Case Study -1

Application of Ensemble Techniques in Campaigning & Target Listing for Banking Products

by

Pradeep Sathyamurthy

Under the guidance of: Prof J. Gemmell

DePaul University

6th October 2017

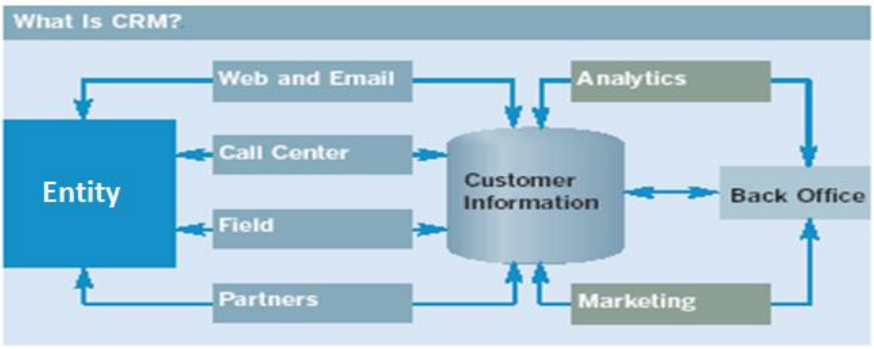
Table of Contents

Abstracts

In this case study, we will research on the benefit of applying machine learning application in Financial domain. As part of this case study we will classify if a user will buy a term deposit product of a Bank or not by deploying simple and advance machine learning techniques like Decision tree and ensemble techniques respectively. Main issue which we generally face is the asymmetric nature in target variable which we have tried to handle using sampling techniques like SMOTE and ROSE on training data and then apply classifiers on it to handle the classification with no bias or noise in data. We measure the performance of each sampling technique and classification techniques to choose the best sampling technique and best model for our classification problem. We will then conclude looking at the dominating variables in dataset based on which we can predict the buying nature of the customer and how it helps bank in return in terms of profit margin.

Introduction

Banking is one of the prominent industry which used statistical analysis in early years and scaled up to a great level in using machine learning at present to stay competitive in market. Every bank around the world now is operating under a core banking technology where the different banks are interlinked with each other. Application of machine learning in such domain is are in different forms like fraud detection, credit risk rating, etc., One of the main application in core banking technology is Customer Relationship Management (CRM) platform which deals mainly with customers maintenance and needs.



CRM is mainly composed of customer creation and maintenance. However, it other important application includes Marketing and Campaigning. Unless a bank understands its customer well any kind of marketing or campaigning will not yield any profit. As part of this case study we are going to see how a simple and advance machine learning techniques help bank in identifying right customer for Marketing and campaigning.

Process Followed in this case study:

- EDA is done in Python for data analysis and visualization
- Feature engineering was done in Python to normalize the value and make the units of variable common
- Data split was done in R to split the training and testing data with 80-20 split ration of total sample size
- Target data imbalance was handled in R using under, over, SMOTE and ROSE sampling technique
- Model building and evaluation was done in R for decision tree, KNN, Naïve Bayes, Logistic regression, Random forest and Logit Boosting.

Data Description

Dataset is obtained from UCI data repository and can be accessed with internet by clicking here. Dataset consist of 45211 instances with 16 features being an independent variable and 1 binary feature being the target variable which would be subjected for classification.

Numerical Independent Variables:

1. Age: age of the user
2. Balance: average yearly balance in euros
3. Day: last contact day of the month
4. Duration: last contact duration, in seconds
5. Campaign: number of contacts performed during this campaign and for this client
6. Pdays: number of days that passed by after the client was last contacted from a previous campaign
7. Previous: number of contacts performed before this campaign and for this client

Categorical Independent Variables and its unique values:

8. Job: type of user job `{'admin.', 'retired', 'technician', 'student', 'services', 'housemaid', 'self-employed', 'management', 'unknown', 'unemployed', 'entrepreneur', 'blue-collar'}`
9. Marital: marital status `{'divorced', 'married', 'single'}`
10. Education: highest education level of user `{'secondary', 'primary', 'unknown', 'tertiary'}`
11. Default: Does user has credit in default? `{'yes', 'no'}`
12. Housing: Does user has housing loan? `{'yes', 'no'}`
13. Loan: Does user has personal loan? `{'yes', 'no'}`
14. Contact: Users preferred communication type `{'cellular', 'unknown', 'telephone'}`
15. Month: last contact month of year `{'feb', 'sep', 'may', 'jan', 'apr', 'mar', 'jul', 'dec', 'aug', 'nov', 'jun', 'oct'}`
16. Poutcome: outcome of the previous marketing campaign `{'yes', 'no'}`

Response Variable to be predicted:

17. y: has the client subscribed a term deposit? `{'yes', 'no'}`

We can even consider features Day as categorical in nature, but this will increase the feature dimension and might make the model complex and even overfit at times. Hence, for this case study I would like to maintain this as a numerical variable only. However, I will standardize these numerical variables based on their nature of distribution so that each numerical variable has same weightage and independent of Units.

Exploratory Data Analysis (EDA):

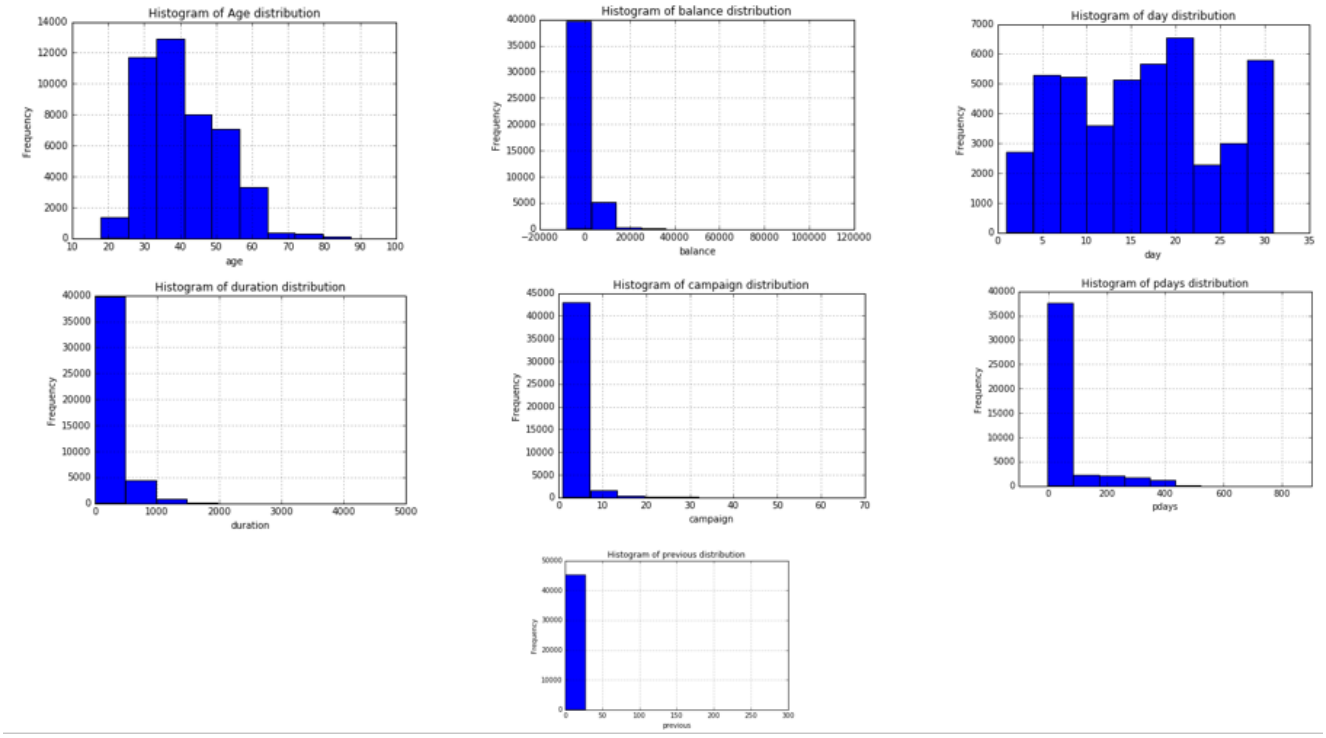
Even before we try to clean a dataset, it is better to visualize the nature of current dataset and then deal with cleaning the dataset if required with an optimum method.

Exploring the numerical attributes:

As explained in dataset description 'age','balance','day','duration','campaign','pdays','previous' act as numerical data in the dataset to classify if a customer buys term deposit or not. Summary statistics of these features are listed below:

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

It would be better to visualize all these numerical variables and explore them further:



Age:

Bank generally target their audience based on the age to sell few banking products like pension account for aged people, housing loan for people who are mid-age, education loan for students who are young and in early 30's. Thus, age play an important role in selling a banking product. In our dataset we see majority of the user have an average age of 40. Since from the above graph we see a normal distribution for age, we can infer that 64% of the population in this dataset are between age 30 and 50 as the standard deviation is 10. Thus, we can hypothesize that users of mid age between 30 and 50 are more intended user to whom campaign can be done to sell the term deposit product and would become a potential buyer.

Balance:

Balance play an important role in cross selling a product, we see that dataset has users with average balance of euros 1362 and its distribution seems right skewed. Thus, median would be the best parameter to explain this variable. Thus, we see median salary for the users in this dataset is approximately equal to euros 500.

Day:

We see a bimodal distribution here and most of the campaigning had taken place in the mid of the month as 15 seems to be the average day when last the campaign was done for a user. Since, this has a bimodal distribution we can expect a little variance from this variable and hence can hypothesize this being not a significant variable to classify a user.

Duration:

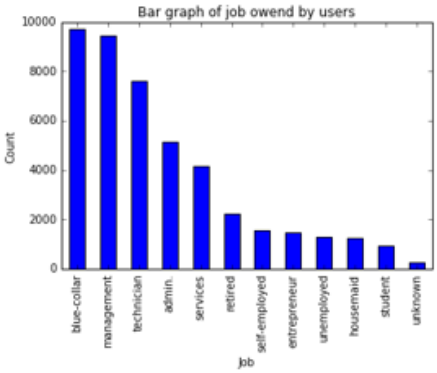
Duration is the time is seconds spend in campaigning a customer about the term deposit product for cross sell. We see a skewed distribution for this variable. Thus, considering median to explain this variable, we see most of the campaign is done for almost 180 seconds which is 3 minutes of duration. ***Intuitively, more time a user spends in listing about a product incline towards buying the same***. With this hypothesis, we can hypothesize that this particular variable will be a significant variable in classifying the users with respective to buying term deposit product.

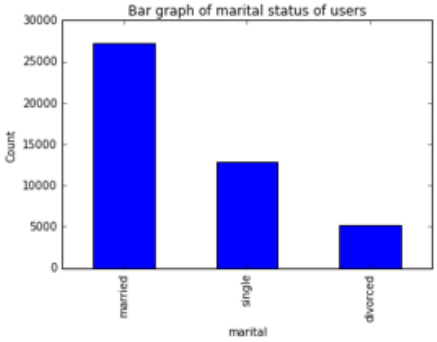
Campaign, Pdays and Previous:

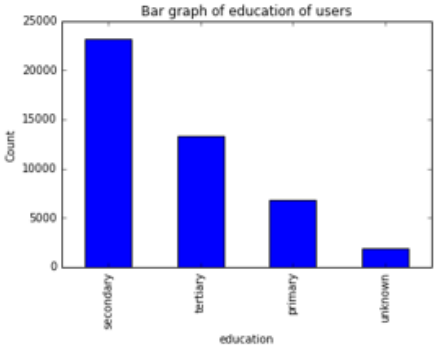
From the graph or from summary statistics I couldn't see any interesting hypothesis on these variables at this state of EDA. However, let's try to build model and then see if these variables of any use.

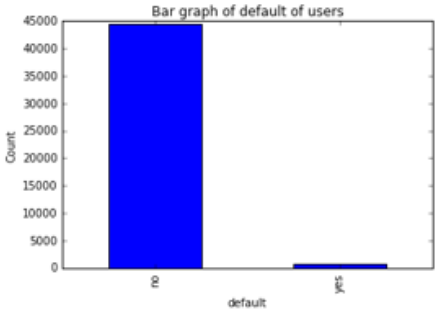
Exploring the categorical attributes

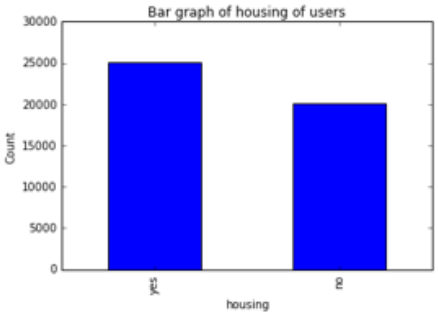
As explained in dataset description 'job','marital','education','default','housing','loan','contact','month','poutcome' act as numerical data in the dataset to classify if a customer buys term deposit or not. It would be better to visualize all these numerical variables and explore them further:

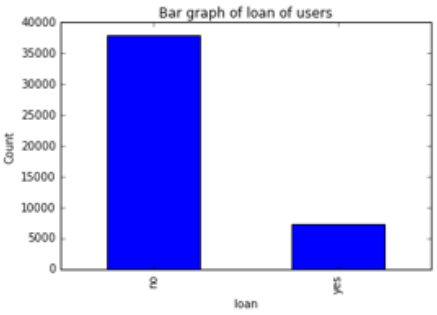


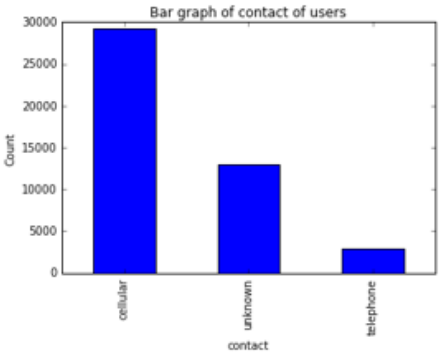


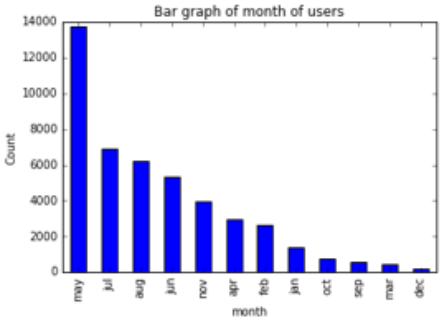


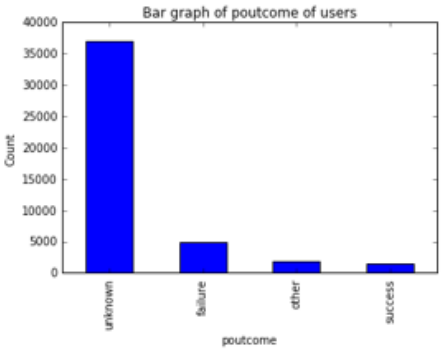










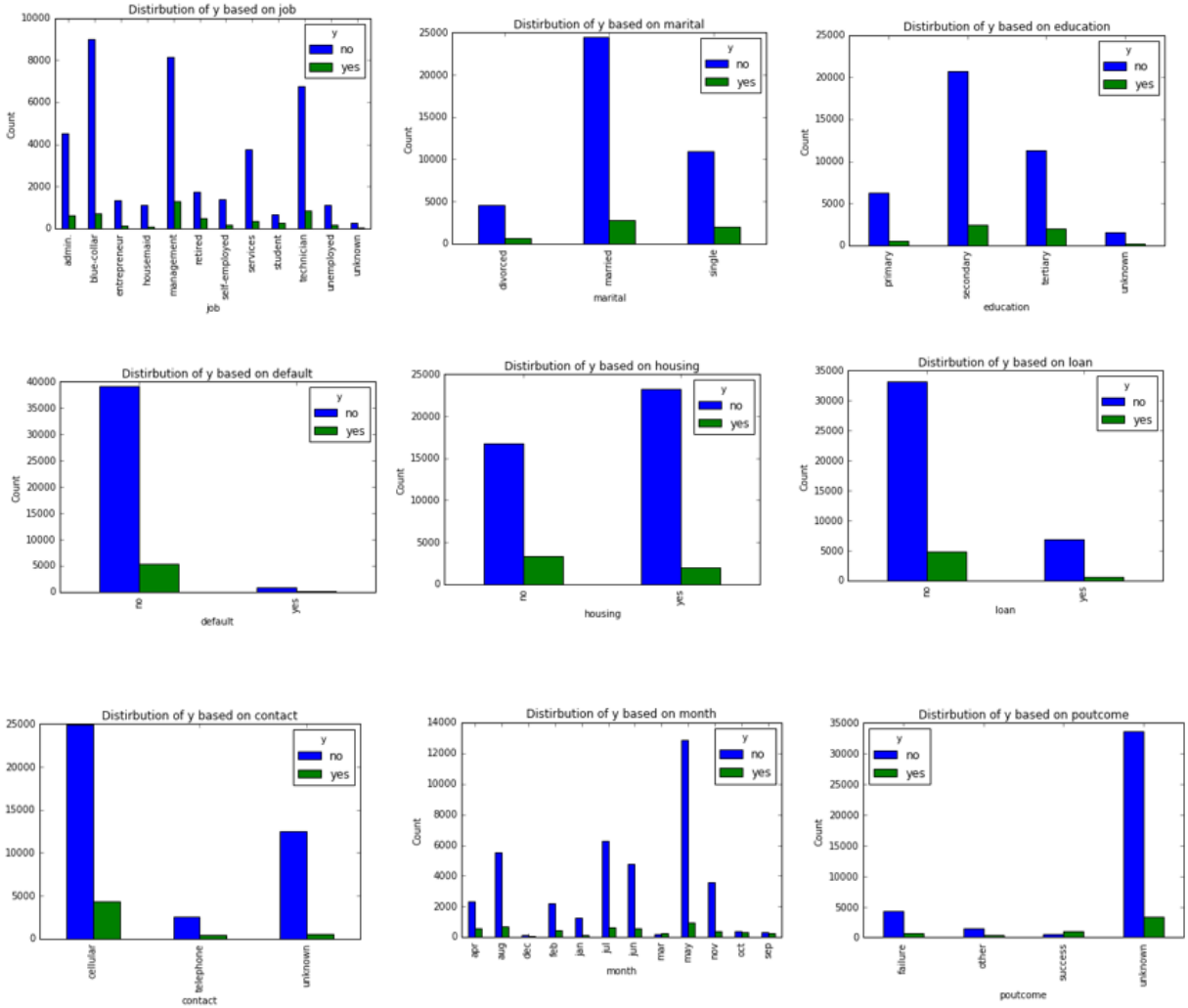


I do not want to go in deep explanation about each categorical variable individually, we will try to highlight very important variables which are interesting to explore further when comparing them with target variable and in model building stage. Bar chart being the best representation for categorical variables we see:

- The dataset is more dominated by users who are in blue-collar jobs and in other white color jobs like management technicians, admins and service.
- We see more users from mid age group who are married and with secondary level education influencing the dataset.
- We see there are majority of users who are not defaulters with no personal loans but have housing loan dominate the dataset allot.
- We also see the preferred communication channel for most of the user being cellular and most campaign happening on the mid of year prominently in the month of May and July.
- We have most of the users whose previous result on campaigning is unknown.

Exploring the impact of categorical feature on target attribute:

In dataset ‘y’ is the target variable which classifies if a user as bough the term deposit or not. As part of this exploratory data analysis let’s see how the target attribute is been affected by the categorical variables in dataset.



Looking at each categorical variable along with the distribution of target variable is more interesting when compare to studying the categorical variables individually. However, hypothesis made in above section can be confirmed visually here with this comparison, thus we observer:

Users with job management followed by technicians, blue-collar and admin are those who purchase term deposit product, from this we can infer that ***users with high job profile can be the main target for campaigning*** about the term deposit product.

We see ***users who are married tends to buy more term deposit product*** when compare to users who are single and divorced. This can be because users who are married tend to have tow source of income and thus more inclined towards buying the term deposit product.

We see ***users who tends to buy TD product have tertiary or at least secondary education***, this is more evident from above result that, high profile user’s general have more education compare to low profile workers.

We also see ***term deposit (TD) product is been bought more by users who have no loan default, with no housing and other personal loans***.

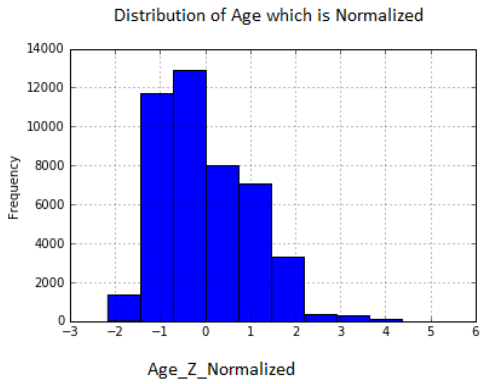
It seems more evident that ***user who have cellular as preferred communication medium generally buy bank product***, while folks who have no interest do not have cell as their preferred medium for communication and max selling happens only in the month of second quarter of the year which is between Apr, may and June.

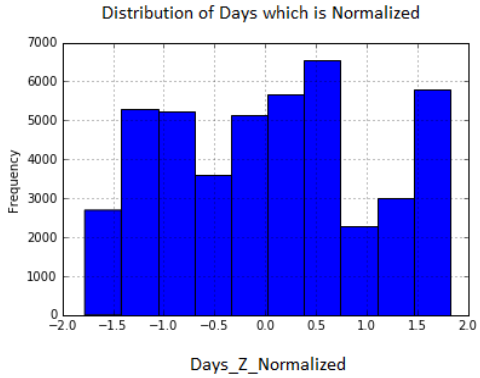
Feature Engineering:

Normalization:

From the exploratory data analysis, we could see there are 7 different numerical variables with different units. Different units being producing different weightage when we build model. Hence it is necessary we remove this influence of weights from each variable and make it unit independent dataset.

Statistical method to achieve this is to standardize the values, so we subject all 7 numerical variables to standardization and make them have equal weightage in the dataset.





We see ***features age and day to have normal and binormal distribution and hence we do z-distribution for these variables alone*** while other numerical variables explained above are subjected to min-max normalization. Main advantage of z-score normalization is that we do not disturb the normality of the data and have its mean to be 0 with standard deviation being 1 for such z-normalized variables. Same was done in python.

Dummy Variables for categorical variables:

Similarly, most of the model building algorithm expect data to be numerical in nature and hence the whole data set after being normalized is converted to be continuous values by creating dummy variables for categorical variables. Thus, ***using panda's python package all categorical variables where converted to dummy variables***.

Correlation analysis on numerical variables:

I performed a correlation analysis on the converted dataset in order to check if there is a existence of any collinearity between variables through which we can do some feature reduction and avoid issues of multicollinearity. I had considered any correlation above and below 0.6 and -0.6 being highly correlated.

	age	balance	day	duration	campaign	pdays	previous
age	True	False	False	False	False	False	False
balance	False	True	False	False	False	False	False
day	False	False	True	False	False	False	False
duration	False	False	False	True	False	False	False
campaign	False	False	False	False	True	False	False
pdays	False	False	False	False	False	True	False
previous	False	False	False	False	False	False	True

However, from the given dataset I did not observe any correlation between variables as we can see True being only in the diagonals of above matrix.

Data Split:

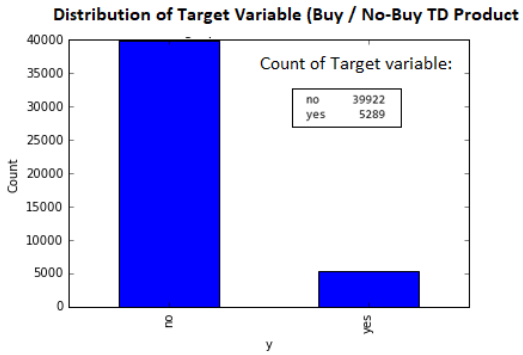
Though in the later model building state we will evaluate a model with n-fold cross validations, it is necessary to have a validation set which is not a part of main dataset to evaluate the model performance. Hence, 80-20% split on entire dataset was made using R with 36169 records in training and 9042 records in testing dataset.

Sampling Techniques to address imbalance in target variable

When we did EDA I was able to observe imbalance in the target variable 'y' which is the target variable to be classified for user who buy term deposit product (yes/no):

We see that the target variable 'y' is totally imbalance, with 39922 instances being 'No' to term deposit product and 5289 instances being 'yes' to term deposit product.

This kind of distribution in dataset would cause bias while building the model and hence need to be treated before we build any model above this given dataset.



Even in the training dataset we see 31972 instances with value 'no' for target variable and 4197 instances with value 'yes' users who will buy term deposit product. We see 88% of the training data set being 'no' and 11% with value 'yes'. This will highly influence the model if subjected to model building as it is and might lead to issue of overfitting the dataset, that is model constructed would work fine only for this training dataset with which model was built and could not be generalized for other data's for classification.

To avoid this, we need to perform some resampling of data and avoid this bias and imbalance in target variable. There are 4 major resampling techniques which can be used to treat this target variable, we will choose the best that works with this dataset for further model building:

1. Over Sampling:

This is a sampling method which is used to treat the biased target variable by exaggerating the minority classes and making them equal to majority class, that is exaggerating records with value 'Yes' that is '1' in dataset.

2. Under Sampling:

This is a sampling method which is used to treat the biased target variable by under playing the majority classes and making them equal to minority class, that is under playing the records with value 'No' that is '0' in dataset.

3. SMOTE Sampling (Both Over + Under Sampling):

This is sampling technique which will over play the minority classes and under play the majority classes in the dataset.

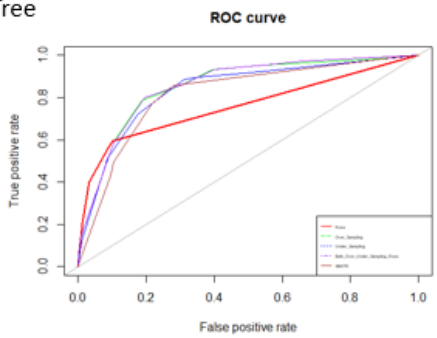
4. ROSS Sampling:

This is another sampling technique which is prominently used to treat such imbalanced target variables.

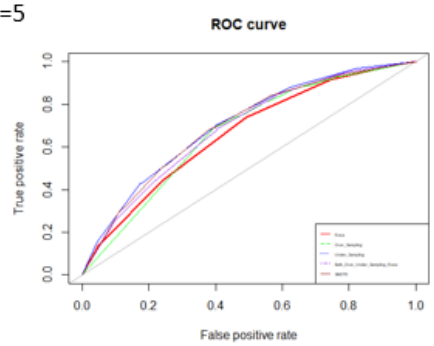
5. Choosing the best Sampling technique for dataset:

I employed all 4-sampling technique and created sampled balanced dataset for the target variable using the 'ROSE' and 'DMwR' package in R. once done, I had built a basic decision tree model and a KNN model with K=5 for all these sampled datasets and below are the results:

Decision Tree



KNN with K=5



Dataset Name	ROC - Decision Tree Algo	ROC - KNN with K=5
Original Imbalanced Dataset	0.833	NA
Over Sampling	0.855	0.667
Uder Sampling	0.839	0.704
Both Over & Under Sampling	*0.858*	*0.686*
ROSE Sampling	0.758	0.662
SMOTE Sampling	*0.817*	*0.695*

SMOTE is the structured implementation of over + under sampling provided by 'DMwR' R package. From the above result, we can say that SMOTE sampled dataset has a best mean ROC of 0.858.

Thus, out of 36169 records from the training dataset, using SMOTE we had built a balanced dataset with respective to the target variable with a record count of 12591 user records who don't purchase term deposit from bank i.e. y=0 and 12591 user records who purchase term deposit from bank i.e. y=1

Thus, with this SMOTE sampled dataset, we will apply below algorithm and evaluate its performance to see if the more advance technique like ensemble improve the accuracy and at the same time not overfitting the model. Below are few classification algorithms which we will use to classify the users who purchase term deposit product or not:

1. Decision Tree

2. Naïve Bayes

3. KNN with K=5

4. Logistic Regression with c=5

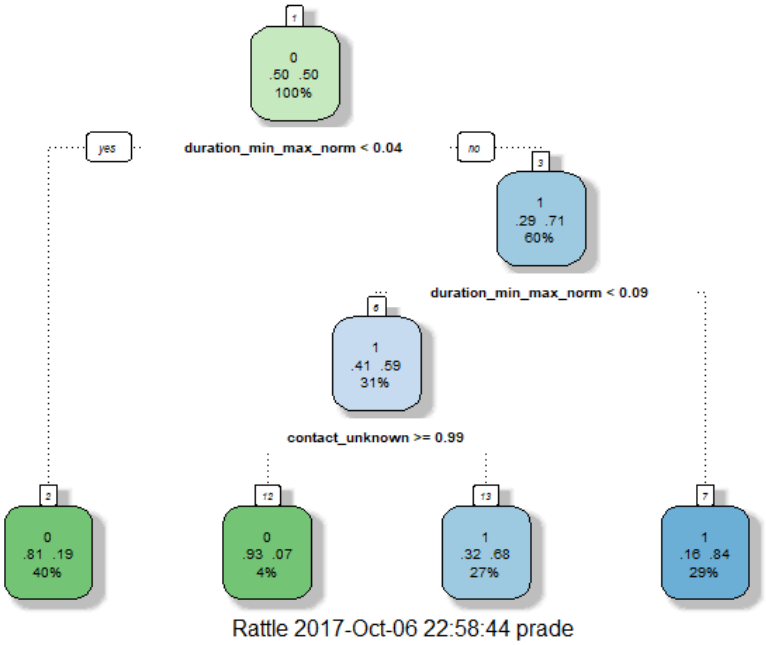
5. Bagging - Ensemble on Decision Tree – Random Forest

6. Boosting – Boosted Logistic Regression

Classification Algorithm and its comparison:

1. Decision Tree:

As hypothesised in our EDA, we see feature duration becomes the root node of the simple decision tree. By reconverting the normalized duration back to its original value, we can infer that efficient call time for campaigning is 473 seconds which is approximately 8 minutes. That is, ***if a call lasts longer than 8 minutes, the chance of buying term deposit product is 84%. Saying that, if the call duration is between 205 seconds and 473 seconds, and the contact method is unknown, then the chance of buying term deposit product is just 68%***. Same proves right even from the equation of logistic equation where duration is the highest influential variable in dataset.



2. Naïve Bayes

This algorithm works on probability measure. By providing the SMOTE sampled dataset to this Naïve Bayes (NB) algorithm below confusion matrix was obtained.

NB	Reference	
Predicted	0	1
0	4219	92
1	3731	1000

Thus, from above confusion matrix we can infer that NB model as an accuracy of 57.7% which means only 57% of the total test cases where classified correctly by NB algorithm which is a poor classification result for this dataset. However, if we look at the specificity measure, NB model produce 91.58% of specificity. Thus, ***to classify customer who will buy term deposit product, we can use NB model for our classification.***

3. KNN with K=5

KNN algorithm works based on the close match of the nearest instances whose similarity is measured by the Euclidean, Manhattan or cosine similarity. Our algorithm in R uses the Euclidean for measuring the similarity. I have used K=5 which means that our KNN model look for 5 nearest neighbour from SMOTE sampled dataset for a given test set and classify if a user will purchase term deposit product or not. Confusion matrix for the same is shown below:

KNN	Reference	
Predicted	0	1
0	5484	433
1	2466	659

From above confusion matrix, we see that neither the accuracy (0.6794) nor the sensitivity (0.6898) or specificity (0.6035) values are too high. So, I infer that ***KNN might not be a right model for classification study for this dataset***.

4. Logistic Regression with c=5

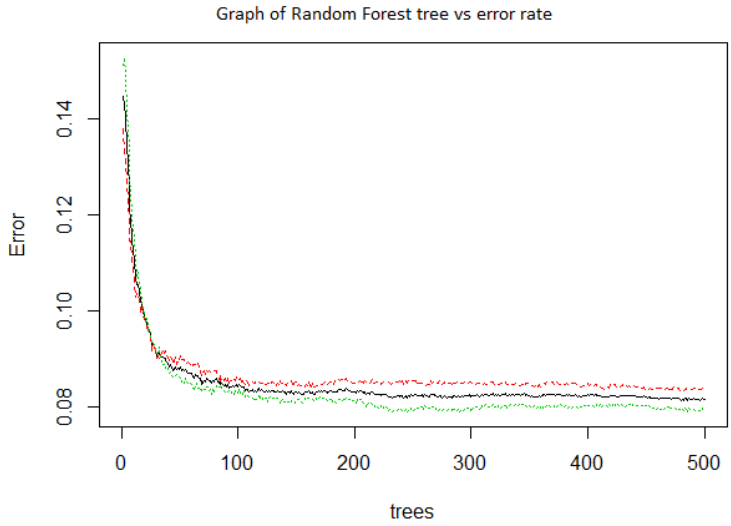
Logistic regression work on a linear function measure to classify the target variable. Out of all individual module discussed so far it is this classifier which produce high accuracy of 83.2%, below is the confusion matrix for the same:

LGReg	Reference	
Predicted	0	1
0	6663	232
1	1287	860

Logistic regression does pretty well in classifying both positive and negative cases, that is, it help to classify both user who buy and who don't buy term deposit product with high accuracy comparatively. And the confidence interval of this accuracy is very close which emphasize the trust we can have on this accuracy measure. So, compare to other individual models discussed so far, we can say that model built with logistic regression produce high Sensitivity of 83.81% Thus, ***this logistic model can be used to classify user who don't buy term deposit product in general.***

5. Bagging - Ensemble on Decision Tree – Random Forest

Bagging is bootstrap aggregation, here we have done this bagging ensemble technique using various decision tree models. As part of this graph we had built 500 random decision trees and measured the error rate that each tree generated.



RF	Reference	
Predicted	0	1
0	6955	293
1	995	799

In above graph, line in black color denotes the overall classification error, red denoted those error caused by positive cases that is users who buy term deposit product and green denotes the error caused by users who don't buy term deposit product. Also from above confusion matrix ***we see that overall accuracy has been improved to 85.76% which is way greater than individual decision tree that produces 73.51% accuracy***. Thus, this ensemble technique has improved model performance which answers our question of this case study.

6. Boosting – Boosted Logistic Regression

Boosting is the single classifier with different samples whose weighted average result is the model output for boosting ensemble learnings. Since, logistic regression did well in predicting the negative cases well, I thought of using a boosting technique on this classifier to increase its classification performance. I used 'Logit Boost' method provided by R to do boosting on Logistic regression model. Below is the confusion matrix obtained:

BoostLG	Reference	
Predicted	0	1
0	7416	517
1	534	575

From this confusion matrix we can compute over all accuracy of the model being 88.38% with 93.28% sensitivity. That is this ***boosted model can predict cases who don't buy term deposit product correctly 93.28% of times***.

Model Comparison and Inference:

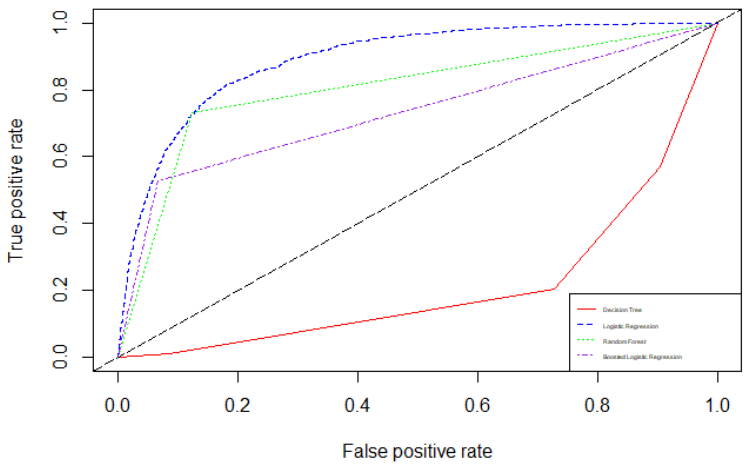
Below is the consolidated evaluation measure of all above algorithms discussed.

	Training time Recorded			Testing time recorded										
SMOTE Sampled Datset	user	system	elapsed	user	system	elapsed	Acc	95% CI (low, high)	Sen	Spe	Precision	Recal	F-Measure	P-value
DT	8.89	0.28	10.61	0.06	0.01	0.12	0.7351	(0.7259, 0.7442)	0.727	0.794	0.794	0.285	0.21	<2e-16
NB	165.57	0.11	166.76	48.33	0.02	48.63	0.5772	(0.5669, 0.5874)	0.5307	0.9158	0.916	0.211	0.172	<2e-16
KNN	157.12	0.05	158.26	24.90	0.03	25.41	0.6794	(0.6697, 0.689)	0.6898	0.6035	0.603	0.211	0.156	<2e-16
LG	5.99	0.10	6.46	0.11	0.00	0.21	0.832	(0.8241, 0.8397)	0.8381	0.7875	0.788	0.401	0.266	<2e-16
RF	1439.92	10.86	1480.22	0.86	0.05	2.83	0.8576	(0.8502, 0.8647)	0.8748	0.7317	0.732	0.445	0.277	<2e-16
Boosted LG	154.47	0.28	157.77	1.01	0.00	1.05	0.8838	(0.877, 0.8903)	0.9328	0.5266	0.527	0.518	0.261	0.622

We see a significant improvement in accuracy and Sensitivity when the dataset is subjected to ensemble technique. We have used two ensemble techniques as part of this case study-1:

1. **Bagging Method** with decision tree classifier was built using the Random Forest algorithm, from above table we can see the ***significant improvement in the model accuracy from 0.7351 for ordinary Decision Tree to 0.8576 accuracy*** with Random forest.

2. **Boosting method** was done on logistic regression model to measure its performance improvement, we see that accuracy rate was 0.832 for simple logistic regression. However, for the boosted logistic regression we see a accuracy improvement of 0.8838. ***Important thing to notice from this boosting result is the improvement in sensitivity from 0.8381 for individual logistic regression model compare to 0.9328 for boosted logistic regression model***.



Conclusion

Thus, we can conclude that ensemble techniques when applied to a dataset produce an efficient result most of the times when compare to one single model classification. However, we cannot say that ensemble method will always improve the performance because, ensemble methods are aggregation methods with multiple algorithms differed by its parameters or multiple samples done with and without replacement and result produced by an ensemble model is the average of all model performance and hence, at times there are chances where results produced by ensemble technique might go less efficient.

However, for our dataset we see a considerable improvement with respective to model performance due to the application of ensemble technique both for bootstrap aggregation and boosting techniques applied to the dataset and we can finally suggest to Bank with this dataset that:

1. ***To classify a user who will buy term deposit product use model built out of Naïve Bayes*** if the intension is just classification. ***To classify a user who will not buy term deposit product use model built out of boosted logistic regression*** if the intension is just classification.

2. ***If the intention is to explain visually, then decision tree model is best***. ***To achieve over all classification accuracy from a single model Random forest works better***.

Reference:

http://jmlr.csail.mit.edu/papers/volume15/delgado14a/delgado14a.pdf

http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0094137

https://datascience52.files.wordpress.com/2017/02/machine-learning-on-uci-adult-data-set-using-various-classifier-algorithms-and-scaling-up-the-accuracy-using-extreme-gradient-boosting.pdf

https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research

http://www.sciencedirect.com/science/article/pii/S016792361400061X?via%3Dihub#ab0010

http://www.dominikkowald.info/documents/2016ht_airline.pdf

Python JUPYTER notebook for Data Manipulation and Cleaning:

Bank_Direct_Marketing.ipynb

JUPYTER notebook in HTML format:

Bank_Direct_Marketing.html

Click here to download html file…

R-Code for Sampling and Model Construction:

Data Split:

```
# Setting the working directory
setwd("D:/Courses/CSC529 - Python/Case_Study1/Submission")

# Loading the file to R
data_raw_dummies = read.csv(file="data_treated.csv",header = TRUE)
# Removing the index generated out of pandas library
data_raw_dummies = data_raw_dummies[2:42]

#######################################
# Data Splitting - 80 - 20 split
#######################################

# Installing the necessary library package
library(caret)
# Creating a random index to split the data as 80 - 20%
idx = createDataPartition(data_raw_dummies$y_yes, p=.80, list=FALSE)
#print(idx)
# Using the index created to create a Training Data set - 131 observations created
train_data1 = data_raw_dummies[idx,]
# Using the index created to create a Testing Data set - 31 observations created
test_data1 = data_raw_dummies[-idx,]


# Data describe
table(train_data1$y_yes)
table(test_data1$y_yes)

#check classes distribution
prop.table(table(train_data1$y_yes))
```

Sampling to address imbalance

```
install.packages("ROSE")
library(ROSE)
accuracy.meas(test_data1$y_yes, pred.treeimb)

# ROC coverage
roc.curve(test_data1$y_yes, pred.treeimb, plotit = T)

help("ovun.sample")
# Over sampling
data_balanced_over <- ovun.sample(y_yes ~ ., data = train_data1, method = "over",N = 63822)$data
table(data_balanced_over$y_yes)


# under sampling without replacement
data_balanced_under <- ovun.sample(y_yes ~ ., data = train_data1, method = "under", N = 8516, seed = 1)$data
table(data_balanced_under$y_yes)

# Doing both
data_balanced_both <- ovun.sample(y_yes ~ ., data = train_data1, method = "both", p=0.5, N=8516, seed = 1)$data
table(data_balanced_both$y_yes)

# Applying ROSE method
data.rose <- ROSE(y_yes ~ ., data = train_data1, seed = 1)$data
table(data.rose$y_yes)

# Applying SMOTE method
install.packages("DMwR")
library(DMwR)
set.seed(1234)
help("SMOTE")
str(train_data1)
train_data2 = train_data1
train_data2$y_yes = as.factor(train_data2$y_yes)
data.smote <- SMOTE(y_yes~., train_data2, perc.over= 200, perc.under = 150)
table(data.smote$y_yes)
```

Comparing and choosing best sampling technique using Decision Tree classifier evaluation

```
# Now lets build a DT on each of the different sampling done
tree.rose <- rpart(y_yes ~ ., data = data.rose)
tree.over <- rpart(y_yes ~ ., data = data_balanced_over)
tree.under <- rpart(y_yes ~ ., data = data_balanced_under)
tree.both <- rpart(y_yes ~ ., data = data_balanced_both)
tree.smote <- rpart(y_yes ~ ., data = data.smote)

#make predictions on unseen data
pred.tree.rose <- predict(tree.rose, newdata = test_data1)
pred.tree.over <- predict(tree.over, newdata = test_data1)
pred.tree.under <- predict(tree.under, newdata = test_data1)
pred.tree.both <- predict(tree.both, newdata = test_data1)
pred.tree.smote <- predict(tree.smote, newdata = test_data1)


#AUC ROSE - 0.758
roc.curve(test_data1$y_yes, pred.tree.rose,plotit=TRUE, col='Red')
#AUC Oversampling - 0.855
roc.curve(test_data1$y_yes, pred.tree.over,plotit=TRUE,add.roc=TRUE, col='Green')
#AUC Undersampling - 0.839
roc.curve(test_data1$y_yes, pred.tree.under,plotit=TRUE,add.roc=TRUE, col='Blue')
#AUC Both - 0.858
roc.curve(test_data1$y_yes, pred.tree.both,plotit=TRUE,add.roc=TRUE, col='Purple')
#AUC SMOTE - 0.817
roc.curve(test_data1$y_yes, pred.tree.smote[,2],plotit=TRUE,add.roc=TRUE, col='Brown')

legend('bottomright',
       legend=c('Rose', 'Over_Sampling', 'Under_Sampling', 'Both_Over_Under_Sampling_Rose','SMOTE'),
       col = c('red', 'Green', 'Blue', 'Purple','Brown'),
       lty = 1:4,
       cex = 0.3)
```

Comparing and choosing best sampling technique using KNN (K=5) classifier evaluation

```
# train a KNN model with K = 5
# Now lets build a KNN on each of the different sampling done
help("knn3")
knn.rose <- knn3(y_yes ~ ., data = data.rose, k=5)
knn.over <- knn3(y_yes ~ ., data = data_balanced_over, k=5)
knn.under <- knn3(y_yes ~ ., data = data_balanced_under, k=5)
knn.both <- knn3(y_yes ~ ., data = data_balanced_both, k=5)
knn.smote <- knn3(y_yes ~ ., data = data.smote, k=5)


#make predictions on unseen data
pred.knn.rose <- predict(knn.rose, newdata = test_data1)
pred.knn.over <- predict(knn.over, newdata = test_data1)
pred.knn.under <- predict(knn.under, newdata = test_data1)
pred.knn.both <- predict(knn.both, newdata = test_data1)
pred.knn.smote <- predict(knn.smote, newdata = test_data1)

#AUC ROSE - 0.662
roc.curve(test_data1$y_yes, pred.knn.rose[,1],plotit=TRUE, col='Red')
#AUC Oversampling - 0.667
roc.curve(test_data1$y_yes, pred.knn.over[,1],plotit=TRUE,add.roc=TRUE, col='Green')
#AUC Undersampling - 0.704
roc.curve(test_data1$y_yes, pred.knn.under[,1],plotit=TRUE,add.roc=TRUE, col='Blue')
#AUC Both - 0.686
roc.curve(test_data1$y_yes, pred.knn.both[,1],plotit=TRUE,add.roc=TRUE, col='Purple')
#AUC Smote - 0.695
roc.curve(test_data1$y_yes, pred.knn.smote[,1],plotit=TRUE,add.roc=TRUE, col='Brown')

legend('bottomright',
       legend=c('Rose', 'Over_Sampling', 'Under_Sampling', 'Both_Over_Under_Sampling_Rose','SMOTE'),
       col = c('red', 'Green', 'Blue', 'Purple','Brown'),
       lty = 1:4,
       cex = 0.3)
```

Naïve Bayes Algorithm on SMOTE sampled data

```
# Naive Bayes Method classifier
# Recording the time - starts
ptm <- proc.time()
model1_nb <- train(y_yes~., data = data.smote, method='nb', trControl = trainctrl)
# Stop the clock
proc.time() - ptm

# Recording the time - starts
ptm <- proc.time()
pred_model1_nb <- predict(model1_nb, test_data1)
# Stop the clock
proc.time() - ptm

# Model Evaluation
confusionMatrix(pred_model1_nb, test_data1$y_yes)
accuracy.meas(pred_model1_nb,test_data1$y_yes)
```

Decision Tree Algorithm on SMOTE sampled data

```
# We will work on sample data produced by SMOTE as it has a reliable ROC curv with both classifier
library(caret)
# Decision Tree using 5-fold cross validation (ACC = 0.72, Sen = 0.71, Spe = 0.80)
help("trainControl")
trainctrl <- trainControl(method = "cv", number = 5)

# Recording the time - starts
ptm <- proc.time()
model1_tree <- train(y_yes~., data = data.smote, method='rpart', trControl = trainctrl)
# Stop the clock
proc.time() - ptm

# Recording the time - starts
ptm <- proc.time()
pred_model1_tree <- predict(model1_tree, test_data1)
# Stop the clock
proc.time() - ptm

# Model evaluation
confusionMatrix(pred_model1_tree, test_data1$y_yes)
accuracy.meas(pred_model1_tree,test_data1$y_yes)

library(rattle)
fancyRpartPlot(model1_tree$finalModel)

# Variable ranking based on importance
model1_tree$finalModel$variable.importance
#duration_min_max_norm       contact_unknown      contact_cellular  balance_min_max_norm
#3765.194048             693.614990             351.788572             158.543329
#campaign_min_max_norm            housing_no           housing_yes                 pdays
#117.341840              52.962845              52.962845              49.847384
#poutcome_unknown              previous            day_z_norm          month_z_norm
#49.785075              49.535838              31.972355              31.972355
#age_z_norm
#3.955343
```

KNN Algorithm with K=5 on SMOTE sampled data

```
# KNN method classifier
# Recording the time - starts
ptm <- proc.time()
model1_knn <- train(y_yes~., data = data.smote, method='knn', trControl = trainctrl)
# Stop the clock
proc.time() - ptm

# Recording the time - starts
ptm <- proc.time()
pred_model1_knn <- predict(model1_knn, test_data1)
# Stop the clock
proc.time() - ptm

# Computing metrics
confusionMatrix(pred_model1_knn, test_data1$y_yes)
accuracy.meas(pred_model1_knn, test_data1$y_yes)
```

Logistic Regression Algorithm on SMOTE sampled data

```
# Logistic Regression with 5 fold (ACC = 0.844, Sen = 0.85, Spe = 0.78)
# Recording the time - starts
ptm <- proc.time()
model2_logistic <- train(y_yes~., data = data.smote, method='glm', trControl = trainctrl)
# Stop the clock
proc.time() - ptm

# Recording the time - starts
ptm <- proc.time()
pred_moel2_logistic <- predict(model2_logistic, test_data1)
# Stop the clock
proc.time() - ptm

confusionMatrix(pred_moel2_logistic, test_data1$y_yes)
accuracy.meas(pred_moel2_logistic, test_data1$y_yes)

model2_logistic$finalModel
```

Random Forest Algorithm on SMOTE sampled data – Ensemble Bagging

```
# Ensemble approach on decision tree - (ACC = 0.8654, Sen = 0.87, Spe = 0.81)
# Random Forest: training with 5-fold CV (takes time to train and find the best model)

# Recording the time - starts
ptm <- proc.time()
model3_rf <- train(y_yes~., data = data.smote, method='rf', trControl = trainctrl)
# Stop the clock
proc.time() - ptm

# Recording the time - starts
ptm <- proc.time()
pred_moel3_rf <- predict(model3_rf, test_data1)
# Stop the clock
proc.time() - ptm

confusionMatrix(pred_moel3_rf, test_data1$y_yes)
accuracy.meas(pred_moel3_rf, test_data1$y_yes)

plot(model3_rf$finalModel)
model3_rf$finalModel$importance
```

Boosted Logistic Algorithm on SMOTE sampled data – Ensemble Boosting

```
# Boosted logistic regression
# Recording the time - starts
ptm <- proc.time()
model4_boosted <- train(y_yes~., data = data.smote, method='LogitBoost')
# Stop the clock
proc.time() - ptm

# Recording the time - starts
ptm <- proc.time()
pred_moel4_boosted <- predict(model4_boosted, test_data1)
# Stop the clock
proc.time() - ptm

# Evaluation
confusionMatrix(pred_moel4_boosted, test_data1$y_yes)
accuracy.meas(pred_moel4_boosted, test_data1$y_yes)
model4_boosted$finalModel$Stump
model4_boosted$finalModel$Stump
```

Evaluation with ROC curve plot

```
# ROC curves of all models

library(ROCR)

tree.pred <- prediction(predict(model1_tree$finalModel, test_data1, type='prob')[,1], test_data1$y_yes)
tree.perf <- performance(tree.pred, "tpr", "fpr")
logit.pred <- prediction(predict(model2_logistic$finalModel, test_data1, type='response'), test_data1$y_yes)
logit.perf <- performance(logit.pred, "tpr", "fpr")
rf.pred <- prediction(as.numeric(predict(model3_rf$finalModel, test_data1)), as.numeric(test_data1$y_yes))
rf.perf <- performance(rf.pred, "tpr", "fpr")
bglm.pred <- prediction(as.numeric(predict(model4_boosted$finalModel, test_data1)), as.numeric(test_data1$y_ye
bglm.perf <- performance(bglm.pred, "tpr", "fpr")

plot(tree.perf, col = 'red', lty=1)
plot(logit.perf, add = TRUE, col = 'blue', lty=2)
plot(rf.perf, add=TRUE, col='green', lty=3)
plot(bglm.perf, add=TRUE, col='purple', lty=4)
abline(0, 1, lty = 5)

legend('bottomright',
       legend=c('Decision Tree', 'Logistic Regression', 'Random Forest', 'Boosted Logistic Regression'),
       col = c('red', 'blue', 'green', 'purple'),
       lty = 1:4,
       cex = 0.3)
```