Project Report

WSDM - KKBox's Churn Prediction Challenge

Chetan Kumar

1. Reflection statement about project benefits to company/ community

The task or goal of our project is to predict churn of subscribers for KKbox which is a music streaming service and KKbox being a subscription business, accurately predicting churn is critical to long term success and any variation in churn can drastically affect profits. So, coming up with a model to accurately predict churn can help KKbox try and retain more customers and increase their profits.

2. Abstract or executive summary

Using Machine Learning and Business Analytics techniques in businesses is increasingly gaining popularity for their potential to evolve workspaces and change entire industries. Here we will be using machine learning techniques to predict churn of subscribers for a music streaming service called KKbox. We use supervised machine learning techniques such as Logistic regression, Decision trees, neural networks, SVM and random forest to predict the target class “churn” and show the obtained results. We compare these results and evaluate the models in this project. We also visualize different aspects of data to determine any trends or patterns.

3. Introduction

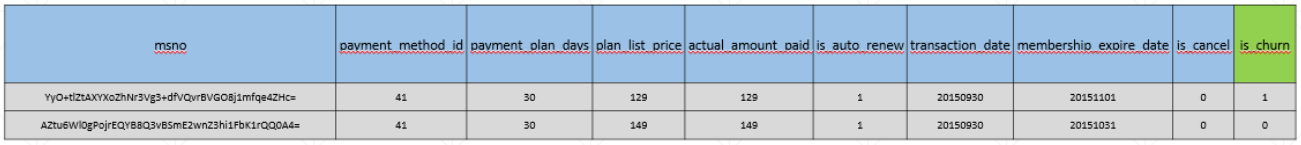
Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed [1]. In simple terms machine learning are processes which fallow set of rules designed for problem solving. They do so by finding patterns in data. The Machine learning tasks can be classified into two broad categories, Supervised and Unsupervised learning. Our task in the project falls under supervised learning as we have labeled data and a predefined output class with discrete outputs. The machine learning task of our project is classification, where inputs needs to be divided into two or more classes (two classes in our case, Binary classification) and our goal is to come up with a model to assign unknown values to these classes. Our data consisted of customers with several attributes as input and the predicted class is Churn or No Churn.

The dataset we chose was obtained from Kaggle’s competition dataset and it was a challenge from the ACM International Conference on Web Search and Data Mining (WSDM 2018) to build an algorithm which would predict if the subscriber will churn or not on a dataset donated by KKbox. KKbox is Asia’s leading music streaming service, holding world’s most comprehensive Asia pop-music library with more than 40 million tracks. They provide access to their music library to millions of users supported by advertisement and paid subscriptions. This model is hugely dependent on accurately predicting the churn of their paid users. Currently KKbox makes use of survival analysis techniques to determine the residual membership life time for each subscriber.

Our task here was to come up with a model which would accurately predict churn. We tried multiple machine learning algorithms and we chose the best out of them. We tried five algorithms which include Logistic regression, Decision trees, neutral networks, random forest and SVM. We try different parameters based on the model, like varying the split ratio of train and test, varying the number of trees used in decision trees, number of hidden layers in neural networks etc., to check if the accuracies varied and we record the best results obtained. We also show some interesting visualization to find some interesting patterns in the data.

4. Dataset Description

Our dataset consists of two files, the train dataset and the transaction dataset. The train dataset consists of 2 variables and the transaction dataset consists of 9 variables. We merged the two datasets by the common variable msno present in both the datasets. We do so to add the is_churn column from the train dataset to the transaction dataset. The merged dataset consisted of 10 variables.



msno	payment_method_id	payment_plan_days	plan_list_price	actual_amount_paid	is_auto_renew	transaction_date	membership_expire_date	is_cancel	is_churn
YyO+tlZtAXYXoZhNr3Vg3+dfVQvrBVGO8j1mfqe4ZHc=	41	30	129	129	1	20150930	20151101	0	1
AZtu6Wl0gPojrEQYB8Q3vBSmE2wnZ3hi1FbK1rQQ0A4=	41	30	149	149	1	20150930	20151031	0	0

Attribute information:

- msno: user_id
- payment_method_id: payment method
- payment_plan_days: length of membership plan in days
- plan_list_price: in New Taiwan Dollar (NTD)
- actual_amount_paid: in New Taiwan Dollar (NTD)
- is_auto_renew: does the user auto renew this subscription or not
- transaction_date: format %Y%m%d
- membership_expire_date: format %Y%m%d
- is_cancel: whether or not the user canceled the membership in this transaction.
- is_churn:1: churn and 0: no churn

The class to be predicted here is is_churn and the churn distribution is 90% of data is no churn and 10% is churn as seen in the pi-chart (figure 1). The class distribution of our data was, users churn only 10% of the time and they don't churn 90% of the time. There is a biasness in the class distribution, where majority of the data is for no churn.

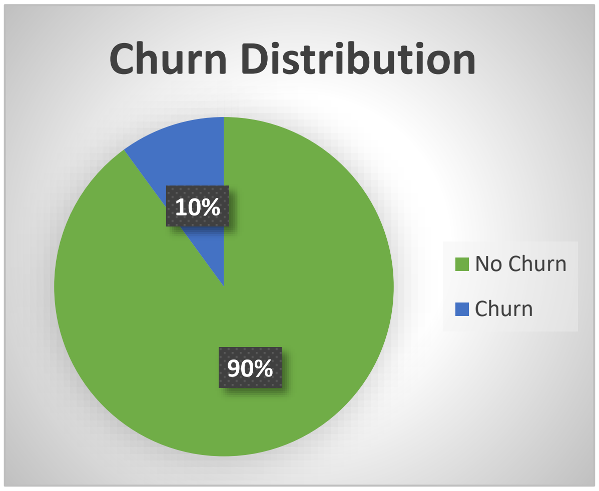


Figure:1

5. Literature survey related to the project

Classification is a task of categorization based on similar or shared qualities and characteristics. For the binary classification task, we have made use of 5 machine learning algorithms here.

The first method we used is logistic regression where the dependent variable is categorical, and in our case the output variable is a binary dependent variable where the output is either “0” or “1” which represent no_churn and churn. Logistic regression works by measuring the relationship between the dependent variable and multiple independent variables by estimating probabilities using the logistic function. Logistic regression is a special case of the generalized linear model.
The second model was Decision Trees. They are a type of Supervised Learning method, which works for both categorical and continuous input and output variables. Decision trees are graphic module of Decision Matrix Analysis, and consist of three types of nodes: decision nodes, chance event nodes and terminating nodes. They split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter or differentiator in input variables and if there is a high non-linearity and complex relationship between dependent and independent variables, a tree model will outperform a classical regression method.
The third machine learning model implemented was Neural Networks. An Artificial Neural Network (ANN) is an information processing paradigm; inspired by the way biological nervous systems, such as the brain processes information. It is composed of many highly interconnected processing elements (neurons) working in unison to solve specific problems. Neural Network model was chosen on our datasets is because of its adaptive learning ability which learns how to do tasks based on the data given for training or initial experience. Also, self-organization, where an ANN creates its own organization or representation of the information it receives during learning time. ANN also supports Real Time Operation where computations can be carried out in parallel, along with special hardware devices being designed and manufactured taking advantage of this capability.
The fourth method, Random Forest is an ensemble learning method for tasks like classification and regression that work by building a multitude of decision trees at training and outputting the overall class for classification and mean prediction for regression obtained by the individual trees. Random forest overcomes a common problem with decision trees of overfitting the training set.
The final method, Support vector machines (SVM) is a supervised learning model with algorithms used for classification and regression analysis tasks. SVM is a non-probabilistic binary linear classifier. In SVM, the representation of observations are as points in space, mapped so that operations of other categories are separated by a clear gap which is as wide as possible. New observations are mapped into the same space and their category is determined by the side of the gap or the space where they fall in.

6. Project methodology

The methodology or the steps we followed were:

I. Dataset selection: Our first step was to select appropriate dataset which suited the project requirement and also allow us to implement the techniques and concepts we learnt in class.

II. Dataset cleaning: The dataset we chose had multiple CSV files which needed to be merged and also contained duplicate values which had to be removed to obtain a clean dataset on which the models can be applied.

III. Applying different machine learning models: Our first task here was to determine the models/methods which would suit our task of binary classification.

IV. Evaluating and comparing the results: Here we evaluate the obtained results by trying various parameters on different models. We also compare the results obtained by different methods.

V. Visualizations: We also created the visualizations on the dataset to find interesting patterns which we did not find using only machine learning techniques.

7. Analysis and Results

Logistic Regression

First we applied logistic regression onto the KKbox dataset. For the evaluation matrics we used different dataset splits for training and testing set. We used 70%, 80% and 90% for training while 30%, 20%, 10% for testing respectively. Results for 70% training and 30% testing are obtained as following.

```
summary(lrmodel)

##
## Call:
## glm(formula = is_churn ~ +payment_method_id + payment_plan_days +
##     plan_list_price + actual_amount_paid + is_auto_renew + transaction_dat
e +
##     membership_expire_date + is_cancel, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.2873  -0.2867  -0.2722   2.9597
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             3.142e+03  3.601e+01   87.26   <2e-16 ***
## payment_method_id      -2.598e-02  1.250e-03  -20.78   <2e-16 ***
## payment_plan_days       4.228e-02  6.902e-04   61.26   <2e-16 ***
## plan_list_price         1.129e-02  2.557e-04   44.13   <2e-16 ***
## actual_amount_paid     -1.400e-02  2.315e-04  -60.50   <2e-16 ***
## is_auto_renew          -1.794e+00  1.339e-02 -133.96   <2e-16 ***
## transaction_date       -1.380e-04  7.199e-07 -191.74   <2e-16 ***
## membership_expire_date -1.781e-05  1.326e-06  -13.43   <2e-16 ***
## is_cancel               3.505e+00  1.504e-02  233.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 515933  on 792625  degrees of freedom
## Residual deviance: 366475  on 792617  degrees of freedom
## AIC: 366493
##
## Number of Fisher Scoring iterations: 7
```

From the summary of logistic regression we can see all the variables are significant in predicting churn.

```
tab3
```

```
##           Actual
## predicted      0      1
##         0 707463  31072
##         1   5765  48326

# Miss classification error for training data
(1-sum(diag(tab3))/sum(tab3)) * 100

## [1] 4.647463

# Accuracy
(sum(diag(tab3))/sum(tab3)) * 100

## [1] 95.35254

# Confustion matrix for test data
tab4

##           Actual
## predicted      0      1
##         0 301091  21992
##         1   4144  12183

# Miss classification error for test data
(1-sum(diag(tab4))/sum(tab4)) * 100

## [1] 7.661313

# Accuracy
(sum(diag(tab4))/sum(tab4)) * 100

## [1] 92.33869
```

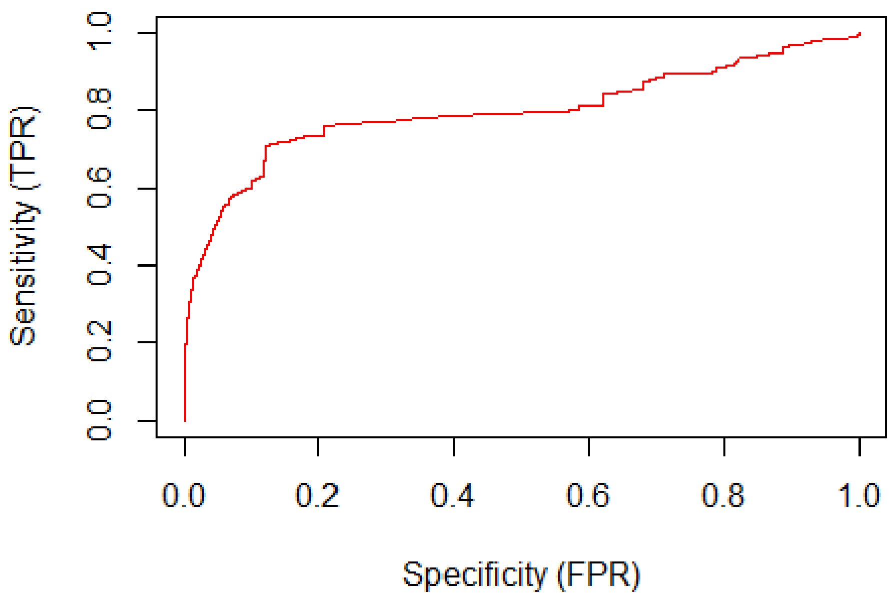
Training accuracy obtained from logistic regression is around 92.3% while error is 7.67%. Test accuracy obtained is 92.29% while error is 7.7%.

		Actual	
		Positive	Negative
Predicted	Positive	12183	4144
	Negative	21992	301091

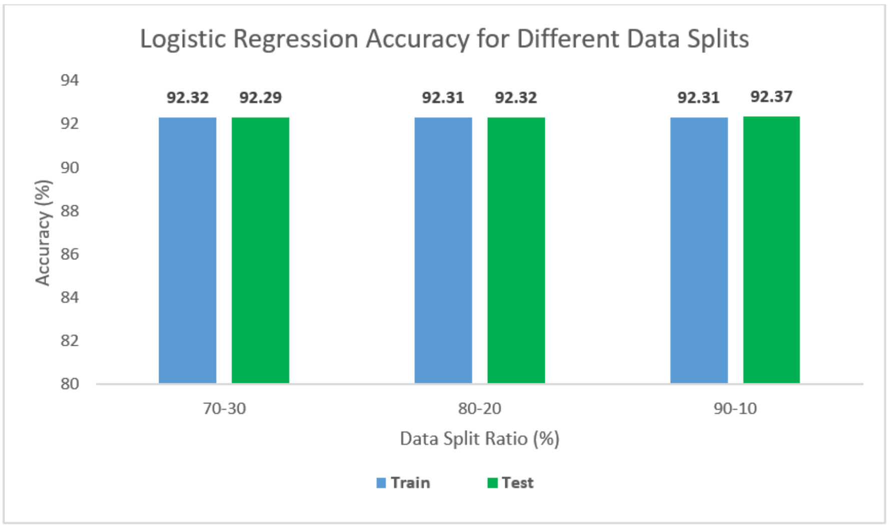
From above confusion matrix we can see most of the negative class values are predicted correctly while most positive values are predicted incorrectly which is why accuracy of the model is good overall. Next we will plot ROC curve in order to examine sensitivity and specificity which gives ratio between true positive rate and false positive rate.

ROC Curve

```
plot(perf, ylab='Sensitivity (TPR)', xlab='Specificity (FPR)', col="red")
```



We calculated ROC curve for the logistic regression model. ROC curve shows relation between true positive and false positive rate. From this ROC curve we got AUC (Area Under the Curve) is around 79.8%. It shows that how many true positives are predicted from the all the predicted positives.



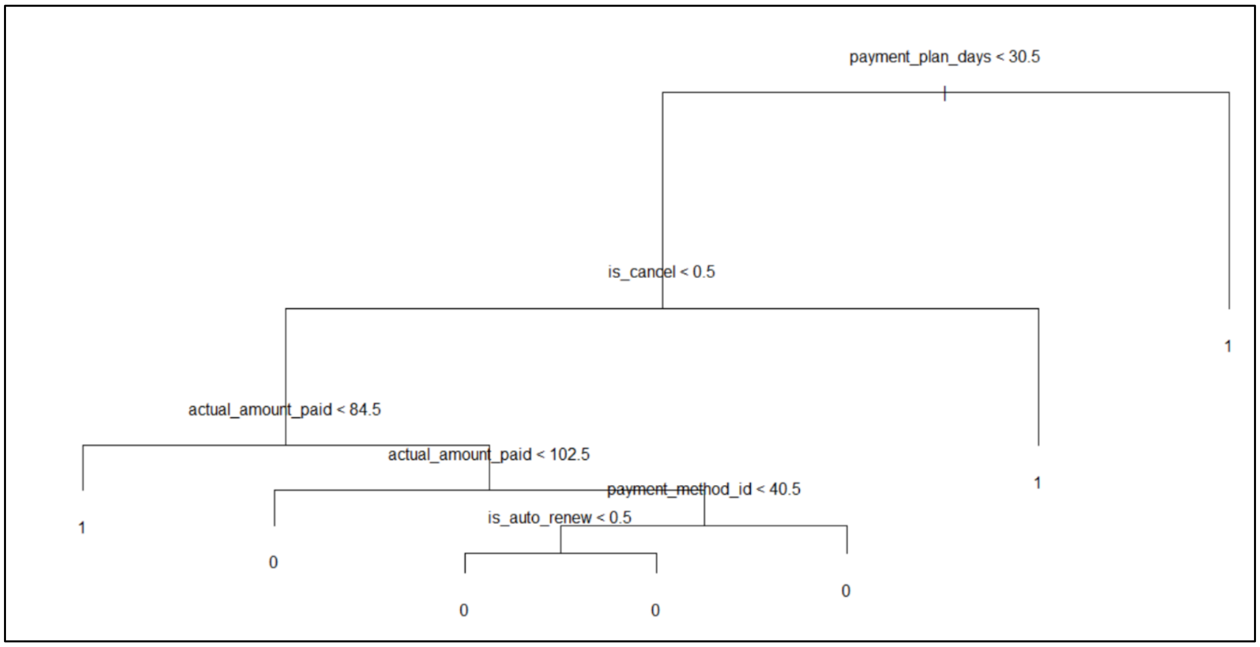
Further we tested our model on different train and test split and fond the following results as shown in above graph. We can see that training and testing accuracy for each split remains almost same, there is no any significant change between the accuracy if we change the split ratio.

Decision Tree

Second model we applied was decision tree, in order to see if there is any increase in the performance.

```
tree_model <- tree(is_churn ~ + payment_method_id + payment_plan_days +
                      plan_list_price + actual_amount_paid + is_auto_renew +
                      transaction_date + membership_expire_date + is_cancel,
                    data = train, method = 'class')
```

```
plot(tree_model)
text(tree_model, pretty = 0)
```



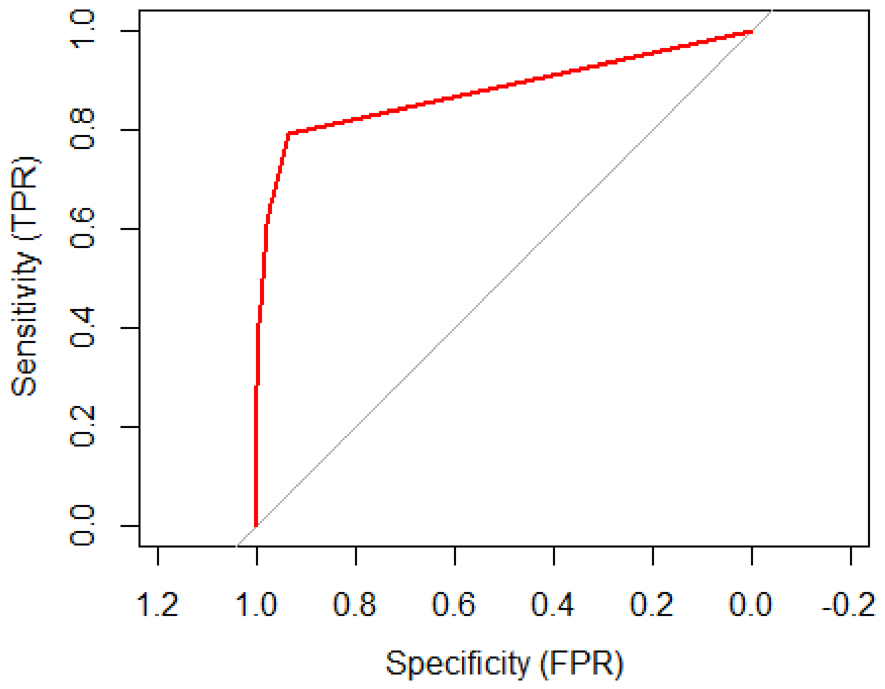
From the plot we can that first tree is split at payment plan days, if its less than 30 then it further splits at is_cancel. It is split at 30 days because most subscriptions are only for a period of 30 days and it further investigates for that period. Second split is at is_cancel which shows that people canceled the subscription or not. Because people can become active after cancellation, it doesn't mean that people churn if the cancel the subscription. Then it splits on actual amount paid, payment method id and is auto renewal. Auto renewal is also important that if your subscription time ends then do you allow to auto renew and deduct the payment automatically from the card saved into your account or you don't want to allow.

```
# Confustion matrix for training data
p3 <- predict(tree_model, train, type = "response")
tab3 <- table(predicted = p3, Actual = train$is_churn)
tab3

##          Actual
## predicted      0      1
##         0 707463  31072
##         1   5765  48326

# Miss classification error for training data
(1-sum(diag(tab3))/sum(tab3)) * 100

## [1] 4.647463

# Accuracy
(sum(diag(tab3))/sum(tab3)) * 100

## [1] 95.35254

# Confusion matrix - test
p2 <- predict(tree_model, test, type = 'response')
```

```
# Confusion matrix - test data
(tab2 <- table(predicted = p2, Actual = test$is_churn))

##          Actual
## predicted      0      1
##         0 302749  13777
##         1   2486  20398

# Miss classification error
(1 - sum(diag(tab2))/sum(tab2)) * 100

## [1] 4.79155

#Accuracy
(sum(diag(tab2))/sum(tab2)) * 100

## [1] 95.20845
```

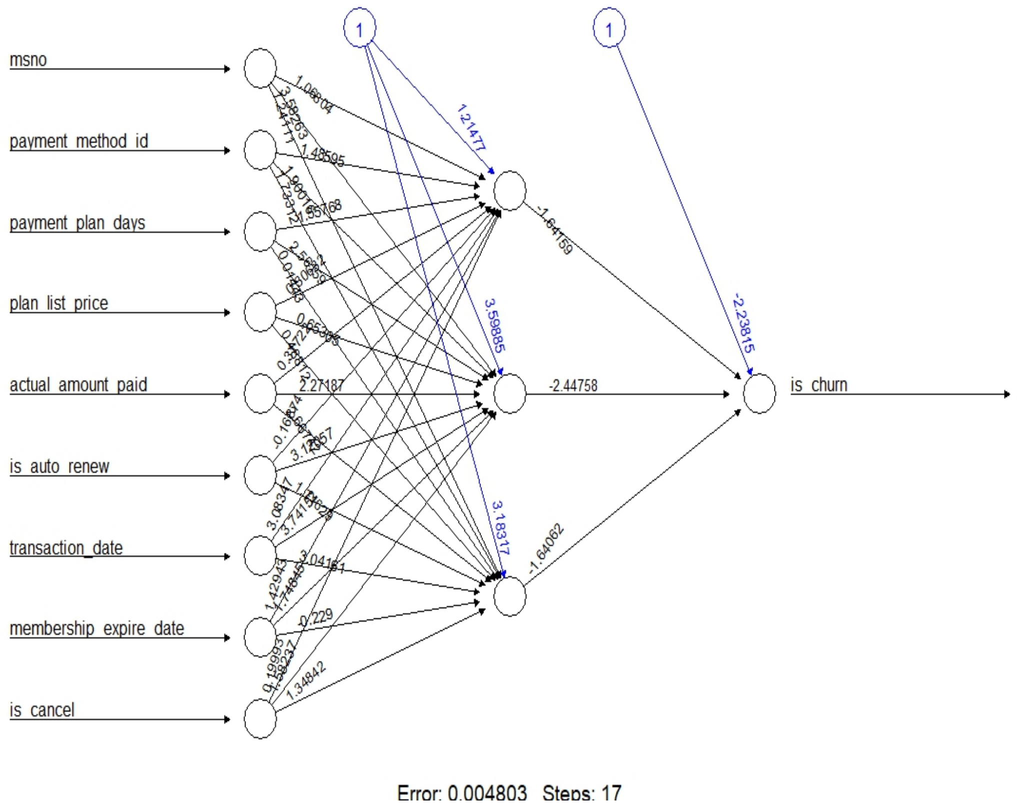
Using decision tree we got around 95% accuracy for training and testing both, which is higher than logistic regression where we got around 92%. Also if we see confusion matrix then prediction for true positive has also increased as compared to prediction of true positive value in logistic regression. We also plotted ROC plot for decision tree and found out that it gives 87.73% AUC which is higher than logistic regression. It gives higher AUC because sensitivity predicted in this algorithm is higher than logistic regression model. ROC curve is given below.

```
prob1 <- predict(tree_model1, newdata=test, type="prob")
auc1 <- auc(test$is_churn, prob1[,2])
auc1

## Area under the curve: 0.8773

plot(roc(test$is_churn, prob1[,2]), ylab='Sensitivity (TPR)', xlab='Specifici
ty (FPR)', col="red", xlim=c(1,0))
```



Neural Network

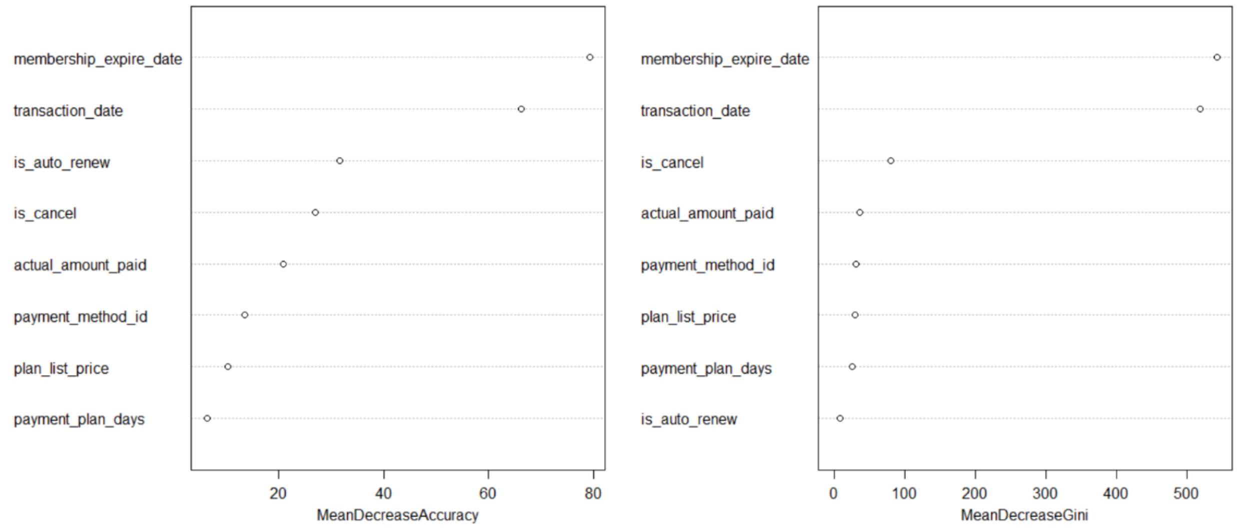
Then we applied neural network in order to see is there any difference in the results. Graph for neural network is given below. Also if we see below for the training and testing confusion matrix, it shows that neural network is only predicting for one class that is 0, it is not predicting for other class. Even the accuracy is around 90% but it does not give prediction for other class that is 1.



```
# Prediction train
pred <- compute(nmodel,train1[,-1])
#  Confusion matrix - train data
p5 <- pred$net.result
p5 <- ifelse(p5>0.5,1,0)
tab5 <- table(predicted=p5, Actual = train1$is_churn)
tab5

##          Actual
## predicted      0 0.0000000494295410808174
##         0 713228                      79398

# Miss clacification Error - train data
(1- sum(diag(tab5)/sum(tab5))) * 100

## [1] 10.01708246

# Accuracy - train data
(sum(diag(tab5)/sum(tab5))) * 100

## [1] 89.98291754
```

```
# Prediction test
pred <- compute(nmodel,test1[,-1])
# Confusion matrix - test data
p6 <- pred$net.result
p6 <- ifelse(p6>0.5,1,0)
tab6 <- table(predicted=p6, Actual = test1$is_churn)
tab6

##           Actual
## predicted      0 0.000000494295410808174
##          0 305235                    34175

# Miss clacification Error - train data
(1- sum(diag(tab6)/sum(tab6))) * 100

## [1] 10.06894317

# Accuracy - train data
(sum(diag(tab6)/sum(tab6))) * 100
```

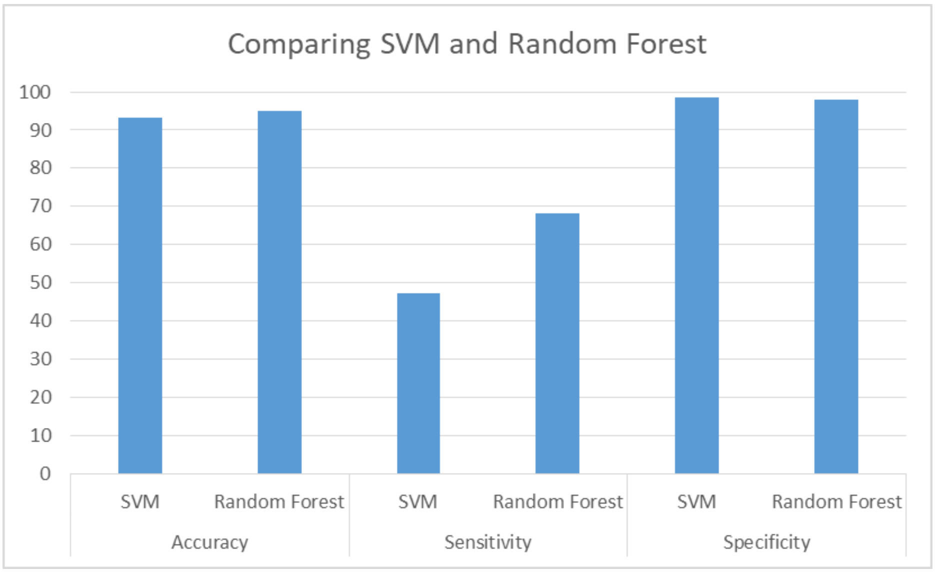
Random Forest



We calculate the variable importance using random forest and find out membership expiration date and transaction date are very important in predicting churn. Along with this as we see in above graph auto renew and is cancel attributes also make their effects on the churn class.

SVM and Random Forest

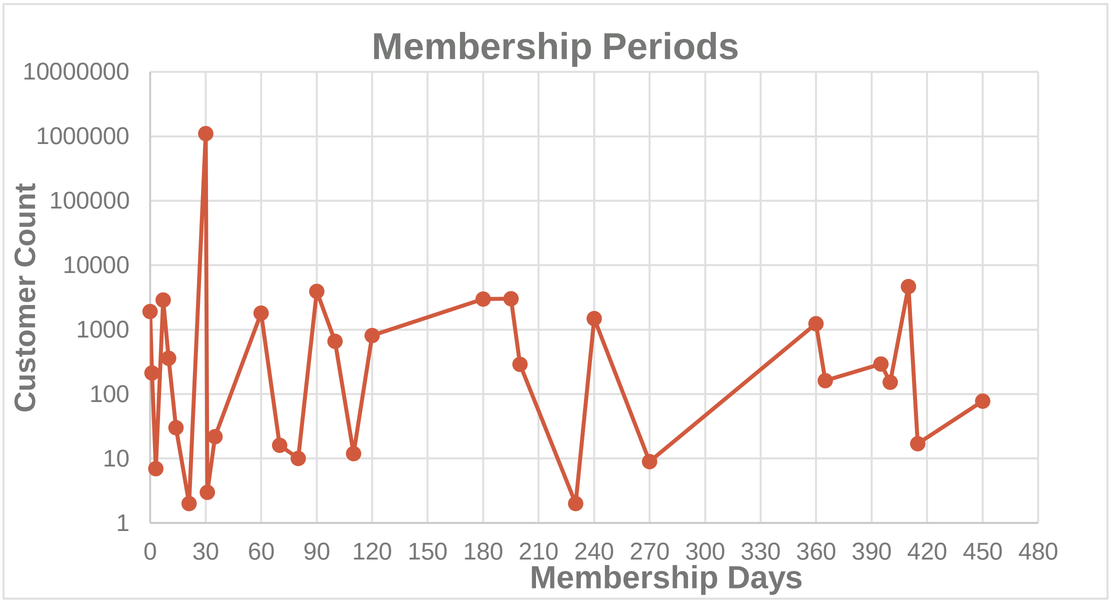
Then we applied SVM and Random Forest on KKbox dataset. SVM and Random Forest were only applied to 10% of data because for whole data it was taking too much time to compute and giving memory error. When we checked the performance, random forest gave 94.96% accuracy while SVM gave a little bit less which was 93.33%. We computed accuracy, sensitivity and specificity for both the models and presented in the below graph.



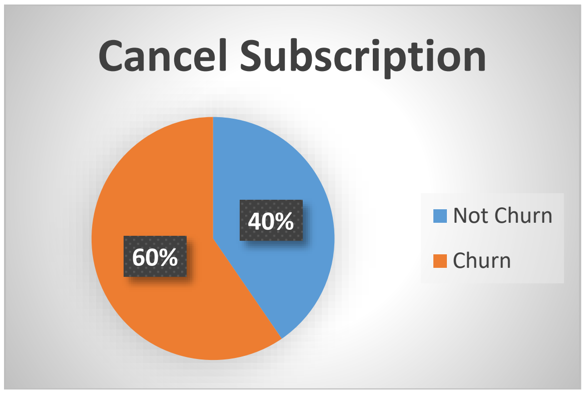
As we can see from the above graph accuracy and specificity for both the models remained around similar while sensitivity is different. Random Forest seems to perform well in case of sensitivity as compared to SVM which gives around 68% sensitivity while SVM gave only 48% sensitivity which is equal to random guess because its less than 50%.

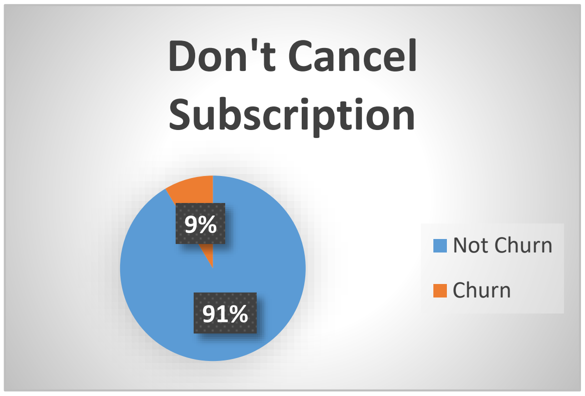
Visualizations

We also visualized certain attributes of data to find out any trend, pattern or any other relation between independent variables and the response variable. The visualizations are given below,



First we wanted to see, what is the relationship between number of customer and period of their membership in days. This is a one month data where people have subscribed from 1 day to 450 days, like at the beginning of the subscriptions they have specified how long they want to stay with the company and they may extend afterwards. So, the results which we found from above graph was maximum number of customers have subscribed for only 30 days which was 1104946, it may be due to some marketing offer or first 30 days free subscription. For the other days we can see values have been up and down while there is sharp decline at 230 days subscription. Only few people have subscribed for the 450 days which is 78.





We visualized another aspect of dataset which gives relation between people cancelling or not cancelling the subscription and if it effects churn.

The first pie chart shows that out of the subscribers who cancel their subscription, 60% of them also churn and only 40% don't churn after cancellation which shows a strong relation which is people who will cancel there subscription have high chances of churn. The second pie chart shows what percent of people churn or don't churn when the subscriber doesn't cancel. It is seen that 91% of the subscribers who don't cancel do not churn and only 9% of subscribers churn without canceling their subscription.

It can be concluded that people who cancel the subscription have higher probability of leaving the company than people who don't cancel their subscription.

8. Conclusions

In this project, our goal was to predict churn, we made use of various machine learning models and after trying all the relevant models, the conclusion we came to is, the highest accuracy was obtained by decision trees, the highest sensitivity was obtained by random forest and the highest specificity was obtained by decision trees. We also created visualizations which revealed some interesting findings such as frequency of membership period and relation between is_cancel and is_churn.

9. References

Dataset: https://www.kaggle.com/c/kkbox-churn-prediction-challenge#description
[1] http://ieeexplore.ieee.org/document/5392560/
[2] https://chatbotsmagazine.com/how-machine-learning-algorithms-can-help-your-business-beeb1e6a221
[3] Machine Learning: A Probabilistic Approach by Kevin P. Murphy
http://dsd.future-lab.cn/members/2015nlp/Machine_Learning.pdf
[4] http://www.saedsayad.com/logistic_regression.htm
[5] https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html
[6] https://towardsdatascience.com/what-is-a-decision-tree-22975f00f3e1

Appendices

This section contains the code of our project which we used to analyze the dataset.

Logistic Regression

```
## Loading train data which contains msno and is_churn
train_v2_data = read.csv(file.choose(), header = T)
## loading transaction data which contains payment method, plan, etc.
transaction_v2_data = read.csv(file.choose(), header = T)
## merging both data inorder to have is_churn column with transaction data
train_transaction_v2 <- merge(train_v2_data, transaction_v2_data, by="msno")
# Factor variables
train_transaction_v2$is_churn <- as.factor(train_transaction_v2$is_churn)
# Partition data - train 70%, test 30%
set.seed(511)
ind <-sample(2,nrow(train_transaction_v2), replace = T, prob = c(0.7, 0.3))
train <- train_transaction_v2[ind ==1,]
test <- train_transaction_v2[ind ==2,]
#library(caret)
lrmodel <- glm(is_churn ~ + payment_method_id + payment_plan_days +
                plan_list_price + actual_amount_paid + is_auto_renew +
                transaction_date + membership_expire_date + is_cancel,
               data = train, family = "binomial")
summary(lrmodel)
# Confustion matrix for training data
p3 <- predict(lrmodel, train, type = "response")
p3 <- ifelse(p3>0.5,1,0)
tab3 <- table(predicted = p3, Actual = train$is_churn)
tab3
# Miss classification error for training data
(1-sum(diag(tab3))/sum(tab3)) * 100
# Accuracy
(sum(diag(tab3))/sum(tab3)) * 100
# Confustion matrix for test data
p4 <- predict(lrmodel, test, type = "response")
p4 <- ifelse(p4>0.5,1,0)
tab4 <- table(predicted = p4, Actual = test$is_churn)
tab4
# Miss classification error for test data
(1-sum(diag(tab4))/sum(tab4)) * 100
# Accuracy
(sum(diag(tab4))/sum(tab4)) * 100
```

Logistic Regression ROC Curve

```
#################### ROC curve for logistic regression #####################

prob <- predict(lrmodel, newdata=test, type="response")
pred <- prediction(prob, test$is_churn)
```

```
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, ylab='Sensitivity (TPR)', xlab='Specificity (FPR)', col="red")
auc <- performance(perf, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

Decision Tree

```
tree_model <- tree(is_churn ~ + payment_method_id + payment_plan_days +
                      plan_list_price + actual_amount_paid + is_auto_renew+
                      transaction_date + membership_expire_date + is_cance,
                    data = train, method = 'class')
plot(tree_model)
text(tree_model, pretty = 0)
# Confustion matrix for training data
p3 <- predict(tree_model, train, type = "response")
tab3 <- table(predicted = p3, Actual = train$is_churn)
tab3
# Miss classification error for training data
(1-sum(diag(tab3))/sum(tab3)) * 100
# Accuracy
(sum(diag(tab3))/sum(tab3)) * 100

# Confusion matrix - test
p2 <- predict(tree_model, test, type = 'response')
# Confusion matrix - test data
(tab2 <- table(predicted = p2, Actual = test$is_churn))
# Miss classification error
(1 - sum(diag(tab2))/sum(tab2)) * 100
#Accuracy
(sum(diag(tab2))/sum(tab2)) * 100
```

Decision Tree ROC Curve

```
library(pROC)
prob1 <- predict(tree_model1, newdata=test, type="prob")
auc1 <- auc(test$is_churn, prob1[,2])
auc1
plot(roc(test$is_churn, prob1[,2]), ylab='Sensitivity (TPR)', xlab='Specificity (FPR)',
col="red", xlim=c(1,0))
```

Neural Network Code

```
## Loading train data which contains msno and is_churn
train_v2_data = read.csv(file.choose(), header = T)
## loading transaction data which contains payment method, plan, etc.
transaction_v2_data = read.csv(file.choose(), header = T)
## merging both data inorder to have is_churn column with transaction data
train_transaction_v2 <- merge(train_v2_data, transaction_v2_data, by="msno")
data_sample_v2 <- train_transaction_v2
data_sample_v2$is_churn <- as.numeric(data_sample_v2$is_churn)
data_sample_v2$msno <- as.numeric(data_sample_v2$msno)
```

```
## Neural Network
#Normalize
data <- (data_sample_v2 - min(data_sample_v2,
na.rm=TRUE))/(max(data_sample_v2,na.rm=TRUE) -
                              min(data_sample_v2, na.rm=TRUE))
# Partition
set.seed(511)
ind <- sample(2, nrow(data), replace =T, prob = c(0.7,0.3))
train1 <-data[ind==1,]
test1 <- data[ind ==2,]
# Neural network model
library(neuralnet)
n <- names(train1)
f <- as.formula(paste("is_churn ~", paste(n[!n %in% "is_churn"], collapse = " + ")))
#nmodel <- neuralnet(f, data = train, hidden = , stepmax=1e6)
nmodel <- neuralnet(f, data=train1, hidden=3,
              linear.output = FALSE)
#nmodel
plot(nmodel)
p <- nmodel$net.result[[1]]

# Prediction train
pred <- compute(nmodel,train1[,-1])
#  Confusion matrix - train data
p5 <- pred$net.result
p5 <- ifelse(p5>0.5,1,0)
tab5 <- table(predicted=p5, Actual = train1$is_churn)
tab5
# Miss clacification Error - train data
(1- sum(diag(tab5)/sum(tab5))) * 100
# Accuracy - train data
(sum(diag(tab5)/sum(tab5))) * 100
# Prediction test
pred <- compute(nmodel,test1[,-1])
# Confusion matrix - test data
p6 <- pred$net.result
p6 <- ifelse(p6>0.5,1,0)
tab6 <- table(predicted=p6, Actual = test1$is_churn)
tab6
# Miss clacification Error - train data
(1- sum(diag(tab6)/sum(tab6))) * 100
# Accuracy - train data
(sum(diag(tab6)/sum(tab6))) * 100
```

SVM

```
## SVM
## Loading train data which contains msno and is_churn
```

```
train_v2_data = read.csv(file.choose(), header = T)
## loading transaction data which contains payment method, plan, etc.
transaction_v2_data = read.csv(file.choose(), header = T)
## merging both data inorder to have is_churn column with transaction data
train_transaction_v2 <- merge(train_v2_data, transaction_v2_data, by="msno")
## Taking out sample 10%
set.seed(511)
ind <- sample(2, nrow(train_transaction_v2), replace = T, prob = c(0.1, 0.9))
#ind
data_sample <- train_transaction_v2[ind == 1,]
#data_sample <-merged_v2_data
#dividing the sample further
ind2 <- sample(2, nrow(data_sample), replace = T, prob = c(0.7, 0.3))
train <- data_sample[ind2==1,]
test <- data_sample[ind2==2,]

library(e1071)
model_svm <- svm(is_churn ~ + payment_method_id + payment_plan_days +
            plan_list_price + actual_amount_paid + is_auto_renew +
            transaction_date + membership_expire_date + is_cancel , train)
summary(model_svm)

# Prediction
p1 <- predict(model_svm, train)
# Confusion matrix - training data
tab1 <- table(predicted = p1, Actual = train$is_churn)
tab1
# Miss classification error
(1 - sum(diag(tab1))/sum(tab1))
# Accuracy
(sum(diag(tab1))/sum(tab1))

# Confusion matrix - test
p2 <- predict(model_svm, test)
# Confusion matrix - test data
(tab2 <- table(predicted = p2, Actual = test$is_churn))
# Miss classification error
(1 - sum(diag(tab2))/sum(tab2))
#Accuracy
(sum(diag(tab2))/sum(tab2))
```

Random Forest

```
data <- sample_data
# Factor variables
data$is_churn <- as.factor(data$is_churn)
# Partition data
set.seed(511)
ind <- sample(2, nrow(data), replace = T, prob = c(0.7, 0.3))
```

```
train <- data[ind==1,]
test <- data[ind==2,]

# Random forest model
library(randomForest)
system.time(rf <- randomForest(is_churn ~ + payment_method_id + payment_plan_days +
                               plan_list_price + actual_amount_paid + is_auto_renew +
                               transaction_date + membership_expire_date + is_cancel, train,
                                importance = T,
                                ntree = 100,
                                mtry = 8))
rf

# confusion matrix - train data
#install.packages("caret")
library(caret)
p1 <- predict(rf, train)
confusionMatrix(p1, train$is_churn)

# confusion matrix - test
p2 <- predict(rf, test)
confusionMatrix(p2, test$is_churn)
# Variable importance
varImpPlot(rf)
plot(rf)
```