STAT 860: Applying Machine Learning Algorithms for Predicting Subscription of Bank Telemarketing Customers

Sharmin Hossain- Student ID: 1337949

10 May 2022

Introduction

Marketing is technique of exposing the target customers to a product via suitable systems and channels.Telemarketing is form of direct marketing in which salesperson approaches the customer either face to face or phone call and persuade him to buy the product.Making right decisions in organizational operations are sometimes proved a great challenge where the quality of decision really matters.

Decision Support Systems (DSS) are classified as a particular class of computerized facts and figures that helps the organization or administration into their decision making actions.Data mining (DM) plays a vital role to support the Decision support systems which are based on the data obtained from the data mining models: rules, patterns and relationship. A specific technology used within the DSS is Machine learning (ML) that combines data and computer applications to accurately predicting the results. The fundamental principle of machine learning is to construct the algorithms that can obtain input data and then predict the results or outputs by using the statistical analysis within satisfactory interval. ML allows the DSS to obtain the new knowledge which helps it to make right decisions.

Bank is one of the organisations that uses telemarketing method for selling banking products or services. Telemarketing is a popular method used by bank to selling, because bank products and services sometimes too complicated for some users to understand.Bank as financing organisation really cares about good reputation and good branding, and one of bad thing do telemarketing can interfere reputation itself. So we need to find out which our target will not buy product or service if bank offer product or service using telemarketing. It can help protect bank reputation by not disturbing target that we already know will not buy the product.

Dataset

The data which is used in this project is the data of customers of a Portuguese banking institution. Data set which is utilized for this project has been taken from University of California, Irvine (UCI) machine learning repository website which is openly available for the public for research purpose. This data set reflects the real information which is related with direct marketing campaigns of a Portuguese retail bank, from period of May 2008 to November 2010, in a total of 45,211 phone contacts (observation) and 17 attributes including response variable. The marketing campaigns were based on phone calls. In many of the cases, more than one contact to the same customer was essential in order to know that if the product (bank term deposit) will be subscribed ('yes') or not subscribed ('no'). The details of 17 attributes are given below:

Variable	**Description**
age	numeric, age of client
job	categorical, type of job (admin, unknown, unemployed, management, housemaid, entrepreneur, student, blue-collar, self-employed, retired, technician, services)
marital	categorical, marital status (married, divorced, single. Here "divorced" states the both divorced or widowed)
education	categorical (unknown, secondary, primary and tertiary)
default	binary, customer credit is in default (yes,no)
balance	numeric, average yearly balance (in euros)
housing	binary, status of housing loan (yes,no)
loan	binary, clients personal loan (yes,no)
contact	categorical, contact communication type (unknown, telephone, cellular)
day	numeric, the last contact day of the month range (1-31)
month	categorical, last contact month of the year
duration	numeric, last contact duration (in seconds)
campaign	numeric, number of contacts performed during this campaign
pdays	numeric, number of days that passed by after the client was last contacted from a previous campaign
previous	numeric, number of contacts which are made before this campaign
poutcome	categorical, result or outcome of the previous marketing campaign (unknown, other, failure, success)
y	binary, (desired target) output variable whether client subscribed a term deposit or not

Figure 1: Variable Description

Analysis Method

In this project, I will use machine learning to understand pattern and predict classification , I will use several predictive model to predict using training and testing data. Predictive models I will use are Decision Tree and Random Forest and Support Vector Machine (SVM).

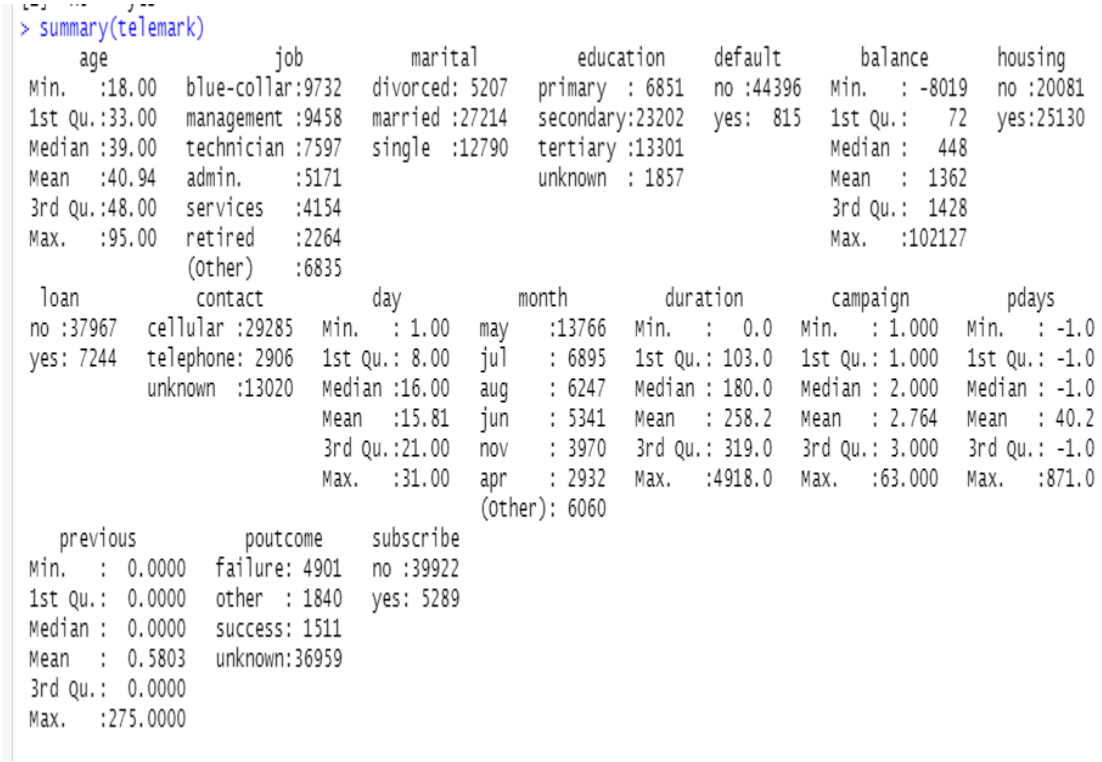
I will compare the result of prediction and see the performance from each model. This 3 models are categorized as supervised learning. Supervised learning is popular to predict pattern, this pattern can learn from train data and do ETL (Extract Transform Load) to get feature information. Based from features, I will compare the prediction results and make the final decision which model works best in this case.

Data Preparation & Data Wrangling

After importing necessary libraries, i have loaded the dataset in my R program. Here, i have showed a glimpse of my database variables.

```
> telemark <- read.csv("bank_full.csv",sep=";")
> glimpse(telemark)
Rows: 45,211
Columns: 17
$ age       <int> 58, 44, 33, 47, 33, 35, 28, 42, 58, 43, 41, 29, 53, 58, 57, 51, 45, 57, 60, 33, 28, 56, 32, 25~
$ job       <chr> "management", "technician", "entrepreneur", "blue-collar", "unknown", "management", "managemen~
$ marital   <chr> "married", "single", "married", "married", "single", "married", "single", "divorced", "married~
$ education <chr> "tertiary", "secondary", "secondary", "unknown", "unknown", "tertiary", "tertiary", "tertiary"~
$ default   <chr> "no", "no", "no", "no", "no", "no", "no", "yes", "no", "no", "no", "no", "no", "no", "no", "no~
$ balance   <int> 2143, 29, 2, 1506, 1, 231, 447, 2, 121, 593, 270, 390, 6, 71, 162, 229, 13, 52, 60, 0, 723, 77~
$ housing   <chr> "yes", "yes", "yes", "yes", "no", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes~
$ loan      <chr> "no", "no", "yes", "no", "no", "no", "yes", "no", "no", "no", "no", "no", "no", "no", "no", "n~
$ contact   <chr> "unknown", "unknown", "unknown", "unknown", "unknown", "unknown", "unknown", "unknown", "unkno~
$ day       <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
$ month     <chr> "may", "may", "may", "may", "may", "may", "may", "may", "may", "may", "may", "may", "may", "ma~
$ duration  <int> 261, 151, 76, 92, 198, 139, 217, 380, 50, 55, 222, 137, 517, 71, 174, 353, 98, 38, 219, 54, 26~
$ campaign  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1~
$ pdays     <int> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1~
$ previous  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
$ poutcome  <chr> "unknown", "unknown", "unknown", "unknown", "unknown", "unknown", "unknown", "unknown", "unkno~
$ y         <chr> "no", "no", "no", "no", "no", "no", "no", "no", "no", "no", "no", "no", "no", "no", "no", "no"~
> #checking if there is a missing value
```

Figure 2: Database Variables



```
> summary(telemark)
      age                job          marital          education     default        balance       housing
 Min.   :18.00   blue-collar:9732   divorced: 5207   primary  : 6851   no :44396   Min.   : -8019   no :20081
 1st Qu.:33.00   management :9458   married :27214   secondary:23202   yes:  815   1st Qu.:    72   yes:25130
 Median :39.00   technician :7597   single  :12790   tertiary :13301               Median :   448
 Mean   :40.94   admin.     :5171                    unknown  : 1857               Mean   :  1362
 3rd Qu.:48.00   services   :4154                                                  3rd Qu.:  1428
 Max.   :95.00   retired    :2264                                                  Max.   :102127
                 (Other)    :6835
  loan           contact           day             month          duration        campaign          pdays
 no :37967   cellular :29285   Min.   : 1.00   may    :13766   Min.   :   0.0   Min.   : 1.000   Min.   : -1.0
 yes: 7244   telephone: 2906   1st Qu.: 8.00   jul    : 6895   1st Qu.: 103.0   1st Qu.: 1.000   1st Qu.: -1.0
             unknown  :13020   Median :16.00   aug    : 6247   Median : 180.0   Median : 2.000   Median : -1.0
                               Mean   :15.81   jun    : 5341   Mean   : 258.2   Mean   : 2.764   Mean   : 40.2
                               3rd Qu.:21.00   nov    : 3970   3rd Qu.: 319.0   3rd Qu.: 3.000   3rd Qu.: -1.0
                               Max.   :31.00   apr    : 2932   Max.   :4918.0   Max.   :63.000   Max.   :871.0
                                               (Other): 6060
    previous          poutcome     subscribe
 Min.   :  0.0000   failure: 4901   no :39922
 1st Qu.:  0.0000   other  : 1840   yes: 5289
 Median :  0.0000   success: 1511
 Mean   :  0.5803   unknown:36959
 3rd Qu.:  0.0000
 Max.   :275.0000
```

Figure 3: Summary of Variables

As missing values (NA) can be a problem for modeling, i need to check if there are any missing values in my dataset or not. Turns out there are no missing values here.

```
> table(is.na(telemark))

 FALSE
768587
>
```

Figure 4: Checking Missing Values

Here, I can see based on column description some our imported variables have incorrect data types. So, I will change the data type refer to column description.
Then, to visualize the numeric variables, I have plotted a histogram here.

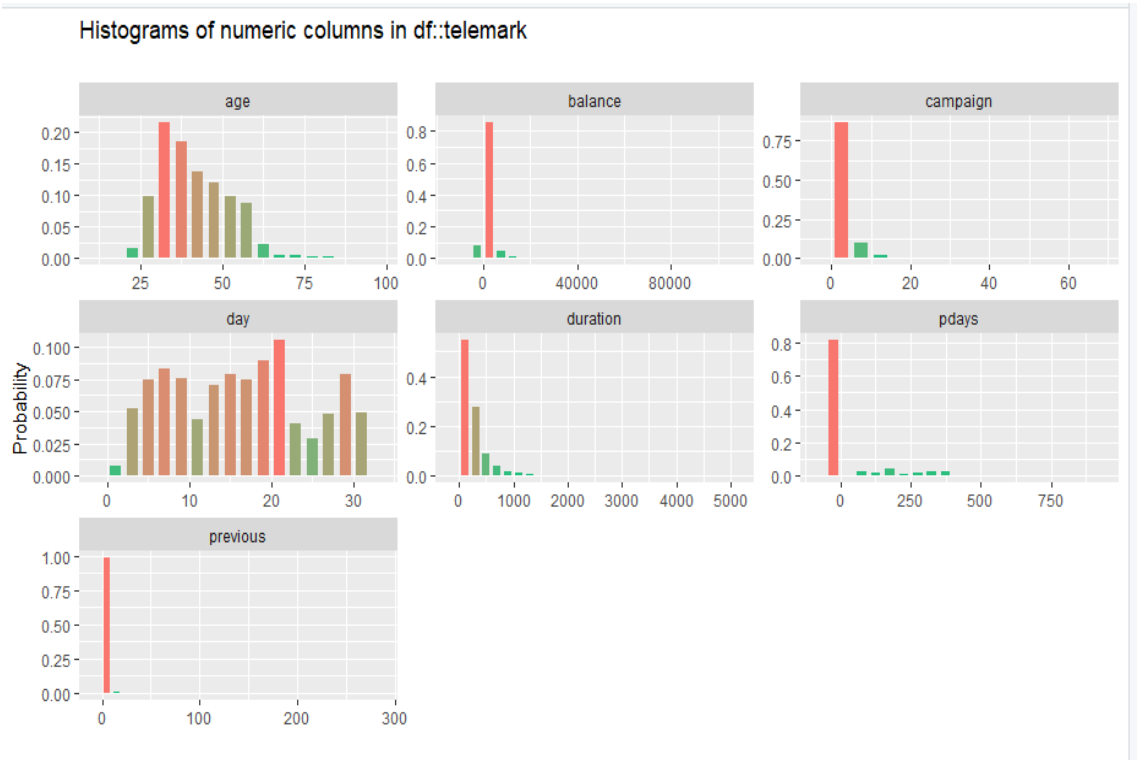


Figure 5: Histogram for Numeric Variables

Here, my target variables before are “Y” and I changed it to “subscribe”. Here target variables consist of 2 levels “yes” means users agree to subscribe or buy product and “no” means users didn’t agree or reject offers from telemarketing.

```
> levels(telemark$subscribe)
[1] "no"  "yes"
>
```

Figure 6: Levels of Response Variable

Exploratory Data Analysis

When conducting a supervised classification with machine learning algorithms, one recommended practice is to work with a balanced classification dataset. Checking proportion of our response variables ’subscribe’ I found that our response variable ’subscribe’ is Imbalanced. It means that data refers to a sit-

uation where the number of observations is not the same for all the classes in a classification dataset. To avoid loss of variance, I will use up-sampling to balance the proportion.

```
> #Checking proportion of Response Variable (Found Imbalanced Dataset)
> prop.table(table(telemark$subscribe))

       no       yes
0.8830152 0.1169848
>
```

Figure 7: Checking Proportion of Response variable

Looking at the correlation plot, i can see that there are some predictor variables that have correlation with other predictor variables. These variables are previous with pdays, campaign with day, and balance with age. I will not remove any variables based on this but will keep in mind that this correlation cause some issues with some models like Naive Bayes.

Figure 8: Correlations of Variables

Cross Validation

Cross-validation (CV) is a statistical method that can be used to evaluate the performance of models or algorithms where the data is separated into two subsets namely learning process data and validation / evaluation data. In this case we will separate data with proportion 80% dataset for data training and rest 20% we use as data test.

```
> prop.table(table(telemark_train$subscribe))

       no       yes
0.8830181 0.1169819
>
```

Figure 9: Checking proportion of target variable on train data

Here, i found target variable still imbalance, so i will try to make it balance using SMOTE method. SMOTE is a oversampling technique which synthesizes a new minority instance between a pair of one minority instance and one of its K nearest neighbor.

```
> #Checking proportion of Response Variable of training dataset again (Found Balanced Dataset)
> prop.table(table(telemark_train_upsample$subscribe))

 no yes
0.5 0.5
>
```

Figure 10: Checking again proportion of target variable on train data

Now i have my balanced proportion data which is ready for modeling.

Modeling

Decision Tree

Decision Trees play s significant role in the field of data mining as they are really fast to construct as compared to the other data mining methods. They can easily handle the data even if it comprises of mixture of numeric and categorical predictor variables.The algorithm of DTs is to split the data-set to accomplish a homogeneous classification for the dependent variable. At every part, objective of algorithm goes for diminishing the entropy of the dependent variable in the

subsequent datasets by selecting the ideal part from various explanatory variables.

First, i created a model for general decision tree, though Accuracy, Sensitivity, Specificity of the model are quite good, our positive pred value was small (39%). I will try to tune it with some criteria such as mincriteria, minsplit and minbucket. Then the positive pred value went up to 96%.

```
> model_dtree <- ctree(subscribe ~ ., telemark_train_upsample)
> width(model_dtree)
[1] 113
> depth(model_dtree)
[1] 17
>
> dtree_prediction <- predict(model_dtree, telemark_test)
> dtree_prediction_raw <- predict(model_dtree, telemark_test,type = "prob")
>
> dtree_matrix <- confusionMatrix(dtree_prediction, telemark_test$subscribe, positive = "yes")
> dtree_matrix <- matrix_result(dtree_matrix, "Decision Tree")
> dtree_matrix
         Model  Accuracy Sensitivity Specificity Pos Pred Value
1 Decision Tree 0.8558001    0.805293   0.8624922      0.4369231
>
> model_dtree_tuning <- ctree(subscribe ~ ., telemark_train_upsample,
+                             control = ctree_control(mincriterion = 0.1, minsplit = 100, minbucket = 60))
>
> dtree_prediction_tuning <- predict(model_dtree_tuning, telemark_test)
>
> dtree_matrix_tuning <- confusionMatrix(dtree_prediction_tuning, telemark_test$subscribe)
> dtree_matrix_tuning <- matrix_result(dtree_matrix_tuning, "Decision Tree Tuning")
> dtree_matrix_tuning
                Model  Accuracy Sensitivity Specificity Pos Pred Value
1 Decision Tree Tuning 0.8441889    0.851221   0.7911153      0.9685095
>
```

Figure 11: Decision Tree Model

Random Forest

Random forest is one of the most broadly used machine learning algorithm for classification. Either the response variable is continuous or categorical, it works in both cases. According to Friedman et al. (2001) random forests starts to become stable at around 200 trees, whereas at 1000 trees the boosting of this still keeps on improving. If trees are much smaller or there is a presence of shrinkage then process of boosting starts to reduce. In contrast to the other non linear estimators, it is possible to fit the RF in one sequence with the cross

validation being completed. The training can be finished if the OOB (Out of Bag) error stabilizes itself.

```
> rforest_matrix
        Model  Accuracy Sensitivity Specificity Pos Pred Value
1 Random Forest 0.8575694   0.8336484   0.8607389      0.442327
>
```

Figure 12: Random Forest Model

After the general modeling, we can see the positive pred value is pretty low (44%). So i tried to fine tune it with a cutoff value by plotting those 4 points and see if i can take a cutoff from there. But unfortunately the pred value did not improve though the sensitivity got improved a bit.

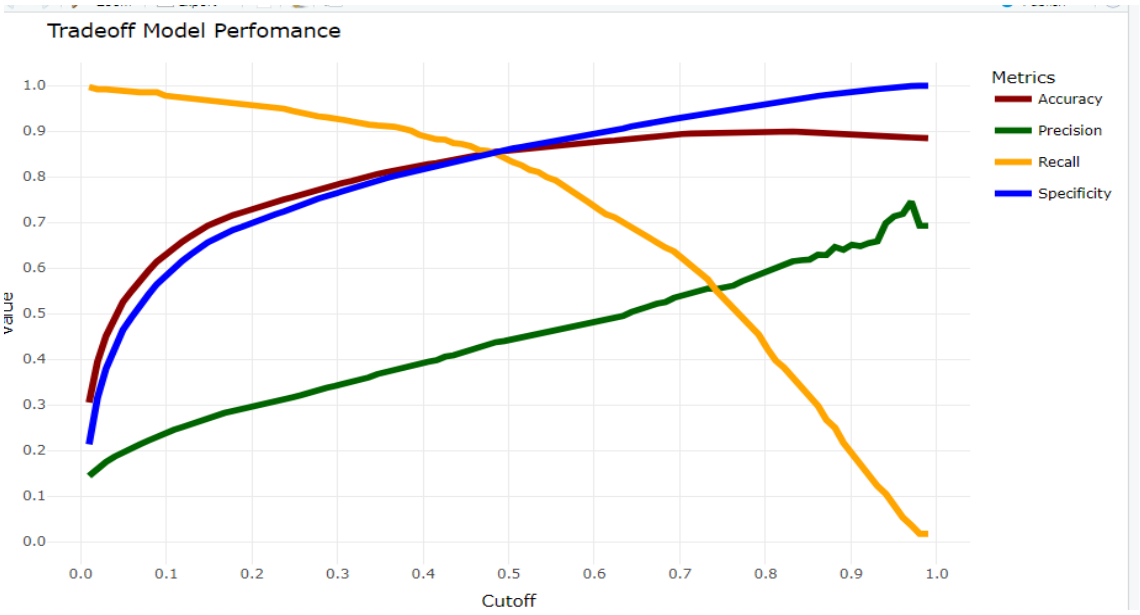


Figure 13: Setting Cutoff Value

```
>
> rforest_matrix_tuning <- confusionMatrix(rforest_predict_tuning$label, telemark_test$subscribe, positive = "yes")
> rforest_matrix_tuning <- matrix_result(rforest_matrix_tuning, "Random Forest Tuning")
> rforest_matrix_tuning
              Model  Accuracy Sensitivity Specificity Pos Pred Value
1 Random Forest Tuning 0.8509344   0.8563327   0.8502192      0.4310181
>
```

Figure 14: New Random Forest Model

Support Vector Machine

Support vector machines has been effectively utilized in numerous applications over the recent years. The great speculation capacity of SVM is that, it separate the two classes for which these are generally suitable. Among all the strategies of classification, the SVM has been broadly known.For the purpose of classifying the data point, a SVM calculates the dot product for the given test point with each of support vector either in the feature space or in the space of input after the conversion through a function i.e. kernel. In this manner, the time for the execution rises with the rising of number of support vectors.

Here, i have both tried with linear and radial SVM then decided to go with the linear model. Fine tuned with a lower cost value.

```
> svm_matrix
  Model  Accuracy Sensitivity Specificity Pos Pred Value
1   SVM 0.8402079   0.7533081    0.851722      0.4023221
>
> #For Radial
> model_svm1<- svm(subscribe~., data = telemark_train_upsample, kernel = "radial", cost = 1, gamma = 1)
> svm_prediction1 <- predict(model_svm1, telemark_test)
> svm_matrix1 <- confusionMatrix(svm_prediction1, telemark_test$subscribe, positive = "yes")
> svm_matrix1 <- matrix_result(svm_matrix1, "SVM")
>
> #Final Checking
> model_svm2<- svm(subscribe~., data = telemark_train_upsample, kernel = "linear", cost = 0.01, gamma = 1)
> svm_prediction2 <- predict(model_svm2, telemark_test)
> svm_matrix2 <- confusionMatrix(svm_prediction2, telemark_test$subscribe, positive = "yes")
> svm_matrix2 <- matrix_result(svm_matrix2, "SVM")
> svm_matrix2
  Model  Accuracy Sensitivity Specificity Pos Pred Value
1   SVM 0.8382174   0.7599244   0.8485911      0.3994039
> svm_prediction_tuning <- predict(model_svm2, telemark_test)
>
> svm_matrix_tuning <- confusionMatrix(svm_prediction_tuning, telemark_test$subscribe)
> svm_matrix_tuning <- matrix_result(svm_matrix_tuning, "SVM Tuning")
> svm_matrix_tuning
       Model  Accuracy Sensitivity Specificity Pos Pred Value
1 SVM Tuning 0.8382174   0.8485911   0.7599244      0.9638691
>
```

Figure 15: Support Vector Machine

Result

From summary result, i can see that random forest's accuracy is highest among all (85.7%). But in our case, we are more focused on people who don't want telemarketing calls because they are not going to be subscribed as per prediction which is formally known as sensitivity. Based on Sensitivity parameter, modified random forest performed the best among these models.

```
> result <- rbind(dtree_matrix, dtree_matrix_tuning, rforest_matrix, rforest_matrix_tuning,svm_matrix2,svm_m
ng)
> result
                Model  Accuracy Sensitivity Specificity Pos Pred Value
1       Decision Tree 0.8558001   0.8052930   0.8624922      0.4369231
2 Decision Tree Tuning 0.8441889   0.8512210   0.7911153      0.9685095
3       Random Forest 0.8575694   0.8336484   0.8607389      0.4423270
4 Random Forest Tuning 0.8509344   0.8563327   0.8502192      0.4310181
5                 SVM 0.8382174   0.7599244   0.8485911      0.3994039
6           SVM Tuning 0.8382174   0.8485911   0.7599244      0.9638691
>
> result %>% arrange(desc(Sensitivity))
                Model  Accuracy Sensitivity Specificity Pos Pred Value
1 Random Forest Tuning 0.8509344   0.8563327   0.8502192      0.4310181
2 Decision Tree Tuning 0.8441889   0.8512210   0.7911153      0.9685095
3           SVM Tuning 0.8382174   0.8485911   0.7599244      0.9638691
4       Random Forest 0.8575694   0.8336484   0.8607389      0.4423270
5       Decision Tree 0.8558001   0.8052930   0.8624922      0.4369231
6                 SVM 0.8382174   0.7599244   0.8485911      0.3994039
>
```

Figure 16: Result Summary

Conclusion

Conclusion is Random Forest with threshold tuning can gave us best performance for Sensitivity (True Positive Rate). Other models can be simulated for better results such as logistic regression and neural networks.

References

1. https://www.sciencedirect.com/science/article/pii/S016792361400061X
2. http://archive.ics.uci.edu/ml

R Codes

```
# Data wrangling Library
library(tidyverse)
library(dplyr)

# Visualize data
library(ggplot2)
library(inspectdf)
library(GGally)
library(plotly)

# SVM
library(e1071)

# Splitting Data
library(rsample)

# Random Forest
library(randomForest)

# Smote for unbalanced data
library(DMwR)

# ROCR
library(ROCR)

# Confussion Matrix
library(caret)

# Decision Tree
library(partykit)

#Extra Installs
library(dplyr)
library(GGally)
library(naivebayes)
library(tidyr)
library(plyr)


source("matrix_result.R")
source("metrics.R")
```

```
#Read the dataset and See summary
telemark <- read.csv("bank_full.csv",sep=";")
glimpse(telemark)

#Checking if there is a missing value
table(is.na(telemark))

#Changing data types
telemark <- telemark %>%
  mutate(job = as.factor(job),
         marital = as.factor(marital),
         education = as.factor(education),
         default = as.factor(default),
         housing = as.factor(housing),
         loan = as.factor(loan),
         contact = as.factor(contact),
         month = as.factor(month),
         poutcome = as.factor(poutcome),
         subscribe = as.factor(y)) %>%
  select(-c(y))

#Visualize numeric variables
numericCols <- unlist(lapply(telemark, is.numeric))
show_plot(inspect_num(telemark[,numericCols]))

#Levels of response variable
levels(telemark$subscribe)

#Overall Data Structure
summary(telemark)

#Checking proportion of Response Variable (Found Imbalanced Dataset)
prop.table(table(telemark$subscribe))

#Checking correlation between predictor variables
show_plot(inspect_cor(subset(telemark, select = -c(subscribe))))
ggcorr(telemark, label = T)

#Set up training & testing data
set.seed(1)
split <- initial_split(data = telemark, prop = 0.8, strata = subscribe)
telemark_train <- training(split)
telemark_test <- testing(split)

#Checking proportion of Response Variable of training dataset (Found Imbalanced Dataset)
prop.table(table(telemark_train$subscribe))
```

```
#Applying Smote Technique
telemark_train_upsample <- SMOTE(subscribe ~ ., as.data.frame(telemark_train),
perc.over = 100, perc.under = 200)

#Checking proportion of Response Variable of training dataset again
#(Found Balanced Dataset)
prop.table(table(telemark_train_upsample$subscribe))

#Applying Different Modeling

#Decision Trees

model_dtree <- ctree(subscribe ~ ., telemark_train_upsample)
width(model_dtree)
depth(model_dtree)

dtree_prediction <- predict(model_dtree, telemark_test)
dtree_prediction_raw <- predict(model_dtree, telemark_test,type = "prob")

dtree_matrix <- confusionMatrix(dtree_prediction, telemark_test$subscribe,
positive = "yes")
dtree_matrix <- matrix_result(dtree_matrix, "Decision Tree")
dtree_matrix

model_dtree_tuning <- ctree(subscribe ~ ., telemark_train_upsample,
                            control = ctree_control(mincriterion = 0.1,
                            minsplit = 100, minbucket = 60))

dtree_prediction_tuning <- predict(model_dtree_tuning, telemark_test)

dtree_matrix_tuning <- confusionMatrix(dtree_prediction_tuning,
telemark_test$subscribe)
dtree_matrix_tuning <- matrix_result(dtree_matrix_tuning,
"Decision Tree Tuning")
dtree_matrix_tuning

#Random Forest

ctrl <- trainControl(method = "repeatedcv", number = 5,repeats = 3)

model_rforest <- train(subscribe ~ ., data = telemark_train_upsample,
method = "rf", trControl = ctrl, ntree = 100)
model_rforest

varImp(model_rforest)
```

```
model_rforest$finalModel

plot(model_rforest$finalModel)
legend("topright", colnames(model_rforest$finalModel$err.rate),col=1:6,
cex=0.8,fill=1:6)

rforest_predict <- predict(model_rforest, telemark_test)
rforest_predict_raw <- predict(model_rforest, telemark_test, type = "prob")

rforest_matrix <- confusionMatrix(rforest_predict, telemark_test$subscribe,
positive = "yes")
table <- as.table(rforest_matrix)
table <- as.data.frame(table)

rforest_matrix <- matrix_result(rforest_matrix, "Random Forest")
rforest_matrix

co <- seq(0.01,0.99,length=100)
result <- matrix(0,100,4)

# apply function metrics
for(i in 1:100){
  result[i,] = metrics(cutoff = co[i],
                       prob = rforest_predict_raw$yes,
                       ref = as.factor(ifelse(telemark_test$subscribe ==
                       "yes", 1, 0)),
                       postarget = "1",
                       negtarget = "0")
}

# visualize
ggplotly(tibble("Recall" = result[,1],
                "Accuracy" = result[,2],
                "Precision" = result[,3],
                "Specificity" = result[,4],
                "Cutoff" = co) %>%
           gather(key = "Metrics", value = "value", 1:4) %>%
           ggplot(aes(x = Cutoff, y = value, col = Metrics)) +
           geom_line(lwd = 1.5) +
           scale_color_manual(values = c("darkred","darkgreen","orange", "blue")) +
           scale_y_continuous(breaks = seq(0,1,0.1), limits = c(0,1)) +
           scale_x_continuous(breaks = seq(0,1,0.1)) +
           labs(title = "Tradeoff Model Perfomance") +
           theme_minimal() +
           theme(legend.position = "top",
```

```
                panel.grid.minor.y = element_blank(),
                panel.grid.minor.x = element_blank()))

rforest_predict_tuning <- rforest_predict_raw %>%
  mutate(label = as.factor(ifelse(yes >= 0.48, "yes", "no"))) %>%
  select(label)

rforest_matrix_tuning <- confusionMatrix(rforest_predict_tuning$label,
telemark_test$subscribe, positive = "yes")
rforest_matrix_tuning <- matrix_result(rforest_matrix_tuning,
"Random Forest Tuning")
rforest_matrix_tuning


#SVM

#For Linear
model_svm<- svm(subscribe~., data = telemark_train_upsample, kernel = "linear",
cost = 1, gamma = 1)
svm_prediction <- predict(model_svm, telemark_test)
svm_matrix <- confusionMatrix(svm_prediction, telemark_test$subscribe,
positive = "yes")
svm_matrix <- matrix_result(svm_matrix, "SVM")
svm_matrix

#For Radial
model_svm1<- svm(subscribe~., data = telemark_train_upsample, kernel = "radial",
cost = 1, gamma = 1)
svm_prediction1 <- predict(model_svm1, telemark_test)
svm_matrix1 <- confusionMatrix(svm_prediction1, telemark_test$subscribe,
positive = "yes")
svm_matrix1 <- matrix_result(svm_matrix1, "SVM")

#Final Checking
model_svm2<- svm(subscribe~., data = telemark_train_upsample,
kernel = "linear", cost = 0.01, gamma = 1)
svm_prediction2 <- predict(model_svm2, telemark_test)
svm_matrix2 <- confusionMatrix(svm_prediction2, telemark_test$subscribe,
positive = "yes")
svm_matrix2 <- matrix_result(svm_matrix2, "SVM")
svm_matrix2

summary(model_svm2)

svm_prediction_tuning <- predict(model_svm2, telemark_test)
```

```
svm_matrix_tuning <- confusionMatrix(svm_prediction_tuning,
telemark_test$subscribe)
svm_matrix_tuning <- matrix_result(svm_matrix_tuning, "SVM Tuning")
svm_matrix_tuning

#All Combined

result <- rbind(dtree_matrix, dtree_matrix_tuning,
rforest_matrix, rforest_matrix_tuning,
svm_matrix2,svm_matrix_tuning)
result

result %>% arrange(desc(Sensitivity))
```