CA400 Final Year Project: Technical Guide

Predictive Analytics Toolkit

Sean Quinn 13330146

Supervisor: Mark Roantree

21/05/17

1. Introduction

1.1 Overview

This project is a predictive analytics platform which provides the user with a suite of customisable classification algorithms. This interactive application boasts a graphical user interface which provides a fast and user friendly way of configuring, optimising and running the algorithms as well as presenting the results in a graphical and easy to interpret way. Classification algorithms are at the core of predictive analytics and in order to maximise the value provided by the system I have chosen four of the most widely adopted prediction algorithms. The K-Nearest Neighbours, Decision Tree, Naive Bayes and Rocchio classification algorithms. The combination of these four algorithms along with a user friendly interface and aesthetic visual outputs provides the user with a powerful tool for making predictions based on their data.

This project addresses a number of problems; which classifier works best for my data? How do I maximise the performance of this algorithm? Which attributes in my data are the most important? Which are the most influential? By offering the user a range of algorithms and configurations, as well as the ability to perform principle component analysis by picking and choosing which attributes to use in the configurations we have addressed these issues.

1.2 Glossary

- **Data Mining:**
  Data mining is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.
- **Predictive Analytics:**
  Predictive analytics is the branch of advanced analytics which is used to make predictions about unknown future events. Predictive analytics uses techniques to analyze current data to make predictions about the nature of future data.
- **API:**
  Application Program Interface - is a set of subroutine definitions, protocols or tools for building software and applications. It is used to specify the interface between two pieces of software and manage how they interact.
- **REST:**
  REST is a type of web service where requests made to a resource's URI will elicit a response in XML, HTML, JSON or some other defined format. By making use of a stateless protocol and standard operations REST systems aim for fast performance, reliability, and the ability to grow, by using reused components that can be managed and updated without affecting the system as a whole, even while it is running.
- **GUI:**

Graphical User Interface - is a type of user interface that allows users to interact with electronic systems through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

- **K-Nearest Neighbours (KNN) Algorithm:**
  K-Nearest Neighbours is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification.
- **Decision Trees (DT) Algorithm:**
  Decision tree learning uses a decision tree as a predictive model which maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves).
- **Naïve Bayes (NB):**
  Naive Bayes classifiers are simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
- **Rocchio Algorithm:**
  Rocchio classifiers are based on the vector space model and works by calculating a single representative centroid for each class. Records are then classified by selecting the closest centroid using a particular distance formula.

2. General Description

2.1 Motivation

The motivation for this particular project came from a deep personal interest in data analytics and a curiosity with the predictive algorithms. I was very eager to get an opportunity to write my own versions of these algorithms to see how they performed and see if I would be able to develop a system with real analytical power. It seems that there is a very limited set of analytical tools available for data science and all of these tools require code to be written. For me the most enjoyable part of the project was figuring out and coding the four algorithms. The challenge I set myself was to expose these algorithms through a graphical user interface so that they can be used without writing any code but to do so in such a way as to avoid diminishing their power. I think this is something which has not been done to date.

2.2 Business Context

This system is designed as a standalone application which could potentially be used by any business or organisation which has a need to investigate data. It would prove useful for any organisation which has the aim of mining patterns and insights from data and making predictions. It would allow an organisation to make use of these complex algorithms and take advantage of their results without needing to have a data scientist or somebody with a deep understanding of data analytics in house. The application only requires that you understand your own data to a certain degree

and have a basic understanding of the algorithms involved. It will bring advanced analytics to the non-expert analyst. As such it may be of value to students or smaller organisations where there is not an in house data scientist. Although the application is designed to cater for the needs of the non expert user, a data scientist would be better able to understand the underlying algorithms and so take more advantage of the features provided by the application. So perhaps the best usage environment for the application would be as a tool for data scientists to assist them in their analysis.

2.3 Research

There has been a number of significant research elements involved in this project. Particularly in the early stages of the project. I had to gain a thorough understanding of each of the four algorithms and their strengths and limitations before I was able to even consider implementing them. I also examined several other algorithms in detail before I concluded the four I chose were the best fit for the project. In particular the fact that so many different versions of the algorithms exist in practice. I endeavoured to implement the most theoretically pure version of each algorithm and then build up configuration options on top of that. In terms of software development, there was a lot of things which were new to me and required research. This application was the first time I implemented a REST API, the first time I used the spring framework, the first time I developed a dynamic front end and the first time I developed something large in javascript. In particular working with Spring and understanding how REST works took a number of weeks of research before I was in a position to write code. Since the completion of the system I have been attempting to apply my application to datasets, gathering different types of datasets to see how well the system responds and performs.

2.4 System Functions

This system is designed to allow the user to run the algorithms with as little user effort as possible. So as such the graphical user interface strives to be as simplistic and user friendly as possible.
The home page of the application serves as an overview on the datasets which have been uploaded. Here the user can upload as many datasets as they wish in a given session. This page allows the user to upload, delete and select a dataset for analysis.
Once a dataset has been selected for analysis the user will then be brought to an algorithm configuration page, where they can select any number and combination of the algorithms to run. The page contains form validation and information icons for each tile, so the user is given the maximum help possible to configure the algorithms. There is also a copy function on each tile to increase the speed of populating the tiles. If the user fills in a single configuration and then selects copy all the other configurations will be populated with the same values, so the user only has to scan over each and make the required changes and algorithm specific selections. Once the user selects run they will be brought to the results page.

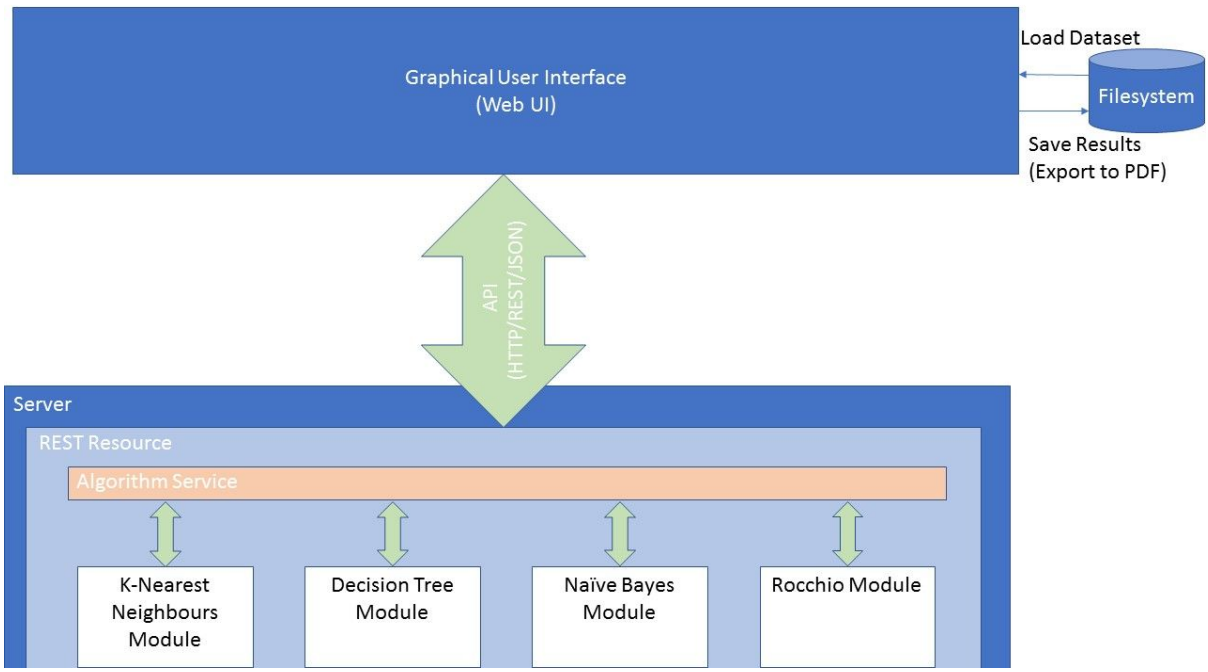
The results page displays a bar chart summary which allows the user to quickly analyse the results of their tests. The user can scroll down or click on the bars of the bar chart to see detailed information on any of the run results. This includes the configuration used, the number of tests ran and the overall percentage accuracy. The user can save the results as a PDF or print them. When the user is finished with the results they can select finish to return to the home page.

2.5 User Characteristics and Objectives

The intended user of the analytics application is anyone who wishes to conduct data mining or is involved in using predictive analytics techniques. The user will need to have some basic knowledge of data preprocessing in order to ensure the dataset provided to the application is in an appropriate format for the algorithms to run on it. The required format for the input data file is clearly specified to the user in the GUI. There will be some preprocessing such as normalisation carried out by the algorithms but other dataset specific features of preprocessing such as data cleansing or data integration will not be carried out by the application. In order to take advantage of this systems features fully the user must understand the nature of the data they are performing the analysis on. The intended user must understand the meaning of the outputs of these algorithms and they must have a problem or curiosity in relation to the data which can be satisfied by running these algorithms. This is intended to be used by someone with some knowledge of data analytics but does not require the user to be competent in programming or understand the inner workings of the algorithms.

3. Design

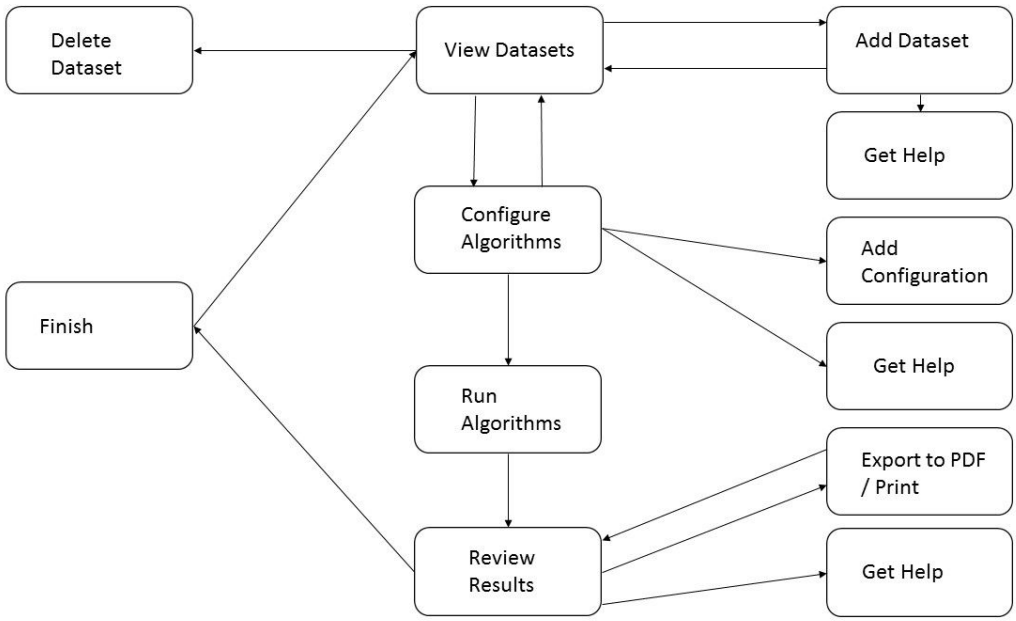
3.1 Architecture Diagram



3.2 Architecture Overview

The architecture above illustrates the structure of the application. The user interacts with the system through the browser based user interface. They upload the CSV format data sets from their local machine to the GUI. They can also save and export the results of their analysis in PDF format from the GUI to their local machine. The user interface interacts with the server through a REST API. A REST controller receives the API calls and then calls the appropriate methods in the algorithm service, which is the class responsible for orchestrating the state of the system. The algorithm service deals with processing and storing uploaded datasets, passing algorithm configurations to the correct algorithm modules and then storing the results. The algorithm modules themselves simply take the configuration parameters, run the algorithm and then pass the results back to the algorithm service.

3.3 High Level Design Diagram



3.4 High Level Design Description

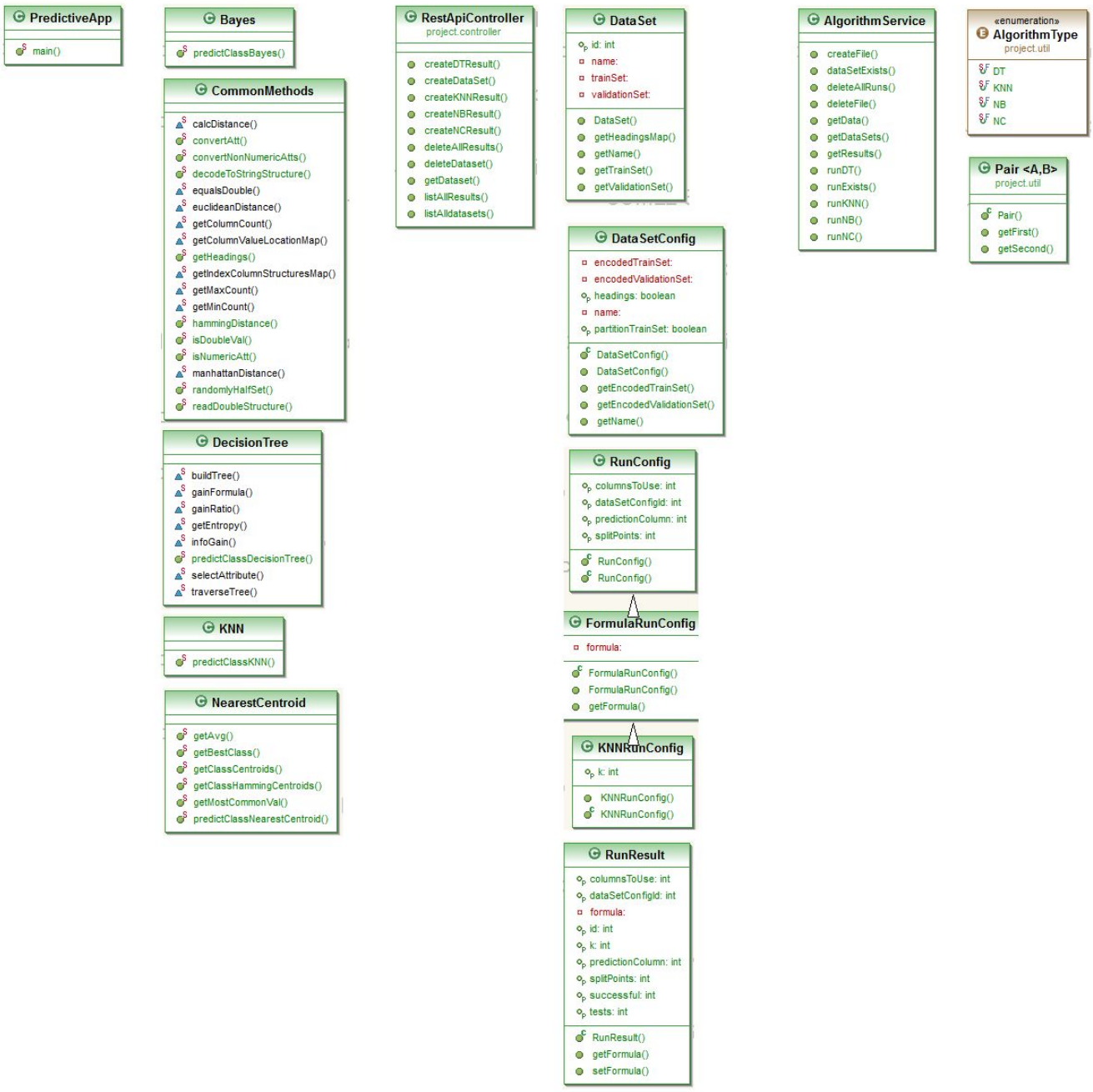
The above diagram illustrates the high level design of the systems functionality. At any point in the systems operation the user will be using one of the 12 functions above.
When the program is started the user will be brought to the datasets page. On this page the user can upload a dataset, delete a dataset or select a dataset to run analysis on. If the user selects upload dataset they will be brought to the add dataset page. From this page they can avail of a get help function. Once they have successfully uploaded a dataset or else canceled they will be brought back to the datasets page.

If the user decided to run analysis on a dataset they will then be brought to the configure algorithms page. From here they can add algorithm configurations to the current run. They can also avail of a get help function on this page. They can then either return home to the datasets page and abandon any configurations or run the algorithms.
When the user runs the algorithms they will then be brought to the review results page. From this page the user can avail of the Print/Save function and the get help function. Finally when the user is finished with the results they will finish and be brought back to the datasets page.

3.5 Object Model

The 6 columns in the below diagram correspond to the following 6 packages:

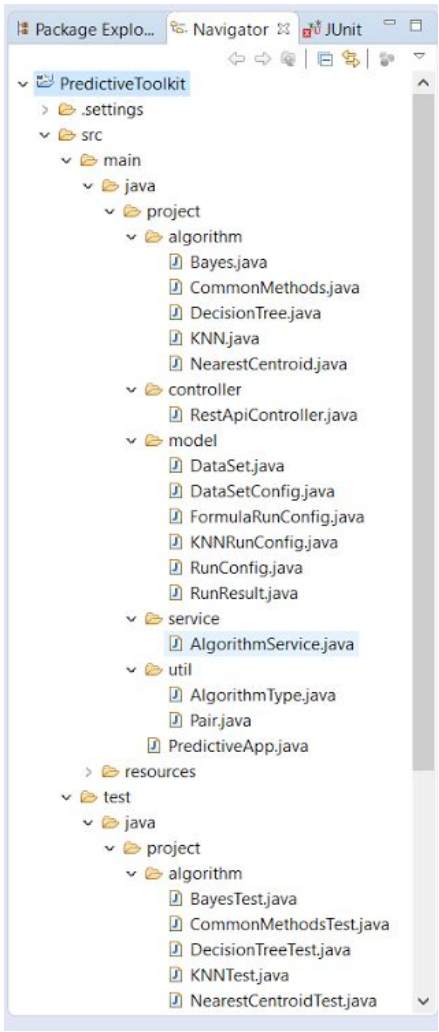
1. project
2. project.algorithm
3. project.controller
4. project.model
5. project.service
6. project.util

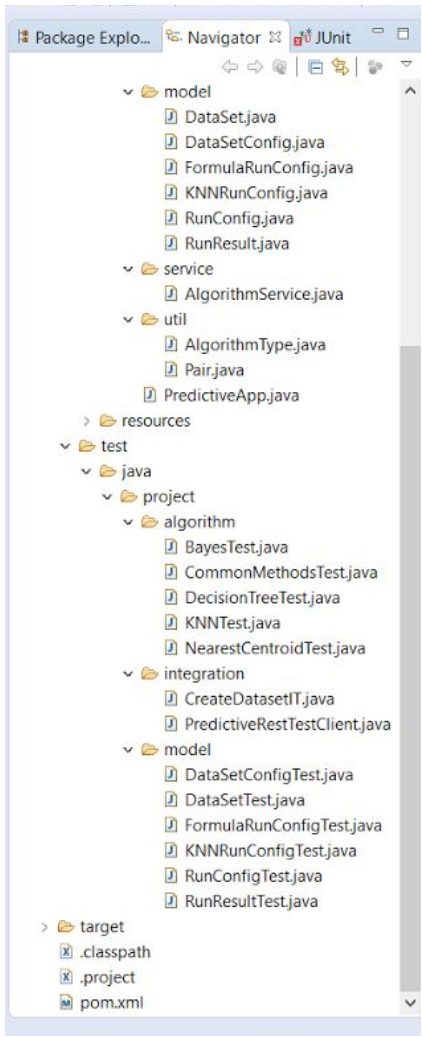


4. Implementation

4.1 Java

The entire back end of this project was completed in Java. Of all the technologies used in this project java was the only technology which I had a strong grounding in coming into the project. I chose java as it is a powerful object oriented language with a large selection of data structures. It was the perfect fit for the development of the complex algorithms. I developed each of the algorithms as independent modules, this increased the scope to add or change algorithms down the line. I also defined a “common methods” utility class which contains all the methods which are used across multiple classes. So the only methods which appear in each algorithm are the ones specific to that algorithm. The most complex code in the project was probably the code used to build each of the classifiers. In particular the recursive method used to build my decision tree classifier stood out as one which took a lot of time and effort to perfect. Below I have included screenshots of the java class structure and the build decision tree method.



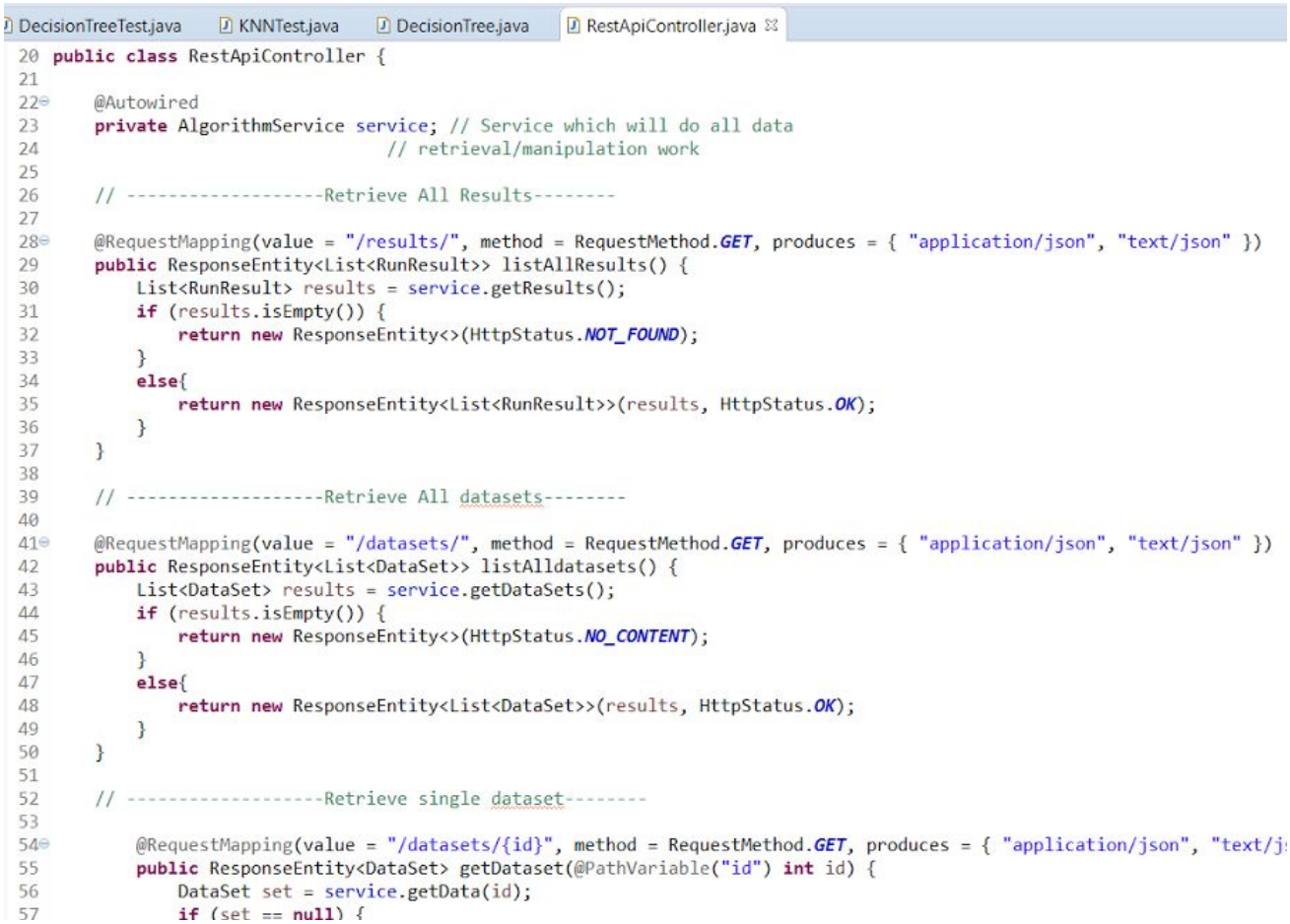



```java
static Node buildTree(Node root, String formula) {
        //check if leaf
            //1 class left - entropy will be 0
            ArrayList<ArrayList<Double>> currentColumns = root.getData();
            Double entropy = getEntropy(currentColumns, (currentColumns.size()-1));
            if(CommonMethods.equalsDouble(entropy,0.0)){
                root.setLeaf(true);
                return root;
            }

            //fully split - usedAttributes will be = to columns.size -1

            if(root.getUsedAttributes().size() == (currentColumns.size()-1)){
                root.setLeaf(true);
                return root;
            }

        //if not - split & get children and call on them

```

```
19
20      int bestAttribute = selectAttribute(currentColumns,formula, root.getUsedAttributes());
21
22      HashMap<Double,ArrayList<Integer>> columnValueIndexMap = CommonMethods.getColumnValueLocationMap(currentColumns, bestAttribute);
23
24      HashMap<Double,ArrayList<ArrayList<Double>>> indexColumnStructuresMap = CommonMethods.getIndexColumnStructuresMap(
25      CommonMethods.createRowStructure(currentColumns), columnValueIndexMap);
26
27      root.setSplitIndex(bestAttribute);
28
29      ArrayList<Integer> atts = root.getUsedAttributes();
30      atts.add(bestAttribute);
31
32      for (Double a: indexColumnStructuresMap.keySet()) {
33          Node currentChild = new Node(indexColumnStructuresMap.get(a),atts);
34          root.children.put(a,currentChild);
35      }
36
37      for(Double x: root.children.keySet()){
38          buildTree(root.children.get(x),formula);
39      }
40
41      return root;
42  }
```

4.2 Spring

I decided to use the Spring boot framework as it provides an embedded tomcat server as well as a framework for implementing REST API's. Spring simplified the process of defining the API and allowed me to get a functioning API up and running in a relatively short space of time. Using the spring framework demanded a significant refactoring and restructuring of my back end to adhere to the requirements of the framework. Once the changes were made the programs API was easily implemented. Using the Spring REST controller and service architecture insured that I followed best practice as this was the first time I authored an API. Below is a screenshot of some of the code in the REST controller class.



4.3 Bootstrap & HTML

I used the Bootstrap framework to design my front end. I used the SB Admin 2 bootstrap theme. This allowed me to achieve consistent and attractive styling on my front end without having to spend time defining stylesheets. I was then able to build upon the bootstrap components and customise them to suit my project by writing custom html. Below is a screenshot of some html from the configure algorithms page.

```
103     <hr style="border-bottom= 20px solid black;">
104
105     <div class="row">
106         <div class="col-lg-1"></div>
107         <div class="col-lg-10">
108             <!--<div class="panel panel-default">-->
109             <div class="panel-body">
110                 <div class="row">
111                     <div class="col-lg-12">
112                         <h4>Add Classifier to List</h4>
113                     </div>
114                 </div>
115                 <div class="row">
116                     <div class="col-lg-12">
117                         <div class="col-lg-3">
118                             <button type="button"
119                                 class="btn btn-primary btn-outline btn-lg btn-block"
120                                 style="border-color: purple; color: purple" id="addKNN">
121                                 <i class="fa fa-plus " style="color: White;"></i> 
122                                 K-Nearest Neighbours
123                             </button>
124                         </div>
125                         <div class="col-lg-3">
126                             <button type="button"
127                                 class="btn btn-warning btn-outline btn-lg btn-block" id="addNB">
128                                 <i class="fa fa-plus " style="color: White"></i>  Naive
129                                 Bayes
130                             </button>
131                         </div>
132                         <div class="col-lg-3">
133                             <button type="button"
134                                 class="btn btn-danger btn-outline btn-lg btn-block" id="addDT">
135                                 <i class="fa fa-plus " style="color: White"></i>  Decision
136                                 Tree
137                             </button>
138                         </div>
139                         <div class="col-lg-3">
140                             <button type="button"
```

4.4 JavaScript

The approach I took to designing the front end was to develop the static design of the website in html and bootstrap first and then add in the javascript functions to bring the page to life. I made all the API calls to the server using ajax. Spring boot uses the jackson libraries to convert the JSON objects into java objects in the back end. I used standard javascript and JQuery to create my dynamic front end. I created methods to dynamically insert and remove html, perform form validation, added popover help functions, collect values from dynamic forms and create graphs. Below is a screenshot of part of the function which displays all of the results on the results page.

```
$.ajax({
    url: "http://localhost:8080/PredictiveToolkit/api/datasets/" + dataSetConfigId
}).then(function(headData) {
    var headings = headData.headingsMap;

    $.ajax({
        url: "http://localhost:8080/PredictiveToolkit/api/results/"
    }).then(function(data) {

        var data1 = new Array();

        for (var i = 0; i < data.length; i++) {

            //split points
            var splitPoints = data[i].splitPoints;
            var splitCode = "";
            if (splitPoints != null){
                var splitString = "";
                var sArrayLength = splitPoints.length;
                for (var j = 0; j < sArrayLength -1; j++){
                    splitString += columns[j] + ", ";
                }
                splitString = splitString + splitPoints[sArrayLength.length-1];
                splitCode = '<div class="list-group-item"><i class="fa fa-gear fa-fw"></i> ClassificationAttribute Split F
            }

            //attributes
            var columns = data[i].columnsToUse;
            var columnString = "";
            var arrayLength = columns.length;
            for (var g = 0; g < arrayLength -1; g++) {
                var stringValue = getKeyByValue(headings,columns[g]);
                if((stringValue.substring(0,1) === "\"")&&(stringValue.substring((stringValue.length-1),stringValue.lengtH
                    stringValue = stringValue.substring(1,(stringValue.length-1));
                }
                columnString += stringValue + ", ";
```

5. Problems & Resolution

I encountered many problems and difficulties during the development of my application. Below I have detailed just a few of the more interesting problems I was faced with.

5.1 Handling HTTP responses

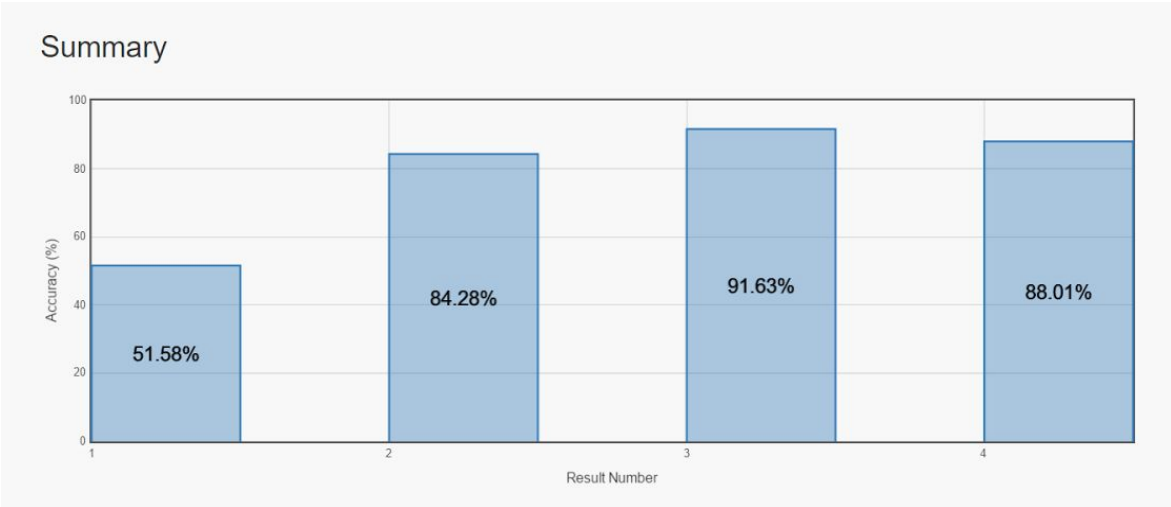
During the development of my home page I encountered an error whereby the placeholder html which displays “Add data set to begin analysis” was not rendering and I could not see any problem with the code responsible for displaying it. It turned out the problem was a http response error which was causing the javascript to halt. By design, on load of the home page I send a GET request to the datasets API to load and display the data sets on the dashboard, my logic allowed that when the called returned “undefined” it meant that there was no datasets on the server, so I display the placeholder html. But the actual http response was returning a HTTP NOT FOUND which was triggering javascripts error handler and halting the script. I eventually realised that I should have been returning a HTTP NO CONTENT instead as this is not considered an error and would not call the handler.

5.2 Popovers

Surprisingly enough, the feature of the front end that caused me the most difficulty was the popover I used for the information icons on the create dataset, configure algorithms and results pages. I found the way that popovers are added to be very different to other aspects of front end development in javascript. My original approach to this problem was to attempt to add in the popover as a dynamic html element when the user clicks the info button. After much research and hardship I eventually figured out how to get it working. The html of the popover must be placed inside the data-content attribute of the information icon and the data-toggle of the icon must be set to popover, the popover is then enabled in the javascript when the page is loaded.

5.3 Charting - Extending libraries

I found it impossible to find a library which could cater exactly to the needs of my charts. I used Morris JS to do the donut charts for each of the run results. By default morris donut charts start down the bottom for some reason, I really didn't like the look of this so I started research how it could change it to get it to start at the top and go clockwise. In the end I had to manually override the Morris.Donut.prototype.redraw function to redefine the start position at the top.
I used Flot charts to do my summary bar chart, here I encountered difficulties because by default Flot does not give you axis labels or labels on the bars of the chart. I solved this by finding two extra extension libraries online which people had developed as a solution to this; flot.axislabels.js and flot.barnumbers.js, once I had also included these I was able to define these values in the properties of the chart. Below I have a screenshot of each of the charts.



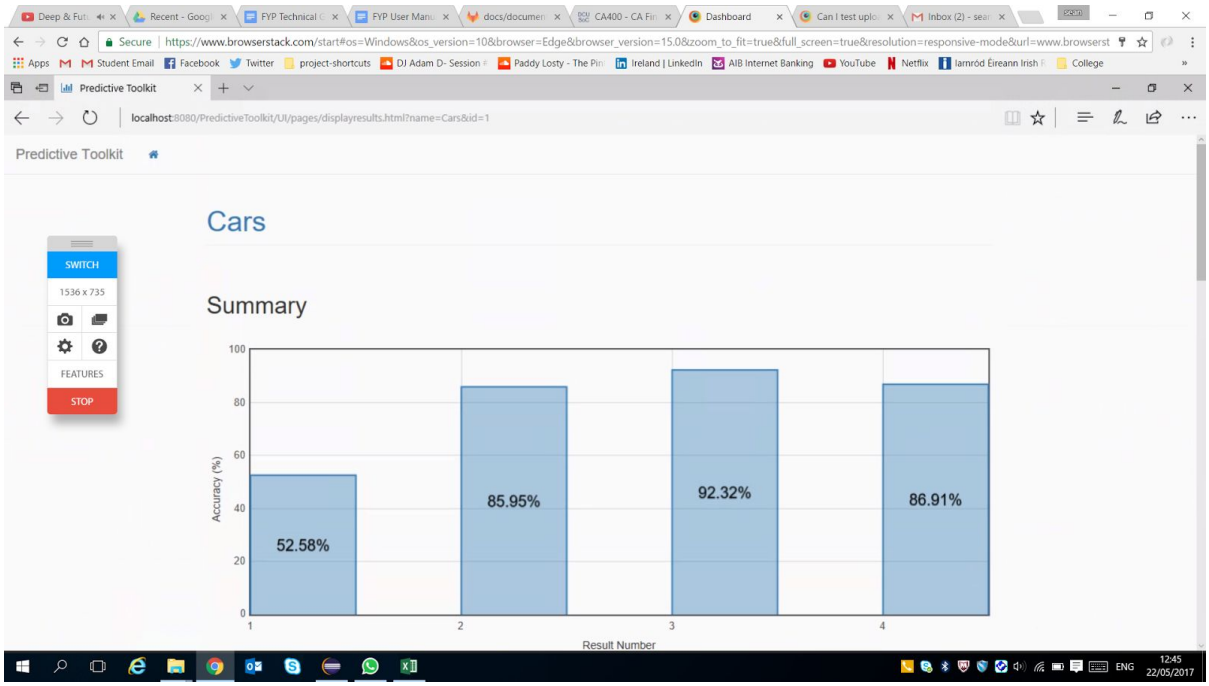
6. Validation

6.1 GUI and Browser Testing

I tested the performance and compatibility of my front end using BrowserStack. BrowserStack is a cloud-based cross-browser testing tool that enables developers to test their websites across various browsers on different operating systems and mobile devices, without requiring users to install virtual machines, devices or emulators. This gave me greater freedom to more extensively test the system. This project was designed as a desktop application so browser compatibility and consistency across the most popular internet explorer's was essential.

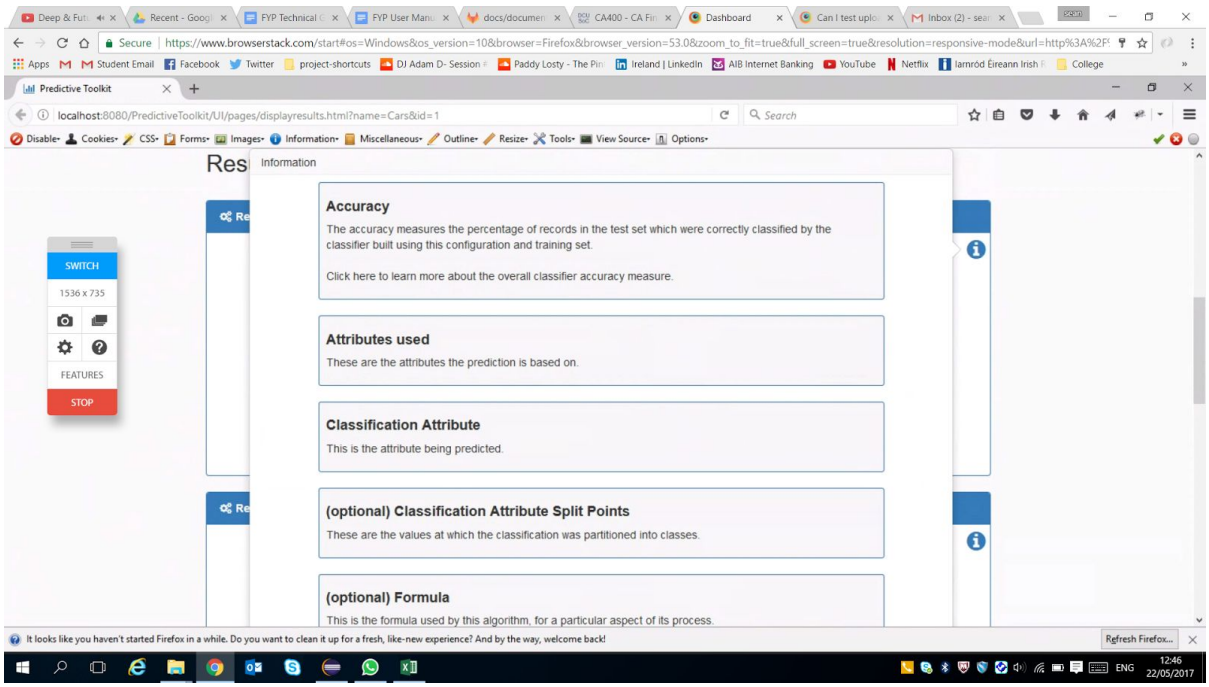
6.1.1 Edge

All aspects and features of the application performed exactly as expected in the microsoft edge browser. The application is fully compatible with this browser. Below is a screenshot of the application being tested in Edge using BrowserStack.



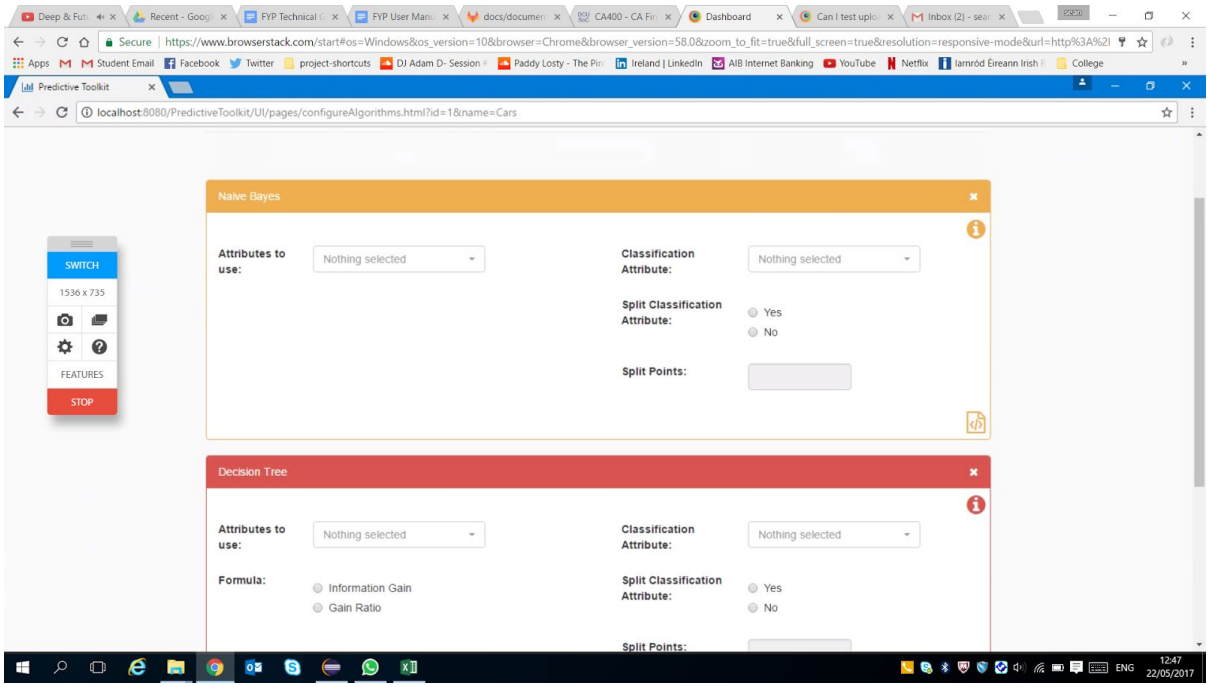
6.1.2 Firefox

All aspects and features of the application performed exactly as expected in the Mozilla Firefox browser. The application is fully compatible with this browser. Below is a screenshot of the application being tested in Firefox using BrowserStack.



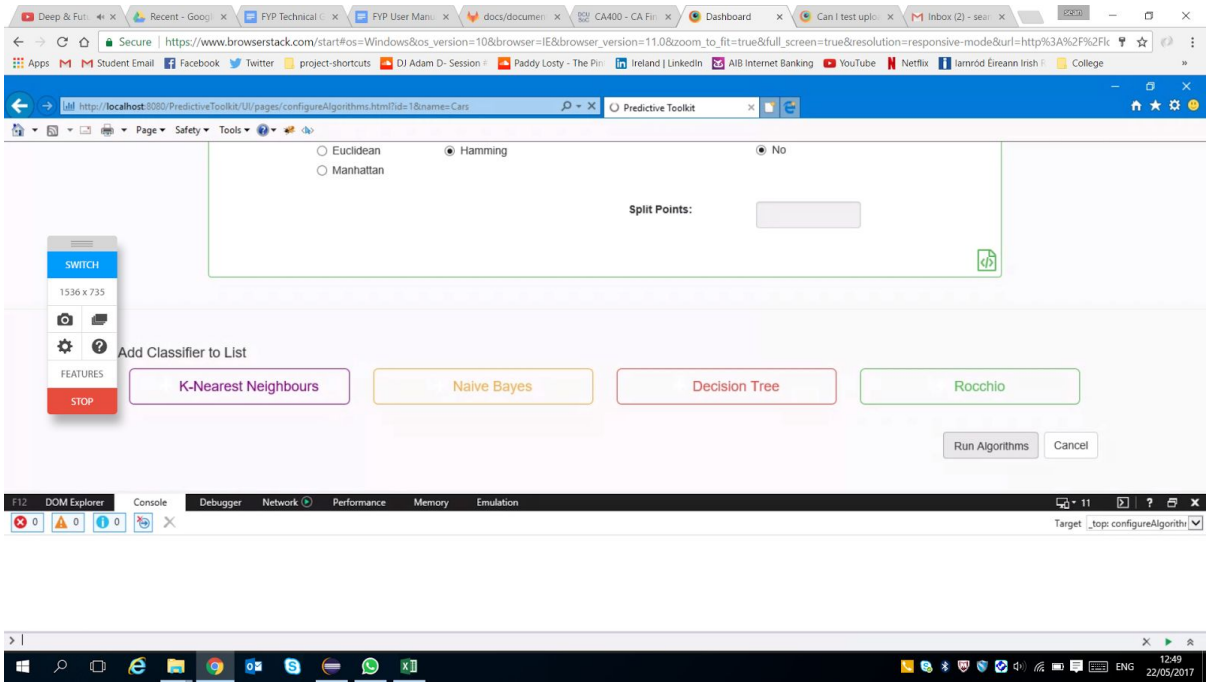
6.1.3 Chrome

All aspects and features of the application performed exactly as expected in the Google Chrome browser. The application is fully compatible with this browser. Below is a screenshot of the application being tested in Chrome using BrowserStack.



6.1.4 Internet Explorer (IE)

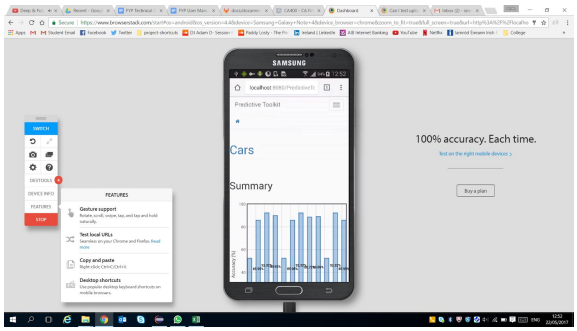
All aspects and features of the application worked without error in the Internet explorer browser. The application is fully compatible with this browser. However during the testing process I noted a significant lag in the time taken to run the algorithms, where this was near instant in other browsers it took up to 5-6 seconds in internet explorer. No errors were thrown but the delay leads me to conclude that Internet explorer is not favourable for running this application, though it could certainly be used if another browser was not available. Below is a screenshot of the application being tested in Edge using BrowserStack.



6.1.5 Mobile & Tablet

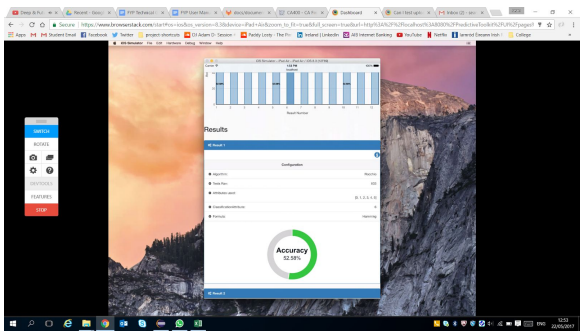
Although I designed this application to be a purely desktop based application, I also performed testing on a number of mobile and tablet devices and operating systems. Mostly just as an experiment and to see if there would be future potential to adapt this application to mobile or tablet. The dataset configuration and algorithms configuration page did not render correctly on any device. The data sets home page and the results page rendered correctly on some devices.

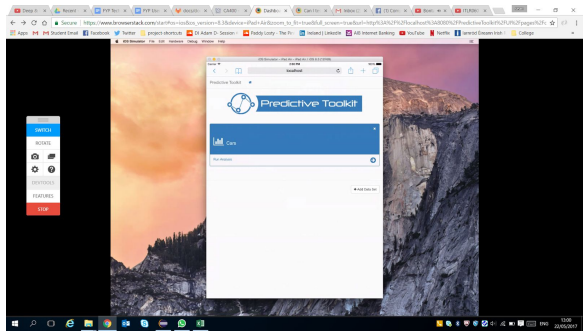
Below shows a screenshot of the application being tested on a samsung galaxy android device. None of the pages displayed in a satisfactory way.

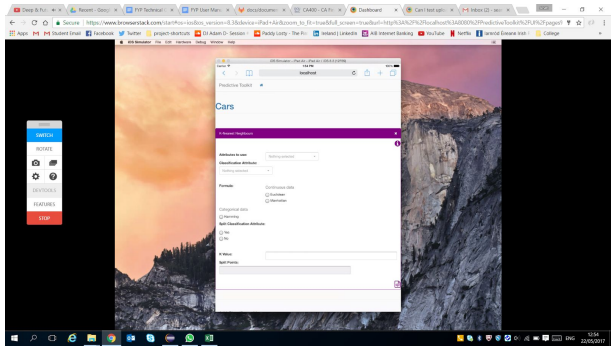


Below shows some screenshots from testing on an Ipad air device using the IOS operating system. The datasets and results page rendered satisfactorily as can be

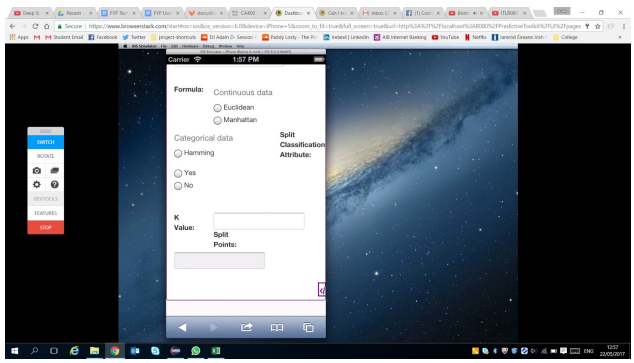
seen in the first two screenshots. The configuration pages did not render satisfactorily as can be seen in the final screenshot.



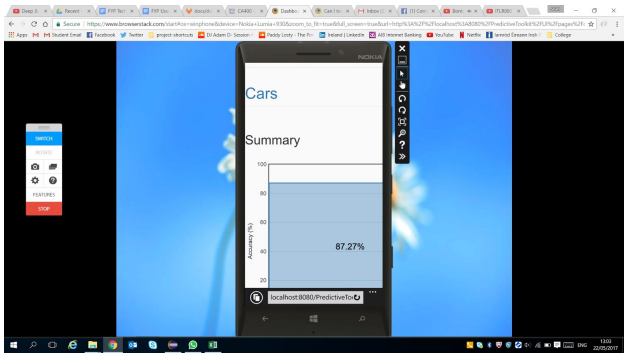




Below shows a screenshot of the application being tested on a iphone 5 IOS device. None of the pages displayed in a satisfactory way.



Below shows a screenshot of the application being tested on a windows phone. None of the pages displayed in a satisfactory way.



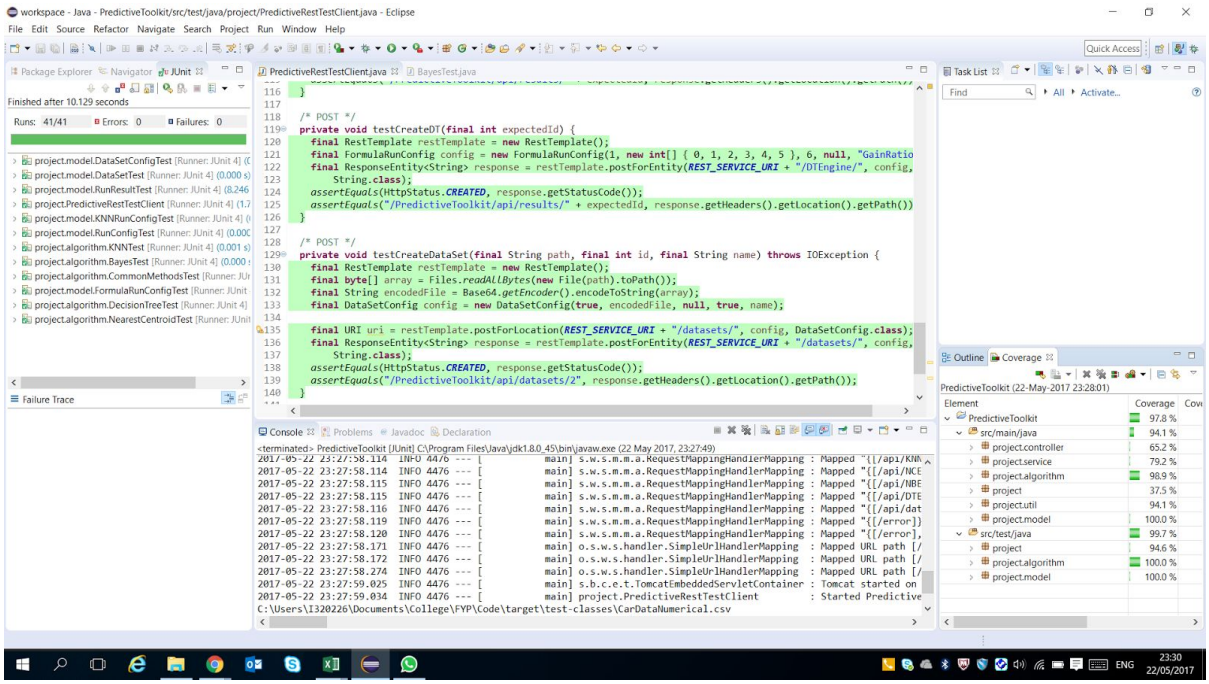
6.2 Unit & Integration testing

As this project required a very large and complex back end to be written I used a test driven development approach for the implementation of the algorithms. This meant that in most cases the test methods were written before or in conjunction with the functional methods. I developed my application using the eclipse IDE, eclipse offers a number of features which make testing a more streamlined and visible process. I used two plugins for my testing, the first is the JUnit framework, which is the standard for unit testing java code. Junit offers libraries and annotations as well as the ability to run an entire suite of tests and see visual feedback on which tests passed or failed. The second plugin I used was the EclEmma code coverage plugin. This plugin builds upon the functionality offered by the JUnit framework to offer extended reporting on the projects testing. When you run a detailed code coverage report it offers a percentage breakdown of the coverage of the application. As well as highlighting, so you can see exactly which lines of code are ran by your tests and which are not. At the conclusion of the project I have achieved an overall code coverage of 97.8%, which I am extremely satisfied with. Below I have included some screenshots of my unit testing and code coverage analysis.

Outline | Coverage

PredictiveToolkit (22-May-2017 23:28:01)

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
PredictiveToolkit	97.8 %	10,306	230	10,536
src/main/java	94.1 %	3,290	207	3,497
project.controller	65.2 %	174	93	267
project.service	79.2 %	293	77	370
project.algorithm	98.9 %	2,538	27	2,565
project	37.5 %	3	5	8
project.util	94.1 %	80	5	85
project.model	100.0 %	202	0	202
src/test/java	99.7 %	7,016	23	7,039
project	94.6 %	406	23	429
project.algorithm	100.0 %	6,022	0	6,022
project.model	100.0 %	588	0	588



6.3 Acceptance testing

I conducted extensive acceptance testing on completion of my project. This involved testing the overall functionality of the system and addressing if it met the requirements specified in the technical specification at the outset of the project. I aimed to test the application like a user would and explore all paths and functions. The acceptance testing phase was the most successful phase of testing the application in terms of the errors discovered. I tried to break the system and by doing this uncovered shortcomings in the way I handle erroneous data. During this phase I discovered and fixed design errors, front-end errors and back-end errors and then performed regression testing to ensure the changes had not disabled the project. I am now much more confident that the system is robust and stable.

6.4 Performance & Stress testing

I conducted performance testing of the algorithms. Some algorithms require significant processing so minimising the time taken to execute through efficient programing was a requirement of the system. During the main development phase of each of the algorithms I recorded the execution times for fixed size datasets. After including as many optimisations as feasible I was able to achieve an average execution speed of less than one second for each algorithm for a dataset of 1000 records. As each algorithm has a different time complexity, this execution time will

increase at different rates for each of the algorithms as the number of records in the dataset increases.
I stress tested my program in two ways; by uploading massive datasets and by attempting to run a large amount of algorithms at once. The results of this stress testing on performance were as expected, the system took longer to complete the execution of the algorithms. This is satisfactory as the system did not fail in any way and rendered good results.

6.5 External testing

In addition to my own testing I also got the opinions of a number of external testers. Before I approached any of these individuals I first applied for and was granted ethical approval for a low risk study. I made contact with Davra Networks, a Dublin based company who I had already established a relationship having used their product to complete my third year project last year. One of the user interaction designers from Davra agreed to evaluate my project for me. At the time I was in the early stages of developing my front end and he was quickly able to point out errors in the logic and flow of my program before I had gone too far developing the UI. The meeting with the Interaction designer helped me understand that there was multiple different ways I could represent the feature set I had developed into a back end. I left the meeting with a new perspective on the UI and with the understanding that simplicity needed to be the focus. I subsequently changed a large part of the design of my UI to cut out unnecessary paths through the application.

On completion of my project I met with a data scientist who works for CitiBank. Her job involves writing predictive models in R to predict the risk of individuals defaulting on loans, so she was very familiar with the algorithms used in my project and is the type of individual who would be the intended end user of this application. She was impressed with my application and in particular with the fact that I coded my own versions of the algorithms and that they perform very similar to the standard libraries she uses. She noted that this type of highly graphical application was a bit atypical for data scientists, who are more used to working with the likes of R or SPSS where the only graphical element is the resulting graphs. She did agree that my application was much faster at configuring and running a large amount of classification algorithms simultaneously than writing the code with R scripts or python.

I had significant input from my supervisor all throughout the development of this application and his advice on certain issues was invaluable.

7. Results

7.1 Future Work

There is massive scope to continue the development of this application. I had an enormous backlog of features which I wanted to implement surplus to what was delivered. In terms of the algorithms, the two main features I would have wanted to include would be; a binary split decision tree which can work with continuous data, my current implementation of the decision tree only works on categorical data, and secondly, I would also have liked to have added support for mixed data sets, where some of the attributes are categorical and some are continuous. At present my algorithms will treat the data as either one or the other. I would like to have added a support vector machine classifier as this is a widely used algorithm in industry. I would have liked to add a feature where the user could mark a result to be included in the next configuration and have an option to go straight back to the configuration page to run again. At present the only way of saving the results is to save as a PDF and then re-enter the configuration at the next run. Another architectural addition which would greatly improve the performance of this system would be the ability to execute the algorithms in parallel, if this application were ever to be deployed at scale then this requirement would increase in importance.

7.2 Conclusion

In conclusion I am very satisfied with the way the project turned out. I fulfilled the requirements I set out in my technical specification and managed to add more features on top of that. The end result has exceeded my original vision of the system. My algorithms perform satisfactorily in terms of execution times and overall accuracy in comparison to established machine learning libraries. My goal was to allow a user to efficiently and effectively use these algorithms without writing a single line of code and I feel I have achieved that goal. The development of the application has been an immensely enjoyable experience for me, I gained exposure to a wide variety of technologies and have built a solid foundation of knowledge to kick start my future in the data analytics industry.