

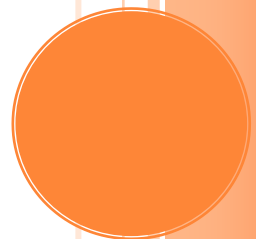
CUSTOMER CHURN MODEL

Betfair Australia

Prediction of customers who are about to fall in to churned customer category and there

Amal Joy: S3644794

2/6/18



1 TABLE OF CONTENTS

1	Table of Contents.....	1
2	Executive Summary.....	2
3	Introduction.....	3
4	Materials and Methods	4
4.1	Data Collection	4
4.2	Data Exploration	4
4.3	Data Visualization.....	5
4.4	Data Cleaning.....	9
4.5	Feature Selection.....	9
4.6	Data Preparation.....	10
4.7	Data modeling.....	10
4.7.1	<i>Configuring the learning task.....</i>	<i>11</i>
4.7.2	<i>Configuring the learner.....</i>	<i>11</i>
4.7.3	<i>Training the learner.....</i>	<i>11</i>
4.7.4	<i>Make prediction</i>	<i>11</i>
4.7.5	<i>Performance Evaluation</i>	<i>12</i>
4.7.6	<i>Prediction using different models.....</i>	<i>12</i>
4.8	Resampling	12
4.9	Hyper-parametric Tuning.....	13
5	Results	14
5.1	Resampling Results.....	15
5.2	Hyper-parametric Tuning Results.....	15
6	Discussion.....	15
7	List of Figures	16
8	List of Tables.....	16
9	Bibliography	17
10	Appendices.....	18
	Appendix A:.....	18
	Appendix B:.....	18
	Appendix C:.....	18
	Appendix D:	18
	Appendix E:.....	19
	Appendix F:.....	19
	Appendix G:	20

THE CUSTOMER CHURN MODEL

Betfair Australia

2 EXECUTIVE SUMMARY

For the last few years Betfair is losing too many customers and is finding a solution to retain its customers. Aim of this project is to build a customer churn model to predict the customers who are about to get churned so that Betfair can implement different business strategies to retain those customers before they actually leave. The tools I am using for this analysis are R-studio and Tableau. Package mlr was chosen as the modeling package. The data for the purpose of prediction was provided by Betfair. After proper data exploration and visualization, important features for the customer churn prediction model was identified. The 8 different classification models were applied on the data in separate steps of configuring the learner task, making the learner, training the learner, prediction and performance evaluation. Out of the 8 different models, Random Forest was chosen as the best model. Cross-validation was done using random forest was done and obtained a mean miss classification error rate of 0.1278126. Hyper-parametric tuning of the random forest model was performed using package mlrHyperopt. There was only 0.05% improvement in the model accuracy after Hyper-parametric tuning. The model obtained is good enough to predict the customers who are about to fall in the churned customer category. Applying this model on the real-time data in Betfair can save huge money in revenue.

3 INTRODUCTION

Being an online gambling, Betfair operates world's largest online betting exchange (Wikipedia, 2017). The company was established in 2000, by Andrew Black and Edward Wray, from where it rose to the position of the largest betting company in the UK and the largest betting exchange in the world. Betfair's idea of exchange derived from the capital markets. A customer who is betting an amount of 1,000 \$ against one of the possible outcomes of a sports event might be matched with tens or hundreds of other players who bet for that outcome. Thus, Betfair engages its customers by placing bets against each other's choices, thereby charging a commission of 5 to 10% on revenue made by winning customers. For the last few years Betfair is losing too many customers and is finding a solution to retain its customers. For the last 3 years, almost 9,390 customers joined Betfair and almost 17,166 customers left Betfair. This created an alarming situation in Betfair.

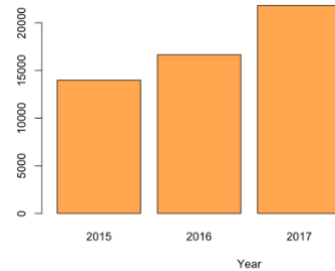


Figure 1 Number of customers churned in the past 3 years

The aim of this project is to build a customer churn model to predict the customers who are about to get churned so that Betfair can implement different business strategies to retain those customers before they actually leave. The tools I am using for this analysis are R-studio and Tableau. For modeling purpose, I chose one of the latest modeling package available for; MLR. The data used in this report is provided completely by Betfair Australia. I did the feature selection by actually experiencing the betting with Betfair and analyzing different customers who actually left Betfair. I had the information for some of the most important customers who stopped betting with Betfair. The customer feedbacks from those customers helped me in identifying the most important features to be considered for preparing the customer churn model. Through this analysis, I hope to predict those valued customers who have a high chance of leaving the company and thereby helping the business team to consider their case for necessary action.

The assumptions taken in this project are:

1. The attributes chosen in this project are enough to accurately predict the customers who are about to get churned
2. There is enough data in the model to accurately predict the customers who are about to get churned.

4 MATERIALS AND METHODS

I have used R program for data cleaning and data exploration. MLR package was made utilized to train the model. Feature selection was done using churned customer feedbacks and other visual analysis.

4.1 Data Collection

All of the data used in this report is owned and maintained by Betfair Australia. The data have been provided in a summarized and structured way to be used in the model by the data science team in Betfair Australia. Going through the business of Betfair, I came to know that their area of interest is in horse betting from which they generate most of their revenue. So, I have decided to go for a model which predicts the customer churn in horse betting.

Data included customer betting details from 2000 to 28 February 2018. But the customer churn data was available only from 2015. Data was collected in two phases. In the first phase, the basic data from customer feedbacks and visualization results were used. In the second phase, there were more meaningful data which included the pattern in which the data changed towards the final days of the customer.

4.2 Data Exploration

To check if there are any issues in the data, I produced the summary of the original dataset using the 'summary' function in R, before data preparation and adding the calculated fields. The output is shown below.

```

Last_Month_num_of_days_bet Previous.Months_num_of_days_bet Last_Month_profit_loss
Min. : 0.000 Min. : 1.000 Min. : -2461802
1st Qu.: 0.000 1st Qu.: 4.875 1st Qu.: -30
Median : 0.000 Median : 7.573 Median : 0
Mean : 4.671 Mean : 9.061 Mean : 499
3rd Qu.: 6.000 3rd Qu.: 11.831 3rd Qu.: 0
Max. : 31.000 Max. : 30.361 Max. : 13535912
Previous.Months_profit_loss Last_To_Last_Month_profit_loss Last_Month_total_bets
Min. : -3083336 Min. : -4851453 Min. : 0
1st Qu.: -2575 1st Qu.: -546 1st Qu.: 0
Median : -638 Median : 0 Median : 0
Mean : -1870 Mean : -303 Mean : 3840
3rd Qu.: 21 3rd Qu.: 0 3rd Qu.: 418
Max. : 7273185 Max. : 20234608 Max. : 12733600
Previous.Months_total_bets Last_To_Last_Month_total_bets Last_Month_turnover
Min. : 15 Min. : 0 Min. : 0
1st Qu.: 333 1st Qu.: 0 1st Qu.: 0
Median : 832 Median : 0 Median : 0
Mean : 3708 Mean : 4072 Mean : 205836
3rd Qu.: 2176 3rd Qu.: 660 3rd Qu.: 12430
Max. : 7920956 Max. : 13554398 Max. : 323013180
Previous.Months_turnover Last_To_Last_Month_turnover customer_flag
Min. : 474 Min. : 0 Active:11312
1st Qu.: 9400 1st Qu.: 0 Churn :12961
Median : 27341 Median : 0
Mean : 240334 Mean : 192940
3rd Qu.: 94395 3rd Qu.: 16928
Max. : 255442396 Max. : 520653241

```

Figure 2 Summary of the original data before data preparation

It can be observed that the dataset is already balanced as a result of the data filtering done in the previous steps. There are 11,312 active customers and 12,961 churned customers in the dataset. There are many zero values in the dataset and this is acceptable as these values are from those customers who already got churned. Rest everything seems alright with the dataset.

The check for any 'NA' values in the dataset showed that there are no null values present in the dataset under consideration (See [Appendix A](#) for the code). The output of analyzing the structure of the dataset after data preparation is shown below.

```
Classes 'tbl_df', 'tbl' and 'data.frame':    24273 obs. of  12 variables:
 $ Last_Month_num_of_days_bet      : num  0 19 29 29 0 29 0 0 11 8 ...
 $ Previous_Months_num_of_days_bet: num  17.74 13.39 19.56 19.79 5.64 ...
 $ Last_Month_profit_loss          : num  0 -8295 63935 -6820 0 ...
 $ Previous_Months_profit_loss     : num  4674 640 -76841 -3810 -4221 ...
 $ Last_To_Last_Month_profit_loss : num  0 -2639 -147616 -3403 -327 ...
 $ Last_Month_total_bets           : num  0 1463 20603 26004 0 ...
 $ Previous_Months_total_bets      : num  1778 1135 6957 7354 1004 ...
 $ Last_To_Last_Month_total_bets   : num  0 1243 16192 13024 253 ...
 $ Last_Month_turnover             : num  0 25163 10852444 218237 0 ...
 $ Previous_Months_turnover        : num  67987 19954 1426867 90589 484945 ...
 $ Last_To_Last_Month_turnover     : num  0 18939 7049088 114154 8781 ...
 $ customer_flag                   : Factor w/ 2 levels "Active","Churn": 1 1 1 1 2 1 2 2 1 1 ...
```

Figure 3 Data Structure exploration

All the variables except the 'customer_flag' is numerical variables. The data-frame contains 24,273 observations and 8 variables.

4.3 Data Visualization

I used Tableau for the data visualization and exploring various relation among the variables. Feature selection was done using the insights gained by creating more interactive visualizations. Many of the abnormalities with the data was identified with the help of data visualizations. It was found that the data is flooded with less important features as they have hardly any effect on determining the probability of a customer to get churned. Also, there was many customers who joined Betfair for just performing few bets and then left their user inactive. There is no point in predicating the churn probability of those customers as they don't contribute much to the revenue of the Betfair. Through this project Betfair trying to retain those customers who are very serious in their participation in market generation and had provided thousands of dollars in revenue for Betfair.

The visualization below shows the relations between revenue generated by the customer, profit/loss of the customer, and the highest discount rate provided to the top 20 churned customers in terms of their revenue.

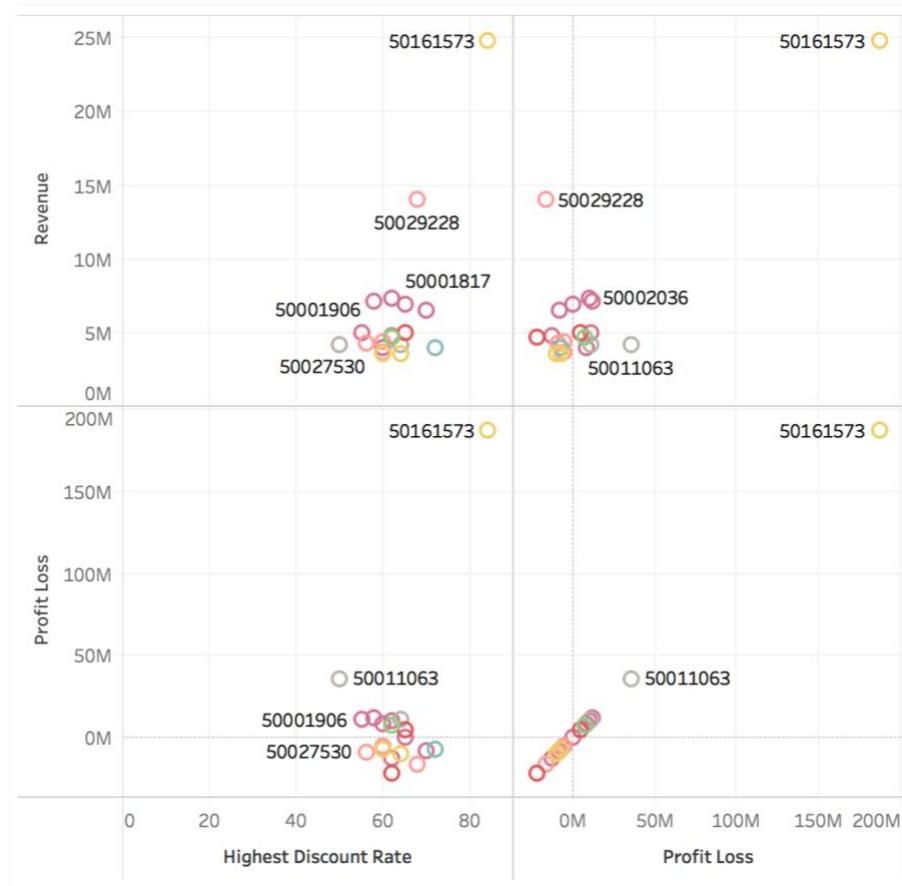


Figure 4 Revenue v/s Profit / loss of the top 20 churned customers

It can be seen that there are few exceptional customers who made huge revenue to the company. This project is mostly about predicting those customers and trying to retain them as long as possible, because according to Betfair's business these customers are the market generators and that's how the market become active for other customers. There seems to have a positive relation among the revenue provided by a customer and the discount rate provided to the customer. Most of the top 20 churned customers were provided with high rates of discount on the commissions. So clearly it was not a major reason for churn as Betfair's standard rate on commission was 5 to 10 % and most of them were given more than 60% of reduction in that rate.

The scatter plot for discount rate and revenue is shown below. It is evident from the graph that people who provided more revenue were given more discounted rate on the commission.

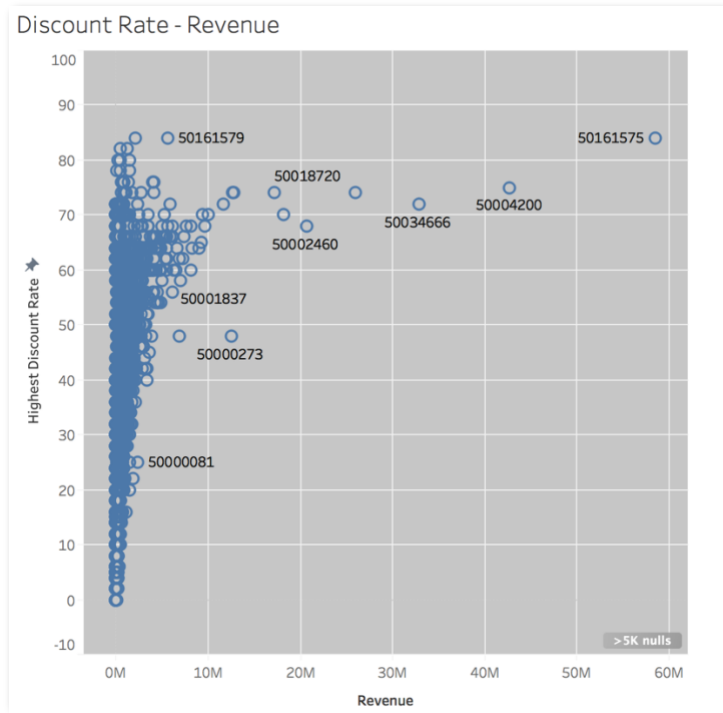


Figure 5 Scatter plot of discount rate and revenue generated.

The diagram below shows the revenue produced by active and churned customers in terms of number of years they served as customer.

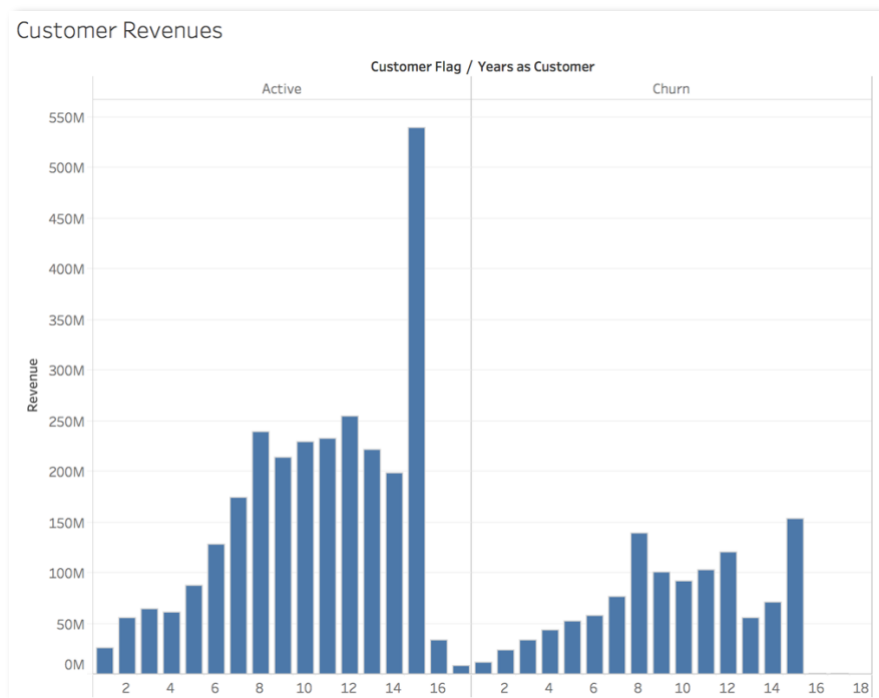


Figure 6 Revenue in terms of number of years served as customer

In spite of the fact that the number of active customers are comparatively very much less than the number of churned customers, it is interesting to note that the revenue produced by the active customers are far higher than that of the churned customers. This turn light to the fact that not all churned customers were producing much revenue. There is lot of customers who are producing almost nothing in terms of revenue. Trying to predict the churn of those customers in retaining them are not worth the efforts. So, the data need to get polished before applying to the model.

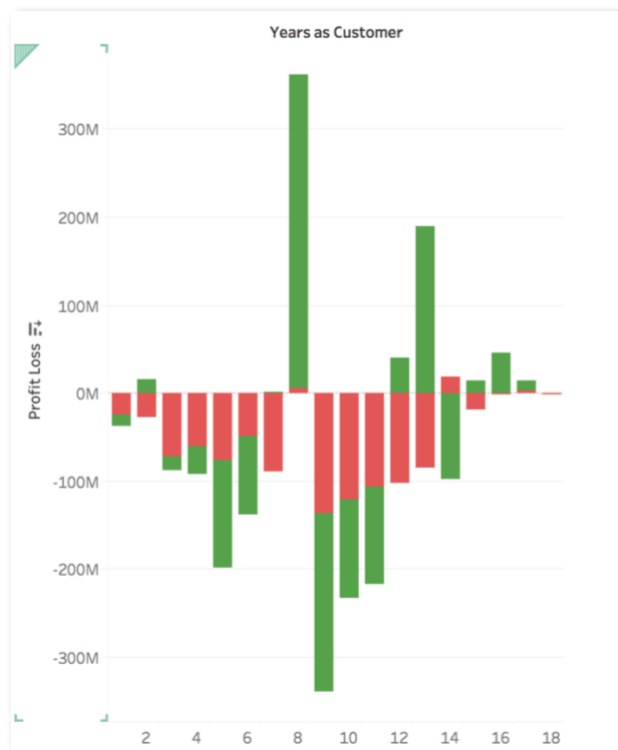


Figure 7 Bar plot of profit/loss vs number of years as customer.

The visualization below shows the longest serving customers who got churned.

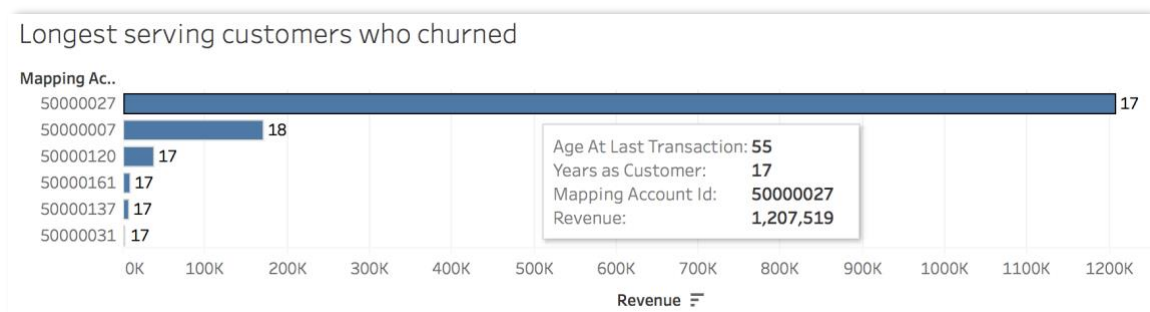


Figure 8 Bar plot of top longest serving customer in terms of revenue who got churned

These customers transactions very studied carefully and some of the reasons for their decision to leave was identified by the Betfair analytics team. Some of these customers are also given customer feedback form to give the reason for leaving Betfair. From all these information, the important features that contribute for the prediction of customer churn was identified.

4.4 Data Cleaning

All those customers who were identified as not producing enough revenue was removed from the dataset. The threshold was kept at 2500\$, as this figure was producing a balance in the data distribution. Also, those customers who had not served enough time is not capable of being a good resource for prediction since they have little information available about the customer history and betting pattern. So, those customers were also removed who haven't served more than 6 months with Betfair. Other sanity checks were applied on the data, as seen in the data exploration section of this document, for null value detection ([Appendix A](#)) and impossible values in the interested features. There weren't any notable impossible values present in the filtered data.

4.5 Feature Selection

Feature selection for the model building was done by analyzing the correlation between the features, univariate feature selection, recursive feature elimination, recursive feature elimination with cross validation and finding of important features based on the tree from random forest classification. The final model was supplied with the best possible features for the prediction. The correlation diagram displayed on the right side shows various correlation issues present in the data (See [Appendix B](#) for the code).

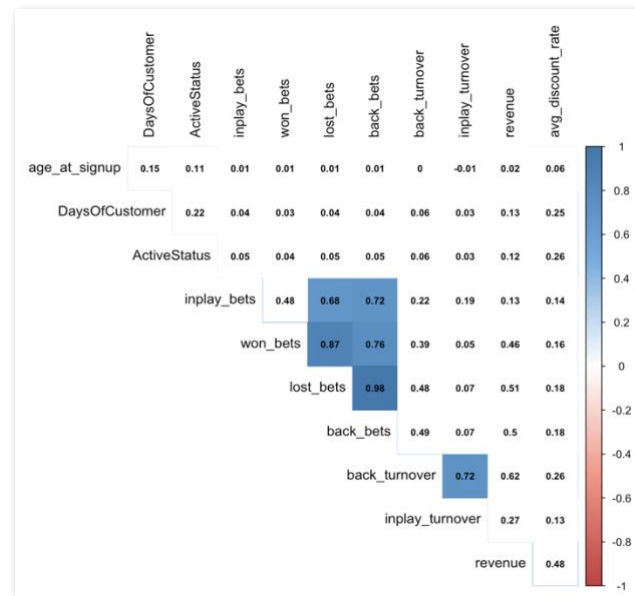


Figure 9 Correlation Diagram of the features

4.6 Data Preparation

The data collected in the phase 1 was having customer account ID as the primary key. But the extra features added to the churn data included column names 'period' which contained the repeated rows of 'last_month', 'last_to_last_month' and 'previous_months'. So, the account ID together with the 'period' column formed the primary key for the phase two data. This data was combined to form a single table by using the 'unite' and the 'spread' function in 'tidyR' and 'inner_join' from 'dplyr' (See [Appendix C](#) for the related code). While joining both the tables, there are 1,946 entries where previous months active period was zero. These customers are those who are new to Betfair and have been active for less than two months. For the purpose of this analysis I removed these customers from the model.

Since the data was highly imbalanced; 16201 active customers and 52,465 churned customers; I tried to apply oversampling on the data. But oversampled model was overfitting the data and I had to resort to other techniques of including only those customers who are really contributing to revenue generation. I also removed those customers who are active in Betfair for more than 4 months and have produced a total revenue of \$2500 or more.

Afterwards, during the progress of the model, to reduce dimensionality and to avoid possible correlations among the variables which may leak duplicate information in to the model, some of the variables were combined to form a calculated field in the model. The formula used for making the calculated fields can be found in the [Appendix D](#). After going through various iterations of the model and by expert business advices from Betfair analytics team I chose to go with only a handful of variables for my model. These final variables include the ratio of number of days the customer did the actual betting in the last month and previous month, profit and loss in the last month and last to last month, total bets in last and last to last month, turnover in the last and last to last month.

4.7 Data modeling

The data modelling for the customer churn model was done using a very new package available in R's CRAN repository, called 'mlr'. R program by default does not make use of parallelization. But parallelization was necessary as the data set was big and needed many iterations in case of feature selection and cross-validation. I utilized mlr's parallel computing capabilities to reduce the computing time (See [Appendix E](#) for the code). The data was split in a ratio of 80:20, 80% assigned for training and 20% assigned for testing. Training a

learner means fitting a model to a churn data-set. The model was trained using the training dataset and then it was tested on the testing dataset. The MLR package provides a unified and powerful interface for building machine learning models in R. Within the MLR framework, a supervised machine learning typically follow the steps below (Bischl, et al., 2016):

- ✚ Configuring learning task
- ✚ Configuring learner
- ✚ Training a learner
- ✚ Making prediction using the trained learner
- ✚ Performance evaluation

4.7.1 Configuring the learning task

The learner was configured by setting the data as the training data and the target variable as 'Customer_Flag'(churned or not). Active customer class was chosen as the positive class. This makes more sense as predicting the churned customer as active is more dangerous than predicting active customer as churned. The code for configuring the learner task is given below.

```
classif.task <- makeClassifTask(id = "customer_flag", data = TrainingData,
target = "customer_flag", positive = "Active")
```

4.7.2 Configuring the learner

Making the learner includes specifying the type of classification model intended to apply on the data and the prediction type of the model. Prediction type can be either probability or response depending up on the business. For this project, I have chosen the prediction type as Probability. The R code for configuring a learner is given below.

```
classif.lrn <- makeLearner(cl = "classif.randomForest", predict.type = "prob")
```

4.7.3 Training the learner

In this step, the information from the learner task is used to train the learner using a random subset of the data as training set. The R code for training a learner is given below.

```
Model = mlr::train( classif.lrn, classif.task)
```

4.7.4 Make prediction

Using the values from the testing data, predict the response variable for the testing data using the trained learner. The R code for predicting the response variable of testing data is given below.

```
pred = predict(model, newdata = TestingData)
```

4.7.5 Performance Evaluation

As performance evaluation, the mean misclassification error, accuracy and false positives of the prediction was calculated.

```
performance(pred, measure = list(mlr::fp, mlr::auc, mlr::mmce))
```

4.7.6 Prediction using different models

Eight different models were applied on the data and best of eight models were chosen as the final model after cross-validation. The eight models which was chosen for prediction include:





- 1) **k-Nearest Neighbor**
- 2) **kSVM**
- 3) **Linear Discriminant Analysis**
- 4) **Gradient Boosting Machine**
- 5) **Neural Network**
- 6) **Support Vector Machine**
- 7) **Decision Tree**
- 8) **Random Forest**

These eight models were run in a loop so that every time when there is a change in any parameters or a change in the data, the loop was run and it will produce a table of model evaluation results (See [Appendix F](#) for the code).

4.8 Resampling

Resampling methods involve repeatedly drawing samples from a training data and refitting a model of interest on each sample to obtain more information about the fitted model.

Supported resampling strategies in mlr include:

-  **CV:** Cross-validation
-  **LOO:** Leave-one-out cross-validation
-  **RepCv:** Repeated cross-validation
-  **Bootstrap:** Out-of-bag bootstrap and other variants

- ✚ **Subsample:** Subsampling, also called Monte-Carlo cross-validation
- ✚ **Holdout:** Holdout (training/test)

In this project, I have used cross validation as the resampling strategy. This is done by dividing the data sample into a training set to train the model, and a test set to evaluate it, like we did before. I have chosen 10-fold cross validation here, so that the data is randomly divided into 10 equal size subsamples. Of the 10 partitions, a single partition is set as the validation data for testing the model, and the remaining 9 partitions are used as training data. The cross-validation process is then repeated 10 times, with each of the 10 partitions used exactly once as the testing data. The average of the 10 results obtained is then used to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once (openml.org, n.d.).

4.9 Hyper-parametric Tuning

A set of optimal hyper-parameters for a learner is chosen by the Hyper-parametric tuning in mlr. While model parameters are learned during training, hyper-parameters must be set before training. In the case of a random forest, hyper-parameters include the number of decision trees in the forest and the number of features considered by each tree when splitting a node. The best method to find the optimal parameters is to try many different combinations and then evaluate the performance of each model.

I used mlrhyperopt as the hyperparametric tuning controller for the model (See [Appendix G](#) for the code). The advantages of using mlrhyperopt is that it provides default search spaces for the most important learners in mlr and an API to add and access custom search spaces from the mlrHyperopt Database (Richter, 2017).

5 RESULTS

The results obtained (at seed = 300) for the eight different models we applied in this project is given below.

Table 1 Results of model evaluation at seed = 300

Model	Predictor	mmce	Accuracy	False_Positive
Gradient Boosting Machine	pred4	0.1320040	0.8679960	227
rpart	pred7	0.1326632	0.8673368	234
ksvm	pred2	0.1334871	0.8665129	283
random Forest	pred8	0.1336519	0.8663481	234
neural network	pred5	0.1348055	0.8651945	253
svm	pred6	0.1377719	0.8622281	264
kknn	pred1	0.1465063	0.8534937	356
lda	pred3	0.1934740	0.8065260	80

The same models at different seed (seed = 5342) is given below:

Table 2 Results of model evaluation at seed = 5342

Model	Predictor	mmce	Accuracy	False_Positive
rpart	pred7	0.1300264	0.8699736	225
Gradient Boosting Machine	pred4	0.1310152	0.8689848	221
random Forest	pred8	0.1326632	0.8673368	268
ksvm	pred2	0.1336519	0.8663481	321
neural network	pred5	0.1344759	0.8655241	227
svm	pred6	0.1418919	0.8581081	287
kknn	pred1	0.1430455	0.8569545	366
lda	pred3	0.1947924	0.8052076	83

It can be seen that at different seed value the performance of the models varies widely. After 10 different iterations, it was found that only Random Forest was having the stable results in all the iterations and it maintained a decent position in the ranking table. So, Random forest was chosen as the final model for fitting the data.

The modeling results of the random forest model is given below.

Table 3 Performance Evaluation of random forest model

```
Model for learner.id=classif.randomForest; learner.class=classif.randomForest
Trained on: task.id = customer_flag; obs = 18205; features = 7
Hyperparameters:
      fp      auc      mmce
268.000000  0.9105061  0.1326632
predicted
true      Active Churn -err.-
Active    2321   537    537
Churn     268   2942   268
-err.-    268   537    805
```

There were 268 false positives in the models. Area under the curve (auc) was found to be 0.9105 and mean miss classification error(mmce) was 0.1326.

5.1 Resampling Results

The aggregated miss classification error was found to be 0.1278126 after 10-fold cross validation. Miss classification error was less than what I have obtained using manual train-test splitting.

5.2 Hyper-parametric Tuning Results

The results obtained for the hyper-parametric tuning is given below:

```

+ Op. pars: cp=0.537
+ maxdepth=3
+ minbucket=16
+ minsplit=26
+ mmce.test.mean=0.1203209
  
```

6 DISCUSSION

- 1) The variation of results at different seed values were may be due to some models' overfitting on the data. These models will have various performance results at various samples.
- 2) Random forest is a powerful model and it takes random number of subsamples at the model creation. Due to this, the chance of overfitting on the data is less with random forest.
- 3) Miss classification error was less than that obtained using manual train-test splitting. This means that we can rely on this model and make sure that it is not an overfitting model.
- 4) There was only 0.05% decrease in the Miss classification error obtained after Hyper-parametric tuning. This is due to the fact that the Random Forest by itself is a powerful ensemble. It tunes its parameters while training the model. So, Hyper-parametric tuning has a very little effect on the Random Forest model.
- 5) The model was tested on real-time data in Betfair. The data set used in the model was for a period until 28 Feb 2018. New data set which is till the end of April was fed in to the model. The accuracy obtained on the real-time data was 75%. \$91,521.9 could have been generated for the last 3 months if 25 % of the predicted churned customers were retained through proper business strategies.
- 6) Updating of the model every 3 months is recommended for best results.

7 LIST OF FIGURES

Figure 1 Number of customers churned in the past 3 years	3
Figure 2 Summary of the original data before data preparation.....	4
Figure 3 Data Structure exploration	5
Figure 4 Revenue v/s Profit / loss of the top 20 churned customers	6
Figure 5 Scatter plot of discount rate and revenue generated.....	7
Figure 6 Revenue in terms of number of years served as customer	7
Figure 7 Bar plot of profit/loss vs number of years as customer.	8
Figure 8 Bar plot of top longest serving customer in terms of revenue who got churned.....	8
Figure 9 Correlation Diagram of the features	9

8 LIST OF TABLES

Table 1 Results of model evaluation at seed = 300	14
Table 2 Results of model evaluation at seed = 5342	14
Table 3 Performance Evaluation of random forest model	14

9 BIBLIOGRAPHY

Bischl, B. et al., 2016. mlr: Machine Learning in R. *Journal of Machine Learning Research* , 17(15-066), pp. 1-5.

Dalinina, R., 2017. *Building Customer Churn Models for Business*. [Online] Available at: <https://www.datascience.com/blog/what-is-a-churn-analysis-and-why-is-it-valuable-for-business> [Accessed 10 04 2018].

Kelleher, J. D. & Namee, B., 2015. *Fundamentals of machine learning for predictive data analytics : algorithms, worked examples, and case studies*. s.l.:The MIT Press.

Klepac, G., Kopal, R. & Mrcic, L., 2014. *Developing churn models using data mining techniques and social network analysis*. s.l.:Information Science Reference.

Kumar, V. & Petersen, J. A., 2012. Customer Churn. In: *Statistical Methods in Customer Relationship Management*. s.l.:Chichester, UK: John Wiley & Sons, Ltd, pp. 149-165.

openml.org, n.d. *10-fold cross validation*. [Online] Available at: <https://www.openml.org/a/estimation-procedures/1>

Richter, J., 2017. *Parameter tuning with mlrHyperopt*. [Online] Available at: <https://mlr-org.github.io/Parameter-tuning-with-mlrHyperopt/>

Wikipedia, 2017. *Betfair*. [Online] Available at: <https://en.wikipedia.org/wiki/Betfair> [Accessed 08 April 2018].

10 APPENDICES

Appendix A:

```
table(is.na(DataSet))
# This code will give the output as a logical binary output with TRUE of FALSE
# values. Count of TRUE values indicates number of null values in the dataset.
```

Appendix B:

```
## Plotting the correlation
require(psych)

#NumericalSet <-subset(DataSet, select= -c(current_age))
NumericalSet <-DataSet
NumericalSet$ActiveStatus <- ifelse(NumericalSet$customer_flag=='Active',1,0)
NumericalSet <-subset(NumericalSet, select= -c(customer_flag))

cor_data <- cor(NumericalSet)

## Customizing the correlogram
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD",
"#4477AA"))
p.mat <- cor.mtest(cor_data)$p
corrplot(cor_data, method = "color", col = col(200),
type = "upper", order = "hclust", number.cex = .7,
addCoef.col = "black", # Add coefficient of correlation
tl.col = "black", tl.srt = 90, # Text label color and rotation
# Combine with significance
p.mat = p.mat, sig.level = 0.01, insig = "blank",
# hide correlation coefficient on the principal diagonal
diag = FALSE)
```

Appendix C:

```
# using method of spread
library(tidyr)
newDataSet <- churn_extra_features %>%
  gather(variable, value, -(mapping_primary_account_id:period)) %>%
  unite(temp, period, variable) %>%
  spread(temp, value, fill = 0)

newDataSet <- newDataSet[, -which(names(newDataSet) %in%
c("Last_To_Last_Month_prev_months_active", "Last_Month_prev_months_activ
e"))]

library(dplyr)
colnames(newDataSet)[1] <- "mapping_account_id"
colnames(newDataSet) <- make.names(colnames(newDataSet), unique=TRUE)
RaceData <- inner_join(RaceData, newDataSet, by="mapping_account_id")
```

Appendix D:

- Last month's data was converted in to a ratio of last month data and the previous months data.

- Last to last month's data was converted in to a ratio of last to last month and previous month's data.

Appendix E:

```
#set parallel backend
library(parallel)
library(parallelMap)
parallelStartSocket(cpus = detectCores())
```

Appendix F:

```
# Creating training and Testing dataset

Split<-0.80
SplitIndex <- sample.int(nrow(DataSet), round(nrow(DataSet) * Split))
TrainingData <- DataSet[SplitIndex,]
TestingData <- DataSet[-SplitIndex,]

classif.task <- makeClassifTask(id = "customer_flag", data = TrainingData,
target = "customer_flag",positive = "Active")

## Making the learner
library(data.table)
lrns = as.data.table(listLearners())

View (lrns)

library(kknn)
library(gbm)
library(e1071)
library(xgboost)

classif.lrn.knn <- makeLearner(cl = "classif.kknn", predict.type = "prob")
classif.lrn.ksvm <- makeLearner(cl = "classif.ksvm", predict.type = "prob")
classif.lrn.lda <- makeLearner(cl = "classif.lda", predict.type = "prob")
classif.lrn.gbm <- makeLearner(cl = "classif.gbm", predict.type = "prob")
classif.lrn.nnet <- makeLearner(cl = "classif.nnet", predict.type = "prob")
classif.lrn.svm <- makeLearner(cl = "classif.svm", predict.type = "prob")
classif.lrn.xgb <- makeLearner(cl = "classif.xgboost", predict.type = "prob")
classif.lrn.rpart <- makeLearner(cl = "classif.rpart", predict.type = "prob")
classif.lrn.rf <- makeLearner(cl = "classif.randomForest", predict.type =
"prob")
list <- c("kknn","ksvm","lda","Gradient Boosting Machine","neural
network","svm","rpart","random Forest")

#set parallel backend
library(parallel)
library(parallelMap)
parallelStartSocket(cpus = detectCores())

classif.lrn1 <- classif.lrn.knn
classif.lrn2 <- classif.lrn.ksvm
classif.lrn3 <- classif.lrn.lda
classif.lrn4 <- classif.lrn.gbm
classif.lrn5 <- classif.lrn.nnet
classif.lrn6 <- classif.lrn.svm
classif.lrn7 <- classif.lrn.rpart
```

```

classif.lrn8 <- classif.lrn.rf

j=1:8

library(foreach)
for (i in j) {
  classif.lrn <- (get(paste("classif.lrn", i, sep="")))

  ## Creating the model
  assign(paste("mod",i, sep=""), mlr::train( classif.lrn, classif.task))
  print(get(paste("mod", i, sep="")))

  ## Prediction

  assign(paste("pred",i, sep=""), predict(get(paste("mod", i, sep="")), newdata =
  TestingData))

  #listMeasures( classif.task)

  print(performance(get(paste("pred", i, sep="")),measure = list(mlr::fp,
  mlr::auc, mlr::mmce)))
  print(calculateConfusionMatrix(get(paste("pred", i, sep=""))))

  # creating a data table of mmce values
  mmce <- data.table(
    Model=list[j],
    Predictor=foreach (i = j, .combine="c") %do%
      paste("pred", i, sep=""),
    mmce=foreach (i = j, .combine="c") %do%
      print(performance(get(paste("pred", i, sep="")))),
    Accuracy=foreach (i = j, .combine="c") %do%
      print(performance(get(paste("pred", i, sep="")),measure = mlr::acc)),
    False_Positive=foreach (i = j, .combine="c") %do%
      print(performance(get(paste("pred", i, sep="")),measure = mlr::fp))
  )

  View(mmce)

```

Appendix G:

```

# mlrHyperOpt
# devtools::install_github("berndbischl/ParamHelpers") # version >= 1.11
# needed.
# devtools::install_github("jakob-r/mlrHyperopt", dependencies = TRUE)
library(mlrHyperopt)
library(parallel)
library(parallelMap)
parallelStartSocket(cpus = detectCores())
res = hyperopt(classif.task, learner = "classif.randomForest", show.info = TRUE)
res

```