R code for Santander customer transaction prediction without sampling and prediction using Logistic regression, Decision trees, Random Forest , Naïve Bayes and KNN classification models

```
# remove the objects stored
rm(list=ls())
#set working directory
setwd("G:/Edwiser material/Project/Santandarcustomer problems/Edwiser project")
#check the set directory
getwd()
# Load required Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
    "dummies", "e1071", "Information", "MASS", "rpart", "gbm", "ROSE", 'sampling',
    'DataCombine', 'inTrees','dplyr','class','scales', "InformationValue",
    "RDocumentation", 'mlbench')
#Installing packages(x)
lapply(x, require, character.only = TRUE)
#clear all the x objects
rm(x)
# Load the both train and test data
train = read.csv("train.csv")
test = read.csv("test.csv")
# check for the dimensions of the data
dim(train)
dim(test)
str(train)
str(test)
head(train,4)
head(test,4)
#Remove the ID_code column  which is nothing but the code and does not have information
train$ID_code=NULL
```

```
test$ID_code=NULL
############DATA PREPROCESSING###############
#####Missing value Analysis
#check the missing values in train data
missing_val_train=data.frame(apply(train, 2, function(x){sum(is.na(x))}))
#Check the missing values in the test data
missing_val_test=data.frame(apply(test, 2, function(x){sum(is.na(x))}))
#Check the distribution of the target variable
ggplot(train, aes_string(x=train$target))+ geom_bar(stat="count",
fill="DarkSlateBlue",)+
 theme_bw()+ xlab("Target")+ ylab('Count')+ ggtitle("Distribution of target classes")+
 theme(text = element_text(size = 15))
###############Outlier Analysis#################
#convert the target variable from numeric to factor
train$target=as.factor(train$target)
#Select all the numeric variables in train data
numeric_index=sapply(train, is.numeric)
numeric_data=train[,numeric_index]
cnames=colnames(numeric_data)
#plot the boxplot for checking the outliers in the train data
#for (i in 1:length(cnames))
#{
#  assign(paste0("gn",i), ggplot(aes_string(y=(cnames[i]), x="responded"), data=
subset(marketing))+
#         stat_boxplot(geom = "errorbar", width = 0.5)+
#         geom_boxplot(outlier.colour = "red", fill = "purple", outlier.shape = 18,
#                      outlier.size = 1, notch = FALSE)+
#         theme(legend.position = "bottom")+
#         labs(y=cnames[i], x= "responded")+
#         ggtitle(paste("Bosplot of responded for", cnames[i])))
#}
#Plot plots together
```

```
#gridExtra::grid.arrange(gn1,gn5,gn2, ncol=3)
#gridExtra::grid.arrange(gn6,gn7,ncol=2)
#gridExtra::grid.arrange(gn8,gn9,ncol=2)
#remove outliers using boxplots
#loop to remove all outliers from all variables
#for(i in cnames){
#  print(i)
#  val=train[,i][marketing[,i]%in%boxplot.stats(train[,i])$out]
#  print(length(val))
#  train=train[which(!train[,i] %in% val),]
#}

dim(marketing)

#replace all the outliers replace with NA and impute with knn
#for(i in cnames){
#  val=marketing[,i][marketing[,i] %in% boxplot.stats(marketing[,i])$out]
#  marketing[,i][marketing[,i] %in% val] =NA
#}
#sum(is.na(marketing))
#impute all the outliers which has been replaced as NA  with KNN imputation
#marketing=knnImputation(marketing, k= 5)
###############Feature selection#####################
#Correlation plot analysis
#corrgram(train[,numeric_index], order= F,
#       upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation plot")
####################feature scaling#################################
#normalization check
qqnorm(train$var_0)
hist(train$var_1)
###standardization of the data as most of the data is normally distributed
#reload the data
```

```
#for( i in cnames){
#  train[,i]=(train[,i]-mean(train[,i]))/
#    sd(train[,i])
#}

#for( i in cnames){
#  test[,i]=(test[,i]-mean(test[,i]))/
#    sd(test[,i])
#}

summary(train$var_0)
range(train$var_0)
dim(train)
#Remove all the objects except the required ones
rmExcept(c("train","test","cnames"))
##################Sampling#####################
train.index=createDataPartition(train$target, p=0.75, list=FALSE)
train=train[train.index,]
test=train[-train.index,]
#############Model development#############
#####Logistic Regression
#Train logistic regression model on train data
Model_LR=glm(target~., data=train, family="binomial")
#Summary of logistic model
summary(Model_LR)
#Predict the test data target values with the trained logistic regression model
Predictions_LR = predict(Model_LR, newdata=test, type="response")
#convert probabilities into binary classes
Predictions_LR = ifelse(Predictions_LR >0.5, 1, 0)
#Evaluate performance of the logistic model with test data target values and predicted values
confMatrix_LR=table(test$target, Predictions_LR)
```

```
confMatrix_LR
#Plot roc curve
plotROC(test$target, Predictions_LR)
#calculating auroc value
AUROC(test$target, Predictions_LR)
#Precision value
precision(test$target,Predictions_LR)

#convert predictions into factors
Pred=as.factor(Predcitons_LR)
#Calculate recall value
recall(test$target,Pred)
#ACCUARCY:
#FNR:
######Random Forest Model########
Model_RF=randomForest(target~., train, importance=TRUE, ntree=50)
#Summary of the RF model
summary(Model_RF)
##Extract rules from the trained random forest model
#transform random forest object to intrees format
treeList=RF2List(Model_RF)
#Extract rules
exec=extractRules(treeList, train[,-1])
#Visualelise rules
exec[1:2,]
#Making rules more readable
readableRules=presentRules(exec, colnames(train))
readableRules[1:2,]
#Get the rule matrix
ruleMetrics=getRuleMetric(exec, train[,-1], train$target)
#Evaluate the rules
ruleMetrics[1:2,]
```

```
#Predict the test target variable class values with trained random forest model
Predictions_RF = predict(Model_RF, test[,-1])
#Evaluate the performance of the Random forest model
confMatrix_RF = table(test$target, Predictions_RF)
confMatrix_RF
#plot ROc curve to RF predictions
pred=as.numeric(Predictions_RF)
plotROC(test$target, pred)
AUROC(test$target, pred)
#precision calculation
precision(test$target, pred)
pred=as.factor(Predictions_RF)
recall(test$target, pred)
#Accuarcy:
#FNR:
#####Decision tree Model
Model_C50=C5.0(target~., data=train, trails=10, rules=TRUE)
summary(Model_C50)
#Write rules into disk
write(capture.output(summary(Model_C50)), "c50rules.txt")
#predict the test target values with trained decision tree model
Predictions_c50=predict(Model_C50, test[,-1], type='class')
#Evaluate of the performance of the c50 model on test target values with actual values
confMatrix_DT=table(test$target, Predictions_c50)
confMatrix_DT
#plot ROC curve to RF predictions
pred=as.numeric(confMatrix_DT)
plotROC(test$target, pred)
AUROC(test$target, pred)
#precision calculation
precision(test$target, pred)
pred=as.factor(confMatrix_DT)
```

```
recall(test$target, pred)


#ACCURACY:
#FNR:
######NAIVE BAYES ALGORITHM#################
Model_NB=naiveBayes(target~., data=train)
#predict on the test data target values
Predictions_NB=predict(Model_NB, test[,2:201], type="class")
#build a confusion matrix
confMatrix_NB=table(observed=test[,1], predicted=Predictions_NB)
confMatrix_NB
#Plot ROC
pred=as.numeric(Predictions_NB)
plotROC(test$target,pred)
AUROC(test$target, pred)
precision(test$target, pred)
pred=as.factor(Predictions_NB)
#calculate recall
recall(test$target,pred )
###########KNN Classification model##########
KNN_Predictions=knn(train[,2:201], test[,2:201], train$target, k=3)
#build a confusion matrix on actual and knn predicted values
confMatrix_KNN=table(test$target, KNN_Predictions)
#Convert predictions into numeric data
pred= as.numeric(NB_Predictions)
#plot ROC for predictions and target values
plotROC(test$target,  pred)
AUROC(test$target, pred)
precision(test$target, KNN_Predictions)
recall(test$target, KNN_Predictions)
#Accuarcy:
#FNR:
```

R code for Santander customer transaction prediction with Under sampling and prediction using random forest model

```
# remove the objects stored

rm(list=ls())


#setting working directory

setwd("G:/Edwiser material/Project/Santandarcustomer problems/Edwiser project")


#checking the set directory

getwd()


# Loading required Libraries

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
    "dummies", "e1071", "Information", "MASS", "rpart", "gbm", "ROSE", 'sampling',
    'DataCombine', 'inTrees','dplyr','class','scales', "InformationValue",
    "RDocumentation", 'mlbench')


#Installing packages(x)

lapply(x, require, character.only = TRUE)

rm(x)


# Loading the both train and test data

train = read.csv("train.csv")

test = read.csv("test.csv")


# checking for the dimensions of the data
```

```
dim(train)

dim(test)

str(train)

str(test)

head(train,4)

head(test,4)


#Removing the ID_code column  which is nothing but the code and does not have information

train$ID_code=NULL

test$ID_code=NULL


#converting the target variable from numeric to factor

train$target=as.factor(train$target)


#Selecting all the numeric variables in train data

numeric_index=sapply(train, is.numeric)

numeric_data=train[,numeric_index]

cnames=colnames(numeric_data)


#Removing all the objects except the required ones

rmExcept(c("train","test","cnames"))


#####################Sampling##############################

trainDataIndex = createDataPartition(train$target, p=0.7, list = F)

train = train[trainDataIndex, ]

test = train[-trainDataIndex, ]


# Class distribution of train data

table(train$target)
```

```
'%ni%' <- Negate('%in%')  # define 'not in' func
options(scipen=999)  # prevents printing scientific notations.

#Down Sampling
set.seed(100)
train = downSample(x = train[, colnames(train) %ni% "target"],
                   y = train$target)

table(train$Class)

##################Model development###############
######Random Forest Model########
Model_RF=randomForest(Class~., train, importance=TRUE, ntree=50)

#Summary of the RF model
summary(Model_RF)

##Extracting rules from the trained random forest model
#transforming random forest object to intrees format
treeList=RF2List(Model_RF)

#Extracting rules
exec=extractRules(treeList, train[,-1])

#Visualizing few rules
exec[1:2,]

#Making rules more readable
readableRules=presentRules(exec, colnames(train))
```

```
readableRules[1:2,]

#Getting rule matrix
ruleMetrics=getRuleMetric(exec, train[,-1], train$target)

#Evaluating the rules
ruleMetrics[1:2,]

#Predicting the test target variable class values with trained random forest model
Predictions_RF = predict(Model_RF, test[,-1])

#Evaluating the performance of the Random forest model
confMatrix_RF = table(test$target, Predictions_RF)

confMatrix_RF

#plotting ROc curve to RF predictions
pred=as.numeric(Predictions_RF)

plotROC(test$target, pred)

AUROC(test$target, pred)

#precision calculation
precision(test$target, pred)

pred=as.factor(Predictions_RF)

recall(test$target, pred)
```

#Accuracy:

#FNR:

R code for Santander customer transaction prediction with UP sampling and prediction with random Forest Model

```
# remove the objects stored
rm(list=ls())

#setting working directory
setwd("G:/Edwiser material/Project/Santandarcustomer problems/Edwiser project")

#checking the set directory
getwd()

# Loading required Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
    "dummies", "e1071", "Information", "MASS", "rpart", "gbm", "ROSE", 'sampling',
    'DataCombine', 'inTrees','dplyr','class','scales', "InformationValue",
    "RDocumentation", 'mlbench')

#Installing packages(x)
lapply(x, require, character.only = TRUE)
rm(x)

# Loading the both train and test data
train = read.csv("train.csv")
test = read.csv("test.csv")

# checking for the dimensions of the data
```

```
dim(train)

dim(test)

str(train)

str(test)

head(train,4)

head(test,4)


#Removing the ID_code column  which is nothing but the code and does not have information

train$ID_code=NULL

test$ID_code=NULL


#converting the target variable from numeric to factor

train$target=as.factor(train$target)


#Selecting all the numeric variables in train data

numeric_index=sapply(train, is.numeric)

numeric_data=train[,numeric_index]

cnames=colnames(numeric_data)


#Removing all the objects except the required ones

rmExcept(c("train","test","cnames"))


trainDataIndex = createDataPartition(train$target, p=0.7, list = F)

train = train[trainDataIndex, ]

test = train[-trainDataIndex, ]


# Class distribution of train data

table(train$target)


'%ni%' <- Negate('%in%')  # define 'not in' func

options(scipen=999)  # prevents printing scientific notations.
```

```
#Up Sampling
set.seed(100)
train <- upSample(x = train[, colnames(train) %ni% "target"],
                  y = train$target)
table(train$Class)

##################Model development###############

######Random Forest Model########
Model_RF=randomForest(Class~., train, importance=TRUE, ntree=50)

#Summary of the RF model
summary(Model_RF)

##Extracting rules from the trained random forest model
#transforming random forest object to intrees format
treeList=RF2List(Model_RF)

#Extracting rules
exec=extractRules(treeList, train[,-201])

#Visualizing few rules
exec[1:2,]

#Making rules more readable
readableRules=presentRules(exec, colnames(train))

readableRules[1:2,]

#Getting rule matrix
ruleMetrics=getRuleMetric(exec, train[,-201], train$Class)
```

```
#Evaluating the rules
ruleMetrics[1:2,]

#Predicting the test target variable class values with trained random forest model
Predictions_RF = predict(Model_RF, test[,-1])

#Evaluating the performance of the Random forest model
confMatrix_RF = table(test$target, Predictions_RF)

confMatrix_RF

#plotting ROC curve to RF predictions
pred=as.numeric(Predictions_RF)

plotROC(test$target, pred)

AUROC(test$target, pred)

#precision calculation
precision(test$target, pred)

pred=as.factor(Predictions_RF)

recall(test$target, pred)

#Accuracy:
#FNR:
```

Finalizing the model and Predictions on the test Data

```
##########Finalizing and Saving Model and Predicting the test cases
####################

rm(list =ls())

# Loading the both train and test data

train = read.csv("train.csv")

test = read.csv("test.csv")


#Removing the ID_code column  which is nothing but the code and does not have
information

train$ID_code=NULL

test$ID_code=NULL


#converting the target variable from numeric to factor

train$target=as.factor(train$target)


# Class distribution of train data

table(train$target)


'%ni%' <- Negate('%in%')  # define 'not in' func

options(scipen=999)  # prevents printing scientific notations.


#Up Sampling

set.seed(100)

train <- upSample(x = train[, colnames(train) %ni% "target"],

                  y = train$target)

table(train$Class)



######Random Forest Model########

final_Model_RF=randomForest(Class~., train, importance=TRUE, ntree=50)
```

```
# Saving the trained model
saveRDS(final_Model_RF, "./final_Model_RF_R.rds")

# loading the saved model
model <- readRDS("./final_Model_RF_R.rds")
print(model)
#Lets now predict on test dataset
test_predictions = predict(model, test)
#converting random forest predictions into data frame
RF_pred = data.frame("customer_transaction" = test_predictions)
#load the test data again
test=read.csv("test.csv")
#binding of random forest prediction to test data
test_RF_pred=cbind(test, RF_pred)
# writing (save) the predicted customer transaction to  disk as .csv format
write.csv(test_RF_pred,"test_RF_predictions_R.csv",row.names = FALSE)
```