Research Practicum Project Report

Dynamic Travel Time Prediction

Stephen Moles

A thesis submitted in part fulfilment of the degree of

MSc. in Computer Science (Conversion)

Group Number: 5

COMP 47360

UCD School of Computer Science

University College Dublin

August 26, 2017

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. 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

Over 115 million passengers use Dublin bus in a year. With such high figures, providing accurate estimates for waiting times and journey times is crucial to create an efficient service that is free of frustration and anxiety. The current Dublin Bus web application is lacking in such regard through its provision of real-time waiting estimates which can increase or stay at the same for many minutes. In addition, planning a future journey, calculating a fares cost, and planning your route are all segregated features which make site navigation an arduous experience. Accordingly, this paper reports on a project conducted to develop a web application that generates more accurate time estimations for bus arrival and journey times. In terms of the methodology, the Dublin Bus API is used to find arrival times based on the closest stop to a bus, with historical data then used to build predictions from a random forest regression model. A dynamic web application is also built to provide valuable and relevant bus information on an integrated platform. This paper therefore documents the approach to the problem and describes the solution taken. The data model results are promising and suggest an alternative approach to bus prediction than currently on offer by Dublin Bus.

Acknowledgements

This project was supported by the staff in UCD School of Computer Science. I am grateful to all those with whom I had the pleasure of working with over the course of this project.

A special thanks to Darragh our Coordination Lead, Lucas our Code Lead, and Peng our Customer Lead. Working with you all throughout this project as part of team BusLightyear has been an unforgettable experience. I wish you every success both in this project, and in future endeavours.

Table of Contents

Chapter 1: **Introduction**

1.1 Project Motivation

On an average weekday, 400,000 people use Dublin Bus. This amounts to 70% of all bus commuters in Dublin during peak times (Dublin Bus, 2017). With such large numbers of passengers relying on its service, travellers expect to know accurate arrival and journey times. It attracts additional ridership, increases user satisfaction, and reduces levels of congestion (Jeong, 2004). However, factors such as traffic, weather, and variations in passenger demand cause unreliability of service, making it difficult for buses to adhere to a static timetable (Shalaik, 2012: 9). Therein lies our business problem. Our challenge is to create a mobile-friendly web application that generates accurate estimations for bus travel times using historic data from November 2012 and January 2013. This paper describes our approach to the assigned problem, details our proposed solution, and outlines our achievements.

The established benchmark for this application is the current Dublin Bus model. This project takes both its front and back-end design as a benchmark. Dublin Bus deliver their travel information

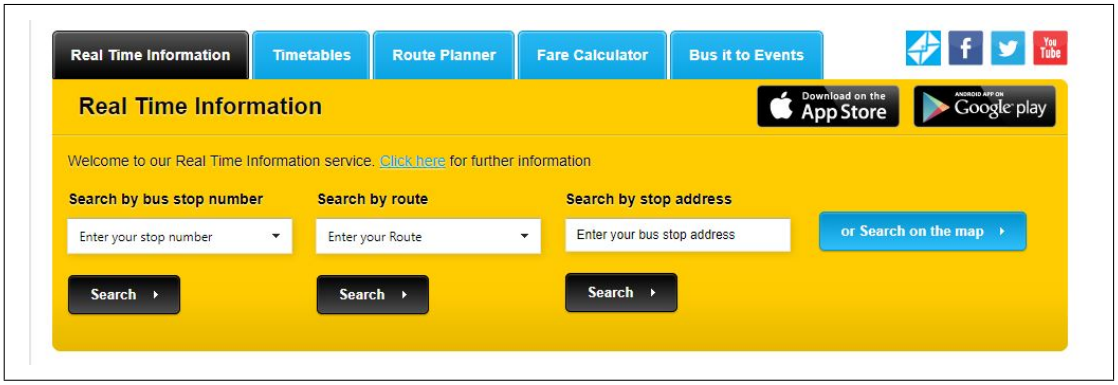

Figure 1.1: Dublin Bus index page benchmark.

through four methods: 1) a static timetable with generic travel predictions for future journeys, 2) a real-time application which is based on a GPS Automatic Vehicle Location (AVL) system to deliver accurate stop arrival time information, 3) a route planner which allows users to plan routes and destinations, and 4) a fare calculator. Dublin Bus currently provide this information on both web and mobile-based applications. However, these features are not user-friendly and fail to meet the demands of average users.

Searching for static timetable redirects the user to select their desired route from a list of possible routes. After selecting, the user is presented with bus frequency information, when the bus should leave its source, and a static estimation of arrival time of the bus in a locality. This information is incompatible with the modern bus user. Personalised fare information, a visual representation of the bus trajectory, or specific stop arrival timing is not offered. This method of data delivery is segmented and inaccessible.

Dublin Bus provide a real-time application which provides users with the most uptodate information about their bus arrival time at their stop. However, accessing this information on their website is just as inaccessible, and the provided information may ultimately change due to un-

foreseen factors. Similarly, it does not provide the user with personal fare or journey travel time information. The reality is that this benchmark is quite dated, segmented, and does not offer an adequate user experience. Rather than tailoring data to suit the user, it delivers generic, general, and often inaccurate information. While real-time information evidently adds value to the service, with a reported 30% uptake in passenger numbers (Murphy & Sweeney, 2013), the service is "complex and prone to issues arising from traffic congestion, communications, and operational issues" (Transport Ireland, 2017), and as a result, waiting times can increase or stay at the same time for many minutes (Dublin Bus, 2017).

This report will be structured by a series of chapters that aim to capture the product life cycle of this piece of software. Sections will highlight undertaken research into the domain, and provide a detailed high-level description of the adopted approach. By compartmentalising the project, it will outline strategies employed to deal with front-end, back-end, and deployment issues, make reference to specific design aspects, and evaluate team achievements. It will conclude by making reference to potential optimisations of the proposed solution, while critically assessing both the project's strengths and weaknesses.

Taking this information into consideration, while developing a prediction model is at the project's core, it is also dedicates much time to improving the user experience (UX). For this reason, our aim is to measure the project's success based on not only on the how address the project specification, but also how it delivers the user's requirements.

Our development life cycle was divided into two distinct prototypes across six two-week sprints. The first prototype was established to meet the project specification in its most basic form. To this end, a basic website would be created that could take an input and deliver a predicted output. The second prototype was built on re-factoring this model for improved accuracy and efficiency. It would also include innovative features that go beyond the basic requirements.

While at-stop real-time information has significantly modernised the service, real-life changes in traffic conditions may result in long at-stop wait times, ultimately offering users little relief. Our project scope will therefore deliver an improved application and data model based on researched features that adhere to the demands set by our project specification and user stories, and that meet our defined acceptance and testing criteria.

1.2 Technical Considerations

Python was the programming language of choice for this project. It was the language with which people were most comfortable, and provided extensive libraries such as pandas, stats-models, and sci-kit learn which were central to our predictive model. It also offered benefits in deployment with Django, our web framework, supporting a python command-line Database API. Jupyter Notebook was used for cleaning and modelling the data.

The front-end was implemented with HTML5 and CSS3 with the use of the front-end framework Bootstrap. The core website functionalities were developed through JavaScript and its jQuery library. The back-end consists of a UCD-provided server running Ubuntu 14.04. On the server, a MySQL database stored our cleaned data tables for querying. Deployment of the application was through Django web framework, with an Nginx web server handling static requests, and Gunicorn application server handling dynamic requests.

1.3 Innovations

Five innovative features were time-lined: The first would find the shortest routes serving a users desired stops. The second was geolocation functionality, which would additionally locate nearby stops, and outline route which served those stops. Thirdly, development of an additional page aimed at tourists, which would provide visual and textual information about events and points of interest. To complement this feature, web-page translation was also implemented through a Google plug-in. Our final innovation was the web application itself. Rather than develop an Android application, the group opted to dedicating time to develop a comprehensive multi-paged website, with the aim of improving the UX.

Chapter 2: **Background Research**

2.1 Context Analysis

An increase in predictive efficiency it is hoped will have the same knock-on effect of increasing passenger numbers for Dublin Bus, as it did with the introduction of real-time data (Murphy & Sweeney, 2013: 2). While the literature highlights this success, there are a variety of factors which can cause discrepancies with prediction models. It was therefore of great importance to focus on first establishing the advantages and disadvantages of established prediction models in delivering accurate dynamic travel information. Following consultation of relevant literature, the group opted to implement a Random Forest regressor model.

Increased usage of public transport can be attributed to multiple factors: Ability to deliver accurate bus information, location of bus stops, bus frequency, strong network of other transport connections, and to a lesser extent, rapport between staff and customers (Harrison et al, 1998: 225). While most bus operators to some degree deliver on these standards nowadays, the accuracy at which they deliver on them is still a contentious issue. Arguably, most passenger frustration is caused by two main factors: Inaccuracies in bus arrival, departure and journey time, and by the wait-time required by customers at their bus stop (Caulfield & O' Mahony, 2009: 9).

With daily passenger numbers of nearly 50,000 (Murphy & Sweeney, 2013: 3), it is important to acknowledge that different users have different expectations from the service. Taking for example the use of real-time data, surveys indicate age profiling is an important factor for measuring its usage, returning a p-value of 0.01. The survey indicates that younger demographics are heavily reliant on real-time information in comparison with their senior counterparts (Murphy & Sweeney, 2013: 4).

It outlines further that since the introduction of real-time data, 63.2% of users indicate the service's reliability has increased. Analysing dependence on real-time information across low, medium, and high frequency bus routes, Murphy and Sweeney note a marginal increase of 29.9% in passenger numbers. Perhaps most relevant is that across all types of routes a significant majority, 45.6%, rely primarily on the electronic real-time display at the stop. A lower but equally significant 31% and 12% rely on mobile applications and computers respectively. This fact highlights the importance of developing a model that accurately predicts journey travel time. If as indicated from this survey a majority of passengers depend on at stop information, displaying inaccurate information can directly lead to customer dissatisfaction, frustration, and an increased negative perception of the service. The most interesting finding from the report highlights that public perception of public transport can fluctuate without changes to service operations itself. This supports the argument that better handling of data can be a cost-effective measure to improve the service's accuracy and in turn, improve public perception.

Others validate this argument, with Caulfield and O' Mahony (2009: 3) stating that accurate real-time data reduces the perceived waiting time at bus stops. They refer to a study on the reduction in wait-time after the introduction of real-time data, where Warman (qtd. in Caulfield & O' Mahony, 2009: 4) indicated that 46% felt reassured following its introduction.

Caulfield and O' Mahony's performed their own survey on Dublin-based office workers in the Central Business District (CBD). Their results indicated that out of the 53% who regularly commute via public transport, 29% of these use Dublin Bus. Highlighting the current inaccuracies of the

real-time data, 79% of respondents were apprehensive that the bus would not arrive on time, 70% were concerned that the bus had already departed their stop, while 73% doubted the overall credibility of the departure information (Caulfield & O'Mahony, 2009: 9).

Taking into consideration first-time users such as tourists, 73% indicated they would rely on real-time information, emphasizing the importance of delivering a good first impression of the service. However, survey results showed that on a scale of 1 - 5 (with 5 indicating extremely satisfied), respondents rated the bus as the most unpopular mode of public transport when considering User Experience (UX). Their survey (2009: 11) was based on the following criteria: Provision of visual information such as mapping of routes (1.88/5), accurate timetabling (2.05/5), trip costing information (1.99/5), and website user interface (2.17/5). These results only justified further the need for improving on the current benchmark website.

These statistics give a real insight to public perception, and the impact an increase in efficiency could have on the service. They were particularly relevant while designing our new application and model.

Focusing on the issue of predictive modelling, there have been numerous strategies employed to address accuracy problems, however no universally accepted approach has been deemed as the most effective (Gurmu, 2010: 6). Therefore, a review was conducted on prescribed and non-prescribed readings to analyse the five most widely accredited models (Shalaik, 2010: 42). They are as follows:

1. Historic data based models
2. Time series models
3. Regression models
4. Kalman filtering models
5. Machine Learning models

As previously stated, this project focuses on implementing a Random Forest Regressor model based on its efficiency with non-linear data, and the teams previous familiarity with it.

2.2 Historic data based models

What is evident from the literature is that each model's suitability is dependent on its context: That is to say, while one model may perform well at predicting rural bus routes journeys, the same model may be inadequate for densely populated urban area, producing inaccurate predictions due to the weight of the influencing factors. Examples of this can be seen from the Historic data based models. They function on the premise that future journeys can be modelled as an average of a set of historic journeys (Gurmu, 2010: 7).

Models based on this method therefore assume that a factor such as traffic will on average have the same impact on travel times for future journeys as they did on past journeys. Lin and Zeng (qtd. in Gurmu, 2010: 7) argue that journey times on certain days of the week, or at certain times of the day will be similar at that point in the future. There is an element of truth to this, insofar as commuting each weekday at 8 am is likely to result in longer travel times, than perhaps travelling at that same time on weekends. Time is just an example of one of the factors that can influence journey time. Similarly, public and bank holidays may also alter the level of traffic

congestion. For that reason, these models may be true in the context of rural bus routes where long traffic delays are not commonplace, but untrue in major cities. For this reason, it was not recommended to base our predictions entirely on historic-based model.

2.3 Regression models

Gurmu (2010: 10) argued further that for historic data based models to perform accurately, they required expansive datasets, limited feature variation, and that the historic data be similar to the current real-time data. As these conditions were not met, we therefore researched developing a regression model. Linear regression models "predict and explain a dependent variable with a mathematical function formed by a set of independent variables" (qtd. in Gurmu, 2010: 11). While they do not perform well in this context, they allow for verifying feature importance. For example, in Patnaiks et al (qtd. in Gurmu, 2010: 11) and our own implementation, weather was not categorised as an important modelling feature. This was quite a surprising result, but could be due to its unpredictability, or the difficulties that arise with Linear Regression and inter-correlation of certain features.

2.4 Kalman Filtering & Machine Learning models

Other established models such as Kalman filtering and Machine Learning models were not deemed feasible for this project. Chien (qtd. in Gurmu, 2010: 12) described how Kalman filtering was used to design algorithms that tracked bus locations and modelled statistical estimates for bus arrival times. They combine real-time Automatic Vehicle Location (AVL) data with a historic data based prediction model. Bin et al (qtd. in Gurmu, 2010: 18) argue that while Kalman filtering and Machine Learning models can yield better results than Historic data based and Linear Regression models. However, it is worth noting that they do not yield the same accuracy with multiple time steps. Two of the most prominent Machine Learning models are Artificial Neural Networks (ANN) and Support Vector Machines. ANN models are implemented by imitating the synaptic processes of the neurons in the human brain. Although known to deliver accurate predictions based on noisy and complex data, due to their complexity they were not feasible for this project. It was argued in Gurmu that the modelling of SVMs is not common in this area largely due to the high computational cost involved. Based on this feedback, these models were disregarded for the purpose of this project.

2.5 Model Justification

While Linear Regression proved not to be an optimal model for prediction in this context, the group explored other potential regression models. Random Forest regressor was an alternative from which we decided we would base our model. It is more flexible than other regression models, and is not as constrained by inter-correlation as is the case with Linear Regression. A non-linear model, it is based on communication between variables and decision trees. Using an approach known as ensembling, these trees are divided into smaller sub-trees to produce accurate predictions. Key advantages are that it is not susceptible to over fitting, with a recognised ability

to handle noisy and error-prone data well. This report will provide the implementation of this model in a later chapter. Our model was used in conjunction with the Dublin Bus API, on the basis that its use would increase the model's accuracy.

Chapter 3: **Deliverable Project:**

3.1 Requirements

The importance of customer satisfaction, the deliverance of accurate data, and the user's perception of the service are some of the key factors to take from the review of the domain's current literature. Given these factors and the project specification, we therefore set out to formalise our basic project requirements. During a series of white-boarding sessions the group identified causes that may alter accuracy, and the effects that these causes may have. Some of the highlighted causes were;

- school holidays
- time of the day, day of the week, or month of the year
- weather
- events or specific geographical locations (points of interest)
- road collisions
- road works or construction

The effects are numerous: Traffic jams, varied day and directional levels of congestion, diversions, and fluctuations in bus passenger numbers. There are many examples which highlight the unpredictability of these outcomes. The following is one such example: Take for example a collision on a major road. This is likely to cause greater setbacks if it occurs at rush hour heading city-bound, than were it to occur outside of these hours heading in the opposite direction. Identifying such key influential factors greatly improved the efficiency of our model.

After identifying the above factors, the group next focused on formalising who this application would benefit. There are two beneficiaries in this case: Primarily, the customers of Dublin Bus are the main beneficiaries. However, should this application be a success Dublin Bus themselves may benefit from increased passenger numbers, as was the case with the introduction of real-time data at bus stops. Two classes of users were identified: The frequent or daily users such as school children, students, workers, or old-age pensioners (OAPs). The second class are the occasional users: Tourists, concert or event-goers, shoppers, or business people travelling to and from airports or ports.

After establishing our client-base, we identified some external factors that may impact bus demand. Given the recent upturn in the economy in 2017, it is accurate to state that more people are back at work than in 2012, and therefore more people will be participating in the daily commute. Similarly, Dublin is better connected than it was 5 years ago, demonstrated by the expansion of Dublin Bikes, the reduction of private car use in key city centre locations, and the introduction of longer Luas carriages following the completion of Luas Cross-city. Although difficult to measure their impact on bus passenger numbers, they will undoubtedly have an impact on bus capacity and frequency.

This research has emphasized that in its current form, the Dublin Bus model is quite limited, and does not efficiently serve its customers. It provides generic travel information and data that

often does not pertain to that specific user. As a corrective approach, we therefore developed our application guided the wants and needs of an average user.

3.2 Data Analytics

3.2.1 Data Collection & Availability

The provided data was historic Dublin Bus data that was recorded for the November period of 2012, and January period of 2013 (Dublinked, 2016). This data would be at the core of the predictive model. Following group research, additional data was acquired from Citizens Information and other government agencies which would aim to help improve the accuracy of the model. The data used in this project is as follows:

- 2012 and 2013 Historic Dublin Bus data.
- Historical weather data for rain, temperature, wind, and weather summary was scrapped in 30-minute intervals from the Wunderground weather API (max API calls, 250/day). Our aim was to include this in our predictive model. However, as the report will justify, this was not feasible or beneficial. (Wunderground, 2017).
- The group sourced additional 2017 schedule data (Gov.ie, 2017) and stop data (Dublinked, 2017). The stop data would later be compared with the 2013 stop data to construct an accurate list of operational stops for our model. It would also provide full stop naming, and more accurate geolocations for all bus stops.
- The Dublin Bus real-time API data was scraped and used to retrieve an estimated time of arrival for buses based on their routes and stops. (Dublinked, 2017).
- 2012/2013/Current School Calendar. (TUI, 2017).
- 2012, 2013, and current Public Holidays Calendar information was compiled. Citizens Information, 2017).

The inclusion of this additional data expanded the scope of our modelling and allowed for the consideration of more variables or scenarios, which had they not been included would have reduced confidence in our model. A brief example of this is taking Public and School holiday information into consideration, as these dates would likely result in lower traffic congestion and bus passenger numbers.

3.2.2 Data Reduction

After establishing our dataset, the team implemented a strategy to model and understand our data. The CRISP DM methodology had been used to support previous project work, and was therefore adopted by the group. This compartmentalised the analytic tasks for our team and set parameters for our data exploration, cleaning, and modelling processes. The methodology devotes separate sections to business and data understanding, data preparation and modelling, and an evaluation period to verify that our developed solution addresses our business model before deployment. It provided an iterative framework for dealing with the data which was noisy, messy, and inaccessible. The project specification and user stories provided the team with clarity and a

business-oriented approach to our modelling, guiding our focus in addressing the concerns of this products main beneficiaries. Darragh our Coordination Lead was responsible the teams modelling.

Data exploration and understanding consisted of various group white-boarding sessions. This provided an environment in which team members could become familiar with the data, identify key Data Quality Issues, and outline strategies for handling issues such as missing values, irregular cardinality, and outliers. It was important to distinguish between invalid and valid data, to therefore able to take the appropriate corrective action. The team dedicated almost two weeks to this part of the process to be confident in their initial approach. Initial exploration of the prescribed historic data identified roughly 83 million cells of data (82,832,358). The first task was to concatenate and compress this data into four files, two for each of the respective years, through Jupyter Notebook. Within data-frames, tables and visualisations were constructed to plot features to better understand their characteristics and record issues to be addressed later in the Data Quality Plan.

Data preparation and cleaning was an iterative process that lasted beyond our fourth sprint. As most of the literature highlighted, our model would only be as good as the data we presented it. It was therefore of key importance to be confident in our cleaning. This process was divided into ten major phases, with each iteration closer to balancing the trade-off between accuracy and computational costs. The group implemented a Data Quality Plan as a strategy to deal with Data Quality Issues. This established modelling parameters and a course of action for dealing with both valid and invalid data.

In relation to missing values, the team applied a complete case analysis approach. If missing values had a significant effect on predictive accuracy, imputation was performed on features with less than 30% of their values affected. For features with more than 30% of missing values, it was agreed that dropping that feature was the best course of action. Outliers can be both valid and invalid data, therefore a delicate approach was required for dealing with them. Thresholds were established for outliers which allowed for the inclusion of minor fluctuations in feature values, but also set a cut-off point for removal of those beyond that threshold. Other criteria for the dropping of features was to study the feature importance attribute. It helped to justify early inclusion or exclusion of features for the model. The diagram below highlights the importance of each short-listed feature.

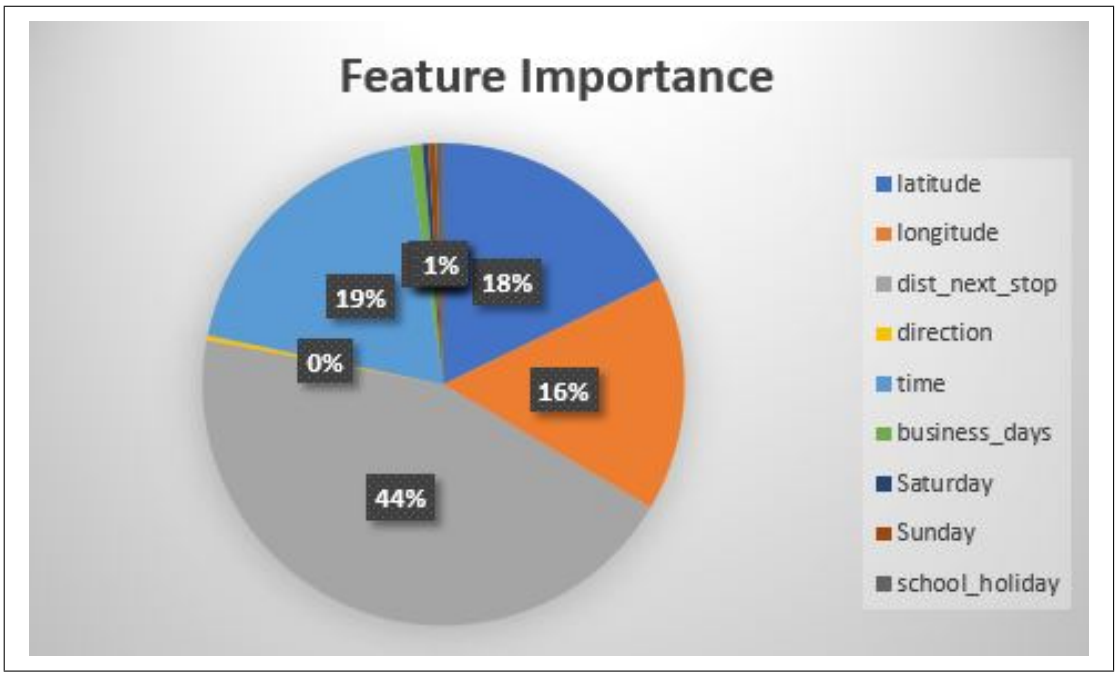

Figure 3.1: Representation of Feature Importance from short-listed features.

After effective cleaning of the dataset, the Random Forest regressor model was implemented. This model was agreed upon, based on findings from the background research outlined in Chapter 2 of this report. Its implementation will be addressed at length in Chapter 5 of this report.

3.3 Application

3.3.1 Overview

As discussed in our projects requirements, our main aim is to identify from the available information; who our customers are and how we could improve on the current benchmark, tailoring an accurate and optimised application for that client-base. Our application was constructed across two main prototype phases. In its most basic format, the application functions through the front-end, whereby the user submits a form which queries the back-end database, prediction model, and API, and returns both visual and textual information tailored for that user.

3.3.2 Front-end

Efficient planning and design gave the group a detailed blueprint to work from. It was decided that the front-end framework Bootstrap would allow us to achieve and implement both our user stories and acceptance criteria. The group decided that while an Android application would have been a significant innovation, in terms of time management the workload to deliver such an application was a black box. Additionally, by focusing our time on making a comprehensive, mobile-responsive web application, creating an Android application would have been a smaller victory for the team.

Regarding the front-end, we reached all our targeted goals. Its design was first drafted via Brackets IDE, allowing for early testing of what features were achievable and what aesthetically would deliver on the team's specification. Extensive research was carried out establishing how in terms of usability the Dublin Bus benchmark compared with other European and United States (US) counterparts. Some features from short-listed websites for Barcelona, London, Zurich, and Washington D.C fed into the group's design and white-boarding sessions.

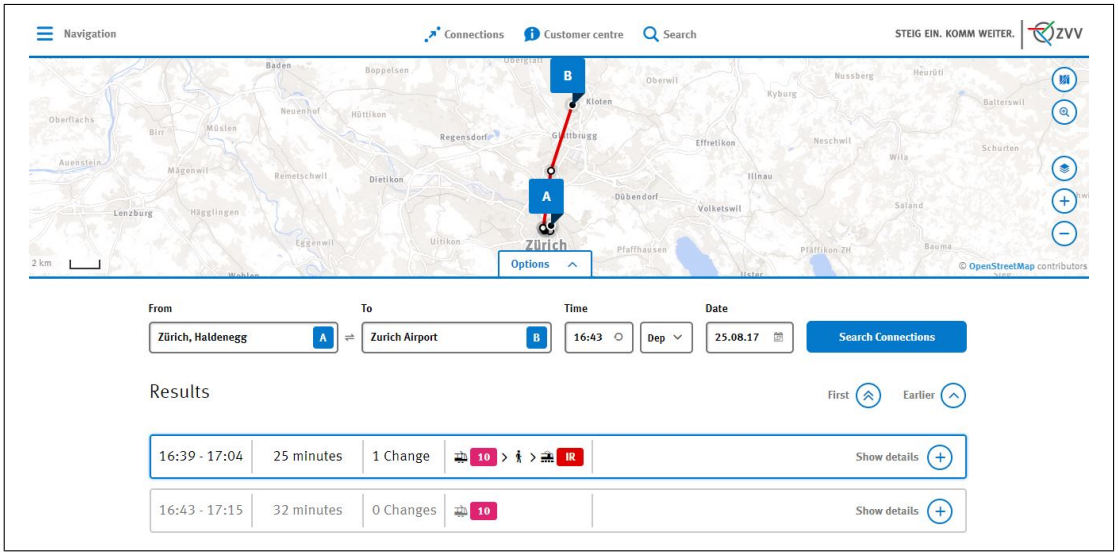

Figure 3.2: Transport web site for Zurich's ZVV .

The developed features of the website became the acceptance criteria for our user stories. This served to help measure our achievements and how we conformed to our requirements.

The application contains six individual web pages; index.html, map.html, tickets_ fares.html, tourism.html, connections.html, and contact.html. All are accessible via the side navigation bar, apart from map.html which can be accessed following submission of the form on index.html.

The core of the application is within index.html and map.html providing the user with data customised to their journey. The forms allow for two methods of input which will be discussed in the following section. A geolocation feature to highlight nearby bus stops and routes was implemented. Map.html provides this same form functionality, should the user want to reset or change their journey without multiple page navigations as experienced in the Dublin Bus benchmark site. There are three secondary pages; tickets_ fares.html, connections.html, and contact.html. These pages provide static, generic, but useful information for the users. They were designed with the purpose of giving the user options, such as other transport methods, service operating hours, and pricing parameters for each type of customer, allowing for alternative planning should the desired bus not be available. In line with the current user stories, we created a tourism page to provide relevant static and dynamic data about events and top attractions in the city.

3.3.3 Back-end

Our back-end design was structured to support this seamless user interface. There are two main components two our back-end: The standard route planner presents the user their predicted journey time from source to destination for that route. The second implementation is for our Plan My Journey/Shortest Path, whereby a source and destination is provided and the route field is left empty, supplying the user with data about the routes that serve their stops and the distance required to travel.

This core functionality is implemented by JavaScript events, an intricate number of python functions stored within Django views, and the running of the Dublin Bus API in tandem with our Random Forest regressor model. The algorithms were designed by our code lead Lucas in conjunction with our adopted requirements.

Inputting a route results in the population of the source and destination drop down menus. Designed specifically to personalise the user experience, selection of a source stop alters the choice of destination stops to allow only the selection of stops following on from the source stop, up until the end of that route. The output format is the same as with the previous algorithm, highlighting the arrival times of the next three buses to that stop, for the selected route. The results are represented both visually and textual on the map.html. Depending on the type of journey selection, different python functions will be called to run the model in the appropriate fashion.

Some of the selected tools were ones with which the group had no experience in implementing. However, in the case of our back-end database we chose to implement a MySQL database. There were perhaps other databases choices such as PostgreSQL, whose geo-spatial queries perhaps may have been more adaptable to our project. However, as the database was going to be an integral part of our project, we chose to implement one with which we were familiar.

3.3.4 Deployment

Our method of deployment was through the Python web framework Django, which was implemented on our UCD-provided Ubuntu 14.04 server. The team had previous experience with the Flask framework, and therefore opted to implement Django to gain new experience.

- Access to the site through its homepage (index.html) is supported across secure networks at: http://137.43.49.41:8000/dublinbuspredict/

Django is structured by two main folders and comes with many default empty files. The first is a directory that stores default installation python files required to run the Django application. The second is the project folder itself, which contains the project's static and template content for the web-pages, styling, and configurable files for content rendering. It contains numerous default files such as models, views, URLs, and tests. This was more accessible than the experience with Flask, with Django also providing a UI for administrative purposes.

The project was hosted by matching URL patterns to requests from the front-end. Each URL pattern matches to a view, or a python function which takes a user requests and gives back a response. Django offers database blueprints and connections through migrations and use of the default models.py file. The team successfully implemented Django for this project. Chapter 5 will focus on outlining some of these issues and strategies team members employed overcome these difficulties.

3.4 Project Management

3.4.1 Agile Methodology

The group followed the Agile methodology throughout the project. In the initial planning phase, the project was divided into 6 individual two-week sprints. While each team member was designated a role within this project, these roles were not strictly adhered to. The initial weeks which encompassed the first sprint were dedicated to team research. Different research topics were delegated to each member. Stand up meetings were regular but often not daily. We spent an extended period working through the project aspects with group white-boarding sessions. This was the most effective strategy for the group as it allowed us to debate ideas, generate a blueprint based on user stories and the project specification, and plan extensively for User Experience (UX) and User Interface (UI) design.

There was daily communication between team members unless otherwise notified. Most meetings took place locally on campus, however remote contact was maintained via Slack, a project development chat application, and Appear, a video conferencing application. Team backlog and progress was maintained on a group Trello account. Each team member was required to regularly update progress, acknowledge when a task had been taken from the backlog, and quality control check other team members progress. This allowed each member to know specifically what other member were working on. There was a designated section for blocked tasks, which was a good strategy for highlighting issues or difficulty within the project.

Group documentation and resources were stored on a shared Google Drive. This contained directories for each sprint and their respective documentation, for group presentations and their resources, and for group information, such as read-me documentation developed by team members.

Frequent recording of progress became a hindrance and quite time consuming as we delved into the core of the project. As the team frequently coded together, by the 4th sprint team members posted updates solely to Slack, rather than maintaining excessive documentation.

3.4.2 Version Control

The group made use of two GitHub repositories. The first was under my account, and can be accessed here. This repository was largely used during the first prototype stage, and does not form part of the final submission. From this, each team member created branches for remote working, with each team member required to push, pull, and delete branches on a regular basis. As the complexity of the project increased, multiple merge conflicts arose within in the group. Some devised strategies to rectify these were to zip and send code for one team member to merge and push. The other team members could then pull the most up-to-date down, rather than filtering through merge conflicts. However much time was being wasted on these conflicts. As a more permanent solution, the group opted to establish a second repository, which can be accessed here, from which all further project work would be pushed to and pulled from. The first repository serves purely for historical reference and for tracking group progress and commits. While not ideal, conflicts are a part of the development process. The important message to take home from this experience, is to always maintain multiple versions of project progress, and to maintain communication with team members as frequent as possible. Basic communication can often prevent issues before they manage to escalate.

Chapter 4: **Personal Contribution**

4.1 Overview

Each team member was assigned a certain role within the group. We were a group of 4 members, and I was the Maintenance Lead. To some degree, our roles were largely blurred. Each member endeavoured to work within an Agile development, where tasks were in a backlog that was maintained by each member. However, as work became more complex, some members were primarily involved in certain tasks, and secondarily involved in others. My primary roles were based on;

1. front-end UI and UX development.
2. maintenance of version control and aspects of project management.
3. research and basic implementation of Django.
4. research and installation of Nginx (web server) and Gunicorn (application server).
5. implementation of web page translations

My secondary contributions were in the data cleaning and preparation phase, and styling the algorithm's output.

4.2 Data Analytics

Along with other group members I was involved in the data exploration and cleaning phases. My specific contribution was cleaning Journey Pattern ID into a usable format. These contained preceding and subsequent zeros. For example, a route such as the 46a was potentially stored as 0046a0000. By developing a python script that sliced these route, I was apply to access routes numbers for use later in the main algorithm.

4.3 Front-end

I was primarily tasked with delivering the UI design and some basic front-end functionalities. I chose to implement Bootstrap through its Content Delivery Network (CDN) version available on the Bootstrap homepage. Neither myself nor any of the team members had previous experience with using this framework. Therefore, the first stage involved researching tutorials, reading through the documentation API, and taking note of potential features that may complement our design. Subsequently, I developed and presented our first prototype to the group.

Bootstrap was effective in that its documentation was accessible and once you had solid foundations, it was relatively straight-forward to build upwards.

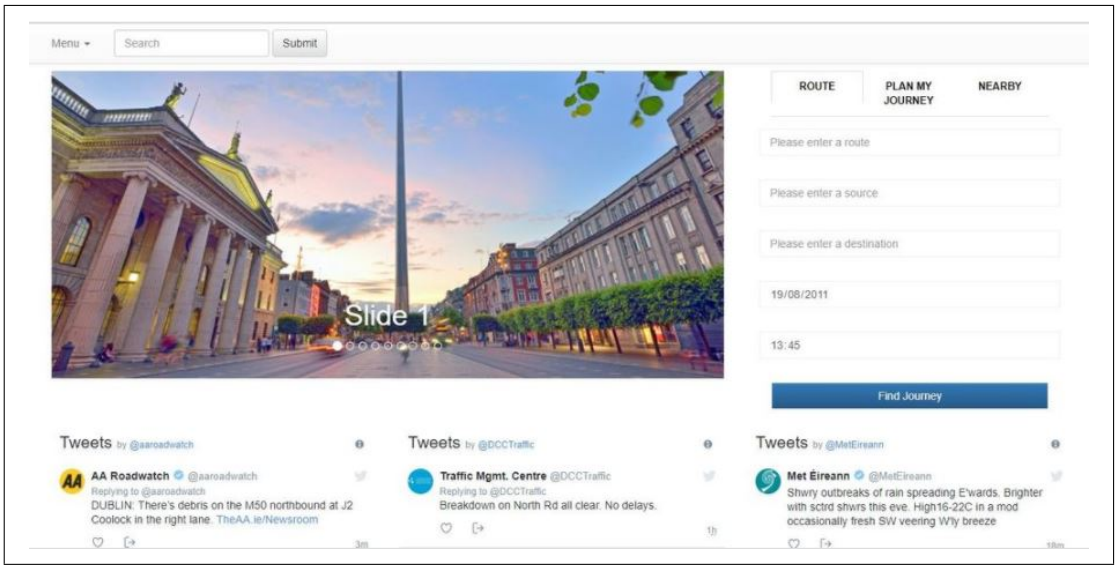

Figure 4.1: Early iteration of index.html from Prototype 1

I implemented basic JavaScript functionalities through its jQuery library for on-click, show/hide, and toggle actions. One of the main issues that quickly arose was that with each iteration and edit of the project's algorithm and back-end, changes in project syntax often resulted in broken code and unwanted edits to the project design. To combat this, in sprint meetings team members discussed any major overhauls to aspects of the project, agreeing to minimise large commits that may be difficult to track changes from.

4.4 Back-end & Deployment

4.4.1 Django

Where this was likely to become an issue was with the groups use of Django, our web framework. As in the case of Bootstrap no team member had any prior experience with Django, which was the main motivation for using it as opposed to Flask. Our early research also indicated it would be better suited to handle larger websites more optimally than Flask. I was responsible for setting up Django locally first, and then it was installed on the group's server.

While it was necessary to delegate some tasks, isolating a key component to one member was not practical. This was raised in some sprint meetings, and to involve other team members in this key process, I developed and uploaded three documents to our group drive. These were two tutorials, outlining the process required to establish your first Django application, and a final document which demonstrated how to run our application. Most of my research focused around YouTube tutorials, Django documentation, and consultation of Stack Overflow for errors that arose. Condensing this information for my team members offered insights into the syntax and structure of Django, and successfully allowed others to follow on from my process.

The key components I was focused on were in understanding its syntax, and then getting our application running. In my case, I firstly established separate python functions so that for each request for a web page, a response for the desired page was sent back to the user. Furthermore, I developed URL patterns which would match to these basic views for each web page. Sample queries were developed to test returning sample data from the database to a HTML page. By

default, Django is connected to a SQLite database which I replaced with our MySQL database through a connection in our settings file. This allowed us to locally establish responses to user requests on our front-end. Django became quite syntactically complex and restrictive far into the project. There was a significant learning curve which perhaps could have been avoided had we opted to use Flask, a framework with which we were quite competent.

4.4.2 Nginx & Gunicorn

Other team members became involved and began to develop some of our core models and queries for application. I was tasked with finding a method to efficiently handle these requests. Django would need an effective web server and load balancer. Previous research and inspiration from group presentations suggested that Nginx could would fit this purpose. No team member had experience with Nginx, and our knowledge was limited with regard to its main competitor, Apache. Where Nginx excelled was in its handling of static and dynamic requests, or namely its concurrent handling of requests and responses. It handled solely requests for static content, passing the burden of dynamic content onto Gunicorn. These requests are divided into numerous worker processes which vastly improved request handling capabilities, simultaneously handling multiple HTTP connections. This was a major benefit, lowering the overhead of handling dynamic requests. However, it is important to highlight that there was involved much configuration between Nginx and its protocols.

4.5 Personal Innovations

4.5.1 Web Application UI

From our design process, I was responsible for the implementation of the team's UI design. The UI consists of six web pages, developed from our nine user stories which are outlined below:

As a user, I need...

1. The journey times from Stop A to Stop B, so that I can plan my journey.
2. To know the arrival times at a bus stop so I know how long I have till the next bus.
3. A platform to schedule my future journey so that I am not late.
4. To see a map of Dublin that gives me visual representation of bus routes, so that I can see what route the bus is taking me.
5. A responsively-designed web application so that I can access the site on my phone.
6. I want to be able to search for a street or use my GPS location so that I can quickly choose an appropriate bus route.
7. To know the journey times from a relevant street to a destination so I can choose the shortest route or an alternative.
8. A simple and intuitive design for OAPs and tourists, so that I can plan my journey around the city
9. To know the cost of my trip so that I can have that inconsideration.

The UX measured whether user stories were accurately implemented and met their set acceptance criteria. Taking the current Dublin Bus site as a benchmark, I focused on improving inefficient features. One key strategy to improve customer satisfaction was to limit the number of clicks required to return results. Multiple clicks are required to return a basic map for a route on the current Dublin Bus benchmark. Our design removed these overheads. A simple submission of a form returns all user-tailored information, with the duplication of the form on the results page map.html, the user could change journey without having to change page.

Previous iterations highlighted that tabbed navigation bars or indeed drop-down navigation bars were cumbersome in the context of our design. The tabular style navigation bar was therefore replaced with a toggle-able sidebar. Ultimately, the side navigation bar left the user with a clear main web-page when the menu was not desired. Our coordination lead Darragh contributed to some of the styling, font, and colour theme of the site. My implementation incorporated mobile and web-devices through its responsive design.

The front-end was also used as part of a strategy to include features which were random or too complex to include in a prediction model. I added twitter feeds to the main index.html page. From the AA Road-watch feed, our team member Peng could slice and select content from tweets only related to Dublin. Initially we had tried to include congestion as a feature in our model, as any form of traffic irregularity was likely to have an impact on journey time. However, its feature importance scored quite low in our prediction model and was therefore dropped. The tweets therefore serve to notify users of any major traffic irregularities or discrepancies with Dublin Bus. They were added to div elements through their individual Ids, and styled to provide accessible updates to users. Iterations of the design were styled by Darragh and I.

Secondary pages such as connections.html and ticketsfares.html, and contact.html were developed to deliver generic static information. I sourced this information from the respective websites of each transport company. Its purpose was to fulfil our user stories that required an intuitive design that could allow for accurate journey planning. Large images did not resize well on mobile devices and therefore cover photos and logos were hidden in responsive mode. This design did not affect the deliverable content and added to the UX.

To serve the tourists that may use the website, I designed and developed HTML and JavaScript pages to display points of interest in the city. Through research of pages such as Trip Advisor, I was unable to source access to a free API with tourist information. As a strategy to overcome this, I compiled an array of the Top 15 tourist attractions in Dublin, and of significant venues around the city. Targeting significant venues was aimed at including venues that were likely to cause a strain on public transport, or result in traffic restrictions. Looping through the locations by their latitude and longitude, markers were output in their respective locations on a Map.

As there were multiple different types of sights around the city, I changed the default Google Map markers to custom built markers with Font Awesome icons. There was an issue with these markers once the user zoomed out. A list of all locations was included to satisfy and appeal to older users unfamiliar with map icons and information boxes. Our team member Darragh connected the page to the database, and focused on adding the 5 closest bus routes that served each attraction.

We aimed to deliver our web application across different languages by implementing a translation tool. Through my research, I came across two options: Google's Cloud-based translation, and their plug-in version. Unfortunately, the Cloud-based version came at a cost and I therefore opted for the plug-in. Through a JavaScript function, I created a drop-down tool bar that would allow translation into any language supported by Google. The tool bar was not aesthetically pleasing, therefore another team member attempted to blend it into our design.

Chapter 5: **Detailed Design & Implementation**

5.1 Front-end

As outlined in the personal contribution section, the UI implementation was a largely straightforward process. Implementing a responsive design for mobiles was the most significant challenge. Additional changes through remote branches by team members often resulted in a broken UI. Excessive use of classes and Id's and inaccurate naming of these classes often resulted in confusion when rectifying a broken design. A costly solution was the intuitive renaming these attributes to simplify the location of further coding errors.

Google Chrome developer tools allowed for progress evaluation. However, following initial deployment there were discrepancies with the responsive design implementation. To identify the source of the problem, on-line sources were consulted to understand if certain JavaScript events, such as on-click, were applicable only to web applications. An extensive code review of each of the back-end functions was carried out to check for syntax or logical errors with the problem eventually being solved. This process highlighted the importance of test driven development. Appropriate implementation of testing can reduce the number of code bugs before deployment. While a time-consuming process, appropriate testing would require less time than searching through code to locate these errors late in the process.

5.2 Back-end

The back-end was implemented through four individual python files; locate_ bus.py, model_ prototype_ 1.py, project_ central.py, and time_ and_ date.py. The implementation of each file is focused on splitting functionality across multiple functions, database querying, try except clause to for error handling, and if/else statements to accurately capture each scenario. Locate bus and project central focus on the first algorithm. Functions were implemented to create a list of stops from the beginning of the line to its destination. The real-time API was queried to locate the closest stop for each of the next 3 buses on that route. From this information, a list of stops is created with the stop_ id, next_ stop, and the status information of each bus. Taking the arrival time at that next closest stop, the model runs through this list of stops and provides a predicted arrival time for each stop and each bus.

The second algorithm's functionality is addressed in the time_ and_ date.py file and focuses on the next three buses. The source arrival times of schedule timetable data is queried and compared to the user input. To get each stop for that trip, the stops with the same trip_ id are queried. The model is run based on the scheduled source stop arrival time to generate predictions for the subsequent stops. The above information is based on the user inputting a date and time for their journey. Should this information be left blank, the algorithm is called to check this information for a maximum of three buses for the next hour. In the case of there only being two buses for this hour, the model is only run twice.

The final algorithm focused on factoring public and school holiday information into the model.

This is completed by cross-checking a compiled list of known public and school holidays. True or false parameters verify if the day is in this list and should therefore be marked as such. If a day is a holiday, for modelling purposes it is treated as a Sunday. To capture this information, days are converted into integers values in a range of 0 to 6; 0 ? 4 refer to business days, 5 refer to Saturdays, and 6 refers to Sundays or holidays.

By compartmentalising each algorithm, the group could achieve the desired results accurately. Iterations to alter these algorithms were therefore not as complex as anticipated, as each aspect was delivered by an individual function.

In the initial stages, there were performance issues associated with the form submission and the handling of model predictions. The case of the forms, there was a cost associated to the loading of the stop information for the source and destination fields. It impacted the overall performance of the web-pages. A strategy to solve this was placing the stops into arrays. Rather than loading all stop information into the fields, only stops following the source were taken from the arrays to populate the destination field. As mentioned before, the model size came at a large computational cost. Our initial prototype involved running the model for each stop in the route, which similarly put a strain on the speed required to return the user's results. To overcome this, the team ran the module once for a route rather than on a stop-by-stop basis. The impact of these minor changes was immediately noticeable, with an average time of 5 seconds for loading predictions onto the map page.

5.3 Deployment

Web servers are of vital importance when deploying an application. For this project, our framework worked in tandem with our web server Nginx to handle requests through the HTTP network protocol. Additional application servers can supplement and improve the management through techniques such as load balancing and session persistence. Our web server Nginx, required significant configuration with our Gunicorn application server to be able to handle user requests. While installation and handling of our application through Nginx was straightforward, configuration of our application server proved to be significantly complex. Two main issues arose: The connection of the server via socket files, and the addition of HTTPS to allow for Geolocation.

Our strategy for connection through socket files involved dedicating much time to shaping our configuration files to follow specific tutorials. Initially implementation was a success, however following on from the inclusion of Geolocation, configuration of the file's server and location blocks was required to allow access to port 443. This was a source of much frustration for team. Conflicts in file configuration hindered the group from deploying the application on the server correctly. By default, Nginx works with the application server and directly parses URIs, therefore much of the required configuration was focused on editing predefined configuration blocks. After dedicating a significant portion of time towards fixing the configuration, the group opted to uninstall and reinstall both the web and application server. Following this, both were successfully installed and configured.

Initial deployment was not successful successful due to issues related to the responsive design. Further investigation into the scope of JavaScript events and how they differ per device was carried out by the team. This did not yield a solution, and therefore an in-depth review of all code pertaining to the forms was undertaken. This is involved de-constructing and reconstructing all related CSS, HTML, and JavaScript . This was highly inefficient but effective. The debugging cost some team members a significant amount of time.

5.4 Analytics

5.4.1 Data Quality Plan Implementation

The implementation of our Data Quality Plan involved the reduction of unimportant features. Features with low importance such as operator, congestion, and latitude or longitude were removed based on this metric. Similarly, features such as Line_ ID and Journey_ Pattern_ ID referred to duplicated information, which therefore supported the deletion of Line_ Id. Journey_ Pattern_ ID was determined a more valuable feature containing access to bus route numbers. These are two isolated instances which directly impacted the quality of our data.

Perhaps the most significant impact was the dropping of at_ stop where the value equalled zero, cutting our data down to just under 19 million cells, at 18,868,110. Most surprisingly, weather data performed poorly in our model. One would assume that this feature would drastically impact journey time and therefore be key to the model. Perhaps the inter-correlation of this feature with others makes it complex for predictions.

Various statistical strategies were employed to the most from the data: Setting a parameter of 500 on the cardinality for columns such as next_ stop ensured that the substantial amounts of data were being recorded for the model. Additional columns such as "duration" and "total_ seconds" were added to list the time between stops and to display an aggregation of this time. Rows that recorded a journey time between stops of over and hour, or less than five seconds, were removed. This was based on a violation of a threshold in schedule data for 2013 which indicated that the longest journey between stops was 27 minutes. This strategy restricted any unwanted alterations of our results, but also left a significant margin of error should some of these outliers be valuable and within reason for consideration.

Separately, normalisation technique such as z-scoring were used to remove outliers from total_ seconds. This has the effect of averaging and standardising extreme values that may skew the overall result. Box-plot visualisations were carried out to verify that any additional outliers were within reasonable conceivable bounds to be included as valid data.

Two new columns, next_ stop and distance_ next_ stop, were added to 2017 Dublin Bus schedule, which made use of the distance field already in the data. A join was made between this and the historic datasets datasets on stop_ id and next_ stop, creating a dataset which only includes fields that match the 2012, 2013, and 2017 schedule. This reduced the dataset further to a size of 2,750,44 rows. Final outliers, such as where duration was equal to 0, and where there was no distance travelled for next_ stop, were removed.

At this point, the group data cleaning will have been completed. The database contains three tables of cleaned information; bus_ stops, bus_ routes, and bus_ timetable. These tables will have incorporated all data taken from the mentioned 2012, 2013, and 2017 datasets. Weather data proved too difficult to model, considering that only information for isolated months was provided. To be able to account for variations in bus usage by day, days of the week were organised into a column, separated as business days, Saturdays, or Sundays, with bank holidays being accounted for as Sunday values.

Missing stops from both 2013 and 2017 datasets were compiled, with 49 in total found and consequentially removed from the dataset. The latitude and longitude were used instead of stop_ id's within the model, as stop_ id's are not linearly assigned. Speed was calculated based on distance and the time taken to travel to the next stop. Dublin City imposes a speed limit of 60km/h, therefore buses above the speed of 80km/h were removed, allowing for some buses who marginally exceed that limit. At this point, all speeds of zero are also removed.

The Random Forest regressor model was based on nine features and a target feature following on from its initial cleaning. The short-listed features were:

- latitude, longitude, distance to next stop, direction, time, business_ day, Saturday, Sunday, and school holidays

The target feature is total seconds. The model is implemented in conjunction with the back-end functions to offer dynamic timing predictions based on these criteria. Throughout the process the were various factors and metrics that would alter the performance of the model.

Firstly was its size. The size of our initial model was close to 3GB. One strategy was the compress the file to a pickle file to decrease computational running costs. A limitation of this was that its functionality was a black box. JobLib, a lightweight pipe-lining tool in python was used to get access to file. The tenth and final version of the model is 84MB. While the files compression contributed to this achievement, significant configuration of estimators and max depth of trees within the Random Forest regressor was required. Max depth controls the size of the trees whereas the number of estimators controls the number of trees.

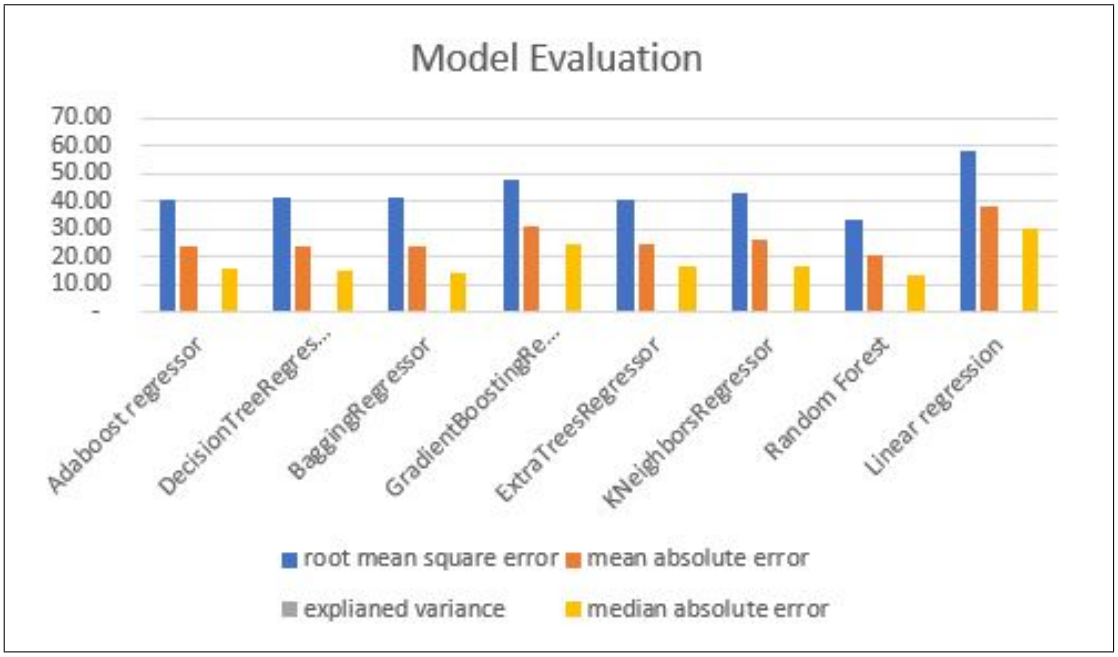

Figure 5.1: Comparison & Evaluation of tested Regression Models

As the chart above indicates, the setting of estimators to 10 and the max depth to 20 balanced the trade-off between the model's size, and ultimately computational cost, and the model's accuracy. The chart also outlines the performance of other tested regressor models. Most notably, the Random Forest regressor has the lowest root mean squared error (RMSE), mean absolute error (MAE), and median absolute error (MDAE). This justifies it as our model choice, producing the lowest margin of error between forecasted and actual values. The only currently identified issue with the model is in balancing the trade-off between computational cost and accuracy.

5.5 Testing & Evaluation

Testing can be divided into front-end, back-end, and analytical validation to verify that the requirements of the product have been delivered. Focusing firstly on the front-end, the main testing criteria were our self-defined acceptance criteria. Due to timing constraints, the team was unable to meet the requirements for the sixth user story:

- To be able to search for a street or use my GPS location so that I can quickly choose an appropriate bus route.

As previously outlined, while geolocation functioned on a local machine, server hosting required specific HTTPS configuration. This was not possible in the time-frame due the inadequate time and task management towards the end of the development cycle. A complete list of user stories was included in section 4.5.1 for cross-referencing purposes.

Appropriate use of if/else statements and try/catch exceptions were the foundation of both front and back-end architecture. This performed accurate error handling to catch all desired functionality. The two core algorithms were measured by testing accuracy, testing performance, and by including error handling capabilities. Testing of the forms with sample input highlighted an initial issue in performance, with the time taken to populate stop information an issue. By limiting the destination stop list to only stops after the source, there was a major improvement in load time. While industry standard testing such as unit tests would have validated that functions were implementing their intended functionality, time constraints did not allow for these.

Regarding the modelling, various estimators and max tree depth values were employed to represent the margin of error in predicting the from the actual values. The model was based on a 70/30 train/test split. In this instance, Random Forest regressor outperforms all the other tested models, given the provided dataset.

Chapter 6: **Conclusion**

6.1 Concluding Remarks

The aim of this project was to design a responsive web-based application that would give Dublin Bus users access to dynamic travel time information. By implementing a model based on cleaned historic and present-day data with the real-time information, our group succeeded and reached beyond our original aims. First and foremost, the application delivers on the core requirement of producing accurate journey travel time estimates. Our Random Forest regressor prediction model has been fitted with cleaned continuous data, with its performance cross-checked with other significant regression models. It performs optimally with minimum computational cost or limitations to accuracy.

Beyond these requirements, the application delivers accurate stop arrival timing, appropriate stage pricing, and total accumulative and stop distancing for the next three buses arriving at a user's stop. It maintains and improves on the quality of the current benchmark's static connection, pricing, and event information. Focusing on the user experience, the application meets the demands of various types of user identified by our user stories. The mobile-optimised version of the site resize depending on the device, maintaining and delivering all of web-based application's core functionality.

6.2 Future Work

Predicting journey time is a challenging problem given the multiple variables that need to be taken into consideration. Our application innovates, incorporates, and builds on established and recognised models, to deliver both a technical and business solution that will improve the accuracy and perception of Dublin Bus. As with the introduction of real-time information, our solution modernises the user experience, but it is not without its limitations.

Increasing and diversifying the data sources that shape the model is at the heart of future development of this solution. The core issue with accurately generating dynamic travel time is not with the service that Dublin Bus is providing, but rather with the data they include with that service. This solution has highlighted that by including important factors such as distance to the next stop, time, latitude, and longitude, the predicted time on a stop-by-stop basis was 23% +- the actual time. It could therefore be argued that the inclusion of additional real-time features, specifically those that would affect a trajectory should be at centre any future work.

One proposal that could be adopted is a data model based on collective awareness through participatory sensing. Rather than AVL sensors, the customer embodies the sensor, delivering real-time quality of service insights. This would drastically increase the quantity of input data. While accessing personal data is still currently a contentious issue, it would allow the users to play a direct role in improving service reliability and efficiency. The EU are currently implementing several collective awareness projects based on creating collaborative, innovative technologies that promote openness, inclusion and participation to counter sustainability challenges.

Crowd-sourcing is a similar adaptation of this model that has helped make business dreams a reality. In the context of bus journeys, perhaps visual or map-based gamification applications or the mining of customer reviews could not only further enhance the quality of service beyond its current form, but also provide the user with some mild entertainment during the morning commute. The future is in the data.

Bibliography

[1] Caulfield. B, O' Mahony. M. A *A stated Preference Analysis of Real Time Public Transit Stop Information, Journal of Public Transportation, Vol. 12, pp. 1-20, 2009, accessed 25/08/2017: available here, 2009.*

[2] Citizens Information. *'Public Holidays', Accessed 25/08/2017: available here, 2017.*

[3] Citizens Information. *Road Traffic Speed Limits Ireland, Accessed 25/08/2017: available here, 2017.*

[4] Data.Gov.IE. *Schedule data for Dublin Bus, Accessed 25/08/2017: available here, 2017.*

[5] Dublin Bus. *Dublin Bus GTFS data, Accessed 25/08/2017: available here, 2013.*

[6] Dublin Bus. *"RTPI Real Time Information FAQs", Accessed 25/08/2017: available here, 2017.*

[7] Dublinked. *Real-time Passenger Information (RTPI) for Dublin Bus, Bus Eireann, Luas and Irish rail, Accessed 25/08/2017: available here, 2017.*

[8] European Commission. *"Digital Single Market: Collective Awareness Platforms for Sustainability and Social Innovation", Accessed 25/08/2017: available here, 2017.*

[9] Gurmu Z. K. *A Dynamic Prediction of Travel Time for Transit Vehicles in Brazil Using GPS Data. A thesis submitted to the department of Civil Engineering Management, A Masters thesis submitted at University of Twente, Accessed 25/08/2017: available here, 2010.*

[10] Harrison. S, Henderson. G, Humphreys. E, Smyth. A *Quality Bus Corridors and Green Routes: Can they achieve a public perception of "performance" of bus service? Accessed 25/08/2017: available here, 1998.*

[11] Jeong R.H. *"The Prediction of Bus Arrival time Using Automatic Vehicle Location Systems Data", A Ph.D. Dissertation at Texas AM University, Accessed on 25/08/2017: available here, 2004.*

[12] Lin, W.H. and Zeng, J. *"Experimental Study of Real-Time Bus Arrival Time Prediction with GPS Data" Transportation Research Record, 1666, pp. 101-109, Accessed 25/08/2017: available here, 1999.*

[13] Murphy. E Sweeney. *Real time bus passenger information take-up and user, Proceedings of The Irish Transportation Research Network 4th Annual Conference, Accessed 25/08/2017: available here, 2013.*

[14] Patnaik. J, Chein. S, and Bladihas. A. *"Estimation of Bus Arrival Times. Using APC Data". Journal of Public Transportation, Vol. 7, No. 1, pp. 1?20, Accessed 25/08/2017: available here, 2004.*

[15] Shalaik. B. *Software for the Control and Analysis of Public Transport, A Ph.D. Dissertation at National University of Ireland, Accessed 25/08/2017: available here, 2012.*

[16] Transport Ireland. *All Bus Stops 2017, Accessed 23/07/2017: available here, 2017.*

[17] Transport Ireland. *How accurate is the information? Accessed 25/07/2017: available here, 2017.*

[18] TUI. *Standardised school year dates 2011/12, 2012/13 and 2013/14, Accessed 25/08/2017: available here, 2017.*

[19] Warman, P. *Measured impacts of real-time control and information systems for bus services. Transport Direct, UK Department for Transport., Accessed 23/07/2017: available here, 2013.*

[20] Wunderground. *A Weather API Designed For Developers, Accessed 25/08/2017: available here, 2017.*