Research Practicum Project Report

Dublin Bus Travel Time Predictor

Ian Fuchs

A thesis submitted in part fulfilment of the degree of

MSc. in Computer Science (Conversion)

Group Number: 10

COMP 47360

UCD School of Computer Science
University College Dublin
October 14, 2018

Project Specification

Bus companies produce schedules which contain generic travel times. For example, in the Dublin Bus Schedule, the estimated travel time from Dun Laoghaire to the Phoenix Park is 61 minutes (http://dublinbus.ie/Your-Journey1/Timetables/All-Timetables/46a-1/). Of course, there are many variables which determine how long the actual journey will take. Traffic conditions which are affected by the time of day, day of the week, month of the year and the weather play an important role in determining how long the journey will take.

These factors along with the dynamic nature of the events on the road network make it difficult to efficiently plan trips on public transport modes which interact with other traffic. This project involves analysing historic Dublin Bus GPS data [1] and weather data [2] in order to create dynamic travel time estimates.

Based on data analysis of historic Dublin bus GPS data, a system which when presented with any bus route, departure time, day of the week, current weather condition, produces an accurate estimate of travel time for the complete route.

Users should be able to interact with the system via a web-based interface which is optimised for mobile devices.

When presented with any bus route, an origin stop and a destination stop, a time, a day of the week, current weather, the system should produce and display via the interface an accurate estimate of travel time for the selected journey.

Abstract

This report is an account of a project undertaken by a group of MSc conversion students in the summer of 2017, in which an attempt was made to develop a mobile-friendly web-based application which would return accurate journey time predictions for trips on individual Dublin Bus routes between any two given stops. These predictions were based on in-depth analysis of Dublin Bus GPS data collected over the course of 56 days in late 2012/early 2013, and Met Eireann weather data from the same time period. The team took the approach of modelling the bus data by treating road segments between consecutive stops as the basic unit and producing predictive models for each one. These were based on the travel time data accumulated from all buses traversing these segments, so as to capitalise on the additional information thus garnered to provide greater accuracy than focusing on individual bus routes alone would. The application was designed to be usable at home or on the go and to deliver predictions up to five days in advance of a planned bus trip, with arrival time predictions for each stop along the journey being viewable on a map within the web-page. However, while the web application worked well on a test set of models, and the modelling itself was moderately successful, the linking of the former with the full set of models caused unforeseen issues that have left the full application inadequately functional at present. We believe that the insights into public transport modelling and mobile application development we gained in the process and which are communicated in this report will nonetheless be worthy of the attention of anyone with an interest in these fields.

Acknowledgments

First of all, I wish to thank my team-mates for their contributions and support, particularly Emma Byrne, who did trojan work on the front-end flask application and database set-up, and Wen-Ting Chang, who I collaborated closely with in our work on the data analysis, preparation, and modelling, and who I would have been completely lost without.

I also wish to thank the project co-ordinators and demonstrators for their kindness and assistance throughout the project's duration: Gavin McCardle, Vivek Nallur, Hamed ZolghadriJahromi, and most especially Ellen Rushe, who spent a great deal of time giving us extremely useful feedback and guidance on our approach, and who promptly and in very helpful detail responded to several late-stage emails from myself about various aspects of the data modelling phase.

Table of Contents

Chapter 1: **Introduction**

This aim of this project was to produce a mobile-friendly web-based application which could return more accurate journey time predictions for trips on single Dublin Bus routes between any two given stops than that provided by the simple schedule that that company provides. This was to be completed via the analysis of historical Dublin Bus data from 2012-2013. Our original aim was to make this application work for buses running in the present day, but unfortunately the differences between the routes when the data was recorded and those currently in operation were too great, and precise data on those differences too difficult to acquire, to allow that ambition to be fulfilled, so we all had to settle for creating an app that functioned as if the routes from 2013 were still in operation at the present time.

In producing this application we attempted to go beyond the requirements in a number of ways. First of all we added a number of additional features to the web application we produced to enhance its functionality and attractiveness, including providing easy access to several different useful and relevant information sources and some social media interaction. Second, we added a map feature that showed the user the route they were to take, and additionally provided arrival times for every stop along the way. We were able to do this thanks to our innovative approach to modelling the network, which was by focusing on individual segments, or stretches of road between stops, and calculating the time taken to traverse each one.

The next section of this report is focused on background research, with a review of some of the literature we found relevant to our task and which helped influence the direction we decided to take the project. Following this we move onto a chapter on our project strategy, including a high-level overview of the problem and the solution we chose to apply to it, both at the data and user interface levels, and the tools and project management strategies we employed in this endeavour. The fourth chapter is the most extensive, and involves taking a look at the CRISP-DM process and how we applied it to our work on the Dublin Bus and Met Eireann weather data, from initial analysis through cleaning and transformation, to our eventual production of models for every individual segment in the Dublin Bus network. Next, we examine the front end and how this was configured, and our efforts to integrate the two components of the project into a working application. After this we focus on one specific and very taxing issue in working with the Dublin Bus GPS data, and that is the high density of outliers contained within it and our efforts to handle these. In the penultimate chapter the author gives his personal reflections on the process of working on the project and what he achieved during it's course, and we finish with some conclusions that can be drawn from this work, and how the efforts made may be expanded upon in the future.

Chapter 2: **Background Research**

Given the ubiquity of public bus networks in urban areas worldwide, and the potentially enormous benefits that a reliable travel duration and bus stop arrival time prediction system could provide to such networks, it's hardly surprising that a great wealth of literature on this topic exists. However, much of what we came across was not entirely relevant to the task and dataset that we had been provided with.

For one thing, much of the literature is concerned with real-time predictions incorporating real-time bus location data, a resource that was not available to us. We did toy with the idea of utilising the Dublin Bus Real-time Passenger Information API as part of our predictive data, but decided against this for reasons explained in Chapter 3.1. For another, many papers were concerned with predicting bus journey times across entire routes, whereas we had been tasked with creating a system to predict the travel time from any one stop along a route to another, which we felt required a different approach.

However, we did glean useful and relevant information from much of our research. We learned that statistical techniques like time-series and regression methods have long been popular in the field, but have been largely superseded in the eyes of many researchers by machine-learning models like Artificial Neural Networks (ANN) and Support Vector Machines (SVM) [3] [4]. The Kalman Filtering Model was another frequently mentioned, but this seemed only applicable when dealing with real-time data. In the 2010 El-Geneidy *et al*. [5] and Bejan *et al*. [6] papers, automatic vehicle location (AVL) technology was discussed; this technology uses GPS devices integrated into vehicles to collect probe data, "a sequence of coordinates recorded over time". Although we were not informed as to the methods used to collect the Dublin Bus data that we were to work with, the descriptions here of this technology and how you might expect the data produced by it to appear conformed with our initial analysis of the dataset.

The latter paper points out a few problems with data collected in such a way. First, it is sparse, being recorded on average every 20 to 30 seconds. Second, the precision of GPS measurement using such systems is not entirely accurate, estimated as being $\pm$30 metres. Finally, one can expect to see in such data "missing observations, meaning that there are times of silence when all or some buses are not transmitting their positions due to communication failure, temporary hardware malfunction, weather, and so on". Our later analysis of the data would confirm the presence of these issues. However, this paper otherwise was of little use to us, being focused on a single route run times.

We were quite excited to come across a pair of papers from the 15th International IEEE Conference on Intelligent Transportation Systems in 2012 [7] [8] which focused specifically on modelling Dublin Bus GPS data, though from over a year earlier than our own dataset. These confirmed that the data collection technology being used was AVL and that it was recorded on average at 20 second time intervals, with a standard deviation of 10 seconds, and also that the GPS spatial error had a zero mean, with a standard deviation of 20 metres.

However, both papers were focused on predicting journeys across a single entire route, with the former paper exploring the use of a modified k-Nearest Neighbours (kNN) algorithm and finding it performing well only for predicting running times (i.e. excluding dwell times at bus stops or time spent at traffic lights), and the latter comparing Linear Regression, kNN, and Kernel Regression, and finding the first two performing similarly and being outperformed by the third. The approach in both cases was to divide the journey into segments of equal length (126m in the first paper,

100m in the second), and consider only the timestamps at the joins of these points. While we liked the segmentation idea, we did not think this approach would be suitable for our purposes, given that bus stops are not so conveniently and evenly distributed.

The papers that most influenced our approach were one from China [9] and one from Hong Kong [10], which also took the approach of modelling by segment, but in this case basing the segments on the stretches of road between two consecutive stops on a route (or more exactly, multiple routes). These both looked only at a very small number of segments in detail, but we felt such an approach could be expanded and as such would be the most applicable to our problem of those we had studied.

To measure accuracy, both used the same suite of metrics: the mean absolute error (MAE), the mean absolute percentage error (MAPE), and the root mean square error (RMSE). The former study used a combination of historical and current data and experimented with pure ANN model, pure SVM model, pure Kalman model, and hybrid ANN-Kalman model and SVM-Kalman models, finding the last of these to give the best results. The latter however used only historical data, and compared SVM (with radial basis function (RBF) kernel), ANN, k-NN and Linear Regression models, with the accuracy of their predictions of bus arrival time being best to worst in that order. We decided to follow roughly their approach in the hope that an SVM model might work as well for us.

The last paper I want to mention here didn't actually form part of our background research - we only came across it a few weeks before the project deadline - but we wish it had. This is a study completed in 2014 which examined similar Dublin Bus data to ours via the same stop-to-stop segment approach outlined above, focusing on predictions for the 46A route [11]. Rather than SVM, this study experimented with a variety of regression tree models, which we ourselves had moved onto by the time we discovered it. As it wasn't background research, this paper will be further discussed in the section on modelling (4.2) rather than here.

Chapter 3: **Project Strategy**

3.1 Problem review and solutions considered

The purpose of this project was to build a web application, optimised for mobile devices, that would accurately predict the travel time between any two stops for any route on Dublin Bus regular services, with the predictions based on the analysis of historic Dublin Bus GPS data [1] and Met Eireann weather data [2]. Such an application is desirable for the following reasons:

1. Passenger journeys on Dublin Bus have increased from 112 million to 125 million from 2013 to 2016 [12], which indicates that these public transport services are in growing demand from both residents and visitors. This is a very positive development from an environmental and traffic management perspective, and one that should be further encouraged as much as possible. An accurate and reliable online method of calculating when a bus will arrive at a certain stop, and when it will reach its destination, could only be of benefit to this aim, especially in this age of almost ubiquitous internet-connected mobile devices.
2. The publicly available Dublin Bus route timetables do not contain arrival/departure times for each stop, but only the scheduled times that the routes depart from their starting stops.
3. The current Dublin Bus app only shows expected arrival times for each stop within the next hour; predicted arrival times beyond this time-frame are unavailable.
4. These predicted arrival times are unreliable in the experience of team members. Most of us have been in the position of waiting at a bus stop while the app counts the minutes down to a bus' arrival, only for it to hit 'due' and disappear from the app with no bus arriving at the physical stop location. Buses arriving when no bus is apparently due is also an occurrence we have experienced.
5. The only alternative bus travel time predictors available are Google Maps and the Transport for Ireland Journey Planner [13], but the travel time predictions for Dublin Bus provided by these seem to be based only on scheduled times per stop per route journey [14]. This information is apparently taken from the GTFS stop_times data [15], which is static and is not modified by any external factors.

3.1.1 Modelling the Bus Data

At first we considered the approach of modelling by full bus route, using distance and travel times from the starting point to the terminus of routes to make predictions. We felt it might be possible to follow a similar approach to that advocated in the pair of papers from the 15th International IEEE Conference on Intelligent Transportation Systems in 2012 [7] [8] mentioned in the background reading section and using distance measured via GPS data to slice the routes into journeys between stops as per the requirements. However we felt that a major problem with this would be that all the segments within a route would be treated as equal, whereas in reality travel times per km on some segments would likely be much slower than on others; for example, traffic tends to be heavier in the inner-city than in suburban areas.

The approach we settled on was to focus on these segments themselves rather than the routes. We divided the data into over 5,500 dataframes, each one representing an individual stretch of road between two consecutive stops on one or more routes. This can be visualised in 3.1, where the yellow markers denote stops and the pink arrows stop-to-stop segments to be individually modelled; the term SSID in the figure is the name we gave to the 'Stop-to-Stop' IDs used to identify each individual segment. These comprise eight digits, with the first four representing the departure StopID and the second four the arrival StopID, with leading zeroes added to any StopID of fewer than four digits (see 4.1 for more clarity on Dublin Bus StopIDs) The target feature of all our models was the travel time for any route that traversed the segment in question.

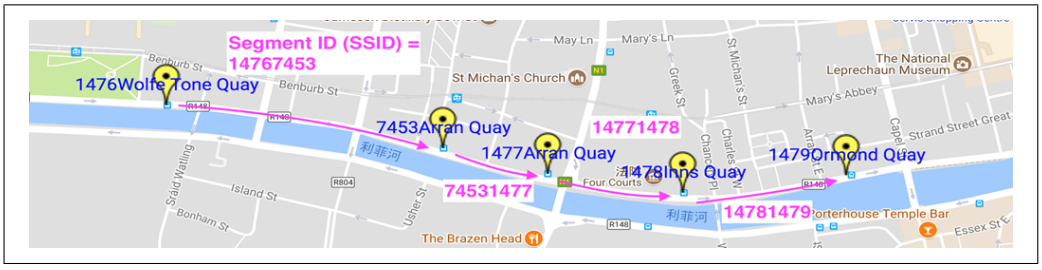


Figure 3.1: Example of stop-to-stop segmentation

Advantages we perceived of this approach included:

- An increase in the amount of data available for modelling over a focus on individual routes, as we would be using data from all the routes that traversed a given segment. This was the main advantage in our eyes, as data quantity is generally considered the primary factor influencing the accuracy of predictive modelling [9] [10].
- A reduction in the size of the dataset we had to manipulate, as all entries that existed between individual stops on a route could be discarded.
- A better ability to see in which areas of the city delays were occurring on routes.
- A greater ability to recognise which parts of the dataset were most problematic.
- More flexibility in filtering out problematic aspects of the dataset, as granted by the higher quantity of data being analysed, at least on segments traversed by more than one route.
- Easier to adapt the model for use with altered bus routes, as defunct segments could be simply 'plugged out' and new ones 'plugged in' (see below).

Initially we believed that it was necessary to make the application functional with respect to present-day Dublin Bus routes, and wished if possible to incorporate Dublin Bus real-time data into our calculations. It soon became apparent however that due to the age of the data, there were very large differences between the routes and stops as they existed then and in Dublin at present. Part of our justification early on for taking the approach of modelling by stop-to-stop segment was that we hoped that this would make it easier to adapt our application to predict for current routes, since we would be able to model all segments that still existed and use the static journey-time data from the GTFS data to fill in for parts that had changed. If we had modelled by route, such adaptation would have been far more difficult.

We spent a substantial amount of time over the first half of the project trying to figure out the differences between routes and stop locations then and now, but found this to be very difficult as no clear information was available and the data was not easy to work with. We tried contacted the National Transport Authority (NTA) for a list of route stop changes and physical stop location changes since the data was collected, but were redirected back to publicly available GTFS and

NaPTAN data we had already looked at and which did not resolve our issues. When we discovered around the midpoint of the project that making the app work for present day routes was not a requirement, we abandoned these efforts.

3.1.2 Incorporating the Weather Data

The weather data was taken from all three Dublin weather stations: Casement to the west of the city centre, Phoenix Park in central North-West, and Dublin Airport to the North of the city centre; not all stations provided data for the same features. The provided bus data was from the periods of 6th-30th November 2012 and 1st-31st January 2013, so the data downloaded from Met Eireann (which gave hourly weather information over the course of several years) was trimmed down to fit these dates. From the available features, we decided to only go with rain (as we felt people may be more likely to take the bus when it's wet) and wind-speed (both because people are less likely to cycle in windy weather and high wind speeds are generally indicative of unpleasant weather). We would have liked to include temperature, but the fact that all the data came from winter and the maximum temperature was in the mid-teens meant that this was not feasible for a year-round input feature.

One major issue with the weather data was that the stations it was derived from did not represent the entire geographical area of the Dublin Bus network. As shown in figure 3.2 below, there are no weather stations near the entire south-east of the network, which is a particular problem for the rain feature, as this can tend to be quite localised. In addition, there were no wind-speed measurements for Phoenix Park. Since omitting these features was not an option given they were part of the requirements, the best solution we could come up with was to take the average of all relevant values from each station for each hour as our descriptive features for weather, and assume that these would have a low importance ranking in areas distant from where the measurements were taken.

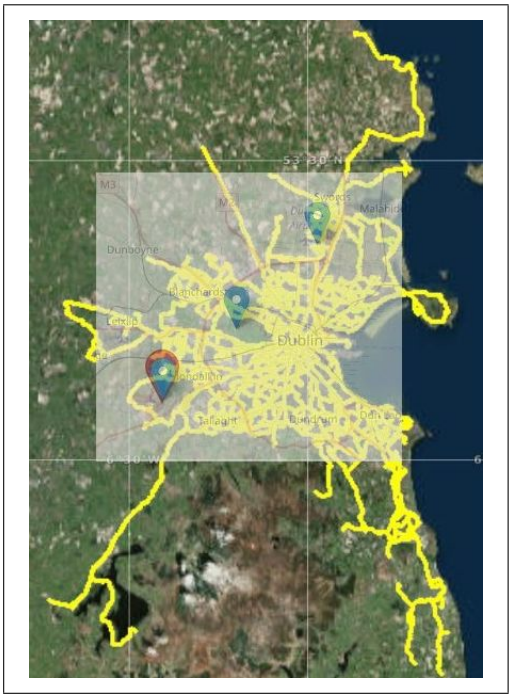


Figure 3.2: Relative positions of the three Dublin weather stations, superimposed on a map of the Dublin Bus Network. From top right to bottom left: Dublin Airport, Phoenix Park, Casement.

3.1.3 Creating the application

We chose to use the OpenWeatherMap API for weather input as it is free (or free enough for our purposes) and we had previous experience in using it on another project. The question of just how far ahead we could predict for was answered for us by the limitations of this API, which is restricted to five day forecasts; we briefly considered incorporating a manual weather input but quickly dismissed this as a pointless novelty. Since we were allowing bus arrival and journey time predictions up to five days in advance, we decided that it would be best to have an interface for the application that was accessible and user-friendly both at home and on the go. Therefore we decided to create an interface that would well on both wide desktop screens and narrow mobile devices.

Due to issues with the reliability of route details in the Dublin Bus dataset (covered in more detail in 4.1.2), we decided to use data from the NTA GTFS dataset for Dublin Bus [15] in the database accessible to the application and used to populate the drop-down menus on the input screen and process the user input. We aimed to match up the location of StopIDs in this dataset with those in the Dublin Bus GPS one, and thus link the models created using the latter with the user input defined by the former.

To calculate bus arrival times and total travelling times, we devised a system as illustrated in figure 3.3 below. The user inputs the desired time and date at which they wish to get a bus on a specified route from a specified stop, and where along the route they would then like to disembark. The models for each segment on the route from that stop back to the start of it are used to predict the amount of time it would take to travel along each one, and the total is subtracted from the desired time of departure. The official timetable for the route in question is then consulted in the database, and the earliest departing bus after the calculated time is selected. The predicted travel times for each segment from the starting point back to the requested departure stop are summed to calculate the bus arrival time at that stop, and then the models for each subsequent segment to the intended destination are used to predict the total journey time.

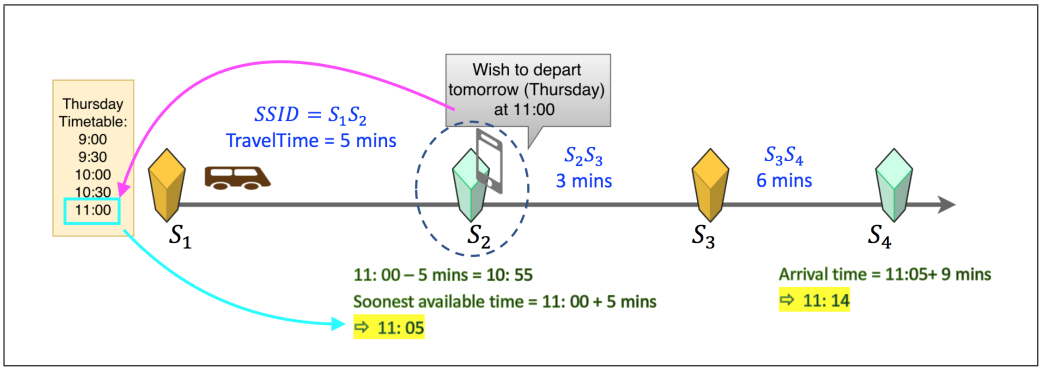


Figure 3.3: Illustration of how time of arrival and of journey are predicted.

3.2 Technologies and tools used

Python 3 is the language that most members of the group have had most experience with, and so we utilised this for the bulk of our work on this project, with the exception of the design and implementation of the web interface itself, which of course required the use of W3C suite of languages, including HTML, CSS, and JavaScript.

For our work on the analysis and manipulation of the supplied data, we generally used the Python

Pandas data analysis library [16] and related tools within the Jupyter Notebook environment [17], though some algorithms were produced outside this environment. For modelling the data, we used the Scikit-learn machine-learning library [18] within Jupyter Notebooks, though we also made some use of the Statsmodels module [19] for preliminary linear regression analysis. For storing data, we used CSV files, the in-built pickle module, and the more efficient Feather module [20], and to calculate distances between the GPS co-ordinates of data elements, we used the GeoPy module [21].

For the work on the application itself, in addition to the above-mentioned W3C suite of languages, we utilised the Flask micro web development framework [22]. This is Python-based and we had all had experience of using it during previous course-work and agreed that it was fit for our purposes. Part of the project specification was to make the web-based interface optimised for mobile devices, so we used the popular Bootstrap framework [23] to achieve this, which again some of us had had experience using. We also used the jQuery JavaScript library [24] in programming the functionality of the website. We used MySQL to set up and manage our database of routes and timetables, etc, and the python module SQLAlchemy to access it from the application.

3.3 Project Management

We agreed to follow an agile methodology for this project, as we had already been trained in this and knew what was required. We met in UCD at least twice weekly for face-to-face discussions and to work on code together and also had regular Skype meetings. Probably our main channel of communication was Slack however, where we regularly shared resources, files, and updates on our progress, and requested help from each other when having difficulties; Appendix A consists of a screen-shot of the Slack analytics from our project group demonstrating the extensive use we made of this interface. We also used GitHub to share files and assemble our application, though we didn't start making regular use of this until the second month of the time-frame, and formed a WhatsApp group for urgent communications.

While we conceived aims for each sprint, these were generally rather loose and often not met for the data analysis, preparation, and modelling side of the project, though the application design and implementation side was more successful. This was partly due to the fact that as we progressed towards the data preparation phase, we regularly encountered new, often severe data quality issues which necessitated doubling back to the data understanding and cleaning phase, and delayed progress substantially. The rather complex approach to the modelling that we had settled upon required close and careful analysis; we could probably have progressed more quickly had we decided to model by route, but we were quite convinced that the extra effort would be worthwhile in the end. In addition, our initial focus on making the application work for present-day routes also wasted a lot of time that should have been spent working with the data that we had.

Chapter 4: **Data Analysis and Modelling**

For the predictive data analytics side of the project, we tried to stick to the Cross Industry Standard Process for Data Mining (CRISP-DM) [25, pp.52-54] [26], which is comprised of six non-linear phases:

1. **Business Understanding:** Understanding the business problem being addressed and deciding whether and how a data analytics project can assist in solving it.
2. **Data Understanding:** Understanding what Data is available to assess and conducting initial assessment of it, including describing, exploring, and verifying it.
3. **Data Preparation:** Selecting which elements are to be used, cleaning the data, and engineering any required features.
4. **Modelling:** Creating predictive machine learning models based on the data.
5. **Evaluation:** Determining how effective models are at what they've been designed to do.
6. **Deployment:** Utilising the models for their intended purpose.

Number 1 has already been addressed in section 3.1 in this report. Now we shall look at points 2 and 3, and 4 and 5 together. We examine these phases in their pairs because due to severe data quality issues, we found that we moved back and forth between no. 2 and no. 3 several times, and because the modelling and evaluation phases were carried out concurrently. The final point shall be covered in chapter (5), as it more properly belongs in this context of discussion of the overall application.

4.1 Data understanding & preparation

The Dublin Bus historical GPS data we worked on initially comprised 56 CSV files, each representing an full day's worth of data from the entire fleet of operational Dublin Bus vehicles. The time periods covered were from the 6th to the 30th of November 2012 and the 1st to the 31st of January 2013. When put together, the total size of the CSV files reached over 8GB and over 8 million rows. Obviously, since we needed to work with the data as a whole at first (as we were not looking at individual routes or dates) before breaking it down by segment, we needed to do as much cleaning as possible without losing any valuable information.

Getting rid of as many unnecessary features and rows as possible was the main focus, but an important preliminary step was to reduce the amount of RAM consumed while working on the dataset as it was practically unusable on most of our machines as it was. Because the files originally came in CSV format, Panda's default datatypes were used, and these are far from optimised. Since the rough order of efficiency is numerical $>$ categorical $>$ object, we initially turned all numerical data into ints or floats, even when they would be more properly classified as categorical, and reduced the the size from 64-bit to 32-bit where we could be sure this would not result in any loss of detail. All remaining objects were converted into categories; thankfully

none had a wide enough range of values for this to be unsuitable. After this, the dataframe was saved in the recently developed feather format, which uses the Apache Arrow columnar memory specification to serialise the data and is thus extremely efficient; it is not designed for long-term storage and may be deprecated in the not too distant future, but it was fit for our purposes [20].

After this, we launched into the analysis and cleaning, an aspect of the project which was extremely time-consuming and difficult. This was due to multiple data quality issues, which will be outlined in the first sections below. Because of this, it took us a long time to get to a stage where we were ready to implement our planned SSID datasets and begin the modelling phase, which was moderately successful, but not quite as much as we had hoped; details on this comprise the second part of this chapter.

4.1.1 Initial Analysis of the Data

The given Dublin Bus GPS dataset consisted of fifteen features. Below I will go through them in two groups: those that were of some value in our analysis, and those that were not.

Valuable features:

- **JourneyPatternID:** This feature consisted of eight characters. The first four, a combination of numbers and sometimes letters, indicated which route (as displayed on the front of a bus) the row represented, with leading zeroes to fill any unused space. The fifth character was binary, 0 or 1, and symbolised which direction of the route the bus was travelling in. The final three consisted of two zeroes and another number. Where this was a one, it generally represented the standard route for that route number, and where it was another number, indicated some variation on that route (e.g. an alternative starting point or finishing point), These alternatives to the standard route were sometimes but not always shorter. This feature was crucial until the final stages.
- **StopID:** This feature indicated which bus stop the vehicle was either at, or had most recently departed. This feature was of course necessary for dividing the dataset by segment.
- **Timestamp:** This feature was in epoch format and so needed to be converted into datetime for analysis. As it indicated the exact time each instance of probe data was transmitted, it was required for producing our eventual intended target feature of Travel Time.
- **TimeFrame:** At first we thought this feature would be disposable as it seemed to duplicate the date value in Timestamp, but it represented the 'working day' for Dublin Bus, which for many routes ran after midnight, so we needed to keep it to identify when in the system each record was from.
- **Lon/Lat:** This pair of features represented the GPS co-ordinates of the vehicle when the probe data was transmitted. It was required for measuring distances and matching up StopIDs in this dataset with the GTFS one (as discussed later in this chapter) so was kept until this work was done.
- **VehicleJourneyID:** This feature represented 'a given run on the journey pattern' [1] *i.e.* a particular route travelling at a particular time and day. This feature was of little value on its own, but a combination of this and timeframe was sufficient to identify a unique journey (though JourneyPatternID was required to know anything about which route was in question).

Valueless features:

- **Direction:** This was constant and direction was already represented within JourneyPatternID, so this feature was immediately dropped.
- **LineID:** This feature purported to represent the route but came in various datatypes (int, float, object) and was often erroneous; we found that this information (number of bus route) was more accurately represented in JourneyPatternID so it was dropped fairly early on.
- **BlockID:** This refers to 'the sequence of trips made by a vehicle in the course of one day of operations, including both revenue and non-revenue trips' [27], and so was irrelevant to our purposes.
- **Operator:** This feature represents, as best we can tell, which of eight depots a vehicle was attached to at a given time. This was unlikely to have any predictive value and anyway was not reproducible as a parameter so was dropped.
- **Delay:** This feature was a cumulative measurement from the start of the route - positive numbers were the amount of seconds a bus was behind schedule and negative values how much it was ahead. There was an argument for using this to cut outliers but we didn't know what an appropriate cut-off point would be and it was otherwise not applicable to the model we settled on (as we focused only on individual segments), so it was dropped.
- **Congestion:** A binary feature true for little over 1% of entries. Again there was in retrospect an argument that we could have used it to identify outliers, but we had no way to determine how it was measured, so we therefore decided that since it was not reproducible as a parameter, it had no value in our analysis.
- **VehicleID:** This refers to the identity of an individual bus being driven. We kept this for the initial analysis of data, but later dropped it as we couldn't see how it could have any predictive value.
- **AtStop:** We kept this feature for a long time as it is a binary feature describing whether a vehicle is actually stopped at a bus stop or not, which was theoretically certainly useful. However, we came across several instances of buses at stops (recognised as such via GPS) with this set to zero, especially at the start of routes but also at other points and sometimes throughout an entire journey. Discussions with bus drivers who were working at the time this data was collected revealed that not all vehicles in the fleet then had fully automatic vehicle location systems installed; on some the driver needed to press a button to signal passing from one stage to the next, and failure to do so at the right times would apparently cause the real-time data prediction system to fail. We suspect this mixture of automatic and manual data collection is at the heart of many of the data quality issues we faced during our analysis. We eventually had to dispose of this feature as it was so unreliable and instead use GPS data to determine whether or not a bus was at a stop (or close enough).

4.1.2 Data Quality Problems and Solutions

One major issue we faced was ascertaining which StopIDs belonged to which route; there were often time gaps of much greater than the average of twenty seconds where multiple stops seemed to have been skipped, but also normal length ones which seemed to skip stops too, as we found that journeys with the same JourneyPatternID often contained different numbers of unique StopIDs. This wouldn't have really been a problem if we were looking at a single route, but it certainly was when we needed to know the correct sequence for all of them and their minor variations (as indicated by the last digit on the JourneyPatternIDs).

It wasn't possible for us to work this out by looking at all the StopIDs that a single JourneyPatternID contained, as we found a number of instances where StopIDs appeared in routes that they didn't seem to belong to or in the wrong direction of routes that they did. Also, the GPS co-ordinates for individual StopIDs sometimes varied from the average to the degree that they couldn't possibly be referring to the correct stop. And then there were instances where buses actually went backwards to a previous stop before resuming a 'forward motion'.

To deal with this, we used the GTFS data from Dublin Bus to ascertain which StopIDs belonged to which route, and then matched them to routes ('trip_ids') in the GTFS data under the assumption that this would be more reliable. There were some mismatches between the sets of data though, with some StopIDs and JourneyPatternIDs existing in one but not the other - we tended to side with the GTFS data and drop rows with StopIDs or JourneyPatternIDs that didn't conform to it.

We did try to research possible route changes via the Dublin Bus *General News Archive* website [28] but for some reason the archive, generally containing several updates a month, is almost blank between November 2012 and March 2013. I did find one community website listing some route changes during November 2012 [29] and we deleted these defunct routes, but we have no idea what other changes were happening. We thought this use of the GTFS data was the best approach available to us for cleaning up the StopID problems, but it's possible that parts of this are erroneous too and that this (or errors we may have made in manipulating and combining the two sets of data) contributed to the problems our application later encountered.

Some additional data quality issues not previously mentioned that we encountered are listed below:

- Over 15% of rows in the total dataset were null for JourneyPatternID and/or StopID (this happened mostly in conjunction). These were dropped as analysis demonstrated these were not passenger journeys but buses travelling between depots for later use in passenger journeys or idling at a stop between journeys.
- There were rows containing VehicleJourneyID + TimeFrame combinations which corresponded to more than a single JourneyPatternID. These were dropped as unreliable, as a VehicleJourneyID by definition should not correspond to more than one in a single TimeFrame.
- Rows corresponding to unique bus journeys (VehicleJourneyID + JourneyPatternID + Timeframe combination) with fewer than 5 unique StopIDs were dropped as this is too short for a real bus journey and often corresponded to situations like the one illustrated in figure 4.1 below, where a bus is idling between stops. This cut out over 5 million rows.

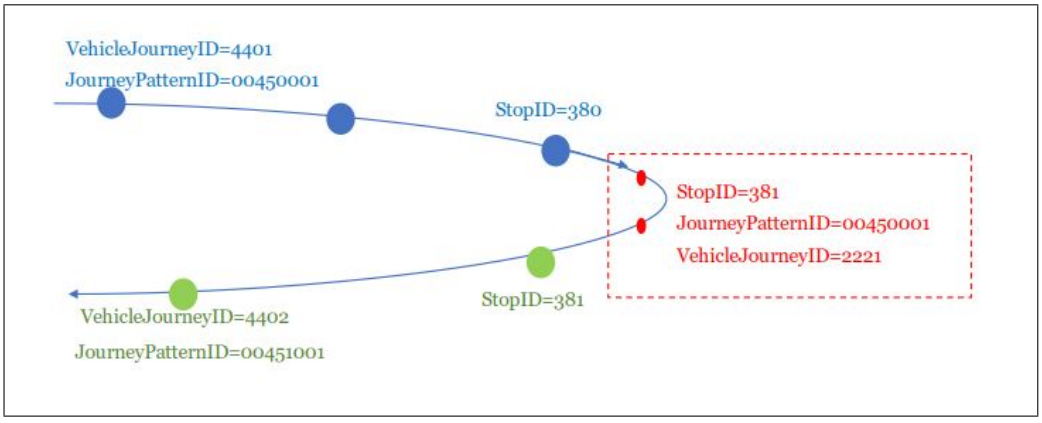


Figure 4.1: Static VehicleJourneyID between actual journeys

4.1.3 Data Transformation

Implementing the SSID feature involved matching up appropriate pairs of consecutive StopIDs on a route via the GTFS data and calculating the time between them. This was simple enough when the bus stopped at both stops of the segment (now calculated via GPS data rather than the AtStop feature), but of course it was often less simple than that. As a general rule, we took only the first instance of a StopID at a stop into account, unless it was the start of a route, where we took the last. Where one or neither of the buses in a journey stopped at a stop, we took the first instance of probe data after a stop had been passed as the start of the segment. The exception was when the first instance of that StopID was further from the referenced stop than the last instance of the previous StopID, in which case we relabelled that last instance of the previous StopID as the current StopID. Once this was complete, all other rows were dropped.

There were problems with this approach. Besides the inevitable inaccuracy of taking data instances not at a stop as being the start of a segment (which we hoped would work itself out over the course of the body of data), there are also the following scenarios to consider, illustrated in figure 4.2 below. In Case 1, the bus is going fast enough to register with first StopID A and then StopID C, skipping StopID B. We ignored such instances, which means we may be overstating the average length of time a bus takes to traverse a segment; perhaps in these circumstances we should have divided the time in half instead. In Case 2, the bus barely moves between the two records at StopID A, but the second record is closer to StopID B meaning that the time for SSID_A-B becomes whatever time was measured between the two instances; we have seen Travel Time measurements of as little as two seconds as a result of this scenario.

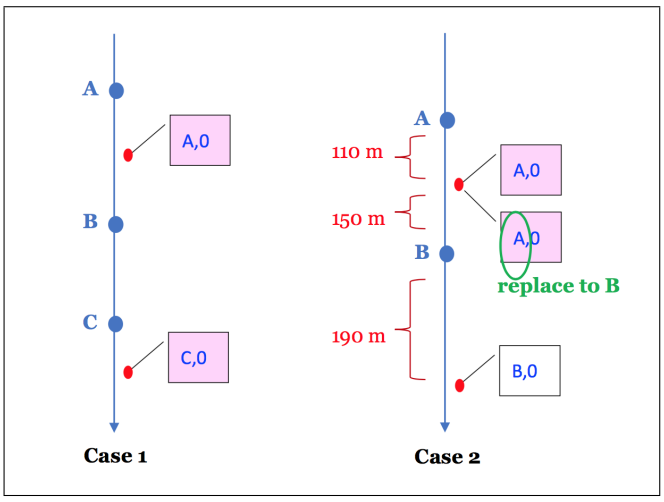


Figure 4.2: Examples of problems with the SSID TravelTime measurement algorithm

Once we had our SSIDs created, we were able to produce CSV files containing only rows from these segments to work on. After some work on these, none of the original Dublin Bus data features were left in the final Analytics Base Table, as can be seen in figure 4.3 on the next page. The newly engineered features, all of which were found to have some predictive value, were:

- **TravelTime:** Our target feature, engineered from the Timestamp feature and representing the time taken in seconds to traverse the relevant segment in a particular unique bus journey. This is a quantitative output, so a regression predictive task is necessary at the modelling phase [30, pp.10].
- **Rain:** Another continuous feature; see Chapter 3.1.2 for details.
- **WindSpeed:** Another continuous feature; see Chapter 3.1.2 for details.

	TravelTime	Rain	WindSpeed	JPID_length	XBuses	SchoolHoliday	Day	HourFrame
0	160	0.033333	14.5	49	1	0	Tuesday	7
1	141	0.033333	14.5	49	1	0	Tuesday	7
2	158	0.033333	14.5	49	1	0	Tuesday	7
3	139	0.000000	14.0	49	1	0	Wednesday	7
4	143	0.000000	14.0	49	1	0	Wednesday	7

Figure 4.3: Analytics Base Table: Sample Dataframe snapshot

- **JPID_length:** This is also a continuous feature, derived from counting the number of stops in the entire route of each JourneyPatternID that traversed a segment. We decided to create this feature after analysis of the mean/median travel times of each JourneyPatternID on a selection of segments used by a substantial number of routes; these showed great variations, with buses with longer routes seeming to take longer on average. Implemented in the application by inputting the number of stops traversed by the bus route selected by the user.
- **XBuses:** This is a boolean feature, also engineered from JourneyPatternID, and representing whether a bus is an 'Express' bus which skips a number of stops on the route and consequently spends longer on segments at which it does stop; we noticed that such buses often had very different TravelTime values from the mean/median.
- **SchoolHoliday:** This is another boolean feature; Jan 1st to Jan 6th inclusive are true for this feature. This was implemented in the application by manually entering Irish school holiday dates listed online.
- **Day:** This was engineered from the TimeFrame feature. We did not bin this feature as initially intended, as patterns varied substantially between SSIDs.
- **Hourframe:** This was engineered from the Timestamp feature. As with 'Day' we did not bin this feature as initially intended, for the same reasons; see figure 4.4 for more detail.

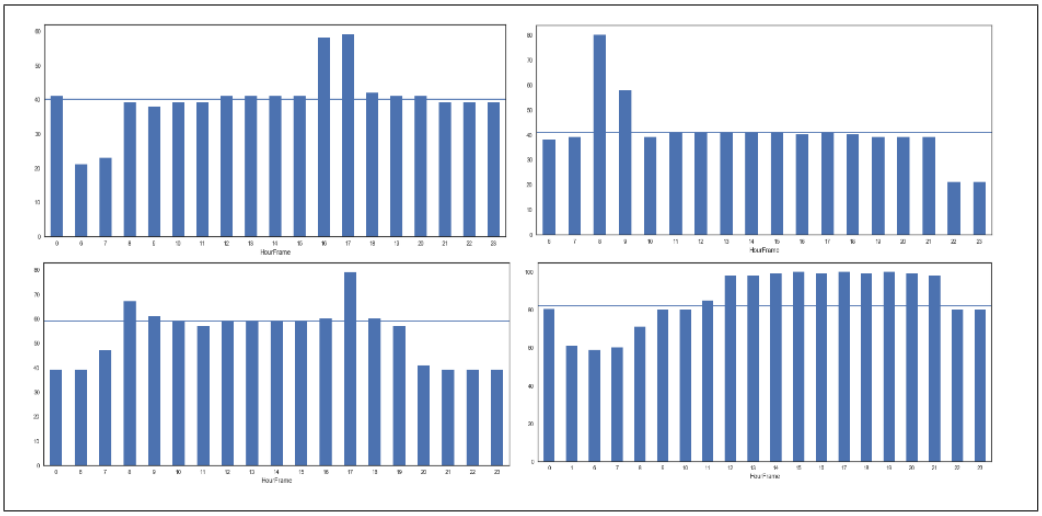


Figure 4.4: Median hourframe values per hour per SSID. Horizontal line is median for all hours. All inner-city. Clockwise from top left: SSID12881289, on S.Circular road, outbound; SSID13481349, opposite SSID12881289, inbound; SSID13551357, on G. George's St, inbound (shares the 9, 16, and 122 routes with SSID13481349, with just four stops between the segments), SSID09090786, on Leeson St, inbound

4.2 Modelling & Evaluation

4.2.1 First Modelling stage

For the modelling phase, we started with linear regression (using Statsmodels) as it was one we were all familiar with from previous coursework, and we wanted to see the p-values to help us evaluate the worthiness of our features. Results were as unimpressive as the literature had prepared us for, but it provided a good benchmark for the next stage when we tried out multiple models. At this point we had not yet created features derived from JourneyPatternID and instead were creating dummy categories for the various JourneyPatternIDs themselves (as well as Hourframe and Day). This approach didn't work very well at all as the vast majority were of little or no predictive value, and we also realised we would have trouble matching these to the trip_ids from the GTFS data that we were using for the timetable/route in the database, which is what led us to exploring other ways of representing the variations from the mean/median that many of these routes demonstrated. Also, we had not normalised the continuous features (at this stage just weather) initially, though when we did do this it had little to no noticeable effect on the predictive value of the model or any of the features.

4.2.2 Second Modelling stage

In this stage we moved onto using Scikit-learn to try out several estimators, on a 70:30 train:test split. We did this on about five SSIDs, and used the default parameters for all of them, in order to see which estimators worked best on these. To measure accuracy, we used Mean Absolute Error (and percentage error - MAE/MAPE), Median Absolute Error (and percentage error - MdAE and MdAPE), and R^2 value.

Our primary focus was on the MdAPE, with an eye on the MAPE and to a lesser extent the R^2 to check there wasn't too much discrepancy between the values. Partly this was because the Absolute Percentage Error metrics gave us a clearer idea of how far the model predictions were from what we wanted them to be, partly because resilience to outliers was very important in evaluating our models, and partly because around this time we happened upon the paper by Gal *et al*. [11] that examined Dublin Bus GPS data from a very similar perspective to ours, and these were the metrics that were focused on there - in fact they only briefly mentioned the R^2 metric in passing, with values similar to what we eventually achieved.

They had also used Root Mean Squared Error, and much of what we researched had promoted its use, but we decided against this as we had reason to believe it was an inappropriate indicator of average error [31], and indeed Gal *et al*.pointed out its unsuitability for this data due to its sensitivity to outliers (see Chapter 6 for more detail on the problems we had dealing with these).

The estimators we tried out were Linear Regression, Support Vector Machine Regression with its various kernels (on the recommendation of our background reading), and decision tree-based models, as these seemed like they might be a good fit [30, pp.352]:

- **Linear Regression** (again): This was more as a baseline comparison than in any hope of using it for our modelling.
- **Support Vector Machine Regression with Linear Kernel:** Results were not much different from Linear Regression, but it ran a lot slower.
- **Support Vector Machine Regression with Polynomial Kernel:** Usually the worst results of all. Also ran very slowly.

- **Support Vector Machine Regression with rbf Kernel:** As expected [10], results were substantially better than the other SVM kernels, and it also ran a lot more quickly.
- **Decision Tree Regression:** Tended to overfit the data severely, with strong results in the training slice and very poor ones in the testing slice.
- **Decision Tree Regression with AdaBoost:** Results were consistent but poor, not very suprising since with a high number of outliers 'the emphasis placed on the hard examples can become detrimental to the performance of AdaBoost' [32]
- **Gradient Boosting Regression:** Usually gave the best results of all, with relatively high consistency between the training and test results, perhaps largely down to its robustness to outliers [30, pp.379].
- **Random Forest Regression:** Surprisingly (as it went against what we've read) tended to overfit the data badly, though not as much as bare Decision Tree Regression.

For the third stage, we decided to proceed with Support Vector Machine Regression with rbf Kernel, Gradient Boosting Regression, and Random Forest Regression (as although the later was disappointing, other groups were apparently having success with it and we thought with some fine-tuning we might get better performance).

4.2.3 Third Modelling Stage

At this stage we tested another ten or so SSIDs in addition to rerunning tests on those we had already examined. We did the same 70:30 train:test split, and then ran RandomizedSearchCV with 5-fold cross-validation and ranges of hyper-parameters recommended by various online sources on the training section, followed by rerunning the best parameters found on the test section. We used RandomizedSearchCV rather than Gridsearch (which was tried once and took several hours) because it was much more efficient, and arguably gives better results [33].

Support Vector Machine Regression gave rather disappointing results, with negative r-squared values common; we were not sure how much we needed to worry about that because it wasn't a metric we were focused on, but it certainly wasn't encouraging. Its ineffectiveness may have been because we hadn't normalised the three continuous descriptive features. Our reasons for not doing so were down to worries about values outside of the range of our dataset being inputted for weather (as the 2012 winter was a particularly mild one) and the fact that this would really have had to be done several steps back in the data preparation stage.

Random Forest worked much better with hyper-parameters tuned than before, but generally not as well as Gradient Boosting Regression, so we turned our focus to the latter. However a substantial amount of time spent adjusting the parameter ranges yielded no real improvement on the default that would generalise over to more than handful of segments, and indeed any set of parameters we found that would improve performance on one or two segments tended to degrade it on others. Even more annoyingly, a few times the default settings returned better MdAPE/MAPE scores than those returned by RandomizedSearchCV. We may well have been trying to tune too many hyper-parameters at once however, or trying to search too wide a range at a time.

The only small improvement we found was changing the loss function. Gal *et al.*all recommended changing this from the default of least squares, in which the initial model is given by the mean of the target values, to least absolute deviation in which it is instead given by the median. We found however that this almost exclusively reduced our scores, with a particularly negative influence on segments with lower numbers of rows. The Huber loss function however, a combination of both which is known for its strong resistance to gross outliers [30, pp.349], was found to generally

produce minor improvements to the results and also tended to minimise the gap between training and testing scores.

One other thing worth briefly discussing is the feature rankings delivered by these tree-based methods, which showed that wind-speed was almost always of greater predictive value than rainfall, contrary to what we expected. This may be because it is less localised and a better indicator of poor weather all round. Also, JPID_length tended to be ranked at or near the top on segments traversed by a number of routes, which shows the usefulness of this feature in this context.

Below is a table with average scores for several of the considered models after running them over all of the segments, and a chart showing lowest to highest MdAPE (and corresponding MAPE) scores across same for the final selected-for-production Gradient Boosting Regression model. The enormous spike at the end is largely caused by segments at the starts of routes, where the data tended to be at it's worst; only 3 out 5,500 or so are over 100% MdAPE, and it's somewhat amusing to note that the worst of them all, at just over 200%, is SSID07670768 - from UCD campus to the Stillorgan Road.

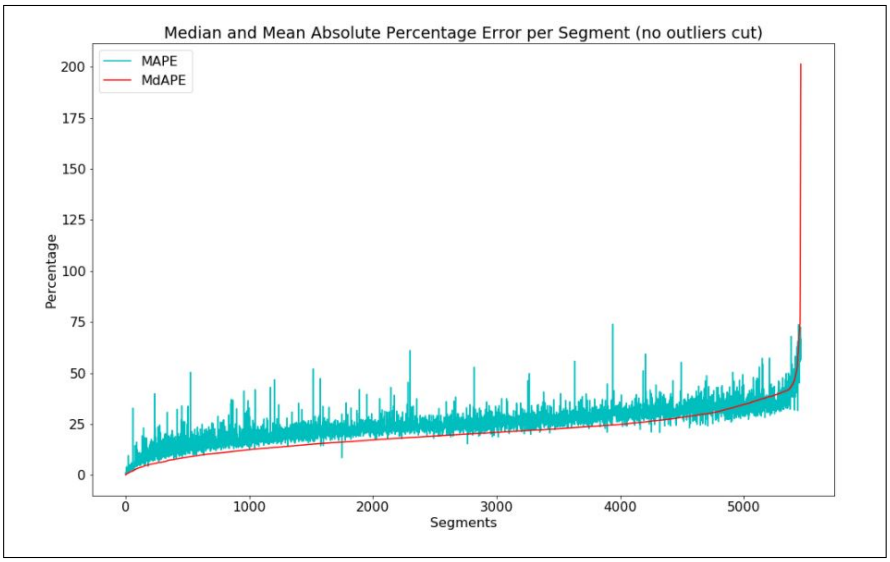


Figure 4.5: Results over all segments for final GBR model

Model [1]	MdAPE	MAPE	R^2
Gradient Boosting Regression (final) [2]	20.69	25.12	0.28
Linear Regression	26	29.48	0.16
Support Vector Machine Regression	21.76	29.15	-0.03
Decision Tree Regression with AdaBoost	34.24	34.58	0.09
Random Forest Regression [3]	25.87	29.72	0.15
Gradient Boosting Regression (preliminary) [4]	19.60	22.47	0.30

Table 4.1: Summary of Average Modelling Scores Across All Segments

[1] Results are using default scikit-learn estimator hyper-parameters and without outliers cut, unless otherwise stated.

[2] loss='huber'

[3] n_estimators=250, max_depth=4, min_samples_leaf=10, min_samples_split=20; NOTE: hyper-parameters chosen from RandomizedSearchCV results as overfitting at default would exaggerate the effectiveness of the estimator.

[4] loss='huber' and outliers cut at 3xIQR at upper and lower bound

Chapter 5: **Front-end and Deployment**

The architecture of the site is as pictured in figure 5.1. The user connects to the site, hosted on the UCD server (as are the models and database, which was moved from Amazon RDS after it was found to be too slow), via the web interface. This front-end of the application has been made mobile-friendly via bootstrap (and with the help of the developer tools in Google Chrome), but is also suitable for desktop usage. The user is able to select a route via a dynamic drop-down menu on the home page (5.2). Once this has been selected, the parameter is passed to Flask which queries the database for the GTFS-data derived information about directions and stops on the route. These fields are then populated for the user to select from. From there the process follows the steps illustrated in figure 3.3. The input forms check for user errors (e.g. selecting a date more than five days into the future or a time outside bus running hours) and returns an appropriate message if one occurs.

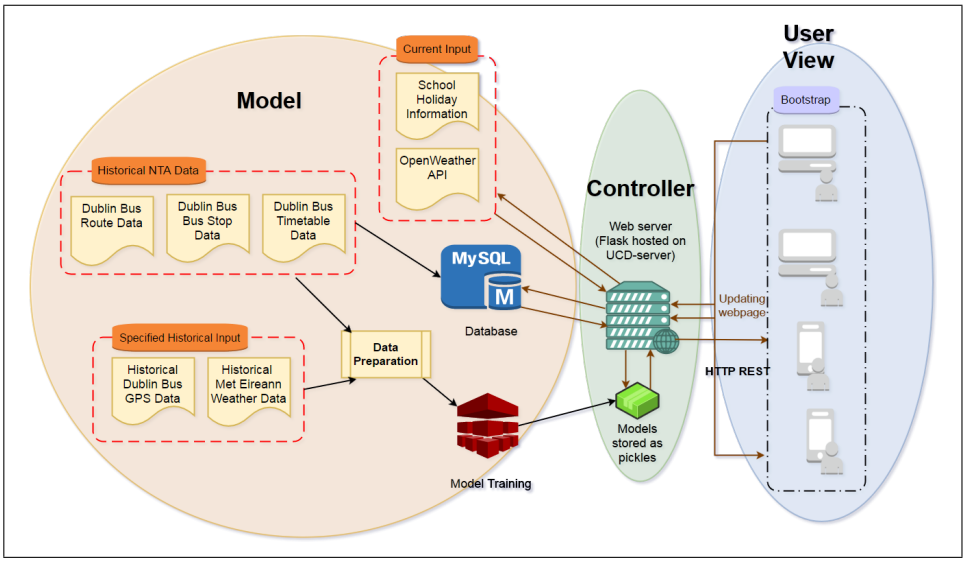


Figure 5.1: Application Architecture.

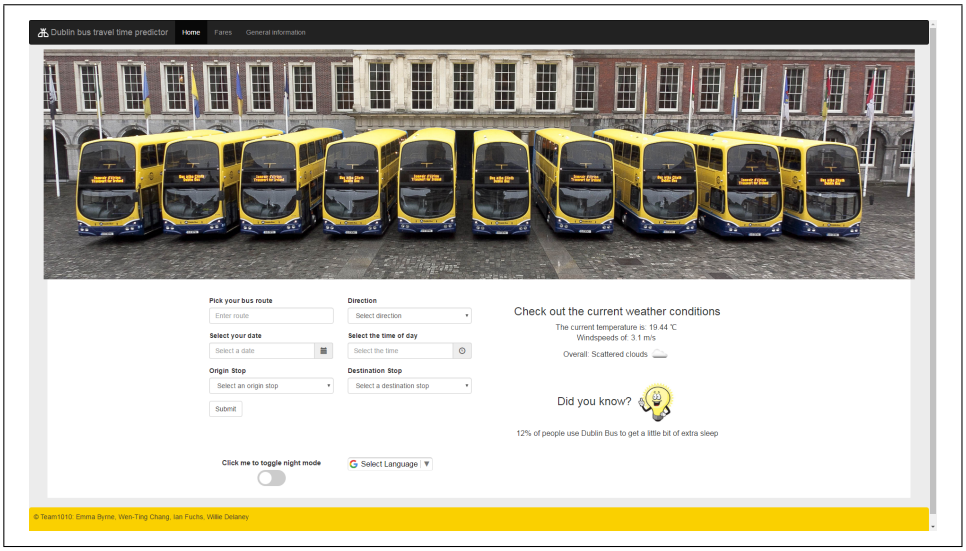


Figure 5.2: Web Application Homepage.

Innovations:

All of the innovations proposed in our group report were implemented, along with a few others. These are:

- A Twitter feed to show road accidents and updates on traffic conditions or bus delays was implemented. Twitter feeds from Luas and IrishRail were also embedded, as well events, diversion information, and airport services information pages.
- Buttons so that a user can tweet directly to our Twitter account were added.
- The ability to translate the entire page (with the exception of a couple of input fields) into a user-specified language the user desires was added via the Google Cloud Translation API.
- Weather information is directly displayed on the home page, as well as a bus trivia feed.
- Predicting bus arrival time at the requested stop was implemented, as shown in figure 3.3.
- A map with the selected route highlighted is displayed when the user's request is fulfilled, as are indicators for each stop along the route, along with a pop-up boxes on hover with predicted arrival times at each stop, as can be seen in 5.3 below.
- A night-mode for battery-saving and to make the display easier on the eyes; this is automatically triggered between 10pm and 6am, but can be toggled on or off at any time.
- A separate page was created with fare information, and a Dublin Bus fare calculator was embedded into this.

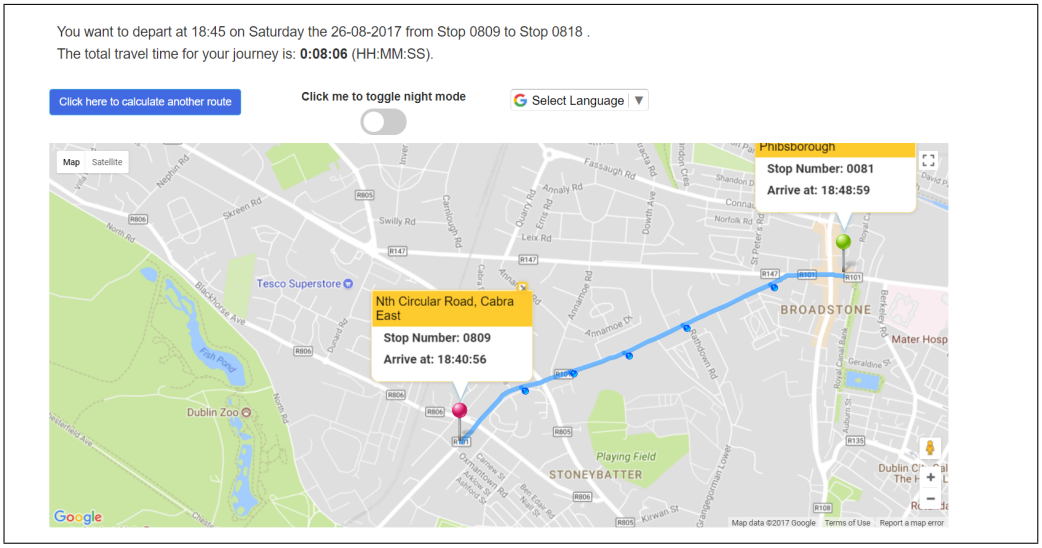


Figure 5.3: Results page of Application.

The website was working properly, if a bit slowly, and the application was successfully retrieving and displaying reasonable predictions from preliminary models for the 46A. However, we were late in producing and adding the final complete set of models for the app, and when this was done, multiple unexpected issues arose - functionality is intermittent with quite regular errors indicating problems with accessing the database, while some routes don't work at all, and those that do sometimes won't display the map or the route shown is incorrect; in the screenshot above the route should be several segments longer than what is displayed and there is a mismatch between the destination shown and that predicted. We didn't have time to properly investigate and fix these issues, so unfortunately this is the final status of the submitted application.

Chapter 6: **Detailed Implementation and Evaluation - Outliers in the data**

The distribution pattern of the majority of segments analysed basically demonstrated a heavily right-skewed unimodal distribution (if we are to ignore the apparent multimodal shapes caused by the method of probe data collection at timed intervals), as demonstrated in the below sample SSID graph 6.1. The multiple outliers exemplified here in the lengthy tail and box-plot caused severe distortion in the data and was a problem we had great difficulty grappling with.

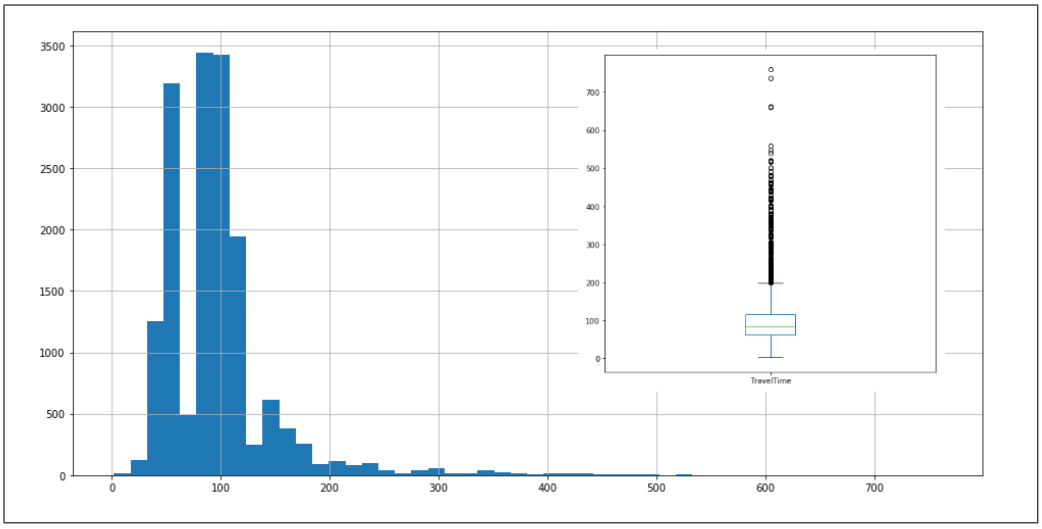


Figure 6.1: Histogram and box-plot of SSID13551357

Close examination of the outliers for SSID09090786 showed that reasons for the highest Travel Time values included buses spending prolonged periods of time sitting at stops, sometimes because they were too far ahead of schedule and sometimes for no obvious reason, and unusually extreme heavy traffic. Generally though, the high outliers on a segment tend *not* to be grouped together by time and date, which indicates that these are more likely to be due to unusual stoppages, and thus should be discarded from the data. To do so we cut upper bound outliers at what we thought was a conservative 3xIQR.

On the other side, there were also sometimes Travel Time values in single digits, as low as two seconds for some segments, which were certainly also erroneous. Cutting the lowerbound by multiples of the IQR usually failed to remove these, so instead we used GeoPy to calculate the minimum time required to traverse each segment at the maximum bus speed of 65kmph, and cut any values below this.

However, as we were about to deploy the models we discovered that this method saw some segments lose over 99% of rows as 'outliers', and in total over 16% of SSIDs were losing 10%+ of their rows in this manner, which was obviously unworkable. We are not sure if this was caused by errors in the use of Geopy or in the GPS data, but we reverted back to 3xIQR on both bounds. However we then discovered that even using this method some segments were losing over 40% of rows and over 8.7% of SSIDs were losing 10%+, so we made a last-minute decision to leave all outliers in. This was not at all ideal, but was surely the only choice we had at that stage as none of us knew of a reliable to way to automate the handling of this problem in this dataset, and it didn't actually affect average MdAPE as much as we feared, increasing it by little over 1%.

Chapter 7: **Personal Contribution**

As is no doubt clear from the rest of this report, my focus in this project was very much on the predictive data analytics side. My contributions to the front-end and database-integration side of the project was largely limited to reviewing the existing work and giving advice/making suggestions on features to be implemented or changes to be made. I also did some reviewing and error-checking of the python code used in these areas of the project. I don't feel my contributions in these areas are really worthy of extensive discussion here.

In working on the data analysis, preparation, and modelling aspects of the project, I collaborated very closely with one other team member, to the extent that for several elements of the completed work it is difficult to say precisely who did what. I shall try to be as clear as possible where the work I did was my own and where the efforts were shared in the following.

I was largely responsible for pushing the idea of modelling the data by stop-to-stop segment in the first place, though I'm unsure now whether this something to be happy about or not. In terms of the actual analysis, I started the ball rolling by analysing an individual day's data in detail ('Exploring_siri20121106.ipynb'), and repeating this process with a handful of other days' data. From this, I and my collaborator discovered some of the more obvious data quality problems and were able to devise initial cleaning steps.

When we were still considering trying to create an app that would work with current bus routes, I contacted the National Transport Authority in a futile attempt to get a list of the changes in bus routes and stops and their locations since 2013, and spent quite a bit of time investigating differences between the routes and stops then and now. I also took the initiative of speaking to bus drivers about what they could recall of the data collection system at that time and discovered that at least some of the data had had some manual input from the drivers.

When we were having trouble working with the entire dataset I researched more efficient alternatives to CSV and pickle files for storing and loading data, trying out first the dask format and then discovering and recommending the feather format. Between this and learning about the most efficient file types for working on the data, I was able to pare down the dataset to under half its original size in 'Initial_Cleaning_of_All_via_Pickle-v3.ipynb' so that we were able to load it into our notebooks to work on in about a minute, as compared to the fifteen minutes or so this took using our original method of stitching together multiple pickle files.

While I was able to make recommendations on subsequent cleaning steps, much of the required coding was beyond my skill level and my collaborator did the bulk of this work, including the code necessary to compare the GTFS data with the original datset, the use of the GeoPy module to identify 'rogue' StopIds, and the creation of the algorithm to transform the original data into SSID-based rows, separated out into a series of route-based CSVs.

Once these had been created, I worked on producing a series of notebooks analysing individual segments and graphing the most important information, and determining the best way to proceed with the modelling, including deciding what additional features we wanted to engineer, though I had plenty of help from my collaborator in figuring out what to do and how to do it at this stage too.

From this point on I mostly worked alone at researching different machine-learning estimators and experimenting with them on over fifteen individual segments, including using RandomizedSearchCV in an effort to improve predictive performance, before finally settling on what I felt was

the best approach. The colleague who had been working with me on the data was at this time mostly engaged with developing the map for the web application and an algorithm for integrating the models with the front-end (which I helped to design). We then worked together on developing a system for producing a complete set of models, though the bulk of the coding work was hers. I incorporated a check into the model production phase on the extent to which outliers were being cut, which led to our late decision to abandon doing so.

I have to admit that I spent much too long trying to discover optimal hyper-parameters for the estimators I was trying out, when my first priority should have just been to produce a full set of models of any type for testing purposes. This was due to my misapprehension that the production of models would be a very time-consuming process; I was very surprised when I found I was able to produce over 5,500 GBR models in little over half an hour. I quickly adapted the process to produce a series of CSV files with modelling results for several of the rejected estimators for comparison purposes.

A subset of test models for the 46A route had shortly beforehand been produced which worked fine with the web app, so we didn't anticipate the possibility of problems once the rest were done, but we had been warned before that scalability issues might arise at that point and I did not pay sufficient heed to those warnings. I feel I have learned a lot over the past few months, but if I had produced a full set of models earlier in the process, we may have had time to fix the issues that arose. The lessons I take from this are to ensure in future that any assumptions I make about any step in a process are grounded in firm evidence, and to avoid getting so wrapped up in whatever tasks I'm working on that I lose sight of the needs of my team and the aims of our project.

Chapter 8: **Conclusions and Future Work**

8.1 Conclusions

There are plenty of positives that we can take from this project. We attempted to create a relatively complex system of bus journey travel time and stop arrival time prediction, wrapped up in an attractive and useful mobile-friendly web interface, and feel we came close to succeeding. We created innovative features for both the web application and the predictive engine that went beyond the project specifications. We explored the given data in great depth, discovered a lot of problems that weren't immediately obvious, and came up with innovative solutions to some of these. We developed a model for the prediction of bus travel times that while not exactly highly accurate, compared quite favourably to the results of the similar approach taken by Gal *et al*. [11] on a much smaller scale (as they only modelled the 46A) and presumably with more resources and experience.

However, the application as it stands is not properly functional, which is a great disappointment to us all. We are still unsure of what went wrong. It seems likely that it has something to do with our attempt to integrate the Dublin Bus GPS and GTFS datasets; perhaps the latter was less accurate than we had assumed, or maybe we matched up the wrong stops somewhere along the line. It is also possible that we made errors in our use of the Python GeoPy module, as this was not something we had had any training in prior to our attempt at this project - the issues we had in using that to try to cut lower-bound outliers indicate that there were some problems with our implementation of it and so there are likely other issues arising from that we didn't spot (though then again these problems could be down to inaccurate GPS data).

Perhaps we didn't take a proper approach to the scale of the data that we had to deal with, in that rather than focusing on one route's worth of segments to start with as was done by Gal *et al*.we focused narrowly on disparate individual segments and then suddenly tried to scale up to an entire network of buses. In retrospect, that certainly seems like much too much of a leap of faith.

8.2 Future Work

The major future work task on this application would of course be to find out what was causing our system failures when we expanded to the full range of models and fix that if possible. It was also quite slow to load the predictions when working properly, so the interface and/or the algorithm for fetching the predicted times via the models could do with optimisation; incorporating parallel processing if possible would be one obvious route to explore if the fault is with the latter.

Given all the issues we had processing the data however, and the reasonably high likelihood that the failure of our system is at least in part as a result of data quality problems (even if that's just errors we made in the process of trying to create a workable set of SSID models), we feel that the application of the process we undertook to more recent data from Dublin Bus would give it a better chance of success. The drivers who told us that the data collection system at the time was

a mixture of automatic and manual also told us that not long afterwards, the remaining buses that required some manual input from the driver were upgraded/replaced and that the entire fleet now operates with fully automatic vehicle location technology, so it would be worthwhile seeing if there is much difference in the data quality now.

It would also likely be necessary to develop an effective approach to identifying and removing invalid outliers, as even with a cleaner dataset there would be clearly invalid Travel Time examples such as those on SSID09090786 discussed in chapter 6. Clustering is one approach that could be viable, but we do not know enough about how this may be implemented to firmly advance it.

Additionally, as previously stated in Chapter 4.2, one major issue we had with modelling was the difficulty in moving beyond the default hyper-parameters for each estimator. We are confident that the Gradient Tree Boosting model was the best performing of those we tested, but no one set of hyper-parameters that worked well on some segments was found to be applicable to all without substantially degrading predictive performance below the level of that provided by the defaults on other segments.

It may perhaps be possible however, to find groups of segments which together may be amenable to an identical approach. We would suggest that applying the following criteria for segment classification could achieve this:

1. **District type:** Segments on roads passing through different kinds of areas revealed different patterns of deviation from mean TravelTime by hour and day (see figure 4.4 for subsequent examples). A suggested set of categories would be: Residential (SSID13481349 and SSID12881289), Commercial (SSID13551357), Business/Education (SSID09090786), and Rural. Perhaps division of the first three into two or more urban zones would be necessary. Additional categories may also be necessary, but these are the most obvious from our analysis.

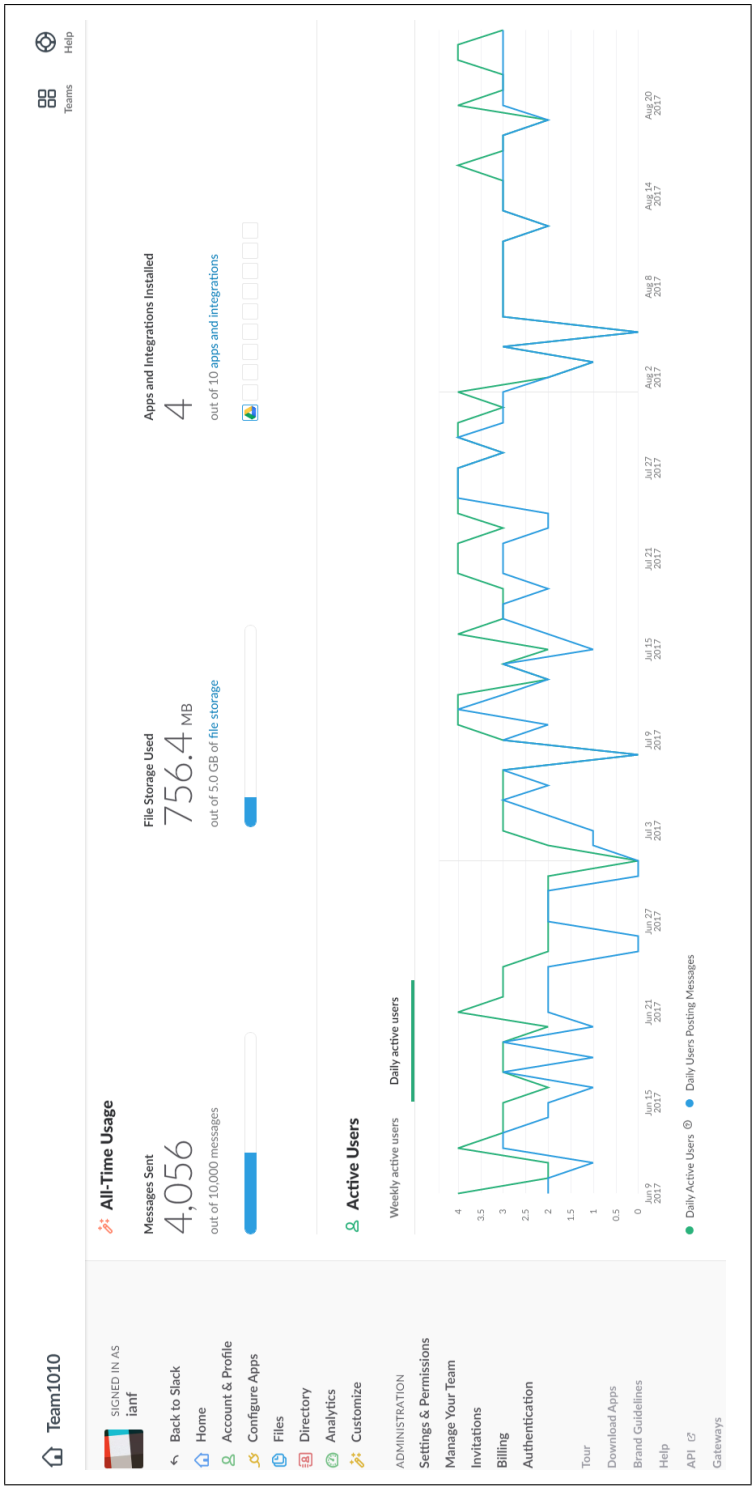
2. **Direction:** Segments on the same stretch of road but travelling in different directions also had different patterns of deviation from mean TravelTime by hour and day (compare SSID13481349 and SSID12881289, for example). A suggested set of categories would be: Towards City Centre, Away from City Centre, and Neither.

3. **Traffic Flow:** This might be most easily captured by counting the total frequency of all buses traversing each segment on a weekly basis, and binning them into categories of ranges, under the reasonable assumption that the more buses that travel along a stretch of road, the more other vehicles will also be travelling on it.

4. **Segment Length:** This was something we failed to consider in our own analysis, but upon reflection is likely to be an important factor in determining the best approach to an individual segment. This should also be binned into categories of ranges.

The existence of traffic lights on a segment may be another factor worth considering. We think it is quite reasonable to expect a high enough proportion of segments sharing a value from each of these criteria to be amenable to the same modelling approach; the number of potential categories is quite high, but would still be substantially smaller than the total number of roughly 5,500 segments existing in the network, and it is quite possible that the same modelling approach may be found to be applicable to more than one category type, which could most easily be discovered if the categories were worked on in descending order of population, with previously developed models tested out on smaller categories before launching into developing new models. In this way, it may be possible to develop a robust and workable Dublin Bus travel time prediction system.

Appendix A: **Slack group activity analysis**



Bibliography

[1] *Dublin Bus GPS sample data from Dublin City Council (Insight Project)*. (28th Jun. 2013). Retrieved from `https://data.dublinked.ie/dataset/dublin-bus-gps-sample-data-from-dublin-city-council-insight-project`

[2] *Display and Download Historical Data*. (n.d.) Retrieved from `http://www.met.ie/climate-request/`

[3] Dhivyabharathi, B., Anil Kumar, B., Vanajakshi, L. 'Real Time Bus Arrival Time Prediction System under Indian Traffic Condition'. (2010). *2016 International IEEE Conference on Intelligent Transportation Systems, Singapore.* 8-22.

[4] Fan, W., Gurmu, Z. 'Dynamic Travel Time Prediction Models Using Only GPS Data'. (2015). *International Journal of Transportation Science and Technology, Vol 4, no. 4,* 353-366.

[5] El-Geneidy, A.M., Horning, J., Krizek, K.J. 'Analyzing transit service reliability using detailed data from automatic vehicular locator systems'. (2010). *Journal of Advanced Transportation, Vol 44, Issue 1,* 66-79.

[6] Bejan, A.I., Gibbens, R.J., Evans, D., Beresford, A.R., Bacon, J., Friday, A. 'Statistical Modelling and Analysis of Sparse Bus Probe Data in Urban Areas'. (2010). *2010 13th International IEEE Conference on Intelligent Transportation Systems, Madeira Island, Portugal.* 1256-1263.

[7] Baptista, A.T., Bouillet, E.P., Pompey, P. 'Towards an uncertainty aware Short-Term Travel Time Prediction Using GPS Bus Data: Case Study in Dublin'. (2012). *2012 15th International IEEE Conference on Intelligent Transportation Systems, Anchorage, Alaska, USA.* 1620-1625.

[8] Sinn, M., Yoon, J.W., Calabrese, F., Bouillet, E.P. 'Predicting arrival times of buses using real-time GPS measurements'. (2012). *2012 15th International IEEE Conference on Intelligent Transportation Systems, Anchorage, Alaska, USA.* 1227-1232.

[9] Bai, C., Peng, Z., Lu, Q., Sun, J. 'Dynamic Bus Travel Time Prediction Models on Road with Multiple Bus Routes'. (2015). *Computational Intelligence and Neuroscience, Volume 2015,* Article ID 432389.

[10] Yu, B., Lam W.H.K., Tam, M.L. 'Bus arrival time prediction at bus stop with multiple routes'. (2011). *Transportation Research Part C 19 (2011)* 1157-1170.

[11] Gal, A., Mandelbaum, A., Schnitzler, F., Senderovich, A., and Weidlich, M. 'Traveling time prediction in scheduled transportation with journey segments'. (2015). *Information Systems, Vol. 64,* 266-280.

[12] '31 million increase in public transport passengers since 2013'. (24th Aug. 2017) in *Transport for Ireland*. Retrieved from `https://www.transportforireland.ie/31-million-increase-in-public-transport-passengers-since-2013/`

[13] *Transport for Ireland - Journey Planner*. (n.d.). Retrieved from `http://www.transportforireland.ie/`

[14] 'Ireland's public transport schedule now included in Google Maps'. (27th Nov. 2013) in *thejournal.ie*. Retrieved from `http://www.thejournal.ie/google-maps-transport-ireland-1194891-Nov2013/`

[15] *Dublin Bus GTFS data*. (22nd Mar. 2013). https://data.dublinked.ie/dataset/dublin-bus-gtfs-data

[16] *pandas*. (n.d.). Retrieved from http://pandas.pydata.org

[17] *Jupyter/IPython Notebook Quick Start Guide*. (n.d.). Retrieved from http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

[18] *scikit-learn - Machine Learning in Python*. (n.d.). Retrieved from http://scikit-learn.org/stable/

[19] *Welcome to Statsmodels's Documentation*. (n.d.). Retrieved from http://www.statsmodels.org/stable/index.html

[20] McKinney, W. *Feather: fast, interoperable data frame storage*. (n.d.). Retrieved from https://github.com/wesm/feather

[21] *Welcome to GeoPy's documentation!* (n.d.). Retrieved from https://geopy.readthedocs.io/en/1.11.0/

[22] *Flask - web development, one drop at a time*. (n.d.). Retrieved from http://flask.pocoo.org

[23] *Bootstrap*. (n.d.). Retrieved from http://getbootstrap.com

[24] *What is jQuery?*. (n.d.). Retrieved from https://jquery.com

[25] Kelleher, J.D., MacNamee, B., D'Arcy, A. *Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies*. (2015). MIT-Press

[26] *What is the CRISP-DM methodology?*. (n.d.). Retrieved from http://www.sv-europe.com/crisp-dm-methodology/

[27] 'Fundamentals of Transportation/Timetabling and Scheduling'. (n.d.) in *Wikibooks*. Retrieved from https://en.wikibooks.org/wiki/Fundamentals_of_Transportation/Timetabling_and_Scheduling

[28] 'General News Archive' in *Dublin Bus*. (n.d.). https://www.dublinbus.ie/News-Centre/General-News-Archive

[29] 'New Bus Timetables' in *Malahide Community Forum*. (12th Nov. 2012). https://malahidecommunityforum.ie/2012/11/12/new-bus-timetables/

[30] Hastie, T., Tibshirani, R., Friedman, J. *The Elements of Statistical Learning, Second Edition*. (2001). New York, NY, USA: Springer New York Inc.

[31] Willmott, C.J., Matsuura, K. 'Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance'. (2005). *Climate Research, Vol. 30,* 79-82.

[32] Freund, Y., Schapire, R.E. 'A Short Introduction to Boosting'. (1999). *Journal of Japanese Society for Artificial Intelligence, 14(5),* 771-780.

[33] Bergstra, J., Bengio, Y. 'Random Search for Hyper-Parameter Optimization'. (2012). *Journal of Machine Learning Research, 13,* 281-305.