PREDICTING CUSTOMER SATISFACTION USING RANDOM FOREST & GRADIENT BOOSTING

TEAM codeR

TANMAY KHARE

SHUBHAM PACHORI

SWATI CHELAWAT

DEEPENDRA SINGH

CONTENTS

INTRODUCTION - HOW CUSTOMER SATISFACTION IS IMPORTANT TO BANKS

In this competitive atmosphere, it becomes highly imperative for banks to understand the factors which might affect customer satisfaction. The knowledge of customer satisfaction among banks is the key to establish and maintain long term relationship with their customers. Thus, it becomes essential for banks to know about their customer's satisfaction well in advance so that they can take immediate steps to retain them with the bank in future.

According to the American Customer Satisfaction Index, there are certain critical elements that helps them to identify the level of satisfaction among bank customers. For example, the number of ATMs and their location, Website, Staff Courtesy, Branch Locations, Interest rate competitiveness etc.

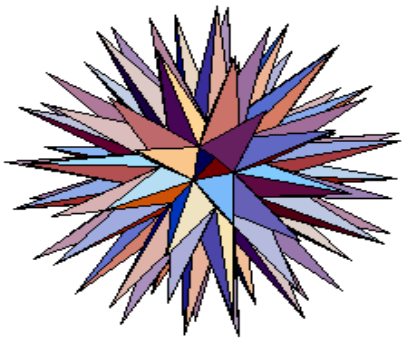
Realizing the value of customer satisfaction, Santander Bank wanted to predict whether their customers are satisfied with their services or not. They could then focus their attention to these customers and take necessary measures to make them happy.

Santander Bank released their dataset on Kaggle and asked the data science community to help them identify dissatisfied customers early in their relationship. This is therefore a classification predictive modeling problem where the target variable is categorical in nature i.e. takes values zero or one, where zero means the customer is satisfied and one means unsatisfied.

SUMMARY

Predicting Customer Satisfaction by banks early in their relationship with customers, helps them to direct their focus on proving better services to such customers.

CURSE OF DIMENSIONALITY



The dataset provided by Santander Bank is divided into training and test and contains 370 numerical predictors. All these features are anonymous in nature and therefore cannot be used for initial insights and visualization.

Therefore, this is a typical case of a high dimensional problem also popularly known as Curse of Dimensionality Problem. The main problem here is, that as the dimensionality of the data increases, volume of the space increases so rapidly that the available data becomes sparse. Also, to support the statistical models, the amount of data increases often grows exponentially with the dimensionality.

In order to deal with this high dimensional problem of our case, the most common approach used is to apply Principal Component Analysis. PCA is an algorithm that helps to reduce the total number of dimension by first checking the correlation between the dimensions and then reducing the dimensions in such a way that a certain correlation between them is maintained. More the dimensions are reduced less the correlation between them would be captured.

Another approach is to use the feature importance method provided by the Random Forest Algorithm. Random Forest Algorithm not only helps to make predictions but also assess variable importance. The R package to use Random Forest Algorithm is **randomForest** which has an *importance* parameter that helps us to select features based on their importance in the algorithm. We have used this approach in our project.

STEPS TAKEN TO REDUCE THE DIMENSIONALITY

REMOVING ZERO AND NEAR ZERO VARIANCE PREDICTORS:

In many situations, the data generation mechanism can generate some features that have a single unique value also known as zero variance predictor. This may sometimes cause the model to crash or end up giving fit to be unstable.

Therefore, we have outlined the below function in R to detect features with zero variance.

```
near_zero_variance_predictor = function(train)
{
  i=1
  features = c()
  for (i in 1:ncol(train))
  {
   if (var(train[,i,])==0)
   {
     features = c(features,names(train[i]))
   }
  i = i+1
  }
  return(features)
}
```

The above function uses the variable I to traverse through all the columns of the data and check if the variance of the column is zero. The columns for which the variance is equal to zero has been stored in the *features* variable and returned. Thus these set of features can be excluded from further analysis.

ANALYZING THE CORRELATION AMONG THE FEATURES

Reducing the number of highly correlated dimensions is a common approach used by Principal Component Analysis. Therefore, we wanted to check the correlation among the features of our dataset. We have used the below function to determine the correlation among variables. After removing the correlated variable, we applied the new dataset to the Random Forest Algorithm and calculated the accuracy of the model. We received a high accuracy by using the feature importance method of RandomForest package which we have explained later.

```
correlated_columns = function(train)
{
  f = c()
for (i in 1:ncol(train))
{
  if (i==370)
    break;
  for (j in i+1:ncol(train))
  {
    if(cor(train[names(train[i])],train[names(train[j])])==1)
    {
      ##f = c(f,names(train[i]),names(train[j]))
      f = rbind(f,names(train[i]),names(train[j]))
    }
    j=j+1
  }
  i=i+1
}
  return(f)
}
```

The above function checks if a column X and Y have equal correlation between them. If the correlation is equal then one of the columns among them is selected and used in the dataset.

USING randomForest AND gbm PACKAGE TO DETERMINE FEATURE IMPORTANCE

It is important to understand which algorithm to choose between Random Forest and Gradient Boosting for making predictions and selecting important features. Both the algorithms have their own pros and cons but some important points are worth keeping in mind before using them.

It is very important to understand how we tune the parameters for both the packages. Random Forest basically has only one parameter to tune which is the number of variables to randomly select at each node. On the other hand, Gradient Boosting have several hyper parameters that include the number of trees, the depth (or number of leaves), and the shrinkage (learning rate).

After using both the models, we got better ROC value from Gradient Boosting. Therefore, Gradient Boosting is usually preferred than Random Forest if we carefully tune the parameters. Please refer the code comment section to see how the parameters are tuned.

The below code shows how we use gbm.fit function from gbm package.

```
model = gbm.fit(x = as.matrix(train), ##Feature matrix
                y = y, ##Dependent variable TARGET
                distribution = "bernoulli", ##Bernouli for binary outcome
                n.trees = ntrees, ##A high value just for intuition purpose
                shrinkage = 0.001, ##Lower the shrinkage better chance of fit,
                                   ##but lowering it too low can be computationally expansive
                interaction.depth = 3, ##We'll use cross validation later to tune this hyper parameter
                n.minobsinnode = 10, ##Lower the value better for in-sample fit but result in overfitti
                nTrain = round(nrow(train)*0.8), ## USe this to select better guess for no of trees in
                #var.monotone = c() ##handy to smooth out of noisy curve
                verbose = TRUE ##Print the output
                )
```

Below snapshot shows how we fit Random Forests.

```
fit = randomForest(as.factor(TARGET)~.,data = final_train_data_set,ntree=10
probs = predict(fit,test,type='prob')
```

CONCLUSION

This project helped us to understand how to deal with high dimensional datasets such as provided by Santander Bank. This was a typical case where the features of the data were anonymous in nature and therefore it becomes nearly impossible to do analysis based on the nature of the feature. Hence, it becomes imperative to use advanced feature engineering techniques to come up with a more robust dataset to make better predictions. We also learned how to tune more complex algorithms such as Random Forest and Gradient Boosting in R. These algorithms not only gives us better accuracy with the predictions than other classification techniques but also helps us to do feature engineering which becomes very critical in model learning.

REFERENCES

https://www.kaggle.com

https://www.quora.com

http://www.theacsi.org

https://www.wikipedia.org

https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

https://cran.r-project.org/web/packages/gbm/gbm.pdf