SCHOOL OF MATHEMATICAL & COMPUTER SCIENCES

DEPARTMENT OF Actuarial Mathematics and Statistics

<u>DISSERTATION SUBMISSION</u>

Please complete all of the following:

Programme	BSc (Hons) Statistical Data Science
Year of Study	Year 3, Malaysia, 2021-22, Semester 2
Surname	Tan
First Name	Yen Shen
ID No	H00336563
Course Code	F70DB
Course Title	Statistics Dissertation B
Title of Project	Customer Behaviour Analytics based on Machine Learning Classification Techniques
Supervisor(s)	Supervisor – Sarat Dass Co-supervisor – Benjamin Choong
Date of Submission	28th March 2022 (28/3/22)

I affirm that this dissertation is my own work, and I have not copied material without giving adequate references.	**Signed**_________Tan Yen Shen ____________ **Dated**_______ *28th March 2022 (28/3/22)*____

Student Declaration of Authorship

Course code and name:	F70DB Statistics Dissertation B
Type of assessment:	**Individual**
Coursework Title:	Customer Behaviour Analytics based on Machine Learning Classification Techniques
Student Name:	Tan Yen Shen
Student ID Number:	H00336563

Student Signature *(Tan Yen Shen):* *Tan Yen Shen*

Date: *28th March 2022 (28/3/22)*

Acknowledgements

I would like to express my deepest gratitude to Dr Sarat and Benjamin Choong for guiding me in this project. It has been a journey of growth for me.

Thank you, Dr Sarat, for your patience, consistent guidance and knowledge.

Thank you, Ben, for your never-ending support and insights.

Table of Contents

Abstract

Customer churn is an important concept for businesses to understand and interpret to make informed decisions. The purpose of churning observed from businesses is that customers leave and choose another provider causing a loss in businesses. In this thesis, we touch on the fundamentals of customer churn, the variables that do and do not affect churn, and we introduce machine learning models and fit Telcos dataset to these models.

The main objective of this project is to find which is the best and most optimal model to use for this Telcos dataset. The machine learning techniques that have been introduced are Decision Tree, Random Forest and Neural Network. We will then compare the performance metrics of these models and discuss which is the most optimal machine learning model.

Keywords: Customer Churn, Decision Tree, Random Forest, Neural Network

1.Introduction

1.1 Basic Concepts

1.12 Customer Churn

The biggest element that businesses look into on determining the growth and competition of the company is customer churn. Firstly, businesses acquire new customers, and then engage with them at satisfactory levels of service in hopes to retain them as existing or retaining customers. It has been revealed that gaining new customers results to be more expensive, in comparison to maintaining and retaining existing customers. (J.Hadden 2006).

In Berson et.al (2000) and Tan Y.S (2021), 'customer churn' is a term used in the wireless telecom service industry that conveys customer movement from one provider to another. 'Customer management' is a term that describes a company's process to retain profitable customers. In a business model, the company segments customers based on profitability and focuses on retaining only those that are profitable. However the telecom service industry is yet to standardise a set of 'profitability' measurements.(see, for example, Hung et.al (2006) For example, current versus life-time, business unit versus corporate, etc.)

Voluntary and involuntary churners are mainly the two types of customer churn. Voluntary churners, also known as active churners, are customers who willingly choose to quit the service and move to the next service. On the contrary, involuntary churners are customers who the company chooses to have their services terminated or suspended by the company, typically due to poor payment history (Blattberg et al 2008). However, when referring to the customer churn problem, we are generally relating to the voluntary churners.

As a result of customer churn, there are switching costs (Klemperer 1988). Switching costs include transaction cost (financial) and learning cost (non-financial). A typical scenario of the former switching cost is due to the availability of cheaper but alternative identical products. Similarly, An example of the latter switching cost is having to re-learn a new product, and the time required for that could come as a cost.

1.13 How is Customer Churn calculated?

Customer churn is focused on the retention element, *r.* At the customer level, churn refers to the probability the customer leaves the company in a given time period. At the company level, churn is the percentage of the company's customer base that leaves in a given time period. Churn is thus one minus the retention rate:

$$\text{Churn} = c = 1 - \text{Retention Rate} = 1 - \text{r}$$

For businesses, the retention rate is defined as the percentage of customers that the business retains over a period of time. In other words, this percentage represents the loyal customers of the business.

The formula for retention rate (Tan Y.S (2021), and Robert et al (2008)) is as follows

$$\text{Retention Rate} = \frac{\textit{Number of customers who continue business for this given time period}}{\textit{Total number of customers at the beginning of that period}} \times 100$$

1.14 Why is it important to calculate Customer Churn?

The elements of churn and retention rate are very important factors in determining the lifetime value (LTV) of a customer of that business. This in turn determines how viable the business is and its growth. The Lifetime Value (LTV) is termed as 'how much revenue a customer is estimated to deliver across their entire time buying from the business'.

Robert et.al (2008) uses a simple retention model where the lifetime value of a customer is:

$$LTV = \sum_{t=1}^{\infty} \frac{m_t r^{t-1}}{(1+\delta)^{t-1}}$$

with:

$$m = Annual\ profit\ contribution\ per\ customer$$
$$c = Annual\ churn\ rate$$
$$\delta = Annual\ discount\ rate$$
$$r = retention\ rate$$
$$t = time$$

Companies that have a subscription service, like telecommunications companies, have a standardised way of measuring customers' value, which is termed as the Average Revenue Per User (ARPU). The calculation for ARPU would be calculated as

$$ARPU = Total\ Revenue\ /\ Average\ Customers$$

Despite this being a way for companies to measure the value of their customers, note that churn does not significantly affect ARPU. Instead, reducing churn will increase LTV. By decreasing the rate at which customers leave the service, we increase the probability of them staying in the service longer.

1.2 Objective

For Semester 2 goal, we shall introduce intelligent and scientific models that allow interpretation and processing of the data. It allows us to optimise and make sense of the data based on the given techniques, benefits and disadvantages in each model. It extracts insights for businesses to make better and informed decisions by providing knowledge and understanding with meaningful results.

Based on the above, my Semester 2 goal is to explore the machine learning techniques given the Telcos dataset:

1. Decision Tree
2. Random Forest
3. Neural Network

Decision Tree machine learning model and algorithm is chosen due to its easy interpretability: a decision tree is like a flowchart which enables the user to see the consequences of their decisions as one moves down along the nodes of the tree. However, decision trees have some disadvantages - they may be susceptible to low predictive accuracy rates and overfitting (Gareth and et al, 2013). Consequently, Random Forests are introduced to overcome the aforementioned disadvantage of Decision Trees. The Neural Network machine learning model is chosen since it is a more enhanced model compared to Logistic Regression in Semester 1 with hidden layers and weights to improve the prediction accuracy.

Using these techniques, we will explore their test accuracies and compare them alongside the Logistic Regression technique that was applied in Semester 1. In this thesis, it is important to ask ourselves in the beginning - what is this analysis for, and how is it different from our approach in Semester 1?

2. Literature Review

There have been many studies conducted based on churn prediction in the mobile telecom industry. The machine learning techniques and data mining strategies applied in these studies range from Naive Bayes to Support Vector Machine. However, for the current literature review, we shall solely focus on the research done based on the three machine learning techniques explored in this study, primarily Decision Trees, Random Forest and Neural Networks.

Hung, Yen and Wang (2006) analysed a dataset from Taiwan's telecom company as an application of the Decision Trees and Neural Network back propagation approach. It showed that both techniques delivered accurate results, with the Neural Network performing better than the Decision Tree.

Burez, Van and Poel (2009) explored the techniques of Logistic Regression and Random Forest for a telecom dataset. This paper also explored the situations of class imbalance and methods like oversampling and undersampling that can be used to tackle that. When comparing the two models used, Random Forest outperforms Logistic Regression.

A study conducted by Sharma and Kumar (2011) used a neural network based approach to predict customer churn in a cellular network service. The outcome of the research for the neutral network obtained a 92.35% predicted accuracy.

There are other machine learning techniques explored in other papers like Bayesian Network and Naive Bayes (Kirui et al, 2013) and Support Vector Machine (Chen, Fan and Sun, 2012), used to explore churn prediction.

From observing the research conducted above, it is usually the case where one or two machine learning techniques are used to explore the given dataset. In this given research, we will analyse the Telecom dataset using three machine learning techniques of Decision Trees, Random Forest and Neural Networks and compare their performance on test accuracy.

3. Data preparation and feature selection

3.1 Content of Telco Customer Churn data

Each row of the Telco dataset represents an individual customer of Telco represents each row, and the customer's attributes are depicted by each column of the data matrix.

The data set includes information about:

- Customers who left within the last month – this column is called Churn.
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies.
- Customer account information – how long they've been a customer- tenure, the type of contract, payment method, paperless billing, monthly charges, and the accumulation of the total charges.
- Demographic about customers – gender, whether they are seniors, and if they have partners and dependents.

No	**Variable**	**Values**	**Type**	**Affected Churn?**
1	customerID	7043 unique values	Character	-
2	gender	**(Male, Female)**	Factor	No
3	SeniorCitizen	**(1,0)** Whether the customer is a Senior Citizen or not	Factor	Yes
4	Partner	**(Yes, No)** Whether customer has a Partner or not	Factor	Yes
5	Dependents	**(Yes, No)** Whether customer has Dependents or not	Factor	Yes
6	Tenure	Number of months the customer has stayed with the company	Numerical	Yes
7	PhoneService	**(Yes, No)** Whether customer has a PhoneService or not	Factor	No
8	MultipleLines	**(Yes, No, No phone service)** Whether customer has a PhoneService or not	Factor	Yes
9	InternetService	**(DSL, Fibre Optic, No)** No means No Internet Service Customer's Internet Service Provider	Factor	Yes
10	OnlineSecurity	**(Yes, No, No internet service)** Whether the customer has online security or not	Factor	Yes

11	OnlineBackup	**(Yes, No, No internet service)** Whether the customer has online backup or not	Factor	Yes
12	DeviceProtection	**(Yes, No, No internet service)** Whether the customer has tech support or not	Factor	Yes
13	TechSupport	**(Yes, No, No internet service)** Whether the customer has tech support or not	Factor	Yes
14	StreamingTV	(**Yes, No, No internet service)** Whether the customer has streaming tv or not	Factor	Yes
15	StreamingMovies	**(Yes, No, No internet service)** Whether the customer has streaming movies or not	Factor	Yes
16	Contract	**(Month – to- month, One year, Two year)** The contract term of the customer	Factor	Yes
17	PaperlessBilling	(**Yes, No)** Whether the customer has paperless billing or not	Factor	Yes
18	PaymentMethod	**(Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))** The customer's payment method	Factor	Yes
19	MonthlyCharges	The amount charged to the customer monthly	Numerical	Yes
20	TotalCharges	The **total** amount charged to the customer	Numerical	Yes
21	Churn	**(Yes, No)** Whether the customer churned or not	Factor	Yes

Table 3.1 21 Variables value, type and whether they affected churn or not

In Semester 1, obtaining a deep understanding of the Telcos dataset, and the steps of data preparation and feature selection were completed. From there, we performed exploratory data analysis and found the variables which affected and did not affect churn.

We conducted statistical tests on each variable with the Churn variable as the dependent variable. These tests included the Chi-squared, Komolgorov Smirnov, and Mann -Whitney U test based on whether the variables were numerical or categorical, with the objective of determining whether they were significant predictors of Churn. To summarise, we found Gender and Phone Service to be variables that did not affect churn. The remaining 17 variables, excluding customerID, affected Churn. The results are tabulated in Table 3.1 above.

We also observed that variables like Paperless Billing and Total Charges are not predictor variables of Churn, but rather response variables. This is because these 2 variables are dependent on other variables such as Senior Citizen and Monthly Charges.

From this understanding about the variables and their influence on Churn, we advised businesses to focus on factors that do affect Churn. These factors also include specific variables that can not be controlled from a business perspective, such as certain demographic factors (for e.g we can't ask Senior Citizen's to be younger). However, highlighting these variables allows us to provide guidance to customers belonging to such groups to optimise their behaviour to minimise expenditure (e.g reducing their spending based on which type of billing is favourable to them)

4. Methodology

4.1 Data Processing

Situations with a quantitative (numerical) response, are referred to as regression problems, while those with a qualitative (categorical) response, are referred to as classification problems. (Referenced 'Introduction of Statistical Machine Learning' (Gareth and et al, 2013))

In this thesis, we will be using the classification method since our dependent variable, Churn, outputs 'Yes' or 'No'. For classification, predicting a qualitative response for an observation can be referred to as classifying that observation, since it involves assigning the observation to a category or class. There are many different types of classification techniques, giving rise to classifiers, that would be used to predict a qualitative response. The classifiers that we will be exploring in this thesis are decision trees, random forest and neural networks.

Apart from choosing a statistical machine learning model based on whether the response is quantitative or qualitative, it is also important to ensure that the explanatory variables used to predict the response are properly coded before the analysis is performed. For example, for logistic regression, we made sure that the variables which are in factor mode are converted to numerical types before analysis. For neural networks (explored much later in this thesis), we made sure to convert the categorical variables to the numerical values of '1's and '0's where this conversion is known as One-Hot-Encoding.

Next, we partitioned the dataset into test and training parts before fitting them into the classifier. The training set is used to train and build the model, and then very lastly, we use it to predict using the test data to see whether the model has accurately output the given results of customer churns.

4.2 Overview of Methodology and Packages

One of the biggest obstacles when fitting these models is overfitting (Gareth and et al (2013)). It happens when it takes the data that is used to train, and trains it too well. But when you try to introduce another dataset that is completely different from the initial dataset, it doesn't fit it as well. Because it learns it too well on the tiny patterns for the initial dataset and therefore didn't do a good job in generalising the model. This obstacle is typically found in decision trees and neural networks and to be mentioned more in Section 4.4 and 4.6 respectively.

We then introduce Random Forest in Section 4.5 which uses the bootstrap method to combat overfitting. Referenced from Gareth and et al (2013), by mathematical situation, with a given set of independent observations $Z_1, \dots Z_n$, each with variance σ^2. To find the variance of the mean of $\overline{Z}$ (of the overall observations), it is derived as σ^2/n. To put it in another way, averaging from a set of observations reduces variance. Reducing variance, results in an increase in test set accuracy results. Therefore, this results in an increase in the bias and boosts the performance of the final model.

As mentioned earlier, overfitting may be one of the biggest problems when it comes to fitting models. So for the Neural Network, we have introduced weight decay as a tool to generalise the model to improve the fitting and accuracy of the model (Anders K and John A.K (1991). More will be discussed regarding Neural Networks in Section 4.6

The main package in the R language that we will be using is the CARET package, which stands for Classification And REgression Training (Kuhn, M, 2008). It allows us to process and create predictive models by introducing tools such as data splitting, pre-processing, and model training which we will be using throughout this project. Alongside that, we use the ggplot package for graphing and data visualisation, and the dplyr package for dataset manipulations and other mathematical calculations. For the models of decision trees and neural networks, we trained the model using CARET, and additional packages such as rpart and nnet. For random forests, we introduced another package in R called randomForest.

4.3 *k* - Fold Cross Validation

Cross Validation (CV) is a resampling technique. For this technique, it's important that we use the training set. It involves fitting the same statistical method many times using different subsets of the training data. Resampling methods help in reducing overfitting and it helps in finding the best (optimal) hyperparameter settings to increase the efficiency of the model. (Hastie T. et al, 2009)

This approach randomly divides the set of observations into k groups, of approximately equal size if applicable. In this scenario, we have used *k* to be 10. These sets of observations are split into a validation set and a training set. A typical CV approach is illustrated in Figure 4.3 below in a 4-fold CV scheme, the training data is partitioned into 4-folds. The first fold, which is depicted by the yellow box in Figure 4.1, is the validation set, whereas the remaining 3 folds are the training dataset. During the training phase, the model (any one of the Decision Trees, Neural Network or Random Forest) is fitted to the training dataset, and accuracy measures are obtained on the validation dataset. This is then repeated for the subsequent 3 iterations - the remaining *k* - 1 fold, as shown as the blue boxes in Figure 4.3.

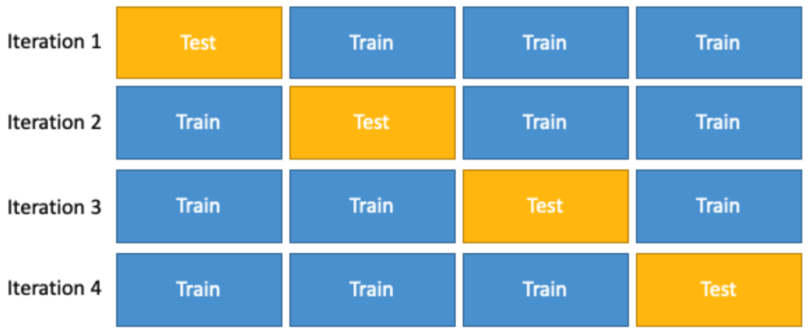

Figure 4.3 Illustratration of the *k*-fold CV approach

Thus, the *k*-fold CV procedure is repeated k times, with a different group of observations treated as a validation set each time. For a qualitative response, this results in k estimates of the test error, MSE_1, MSE_2, ..., MSE_k. Then, the k-fold CV estimate is calculated by finding an average of these values. This formula (Hastie T. et al, 2009) is depicted below as,

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

However, since we have a classification problem, we use the number of misclassified observations instead. The formula is shown as below,

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} Err_i$$

where $Err_i = I(y_i \neq \hat{y}_i)$

For our machine learning techniques in this thesis, we applied the *k*-fold cross validation method to only decision trees and neural networks. For random forest, the method in itself mirrors a bootstrap method, to which more will be described in its own section below.

4.4 Decision Tree based - Classification

A decision tree is like a flowchart which enables the user to see the consequences of their decisions as one moves down along the nodes of the tree. Decision tree based classification method is one of the most powerful and easily-interpretable machine learning techniques that can be used for both classification (qualitative response) and regression (quantitative response).For this given data set, we will be using the approach of classification tree since we are predicting a qualitative response.

To first understand the ways in which a tree can be split into branches, we provide several measures used to evaluate the split of the tree. We mention the two methods below as approaches to splitting the trees; Gini index and entropy. The following formulas are referenced from 'Introduction of Statistical Machine Learning' (Gareth and et al, 2013)

For the Gini index, is defined as

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

$\hat{p}_{mk}$, $0 \leq \hat{p}_{mk} \leq 1$. Gini Index is defined as a measure of total variance across the K classes.

Last method is entropy. It is defined as,

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

With the given range of $\hat{p}_{mk}$, $0 \leq \hat{p}_{mk} \leq 1$. To which $0 \leq \hat{p}_{mk} \log \hat{p}_{mk}$. Entropy measures the expected reduction in information caused by partitioning the attributes. Similar to the Gini index, the entropy will take on small values given that the mth node is pure. It's important to note that, Gini index and the entropy are quite similar numerically. However, the difference in these splitting methods is the computational time. As observed by the formulas above, entropy will take a longer time as it uses the logarithm function in its calculations compared to Gini index which does not. (Gareth and et al, 2013) In R, when we built our decision tree model, the default splitting method was Gini. We have also used the splitting method of entropy in R. When comparing the training accuracies for both they are similar.

The benefits of decision trees are that they are easy to explain and understood. With that it is easily interpreted by a non-expert. Another benefit to this method is that decision trees can handle qualitative predictors without the need to create dummy variables like one hot encoding (this method is used in other techniques).However, the disadvantage to this technique is that they do not have the same level of predictive accuracy as other Machine Learning approaches. Also, decision trees are very not robust. In other words, a small change in the data can cause a large change in the final estimated tree. To add on, decision trees are prone to overfitting. The concept of overfitting has been described in detail earlier on.
To improve the case of overfitting in decision trees, we have introduced Random Forest as the next Machine Learning technique to our dataset.

4.5 Random Forest

Random forest Breiman (2001) is a bootstrap method (a resampling method) that utilises a majority vote to predict classes based on the stemming of data from multiple uncorrelated forest of decision trees. In other words, this approach is an ensemble method that combines a group trees' teamwork effort by working on a common goal by improving the performance of a single random tree. Through this ensemble method, cross validation technique was not introduced.

The situation with Decision Trees discussed in Section 4.4 is that they suffer from overfitting. With that in mind, random forest is introduced to combat that issue, by taking an average from a set of observations. This method was explored in greater detail in section 4.2.

The main distinction of random forest that makes this method unique, is the decorrelation of the trees and the ways in which they choose the features for classification.(Gareth and et al, 2013) Random forest uses variable importance. When building a random forest, at each split of the tree, it is not allowed to consider a majority of the available predictors.

As featured in Figure 4.5.1, there are other variables that are deemed of higher importance than others. By decorrelating the trees, random forests force each split to only consider a subset of the variables. To which the splits will not even consider the strong predictor, giving even the less important variables more chance with a higher importance variable. Then, this method takes the average of the trees, resulting in less variability and hence, more reliable.

For classification, we will use $\sqrt{p}$ where p is denoted as the predictors

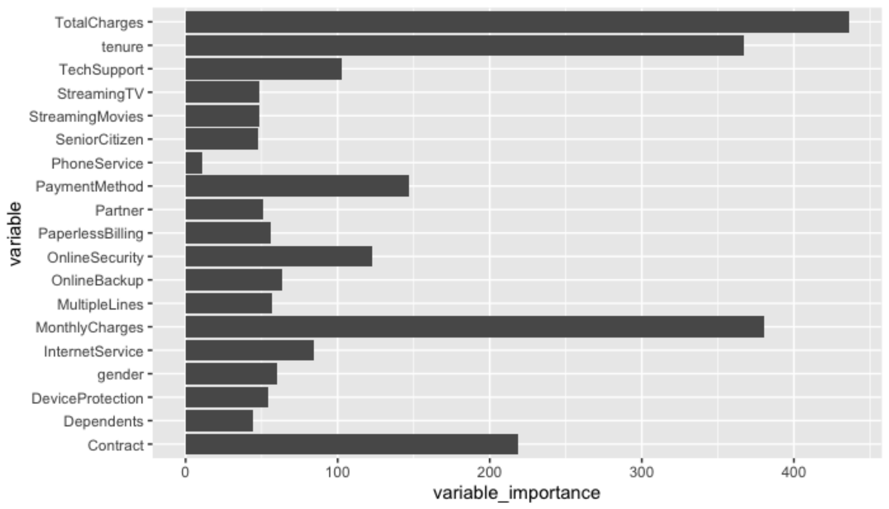

Figure 4.5.1 Variable Importance for Random Forest

As much as random forests has its benefit through the ensemble method and overcoming overfitting, there are a few cons for random forest. First of all, this method may take a very long time to train and have a high cost in runtime, therefore resulting in computational complexity. On top of that, with decision trees deemed as the approach that was easily interpretable, having multiple decision trees takes away the simplicity of the model, and is depicted harder to interpret. (Gareth and et al, 2013)

4.6 Artificial Neural Network

Artificial Neural Network is a complex network that comprises a large set of nodes known as neural cells. This model was introduced to mimic the human biological nervous systems in our nervous system and the human brain that is used to process information and many more. (Markopoulos et al., 2008) The basic structure of a Neural Network consists of multiple associated weighted inputs with a threshold value. When the output of a single node is above the specified threshold value, the node is activated and then sends data to the next layer of the network. Otherwise, no data is passed to the next layer of the network.

Neural networks can be distinguished into single-layer perceptron and multilayer perceptron (MLP) networks. The MLP classifier consists of multiple layers of simple, two state, sigmoid transfer functions that interact by using the weighted inputs. Additional to that, there are intermediate layers knows as hidden layers

(Gareth and et al, 2013) In forward processing, the signals will process through each layer with the sigmoid function $f(x) = (1 + e^{-x})^{-1}$

For classification, we shall use either squared error or cross-entropy (deviance):

$$R(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log f_k(x_i),$$

And the corresponding classifier is $G(x) = argmax_k f_k(x)$

With the softmax activation function and the cross-entropy error function, the neural network model is a linear logistic regression model with additional hidden units, weights added.

As a result of too many weights, neural networks tend to overfit the data. To combat the case of overfitting, a regularisation technique is introduced into the neural networks. This method is known as weight decay(Anders K and John A.K (1991). It is applied to the weights of a neural network. Mathematically, this technique adds a penalty to the error function $R(\theta) + \lambda J(\theta)$, where

$$R(\theta) = \sum_{k,m} \beta^2_{km} + \sum_{m,l} \alpha^2_{ml}$$

And $\lambda > 0$ is a tuning parameter.

4.7 Confusion Matrix and Performance Metric

When assessing the machine learning model, a confusion matrix is introduced to allow us to identify whether the model is doing a good job by comparing the predicted variables to the actual test values. We have the given counts of values tabulated and defined as below.

		Predicted	
		No	Yes
Actual	No	TN	FP
	Yes	FN	FP

Table 4.7.1 Confusion Matrix Table

- True Positive (TP): Positive values are correctly predicted counts
- False Positive (FP): Negative values are incorrectly predicted as positive counts
- False Negative (FN): Positive values predicted as negative counts
- True Negative (TN): Negative values predicted as actual negative value counts

Performance Metrics Formulas

So based on the output of the Confusion Matrix, we can find the performance metrics based on the given values. In deciding whether the machine learning model is performing efficiently, we have the following metrics. We will use this concept in the results section when we compare the models against each other.

Performance Metric	Formula	Definition
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$	The ratio of correctly identified subjects in a pool of subjects.
Sensitivity	$\frac{TP}{TP+FN}$	The ratio of correctly *positive* identified subjects, against all the other subjects that were identified as *positive* in reality.
Specificity	$\frac{TN}{TN+FP}$	The ratio of correctly *negative* identified subjects, against all the other subjects that were identified as negative in reality.
Precision	$\frac{TP}{TP+FP}$	The ratio of correctly *positive* identified subjects, against all the other subjects that were identified as *positive* in test/
F1 Score	$\frac{2(Sensitivity \times Precision)}{Sensitivity + Precision}$	A metric that accounts for precision and sensitivity

Table 4.7.2 Performance Metrics definitions

5. Figures and Tables

This section will be displaying the figures and outputs of the methodologies applied in Section 4. These outputs have been obtained in R.

5.1 Decision Tree

After introducing the two splitting indexes of Gini and Cross entropy, we shall show these outputs in the figures below.

5.12 With Gini Index

The complexity parameter is introduced to control the size of the tree and to select the optimal size of the tree. As defined in section 4.2, Gini Index is defined as a measure of total variance across the *K* classes. It is used to measure node purity - with a smaller index, it indicates that the node contains a particular observation from a single class.

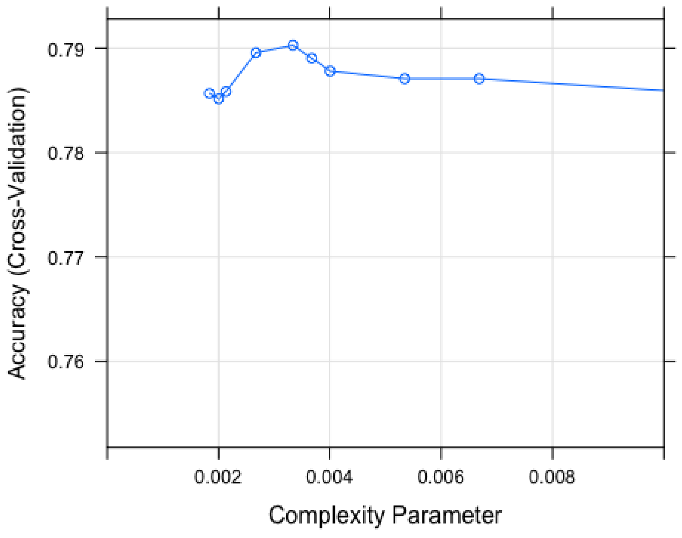

Figure 5.12.1 Complexity Parameter against Training Accuracy with Gini

By observing the graph above, the best tuned model for a decision tree with gini as the splitting index is when the complexity parameter is 0.003342 with the training accuracy attained at 0.7903.

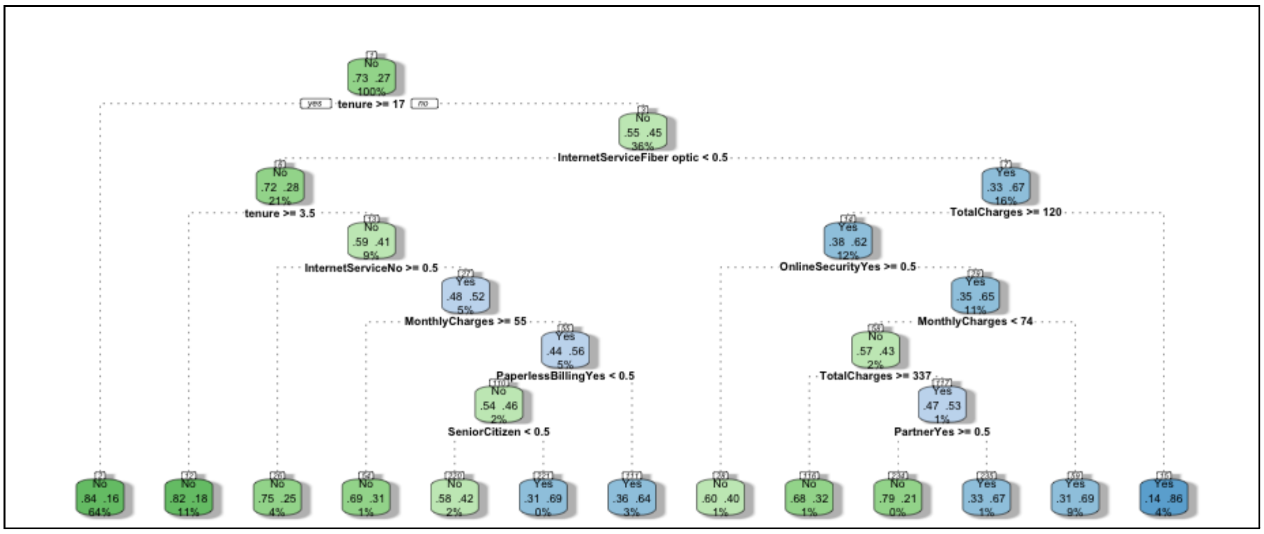

Figure 5.12.2 Decision Tree Graphing with Gini

We start by interpreting the node at the top. At the top, it shows the overall probability of churn rate. With a 73% chance of churning, and a 27% rate of not churning.

The node then asks whether the customer has a tenure year of more than 17 years. If yes, then go down to the left child node, 64% of customers with more than 17 years tenure have a non-churn rate of 84% and a churn rate of 16%. And the node ends there.Now, we start back at the top on whether the customer has more than a 17 years of tenure. If no, then we branch out to the right node. It is observed that 36% of customers with less than 17 years tenure have a 55% of not churning and 45% of churning. With this information provided in the training section, we will output its test accuracy and show it in Section 6, Results.

By this branching alone, we can already see that tenure is a very important feature in predicting churn. We have observed that the relative proportion of customers with more than 17 years tenure has the highest chance of retention. The branching of the tree continues on from there as the decision tree choses the features that impact the likelihood of churning.

5.13 With Cross Entropy

We now know the splitting index of cross entropy. By computation of the cross entropy, it is more complex than the gini as it takes into consideration logarithms. We now plot the complexity parameter and the accuracy rate below.

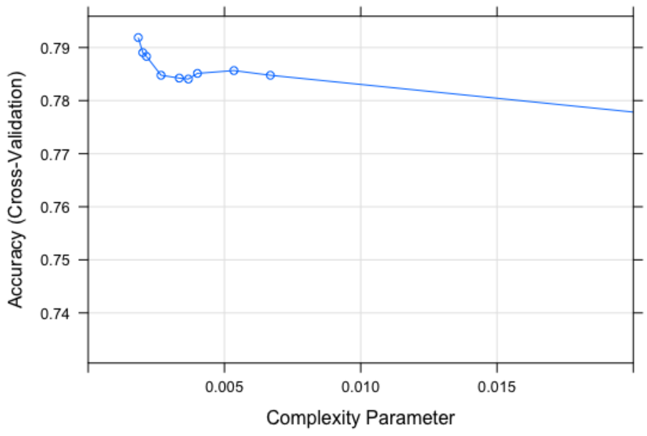

Figure 5.13.1 Complexity Parameter against Training Accuracy with Cross Entropy

By observing the graph above, the best tuned model for a decision tree with gini as the splitting index is when the complexity parameter is 0.001838 with the training accuracy attained at 0.7919. With this information, we will output its test accuracy and show it in Section 6, Results.

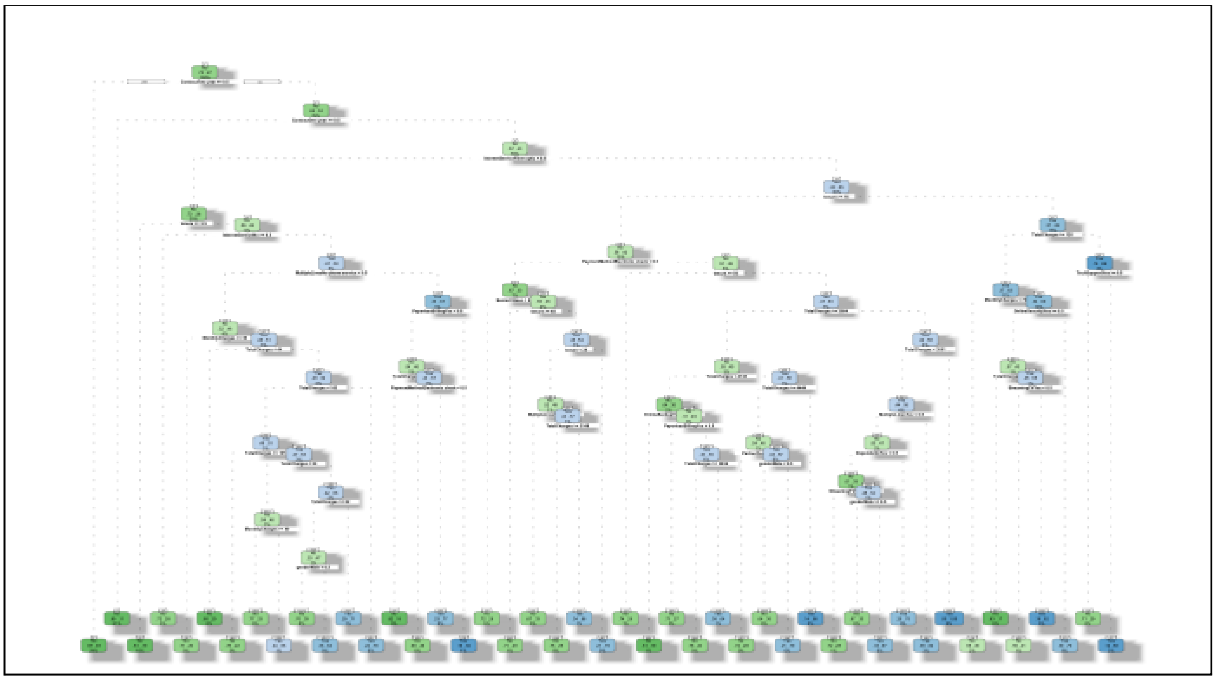

Figure 5.13.1 Decision Tree Graphing with Cross Entropy

When comparing the decision tree graphs of both cross entropy and gini, we have observed that cross entropy takes into consideration more variables for branching that gini. In other words, cross entropy is more informative as it takes into more variables in choosing which variables to branch out of. However, it does take a longer computational time as observed by having more nodes, and the use of log in its formula mentioned in Section 4.4.

5.2 Random Forest with Out of Bag

When processing the data for Random Forest, we took the missing values (N/A set) and applied the Out of Bag (OOB) method to it. We then simulated the dataset with the given 500, and 1,000 trees as below.

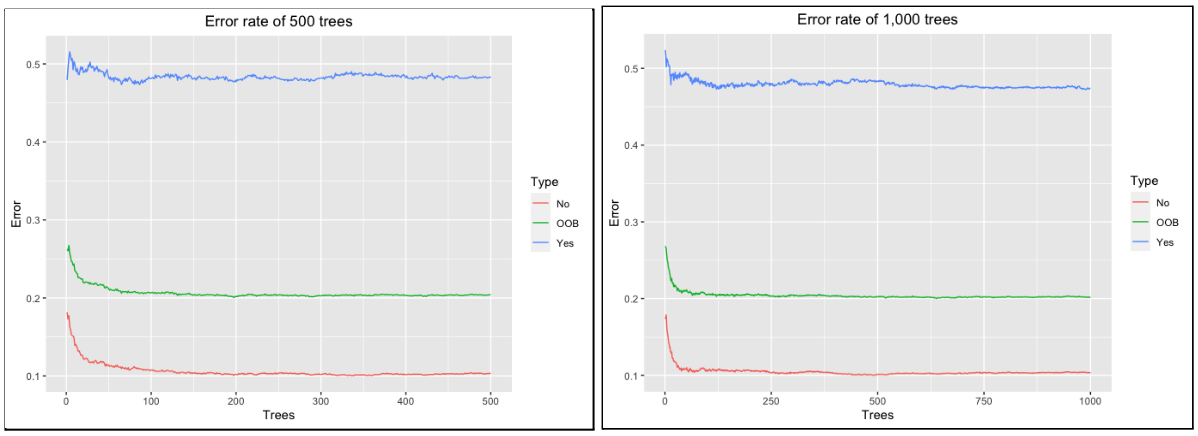

Figure 5.2 Simulation of random forest with 500 and 1,000 trees

Observing the graph above, the red line represents the error rate for calling 'Non-Churn' patients that are OOB. The green is the overall OOB error rate. And the blue line shows specifically the error rate for 'Churn' patients that are OOB. After building a random forest with both 500 and 1,000 trees, the graph makes it clear that it was sufficient with only 500 trees.

From the given output in R for 500 trees, we found the OOB estimate of error rate to be 20.26%. This means that 79.74% of OOB samples were correctly classified in the random forest. Similarly for 1000 trees, we found the OOB estimate of error rate to be 20.29% implying that there were 79.71% correctly classified samples in the random forest. With that in mind, we will output its test accuracy and show it in Section 6, Results.

5.3 Neural Network

We processed the data with one hot encoding and scaled the data. The output of the neural network graphing is shown as below.

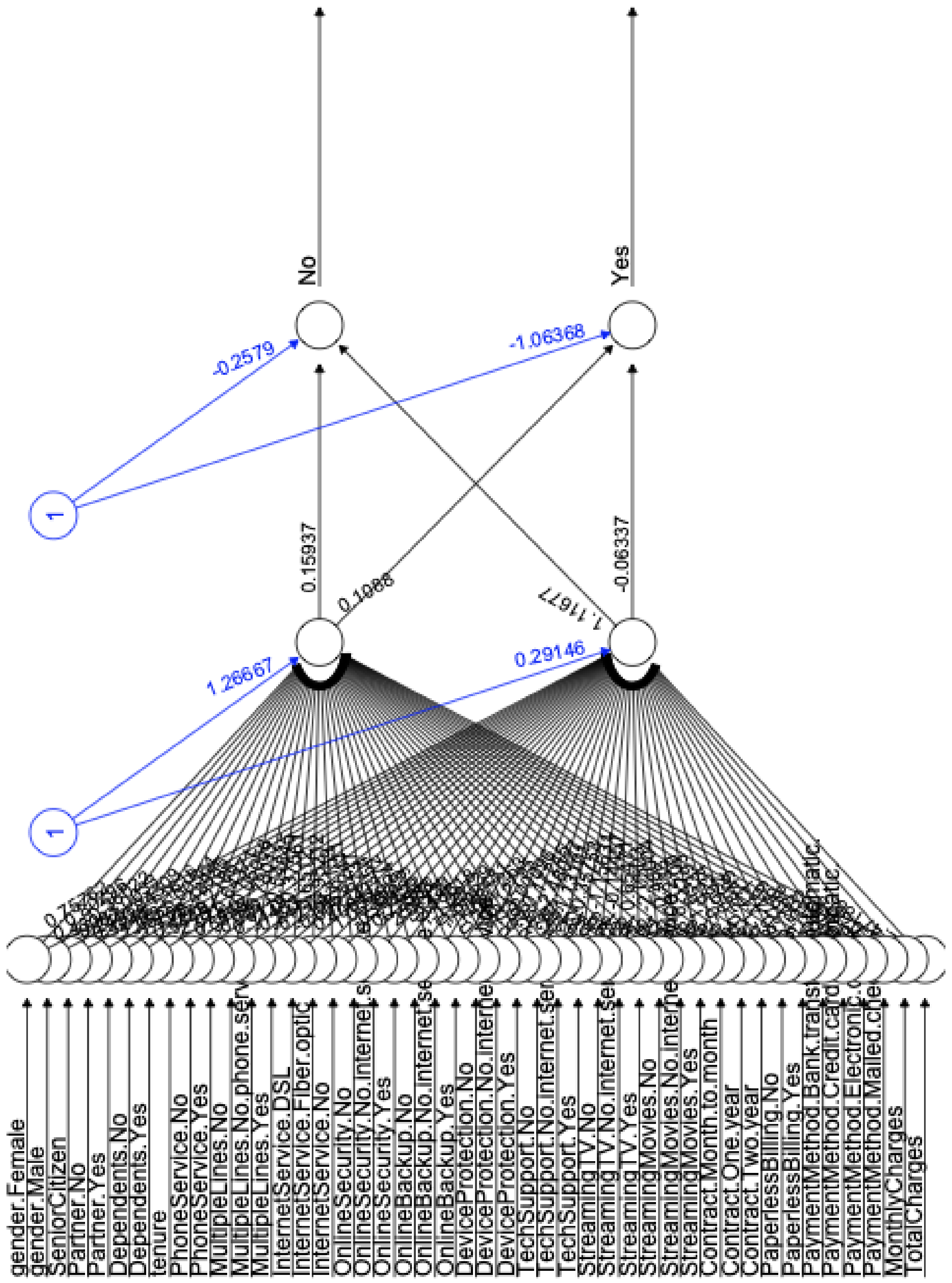

Figure 5.3.1 Neural Network graph fit with Telcos data

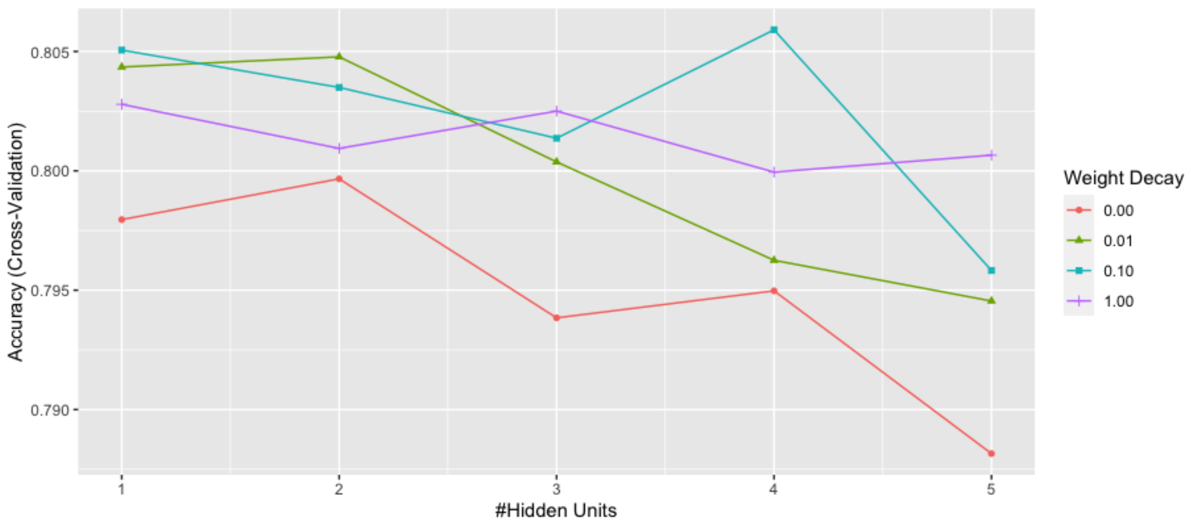

Figure 5.3.3 Neural Network graphs with the range of weight decay and hidden layers

Looking above in the graph the training accuracy for the Neural Networks with weight decays have a higher accuracy compared to one with no weight decay. This is because weight decay helps improve the model by improving the generalisation of the model by suppressing any static noise (unnecessarily feed/information) on the output (Anders K and John A.K (1991). It also suppresses any irrelevant components of the weight vector by choosing the smallest vector that solves the problem. This has been used to combat over-fitting, as mentioned in Section 4.2 Methodology.

In Figure 5.3.3, the maximum training accuracy with 200 iterations was with a neural network of 4 hidden layers and 0.1 weight decay. And with that, this proposed model, we will output its test accuracy and show it in Section 6, Results.

5.4 Confusion Matrix Tables

This outcome was based on the actual training test set and the predicted output of the test set. These tables allow us to compute the given performance metrics mentioned in Section 4.7 and it allows us to visualise the values for wrongly and correctly predicted values.

Decision Tree (with Gini)

		Predicted	
		No	Yes
Actual	No	479	38
	Yes	94	92

Table 5.4.1 Confusion Matrix for Decision Tree with Gini

Decision Tree (with Entropy)

		Predicted	
		No	Yes
Actual	No	473	44
	Yes	92	94

Table 5.4.2 Confusion Matrix for Decision Tree with Cross Entropy

Random Forest (1,000 trees)

		Predicted	
		No	Yes
Actual	No	510	7
	Yes	10	176

Table 5.4.3 Confusion Matrix for Random Forest with 1,000 trees

Neural Network (With 4 hidden layer and 0.1 weight decay)

		Predicted	
		No	Yes
Actual	No	469	48
	Yes	81	105

Table 5.4.2 Confusion Matrix for Neural Network with 4 hidden layer and 0.1 weight decay

6.Results

Now, to compare the machine learning accuracies found in this thesis alongside the Logistic Regression in Semester 1. These training accuracies are tested against the test data results. We used the dplyr package in R to compare the test data and predicted data.

Table 6. Table comparison for the accuracies for each Machine Learning Technique

	Machine Learning Techniques				
	Logistic Regression (Semester 1)	Decision Tree (with Gini)	Decision Tree (with Cross Entropy)	Random Forest (1000 trees)	Neural Network (With 4 hidden layer and 0.1 weight decay)
Test Data Accuracy Rate	0.4780	0.8122	0.8065	0.9765	0.8137

As observed in Table 6 above, there has been a significant improvement in the training accuracy from the methods applied especially from Semester 1 and in comparison between techniques applied in this current thesis, in Semester 2. We have observed that Random Forest outputs the highest training accuracy in comparison to the other models, with a 97.6% accuracy rate. In other words, this means that we are 97.6% confident that the trained model will output with exact output as test data. Simply put, we are 97.6% confident in our prediction.

The benefit of this model and why it outweighs others in terms of accuracy rate is due to the ensemble method that Random Forest has. It reduces overfitting, introduces bootstrapping and in a way generalises the model (Gareth J. et. al (2013) so that when we have a new set of dataset, it will be a very good fit. However, there is a disadvantage to this model. This method may take a very long time to train and have a high cost in runtime, therefore resulting in computational complexity. On top of that, with decision trees deemed as the approach that was easily interpretable, having multiple decision trees takes away the simplicity of the model, and is harder to interpret.

Having a high accuracy but long computational time may take away the effectiveness of the model and whether it would actually be the best model. To put it into perspective, for this given dataset, with 7,000 rows and 21 variables, the computational time was estimated to be around 7 minutes. Hypothetically, if we were to take into consideration a large dataset with almost 7 million rows and more than 21 variables, one can only imagine how long the computational time will be, as it may take hours or even days to produce results.

Other than using accuracy rate as an indicator in determining which one is a good model. It's also important to note that there are other metrics to decide on that may produce different results. We mentioned the other performance metrics such as precision, specificity, sensitivity and F1 Score, that were all defined in Section 4.7. The metrics could determine another model to be better based on the values to True Positive, True Negative, False Positive and False Negative.

7. Conclusion and Discussion

As shown in Section 6, Random Forest outputs the highest accuracy rate in terms of predicting churn. It is then followed by Neural Networks, Decision Trees and lastly, Logistic Regression. We have explored the benefits and disadvantages to all approaches, weighed them out and compared them to each other and the Telcos dataset that was proposed.

It was found that, as Random Forest is the best model with the highest test data accuracy, it was important to weigh out the cons on whether it was preferable to use in this situation. For Telcos dataset having a relatively small dataset, it would be best to use Random Forest. However, for bigger datasets that have millions of rows, it would not be wise to Random Forest due to its long computational runtime. This may result in long processing power, and result in high costs for businesses. Also, the performance metric of accuracy might not be the best metric to use, we may need to use other metrics as a way of determining which model is best as it may output different results.

Now that we have found the best machine learning model, it always cycles back to why this analysis is important: Why churn prediction is important?

From a business point of view, the goal is to always strive to reduce churn. Understanding that acquiring new customers bears a higher host compared to maintaining and retaining existing customers (J.Hadden 2006) is the fuel that drives this research. And with this, from the knowledge that the companies know, it helps them identify why customers want to leave so that early intervention plans can be implemented to convince them to stay on longer.

In an academic approach, it allows us to explore the best learning techniques that allow decisions to be made with backed up research. With that, the latest knowledge is found useful for both future researchers and businesses. Also, it allows businesses to make sound and credible decisions with clear distinction between fact and opinion.

To conclude this thesis and this project overall, we have found the variables that affect churn by analysing the variables that affect churn to find the best model approach to tackle churning customers with these variables. It has helped us shape and build the methods of exploratory analysis and model fitting in data science, and with that allowed stakeholders and the researcher to have a broadened understanding of customer churn prediction.

References

Anders K. and John A.K. (1991) A Simple Weight Decay can Improve Generalisation. *Advances in Neural Information Processing Systems Volume 4.*

Breiman, L., (2001). Random forests. *Machine learning*, *45* (1) , pp. 5-32

Breiman, L., (1996). Bagging predictors. *Machine learning*, *24*(2), pp.123-140

Chen, Z.Y., Fan, Z.P. & Sun, M. (2012) A Hierarchical Multiple Kernel Support Vector Machine for Customer Churn Prediction Using Longitudinal Behavioural Data. European Journal of Operational Research. 223(2). P.461-472.

Gareth, J., Daniela, W., Trevor, H. and Robert, T., (2013). An introduction to statistical learning: with applications in R. Springer.
Hadden J. , Tiwari A. , Roy R. , Ruta D. (2007) Computer assisted customer churn management: state-of-the-art and future trends. Comput. Oper. Res., 34 (10), pp. 2902-2917

Hastie, T., Tibshirani, R., Friedman, J.H. and Friedman, J.H., (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: Springer.

Hung S.-Y. , Yen D.C. , Wang H.-Y. . (2006). Applying data mining to telecom churn management. *Expert Syst. Appl., 31 (3)*, pp. 515-524

J. Burez, D. Van den Poel. (2009). Handling class imbalance in customer churn prediction. *Expert Syst. Appl.*, 36 (3), pp. 4626-4636

Kirui, C., Hong, L., Cheruiyot,W. & Kirui, H. (2013). Predicting Customer Churn in Mobile Telephony Industry Using Probabilistic Classifiers in Data Mining. *International Journal of Computer Science Issues.* 10(2). P.165-172

Kuhn, M. (2008). Building Predictive Models in R Using the caret Package. Journal of Statistical Software, 28(5), 1–26.

Markopoulos, A. P., Manolakos, D. E., & Vaxevanidis, N. M. (2008). Artificial neural network models for the prediction of surface roughness in electrical discharge machining. Journal of Intelligent Manufacturing, 19, 283– 292.

Robert C. Blattberg, Kim B.D and Scott A. (2008) Database Marketing: Analyzing and Managing Customers. Springer

Sharma A., Prabin Kumar P. , (2011). A neural network based approach for Predicting Customer churn in cellular network services. *Int. J. Comput. Appl.*, 27 (11), pp. 26-31

Tan Y.S. (2021) Customer Behaviour Analytics based on Machine Learning Classification Techniques, Statistic Dissertation A

Appendix

```
##### Data Processing ######
#Website link:
https://towardsdatascience.com/predict-customer-churn-the-right-way-using-pycaret-8ba6541608ac
# Youtube link to ggplot tutorial = https://www.youtube.com/watch?v=49fADBfcDD4

file <-
"https://raw.githubusercontent.com/srees1988/predict-churn-py/main/customer_churn_data.csv"

data <- read.csv(file)
View(data) # view the data in a different tab

#library("dplyr")



column_names <- data

name_of_col <- colnames(data)

#structure of the data
str(data)
#        7043 obs. of  21 variables:

class(data)


### loop over the column headings and change characters to factors
  for (i in 1:length(column_names)) {

      if (lapply(data, class)[[i]] == "character") {
        data[,i] <- as.factor(data[,i])

      # data$column<- as.factor(data$column)
      }
  }




data$customerID<- as.character(data$customerID) # CustomerID must be Character because it is
Unique


## check that the data has changed to factor
#structure of the data
str(data)
```

```
data_with_NA <- data

# is.na.data.frame(data)

# to check null values in each column
cbind( lapply( lapply(data, is.na),sum) )

# replace NA columns with 0 values
data <- replace(data, is.na(data), 0)



################################################################################
########## Decision Trees ##########

# install libraries
library(rpart)
library(rpart.plot)
library(caTools)
library(caret)
library(dplyr)

# the question to answer is, will this customer churn?

# dataset1 won't include CustomerID
dataset1 <- subset(data, select = -customerID)
str(dataset1)


set.seed(6563)
# split the data
subset(data, )
training <- createDataPartition(data[ ,"Churn"], p=0.9, list = FALSE)

train_data <- dataset1[training,] # split train and valid
test_data <- dataset1[-training,]

# training_2 <- createDataPartition(train_data[ ,"Churn"], p=0.8, list = FALSE)
# # run CV with training indices 2
# # cp optimised
# # take that optimised
# cv_data <-

set.seed(6563)

# 10-Fold CV
#cross_val <- trainControl(method = "repeatedcv", number = 10, repeats = 10)
cross_val <- trainControl(method = "cv", number = 10)
```

```
set.seed(6563)
# Training the DT Classifier
# by default, the splitting index is gini
tree_train <- train(Churn ~. , data = train_data, method = "rpart",
                    trControl = cross_val,
                    tuneLength = 10)

# plot model accuracy vs cp (complexity parameter)

plot(tree_train)

plot(tree_train, xlim = c(0, 0.01))
# plot the accuracy curve


# print the best tune parameter cp
# that maximises the model accuracy
tree_train$bestTune
# 0.003267974

#max accuracy?
max(tree_train$results)
# [1] 0.7966954

# which depth

tree_train$finalModel

summary(tree_train$finalModel)

par(xpd = NA)
plot(tree_train$finalModel)
text(tree_train$finalModel, pretty = 0)
title(main = "10-CV Classification Tree with Gini")

# or can plot out this way
suppressMessages(library(rattle))
fancyRpartPlot(tree_train$finalModel)

# Make predictions on the test data
predicted_classes_gini <- tree_train %>% predict(test_data)

# Compute model accuracy rate on test data
mean(predicted_classes_gini == test_data$Churn)
#### 0.8122333
```

```
### CHECK WITH CROSS ENTROPY AND SEE RESULTS

set.seed(6563)
tree_cross_entropy <- train(Churn ~. , data = train_data, method = "rpart",
                trControl = cross_val,
                tuneLength = 10,
                parms = list(split = "information"))

# plot model accuracy vs cp (complexity parameter)
plot(tree_cross_entropy)

plot(tree_cross_entropy, xlim = c(0, 0.02)) #readjust

# plot the accuracy curve
ggplot(tree_cross_entropy)
ggplot(tree_cross_entropy) + coord_cartesian(xlim = c(0, 0.02))


# print the best tune parameter cp
# that maximises the model accuracy
tree_cross_entropy$bestTune
# 0.002376708

# max accuracy?
max(tree_cross_entropy$results)
# 0.7906974

# which depth
tree_cross_entropy$finalModel

summary(tree_cross_entropy$finalModel)

par(xpd = NA)
plot(tree_cross_entropy$finalModel)
text(tree_cross_entropy$finalModel, pretty = 0)
title(main = "10-Cross validation Classification Tree with Entropy")

# or can plot out this way
suppressMessages(library(rattle))
fancyRpartPlot(tree_cross_entropy$finalModel)

# Make predictions on the test data
predicted_classes_ce <- tree_cross_entropy %>% predict(test_data)

# Compute model accuracy rate on test data
mean(predicted_classes_ce == test_data$Churn)
```

```
## [1] 0.8065434



##### CREATE A CONFUSION MATRIX
# gini
confusionMatrix(test_data$Churn,predicted_classes_gini )
# Reference
# Prediction  No Yes
# No  479  38
# Yes  94  92




# cross entropy
confusionMatrix(test_data$Churn,predicted_classes_ce )
# Reference
# Prediction  No Yes
# No  473  44
# Yes  92  94

##################################################################################
####################### RANDOM FOREST ########################

## with out of bag for the missing values

library(ggplot2)
library(cowplot)
library(randomForest)
library(dplyr)


data_with_NA

dataset_NA <- subset(data_with_NA, select = -customerID)

?rfImpute #missing value imputations by randomForest

set.seed(6563)

# Impute missing values in predictor data using proximity
# from randomForest.
imputed_data <- rfImpute(Churn ~. , data = dataset_NA)
  # it will produce 5 rows

# ntree      OOB      1      2
# 300:  20.25% 10.55% 47.08%
# ntree      OOB      1      2
# 300:  20.50% 10.55% 48.05%
# ntree      OOB      1      2
# 300:  20.49% 10.46% 48.26%
# ntree      OOB      1      2
# 300:  20.15% 10.32% 47.35%
# ntree      OOB      1      2
# 300:  20.47% 10.63% 47.73%


model_rf <- randomForest(Churn~. , data = imputed_data, proximity = TRUE)

model_rf

# Type of random forest: classification
# Number of trees: 500
# No. of variables tried at each split: 4
#
# OOB estimate of  error rate: 20.26%
## what this means: that 79.74% of OOB samples were correctly classified in the random forest
# Confusion matrix:
#   No Yes class.error
# No  4651 523   0.1010823
# Yes  904 965   0.4836811

### plot variable importance
## mean decrease gini
table1<-data.frame(model_rf$importance)
rowz_z <-table1[,1]
rowz_n <-row.names.data.frame(table1)
barplot(table1)

weee <- data.frame(variable = rowz_n,
          variable_importance = rowz_z)

ggplot(data = weee, aes(x = variable, y = variable_importance)) +
      geom_bar(stat = 'identity')+
  coord_flip()
#####



oob_error_rate <- data.frame(
  Trees = rep(1:nrow(model_rf$err.rate), times = 3),
  Type = rep(c("OOB", "No", "Yes"), each = nrow(model_rf$err.rate)),
  Error = c(model_rf$err.rate[, "OOB"],
  model_rf$err.rate[, "No"],
  model_rf$err.rate[, "Yes"]))
```

```
ggplot(data = oob_error_rate, aes(x = Trees, y = Error)) +
    geom_line(aes (color = Type)) +
  labs( title = 'Error rate of 500 trees' ) +
  theme(plot.title = element_text(hjust = 0.5))
## Blue line = The error rate specifically for Non Churners that
## are OOB.
##
## Green line = The overall OOB error rate.
##
## Red line = The error rate specifically for Yes Churners that
##are OOB.




## REPEAT BUT WITH 1000 TREES
model_rf <- randomForest(Churn~. , data = imputed_data, proximity = TRUE, ntree = 1000)

model_rf

# Call:
#   randomForest(formula = Churn ~ ., data = imputed_data, proximity = TRUE,      ntree = 1000)
# Type of random forest: classification
# Number of trees: 1000
# No. of variables tried at each split: 4
#
# OOB estimate of  error rate: 20.29%
# Confusion matrix:
#   No Yes class.error
# No  4626 548   0.1059142
# Yes  881 988   0.4713751


oob_error_rate <- data.frame(
  Trees = rep(1:nrow(model_rf$err.rate), times = 3),
  Type = rep(c("OOB", "No", "Yes"), each = nrow(model_rf$err.rate)),
  Error = c(model_rf$err.rate[, "OOB"],
           model_rf$err.rate[, "No"],
           model_rf$err.rate[, "Yes"]))


ggplot(data = oob_error_rate, aes(x = Trees, y = Error)) +
  geom_line(aes (color = Type)) +
  labs( title = 'Error rate of 1,000 trees' ) +
  theme(plot.title = element_text(hjust = 0.5))

# Would work just as best with 500 trees,
# as there is a convergence even after 500 trees

######### Calculate the test data accuracy (training accuracy) #########
# Make predictions on the test data
predicted_classes_RF <- model_rf %>% predict(test_data)

# Compute model accuracy rate on test data
mean(predicted_classes_RF == test_data$Churn)
####0.9765458

confusionMatrix(test_data$Churn, predicted_classes_RF)
# Confusion Matrix and Statistics
#
# Reference
# Prediction  No Yes
# No  510   7
# Yes  10 176
#
# Accuracy : 0.9758
# 95% CI : (0.9616, 0.9859)
# No Information Rate : 0.7397
# P-Value [Acc > NIR] : <2e-16
#
# Kappa : 0.9375
#
# Mcnemar's Test P-Value : 0.6276
#
#             Sensitivity : 0.9808
#             Specificity : 0.9617
#          Pos Pred Value : 0.9865
#          Neg Pred Value : 0.9462
#              Prevalence : 0.7397
#          Detection Rate : 0.7255
#    Detection Prevalence : 0.7354
#       Balanced Accuracy : 0.9713
#
#        'Positive' Class : No


##############################################################################
######################## NEURAL NETWORK ########################

# Deep Learning
# Neural Network
# The word 'deep' stems from  the many layers of the neural network
# A neural network that only has two or three layers is just a basic neural network.

str(dataset1)
str(train_data)

library(neuralnet)
library(caret)
library(NeuralNetTools)
library(dplyr)


set.seed(6563)


 #### PROCESS THE DATA: perform one-hot encoding #######
#exclude churn so that we can one hot enconding on others
dataset2 <- subset(dataset1, select = -Churn)
str(dataset2)

#define one-hot encoding function
dummy <- dummyVars(" ~ .", data=dataset2)

#perform one-hot encoding on data frame
final_df <- data.frame(predict(dummy, newdata=dataset2))

#view final data frame
final_df

# combine back the Churn Column
final_df<- cbind(final_df, dataset1$Churn)

#rename to Churn
colnames(final_df)[colnames(final_df) == "dataset1$Churn"] <- "Churn"

######## ######## ######## ######## ########
######## fit the neural network ########
nn_telco_all <- neuralnet( Churn ~ . ,
                    data = final_df,
                    linear.output = FALSE,
                    hidden = 2
                    # specify the nodes per layer)
)

## plot the neural network graphing
plot(nn_telco_all)



#### have the final_df go through cross validation as well

cross_val <- trainControl(method = "cv", number = 10)
```

```
set.seed(6563)
# Training the neural net classifier
#

# took about 7mins to load out
Gneural_net_train <- train(Churn ~. , data = final_df, method = "nnet",
                     trControl = cross_val
                     #,size = 3
                     ,maxit = 200
                     ,preProcess = c("center","scale")
                     ,tuneGrid = expand.grid(size = seq(1,5,1), decay = c(0, 0.01, 0.1, 1))
                     # or decay = 10^seq(-5,0,by=1))
                     # size =Hidden Layers
                     # decay = _ #Weight Decay
                     ,metric= "Accuracy")

ggplot(Gneural_net_train)

# locate the best tuned, the accuracy rate
Gneural_net_train$bestTune
#   size decay
# 7    4   0.1

Gneural_net_train$results

which.max(Gneural_net_train$results$Accuracy)
# 15    4  0.10 0.8059076 0.4716851 0.009576771 0.02545719

max(Gneural_net_train$results$Accuracy)
# max accuracy attained at 0.8059076

## Now, let's test it with the test data ##
######### Calculate the test data accuracy (training accuracy) #########

## based on the highest is attained when model is 2 with decay 0.1
best_neural_net_train <- train(Churn ~. , data = final_df, method = "nnet",
                     trControl = cross_val
                     #,size = 3
                     ,maxit = 200
                     ,preProcess = c("center","scale")
                     ,tuneGrid = expand.grid(size = 4, decay = 0.1)
                     # or decay = 10^seq(-5,0,by=1))
                     # size =Hidden Layers
                     # decay = _ #Weight Decay
                     ,metric= "Accuracy")
```

```
## make test data become one hot encoding as well
nn_test_data <- test_data

#### PROCESS THE DATA: perform one-hot encoding #######
#exclude churn so that we can one hot enconding on others
nn_test_dataset <- subset(nn_test_data, select = -Churn)
str(nn_test_dataset)

#define one-hot encoding function
dummy <- dummyVars(" ~ .", data=nn_test_dataset)

#perform one-hot encoding on data frame
nn_final_test_data<- data.frame(predict(dummy, newdata=nn_test_dataset))

#view final test data for nn
nn_final_test_data

# combine back the Churn Column
nn_final_test_data<- cbind(nn_final_test_data, nn_test_data$Churn)

#rename to Churn
colnames(nn_final_test_data)[colnames(nn_final_test_data) == "nn_test_data$Churn"] <- "Churn"

# Make predictions on the test data
predicted_classes_NN <- best_neural_net_train %>% predict(nn_final_test_data)

# Compute model accuracy rate on test data
mean(predicted_classes_NN == nn_final_test_data$Churn)
####  0.8165007

confusionMatrix(nn_final_test_data$Churn,predicted_classes_NN)

# Confusion Matrix and Statistics
#
# Reference
# Prediction  No Yes
# No  469  48
# Yes  81 105
#
# Accuracy : 0.8165
# 95% CI : (0.7859, 0.8444)
# No Information Rate : 0.7824
# P-Value [Acc > NIR] : 0.014561
#
# Kappa : 0.5001
#
# Mcnemar's Test P-Value : 0.004841
```