the time of observation. However, there are many instances in the dataset for which the values of ‘Lat’ and ‘Long’ were different for observations with the same Stop ID and an ‘At Stop’ value of 1. This suggests that if a bus is within a certain threshold distance from a bus stop, the value of ‘At Stop’ is set to 1. But it also means that it is not possible to derive the true latitude and longitude of the stops entirely from the dataset. Therefore, Dublin bus General Transit Feed Specification (GTFS) information (18) is used. GTFS (19) is an open data format for public transportation schedules and associated geographic information. It provides information about bus stops, bus routes, stop times etc.
Unique bus Stop IDs from the dataset are extracted and searched against bus Stop IDs present in GTFS data. For every matched Stop ID, its information – latitude, longitude and name of the stop – is stored in a Python dictionary data structure. This dictionary is used later in the study when we build models for predicting segment-of-route journey time.

Figure 3.1 shows various bus route trajectories in Dublin obtained by plotting individual GPS observations after cleaning the data. MapBox (20) is used for rendering the map.

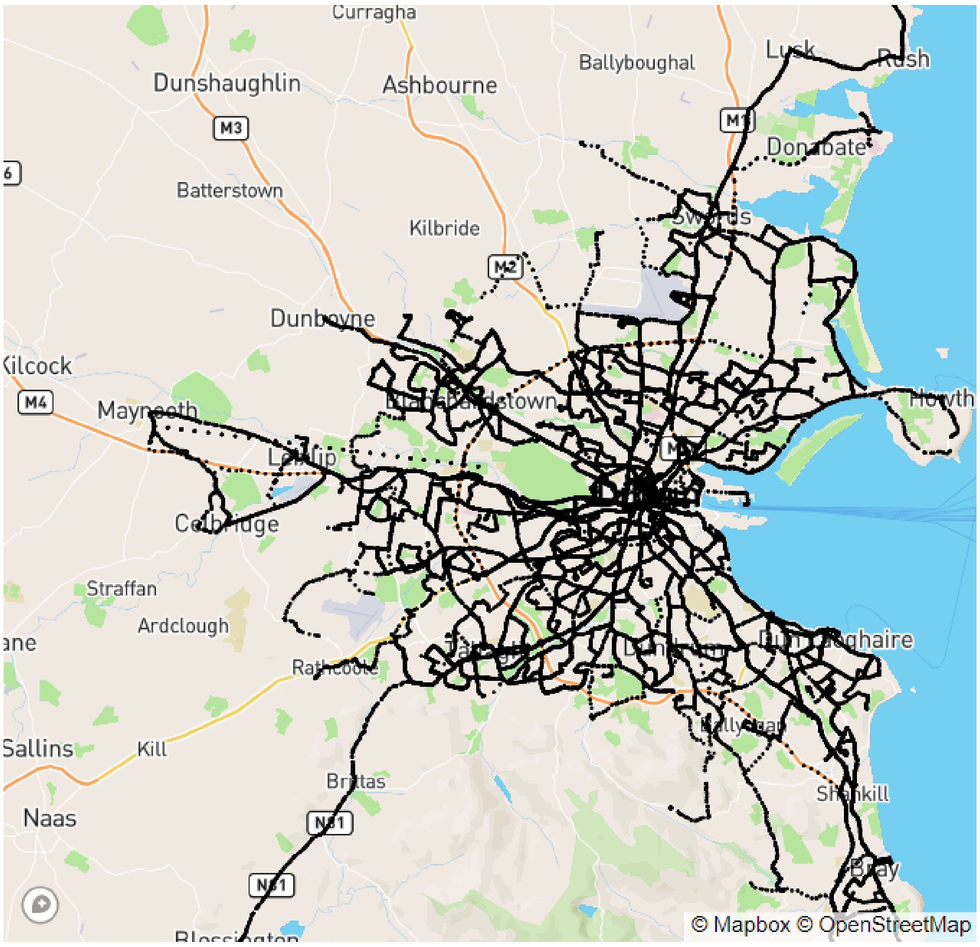


Figure 3.1: Bus routes obtained by plotting individual GPS observations after data cleaning and pre-processing

Chapter 4

Evaluation Metrics

In this chapter, we define the evaluation metrics which have been used to measure and compare the performance of our predictive models.

Let there be N test samples. $Y_{predicted}[i]$ and $Y_{actual}[i]$ denote the predicted journey times and actual journey times $\forall$ i $\in$ N .

- The **Maximum Absolute Error (MaxAE)** measures the maximum deviation of predicted journey time from actual journey time.

$$MaxME = \max |Y_{actual}[i] - Y_{predicted}[i]| \tag{4.1}$$

- The **Mean Absolute Error (MAE)** measures the average magnitude of the individual prediction errors for test data, without considering their direction.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |Y_{actual}[i] - Y_{predicted}[i]| \tag{4.2}$$

- The **Mean Absolute Percentage Error (MAPE)** measures the average percentage deviation of predicted journey time from actual journey time.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N} \frac{|Y_{actual}[i] - Y_{predicted}[i]|}{Y_{actual}[i]} \tag{4.3}$$

- The **Maximum Absolute Percentage Error (MaxAPE)** measures the maximum percentage deviation of predicted journey time from actual journey time.

$$MaxAPE = \max\{\frac{|Y_{actual}[i] - Y_{predicted}[i]|}{Y_{actual}[i]}\} \tag{4.4}$$

- The **Root Mean Squared Error (RMSE)** measures the standard deviation of the residuals (difference between predicted and actual values). Compared to Mean Absolute Error, RMSE puts a large weight on large errors.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Y_{actual}[n] - Y_{predicted}[n])^2}{N}} \tag{4.5}$$

Chapter 5

Analysing Total Bus Journey Times

After the data is cleaned and pre-processed, we analyse the bus journey times from source to destination for some specific bus routes. By total journey time we mean the time taken by the bus to traverse the whole route i.e. from its start stop to end stop. From the data, it is found that for routes (Line IDs) - 40, 46 and 145, maximum GPS data observations are recorded. Hence these three bus routes (40, 46 and 145) are considered for our analysis as we are able to extract significant number of trips from these particular routes.

5.1 Determining Source and Destination Stops for Routes

A bus trip is uniquely identified by a combination of Date, Line ID and Vehicle Journey ID features. Firstly, trips are segregated using Line ID which denotes a particular bus route. Then for each bus route, unique trips are extracted from the data. Thereafter, for each route we iterate over all of its trips. For each trip we store its start and end stop pair and increment its count. The pair for which the count is maximum is chosen as the source and destination stop for that route. It also helps in selecting those stops as the source and destination for which maximum trips are available for analysis.

5.2 Calculating Total Journey Time for Routes

Once the source and destination stops are determined for each bus route, the total journey time of each of the trips is calculated using the timestamp information recorded at the source and the destination stops. There are instances for which multiple GPS observations are recorded at source and destination bus stops. This indicates the bus is either waiting at the source bus stop to start its journey or it has reached the destination and waiting to start its onward journey. In such a case, the timestamp of the last observation made at the source stop for which 'At Stop' = 1 is considered to be the journey start time. Similarly, the the timestamp of first observation made at the destination stop for which 'At Stop' = 1 is considered to be the journey end time. The total journey time is the difference between the chosen timestamps recorded at the destination stop and source stop for each bus trip.

5.3 Analysing Total Journey Time Graphs for Routes

Total journey times for routes 40 and 46 are are calculated over a time period from 6th November to 30th November 2012 and plotted in chronological order in Figures 5.1 and 5.2 .

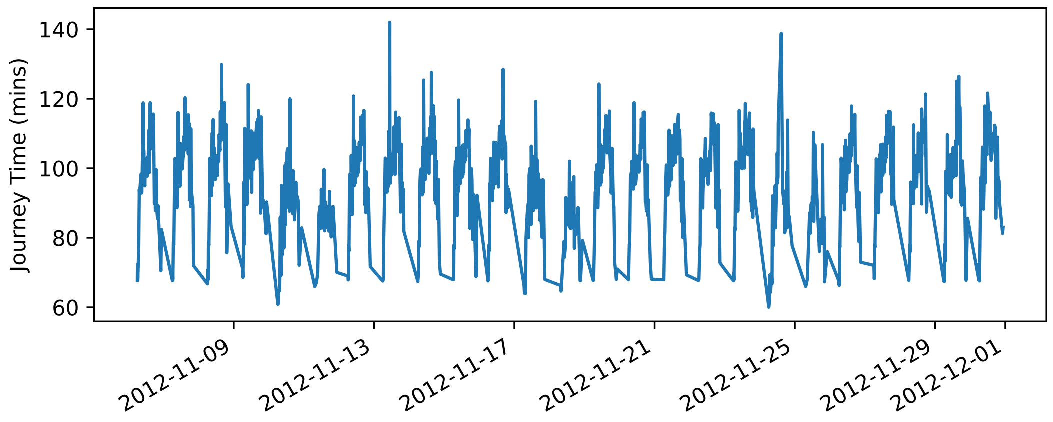


Figure 5.1: Total bus journey times for bus route 40

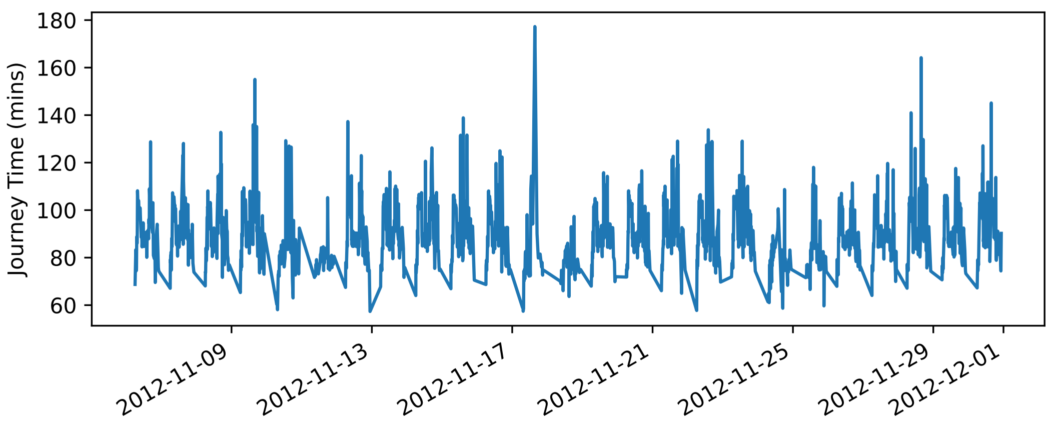


Figure 5.2: Total bus journey times for bus route 46

Both the plots reveal that the bus journey times follow a certain pattern. Journey times over weekdays are somewhat similar but higher than for journeys completed at weekends. Sudden drops in peaks for Saturdays and Sundays account for this feature. It may be because of fewer vehicles on the road leading to less traffic and low passenger demand on Saturdays and Sundays which are not working days in Dublin.

Some unusually high journey times can also be seen in the plots. These observations, called outliers, can be due to inconsistencies in readings of the GPS transmitter installed in the bus or due to accidents or bus breakdowns.

To further analyse the variation in journey times for a particular day, we plot the journey times of trips completed on 6^{th} November 2012 (Tuesday) for route 46. It can be observed from Figure 5.3 that journey times tend to be longer in morning time (07:00 – 09:00) and evening time (16:00 – 17:00) than in the afternoon owing to traffic congestion because of people going or returning from work.

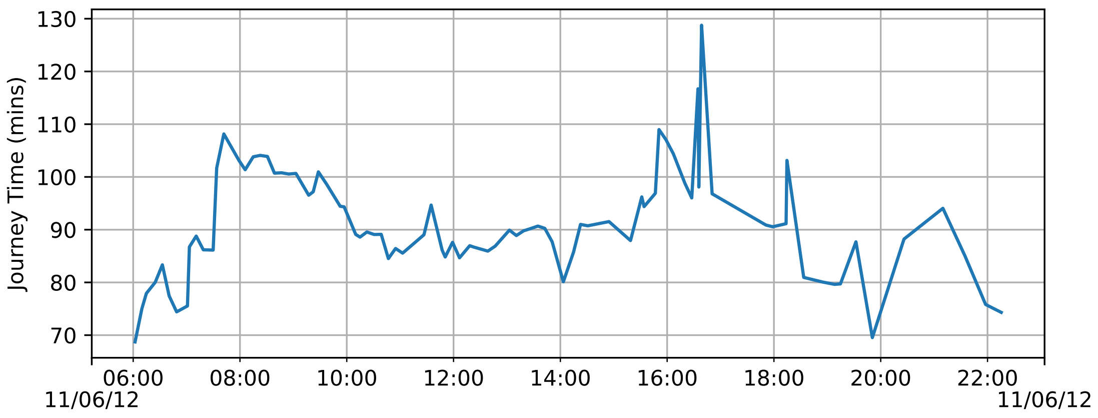


Figure 5.3: Total bus journey times for bus route 46 on 6th November 2012

Chapter 6

Total Bus Journey Time Prediction

After pre-processing the data and analysing journey trips for most observed bus routes, we try to build predictive models to predict journey times for bus trips based on relevant journey features. For this purpose, models are built using Linear Regression and Neural Networks. We build models for specific bus routes as well as generalised models for multiple bus routes. Metrics such as Maximum Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE) are used to compare and evaluate the performance and robustness of the models.

6.1 Removing Outliers

Outliers are observations that are significantly different from other data points. They adversely affect the training process of the machine learning models resulting in less accurate models and poor predictions. So it is important to detect and remove outliers from data before it is being fed to the models. In our case, outliers are journeys which took unusually long times to complete. This may have several causes – bus involved in an accident, bus had to diverge from its route or other special event.

We use a Z-score to identify and remove outliers from the data. A Z-score is a measure of how many standard deviations an observation is away from the group mean. The Z-Score for any observation/data point x is given by:

$$Z - Score = \frac{x - \mu}{\sigma} \tag{6.1}$$

where μ is the mean of the data points and σ is the standard deviation of the data points.

If the value of the Z-score is greater than 3 or less than -3 , then that observation is identified as an outlier. This is illustrated in Figure 6.1 below.

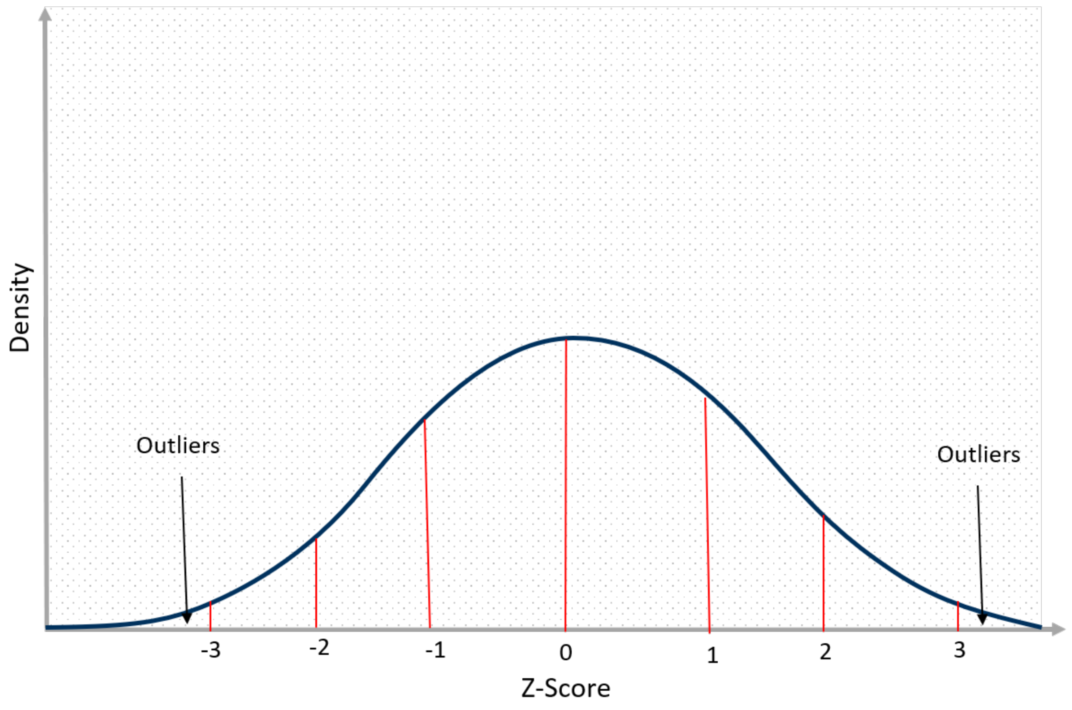


Figure 6.1: Outlier Detection using Z-score

6.2 Predictive Models for a Single Bus Route

6.2.1 Linear Regression Model

Overview

Linear regression (21) is a very simple and useful approach in supervised learning to predict a quantitative response. It models linear relationships between dependent and independent variables and helps to determine how strong those relationships are.

Suppose we have N data samples and m feature variables such that the i^{th} sample is represented as ($Y^{(i)}$,$X_1^{(i)}$,$X_2^{(i)}$,..., $X_m^{(i)}$).

The predictor variable $Y^{(i)}$ is modelled using feature variables $X_1^{(i)}$,$X_2^{(i)}$,..., $X_m^{(i)}$ as follows:

$$Y_{pred}^{(i)} \approx \theta_0 + \theta_1 X_1^{(i)} + \theta_2 X_2^{(i)} + \cdots + \theta_m X_m^{(i)} \tag{6.2}$$

where θ_0 is the intercept and represents the value of Y when $X^{(i)} = 0$ and θ_i are the regression coefficients where i = 1 to N.
The value of θ_i denotes the change in predictor variable Y on changing the feature variable $X^{(i)}$ by 1 unit.

The values of θ_i are chosen such that they minimize the loss function J(θ) given as:

$$J(\theta) = \frac{\sum_{i=1}^{N} \left(Y^{(i)} - Y_{pred}^{(i)}\right)^2}{N} \tag{6.3}$$

We start with random values of θ and then use gradient descent to update the values of θ in the direction which decreases the loss function for each iteration. We stop iterating once θ assumes the value of $\theta_{optimal}$ as shown in Figure 6.2.

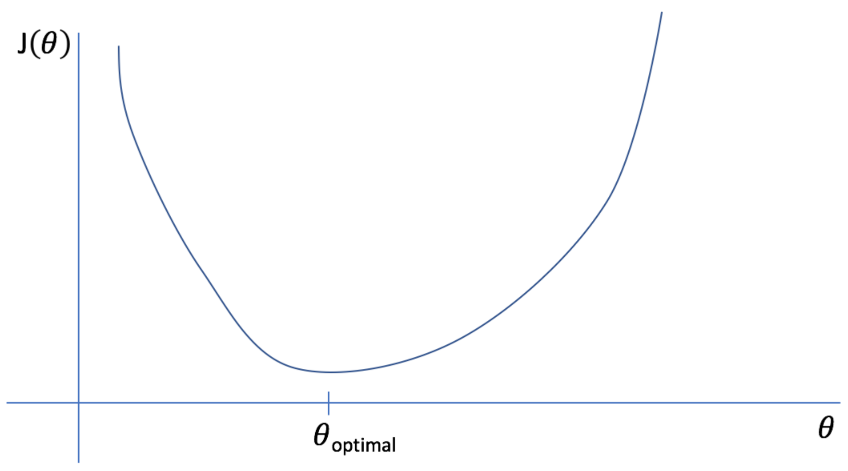


Figure 6.2: Gradient descent for optimal θ

Features Used

We use the features ‘day of week’, ‘hour’ and ‘delay’ to build the model. Therefore these are the independent variables whereas ‘journey time’ is the dependent variable which the model will predict.

Training and Test Data

A total of 1989 journeys of bus route 46 are used of which 80% are used for training and the remaining 20% for testing the model.

Figures 6.3 and 6.4 show the actual journey times vs. predicted journey times for journeys done by bus number 46.

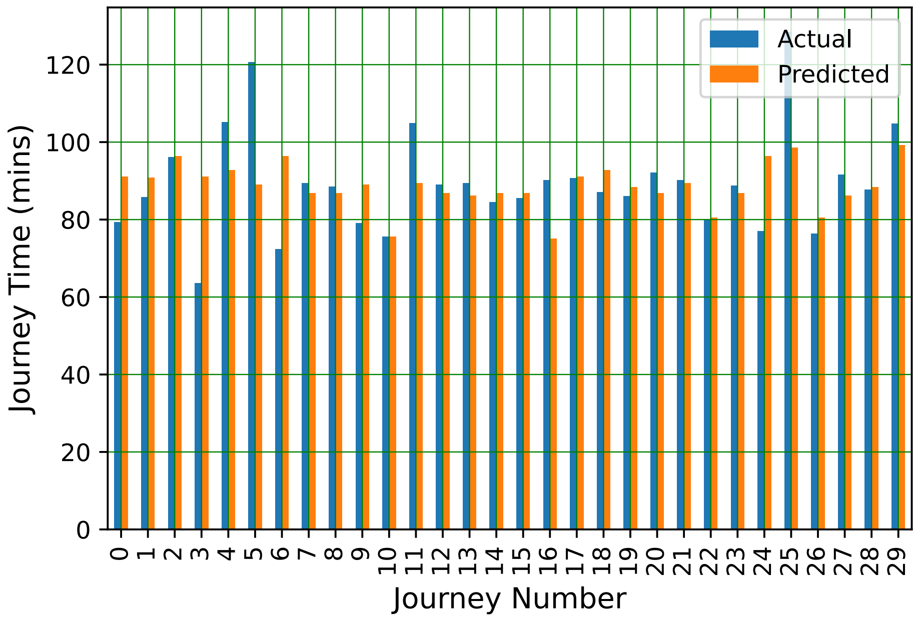


Figure 6.3: LR: Actual vs Predicted Journey time for the first 30 test journeys

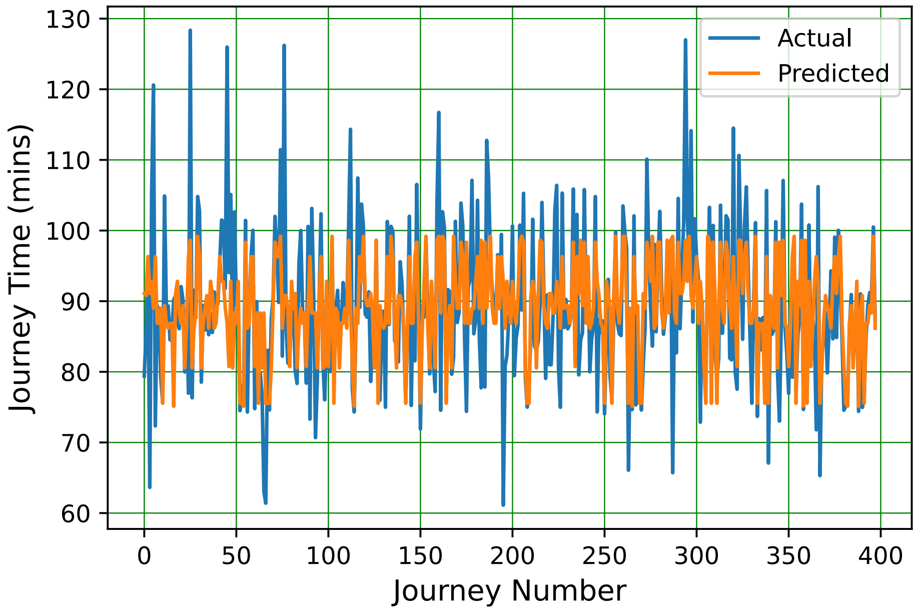


Figure 6.4: LR: Actual vs Predicted Journey time for all test journeys

6.2.2 Artificial Neural Network Model

Overview

Artificial Neural Networks (ANN) (22) are a powerful technique for capturing and modelling non linear complex relationships between inputs and outputs. They are inspired by the way a human nervous system works such as how the brain processes information and are able to recognize relationships and patterns in the environment.

Typically an ANN consists of three layers – Input, Hidden and Output. The input layer takes the inputs, the hidden layer processes the inputs, and the output layer produces the final desired output. Essentially, each layer tries to learn certain weights when they process the training samples using the back propagation technique (23). Figure 6.5 shows a a general architecture of multilayer feed-forward ANN, where x_i is the i^{th} input to the input layer and w_i , w_k are the weights of the connections from input to hidden layer and from hidden layer to output layer respectively. Finally, T represents the output which the ANN predicts.

Activation functions form the core of the ANN. They introduce non linearity in neural networks and help the ANN to learn complex relationships between inputs and output. Each neuron in the network calculates a weighted sum of its input, adds a bias (constant) and then the activation function defined for it determines whether the neuron should be activated or not. So for any neuron its output Y is defined as:

$$Y = f(\sum(weights * input) + bias) \tag{6.4}$$

where f is the activation function, which can be a step function, or a sigmoid function or a tanh function.

For a step-based activation function if Y $>$ 0, the neuron will be activated and will output 1 else the output will be 0. Therefore, the inputs determine which neurons get activated or deactivated.

For the j^{th} neuron in the hidden layer, its output is given as:

$$y_j = f\left(\sum x_i w_i + b_j\right) \tag{6.5}$$

where $\sum x_i w_i$ represents the sum of weighted input from the input layer to the j^{th} node of the hidden layer and b_j is the bias.

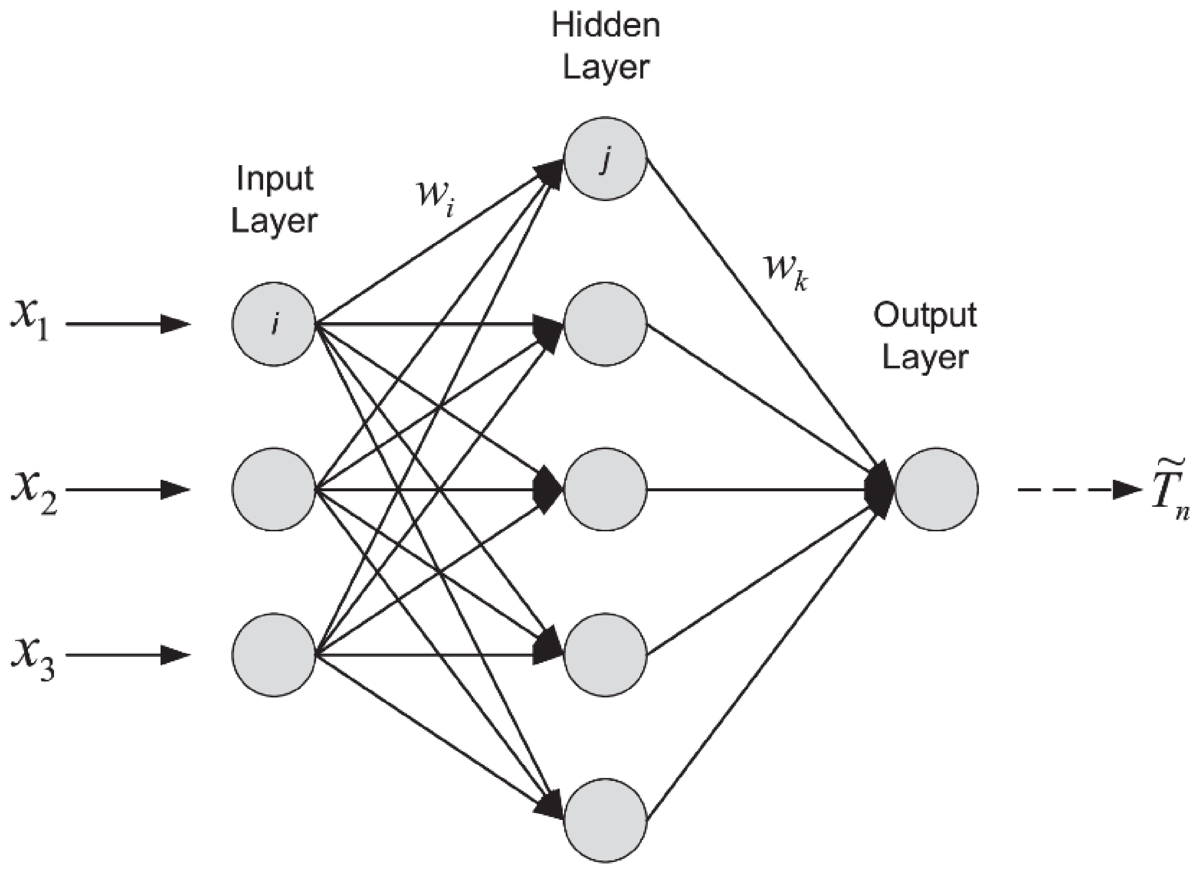


Figure 6.5: Artificial Neural Network

Similarly, the final model output $\widetilde{T_n}$ is given as:

$$\widetilde{T_n} = f\left(\sum y_j w_k + b_o\right) \tag{6.6}$$

where w_k represents the weight of the connections from hidden nodes to the output node and b_o is the bias.

As mentioned in Chapter 2, Artificial Neural Networks have been a popular choice among the researchers for travel time prediction (5) (6) (13). They can identify relationships and patterns in datasets and can approximate any arbitary input-output mapping. However, neural networks require much more data than traditional machine learning algorithms to generalize well and to make better predictions (24).

Features Used

We use the same set of features as used in the Linear Regression evaluation. Therefore features used are 'day of week', 'hour' and 'delay'. One-hot encoding (25) is applied to 'day of week' and 'hour' before being fed into the model.

Model Architecture

We use a four layered feed-forward neural network (Figure 6.6) which consists of an input layer, two dense hidden layers and an output layer. The features are fed into the network through input layer which has 24 neurons. The hidden layers consist of 25 neurons each with each neuron in one hidden layer connected to other neurons in the next layer. Hidden layer neurons are activated by a Rectified Linear Unit (ReLu) (26) activation function. Finally, the output layer consists of one node with linear activation function which gives the predicted journey time. Biases are included in hidden and output layers.

A regularisation technique - Dropout (27) with value of 0.3 is applied to the hidden layers. It reduces the interdependence between neurons thus prevents the neural network model from overfitting and making it robust to various unseen inputs.

To stabilize the learning and make the neural networks converge faster Batch Normalisation (28) is applied to the outputs of each hidden layer. It standardizes the inputs to a layer for each mini-batch and thus significantly reduces the number of training epochs required to train the network.

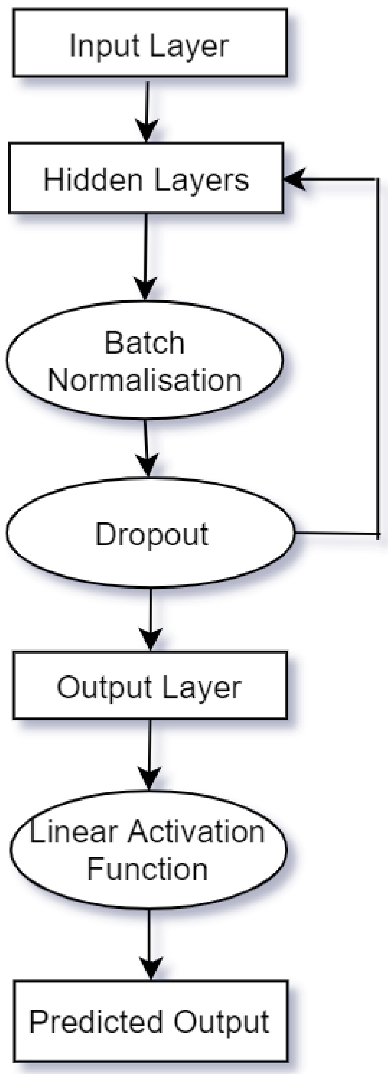


Figure 6.6: ANN: Model Structure

Hyperparameter Tuning

A Hyperparameter is a parameter whose value is used to control the learning process in a machine learning model. Hyperparameter tuning is the process of choosing a set of optimal hyperparameters for a learning algorithm. A neural network takes lots of hyperparameters which need to be set correctly before the training process commences so that the model produces results with the highest accuracy. We use *scikit-learn's* RandomizedSearchCV method (29) (30) for hyperparameter tuning to obtain the best combination of values of the hyperparameters for our model. Table 6.1 summarizes the grid of hyperparameter ranges we provided to RandomizedSearchCV and the optimal values determined by it.

Table 6.1: Hyperparameters for ANN

Hyperparamter	**Parameter values**	**Optimal value**
init_ mode	glorot_uniform, uniform	uniform
batches	10,20,30,50,90,128,512	50
epochs	10,30,50,200	50
lr	1e-2, 1e-3, 1e-4	1e-2
decay	1e-6,1e-9,0	1e-9
activation	ReLu, sigmoid	ReLu
dropout	0, 0.1, 0.2, 0.3	0.3

Training, Validation and Test data

The model is trained using 1591 bus trips and the network is validated over 200 trips of bus route 46 with a batch size of 50 for 50 epochs. ReLu activation functions are used in hidden layers. An Adam optimiser is used to update network weights with a learning rate of 0.01.

The loss model is set to Mean Squared Error (MSE) which the model tries to minimize while training :

$$Loss = \frac{\sum_{i=1}^{N} \left(Y_{actual}[n] - Y_{predicted}[n]\right)^2}{N} \tag{6.7}$$

To prevent the overfitting, we use the early stopping mechanism to stop the training process once the value of loss function for validation set stops decreasing.

The loss curve for the same is shown in Figure 6.7. It can be observed that both the train and validation loss decrease with each epoch and then stabilize.

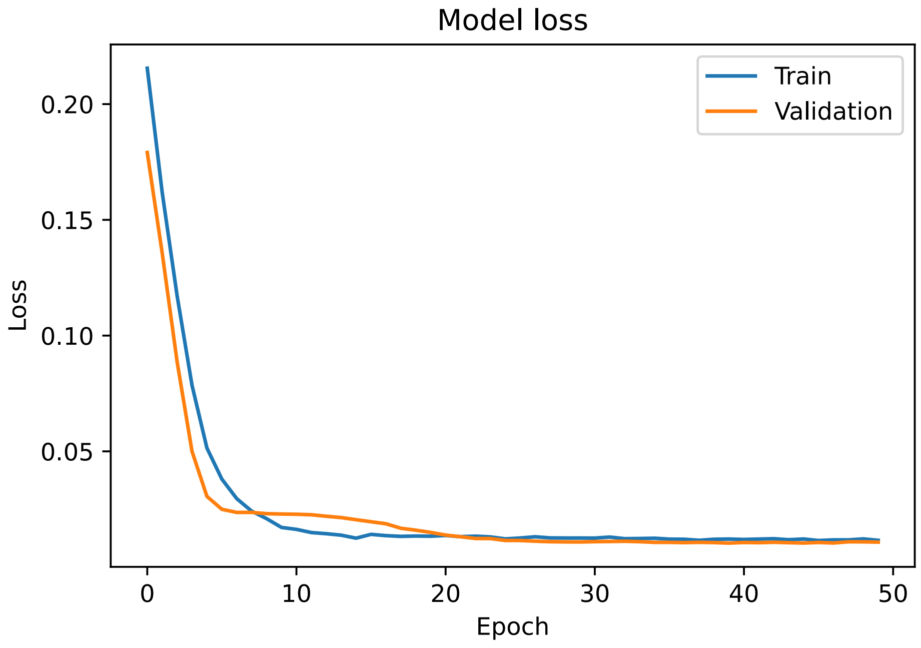


Figure 6.7: ANN: Training and Validation Loss

Once the network is trained, it is tested against 398 trips that are not part of the training set. Figures 6.8 and 6.9 represent the actual journey times and predicted journey times by the ANN for the test journeys.

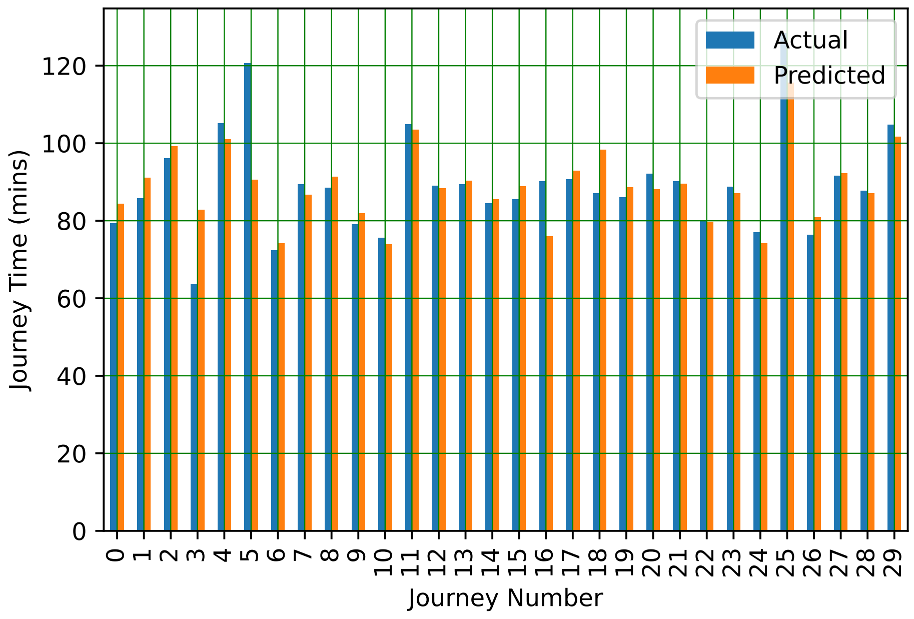


Figure 6.8: ANN: Actual vs Predicted Journey time for first 30 test journeys

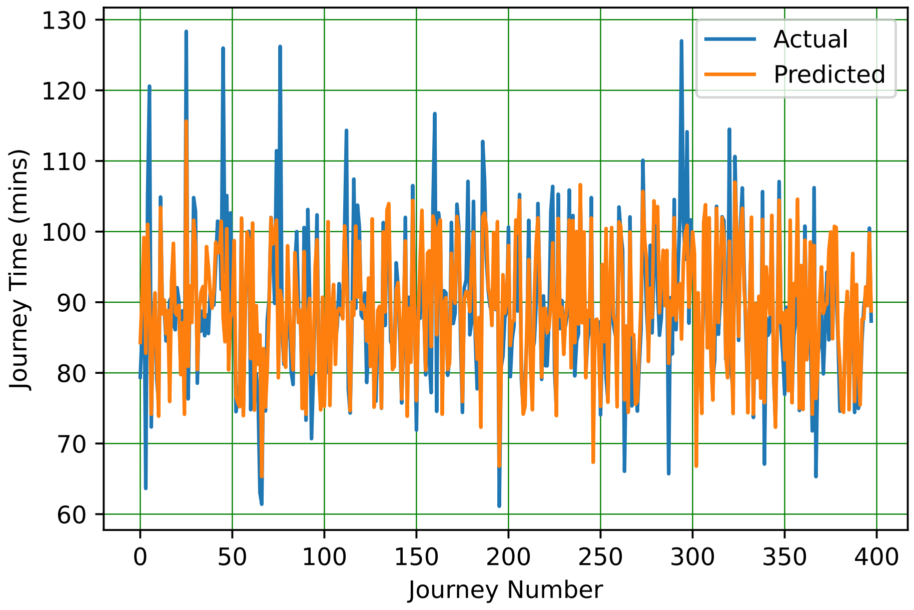


Figure 6.9: ANN: Actual vs Predicted Journey time for all test journeys

6.2.3 Results and Evaluation

To visually compare the performance of the two models for predicting total journey time, we plot their corresponding predictions on the same graph as shown in Figure 6.10. It can be observed that the ANN performs better than Linear Regression for almost all the journeys. In particular, for journeys which have lower travel times, the ANN's predictions are closest to the actual travel time.

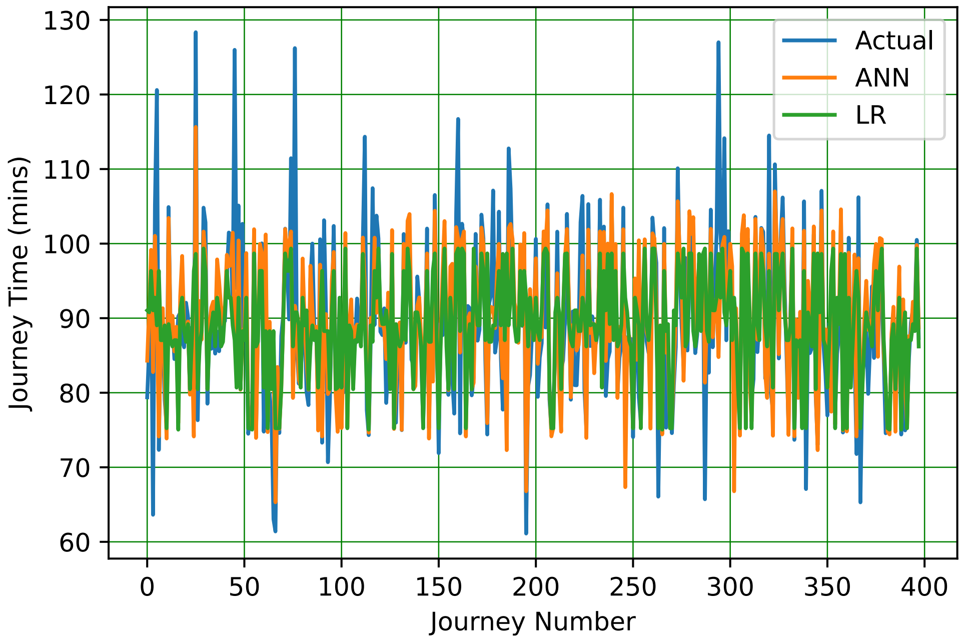


Figure 6.10: Comparison: Linear Regression vs ANN

Figure 6.11 shows the absolute error of both models for the test journeys. We can clearly see that the peaks in the graph are mostly dominated by the Linear Regression model. The graph also reveals that the ANN makes much lower prediction errors than the Linear Regression model.

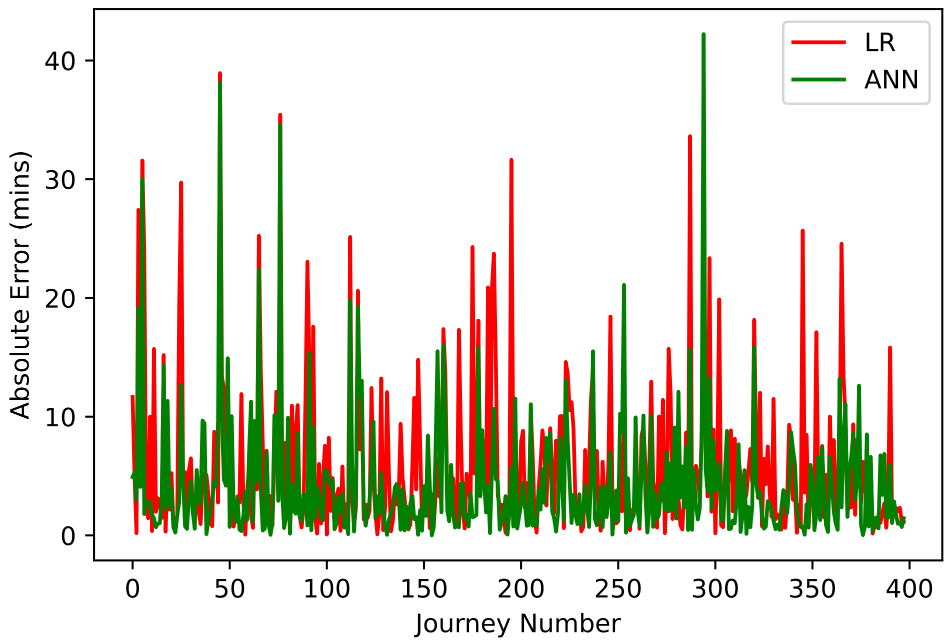


Figure 6.11: MAE for Linear Regression and ANN

To further strengthen the argument, evaluation metrics are calculated for both models and are summarized in Table 6.2. It can be observed that the ANN outperformed the Linear Regression model for all the three evaluation metrics. The MAE signifies that on average Linear Regression predicts total journey times deviating 7.6 minutes from the actual time whereas ANN predictions have a deviation of only 4.3 minutes.

Table 6.2: Results Obtained

Model	**MAE (mins)**	**MAPE (%)**	**RMSE (mins)**
Linear Regression	7.6130	7.2406	10.0342
Artificial Neural Network	4.3491	4.8364	6.7541

A RMSE of 6.75 minutes is observed for ANN which is an improvement of 28% over the Linear Regression model of 10 minutes. This suggests there exists some non-linear relationships between the dependent variable (journey times) and independent features which could not be captured by the Linear Regression model.

However, both models make large prediction errors for some trips with longer travel times. It may be that those trips were completed during rush hour and thus models could not make reliable predictions due to lack of traffic congestion information in the dataset.

6.3 Linear Regression Model to Predict Total Journey Times for Multiple Routes

Both models discussed above make journey time predictions for a single bus route. In this section, we experiment and develop a Linear Regression model to predict total bus journey for multiple routes. Bus routes 40, 46 and 145 which have most GPS observations in the data is used as input.

Features Used

We use the features – 'day of week', 'hour' and 'delay' which were also used in the models for single route. In addition, new features – 'distance of source stop from city centre', 'distance of destination stop from city centre' and 'route length' for each of the routes are computed and used.

Training and Test Data

Input data consists of a total of 4807 journeys for bus routes 40, 46 and 145. It is divided into a training set (80%) and a test set (20%). Breakdown for the training and and test data route-wise is given in Table 6.3.

Table 6.3: Training and Test Data

Bus Route	**Route Length (km)**	**Training Trajectories**	**Testing Trajectories**
40	8.64	1002	237
46	17.53	1616	386
145	22.25	1201	332

Results and Evaluation

Figure 6.12 shows the actual journey times and predicted journey times for all test journeys comprising bus route numbers 40, 46 and 145.

Figure 6.13 shows the predictions made by the model by individual routes.

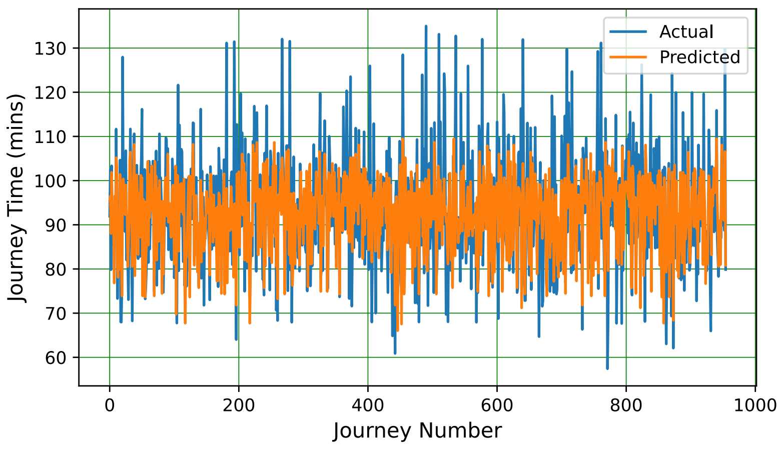
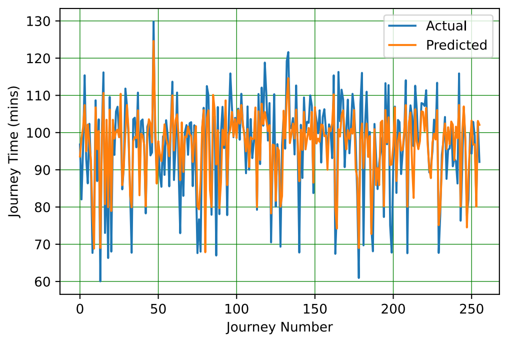
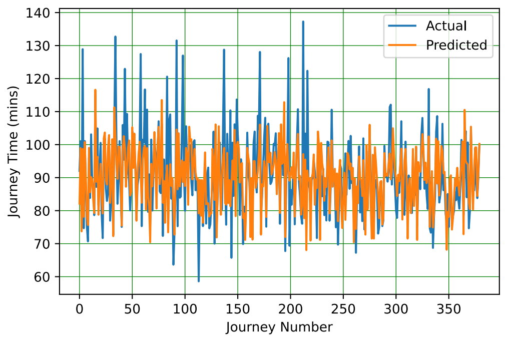


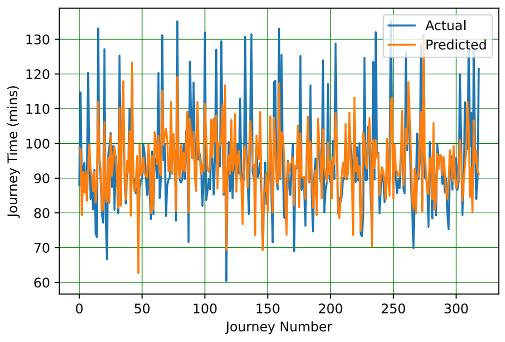
Figure 6.12: Actual vs Predicted Journey times for bus routes 46, 40 and 145



(a) Route 40



(b) Route 46



(c) Route 145

Figure 6.13: Actual vs Predicted Journey times for individual bus routes

An overall MAE of 6.793 minutes, MAPE of 7.29% and RMSE of 9.30 minutes is recorded for the test data. In general, the model's overall performance is reasonably good for journeys with average and low journey times. However it fails to accurately predict journey times for the bus journeys which actually took longer to complete (journeys during peak hour) as observed from Figures 6.12 and 6.13.

Evaluation metrics are also calculated by route and shown in Table 6.4. It is observed that the model's performance for a shorter route (route 40) is comparatively better than for a longer route (route 46, route 145). It suggests that with increasing route length, the external factors affecting travel times become more prominent and unpredictable, thus making it harder for the model to make accurate predictions.

Table 6.4: Evaluation metrics calculated for each route

Bus Route	**MAE (mins)**	**MAPE (%)**	**RMSE (mins)**
40	5.497	6.110	7.204
46	6.355	6.865	8.942
145	6.902	7.067	9.663

Chapter 7

Predicting Overall Journey Time Trends using LSTM

7.1 LSTM Overview

Long Short Term Memory (LSTM) networks (31) are a variant of Recurrent Neural Networks (RNNs) which are suitable for processing long sequence of inputs and predicting time series whereas RNNs suffer from the short term memory. For long sequences, RNN is unable to capture dependency between earlier time steps and the current time step. This is due to the vanishing gradient problem (32). During the back propagation, gradients are calculated which are used to update the neural networks weights. In the vanishing gradient problem, the gradient shrinks as it back propagates through time. As the gradient value becomes extremely small, it does not contribute to much learning of the network. This results in earlier layers of the network stopping to learn and therefore RNN tends to forget the longer sequence of input data which might be useful in predicting the current state.

LSTM overcomes the problem of the vanishing gradient faced by RNNs through its internal mechanism of gates. Its various gates regulate the information and enables the network learn which data in a sequence is important to keep or discard for predicting the current output. By doing that, it passes useful information down the long chain of sequences to make predictions.

Figure 7.1 shows a LSTM network where x_t denotes the input and h_t denotes the hidden state for t^{th} time step.

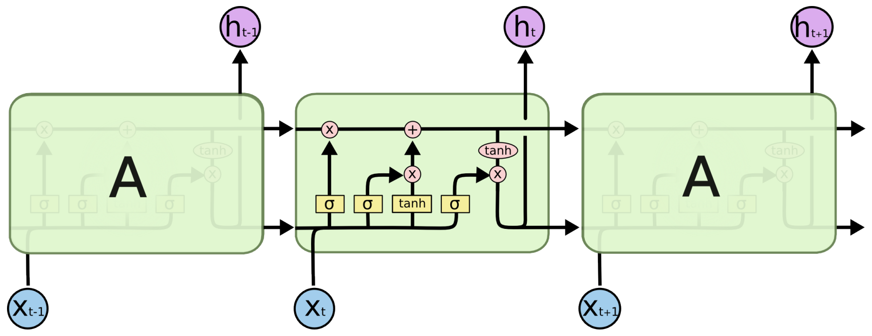
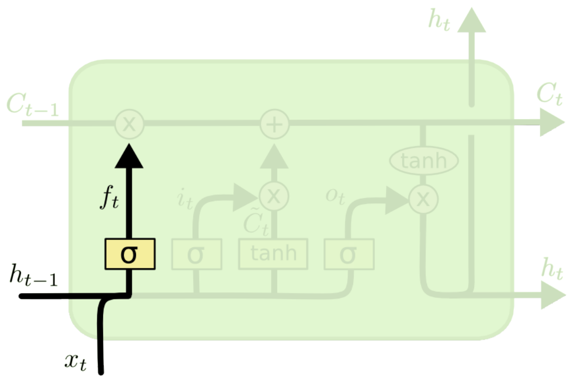


Figure 7.1: LSTM network [1]

The functionalities of the gates are described below:

1. **Forget gate**
   Forget gate (Figure 7.2) determines which information to throw away or to hold. Information from the previous hidden state is passed and information from the current input through the sigmoid function. Values range from 0 to 1. The closest to 0 means forgetting and the closest to 1 means holding off.



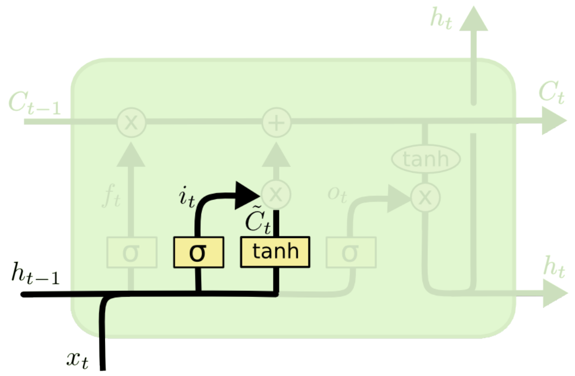
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Figure 7.2: Forget gate [1]

2. **Input gate**
   The input gate is used to the update cell state. First, the previous hidden state and current input are passed into a sigmoid activation function. This determines

[1]Image source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

which values are modified by converting the values between 0 and 1. 0 means insignificant and 1 denotes it is important. The hidden state and current input are also fed into the tanh function to suppress its values between -1 and 1 to help regulate the network. Then the tanh output is multiplied with the sigmoid output. The sigmoid output will determine which information is important to keep from the tanh output. This is illustrated in Figure 7.3.



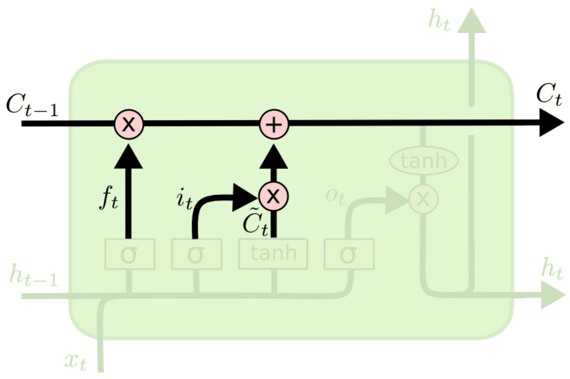
$$i_t = \sigma\left(W_i\cdot[h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C\cdot[h_{t-1}, x_t] \ + \ b_C)$$

Figure 7.3: Input gate [1]

3. **Cell State**
   Now we have enough information to calculate the cell state. We multiply the previous cell state with the forget vector. If the forget vector value is close to 0, there is a possibility of dropping values in the cell state. Then we do pointwise addition of the above resultant value with the output of the input gate. This gives us the value of the next cell state as shown in Figure 7.4.

4. **Output state**
   The output gate determines the value of the next hidden state. As the current output depends on previous hidden state (previous inputs), the previous hidden state and the current input is passed into the sigmoid function. Then the computed value of the next cell state is passed into tanh function and its resultant value is multiplied with the above output of sigmoid function to determine what information the next hidden state should retain. This gives us the value of the

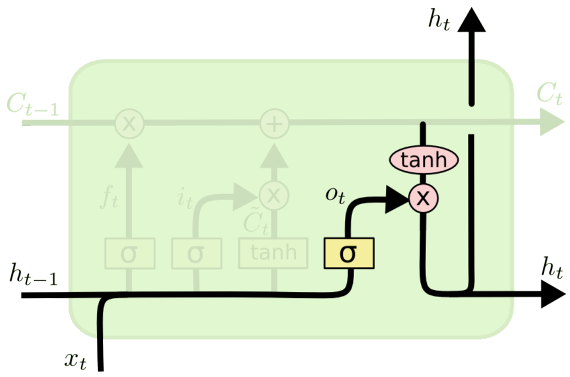
[1] Image source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 7.4: Cell State [1]

next hidden state which is carried to the next time step along with the new cell state (Figure 7.5).



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 7.5: Output gate [1]

7.2 Prediction using LSTM

In recent times, studies have used LSTM to predict travel times for public transport (33) (34). As we have total bus-journey times sequential data spanning from 6th November 2012 to 30th November 2012, we leverage LSTM to predict the future journey times based on past journey times sequence. This can also be helpful in predicting the future trends in bus travel times.

[1]Image source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

7.2.1 Model Architecture

The LSTM model used has one input layer, two dense LSTM layers and an output layer. It is a double stacked LSTM with the output from the first LSTM layer at each time step being fed to the second LSTM layer. The output of the second LSTM layer goes into a dense layer, which is a fully connected neural network. Finally, the dense layer consists of one node on which tanh activation is applied and gives the predicted journey time. An important model parameter – look_back which denotes the number of past journey times to use as input variables to predict the next journey time – is set to 15. The inputs are normalised before being fed into the network. In addition, dropout of 0.2 is applied to LSTM layers to prevent overfitting. The model structure is shown in Figure 7.6.

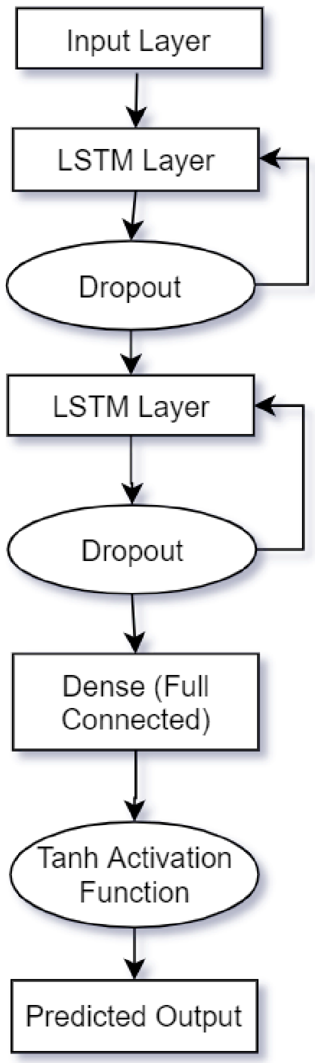


Figure 7.6: LSTM: Model Structure

7.2.2 Features Used

Unlike previous models, our univariate LSTM model takes in only one feature – past sequence of ‘Journey times’ as input. The LSTM model learns a function that maps a sequence of past journey times as input to an output observation. One major difference between ANN and LSTM is that during the training process the ANN assumes that the data samples are independent of each other whereas the LSTM assumes each sample is dependent on the previous samples.

7.2.3 Hyperparameter Tuning

Hyperparameter tuning for finding the optimal set of hyperparameters for the LSTM model is performed using *scikit-learn's* RandomizedSearchCV method. The range of value of hyperparameters searched for and the optimal values obtained are summarized in Table 7.1.

Table 7.1: Hyperparameters for LSTM

Hyperparameter	**Parameter values**	**Optimal value**
init_ mode	glorot_uniform, uniform	glorot_uniform
batches	10,20,30,50,90,128,512	50
epochs	10,30,50,200	200
lr	1e-2, 1e-3, 1e-4	1e-2
decay	1e-6,1e-9,0	1e-6
activation	ReLu, tanh, sigmoid	ReLu
dropout	0, 0.1, 0.2, 0.3	0.3
look_back	3,5,10,15,50,100	15

7.2.4 Training and Test Data

The model is trained on a training set consisting of journeys of bus route number 46 from 6th November 2012 to 25th November 2012. It is then tested to predict journey times for the same bus route from 26th November 2012 to 30th November 2012. The training set consists of 1609 journeys and the test set comprises 402 journeys.

The model is trained with a batch size of 50 samples for 200 iterations. During training, the model tries to minimize the loss function which is set to Mean Squared Error (MSE):

$$Loss = \frac{\sum_{i=1}^{N} (Y_{actual}[i] - Y_{predicted}[i])^2}{N} \tag{7.1}$$

ReLu activation functions are used in the LSTM layers. An Adam optimiser is used to update network weights with a learning rate of 0.01 and a decay of 1e-6 .

7.2.5 Results and Evaluation

Figure 7.7 represents the actual and the predicted total journey times for bus journeys of route 46 completed from 6^{th} November to 30^{th} November 2012 in chronological order. As mentioned earlier, the model, after being trained on the previous 1609 consecutive trips, predicts the journey times for the next 402 trips.

It can be observed that the model is able to predict the peaks and drops in journey times much better than the Linear Regression model and its performance is comparable to that of the ANN we discussed earlier. It reveals that the time taken to complete a future journey depends on the previous sequence of journeys completed. An MAE of 4.28 minutes for LSTM is an improvement of around 43% over the Linear Regression. Values of other evaluation metrics – MAPE of 4.7312 % and RMSE of 6.78 minutes are recorded for the test data.

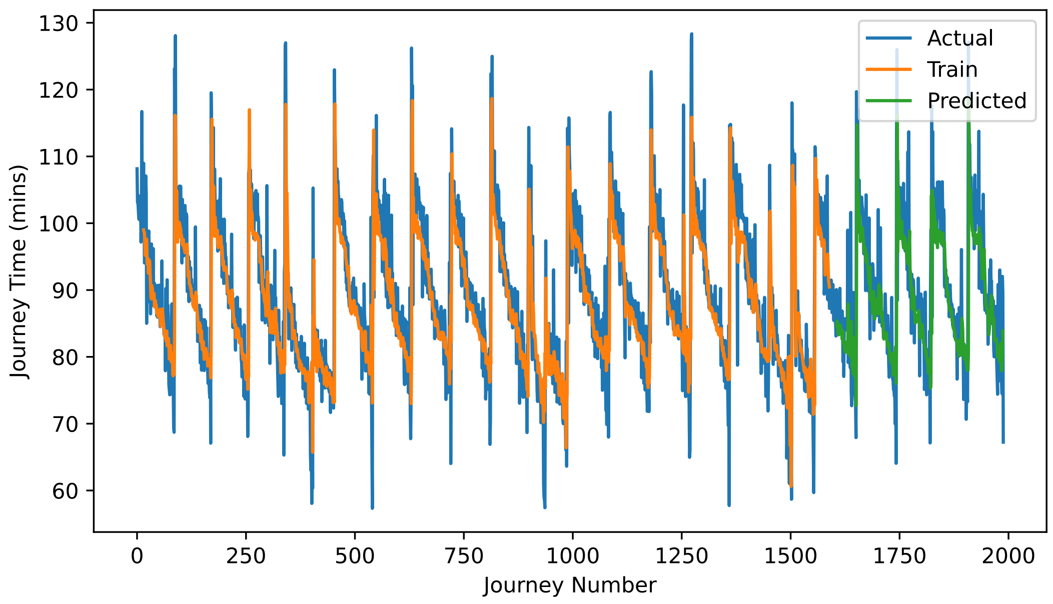


Figure 7.7: LSTM: Actual vs Predicted Journey Time and Future Trends for bus route 46

One key observation to note is that the LSTM model gives much lower prediction errors especially for journeys completed during the rush hours in comparison to Linear Regression and ANN . Even without taking traffic and time-related features such as ‘day of week’, ‘hour’, ‘delay’ as input, it is able to distinguish between peak hours and non peak hours, and also differentiate between weekdays and weekends for which journey times follow different patterns. It suggests that LSTM derives some intrinsic traffic information from the past journey times sequence, memorizes it, retains the information and uses it for future predictions.

Chapter 8

Journey Time Prediction of Route Segments

In the prior chapters, we developed various models to predict the journey time a bus takes to traverse the whole route. Besides this, it is also useful to predict bus journey times for segments of a route. A segment of a bus route is a section of route between any two bus stops. If we can predict how much time a bus takes to travel from Stop A to Stop B, it would provide bus arrival information to a passenger waiting at Stop B and potentially reduce waiting time for passengers at the stop. To accomplish this task, individual bus journeys for route 46 are extracted, cleaned and analysed from the dataset. The data is then fed into machine learning models to predict the travel time between stops for future bus trips.

8.1 Bus Arrival Time Prediction Using Predicted Route Segment Travel Time

Figure 8.1 shows a part of a route having three bus stops – A, B, C and two segments – S1, S2. Let the predicted travel times for the bus to traverse the segments S1, S2 are t1 and t2 respectively. Then to predict when the bus at Stop A will arrive at Stop C, we can just add the predicted segment travel times for S1 and S2 to the time at which the bus was located at Stop A.

$$\textit{Predicted arrival time at } C = \textit{Time at which bus located at } A + t1 + t2 \qquad (8.1)$$

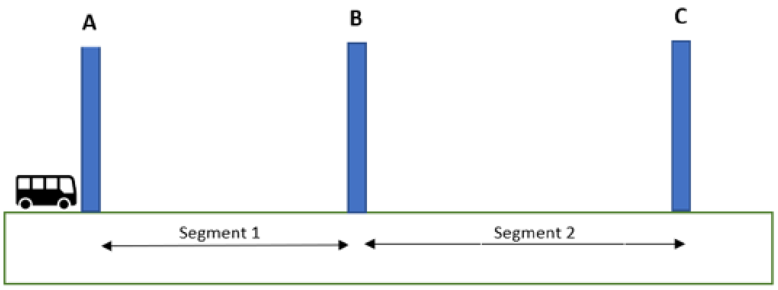


Figure 8.1: Bus Arrival Time

Ideally bus dwell times at Stop A and Stop B should also be accumulated to estimate the arrival time at Stop C. But in our study, we assume that the dwell times are included in the segment travel time, so we do not consider them explicitly.

In the following sections, we develop predictive models to estimate bus travel time between two adjacent stops. Once we have those predicted values, they can be used to calculate travel time for any route segment and can be used to derive the bus arrival time at a stop.

8.2 Cleaning and Extracting Bus Trips

Bus journey trips of route number 46 are considered for analysis. The following steps are performed for cleaning and pre-processing the data to obtain the required valid bus trips to be used in our models.

1. **Segregating individual bus trips**
   A bus trip is uniquely identified by a combination of Date, Line ID and Vehicle Journey ID features. A total of 1569 trips are identified and stored in a Python based dictionary data structure.

2. **Mapping observed GPS points to closest bus stop**
   For each GPS observation, the distance between the point of observation and each bus stop is calculated. If the distance is less than a threshold of 50 metres, the observation is mapped to the corresponding bus stop and it is assumed that at that particular instant the bus was at that particular stop. Simultaneously, the value of the feature 'At Stop' is set to 1.

3. **Determining most frequently marked bus stops**
   A stop is said to be a marked stop if a GPS observation was recorded at the time when the bus was present at that particular bus stop. In other words, for a marked stop, value of 'At Stop' should be equal to 1 for a GPS observation. From all the bus trips of route 46 considered, most frequently marked stops are calculated. Stop number and its count is stored in a hashed map.

4. **Trade-off between number of bus stops and total number of journeys**
   There is a trade-off between number of bus stops and total number of journeys. Not all bus trips would contain the same sequence of marked bus stops. For example, for one bus trip, say stops 1,3,4,5,8 are marked whereas for other two trips stops 1,5,6,7 and stops 1,2,4,5,8 are marked respectively. So we can observe, even if the count of stops 1 and 5 is three measured across all the three trips but the longest most frequent matching sequence 1,5,8 has count of two. Now if we consider only two stops 1 and 5, the most frequent matching sequence 1,5 has count of three . So it implies the more stops we consider in all trips the less the number of trips we obtain for analysis. Though it is beneficial to select the maximum number of stops for better prediction information, the maximum number of journeys data is also desirable for reliable prediction of travel times. Therefore, to maximize the number of stops in the matching sequence and to avoid losing out on journey information, the optimal number of stops to be considered is found to be 20 as shown in Figure 8.2.

   Figure 8.3 shows the route for Line ID 46 and the position of such stops marked on the map of Dublin city.

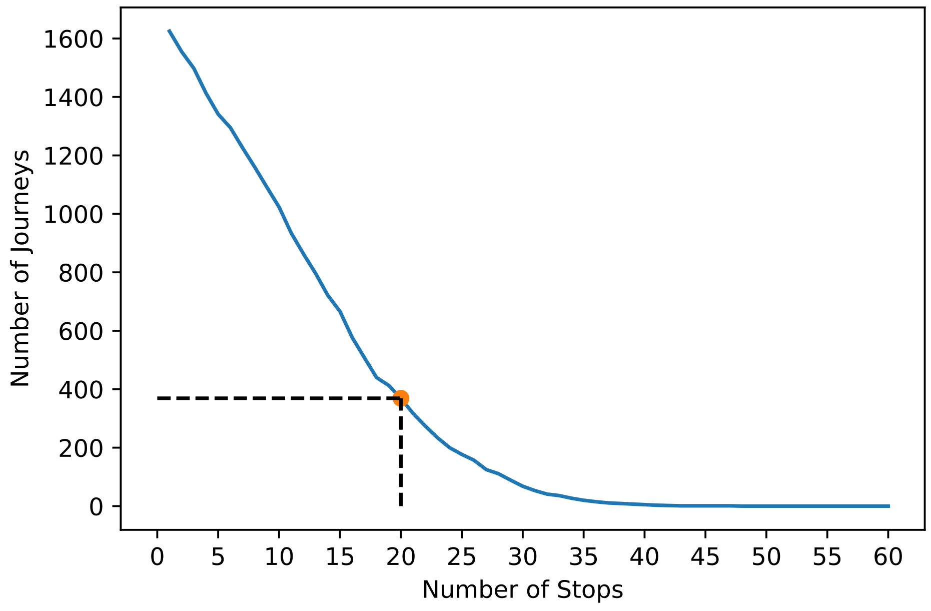


Figure 8.2: Number of Journeys vs Number of Stops

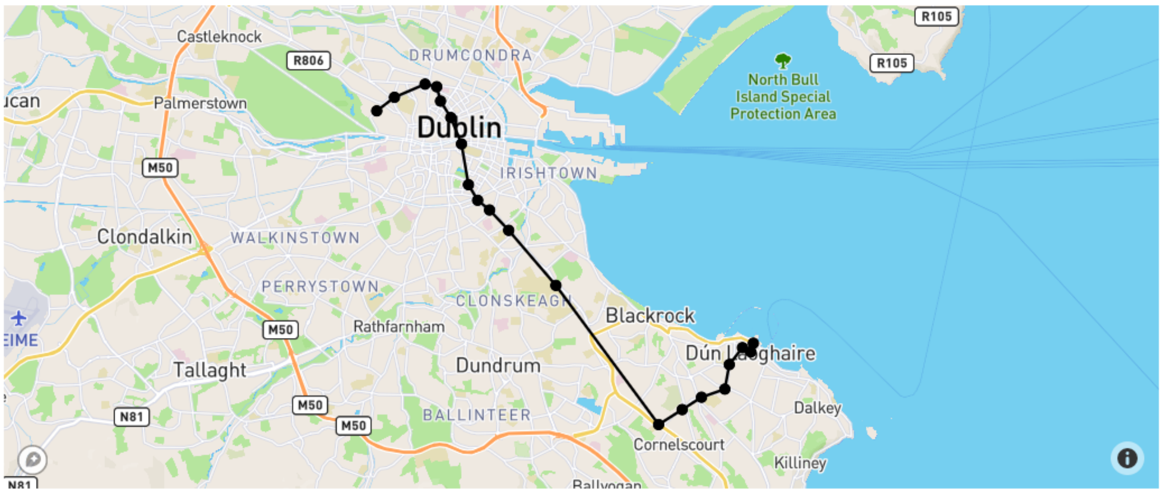


Figure 8.3: Bus Route 46 with 20 most frequent marked stops obtained after cleaning the trips. The road from Cornelscourt to Clonskeagh is a relatively fast urban dual carriageway, so not many GPS observations were recorded within 50 metres from the bus stops.

Stops information and distance between the stops in shown in Table 8.1

Table 8.1: Stops Information

Stop No	**Distance from previous stop (km)**	**Distance from City Centre (km)**
0	0	10.814
1	0.235	10.553
2	0.272	10.298
3	0.633933	10.345
4	0.715	10.737
5	0.703	10.460
6	0.605	10.416
7	0.800	10.385407
8	4.898693	5.513
9	2.066	3.524
10	0.820	2.788
11	0.408	2.469
12	0.510	2.000
13	1.175	0.844
14	0.800	0.391
15	0.544	0.762
16	0.419	1.100
17	0.328	1.393
18	0.9344	2.026
19	0.680	2.503

8.3 Predictive Models for Bus Route Segment Travel Time/Arrival Time

Three predictive models are studied and evaluated.

8.3.1 Historical Averaging Model

The Historical Averaging model predicts the travel time of a future trip from observed historical bus travel time data of past journeys completed in the same daily time period over different days. It is a simple model which does not take any explicit parameters as input. It assumes the traffic conditions in the current time to be same as in the past while making predictions. Analysing the previous trends in bus journey times, a day has been divided into four time periods: period-1 (06:00 — 10:00), period-2 (10:00 — 14:00), period-3 (14:00 – 18:00), period-4 (18:00 — 22:00). Now suppose we have N past bus trips data completed during period-1 such that $t^{i}_{a,b}$ represents the time taken by the bus to travel from stop a to b in the i^{th} trip where i= 1 to N, a < b and a,b $\in$ k. So for a current bus trip c operating during period-1, the historical averaging model predicts the time the bus will take to travel from stop a to stop b is given as:

$$t^{c}_{a,b} = \frac{\sum_{i=1}^{N} t^{i}_{a,b}}{N} \tag{8.2}$$

8.3.2 Linear Regression Model

Overview

The Linear Regression model (as discussed in section 6.2) which captures linear relationships between dependent and independent variables is used to predict the travel times between the stops. Moreover, they also measure the impact that independent variables have over the dependent variables.

Features Used

The model takes in input the features – ‘distance between the stops’, ‘distance of source stop from city centre’, ‘distance of destination stop from city centre’, ‘day of week’, ‘hour of day’ and ‘delay’. It outputs the time taken to travel between the source stop and the destination stop.

8.3.3 Gradient Boosting Model

Overview

Gradient Boosting (35) is a machine learning technique in which models are created and trained in a gradual, additive and sequential manner. Each model learns from the mistakes of previous models to minimize the error. The key principle behind the gradient boosting algorithm is to find a new sub-model to compensate for the residual error created by the previous submodel (36) as shown in Figure 8.4. The term gradient signifies that the residual errors are minimised using gradient descent such that the each new model takes a step in the direction that minimizes prediction error. For implementation purposes, we used the XGBoost (37) library of Python which is an efficient and scalable implementation of gradient boosting framework. Since the model keeps improving as it evolves, gradient boosting model show good results in non linear dependencies as well as making use of regularization parameters that help against overfitting.

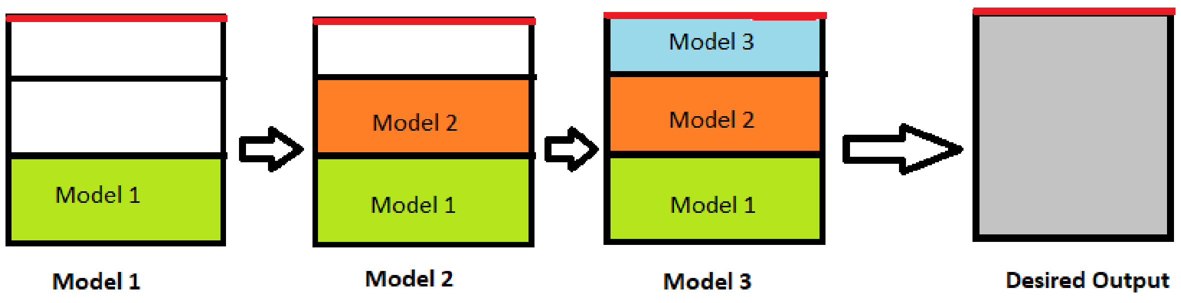


Figure 8.4: Schematic diagram of Gradient Boosting algorithm

Features Used

The same set of feature variables as used in the Linear Regression model are fed to the Gradient Boosting model to predict the bus travel times between adjacent stops.

8.4 Training and Test Data

By considering the 20 most frequently marked stops, a total of 459 bus trips having the same sequence of 20 marked stops are obtained for the period between 6^{th} November and 30^{th} November 2012. Of these, 369 trips took place from 6^{th} to 26^{th} November 2012. We considered it as our training set. The remaining 90 trips made between 27^{th} and 30^{th} November 2012 comprises our test set for which we make the predictions.

8.5 Results and Evaluation

The prediction of the three models and the actual arrival times are shown in Figure 8.5 which represents the predicted and the actual travel time between two adjacent stops for one of the test journeys.

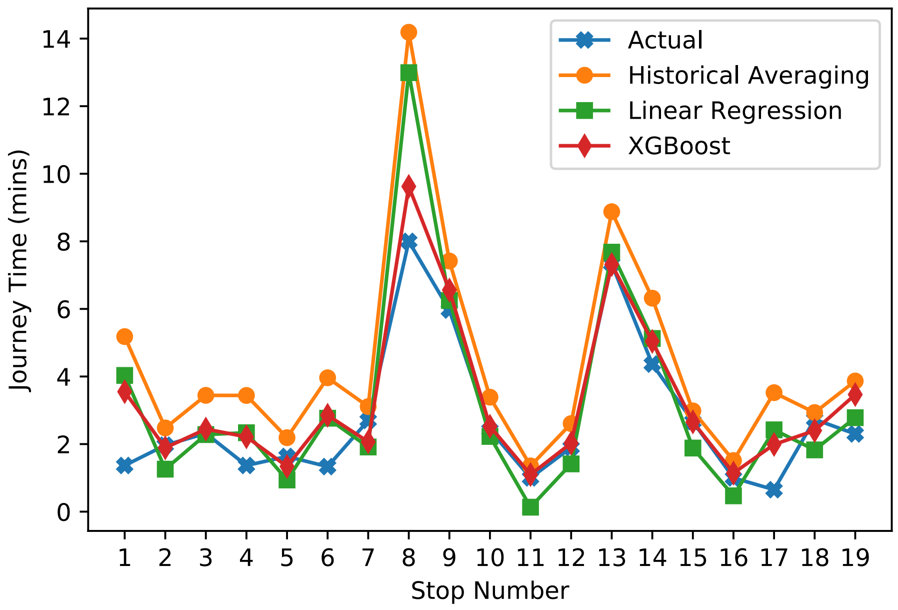


Figure 8.5: Predicted and Actual travel times from the previous stop

Figure 8.6 below represents the absolute deviation, that is, the absolute difference between the predicted and actual travel time between stops for the particular bus trip shown in Figure 8.5.

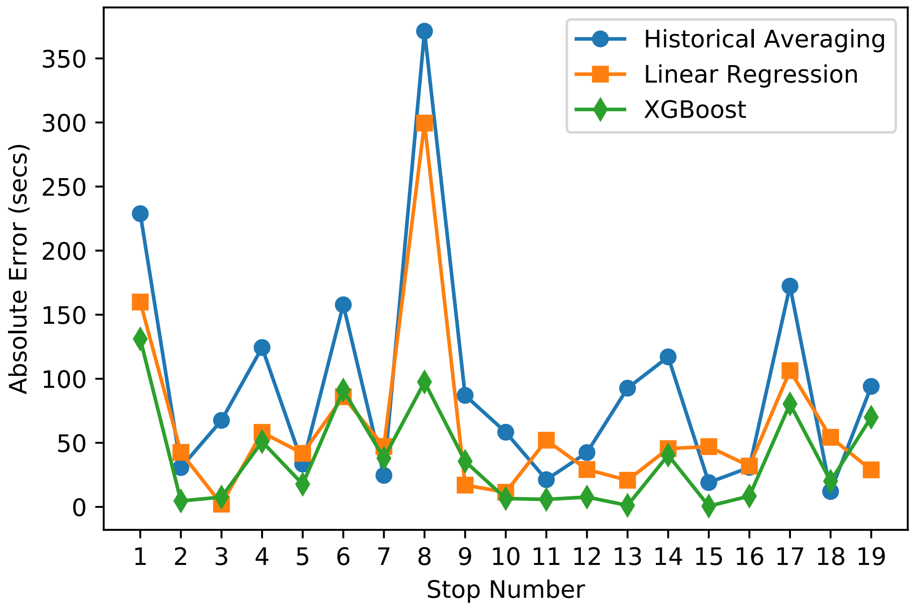


Figure 8.6: Absolute Deviation between Predicted and Actual travel times from the previous stop

It is evident from the above plots that the Gradient Boosting (XGBoost) gives better predictions, more closely approximates the actual travel times and outperforms the other two models. To reinforce this claim and further assess the performance of the models, evaluation metrics – MaxAE, MAE, RMSE are calculated for each model and listed in Table 8.2.

Table 8.2: Evaluation Metrics recorded for the models

Model	**MaxAE (mins)**	**MAE (mins)**	**RMSE (mins)**
Historical Averaging	11.28	1.02	2.37
Linear Regression	10.58	0.93	1.45
Gradient Boosting (XGBoost)	9.62	0.80	1.16

It can be observed from the Table 8.2 that Gradient Boosting has the lowest values for each of the metrics. On the other hand, Historical Averaging has the highest values for all metrics. However, we observe that for all three models the values of MAE are small. This is due to the fact that bus stops in Dublin are quite close to each other with most of them within the range of 300 metres (38). Hence, travel times between adjacent stops are low. Therefore, even a small MAE can cause large deviation from actual time and is undesirable. The observed value of MAE signifies that on average Historical Averaging predicts the bus travel time between two adjacent stops deviating 1.02 minutes from the actual time, Linear Regression predictions have a deviation of 0.90 minutes whereas Gradient Boosting predictions have a deviation of 0.84 minutes. Figure 8.7 shows the MAE values of the models for the test journeys.

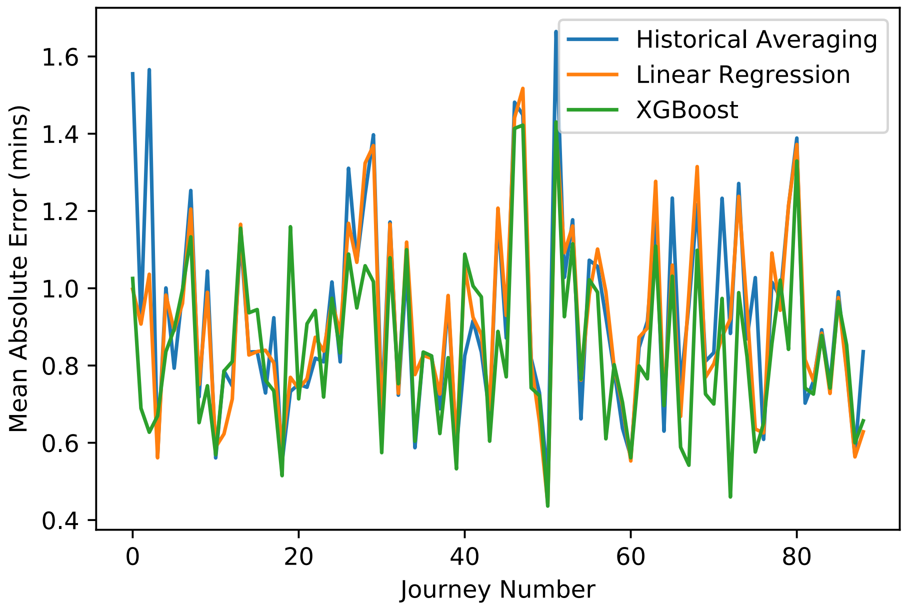


Figure 8.7: Mean Absolute Error (MAE) for test journeys

The table also reveals that the RMSE for the Gradient Boosting is least among the three models. This shows that the predictions by Gradient Boosting model are closer to the actual values than Historical Averaging and Linear Regression.

Figure 8.8 shows the RMSE values calculated for each of the test journeys by each of the three models discussed above.

It can be observed from the above plots that peaks of the graphs are mainly dominated by Historical Averaging and Linear Regression. It signifies that Gradient Boost-

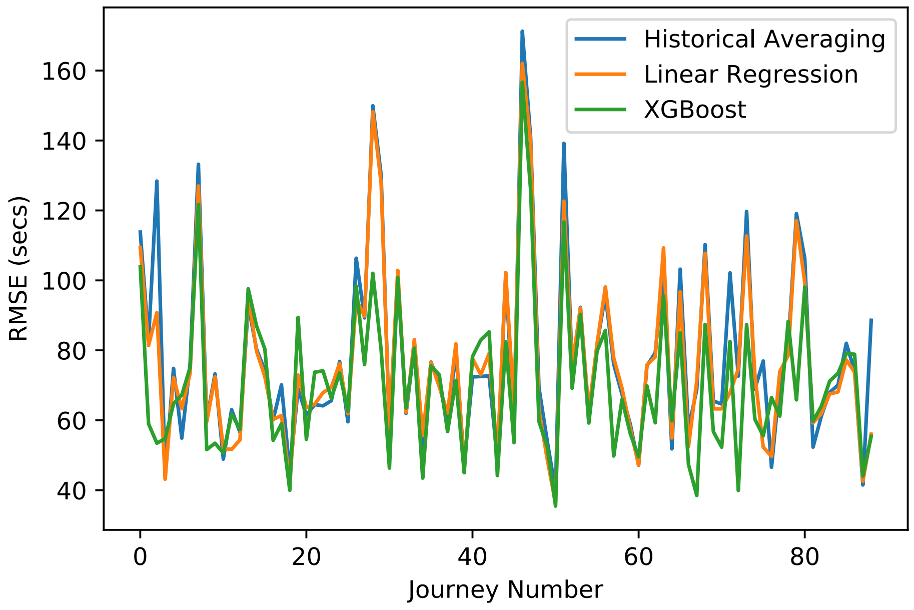


Figure 8.8: Root Mean Squared Error (RMSE) for test journeys

ing (XGBoost) performs better and approximates travel times between the stops best of all.

To further assess the performance and the robustness of the models in predicting the segment travel time, three different segments shown in Table 8.3 are considered.

Table 8.3: Segments

Segments	**Start Stop Number**	**End Stop Number**	**Segment length (km)**
Segment 1	0	7	3.96
Segment 2	0	9	10.927
Segment 3	0	19	17.54

The Mean Absolute Percentage Error (MAPE) is used as the measure of the models' performance. Figure 8.9 shows the MAPE values of the three models for the three segments. It can be observed that Historical Averaging has the highest MAPE, whereas

Gradient Boosting (XGBoost) has the lowest MAPEs for all three segments. The results also reveal a decreasing trend in MAPE with increase in segment length for all the models. Segment length decreases from segment 1 to segment 3. Since smaller segments has fewer bus stops, fewer traffic signals and fewer intersections, large variances in traffic, bus dwell times and signal delays can be expected. For example, there may be instances for short segments when buses don't get stuck at traffic signals and reach the stops ahead of their scheduled time. But there may be other instances when the bus has to wait at every signal thus arriving late at the stops. Moreover, for short segments, journey times are small, and even a small difference between actual times and predicted times will lead to high MAPE. However, with the increase in segment length, the average values of the above mentioned factors tend to be more stable, and better prediction accuracy can be obtained.

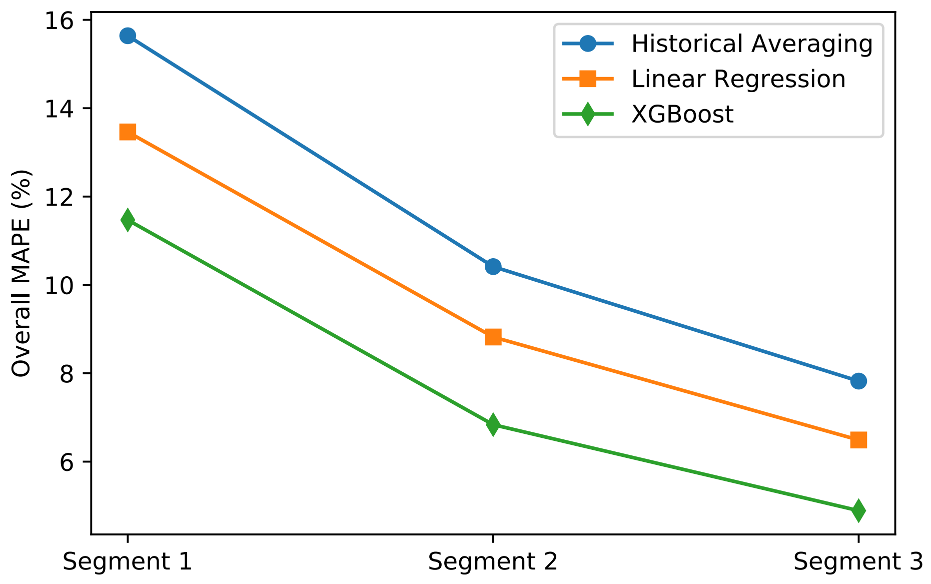


Figure 8.9: Overall MAPE for segments

A lower MAPE value usually suggests that the model is good in making predictions. However, a model with a small MAPE may occasionally yield a prediction with a large deviation. This is undesirable as it may predict the arrival time very different from the actual time. This might lead to passengers missing the buses. Therefore, it is critical to assess the robustness of the model to check if its maximum deviation is within a certain range. Hence, we use MaxMAPE (Maximum Absolute Percentage Error) which is defined as MaxMAPE = max{MAPE} of a segment, to measure the robustness of the models. As it can be seen from Figure 8.10, the maximum MAPE values of the Historical Averaging and Linear Regression models are greater than those of Gradient Boosting for the three segments we considered for analysis. Therefore, Gradient Boosting is found to be more robust than the other two models.

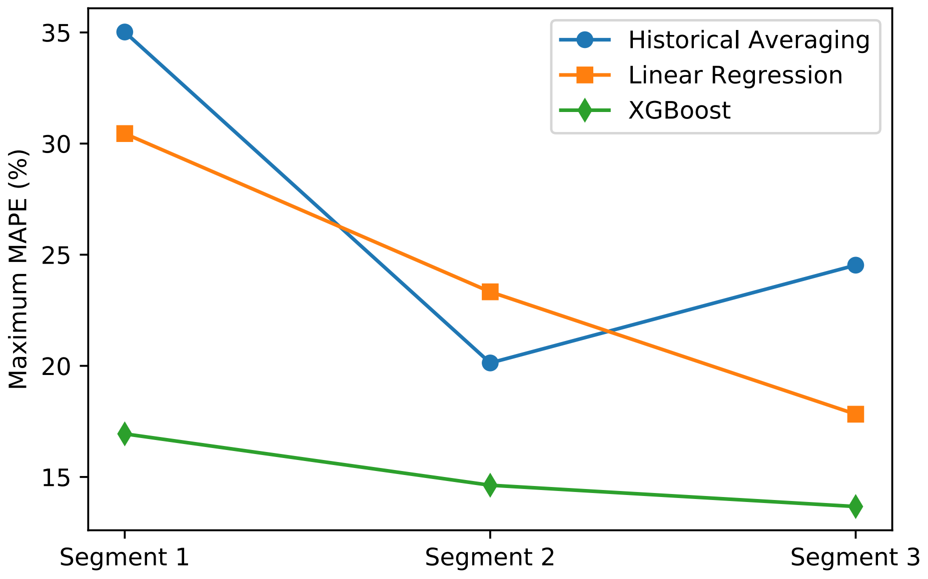


Figure 8.10: MaxMAPE for segments

A further point to be made is that, by looking at the modest performance of the Historical Averaging model, it can be said that bus journey times in Dublin appeared to be consistent over the observed time period back in 2012. The bus schedules appeared to be followed in a relatively disciplined and reliable way. Perhaps, this can be attributed to multiple factors such as – low traffic congestion, strict adherence to lane rules and a small population as compared to other metropolitan areas such as Toronto, Beijing, New York City on which similar studies haven been done.

Chapter 9

Conclusion and Future Work

In this research study we proposed and developed various models to predict overall bus journey times and arrival times using historical AVL/GPS data and prior routes information. The models developed are evaluated on a ground-truth dataset of Dublin buses and their performances are compared.

In the first half of our study, we developed Linear Regression and ANN models to estimate total time a bus takes to traverse the whole route. It is found that the ANN outperformed the Linear Regression model for all evaluation metrics. The RMSE observed for ANN was an improvement of around 28% over the Linear Regression model. This suggests there exists non-linear complex relationships between journey times and factors affecting it which could not be captured by the Linear Regression model.

We also proposed a univariate LSTM to predict the total bus journey times based on the time-tagged history of previous journeys. Experimental results reveals that the LSTM outperformed Linear Regression and its performance is comparable to that of the ANN. This is despite the fact that the LSTM did not take into account time-related features and traffic information for making the predictions. It is able to derive embedded traffic information from the past journey times sequence and thus is also able to differentiate between journeys which took place during rush hours and non rush hours and journeys during weekdays and weekends.

The second half of our study explored techniques and methods to predict bus arrival times at bus stops. From the experimental results obtained, it is found that

Gradient Boosting outperformed Historical Averaging and Linear Regression models on prediction accuracy and robustness. Its strategy of building and combining models in a sequential manner to minimize the errors made by the previous submodels helps to obtain better prediction accuracy. To further assess the performance and robustness of models, three route segments of varying lengths are considered. For each of the segments, Gradient Boosting gives the least MAPE and MaxMAPE of the three models and is found to be more robust in making predictions.

To conclude, results obtained from the study are promising and the proposed Artificial Neural Network and Gradient Boosting models can be used in Intelligent Transport Systems (ITS) to provide reliable real-time journey time and arrival time information to passengers based on historical bus' AVL/GPS data. It can help encourage people to use public transport and enable transport authorities to efficiently manage the available resources.

As part of future work, data regarding external factors such as traffic congestion, weather could be gathered and incorporated into the models for better predictions. One limitation of this study is that we assumed that the bus dwell times are included in the segment travel time, so they are not considered explicitly. To overcome this, some sophisticated techniques could be employed to derive the bus dwell times at stops from the data we used for our research. It could also give a measure of passenger load at the bus stops and might potentially improve bus arrival time predictions at downstream stops. Lastly, another interesting future work direction would be to develop a hybrid prediction model combining ANN, LSTM and Gradient Boosting. It would be interesting to see if the hybrid model could combine the individual strengths of the models to overcome their weaknesses and make better predictions.

Bibliography

[1] TomTom Traffic Index, 01 2020. URL `https://www.globenewswire.com/news-release/2020/01/29/1976528/0/en/TomTom-Traffic-Index-Global-Traffic-Congestion-Up-as-Bengaluru-takes-Crown-of-World-s-Most-Traffic-Congested-City.html`.

[2] Marcy Lowe, Bengu Aytekin, and Gary Gereffi. *Public transit buses: A green choice gets greener.* Center on Globalization Governance and Competitiveness, 2009.

[3] Jayakrishna Patnaik, Steven Chien, and Athanassios Bladikas. Estimation of bus arrival times using apc data. *Journal of public transportation*, 7(1):1, 2004.

[4] Ranhee 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 988–993. IEEE, 2004.

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

[6] Mei Chen, Jason Yaw, Steven I Chien, and Xiaobo Liu. Using automatic passenger counter data in bus arrival time prediction. *Journal of Advanced Transportation*, 41(3):267–283, 2007.

[7] Tao Liu, Jihui Ma, Wei Guan, Yue Song, and Hu Niu. Bus arrival time prediction based on the k-nearest neighbor method. In *2012 Fifth International Joint*

Conference on Computational Sciences and Optimization, pages 480–483. IEEE, 2012.

[8] Amer Shalaby and Ali Farhan. Bus travel time prediction model for dynamic operations control and passenger information systems. *Transportation Research Board*, 2, 2003.

[9] Steven I-Jy Chien, Yuqing Ding, and Chienhung Wei. Dynamic bus arrival time prediction with artificial neural networks. *Journal of transportation engineering*, 128(5):429–438, 2002.

[10] Abolhassan Halati, Henry Lieu, and Susan Walker. Corsim-corridor traffic simulation model. In *Traffic Congestion and Traffic Safety in the 21st Century: Challenges, Innovations, and OpportunitiesUrban Transportation Division, ASCE; Highway Division, ASCE; Federal Highway Administration, USDOT; and National Highway Traffic Safety Administration, USDOT.*, 1997.

[11] Peilan He, Guiyuan Jiang, Siew-Kei Lam, and Dehua Tang. Travel-time prediction of bus journey with multiple bus trips. *IEEE Transactions on Intelligent Transportation Systems*, 20(11):4192–4205, 2018.

[12] Kranthi Kumar Reddy, B Anil Kumar, and Lelitha Vanajakshi. Bus travel time prediction under high variability conditions. *Current Science*, pages 700–711, 2016.

[13] M Zaki, I Ashour, M Zorkany, and B Hesham. Online bus arrival time prediction using hybrid neural network and kalman filter techniques. *International Journal of Modern Engineering Research*, 3(4):2035–2041, 2013.

[14] Bin Yu, Zhong-Zhen Yang, Kang Chen, and Bo Yu. Hybrid model for prediction of bus arrival times at next station. *Journal of Advanced Transportation*, 44(3): 193–204, 2010.

[15] DW Seng, JW Peng, J Chen, and N Zheng. Bus arrival time prediction based on static and dynamic algorithms. *Advances in Transportation Studies*, 2015.

[16] Dublin Bus GPS Data. URL `https://data.smartdublin.ie/dataset/dublin_bus_sample/resource/26c468ec-8e19-4cd1-8ec4-1597b8a54fe2`.

[17] C Carl Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40, 1957.

[18] Transport for Ireland. Dublin Bus GTFS. URL `http://www.transportforireland.ie/transitData/google_transit_dublinbus.zip`.

[19] GTFS Static Overview. URL `https://developers.google.com/transit/gtfs`.

[20] MapBox. URL `https://www.mapbox.com/`.

[21] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[22] A. K. Jain, Jianchang Mao, and K. M. Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, 1996.

[23] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[24] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1): 35–62, 1998.

[25] Michael DelSole. What is One Hot Encoding and How to Do It, 04 2018. URL `https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179`.

[26] Jason Brownlee. A Gentle Introduction to the Rectified Linear Unit (ReLU), 01 2019. URL `https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/`.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] RandomizedSearchCV. URL `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html`.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[32] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[33] AA Agafonov and AS Yumaganov. Bus arrival time prediction using recurrent neural network with lstm architecture. *Optical Memory and Neural Networks*, 28 (3):222–230, 2019.

[34] Yanjie Duan, LV Yisheng, and Fei-Yue Wang. Travel time prediction with lstm neural network. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*, pages 1053–1058. IEEE, 2016.

[35] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[36] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518, 2000.

[37] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, and Yuan Tang. Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4, 2015.

[38] National Transport Authority. Core bus network report. URL `https://data.smartdublin.ie/dataset/dublin_bus_sample/resource/26c468ec-8e19-4cd1-8ec4-1597b8a54fe2`.