Predicting Customer Churn Using WSDM - KKBox's Churn Data

MATH2319 Machine Learning Applied Project Phase II

s3398979 - Vijeta Tulsiyan

31.05.2018

Contents

1 Introduction

The objective of this project is to predict whether a customer of KKBox will churn after his/her subscription expires. KKBox offers subscription based music streaming service. We are required to forecast if a user makes a new service subscription transaction within 30 days after the current membership expiration date. The data is sourced from Kaggle's KKBOX Churn Prediction Challenge (https://www.kaggle.com/c/kkbox-churn-prediction-challenge).In Phase I, we cleaned and explored the data. We visualized the data for each predictor variable with respect to binary target variable (is_churn). In Phase II the cleaned data was saved in a new csv file named "Cleaned_KKBx.csv". This cleaned data had 1324001 rows and 18 variables. The ratio of binary target levels were 96% and 4% for "Not Churned" and "Churned", respectively. The dataset was highly imbalanced. A random sample was taken from the cleaned dataset in such a way that the proportion of binary target was 80% and 20% for "Not Churned" and "Churned", respectively. After this, a random sample of only 50,000 observations was taken maintaining the binary target labels' ratio as 80:20. We dropped one variable "payment_method_id" due to the error "New factor level" while working on R. Feature selection was done using five different binary-classifiers on the processed data. The rest of this report is organised as follow. Section 2 describes an overview of the methodology. Section 3 discusses the fine-tuning process and detailed performance analysis of each classifier. Section 4 compares the performance of the classifiers ROC curve. Benchmarking is performed on different learned to rank them according to their performance. Section 5 critiques our methodology. The last section concludes with a summary.

Feature Selection

We applied binary classifiers to the training data to find relevant features. However, for data modelling we chose to keep all 17 variables. We noticed that the predictor variables that were selected by majority classifiers were: city, bd (age), reg_via (subcription registration method), is_auto_renew (auto renewal of subscription of songs) and num_100 (number of times the song was palyed 100 percent of its length). Feature selection by these 5 classifiers:

Table 1: Percentage of Churn/No Churn Categories

Var1	Freq
0	0.79834
1	0.20166

```
## [1] "Features Selected using Naive Bayes Classifier are: "

## [1] "city"          "bd"            "reg_via"       "is_auto_renew"
## [5] "is_cancel"     "num_50"        "num_75"        "num_985"
## [9] "num_100"

## [1] "Features Selected using Decision Tree Classifier are: "

## [1] "city"                "bd"                  "reg_via"
## [4] "payment_plan_days"   "actual_amount_paid"  "is_auto_renew"
## [7] "num_100"             "num_unq"

## [1] "Features Selected using Random Forest Classifier are: "

##  [1] "city"                "bd"                  "reg_via"
##  [4] "payment_plan_days"   "actual_amount_paid"  "is_auto_renew"
##  [7] "num_25"              "num_75"              "num_unq"
## [10] "total_secs"

## [1] "Features Selected using Gradient Boost Classifier are: "

## [1] "payment_plan_days" "plan_list_price"   "date_diff_days"
## [4] "num_25"            "num_50"            "num_unq"
## [7] "total_secs"

## [1] "Features Selected using Logistic Regression Classifier are: "

##  [1] "city"              "bd"                "reg_via"
##  [4] "payment_plan_days" "is_auto_renew"     "is_cancel"
##  [7] "date_diff_days"    "num_50"            "num_985"
## [10] "num_100"           "total_secs"

## [1] "The mmce for Naive Bayes Classifier: "

## mmce.test.mean
##          0.192

## [1] "The mmce for Decision Tree Classifier: "

## mmce.test.mean
##      0.1907428

## [1] "The mmce for Random Forest Classifier: "

## mmce.test.mean
##      0.1683143

## [1] "The mmce for Gradient Boost Classifier: "

## mmce.test.mean
##      0.2016571

## [1] "The mmce for Logistic Regression Classifier: "

## mmce.test.mean
##       0.189743
```

2 Methodology

We considered five classifiers - Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), Gradient Boosting Machine (GBM) and Logistic Regression (LogReg). The NB was the baseline classifier. Each classifier was trained to make probability predictions so that we were able to adjust prediction threshold to refine the performance. We split the full data set into 70 % training set and 30 % test set. Each set resembled the full data by having the same proportion of target classes i.e. approximately 80 % of customer who didn't churn and 20% of customers who churned. For fine-tuning process, we ran a five-folded cross-validation stratified sampling on each classifier. Stratified sampling was used to cater the slight imbalance class of the target feature.

Next, for each classsifer, we determined the optimal probability threshold. Using the tuned hyperparameters and the optimal thresholds, we made predictions on the test data. During model training (hyperparameter tuning and threshold adjustment), we relied on mean misclassification error rate (mmce) and area under teh curve (auc). In addition to these, we also used the confusion matrix on the test data to evaluate classifiers' performance. We performed benchmarking and ranking to compare the performance fo different models. The modelling was implemented in `R` with the `mlr` package [@mlr].

3 Hyperparameter Tune-Fining

3.1 Naive Bayes

Since the training set might have unwittingly excluded rare instances, the NB classifier might produce some fitted zero probabilities as predictions. To mitigate this, we ran a grid search to determine the optimal value of the Laplacian smoothing parameter. Using the stratified sampling discussed in the previous section, we experimented values ranging from 0 to 30.

```
## [1] "The optimal Laplacian parameter are: "

## $laplace
## [1] 20

## [1] "The mean test error and auc are: "

## mmce.test.mean  auc.test.mean
##      0.7947429      0.6635903
```

3.2 Decision Tree

We tune-fined parameter "minsplit" which represents the minimum number of observation in a node for a split to take place. We experimented values ranging from 20 to 40 using the stratified sampling.

```
## [1] "The optimal minsplit parameter is: "

## $minsplit
## [1] 38

## [1] "The mean test error and auc are: "

## mmce.test.mean  auc.test.mean
##      0.1911142      0.6637514
```

3.3 Random Forest

We tune-fined the number of variables randomly sampled as candidates at each split (i.e. `mtry`). For a classification problem, @Breiman suggests `mtry` $= \sqrt{p}$ where p is the number of descriptive features. In our case, $\sqrt{p} = \sqrt{17} = 4.12$. Therefore, we experimented `mtry` $= 3$, 4, and 5. We left other hyperparameters, such as the number of trees to grow at the default value.

```
## [1] "The optimal mtry parameter is: "

## $mtry
## [1] 5

## [1] "The mean test error and auc are: "

## mmce.test.mean  auc.test.mean
##      0.1621142      0.7846575
```

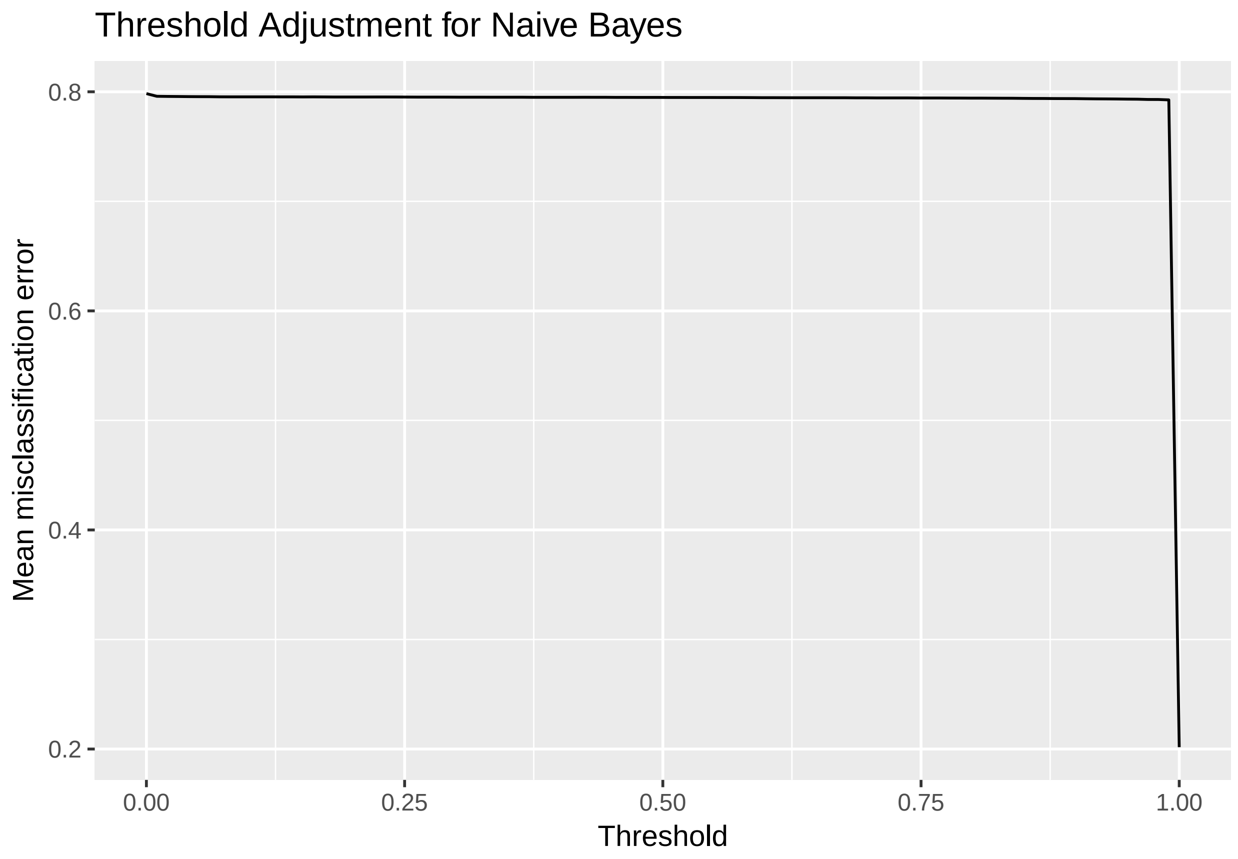
3.4 GBM (Adaboost)

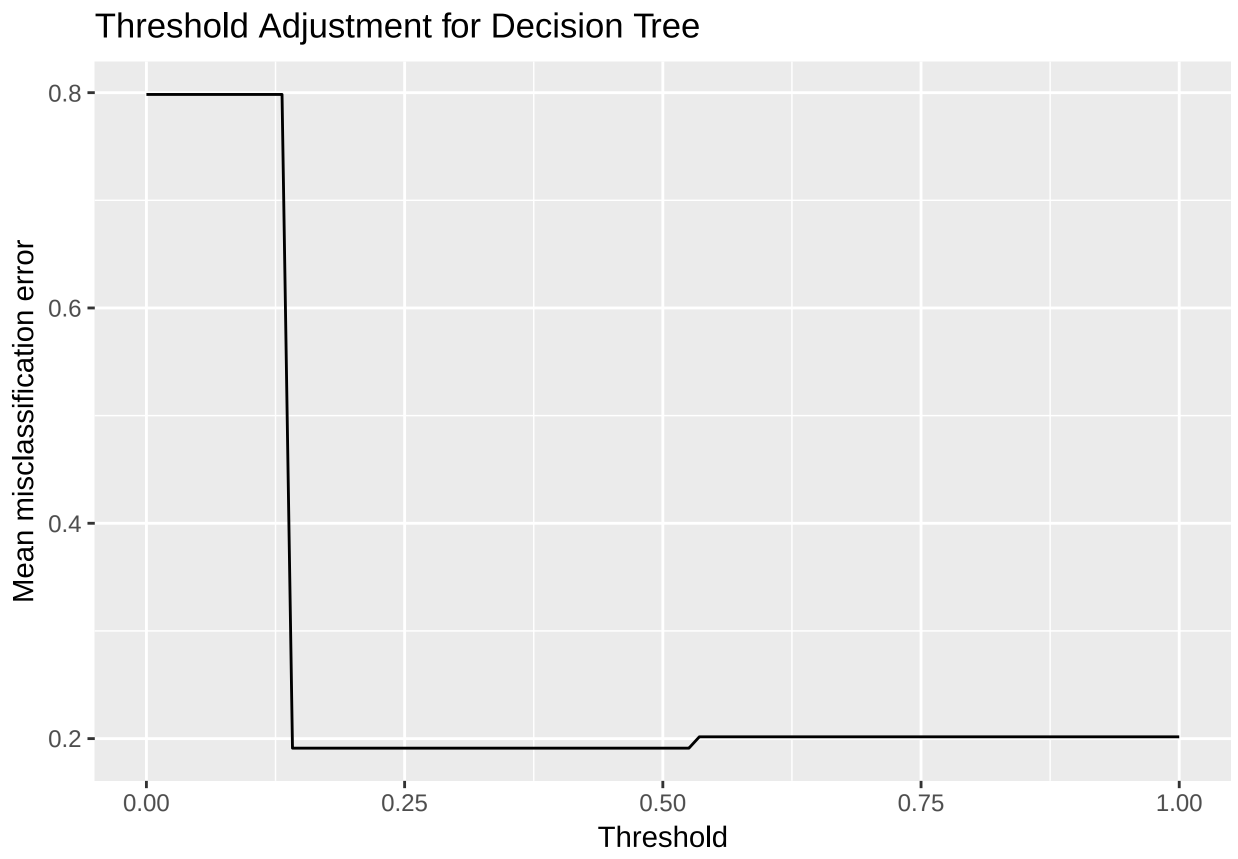
For GBM, the default distribution is "Bernoulli's". For number of trees, we experimented values ranging from 300 to 400 using the stratified sampling.

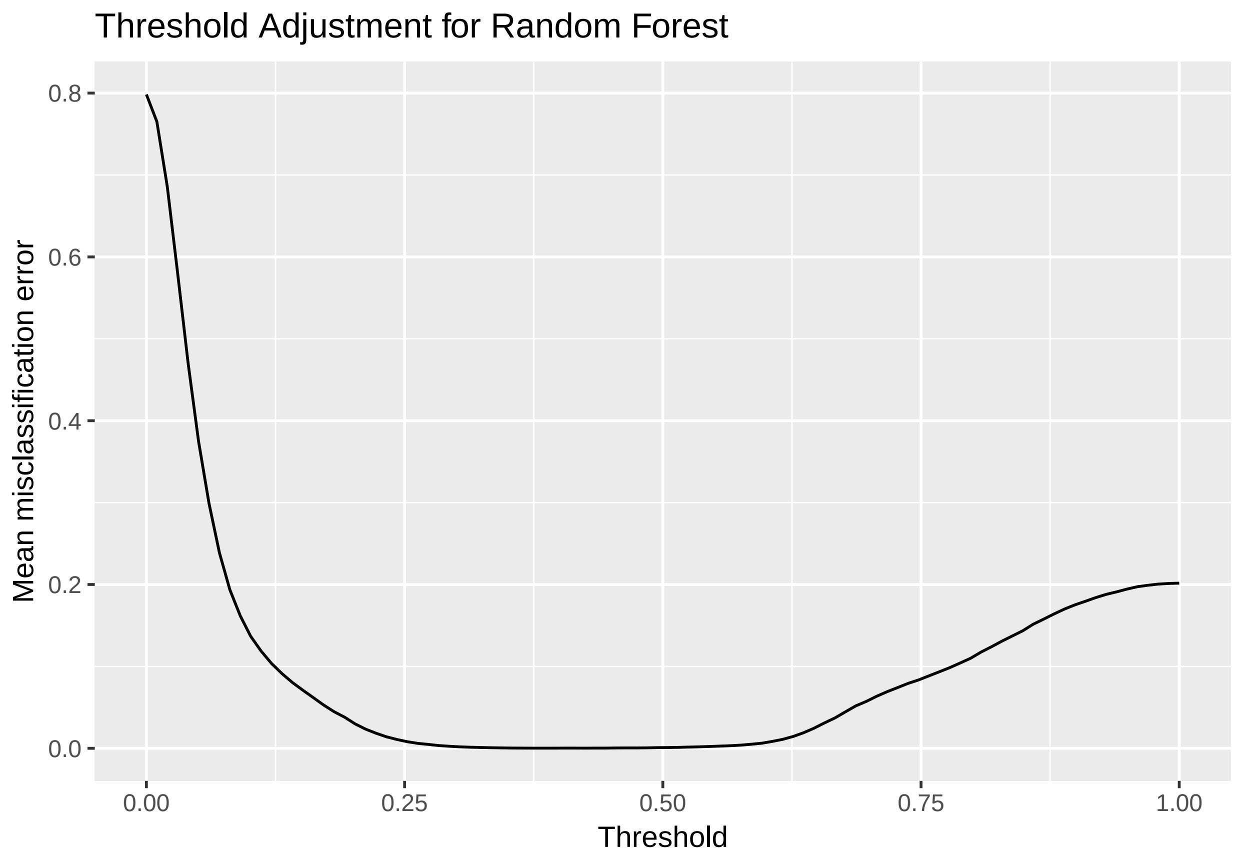
```
## [1] "The optimal number of tree parameter is: "
## $distribution
## [1] "adaboost"
##
## $n.trees
## [1] 322

## [1] "The mean test error and auc are: "

## mmce.test.mean  auc.test.mean
##      0.2016571      0.6637494
```

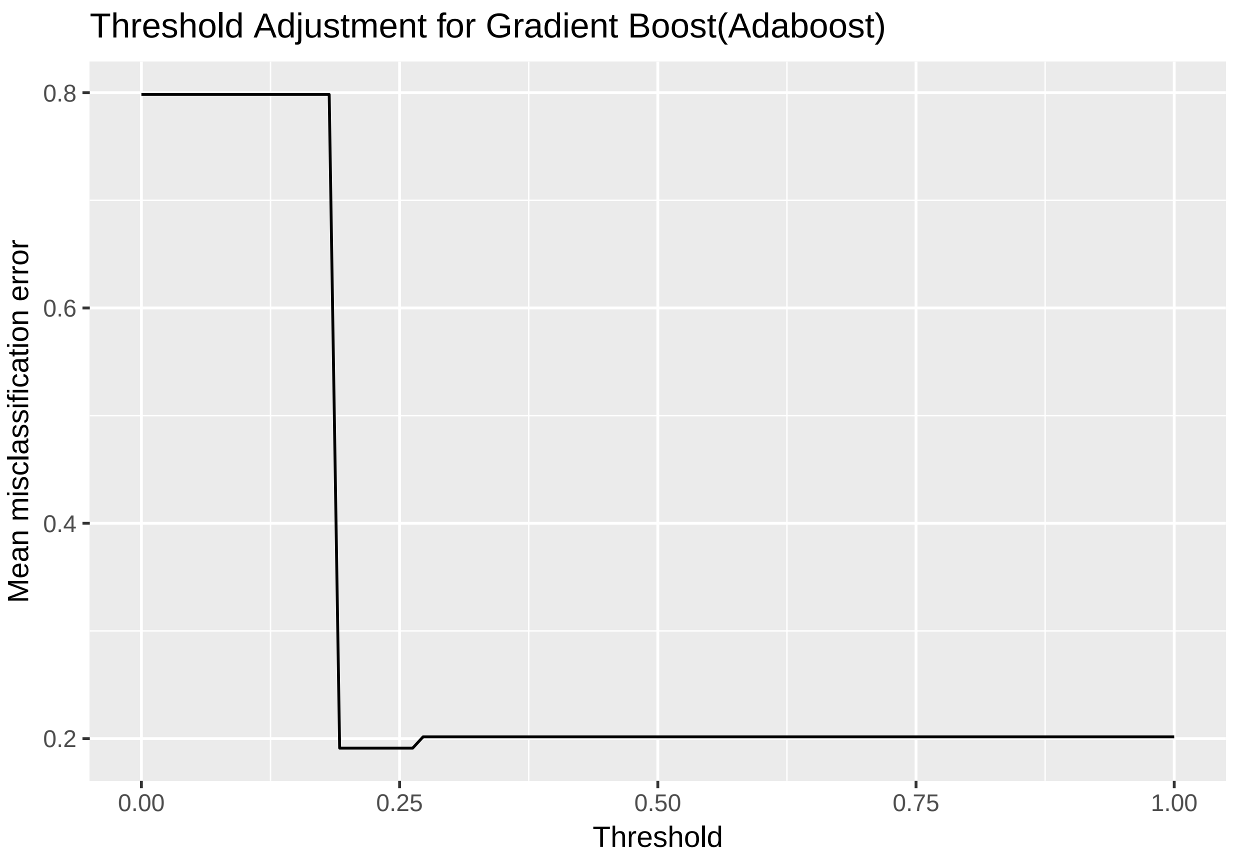
3.5 Threshold Adjustment

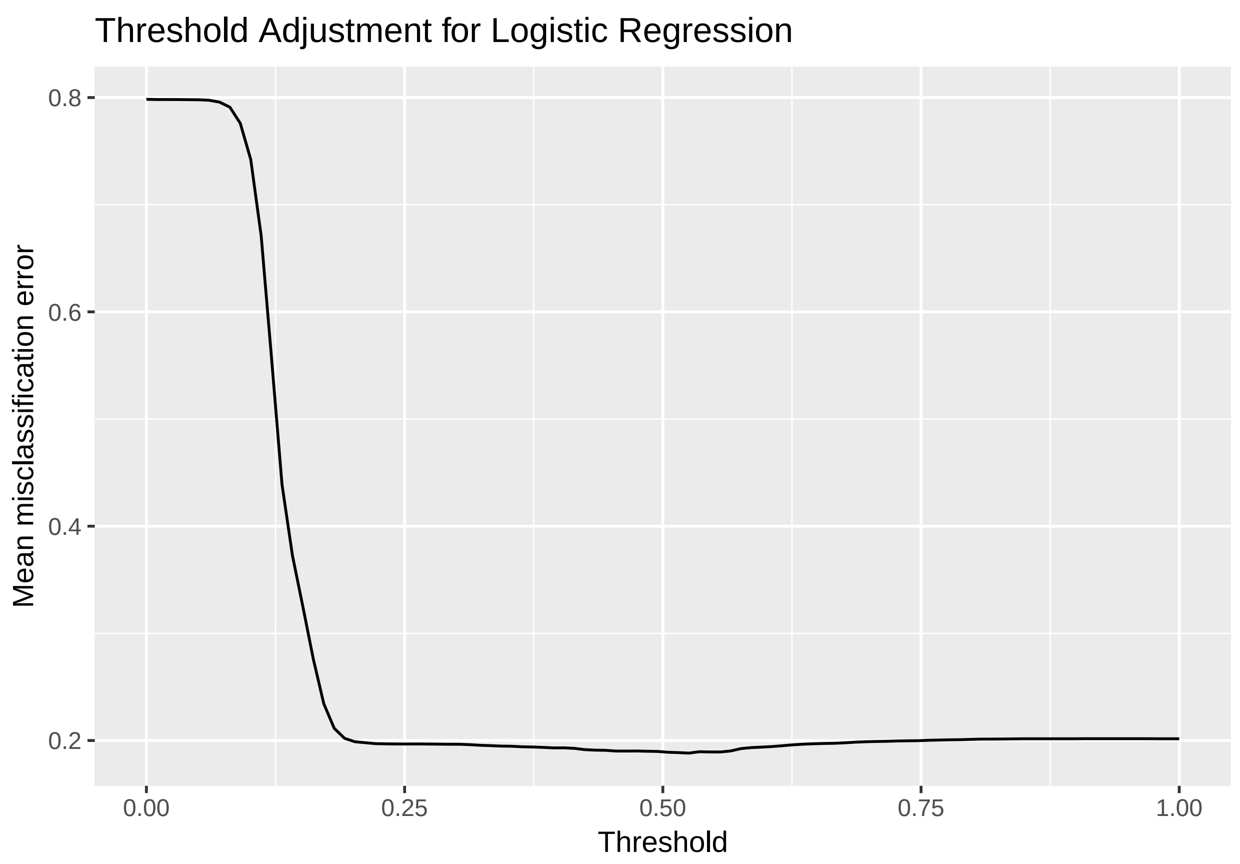
The following plots depict the value of mmce vs. the range of probability thresholds. These thresholds were used to determine the probability of a customer churning out. The threshhold for each classifier areas follows:




Threshold Adjustment for Decision Tree
Mean misclassification error
0.8
0.6
0.4
0.2
0.00
0.25
0.50
0.75
1.00
Threshold


Threshold Adjustment for Random Forest
Mean misclassification error
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
Threshold


Threshold Adjustment for Gradient Boost(Adaboost)
Mean misclassification error
0.8
0.6
0.4
0.2
0.00
0.25
0.50
0.75
1.00
Threshold



```
## [1] "The threshold for Naive Bayes classifier is: "

## [1] 1

## [1] "The threshold for Decision Tree classifier is: "

## [1] 0.1414141

## [1] "The threshold for Random Forest classifier is: "

## [1] 0.3737374

## [1] "The threshold for GBM classifier is: "

## [1] 0.1919192

## [1] "The threshold for Logistic Regression classifier is: "

## [1] 0.5252525
```

4 Evaluation

Using the parameters and threshold levels, we calculated the confusion matrix for each classifier.

The confusion matrix of NB classifer is as follow:

```
## Relative confusion matrix (normalized by row/column):
##         predicted
## true     0         1         -err.-
##   0      1.00/0.80 0.00/0.00 0.00
##   1      1.00/0.20 0.00/0.00 1.00
##   -err.-      0.20      0.00 0.20
##
##
## Absolute confusion matrix:
##         predicted
## true         0 1 -err.-
##   0      11975 0      0
##   1       3025 0   3025
##   -err.-  3025 0   3025
```

The confusion matrix of DT classifer is as follow:

```
## Relative confusion matrix (normalized by row/column):
##         predicted
## true     0         1         -err.-
##   0      0.90/0.86 0.10/0.47 0.10
##   1      0.57/0.14 0.43/0.53 0.57
##   -err.-      0.14      0.47 0.19
##
##
## Absolute confusion matrix:
##         predicted
## true         0    1 -err.-
##   0      10810 1165   1165
##   1       1712 1313   1712
##   -err.-  1712 1165   2877
```

The confusion matrix of RF classifer is as follow:

```
## Relative confusion matrix (normalized by row/column):
##         predicted
## true     0         1         -err.-
##   0      0.90/0.89 0.10/0.42 0.10
##   1      0.44/0.11 0.56/0.58 0.44
##   -err.-      0.11      0.42 0.17
##
##
## Absolute confusion matrix:
##         predicted
## true         0    1 -err.-
##   0      10754 1221   1221
##   1       1335 1690   1335
##   -err.-  1335 1221   2556
```

The confusion matrix of GBM classifer is as follow:

```
## Relative confusion matrix (normalized by row/column):
##         predicted
## true     0         1         -err.-
##   0      0.90/0.86 0.10/0.47 0.10
##   1      0.57/0.14 0.43/0.53 0.57
##   -err.-      0.14      0.47 0.19
##
##
## Absolute confusion matrix:
##         predicted
## true         0    1 -err.-
##   0      10810 1165   1165
##   1       1712 1313   1712
##   -err.-  1712 1165   2877
```

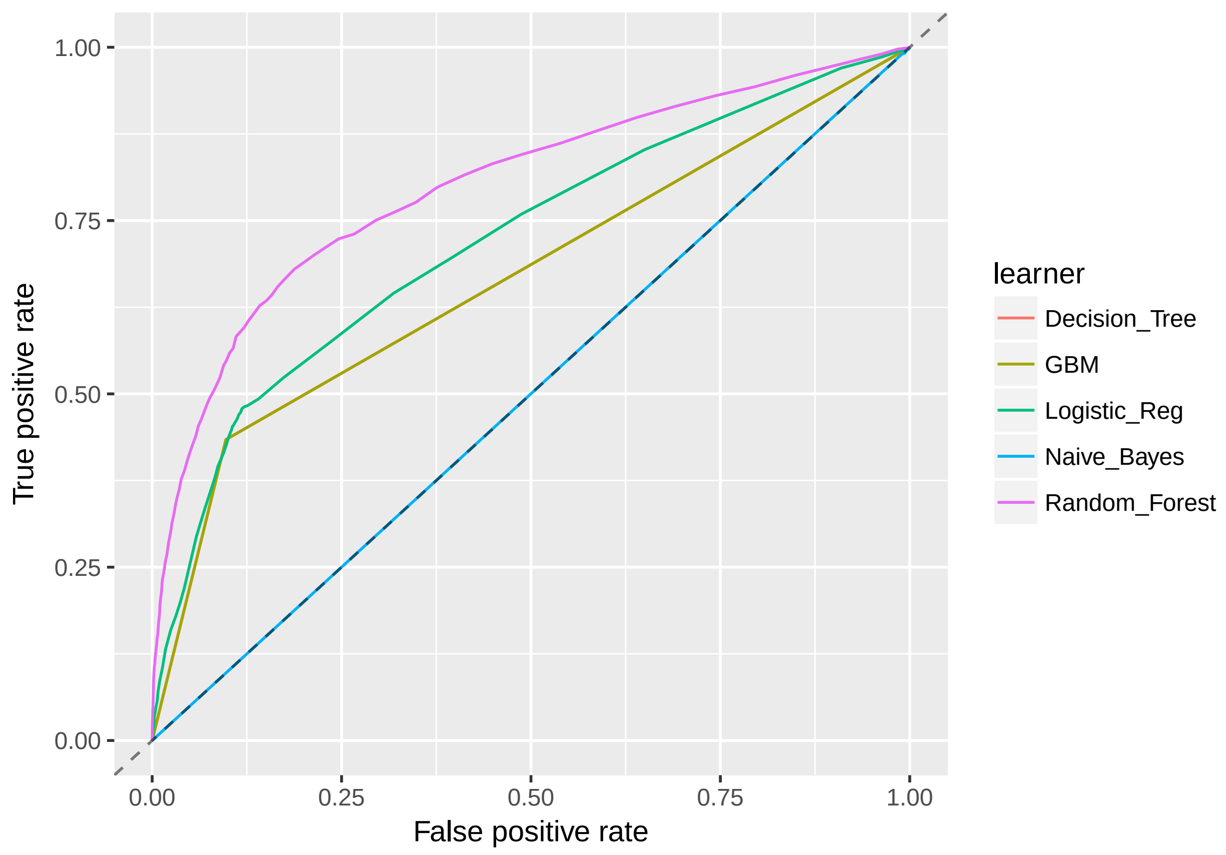
The confusion matrix of Logistic_Reg classifer is as follow:

```
## Relative confusion matrix (normalized by row/column):
##         predicted
## true     0         1         -err.-
##   0      0.96/0.83 0.04/0.43 0.04
##   1      0.78/0.17 0.22/0.57 0.78
##   -err.-      0.17      0.43 0.19
##
##
## Absolute confusion matrix:
##         predicted
## true         0   1 -err.-
##   0      11466 509    509
##   1       2358 667   2358
##   -err.-  2358 509   2867
```

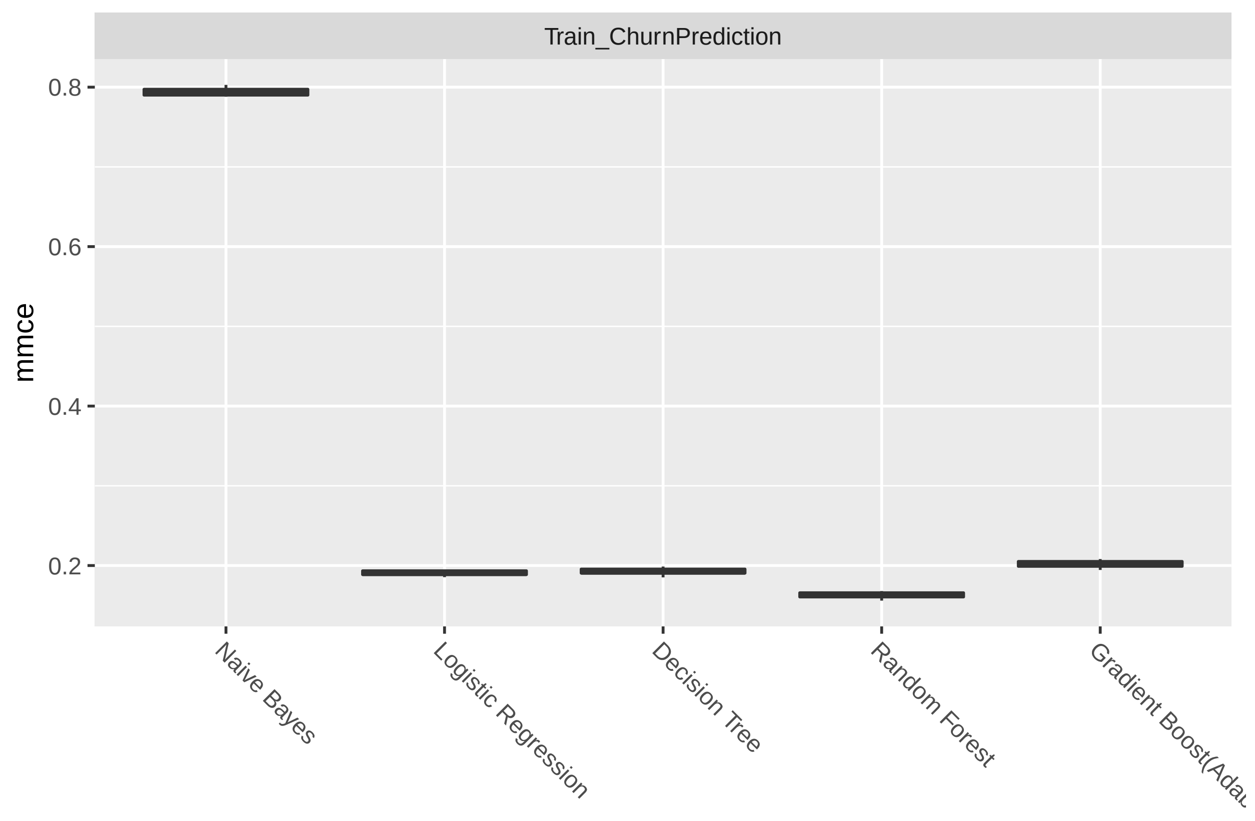
5 Benchmarking

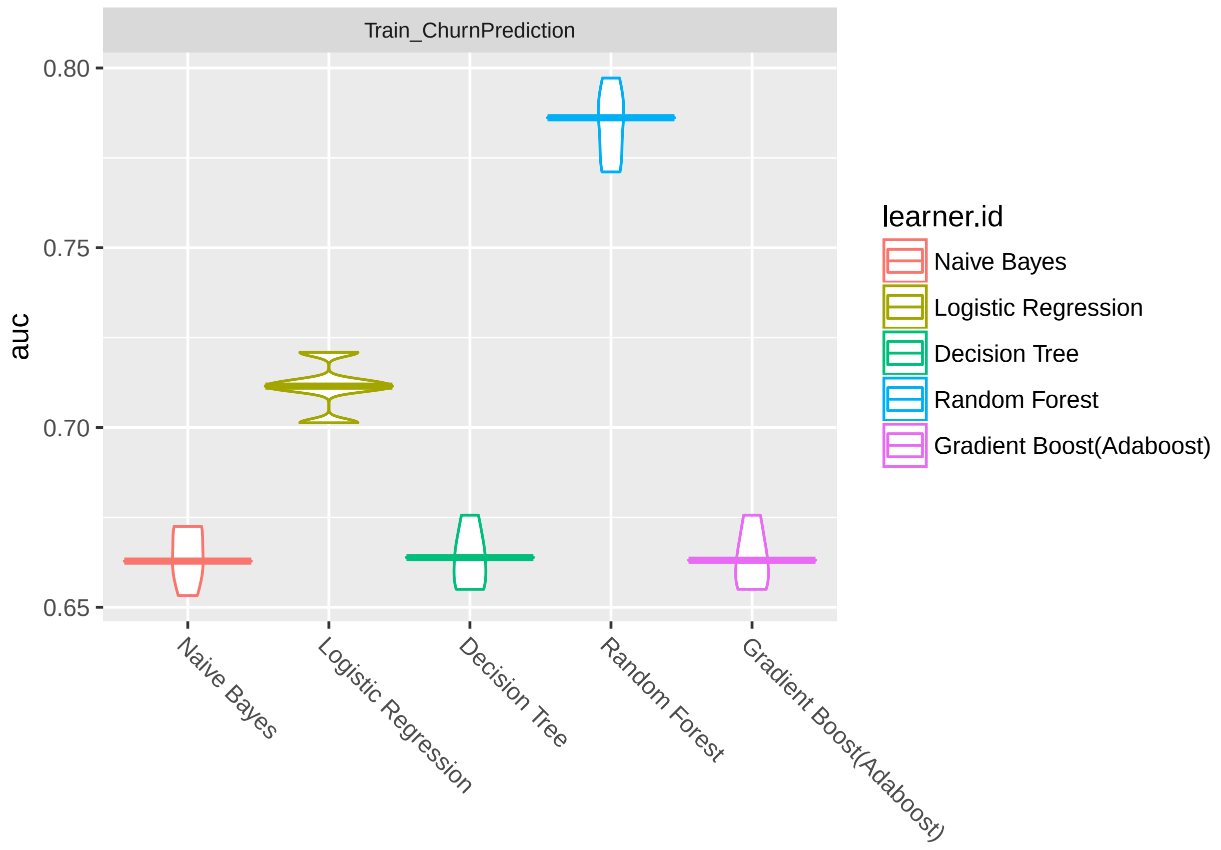
```
## [1] "Performance measures of Naive Bayes classifier is: "

##      mmce       tpr       auc
## 0.2016667 0.0000000 0.6723517

## [1] "Performance measures of Decision Tree classifier is: "

##      mmce       tpr       auc
## 0.1918000 0.4340496 0.6683818

## [1] "Performance measures of Random Forest classifier is: "

##      mmce       tpr       auc
## 0.1704000 0.5586777 0.7998591

## [1] "Performance measures of GBM classifier is: "

##      mmce       tpr       auc
## 0.1918000 0.4340496 0.6683818

## [1] "Performance measures of Logistic Regression classifier is: "

##      mmce       tpr       auc
## 0.1911333 0.2204959 0.7212293
```

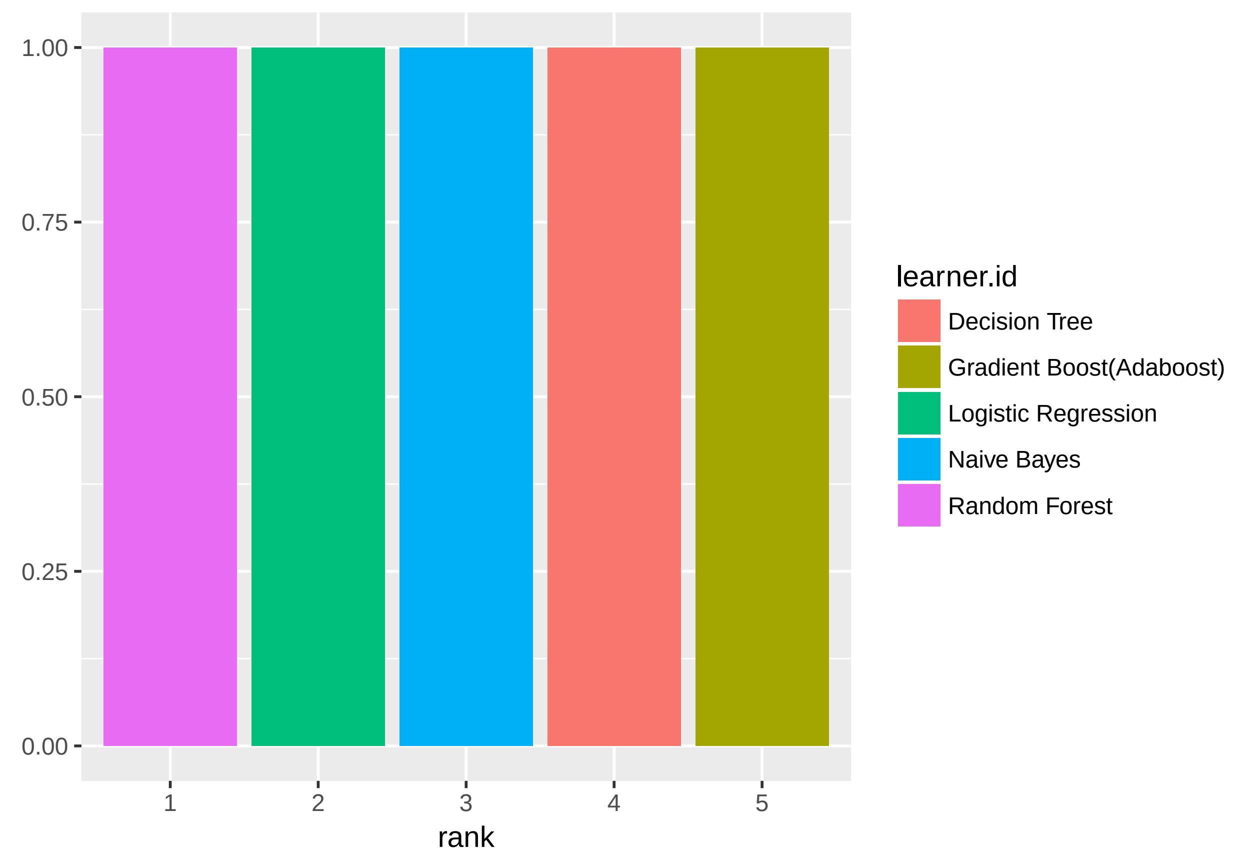
We plotted ROC curve to visualise the the area under the curve (auc) and analysed the performance of each of the five classifiers. It is evident that of the five, Random Forest is the best performing model.



We also plot boxplot and violin plot to benchmark the performance of the classifiers. By default, benchmark is mlr pick the first measure in the list, which is auc in this case. We can change it to mmce if needed. We made a rank matric to rank their performance. Random Forest model stood first.


Train_ChurnPrediction
mmce
0.8
0.6
0.4
0.2
Naive Bayes
Logistic Regression
Decision Tree
Random Forest
Gradient Boost(Ada


Train_ChurnPrediction
0.80
0.75
0.70
0.65
auc
learner.id
Naive Bayes
Logistic Regression
Decision Tree
Random Forest
Gradient Boost(Adaboost)



# Discussion

The previous section showed that all classifiers did not perform accurately in predicting the customer churn despite the stratified sampling. This suggested class imbalance. A better approach would be a cost-sensitive classification where we could have allocated more cost to true positive groups i.e. the correctly predicted churn class. Another alternative would be under- or oversampling to adjust the class balance, despite the risk of inducing biases. These methods deal with handling imbalanced data by resampling original data to provide balanced classes. We can use ensemble methods such as bagging and boosting on the imbalaced dataset. Bagging handles overfitting nicely. Unlike boosting, bagging allows replacement in the bootstrapped sample. Random Forest uses bagging approach and GBM uses boosting approach to deal with imbalanced datasets. For this dataset, GBM model performed badly, as evident from confusion matrix. Random model performed better with hyper parameter tuning. Logistic regression was the second best performer with deafult parameters. We can try all 5 models using respective features selected (as shown above) to check if it helps to improve the performance.

6 Conclusion

Among five classifiers, the Random Forest produces the best performance in predicting churn. We split the data into training and test sets. Via a stratified sampling, we determined the optimal value of the selected hyperparameter of each classifier and the probability threshold. Despite this, the imbalance class issue still persisted and therefore reduced the class accuracy of the chustomer curned. Bagging method seems to perform well, but we can consider cost-sensitive classification and under/over-sampling methods to check if it further improves the result or not.

7 Reference

*www.kaggle.com

*RMIT University Lecture and Tute notes