

### ***Copyright Notice***

Staff and students of the University of the West of England are reminded that copyright subsists in this work.

**Students and staff are permitted to view and browse the electronic copy.**

**Please note that students and staff may not:**

- **Copy and paste material from the dissertation/project**
- **Print out and/or save a copy of the dissertation/project**
- **Redistribute (including by e-mail), republish or reformat anything contained within the dissertation/project**

The author (which term includes artists and other visual creators) has moral rights in the work and neither staff nor students may cause, or permit, the distortion, mutilation or other modification of the work, or any other derogatory treatment of it, which would be prejudicial to the honour or reputation of the author.

This dissertation/project will be deleted at the end of the agreed retention period (normally 5 years). If requested the Library will delete any dissertation/project before the agreed period has expired. Requests may be made by members of staff, the author or any other interested party. Deletion requests should be forwarded to: Digital Collections, UWE Library services, e-mail [Digital.Collections@uwe.ac.uk](mailto:Digital.Collections@uwe.ac.uk).

# Exploring How Machine Learning Could Be Used to Predict Customer Churn for Cloud Service Providers to Improve Customer Relationship Management

Thomas Duckett

University West of England

15006781

UFCFXK-30-3

Digital Systems Project

## Abstract

The process of a customer leaving one company to join another is called churn (Shrikhande and Verma, 2018). Since it is generally more profitable to satisfy existing customers than to find new customers, there is a desire to improve customer retention. Identifying customers that are likely to churn is not a simple task and requires a large amount of data analysis. Research will be conducted on how machine learning may be able to help classify these customers.

An application has been designed and developed using the information collected during the research, taking into consideration the gaps identified in the literature and the further work suggested by some authors. The applications sole purpose is to allow users to experiment with a variety of machine learning algorithms and compare the results. This study uses a Cloud Service Providers customer dataset.

## Acknowledgements

I would firstly like to thank my supervisor, Dr. Phil Legg, for reviewing my work and guiding me through the project. Phil has offered me valuable feedback and advice during every stage and provided new insights into aspects of the project that had not been previously considered.

I would also like to thank the [REDACTED] for employing me and working with me during the project to provide a real customer dataset that had not yet been used for research in customer churn prediction.

Finally, I would like to thank the UI testers for their time and feedback.

# Table of Contents

Abstract.....	2
Acknowledgements.....	3
Table of Contents.....	4
Table of Figures .....	7
Table of Tables .....	8
1 Introduction.....	9
2 Literature Review.....	9
2.1 Previous Work .....	11
2.2 Identified Gaps.....	13
2.3 Project Scope .....	14
3 Requirements .....	14
3.1 Functional .....	15
3.2 Non-Functional .....	16
4 Methodology .....	16
4.1.1 Planning .....	17
5 Design .....	18
5.1 System Architecture.....	18
5.1.1 Tools .....	18
5.1.2 Architecture.....	19
5.1.3 User Stories.....	19
5.1.4 Use Case Diagram.....	21
5.1.5 State Diagram.....	22
5.2 User Interface.....	23
5.2.1 Tools .....	23
5.2.2 Layout .....	23
5.2.3 Colours.....	24
5.2.4 Icons.....	24
5.2.5 Charts .....	24
5.3 Database.....	25
6 Implementation .....	26
6.1 User Interface (Sprint 1) .....	26
6.1.1 Languages .....	26
6.1.2 Layout .....	28
6.1.3 Icons.....	29

6.1.4	Charts .....	29
6.1.5	Navigation.....	30
6.1.6	Accessibility.....	30
6.1.7	Error Handling .....	30
6.1.8	Testing.....	31
6.1.9	Requirements Satisfaction.....	32
6.2	Backend (Sprint 2) .....	32
6.2.1	Languages .....	32
6.3	Database (Sprint 2) .....	33
6.3.1	Data Imports.....	33
6.4	Security .....	33
6.4.1	Secure Login .....	33
6.4.2	User Sessions .....	34
6.4.3	Private Routing .....	34
6.4.4	SQL Injection.....	34
6.4.5	Testing.....	34
6.4.6	Requirements Satisfaction.....	34
6.5	Machine Learning (Sprint 3).....	35
6.5.1	Languages .....	35
6.5.2	Data Processing.....	35
6.5.3	Models.....	37
6.5.4	Features.....	41
6.5.5	Parameters.....	41
6.5.6	Testing.....	42
6.5.7	Requirements Satisfaction.....	42
6.6	Software Testing (Sprint 4).....	43
6.6.1	Formal Test Cases.....	43
6.6.2	Friendly User Testing .....	43
6.6.3	Real World Outcome Validation.....	44
6.6.4	Reflection on Test Results .....	45
7	Project Evaluation.....	45
7.1	Limitations .....	45
7.2	Improvements .....	46
7.3	Reflection.....	47
7.4	Feedback .....	47
8	Conclusion .....	49
8.1	Further Work.....	50

9	References.....	51
9.1	Literature.....	51
9.2	Software and Libraries.....	55
10	Appendix 1: Test Cases .....	57
10.1	User Actions.....	57
10.2	Navigation.....	59
10.3	Generating Predictions.....	60
10.4	Viewing Results .....	62
10.5	Admin .....	63
10.6	Notifications.....	65

## Table of Figures

Figure 1: Gantt Chart. ....	17
Figure 2: System Architecture. ....	19
Figure 3: Use case diagram. ....	21
Figure 4: State diagram. ....	23
Figure 5: Initial user interface design. ....	24
Figure 6: Entity relationship diagram. ....	25
Figure 7: An example parent React.js component. ....	27
Figure 8: An example child React.js component. ....	28
Figure 9: User Interface on desktop (left) and mobile (right). ....	29
Figure 10: Example error banner. ....	30
Figure 11: Example chart with new colours. ....	31
Figure 12: Routing components. ....	32
Figure 13: Example salt and hash. ....	33
Figure 14: Data processing. ....	36
Figure 15: Features data frame. ....	36
Figure 16: Features formatted for the algorithms (numpy array). ....	37
Figure 17: Splitting the data into training and testing sets. ....	37
Figure 18: Model selection. ....	38
Figure 19: Fitting a Decision Tree and making predictions (Scikit-Learn). ....	38
Figure 20: Fitting and Random forest and making predictions (Scikit-Learn). ....	39
Figure 21: Neural network feature scaling. ....	40
Figure 22: Fitting a Neural Network and making predictions (Keras). ....	40
Figure 23: Feature selection. ....	41
Figure 24: Parameter adjustments (Random Forest). ....	42
Figure 25: First state diagram. ....	48
Figure 26: Old and new chart colours. ....	49



## Table of Tables

Table 1: Previously used dataset properties. ....	10
Table 2: Comparison of some previous results. ....	13
Table 3: Functional requirements. ....	15
Table 4: Non-functional requirements. ....	16
Table 5: User Stories.....	20
Table 6: Admin Stories. ....	20
Table 7: Table names. ....	26

# 1 Introduction

The process of a customer leaving one company to join another is called churn (Shrikhande and Verma, 2018). It is generally more profitable to retain and satisfy existing customers than it is to attract new customers who can be characterised by having a high attrition risk (Reinartz and Kumar, 2003) (Amin et al. 2003). Customer retention is also important if companies want to gain a strategic advantage and survive in an increasingly competitive business environment (Clark, 1997). Reichheld and Kenny (1990) measured the impact of customer retention over 24 service businesses and concluded that an improvement in retention can lead to profit swings of 25% to 80%.

Customer churn prediction is the practice of assigning a probability of churn to each customer to rank them from most likely to churn, to least likely to churn (Coussement, Lessmann, and Verstraeten, 2017) (Gordini and Veglio, 2017). The general goal is to discover hidden patterns and associations in customer behaviour. The most likely to churn would receive marketing campaigns or incentives to continue business with the company (Coussement, Lessmann, and Verstraeten, 2017) (ÓSkarsdóttir et al. 2017).

It has been confirmed that revenue and reputation can be improved by implementing customer churn prediction (ÓSkarsdóttir et al. 2016). While it can take years to realise the full potential of improving retention, there have been cases of visible improvement within the first six months of implementing a retention strategy (Reichheld and Kenny 1990).

For this project, I consider research in the area of predicting customer churn, how this can be predicted in different industries, and how machine learning can potentially be applied to this challenge in an example of a Cloud Service Provider. I explore the results of previous research that have used a variety of models across different datasets to understand how machine learning techniques have been applied for assessing customer churn. Ultimately, I ask to what extent is it possible to predict customer churn for a Cloud Service Provider?

# 2 Literature Review

Previous research has attempted to address churn prediction for a variety of different datasets. Many examples in the literature have addressed the topic of Telecommunication (Shrikhande and Verma, 2018) (Dala et al. 2018) (Coussement, Lessmann, and Verstraeten, 2017) (Umayaparvathi and Iyakutti, 2016) due to wider availability of public datasets. Other domains

include Business-to-Business e-commerce (Gordini and Veglio, 2017), newspaper publishing (Coussement, Benoit, and Van Den Poel, 2010), pay-tv (Burez and Van Den Poel, 2007), insurance providers (Soeini and Rodpysh, 2012), financial services (Van Den Poel and Larivière, 2004), online gaming (Kawale, Pal, and Srivastava, 2009), and banking (Devi Prasad and Madhavi, 2012). Datasets used in previous literature show great variance in the percentage of churned customers. Some are compared in Table 1.

<b>Literature</b>	<b>Number of Customers</b>	<b>Number Churned</b>	<b>Percentage Churned</b>
Shrikhande and Verma, 2018 Dala et al. 2018 Umayaparvathi and Iyakutti, 2016	3,333	483	14.5%
(De Caigny, Coussement, and De Bock, 2018)	3,827	~1077	28.14%
Umayaparvathi and Iyakutti, 2016	24,000	3,150	13%
Umayaparvathi and Iyakutti, 2016	50,000	3,672	7.5%
Umayaparvathi and Iyakutti, 2016	70,831	20,505	28.9%
(De Caigny, Coussement, and De Bock, 2018)	427,833	~ 47,660	11.14%
(De Caigny, Coussement, and De Bock, 2018)	631,627	~ 15,980	2.53%

*Table 1: Previously used dataset properties.*

The ‘class imbalance’ problem occurs if the class variable is heavily skewed (Japkowicz, 2000). This is a common problem for customer churn as the number of non-churners is usually much greater than churners (De Caigny, Coussement, and De Bock, 2018) (Burez and Van Den Poel, 2009), as shown in Table 1. This will be taken into consideration when analysing the dataset used in this study.

The class imbalance problem is often apparent in research around malware detection, since the malicious files will be fewer in number than benign ones in a real-world system (Markel and Bilzor, 2014). (Miao et al. 2016) introduce a non-standard machine learning technique of a one-class classification support vector machine, to explore the problem of malware detection with class imbalance. This technique is used to build a classification model in a situation where a class is poorly sampled or even absent (Miao et al. 2016). One-class classification may be useful for datasets where there are very few churned customers in comparison to non-churn.

As stated by (Huang, Tahar Kechadi, and Buckley, 2012), robust features may not be found in a single dataset but instead will require multiple tables or files to be joined. Generally, the features used for the Telecommunication industry include customer demographics, contractual data, customer service logs, call details, complaint data, billing data, and payment information (Huang, Tahar Kechadi, and Buckley, 2012) (Hadden et al. 2008) (Hung, Yen, and Wang, 2006) (Hadden et al. 2007) (Wei and Chiu, 2002). This illustrates the complexity of the problem as there are many factors to consider when trying to identify what may cause a customer to churn.

## 2.1 Previous Work

Models can either be binary (churn or not churn) or probabilistic classifiers (probability of churning) (Burez and Van Den Poel, 2009). Probabilistic classifiers are the most useful, as when combined with other customer metrics allow for better prioritisation. It has been demonstrated that Logistic Regression and Decision Tree classifiers are a good starting point for companies that wish to begin predictive modelling of customer churn (Neslin et al. 2006).

Logistic Regression has been a popular approach (Coussement, Lessmann, and Verstraeten, 2017) (De Caigny, Coussement, and De Bock, 2018) (Owczarczuk, 2010) (Huang, Tahar Kechadi, and Buckley, 2012) and has provided good and robust results in the past (Neslin et al. 2006) (Buckinx and Van Den Poel, 2005) but has said to oversimplify the real relationships in the data (Allison, 1999). Accuracies of 80.02% (Dala et al. 2018) and 87.16% (Guo-En and Wei-Dong, 2008) have been achieved on different datasets. A simple Logistic Regression model was comparable with more advanced single model algorithms (Coussement, Lessmann, and Verstraeten, 2017) but has been outperformed by Generalised Additive Models in the past (Coussement, Benoit, and Van Den Poel, 2010). Burez and Van Den Poel (2009) compared their Logistic Regression model to a Random Forest and found they would outperform each other in different cases.

Decision Trees have achieved from 83.86% (Guo-En and Wei-Dong, 2008) to 94.08% (Dala et al. 2018) accuracy, but when combined with Logistic Regression, the performance improved to be comparable to Random Forests (Coussement, Lessmann, and Verstraeten, 2017).

Support Vector Machines (SVMs) have also performed well with accuracies of 88.56% (Shrikhande and Verma, 2018), and 89.67% when based on the ‘Area Under the Curve’ (AUC) parameter selection technique (Gordini and Veglio, 2017). Guo-En and Wei-Dong (2008) found their SVM could perform better than Artificial Neural Networks, Decision Trees, Logistic Regression, and Naïve Bayes when their model achieved an accuracy of 90.88% and the others achieved 89.83%, 83.86%, 87.15%, and 87.82% respectively.

(Amin et al. 2003) have experimented with Rough Set Theory and Genetic Algorithms. Their Genetic Algorithm achieved an incredible overall accuracy of 98.2% with minimum misclassification error on their dataset.

Tsai and Lu (2009) used Self-Ordering Maps (SOM) combined with back-propagation Artificial Neural Networks (ANNs). Their baseline ANN gave 92.80% accuracy but when combined with another ANN it achieved 94.32% accuracy, and gave 93.06% with SOM. The ANN + ANN model proved to be more stable than the other two models (Tsai and Lu, 2009). (Ammar and Maheswari, 2017) also created a hybrid model using Fire Fly (FFA) and Particle Swarm (PSO) algorithms. The hybrid provided more accurate churn predictions than their FFA and PSO algorithms with 1.41% and 17% higher accuracies respectively.

<b>Literature</b>	<b>Model</b>	<b>Accuracy</b>
Dala et al. 2018	Logistic Regression	80.02
	Decision Tree	94.08%
Gordini and Veglio, 2017	Support Vector Machine (AUC parameter selection)	89.67%
	Artificial Neural Network	85.1%
	Logistic Regression	83.8%
Guo-En and Wei-Dong, 2008	Support Vector Machine	90.88%
	Artificial Neural Network	89.83%
	Decision Tree	83.86%
	Logistic Regression	87.16%
	Naïve Bayes	87.82%
Tsai and Lu, 2009	Artificial Neural Network	92.80%
	Hybrid Artificial Neural Networks	94.32%

	Hybrid Self-Ordering Map Neural Network	93.06%
(Ammar and Maheswari, 2017)	Fire Fly Algorithm	~88%
	Particle Swarm Optimisation	~86.59%
	Hybrid Fire Fly and Particle Swarm Optimisation	~71%

*Table 2: Comparison of some previous results.*

Although there are many ways to measure the performance of a predictive model, overall accuracy is great for a general comparison. As shown in Table 2, results for one algorithm can vary. This could be due to the quality of the dataset, the available features, or the model parameters. The best algorithm for a given dataset may also change depending on the algorithms ‘staying power’ (Neslin et al. 2006). A tool to easily compare machine learning techniques on a given dataset would be most useful.

## 2.2 Identified Gaps

Literature shows that no consensus exists on which model is the most accurate for churn prediction (Gordini and Veglio, 2017) and it has been discussed that models get old quickly and require constant updates (Owczarczuk, 2010). The results of different models are also often conflicting (Gordini and Veglio, 2017) and selecting the parameters of the algorithm and choosing the strategy is also a problem for customer churn prediction (Dala et al. 2018).

Further research has been desired in comparing other models and their results (Coussement, Benoit, and Van Den Poel, 2010) (Amin et al. 2019); particularly logistic regression, decision trees, neural networks, and random forests (Coussement, Benoit and Van Den Poel, 2010), and that the comparison takes place on datasets from new domains (Tsai and Lu, 2009). It has been suggested that profiles of each customer should be generated as it would be useful for organisations to base decisions on (Amin et al. 2003).

With this information in mind, it would be desirable to have a system that would allow you to try multiple techniques and compare the results.

## 2.3 Project Scope

As there has been a desire to try churn prediction with data from new domains (Tsai and Lu, 2009), real data from a Cloud Service Provider (CSP) will be used. Due to business confidentiality and privacy, there are no public CSP data sets available, so an agreement has been made with a CSP to use their anonymised data to conduct this further research.

This CSP often engages in streams of acquisitions and is therefore considered a ‘serial acquirer’ (Fuller, Netter, and Stegemoller, 2002). Since customer retention is used as a performance indicator when looking at long-term acquisition performance (Zollo and Meier, 2008) it is important for serial acquirers to minimise churn. Company value will also be improved if acquired customers are retained (Degbey, 2015). Therefore, any work to improve retention for this CSP could be expected to have a positive impact on business.

A system will be designed and developed with the previously mentioned findings in mind. The requirements will consider that there is no consensus on the best model to use for churn prediction (Gordini and Veglio, 2017), there is a desire for a further comparison of models (Coussement, Benoit and Van Den Poel, 2010) (Amin et al. 2019), that feature and parameter selection is important (Dala et al. 2018), and that a profile for each customer would be useful (Amin et al. 2003).

## 3 Requirements

10 functional and 10 non-functional requirements are listed in the next sections. These are shaped around the gaps identified during research and use the MoSCoW method for prioritisation. MoSCoW can be defined as:

- Must: These requirements must be satisfied and implemented to consider the project a success.
- Should: These features are not essential but would provide great value to the project if implemented.
- Could: These requirements could be included but are not essential.
- Would/Won't: These requirements are not included but may be implemented in the future.

### 3.1 Functional

Requirement	Priority (MoSCoW)	Description
<b>F1</b>	MUST	The system must allow the user to make a prediction of customer churn, based on the selection of different machine learning algorithms and tuning parameters.
<b>F2</b>	MUST	The system must allow users to choose from a selection of different dataset features.
<b>F3</b>	MUST	The system must provide a profile for each customer that displays information about their behavior and results from a chosen prediction, including the confidence that the customer is won and the contributions for that customer.
<b>F4</b>	MUST	The system must restrict access in the form of a secure login feature.
<b>F5</b>	SHOULD	The system should allow users to update their profile and change their password.
<b>F6</b>	SHOULD	The system should train algorithms on a dedicated server.
<b>F7</b>	SHOULD	The system should allow for different user roles, where admins can manage other users and access to the system.
<b>F8</b>	COULD	The system could provide help to users in the form of questions and answers.
<b>F9</b>	COULD	The system could notify users when model training is complete.
<b>F10</b>	WOULD	The system would be able to communicate with well-known off-the-shelf customer management systems to reduce implementation time for other organizations.

*Table 3: Functional requirements.*



### 3.2 Non-Functional

Requirement	Priority (MoSCoW)	Description
NF1	MUST	The system must have a user interface that is easy to interpret and navigate.
NF2	MUST	The system must be scalable and allow for new models, features and parameters to be added over time.
NF3	MUST	The system must be easy to maintain.
NF4	MUST	The system must be accessible by multiple users at any one time.
NF5	SHOULD	The system should be accessible on a range of devices.
NF6	SHOULD	The system should retrieve data quickly.
NF7	SHOULD	The system should present data in chart or tabular formats.
NF8	COULD	The system could have a maximum response time of 1 second
NF9	WOULD	The system would not be down for more than 1 minute during peak time and 5 minutes during off peak time.
NF10	WOULD	The system would be extensively documented.

*Table 4: Non-functional requirements.*

## 4 Methodology

I followed the Agile methodology for the implementation of this software. I decided on an Agile approach as I knew I was likely to come across unforeseen problems and receive feedback during the development which would encourage changes.

I decided to use four-week sprints to divide the project into smaller milestones. I adapted a test first approach for each sprint but instead of using an automated unit test framework (Somerville, 2016) I took a manual approach. By writing the tests first, I gave myself a clear understanding of the desired outcome of the code before development. At the end of each sprint I would test the new components and functionality against the test cases.

I continuously refactored the code during development in order to keep it simple and maintainable as described by Somerville (2016). After each sprint I would attempt to tidy up each component and move any duplicate code into its own component or function.

#### 4.1.1 Planning

I used TeamGantt (2018) to create a Gantt chart for my project. I would modify the Gantt Chart during the development of the application as requirements changed and new ideas or obstacles became apparent. Figure 1 is the complete Gantt Chart after development.

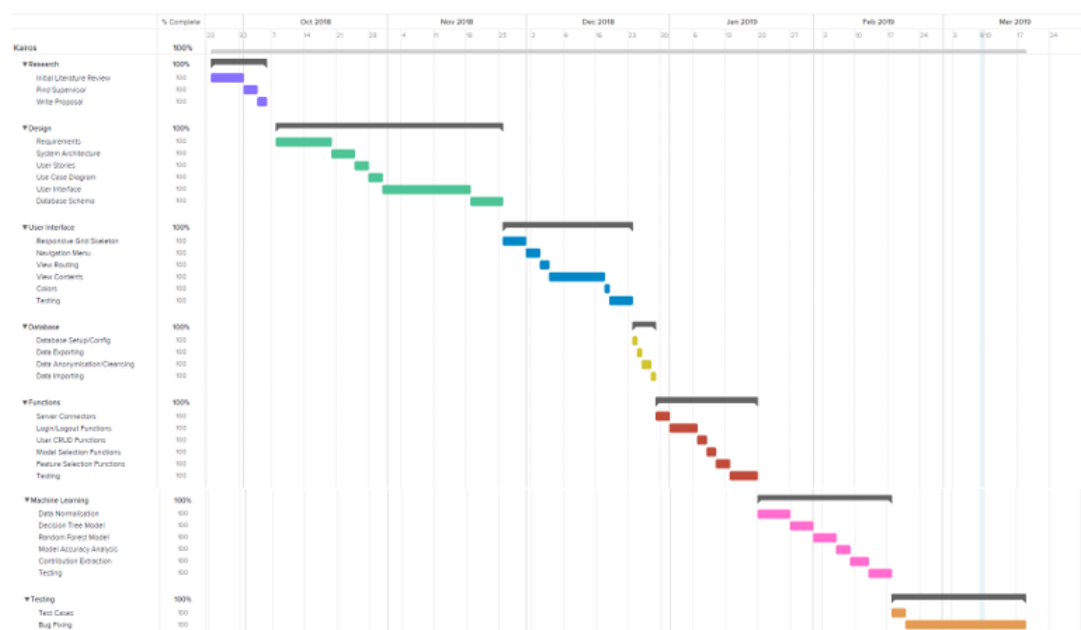


Figure 1: Gantt Chart.

##### 4.1.1.1 Sprint 1: Development of the User Interface

The first task of this sprint is to create a fully responsive skeleton. Once the skeleton containers have been created, other components can be added to them without needing to style each of them for both desktop and mobile views.

The first components will be the navigation menus. Once these are in place, I can set up the view routing and necessary security. The contents of each view will be implemented afterwards and finally the colours will be added.

The user interface will then be tested against the test cases to ensure all views and components display correctly for a range of viewport sizes.

#### 4.1.1.2 *Sprint 2: Configuring the Database*

In this sprint I set up and configure the database. I also export, cleanse and anonymise the data before finally importing it. This process is not very lengthy as the database is supposed to represent that of the organisations as closely as possible. Any data processing required before training the different machine learning models will be carried out by the application.

The remaining time is allocated to writing the functions for querying the database. These include the creation of user profiles and the ability to update them, and the ability to select models and features.

I have allocated some time at the end of the sprint to test the functions against their test cases.

#### 4.1.1.3 *Sprint 3: Machine Learning*

All of the machine learning will be introduced during this sprint. I'll add some models and the ability to return model accuracies and extract the contribution data for each customer. All of the results and contributions will be stored in the database and displayed using the graphs and charts on the user interface.

#### 4.1.1.4 *Sprint 4: Testing*

This final sprint is dedicated to testing and fixing any bugs that may be found. If there is a limited number of bugs, more models and features could be added and tested with the remaining time. The tests will be carried out using the test cases (Appendix 1) and the functional and non-functional requirements.

## 5 Design

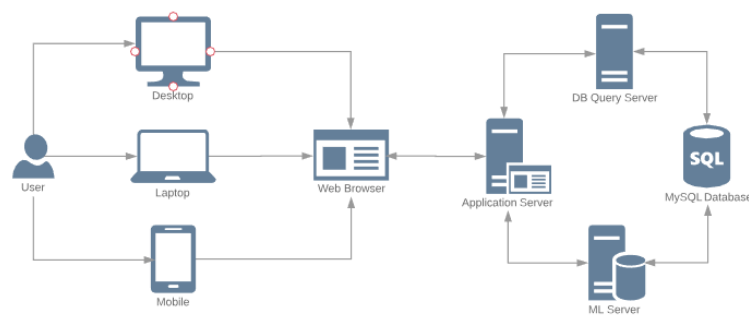
### 5.1 System Architecture

#### 5.1.1 Tools

I used Lucidchart (2018) to create the system architecture diagram (Figure 2), use case diagram (Figure 3), and state diagram (Figure 4) for this system. Lucidchart is a free online cloud-based application with a wide variety of features for building any kind of diagram.

### 5.1.2 Architecture

The application follows the Model-View-Controller (MVC) architectural pattern. The model represents the applications data. In this case, the model is the customer data stored in the MySQL database. The view is the applications user interface which updates itself as the model changes. The controller is the part of the application that handles any input and updates the model when necessary. The controllers for this application are the servers that handle the machine learning and database queries. Figure 2 is a high-level diagram of the system architecture.



*Figure 2: System Architecture.*

### 5.1.3 User Stories

My first step towards some more detailed designs was to create user stories. This allowed me to think about what a user would want to be able to achieve through using the system and why. Since I want there to be users with different privileges, I have split the user stories into the two most common roles; regular users (Table 5), and administrators (Table 6).

Story	Description
Login	As a user, I want to be able to login securely, so I know that my information is safe.
Generate Predictions	As a user, I want to be able to generate new predictions, so I can ensure they are up to date in case of poor model staying power.
View Predictions	As a user, I want to be able to view predictions, so I can compare results of different model-feature combinations.

View a Customer Profile	As a user, I want to be able to view a customer's profile, so I can identify the reasons why they may have been predicted as likely to churn and act towards retaining them.
Update Own Profile	As a user, I want to be able to update my own profile, so that I can keep my information up to date and change my password regularly.
View Help	As a user, I want to be able to seek help within the application, so that I can quickly resolve any questions without waiting for a response from a system administrator.
View Notifications	As I user, I want to be notified when key processes have completed, so that I can continue to use the application during any wait periods.

*Table 5: User Stories.*

Story	Description
Add Users	As an admin, I want to be able to add new users to the application, so that they can log in and use the applications features.
Edit Users	As an admin I want to be able to edit users' details, such as their role, so that I can manage user permissions and help recover lost accounts.
Lock Out Users	As an admin, I want to be able to lock users out of the application, so that I can end the access of users that are no longer authorized by the company.
Edit Help	As an admin, I want to be able to add and remove questions and answers in the help view, so that I can keep this section up to date and add any new questions that may be frequently asked by users.
User Functions	As an admin, I want to be able to perform all the functions available to regular users, so that I can use all the applications features as they would.

*Table 6: Admin Stories.*

### 5.1.4 Use Case Diagram

I used the user stories to create a use case diagram (Figure 3). This adds some extra detail such as how other systems are involved and how the user stories can be broken down into a series of smaller actions. This allowed me to think about how the user would navigate the application to carry out each particular action.

The diagram illustrates User and Admin actors, and 3 System actors. The Admin actor extends the User actor and is able to carry out any task the user can. The Database actor represents the applications database (model) and is queried by the Backend API actor each time the user performs a task that is associated to the Backend API (controller).

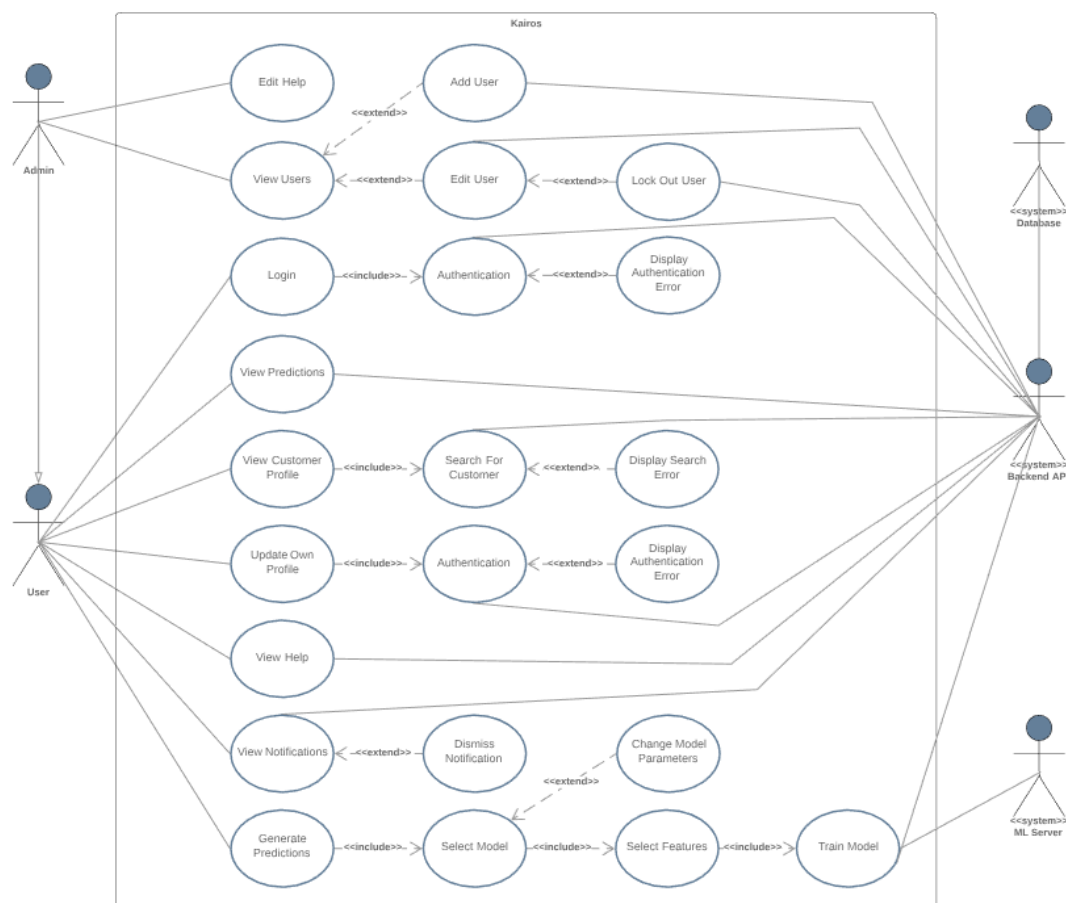


Figure 3: Use case diagram.

### 5.1.5 State Diagram

The state diagram (Figure 4) demonstrates how the state of the application will change as the two user actors navigate the application and perform the actions associated to them in the use case diagram (Figure 3).

The user begins by logging in to the application. It is important that they cannot navigate to any other state before authentication. Once authenticated, the user will be able to navigate the application freely. The 'Navigate' task encapsulates these 5 states as the user would be able to navigate to them at any time, from any state:

- Edit users.
- Train model.
- Edit profile.
- View content.
- Search.

The 'edit users' state is an exception and can only be entered by system administrators.

Each of these states contain sub-processes that can be followed after entering that state. The user can leave these states at any time by navigating elsewhere. The user is able to logout through the navigation task and reach the end state. If the user closes the application without logging out, they will remain in the navigation state and continue from here when returning to the application.

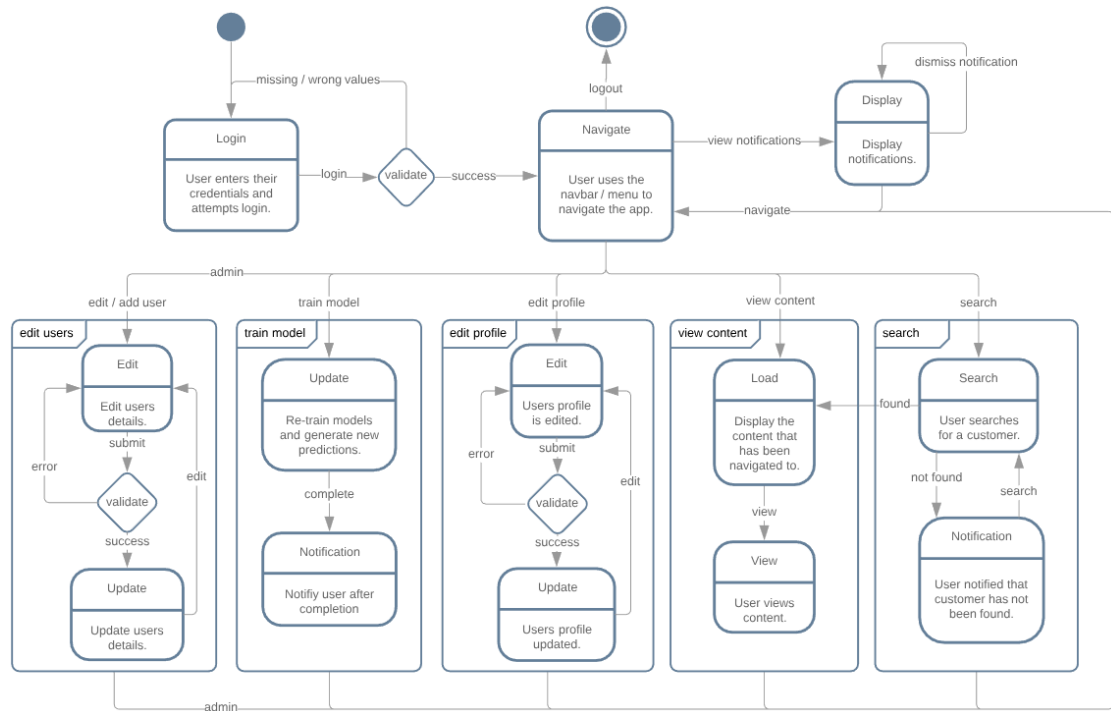


Figure 4: State diagram.

## 5.2 User Interface

After understanding how the users were going to interact with the system, I designed the user interface.

### 5.2.1 Tools

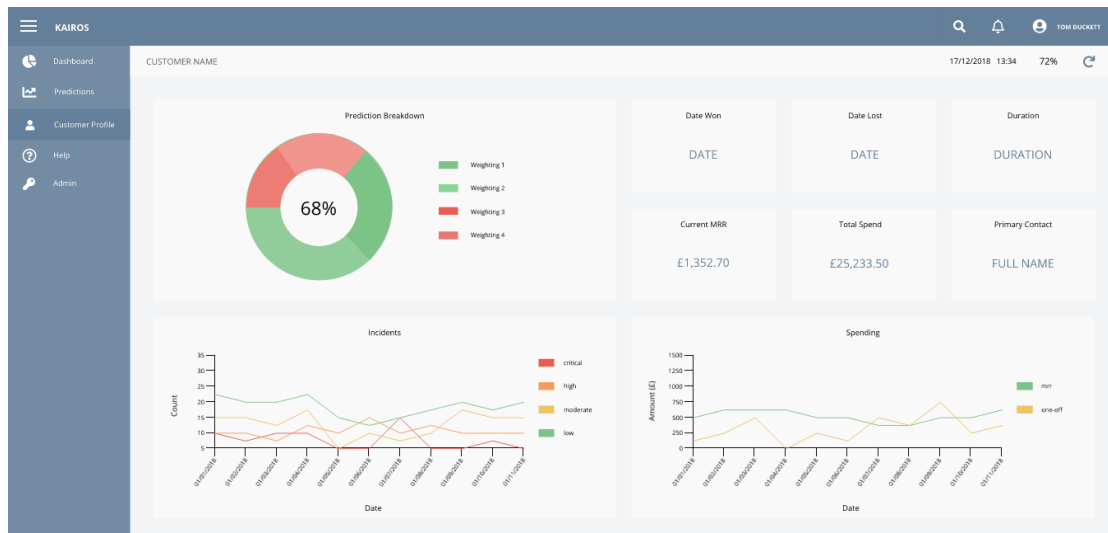
Gravit (2018) was used while designing the user interface. Gravit is a free online graphical design tool with an intuitive interface and a set of simple yet complete tools. I chose Gravit due to its built-in cloud storage feature, allowing me to work on my designs anywhere and back them up instantly. The tool can also be used in the browser so there is no need to download the software.

### 5.2.2 Layout

The basis of my design uses the universally recognised layout for content management systems to be as intuitive as possible for new users. It has a navigation bar along the top and a menu with a link to each view down the left-hand side. Since I would like users to be able to navigate to another view at any time, the navigations will always be visible. I will allow the user to hide them on smaller view-ports, so more space will be available for the view content.



(Figure 5) shows the interface design of the ‘Customer Profile’ view. I have included this particular view design as a desire for it has been shown in literature (Amin et al. 2003).



*Figure 5: Initial user interface design.*

### 5.2.3 Colours

I used two different free online tools to help me decide on the colours of the user interface; Colormind (2018) and Coolors (2018). Colormind allows you to generate colour palettes and changes the appearance of their website to give the user an idea as to how the palette may look. Coolors allows the user to export palettes or save them for later use. Both tools allow you to manually adjust each of the colours in the palette and provides their Hex and RGB codes.

### 5.2.4 Icons

I have tried to incorporate icons where possible in an attempt to increase an understanding of functions while reducing the amount of text on the UI. The icons will also prove valuable on mobile devices where the viewport is smaller, and space is limited.

### 5.2.5 Charts

Since one of the primary functions of this application is to display data in a comprehensible way, I have included graphs and charts in the interface design. This aligns with requirement NF7 (Table 4).

### 5.3 Database

I decided to use a MySQL relational database for my application as this is what the Cloud Service Provider uses for their business systems and the application would be designed to directly query them if put into production. MySQL is also used by some of the biggest names in industry (MySQL, 2019). As the customer data is being exported from a relational MySQL database, minimal reorganisation will be required before importing.

The Entity Relationship (ER) diagram (Figure 6) was drawn in MySQL Workbench (Oracle, 2018) as it provides tools to instantly convert your ER diagram into a fully functional database. This diagram creates a relationship by adding a foreign key into a table to reference the primary key of another. The data type for each column is also displayed.

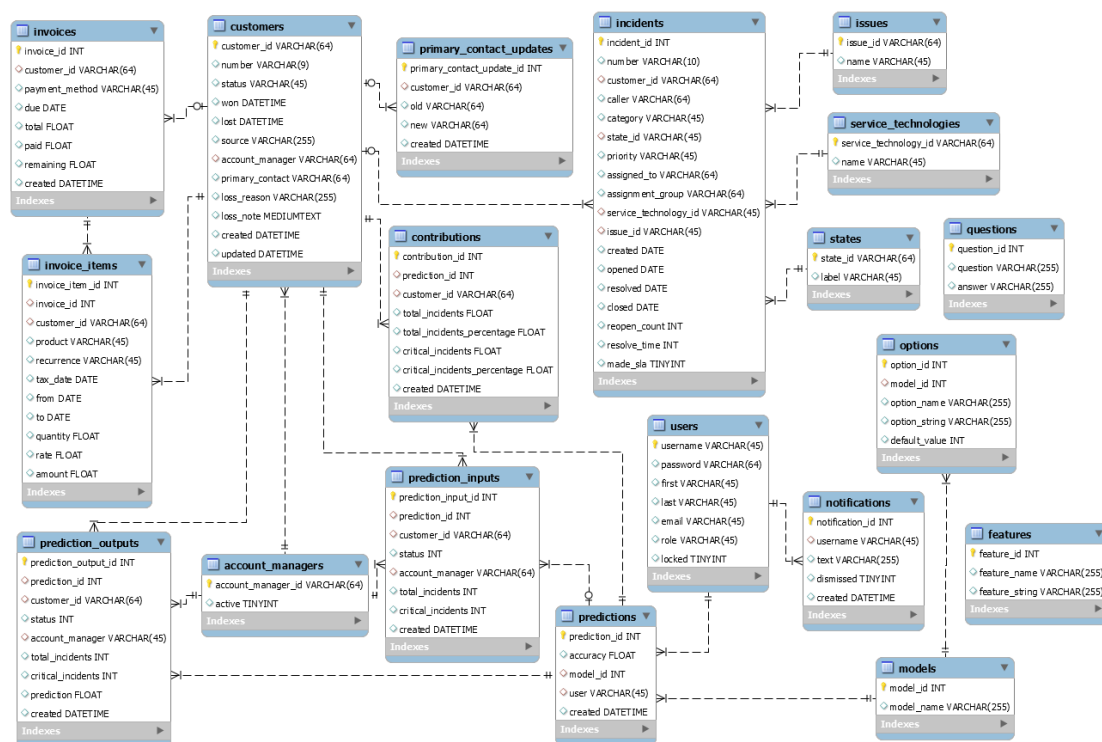


Figure 6: Entity relationship diagram.

The database design contains tables that would not need to be created if implementing this application into a company's active environment. These tables hold the customer records and their associated behaviours such as their spending and incident tickets raised.

These tables have been included into the design as I will be importing customer data into them instead of connecting to an existing database. This is primarily to avoid possible data breaches

as the security of this application is not the primary focus for this research. Columns that contain names or other sensitive information have been removed for this reason. It can also be assumed that if a company has decided to implement a customer churn predictor, they will already have existing customer data.

Existing Company Tables	Application Specific Tables
customers	predictions
invoices	prediction_inputs
invoice_items	prediction_outputs
incidents	contributions
primary_contact_updates	models
account_managers	features
issues	options
service_technologies	users
states	notifications
	questions

*Table 7: Table names.*

## 6 Implementation

During the implementation I closely followed the sprints I had defined during the planning stages. These split the project into three development milestones; the user interface, the database, and the machine learning.

### 6.1 User Interface (Sprint 1)

#### 6.1.1 Languages

The user interface was built using React.js (2019), a JavaScript library developed by Facebook for building user interface components. It uses Babel - a JavaScript compiler, to translate mark-up languages into JavaScript, and JSX - an XML/HTML syntax extension for JavaScript that uses ES6. I used HTML5 elements inside of the React components and CSS3 for the styling.

I decided to use React for these reasons:

- React has a component-based architecture which promotes clean and modular reusable code.
- Each React component can be imported and used multiple times anywhere in the UI, removing the need to duplicate code.
- React uses a Virtual DOM and handles state changes of individual components. This allows each component to be updated on the UI without the need to reload the page or parent components. This is perfect for single-page applications.

```
import React, { Component } from 'react';
/* import the container component */
import Container from '../Container/Container'

class View extends Component {
  /* the JSX to be rendered as HTML in the browser */
  render () {
    return (
      <div className='view'>
        /* below, the container component has been used twice */
        /* a prop called 'name' has also been passed to the container */
        <Container name='container 1' />
        <Container name='container 2' />
      </div>
    )
  }
}
```

*Figure 7: An example parent React.js component.*

Figure 7 shows the creation of a very simple React.js component. This ‘View’ component is considered to be a parent because some ‘Container’ components have been imported and injected into its render function. The name of a component is defined by its class name.

The ‘Container’ child component is shown in (Figure 8). Although it is not shown, it is also possible to import and use other components inside of this component. There is no limit to how many components can be nested inside another.

```

class Container extends Component {
  constructor(props){
    super(props)
    /* a components state */
    this.state = {
      some_value: 'a value'
    }
  }

  /* if this function is called it will update the components state asynchronously */
  changeValue = () => {
    /* updating a components state causes the component to re-render */
    this.setState({
      some_value: 'a new value'
    })
  }

  /* the JSX to be rendered as HTML in the browser */
  render () {
    return (
      <div className='container'>
        {/* the value of the 'name' prop replaces 'this.props.name' when compiled */}
        <p>this is a {this.props.name} component</p>
        <p>it can also retrieve {this.state.some_value} from it's state</p>
        {/* if this button is clicked, the above 'p' tag will change */}
        <button onClick={this.changeValue}>update state</button>
      </div>
    )
  }
}

export default Container

```

Figure 8: An example child React.js component.

If the ‘*changeValue*’ function is called, the ‘Container’ components will re-render due to a state update. This will not cause their parent component to re-render too. If the ‘View’ component had other elements defined in its render statement, they would be left alone.

To create a barebones React.js frontend build pipeline, I used the ‘*create-react-app*’ command in the Windows Command Prompt. I can also create an optimised build of the application using the ‘*npm run build*’ command.

### 6.1.2 Layout

Bootstrap (2019) is a popular CSS framework for developing responsive websites. I wanted the application to work well on mobile and desktop devices, so I used the Bootstrap grid system to handle a wide range of screen sizes.

Figure 9 is a screenshot of the user interfaces ‘Customer Profile’ view on both desktop and mobile devices, demonstrating how the Bootstrap grid system handles the change in view-port size by repositioning the elements.

The user interface differs slightly from its original design. I have added a couple of extra tiles to hold survey score values as I was unaware of this data during the design stages. The charts also have a grid background to help identify where a point lies on each axis, and a faded fill below each line to add more colour to the view. Finally, each tile holds some extra information about the customer.

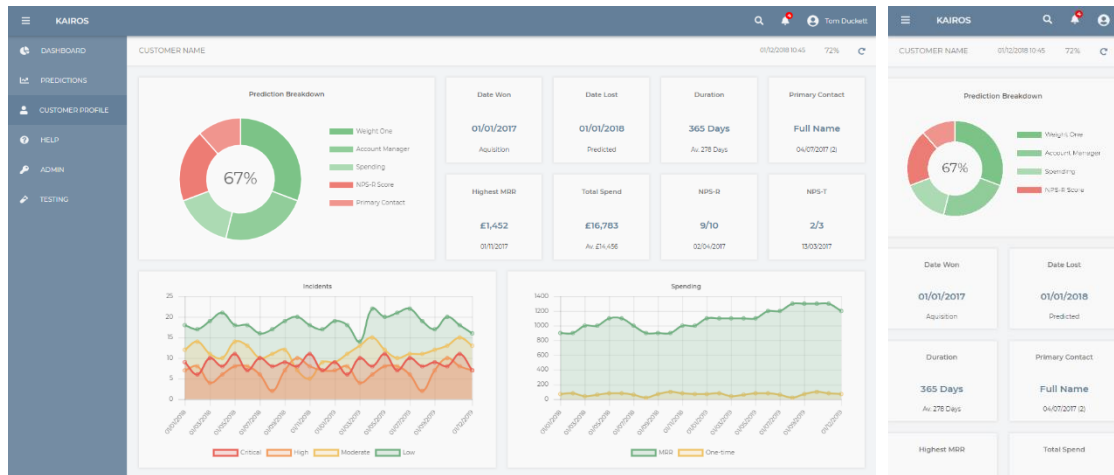


Figure 9: User Interface on desktop (left) and mobile (right).

### 6.1.3 Icons

All of the icons on the user interface are from FontAwesome's (2019) library. These icons are fonts rather than images so they can be easily manipulated to change their colours. These fonts are vectors, rendered by the CSS so they generate a clear image on any resolution. They also provide great backward-compatibility across browsers.

One of the most beneficial advantages of font-based icons is that the load time of the web page is reduced primarily due to the much smaller file sizes. Secondly, multiple HTTP requests are required to retrieve multiple images, whereas the fonts can be loaded from a content delivery network (CDN) server and easily cached.

### 6.1.4 Charts

I used the Chart.js (2019) library with a React.js wrapper for the applications charts. Chart.js offers fully customisable JavaScript graphs and charts. Customisations include chart styles and colours, legend positioning, axis scaling and the ability to add multiple datasets to one chart. These datasets can be hidden by the user by clicking their names in the legend. This is useful for further analysis and comparisons.

### 6.1.5 Navigation

To allow the application to remain single-page, I used the React Router library to handle all browser routing. When a link is clicked, this library displays a view component instead of reloading the whole page. It also manages the applications navigation history to ensure the browsers back/forward functions navigate you to previous views instead of the previously visited website.

### 6.1.6 Accessibility


Accessibility can be improved by using the correct HTML elements for their correct purposes. For example, button elements come with built-in functionality that allow the user to use keyboard shortcuts and tab between them.

Screen readers will recognise elements and tell the user if the current text is a heading, or a paragraph. Many screen readers will also allow the user to jump between elements. Appropriate sectioning elements help prevent confusion when reading out content. For example, I have wrapped the navigations in ‘<nav>’ elements. Screen readers will not provide the user with enough information or control if every element is a simple ‘<div>’ element.

Focusable elements are given a default highlighted style in every browser so that you can tell when they are focused. This is particularly useful for users who may be visually impaired. This styling can be easily removed and often is by designers who find it aesthetically unpleasing. I have decided to keep the styling as I value accessibility over style.

### 6.1.7 Error Handling

If an error occurs, the user is notified in the form of a red warning banner. These banners do not interrupt interaction with the interface like regular browser alerts do. The banner will be dismissed when the user navigates to a new view or clicks the dismiss button on the banner.

A horizontal red banner with a light red background. It contains the text "Sorry! There was an error retrieving this content." on the left and a small red 'x' icon on the right, which serves as a dismiss button.

Sorry! There was an error retrieving this content.

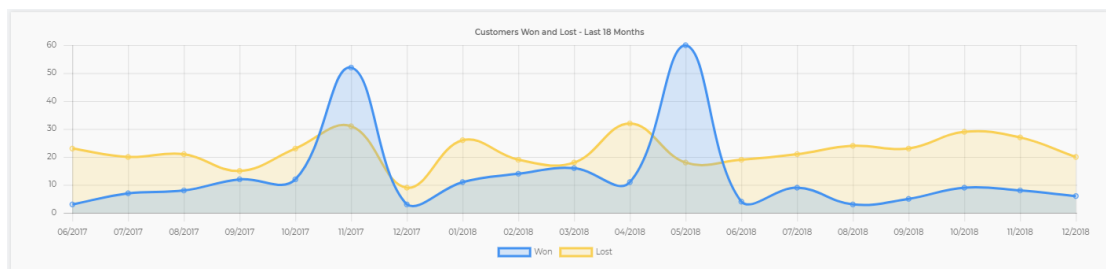


*Figure 10: Example error banner.*

### 6.1.8 Testing

I used Google Chrome's built in device emulator to test the applications interface on different device screen widths while developing. I also tested on different browsers to ensure the user interface remained consistent and displayed correctly. To test that the user interface was intuitive and easy to use, I asked 5 people with ages ranging from 18-51 to navigate the user interface and provide feedback, although I had not conducted this particular test until after the next sprint. One user suggested the colours of the charts may cause problems for users that suffer with colour blindness.

People with deuteranomaly and protanomaly are known as red-green colour blind. They generally have difficulty distinguishing between reds, greens, browns and oranges. This is the most common type of colour blindness (Colour Blind Awareness). After learning this, I experimented with some new colour schemes using Colormind (2018) and Coolers (2018). I decided on blues and yellows as these seem to be the best choice to accommodate for the different types of colour blindness (Figure 11).



*Figure 11: Example chart with new colours.*

The 'Navigation' test cases (Appendix 1) were completed at the end of this sprint. All the tests passed except one - 'N6: Attempt to navigate to an admin view as a regular user'. This was a simple fix as I had simply forgotten to enclose the code handling this particular route with the admin Boolean variable (Figure 12).



```

{/* private */}
<PrivateRoute exact path='/dashboard' component={Dashboard} />
<PrivateRoute exact path='/predictions' component={Predictions} />
<PrivateRoute exact path='/customer_profile' component={CustomerProfile} />
<PrivateRoute exact path='/train_model' component={TrainModel} />
<PrivateRoute exact path='/help' component={Help} />
<PrivateRoute exact path='/profile' component={Profile} />
{/* admin */}
{admin && <PrivateRoute exact path='/admin' component={Admin} />}

```

*Figure 12: Routing components.*

### 6.1.9 Requirements Satisfaction

This sprint contributes to satisfying the following functional and non-functional requirements (Tables 3 and 4):

- NF1: The system must have a user interface that is easy to interpret and navigate.
- NF5: The system should be accessible on a range of devices.
- NF7: The system should present data in chart or tabular format.

## 6.2 Backend (Sprint 2)

The backend of the application works as an API between the web-client and the database. It consists entirely of functions that are called by the HTTP requests sent by the web client.

During this sprint, all of the functions to handle user actions, such as logging in/out and updating their profile, retrieving data from the database, and adding/removing help questions were implemented.

The backend is hosted on port 4000 of my localhost to simulate another server.

### 6.2.1 Languages

The JavaScript backend uses Node.js (2019), a free, open source, interoperable JavaScript server environment. I decided to use Node.js over PHP or ASP as unlike these technologies, it is fully asynchronous which eliminates any waiting between requests. Node.js is very memory efficient as it runs asynchronously on a single thread.

## 6.3 Database (Sprint 2)

The database was created using MySQL Workbench (Oracle 2018). I was able to easily convert the ER diagram I had previously created (Figure 6) into a fully functional relational database, using the tools provided by the workbench.

### 6.3.1 Data Imports

All of the customer data had to be exported from the Cloud Service Providers systems and anonymised before being imported into the applications database using the MySQL Workbench (Oracle, 2018) import feature. I did not directly query their database to reduce the risk of a data breach. If this application was to be put into production, data anonymisation and importing would not be necessary, and security would need to become a priority.

## 6.4 Security

Once the database and backend were configured, I began building the login functionality.

As the system is a web-application, the below security measures have been put in place to prevent unauthorized access. However, as the application is running on a development server, it does not hold a certificate for HTTPS. Services such as Firebase could be used to host the application with a certificate, however this would also require the database to be hosted in the cloud to continue communication with the web-client.

### 6.4.1 Secure Login

I used the Bcrypt (NPM, 2019) library for hashing passwords. This hashing technique generates a new, completely random salt every time, which is stored with the password in the database. Therefore, even if you input the same value into the hashing function twice, you will receive two different hashes. To compare passwords, the function retrieves the salt from the password stored in the database and uses this to hash the users input and compare.

```
Salt: $2b$10$/DXiVVE59p7G5k/4Klx/e
```

```
Hash: $2b$10$/DXiVVE59p7G5k/4Klx/ezF7BI42QZKmoOD0NDvUuqxRE5bFFBLy
```

*Figure 13: Example salt and hash.*

### 6.4.2 User Sessions

After a successful login, the user's authentication token will be stored in local storage. If their token exists and is valid when visiting the application, they will automatically log in. The user's token will be removed from storage after logging out of the application. At this stage the token generation is just a simulation and does not provide actual security yet. Before going live, the token generation would have to be revisited and the tokens sent to an authorisation service, such as Netlify's (2019) Identity for validation.

### 6.4.3 Private Routing

Private routing was used to prevent unauthorised access to different views of the application. If a user has not been authenticated during login, they will only be able to navigate to the login view. An attempt to navigate elsewhere will result in redirection to the login page. If an authenticated user tries to access admin views without the admin role, they will be redirected to the dashboard view.

### 6.4.4 SQL Injection

All user inputs are escaped before querying the database to prevent SQL injection. To do this I have used the `SqlString` (NPM, 2018) library.

### 6.4.5 Testing

I completed the user actions, viewing of results, and generating predictions test cases (Appendix 1). At this time some dummy functions were created to simulate the machine learning process by generating random results and writing them to/retrieving them from the database. These functions would then be used as part of the 'friendly user testing' explained in the 'Testing' section. During development I also ensured the asynchronous functions and their call-backs executed correctly and that promises resolved in the correct order.

### 6.4.6 Requirements Satisfaction

This sprint contributes to satisfying the following functional and non-functional requirements (Tables 3 and 4):

- F4: The system must restrict access in the form of a secure login feature.
- F5: the system should allow users to update their profile.
- F7: The system should allow for different user roles.

- F8: The system could provide help to users in the form of questions and answers.
- NF3: The system must be easy to maintain.
- NF4: The system must be accessible by multiple users at any one time.
- NF6: The system should retrieve data quickly.

## 6.5 Machine Learning (Sprint 3)

### 6.5.1 Languages

Although I used JavaScript to build the user interface and query the database, I decided to use Python for the machine learning algorithms. JavaScript does provide machine learning libraries such as Tensorflow.js (2019), however I found the Python libraries were easier to interpret, more readily available, and provided more examples and support online.

### 6.5.2 Data Processing

The dataset contains a total of 3144 customers, 1473 are non-churners and 1671 are churners. As the classifier for this dataset is not heavily skewed, the class imbalance problem does not occur (Japkowicz, 2000).

```

# create dataframe using features and set the customer id as the index
features = ['customer_id', 'status', 'account_manager'] + features
dataframe = pd.DataFrame(inputs, columns = features)
features = dataframe.set_index('customer_id')

# replace NaN with 0
for i in features:
    features[i] = features[i].fillna(0)

# save account managers
account_managers = features['account_manager']
features = features.drop('account_manager', axis=1)

# save copies of dataframe
result = features
customers = features

# one-hot encode the data using pandas get_dummies
features = pd.get_dummies(features)

# save the labels
labels = np.array(features[label])

# remove the labels from the features - axis 1 refers to the columns
features = features.drop(label, axis = 1)

# saving feature names for later use
feature_list = list(features.columns)

# convert features to numpy array
features = np.array(features)

```

*Figure 14: Data processing.*

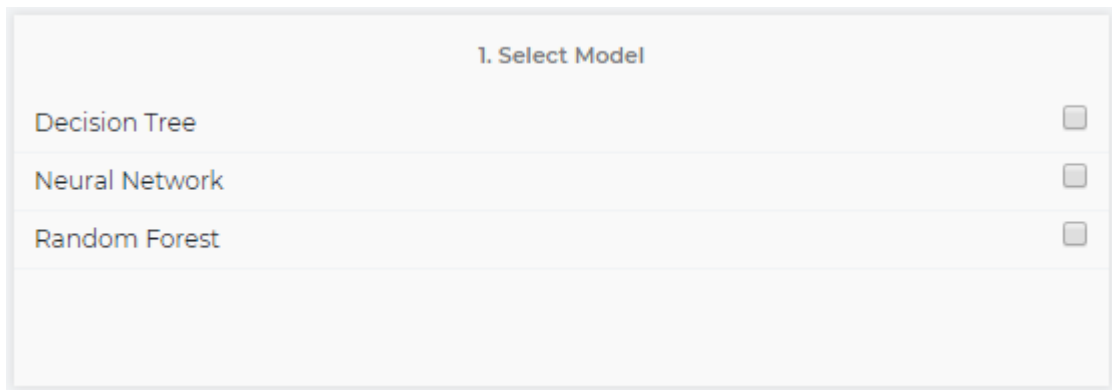
(Figure 14) shows the processing of the data. The ‘features’ and ‘input’ variables are returned from other functions - ‘features’ is an array of strings containing feature names, and the ‘input’ is the result returned by the ‘cursor.fetchall()’ MySQL library function. These are then combined to create a data frame (Figure 15).

customer_id	status	total_incidents	...	moderate_incidents	low_incidents
000342a7db434780591972fabf9619fd	2	17.0	...	2.0	14.0
000e93833c98250003bf5d442c31d2a6	1	2.0	...	1.0	1.0
00130dd337e19240cee4db9643990e52	2	19.0	...	2.0	16.0
005d9a1659980100e50a18315ecfdede	2	4.0	...	1.0	3.0
006d125659980100e50a18315ecfde04	2	32.0	...	11.0	18.0

*Figure 15: Features data frame.*

Next, all NaN’s are replaced with 0s, the customer account managers are removed, and the rest of the data is one-hot encoded. Finally, the features are formatted into a numpy array (Figure 16).

```
[5 rows x 6 columns]
[[17.  1.  0.  2. 14.]
 [ 2.  0.  0.  1.  1.]
 [19.  0.  0.  2. 16.]
 ...
 [19.  0.  0.  2. 14.]
 [ 3.  0.  0.  0.  3.]
 [ 8.  0.  0.  5.  3.]]
```



1. Select Model	
Decision Tree	<input type="checkbox"/>
Neural Network	<input type="checkbox"/>
Random Forest	<input type="checkbox"/>

*Figure 18: Model selection.*

### 6.5.3.1 Decision Tree

Decision trees use a method known as ‘divide and conquer’ to construct a binary tree. This method searches for an attribute with the best information gain to be the root node. It then divides the tree into sub trees, repeating recursively. It halts if a leaf node is reached or there is no information gain. Rules can be generated by traversing the branches of the trees (Huang, Tahar Kechadi, and Buckley, 2012). Decision trees are well suited for categorical variables because they use frequency comparisons to build the model and can handle many categories simultaneously (Coussement, Lessmann, and Verstraeten, 2017).

The applications decision tree was built using the Scikit-Learn (2019) library. I used a regressor decision tree to receive a probability prediction rather than a classification.

```
# train a decision tree regressor model
dt = tree.DecisionTreeRegressor(max_depth = max_depth,
                                max_leaf_nodes = max_leaf_nodes,
                                min_samples_split = min_samples_split,
                                min_samples_leaf = min_samples_leaf,
                                max_features = max_features,
                                presort = presort)
dt.fit(train_features, train_labels)

# make predictions on the test data
predictions = dt.predict(test_features)
```

*Figure 19: Fitting a Decision Tree and making predictions (Scikit-Learn).*

### 6.5.3.2 Random Forest

Random forests use decision trees as the base classifier. At each node in the tree, a subset of the available features is randomly selected and the best split available for those features is selected for that node (Burez and Van Den Poel, 2009). As they are often biased towards the majority class, a heavier penalty (larger weight) can be place when miss classifying the minority class (Burez and Van Den Poel, 2009). The number of base classifiers in an ensemble such as a random forest is an important parameter to be optimised (Coussement, Lessmann, and Verstraeten, 2017).

The applications random forest was built using the Scikit-Learn (2019) library. I used a regressor random forest to receive a probability as a prediction rather than a classification.

```
# train a random forest regressor model
rf = RandomForestRegressor(n_estimators = estimators,
                           max_depth = max_depth,
                           min_samples_split = min_samples_split,
                           min_samples_leaf = min_samples_leaf,
                           max_features = max_features,
                           max_leaf_nodes = max_leaf_nodes,
                           warm_start = warm_start)
rf.fit(train_features, train_labels)

# make predictions on the test data
predictions = rf.predict(test_features)
```

*Figure 20: Fitting and Random forest and making predictions (Scikit-Learn).*

The functions used for the random forest are identical to the decision tree. The only difference is the model name and the adjustable parameters. Since both algorithms also take the same input, it is easy to configure the application to use either one.

### 6.5.3.3 Neural Network

Neural networks attempt to simulate biological neural systems and can be distinguished as either single or multi-layer perceptron's (Tsai and Lu, 2009). Supervised feed-forward neural networks consist of an input layer, hidden layers and an output layer. Usually the activation function is a sigmoid function (Huang, Tahar Kechadi, and Buckley, 2012) so I have decided to use this function for the final layer.



The applications neural network was built using the Keras (2019) library. I used a classification neural network as I was having issues with the probability function still only returning a classification.

```
# feature scaling
sc = StandardScaler()
features = sc.fit_transform(features)
train_features = sc.fit_transform(train_features)
test_features = sc.transform(test_features)
```

*Figure 21: Neural network feature scaling.*

Neural networks have difficulties in accepting string values for features such as names of people or products. These features need to be rewritten as numerical values and normalized (Huang, Tahar Kechadi, and Buckley, 2012). I decided to scale the data for this algorithm (Figure 21), which also helps to increase training speed and reduce the chances of getting stuck in a local optimum. To do this I used the ‘StandardScalar’ function provided by Scikit-Learn (2019).

```
# train a neural network classification model
model = Sequential()
model.add(Dense(4, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(
    loss = 'mean_squared_error',
    optimizer = sgd,
    metrics=['accuracy']
)

model.fit(train_features,
          train_labels,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(test_features, test_labels))

# make predictions on the test data
predictions = model.predict_classes(test_features)
```

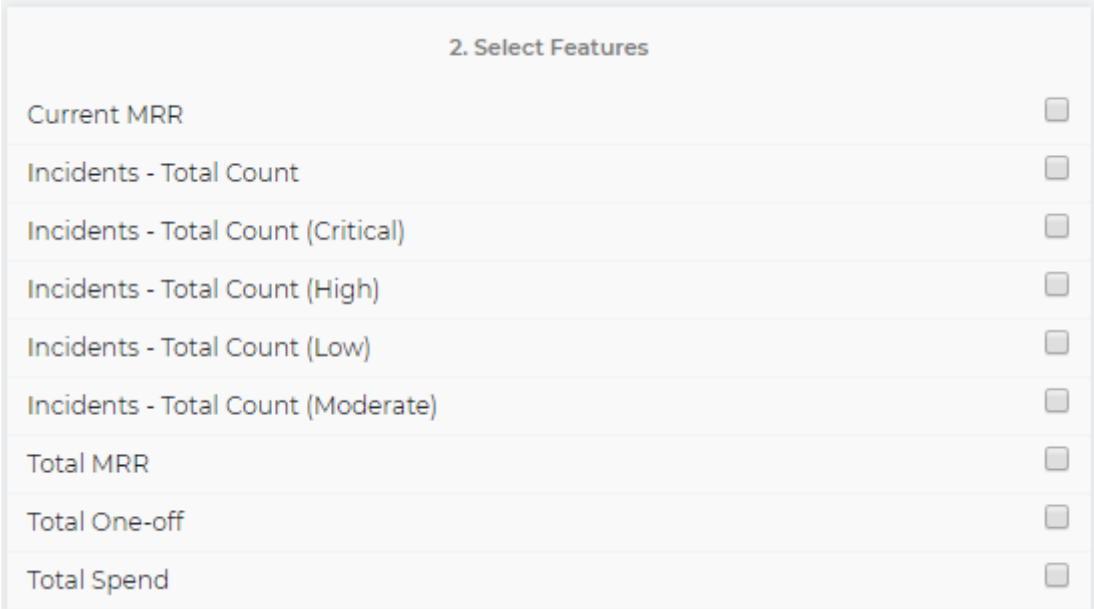
*Figure 22: Fitting a Neural Network and making predictions (Keras).*

#### 6.5.4 Features

The currently available features include:

- Incident count.
- Incident count by priorities.
- Total spends.
- Monthly Recurring Revenue (MRR) data.
- One-off payment data.

There are plans to include the customer's account manager, customer source, primary contact changes, customer lifetime duration, and satisfaction survey scores as features in the near future.



The screenshot shows a web interface titled "2. Select Features". It contains a list of features, each with a checkbox to its right. The features listed are:

Feature Name	Selected
Current MRR	<input type="checkbox"/>
Incidents - Total Count	<input type="checkbox"/>
Incidents - Total Count (Critical)	<input type="checkbox"/>
Incidents - Total Count (High)	<input type="checkbox"/>
Incidents - Total Count (Low)	<input type="checkbox"/>
Incidents - Total Count (Moderate)	<input type="checkbox"/>
Total MRR	<input type="checkbox"/>
Total One-off	<input type="checkbox"/>
Total Spend	<input type="checkbox"/>

*Figure 23: Feature selection.*

#### 6.5.5 Parameters

Model parameters will affect the performance of a model and the execution time. The user is able to adjust the parameters of each of the models during experimentation to find the best combination.

The parameters in (Figure 24) will change depending on the selected model. Default values will be presented to the user to give them an idea as to the input these fields expect. There is also validation in place to ensure a user does not leave the fields empty or input invalid values.

3. Adjust Parameters	
Max Depth	<input type="text" value="5"/>
Max Features (<= No. Features Selected)	<input type="text" value="1"/>
Max Leaf Nodes	<input type="text" value="10"/>
Min Samples Leaf	<input type="text" value="1"/>
Min Samples Split	<input type="text" value="2"/>
No. of Trees	<input type="text" value="20"/>
Test Size	<input type="text" value="0.5"/>
Warm Start	<input type="text" value="0"/>

*Figure 24: Parameter adjustments (Random Forest).*

#### 6.5.6 Testing

The ‘Generating Predictions’ and ‘Viewing Results’ test cases (Appendix 1) were completed at the end of this sprint. These test the training of models following user input, and the way the results are displayed in the ‘prediction’ and ‘customer profile’ views.

Initially tests ‘GP7: Train with no model selected’, and ‘GP8: Train with no features selected’ failed. I fixed these before moving onto the final sprint to do a complete retest of all system functions. All other tests passed successfully.

#### 6.5.7 Requirements Satisfaction

This sprint contributes to satisfying the following functional and non-functional requirements (Tables 3 and 4):

- F1: The system must allow the user to make a prediction of customer churn.
- F2: The system must allow the user to choose from a selection of different dataset features.
- F3: The system must provide a profile for each customer.
- F6: The system should train its algorithms on a dedicated server.
- NF2: The system must be scalable and allow for new models, features and parameters to be added over time.
- NF7: The system should present data in chart and tabular formats.

## 6.6 Software Testing (Sprint 4)

The testing of the system was separated into three distinct approaches, the first being a series of ‘formal test cases’ (Appendix 1) that were created during the design stages. I tested everything that had been implemented during a sprint, at the end of that sprint, but also allocated this final sprint to thoroughly test the application again. I also tested the application against the functional and non-functional requirements.

The second aspect of the strategy was ‘friendly user testing’, focused on the user interface and overall usability of the system. This involved different users operating the system, providing feedback on the look, feel, and usability. These users ranged in age from 18-51, in an attempt to capture feedback relating to the majority of would-be-users in a real-world deployment.

Finally, a ‘real world outcome validation’ test cycle was conducted, running real data through the system and comparing the output with known results of actual customers. The cloud service provider was able to provide the details and statuses of the customers from the month following that of the exported data. I was then able to compare my systems predictions with the actual outcomes of the same customers.

### 6.6.1 Formal Test Cases

The formal test cases (Appendix 1) are split into user actions, navigation, generating predictions, viewing results, admin functions, and notifications.

The user test cases covered logging in and out, re-visiting a session, and updating the user’s profile. The navigation tests focused on ensuring each view was accessible and displayed correctly. The tests for generating predictions and viewing results encapsulated the process of choosing a model, set of features, and adjusting the model parameters to generate a set of prediction results which could then be viewed by the user. Lastly, the admin test cases covered the management of user accounts, such as editing users details or locking/unlocking their accounts.

All of the notification test cases failed as this feature was not implemented due to time constraints.

### 6.6.2 Friendly User Testing

The users were asked to navigate the interface and perform some actions using a dedicated admin testing account, on both desktop and mobile devices. At this stage dummy client-side

functions were used to simulate the generation of predictions since this functionality had not yet been implemented.

Actions:

- Log in to the system with a username of 'test.user' and a password of 'testing'.
- Navigate to your profile and change the password.
- Log out of the system and log back in using your new password.
- Change the last name of the user 'Another Test' to 'User'.
- Generate a set of predictions using any model and features.
- View the profile of a customer with these results.
- Create a new set of results using a different model and set of features.
- View the previous customer profile again with the new results.
- View the results for a different customer.
- View the accuracies of both models used.

A questionnaire with the following questions was then filled out by each user:

- How intuitive do you find the user interface?
- How appealing is the user interface?
- Did you have any problems navigating the system or finding the features desired?
- If you could add another feature, what would it be?
- What would you change in terms of design, layout or the process of navigating the user interface?
- What overall score out of 10 would you give the user interface?

The results were positive, with an average score of 9 for the user interface and minimal problems experienced.

### 6.6.3 Real World Outcome Validation

The new extract of customer data contained 5 newly churned customers. 3 of these had been predicted to have a high probability of churn when using my system after some experimentation.

This gives me confidence that the system initially works but could be further improved. More experimentation would be needed to create more accurate predictions. Further to this, more features may need to be implemented to find the best combination.

#### 6.6.4 Reflection on Test Results

Each of the three stages of testing have all contributed to the testing process in a different but valuable way. The structured test cases have allowed me to exercise the solution against the requirements and ensure that the basic features and end to end data flows operate as required. This is the minimum level of testing that should be performed to ensure a working solution and to satisfy me that the requirements have been covered at a detailed level. The end to end test cases have been useful to validate the data flows and data management aspects of the system.

My approach to the software development means that I have tested individual aspects of the code as the solution was produced, ensuring that each component written works correctly. The final phase of structured test cases revalidates the features and ensures that when operating together, each aspect of the solution continues to work correctly. Small defects were discovered and quickly corrected. On this basis I have confidence that the solution would operate as designed.

The 'friendly user testing' found a very limited number of bugs, such as a couple of badly formatted components in mobile view, but it did highlight the need to adjust chart colours to maximise the usability of the system. These results and feedback proved beneficial.

Finally, the 'real world outcome' validation was a critical step in terms of gaining confidence that the solution as a whole was able to provide the required insight into likely customer actions. Overall the initial results are promising with a series of matches made between the system and the actual actions taken by the customers to which the data relates. However, I would recommend that a marketing strategy to be implemented only when the accuracy of the results have improved.

I would recommend that automated tests be designed and implemented to allow the system to be effectively re-tested for regression issues at the point at which changes are made. Manually testing the system was time consuming, although the results were valuable and allowed me to improve the system further.

## 7 Project Evaluation

### 7.1 Limitations

Firstly, it is hard to see the impact this application could have on improving customer retention without using it for a generous period of time and implementing a marketing strategy for improving relationships with predicted churners. It is clear from the real-world validation test

phase that initial results are promising but that a prolonged period of parallel running of the solution will be needed before a business would be confident to take specific actions with respect to actual customers. This is aligned with feedback I have received from the cloud service provider who see the initial results but have a desire to run the system for a longer period of time to understand the reliability of the results and build trust and confidence in the system.

Secondly, in its current state it would require a backend rebuild if implementing it for a company with a different database structure. Further considerations would need to be made to the flexible nature and structure of some incoming data and for the solution to be easily adaptable if data sources change or are extended.

## 7.2 Improvements

If data confidentiality and security was not an issue for this project, the application could have been designed to work with various APIs in order to query data from popular customer management systems. I would also utilise cloud services such as Microsoft Azure or Amazon Web Services to host each component.

Changes could be made to allow users to select a date range for the time series charts. A customer's previous probability scores could also be displayed on their profile page, and a chart to show the scores over time.

More models and features could be added, and the ability to select multiple models to create ensembles would also allow for further analysis and comparisons. New models may prove to be computationally expensive and this could become an important statistic to report on for model comparison. Other model validation techniques, such as confusion matrices, could be implemented. A feature could be added to allow users to choose between a classification or probability algorithm for each model.

Models could be stored and retrieved. This would allow reuse of the model after a period of time to test the models staying-power. Visuals of each model could also be displayed, to provide a better understanding as to how each algorithm made its decision.

An LSTM model could be implemented to make predictions on time-series data. For example, you could predict how features such as the number of incidents, or monthly spend, may look a week from now. A step further would be to make these predictions for each feature, for every customer, and process the newly generated dataset through one of implemented models to predict how the results may look a week from now.

To potentially improve accuracy of the models, an option to train them with a Genetic Algorithm could be implemented. This would drastically increase runtime but would automate the experimentation process. To further automate the process, training could be scheduled to run automatically after a given time period.

### 7.3 Reflection

The research proved valuable and it was very beneficial to understand the other approaches taken by the authors to solve this problem for other industries. By gathering an understanding of the desired comparisons to be made, I was able to prioritise the implementation of specific models.

The requirements and designs I had created were descriptive and complete enough for me to understand the direction of the project throughout its entire lifetime. They kept me from straying from the objectives path as I often drastically change aspects of a system when jumping straight into the code. Productivity was increased as I did not waste time trying to implement changes in the code, such as the change to user interface colours, without first altering the design, which proved to be much more efficient.

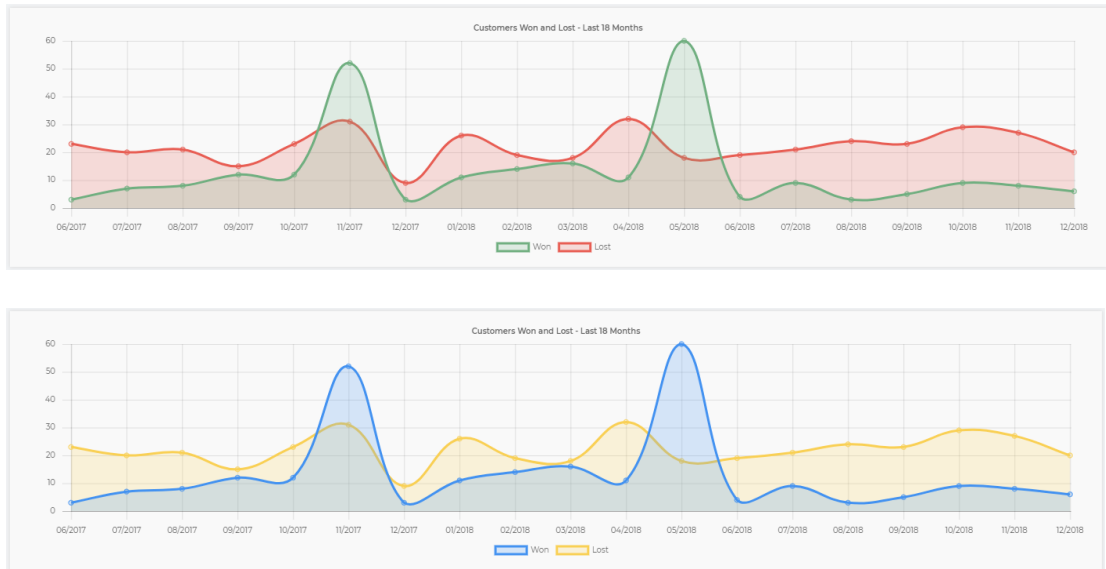
I should have asked for more feedback during the design stage, but the ‘friendly user testing’ process helped to rectify this, and fine tune aspects of the design related to the user interface and operability.

### 7.4 Feedback

In the first few meetings with Phil, my supervisor, we brainstormed the different languages, libraries and approaches that may be suitable for this type of project. Phil had pointed me in the direction of Flask, a simple python server, which I then used during development to host the machine learning. During the design stages, Phil provided me with valuable feedback on my requirements and state diagram. Figure 25 shows my first attempt at the state diagram, where Figure 4 shows the final version. I was able to create a more detailed and more interpretable diagram following Phil’s feedback – the most useful of which was to use bounding boxes around each of the sub-processes as they may loop on themselves without navigating to a new page.







*Figure 26: Old and new chart colours.*

I demonstrated the system to the cloud service provider on multiple occasions to capture feedback from a stakeholder perspective. All feedback was positive, and they appeared impressed with the system and the available functions to users. Prior to implementing the feature selection functionality, they suggested that capturing incident data by priority may prove more useful rather than generalising all incidents. It was advised to prioritise the incident features as they believed they could see a trend in the current data, where customers had threatened to cancel services after an increase in raised incident tickets. They then felt that spending data would be the next most useful to capture. I listened to this advice and prioritised the implementation of incident and spending features during the development of this part of the system.

## 8 Conclusion

The overall aim of the application is to allow users to generate predictions for customer churn. The application allows users to select a model, select a number of features and adjust the model parameters. The results of these models can also be compared and a previous result for any customer can be retrieved. Each customer has a profile which displays the confidence score of the chosen prediction, and a breakdown of how each feature has contributed. I would conclude that overall the implementation has been a success.

At this stage a ‘proof of concept’ can be considered to have been completed. A longer period of evaluation will be needed to allow the solution to be tuned for increasingly confident results and to potentially consider other data sources.

From a personal perspective, the project has allowed me to further understand the software development process, to experience the full lifecycle of development based on real data and real business requirements, and to appreciate the contribution that different approaches to testing can bring.

## 8.1 Further Work

The next steps would be to add some of the additional features previously mentioned, and to include regression and classification functions for each model. More models could also be added once the system has proven to be robust and provide confidence with the current models.

Time series analysis could also be implemented using LSTMs, and genetic algorithms used to train the models. Previously used models could also be stored.

## 9 References

### 9.1 Literature

Allison, P.D. (1999) Logistic Regression Using SAS®: Theory and Application. Cary: SAS Institute.

Amin, A., Al-obeidat, F., Shah, B., Adnan, A., Loo, J. and Anwar, S. (2019) Customer Churn Prediction in Telecommunication Industry Using Data Certainty. *Journal of Business Research* [online]. 94, pp. 290-301. [Accessed 01 February 2019].

Amin, A., Anwar, S., Adnam, A., Nawaz, M., Alawfi, K., Hussain, A. and Huang, K. (2017) Customer Churn Prediction in the Telecommunication Sector Using a Rough Set Approach. *Neurocomputing* [online]. 237, pp. 242-254. [Accessed 15 October 2018].

Ammar A.Q. Ahmed and Maheswari, D. (2017) Churn Prediction on Huge Telecom Data Using Hybrid Firefly Particle Swarm Optimization Algorithm Based Classification. *IOSR Journal of Computer Engineering* [online]. 19 (4), pp. 30-39. [Accessed 03 November 2018].

Buckinx, W. and Van Den Poel, D. (2005) Customer Base Analysis: Partial Defection of Behaviourally Loyal Clients in a Non-contractual FMCG Retail Setting. *European Journal of Operational Research* [online]. 164 (1), pp. 252-268. [Accessed 13 October 2018].

Burez, J. and Van Den Poel, D. (2007) CRM at a Pay-tv Company: Using Analytical Models to Reduce Customer Attrition by Targeted Marketing for Subscription Services. *Expert Systems with Applications* [online]. 32 (2), pp. 277-288. [Accessed 11 November 2018].

Burez, J. and Van Den Poel, D. (2009) Handling Class Imbalance in Customer Churn Prediction. *Expert Systems with Applications* [online]. 36 (3), pp. 4626-4636. [Accessed 10 October 2018].

Clark, M. (1997) Modelling the Impact of Customer-employee Relationships on Customer Retention Rates in a Major UK Retail Bank. *Management Decision* [online]. 35 (4), pp. 293-301. [Accessed 17 November 2018].

Colour Blind Awareness. (no date) Types of Colour Blindness. Available from: <http://www.colourblindawareness.org/colour-blindness/types-of-colour-blindness/>

Coussement, K., Benoit, D.F. and Van Den Poel, D. (2010) Improved Marketing Decision Making in a Customer Churn Prediction Context Using Generalized Additive Models. *Expert Systems with Applications* [online]. 37 (3), pp. 2132-2143. [Accessed 18 October 2018].

Coussement, K., Lessmann, S. and Verstraeten, G. (2017) A Comparative Analysis of Data Preparation Algorithms for Customer Churn Prediction: A Case Study in the Telecommunication Industry. *Decision Support Systems* [online]. 95, pp. 27-36. [Accessed 13 October 2018].

Dala, D. K., Patidar, K., Chouhan, S. and Sharma, N. (2018) Predicting Customer Churn in Telecom Sector Using Cart. *Journal of Applied Science and Computations* [online]. 5 (9), pp. 179-186. [Accessed 08 October 2018].

De Caigny, A., Coussement, K. and De Bock, K.W. (2018) A New Hybrid Classification Algorithm for Customer Churn Prediction Based on Logistic Regression and Decision Trees. *European Journal of Operational Research* [online]. 269 (2), pp. 760-772. [Accessed 24 October 2018].

Degbey, W.Y. (2015) Customer Retention: A Source of Value for Serial Acquirers. *Industrial Marketing Management* [online]. 46, pp. 11-23. [Accessed 27 October 2018].

Devi Prasad, U and Madhavi, S. (2012) Prediction of Churn Behaviour of Bank Customers Using Data Mining Tools. *Indian Journal of Marketing* [online]. 42 (9) [Accessed 17 November 2018].

Fuller, K., Netter, K. and Stegemoller, M. (2002) What Do Returns to Acquiring Firms Tell Us? Evidence from Firms That Make Many Acquisitions. *The Journal of Finance* [online]. 57 (4), pp. 1763-1793. [Accessed 07 November 2018].

Gordini, N. and Veglio, V. (2017) Customers Churn Prediction and Marketing Retention Strategies. an Application of Support Vector Machines Based on the Auc Parameter-selection Technique in B2b E-commerce Industry. *Industrial Marketing Management* [online]. 62, pp. 100-107. [Accessed 10 October 2018].

Guo-En, X. and Wei-Dong, J. (2008) Model of Customer Churn Prediction on Support Vector Machine. *Systems Engineering — Theory & Practice* [online]. 28 (1), pp. 71-77. [Accessed 24 October 2018].

Hadden, J., Tiwari, A., Roy, R. and Ruta, D. (2007) Computer Assisted Customer Churn Management: State-of-the-art and Future Trends. *Computers & Operations Research* [online]. 34 (10), pp. 2902-2917. [Accessed 15 October 2018].

Hadden, J., Tiwari, A., Roy, R. and Ruta, D. (2008) Churn Prediction: Does Technology Matter? *International Journal of Industrial and Manufacturing Engineering* [online]. 2 (4), pp. 524-530. [Accessed 15 October 2018].

Huang, B., Tahar Kechadi, M. and Buckley, B. (2012) Customer Churn Prediction in Telecommunications. *Expert Systems with Applications* [online]. 39 (1), pp. 1414-1425. [Accessed 15 October 2018].

Hung, S.-Y., Yen, D.C. and Wang, H.-Y. (2006) Applying Data Mining to Telecom Churn Management. *Expert Systems with Applications* [online]. 31 (3), pp. 515-524. [Accessed 15 October 2018].

Japkowicz, N. (2000) Learning from Imbalanced Data Sets: A Comparison of Various Strategies. AAAI Press [online]., pp. 10-15. [Accessed 15 October 2018].

Kawale, J., Pal, A. and Srivastava, J. (2009) Churn Prediction in MMORPGs: A Social Influence Based Approach. *2009 International Conference on Computational Science and Engineering* [online]. 4, pp. 423-428. [Accessed 17 November 2018].

Markel, Z and Bilzor, M. (2014) Building a Machine Learning Classifier for Malware Detection. *2014 Second Workshop on Anti-malware Testing Research (WATeR)* [online]. [Accessed 01 March 2019].

Miao, Q., Liu, J., Cao, Y. and Song, J. (2016) Malware Detection Using Bilayer Behavior Abstraction and Improved One-class Support Vector Machines. *International Journal of Information Security* [online]. 15 (4), pp. 361-379. [Accessed 01 March 2019].

MySQL (2019) MySQL Customers. Available from: <https://www.mysql.com/customers/> [Accessed 10 December 2018].

Neslin, S.A., Gupta, S., Kamakura, W., Lu, J. and Mason, C.H. (2006) Defection Detection: Measuring and Understanding the Predictive Accuracy of Customer Churn Models. *Journal of Marketing Research* [online]. 43, pp. 204-211. [Accessed 03 November 2018].

ÓSkarsdóttir, M., Bravo, C., Verbeke, W., Sarraute, C., Baesens, B. and Vanthienen, J. (2017) Social Network Analytics for Churn Prediction in Telco: Model Building, Evaluation and Network Architecture. *Expert Systems with Applications* [online]. 85, pp. 204-220. [Accessed 24 October 2018].

ÓSkarsdóttir, M., Bravo, C., Verbeke, W., Sarraute, C., Baesens, B. and Vanthienen, J. (2016) A Comparative Study of Social Network Classifiers for Predicting Churn in the Telecommunication Industry. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* [online]., pp. 1151-1158. [Accessed 02 November 2018].

- Owczarczuk, M. (2010) Churn Models for Prepaid Customers in the Cellular Telecommunication Industry Using Large Data Marts. *Expert Systems with Applications* [online]. 37 (6), pp. 4710-4712. [Accessed 27 October 2018].
- Reichheld, F.F. and Kenny, D.W. (1990) The Hidden Advantages of Customer Retention. *Journal of Retail Banking* [online]., p. 19+. [Accessed 10 October 2018].
- Reinartz, W. J and Kumar, V. (2003) The Impact of Customer Relationship Characteristics on Profitable Lifetime Duration. *Journal of Marketing* [online]. 67 (1), pp. 77-99. [Accessed 15 October 2018].
- Shrikhande, P A and Verma, A. (2018) Survey on Customer Churn Analysis in Telecom Sector Using Decision Tree. *Journal of Applied Science and Computations* [online]. 5 (9), pp. 330-335. [Accessed 08 October 2018].
- Soeini, R.A. and Rodpysh, K.V. (2012) Applying Data Mining to Insurance Customer Churn Management. *International Proceedings of Computer Science and Information Technology* [online]. 30, pp. 82-92. [Accessed 17 November 2018].
- Sommerville, I. (2016) *Software Engineering*. Tenth edition, Global edition ed. Pearson Education.
- Tsai, C and Lu, Y. (2009) Customer Churn Prediction by Hybrid Neural Networks. *Expert Systems with Applications* [online]. 36 (10), pp. 12547-12553. [Accessed 01 November 2018].
- Umayaparvathi, V. and Iyakutti, K. (2016) A Survey on Customer Churn Prediction in Telecom Industry: Datasets, Methods and Metrics. *International Research Journal of Engineering and Technology* [online]. 3 (4), pp. 1065-1070. [Accessed 15 October 2018].
- Van Den Poel, D. and Larivière, B. (2004) Customer Attrition Analysis for Financial Services Using Proportional Hazard Models. *European Journal of Operational Research* [online]. 157 (1), pp. 196-217. [Accessed 10 November 2018].
- Wei, C.-P. and Chiu, I.-T. (2002) Turning Telecommunications Call Details to Churn Prediction: A Data Mining Approach. *Expert Systems with Applications* [online]. 23 (2), pp. 103-112. [Accessed 15 October 2018].
- Zollo, M. and Meier, D. (2008) What Is M&A Performance? *Academy of Management Perspectives* [online]. 22 (3), pp. 55-77. [Accessed 07 November 2018].

## 9.2 Software and Libraries

Bootstrap (2019) [computer program]. Available from: <https://getbootstrap.com/> [Accessed 20 November 2018].

Chart.js (2019) [computer program]. Available from: <https://www.chartjs.org/> [Accessed 20 November 2018].

Colormind (2018) [computer program]. Available from: <http://colormind.io/> [Accessed 9 November 2018].

Coolors (2018) [computer program]. Available from: <https://coolors.co/> [Accessed 9 November 2018].

FontAwesome (2019) [computer program]. Available from: <https://fontawesome.com/> [Accessed 20 November 2018].

Gravit (2018) [computer program]. Available from: <https://gravit.io/> [Accessed 9 October 2018].

Keras (2019) [computer program]. Available from: <https://keras.io/> [Accessed 10 November 2018].

Lucidchart (2018) [computer program]. Available from: <https://www.lucidchart.com> [Accessed 8 October 2018].

Netlify Identity (2019) [computer program]. Available from: <https://www.netlify.com/docs/identity/> [Accessed 14 December 2018].

Node.js (2019) [computer program]. Available from: <https://nodejs.org/en/> [Accessed 20 November 2018].

NPM. Bcrypt (2019) [computer program]. Available from: <https://www.npmjs.com/package/bcrypt> [Accessed 18 November 2018].

NPM. SqlString (2018) [computer program]. Available from: <https://www.npmjs.com/package/sqlstring> [Accessed 18 November 2018].

Oracle. MySQL Workbench 8.0 (2018) [computer program]. Available from: <https://www.mysql.com/products/workbench/> [Accessed 10 December 2018].



React.js (2019) [computer program]. Available from: <https://reactjs.org/> [Accessed 20 November 2018].

Scikit-Learn (2019) [computer program]. Available from: <https://scikit-learn.org/stable/> [Accessed 10 November 2018].

TeamGantt (2018) [computer program]. Available from: <https://www.teamgantt.com/> [Accessed 8 October 2018].

Tensorflow.js (2019) [computer program]. Available from: <https://www.tensorflow.org/js> [Accessed 10 November 2018].

## 10 Appendix 1: Test Cases

### 10.1 User Actions

ID	Test	Steps	Expected Outcome	Actual Outcome	Pass / Fail
UA1	Login in with correct credentials.	<ol style="list-style-type: none"> <li>1. Type correct username and password into login view.</li> <li>2. Click login.</li> </ol>	User logged in successfully.	User logged in successfully.	Pass
UA2	Login with incorrect credentials.	<ol style="list-style-type: none"> <li>1. Type incorrect username and password into login view.</li> <li>2. Click login.</li> </ol>	User prevented from logging in an error message displayed.	User prevented from logging in an error message displayed.	Pass
UA3	Log in while the server is down.	<ol style="list-style-type: none"> <li>1. Disable server.</li> <li>2. Type correct username and password into login view.</li> <li>3. Click login.</li> </ol>	User prevented from logging in an error message displayed.	User prevented from logging in an error message displayed.	Pass
UA4	Login without password.	<ol style="list-style-type: none"> <li>1. Type correct username, leaving password field empty.</li> <li>2. Click login.</li> </ol>	User prevented from logging in an error message displayed.	User prevented from logging in an error message displayed.	Pass
UA5	Log out.	<ol style="list-style-type: none"> <li>1. Hover over user's name on the navigation bar.</li> <li>2. Click logout.</li> </ol>	User logged out and redirected to login page.	User logged out and redirected to login page.	Pass

<b>UA6</b>	Edit name	<ol style="list-style-type: none"> <li>1. Navigate to user profile view.</li> <li>2. Change value in name field.</li> <li>3. Enter password</li> <li>3. Click save changes.</li> </ol>	Changes are saved successfully, and confirmation message appears.	Changes are saved successfully, and confirmation message appears.	Pass
<b>UA7</b>	Edit password	<ol style="list-style-type: none"> <li>1. Navigate to user profile view.</li> <li>2. Input current password and new password.</li> <li>4. Save changes.</li> </ol>	Changes are saved successfully, and confirmation message appears.	Changes are saved successfully, and confirmation message appears.	Pass
<b>UA8</b>	Save changes with no password.	<ol style="list-style-type: none"> <li>1. Navigate to user profile view.</li> <li>2. Change value in name field.</li> <li>3. Click save changes.</li> </ol>	Error message displayed and changes not saved.	Error message displayed and changes not saved.	Pass
<b>UA9</b>	Save changes with incorrect password.	<ol style="list-style-type: none"> <li>1. Navigate to user profile view.</li> <li>2. Change value in name field.</li> <li>3. Enter incorrect password.</li> <li>3. Click save changes.</li> </ol>	Error message displayed and changes not saved.	Error message displayed and changes not saved.	Pass
<b>UA10</b>	Resume session.	<ol style="list-style-type: none"> <li>1. Log in.</li> <li>2. Close the browser/tab with the application running.</li> <li>3. Navigate back to the application.</li> </ol>	User remains logged in and the session resumes.	User remains logged in and the session resumes.	Pass

## 10.2 Navigation

ID	Test	Steps	Expected Outcome	Actual Outcome	Pass / Fail
N1	Navigate to the dashboard view.	1. Click dashboard menu item.	Redirected to the dashboard view.	Redirected to the dashboard view.	Pass
N2	Navigate to the predictions view.	1. Click predictions menu item.	Redirected to the predictions view.	Redirected to the predictions view.	Pass
N3	Navigate to the customer profile view.	1. Click customer profile menu item.	Redirected to the customer profile view.	Redirected to the customer profile view.	Pass
N4	Navigate to the help view.	1. Click the help menu item.	Redirect to the help view.	Redirect to the help view.	Pass
N5	Navigate to the admin view as an admin.	1. Login as an admin. 2. Click the admin menu item.	Redirected to the admin view.	Redirected to the admin view.	Pass
N6	Attempt to navigate to the admin page as a regular user.	1. Log in as a regular user. 2. Navigate to <a href="http://localhost:3000/#/admin">http://localhost:3000/#/admin</a> in the URL.	Redirect to the dashboard view.	<del>Able to view admin page.</del> Redirect to the dashboard view.	<del>Fail</del> Pass
N7	Navigate to the user's profile.	1. Hover over user's name on the navigation bar.	Redirect to user profile view.	Redirect to user profile view.	Pass

		2. Click the user profile link.			
<b>N8</b>	Attempt to navigate to any view while not logged in.	1. Logout current user. 2. Navigate to <a href="http://localhost:3000/#/predictions">http://localhost:3000/#/predictions</a> in the URL.	Remains on login page.	Remains on login page.	Pass

### 10.3 Generating Predictions

<b>ID</b>	<b>Test</b>	<b>Steps</b>	<b>Expected Outcome</b>	<b>Actual Outcome</b>	<b>Pass / Fail</b>
<b>GP1</b>	Select model.	1. Navigate to train model view. 2. Select model in models list.	Model should be selected.	Model should be selected.	Pass
<b>GP2</b>	Select features.	1. Navigate to train model view. 2. Click on features in features list.	Features become selected.	Features become selected.	Pass
<b>GP3</b>	Adjust model parameters using numeric values.	1. Navigate to train model view. 2. Select a model from the model list. 3. Input numeric values into model parameter options.	Successful input. No error message displayed.	Successful input. No error message displayed.	Pass

<b>GP4</b>	Adjust model parameters using string values.	<ol style="list-style-type: none"> <li>1. Navigate to train model view.</li> <li>2. Select a model from the model list.</li> <li>3. Input character into the model parameter options.</li> </ol>	Error message displayed.	Error message displayed.	Pass
<b>GP5</b>	Adjusting model parameters using empty values.	<ol style="list-style-type: none"> <li>1. Navigate to train model view.</li> <li>2. Select a model from the models list.</li> <li>3. Select at least one feature.</li> <li>4. Remove model parameter values.</li> <li>5. Click train.</li> </ol>	Error message displayed and model training does not commence.	Error message displayed and model training does not commence.	Pass
<b>GP6</b>	Train with model selected	<ol style="list-style-type: none"> <li>1. Navigate to train model view.</li> <li>2. Select a model from the models list.</li> <li>3. Select at least one feature.</li> <li>4. Click train.</li> </ol>	Model training commences and progress bar updates.	Model training commences and progress bar updates.	Pass
<b>GP7</b>	Train with no model selected	<ol style="list-style-type: none"> <li>1. Navigate to train model view.</li> <li>2. Leave all models unselected.</li> <li>3. Select at least one feature.</li> <li>4. Click train.</li> </ol>	Error message displayed and model training does not commence.	<del>Server error.</del> Error message displayed and model training does not commence.	<del>Fail</del> Pass
<b>GP8</b>	Train with no features selected.	<ol style="list-style-type: none"> <li>1. Navigate to train model view.</li> <li>2. Select a model from the models list.</li> </ol>	Error message displayed and model training does not commence.	<del>Server error.</del> Error message displayed and model training	<del>Fail</del> Pass

		3. Do not select any features. 4. Click train.		does not commence.	
<b>GP9</b>	Select multiple models.	1. Navigate to train model view. 2. Select multiple models in models list.	Only one model remains selected.	Only one model remains selected.	Pass

## 10.4 Viewing Results

ID	Test	Steps	Expected Outcome	Actual Outcome	Pass / Fail
<b>VR1</b>	View previous models results.	1. Navigate to train model view.	Results displayed on a table in the view.	Results displayed on a table in the view.	Pass
<b>VR2</b>	View a prediction score for a customer.	1. Navigate to customer profile view.	Prediction score displayed on chart.	Prediction score displayed on chart.	Pass
<b>VR3</b>	View the contributions towards a customer prediction score.	1. Navigate to customer profile view.	Contributions towards the score are displayed on a chart.	Contributions towards the score are displayed on a chart.	Pass (RF & DT)
<b>VR4</b>	Search for a customer to view their results.	1. Navigate to customer profile page. 2. Ensure date selected in date drop down field is correct.	Customer profile displayed.	Customer profile displayed.	Pass

		3. Type existing customer number in search fields on view banner and press enter.			
<b>VR5</b>	Select date of a previous prediction to view previous results.	1. Navigate to customer profile page. 2. Ensure date selected in date drop down field is correct. 3. Type existing customer number in search fields on view banner and press enter. 4. After customer profile is displayed, change date in date drop down field.	Results change to results of selected date.	Results change to results of selected date.	Pass
<b>VR6</b>	Search for customer that does not exist.	1. Navigate to customer profile page. 2. Ensure date selected in date drop down field is correct. 3. Type invalid customer number into the customer search field and press enter.	Error message displayed.	Error message displayed.	Pass

## 10.5 Admin

ID	Test	Steps	Expected Outcome	Actual Outcome	Pass / Fail
----	------	-------	------------------	----------------	-------------



<b>A1</b>	Add a new user.	<ol style="list-style-type: none"> <li>1. Navigate to admin view.</li> <li>2. Enter the new users details into the form.</li> <li>3. Submit form.</li> </ol>	New user is created.	New user is created.	Pass
<b>A2</b>	Edit a user's details.	<ol style="list-style-type: none"> <li>1. Navigate to admin view.</li> <li>2. Double click cell of user in table that should be changed.</li> <li>3. Delete value and type new value.</li> <li>4. Click off cell.</li> </ol>	<p>Users details changed.</p> <p>Success message displayed.</p>	<p>Users details changed.</p> <p>Success message displayed.</p>	Pass
<b>A3</b>	Lock out a user.	<ol style="list-style-type: none"> <li>1. Navigate to admin view.</li> <li>2. Double click the 'locked' cell of the user to be locked out.</li> <li>3. Enter '1' in place of the '0'.</li> <li>4. Click off the cell.</li> <li>5. Attempt to log in as the locked-out user.</li> </ol>	User should be denied access when attempting to log in.	User should be denied access when attempting to log in.	Pass
<b>A4</b>	Unlock users account.	<ol style="list-style-type: none"> <li>1. Navigate to admin view.</li> <li>2. Double click the 'locked' cell of the user to be unlocked.</li> <li>3. Enter '0' in place of the '1'.</li> <li>4. Click off the cell.</li> <li>5. Attempt to log in as the locked-out user.</li> </ol>	The user should be allowed access if their credentials are correct.	The user should be allowed access if their credentials are correct.	Pass

<b>A5</b>	Add a new question and answer to the help view.	<ol style="list-style-type: none"> <li>1. Navigate to the help view.</li> <li>2. Click the '+' icon on the view banner.</li> <li>3. Fill out the question and answer.</li> <li>4. Submit the form.</li> </ol>	The new question and answer should immediately appear at the bottom of the list.	The new question and answer should immediately appear at the bottom of the list.	Pass
<b>A6</b>	Remove a question from the help view.	<ol style="list-style-type: none"> <li>1. Navigate to the help view.</li> <li>2. Click a questions 'trash' icon.</li> </ol>	The question should instantly be removed from the view.	The question should instantly be removed from the view.	Pass

## 10.6 Notifications

<b>ID</b>	<b>Test</b>	<b>Steps</b>	<b>Expected Outcome</b>	<b>Actual Outcome</b>	<b>Pass / Fail</b>
<b>N1</b>	Generate notification	<ol style="list-style-type: none"> <li>1. Navigate to train model view.</li> <li>2. Select a model a model and features.</li> <li>3. Click train.</li> </ol>	When results have been generated, the user should receive a notification.	Nothing.	Fail
<b>N2</b>	View notification	<ol style="list-style-type: none"> <li>1. Hover over notifications icon on the navigation bar.</li> </ol>	Notifications should be displayed.	Nothing.	Fail
<b>N3</b>	Dismiss notification	<ol style="list-style-type: none"> <li>1. Hover over notifications icon on the navigation bar.</li> <li>2. Click the 'x' button on the notification you wish to dismiss.</li> </ol>	The notification should disappear.	Nothing.	Fail