



**DUBLIN INSTITUTE
of TECHNOLOGY**

Institiúid Teicneolaíochta Bhaile Átha Cliath

Formula 1 Prediction and Data Analysis Final Year Project Report

DT228

BSc in Computer Science

Eoin Farrell

Deirdre Lawless

David Carroll

School of Computing

Dublin Institute of Technology

05-04-2014

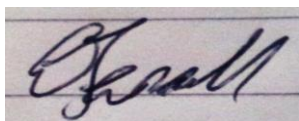
Abstract

The aim of this project is to develop a web application to predict the outcome of a Formula 1 race using previous race results. It will utilise linear regression as the data mining prediction model using data from race weekends including practice, qualifying and race data covering the last four years. The web application will be hosted on Google App Engine and utilise the available Google toolkits and API's. Sports science is a new and expanding field of research in many competitive sports and many teams are now choosing to utilise data mining techniques to improve different aspects of their game such as training, coaching, and selection of player's as well as prediction. As there is no readily available F1 dataset, the dataset will be created from data available from the F1 governing body website along with data stored by Forix, a data collection website for motor racing. The data is collected, cleansed, sorted and analysed using Excel and visual basic code before been uploaded to Google cloud storage, where it can be accessed by the web application. The project will aim towards building a user community around the application with the development of a fantasy F1 style member's league, news section and chat section. Clear tutorials will created for users to allow them to use the prediction model for F1 races.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

A handwritten signature in dark ink, appearing to read 'E. Farrell', is written on a light-colored background.

Eoin Farrell

05-04-2014

Acknowledgements

I would like to thank everybody who helped me throughout this project, providing support on feedback on its design. I would like to especially thank my supervisor Deirdre Lawless who offered guidance throughout the project.

I would also like to thank Erica and my parents Donal and Annette for supporting me throughout this endeavour.

Table of Contents

Abstract.....	1
Declaration.....	2
Acknowledgements.....	3
1. Introduction & Background	1
1.1. Overview of Formula 1.....	2
1.2. Project aims and Objectives.....	3
1.3. Projected Challenges.....	4
1.4. Document Roadmap	4
2. Research.....	6
2.1. Data Mining Overview.....	6
2.2. Data Mining in Sports.....	8
2.2.1. Soccer.....	9
2.2.2. Baseball.....	9
2.1. Datasets & Correlation.....	9
2.2. Data Mining Algorithms	10
2.2.1. Naïve Bayes Classifier.....	10
2.2.2. Linear Regression.....	11
2.2.3. Support Vector Machine.....	14
2.3. Training, Testing & Validation	15
2.3.1. Hold Out Testing	16
2.3.2. K Fold Cross Validation Testing.....	16
2.4. Web Application & User Communities	17
2.5. Conclusion	19
3. Design Methodologies	20
3.1. Methodologies Requirements.....	20
3.2. CRISP-DM.....	20
3.3. Scrum.....	23
3.4. Conclusion	25
4. Architecture Overview.....	26

4.1.	High Level Overview	26
4.2.	Cloud Hosting Providers	28
4.2.1.	Google App Engine	28
4.2.2.	Amazon Elastic Compute Cloud (EC2)	29
4.2.3.	Conclusion	29
4.3.	Data Manipulation	30
4.3.1.	FIA.com	30
4.3.2.	Forix.com	30
4.3.3.	Adobe Acrobat XI Pro	31
4.3.4.	Microsoft Excel	31
4.3.5.	Rapid Miner	31
4.4.	Development Environment	32
4.4.1.	Eclipse	32
4.4.2.	Google App Engine Eclipse Plugin	32
4.4.3.	MySQL Workbench	33
4.4.4.	Sublime Text 2	33
4.4.5.	Dropbox	33
4.4.6.	TeamViewer	33
4.4.7.	Google Analytics	34
4.5.	Utilised Technologies	34
4.5.1.	Java	34
4.5.2.	JUnit	34
4.5.3.	JavaScript Object Notation (JSON)	35
4.5.4.	AngularJS - JavaScript	35
4.5.5.	SQL	35
4.5.6.	Hypertext Mark-up Language (HTML)	35
4.5.7.	Cascading Style Sheets (CSS)	35
4.5.8.	Greensock	36
4.5.9.	Asynchronous JavaScript and XML (Ajax)	36
4.6.	Conclusion	36
5.	Implementation	37
5.1.	Dataset Creation & Evaluation	37

5.1.1.	Data Modelling.....	37
5.1.2.	Business & Data Understanding	37
5.1.3.	Data Preparation.....	40
5.1.4.	Data Modelling & Model Evaluation.....	43
5.2.	Application Development.....	44
5.2.1.	Sprint 1 – Backend Server	44
5.2.2.	Sprint 2 - Front-end Website	57
5.2.3.	Sprint 3 – Testing & Evaluation.....	80
6.	Future Work, Critical Analysis & Conclusion.....	89
6.1.	Future Work	89
6.1.1.	Multiple Linear Regression Algorithm	89
6.1.2.	Reorganise & improve the Member’s Prediction League.....	89
6.1.3.	Standardise and expand Search Database.....	89
6.1.4.	Browser Control Buttons	89
6.1.5.	Improve Google Analytics tracking	90
6.1.6.	Chat function.....	90
6.1.7.	Administrator level user account.....	90
6.2.	Critical Analysis.....	90
6.3.	Conclusion	91
7.	Bibliography	92
8.	Appendix	96
8.1.	Google Analytics Browser Usage.....	96
8.2.	Google Analytics Operating System Usage	96
8.3.	Google Analytics Mobile Device Screen Sizes	96

Table of Figures

Figure 1 SCATTERPLOT SHOWING RELATIONSHIP BETWEEN SHEEP POPULATION AND CHEESE PRODUCTION AGAINST STOCK MARKET VALUE [11].....	7
FIGURE 2 GRAPH SHOWING DATA UTILISED BY DRIVES AND TEAMS FROM THE MONACO GP.....	8
Figure 3 FORMULA FOR PEARSON’S PRODUCT MOMENT CORRELATION [15]	10
Figure 4 FORMULA FOR NAÏVE BAYES [17]	11
FIGURE 5 LINEAR REGRESSION ALGORITHM.....	12
FIGURE 6 LINEAR REGRESSION COEFFICIENTS ALGORITHMS.....	12
Figure 7 PSEUDO CODE FOR LINEAR REGRESSION [20]	12
Figure 8 SCATTERPLOTS SHOWING SIMILAR STRAIGHT LINES OF BEST FIT [19]	13
Figure 9 SCATTERPLOTS SHOWING SIMILAR RISING LINES OF BEST FIT [19]	13
Figure 10 EXAMPLE SCATTERPLOT AND HYPERPLANE CREATED BY SVM [24].	14
Figure 11 THE PSEUDO CODE FOR SVM [24].	15
FIGURE 12 MAE AND RMSE ERROR ALGORITHMS	16
FIGURE 13 WEB APPLICATION ARCHITECTURE OVERVIEW.....	17
FIGURE 14 WEBSITE USABILITY GUIDELINES PROCESS FLOW [30]	18
Figure 15 CRISP-DM OVERVIEW DIAGRAM [34].....	21
Figure 16 EXAMPLE DIAGRAM SHOWING THE PROCESS WITHIN EACH STAGE OF CRISP-DM METHODOLOGY [32]	23
Figure 17 EXAMPLE DIAGRAM SHOWING THE STAGES OF SCRUM DEVELOPMENT AND ITS ITERATIONS [36]...	24
FIGURE 18 ARCHITECTURE OVERVIEW	27
FIGURE 19 INITIAL PEARSON’S CORRELATION TESTING	39
FIGURE 20 RANKING RACE WEEKEND TIMES.....	40
FIGURE 21 EXAMPLE VBA TRANSFER CODE	41
FIGURE 22 OVERVIEW OF SECTION OF MASTER EXCEL WORKBOOK	42
FIGURE 23 EXAMPLE FORIX TRANSFER WORKSHEET	43
FIGURE 24 SEQUENCE DIAGRAM OUTLINING LINEAR REGRESSION PROCESS	45
FIGURE 25 SCREENSHOT OF EARLY TESTING WEB APPLICATION	46
FIGURE 26 JAVASCRIPT CODE OUTLINING AJAX CONNECTION TO PREDICTION SERVLET	47
FIGURE 27 MULTIPLE LINEAR REGRESSION FORMULA & MULTIPLE LINEAR REGRESSION WITH MATRICES FORMULA	48
FIGURE 28 JAVA CODE OUTLINING MATRIX LINEAR REGRESSION	49
FIGURE 29 SIMPLE LINEAR REGRESSION COEFFICIENTS ALGORITHMS	50
FIGURE 30 JAVA CODE OUTLINING CREATION OF SIMPLE LINEAR REGRESSION MODEL	51
FIGURE 31 JAVA CODE OUTLINING THE EVALUATION OF THE MODEL ON TEST DATASET	52

FIGURE 32 JAVA CODE OUTLINING TENFOLD CROSS VALIDATION	54
FIGURE 33 JAVA CODE OUTLINING THE RANK LIKELIHOOD FUNCTION.....	56
FIGURE 34 INITIAL USE CASE DIAGRAM	58
FIGURE 35 FINISHED USE CASE DIAGRAM	59
FIGURE 36 LOW FIDELITY DIAGRAM OF THE WEBSITE	60
FIGURE 37 MEDIUM FIDELITY DIAGRAM OF THE WEBSITE	61
FIGURE 38 INITIAL APPLICATION HOMEPAGE.....	62
FIGURE 39 APPLICATION HOMEPAGE AFTER USER FEEDBACK	62
FIGURE 40 JAVASCRIPT CODE OUTLINING CODE USED TO CHANGE WEBSITE PAGE	63
FIGURE 41 BACK-END SERVER CLASS STRUCTURE	64
FIGURE 42 DATABASE ENTITY RELATIONSHIP DIAGRAM	66
FIGURE 43 SEARCH DATABASE PAGE	68
FIGURE 44 GOOLE USER LOGIN	69
FIGURE 45 USERPREDICTION SERVLET	70
FIGURE 46 GETNEXTTRACE JAVASCRIPT FUNCTION	71
FIGURE 47 MEMBER'S PREDICTION LEAGUE PAGE	72
FIGURE 48 UPDATEUSERPREDICTION JAVASCRIPT FUNCTION	73
FIGURE 49 UPDATE DATABASE PAGE	74
FIGURE 50 SEARCHTRACK SERVLET.....	75
FIGURE 51 CONVERTDATA FUNCTION	76
FIGURE 52 RACE PREDICTION TUTORIAL PAGE	77
FIGURE 53 RACE PREDICTION PAGE.....	79
FIGURE 54 JSONLINEARREGRESSIONTEST UNIT TEST	81
FIGURE 55 USER FEEDBACK FORM PART 1.....	82
FIGURE 56 USER FEEDBACK FORM PART 2	83
FIGURE 57 COLOUR CHART FOR APPLICATION DESIGN	85
FIGURE 58 FINALISED HOMEPAGE	87
FIGURE 59 GOOGLE ANALYTICS SHOWING WEBSITE USE	88

1. Introduction & Background

Formula 1 (F1) represents the pinnacle of motor sport racing, technology and engineering blended into one sport, travelling across five continents during a season, pulling hundreds of millions of viewers from around the world [1]. The roots of F1 lie in pre-war grand prix racing when, in 1950, these races were brought together to form the inaugural Formula 1 championship event. The term “Formula” in the name references the rules and regulations governing the sport set by the sport’s governing body, Fédération Internationale de l'Automobile (FIA). The “1” in the name represents that Formula 1 is the highest level of international motorsport recognised by the FIA.

After every Formula 1 race weekend the FIA release PDF documents containing detailed accounts of all laps performed on the specified weekend along with detailed analysis of any decisions or incidents that may have occurred. This data is held on the FIA’s website until the end of the racing season. From this data, and using data from other sources, it is possible to build a dataset against which data mining algorithms could be run to generate predictions for future races.

Research around F1 generally revolves around the engineering and design of the cars and around the strategy used by teams during a race. Examples of such research include [2] [3]. Timing information is hugely important in F1, with differences between driver times sometimes in the hundreds of a second. Qualifying for an F1 race is determined through three sessions with the fastest single lap by a driver placing them in first position (pole) at the start of the race. This session displays the importance of an accurate timing system to rank the drivers.

From my research into how data mining can be applied to F1 data to make predictions, it was found to have received limited attention to date. A possible reason for this is the work involved in creating a dataset of information from the PDF’s the FIA creates and the fact that they only store the data for year. The reason the FIA do not archive the data for public use is unknown and could be due to the technical changes that come with a new season. These technical changes can be vast and leave the previous year’s data as an incorrect comparison point for timing information and this could be one reason why the data is not archived. As well as this, any data the FIA do provide must be parsed from the PDF format into a more workable and editable format before data mining techniques can be applied to the data. In comparison, other sports (mostly American sports) archive their data and allow easy access to these complete data sets for users. An example of these data sets is available at [4]. This research could offer some insights that would be useful to developing data mining applications for Formula 1. In particular, the usefulness of data mining for a sport which is directly comparable to Formula 1, NASCAR – stock car racing, has been researched. For this

reason, most of my research around data mining in sports will be based upon those which have a significant body of research, e.g. soccer, basketball and NASCAR in particular.

1.1. Overview of Formula 1

Formula 1 has grown since its initial season, hosting seven races with a total of six cars in 1950, to now hosting on average 20 races per season and currently 22 cars with 11 teams [5]. During the 1980's the sport's popularity quickly rose, as Bernie Ecclestone took charge of the sports media rights with Formula One Management Limited as drivers began to become household names. Still to this day, the company and Ecclestone still to this day control the media and advertising side of F1 while the FIA run the rules and regulations side [1].

With every season, a team is in charge of building and designing its car to make sure it adheres to that seasons F1 regulations. With the cost of such developments in aerodynamics, engine performance and electronics, along with the cost of running and transporting the cars and its drivers and team staff, team budgets can range from tens of millions to hundreds of millions of Euro [6]. Each team may have up to four drivers during a season and can swap and change drivers during the free practice sessions of a race weekend.

Each race of the Formula 1 season takes part over three days from Friday to Sunday with five different driving sessions taking place. On the Friday, free practice one and two take place. These sessions last an hour and a half each and allow teams to test various car set ups and allow drivers to become familiar with the track. On Saturday morning, free practice three takes place, during which teams and drivers have their last chance to make and test any final car adjustments. After this, qualifying takes place over three short sessions. The goal during each session is for each driver to place a fast enough lap time to make it to the next round and the final results are used to create the starting position for drivers in the race. The first session lasts 20 minutes and at the end the six slowest drivers are relegated from the qualifying session. In order for the driver to take part in the race, their best lap time must also be within 107% of the fastest lap time set unless exceptional circumstances are determined by the race stewards. Qualifying session two runs in the same manner to the first session but lasting 15 minutes with drivers competing for the fastest lap and again the six slowest drivers are dropped. Qualifying session three lasts 10 minutes between the ten remaining drivers to decide the top ten starting positions for the race, where the driver with the fastest lap award pole position (first position).

On Sunday the race is run with drivers starting in the position determined from their Saturday qualifying session. Though tracks may vary in length, the time to complete a race is usually an hour and a half as all races must last up to the number of laps needed to exceed 305km, with a maximum time of two hours set on a race. During the race drivers may be given penalties for dangerous driving and if a crash occurs the safety car may be deployed to slow the race until all debris is cleared from the track. Along with this, each driver must use two different sets of tires during the race, though teams sometimes determine it beneficial to use more sets of tires to avoid them wearing out. With refuelling now banned from F1, the need to

change tires brings an element of strategy to races as teams decide when best to pit and how often. After the race, each car must return to 'parc ferme', where cars can be tested for irregularities and drivers can be weighed and prepared for post-race interviews and award ceremonies.

After the race, drivers and teams who finished between 1st and 10th are awarded world championship points. This points system, ranging from a maximum of 25 to a minimum of 1 point, determines the driver's position in the championship standings, and also to determine the overall driver and team winners at the end of the season.

As already mentioned, time and the data obtained from the cars is a very important aspect of F1. The timing information from the driver's laps can determine their starting position and factors heavily into the strategy used by teams. This time and is collected by the FIA and any timing information that does not relate to the individual build of the car is released to the public. With this timing information available to the public it can be utilised with data mining techniques. Data mining is a knowledge discovery process to analysis data to find new information. Though teams use data mining techniques, to create race strategies for their drivers, no project in the public domain from my research has focused on predicting the results of F1 races from the available information using data mining techniques, which my project will focus on.

1.2. Project aims and Objectives

The primary aim of this project is to create a web application with the ability to read in previous results from Formula 1 race weekends. These results will then be divided between training and test data and placed into a data mining algorithm which will then output a rank of its predictions for the finish position of drivers. The web application must be able to visualise this process and its results in a meaningful way that is easy to understand by a user with little data mining knowledge.

A secondary aim of this project will be the intention of building a user community around the web application. A web application focused on building a user community can help encourage users to return to the application to interact with the website and find new updated information. To facilitate this tutorials will be created to give an overview of the data mining and the design of the F1 dataset, allowing user to understand how to interact with the prediction model. Further to this and to encourage users to return to the website members sections will be added that will allow users to compete against one another in fantasy F1 style leagues. Other features available to users will be to search driver and track information and provide an engaging home page that will focus their attention on first visit.

The following list of objectives has been created for this project and outlines all the features and work that needs to be completed:

- Research and locate sources of F1 data related to in season and previous season race and pre-race performance.

- Design and develop a dataset to support the prediction of race results from in season race and pre-race performance.
- Investigate the dataset and perform initial testing of the dataset in RapidMiner to establish a liner regression algorithm to be used in prediction of race results from in season performance data.
- Implement linear regression algorithm to be used in prediction of race results from in season performance data as part of a prediction application using appropriate development techniques.
- Investigate requirements for, design and develop web application to allow interaction with prediction algorithm using an appropriate development method.
- Expand application to provide features to support a user community interested in F1 and in interacting with the F1 race prediction algorithm.
- Plan and execute testing of the prediction algorithm by this user community to investigate its usefulness to this community and amend the application to reflect this feedback as needed.
- Plan and execute usability testing of the website to gauge problems users may have and amend the website to reflect the requirements as needed.

1.3. Projected Challenges

The main challenge that I will confront while developing this web application will be to learn and understand how to implement linear regression correctly so as to generate reliable predictions that match the predictions already made by Rapid Miner. The creation of the linear regression algorithm will be the centre point of this project and for this reason will be a challenge to implement.

The creation of the dataset and the collection of timing information for the prediction of F1 race results will be a long process that could overrun and effect other areas of the project. Along with this learning and understanding how to use the cloud hosting environment and its API's will be an important task that may cause problems with the transfer of data from client to server.

1.4. Document Roadmap

Over the next five chapters I will detail the development of the project from initial research to final usability testing. Each chapter is summarised below.

Chapter 2 will detail the prior research that needed to inform the development of the regression algorithm and the application to support the user community. It will cover the area of sports data mining, the building of user community groups, the design of a dataset and the data mining prediction algorithms research for this project.

Chapter 3 will focus on outlining the project development cycles and methods that will be followed during the development of the application. This process will introduce a structured approach to development.

Chapter 4 will introduce the tools and technologies that will be utilised throughout the development of this application. It will also give an overview of the cloud hosting that the project will utilise and the development environment of the project.

Chapter 5 will focus on the core development of the application, this will include the creation of the F1 dataset and the linear regression algorithm, development of the web application and the full testing process that will be utilised throughout development.

The concluding chapter, Chapter **Error! Reference source not found.**, presents the challenges I faced during the development of this application, how I dealt with these and alternate solutions I considered or could consider in the future. The chapter then outlines the development that I plan to complete in the future. The chapter will conclude by presenting an overall summary of the project and its achievements.

2. Research

This chapter will focus on the research undertaken before beginning the development of the data mining project. The chapter will begin with an overview of what data mining is, how it can be used and a discussion of its popularity and the pitfalls due to a lack of knowledge this can bring. The chapter will next outline how data mining can and already has been utilised in a variety of sports, with example sports discussed. The focus of the chapter will then move on to how to implement data mining, how to create the dataset, outline possible data mining algorithms and how to test the data mining model. The chapter will then conclude with research into web applications and how to develop for a user community driver application.

2.1. Data Mining Overview

Data mining is the process of knowledge discovery from a database of information. Patterns and trends are analysed from the data using concepts taken from artificial intelligence, machine learning and statistics. One of the first areas of data mining use was in calculating credit risk within the financial sector in the 1960's when Fair and Isaac used statistical analysis to understand, recognise and attempted to predict this credit risk using collected data from application forms and behavioural data [7]. Since this first incarnation of data mining, it has evolved to match the increasing computing power available to users and has spread to many industries outside of the financial world, including market research, stock markets, research science and many other areas [7]. In recent years, with increased understanding of data mining and its usefulness, increased availability of large datasets of information and increased access to data mining tools has led to data mining becoming an increasingly popular field of exploration in both industry and education.

In recent years, Big Data has become the go to buzz word within media. SAS identified Big Data as a "popular term used to describe the exponential growth and availability of data, both structured and unstructured" [8]. Without data mining techniques then, big data is just a large dataset of information without the ability to inform us of the hidden patterns and sequences that may be hidden within. As an example, this shows us the need for data mining as a toolset within the business world, as big data is an important area of research there and brings with it the opportunities of "greater operational efficiencies, cost reductions and reduced risk" [8]

For data mining to make predictions on a dataset, a statistical model is created that "reflects the important aspects of the object of study with some degree of realism" [9]. The model therefore creates a prediction by exploring the relationship between the explanatory variables (input variables) and the relevant response variable (the predictor/outcome variable/race result). There are four types of explanatory variables when detailing a dataset [10]:

- Predictive variable: An explanatory variable that provides an insight into the response variable, thus is fitted for prediction
- Interacting variable: An explanatory variable that by itself does not provide any insight into the response variable, but when combined with another variable, known as a computed variable, provides an insight to the response variable, thus is fitted for prediction.
- Redundant variable: An explanatory variable that does not offer additional information that has already been provided by another explanatory variable. Characterised when has a very high correlation with another explanatory variable.
- Irrelevant variable: An explanatory variable may or may not display a correlation to the response variable but is one that a domain field expert believes does not affect the outcome of the response variable.

A scatter plot can be created to show the relationship between these variables and how a line of best fit would work in the data, allowing as many of the data points to be as close to the line as possible [9].

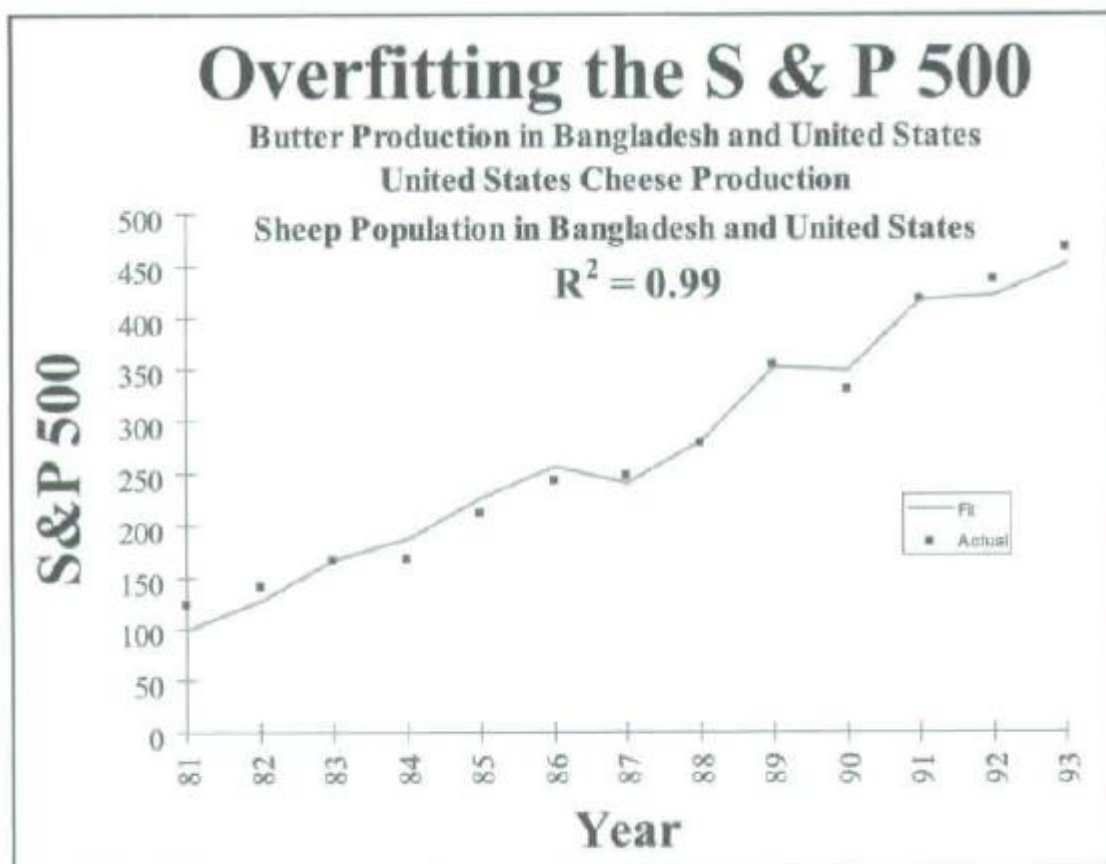


FIGURE 1 SCATTERPLOT SHOWING RELATIONSHIP BETWEEN SHEEP POPULATION AND CHEESE PRODUCTION AGAINST STOCK MARKET VALUE [11]

When designing a data mining project, it is necessary for a data analyst to examine these models and the explanatory variables which are used to determine if the two variables do logically relate to each other, or if it is just luck that allows a statistically significant model to

be created for the given time period of data. As shown in the above scatter plot, the author was able to generate a line of best fit, with an accuracy of 99% using the sheep population of Bangladesh and the US, cheese production in the US and butter production in Bangladesh and the US as explanatory variables against his response variable of the S&P 500 index. He was able to use these datasets (1983 – 1993) to create his predictive model, but as stated in the document, outside of this timeline, this model would be useless as there is no direct relationship between the variables outside of the dataset. The explanatory variable in this experiment can be considered an irrelevant explanatory variable as discussed previously. The documenting of this analysis of data mining is used to give a brief overview of the pitfalls of data mining and how it can be used incorrectly, if not examined correctly.

2.2. Data Mining in Sports

As mentioned previously data mining with regard F1 race results has been limited from my research but F1 teams place high importance on the data they collect and how they use it. F1 teams rely heavily on data during a race and along with this FIA data, teams rely on over 150 sensors that measure and log details such as speed, oil temperature, aerodynamics, tire degradation, etc. [12] [13]. During a typical race these sensors will generate over 25GB of data. When this is compounded with information from five sessions per weekend, 22 race weekends for the 2014 season and preseason practice sessions for 22 cars, it leads to a large amount of data generated throughout the year. This data is used by the teams mostly to design race strategies that can adapt to changes during a race. Though data from these sensors is not publicly (thus not relevant to my project), this information shows how important data and timing information is within the world of F1. The below image displays an output of some of these sensors that the drivers and team analysis.

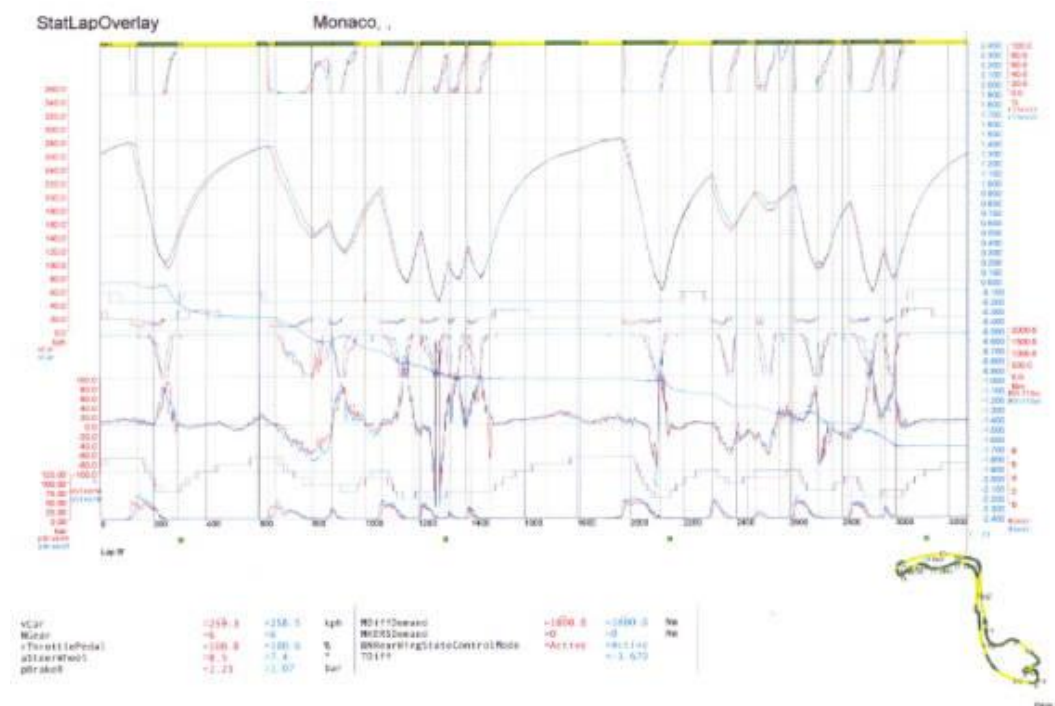


FIGURE 2 GRAPH SHOWING DATA UTILISED BY DRIVES AND TEAMS FROM THE MONACO GP

Sports science is a new field of research in many competitive sports and many teams are now choosing to utilise data mining techniques to improve different aspects of their game such as training, coaching, and selection of player's as well as prediction [14]. The below examples show how data mining techniques can be utilised in sport to predict not only results but player injuries and players' abilities to fit into a team.

2.2.1. Soccer

AC Milan, a professional Italian club has begun using a biomedical tool developed by Computer Associates that allows them to predict player injuries during a season [14]. The system is fed each players medical statistics along with their performance from each training session. When a player has fallen below the baseline expectations for that training session the system raises an alert that the player may have sustained an injury or that an ongoing injury has worsened. Due to the cost associated with player injuries and time out from playing this software is used by AC Milan to save money and improve player's match time.

2.2.2. Baseball

One of the most famous instances of data mining in sports is told by Michael Lewis in the book MoneyBall [14]. It tells the story of how the manager Billy Beane of the Oakland A's baseball team relied upon data mining techniques to pick players for his team based on previous performance. The idea went against traditional baseball management as Beane sensed current scouts would only pick players who played the game a certain way, indifferent of other players proven performance. With a low cost budget Beane was able to pick and choose players that were highlighted to him through data mining knowledge and create a team on a small budget that was able to compete against bigger clubs such as the New York Yankees.

2.1.Datasets & Correlation

One of the most time consuming processes in undertaking a data mining project is creating the dataset of information. This dataset can be created from a variety of sources, it could be scraped from the web, it could be data pulled together from multiple systems or it might already be readily available to the data analyst. This process can take time as the data may have to be collected, cleansed of errors and transformed to the correct format before entering the dataset.

Once the dataset is collected, the next step is to understand the data that has been collected. Without examining the variables held within the dataset, we may include variables in our data mining techniques and algorithms that do not help and may in some cases cause our model to be less accurate, outlining the importance of selecting the explanatory variables. For this reason, Pearson's Product Moment Correlation is used to measure the correlation (measuring how well the variables are related) between the explanatory variable and response variable. The formula generates results that can range from -1 (perfect negative correlation) to 1 (perfect positive correlation). A relationship of 0 states no relationship between the two

variables thus the greatest variation is found between the data points of the variables. [15]. Below is the formula for Pearson's Product Moment Correlation:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

FIGURE 3 FORMULA FOR PEARSON'S PRODUCT MOMENT CORRELATION [15]

Without prior knowledge of how the data domain functions an analyst may not realise the potential for new computed variables and how they may relate to the response variable in a meaningful way. Likewise a data analyst may incorrectly believe some variables have a relationship to the response variable of the dataset due to using Pearson's Correlation algorithm showing a high correlation, when in fact the explanatory variable is an irrelevant variable with no actual link to the response variable. This outlines the importance when creating a dataset and using correlation functions that the data analyst is familiar with the area or has access to a domain expert.

2.2. Data Mining Algorithms

This section will give a brief overview of some of the data mining algorithms investigated for this project along with an explanation on how they work and a brief discussion on why I choose the algorithm I used for this project. Along with this I will give a brief description of the methods used to train these algorithms and the validation that is required.

2.2.1. Naïve Bayes Classifier

The Naïve Bayes classifier is based on applying Bayes' rule (from Bayesian statistics) and is classed as a simple probabilistic classifier (probability statistics) and assumes independence of the explanatory variables (variables do not relate to one another, thus the algorithm is naive) [16] and is widely used in text recognition and spam filtering [17]. It was identified as one of the top 10 data mining algorithms by the Institute of Electrical and Electronics Engineers Association (IEEE) from a three stage survey aimed at industry professionals [17]. The survey found it to be popular due to its relative simplicity to implement allowing even unskilled users in classification algorithms to understand its process in classifying variables.

The independence assumed by naïve Bayes may seem like a downfall for the algorithm but when combined with datasets that have had redundant explanatory variables removed through the use of attribute selection procedures [16] it becomes a powerful method of data mining. Using a supervised learning setting (learning a labelled training dataset), these classifiers can be trained very effectively allowing the algorithm to learn the maximum likelihood of a response variable given the particular feature set of the class. One problem with naïve Bayes is if a particular attribute does not appear in the training set but appears in the test set, then it will set the response variable to 0, no matter the value of other explanatory variables. This problem can be avoided through minor tweaks to the way it

evaluates probabilities from frequencies. Though naïve Bayes is one of the oldest algorithms used in data mining, it can still be relied upon to generate useful outcomes and is still widely used due to its simplicity and accuracy [17].

To compute the probability of a sample set of explanatory variables being a part of a class of response variables the following formula is used [17]:

$$\ln \frac{P(1|x)}{P(0|x)} = k + \sum_{j=1}^p w_j,$$

FIGURE 4 FORMULA FOR NAÏVE BAYES [17]

- $P(i|x)$ defines the probability of the response variable in terms of the explanatory variables x_1, x_2, \dots, x_j
- The training set is divided into two different sets with $i = 1$ for higher values and $i = 0$ for lower values
- k stands for the probability of the response variable equals to an undetermined class when $i = 1$ / divided by the probability of the response variable equals to an undetermined class when of $i = 0$ and no other information (explanatory variables) n are available
- w_j calculates the conditional distribution for x in both sets i and then divides the result for $i = 0$ by the result of $i = 1$

2.2.2. Linear Regression

Linear regression first appeared in 1927 in a paper on sun-spot analysis and has since been adopted for analysis in neuro-physics, seismology as well as speech communication [18]. Linear regression is considered to be a staple method in the foundation of statistics and is a method that is ideal for use when the response variable is numeric [16]. The idea behind linear regression is the idea of letting the response variable equal to a combination of the explanatory variable given a pre-determined weight. From this linear regression models the relationship between these variables and the response variable to create a line of best fit (which ideally creates a straight line). This line of best fit attempts to reach as many of the data points as possible, or if not, to get close to the majority of data points as possible. From this understanding we can say the line of best is the average relationship between the two variables with some variation around it. When multiple explanatory variables are used it is known as multiple linear regression [9]. Linear regression calculates an adjusted R^2 , which represents the variance around the average fit. This tells us how many of the data points are covered by the line of best fit. Linear regressions by nature though suffer from the idea of a linear line, as the dataset may not be constructed in a linear fashion and thus a best fitting straight line may not be representative of the entire population of data points.

The model for Simple Linear Regression is:

$$Y' = B_0 + B_1X + e$$

FIGURE 5 LINEAR REGRESSION ALGORITHM

- B_0 represents the intercept of the y axis (unknown)
- B_1 represents the slope (unknown)
- e represents a random error component that is the distance between the data points and the line. These errors are assumed to have an average of 0 along with the assumption that the errors do not relate to one another. As the data points cancel each other out to give an average of 0, the error component is usually dropped from the formula
- X represents the explanatory variable
- Y' represents the predicted response variable

The parameters B_0 and B_1 are known as regression coefficients with the slope of B_1 equal to the change in the mean distribution of y by the change in x [19]. If the dataset of x contains $x=0$, then the intercept of the line on the y axis is stated by Montgomery [19] as “the mean of the distribution of the response y when $x=0$ ”. The formula below outlines the equation as described for the slope B_1 and the intercept B_0 .

$$B_1' = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2}$$

$$B_0' = \bar{y} - B_1'\bar{x}$$

FIGURE 6 LINEAR REGRESSION COEFFICIENTS ALGORITHMS

Let \mathbf{X} be a $p \times q$ feature matrix.

Let \mathbf{Y} be a corresponding $p \times r$ label matrix.

Let \mathbf{M} be a $r \times q$ matrix.

Let \mathbf{b} be a r -dimensional vector.

Initialize \mathbf{M} with small random values.

Initialize \mathbf{b} to $\mathbf{0}_r$

$\eta = 0.05$

for i from 0 to 20

```

.   for each row  $\mathbf{x}_j \in \mathbf{X}$  in random order
.   .   for each  $y_{j,k} \in \mathbf{y}_j$ 
.   .   .    $e = y_{j,k} - (\mathbf{m}_k \cdot \mathbf{x}_j + b_k)$ 
.   .   .    $b_k = b_k + \eta e$ 
.   .   .   for each  $x_{j,l} \in \mathbf{x}_j$ 
.   .   .   .    $m_{k,l} = m_{k,l} + \eta e x_{j,l}$ 
.    $\eta = 0.85\eta$ 
```

FIGURE 7 PSEUDO CODE FOR LINEAR REGRESSION [20]

The below figures show examples of scatter plots with the explanatory variable marked with the response variable along with the line of best fit shown [19]. These four examples illustrate

how the data points can vary within a dataset but output a similar line of best fit. The first scatterplot in figure 6 shows a distribution of data points that are all close to the line of best fit with only a small variance in distance between the closest point to the line and the furthest point to the line. The second scatterplot in figure 6 displays the same line of best fit but with a different data set. In this example there is a higher distribution between the data points leading to a larger variance in distance between the closest point and the furthest point to the line. These two examples show how different data sets can give a similar line of best fit as their average values match up but their data point distribution can vary greatly. Figure 7 displays a similar situation with a variance between the data points of scatterplot one and two while displaying the same line of best fit but with an upwards direction.

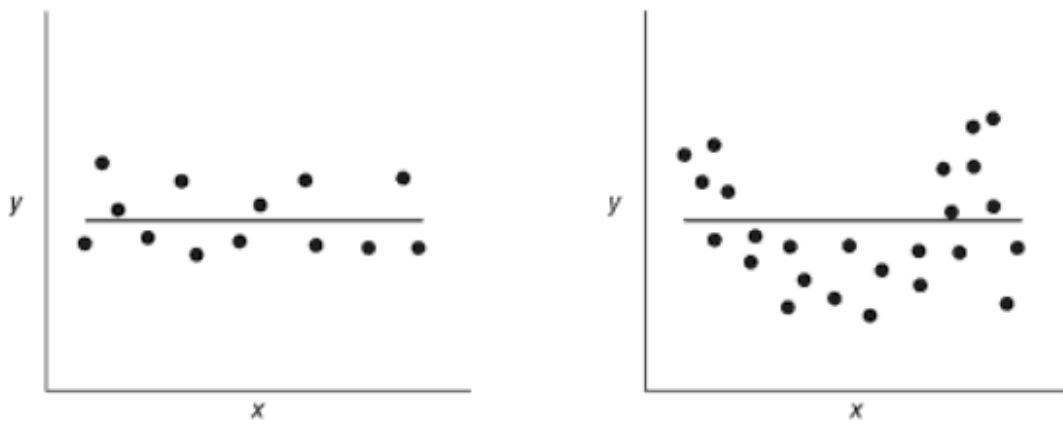


FIGURE 8 SCATTERPLOTS SHOWING SIMILAR STRAIGHT LINES OF BEST FIT [19]

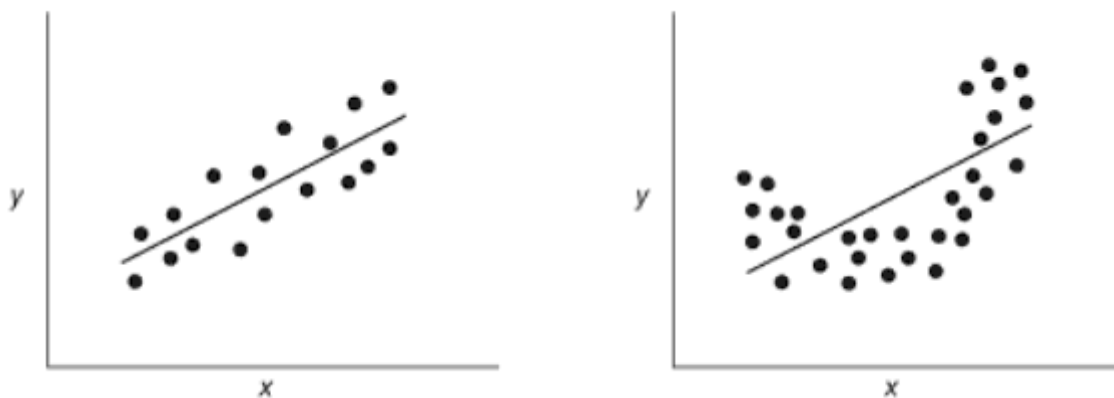


FIGURE 9 SCATTERPLOTS SHOWING SIMILAR RISING LINES OF BEST FIT [19]

Logistic regression, an extension of linear regression was also analysed briefly for the research of this project [21]. Logistic regression operates in a similar manner to linear regression, applying a line of best fit to the data to make predictions but focuses on data that contains a binary response variable. A binary response variable is a variable that can only be true or false. As the F1 dataset contains perfectly linear data values ranging from 1 – 22/24 for each race and not a binary response variable, it was decided logistic regression was not suitable. If the

prediction question was altered to determine if a driver would finish a race or not (true or false), then logistic regression could be considered.

2.2.3. Support Vector Machine

Support vector machine (SVMs) is a relatively new algorithm in the field of data mining first appearing in 1995 [22] and is considered to be a must-try, due to its accuracy and robustness due to only needing a small training dataset [17]. The disadvantage though for SVM is its computational inefficiencies requires a lot of power to run the algorithm and without adaptation it does not match well with training data that is not linearly separable. SVM has shown to be useful in the areas of handwriting recognition, image classification and bioinformatics analysis [23].

SVM is built around linear modelling and instance based learning by creating a linear hyperplane between two sets of data classes with a margin both sides of this line and as stated by Kotsiantis [24] finds the “largest margin either side of the hyperplane that separates one class from another”. When this is created, given a linear dataset, only points that lie on the margin are represented while all other data points on the graph are disregarded; these points are called support vector points. This means that SVMs chooses a low percentage of the data points available and is therefore more suited to datasets that contain a high number of explanatory points [24] [25].

Without adaptation, SVMs can sometimes fail to create a hyperplane if data is found to contain “misclassified instances” [24]. To overcome this problem SVMs can be altered to create soft margins that allow misclassified data through the use of positive slack variables.

The below figure shows a scatter plot of the explanatory variables and response variables. The hyperplane is situated in the middle with margins on either side. It also highlights the support vector points that are along this margin.

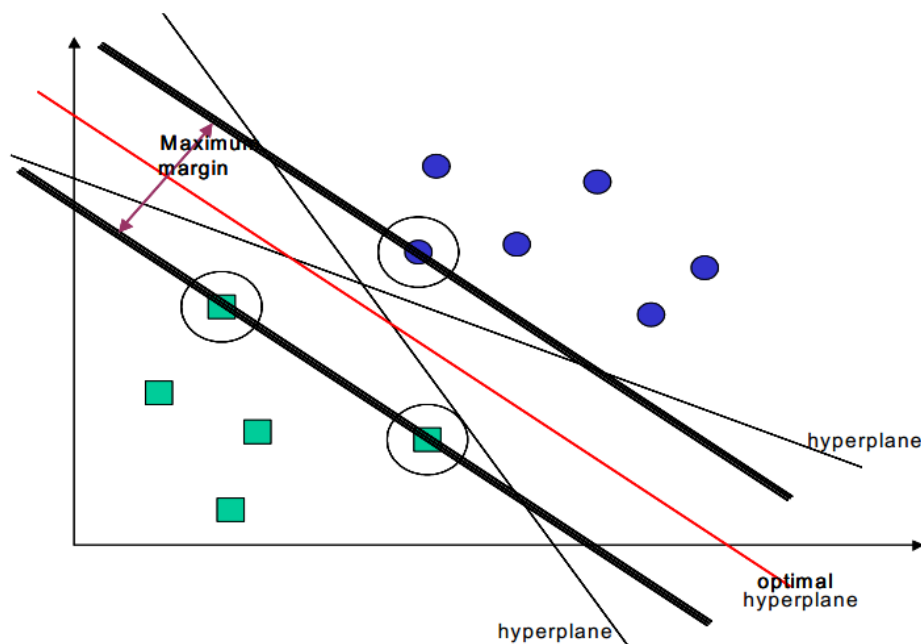


FIGURE 10 EXAMPLE SCATTERPLOT AND HYPERPLANE CREATED BY SVM [24].

1) Introduce positive Lagrange multipliers, one for each of the inequality constraints (1). This gives Lagrangian:

$$L_P \equiv \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \alpha_i y_i (x_i \cdot w - b) + \sum_{i=1}^N \alpha_i$$

2) Minimize L_P with respect to w , b . This is a convex quadratic programming problem.

3) In the solution, those points for which $\alpha_i > 0$ are called "support vectors"

FIGURE 11 THE PSEUDO CODE FOR SVM [24].

2.3. Training, Testing & Validation

When running data mining algorithms for predictions the data is divided into two parts, training data and testing data. Training data includes your background data, data where you already know the outcome/response variable. This data allows the chosen algorithm to learn the type of data it is dealing with, along with learning the patterns the data holds and how these patterns affect the response variable. The larger the training data set, the more accurate a model will be, as bad data and unusual data will have less of an impact on the model. As a ratio for the time period you want to predict, you should have a ratio of 3/1 for training data against testing data [26]. Simplified in regards my project, this means that if I want to predict the results of one F1 season, I should aim to have, at least, the data from the previous three seasons to compute this. The algorithm I choose will then be able to learn from the three season's worth of data, what explanatory variables contribute to the outcome of a race such as free practice results and qualifying result.

After a base model has been created and before it is applied to the test data, validation can be used to judge the models accuracy or error rate and potentially improve its performance [16]. Although the error rate generated from testing the training data is generally not accepted to be a good indicator of the model's accuracy on the test data as the training data may not be a good representation of what the testing data contains, it is still a useful bit of information to understand.

Two error evaluation statistics were investigated for this project, root mean squared error and mean absolute error [27]. These two determine the error rate by averaging the distance from the actual response variable to the predicted response variable. RMSE squares each

iteration of this distance, and then gets the square root of the average distance. In contrast MAE gets the absolute value of each iteration (negative numbers are transformed to their positive value) before getting the average distance. As MAE uses the absolute distance it is less susceptible to large differences that may square the error rate. The two formulas are displayed below:

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

FIGURE 12 MAE AND RMSE ERROR ALGORITHMS

To perform these error evaluations two types of testing were researched for this project that can be used on the training data, hold out testing and cross validation testing.

2.3.1. Hold Out Testing

Hold out testing takes one portion of the training data and uses it for training the model and then the remaining training data is used for testing against this model to determine its error rate before it is used on the test data. Generally one third of the training data is set aside for testing and two thirds for the training [16]. The error rate can then be determined from this subsection of training data used for testing purposes. To safeguard against uneven representation of the test data, it is good practice to use stratified hold out testing, which randomly selects the data to use for testing purposes. The disadvantage of this method is that ideally you want a large size training set, the larger a training set the more accurate a model can be created, although as stated by [16] “the returns begin to diminish once a certain volume of training data is exceeded.” If the dataset you are working on is not of a vast size, then hold out testing can waste training data for testing purposes, meaning you are not maximizing the use of data available.

2.3.2. K Fold Cross Validation Testing

Cross validation testing is an alternative to hold out testing and benefits from maximizing the use of the data available for testing purposes, though can be more computationally intensive. Cross validation testing randomly divides the data into K folds. Data of size K-1 is then used for training while the remaining sample is used for testing, this process is then repeated K times. This allows multiple versions of the training data to be used for both testing and

training and allows all sections of the training data to be used as testing data at least once before creating the model. Stratification can also be used for cross validation testing to randomly select the K folds from the training set. Ten is generally considered to be the ideal number for K. Furthermore it is standard to run the cross validation testing ten times on the training data and averaging the results to get the error rate. This leads to a total of 100 times the algorithm is invoked on the training data when used with 10 fold cross validation, showing the computationally intensive side to the algorithm.

2.4. Web Application & User Communities

A web application is a website that allows users to query a database of information to dynamically serve and generate content to the client [28]. This style of website allows users to choose the data they wish to see, dynamically creating the website around them. A variety of different web technologies can be utilised in the development of a web application including HTML, CSS and JavaScript on the client side and Java servlets, PHP and SQL on the server side. The user's web browser is a key component in the creation of a web application as it intercepts the user's interaction with the web application. From this it determines the calls to the server, thus defining the dynamically returned content. The below image illustrates the typical architecture of a web application.

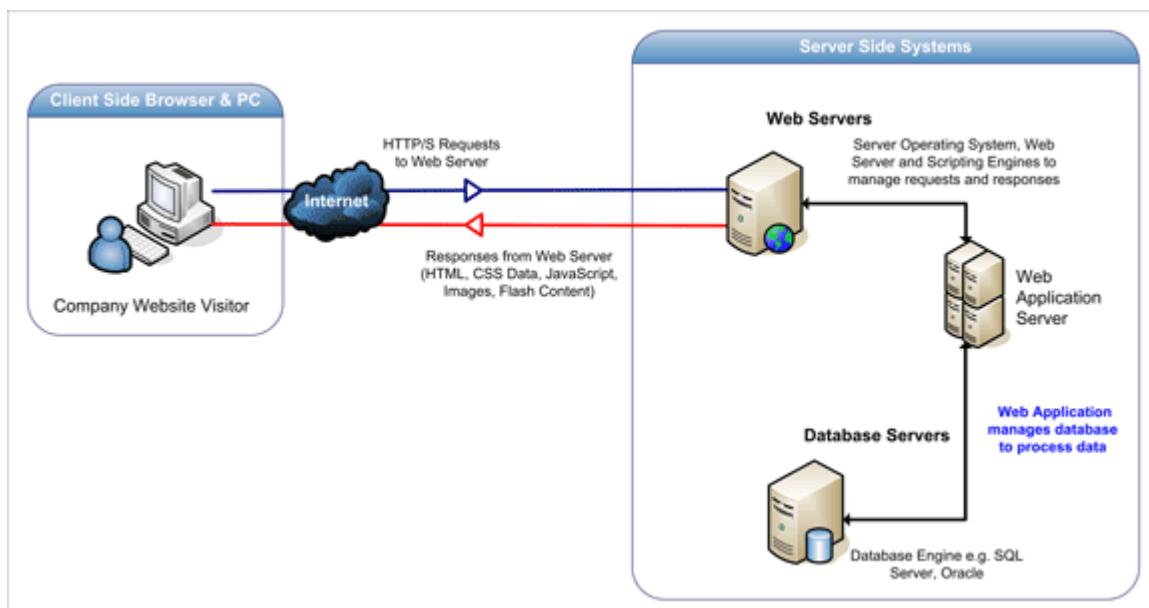


FIGURE 13 WEB APPLICATION ARCHITECTURE OVERVIEW

A disadvantage to web applications is the increased security measures that need to be taken to secure sensitive data that may be held on the database of the server. Web applications that have not been designed with security in mind can be problem for its users, as their data may be at risk.

A user community based web site is a web site that encourages user interaction with it and other members. This type of website encourages users to return to the website, to find new content regularly [29]. This can be facilitated through the development of interactive forms

and competitions on the website that allow users to compete against one another. This allows users to interact with one another through indirect communication. To expand upon this allowing members to directly communicate with each other through forums and chat features can help to grow the community interaction. Providing users with content about current trends and topics, that are relevant to your web site, can help gather discussion with your members. During the development of any website especially a community website it is important to follow website design guidelines to encourage these users to return and not be put off by poor design. The US government provide a full design guideline that can be followed for this [30]. The diagram below outlines a step by step process flow that can be followed in the development of a web application. To understand how this diagram may be relevant to my development, F1Predictions.net was analysed as a F1 website focusing on user interaction through prediction's and competitions [31].

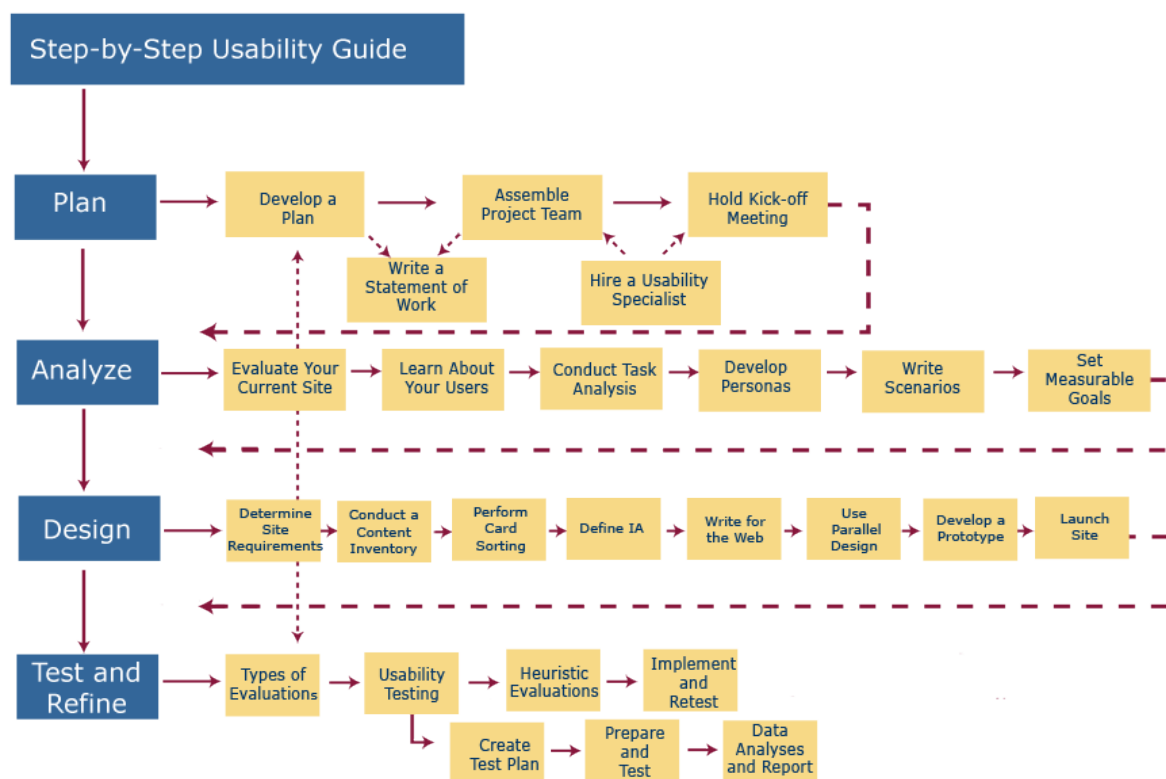


FIGURE 14 WEBSITE USABILITY GUIDELINES PROCESS FLOW [30]

[HTTP://WWW.F1PREDICTIONS.NET/](http://www.f1predictions.net/)

2.5. Conclusion

From my research I have chosen to implement linear regression as my choice of prediction algorithm. I chose linear regression as it has a long history within statistics and data mining and due to this there is a large amount of research and documentation around it that I can reference and help me build my prediction algorithm. As stated before linear regression is designed to work best when the response variable is a numeric value and for this reason it is well suited to my project.

My research on web applications and designing one for the purposes of user interaction, has laid out clear points that I need to address during the development cycle of the application. This includes focusing on providing user content and allow user interaction within the application and to develop it with a focus on data security.

3. Design Methodologies

As my project is both a data mining project and a computer science development project, two separate design methodologies will be used. Initially the focus will be on the data mining side of the project so Cross Industry Standard Process for Data Mining (CRISP-DM) methodology will be utilised. After the data has been processed into a useable form and initial testing has been done using data mining toolkits, the project will move towards the development role and Scrum will be utilised.

3.1. Methodologies Requirements

When researching for a design methodology for the creation and initial model testing of my dataset, the main requirement was that it would support the development of the dataset and allow me to recognise and elevate problems easily. The use of a data mining process would also allow the division of the workload into specific sections, allowing data to be filtered out as a better understanding of the data is achieved. Overall these requirements of the data mining methodologies would improve my time management by outlining the workload needed to be completed in the correct order and prevent errors from passing through the development as they are highlighted in the initial data understanding stage. Two data modelling process methodologies were researched, CRISP-DM and Sample, Explore, Modify, Model, Assess (SEMMA).

An endless number of design methodologies exist for software development. The requirements for a development methodology for this project would be to facilitate a quick development cycle, allowing new versions to be pushed to users for testing. The methodology must also avoid a linear approach to development, allowing the development to revert back to previous stages when the need arises. As new versions are pushed to the user for testing, the methodology must also facilitate a reflection period on the feedback received and allow this feedback to be carried into the next iteration of the application. These requirements and outlining of stages in the development process would allow easy interaction with users, highlight problems with the design both during planning and after implantation and would improve time management as the most important aspects of the project are addressed. Two development data modelling process methodologies were researched, SCRUM and waterfall model.

3.2. CRISP-DM

For this project I will be following a specific data mining methodology, CRISP-DM. CRISP-DM was developed with cooperation from a number of data mining companies, with the goal of creating a tool agnostic data mining methodology [32]. This is in contrast to SEMMA which was developed by SAS and was aimed at data mining using its software application toolset.

SEMMA was not chosen for this reason, as I will not be using any SAS applications during development.

CRISP-DM describes the commonly used steps undertaken by data miners to complete a relevant task and has been known as the “de facto” [32] standard for the industry by analysts. According to a 2007 poll [33] of data analysts CRISP-DM was ranked as the most used data mining methodology. In section two of this report I have outlined some of the detailed processes that I am undertaking, with this assignment that fit into the CRISP-DM structure.

CRISP-DM is not a strict form of design methodology, as it believes moving back and forward between states is necessary. The outcome of one phase thus influences the next phase to be performed. CRISP-DM also illustrates the necessity for a data mining project to constantly evolve, even after a solution has been deployed. From the initial deployment, new business questions or scenarios may arise from the results, allowing subsequent data mining processes to benefit from the initial results [32]. CRISP-DM outlines the following structure:

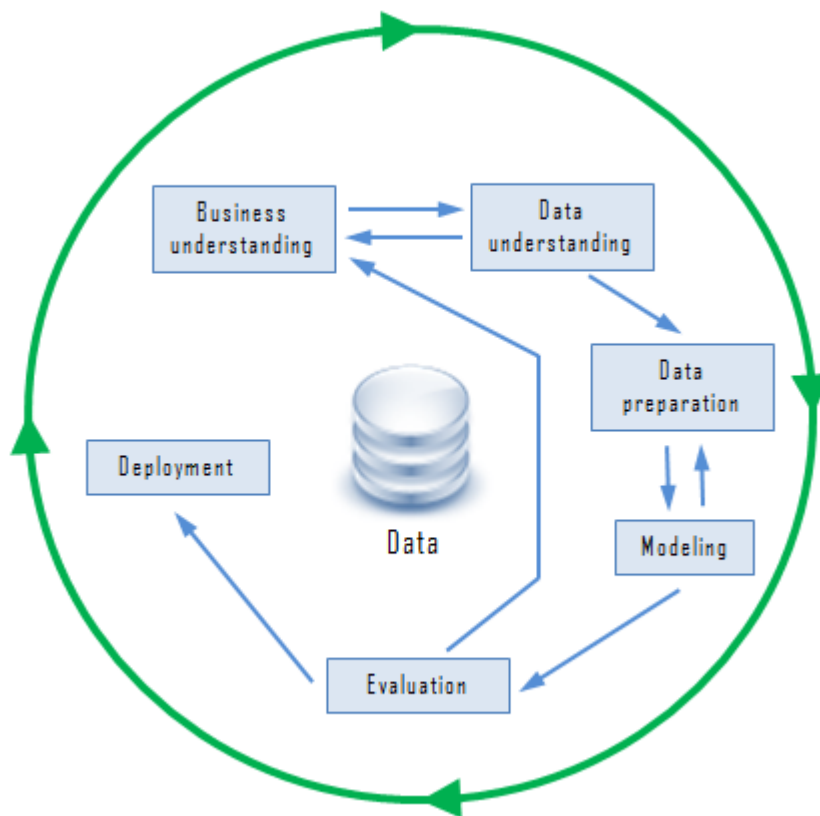


FIGURE 15 CRISP-DM OVERVIEW DIAGRAM [34]

Business Understanding

The first step in CRISP-DM is to create a data mining problem definition. This can be achieved through analysing and gaining an understanding the business and project objectives. Task overview:

- Determine Business Objectives
- Assess Situation

- Determine Data Mining Goals
- Produce Project Plan

Data Understanding

This stage introduces data to the project. The first step is the data collection. Depending on the source and quality of the data, the data collection can take a lot of time. After the data has been collected it must then be analysed to understand the quality of the data at hand, any errors that may be present within the data and any initial insights that could be gained from examining it. Task overview:

- Collect Initial Data
- Describe Data
- Explore Data
- Verify Data Quality

Data Preparation

Data preparation is the final stage in completing the dataset before it is put into a modelling tool. After understanding and analysing the data, new computed columns may be found that are relevant to the model and can be added in. Data may also need to be cleansed in this section (cleaning data of any errors or bad data) and transformed (converting datasets to allow use in the modelling tool). Task overview:

- Select Data
- Clean Data
- Construct Data
- Integrate Data
- Format Data

Modelling

At the modelling stage the dataset is placed into the data mining toolkit, where upon the chosen modelling technique can be run on the data. Movement between this stage and the data preparation stage is common as the dataset design may change as new styles of models are applied to the model. Task overview:

- Select Modelling Techniques
- Generate Test Design
- Build Model
- Assess Model

Evaluation

Before deploying the now complete model, it needs to be analysed first, to determine if it meets all of the business objectives as outlined in stage 1. To do this I will need to review the steps undertaken so far to create this model so as to ensure it meets the business

requirements. From this, it may also be possible to generate further ideas relevant to the dataset that can be generated, either before or after deployment. Task overview:

- Evaluate Results
- Review Process
- Determine Next Steps

Deployment

After the model has been created and evaluated it then must be organised and presented in an intuitive manner to the end user. This stage can vary within the time scale as it may require a simple report or could be feeding a large website or program. For this reason it is important to understand how to correctly present the data to the end user. Task overview

- Plan Deployment
- Plan Monitoring and Maintenance
- Produce Final Report
- Review Project

Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment
Determine Business Objectives <i>Background Business Objectives Business Success Criteria</i>	Collect Initial Data <i>Initial Data Collection Report</i>	Select Data <i>Rationale for Inclusion/Exclusion</i>	Select Modeling Techniques <i>Modeling Technique Modeling Assumptions</i>	Evaluate Results <i>Assessment of Data Mining Results w.r.t. Business Success Criteria Approved Models</i>	Plan Deployment <i>Deployment Plan</i>
Assess Situation <i>Inventory of Resources Requirements, Assumptions, and Constraints Risks and Contingencies Terminology Costs and Benefits</i>	Describe Data <i>Data Description Report</i>	Clean Data <i>Data Cleaning Report</i>	Generate Test Design <i>Test Design</i>	Review Process <i>Review of Process</i>	Plan Monitoring and Maintenance <i>Monitoring and Maintenance Plan</i>
Determine Data Mining Goals <i>Data Mining Goals Data Mining Success Criteria</i>	Explore Data <i>Data Exploration Report</i>	Construct Data <i>Derived Attributes Generated Records</i>	Build Model <i>Parameter Settings Models Model Descriptions</i>	Determine Next Steps <i>List of Possible Actions Decision</i>	Produce Final Report <i>Final Report Final Presentation</i>
Produce Project Plan <i>Project Plan Initial Assessment of Tools and Techniques</i>	Verify Data Quality <i>Data Quality Report</i>	Integrate Data <i>Merged Data</i>	Assess Model <i>Model Assessment Revised Parameter Settings</i>		Review Project <i>Experience Documentation</i>
		Format Data <i>Reformatted Data</i>			
		<i>Dataset Dataset Description</i>			

FIGURE 16 EXAMPLE DIAGRAM SHOWING THE PROCESS WITHIN EACH STAGE OF CRISP-DM METHODOLOGY [32]

3.3. Scrum

For the development and design of the project I will be using Scrum methodology. Scrum was chosen as it does not fixate on a strict linear approach to development unlike the waterfall model. The waterfall model follows a straight timeline from concept development through to testing and does not facilitate freely moving from one state to a previous state. As Scrum is based on the Agile approach of development, it allows this free movement from state to state. This is important during development as if a user raises a small problem that hinders the use of the application, the development stage can be quickly reverted to, to alleviate the problem

without having to wait till the complete process restarts. This was one of the main reasons for the choice of Scrum over the Waterfall model.

Scrum design methodology was designed to overcome the shortfalls associated with development while using agile methodology. One of the main area's it attempted to improve upon from agile was placing emphasis on meeting deadlines when stakeholders apply pressure to the development [35]. Scrum revolves around the idea of empirical process control, this is to say Scrum does not attempt to use a best guess or forecast to plan release schedules, instead it uses the real progress of the project to determine its schedule. The development period is divided into sprints, which are short periods of development usually lasting between one and three weeks. Before these Scrums begin detailed requirements are outlined by the developers and stakeholders. When a sprint finishes the developers and stakeholders meet again to discuss the progress achieved during the sprint and agree with what the next step of the project is. These regular meetings allow the project idea and schedule to remain on course and adjust to both internal and external changes while reducing speculation around what the projects aims and schedule are. During these meetings, development back logs are outlined and sorted in order of priority. During the sprint the goal is to work through the back log starting with the most important items and completing their implementation.

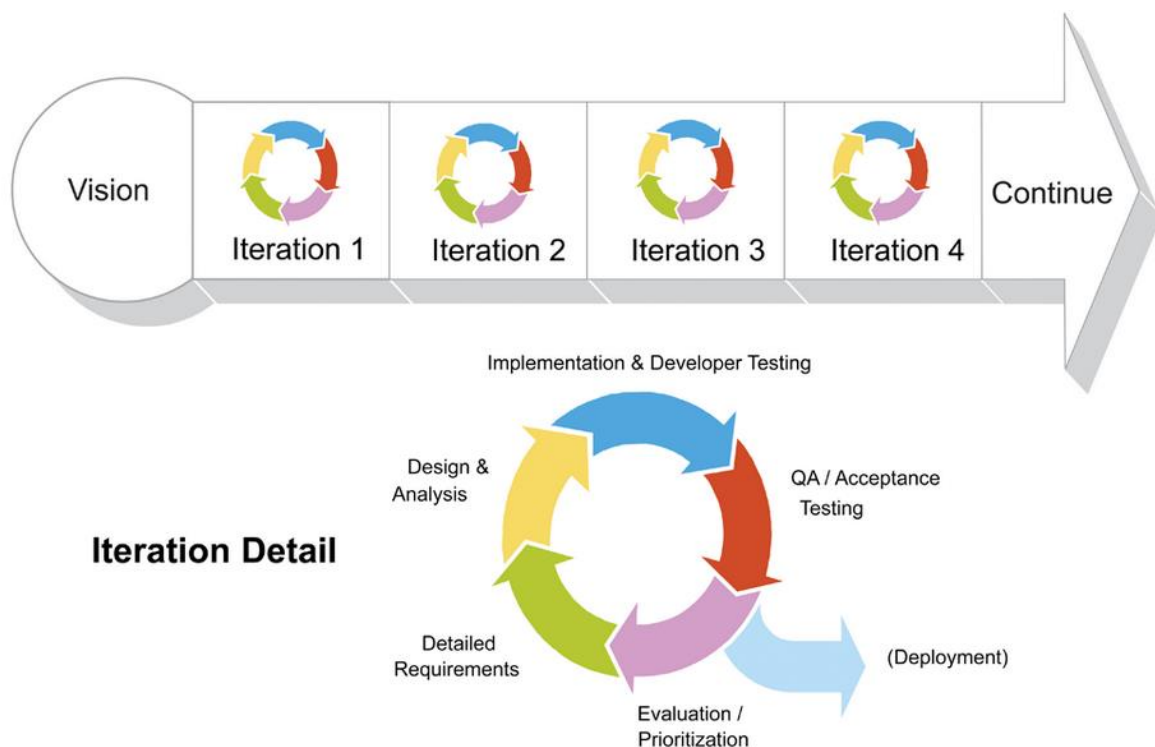


FIGURE 17 EXAMPLE DIAGRAM SHOWING THE STAGES OF SCRUM DEVELOPMENT AND ITS ITERATIONS [36]

Within the Scrum there are three main roles, the product owner, the Scrum master and the team members along with end users and external stakeholders outside this scope. As this is an individual project some aspects of Scrum will be adopted to meet my needs such as this division of roles as I will fit into both the product owner and the development team.

3.4. Conclusion

This chapter focused on discussing the methodologies that will be utilised during the development of the application. Requirements for the methodologies was outlined with explanations on why each was chosen along with a comparison to alternative methodologies.

4. Architecture Overview

This section describes the architecture of the system and the technologies chosen to implement it. A high level overview of the architecture of the system is first given before detailed analysis of the different hosting packages available are introduced and described before a choice is detailed. The chapter finally outlines the tools and technologies that are utilised throughout development.

4.1. High Level Overview

The architecture of this system will revolve around the use of a cloud hosting package to run the data mining algorithms and then push these results in the form of web pages to the user. Cloud hosting has been adopted for this project as it allows me to create and host the project at a reduced cost to traditional web hosting, along with not having to worry about underlying servers and technologies. A third benefit to cloud hosting is the ease of increased scalability when needed, if a large amount of user attempt to access the site. Starting out this may not be important to this project, but in the long run it is advantage to have in case of increased user flow. The development of this project will be undertaken in Java as it is a popular programming language with easy to use documentation and has history in the creation of data mining programs such as Weka and RapidMiner [37] [38]. The prediction algorithms will sit on an instance of one of the cloud hosting packages detailed below. The app engine will feed users with the web application that they can interact with. Users can then call the App engine server to start the prediction algorithm on the F1 data that will be held on the cloud instance with the web application. Once the prediction algorithm collects its results they will be presented back to the user. The below diagram gives an overview of the complete architecture of the system including the production and development environments and methods in which users interact with the system.

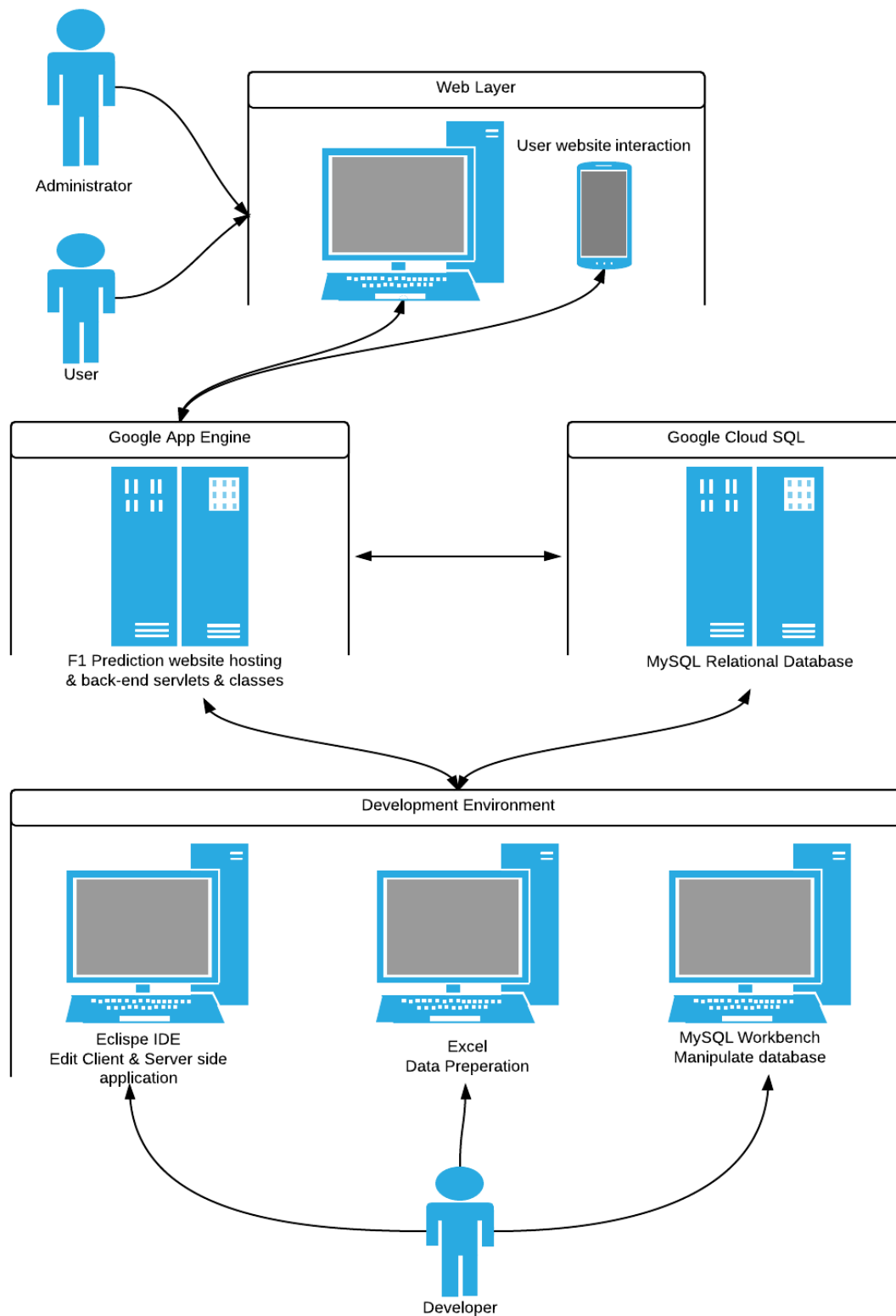


FIGURE 18 ARCHITECTURE OVERVIEW

As can be seen from the above diagram, the application will be developed to support three different stakeholders. The developer of the system, will interact mostly with the development environment, the development environment covers the data mining and data management workflow along with the workflow of developing and creating the application.

When the developer has completed a new development version they can push it to the production environment app engine and database. The user interacts with the application from their computer, possibly a desktop, laptop, phone or tablet. The user is the day to day person who visits the website to find new information and query the database for information. The administrator interacts with the application in a similar methods to the user, but has access to additional tools. These additional tools allow the administrator to control the content on the site.

4.2. Cloud Hosting Providers

The first requirement for the cloud hosting was an ability to support Java as a development language. After this a cloud hosting package that allowed easy access to additional API's, specifically a user login API was desired, to allow easy expandability in the future and to offset the requirement of creating a secure user sign up system with the necessity to store sensitive user information. Also expected from the cloud hosting package is an easy to use system with a focus on development rather than hosting and environment management. An additional bonus is the cost of the cloud hosting and access to a possible free tier. Amazon Elastic Cloud (EC2) and Google App Engine were chosen as cloud hosting packages to investigate, as the two largest cloud hosting packages that support Java programming unlike Microsoft Azure which is aimed towards .Net/C# developers.

4.2.1. Google App Engine

Google App Engine [39] allows developers to create applications to run on servers provided by Google without having to complete the work or have the knowledge required to set up virtual machines, load balancing, storage space and scaling to increased user demands. App Engine works on the promise of Google handling these different tasks themselves so that developers can focus on creating and improving their web apps in the knowledge the backend is handled by Google themselves. This brings many advantages such as the stability and consistency Google is known for with handling services such as YouTube carried into App Engine. Along with this Google focus on the security of you application and automatically encrypt your application data and have achieved various security certificates for their App Engine platform (ISO 27001, SSAE-16, SOC 1, SOC 2, SOC 3, etc.). Google App Engine also blocks external requests to your data, removing part of the security threat raised in the research chapter.

App Engine provides developers with an appspot.com web address to host their web app and if required allows this to be changed to a custom domain name. App Engine provides a choice of programming languages and in the case of Java allows developers to use standard Java toolsets such as Java servlets.

<https://developers.google.com/appengine/?csw=1>

<http://aws.amazon.com/>

<http://azure.microsoft.com/en-us/>

Google also allows developers to integrate other Google API's into their web app such as allowing users to login into your web app with their Google id. Task scheduling and dynamic web serving for common web technologies such as HTML, CSS and JavaScript is also provided.

Storage is provided to users in three different forms:

- App Engine Datastore: NoSQL schemaless object database
- Google Cloud SQL: Standard relational SQL database based on MYSQL.
- Google Cloud Storage: Storage of objects and files in bytecode, up to terabytes in size.

4.2.2. Amazon Elastic Compute Cloud (EC2)

Amazon EC2 [40] takes the opposite approach to Google App Engine as it offers developers the possibility to endlessly customise the back end of their web application. Amazon EC2 allows developers dictate their choice in virtual machines (VM), software, hardware, security and load balancing. This choice gives developers access to any programming language they wish to install on their VM along with other software tools and the choice of hardware configurations including GPU compute instances that are not available on Google App Engine. EC2 allows developers complete control over all of their EC2 instances with root access to all VMs'. Through EC2 developers are also given access to the many other web services that Amazon run through various API's, this includes the ability to allow users login through their Amazon, Facebook or Google account. Similar to Google App Engine, Amazon EC2 allows developers to deliver dynamic web content using common technologies using their CloudFront service.

Storage is provided to users in the following forms:

- Amazon Simple Storage Service: Web interface to store any type of data
- Amazon Relational Database Service: Choose from MYSQL, Oracle and Microsoft SQL database management engines
- Amazon SimpleDB: Core database functions
- Amazon Simple Queue Service: Store messages sent between computers to control data transfer between different instances

4.2.3. Conclusion

For my project I have chosen to use Google App Engine. I chose Google App Engine due to its simplicity and though I can see the advantages of Amazons cloud hosting package and the ability to fine tune the individual instances to suit a development project, my web application does not require this complexity and customisation. App Engine allows easy interaction with a vast array of Google API's that can help expand my project in the future. For storage purposes Google Cloud SQL will be utilised to store the prediction datasets along with other relevant information on drivers, tracks, etc. As both cloud hosting packages have free tiers that should be enough to handle my development this factor was disregarded when choosing which package to use.

4.3. Data Manipulation

This section outlines the tools and websites used during the creation and initial testing of the F1 dataset, to be used for predicting race results from previous race information times. This includes programs and websites that were used during the data preparation stage to select, clean, construct, integrate and format data. A number of tools and websites were required for this including, a reliable source of data, an application that would allow data to be stored, cleansed, possible intermediate programs to support the transfer of the raw data to the storage application and finally an application allowing access to a number of data mining algorithm to test the dataset against. As data preparation can be one of the longest stages in a data analytics, any programs that improve this would be beneficial.

4.3.1. FIA.com

During the data understanding stage of this project, FIA.com, the official website of the sport's governing body was used to gather data from most recent season, 2013. FIA.com was the first choice for a reliable source of data as they are the F1 governing body and thus are in charge of all timing information. After every race weekend the FIA as the governing body of F1, release PDF documents that detail all race events, decisions and times. This can be upwards of 50 PDF documents per race. For this project only the timing information from all 5 of the race weekend events, free practice 1, 2 & 3, qualifying and the race, is relevant. This data was later used during the data preparation stage to assist building the dataset for the application. Unfortunately FIA.com only provides these PDF documents for the current racing season. This meant that during the data preparation stage another data source would need to be utilised in order to develop the F1 dataset. Utilising multiple sources of data while creating a dataset can lead to problems but as the FIA is the only primary source of F1 timing data, any other data sources used would be based on information provided by them. This would avoid the problems of inconsistencies of data appearing.

4.3.2. Forix.com

Forix.com, run by AutoSport magazine, is a data collection website for many racing series including F1. Many alternative websites were found, however Forix was chosen due to its easy to use lay out, its substantial collection of historic data and its reputation as it is used by EuroSport, a large sports television network as their data set for racing analysis. The downside to Forix was the lack of an easily downloadable format, so data had to be instead copied and pasted into Microsoft Excel before cleansing.

<http://www.fia.com/>

<http://www.forix.com/>

4.3.3. Adobe Acrobat XI Pro

Adobe Acrobat XI Pro is Adobes premium PDF viewer, available through their creative cloud service. XI Pro allows users to view, edit and create PDF documents. XI Pro was utilised as a data transfer tool and was chosen as it is developed by Adobe, the original developers of the PDF file system. Acrobat XI Pro was used to convert the PDF race documents released on FIA.com into individual Excel files which could be easily editable. Batch processes were set up to run this conversion as 550 PDF documents were obtained from the FIA for the 2013 season.

4.3.4. Microsoft Excel

Microsoft Excel is a spreadsheet program bundled with Microsoft Office, that has become the industry standard for spreadsheet programs as it is used in businesses across all sectors to manage, manipulate, understand and question data. Microsoft Excel was chosen for these purposes due to its popularity and having used it extensively in the past along with a knowledge visual basic for applications code (VBA). During the data understanding stage of this project and data preparation stage of this project, Microsoft Excel was the main program used to understand the data and to clean, construct, integrate and format this data to construct the dataset for the application. To achieve this an array of Excel functions were used along with VBA code. The use of Excel functions and VBA code, would help to reduce the time needed to construct the dataset.

4.3.5. Rapid Miner

Rapid Miner is an industry leading data analytics and data mining program and was voted the most used analytics program used by data analysts in 2013 [41]. With no previous experience using a data mining tool kit, an easy to use application was the main requirement. Rapid Miner was chosen due to its popularity and from its ease of use when testing it against alternatives such as Weka. After creating a sub sample of the dataset, Rapid Miner was utilised to understand the data and to assess the viability of the Linear Regression algorithm on the dataset as Rapid Miner provides Linear Regression methods to test against datasets. Rapid Miner was also used to test the viability of cross validation and Naïve Bayes on the dataset.

<http://www.adobe.com/ie/products/acrobatpro.html>

<http://office.microsoft.com/en-us/excel/>

<http://office.microsoft.com/en-us/excel/>

4.4. Development Environment

This section outlines the tools and programs that are used during the development process of this application. This includes programs used to develop the backend framework of the application, including running of the linear regression algorithm, tools to develop the front-end framework of the website, data storage environments and website monitoring tools. For this development a list of tools were required including, a Java development environment, an easy to use method to interact with Google App Engine, a relational database management system for local and remote databases and a text editor. During development it was not feasible to restrict the development of the application to one machine. For this reason two separate computers were used for the creation of this application, to achieve this a framework was needed to support to cross system development process. This process allowed for easier transfer of code & data from one computer to another, providing a free flowing development environment.

4.4.1. Eclipse

Eclipse is an integrated development environment (IDE), mostly focused on Java development, but other languages can be used through the use of plugins. Eclipse was chosen as the primary IDE due to previous experience with it and its easy integration with Google App Engine. Eclipse includes handy features such as code completion, error checking, breakpoint debugging, code refactoring, along with many other features for Java development. Eclipse was utilised to create the backend of this program that would link the website to the database and run functions such as the linear regression algorithm and Pearson's correlation algorithm.

4.4.2. Google App Engine Eclipse Plugin

To support development, Google provides an Eclipse plugin for Google App Engine. This plugin provides access to Google Java library's that are necessary for creating a Google App Engine. This plugin was utilised as it allowed easier inaction with the Google App Engine production environment compared to command line tools provided by the App Engine SDK. The plugin also provides a local development server which is used to test applications before uploading using the plugin's ability to upload the application to the production environment of Google App Engine. Google App Engine was used in addition to Eclipse to develop the backend framework of the application.

<https://www.eclipse.org/>

<https://developers.google.com/appengine/docs/java/tools/eclipse>

4.4.3. MySQL Workbench

MySQL workbench is a relational database management system (RDMS) developed by Oracle that provides an easy to use graphical interface, allowing developers easy access to create, edit and update both local and remote SQL relational databases. MySQL was chosen as RDMS as Google Cloud SQL is based upon MySQL. Sticking with a similar architecture to the Google backend could reduce the chance of problems arising. During development, a local database was created using MySQL workbench to mimic Google Cloud SQL which is based upon MySQL for testing purposes. Before the application was uploaded to the production environment, MySQL Workbench was once again used to interact with Google Cloud SQL to create the production relational database.

4.4.4. Sublime Text 2

Sublime text 2 is a text editor that allows developers to edit source code of any document, providing features such as spell checking, auto completion and syntax highlighting. Sublime Text 2 was chosen as it provided a better file navigation system and code completion for HTML and JavaScript when compared to Notepad++. Sublime Text 2 was utilised during the development stages of the front end website, to create the HTML, CSS and JavaScript of the website.

4.4.5. Dropbox

Dropbox is a file sharing and cloud backup application that allows users to store files on Dropbox servers for backup and allows access to these files from anywhere. While testing with Microsoft OneDrive it was necessary to choose which files to store offline and online, leading to confusion when new files weren't readily available on the second computer. Dropbox was chosen for this task as it automatically downloaded all updated files, alleviating the problem of OneDrive. Dropbox was used throughout development to continue development on a laptop, while allowing access to the most up to date files of the application. Dropbox also stores the last 30 versions of a file and for this reason was able to provide a degree of version control for individual files that was utilised throughout development.

4.4.6. TeamViewer

TeamViewer is a remote desktop application allowing users easy access to control their computer from a remote location. TeamViewer was chosen due to previous experience using the application. During the development of this application, the full development suite was not installed on my laptop (Google App Engine Eclipse Plugin & MySQL workbench) so while developing new features TeamViewer was utilised to access my development environment PC to test the new feature.

<http://www.mysql.com/products/workbench/>

<http://www.sublimetext.com/2>

<https://www.dropbox.com/home>

<http://www.teamviewer.com/en/index.aspx>

4.4.7. Google Analytics

Google Analytics is a web application that allows website administrators monitor the actions of website users and collect data on individual users such as how they reached the website, their location, type of device they are using, etc. Google Analytics was chosen due to past experience using it web application. Google Analytics was utilised during the testing of this of the application to monitor the actions of users on the website.

4.5. Utilised Technologies

In this section I will discuss the different technologies that are used during the development process of this application. A number of different technologies were required for this project including client side display and scripting languages, server side languages, testing frameworks, database interaction languages and methods to store data for transfer.

4.5.1. Java

Java is an object oriented programming language developed by Oracle that allows its programs to run on a variety of machines and environments as it is compiled to run in the Java Virtual Machine (JVM). Java was chosen due to its popularity and its exposure in its use for already established data mining tools. Java is one of the most popular languages worldwide and is used to power servers, phones, desktop programs and can be used to run within web browsers. Within my application Java was employed to develop the backend server of the application, this included Java servlets to interact with the front-end web application, Java Database Connection API to connect to remote databases and Java classes to store and manipulate data.

4.5.2. Junit

Junit is a unit testing framework developed for Java, allowing developers to utilise black box testing for their application. Unit testing was chosen to support a development of clean functional code in the backend server. Junit was chosen for this Java class testing due to its popularity in Java development, readily available tutorials and past experience with using it. Black box testing is a method of testing the outcome of a section of code without examining the internal workings of the code. Junit was utilised throughout development to determine if backend Java classes were performing as expected.

<http://www.google.ie/analytics/>

<http://www.java.com/en/>

<http://junit.org/>

4.5.3. JavaScript Object Notation (JSON)

JSON was developed as an alternative to XML to store and transfer data objects in a human readable fashion between endpoints in a system. This technology was necessary for my client server architecture to allow each endpoint to communicate with each other. JSON was chosen due to its simplicity compared to XML. For this application JSON was utilised to transfer data between the web client and the backend server, with GSON (a Google JSON parser) incorporated in the server to parse this JSON upon arrival and then create new JSON messages to return to the web client.

4.5.4. AngularJS - JavaScript

AngularJS is a JavaScript framework that attempts to improve the default implementation of JavaScript for the development of web applications by dividing the application into a model view controller style framework. AngularJS was chosen over straight JavaScript and alternatives such as JQuery due to its clean layout and division of the model data and application user view. JavaScript thus AngularJS allows a developer to create client side scripts that run tasks on the client's machine when called. In my application AngularJS was utilised to allow the user to interact with the website, allowing them to traverse sections of the website, control Ajax requests and responses to the backend server and to modify the content of the website.

4.5.5. SQL

SQL is a programming language designed for interacting with relation databases to manage and manipulate and is developed by the International Organisation for Standardization (ISO) & the American National Standard Institute (ANSI). SQL was used in this project to create the data structure of the database, upload the original dataset and serve this dataset to users and to manage new users and their race predictions.

4.5.6. Hypertext Mark-up Language (HTML)

HTML is a publishing language used primarily for content creation on the web and is a standard governed by the W3 organisation. HTML was used in this project to create the elements that make up the website application.

4.5.7. Cascading Style Sheets (CSS)

Cascading style sheets are a styling language used to design and style elements in HTML and is a standard governed by the W3 organisation. CSS was utilised in this application to style the look and feel of the web application.

<https://developer.mozilla.org/en/docs/Web/JavaScript>

<http://angularjs.org/>

<http://www.mysql.com/>

<http://www.w3.org/>

<http://www.w3.org/Style/CSS/Overview.en.html>

4.5.8. Greensock

Greensock is a JavaScript animation toolset designed to enhance the animation functions built into HTML5 & CSS3 that allow complex animations and designs for HTML elements. Greensock was chosen due to its added functionality over CSS animations and its improved performance compared to standalone JavaScript animation. Greensock was used in this application to develop the animations between pages and to hide previous pages.

4.5.9. Asynchronous JavaScript and XML (Ajax)

Ajax is a web client technique utilising elements from different web languages to transfer data from a client web browser to a server while allowing the subsequent code to continue executing. Ajax is not restricted to just transferring XML documents as it is just a technique and can be used with other methods such as JSON which I have used. In this application Ajax was used to set up the transfer of data to the Java Servlets backend and to receive the responses from the server.

4.6. Conclusion

This chapter has outlined a high level overview of the web application along with the different users that will interact with it. The method in which these users interact with the application and the content that is relevant to them. The chapter then outlined the different technologies, tool sets, API's and websites that will be utilised throughout the development of this project. Requirements for the different technologies were outlined and clear points were drawn on why each tool and application was chosen. This chapter should give the reader an understanding of the development environment,

<http://www.greensock.com/>

<http://www.w3.org/standards/>

5. Implementation

This chapter will highlight the key stages in the development process of this application from the outlining of the requirements and design through to their implementation and will examine the methods and techniques used to achieve this. This chapter has been divided into two sections, the first focusing on the development of the dataset and modelling it and the second section focusing on the development of the web application and additional features.

5.1. Dataset Creation & Evaluation

5.1.1. Data Modelling

This section will outline the methods used to build, test and evaluate the F1 dataset that would be eventually used in my application and the problems I faced. The full dataset would be made up of data from the 2010 season to the 2013 season as this warranted a large amount of data that I could reference against, while following the rule of 3 parts training data, 1 part test data, if it was decided to do prediction for a full season. During this process I broke the data into three separate iterations (similar to how scrum sprints), downloading and focusing on one iteration at a time to fully understand it. This allowed me to understand the dataset in chunks, starting from a small dataset and growing to the full large dataset, instead of creating the full dataset at the start. This meant that I could examine a small dataset at the beginning, I could begin to understand which explanatory variables warranted further investigation into their relation with the race finish. This would then avoid the problem of finding explanatory variables that were not reliable, and thus didn't warrant being inserted into the full dataset and wasting time in their collection. These three iterations included:

- Australian GP, first race of 2013 season
- 2013 season data
- 2010 – 2013 season data

5.1.2. Business & Data Understanding

The first step in this project was to understand and outline the questions I wanted answered using the available data. With a knowledge of Formula 1 already I decided that a viable question “was who would win a given race?” and from this I decided to extend the question to “what would the final race ranking be of a particular race?” given a historical dataset. This question could be asked on both future races before they had occurred and on past races that had already been completed. By predicting past races, the predicted race results could be compared to the real result of the race.

To understand the data that was available from the FIA website, the first instance outlined, the first race of the 2013 season, the Australian GP, was analysed. This dataset contained documents regarding press conferences, stewards' decisions on technical regulations and timing information for the different race weekend sessions. Within the F1 dataset there is

upwards of 40 explanatory variables that could potentially be used to help generate the prediction model. Without examining these variables we may include variables in our calculations that do not help and may in some cases our model to be less accurate, a subset of these explanatory variables are outlined below:

- Free Practice 1 & 2 & 3 times
- Free Practice 1 & 2 & 3 gap
- Free Practice 1 & 2 & 3 lap count
- Qualifying 1 & 2 & 3 times
- Qualifying 1 & 2 & 3 speed trap 1
- Qualifying 1 & 2 & 3 speed trap 2
- Qualifying 1 & 2 & 3 average speed
- Start Position
- Race speed trap 1
- Race speed trap 2
- Race average speed

With these explanatory variables in mind, we must first find what explanatory variables have statistical relevance against our race result (response variable). For this reason, Pearson's Product Moment Correlation is utilised as outlined in Chapter 2. Though 40 explanatory variables may seem a large amount, other variables that could influence a race result are not taken into account in our dataset. This is the area of data mining where insight into the dataset and the area around it is important. Without prior knowledge of how F1 works an analyst may not realise the potential for new computed variables and how they may relate to the response variable in a meaningful way.

Due to the lack of analysis around F1 data and computed columns and their effect on results, my research in this area is based around similar analysis in NASCAR racing. Within NASCAR, many different research projects have been undertaken to determine how different factors affect race results, such as driver experience [42], vehicle failures and team costs [43], the reward system [44] and previous related race experiences and car and team characteristics [45]. From this research new variables were created around a drivers racing experience, previous results this season and previous results at a given track. Ideally with more time variables around safety car laps, the F1 reward system and the reliability of cars by team would be analysed to see if they can achieve a similar relationship with the finish position as has been identified in the above articles.

Initial tests were conducted within Excel on these explanatory variables, recreating Pearson's Correlation formula using Excel functions, to learn the correlation between the explanatory variables and the response variable of the race result. The image below shows this process in Excel with some of the explanatory variables.

AUS Start Position V Finish Position							AUS	
Car Number	Driver	Start Position X	Finish Position Y	XY	X2	Y2	Car Number	Driver
1	S.Vettel	1	3	3	1	9	1	S.Vettel
2	M.Webber	2	6	12	4	36	2	M.Webber
3	F.Alonso	5	2	10	25	4	3	F.Alonso
4	F.Massa	4	4	16	16	16	4	F.Massa
5	J.Button	10	9	90	100	81	5	J.Button
6	S.Perez	15	11	165	225	121	6	S.Perez
7	K.Raikkonen	7	1	7	49	1	7	K.Raikkonen
8	R.Grosjean	8	10	80	64	100	8	R.Grosjean
9	N.Rosberg	6	20	120	36	400	9	N.Rosberg
10	L.Hamilton	3	5	15	9	25	10	L.Hamilton
11	N.Hulkenberg	11	22	242	121	484	11	N.Hulkenberg
12	E.Gutierrez	18	13	234	324	169	12	E.Gutierrez
14	P.diResta	9	8	72	81	64	14	P.diResta
15	A.Sutil	12	7	84	144	49	15	A.Sutil
16	P.Maldonado	17	21	357	289	441	16	P.Maldonado
17	V.Bottas	16	14	224	256	196	17	V.Bottas
18	J.Vergne	13	12	156	169	144	18	J.Vergne
19	D.Ricciardo	14	19	266	196	361	19	D.Ricciardo
20	C.Pic	22	16	352	484	256	20	C.Pic
21	G.Garde	21	18	378	441	324	21	G.Garde
22	J.Bianchi	19	15	285	361	225	22	J.Bianchi
23	M.Chilton	20	17	340	400	289	23	M.Chilton
22	Σ	253	253	3508	3795	3795	22	Σ
Pearson Correlation Coefficient:							Pearson Correlation	
0.6758893							0.6905703	
AUS FP2 Classification V Finish Position							AUS	
Car Number	Driver	FP2 Classification X	Finish Position Y	XY	X2	Y2	Car Number	Driver
1	S.Vettel	1	3	3	1	9	1	S.Vettel
2	M.Webber	2	6	12	4	36	2	M.Webber
3	F.Alonso	6	2	12	36	4	3	F.Alonso
4	F.Massa	8	4	32	64	16	4	F.Massa
5	J.Button	11	9	99	121	81	5	J.Button
6	S.Perez	13	11	143	169	121	6	S.Perez
7	K.Raikkonen	4	1	4	16	1	7	K.Raikkonen
8	R.Grosjean	5	10	50	25	100	8	R.Grosjean
9	N.Rosberg	3	20	60	9	400	9	N.Rosberg
10	L.Hamilton	7	5	35	49	25	10	L.Hamilton
11	N.Hulkenberg	10	22	220	100	484	11	N.Hulkenberg
12	E.Gutierrez	15	13	195	225	169	12	E.Gutierrez
14	P.diResta	12	8	96	144	64	14	P.diResta

FIGURE 19 INITIAL PEARSON'S CORRELATION TESTING

From analysing these results and the explanatory variables themselves, I chose to remove any variables that were related to the race, such as average speed during the race, as these would not be beneficial to predicting the result of a race before it began. Along with this explanatory variables that included a driver's time (in the format: minute: seconds: milliseconds) to complete a lap were disregarded due to their complexity to perform tests against. Instead explanatory variables that depicted time were transformed into a ranking of the drivers for their time. The below table illustrates this process.

FP3 TIME	FP3 LAPS	FP3 GAP	FP3 RANK
1'55.000	14	0.901	7
1'55.860	8	1.761	12

1'54.533	13	0.434	4
1'54.368	13	0.269	2
1'54.646	16	0.547	5
1'54.500	15	0.401	3
1'54.739	14	0.64	6
1'54.099	14	0	1
1'56.259	15	2.16	14
1'55.461	13	1.362	10
1'55.331	15	1.232	8
1'56.811	16	2.712	18
1'55.521	13	1.422	11
1'55.432	11	1.333	9
1'56.295	17	2.196	15
1'56.504	16	2.405	16
2'01.259	5	7.16	21
1'59.789	15	5.69	20
0	0	0	0
2'04.001	11	9.902	22
1'56.063	14	1.964	13
1'56.530	14	2.431	17
1'59.173	10	5.074	19
0	2	0	0

FIGURE 20 RANKING RACE WEEKEND TIMES

From this testing a list of explanatory variables were created that would be collected for the full 2013 season before rerunning these tests again to get a final list to be used within the model.

5.1.3. Data Preparation

The full dataset for the 2013 season, the second iteration of the dataset, was collected from the FIA website. As mentioned previously this dataset came in the form of PDF documents, with 550 PDF documents in total. From the included timing documents the data was extracted from the PDF's using a batch conversion process set up in Adobe Acrobat XI Pro to convert them to individual Excel files. A master Excel workbook was created that would hold all this information sorted by track and then driver number, this excel file would become the basis for the final application dataset. VBA code and Excel functions were used to cleanse the data of the individual Excel files, removing unwanted aspects from the PDF to Excel conversion process, the code then selected the needed information and transferred it into the correct position of the master excel workbook. An example of this VBA code is shown below:

```

Sub Macro2()

    Dim w As Workbook
    Set w = ActiveWorkbook

    Rows("5:5").Select
    Selection.Delete Shift:=xlUp
    Cells.Select
    Range("A5").Activate
    With Selection
        .VerticalAlignment = xlTop
        .Orientation = 0
        .AddIndent = False
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Range("C6:J27").Select
    ActiveWorkbook.Worksheets("Table 1").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Table 1").Sort.SortFields.Add Key:=Range("C6"), _
        SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Table 1").Sort
        .SetRange Range("C6:J27")
        .Header = xlNo
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
    Range("H6:J27").Select
    Selection.Copy
    Windows("Driver_Race_Weekend.xlsx").Activate
    ActiveSheet.Paste

    w.Activate
    ActiveWorkbook.Close False

End Sub

```

FIGURE 21 EXAMPLE VBA TRANSFER CODE

Within the master Excel workbook, functions were set up to rank the position of drivers timed laps. This was a very manual and time consuming process due to having to run the code for each file individually. Upon completion of the 2013 data, the dataset was verified by performing spot checks on the data against the original PDF's to determine if any files had brought across incorrect data. Below is a screenshot of a subsection of the master Excel dataset.

Year	Car Number	Driver	Circuit	Previous Race	Previous Race	Previous Race	FP1 Time	FP1 Lap	FP1 Gap	FP1 Rank	FP2 Time	FP2 Lap	FP2 Gap	FP2 Rank	FP3 Time	FP3 Lap	FP3 Gap	FP3 Rank	Q1 Time	Q1 Lap
2013	1	S.Vettel	Bahrain	3	1	4	1:34.790	20	0.303	3	1:34.282	29	0.128	2	1:33.348	15	0.101	2	1:33.327	1
2013	2	M.Weber	Bahrain	6	7	0	1:35.101	19	0.614	6	1:34.184	26	0.030	1	1:33.360	19	0.133	3	1:33.366	2
2013	3	F.Alonso	Bahrain	2	8	1	1:34.564	19	0.077	1	1:34.310	29	0.156	3	1:33.247	12	0	1	1:32.878	3
2013	4	F.Massa	Bahrain	4	15	6	1:34.487	11		N/A	1:34.552	34	0.338	5	1:33.343	20	0.702	11	1:33.780	4
2013	5	J.Button	Bahrain	9	10	5	1:35.069	22	0.582	5	1:35.356	32	1.202	10	1:34.117	17	0.870	12	1:34.071	5
2013	6	S.Perez	Bahrain	11	6	11	1:35.640	23	1.153	10	1:35.589	37	1.435	12	1:34.282	18	1.035	13	1:34.310	6
2013	7	K.Raikkonen	Bahrain	1	2	2	1:35.345	17	0.858	8	1:34.154	31		N/A	1:33.446	21	0.199	4	1:33.827	7
2013	8	R.Grosjean	Bahrain	10	3	9	1:35.611	14	1.124	9	1:34.631	33	0.477	6	1:33.464	19	0.217	6	1:33.498	8
2013	9	N.Rosberg	Bahrain	0	9	0	1:34.621	22	0.134	2	1:34.666	37	0.512	7	1:33.764	19	0.517	9	1:33.364	9
2013	10	L.Hamilton	Bahrain	5	5	3	1:35.792	16	1.305	12	1:34.376	29	0.822	9	1:33.455	19	0.208	5	1:33.498	10
2013	11	N.Hulkenberg	Bahrain	0	12	10	1:36.755	20	2.268	16	1:36.133	36	1.979	14	1:33.322	17	0.675	10	1:34.409	11
2013	12	E.Gutierrez	Bahrain	13	18	0	1:37.214	21	2.727	17	1:36.616	34	2.462	17	1:33.592	29	6.345	22	1:34.730	12
2013	14	P.di Resta	Bahrain	8	4	8	1:34.343	17	0.462	4	1:34.543	35	0.589	4	1:33.700	15	0.453	8	1:33.762	13
2013	15	A.Sutil	Bahrain	7	13	0	1:35.119	19	0.632	7	1:34.332	33	0.178	8	1:33.596	17	0.349	7	1:34.048	14
2013	16	P.Maldonado	Bahrain	0	11	14	1:36.438	17	2.011	15	1:36.279	33	2.125	15	1:34.833	17	1.586	17	1:34.425	15
2013	17	V.Bottas	Bahrain	14	14	13	1:35.783	16	1.296	11	1:36.579	28	2.425	16	1:34.611	17	1.364	15	1:34.425	16
2013	18	J.Vergne	Bahrain	12	0	12	1:36.014	19	1.527	13	1:35.506	36	1.352	11	1:34.678	16	1.431	16	1:34.314	17
2013	19	D.Ricciardo	Bahrain	0	16	7	1:36.485	20	1.398	14	1:35.761	33	1.607	13	1:34.577	16	1.330	14	1:34.120	18
2013	20	C.Pic	Bahrain	16	17	16	1:37.850	20	3.363	18	1:37.061	33	2.307	18	1:35.816	16	2.569	18	1:35.283	19
2013	21	G.Garde	Bahrain	18	21	18	1:38.401	20	3.914	19	1:37.910	34	3.816	21	1:36.939	16	3.632	20	1:36.304	20
2013	22	J.Bianchi	Bahrain	15	15	15	1:40.215	7	5.128	21	1:37.363	29	3.209	20	1:36.731	17	3.484	19	1:36.178	21
2013	23	M.Chilton	Bahrain	17	20	17	1:39.445	12	4.958	20	1:37.313	33	3.159	19	1:37.630	7	4.383	21	1:37.118	22
2013	231.000			171	231	171	0.000	372.000	35.507	N/A	0.000	678.000	30.016	N/A	0.000	363.000	30.212	237.000	0.001	
2013	1	S.Vettel	Spain	1	4	1	1:29.457	11	4.205	18	1:22.808	34		N/A	1:22.229	17	0.328	5	1:22.158	1
2013	2	M.Weber	Spain	7	0	7	1:29.473	21	4.221	19	1:22.831	36	0.083	2	1:22.044	17	0.143	3	1:22.342	2
2013	3	F.Alonso	Spain	8	1	8	1:25.252	20		N/A	1:22.825	35	0.017	1	1:22.254	15	0.353	6	1:22.264	3
2013	4	F.Massa	Spain	15	6	15	1:25.455	20	0.203	1	1:23.110	37	0.302	4	1:21.901	13	0	1	1:22.432	4
2013	5	J.Button	Spain	10	5	10		6			1:24.306	35	1.438	11	1:23.151	13	1.250	12	1:23.166	5
2013	6	S.Perez	Spain	6	11	6	1:27.135	6	1.883	12	1:24.854	31	2.045	12	1:23.373	13	1.472	14	1:23.116	6
2013	7	K.Raikkonen	Spain	2	2	2	1:26.614	21	1.362	7	1:23.030	32	0.222	3	1:21.907	14	0.006	2	1:22.210	7
2013	8	R.Grosjean	Spain	3	9	3	1:26.042	21	0.790	3	1:25.851	36	3.043	17	1:22.063	13	0.168	4	1:22.613	8
2013	9	N.Rosberg	Spain	9	0	9	1:26.621	21	1.369	8	1:23.398	45	0.590	6	1:22.839	26	0.938	11	1:21.913	9
2013	10	L.Hamilton	Spain	5	3	5	1:26.374	19	1.122	5	1:23.140	35	0.332	5	1:22.740	24	0.839	9	1:21.728	10
2013	11	N.Hulkenberg	Spain	12	10	12	1:27.061	24	1.809	11	1:25.167	38	2.359	14	1:23.388	18	1.487	16	1:23.058	11
2013	12	E.Gutierrez	Spain	18	0	18	1:27.250	26	1.938	13	1:25.441	37	2.533	16	1:23.371	21	1.470	13	1:23.218	12
2013	14	P.di Resta	Spain	4	8	4	1:26.755	16	1.503	9	1:24.104	25	1.296	9	1:22.574	11	0.673	7	1:22.563	13
2013	15	A.Sutil	Spain	13	0	13	1:26.212	24	0.960	4	1:23.840	37	1.032	7	1:22.729	17	0.828	8	1:22.652	14
2013	16	P.Maldonado	Spain	11	14	11	1:27.576	24	2.324	14	1:25.321	32	2.513	15	1:23.385	17	1.484	15	1:23.316	15
2013	17	V.Bottas	Spain	14	13	14	1:26.456	20	1.204	6	1:24.888	38	2.080	13	1:23.660	16	1.759	17	1:23.260	16
2013	18	J.Vergne	Spain	0	12	0	1:25.667	25	0.415	2	1:24.058	31	1.250	8	1:22.759	15	0.858	10	1:22.775	17
2013	19	D.Ricciardo	Spain	16	7	16	1:26.340	26	1.688	10	1:24.175	32	1.367	10	1:23.767	17	1.866	18	1:22.905	18
2013	20	C.Pic	Spain	17	16	17	1:28.373	14	3.121	15	1:26.930	35	4.122	20	1:24.775	18	2.874	19	1:23.070	19
2013	21	G.Garde	Spain	21	18	21	1:28.600	19	3.348	16	1:25.963	30	3.155	18	1:25.250	18	3.349	22	1:24.661	20
2013	22	J.Bianchi	Spain	19	15	19	1:28.887	14	3.635	17	1:26.078	31	3.270	19	1:24.793	16	2.892	20	1:24.713	21
2013	23	M.Chilton	Spain	20	17	20	1:30.314	13	5.062	20	1:26.310	26	4.162	21	1:25.135	17	3.234	21	1:24.996	22
2013	190.000			231	171	231	0.000	346.000	37.663	N/A	0.000	651.000	29.324	N/A	0.000	320.000	23.896	217.000	0.000	1
2013	1	S.Vettel	Monaco	4	1	4	1:17.380	23	1.185	9	1:16.014	33	1.255	8	1:15.261	17	0.883	3	1:14.243	1
2013	2	M.Weber	Monaco	0	7	5	1:17.020	27	0.825	6	1:15.404	42	0.645	4	1:15.550	20	1.172	7	1:15.352	2
2013	3	F.Alonso	Monaco	1	8	1	1:16.282	27	0.087	1	1:15.196	38	0.437	2	1:15.286	17	0.908	4	1:13.712	3
2013	4	F.Massa	Monaco	6	15	3	1:16.334	23	0.193	3	1:15.278	33	0.513	3	1:16.105	8	1.127	16		4
2013	5	J.Button	Monaco	5	10	8	1:17.123	29	0.934	7	1:15.959	40	1.200	7	1:15.976	19	1.598	12	1:13.744	5
2013	6	S.Perez	Monaco	11	6	9	1:17.378	25	1.183	8	1:16.434	41	1.615	11	1:15.958	23	1.580	11	1:14.682	6
2013	7	K.Raikkonen	Monaco	2	2	2	1:17.509	26	1.314	10	1:15.511	39	0.752	5	1:15.380	19	1.002	6	1:15.835	7
2013	8	R.Grosjean	Monaco	9	3	0	1:16.380	21	0.185	2	1:15.718	10	0.359	6	1:15.033	13	0.661	2	1:13.738	8
2013	9	N.Rosberg	Monaco	0	9	6	1:16.135	31		N/A	1:14.759	46		N/A	1:14.378	22	0	1	1:14.620	9
2013	10	L.Hamilton	Monaco	3	5	12	1:16.463	28	0.274	4	1:15.077	51	0.318	1	1:15.311	20	0.353	5	1:13.779	10
2013	11	N.Hulkenberg	Monaco	10	12	15	1:18.193	26	1.938	13	1:16.823	43	2.064	12	1:15.926	25	1.548	10	1:15.547	11
2013	12	E.Gutierrez	Monaco	0	18	11	1:18.754	28	2.559	15	1:16.935	45	2.176	14	1:16.427	26	2.049	17	1:16.917	12

FIGURE 22 OVERVIEW OF SECTION OF MASTER EXCEL WORKBOOK

With the new data collected it was important to reevaluate the explanatory variables before proceeding to collecting the third iteration of the dataset, as was the case with the first iteration. A similar process was used for the first iteration to understand the data, using correlation testing against the dataset and the race results and by individually analysing the explanatory variables. Correlation testing on the 2013 data showed that a few explanatory variables of data were giving positive relations to the response variable and were worth further investigation in collecting similar data for the 2010 – 2012 seasons, this included:

- Driver Number
- Free Practice 3 Result
- Start Position

The following computed columns were also used and though they did not show a highly positive correlation towards the response variable, I decided to include these as they are considered to be a good indication of the performance within not just Formula 1, but in many sports [14] [45].

- Previous Race
- 2nd Previous Race

- 3rd Previous Race

To collect the data for the 2010 – 2012 seasons, the third iteration of the dataset, I needed to use Forix.com as my data source as outlined previously. As you cannot download datasets for a season or an individual race on Forix, the web page for each race session had to be loaded before highlighting the relevant data and then copying this into Excel. Once in Excel a similar approach to the 2013 season was taken, using VBA code and Excel functions to select, cleanse and transfer the needed data into the master Excel workbook. Below is an example worksheet used to transfer data, on the left the data is pasted from Forix before functions on the right sort the data into the correct order. VBA code is used to transfer it to the master Excel sheet.

1	7.00	M.Schumacher	Mercedes	1'29.183		16	214.063			1	J.Button	1'32.194		19	3.011
2	12.00	Nico Hülkenberg	Force India/Mercedes	1'29.292	0.109	19	213.801			2	L.Hamilton	1'32.296		20	3.113
3	15.00	Sergio Pérez	Sauber/Ferrari	1'30.199	1.016	23	211.652			3	M.Schumacher	1'33.039		18	3.856
4	5.00	Fernando Alonso	Ferrari	1'30.341	1.158	13	211.319			4	N.Rosberg	1'33.252		11	4.069
5	14.00	Kamui Kobayashi	Sauber/Ferrari	1'30.709	1.526	14	210.462			5	S.Vettel	1'30.341		13	1.158
6	11.00	Paul Di Resta	Force India/Mercedes	1'31.466	2.283	13	208.72			6	M.Webber	1'31.505		14	2.322
7	6.00	Felipe Massa	Ferrari	1'31.505	2.322	14	208.631			7	F.Massa	1'29.183		16	0
8	20.00	H.Kovalainen	Caterham/Renault	1'31.932	2.749	16	207.662			8	F.Alonso	1'32.184		17	3.001
9	8.00	Nico Rosberg	Mercedes	1'32.184	3.001	17	207.094			9	R.Barrichello	1'34.275		7	5.092
10	1.00	Sebastian Vettel	Red Bull/Renault	1'32.194	3.011	19	207.072			10	N.Hülkenberg	1'32.822		11	3.639
11	2.00	Mark Webber	Red Bull/Renault	1'32.296	3.113	20	206.843			11	R.Kubica	1'31.466		13	2.283
12	24.00	Timo Glock	Marussia/Cosworth	1'32.632	3.449	17	206.092			12	V.Petrov	1'29.292		19	0.109
13	21.00	Vitaly Petrov	Caterham/Renault	1'32.767	3.584	15	205.793			14	A.Sutil	1'30.709		14	1.526
14	10.00	Romain Grosjean	Lotus/Renault	1'32.822	3.639	11	205.671			15	V.Liuzzi	1'30.199		23	1.016
15	3.00	Jenson Button	McLaren/Mercedes	1'33.039	3.856	18	205.191			16	S.Buemi	1'34.604		31	5.421
16	4.00	Lewis Hamilton	McLaren/Mercedes	1'33.252	4.069	11	204.722			17	J.Alguersuari	1'34.485		29	5.302
17	18.00	Pastor Maldonado	Williams/Renault	1'34.108	4.925	21	202.86			18	J.Trulli	1'34.108		21	4.925
18	9.00	Kimi Räikkönen	Lotus/Renault	1'34.275	5.092	7	202.501			19	H.Kovalainen	1'34.312		17	5.129
19	19.00	Bruno Senna	Williams/Renault	1'34.312	5.129	17	202.421			20	K.Chandhok	1'31.932		16	2.749
20	17.00	Jean-Eric Vergne	Toro Rosso/Ferrari	1'34.485	5.302	29	202.051			21	B.Senna	1'32.767		15	3.584
21	16.00	Daniel Ricciardo	Toro Rosso/Ferrari	1'34.604	5.421	31	201.796			22	P.DiResta	0		1	0
22	25.00	Charles Pic	Marussia/Cosworth	1'34.770	5.587	13	201.443			23	K.Kobayashi	1'42.627		16	13.444
23	23.00	N.Karthikeyan	HRT/Cosworth	1'42.627	13.444	16	186.021			24	T.Glock	1'32.632		17	3.449
24	22.00	Pedro de la Rosa	HRT/Cosworth			1				25	L.Grassi	1'34.770		13	5.587

FIGURE 23 EXAMPLE FORIX TRANSFER WORKSHEET

With the full dataset now collected it was decided to analyse the final explanatory variables to find any variables that may cause a problem in the future. While analysing the relation between these explanatory variables and the response variable, I decided to disregard the driver number for use in the model. This was due to the fact that for the 2014 season, driver numbers were no longer going to be assigned by a driver and his team's position in the previous season, but rather chosen at the driver's discretion, meaning it was now essentially a random variable. This meant that the driver number was now an independent variable when compared against the race result as there was no logical link between the driver number and how it would affect the outcome of the race. When the full dataset was complete a similar verification process was used against the full dataset as in the second iteration, comparing data at random to the original Forix data to spot any errors that may have occurred.

5.1.4. Data Modelling & Model Evaluation

As discussed in the research section of this paper, linear regression was chosen as the algorithm that would be used for prediction on the F1 dataset. Before building my application Rapid Miner was used for initial testing against the full dataset. This initial testing allowed the evaluation of linear regression and cross validation on the dataset while highlighting any problems with the dataset and its explanatory variables that may have been missed during the data understand and data preparation stages. This testing also allowed me to compare

the eventual predictions from my application with an industry used application. An initial problem I spotted with the dataset and its use with linear regression, was that if a driver had failed to qualify for a race and thus not warrant a start position, the model would still predict the drivers finish position. A second problem found with the dataset was the addition of the new explanatory variables of a drivers three previous races. At the start of a season, each F1 team has a new car available with possible new drivers, which may differ largely to the previous year's car depending on rule changes. Using the last races of a previous season as the explanatory variables for the first races of a new season would not be beneficial as these explanatory variables and the response variable would have no logical correlation (making them independent variables to the response variable). For this reason the first three races at the start of the season could not be used with the previous races explanatory variable. This is a similar problem that occurred in [14], where matches from the first 2 months of the season (10+ games) were removed from the dataset for prediction. Both of these problems in the dataset were noted so that they could be circumvented while developing the linear regression algorithm.

5.2. Application Development

In this section I will discuss the development of my application, including the front-end website, backend server and database and discuss the design choices that influenced the look and feel of the application and the methods and workings of the back end. The development of this application was divided into three separate sprints as outlined in the scrum methodology. These sprints focused on creating a complete application that was fully tested by the assigned due date. These three sprints included, but were not limited to:

- Starting development and learning about Google App Engine and creation of linear regression algorithm
- Creation of the front end website
- Creation of additional features, testing and bug fixes

5.2.1. Sprint 1 – Backend Server

Sprint 1 would revolve around learning and developing for Google App Engine while also aiming to create a functional linear regression algorithm. To understand how to develop for App Engine, online tutorials on the Google website were used to learn the structure of a relevant application and to learn the best practices of the system. The below sequence diagram gives an overview of how the web application and Java classes interact with one another to create the model prediction.

<https://developers.google.com/appengine/docs/java/gettingstarted/introduction>

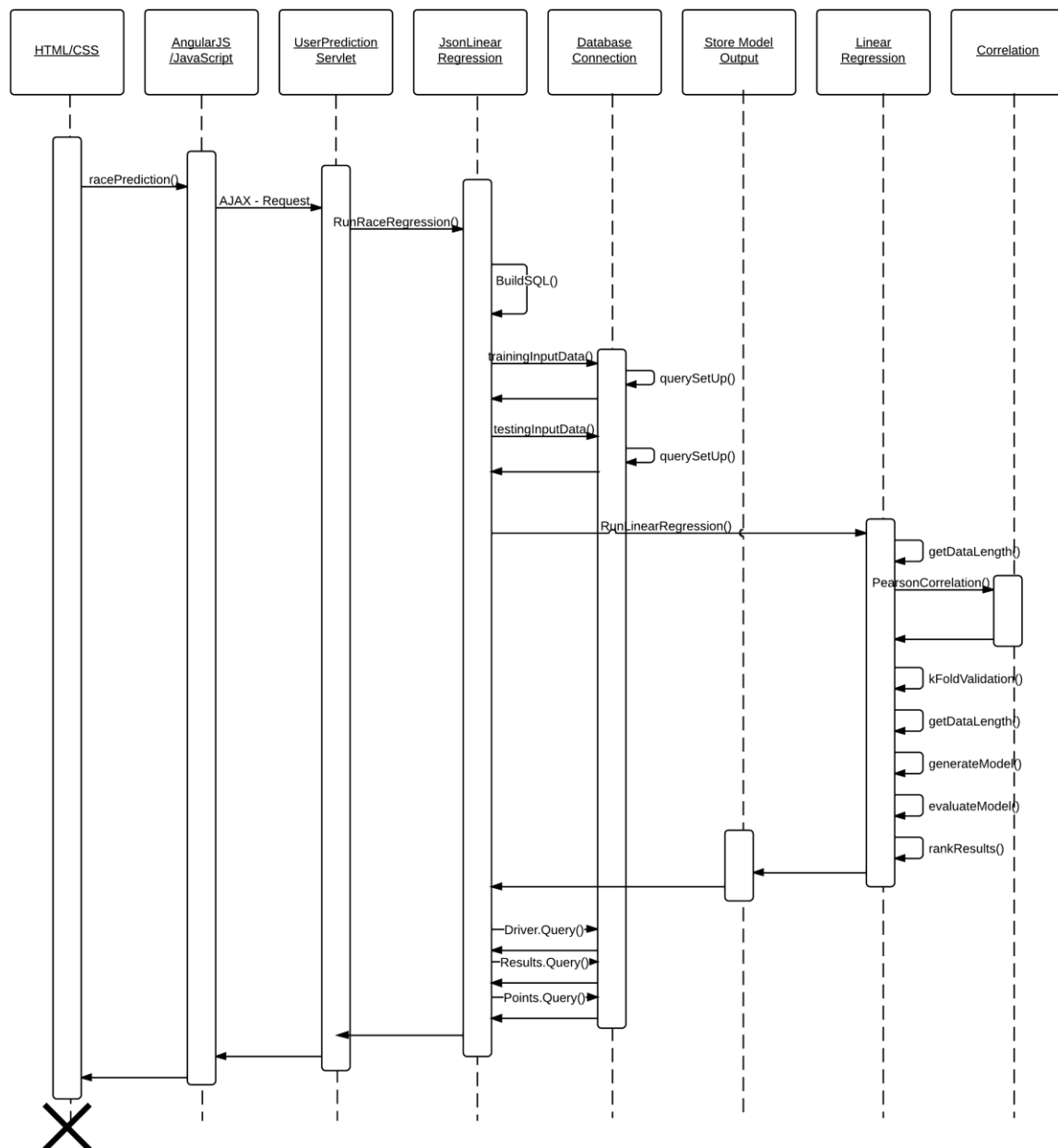


FIGURE 24 SEQUENCE DIAGRAM OUTLINING LINEAR REGRESSION PROCESS

The initial sprint would not focus on the development of the database or the web application, so it was decided a basic web page would be created, with a web form that could parse inputs to a Java servlet. This allowed the sprint to focus on the servlet and how it and other Java classes would be created to run the linear regression algorithm, while creating a full database at first and only working with a basic front-end. The image below shows one of the early iterations, of the front-end during development.

Training: 1st Explanatory Variables	Training: 2nd Explanatory Variables	Training: Response Variables
Testing: 1st Explanatory Variables	Testing: 2nd Explanatory Variables	

Run Prediction

```
{{ formData.name }}  
{{ returnedData }}  
  
{{test}}
```

FIGURE 25 SCREENSHOT OF EARLY TESTING WEB APPLICATION

The web page was the first step in this sprint. Using basic HTML, three input fields were created. The first two input fields were for inputting the training data's explanatory and response variables and the third field was for taking in the explanatory field of the testing data. To control sending this data to the servlet, a button linked to AngularJS was created. When clicked AngularJS would run its relevant function, which created an HTTP Post request, using an Ajax technique to asynchronously send the data to the web servlet and then wait upon the response. The below code snippet, shows an example of the Ajax technique used to

send to the servlet.

```
//this function is used to pass the data that a user has entered for the user prediction algorithm, to the server
//where the linear regression algorithm can be run
$scope.processUserPrediction = function()
{
    $http.post('/UserPrediction', $scope.userData, {headers: {'Type': 'userPrediction'}})
    .success(function(data)
    {
        console.log(data);
        if (data == null || data.error == true)
        {
            // if not successful, bind errors to error variables
            alert("Something went wrong :( Please refresh the page and try again");
        }
        else
        {
            // if successful, bind success message to message
            $scope.returnedUserData = data;
        }
    });
}
```

FIGURE 26 JAVASCRIPT CODE OUTLINING AJAX CONNECTION TO PREDICTION SERVLET

TheUserPredictionn servlet was the first servlet to be developed and its initial focus was to read in the request data passed into it from the webpage and organizes the data into a useable form that the linear regression algorithm could handle. The data that was sent to the server was organised within a JSON structure, a widely popular format, allowing it to be easily organised once in the servlet. To simplify this process, another Google API was utilised, GSON. GSON is one of many available JSON parses, allowing a JSON message to be easily converted into a Java class structure. A Java class, JSONLinearRegression, was created to mimic the format of the JSON message and the variables it contained. GSON then automatically parsed this data into string variables (TrainingVar1, TestingVar2, etc.) of an instantiated version of the class. This class became an intermediate class between the UserPrediction servlet and the LinearRegression class. The linear regression class was designed with a function, runLinearRegression, this function would take in two 2D arrays of doubles which would be used to run the linear regression algorithm before storing the results within its global variables so they could be accessed by JSONLinearRegression. This meant JSONLinearRegression would have to organise its explanatory and response variables to fit them into two 2D arrays before calling upon the LinearRegression class.

To achieve this a function convertData was created in JSONLinearRegression. This function was utilised to convert the string data held in the variables TrainingVar1, TestingVar2, etc., to an array of strings, each divided where a comma was present, which was then removed. Following this the string arrays were converted to an array of doubles for the testing data and training data. The testingData and trainingData could then be passed into the linear regression class.

Once the variables were within the linear regression function, the algorithm now had to be created. For this initial sprint, multi – variant linear regression was attempted using a matrix notation approach as matrices allow for compact expressions to calculate the response variable. The below algorithm illustrates the standard formula for multiple linear regression and its matrix form are shown:

$$\text{Standard: } Y' = B_0 + B_1X + B_1X + \dots + B_kX + e$$

$$\text{Matrix: } Y' = XB + e$$

FIGURE 27 MULTIPLE LINEAR REGRESSION FORMULA & MULTIPLE LINEAR REGRESSION WITH MATRICES FORMULA

In the above matrix formula, as we know X , is our training data and our error rate e can be dropped as it should average to zero, we must calculate B to determine Y' . To determine this, an adapted least squares normal equation for linear regression is used, which is stated below:

$$\text{Adapted least squares normal equation: } B' = (X^T X)^{-1} X^T Y$$

- Where T is a matrix transform
- -1 is the inverse of a matrix
- X is a $k \times n$ matrix of the training explanatory variables
- Y is a $1 \times n$ matrix of the training response variables

When attempting to code this matrix algorithm, two approaches were adopted. The first approach was to code the matrix equations by hand, but this proved difficult to debug when problems arose, as millions of rows of data were created, when multiplying out matrices.

The second approach was to utilise a library to compute the matrix equations. The Apache Commons Math Library was chosen for this as it provided the needed functions. The Apache library allowed me to outline the structure of the algorithm, to multiply, transpose and inverse the matrices with just a few lines of code. The below code snippet, displays both the original code and the Apache library utilised to determine B .

<http://commons.apache.org/proper/commons-math/>

```

for(i=0;i<RowCount;i++)
{
    for(j=0;j<ColumnCount;j++)
    {
        XMatrixTranspose[i][j] = XMatrix[j][i];
    }
}*/

RealMatrix RealXMatrix = new Array2DRowRealMatrix(XMatrix);

RealMatrix RealMatrixTranspose = RealXMatrix.transpose();

//Multiply XMatrix by its transpose matrix
//No need to check if matrices have same column and row
//length as transposed means always will be true
//Cik = ForEach j AijBjk

/*for(i=0;i<RowCount;i++)
{
    for(j=0;j<RowCount;j++)
    {
        for(int k=0;k<ColumnCount;k++)
        {
            XMatrixByTranspose[i][j] += XMatrix[i][k] * XMatrixTranspose[k][j];
        }
    }
}*/

RealMatrix RealMatrixByTranspose = RealXMatrix.multiply(RealMatrixTranspose);

//get the inverse of the matrix from the previous step
//the sides should be equal so no need to check in this case
//first need to get determinant using below formula
//|A| = ForEach ( + ) A1qA2rA3s . . . Anz
//I have used a library for this as matrix size is too large
//to allow me to do it by hand

//RealMatrix RealMatrixTranspose = new Array2DRowRealMatrix(XMatrixTranspose);
//RealMatrix RealMatrixByTranspose = new Array2DRowRealMatrix(XMatrixByTranspose);

MatrixOutput = RealMatrixByTranspose.getData();

System.out.println(MatrixOutput);

RealMatrix Inverse = new LUdecomposition(RealMatrixByTranspose).getSolver().getInverse();

RealMatrix temp = Inverse.multiply(RealMatrixTranspose);
RealMatrix temp1 = new Array2DRowRealMatrix(TrainingExp);

RealMatrix Output = temp.multiply(temp1);

```

FIGURE 28 JAVA CODE OUTLINING MATRIX LINEAR REGRESSION

Unfortunately, this method of using the Apache library to determine the model for multiple linear regression also ran into problems when the inverse attempts to multiply by the realMatrixTranspose (in the formula version when $(X^T X)^{-1}$ attempts to multiply by X^T).

Unable to fix this problem, a new formula was derived based upon simple linear regression, to solve the multiple linear regression algorithm.

Two functions were created for this simple linear regression, the first to generate the prediction model from the trainingData array and the second to evaluate the model and make the required predictions using the testingData array. A function getDataLength was also created to count the column count and row size of the training and test datasets.

The generateModel function, takes in the trainingData array as a parameter. This function is used to evaluate the correct slope and intercept of the line using simple linear regression. The

array variables m and b are created with a size equal to the number of explanatory variables been utilised. These arrays store the results for both the slope and intersect equations on each explanatory variable can be stored. As stated before the formula for the slope and the intercept are as follows, with a predicted B_1' slope and a predicted B_0' intercept:

$$B_1' = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2}$$

$$B_0' = \bar{y} - B_1' \bar{x}$$

FIGURE 29 SIMPLE LINEAR REGRESSION COEFFICIENTS ALGORITHMS

To evaluate these formulas, the columns of the explanatory variables in the training set are looped through, where upon each variable within each column is then looped through within an inner loop, this inner loop represents the sigma notation, within the equation for the slope. With each pass through of the sigma, the code takes the current explanatory training value and minuses the mean of this explanatory variable. It then repeats this process with the response variable before multiply the two together. The answer to this equation is stored within the variable `DifferenceXByY` and covers the top line of the divisor for the equation of the slope. The bottom line of the divisor for the equation of the slope is calculated by repeating this same process, taking the current explanatory training value and minuses the mean of this explanatory variable before squaring itself. The answer to this equation is stored in `DifferenceToMeanx2`.

These two equations are then divided from each other and stored within the array m . To calculate the intercept of the line, the previously calculated slope is multiplied by the average explanatory variable value and is then subtracted from the average response variable value. The code snippet below, displays this code:

```

for(int i = 0;i<ColumnCount;i++)
{
    double averagex = 0;
    double averagey = 0;

    double DifferenceToMeanx = 0;
    double DifferenceToMeany = 0;
    double DifferenceToMeanx2 = 0;
    double DifferenceXByY = 0;

    sumx = 0;
    sumy = 0;
    sumxy = 0;
    sumx2 = 0;
    sumy2 = 0;

    for(int j = 0;j<TrainingLength;j++)
    {
        sumx += trainingData[i][j];
        sumy += trainingData[ColumnCount][j];
        sumxy += trainingData[i][j] * trainingData[ColumnCount][j];
        sumx2 += trainingData[i][j] * trainingData[i][j];
        sumy2 += trainingData[ColumnCount][j] * trainingData[ColumnCount][j];
    }

    averagex = sumx / TrainingLength;
    averagey = sumy / TrainingLength;

    for(int j = 0;j<TrainingLength;j++)
    {
        DifferenceToMeanx2 += (trainingData[i][j] - averagex) * (trainingData[i][j] - averagex);
        DifferenceXByY += (trainingData[i][j] - averagex) * (trainingData[ColumnCount][j] - averagey);
    }

    //this is a previous method I was using
    //this method is using the prince george college method
    //a pdf i found online
    //it worked but decided to change, since other method seems to be
    //the exact formula in most papers found online
    //m[i] = ((TrainingLength * sumxy) - (sumx * sumy)) / ((TrainingLength * sumx2) - (sumx * sumx));
    //b[i] = ((sumx * sumx2) - (sumx * sumxy)) / ((TrainingLength * sumx2) - (sumx * sumx));

    //this method is from simple linear regression http://www.ccsr.ac.uk/publications/teaching/mlr.pdf
    m[i] = DifferenceXByY/DifferenceToMeanx2;
    b[i] = averagey - (m[i] * averagex);

    b0 += b[i];
}

```

FIGURE 30 JAVA CODE OUTLINING CREATION OF SIMPLE LINEAR REGRESSION MODEL

With the generateModel function complete and the slope and intercept stored in the global variables of the class, the evaluateModel function is called. This function is used to apply the previously created model to each row of the training set. The function works by running through each row of the training set and then looping through each explanatory variable and then multiplying that value by the relevant column slope. The code checks to see if the Start Position explanatory variable is being utilised, if it is and that is the current explanatory variable being used, then the code checks to see if the start position is equal to zero. If this is the case, then the predicated response value is set to zero. This check is used to avoid a previous problem found while testing the dataset in Rapid Miner, that if a driver was found to have not started a race, he would still be ranked in the prediction. After this step the response variable is then divided by the number of explanatory variables before adding in the intercept which has already been divided by the number of explanatory variables. This is shown in the code snippet below:

```

private void evaluateModel(double[][] testingData)
{
    /*******
    //Last Edited by: Eoin Farrell - Last Edit Date: 11/03/2014
    //Now that we have the slope and intercept, next step is to pass
    //the testing data through the formula  $y=mx+b$  where m and b are equal
    //to the results from above and x is the passed in testing data to
    //give a response variable = y
    /*******

    double RegressionModel;
    predictionOutcome = new double[TestingLength];

    b0 = b0/ColumnCount;

    //this is used to multiply each testing data point by each explanatory
    //variable slope, and then it divides by the number of explanatory slopes.
    //Thus it essentially gets the average slope of the five pieces of data
    for(int i=0;i<TestingLength;i++)
    {
        RegressionModel = 0;

        for(int j = 0;j<ColumnCount;j++)
        {
            if(StartPos == false)
            {
                RegressionModel += testingData[j][i] * m[j];
            }
            else
            {
                if(j != ColumnCount-1 || testingData[j][i] != 0)
                {
                    RegressionModel += testingData[j][i] * m[j];
                }
                else
                {
                    RegressionModel = 0;
                }
            }
        }

        if(RegressionModel != 0)
        {
            RegressionModel = RegressionModel/ColumnCount;
            predictionOutcome[i] = b0 + RegressionModel;
        }
    }
}

```

FIGURE 31 JAVA CODE OUTLINING THE EVALUATION OF THE MODEL ON TEST DATASET

This is not an ideal method of performing multiple linear regression, essentially averaging over individual simple linear regression methods as it values each explanatory variables contribution towards the response variable equally. This is in contrast to the matrix multiple linear regression outlined before, which weights the explanatory variables by their impact/correlation with the response variable. To overcome this problem, it was decided to introduce a new Correlation class which would take in the unaltered training and datasets. This correlation class would utilise Pearson's correlation algorithm, outlined before. This function would allow the results of the correlation testing performed on each explanatory

variable to be returned to the user, with their prediction model. A user could then use this information, to disregard explanatory variables that they believed had little effect on the race result, from reading these numbers. This would partially avoid low correlation variables being used in the predictions, as users deselect them, but still is not perfect and will hopefully be corrected in future work.

To avoid one of the last problems that surfaced from Rapid Miner testing, a function RankResults was created in the linear regression class. This function ranked the results of the predicted race results, so that each driver would be individually numbered 1 to 22/24 (depending on number of drivers participating in a race).

With the above race predictions stored within the global variables of the LinearRegression class, the JsonLinearRegression class could access these results, to be returned to the user. To return the predicted results, along with other details, a StoreModelOutput class was created. By creating a class to store the results of the prediction in, an easily readable JSON data structure could be created that the client web application would be able to easily interpret. JsonLinearRegression handled creating and populating an array of these class objects (one for each driver) before returning the array to the UserPrediction servlet. The servlet transforms this array into a JSON data structure using the GSON library before returning this to the web application. The returned information, was then dumped in plain text without any formatting to the user, using AngularJS.

To test the predictions made by the prediction model, two new functions were created in the LinearRegression class, kFoldValidation and getLikelyHood. As stated in the training, testing, validation section (Chapter 2) of this document, two methods were outlined for algorithm evaluation, hold out testing and k fold cross validation. During development I decided to implement k fold cross validation as it would allow all sections of the training data to be tested once to give an indication of the accuracy of the model instead of just using one part of the training data for testing, thus not fully utilising all the training data for evaluation. kFoldValidation was a function that implemented this tenfold cross validation on the dataset while calculating the root mean squared error and the mean absolute error, both of which have been examined before in this document. The getLikelyHood function was developed to determine the percentage likely hood of a predicted outcome occurring, given the values in the previous dataset.

As explained earlier, k fold cross validation is the process of dividing the training set into k folds, and then running each individual fold against the remaining training dataset to make a prediction. These predictions can then be compared to the real response variable to determine the fit of the prediction model against the training data. While implementing cross validation, I decided to stick to straight division of the set into 1/10 testing and 9/10 training moving from the start of the dataset till the end. Not using stratification to randomly select the 10 individual parts from the data there was no need to run the cross validation 10 times

on itself. Since the data was divided the same way each time, the results would remain the same. This improved the computation requirement for cross validation, reducing the number of iterations from 100 to just 10. To create this function. The training set is first divided into ten equal parts. This is accomplished by setting a lower bound to zero and an upper bound to the training size divided by ten. This upper and lower bound are then used to select the testing data from the training dataset while the remaining data is used for training. When both datasets have been created, their complete sizes are determined and then a model is created from this cross validation training set and the cross validation testing set is placed through the model. Upon completion of the model evolution, the predicted response values can be compared to the real response values, allowing an easy implementation of the root mean squared error and the mean absolute error. This process is repeated ten times and on each loop the lower and upper bound are increased again by the size of the training set divided by ten. This process allows all data within the function to be used for cross validation once. The code snippet below displays this implementation

```
for(int k = 0;k<10;k++)
{
    //chooses training data that is before starting point of testing data
    for(int i = 0;i<lowerCount;i++)
    {
        for(int j=0;j<trainingData.length;j++)
        {
            tempTrainingData[j][i] = trainingData[j][i];
        }
    }
    //gets the testing fold
    for(int i = lowerCount;i < higherCount ;i++)
    {
        for(int j=0;j < trainingData.length;j++)
        {
            //generates testing fold, while saving its results
            if(j != trainingData.length -1)
            {
                tempTestingData[j][i-lowerCount] = trainingData[j][i];
            }
            else
            {
                testOriginalResults[i-lowerCount] = trainingData[j][i];
            }
        }
    }
    //chooses training data that is after the finish point of testing data
    for(int i = higherCount;i<trainingData[0].length;i++)
    {
        for(int j=0;j<trainingData.length;j++)
        {
            tempTrainingData[j][i-higherCount] = trainingData[j][i];
        }
    }
    getDataLength(tempTrainingData, tempTestingData);
    generateModel(tempTrainingData);
    evaluateModel(tempTestingData);
    predictionOutcomeRanked = RankResults(predictionOutcome);

    //now need to compare predicted results against real results
    //Formulas for various error measurements
    for(int i=0;i<kFoldLengthTest;i++)
    {
        error += (testOriginalResults[i] - predictionOutcome[i]) * (testOriginalResults[i] - predictionOutcome[i]);
        meanAbsoluteErrorTemp += Math.abs(testOriginalResults[i] - predictionOutcome[i]);
    }
    kFoldError += error/kFoldLengthTest;
    rootMeanSquaredError += Math.sqrt(error/kFoldLengthTest);
    meanAbsoluteError += meanAbsoluteErrorTemp / kFoldLengthTest;
    //loop through that process for k
    lowerCount += kFoldLengthTest;
    higherCount += kFoldLengthTest;
}
```

FIGURE 32 JAVA CODE OUTLINING TENFOLD CROSS VALIDATION

A key consideration during the development of the `kFoldValidation` function was how I ranked my results. When a standalone race is put through the linear regression prediction, its results are sorted and ranked, to get rid of decimal places and to give each driver a standalone position between 1 and 22 or 24 depending on the number of race participants. When running cross validation a tenth of the training data is used for testing. During development this accounted to 115 rows of data, equivalent to just over 5 races for each testing iteration. If the same ranking function was used as a normal race prediction, then individual rows would be ranked between 1 and 115 when the actual results would range between 1 and 24 which is just over five times the actual results. The reason for this is the ranking function does not have any indication of when one race ends and the next begins in a list of testing data and cannot handle testing data where only a part of a race may exist. To get around this issue, during cross validation I decided to not use the ranked prediction results to get the error rate, but instead to compare the raw output of the linear regression algorithm to the actual race result. Though this is not strictly correct in comparison to how the algorithm determines the outcome of a single race it did give me a reasonable error rate.

An alternative `k` fold cross validation (`F1kFoldValidation`) function was developed that attempted to circumvent this issue by determining when one race began and another ended, to have each cross validation fold equal to one race. The results from this though did not offer much benefit from my testing in comparison to the first function and ignoring the ranking factor so this function is not called. I believe one of the reasons for this problem is the due to half races that occur in the testing set and the new function could still not handle these race results. The only method to fix this would be to cleanly divide the cross validation folds when one race ends and another begins, but this would be going against the idea of testing evenly sized folds.

As stated the `getLikelyHood` function was developed to get the percentage likelihood of a given prediction coming through. This function utilises probability based formulas that led to the creation of Bayes theorem and thus the creation of Naïve Bayes theorem and as such the function could be expanded in the future to offer an alternative prediction method to linear regression. The function operates by looping through the testing data, selecting the explanatory variables for a given row of the training data before storing it in the `compareTraining` array. This array is then compared to the training dataset, to find any rows of data that match the values of the explanatory variables. When a match is found the response variable of the training row is stored in a vector `trainingResults` (a vector is chosen instead of an array as it allows the vector size dynamically grow compared to an array which must be given a fixed size). The code then checks if the response variable of the training data is the same as the response variable of the testing data. If it is, a variable `correctCount` is incremented. With all the relevant data now determined, to calculate the likelihood percentage, the `correctCount` is divided by the size of the `trainingResults` vector and multiplied by 100. The below code snippet displays this function.


```

for(int i = 0;i<TestingLength;i++)
{
    //first need to get the first drivers test results
    for(int j = 0;j<ColumnCount;j++)
    {
        compareTesting[j] = testingData[j][i];
    }
    //loop through trainingData
    for(int j = 0;j<TrainingLength;j++)
    {
        check = true;
        //get first row of TrainingData
        for(int k = 0;k<ColumnCount;k++)
        {
            compareTraining[k] = trainingData[k][j];
        }
        //compare rows of training and test data
        for(int k=0;k<ColumnCount;k++)
        {
            if(compareTraining[k] != compareTesting[k])
            {
                k = ColumnCount;
                check = false;
            }
        }
        if(check == true)
        {
            trainingResults.add(new Double(trainingData[ColumnCount][j]));
        }
    }
    //this is where the percentage calculation should
    for(int j=0;j<trainingResults.size();j++)
    {
        if(trainingResults.get(j) == testingData[ColumnCount-1][i])
        {
            correctCount++;
        }
    }
}

if(trainingResults.size() == 0)
{
    likelyHood[i] = 0;
}
else
{
    likelyHood[i] = (correctCount/trainingResults.size()) * 100;
}
trainingResults.clear();
correctCount = 0;
}

```

FIGURE 33 JAVA CODE OUTLINING THE RANK LIKELIHOOD FUNCTION

Conclusion

When testing the dataset and LinearRegression class, I choose to use the US Grand Prix from 2013 for testing purposes. I chose this race as all drivers who started the race finished it, negating any discrepancies that could be present from drivers not finishing. Along with this one driver did not start the race, so I was able to make sure the algorithm was at all times not

predicting him to finish the race. From the correlation testing performed on the full training set when I used this race for the test set, I got an RMSE error of 5.8 and an MAE error rate of 4.8. This means that the position of a predicted driver finish can vary as much as 5.8 to 4.8 positions. MAE presents a lower error rating as it is less susceptible to inconsistent variances in the data set. Such variances in the F1 dataset are caused by drivers not finishing races. If a driver was predicted to finish in 20th position but in reality does not finish the race, a position of 0 is placed into the dataset, leading to a large inconsistency, leading to the higher value of the RMSE error rating.

5.2.2. Sprint 2 - Front-end Website

The second sprint in this process focused on the development of the front-end web application. Early on in this process it was decided to develop a single page website design. The reason for this was to allow an easily maintain website that allowed quick and easy changes to be made during development. A single page website allowed this as it would no longer be necessary to manage a multitude of different web pages with individual links to move between. The second reason was the uniqueness it would bring to my website and the freedom it would offer development as JavaScript could easily become the backbone of the development thus allowing stylised and animated interaction take place with the user. This style of web page would attract a user, due to its uniqueness in design. It was also decided to focus heavily on user testing throughout the development of the web application frontend. User testing was utilised throughout the 2nd sprint to understand the thoughts and feelings users had towards the website and appropriate changes were made when necessary.

During the initial design of the web application, it was focused primarily on allowing users to make predictions on F1 races and providing a tutorial to go with this and also offering the user the ability to search through the database of drivers and tracks. The following use case was put forward:

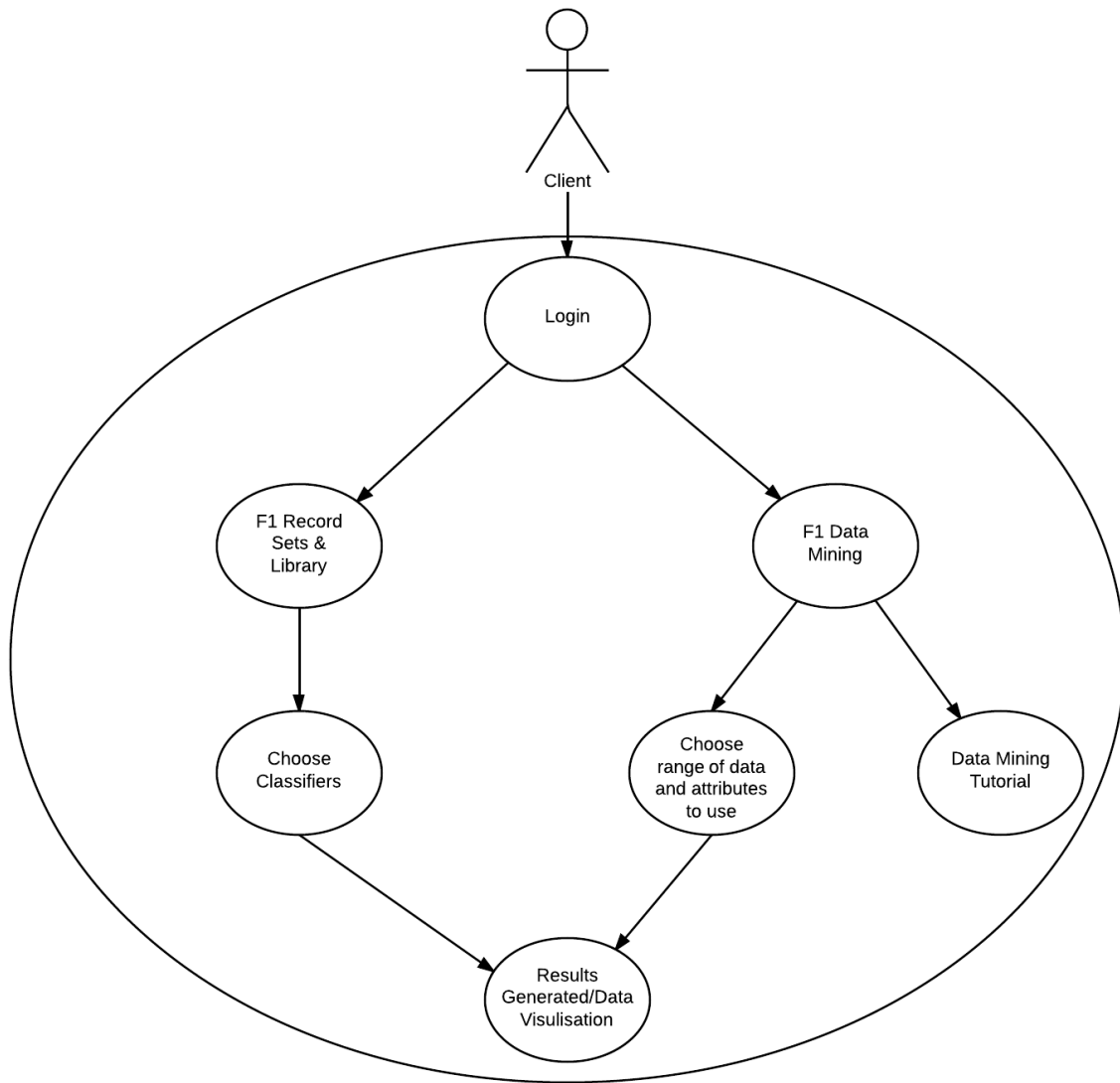


FIGURE 34 INITIAL USE CASE DIAGRAM

During the design and development of the web application, the number of features available to the user expanded, as new ideas were put forward that could be added to the website. This included a user sign in process that would allow users to enter a members league where they could compete at predicting race results, allowing users to import their own non F1 data, to be passed through the linear regression model (this came about from the first sprint 1, as data was hand entered, as a database had not been created at the time) and a members chat section, where they could talk and interact with one another. Other such pages added to the new design was an 'Update Database' page and an about. Below is the use case for the final design of the website.

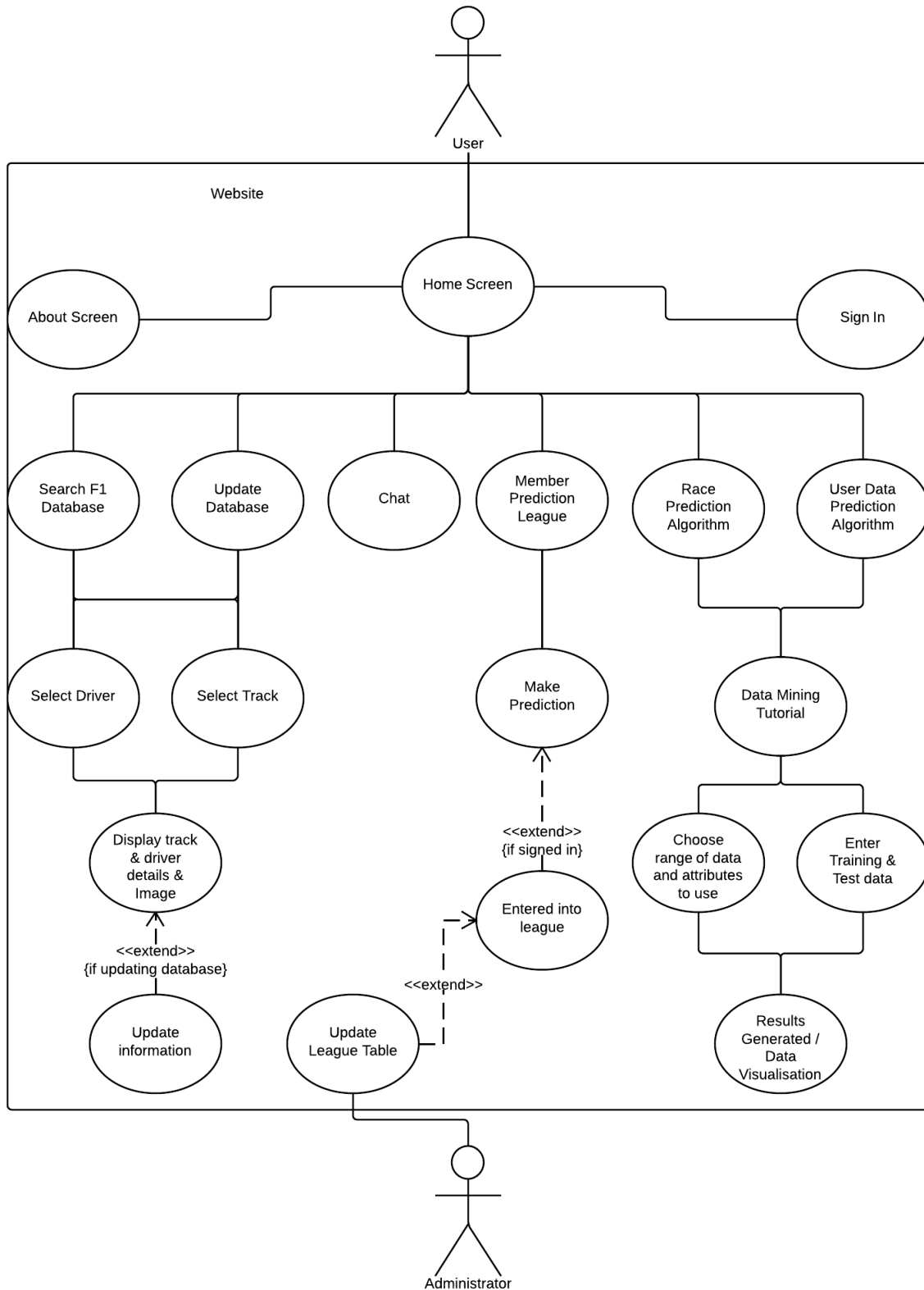


FIGURE 35 FINISHED USE CASE DIAGRAM

To help with the design process of the website, low and medium fidelity diagrams were created, so the layout and design of the website could be finalised. A low fidelity diagram is a basic diagram usually done on paper for a project and a medium fidelity diagram is a slightly

more detailed diagram created in a graphic design program for a project. Below are the low and medium fidelity diagrams for the web application. The low fidelity diagram shown was created earlier in the project and so was based more on the original use case while the medium fidelity diagram was created later in the project lifecycle so relates to the finalised use case.

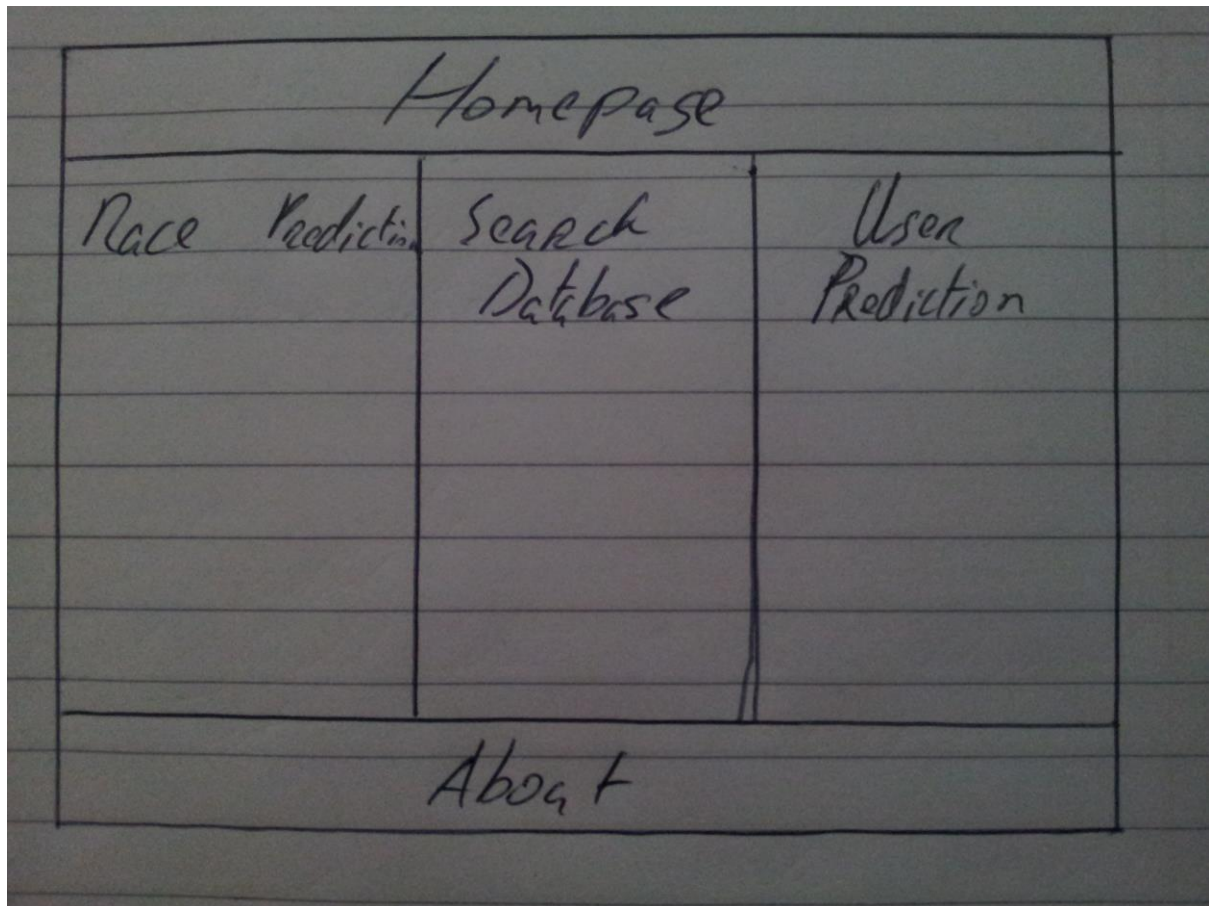


FIGURE 36 LOW FIDELITY DIAGRAM OF THE WEBSITE

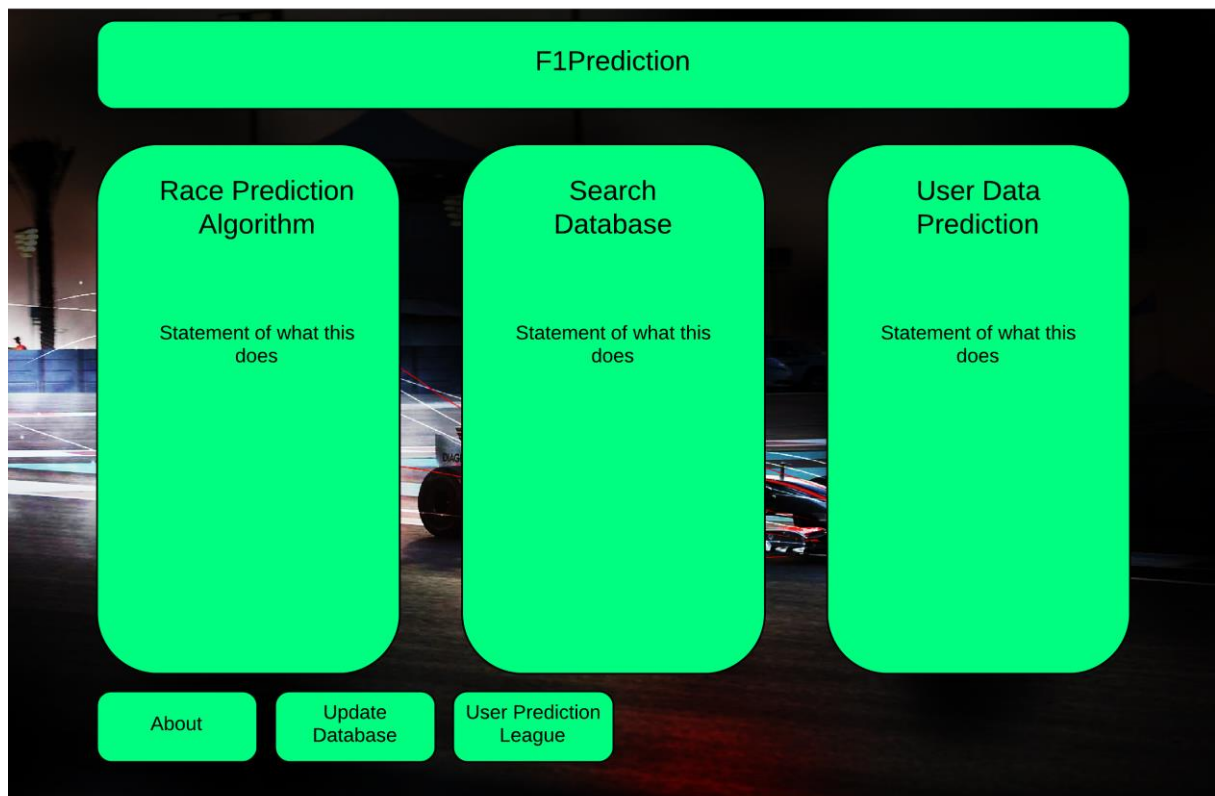


FIGURE 37 MEDIUM FIDELITY DIAGRAM OF THE WEBSITE

The front end website was created using HTML, CSS and AngularJS as the JavaScript framework along with utilising the Greensock JavaScript API that allows complex animations to be created, providing greater performance and flexibility than CSS3 animations.

During the development of the web application, user testing was an important aspect of the design. As iterations of the website were created, I displayed this to users to generate feedback. This feedback helped to shape the finished web application as users pointed out aspects of the web application that they liked and parts that they found difficult to navigate. The HTML structure of the web application was divided into three sections, the header, body and footer, each of which was subdivided into its own individual sections. The styling of this HTML is handled by the theme.css file. This CSS file outlines the styling, alignment, size, colour, etc. of all elements of the web application. The header of the website, was initially planned to be one solid clickable bar. The header bar container, headerContainer, was the parent element of the header and allocated the headers total size and layout of child elements. Within the headerContainer div was the headerBar. This div class outlines the clickable bar that the user can see on the webpage, with text inside of it. Due to feedback from testers and watching people use the website it was obvious users did not understand the header was a clickable button that returned them to the homepage. For this reason, the header bar was altered to provide a more traditional menu style system for users. This menu style system would contain links to the main areas of the website, including the homepage by dividing the headerBar into individual sections/divs. User feedback after making this change, was positive,

with users now better understanding how to navigate the application. The below screenshots, shows the design of the website with the initial single header design and the revised header, altered due to user feedback can be seen in the second screenshot.

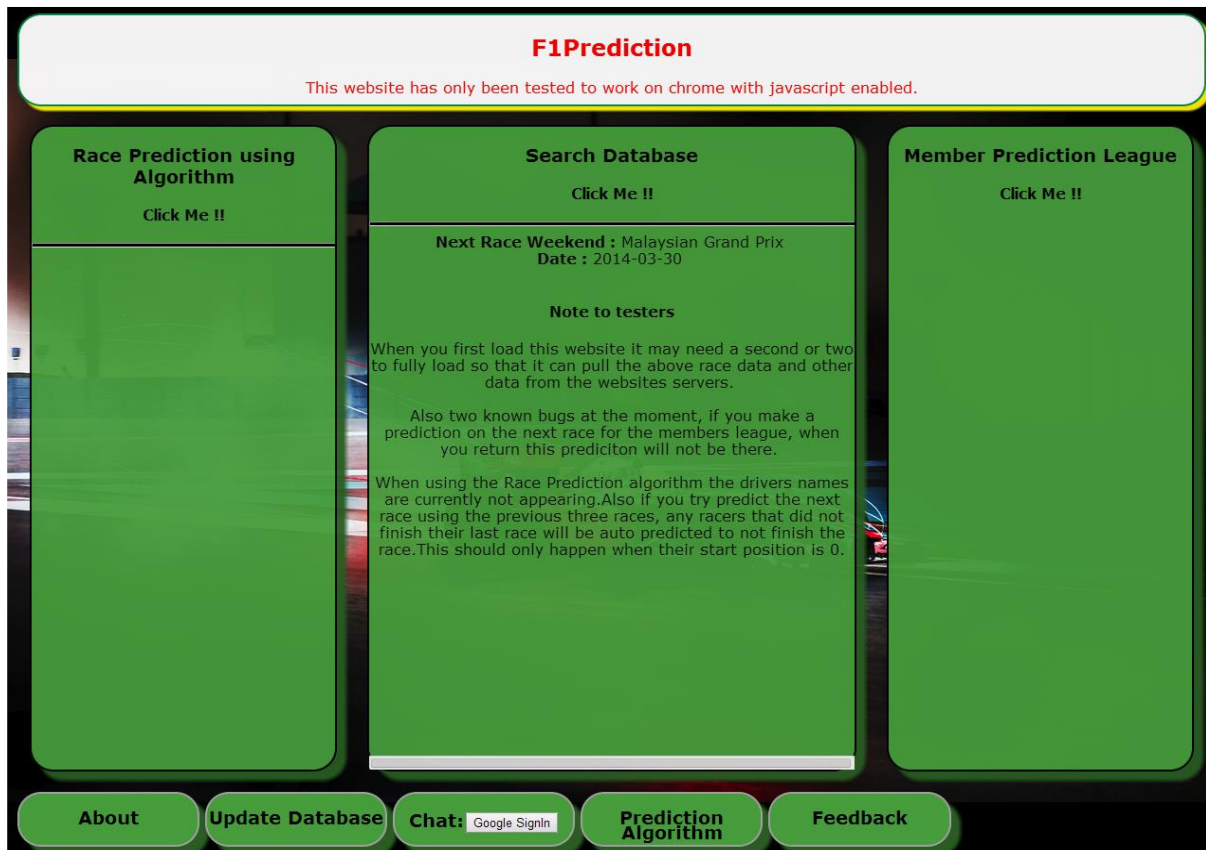


FIGURE 38 INITIAL APPLICATION HOMEPAGE



FIGURE 39 APPLICATION HOMEPAGE AFTER USER FEEDBACK

The body was created using a container div, homeContainer. This homeContainer, similar to the header container, outlined the area of the screen that belonged to the body of the

website and as such the distance from between it and the headers and footers. Within the homeContainer, three sub divs were created (homeBlock1, homeBlock2, homeBlock3), these are the three columns a user see's displayed on screen. These three divs are containers themselves, each containing multiple divs for the pages that are displayed within them, such as mainBlock1 div (left most column of the home page) and racePredicitionBlock3 (right most column of the race prediction page). These divs contain the individual page content, for each page. When the page loads initially, these divs CSS properties are set by JavaScript to either hide or show the page. When a user clicks on one of the homepage columns the AngularJS event handlers call the appropriate function to serve the next page. For example if the user clicks on the middle column, 'Search Database', the event handler ng-click is alerted which in turn calls the function mainBlk2Click. For each page, there is a similar function to hide the current page and show the new page. The mainBlk2Click function stores the ID of the three columns (homeBlock), the page to be hidden (mainBlock) and the page to display (searchDatabaseBlock) in variables. These variables are then passed into a Greensock animation timeline. In the case of mainBlk2Click, the Greensock timeline changes the height of the three columns to 0 along with the opacity. Following this it then changes the CSS style of the mainBlock to none, hiding these divs from the user and changing the display block of the searchDatabaseBlock's to an inline-block, making their content visible. Finally the height of the columns is set back to 100% and their opacity to 1. This timeline code, achieves the goal of providing an attractive user interface for the user while gracefully hiding the previous page and displaying the new one. Code snippet below.

```
//function to bring up search database page for when user clicks homepage block 2
$scope.mainBlk2Click = function()
{
    var block1Transition = document.getElementById("homeBlock1");
    var block2Transition = document.getElementById("homeBlock2");
    var block3Transition = document.getElementById("homeBlock3");

    var main1Transition = document.getElementById("mainBlock1");
    var main2Transition = document.getElementById("mainBlock2");
    var main3Transition = document.getElementById("mainBlock3");

    var search1Transition = document.getElementById("searchDatabaseBlock1");
    var search2Transition = document.getElementById("searchDatabaseBlock2");
    var search3Transition = document.getElementById("searchDatabaseBlock3");

    var tl = new TimelineLite();

    //greensock api to make search database page visible, hide homepage and run animation
    tl.staggerTo([block2Transition,[block1Transition,block3Transition]],1, {height:"0%"},0.4)
    .staggerTo([block2Transition,[block1Transition,block3Transition]],1, {opacity:0},0.4,"-=1.5")
    .to([main2Transition,main1Transition,main3Transition],0.0,{display:'none'},"-=0.3")
    .to([search2Transition,search1Transition,search3Transition],0.0,{display:'inline-block'},"-=0.3")
    .staggerTo([block2Transition,[block1Transition,block3Transition]],1.3, {height:"100%"},0.4,"-=0.3")
    .staggerTo([block2Transition,[block1Transition,block3Transition]],2.5, {opacity:0.9},0.4,"-=1.9");

    //need to get all driver and track names so they can be used in search fields
    $scope.getTrackandDriverData();
}
```

FIGURE 40 JAVASCRIPT CODE OUTLINING CODE USED TO CHANGE WEBSITE PAGE

As can be seen from the above code snippet, a second function, getTrackandDriverData is also called. As the front-end was developed, functions were created to pull data from the server. This meant updating and creating new servlets and Java classes on the backend server to cater

to these new requests. Along with this a database was created to store data for individual drivers, tracks, events and the previous race results dataset, used in the linear regression race prediction. The image below shows the class layout of the complete server.

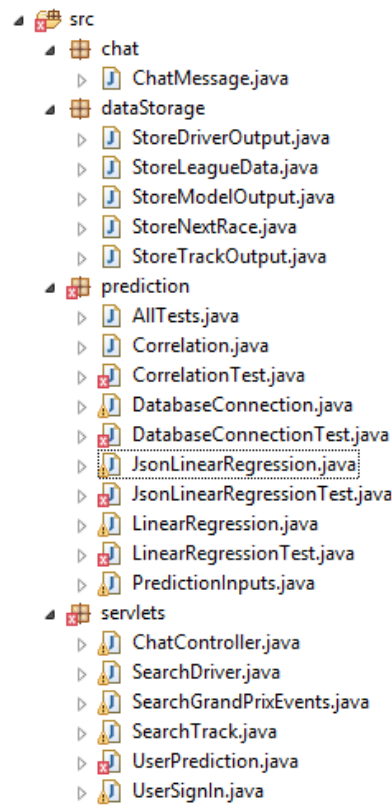


FIGURE 41 BACK-END SERVER CLASS STRUCTURE

The footer bar of the website was created in a similar fashion to the revised header bar. The footer bar container, footerContainer, was the parent element of the footer and allocated the footers total size and layout of child elements. Inside the footerContainer, individual child divs were created for each footer section/div with links to their respective functions.

The additional servlets and classes are designed to communicate in a similar fashion to how the original linear regression method functioned from Sprint 1, using Ajax calls to pass JSON data structures between client and server. The addition of a database allowed the JsonLinearRegression class to now also query the database for the race dataset as well as taking datasets from the user. The following subsections of this paper will detail the new features of the website that were developed during sprint two, giving an overview of their design and examine the relevant classes and their JavaScript functions. Along with this I will detail the design and implementation of the database and the alterations made to the JsonLinearRegression class to use this database.

Database

A database was created using MySQL Workbench, SQL and the MySQL Workbench plugin for Excel in a local instance for testing purposes during development. When complete the local database would be uploaded to an instance of Google Cloud SQL. The design of the database

focused on three areas, storing race results from the master Excel dataset, storing track and driver information to allow users to search through information and to store member's data and their race predictions. The design of the race results table followed the design of the initial master Excel workbook and the chosen explanatory variables. One table was created that would house this data for each individual race of the previous four years. To avoid the overuse of driver names, tracks and seasons as is evident from the original master Excel workbook, the table was normalised and three separate tables were created. This allowed the database to easily store added information for the drivers, tracks and race events that could be presented to users without duplicating large amounts of data. The MySQL Workbench plugin for Excel allowed the Excel data to be easily uploaded to these tables, without having to hand code over two thousand SQL insert statements. Along with these tables a usersPredictionLeague was created that would store the details of users of the website that were taking part in the members prediction league. From this a child table was created that would then store the data for each user for each given race weekend, userRacePredictions. This table was linked to the race weekend table and the drivers' table. If this database was to be recreated, the design of the userRacePredictions table would be altered as the current design contains a lot of clutter within the ERD, focusing on joins from individual results in the userRacePredictions table and the drivers table. This could be alleviated by creating a child table off userRacePredictions, where each individual driver prediction would be contained as its own row of data, leading to a single column joining to the drivers and userRacePredictions table. The complete entity relationship diagram is shown below.

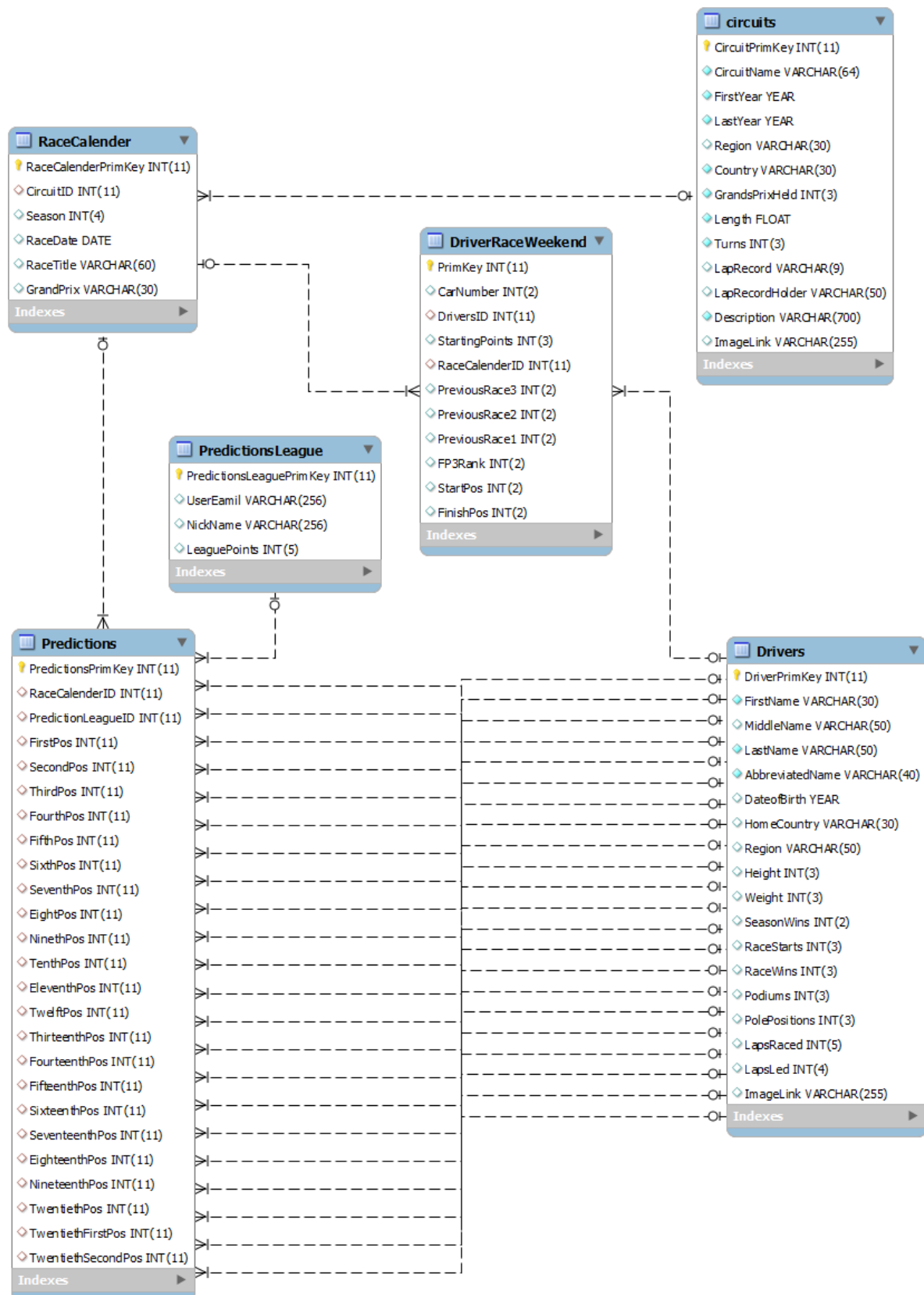


FIGURE 42 DATABASE ENTITY RELATIONSHIP DIAGRAM

To connect to the database, a database class was created within the server. The database was designed so that it could be instantiated by other classes and functions could be created to

pull specific information from the database for a given class or a class could pass a string into the database to return a single column of numbers or strings. To achieve this a constructor was created that allowed the database to connect to the relevant database, local MySQL Workbench within the development environment and Google Cloud SQL within the production environment. The constructor checks the system environment variables and then stores the correct URL location of the database given the environment.

Search Database

The search database function was added to the project to allow users to search for their favourite driver and or track. Upon clicking the 'Search Database' button in the middle column of the home page the JavaScript function `mainBlk2Click` is called, which can be seen previously in [FIGURE 40](#). A similar function is called when the user clicks the 'Search Database' button in the header, but without the Greensock animation. As described before this function changes the page to the 'Search Database' page and then calls the `getTrackAndDriverData` function. This function is used to contact the `SearchTrack` and `SearchDriver` servlets, both of which function in a similar matter, so I will outline how the `searchDriver` servlet operates. Within the `searchDriver` servlet a new instance of the database class is instantiated before calling the `getDriverData` function of the database class. This function within the database class was created to return all relevant information from the database table `Drivers` and returns an array of the class `StoreDriverOutput`. This class is similar to the `StoreModelOutput` described previously allowing for the storage of driver data within the Java code and allowing easy transition to a JSON structure for transfer to the client.

The `getDriverData` function calls the `querySetUp` function in the database class. This function is utilised to execute all query's that are made within the database class. The function first uses the URL generated within the constructor to make a JDBC (Java Database Connectivity) driver to connect to the database. The function then alters the database query to insert a count function to determine the number of rows returned before executing the driver's query which is stored in the `resultSet`. This `resultSet` variable can then be transferred into the `StoreDriverOutput` which can then be returned to the client for processing by the servlet.

When the JSON data is returned to the client, AngularJS stores this data within a variable of an array of objects, `DriverSearch`. AngularJS utilises this array to access the driver data columns, stored in the individual objects. This is presented to the user using a HTML select statement and displays the `driver.AbbreviatedName`. When a user selects a driver's name from the select statement in the leftmost column, the drivers data is populated in the middle column along with an image of the driver appear below the select statement. This is achieved as AngularJS updates `DriverSearch` to reference to the currently selected driver. This is a similar process to how the application returns the track information. A screenshot of the 'Search Database' page can be seen below (this screenshot was taken using the final build of the application, after changes in design outlined in the 3rd sprint).




FIGURE 43 SEARCH DATABASE PAGE

User Sign In

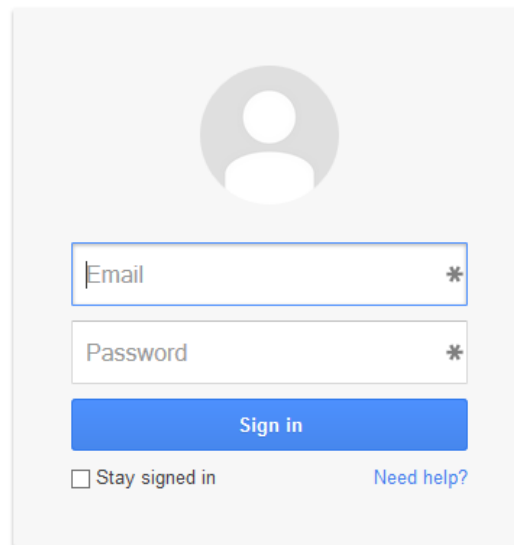
To encourage a user community style web application it was decided that an important aspect of this would be allowing users to login into the website. Allowing users to login could expand the scope of the website to allow them to interact more with the application and other users. This could be achieved by creating a members league where users could compete against one another to predict the next F1 race.

Sign in through Google accounts was utilised for this as it allowed users to sign into the web application without having to create a new account, appealing to some users. This also meant that a user sign up process was not needed to gather user information and my web application would not have to be concerned with storing private, sensitive information. As detailed in chapter two this is a security risk with web applications, as storing sensitive information that may be used against a user in the case of a security breach such as their password can be dangerous. As I utilised the Google logins, their private information such as their password would remain with Google and I would never get access to this, removing the responsibility of security from my application. Below is a screenshot which illustrates the Google sign in page.



One account. All of Google.

Sign in with your Google Account



The sign-in form is a light gray box containing a large gray person icon at the top. Below the icon are two input fields: 'Email' and 'Password', each with a blue border and a small asterisk icon on the right. Below these fields is a blue 'Sign in' button. At the bottom of the box, there is a checkbox labeled 'Stay signed in' and a blue link labeled 'Need help?'.

[Create an account](#)

One Google Account for everything Google



FIGURE 44 GOOLE USER LOGIN

When a user first loads the website, the application must first check to see if the user is already logged into the application from a previous visit. This is achieved through the use of the AngularJS function `$scope.$watch('$viewContentLoaded', function())`. This function is called when the content of the page begins loading. The function then sends a request to the `UserSignIn` servlet with the header `checkSignIn`. This servlet checks the parameters of the request to see if it contains user information to determine if a user is signed in.

The servlet achieves this by utilising the user's library provided by Google, as a part of App Engine. The servlet gets the `currentUser` information from the request and if this data is null then a false message is returned to the client. Likewise if the `currentUser` is not null a true message is returned to the client. With this information, the AngularJS function stores the

true or false message in the scope variable SignedIn. This SignedIn variable is used to determine whether the sign in button should be displayed to the user or not.

If a user has not signed into the application on this visit then a Sign In button is presented to the user within the header. When the user clicks this button the AngularJS function SignIn is called. This function contacts the UserSignIn servlet with the header SignIn in order to receive a web link to redirect the user to, to allow them to sign into their Google account. The servlet creates this URL by utilising the createLoginURL call from the userService library provided by App Engine. This URL is then returned to the client where the browser can be redirected to the link. A code snippet of the UserSignIn servlet is shown below.

```

if("checkSignIn".equals(RequestType))
{
    if (user == null)
    {
        test = "false";
    } else
    {
        test = "true";
        System.out.println(user.getEmail());
    }
    resp.setContentType("application/json");
    resp.setCharacterEncoding("UTF-8");
    resp.getWriter().write(test);
}
else if("SignIn".equals(RequestType))
{
    if(user != null)
    {
        resp.sendRedirect("/");
        System.out.println("in here");
    }
    else
    {
        test = userService.createLoginURL(req.getRequestURI());
        System.out.println(test + " : out here");
    }

    Gson ReturnedJson = new Gson();

    //The string is initialised to }}}',\n This is a specific string
    //that is used to protect against a Json security vulnerability
    String ReturnedjsonStr = "}}}',\n";
    ReturnedjsonStr += ReturnedJson.toJson(test);
}

```

FIGURE 45 USERPREDICTION SERVLET

Members Prediction League

When a user clicks on the rightmost column of the homepage, 'Members Prediction League', a JavaScript function mainBlk3Click is called. A similar function is called when the user selects

the 'Members Perditiion League' button in the header, but without the Greensock animation. This function is used to change the page and also call the functions getNextRace and getUserLeague. The getNextRace function is used to get the information on the next race in the F1 calendar to return this to the user so they are aware of the race they will be predicting the results for along with returning the data for a user's previous predictions for this race, if they are returning to alter their prediction. This function queries the userPrediction servlet to get the needed information. The userPrediction servlet was chosen to handle the 'Members Perditiion League' as it is just another form of race prediction, so it was easiest to keep all prediction functions together in the one servlet. To allow the servlet to differentiate the different requests it would receive a header system was implemented. Within the AngularJS Ajax request, an additional header, "Type", is added to the standard HTTP Post headers which can be read within the userPrediction class so the correct code is used. The below code snippet displays this header addition to the Ajax request.

```
//function to get the data of the next race for use on the homepage and the members prediction league page
$scope.getNextRace = function()
{
    //need to get next race information
    //this is used for display on the members race prediction page
    $http.post('/UserPrediction', $scope.returnedNextRace, {headers: {'Type': 'nextRace'}})
        .success(function(data)
        {
            console.log(data);
            if (data == null)
            {
                // if not successful, bind errors to error variables
                alert("Something went wrong :( Please refresh the page and try again");
            }
            else
            {
                // if successful, bind success message to message
                //$scope.returnedTrackData = data;

                $scope.returnedNextRace = data;
                //original data used to compare against when a user changes the position of a driver in their position
                $scope.returnedNextRaceOriginal = angular.copy($scope.returnedNextRace);
            }
        });
}
```

FIGURE 46 GETNEXT RACE JAVASCRIPT FUNCTION

Using this header the UserPrediction servlet can read the type of request so to understand how it was initiated and what information to return to the client. After selecting the correct code block within the servlet and instantiating an instance of the database class it then checks to see if the user is signed into the website using their Google account. If the user is not signed in, then the servlet calls the getNextRace function within the database class. This function gets the next race of the F1 calendar along with each driver taking part in the race. Each driver is assigned a default prediction in the order they are accessed. The return data from the getNextRace function is stored in an array of the StoreNextRace class to be returned to the client. If the user is signed into the application, then the UserPrediction servlet first checks to see if they are already in the member's league. If a user has not entered the league or has entered the league during a previous F1 race but has not made a prediction on the next F1 race, then the same process is followed as if the user had not been signed in. If a user is signed into the application, is a member of the league and has already made a prediction on the next race, then the servlet returns the users current prediction and is stored in the variables

returnedNextRace and returnedNextRaceOriginal. This is achieved by using a function getNextRaceWithPredictions within the database class. This function operates in a similar fashion to the getNextRace function, but returns the users previous prediction along with it.

With all of the data now returned to the user for the 'Member's Prediction League' it was displayed to the user in the three column design of the website. The left column presented the user with the current leader board for all members, the middle column outlined the instructions on how the leader board and prediction worked and the right column displayed the table of drivers that the user used for their prediction. The below screenshot displays the 'Member's Prediction League' page below (this screenshot was taken using the final build of the application, after changes in design outlined in the 3rd sprint).



FIGURE 47 MEMBER'S PREDICTION LEAGUE PAGE

When a user changed a driver's position a JavaScript function updateUserPrediction was called to change the position of other drivers. This would alleviate the problem of two drivers finishing in the same position for a prediction. When a user changed the predicted finish of a driver, the variable returnedNextRace was updated with the new prediction list. The updateUserPrediction function utilised the returnedNextRaceOriginal variable to compare the original returned list from the server to the new list created by the user. When a difference was found in the list the function could determine whether to increase or decrease the position of the driver. The new list is then copied into the returnedNextRaceOriginal variable so that future comparisons can be made on this list. The below code snippet illustrates this function.

```

//function to sort drivers for members league prediction
$scope.updateUserPrediction = function()
{
    //this is used to find the driver that has changed and store his new and old position
    for(var i = 0; i < $scope.returnedNextRace.length; i++)
    {
        if($scope.returnedNextRaceOriginal[i].prediction != $scope.returnedNextRace[i].prediction)
        {
            var oldPosition = $scope.returnedNextRaceOriginal[i].prediction;
            var newPosition = $scope.returnedNextRace[i].prediction;
        }
    }

    //this is then used to change any other values that have been effected by this change
    for(var i = 0; i < $scope.returnedNextRace.length; i++)
    {
        //finds driver that currently occupies the new drivers position
        if($scope.returnedNextRaceOriginal[i].prediction == newPosition)
        {
            if(newPosition > oldPosition)
            {
                $scope.returnedNextRace[i].prediction--;
            }
            else
            {
                $scope.returnedNextRace[i].prediction++;
            }
        }
        //find driver that is in the middle of new drivers decreasing move
        else if($scope.returnedNextRaceOriginal[i].prediction > newPosition && $scope.returnedNextRaceOriginal[i].prediction < oldPosition)
        {
            $scope.returnedNextRace[i].prediction++;
        }
        //find driver that is in the middle of new drivers increasing move
        else if($scope.returnedNextRaceOriginal[i].prediction < newPosition && $scope.returnedNextRaceOriginal[i].prediction > oldPosition)
        {
            $scope.returnedNextRace[i].prediction--;
        }
    }
    //store the new ranking
    $scope.returnedNextRaceOriginal = angular.copy($scope.returnedNextRace);
}

```

FIGURE 48 updateUserPrediction JavaScript FUNCTION

When a user was happy with their prediction selection they could click the ‘Insert Prediction’ button in the middle column. When clicked the function insertUserPrediction was called. This function calls the UserPrediction servlet but with a different type header in the Ajax call, insertPrediction. The Java servlet can interpret this header to select the correct code block. This code block uses GSON to parse the request JSON data, containing the prediction, into a class PredictionInputs. This class was designed to match its variables with the design of the JSON request data and to create the correct SQL statement to either insert the user’s predictions or update their previous prediction.

Update Database

The ‘Update Database’ feature on the website allows a user to update information on a track that they feel may be incorrect or out of date. When a user clicks into the page, they are presented with a selection box in the left column. This selection box, functions in an identical manner to the track selection box on the ‘Search Database’ page, allowing a user to select a track, with its details then displayed on the middle column. The difference in the ‘Update Database’ page is that the track details are displayed in a user editable field. This allows users to alter the information presented to them and then by clicking the ‘Update Database’ button found below the input fields, the new track information is sent back to the server which then

updates the database. The below screenshots display the 'Update Database' page of the website (this screenshot was taken using the final build of the application, after changes in design outlined in the 3rd sprint).

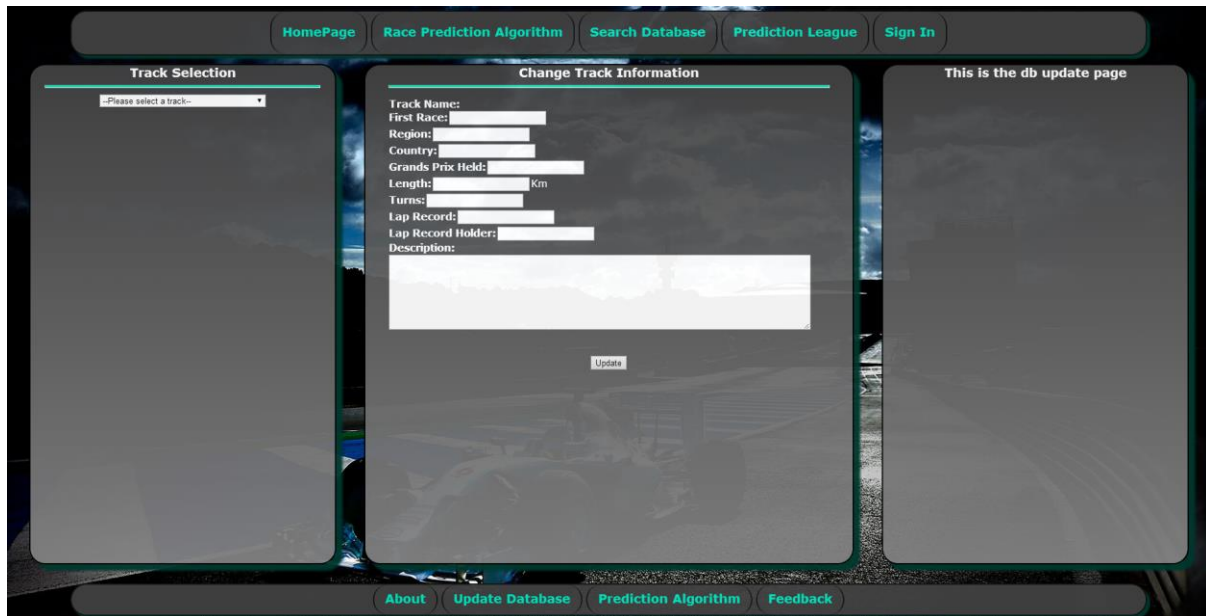


FIGURE 49 UPDATE DATABASE PAGE

When the 'Update' button is clicked, it calls the AngularJS function `submitTrackDataUpdate`. This function calls the `SearchTrack` servlet with the header `trackDataUpdate` to alert the servlet that it is not requesting track data but wants to update the track information, before requesting the new updated information. This header allows the `SearchTrack` servlet to execute the correct code block for this update. The Java servlet first decodes the request string, which contains the JSON data of the track update and stores it in the string variable `temp`. This `temp` string variable is then converted using GSON to an object of `StoreTrackOutput`. This allows all of the fields of the JSON string to be divided up into individual fields. With the individual fields accessible, an SQL insert query is created before calling the `runUpdateQuery` of an instantiated object of the database class. The `runUpdateQuery` is a simple function, designed to only push data to the database and not to return anything. When the update has completed an empty string is returned to the client to notify them that update has occurred. The below code snippet displays the SQL query utilised for the insert function.

```

String RequestType = req.getHeader("Type");

DatabaseConnection dbConnection = new DatabaseConnection();

if("trackDataUpdate".equals(RequestType))
{
    String temp = sb.toString();
    StoreTrackOutput inputData = JsonParser.fromJson(temp, StoreTrackOutput.class);

    //need to now update the database with the new info

    dbConnection.query = "update F1Prediction.Circuits SET CircuitName = '" + inputData.CircuitName
        + "', Firstyear = " + inputData.FirstYear + ", Lastyear = "
        + inputData.LastYear + ", Region = '" + inputData.Region + "', Country = '" + inputData.Country
        + "', GrandsPrixHeld = " + inputData.GrandsPrixHeld
        + ", Length = " + inputData.TrackLength + ", Turns = " + inputData.Turns + ", LapRecord = '"
        + inputData.LapRecord + "', LapRecordHolder = '"
        + inputData.LapRecordHolder + "', Description = '" + inputData.Description + "' where CircuitName = '"
        + inputData.CircuitName + "'";

    dbConnection.runUpdateQuery();
}

```

FIGURE 50 SEARCHTRACK SERVLET

Race Prediction & User Data Prediction

With the creation of the new web application, a new page had to be created for the race prediction algorithm that allowed users to select from data held in the database. For this a new page was created that would display a tutorial to the user on how the prediction model worked and how to use it. After the tutorial the user would be brought to the race prediction page where they could select the race they wanted to predict and they could select the explanatory variables to be used in the prediction. Along with this page, it was decided to keep the old style of race prediction, from the initial web application of the first sprint. This page would allow users to enter their own datasets that may not be relevant to F1, to make predictions using the linear regression algorithm. This would be the user data prediction page and would also contain a similar tutorial page as the race prediction page.

The user data prediction page could be accessed from the footer bar of the new website design but did not require much change from the initial sprint, except for the creation of the interim page for the tutorial and added validation of the input fields. This added validation was performed within the ConvertData function within the JSONLinearRegression class outlined in the 1st sprint. The ConvertData function was extended to check the length of the user entered data in the training and testing fields. All training fields entered by the user had to have the same number of data points when compared to each other. Similarly all testing fields entered by the user had to have the same number of data points when compared to each other. Along with this length check, Java regular expressions were utilised to determine if the content of the user inputted data was correct (contained only numbers, decimal points and commas to divide the data points). Other than these changes and the small change of the addition of a header in the Ajax call to the UserPrediction servlet the prediction process remained the same. The below code snippet displays a part of the data validation performed in the ConvertData function.

```

//declare variables for regular expression comparison
Pattern pattern = Pattern.compile("^0-9,.*$");

//if statements to count number of columns,
//make sure all explanatory and response variables
//are the same length and that where an explanatory
//variable is found a response variable must be found
if(TrainingVar1 != null)
{
    TrainingVar1Array = TrainingVar1.split(",");
    TrainingLength = TrainingVar1Array.length;
    if(TestingVar1 == null || false == pattern.matcher(TrainingVar1).matches() || false == pattern.matcher(TestingVar1).matches())
    {
        return false;
    }
    else
    {
        columnCount++;

        TestingVar1Array = TestingVar1.split(",");
        TestingLength = TestingVar1Array.length;
    }
}
else
{
    return false;
}
if(TrainingVar2 != null)
{
    TrainingVar2Array = TrainingVar2.split(",");
    if(TestingVar2 == null || TrainingVar2Array.length != TrainingLength || false == pattern.matcher(TrainingVar2).matches())
    {
        return false;
    }
    else
    {
        columnCount++;

        TestingVar2Array = TestingVar2.split(",");

        if(TestingVar2Array.length != TestingLength || false == pattern.matcher(TestingVar2).matches())
        {
            return false;
        }
    }
}
if(TrainingVar3 != null)
{

```

FIGURE 51 CONVERTDATA FUNCTION

The race prediction page could be accessed from the header and the right column of the homepage, when upon clicking the user would be presented with the tutorial page, using similar JavaScript functions outlined before to change page. The tutorial page was designed to give the user an understanding of what data mining is in the leftmost column, an overview of the dataset and the linear regression model used in the middle column and a description on how to use all the functions of the race prediction page in the rightmost column. When the user was finished reading through the tutorial they could press the continue button which would bring them to the prediction page again utilising JavaScript functions similar to those already described. The below image displays the tutorial page (this screenshot was taken using the final build of the application, after changes in design outlined in the 3rd sprint).

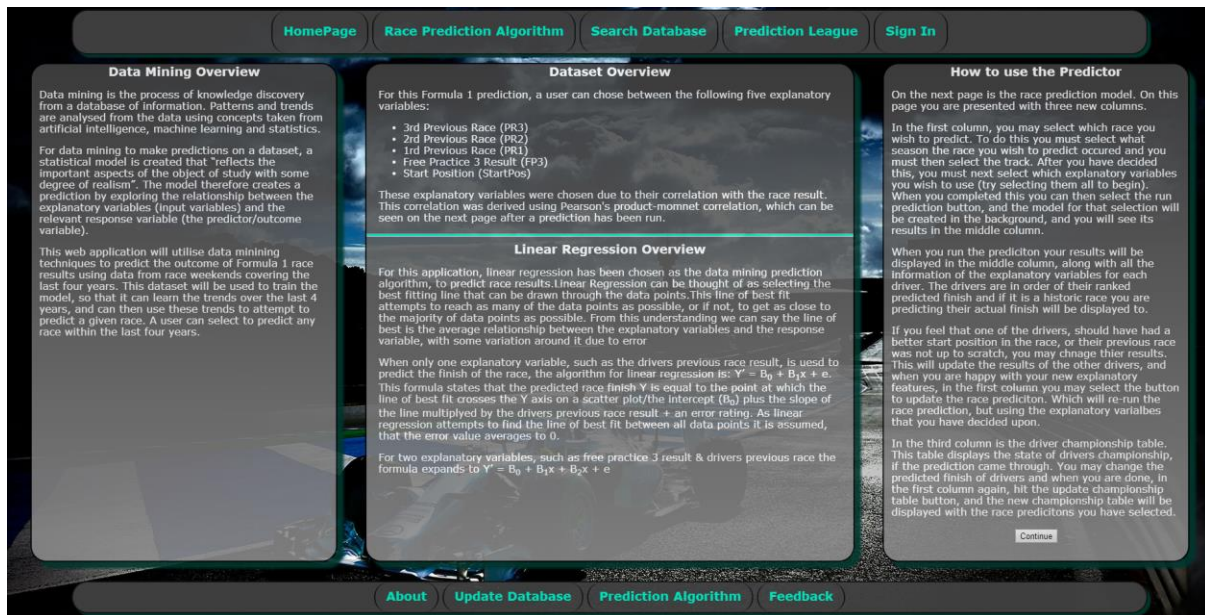


FIGURE 52 RACE PREDICTION TUTORIAL PAGE

On the race prediction page the user was presented with a season and track selection in the leftmost column. Below this was checkbox's for the explanatory variables selection and a 'Run Prediction' button followed by fields for the error margins from the prediction results. When a user selected a season, an AngularJS function `getGrandPrix` was called. This function queried a new servlet, `SearchGrandPrixEvents`, to get all the grand prix races that were held during that year. This servlet created an SQL query that would return all F1 track names that were used in the requested season and would eliminate races that did not have information for the previous race explanatory variables (the first three races of the season as outlined in the dataset creation & evaluation section). The SQL query was then executed by calling the `runStringQuery` function within an instantiated object of the database class. This function was designed to accept general queries that only request one column of string information, and returned this information in an array of individual strings for each row of data. Unfortunately the setup of this query resulted in each race been returned multiple times per the number of drivers taking part in the race. The use of the SQL keyword `DISTINCT` was attempted but this led to SQL errors been returned in the development environment. For this reason the SQL output variable, `dataOutput`, had to be passed into a Java set. A Java set is a collection of items, but disregards all duplicate variables which was then converted back into an array of strings variable, `dataOutput2`, now only containing distinct circuit names. This variable was returned to the client and was used to populate the track selection field ordered by the spelling of the tracks, on the race prediction page.

The user could now select the race they wanted to predict and the checkbox's for the individual explanatory variables they wished to use. When the user pressed the 'Run Prediction' button, the AngularJS function `processRacePrediction` passed the selected season and track and the selected explanatory variables to the `UserPrediction` servlet with a type header of `updateRacePrediction`.

The Java servlet finds the correct code block using this header information and then stores the JSON data passed into it, in the JSONLinearRegression class, the same class used for the user data prediction functionality of the web application. A separate function `runRacePrediction` was created to handle the prediction of race results instead of the user data. This function was used to query the database for the race prediction dataset. To do this it called a function `BuildSQL`, which was used to build the individual SQL queries needed to pull the training data and testing data and other relevant data from the dataset depending on the user's selection. This included the drivers that took part in the race for the testing data, the championship points the drivers had before the selected race and their actual finish at the race. The `runRacePrediction` function utilised these SQL queries and the newly created functions in the database class `getTrainingData` and `getTestingData`, to build the training and testing datasets to then be passed into the `LinearRegression` class for prediction. The `LinearRegression` class handled this data in the same method as described in the 1st sprint. The `JSONLinearRegression` class stored the prediction results, along with the returned data from the driver names, driver points and actual finish results to the `UserPrediction` servlet using the `StoreModelOutput` class, again in a similar fashion as outlined in the 1st sprint, to then be returned to the client.

When this returned data reached the client, it was placed into two variables, `returnedDataOriginal` and `returnedData`. The middle column of the website contained a table that was used to display the results from the prediction. The data from the `returnedData` variable was used to populate this table to show the user the driver's names, explanatory variables, response variable, ranked response result and actual race result of test set. Along with this the correlation testing and root mean squared error and mean absolute error were returned to the user in the left column.

With the left and middle column containing data, it was decided to incorporate an F1 championship leader board into the right column. This leader board would display a table with all drivers and their championship points they had before the predicted race, and would then show the predicted points after the race, for the given linear regression race prediction. After the AngularJS function `processRacePrediction` stored the race prediction data within the variables `returnedDataOriginal` and `returnedData` it then called the function `UpdateChampionshipTable`. This function looped through the `returnedData` variable and read the drivers current points and their ranked predicted finish position. The function then added on the number of points they would gain for finishing in their predicted place and then storing this as their predicted points to be displayed in the championship leader board. An example of this is if a driver had 100 points and they were predicted to finish in second, their predicted points would be 118. After testing the race prediction page of the web application it was decided to expand the size of the middle column for the whole website. This was due to the amount of room the race prediction results table took up on the race prediction page and to maintain the consistent feel across the website. This change allowed more room for data in the middle column on other pages of the web application, such as the 'Search Database' page

too. A screenshot of the final race prediction screen is show below (this screenshot was taken using the final build of the application, after changes in design outlined in the 3rd sprint).



FIGURE 53 RACE PREDICTION PAGE

Now that all the data from the prediction was presented to the user, it was decided to allow the user to alter the data. In the championship leader board table users could change the predicted finish position of the driver. A similar function to the updateUserPrediction as utilised in the members prediction league was created, updateDriverChampionshipRanking, to reevaluate the position of drivers after every position change, to verify drivers could not be predicted to finish in the same position. When a user had completed moving the drivers around they could press the button "Update Championship Table" found in the left column. This button would recall the AngularJS function UpdateChampionshipTable to re-evaluate each drivers predicted championship points

A similar idea was used in the race prediction table in the middle column. It was decided to allow users to alter the response variables of drivers, so if they felt a driver should have had a better finish in a previous race, they could change this value and see how it would affect their predicted race result. Again when a user changed the explanatory variable for a driver, a similar AngularJS function was called to updateUserPrediction as used in the 'Members Prediction League' to rearrange driver positions to avoid errors. Each column had its own function for this, e.g. changing the previous race explanatory variable called the function updatePR3. When the user was finished altering the position of drivers, they could click a button 'Update Prediction Table'. This button would re-run the race prediction but with the new explanatory variables. This worked in a similar fashion to the normal race prediction outlined just prior in this section except for some small changes.

When the user selected the 'Update Prediction Table' button, the explanatory variables from the table were sent to the UserPrediction servlet using the AngularJS function updateRacePrediction with the type header updateRacePrediction which stored the user

altered explanatory variables in an array of objects for StoreModelOutput class. This array of object was then passed into a new function updateRacePredicition in the JSONLinearRegression class. This function is similar to the runRacePredicition function in the JSONLinearRegression class as outlined before but does not request the testing dataset, driver's names and driver's points from the database, as these variables are already present in the array of objects passed into it. The remaining parts of the function work in a similar method to the normal race prediction, to run the linear regression algorithm and return and display the results to the user.

5.2.3. Sprint 3 – Testing & Evaluation

The third sprint in the application development process focused on creating a tested and complete web application. The finished web application needed to allow users to easily navigate the website and allow them to interact with the websites features while avoiding any errors that may occur, but alerting the user when they occur and how to fix the problem. The three types of testing used in the creation of the web application included unit testing, user testing and developer testing.

Unit Testing

Unit testing is a form of black box testing, allowing a system to be automatically tested, while only worrying about the input and output data, not considering the inner workings of a function. Unit testing was utilised during this project, but only to a small degree, three Junit test classes were created, CorrelationTest to test Pearson's Correlation formula, DatabaseConnectionTest to test the functions of the database class and JsonLinearRegressionTest to test the functions of the JSONLinearRegression class.

The CorrelationTest class was used to compare correlation results created in Microsoft Excel to the results created by the Correlation class of the application using the same test data. The test passed meaning, the algorithm was running correctly.

The DatabaseConnectionTest class was used to test the database connection class of the application to determine if it was returning the correct data for a race. Unfortunately a problem raised with Junit unable to connect to the database occurred as it couldn't connect to the local database using the JDBC driver. This was not a problem that was effecting the development of production environment application, so was put down to a Junit error.

The JsonLinearRegressionTest class was used to compare the linear regression prediction results of the application against some simple datasets. One problem was raised from this testing was that if a driver did not finish a previous race or the free practice they would be placed at an advantage to someone who did complete these sessions. From examining the problem this was due to drivers who did not finish a race or practice session been placed as 0 in the database. This is a similar problem to when a driver was not starting a race but was still ranked as discussed before. To avoid this issue, the algorithm was altered so that when a race was been predicted that any explanatory variables other than the starting position that were

equal to zero would be rounded up to 24 (worst case scenario between races of length 22 and 24). The Junit test that alerted this problem is shown in the code snippet below.

```
@Test
public void testRaceLinearRegression()
{
    //simulate race event were training var is previous race
    //can't use runRacePrediction as that will only use SQL for data

    TestModel.TrainingVar1 = "3,4,5,8,7,9,2,6,19,1,11,0,18,0,10,0,12,13,15,14,16,17";
    TestModel.TrainingVar1 = "1,2,5,4,10,15,7,8,6,3,11,18,9,12,17,16,13,14,22,21,19,20";
    TestModel.TrainingResp= "3,6,2,4,9,11,1,10,20,5,22,13,8,7,21,14,12,19,16,18,15,17";
    TestModel.TestingVar1="6,7,0,9";
    TestModel.TestingVar1="4,5,9,10";

    StoreModelOutput[] Outcome = TestModel.RunLinearRegression();

    assertEquals(1,Outcome[0].PredictedFinishPosRanked,0);
    assertEquals(2,Outcome[1].PredictedFinishPosRanked,0);
    assertEquals(4,Outcome[2].PredictedFinishPosRanked,0);
    assertEquals(3,Outcome[3].PredictedFinishPosRanked,0);
}
```

FIGURE 54 JSONLINEARREGRESSIONTEST UNIT TEST

User Testing

Throughout the development of the web application frontend, user testing was an important aspect and helped design the finished website. During the 2nd sprint mostly friends and family were asked to test my website but for the 3rd sprint a wider audience was utilised. To attract this wider audience, the website was posted onto a popular Irish forum under the website testing category along with users from the 2nd sprint who were asked to revisit the website and try out the now finished functionality of the website. To gather more detailed user information during the 3rd sprint, a feedback link was added within the footer that redirected users to a google doc's questionnaire where they could answer simple questions to understand the type of user they are, outlining any problems they had with the website and leave any additional feedback comments. The two below screenshot shows the questions asked in this questionnaire.

<http://www.boards.ie/vbulletin/showthread.php?t=2057181839>

Usability Testing & Bug Report

Thank you for visiting F1Prediction.appspot.com and providing feedback on it. This feedback will allow me to better understand my user base, find unknown bugs, problems and poor design features of the website and thus help me evaluate my development methods.

Have you tested the website?What did you think?What should I change?Did you find a bug? Let me know below - Eoin Farrell

Are you a Formula1 fan?

1 2 3 4 5
Not interested ☐ ☐ ☐ ☐ ☐ Very interested, don't miss a race

Do you have any prior knowledge of data prediction & data analytics?

1 2 3 4 5
None ☐ ☐ ☐ ☐ ☐ Expert

Do you believe it is possible to predict the winner of sports competition using maths?

1 2 3
No ☐ ☐ ☐ Yes

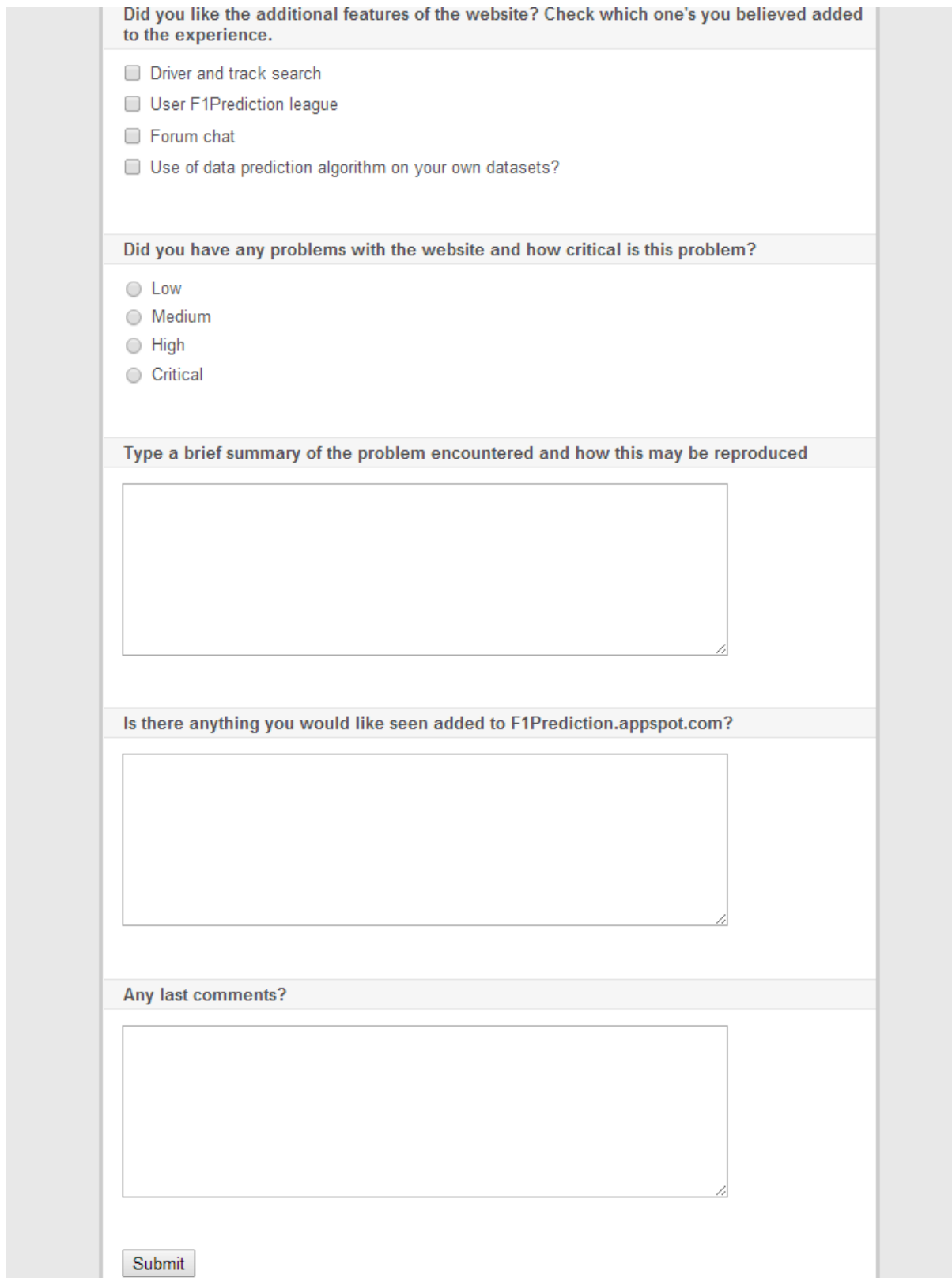
Do you currently take part in any fantasy football style competitions?

☐ Yes
☐ No

Did you like the design of F1Prediciton.appspot.com, please leave comments at the end

1 2 3 4 5
Hated it ☐ ☐ ☐ ☐ ☐ Loved it

FIGURE 55 USER FEEDBACK FORM PART 1



Did you like the additional features of the website? Check which one's you believed added to the experience.

- ☐ Driver and track search
- ☐ User F1Prediction league
- ☐ Forum chat
- ☐ Use of data prediction algorithm on your own datasets?

Did you have any problems with the website and how critical is this problem?

- ☐ Low
- ☐ Medium
- ☐ High
- ☐ Critical

Type a brief summary of the problem encountered and how this may be reproduced

Is there anything you would like seen added to F1Prediction.appspot.com?

Any last comments?

Submit

FIGURE 56 USER FEEDBACK FORM PART 2

From talking to users and the answers in the questionnaire, the following points were brought to my attention, in some cases by multiple users.

Does the spinning animation need to be so prolonged?

While talking to users and from the feedback report, it was brought to my attention that while a lot of users liked the addition of the animation on the website, some users didn't like the spinning animation present on the race prediction tutorial page. One user questioned why did the spinning need to be so prolonged while another didn't think it was as attractive as the other animation styles used on the site. From this user feedback it was decided to reduce the length of time it took for the spinning animation to complete instead of removing it altogether as only two users raised the issue. The total time from start to finish was reduced from 2.5 seconds to 1 second and the columns went from spinning 720° to 360°.

Reorganise the member's race prediction driver order

In the feedback report, one member questioned the layout of the driver's prediction table in the 'Members Prediction League'. Users are given the list of drivers for the next race in one column along with their start position in the second column. In the third column the user inputs the position they believe the driver will finish. This feedback query believed that it should be done in the opposite fashion, the user is presented with a list of numbers and then they must pick which driver will finish in that position. This was considered for change and would allow an easier prediction for users that is more focused on choosing the driver then choosing the position. This would also follow the selection design utilised by F1Predictions.net, which was outlined in the research section. Unfortunately due to time constraints this feedback item was listed as a low priority as the system currently in place was fully working. This feedback item will be looked into, in the future of the website though.

Standardise the image sizes in the 'Search Database' section

This was addressed by one user in the feedback section who liked the 'Search Database' section of the application, believed the driver images should be standardised, so that all images were a consistent size. This was deemed to be a technically easy fix for the website, as it just required swapping the image files. The problem in fixing this though is finding consistent images of the drivers. The majority of the drivers, those taking part in the 2014 season are all sized to 130px*185px, due to all driver images been available on the official F1 website. For drivers from previous years, google images was used to gather photos. This feedback item will be looked into in the future of the website though as it is a technically easy but possibly time consuming fix.

Don't like website colour design

During the user testing of the application, the colour design of the website was the point most raised by users and was thus decided the most urgent feedback change need. One user summed it up as follows: *"A big part of the appeal of formula one is the high tech, flashy nature of it. Any website or app that wants to seriously attract interest in this area must reflect that above all in its design. Love the background pictures but the menus need to be similarly exciting!"*. After reading this feedback from a number of users, I realised the green columns of the website were not the best fit and would have to be changed. I first looked over the

official Formula 1 website, to get an idea of the colours they use. From analysing there website the following colours were outlined as displayed in the picture below.

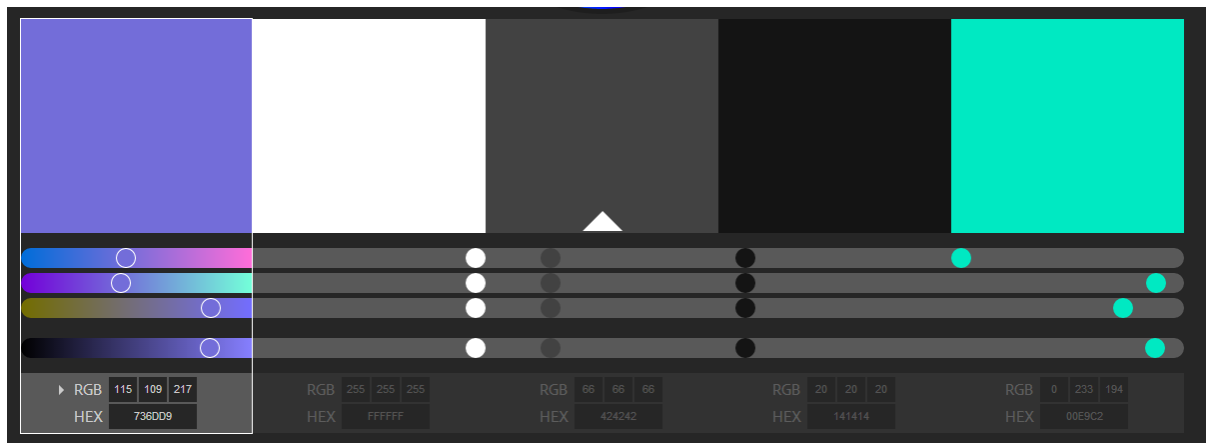


FIGURE 57 COLOUR CHART FOR APPLICATION DESIGN

With these colours in mind, I decided to use the white, grey, black and turquoise as the main colours for my website as they matched the team colours of the Mercedes F1 team. As I was using the Mercedes F1 team colours I also decided to change the background image to one from their official website. Along with this the header and footer bars were changed from individual divs to a connected div. To highlight to users that these individual sections/divs of the header and footer bar were clickable, when the user hovers over the sections, the background colour would change to a black and the mouse icon changed to a pointer. In the future this idea could possibly be expanded to allow users to select the design of the website based on their favourite team. After going back to users with the new website design the feedback was a lot more positive.

No browser back button & refresh page doesn't work

This problem was raised by one user and does highlight a usability problem with the application design and single page websites in general. From examining the issue it seems new controls exist in HTML5 that allow developers to alter a client browser's history [46]. This would allow control over the browser back, forwards and refresh buttons. From this research it was decided that the change could be made to the website, if time was left over at the end, but unfortunately this was not the case. In future development this usability problem will be addressed.

Easier access to driver information while making a prediction

The above problem was mentioned by my supervisor, Deirdre Lawless, when she was attempting to make a prediction on a race. Deirdre believed when a user was making a race prediction it would be nice for users if they could quickly get information on a driver. This feedback item was determined to be an easy addition to the website that would make the

<http://www.mercedesamgf1.com/en/>

website much easier to navigate for the user. This function was developed, so that when a user is displayed a list of driver names they could now click on the driver name and it would redirect them to the 'Search Database' page and the driver would be populated with the driver they selected. Along with this, the race track they are trying to predict would be loaded into the track section of the 'Search Database' page. A new AngularJS function `getDriver` was created to perform this function which took in the driver name and track name as parameters. This function would call the `headerDBClick` function to change to the 'Search Database' page and would then call the `getTrackandDriverData` function, while passing in the driver and track parameters. The `getTrackandDriverData` function was altered so that when a driver and track were passed into it, it would display the chosen track and driver. Otherwise, if a user just clicked on the 'Search Database' button in the header, no parameters would be passed into the `getTrackandDriverData` function and it would not automatically display any driver.

Developer Testing

After allowing users to test the web application, I analysed the website to pick out areas of the website that I was not happy with. One such part of the website was the homepage. The homepage was very empty and bare, the three large columns contained the names of the pages they linked to and a development note to testers on browser support. This went against some of the design requirements researched in the research chapter, outlining how to attract visitors to a website to build a user community. To attract users from the first time they loaded the website, I decided the homepage needed a content overhaul. This would allow the website to advertise itself better to users as it could engage with them by presenting relevant information to them. To facilitate this, it was decided to remove the ability to click the homepage columns to go to the race prediction, 'Search Database' and 'Members Prediction League' pages. These pages were already reachable from the header, and keeping their links in the columns of the homepage would take up needed space. Without removing it, it could also possibly confuse users as they may accidentally click on one of the columns while reading the new homepage content and then unexceptionally be brought to a new page.

The first step in this process was to create a twitter widget for the right column, with F1 drivers past and present and F1 teams and management. The creation of the widget was an easy change to the website, creating the widget using a quickly created list of personalities on twitter and following their instructions on generated the HTML code to display this list. The HTML code was then altered by following the twitter widget API notes on their developer website to remove footers and headers from the widget and to give it a transparent look, allowing it to seamlessly be incorporated into the website design.

A second change to the home page was the addition of a news section. This would attract users back to the website, to read the latest F1 headlines. A list of news headlines were displayed in the left column of the homepage and then when a user clicked a news link it would be displayed in the middle column. Along with this the details of the next race were

<https://dev.twitter.com/docs/embedded-timelines>

displayed below the news headings, giving users details on the next races, the race date and an image of the track. Clicking the race name also redirected the user to the 'Search Database' page, to find more information on the track. The finalised homepage is shown in the screenshot below.

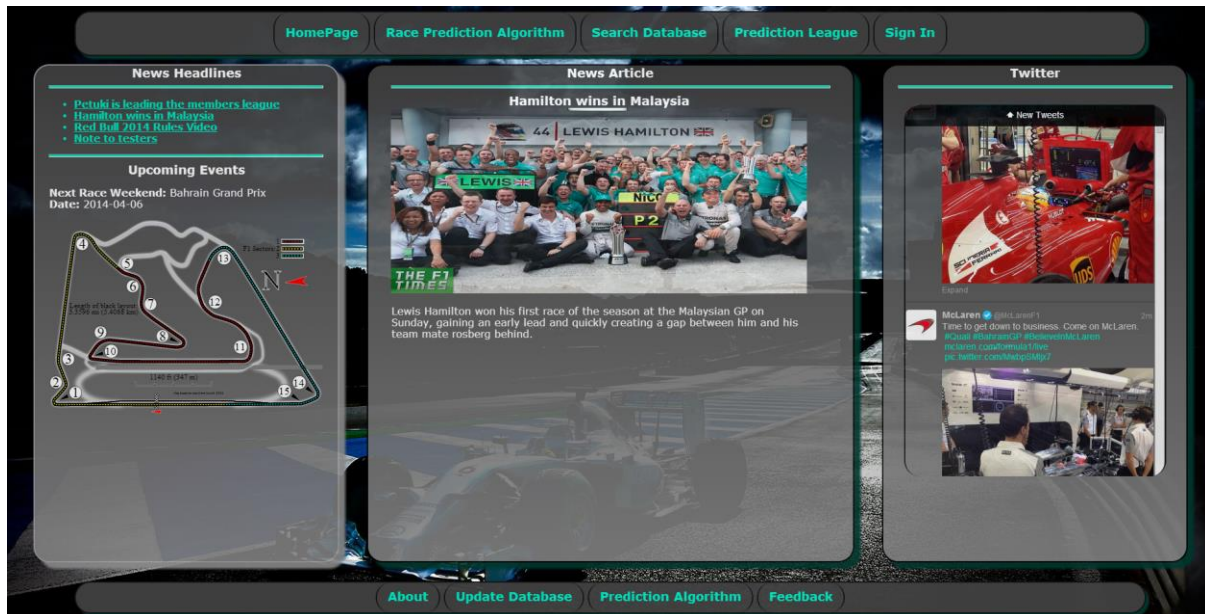


FIGURE 58 FINALISED HOMEPAGE

Google Analytics was utilised for developer testing to gather information on the number of users that were testing my site, information on the devices and browsers they used and to understand how they navigated the application. From looking through the data, the website was tested by 43 unique visitors. These users navigated the website on a number of different devices and browsers and surprisingly a large number of visits were from mobile users. Most of the developer testing was focused aimed towards desktop use, with multiple screen sizes tested ranging from 11in – 24in. With this focus in mind, surprisingly from the mobile users I did talk to, most were happy with the website experience. Understanding the method in which users navigate the site was not possible through the standard implementation of Google Analytics with my web application. This is due to Google Analytics not being able to understand my single page application and cannot differentiate between pages. This can be overcome by tailoring Google Analytics for single page websites and will be looked into in the future. The below diagram outlines the number of visits to the site including numbers with the remaining diagrams in the..... As I used the website myself from both my laptop and my desktop, the numbers may be skewed a bit to my testing environment of Chrome and Windows, but the diagrams still give an indication of the users who utilised the site.

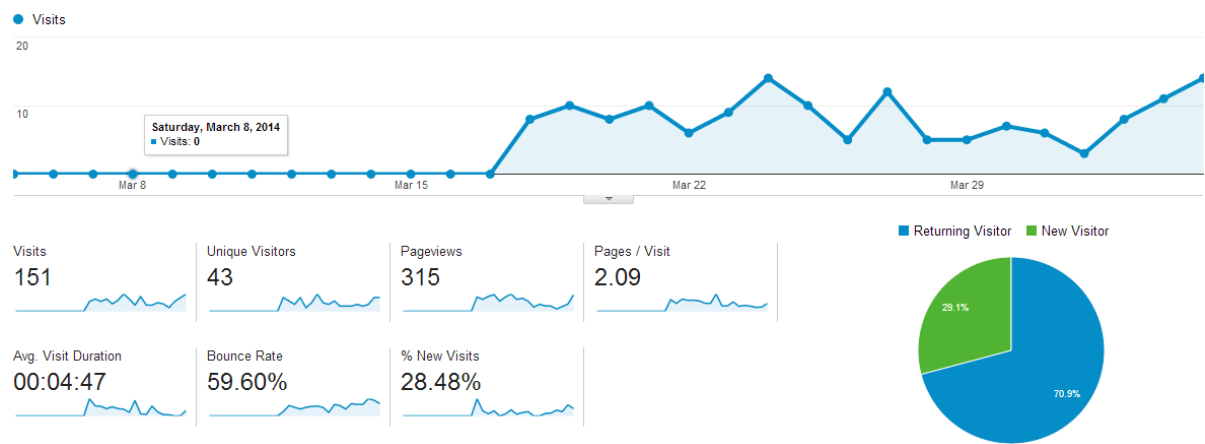


FIGURE 59 GOOGLE ANALYTICS SHOWING WEBSITE USE

6. Future Work, Critical Analysis & Conclusion

This chapter outlines the development that I plan to undertake to improve this project in the future and address areas where problems arose during development. It will also detail the challenges I faced during the development of this application and how these challenges were approached and alternate solutions I considered or could consider in the future. The chapter will conclude by presenting an overall summary of the project and its achievements.

6.1. Future Work

This section will outline areas of the web application that I will revisit in the future to improve.

6.1.1. Multiple Linear Regression Algorithm

The current web application is utilising a form of simple linear regression and averaging across multiple explanatory variables to make predictions as described in chapter 5. As mentioned this is not the ideal method for performing multiple linear regression as this does not factor in the correlation between explanatory variables and the response variable. There is room for improvement and I have already started testing multiple linear regression using gradient descent, which can be seen in the appendix.

6.1.2. Reorganise & improve the Member's Prediction League

This was one of the user feedback items discussed in the user testing section. The user wanted the Members Prediction table to be reorganised to focus on predicting the driver for a given finish position instead of the finish position for a given driver. Along with this other improvements could be introduced such as allowing users to see the prediction's made by other users.

6.1.3. Standardise and expand Search Database

Currently the driver images of the Search Database page are not a standardised size, with older driver's image varying in proportions. This currently leads to an inconsistent feel while scrolling through drivers and was mentioned by one user. This is something that will be improved upon in the future, along with attempting to expand the information for drivers and tracks. This could include displaying a drivers and tracks previous race results as this data is already stored as part of the Race Prediction.

6.1.4. Browser Control Buttons

As the website is a single page web application, the browser control buttons (back, forward, refresh) currently do not operate correctly as raised by a user. The control buttons of the browser are the standard method for navigating websites so it is important this is improved, to improve the applications usability. This problem has already been investigated and through the HTML5 History API, I plan to rectify this problem.

6.1.5. Improve Google Analytics tracking

Similar to the problem presented by the browser control buttons, Google Analytics cannot currently measure the paths a user follows through the application. In order to monitor future use of the application and understand the most utilised areas of the site this will be an area of focus in the future. Already I have looked into the problem and through utilising Google Analytics tracking code API, a website can track site events to improve its reporting ability.

6.1.6. Chat function

To improve user interaction on the website and allow users to directly communicate with each other, implementing a chat/forum system for users, was investigated for this project. Unfortunately due to time constraints and difficulty implementing the system, the chat function had to be dropped. The chat function will use the Google Channels API which allows the server to send messages to other clients that are connected without having to reload the web application. The code for this can be seen in the server implementation with a chat servlet and chat client management class already created. This will be a focus in the future as it will be a huge benefit to helping build the applications user community.

6.1.7. Administrator level user account

In chapter 4, the system architecture was outlined, mentioning the different users and how they would interact with the system. An administrator role was discussed and talked about, but unfortunately this functionality was not implemented due to time constraints. In the future and especially with the development of a chat function, an administrator role will be important to manage parts of the application. From investigating this feature, it seems Google already have an administrator parameter for user accounts so this should be easily implementable in the future.

6.2. Critical Analysis

One of the most important aspects of the web application and this project, was to predict Formula 1 race results. Unfortunately the method used to achieve this is not perfect and I believe I may not have focused enough research into how to implement multiple linear regression early enough. This section of the project can be improved upon in the future given a better understanding of multiple linear regression but can be outlined as one of the areas where preparation was deficient.

Another aspect of the project that could have been improved was the collection of the dataset. This took a considerable amount of time due to it requiring a very manual process. Possibly with better investigation a more automated process could have been created, saving time. At the moment any updates to the race prediction dataset are a manual process as well and an automated process at the start could have improved the long term running of the application.

6.3. Conclusion

From completing this project I have improved my knowledge on the development of an application. This is the largest and most technically difficult software project that I have undertaken and I have learnt a lot from the experience. Most of this is around the project planning stage and how important it is to stick to a development life cycle with clear goals at each stage. Along with this early interaction with user groups can help develop a project that is not just focused on the creative ideas of the developer but also facilitates the user.

Outside of the project planning and implementation scope my knowledge on the process of data mining and the different algorithms that support it has grown considerably. I believe I have a much better understanding of the abilities that data mining can bring to a project and the advantages it has in discovering trends and information from data.

Compared to previous projects creating websites I believe my understanding of how to design modern attractive websites for users has improved. Along with this I have learnt new methods and techniques for creating websites with powerful backend servers that can feed and analyse data to the user.

Overall I feel the project has been a great success and it is great to know users have tested and used my web application to make and generate predictions. Now that it is live and available to users I hope to see the application grow and reach a large user base which I believe is achievable with some of the positive feedback received during user testing.

7. Bibliography

- [1] Formula One Management Limited, "Formula1.com," Formula One Management Limited, [Online]. Available: <http://www.formula1.com/default.html>. [Accessed 5 12 2013].
- [2] A. C. Portillo, "CFD Analysis of Winglets," *Glyndwr University Wrexham*, 2011.
- [3] I. Horlock, "Prediction of Formula One Race Results using Driver Characteristics," *University of Bristol*, 2009.
- [4] sports-reference, "sports-reference.com," sports-reference, [Online]. Available: <http://www.sports-reference.com/>. [Accessed 4 12 2012].
- [5] AutoSport Magazine, "Forix.com," AutoSport Magazine, [Online]. Available: <http://www.forix.com/>. [Accessed 10 11 2013].
- [6] NBC Sports, "What's it cost to compete in Formula One? An IndyCar comparison," NBC Sports, [Online]. Available: <http://motorsportstalk.nbcsports.com/2013/05/22/whats-it-cost-to-compete-in-formula-one-an-indycar-comparison/>. [Accessed 12 28 2013].
- [7] A. Kerkes and S. MacLean, "The rise of data mining: opportunity or threat," [Online]. Available: <http://www.lewers.com.au/uploads/data-mining.pdf>.
- [8] SAS, "What is big data?," [Online]. Available: <http://www.sas.com/big-data/>.
- [9] G. A. F. Seber and A. J. Lee, "Linear Regression Analysis," Wiley, pp. 2-5.
- [10] J. Kelleher, "Similarity-based Learning, Extensions & Variations," in *DIT Lecutre*, Dublin,] 2014.
- [11] D. J. Leinweber, "Stupid Data Miner Tricks: Overfitting the S&P 500," 2007.]
- [12] Forbes, "Sauber F1 Analytics," Forbes, [Online]. Available:] <http://www.forbes.com/sites/netapp/2013/01/28/sauber-f1-analytics/>. [Accessed 2 10 2013].
- [13] Computer Weekly, "McLaren uses high speed data analytics to gain Formula 1 edge,"] Computer Weekly, [Online]. Available:

<http://www.computerweekly.com/news/2240184539/McLaren-uses-high-speed-data-analytics-to-gain-Formula-1-edge>. [Accessed 3 10 2013].

- [14 C. Cao, "Sports Data Mining Technology Used in Basketball Outcome Prediction," pp. 35-39, 2012.]
- [15 Statisticshowto.com, "What is the Pearson Correlation Coefficient?," Statisticshowto.com, [Online]. Available: <http://www.statisticshowto.com/what-is-the-pearson-correlation-coefficient/>. [Accessed 03 12 2013].]
- [16 I. H. Witten, E. Frank and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2011.]
- [17 IEEE, "Top 10 Algorithms in Data Mining".]
- [18 A. O. Cinneide, "Linear Prediction, The Technique, Its Solution and Application to Speech," 2008.]
- [19 D. C. Montgomery, E. A. Peck and G. G. Vining, Introduction to Linear Regression Analysis, Wiley, 2012.]
- [20 M. S. Gashler, "Linear Regression," Department of Computer Science and Computer Engineering, [Online]. Available: <http://uaf46365.ddns.uark.edu/ml/projects/project3.html>. [Accessed 02 12 2013].]
- [21 C. Shalizi, "Logistic Regression," [Online]. Available: <http://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf>. [Accessed 21 03 2014].]
- [22 V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, 1995.]
- [23 Oracle, "Support Vector Machines," Oracle, [Online]. Available: http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_svm.htm#CHDDJFDJ. [Accessed 06 12 2013].]
- [24 S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," pp. 249-268, 2007.]
- [25 A. Egan, "Usefulness Analysis of Q&A Posts," 2012.]

- [26 M. Bailey, Lies, Damn Lies and Statistics, White Box Publishing, 2004.
]
- [27 Oracle, "Regression," Oracle, September 2007. [Online]. Available:
] <http://www.comp.dit.ie/btierney/oracle11gdoc/datamine.111/b28129/regress.htm>.
[Accessed 25 2 2014].
- [28 Acunetix, "Web Applications: What are They? What of Them?," Acunetix, [Online].
] Available: <http://www.acunetix.com/websitesecurity/web-applications/>. [Accessed 05
04 2014].
- [29 Website ToolBox, "Promoting your forum," Website Toolbox, [Online]. Available:
] <http://www.websitetoolbox.com/support/290>. [Accessed 29 02 2014].
- [30 Digital Communications Division in the U.S. Department of Health and Human Services'
] (HHS) Office of the Assistant Secretary for Public Affairs, "Usability Guidelines," Digital
Communications Division in the U.S. Department of Health and Human Services' (HHS)
Office of the Assistant Secretary for Public Affairs, 2006. [Online]. Available:
<http://guidelines.usability.gov/>. [Accessed 29 02 2014].
- [31 F1Predictions.net, "Make F1 predictions to win prizes - Better than Fantasy F1,"
] F1Predictions.net, [Online]. Available: <http://www.f1predictions.net/>. [Accessed 27 02
2014].
- [32 P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer and R. Wirth,
] "CRISP-DM 1.0," pp. 10-29, 2000.
- [33 KDNuggets, "Data Mining Methodology Poll," KDNugget, August 2007. [Online].
] Available: http://www.kdnuggets.com/polls/2007/data_mining_methodology.htm.
[Accessed 02 December 2013].
- [34 Zentut, "Data Mining Processes," Zentut, [Online]. Available:
] <http://www.zentut.com/data-mining/data-mining-processes/>. [Accessed 6 12 2013].
- [35 Scrummethodology.com, "Scrum Methodology," CollabNet, [Online]. Available:
] <http://scrummethodology.com/>. [Accessed 07 12 2013].
- [36 M. James, "Scrum Reference Card," Scrum Reference Card, [Online]. Available:
] <http://scrumreferencecard.com/scrum-reference-card/>. [Accessed 07 12 2013].
- [37 University of Waikato, "Downloading and installing Weka," University of Waikato,
] [Online]. Available: <http://www.cs.waikato.ac.nz/ml/weka/downloading.html>.
[Accessed 28 10 2013].






- [38] RapidMiner, "Installing RapidMiner (any platform) Step 1: Installing Java," RapidMiner, [Online]. Available: <http://rapidminer.com/support/installation-guide/#installing-java>. [Accessed 28 10 2013].
- [39] Google, "Google App Engine Platform as a Service," Google, [Online]. Available: <https://developers.google.com/appengine/>. [Accessed 07 12 2013].
- [40] Amazon, "Amazon Web Services," Amazon, [Online]. Available: <http://aws.amazon.com/>. [Accessed 07 12 2013].
- [41] KDNuggets, "What Analytics, Big Data, Data mining, Data Science software you used in the past 12 months for a real project?," KDNuggets, May 2013. [Online]. Available: <http://www.kdnuggets.com/polls/2013/analytics-big-data-mining-data-science-software.html>. [Accessed 25 03 2014].
- [42] M. Allender, "The Role of Driver Experience in Predicting the Outcome of NASCAR Races: An Empirical Analysis," *The Sports Journal*, vol. 12, no. 2, 2009.
- [43] M. Allender, "Are There a Higher than Expected Number of Early Life Critical Part Failures in NASCAR Vehicles? A Reliability Study," *The Sport Journal*, vol. 10, no. 1, 2007.
- [44] P. V. Allmen, "IS the reward system in NASCAR efficient?," *Journal of Sports Economics*, vol. 2, no. 1, pp. 62-79, 2001.
- [45] C. B. Pfitzner and T. D. Rishel, "Do Reliable Predictors Exist for the Outcomes of NASCAR Races?," *The Sport Journal*, vol. 8, no. 2, 2005.
- [46] W3, "HTML5 A vocabulary and associated APIs for HTML and XHTML," W3, 2014 February 2014. [Online]. Available: <http://www.w3.org/TR/html5/browsers.html#history>. [Accessed 19 March 2014].

8. Appendix


8.1. Google Analytics Browser Usage

1. Chrome	118		78.15%
2. Firefox	17		11.26%
3. Android Browser	9		5.96%
4. Safari	4		2.65%
5. Internet Explorer	3		1.99%

8.2. Google Analytics Operating System Usage

Operating System	Visits	% Visits
1. Windows	104	 68.87%
2. Android	41	 27.15%
3. iOS	3	 1.99%
4. Macintosh	2	 1.32%
5. Linux	1	 0.66%

8.3. Google Analytics Mobile Device Screen Sizes

Screen Resolution	Visits	% Visits
1. 360x592	27	 61.36%
2. 480x800	5	 11.36%
3. 320x480	2	 4.55%
4. 598x360	2	 4.55%
5. 600x912	2	 4.55%
6. 800x1220	2	 4.55%
7. 320x568	1	 2.27%
8. 360x640	1	 2.27%
9. 540x960	1	 2.27%
10. 800x1125	1	 2.27%

