



**DUBLIN INSTITUTE
of TECHNOLOGY**
Institiúid Teicneolaíochta Bhaile Átha Cliath

CrimAnalytics

A Crime Prediction Web Application

Final Year Project Report

DT228
BSc in Computer Science

Seán Jennings
Patricia O'Byrne

School of Computing
Dublin Institute of Technology

4th April 2017

Abstract

The objective of this project is to develop a web application that can predict the crime rates in an area. This project can be separated into two main areas.

The first area is the creation of the predictive model. This model will provide the ability to predict crime rates in an area. Research was performed to identify the best suited predictive algorithm. Known data on crime and the factors that influence it will also have to be collected.

The second area is to build the CrimAnalytics web application. The web application will allow users to access the predictive power of the model. This process will include the design, implementation and testing of the web application.

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 cursive script, reading "Seán Jennings", is written over a horizontal line.

Seán Jennings

4th April 2017

Acknowledgements

I would like to thank everybody who helped me throughout this project, providing support and feedback. I would like to especially thank my supervisor Patricia O'Byrne who offered invaluable guidance and advice throughout the year.

I would also like to thank my parents Kevin and Anita for supporting me throughout this endeavour.

Table of Contents

1. INTRODUCTION	1
1.1. OVERVIEW OF THE PROJECT AND THE BACKGROUND BEHIND IT.	1
1.2. PROJECT OBJECTIVES.....	2
1.3. PROJECT CHALLENGES	3
1.3.1. <i>Data and Feature Preparation</i>	3
1.3.2. <i>Researching Prediction Techniques and R Implementation</i>	3
1.3.3. <i>Selection of Project Framework and Setup</i>	4
1.3.4. <i>Separation of Concerns in Codebase using Layers</i>	4
1.4. STRUCTURE OF THE DOCUMENT	4
2. RESEARCH	5
2.1. PROBLEM AND SOLUTION IDENTIFICATION RESEARCH	5
2.1.1. <i>Alternative Existing Solutions</i>	5
2.1.2. <i>Datasets & Correlation</i>	6
2.1.3. <i>Predictive Algorithms</i>	6
2.1.4. <i>Training, Testing & Validation</i>	10
2.2. TECHNOLOGY DECISIONS	10
2.2.1. <i>Java Servlets & JSP</i>	10
2.2.2. <i>HTML, CSS and JavaScript</i>	11
2.2.3. <i>R and the Java/R Interface</i>	12
2.2.4. <i>Maps & Places APIs</i>	13
2.3. RESULTANT FINDINGS AND REQUIREMENTS	13
3. DESIGN	14
3.1. DESIGN METHODOLOGY CHOICE	14
3.2. CRISP-DM	15
3.3. AGILE	17
3.4. DESIGN OF PROJECT COMPONENTS	18
3.4.1. <i>User Interface Design</i>	18
3.4.2. <i>Source Code Layout</i>	21
3.5. PROJECT FEATURES AND USE CASES.....	23
4. ARCHITECTURE & DEVELOPMENT	23
4.1. SYSTEM ARCHITECTURE	24
4.1.1. <i>Architecture Overview</i>	24
4.1.2. <i>Backend Services using Docker Containers</i>	25
4.2. DEVELOPMENT COMPONENTS AND CHALLENGES	26
4.2.1. <i>Java Services</i>	26
4.2.2. <i>User Interface Implementation</i>	26
4.2.3. <i>Model Training in R</i>	26
4.3. EXTERNAL APIS AND PERSONAL CODE	27
4.3.1. <i>External APIs</i>	27
4.3.2. <i>Personal Back-End Code</i>	30
4.3.3. <i>Personal Front-End Code</i>	38
5. SYSTEM VALIDATION	40
5.1. TESTING	40
5.1.1. <i>Model Testing and Evaluation</i>	40
5.1.2. <i>Java Unit Testing</i>	43
5.1.3. <i>Java Integration Testing</i>	44

DIT
DT228 B.Sc. (Hons.) in Computer Science
Final Report 2016/17

5.1.4.	<i>User Testing and Web Application Evaluation</i>	<i>44</i>
5.2.	IMPLEMENTED UNIT TESTS.....	48
5.3.	DEMONSTRATION	49
5.4.	DEMONSTRABLE FEATURES	50
6.	PROJECT PLAN	52
6.1.	PROJECT PLAN ANALYSIS.....	52
6.2.	REVIEW OF CHANGES	53
7.	CONCLUSION	53
7.1.	ANALYSIS OF PROJECT ELEMENTS AND LEARNING OUTCOMES.....	53
8.	BIBLIOGRAPHY	55

1. Introduction

1.1. Overview of the project and the background behind it.

In recent years, the use of predictive models has become more prevalent due to the many benefits that they can provide. Predictive Analytics and Modelling has proven to be useful across many industries such as Marketing, Finance, Healthcare, Manufacturing and the Public Sector. This project will consider the use of Predictive Analytics and Modelling in Crime Prediction. There are few examples of production software on the market that apply Predictive Analytics to Crime Prediction. The few existing solutions are still being rolled out and are still in their proof of concept phase.

There are several benefits of a Crime Prediction model to both citizens and local government. A model for Crime Prediction can help both groups to gain a deeper understanding of the causes of crime. The crime model will use recent crime data during its training. This data is made up of counts of different categories of crime (burglary, theft, fraud, damage to property etc.) in every Garda Station in Ireland. This data was combined with locational data (educational, recreational, transport etc.) that can explain the different categorical crime rates in an area.

Both citizens and local government will be able to access this model through the CrimAnalytics web application. Both user groups will be able to use the CrimAnalytics web application as a tool to view crime rates in any area. Although the model will be trained with data from Ireland, it will be of use to citizens or governments of any country to use on areas near them. This will enable citizens to make informed decisions, such as the purchasing of real estate or office space. The application will also act as a useful tool for local government and county councils. Using CrimAnalytics, local government will be able to see what changes can be made to lower crime rates. The web application will show how changes in the numbers of different local features affect the different categories of crime. As with any predictive model, there are explanatory variables for the target that is being modelled. A predictive model must be trained before it can be of any use. In most cases, a predictive model is trained using data that contains known past outcomes of the target that is being modelled (Scriven, 2016).

To build a predictive model successfully, this crime data must be combined with locational data that can explain the known crime rates. A predictive algorithm will go over the combined crime and locational data and produce a model that can predict crime rates. The finished model should be able to take a set of explanatory locational data points from a location that a user specifies. The model should then produce a set of values of crime rates around the location. The CrimAnalytics web application will be built to enable a user to see this process and its results. This will be achieved with graphical user interfaces such as maps for locations and graphs for the resultant crime rates.

1.2. Project Objectives

Several objectives were set to enable this project to achieve its aim. The initial objectives relate to the gathering of data. The two objectives here are the identification of sources of historical crime rate information and data that can explain the crime rate values. After those two objectives are completed, the next will be to combine and prepare the data so that it can be used to build a predictive model. The building of the model itself using a predictive algorithm is also an objective. The final objective will be to build the CrimAnalytics web application to allow users to access the predictive abilities of the model. Users will be able to select an area on a map and the application will show them the crime rates in that area. This objective can be split into the creation of the back and the front end of the CrimAnalytics web application. The backend includes the implementation of the logic required to carry out the use cases for training and applying the model. The frontend includes the implementation of tools for users to choose geographical locations and view the crime rates there. The tools to achieve this will be maps and charts.

The first objective is to gather the recent crime data. This data will contain known crime rates for given areas and will be used to train the model i.e. what crime rate values to expect when combined with the explanatory data. In the case of this project, the recent crime data is a data set provided by the Irish Government of recorded crime offenses, grouped by each individual Garda Station (Scriven, 2016). This dataset provides the number of different offences recorded by every Garda Station in Ireland. The types of crimes in this data set will act as the dependent variables.

The next objective will be to find a data source that can explain the counts for the crime categories. In the case of this project, locational data will be used to explain the crime rate values in an area. This locational data will be counts of how many types of features are in the area. These features will be grouped into categories that will act as data points i.e. Education, Food, Government, Health, Recreation, Stores, Services, Transport and Worship. These will act as the explanatory variables for the predictive model. Once this data has been combined it will require preparation and verification before it is suitable to use for the training of the predictive model.

The next objective will be to combine both data sources. This will ensure the removal of any inconsistencies in the data. If left as is, these inconsistencies could result in some pieces of data being left out when training the model or could make the model less accurate if left malformed. In the case of this project, this means ensuring that the recent crime data and the explanatory data are describing the same place and time. Doing so will ensure that the model is properly trained and the result is as accurate as possible. The more accurately the model is trained, the more the results of applying the model can be trusted. This is vitally important, as the

model will be expected to operate on areas that it might not have been trained on and still provide accurate predictions.

The next objective is to build the predictive model. The first step to achieving this objective will be to research and choose the predictive algorithm that will create the model. The algorithm must produce an effective model that can produce crime values when given a set of explanatory variables. A language and engine will also need to be researched and chosen that will enable the training of the predictive model. These objectives detailed so far match up with the general steps in the CRISP-DM methodology, which is covered in later chapters. This methodology helps create an accurate data model using an iterative process.

The last objective is to build the web application, “CrimAnalytics” that will allow users to access and use the predictive model. The first part of this objective is to build the backend that will contain the logic required to satisfy the use cases of the application. The next part of this objective is the implementation of a frontend that will allow users to access the predicted crime rates. They will be able to choose any area on a map and receive a report of the crime rates there from the predictive model. Users will be able to view the resultant crime rates and see how changes in the number of locational features affects those crime rates.

1.3. Project Challenges

This section details the key challenges that were overcome to complete this project. Each challenge will be covered in relation to the project objectives from the previous section. The exact details of how each challenge was approached is covered in later sections.

1.3.1. Data and Feature Preparation

One of the project challenges was the data and feature preparation. This ensured that the data was accurate when being used in training a predictive model. The preparation of the data and features included several steps. The first step was to correct some of the place names in the government crime data set. If left as is the incorrect place names would cause later processes to fail. Another challenge was the addition of coordinates to the original data set. This was to ensure the correct collection of locational data for a given area.

1.3.2. Researching Prediction Techniques and R Implementation

Another challenge of this project was researching prediction techniques and implementing them in the R language. The purpose of the research into prediction techniques was to identify one that was suited to the specific problem that the project aimed to solve. The aim of the research was to fully understand the mechanics of several prediction algorithms and decide on the one most suited to the problem of crime prediction. Another challenge that arose after I chose a prediction algorithm was the implementation of it in the R language. The challenges here were learning

the R language, specifically its various data types and methods of data transformation. The processes I had to implement in R were to load my data, train a model with that data, and apply the model to new data. Finally, the R script returned the prediction values as its result.

1.3.3. Selection of Project Framework and Setup

The research and selection of potential frameworks to use in solving the problem of crime prediction was also a challenge. This research was performed with the aim of choosing a framework that would enable the implementation of a solution to the problem of crime prediction. The framework I chose had to include languages for the front and back end that could communicate easily, a database to store the training data, a prediction engine to build a predictive model and available drivers to connect these parts. This research also included the identification of an Integrated Development Environment that would allow for front and back end development.

1.3.4. Separation of Concerns in Codebase using Layers

Another aspect of this project that posed a challenge was ensuring a separation of concerns in the code base. The purpose of achieving a separation of concerns is to give the project a clean structure that is easy to read and make changes to. The implementation of the processes that this project is made up of have been separated into tiers of functionality. Each tier is encapsulated in its own package in Java. This work has allowed me to approach each problem in a uniform way and has made implementing software changes and tests much faster.

1.4. Structure of the document

The structure of this document will match the progression of the project itself, from research and design to architecture and implementation. This first section has given an overview of this project, its objectives and key challenges.

The research section gives details of research done into the area of Predictive Analytics with Crime Data. There will also be research and comparison of predictive technologies and algorithms that were key to the completion of this project.

The design section details the design methodologies utilised during this project. It contains an in-depth consideration of the steps laid out in each methodology and the reasons behind each methodology's effectiveness. It also provides details on how exactly each methodology is utilised in relation to this project specifically.

The architecture and development section contains details of the implementation of the project. This will include information of how the researched technologies were implemented. This section details all the components that make up the project and how they communicate with each other.

The system validation section contains details of the testing performed and the demonstrable features of the project. The testing will cover both in-application tests as well as real world user testing. The demonstrable features will showcase the solutions created to satisfy the web application's use cases.

2. Research

This section will discuss the background research performed for this project. The research in this report is on Predictive Analytics and its uses in crime detection. This include research in key areas of this project such as Predictive Analytics, Data Mining, Regression Algorithms and Web Application Stack Technologies. Several of these sections include comparisons of potentially useful technologies and predictive algorithms. The resultant findings and requirements are also detailed at the end of this chapter.

2.1. Problem and Solution Identification Research

This research will look at the problem of predicting crime rates. The aim of this section is to evaluate existing solutions and techniques in crime prediction. The evaluation of these techniques will aid in the identification of a solution within the scope of this project.

2.1.1. Alternative Existing Solutions

Several product and services exist in Crime Analytics. Most of these products and services focus on real time analytics in a single city. They are designed as assistive tools for local law enforcement. The products main features are heat maps of a city showing where crime is most likely to occur. This is to enable law enforcement to assign their people in areas where they will be most effective at preventing crime. One such product, PredPol is currently being tested in both California and Georgia in the US. ('PredPol', 2015)

Several companies who are already invested in Predictive Analytics are also considering its applications in Crime Prevention. One such company is IBM. They initially released a paper documenting the potential of their existing IBM SPSS (Statistical Package for the Social Sciences) product to assist in crime prevention (IBM Corporation, 2010). This was then developed into a working prototype in the City of Lancaster, California. This involved the building of a predictive model which when utilised, resulted in a thirty-five per cent reduction in crime in the city. Nucleus, a research company documented its success in a report and subsequent awarding of a prize for the highest Return on Investment research project (Nucleus Research Inc., 2012).

2.1.2. Datasets & Correlation

One of the lengthiest processes in a data mining and predictive analytics project is the creation of the data set. The final data set may have been the result of importation of data from multiple sources. In this project both crime and locational data was combined to produce the training data for the predictive model. It could be scraped from web pages, pulled from various APIs or it may already be a fully formed and available to a data analyst. Thus, this process of data collection can take time. Even after the collection of the relevant data has been completed, it may still need further work. This can include processes such as cleansing of errors in the data set and the transforming of data into types compatible with both each other and the target database.

When the data has been successfully collected, the next step is to gain an understanding of the data that has been collected. This step includes the examining of the column names and types in the dataset. Understanding the reasons behind the choices made to store data in a specific format is necessary to make accurate and informed predictions going forward. Doing so ensures that variables that may cause the model to be less accurate are not included in the training process.

The aim of the data mining process in this project is to collect and combine data that is relevant to the crime rates in an area. The data understanding was gained as the candidate sources for the training data were chosen. The data preparation will ensure that the data is uniform as without anomalies. The objective of the data mining process is to create suitable training data. This training of the model will enable it to predict the amounts of different types of crime in an area. Thus, the data mining process is a key step to ensuring the quality one of the key features in this project.

2.1.3. Predictive Algorithms

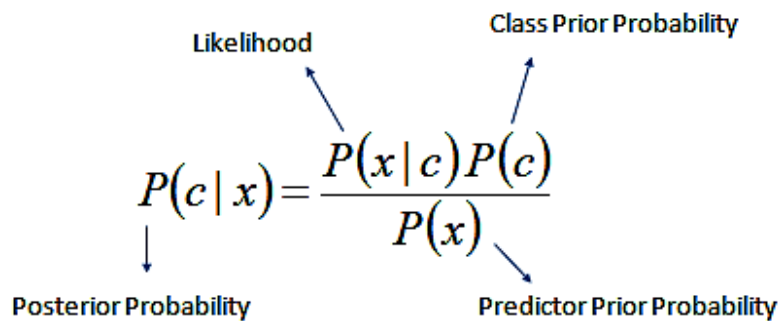
This section will compare the processes and resultant models that can be produced by several predictive algorithms. The aim of this research is to identify the algorithm best suited to solving the problem of creating a predictive model for crime. Each algorithm's method for training models has been analysed and judged on its suitability for the previously collected explanatory and dependent variables.

2.1.3.1. *Naïve Bayes Classifier*

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation, which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well. Thus, is widely used because it often outperforms more sophisticated classification methods.

Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. The Naïve Bayes classifier assumes that the effect of the

value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$


$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Figure 1: Naive Bayes Algorithm (Dr. Saed Sayad, 2010)

$P(c|x)$ is the posterior probability of class (target) given predictor (attribute).

$P(c)$ is the prior probability of class.

$P(x|c)$ is the likelihood that is the probability of predictor given class.

$P(x)$ is the prior probability of predictor.

(Byrski, Oplatková, Carvalho, & Kisiel-Dorohinicki, 2012)

2.1.3.2. Linear Regression

Linear regression describes, and tests for evidence of a linear relationship of given slope between continuous variables. The overall approach was further developed by (Pearson, 1905) and (Fischer, 1922).

There are several steps to Linear Regression. First, it estimates parameters describing the relationship. As this is linear regression all that needs to be estimated is the slope angle and one of the axis intercepts. Secondly, it conducts hypothesis tests based on the overall variability in the data. It also tests how the response variable y varies with the predictive variable x. This is done over many iterations of comparisons between the two variables. It asks is the null hypothesis can be rejected. The null hypothesis states that the true gradient, relating y to x is zero.

Linear Regression works under the assumption that the world is rather linear in that there is a population-level intercept and a population-level slope. It is important to note that error, although it is called error, it is really an inadequacy of the model in explaining all the variability in the data, rather than an inadequacy of the data or even the experimenter.

In the real world, full populations are rarely dealt with. More often, a sample of a population to date can be taken. If there is a relationship between y and x and the aim was to infer things about the population from which that sample was taken.

A degree of uncertainty can be put into the estimates of y and x based on the strength of the relationship and the variability in the data themselves.

Regression assumes that the predictor is measured exactly so the line of best fit can be estimated simply by minimising the vertical deviations (squared) of the line to the data. Even if the predictor is not measured or controlled exactly, it is still possible to conduct a Linear Regression so long as proper considerations have been made about correlation.

The first thing that Linear Regression creates is an overarching analysis of variance. This is done by getting an index of deviation of response data from the grand mean. This is calculated as the total sum of squared deviations of data points around the grand mean. Another measure of the variability of data is to get an index of deviation of response data from predictions. Using the line of best fit it is possible to work out the sum of the squared residuals. The final measure that can be done involves the deviation of the line of best fit from the mean. For each data point, there is the sum of squares of the fitted model to the overall grand mean. These squared deviations are additive. The total variability in the data as expressed as sums of squares is equal to the variability explained by the regression and the variability not explained by regression.

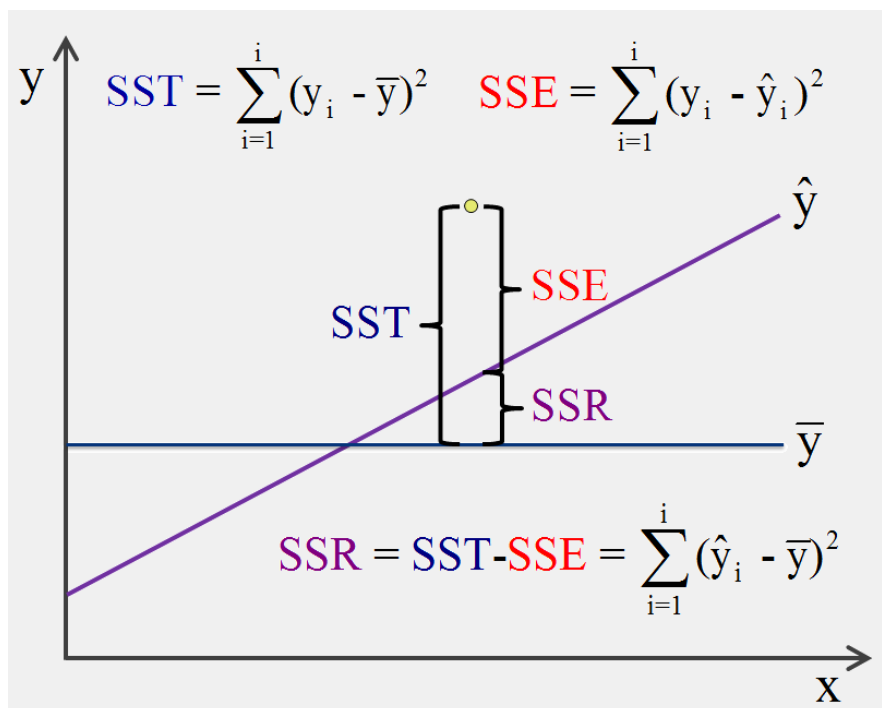


Figure 2: Diagram showing the additive nature of the three types of variance. (Shah, 2015)

The ways in which each of the sums of squares are calculated can be seen in Figure 2.

SSR (Explained Variability, Regression Sum of Squares)
=
SST (Total Sample Variability, Total Sum of Squares)
-
SSE (Unexplained Variability, Error Sum of Squares)

These sums of squares values allow for the calculation of the R^2 statistic, also known as the coefficient of determination. It is the fraction of variability (as indicated by the sum of squares) explained by the line that has been fitted.

$$\text{Coefficient of Determination} \rightarrow R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

$$\text{Sum of Squares Total} \rightarrow SST = \sum (y - \bar{y})^2$$

$$\text{Sum of Squares Regression} \rightarrow SSR = \sum (y' - \bar{y}')^2$$

$$\text{Sum of Squares Error} \rightarrow SSE = \sum (y - y')^2$$

Figure 3: Coefficient of Determination (Sayad, 2010)

The R^2 statistic is a score from zero to one on the accuracy of the model. It is a measure of the overall variance in the predicted and actual values. As it describes the ability of the model to predict for the dependent variable, it is a key output of Linear Regression.

2.1.3.3. Logistic Regression

In some cases, lines of best fit are not useful in creating predictive models. Depending on the layout of the scatter plot, this line may be inaccurate to the point where it serves no use as a predictor for the dependent variable. Logistic regression provides a technique to create a predictive model in such situations. Logistic regression allows for modelling of the probability of an event occurring based on the values of the independent variables, which can be either categorical or numerical. It goes on to estimate the likelihood that an event will occur for a random observation versus the likelihood that it will not. This allows for the prediction of an array of variables on a binary response variable.

The model that this algorithm creates allows for the classification of observations by estimating the probability that an observation is part of a specific category. This decision model can then be applied to new data. Based on the values that it has categorised from the scatter plot, it can now predict for new events. This type of regression algorithm is best suited for predictions on dichotomous variables.

It allows for the input of quantities data to predict the probability of a binary event. (Cox, 1958)

2.1.3.4. Comparison and Decision

To solve the problem of crime prediction within the scope of this project, the Multiple Linear Regression algorithm will be used. This algorithm has proven to best match the solution of combining multiple crime categories with multiple locational categories to create a predictive model. This algorithm facilitates the creation and combination of models for each crime category. Thus, the model trained by this algorithm will be able to take counts of locational features and return the counts for the crime categories. This algorithm also makes it easy to add new explanatory variables and decrease the variance in the predictions from the model.

2.1.4. Training, Testing & Validation

When predictive algorithms are being used, the data is divided into two distinct parts, the training data and the testing data. The training data is the data that has known dependent variables. Using these known values, predictive algorithms can look at the array of independent variables that affect them and build a model to predict them. Using the same independent variables as before a trained model should be able to predict for unknown dependent variables. The more rows of data that are provided during the training phase, the more accurate the model will be.

2.2. Technology Decisions

An overview of the technologies evaluated and selected or rejected and the rationale behind the key decisions.

2.2.1. Java Servlets & JSP

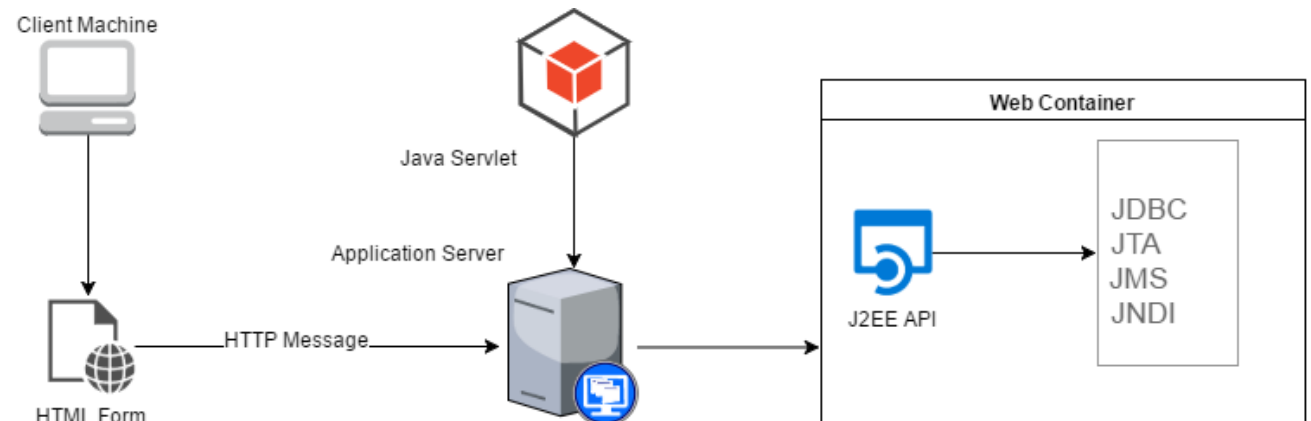


Figure 4: Diagram showing the flow of a user request to a Java Servlet. ('draw.io', 2016)

Figure 4 shows the architecture involved in the communication between Java Server Pages on the front end and Java Servlets on the back end.

The first step in this process is a Servlet being accessed by the client machine. The client uses a HTML Form, which is a special type of HTML document. It contains

special parameter information that can be passed to the servlet. When the HTML Form is utilised, a HTTP message is created. This HTTP message is formatted in two portions. The header of this HTTP message identifies the servlet that will receive this message. The body of this message contains all the information that the client is sending to the servlet.

The next step is the message first reaching the Application Server. From here, control is passed to a web container. The servlet program that the client is talking to is contained and deployed inside this web container. The role of this web container is to support all the activities of a Java servlet program. It does this by providing access to a set of programming interfaces known as Java 2 Platform, Enterprise Edition (J2EE). This set of J2EE services allow the server to perform a wide variety of tasks i.e. access databases, access message queues, and work with session information. With the application server deployed, the servlet can now be open to the client's requests. The main J2EE service that has been used as part of this project is the Java Database Connectivity (JDBC) API. This API provides access to relational types of structures. It provided a set of classes that allow for the mimicking of tables, rows, statements, results and supported types (Oracle, 2013).

In conclusion, I believe Java's HTTP request and response handling functionality will be of great use in this project. It provides a specific layer for handing request from and sending responses to the client. This functionality helps to address the challenge of maintaining a separation of concerns in the codebase.

Java Servlets and JSPs will enable the front and back end of the web application to communicate easily and addresses the challenge of separating concerns in the code base. Java Server Pages allow for quick and simple contacting of the server side of the application. The Java Servlets on the server side allow for simple delegation of user commands to specific services. The combination of these two technologies provides a simpler code base that is easier to read and easier to add functionality.

2.2.2. HTML, CSS and JavaScript

Hypertext is text, which contains links to other texts. In this context markup means to annotate the content in a structured manner. In HTML, this is done with a series of tags. These tags are used to convey a certain typesetting instruction (Mozilla, 2017c).

CSS is a Style Language that is applied to any Markup Language, such as HTML. The purpose of CSS is to control the appearance of a web page. It allows users to create style declarations and apply them to HTML elements or portions of HTML elements. These style declarations control the form and appearance of the HTML elements. The style declarations are made by creating value pairs of different styling properties. These declarations of properties can then be referenced when rendering web pages (Mozilla, 2017a).

JavaScript acts as the programming language of HTML and the web. It is a client side scripting language that exists and executes on the client's browser only. Its main use is the control of rich web features in the application. It allows for more in depth and reactive programming based on user interface interaction. It can also be used to display information more complexly. Many libraries assist in the creation of visualisations of data (Mozilla, 2017b).

Library	Clickable Points	Animation Capability	Documentation
Chartist.js	N	**	**
Chart.js	Y	*	**
HighCharts	Y	***	***

Table 1: Feature comparison of JavaScript visualisation libraries.

In Table 1, there is a comparison of the features available in several of the visualisation libraries available for JavaScript. Both Chartist.js and Chart.js whilst being very popular and widely used libraries, clearly have differing approaches and perform better in different areas. Considering this, I have decided to use HighCharts as it provides charts that have points that give a summary of their data, the ability to have animated elements and very good documentation for their use. It is also completely free to use in non-commercial projects such as this one. HighCharts has great documentation including practical examples that developers can edit and re-execute in real time ('Chartist.js', 2014, 'Chart.js', 2013, 'Highcharts', 2011).

In conclusion, the combination of HTML, CSS, JavaScript and the HighCharts library provide all the features required to build the front-end of this project. In combination, they will create the user interface of the CrimAnalytics web application. The goal when using these technologies will be the display of the resulting predicted crime rates in each area. The combination of these technologies addresses the challenge of ensuring a separation of concerns in the front-end of the application.

2.2.3. R and the Java/R Interface

The open source programming language known as R has emerged as the de facto standard for computational statistics and predictive analytics. The two creators of the language, Robert Gentleman and Ross Ihaka from New Zealand wanted a better statistical platform for their students. The R language was the result of their work. It is modelled after the statistical language S.

The pair continued to work on the language along with many others creating new tools for R. In this work, they found that there was great demand for these tools and found many new applications for the language very quickly. Today, R is used in a wide variety of areas including mapping broad social and marketing trends online and developing financial and climate models. There are now thousands of user-

created libraries available. These libraries all enhance the functionality of R in a unique way. It has gone on to receive widespread crowd sourced quality validation from leaders in the software industry, allowing for further improvements. It allows experts to quickly interpret, interact with and visualise data. ('R: The R Project for Statistical Computing', 2016)

A library called JRI (Java/R Interface) exists for Java and enables R to run inside of Java applications. This can be used to access the predictive capabilities of R from this Java based web application. It provides an extensive set of tools to do this. These include the handling of temporary data and access to external R scripts ('JRI - Java/R Interface', 2007).

Both the R language and the engine it provides to execute R code will address the challenge of training the predictive model. It is the most popular language for performing predictive analytics. As it is open source, there are implementations of a huge variety of both regression and classification algorithms available for developers to use. Using the Java to R interface, the challenge of integrating with the web application's technology stack is addressed.

2.2.4. Maps & Places APIs

The Google Places API gives access to Google's database of over eighty million places. This database contains detailed information about every place. The API is accessible via a web service in either JSON or HTML or via libraries in both JavaScript and Java. To access any Google API, a key is required. These keys come with usage limits on requests per second and per day. These keys can be acquired from the Google's developer console site. The first step on the developer console is to create a project. This project acts as a container for the APIs that the project uses and any other Google service. The key is used as an argument that is passed with every request the project makes. This is done to track the user's usage levels and ensure that they do not exceed their given limit.

This API will provide the explanatory locational variables needed to train the model. The list on the left of Figure 17 shows the list of these explanatory variables, with an example of the sub-categories that they represent seen on the right. In order to train the predictive crime model, the number of features in each area will be collected and added to the recent crime data (Scriven, 2016).

2.3. Resultant Findings and Requirements

This chapter will discuss my findings into this research in the areas of Predictive Analytics, Data Mining and Crime Tracking and Prediction. This research also includes information on the technologies and approaches required to complete a project in these areas.

Through my research into Predictive Analytics, I have discovered that it is possible to build a predictive model based on the government crime data set. This

can be done by collecting feature information of the surrounding area to act as the independent variables. The types of crime provided in the government data set will act as the dependent variables for the models. Depending on the support in R and the Java/R interface to perform Multivariate Linear Regression, there may be a single model for predicting all crime types or a single model for each type. Once the model or models have been trained with all the known values from the data set, they can be validated against new data.

As part of this research, I have also considered existing solutions in crime prediction. I discovered that many of these solutions approach the area from a different aspect, as real time, assistive tools for law enforcement agencies. These solutions aim to provide these law enforcement agencies with predictions of the areas in which crime is likely to occur. This project aims to create more widely applicable models. The models generated by this project could theoretically be used in any city or town as a planning tool to reduce crime rates.

I have also researched the technologies that I will need to utilise to complete this project. These include technologies for the front and back end of the CrimAnalytics web application as well as those that will enable its predictive functionality.

3. Design

3.1. Design Methodology Choice

There are two main processes involved in the development of this project, the creation of the predictive model and the development of the CrimAnalytics web application to access the model. As such, two methodologies have been chosen.

The methodology for the creation of the predictive model is the Cross Industry Standard Process for Data Mining, commonly known by its acronym CRISP-DM. It provides an extensive, cyclical list of steps to be carried out when performing data mining and producing a model.

For the development of the web application, the Agile Methodology has been chosen. It provides the ability to plan all for all items that need to be completed to release a complete piece of software. The use of this methodology will allow for the planning of all the work that needs to be completed before any implementation begins. This has the benefits of setting goals early in the project and being realistic about what is achievable in the time available.

The following sub-sections will detail the mechanics of both methodologies. The ways in which these methodologies are being utilised in this project will also be discussed.

3.2. CRISP-DM

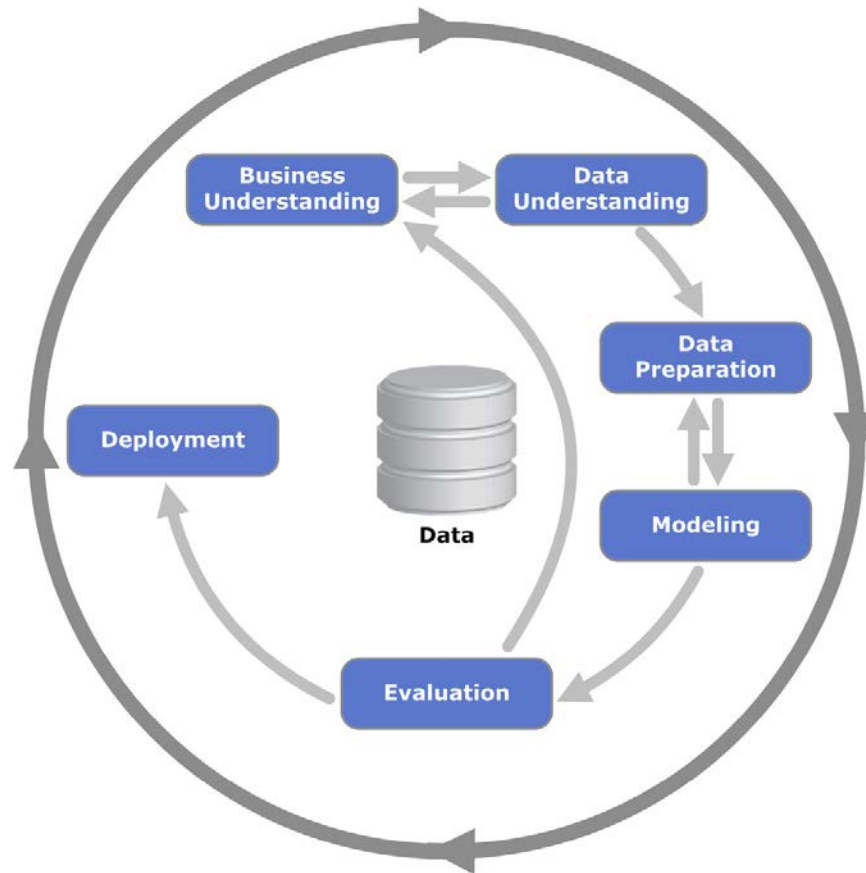


Figure 5: Diagram showing the steps of the CRISP-DM process. (Jensen, 2012)

While there are many other data mining processes, including SAS Institute's SEMMA model which stands for Sample Explore Modify Model and Assess, I will focus on using the CRISP methodology. The CRISP-DM model is a very popular methodology that provides a structured approach to planning a data mining project. CRISP-DM stands for Cross Industry Process for Data Mining. The six phases of the CRISP Data Mining process are business understanding, data understanding, data preparation, modelling, evaluation and deployment.

The first step in the CRISP-DM process is business understanding. In this stage the aim is understand the problem to be solved. It is an iterative process of discovery alongside the data understanding phase as seen in Figure 5. The objective of this stage is to figure out the project goals from a business perspective. Creativity often plays a massive role in the business understanding stage. The specific steps in the business understanding stage are to determine the business objectives, assess the situation, determine the data mining goals and produce a project plan.

The second phase is the data understanding stage. The prerequisite for this stage is an understanding of the strengths and limitations of the data. This is because the data rarely exactly matches the problem that is being looked at. It's also important to recognize that data costs money and that some data may not be

available. So, the data scientist needs to evaluate the costs and benefits of all the different potential data sources. What one may find is that the initial solution paths may already start to diverge in the second stage as the data comes up to surface. In this stage the specific steps are number one, collect initial data, number two, describe the data number three explore the data and number four verify the quality of the data.

In the third stage of the CRISP model there is data preparation. Analytics technologies often require the data be in a form that's different from how the data was initially provided and so conversion may be necessary. Some common data preparation examples include converting data to tabular formats, removing or inferring missing values, converting data to different types and scaling numerical values so they are comparable. In this stage, the specific steps are to selected data clean the data, construct the data and integrate the data.

In the fourth stage of the CRISP-DM model the concept of modelling is first encountered. This is the primary stage where the data mining algorithms are applied to the data with the output hopefully being some sort of model a pattern. It's really important to note that not all patterns are valid. As data comes to the light, new models are built and this might involve going back to the preparation stage if the models just are not very telling. In this stage the specific steps are to select the modelling technique, generate tests for model robustness, build the model and assess the model.

The fifth stage of the CRISP-DM model is the evaluation stage. The purpose of this stage is to assess the data mining results, both from a quantitative and qualitative perspective and determine if the results are both justifiable and feasible. Not all patterns are valid because patterns can be found in any data. It's encouraged to do this before the deployment stage because it usually ends up being cheaper and safer than simply skipping the stage and going directly from modelling to deployment. One of the questions often asked during this evaluation stage is does the model satisfy the original business goals. This is iterative with business understanding which means that the process may go back to the business understanding stage if the model is not meaningful. It's also important to note that just because the model passes the predetermined tests doesn't mean that it's going to work in the real world from a practical standpoint. In this stage the specific steps are to evaluate the results, review the process and determine the next steps. In the very last stage of the CRISP-DM model, there is have the deployment stage. This is where the results of the data mining are put into use so that the user can realize the benefits from the project. (Chapman et al., 2000)

All of the stages set out in this methodology will be beneficial to the successful training and deployment of a predictive crime model. The iterative cycle that it proposes allows for the improvement of the model during the course of this project. As the model is a key component in this project, this use of this methodology will be of great benefit.

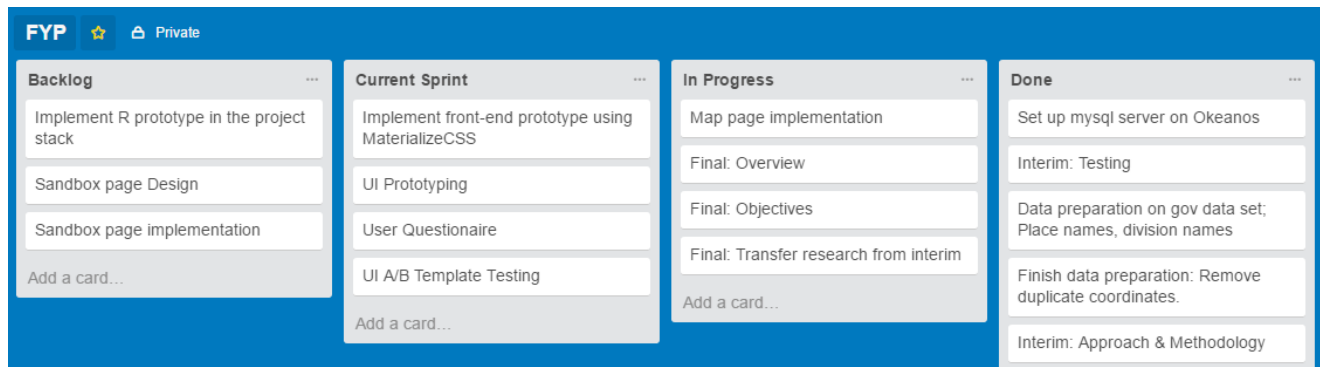


Figure 7: Snapshot of Trello issue tracking platform. ('Trello', 2016)

The scheduling for these sprints has been laid out as a Gantt chart. This chart represents the scheduling of all the possible sprints before the project deadline. I have chosen Trello as my issue tracking platform. This will contain a backlog of tasks that need to be completed. It will act as a repository in which tasks can be added to at any time. Its purpose is to act as a pool of items that can be brought into any given sprint. Once the tasks for a sprint have been decided, the sprint can begin. The progress on these tasks can be visually tracked by marking issues currently being worked on as "in progress". Once a task has been completed it can be moved to the done column. Trello as an issue tracking platform can support the carryover of tasks to the next sprint dynamically. It does this by making the current sprint a permanent feature. This means that the stage columns are always present and can have items moved between them depending on their status. Overall progress on a project can still be tracked as the moving of any item from one column to another is tracked in a log. ('Trello', 2016)

3.4. Design of Project Components

3.4.1. User Interface Design

This User Interface was designed with the use of Nielsen's Heuristics (Nielsen, 1995). Buttons and text boxes were designed to have labels and placeholders that match between the system and the real world. They were designed to be worded in a user-friendly manner whilst remaining relevant to the background operations that they trigger. User control and freedom was also utilised here. The sidebar navigation ensures that users don't get lost in the application and can navigate to any page. This navigation technique also encourages the principle of recognition rather than recall. Error prevention is implemented either in the web application itself or by redirecting to another page. Minimalist design is also used throughout, only displaying key information about how the application and its key features.

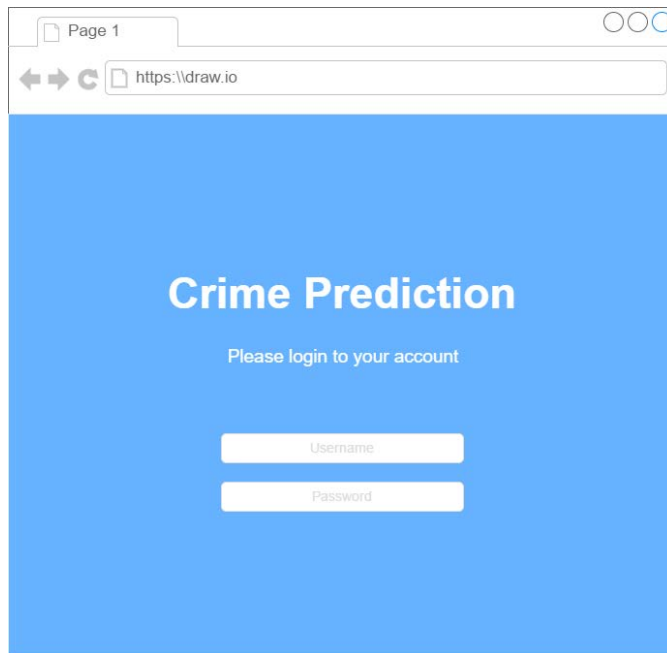


Figure 8: Wireframe of the login page.

The wireframe for the login page to this web application can be seen in Figure 8. The login and registration pages have been designed as simply as possible. With the use of the MaterializeCSS library, both logging in and registration should be achievable on the same page. The form will be designed dynamically and the relevant inputs will be displayed depending if a user is logging in or registering their account.

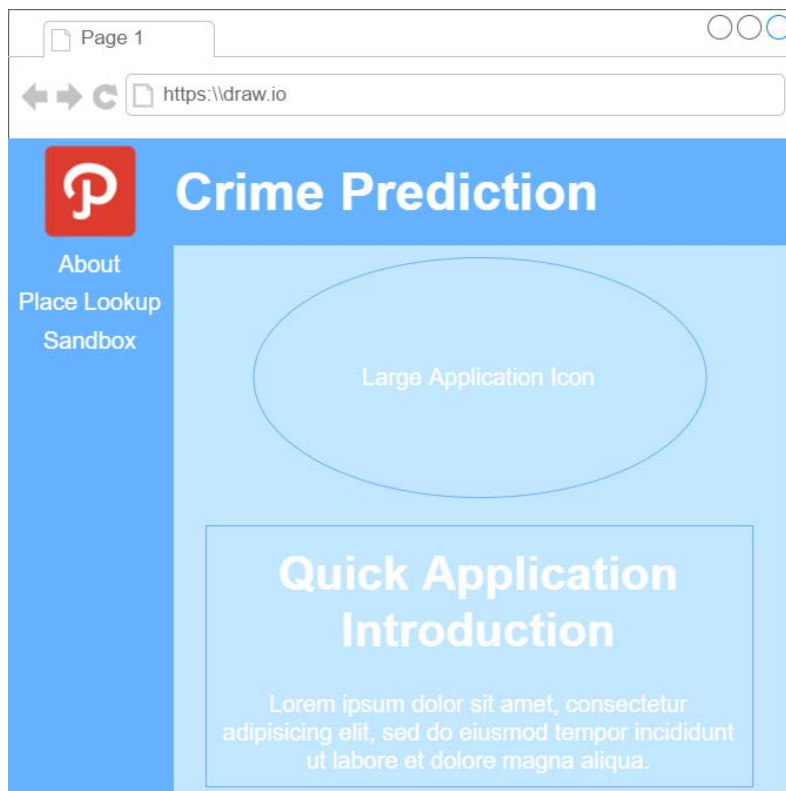


Figure 9: Wireframe of main page after a user logs into the application.

The wireframe for the page that the user will see after they login can be seen in Figure 9. This will act as a welcome page to the application. It will offer a brief explanation of the application. A full explanation can be found in the about section of the website. The sidebar navigation that is introduced on this page will be constant throughout the application after sign in. It will contain a search box and links to other pages that the user can access. To enable quick navigation to the core page of the web application, a “Get Started” button will redirect users to the place search page. For experienced users, the sidebar links will allow for quick access to any section of the application that they need.

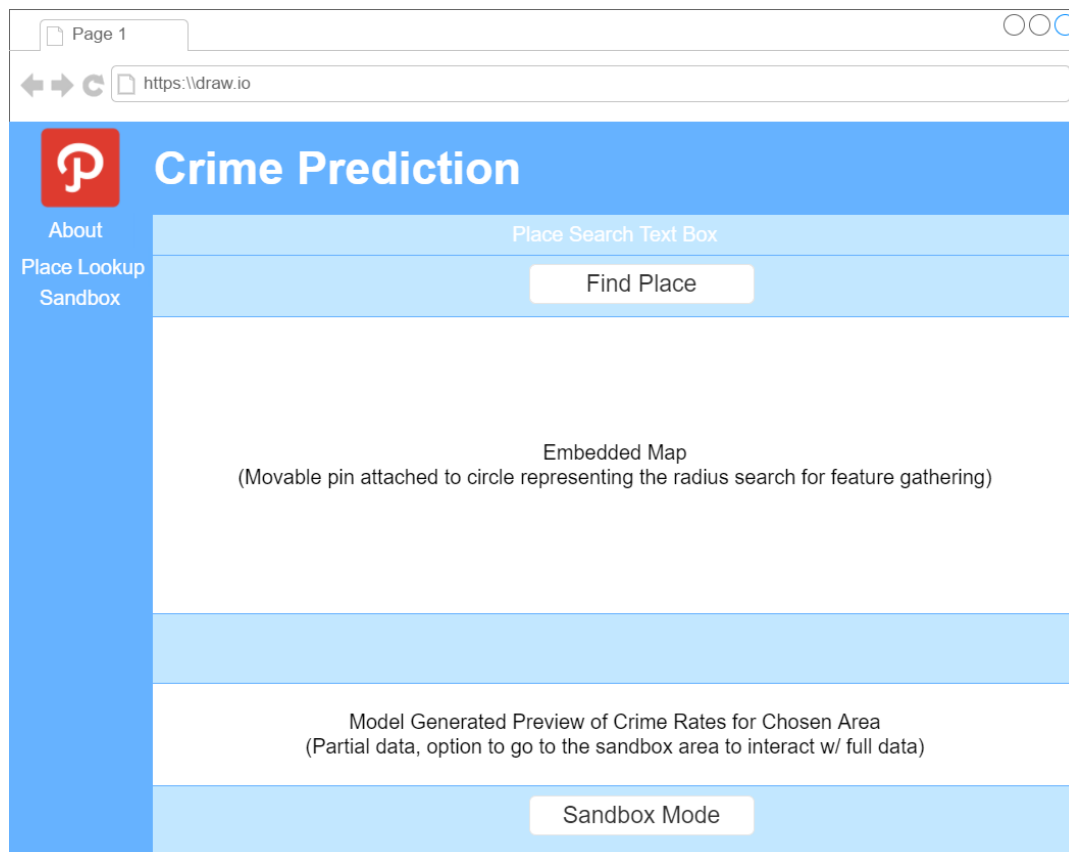


Figure 10: Wireframe showing the place search page.

The wireframe for the place search page can be seen in Figure 10. The purpose of this page is to allow a user to choose a place on a map that they would like to apply the model to.

3.4.2. Source Code Layout

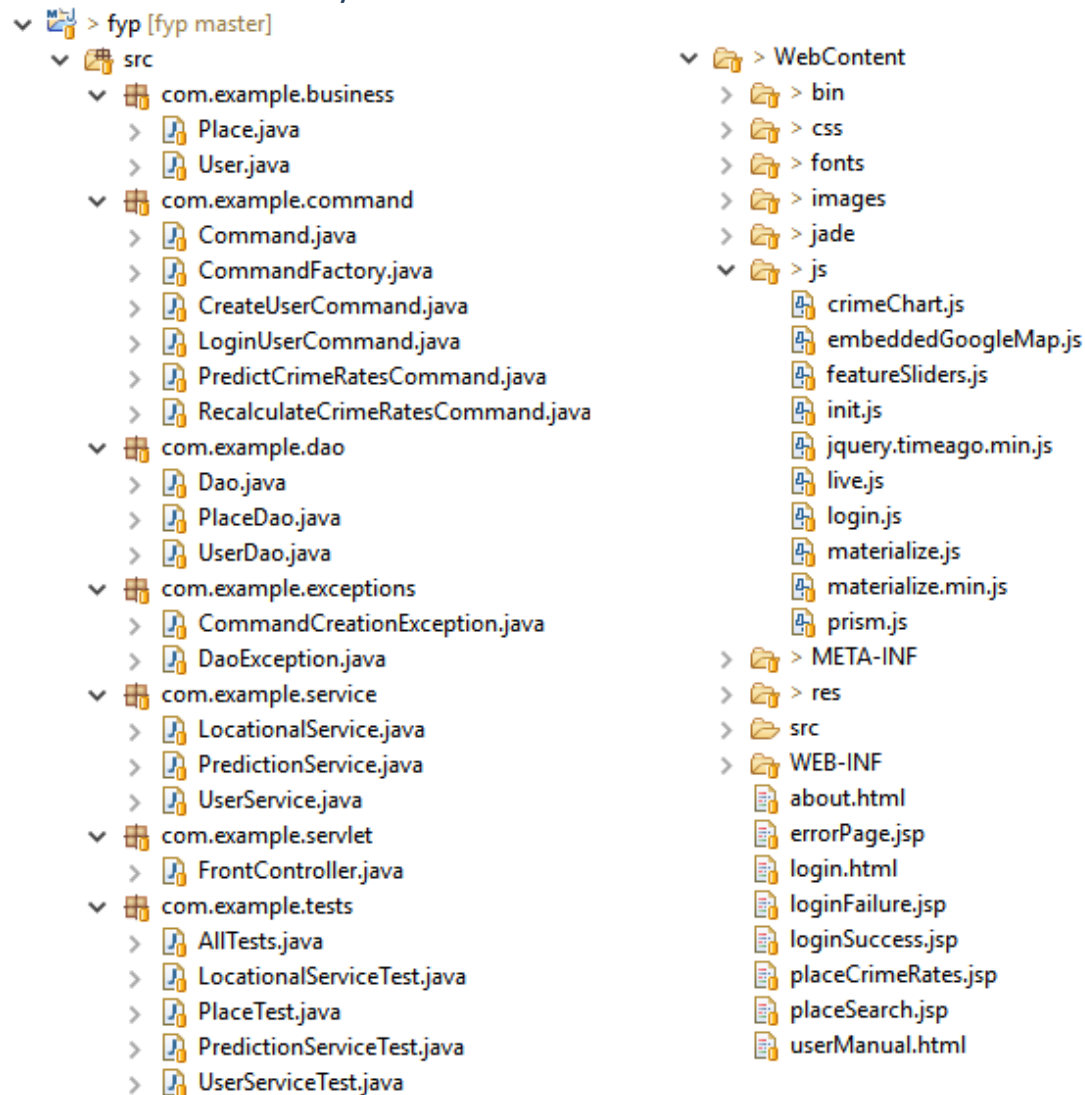


Figure 11: Layout of the source code for the CrimAnalytics web application as seen in the Eclipse IDE.

The layout of the source code for this project aims to solve the challenge of a separation of concerns. As seen in Figure 11, this is achieved by implementing different tiers of logic separately, in dedicated layers. To achieve this separation of concerns in the Java language, there is the concept of Java packages. A package allows a developer to group classes together based on related functionality. The package naming scheme and functionality groupings will be explained in this section.

3.4.2.1. *com.example.servlet*

This package contains logic related to receiving requests and sending back responses. Classes in this package aim to read HTTP requests and return the correct response. In relation to this project's web application, the requests can be to retrieve other pages or submit a command to the back end.

3.4.2.2. *com.example.command*

The classes contained in this package contain logic that processes a user's command and initiates the specific functionality that the command requires. Classes here will often instantiate all the necessary objects and services and pass on the relevant information from the HTTP requests.

3.4.2.3. *com.example.business*

The classes in this package contain simple classes that represent the relevant entities related to the project. These are usually concrete classes that represent specific entities and their fields. These classes act as wrapper classes that contain fields and getters and setters for those fields. These classes allow for simpler and cleaner access to these entities from anywhere in the codebase.

3.4.2.4. *com.example.service*

The classes in this package contain most of the detailed logic on how to interact with the entities from the business layer. This logic is mainly implemented as many methods that conduct specific operations on the data from the database access object layer.

3.4.2.5. *com.example.dao*

The database access object or DAO package contains the classes that communicate directly with the MySQL database. The "Place" entity in the database can be seen in Figure 26. The logic in this package is often directly requested by classes from the service package. The methods often build a specific type of query and take in the query filters as their parameters. The JDBC API is used in this package to achieve this communication. It is used to both build SQL queries to send to the database and process the response data. Each method will convert the response data to a Java type before returning it.

3.4.2.6. *com.example.exceptions*

The classes contained in this package contain logic to build meaningful exceptions that occur at the different levels of this structure. There are classes for each layer to handle exceptions and extract the relevant information from them. These classes exist to avoid useful information for bug fixing being lost in generic exception classes.

3.4.2.7. *com.example.tests*

The classes contained in this package contain the logic to create Unit and Integration Tests. To accomplish this, the JUnit framework was utilised. Thus, this package contains individual unit tests as well as a test suite. The test suite allows for the running of all the tests together. This makes tests quicker to check and changes easier to make.

3.5. Project Features and Use Cases

Clearly identifying the list of features and use cases supported within the project.

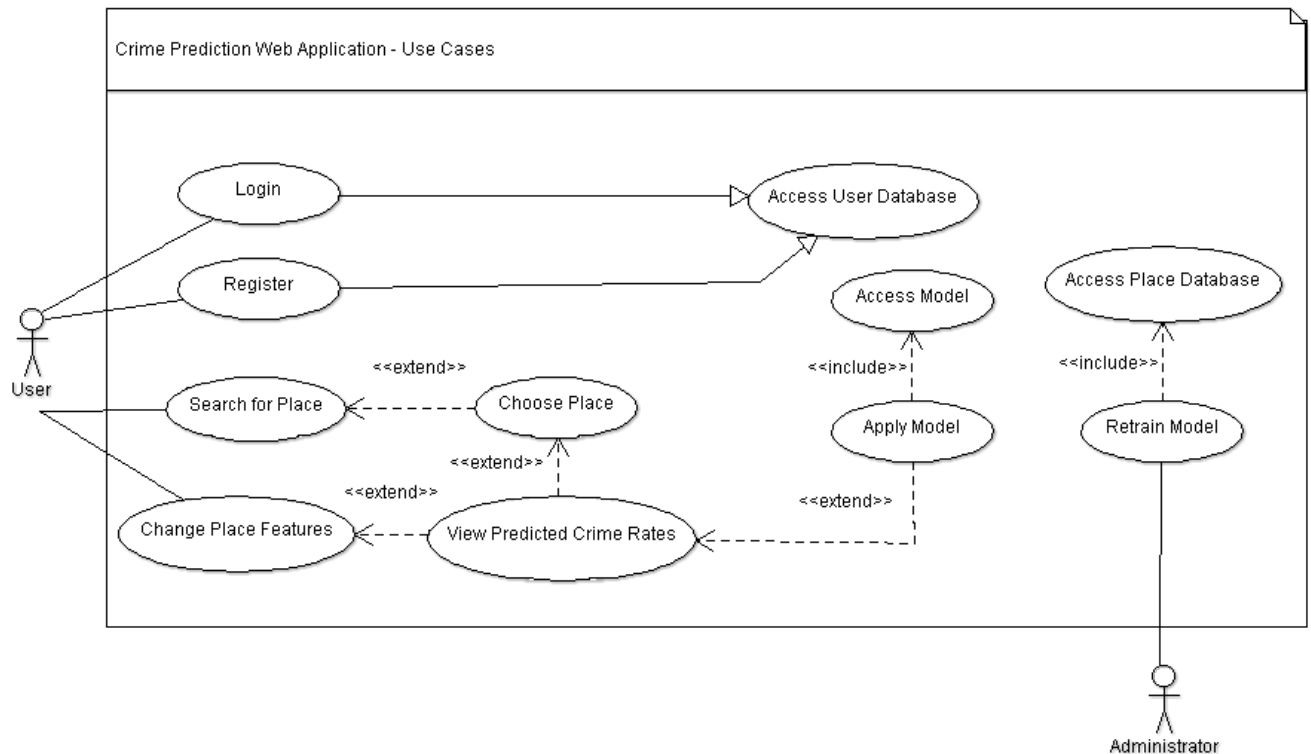


Figure 12: Use Case Diagram for the CrimAnalytics Web Application.

The use case diagram for the CrimAnalytics web application can be seen in Figure 12. Users of the application will have access to login and registration features before using the web application. They can also access and about page for the web application. This page will discuss how the web application works as well as the predictive model. Users will also have access to graphically search for a place. They will be able to choose a place and view the associated crime rates. Users will also be able to change the locational features of a place they have searched for and see how the changes affect the crime rates.

4. Architecture & Development

My solution will create a geographical based predictive model for the amount and type of crime in an area. The model will utilise geographic features from the Google Places API to predict for the crime types found in the government crime data set.

The front-end will be created using HTML, CSS and JavaScript. These technologies are well equipped to build the client side of the CrimAnalytics web application. HTML will be used to create the layout of the pages that will make up this web application. CSS will provide the styling for the web application. I plan to use an existing CSS library, Materialize CSS. This library will enable me to use a

material design theme throughout the application. Pure JavaScript along with the HighCharts library will allow for the creation of the rich elements in the web application. The main element here being the charts that will display the results of the application of the predictive model.

The back-end of this project will be built using Java Servlets and an MVC framework, also implemented in Java. The servlets will handle the communication between the client's embedded JSP scripts. This will allow for full separation between the business and presentation layers. The Java to R communication library called JRI will enable the Java classes in the business layer to communicate with the R engine. With this communication set up, the Java classes on the server side will be able to perform the training and application of predictive models.

4.1. System Architecture

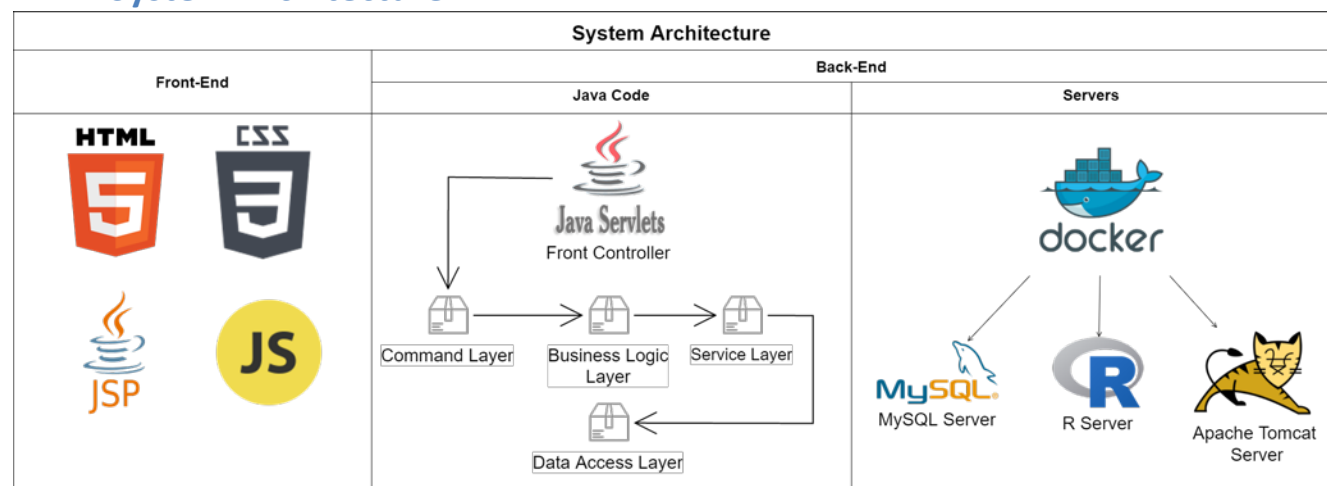


Figure 13: Technical Architecture. ('draw.io', 2016)

The diagram displayed above in Figure 13 describes the overall technical architecture and functionality of this project. This shows the flow of information between the client and the server. It also shows the different layers and technologies that allow for the data transfer between the client and the server. This design follows the MVC framework. This framework allows for the complete separation of the calculations and the interface.

4.1.1. Architecture Overview

The front-end is one component in this application's architecture. This layer utilises HTML, CSS and embedded JSP scripts. The JSP scripts will allow for the communication to the servlets. These will handle operations such as logging in and choosing specific areas in which to apply the predictive model. The response objects sent from these Java servlets can be easily displayed using JSP tags that mirror functionality from Java. It is the job of these servlets to receive HTTP requests and delegate to the service layer.

Another component in the architecture is the Java code. This code contains logic to access the model and perform key operations. In the case of the project, the training data is stored in the MySQL database and the methods that operate it exist in the Data Access Layer. These methods are contained within individual data access objects for each separate entity in the database.

The “Front Controller” is a Java Servlet that coordinates interactions between the front and back end. This controller’s purpose is to take commands from the front-end and call the relevant services. The command layer contains simple code that simply reflects what the process is doing. The “Service Layer” contains the complex logic that drives the application. With the “Data Access Layer” containing the logic required to retrieve data, these service classes are free to hold all their complexity in the operations that it performs for the client.

This separation on concerns should make it easier to create and work on tasks that are more specific during a sprint. The fact that each layer has defined boundaries means that the definition of done for any task can be more easily defined. Because of using this framework, tests can be implemented more easily also. As each layer is now decoupled, tests do not need to make calls to other layers. For example, a data access object can be easily mocked for a test of a service class. All the data access objects fields can be mocked also. In this case, it avoids having to set up a connection to the database and allows the test to execute faster.

4.1.2. Backend Services using Docker Containers

As seen in Figure 13, Docker was utilised for all the backend servers that the project required. Docker offers the ability to carve up any running Linux system into small containers. Each of these containers are sealed off from the Linux system itself and from other containers. A container can contain its own processes and files within itself, isolated from anything else on the system. These containers are designed to be portable and can easily move from one system to another. Docker contains the functionality to allow for the creation of these containers and their transport to and from systems (Docker Inc., 2015).

Docker is an open-source and community driven platform. Thus, there are many recipes available to create different types of containers. These recipes provide containers with all the dependencies and configuration files they need to run within themselves. In the context of this project that means that the R, MySQL and Apache Tomcat servers could all be set up with relative ease. Docker also offers the ability to open access ports to the systems they are running on. This allows for the containers to be contactable via the host system. This address follows the pattern of “host_address:container_port”. Because of the use of Docker in this project, the architecture has become very portable. Once the code for the project is loaded onto the three containers, the web application is fully deployed. The three containers could be deployed on any Linux system and the resulting web application would be identical in every regard (Registered Users, Predictive model and Web Application).

DIT
DT228 B.Sc. (Hons.) in Computer Science
Final Report 2016/17

```
docker pull stevenpollack/docker-rserve
docker run --name rserve -p 6311:6311 -d stevenpollack/docker-rserve

docker pull mysql:latest
docker run -p 3306:3306 --name mysqlServer -e MYSQL_ROOT_PASSWORD=***** -d mysql:latest

docker pull tomcat:7-jre8
docker run -d -it --rm -p 8080:8080 --name tomcat -d tomcat:7-jre8
```

Figure 14: Docker commands to set up the R, MySQL and Tomcat servers.

```
sean@snf-39667:~$ docker ps
```

CONTAINER ID	IMAGE	PORTS	NAMES
eac945568747	webserver	0.0.0.0:8080->8080/tcp	tomcat
03a7252bdb89	stevenpollack/docker-rserve	0.0.0.0:6311->6311/tcp	rserve
c11177e06996	mysql:latest	0.0.0.0:3306->3306/tcp	mysql

Figure 15: The successfully deployed containers and their exposed ports.

4.2. Development Components and Challenges

4.2.1. Java Services

One of the key development components in this project are the classes contained in the service layer of the web application. This layer contains the core logic of the web application. These classes are made easier to implement due to the separation of concerns in the codebase. This layout can be seen in chapter 3.4.2 *Source Code Layout*. These services include the Prediction Service, Locational Service and User Service. The Prediction Service contains the code used to access the R Engine and the model that the web application is based on. The Locational Service contains the code that utilises the Google Places API and retrieves all the locational data needed to submit to the predictive model.

4.2.2. User Interface Implementation

The user interface implementation was achieved using a combination of HTML, Java Server Pages, CSS and JavaScript. There is also a separation of concerns in the user interface aspect of this project. The HTML and Java Server Pages provide the basic structure for each page. The MaterializeCSS framework combined with the personally written JavaScript provide the functionality of the user interface. The MaterializeCSS framework provides reactive components on each page that perform well on a multitude of devices. The JavaScript handles all the data manipulation required to show the results from the predictive crime model.

4.2.3. Model Training in R

Another key component is the model training and application. This is all handled with the use of the R language. In R, the model is created using a Multiple Linear Regression algorithm and my formula of locational and past crime data. As a standalone R instance is always available to the web application, the trained model can be queried and return results easily. All the R code needed to facilitate this training and application of the model is sent from the Java based backend.

4.3. External APIs and Personal Code

4.3.1. External APIs

4.3.1.1. *Google Maps and Places APIs*

Both APIs provide geographic data, but in different forms. The Google Maps API focuses on representing its results on graphical maps, whereas the Google Places API focuses on returning geographic information in data transfer objects e.g. JSON and XML (Crouse, 2015; Google Developers, 2017).

The Google Places API was utilised to retrieve the locational data for the model. This was a key tool in this project as it provided the explanatory variables for the model. This API allows users to search for and retrieve rich information about local places.

To achieve this, the “Radar Search” section of the API was utilised. Whilst the “Place Search” provides more data on each place, each request will return only a maximum of 60 places in its response. This would be unsuitable as initial testing proved that the places in the government data set would have edge cases where the number of features would exceed the APIs limit. The “Radar Search” sacrifices the more detailed results for a larger number of results overall, at 200 per request. This sacrifice was acceptable as the model only required a count of the features in the response from the API.

The usefulness of both the Google Places and Maps APIs has resulted in libraries across many languages that enable communication with them. In this project, it is being contacted in both the front and back end to achieve different tasks. On the front-end the Google Maps API has been used to generate an embedded map in the web application using JavaScript. As both APIs are very similar in how they operate, the map can easily show users how their place search will operate. The embedded Google Map shows the user a moveable pin with a radius showing the area that their search will be applied to. This ties in with how requests for locational data are made in the backend using Java. The parameters in the Google Places API request are the geographic coordinates on the pin on the front end as well as the radius of the circle from the front end also.

The first major use of the Places API in this project is for its geolocation services, also known as geocoding. This service will allow for the translation of the place names contained in the government crime dataset to their corresponding coordinates. The geocoding web service will take the place name string and return the coordinates of the centre of that area. These coordinates will be used as the centre points of radius searches when collecting the locational explanatory variables later in the process.

```
97. GeocodingResult[] results = GeocodingApi.geocode(context, placeNames.get(i)).await();
98.
99. if (results.length > 0) {
100.     System.out.println(results[0].geometry.location);
101.     dao.updatePlaceLocation(placeNames.get(i), results[0].geometry.location);
102. } else {
103.     failed.add(placeNames.get(i));
104. }
```

Figure 16: Current usage of the Maps API Geocoding feature. [src/com.example.service.LocationalService.java]

In Figure 16 a method performs a ‘geolocation’ process. This process is provided by the Places API. Geolocation is the process of converting a place name to its geographical coordinates. The Java implementation in Figure 16 goes through each place name and gets its coordinates. The data access layer handles the retrieval and updating of each place.

The next use of the Google Places API is for the rich data it can return about the places in each area. Using the coordinates from the previous step a “nearby search” can now be performed using the Places API. This API call also requires a radius parameter, which is a radius in metres around the coordinates. This call returns a list of places with their details. This data can be passed back in either JSON or XML format. The Places Java library provides a wrapper class for a place object. It is also possible to pass several filters to the API to tailor the list that is returned. This is what will be done to gather specific place data for the predictive model.










Name	
 education	amusement_park
 food	art_gallery
 government	aquarium
 health	bar
 recreational	bowling_alley
 services	campground
 stores	casino
 transport	movie_rental
 worship	movie_theater
	museum
	night_club
	park
	rv_park
	stadium
	zoo

Figure 17: Groups of place types from the API that will be relevant to the predictive model.

```
163. try {
164.     features = client.getPlacesByRadial(lat, lng, radius, 200, Param.name("types").value(subFeatures));
165. } catch (GooglePlacesException e) {
166.     if (e.getCause() instanceof NoResultsFoundException) {
167.         System.out.println("No " + feature + "s found at " + lat + ", " + lng + ".");
168.     } else {
169.         throw e;
170.     }
}
```

Figure 18: Usage of the Place API to collect place features. [src/com.example.service.LocationalService.java]

Above, in Figure 18 there is an implementation of this feature gathering functionality. A trade is made in the selection of the specific functionality here.

Whilst a *nearby places* search returns objects with more information about each place, a *radar* search has a higher limit for places that can be retrieved by one API call. As all that is needed from this part of the API is a count for each type of place around an area, the *radar* search was chosen. The latitude and longitude coordinates are the first two parameters passed to this method. Then a radius (in metres) around the coordinates to search in is passed in. The next parameter is a filter on the maximum number of Place objects to return (Crouse, 2015) (Google Developers, 2016).

4.3.1.2. *Java Database Connectivity API*

The Java Database Connectivity API provides a way for the Java based backend to communicate with the MYSQL database. To achieve this, it provides a wrapper “Connection” class that holds all the variables need to connect and login to the database. It also provides similar wrapper classes for querying and receiving results from the database. These classes are called “Statement” and “ResultSet”. The Statement class allowed for the building of SQL queries in the Java language. The “ResultSet” class made it easy to receive the desired data back from requests and convert it to types that Java understands.

4.3.1.3. *JSTL – JavaServer Pages Standard Tag Library*

This library provides a collection of tags that mirror common functionality in JSP applications. This allows an otherwise HTML front-end page to communicate with a Java based back-end. The main use of this library is its ability to send and parse Java objects. This allows the front-end to access potentially complex Java objects with ease. Thus, richly populated data can be transferred to the front-end of an application for display.

4.3.1.4. *R Engine Interface*

The R Engine interface acts similarly to the JDBC API. It allows for communication between Java applications and an R server. This will be very useful for this project as the model that lies at the core of the functionality, is trained, accessed and applied via a remote R server. This interface allows the Java back-end to formulate the data to apply the model to. This data can then be easily sent on to the remote R server. The interface also offers several tools to parse returning data into Java primitives.

4.3.2. Personal Back-End Code

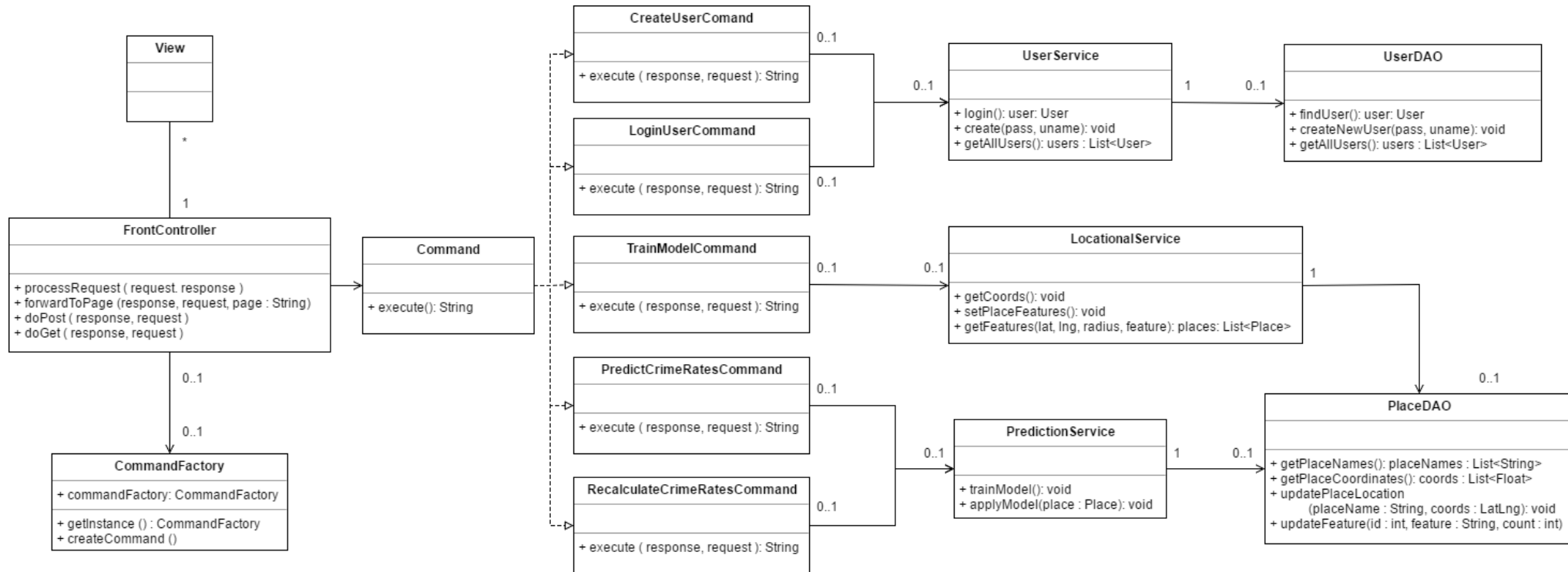


Figure 19: Class Diagram for the CrimAnalytics web application.

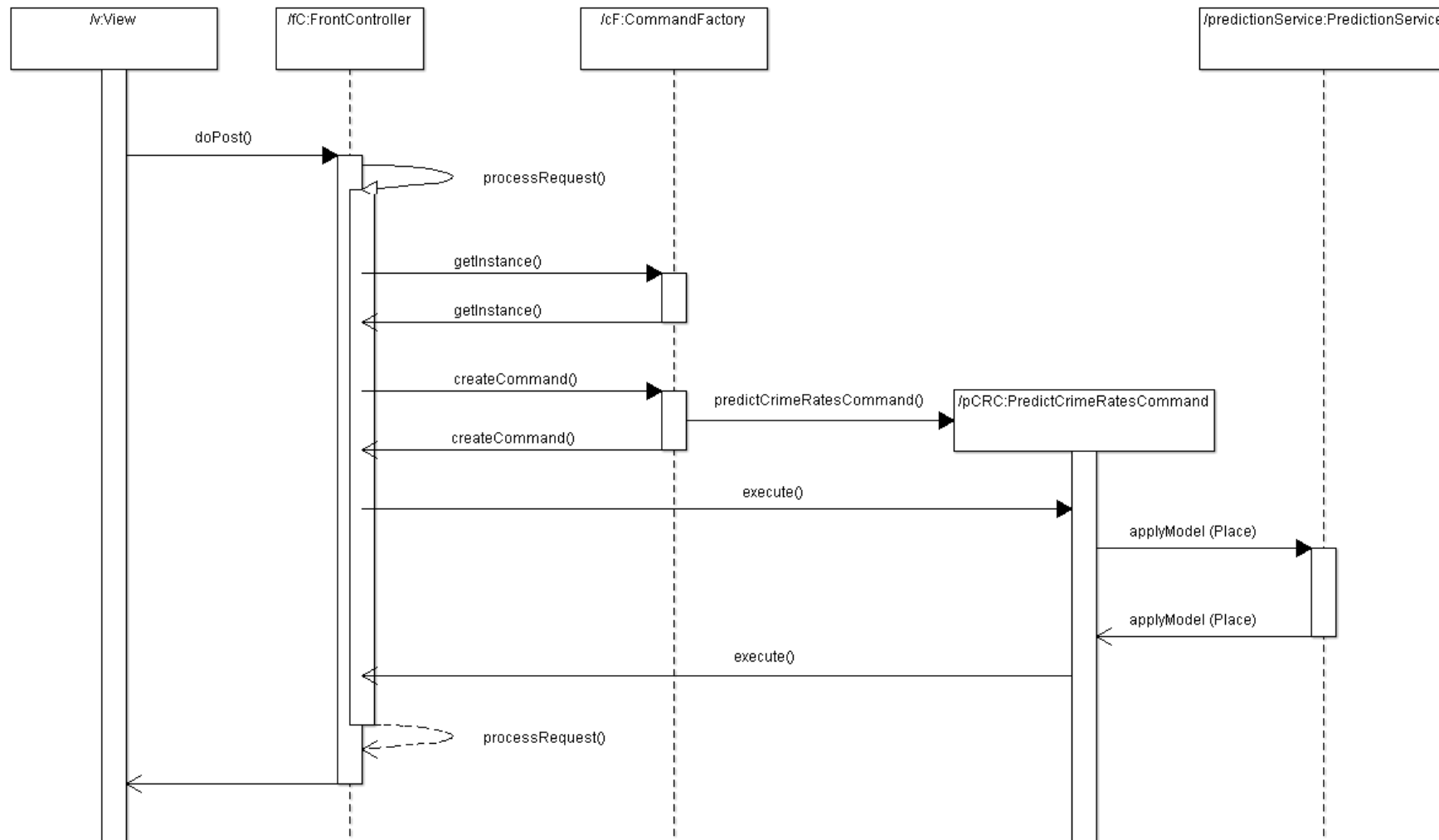


Figure 20: Sequence Diagram showing the application of the predictive model.

The personal code on the back end is written in Java. A complete class diagram of all the personal Java code can be seen in Figure 19. A detailed description of how this code laid out in separate packages can be seen in *Section 3.4.2 Source Code Layout*.

The sequence diagram shows how the CrimAnalytics back-end handles a command to predict the crime rates in an area. The command is first received by the FrontController. The FrontController is a Java Servlet that accepts HTTP requests, processes their command request and redirects it to a matching command class. In this case the HTTP request sends a “PredictCrimeRates” command and is redirected to the “PredictCrimeRatesCommand” class. This class contains all the logic required to process any other parameters sent with the command and pass them on to the relevant service class. In this case the service class is the “PredictionService” class. This class contains the “applyModel” method that contains all the logic to take in the command parameters, pass them to the R model and return the resultant crime rate values. The predicted crime rates from the model are then returned to the view.

To facilitate the training of the model, personal SQL was also written. When training the model, the recent crime data and locational data had to be merged together from different sources. To easily achieve this, both sources were parsed in Java and sent to the database where they could exist as a single entity. Each row in the database represents a place that was used to train the model. Each place has the known crime rates. Each place also has several locational features that explain the exact crime rates in that place. All the attributes relating to a place in the database can be seen in Figure 26.

```
03.  /**
04.   * Common method to process all client requests (GET and POST)
05.   *
06.   * @param request
07.   *       From the front-end.
08.   * @param response
09.   *       From the back-end.
10.   */
11.  private void processRequest(HttpServletRequest request, HttpServletResponse response) {
12.
13.      String forwardToJsp = null;
14.      String action = request.getParameter("action");
15.
16.      // Check if this is not a login request.
17.      if (!action.equalsIgnoreCase(LOGIN_ACTION)) {
18.
19.          // If not a login request then need to check that user is
20.          // logged in before processing ANY requests.
21.
22.          // Check to see if the session id coming from the client matches the
23.          // id stored at login.
24.          HttpSession session = request.getSession();
25.
26.          // If user not logged in.
27.          if (session.getId() != session.getAttribute("loggedSessionId")
28.              && !action.equalsIgnoreCase(CREATE_ACTION)) {
29.              forwardToJsp = "/loginFailure.jsp";
30.              forwardToPage(request, response, forwardToJsp);
31.              return;
32.          }
33.      }
34.
35.      // Process the command contained in the request.
36.      CommandFactory factory = CommandFactory.getInstance();
37.      Command command = null;
38.
39.      try {
40.          command = factory.createCommand(action);
41.          forwardToJsp = command.execute(request, response);
42.      } catch (CommandCreationException e) {
43.          e.printStackTrace();
44.          forwardToJsp = "/errorPage.jsp";
45.      }
46.
47.      forwardToPage(request, response, forwardToJsp);
48.  }
```

Figure 21: processRequest method from the FrontController class. [src/com.example.servlet.FrontController.java]

One of the main methods in the implementation of the Java Servlet can be seen in Figure 21. This method takes an action string from the user interface and connects it with the corresponding command class. It ensures that the user is logged in before processing any actions. The Command Factory is then used to instantiate and execute the command. The command will perform the necessary operations to complete the action and return the page name to send the user to.

DIT
DT228 B.Sc. (Hons.) in Computer Science
Final Report 2016/17

```
21. @Override
22. public String execute(HttpServletRequest request, HttpServletResponse response){
23.
24.     HttpSession session = request.getSession();
25.     LocationalService locationalService = new LocationalService();
26.     PredictionService predictionService = new PredictionService();
27.     String forwardToJsp = "";
28.
29.     // Retrieve the coordinates from the request.
30.     Float lng = new Float(request.getParameter("lng"));
31.     Float lat = new Float(request.getParameter("lat"));
32.     String placeName = request.getParameter("placeName");
33.
34.     if (lng != null && lat != null){
35.
36.         HashMap<String, Integer> featuresHashMap = locationalService.findFeatures(lat, lng);
37.
38.         Place p = new Place(featuresHashMap.get("educational"), featuresHashMap.get("food"),
39.                             featuresHashMap.get("government"), featuresHashMap.get("health"),
40.                             featuresHashMap.get("recreational"), featuresHashMap.get("stores"), featuresHashMap.get("services"),
41.                             featuresHashMap.get("transport"), featuresHashMap.get("worship"));
42.
43.         p.setName(placeName);
44.
45.         // Delegate command to specific functionality in the corresponding
46.         // service class.
47.         session.setAttribute("place", predictionService.applyModel(p));
48.
49.         if (!featuresHashMap.isEmpty()){
50.             forwardToJsp = "/placeCrimeRates.jsp";
51.         }else{
52.             forwardToJsp = "/loginSuccess.jsp";
53.         }
54.     }
55.     return forwardToJsp;
56. }
```

Figure 22: Predicting crime rates. [src/com.example.command.PredictCrimeRatesCommand.java]

An example of how a Command executes can be seen in Figure 22. This example shows the command for the key feature of predicting the crime rates in an area. The coordinates are extracted from the request object and used to create a new Place object. The place object is passed to two separate service classes to complete the command. These service classes contain the specific functionality needed to run commands. In this case the locational services is used to find the locational features in the area and the prediction service is used to predict the crime rates. Depending on the success or failure of these services, different pages can be returned to the user.


```
141. /**
142.  * Use the Google Places API to get counts for the locational feature
143.  * categories.
144.  *
145.  * @param lat
146.  *         Latitude.
147.  * @param lng
148.  *         Longitude.
149.  * @param radius
150.  *         Radius in kilometres to perform the search in.
151.  * @param feature
152.  *         The feature category to be searched for.
153.  * @param subFeatures
154.  *         The Places API tags that make up the locational feature
155.  *         category.
156.  * @return {@link Place} with all locational variables populated.
157.  */
158. private List<Place> getFeatures(final float lat, final float lng, final int radius, final String feature,
159.                                final String subFeatures) {
160.     final GooglePlaces client = new GooglePlaces(PLACES_API_KEY);
161.     List<Place> features = new ArrayList<Place>();
162.
163.     try {
164.         features = client.getPlacesByRadar(lat, lng, radius, 200, Param.name("types").value(subFeatures));
165.     } catch (GooglePlacesException e) {
166.         if (e.getCause() instanceof NoResultsFoundException) {
167.             System.out.println("No " + feature + "s found at " + lat + ", " + lng + ".");
168.         } else {
169.             throw e;
170.         }
171.     }
172.
173.     return features;
174. }
```

Figure 23: Finding the locational features. [src/com.example.service.LocationalService.java]

The implementation of the feature finding functionality referenced in Figure 22 can be seen in Figure 23. This is a method in a class from the service layer. This means that all the low-level functionality can be found here. In this case, the core of the locational feature finding can be seen here. The longitude, latitude and radius for the search are passed to the Google Places API. The API method “getPlacesByRadar” takes the feature name, coordinates and radius and inputs and returns the counts for the different locational features in the area e.g. educational, recreational, services etc.

```
51. /**
52.  * Apply a place to the predictive model and return the predicted crime
53.  * rates.
54.  *
55.  * @param place
56.  *      The {@link Place} to apply the model to.
57.  * @return The {@link Place} with the crime rate category variables
58.  *      populated.
59.  */
60. public Place applyModel(Place place) {
61.     try {
62.
63.         final String placeFeaturesString = String.format(
64.             "educational=%s, food=%s, government=%s, health=%s, recreational=%s,
65.             stores=%s, services=%s, transport=%s, worship=%s",
66.             place.getEducational(), place.getFood(), place.getGovernment(), place.getHealth(),
67.             place.getRecreational(), place.getStores(), place.getServices(), place.getTransport(),
68.             place.getWorship());
69.
70.         this.trainModel();
71.
72.         c.eval("preds = round(predict(MLR_Model, data.frame(" + placeFeaturesString + "));"
73.             + "preds[preds < 0] <- 0;");
74.         REXP predictedCrimeRates = c.eval("preds");
75.         int[] dependentVariables = predictedCrimeRates.asIntegers();
76.         System.out.println(Arrays.toString(dependentVariables));
77.
78.         // Field setting excluded for brevity.
79.
80.     } catch (Exception e) {
81.         e.printStackTrace();
82.     } finally {
83.         if (c != null) {
84.             c.close();
85.         }
86.     }
87.
88.     return place;
89. }
```

Figure 24: Predicting the crime rates in an area. [src/com.example.service.PredictionService.java]

The implementation of the crime rate prediction functionality referenced in Figure 22 can be seen in Figure 24. This method is also in the service layer. This specific functionality is to the application of a place to the model. To achieve this the JRI library is used. This can be seen in the “c” variable of type RConnection and the response wrapper type REXP (R Expression). The locational data is extracted from the “place” object and passed to the model. The predicted crime rates are extracted from the R Expression. The “place” object is then updated with these crime values and returned.

```
158.      /**
159.       * Update a locational feature count for a {@link Place}.
160.       *
161.       * @param placeID
162.       *       The ID of the {@link Place} to update.
163.       * @param feature
164.       *       The feature category to update.
165.       * @param count
166.       *       The number of the features to set.
167.       * @throws DaoException
168.       *       If a data access exception occurs.
169.       */
170.     public void updateFeature(final int placeID, final String feature, final int count) throws DaoException {
171.         System.out.println("Updating feature...");
172.         Connection con = null;
173.         PreparedStatement ufps = null;
174.         ResultSet ufrs = null;
175.         try {
176.             con = this.getConnection();
177.
178.             String updateFeatureQuery = "UPDATE test_fyp SET " + feature + " = ? WHERE placeID = ?";
179.             ufps = con.prepareStatement(updateFeatureQuery);
180.             ufps.setInt(1, count);
181.             ufps.setInt(2, placeID);
182.             System.out.println(ufps);
183.
184.             ufps.executeUpdate();
185.
186.         } catch (SQLException e) {
187.             throw new DaoException("updateFeature " + e.getMessage());
188.         } finally {
189.             try {
190.                 if (ufrs != null) {
191.                     ufrs.close();
192.                 } if (ufps != null) {
193.                     ufps.close();
194.                 } if (con != null) {
195.                     freeConnection(con);
196.                 }
197.             } catch (SQLException e) {
198.                 throw new DaoException("updateFeature " + e.getMessage());
199.             }
200.         }
201.     }
202. }
```

Figure 25: Code from the data access layer used during the model training.

Some code from the data access layer can be seen in Figure 25. This updateFeature method was utilised adding the locational data to the crime data in the MySQL database. This method persists the number of a specific locational feature in the database. The code written in this data access layer provided the ability to combine and prepare the training data for the model.

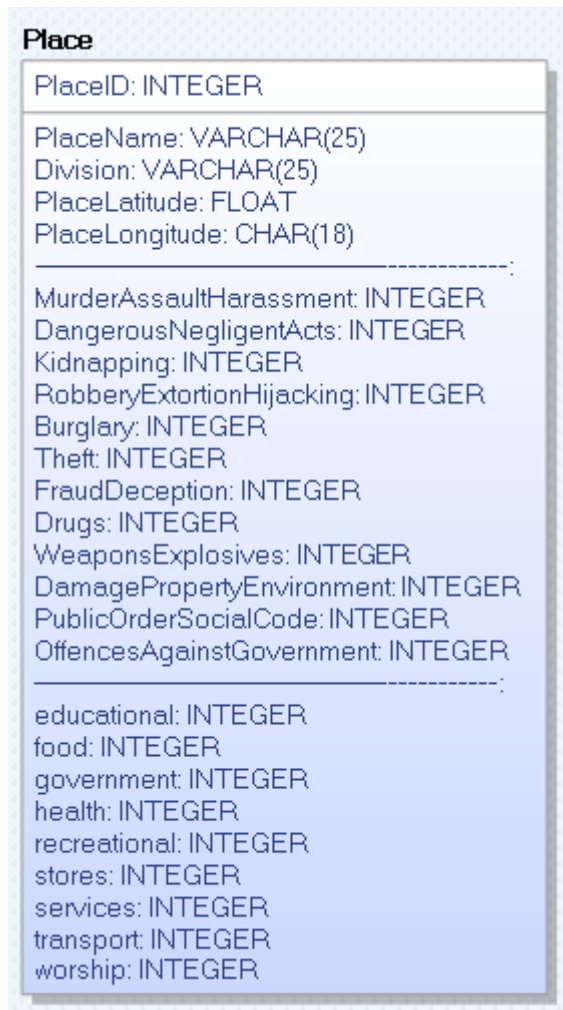


Figure 26: The “Place” entity.

The Place class is used throughout the codebase and the database entity for it can be seen in Figure 26. This entity contains the basic information about the place along with the number of locational features and crime rates. This entity has a corresponding Java class that wraps these fields so they can be transported around the back-end of the CrimAnalytics web application.

4.3.3. Personal Front-End Code

The personally written code is mostly comprised of HTML and JavaScript. The CSS is mostly provided by the MaterializeCSS library with some personal editing of sizing, colouring and additional other styling as needed.

```
68. <div class="section no-pad-bot" id="index-banner">
69.   <div class="container">
70.     <h1 class="header center-on-small-only">CrimAnalytics</h1>
71.     <div class="row center">
72.       <h4 class="header col s12 light center">A project to build and
73.         utilise a predictive crime model.</h4>
74.     </div>
75.     <div class="row center">
76.       <a href="placeSearch.jsp" id="download-button"
77.         class="btn-large waves-effect waves-light">Get Started</a>
78.     </div>
79.     <div class="row center">
80.       <a class="red-text text-lighten-4"
81.         href="https://github.com/seanjennings/fyp">Github Repository</a>
82.     </div>
83.     <br>
84.   </div>
85. </div>
86. </div>
```

Figure 27: Example of Personal HTML with the “class” tag references to MaterializeCSS styling.
[WebContent/loginSuccess.jsp]

The HTML code written for the web application provides features available on every page such as the sidebar navigation, headers and footers. The HTML also provides the positioning of containers for other, more complex components. For the complex components, such as the map search, crime rates chart and crime rate sliders, JavaScript is used. The use of JavaScript allows the complex components to gain richer features that could not be possible using HTML alone.

```
13. map = new google.maps.Map(document.getElementById('mapCanvas'), {
14.   zoom : 12,
15.   center : results[0].geometry.location,
16.   mapTypeId : google.maps.MapTypeId.ROADMAP,
17.   scrollwheel: false,
18.   streetViewControl: false,
19.   styles : [...]
20. });
```

Figure 28: Embedded map script code snippet. [WebContent/js/embeddedGoogleMap.js]

The JavaScript that was written to display the map on the “Place Search” page can be seen in Figure 28. The Google Maps JavaScript API was used to achieve this. The addition of the marker on the map and the circle around it are implemented similarly. Event listeners are placed on the marker to track it as the user moves it.

```
14. Highcharts.chart('container', {
15.   chart: {
16.     type: 'column'
17.   },
18.   credits: {
19.     enabled: false
20.   },
21.   exporting: {
22.     enabled: false
23.   },
24.   title: {
25.     text: 'Crime Rates within two kilometres of '
26.     + document.getElementById("placeName").value
27.     + '<br>(Six Month Period)'
28.   },
29.   subtitle: {
30.     text: 'Results produced by predictive model'
31.   },
32.   xAxis: {
33.     categories: [
34.       'Murder, Assault or Harassment',
35.       'Dangerous or Negligent Acts',
36.       'Kidnapping',
37.       'Robbery, Extortion or Hijacking',
38.       'Burglary',
39.       'Theft',
40.       'Fraud or Deception',
41.       'Drugs',
42.       'Damage to Property or the Environment',
43.       'Public Order and Social Code Offences',
44.       'Offences Against the Government'
45.     ],
46.     crosshair: true
47.   },
48.   yAxis: {
49.     min: 0,
50.     title: {
51.       text: 'Number of crimes recorded'
52.     }
53.   },
```

Figure 29: Implementation of the crime rate chart using HighCharts. [WebContent/js/crimeChart.js]

The implementation of the chart that is displayed on the “Place Crime Rates” page can be seen in Figure 29. The model response values are piped in here and displayed to the user as the crime rates of their selected area. A HighCharts bar chart is used with the crime categories on the x-axis and the numbers for each on the y-axis.

5. System Validation

5.1. Testing

5.1.1. Model Testing and Evaluation

The testing and evaluation of the predictive model is a key component of this project. The accuracy of the model will contribute massively to the quality of this project’s outcome. The two main measures that will be used to evaluate the model are the R^2 and p-value.

The sums of squares values seen in Figure 2 allow for the calculation of the R^2 statistic, also known as the coefficient of determination. It is the fraction of variability (as indicated by the sum of squares) explained by the line that has been fitted.

$$\text{Coefficient of Determination} \rightarrow R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

$$\text{Sum of Squares Total} \rightarrow SST = \sum (y - \bar{y})^2$$

$$\text{Sum of Squares Regression} \rightarrow SSR = \sum (y' - \bar{y}')^2$$

$$\text{Sum of Squares Error} \rightarrow SSE = \sum (y - y')^2$$

Figure 30: Coefficient of Determination (Sayad, 2010)

The R^2 statistic is a score from zero to one on the accuracy of the model. It is a measure of the overall variance in the predicted and actual values. As it describes the ability of the model to predict for the dependent variable, it is a key output of Linear Regression.

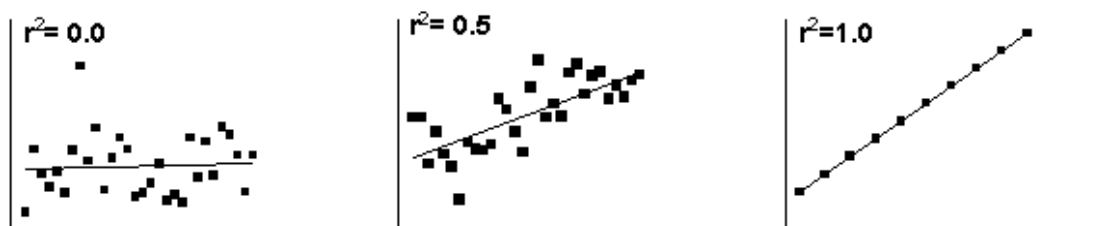


Figure 31: Visual examples of R^2 values and how they represent variance. (Markham, 2016)

A visual reference for how the R^2 statistic represents variance in regression models can be seen in Figure 31. The process of calculating this value is done after the model has been trained. With the training complete the calculation can compare the models line of best fit with the data that was used to train it. This comparison provides a vital measure for the accuracy of regression models.

The p-value is a measure of the strength of the evidence against the null hypothesis. In the case of this project the null hypothesis is that locational data has no effect on the rates of crime in an area. The p-value represents how strongly there is evidence against the null hypothesis. In the case of this project, a low value implies

DIT
DT228 B.Sc. (Hons.) in Computer Science
Final Report 2016/17

that there is a relationship between the locational features and crime rates in an area. The lower the p-value is, the stronger the evidence is against the null hypothesis.

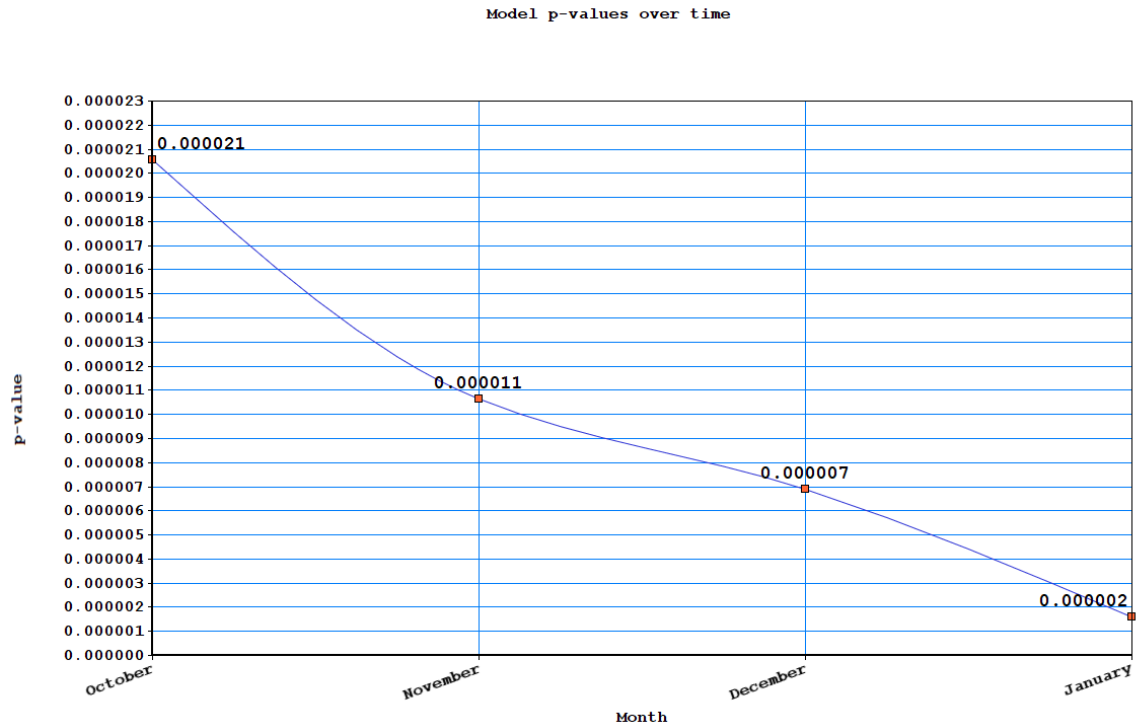


Figure 32: Model p-values over time.

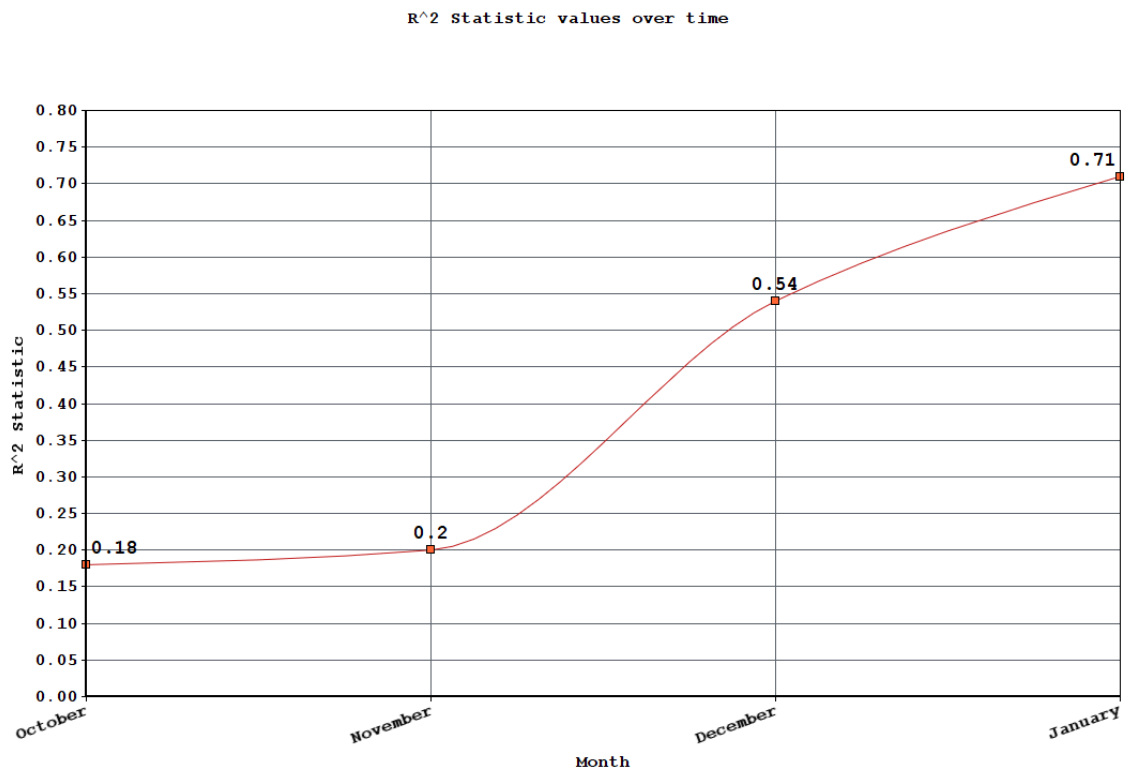


Figure 33: Model R^2 values over time.

The model was evaluated at several intervals during development. The results of these evaluations can be seen in Figure 32 and Figure 33. The R^2 statistic raised significantly over the course of the model's development. This indicated that the model was decreasing the variance in its predictions and becoming more accurate. This was due to ongoing data preparation and the removal of outliers over time. The outliers mostly consisted of certain crimes being prosecuted in different areas than where they were committed. This was due to there being better facilities in the form of Garda Stations or Courts in other areas. The areas with these better facilities had their crime rates artificially inflated. To deal with this, those places were excluded during the training of the model, resulting in better accuracy.

Similarly, the p-value decreased over time, indicating a strengthening in evidence against the null hypothesis, that locational features do not affect crime. Although the government crime data set contained historical data dating from 2003 to 2016, the training data was limited to that in 2016. This was done to ensure that the model was being trained with crime data that was relevant to the up to date locational data. These processes resulted in a more accurate line of best fit in the model. Thus, the variance in the predicted values fell and gave more accurate results for the different crime rates. The p-value was low to begin with and continued to lower as the training data was improved. This was encouraging as it proved that there was strong evidence to suggest that locational features effected the crime rates in an area.

5.1.2. Java Unit Testing

Unit tests were created for all the key Java classes and their methods. These tests will cover both positive and negative scenarios. This means that both correct and incorrect data will be passed to a method. This will test the method to see if it works correctly in positive use cases and handles errors correctly in negative use cases. This testing will be achieved with the JUnit library for Java. The use of the separation of concerns in the codebase allows for simple test implementation. All the objects that a class needs to carry out a test can be easily mocked using existing classes from other layers. This also means that the tests will run very quickly as actual calls to other layers do not need to be made and will be mocked instead.

The reason for using the Unit Testing methodology is to catch more bugs in the system during development instead of at the end of the project. This methodology ensures that developers know what is required for their work. This is because following the methodology ensures that there are previous components that have and a specific purpose. This makes future work that builds on them clearer and easier to integrate.

This Unit Testing was achieved with the use of the JUnit framework. It allows developers to write repeatable tests for Java methods. Developers can set up expected inputs and outputs for a method. The tests are comprised of assertions that the actual output matches the expected output. These tests ensure that any

changes to existing code do not cause any unexpected behaviour. As each of these tests are tied to a single class, a JUnit test suite has also been implemented. This test suite combines these individual tests together. Thus, all the tests can be run together to quickly and easily spot any failures in the code.

5.1.3. Java Integration Testing

Integration testing was also performed on the web application during this project. This testing uses the same JUnit framework used to implement the unit tests from the previous section. However, the unit tests are not sufficient in testing end-to-end functionality. To achieve this, integration tests were implemented. The purpose of these tests was to ensure that the layers in this project communicate with each other correctly. These layers can be seen in how the source code is laid out in Figure 11.

The end-to-end processes reflect those which would be performed when carrying out a real-world use case. When performing a process to satisfy a real use case, all the layers in the codebase are utilised. Because of this, it is important to test that these layers are communicating correctly. The JUnit framework was employed once again to implement these tests. The integration tests are much longer and more complex than the individual unit tests. This is because they must instantiate and manipulate multiple elements across the different layers of the codebase.

5.1.4. User Testing and Web Application Evaluation

The user testing will involve the testing of the CrimAnalytics web application with randomly selected people. The main aim for this methodology is to find out how suitable the user interface is for selecting a place and viewing the predictions for the crime rates there. Thus, testing will allow for feedback on the flow of the screens in the user interface as well as the visual display of the predicted data. This testing will be conducted during the sprints dedicated to user interface development.

I have conducted User Testing to ensure that the Graphical User Interface of this project allows users to effectively access and see results from the predictive crime model. This testing was done by using a questionnaire. This questionnaire covered many different aspects of the project, from menu layouts to data representation.

The application was made available to users at a very early stage in its development cycle, and this is in line with the agile development methodology. Feedback could be quickly acted upon and implemented. New versions of the web app could be deployed to the cloud based tomcat server at any time. This development process made it extremely easy for users to suggest new features or to mention problems that they faced and within minutes, the updated application was made available to all users.

The aim of the questionnaire was to gain insights from the users of the CrimAnalytics web application. The start of the questionnaire gathered information about their interest in crime analytics and if they had any prior knowledge of data prediction. This information gave some context to the feedback and helped in future design choices.

To ensure that this application could be tested with a diverse group of people, different methods were used to receive feedback. The initial user questionnaire was designed using the Google Forms site. This site also offers a shareable link that can be sent to users to fill out. This proved a useful method when the user evaluation was being performed remotely, but was inconvenient when carrying out the testing in person. To solve this, the Google Form was converted into a physical feedback form. With these physical forms, users could more easily provide on the spot feedback. This method also proved very useful when getting feedback from people outside of the easily available young adult user group. The aim of this application is that it can be used and be useful to people across many different age groups and abilities. Thus, having this method of feedback proved very useful in building a more accessible application.

Usability Testing & Bug Report

Thank you for visiting this Crime Prediction web application 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

- Seán Jennings

*Required

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

	1	2	3	4	5	
None	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Expert

Do you believe it is possible to predict crime rates using mathematical algorithms? *

	1	2	3	
No	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Yes

Figure 34: User Feedback Form - Part 1

Did you like the design of this web application, please leave comments at the end *

1 2 3 4 5

Hated it ☐ ☐ ☐ ☐ ☐ Loved it

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

☐ Low

☐ Medium

☐ High

☐ Critical

Type a brief summary of any problems encountered and how this may be reproduced

Your answer _____

Is there anything you would like seen added to this web application?

Your answer _____

Any last comments?

Your answer _____

SUBMIT

Figure 35: User Feedback Form - Part 2

The feedback form that was completed by users can be seen in Figure 34 and Figure 35. Users provided a lot of useful feedback on the CrimAnalytics web application. The feedback also gave me a better idea of how much users knew about predictive analytics. This allowed me to tailor the documentation on the about page. This questionnaire also provided me with several suggestions to improve usability.

The first of these was to prevent the embedded map on the “Place Search” map from automatically grabbing the user’s cursor as they scrolled down the page. This made the page much easier to navigate without losing any functionality of the map. Users could still interact with the marker on the map and use the dedicated zoom buttons on the map to navigate. Users also reported that the sidebar navigation was not working properly on some pages, which was fixed. A bug was reported where the title of the crime rates chart was not properly formulated when users were resubmitting changed locational feature counts. Users also reported some minor inconsistencies in the colour of the user interface in some areas which was also fixed. Another suggestion that resulted from this process was the use of text that would hover over the locational feature icons. This change helped users understand the specific land uses that were contained in each category.

5.2. Implemented Unit Tests

Testing, selection criteria and what are the key use cases within the project.

The Unit Tests in this project were implemented using the JUnit framework. These tests ensure that the functionality that the key use cases depends on does not change. These unit tests can be run before committing any code to ensure that any changes made have not negatively affected existing functionality.

```
34. @Test
35. public void testFindFeatures() {
36.     featuresHashMap = locationalService.findFeatures(lat, lng);
37.
38.     place = new Place(featuresHashMap.get("educational"), featuresHashMap.get("food"),
39.         featuresHashMap.get("government"), featuresHashMap.get("health"),
40.         featuresHashMap.get("recreational"), featuresHashMap.get("stores"), featuresHashMap.get("services"),
41.         featuresHashMap.get("transport"), featuresHashMap.get("worship"));
42.
43.     assertEquals(ZERO, place.getEducational());
44.     assertEquals(ZERO, place.getFood());
45.     assertEquals(ZERO, place.getGovernment());
46.     assertEquals(ZERO, place.getHealth());
47.     assertEquals(ZERO, place.getRecreational());
48.     assertEquals(ZERO, place.getStores());
49.     assertEquals(ZERO, place.getServices());
50.     assertEquals(ZERO, place.getTransport());
51.     assertEquals(ZERO, place.getWorship());
52. }
```

Figure 36: Locational service testing. [src/com.example.tests.LocationalServiceTest.java]

The JUnit test in Figure 36 tests the locational feature retrieval. This is a key process that is run on any place that a user submits. The test specifically tests the “LocationalService.findFeatures” method. The coordinates are passed to the method as usual. The numbers received back from the model are checked to be equal to the expected values using the “assertEquals” method provided by the JUnit framework. If all the asserts fail, an “AssertionError” is thrown and the test will fail. Otherwise the test will pass.

```
34. @Before
35. public void setup() {
36.     place = new Place(10, 10, 10, 10, 10, 10, 10, 10, 10);
37.     predictionService = new PredictionService();
38. }
39.
40. @Test
41. public void testApplyModel() {
42.     predictionService.applyModel(place);
43.
44.     assertEquals(murderAssaultHarassment, place.getMurderAssaultHarassment());
45.     assertEquals(dangerousNegligentActs, place.getDangerousNegligentActs());
46.     assertEquals(kidnapping, place.getKidnapping());
47.     assertEquals(robberyExtortionHijacking, place.getRobberyExtortionHijacking());
48.     assertEquals(burglary, place.getBurglary());
49.     assertEquals(theft, place.getTheft());
50.     assertEquals(fraudDeception, place.getFraudDeception());
51.     assertEquals(drugs, place.getDrugs());
52.     assertEquals(weaponsExplosives, place.getWeaponsExplosives());
53.     assertEquals(damagePropertyEnvironment, place.getDamagePropertyEnvironment());
54.     assertEquals(publicOrderSocialCode, place.getPublicOrderSocialCode());
55.     assertEquals(offencesAgainstGovernment, place.getOffencesAgainstGovernment());
56. }
```

Figure 37: Prediction testing. [src/com.example.tests.PredictionServiceTest.java]

The JUnit test in Figure 37 tests the crime rate prediction functionality. Once the locational features have been retrieved, this is the next key step. The locational data is passed to the model and the crime rates are returned. This test passes known values to the model and asserts that the returned values are equal to the corresponding known crime rates. This test ensures that the model is not accidentally modified, which would cause it to return incorrect crime rate predictions.

5.3. Demonstration

A video demonstration of the functionality in the CrimAnalytics web application was created for demonstration purposes. This will be of use to those who are unable to view the application in person. This video demonstrates all the functionality of the CrimAnalytics web application, including searching for places, viewing the crime rates and changing the locational features. The video shows the use of the web application on both desktop and mobile environments. The video can be viewed at the following link:

<https://www.youtube.com/watch?v=w-AMtQAtXYY>

5.4. Demonstrable Features

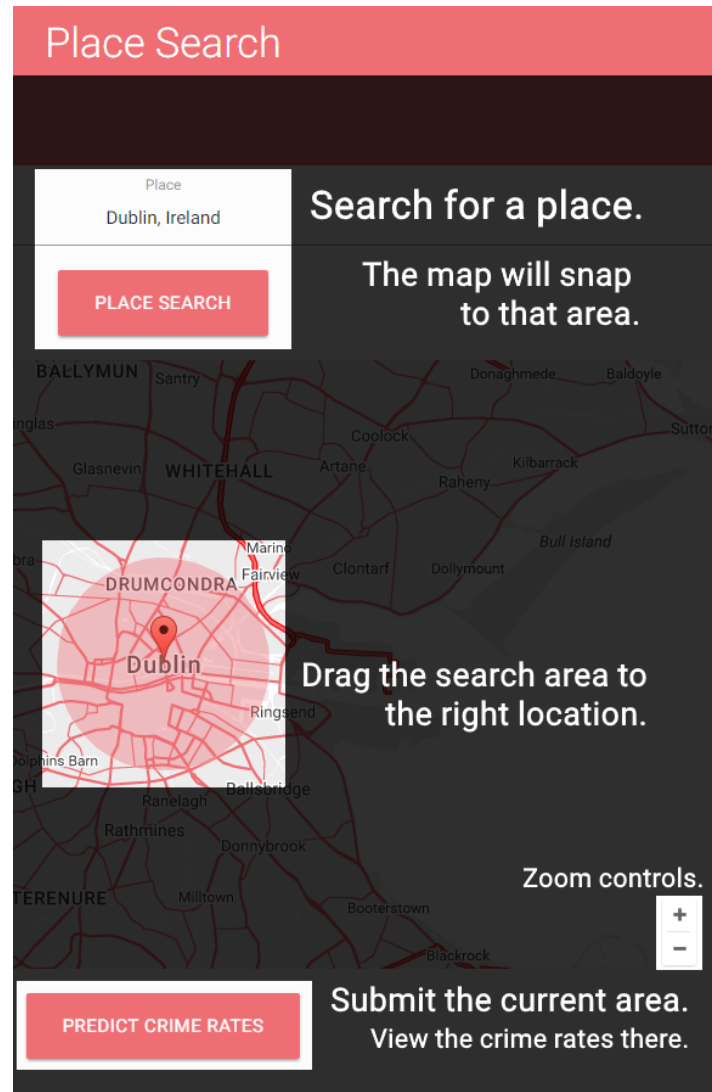


Figure 38: Place Search feature tutorial.

The place search feature can be seen in Figure 38. This feature allows users to search for a place on a map. To achieve this, the page incorporates a map that is embedded into the page. This map can be used to specify a location. The marker can be picked up by the mouse pointer and moved around the map to new locations. The area that is applicable to the crime rates is shown as a circle around the map marker. It represents the radius search that is performed to retrieve the locational features for the model. The purpose of this feature is to submit geographic coordinates to the Java back-end. There the coordinates will be used to attain the locational feature information in a two-kilometre region. The area contained within the marker radius is the same area that the crime rates reference.



Figure 39: Place Crime Rates viewing feature tutorial.

The place crime rates feature can be seen in Figure 39. This feature allows users to see the results from the predictive model as a visual chart. The chart shows the different crime categories that the model predicts for. The counts for the number of crime committed in these categories are also displayed in the chart. The chart depicts the prediction standards as defined by the model. These standards are the predictions being within two kilometres from the point and within a six-month period.

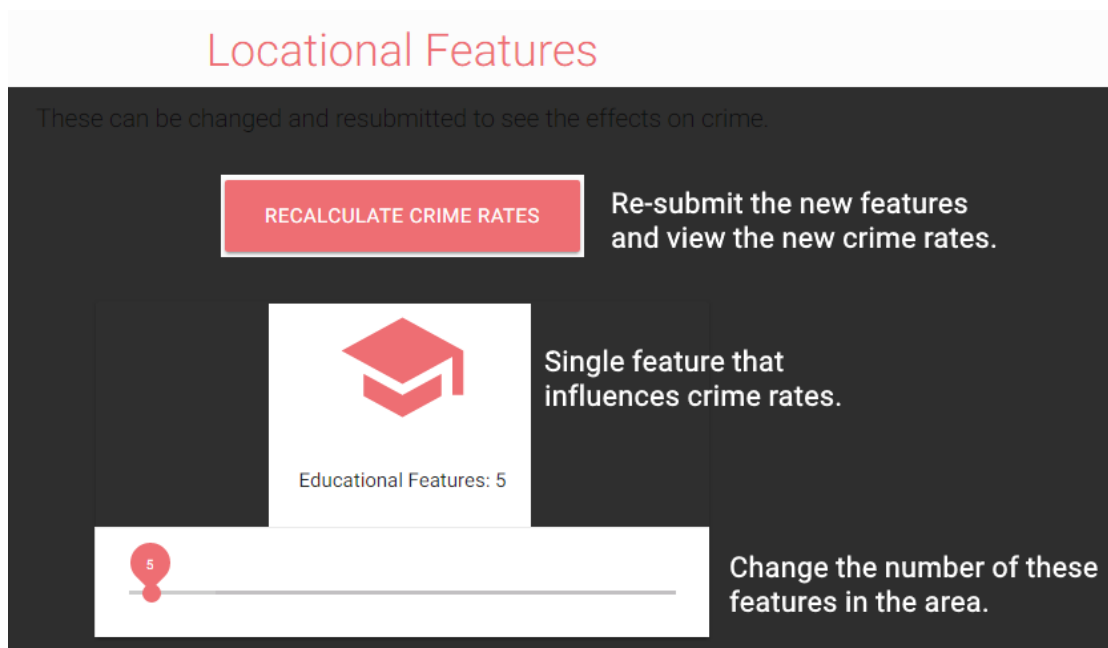


Figure 40: Place Locational Feature editing tutorial.

The locational category editing feature can be seen in Figure 40. This feature allows users to change the counts of the locational features. Once changed, they can be resubmitted to the model. The crime rates chart will then update with the newly predicted counts.

6. Project Plan

6.1. Project Plan Analysis

Fortnight Starting	Sprint	Project Work	Report Work
07/11/2016	1	Data Preparation - Remove duplicate coordinates cases - Fix unfound place names	Interim
21/11/2016	2	Collection of feature data from Places API Set up working prototype of Multiple Linear Regression in R Console	
19/12/2016	3	Use Java-to-R(JRI) to build on prototype (Connect MySQL data to R Engine functions)	
09/01/2017	4	Start implementation of UI wireframe.	
23/01/2017	5	Define data structure of data passed to the client for chart display.	
06/02/2017	6	Implement chart display	
20/02/2017	7	Implement other misc. UI features.	Final
06/03/2017	8	Code clean up Code Documentation User documentation	
20/03/2017	9	Final changes	

Figure 41: Project Plan derived from Gantt Chart and Sprint Planning.

The initial project plan can be seen in Figure 41. This plan was a combination of a Gantt Chart and Sprint Planning. The Gantt Chart's purpose was to schedule each of the two week long sprints. This helped avoid any timing conflicts with weeks that

exams were scheduled. The resulting uninterrupted sprints proved to be very useful as the project progressed. This scheduling process allowed for practical planning using time that could be properly dedicated to this project.

The other aspect of this project plan was the Sprint Planning. This was completed early in the lifespan of this project. It allowed for the definition of the major goals in each of the two-week sprints. This helped greatly as although there were many small tasks in each sprint, the main objective was clearly laid out. The time management of the small tasks was handled using the Trello issue tracking platform. This allowed for the dynamic movement of these small tasks between sprints. Small tasks were moved to subsequent sprints to ensure that the major goals were achieved in time. Repeating this process ensured that the key features were implemented on time.

6.2. Review of Changes

Most of the changes in the project plan arose as the two separate components, the model and the web application came together. The user interface of the web application was very dependent on the functionality of the model. To be able to give the front-end enough development time, the schedule for the simpler parts of the user interface were brought forward. These tasks included the implementation of application wide features such as the sidebar navigation, headers and footers of the pages. When the time originally scheduled for user interface came, I could focus on the more complex user interface features, such as the “Place Search” and “Place Crime Rates” pages. This proved very useful as I did not have as much experience in front-end web development as I did in back-end Java development. The extra time I had to focus on the key user features allowed me to fix several bugs that would have significantly decreased the usability on the user interface.

7. Conclusion

7.1. Analysis of Project Elements and Learning Outcomes

One of the key elements in this project was the building of the predictive model. During the research stage, I learnt a lot about the different analytical techniques used in Predictive Analytics. During this research, I learnt about how several predictive algorithms worked and which problems they could solve. I also learnt a lot about the data mining processes and how they enable better predictive analytics. Most of this learning came from the use of the steps in the CRISP-DM methodology. I has underestimated the usefulness of these steps for a while but when I ran into issues during the modelling stage, the benefits became clear. The ability to return to the “Data Understanding” and “Data Preparation” stages provided me with the ability to improve the accuracy of the model. This was achieved by revisiting the data on several occasions and identifying the sources of any inconsistencies. I was then able to better prepare the data that would be used to train the model. With better quality data to train on, the model’s accuracy in predictions could increase.

There is still the possibility for future work on the predictive model. The addition of more data that could explain crime would have the effect of making the model even more accurate in its predictions. The locational data that is currently being used has the benefit of making the model applicable to any location in the world. Thus, any new data would need to be available on a similar scale.

Another key element in this project was the development of the web application. As this was the first web application that I have personally developed from end to end, I have learnt a lot. As I went through the process of analysing and choosing the technologies for this project, I gained new skills in Software Architecture. One of the biggest learning opportunities here was being able to do research into the technologies available. I could analyse each component in depth during this research. Thus, I gained skills in identifying how different technologies can combine to solve a problem. Whilst many technology combinations are possible, some are better than others. The reasons behind the suitability of a combination of technologies varies and many different aspects must be considered. These aspects can range from the platforms supported to the development environments and libraries that are available. This process of researching technologies is a huge asset when defining the scope of a project. The more work that is done in this area, the easier it is to understand what is realistically possible before starting development on a project. In this project, these new skills that I obtained ensured that a minimal amount of time was spent setting up servers, development environments, APIs and libraries. These new practical skills are ones which will prove very useful for any future projects.

There are several possibilities for future work on the web application. In its current state, it acts as a tool to access the model, manipulate the input values and see the changes. Thus, there are several opportunities to expand the web application. Some examples of future work could be adding more choices for prediction visualisation and the persisting of user created crime predictions.

8. Bibliography

- Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M.,
... Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved
20 November 2016, from <http://agilemanifesto.org/>
- Byrski, A., Oplatková, Z., Carvalho, M., & Kisiel-Dorohinicki, M. (2012). *Advances in
Intelligent Modelling and Simulation: Simulation Tools and Applications*.
Springer.
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R.
(2000). CRISP-DM 1.0: Step-by-step data mining guide. *Journal of Data
Warehousing*, 5.4, 13–22.
- Chartist.js. (2014, April 6). Retrieved 16 October 2016, from
<http://gionkunz.github.io/chartist-js/examples.html>
- Chart.js. (2013, October 16). Retrieved 16 October 2016, from
<http://www.chartjs.org/docs/>
- Cox, D. R. (1958). The Regression Analysis of Binary Sequences. *Journal of the Royal
Statistical Society. Series B (Methodological)*, 20(2), 215–242.
- Crouse, W. (2015). Google Places API (Java). Retrieved 20 November 2016, from
<https://github.com/windy1/google-places-api-java>
- Docker Inc. (2015, May 14). What is Docker? Retrieved 4 April 2017, from
<https://www.docker.com/what-docker>
- Dr. Saed Sayad. (2010). Naive Bayesian. Retrieved 6 November 2016, from
http://www.saedsayad.com/naive_bayesian.htm
- draw.io. (2016). Retrieved 26 November 2016, from <https://www.draw.io/>

Fischer, R. (1922). The goodness of fit of regression formulae, and the distribution of regression coefficients. *Journal of the Royal Statistical Society*.

Google Developers. (2016, October 12). Places API Web Service. Retrieved 20 November 2016, from <https://developers.google.com/places/web-service/search>

Google Developers. (2017). Developer's Guide | Google Maps Geocoding API. Retrieved 16 October 2016, from <https://developers.google.com/maps/documentation/geocoding/intro>

Highcharts. (2011, June 21). Retrieved 16 October 2016, from <http://www.highcharts.com/demo/>

IBM Corporation. (2010). *Public safety: from 'sense and respond' to 'predict and act'*. United States of America. Retrieved from <http://bluelineplanning.com/WordPress/wp-content/uploads/2012/03/Public-Safety.pdf>

Jensen, K. (2012). *CRISP-DM Diagram*. Retrieved from https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png

JRI - Java/R Interface. (2007, October 1). Retrieved 26 November 2016, from <https://rforge.net/JRI/>

Markham, K. (2016, October 18). R2 Statistic - A measure of variance in regression. Retrieved from <https://d2gne97vdumgn3.cloudfront.net/api/file/fYGKw8lZRpit7RhMNJ0W>

Mozilla. (2017a, February 14). CSS reference. Retrieved 17 March 2017, from <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>

- Mozilla. (2017b, February 28). JavaScript. Retrieved 17 March 2017, from
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- Mozilla. (2017c, March 1). HTML. Retrieved 17 March 2017, from
<https://developer.mozilla.org/en-US/docs/Web/HTML>
- Nielsen, J. (1995). 10 Heuristics for User Interface Design. *Nielsen Norman Group*.
Retrieved from <https://www.nngroup.com/articles/ten-usability-heuristics/>
- Nucleus Research Inc. (2012). *IBM ROI Case Study: City of Lancaster, CA*. Retrieved
from <https://nucleusresearch.com/research/single/ibm-roi-case-study-city-of-lancaster-ca/>
- Oracle. (2013, January). Java Servlet Technology. Retrieved 17 March 2017, from
<http://docs.oracle.com/javaee/6/tutorial/doc/bnafdf.html>
- Pearson, K. (1905). *On the general theory of skew correlation and non-linear regression*. London, Dulau and Co. Retrieved from
<http://archive.org/details/cu31924003092917>
- PredPol. (2015). Retrieved from <http://www.predpol.com/about/>
- R: The R Project for Statistical Computing. (2016). Retrieved 26 November 2016,
from <https://www.r-project.org/>
- Sayad, S. (2010). Coefficient of Determination. Retrieved 12 November 2016, from
http://www.saedsayad.com/model_evaluation_r.htm
- Scriven, S. (2016, December 16). Recorded Crime Offences by Garda Station, Type of
Offence and Year - StatBank - data and statistics. Retrieved 11 February 2017,
from

DIT
DT228 B.Sc. (Hons.) in Computer Science
Final Report 2016/17

<http://www.cso.ie/px/pxeirestat/Statire/SelectVarVal/Define.asp?maintable=CJA07>

Shah, J. (2015, April 4). total-regression-error-sum-of-squares.png (PNG Image, 815 × 648 pixels). Retrieved 12 November 2016, from <http://www.dxbydt.com/wp-content/uploads/2015/06/total-regression-error-sum-of-squares.png>

Trello. (2016). Retrieved 27 November 2016, from <https://trello.com>