# The Data Driven Approach to predict the success of Bank Telemarketing

2023-05-08

## Data information

The data is related with direct marketing campaigns of a Portuguese banking institution. The markering campaigns were based on phone calls. Often more than one contact to the same client was required.

## The Business problem I am trying to solve

- The goal of this project is to predict the customer would subscribed bank term deposit or not.
- In order to achieve this objective, first we need to explore and compare which model gives the accurate and better results.
- We will be examining parametric method such as logistic regression, as well as non parametric method such as KNN, to determine the most effective approach.

## Load the library

## load the dataset

```
Bank=read.csv("~/Desktop/SFSU/math449Project/bank.csv",header=TRUE,sep=";")
```

## To see the first five row of the data

```
head(Bank)
```

```
##   age         job marital education default balance housing loan  contact day
## 1  30  unemployed married   primary      no    1787      no   no cellular  19
## 2  33    services married secondary      no    4789     yes  yes cellular  11
## 3  35  management  single  tertiary      no    1350     yes   no cellular  16
## 4  30  management married  tertiary      no    1476     yes  yes  unknown   3
## 5  59 blue-collar married secondary      no       0     yes   no  unknown   5
## 6  35  management  single  tertiary      no     747      no   no cellular  23
##   month duration campaign pdays previous poutcome  y
## 1   oct       79        1    -1        0  unknown no
## 2   may      220        1   339        4  failure no
## 3   apr      185        1   330        1  failure no
## 4   jun      199        4    -1        0  unknown no
## 5   may      226        1    -1        0  unknown no
## 6   feb      141        2   176        3  failure no
```

# to see the data type of Bank.csv

```
str(Bank)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : int  30 33 35 30 59 35 36 39 41 43 ...
##  $ job      : chr  "unemployed" "services" "management" "management" ...
##  $ marital  : chr  "married" "married" "single" "married" ...
##  $ education: chr  "primary" "secondary" "tertiary" "tertiary" ...
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
##  $ housing  : chr  "no" "yes" "yes" "yes" ...
##  $ loan     : chr  "no" "yes" "no" "yes" ...
##  $ contact  : chr  "cellular" "cellular" "cellular" "unknown" ...
##  $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : chr  "oct" "may" "apr" "jun" ...
##  $ duration : int  79 220 185 199 226 141 341 151 57 313 ...
##  $ campaign : int  1 1 1 4 1 2 1 2 2 1 ...
##  $ pdays    : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
##  $ previous : int  0 4 1 0 0 3 2 0 0 2 ...
##  $ poutcome : chr  "unknown" "failure" "failure" "unknown" ...
##  $ y        : chr  "no" "no" "no" "no" ...
```

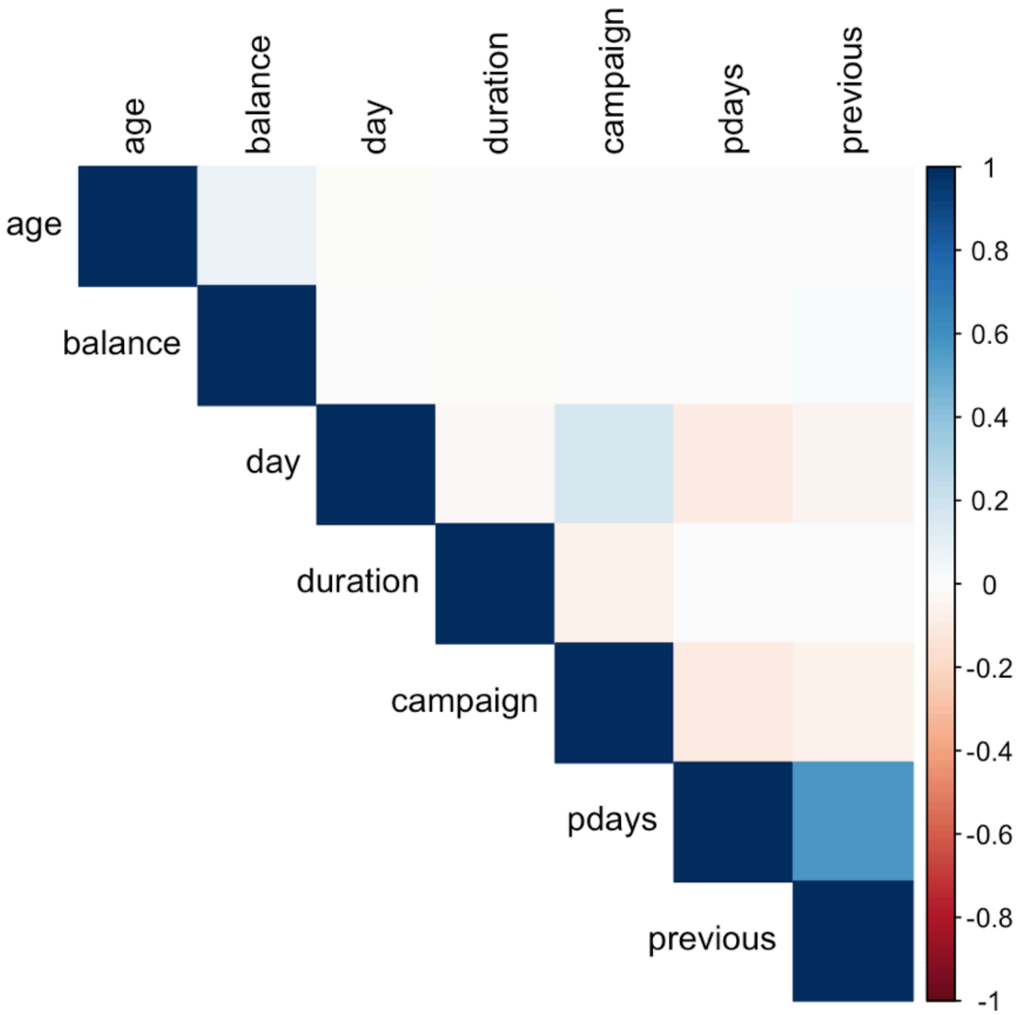
# correlatin between numeric predictors variable

```
BankNumeric=select_if(Bank,is.integer)
corMatrix=cor(BankNumeric)
corrplot(corMatrix,type="upper",method="color",tl.col="black",t1.srt=45)
```

```
## Warning in text.default(pos.xlabel[, 1], pos.xlabel[, 2], newcolnames, srt =
## tl.srt, : "t1.srt" is not a graphical parameter
```

```
## Warning in text.default(pos.ylabel[, 1], pos.ylabel[, 2], newrownames, col =
## tl.col, : "t1.srt" is not a graphical parameter
```

```
## Warning in title(title, ...): "t1.srt" is not a graphical parameter
```




The correlation coefficients between all pairs of predictor variables in the model are less than 0.5. Thus, we don’t need to worry about multicollinearity in this problem.

## count for non-numeric values

```
cols=c("job","marital","education","default","housing","contact","month","poutcome")
for (col in cols){
 counts=table(Bank[,col][!is.numeric(Bank[,col])])
 cat(paste0("Counts for ",col, " column :\n"))
 print(counts)
 cat("\n")
}
```

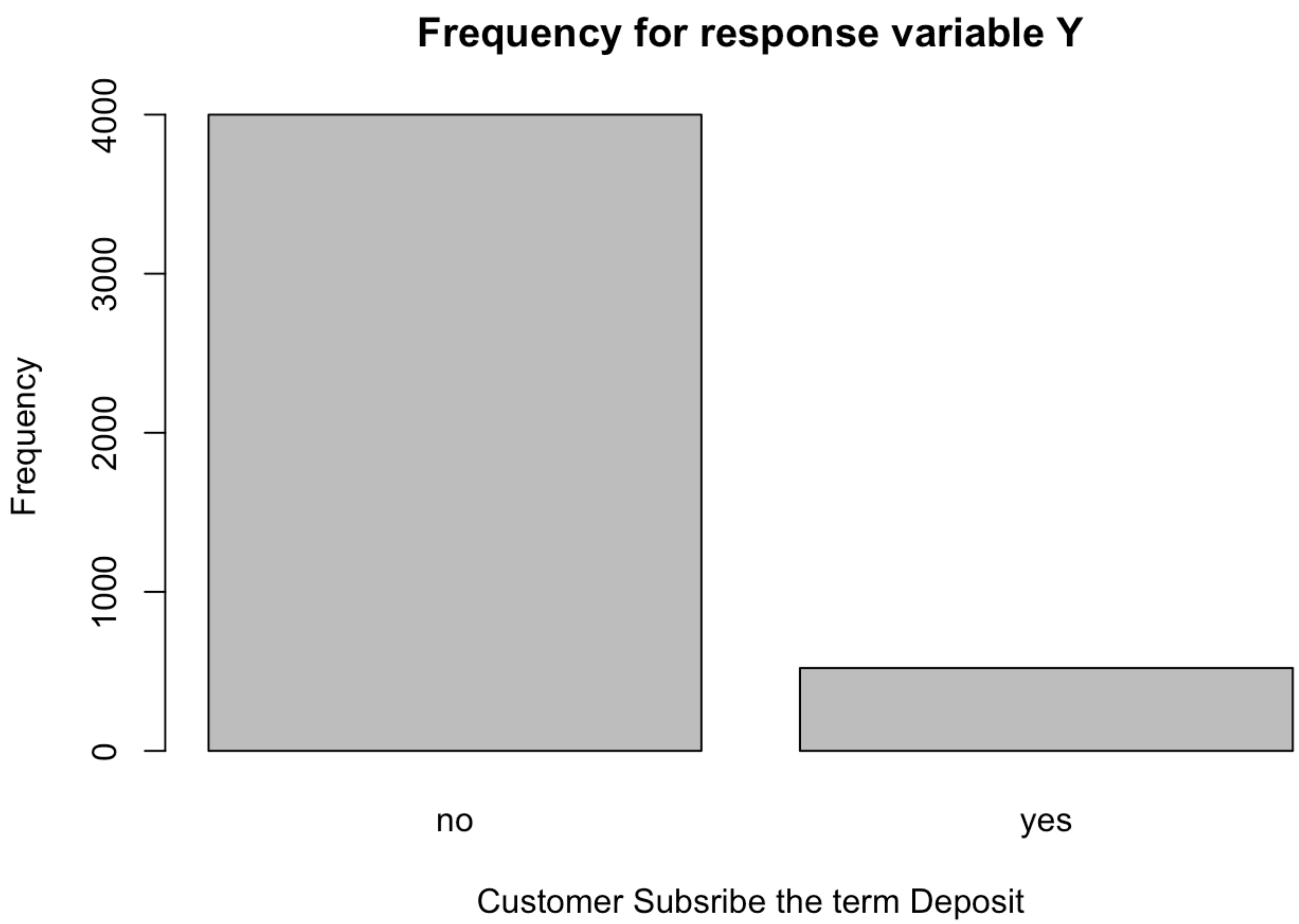
```
## Counts for job column :
##
##         admin.   blue-collar  entrepreneur     housemaid    management
##            478           946           168           112           969
##        retired self-employed      services       student    technician
##            230           183           417            84           768
##     unemployed       unknown
##            128            38
##
## Counts for marital column :
##
## divorced  married   single
##      528     2797     1196
##
## Counts for education column :
##
##   primary secondary  tertiary   unknown
##       678      2306      1350       187
##
## Counts for default column :
##
##   no  yes
## 4445   76
##
## Counts for housing column :
##
##   no  yes
## 1962 2559
##
## Counts for contact column :
##
##  cellular telephone   unknown
##      2896       301      1324
##
## Counts for month column :
##
##  apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
##  293  633   20  222  148  706  531   49 1398  389   80   52
##
## Counts for poutcome column :
##
## failure   other success unknown
##     490     197     129    3705
```

# Graph the non numeric value counts for each column

```
dev.new()
for (col in 1:length(cols)){
 counts=table(Bank[,cols[col]][!is.numeric(Bank[,cols[col]])])
 barplot(counts,main=cols[col],xlab="Non Numeric values", ylab="Count")
}
```

# value counts for response variable

```
countsY=table(Bank$y)
barplot(countsY,main="Frequency for response variable Y",xlab="Customer Subsribe the
term Deposit",ylab="Frequency")
```




### compute the percentage of yes and no

```
percentageYes=countsY[2] / nrow(Bank) * 100
percentageNo=countsY[1] / nrow(Bank) * 100

cat(paste0("Percentage of subscription of term deposit: ",format(percentageYes,nsmall
=2), " %"))
```

```
## Percentage of subscription of term deposit: 11.524 %
```

```
cat("\n")
```

```
cat(paste0("Percentage of no subscription of term deposit: ",format(percentageNo,nsma
ll=2), " %"))
```

```
## Percentage of no subscription of term deposit: 88.476 %
```

Based on the graph and compute percentage, the response variable is unbalanced, with more “No” response than “Yes” which could cause biased in our prediction. This means it may accurately predict the majority class which is “No”, but fail to accurately predict the minority class.

# I will try to make it balance by using the oversampling method

```
Bank=ovun.sample(y~.,data=Bank,method="both",N=nrow(Bank),seed=123)$data
table(Bank$y)
```

```
##
##   no  yes
## 2279 2242
```

Now, the response variable is balanced. . ### convert the response variable into binary 0 means no and 1 means yes

```
Bank$y=ifelse(Bank$y == "yes",1,0)
table(Bank$y)
```

```
##
##    0    1
## 2279 2242
```

Now, the response variable is balanced.

# convert the non numeric vairables into factor

```
Bank$job=as.factor(Bank$job)
Bank$marital=as.factor(Bank$marital)
Bank$education=as.factor(Bank$education)
Bank$default=as.factor(Bank$default)
Bank$housing=as.factor(Bank$housing)
Bank$loan=as.factor(Bank$loan)
Bank$contact=as.factor(Bank$contact)
Bank$month=as.factor(Bank$month)
Bank$poutcome=as.factor(Bank$poutcome)
Bank$y=as.factor(Bank$y)
```

# summary of the original data

```
summary(Bank)
```

```
##       age                      job          marital         education    default
##  Min.   :19.00   management :1026   divorced: 591   primary  : 662   no :4432
##  1st Qu.:33.00   blue-collar: 815   married :2647   secondary:2236   yes:  89
##  Median :40.00   technician : 718   single  :1283   tertiary :1443
##  Mean   :41.91   admin.     : 465                   unknown  : 180
##  3rd Qu.:50.00   services   : 391
##  Max.   :87.00   retired    : 378
##                  (Other)    : 728
##     balance      housing     loan           contact          day
##  Min.   :-2082   no :2241   no :3956   cellular :3195   Min.   : 1.00
##  1st Qu.:  101   yes:2280   yes: 565   telephone: 371   1st Qu.: 9.00
##  Median :  569                         unknown  : 955   Median :16.00
##  Mean   : 1528                                          Mean   :15.73
##  3rd Qu.: 1811                                          3rd Qu.:21.00
##  Max.   :71188                                          Max.   :31.00
##
##      month          duration        campaign          pdays
##  may    :1117   Min.   :   4   Min.   : 1.000   Min.   : -1.00
##  aug    : 676   1st Qu.: 151   1st Qu.: 1.000   1st Qu.: -1.00
##  jul    : 627   Median : 273   Median : 2.000   Median : -1.00
##  jun    : 487   Mean   : 396   Mean   : 2.572   Mean   : 52.79
##  apr    : 384   3rd Qu.: 543   3rd Qu.: 3.000   3rd Qu.: 56.00
##  nov    : 371   Max.   :2769   Max.   :32.000   Max.   :871.00
##  (Other): 859
##     previous          poutcome    y
##  Min.   : 0.0000   failure: 534   0:2279
##  1st Qu.: 0.0000   other  : 251   1:2242
##  Median : 0.0000   success: 362
##  Mean   : 0.7554   unknown:3374
##  3rd Qu.: 1.0000
##  Max.   :20.0000
##
```

First, let start fitting the full model.

# Fit the full model with all the predictors without cross validation, splitting the data and feature selection

```
fullBank=glm(y~.,data=Bank,family="binomial")
summary(fullBank)
```

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = Bank)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2864  -0.5810  -0.1039   0.5901   2.2042
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.212e+00  4.548e-01  -2.664 0.007712 **
## age                 6.587e-03  5.287e-03   1.246 0.212835
## jobblue-collar     -7.218e-01  1.721e-01  -4.193 2.75e-05 ***
## jobentrepreneur    -2.708e-01  2.744e-01  -0.987 0.323705
## jobhousemaid       -4.626e-01  3.107e-01  -1.489 0.136525
## jobmanagement      -3.298e-01  1.797e-01  -1.835 0.066447 .
## jobretired          1.488e-01  2.259e-01   0.658 0.510236
## jobself-employed   -5.089e-01  2.700e-01  -1.885 0.059425 .
## jobservices        -7.598e-01  2.038e-01  -3.728 0.000193 ***
## jobstudent          5.860e-01  2.952e-01   1.985 0.047116 *
## jobtechnician      -5.424e-01  1.678e-01  -3.233 0.001226 **
## jobunemployed      -8.178e-01  2.973e-01  -2.751 0.005941 **
## jobunknown         -1.041e-01  4.514e-01  -0.231 0.817570
## maritalmarried     -2.071e-01  1.337e-01  -1.549 0.121328
## maritalsingle       6.200e-02  1.583e-01   0.392 0.695211
## educationsecondary  2.306e-01  1.432e-01   1.610 0.107475
## educationtertiary   3.541e-01  1.672e-01   2.118 0.034139 *
## educationunknown   -4.775e-01  2.602e-01  -1.835 0.066435 .
## defaultyes          2.775e-01  3.039e-01   0.913 0.361287
## balance            -9.387e-06  1.464e-05  -0.641 0.521400
## housingyes         -2.609e-01  9.849e-02  -2.649 0.008081 **
## loanyes            -1.005e+00  1.420e-01  -7.078 1.46e-12 ***
## contacttelephone   -3.189e-01  1.689e-01  -1.888 0.059089 .
## contactunknown     -1.171e+00  1.436e-01  -8.156 3.47e-16 ***
## day                 1.246e-02  5.886e-03   2.117 0.034274 *
## monthaug           -5.373e-01  1.742e-01  -3.084 0.002044 **
## monthdec           -2.358e-01  5.969e-01  -0.395 0.692865
## monthfeb            3.898e-01  2.088e-01   1.867 0.061850 .
## monthjan           -1.386e+00  2.797e-01  -4.955 7.22e-07 ***
## monthjul           -9.113e-01  1.847e-01  -4.935 8.01e-07 ***
## monthjun            2.458e-01  2.155e-01   1.140 0.254095
## monthmar            1.611e+00  3.109e-01   5.183 2.19e-07 ***
## monthmay           -9.546e-01  1.726e-01  -5.530 3.21e-08 ***
## monthnov           -9.282e-01  1.986e-01  -4.675 2.94e-06 ***
## monthoct            1.889e+00  2.856e-01   6.615 3.72e-11 ***
## monthsep            7.571e-01  3.741e-01   2.024 0.043019 *
## duration            5.927e-03  2.053e-04  28.875  < 2e-16 ***
## campaign           -1.394e-01  2.262e-02  -6.162 7.20e-10 ***
## pdays               1.625e-03  6.796e-04   2.391 0.016798 *
## previous           -7.190e-02  3.768e-02  -1.908 0.056370 .
```

```
## poutcomeother       8.455e-01  2.065e-01   4.095 4.21e-05 ***
## poutcomesuccess     3.245e+00  3.185e-01  10.187  < 2e-16 ***
## poutcomeunknown    -2.179e-01  2.457e-01  -0.887 0.375051
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6267.1  on 4520  degrees of freedom
## Residual deviance: 3593.2  on 4478  degrees of freedom
## AIC: 3679.2
##
## Number of Fisher Scoring iterations: 6
```

Using the full model, many predictor variables are significant to predict whether or not customer will subscribe the term deposit. This might lead to overfitting since we are using all the features to fit the model.

Just to make sure all the variables are really significant for prediction, I will next use feature selection to identify the most important predictors that contribute to a given outcome variable).

By selecting only the most relevant predictors, we can improve the accuracy and interpretability of predictive models.

In this analysis, I will use forward and backward elimination for feature selection

# Backward elimination

```
library(caret)
backwardsModel=step(
  object=fullBank,
  direction = "backward",
  scope=y~.,
  trace=0
)
selectedFeatures=names(coef(backwardsModel))[-1]
print(selectedFeatures)
```

```
##  [1] "jobblue-collar"     "jobentrepreneur"    "jobhousemaid"
##  [4] "jobmanagement"      "jobretired"         "jobself-employed"
##  [7] "jobservices"        "jobstudent"         "jobtechnician"
## [10] "jobunemployed"      "jobunknown"         "maritalmarried"
## [13] "maritalsingle"      "educationsecondary" "educationtertiary"
## [16] "educationunknown"   "housingyes"         "loanyes"
## [19] "contacttelephone"   "contactunknown"     "day"
## [22] "monthaug"           "monthdec"           "monthfeb"
## [25] "monthjan"           "monthjul"           "monthjun"
## [28] "monthmar"           "monthmay"           "monthnov"
## [31] "monthoct"           "monthsep"           "duration"
## [34] "campaign"           "pdays"              "previous"
## [37] "poutcomeother"      "poutcomesuccess"    "poutcomeunknown"
```

From the features selection using backward elimination, I found out that variables “job”,“marital”,“education”, “housing”,“loan”,“contact”,“day”,“month”,“duration”,“campaign”,“pdays”,“previous”,“poutcome” are selected features.

# forward selection

```
forwardModel=step(
  #fullBank is a original model fit
  object=fullBank,
  direction = "forward",
  scope=y~.,
  trace=0
)
selectedFeatures=names(coef(backwardsModel))[-1]
print(selectedFeatures)
```

```
##  [1] "jobblue-collar"     "jobentrepreneur"    "jobhousemaid"
##  [4] "jobmanagement"      "jobretired"         "jobself-employed"
##  [7] "jobservices"        "jobstudent"         "jobtechnician"
## [10] "jobunemployed"      "jobunknown"         "maritalmarried"
## [13] "maritalsingle"      "educationsecondary" "educationtertiary"
## [16] "educationunknown"   "housingyes"         "loanyes"
## [19] "contacttelephone"   "contactunknown"     "day"
## [22] "monthaug"           "monthdec"           "monthfeb"
## [25] "monthjan"           "monthjul"           "monthjun"
## [28] "monthmar"           "monthmay"           "monthnov"
## [31] "monthoct"           "monthsep"           "duration"
## [34] "campaign"           "pdays"              "previous"
## [37] "poutcomeother"      "poutcomesuccess"    "poutcomeunknown"
```

- Backward elimination and forward selection methods choose 13 predictor variables out of 16 from the data.
- From the features selection using forward method, I found out that variables “job”,“marital”,“education”, “housing”,“loan”,“contact”,“day”,“month”,“duration”,“campaign”,“pdays”,“previous”,“poutcome” are selected features.

## create a new data from the selected features(it comes from forward selection)

```
selectedFeaturesCol=c("job","marital","education","housing","loan","contact","day","m
onth","duration","campaign","pdays","previous","poutcome")
selected_features <- c(selectedFeaturesCol, "y")
print(selected_features)
```

```
##  [1] "job"       "marital"   "education" "housing"   "loan"      "contact"  
##  [7] "day"       "month"     "duration"  "campaign"  "pdays"     "previous" 
## [13] "poutcome"  "y"
```

```
newBank <- Bank%>%
  dplyr::select(one_of(selected_features))
```

## Creating training and testing data with selected features

```
train = sample(dim(newBank)[1], dim(newBank)[1]*0.8)
test=-train
newBank.test=newBank[test,]
newBank.train=newBank[train,]
```

# Fit the logistic regression model with selected features using cross-validation.

```
library(caret)
fitControl2 <- trainControl(method = "cv", number = 10)
#fit the modle
newBankFit <- train(y ~ ., data = newBank.train , method = "glm", trControl = fitCont
rol2,family=binomial)
#predicitons
predictions2=predict(newBankFit,newdata = newBank.test,type="prob")
binaryPreds2=ifelse(predictions2[,2]>0.5,1,0)
binaryPredsfactor2=factor(binaryPreds2,levels=c(0,1),ordered=TRUE)
```

# confusion matrix with selected features from forward selection

```
confusion2=confusionMatrix(binaryPredsfactor2,newBank.test$y)
confusion2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 377  78
##          1  78 372
##
##                Accuracy : 0.8276
##                  95% CI : (0.8014, 0.8517)
##     No Information Rate : 0.5028
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6552
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8286
##             Specificity : 0.8267
##          Pos Pred Value : 0.8286
##          Neg Pred Value : 0.8267
##              Prevalence : 0.5028
##          Detection Rate : 0.4166
##    Detection Prevalence : 0.5028
##       Balanced Accuracy : 0.8276
##
##        'Positive' Class : 0
##
```

# ROC curve with selected features from forward selection

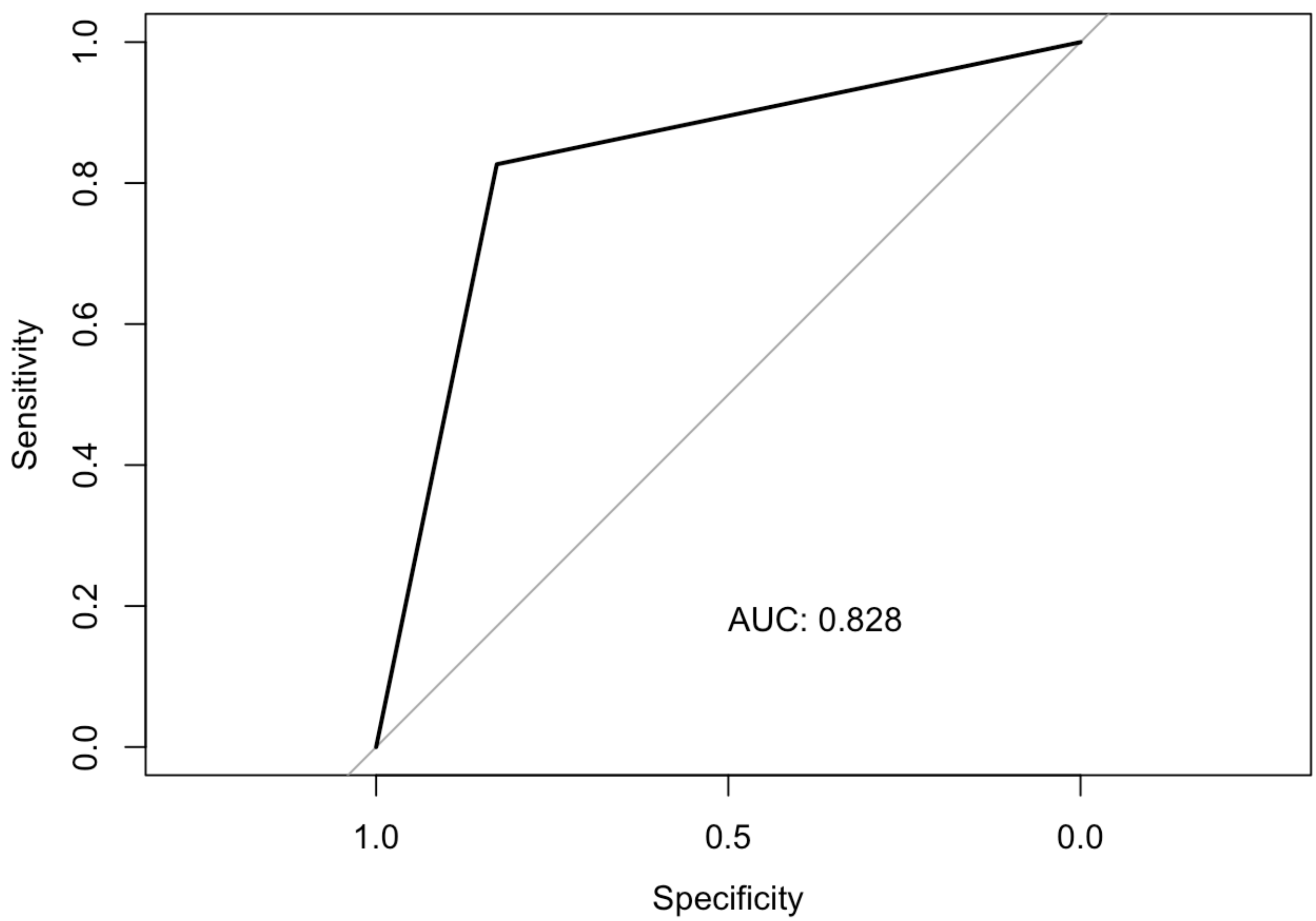
note: roc() function expects predicted class probabilities, not class labels

```
rocobj1 <- roc(newBank.test$y, binaryPredsfactor2)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(rocobj1,print.auc=TRUE,print.auc.x=0.5,print.auc.y=0.2)
```




```
aucScore1=auc(rocobj1)
print(aucScore1)
```

```
## Area under the curve: 0.8276
```

AUC=0.828. It is greater than 0.5. It means the model perform better than random guessing.

# calculate precision and accuracy

```
truePositive2=confusion2$table[2,2]
falsePositive2=confusion2$table[1,2]
precision2=truePositive2/(truePositive2+falsePositive2)
accuracy2=confusion2$overall["Accuracy"]
cat("Precision measures how often the model correctly predicts that customers will su
bscribe the term deposit.")
```

```
## Precision measures how often the model correctly predicts that customers will subs
cribe the term deposit.
```

```
cat("\n")
```

```
cat("\n")
```

```
cat("The precision score using cross validation with the selected feature is"
,round(precision2*100,2), "%.")
```

```
## The precision score using cross validation with the selected feature is 82.67 %.
```

```
cat("\n")
```

```
cat("\n")
```

```
cat("Accuracy measures how often the model correctly predicts, regardless of it is ab
out predicting no subscirbe or subscribe the term deposit.")
```

```
## Accuracy measures how often the model correctly predicts, regardless of it is abou
t predicting no subscirbe or subscribe the term deposit.
```

```
cat("\n")
```

```
cat("The accuracy score using cross validation with the selected feature is"
,round(accuracy2*100,2), "%")
```

```
## The accuracy score using cross validation with the selected feature is 82.76 %
```

# Next fit the model using the random Forest

Advantages: More stable and robust than decision trees, can handle both continuous and categorical features, handles irrelevant features well, can capture non-linear relationships between the input features and the output, can handle large datasets.

Disadvantages: Can be slow and memory-intensive, may not perform well on imbalanced datasets, may not work well with high-dimensional data.

## fit the model using random Forest on selected features with cross validation

```
random.fit=train(y~.,data=newBank.train,method="rf",trControl=fitControl2)
predictions3=predict(random.fit,newdata = newBank.test,type="prob")
binaryPreds3=ifelse(predictions3[,2]>0.5,1,0)
binaryPredsfactor3=factor(binaryPreds3,levels=c(0,1),ordered=TRUE)
```

## confusion matrix with selected features from forward selection

```
confusion3=confusionMatrix(binaryPredsfactor3,newBank.test$y)
confusion3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 431   3
##          1  24 447
##
##                Accuracy : 0.9702
##                  95% CI : (0.9569, 0.9802)
##     No Information Rate : 0.5028
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9403
##
##  Mcnemar's Test P-Value : 0.0001186
##
##             Sensitivity : 0.9473
##             Specificity : 0.9933
##          Pos Pred Value : 0.9931
##          Neg Pred Value : 0.9490
##              Prevalence : 0.5028
##          Detection Rate : 0.4762
##    Detection Prevalence : 0.4796
##       Balanced Accuracy : 0.9703
##
##        'Positive' Class : 0
##
```

# ROC curve with selected features from forward selection

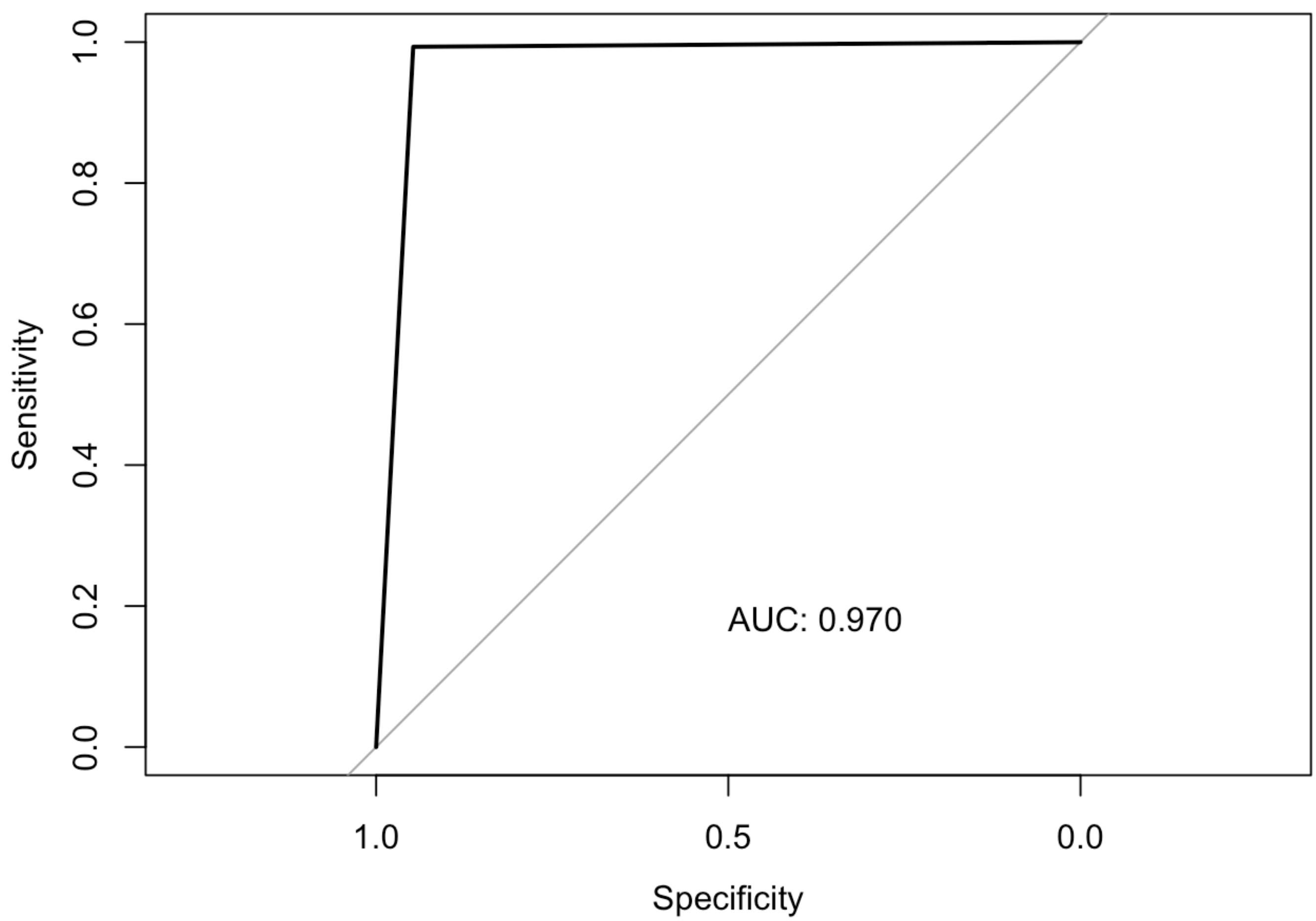
note: roc() function expects predicted class probabilities, not class labels

```
rocobj2 <- roc(newBank.test$y, binaryPredsfactor3)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(rocobj2,print.auc=TRUE,print.auc.x=0.5,print.auc.y=0.2)
```




```
aucScore2=auc(rocobj2)
print(aucScore2)
```

```
## Area under the curve: 0.9703
```

AUC=0.967. It is greater than 0.5. It means the model perform better than random guessing. ### calculate precision and accuracy

```
truePositive3=confusion3$table[2,2]
falsePositive3=confusion3$table[1,2]
precision3=truePositive3/(truePositive3+falsePositive3)
accuracy3=confusion3$overall["Accuracy"]
cat("Precision measures how often the model correctly predicts that customers will su
bscribe the term deposit.")
```

```
## Precision measures how often the model correctly predicts that customers will subs
cribe the term deposit.
```

```
cat("\n")
```

```
cat("The precision score using cross validation with the selected feature is"
,round(precision3*100,2), "%.")
```

```
## The precision score using cross validation with the selected feature is 99.33 %.
```

```
cat("\n")
```

```
cat("\n")
```

```
cat("Accuracy measures how often the model correctly predicts, regardless of it is ab
out predicting no subscirbe or subscribe the term deposit.")
```

```
## Accuracy measures how often the model correctly predicts, regardless of it is abou
t predicting no subscirbe or subscribe the term deposit.
```

```
cat("\n")
```

```
cat("The accuracy score using cross validation with the selected feature is"
,round(accuracy3*100,2), "%")
```

```
## The accuracy score using cross validation with the selected feature is 97.02 %
```

Random Forest significantly increase the accuracy and auc in the new bank data. However, it is computationally expensive.

# Fit the model using Decision trees. It can handle both classification and regression tasks.

some advantages and disadvatages of using Decision trees.

Advantages: It can handle nonlinear relationships and mixed feature types.

Disadvantages: It is prone to overfitting. It is sensitivity to small variations.

# Decision trees

```
treeModel=train(y ~ ., data = newBank.train , method = "rpart", trControl = fitContro
l2)
predictions4=predict(treeModel,newdata = newBank.test)
```

# confusion matrix with selected features from forward selection

```
confusion4=confusionMatrix(data=predictions4,newBank.test$y)
confusion4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 329  72
##          1 126 378
##
##                Accuracy : 0.7812
##                  95% CI : (0.7528, 0.8078)
##     No Information Rate : 0.5028
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5627
##
##  Mcnemar's Test P-Value : 0.0001655
##
##             Sensitivity : 0.7231
##             Specificity : 0.8400
##          Pos Pred Value : 0.8204
##          Neg Pred Value : 0.7500
##              Prevalence : 0.5028
##          Detection Rate : 0.3635
##    Detection Prevalence : 0.4431
##       Balanced Accuracy : 0.7815
##
##        'Positive' Class : 0
##
```

# ROC curve with selected features from forward selection

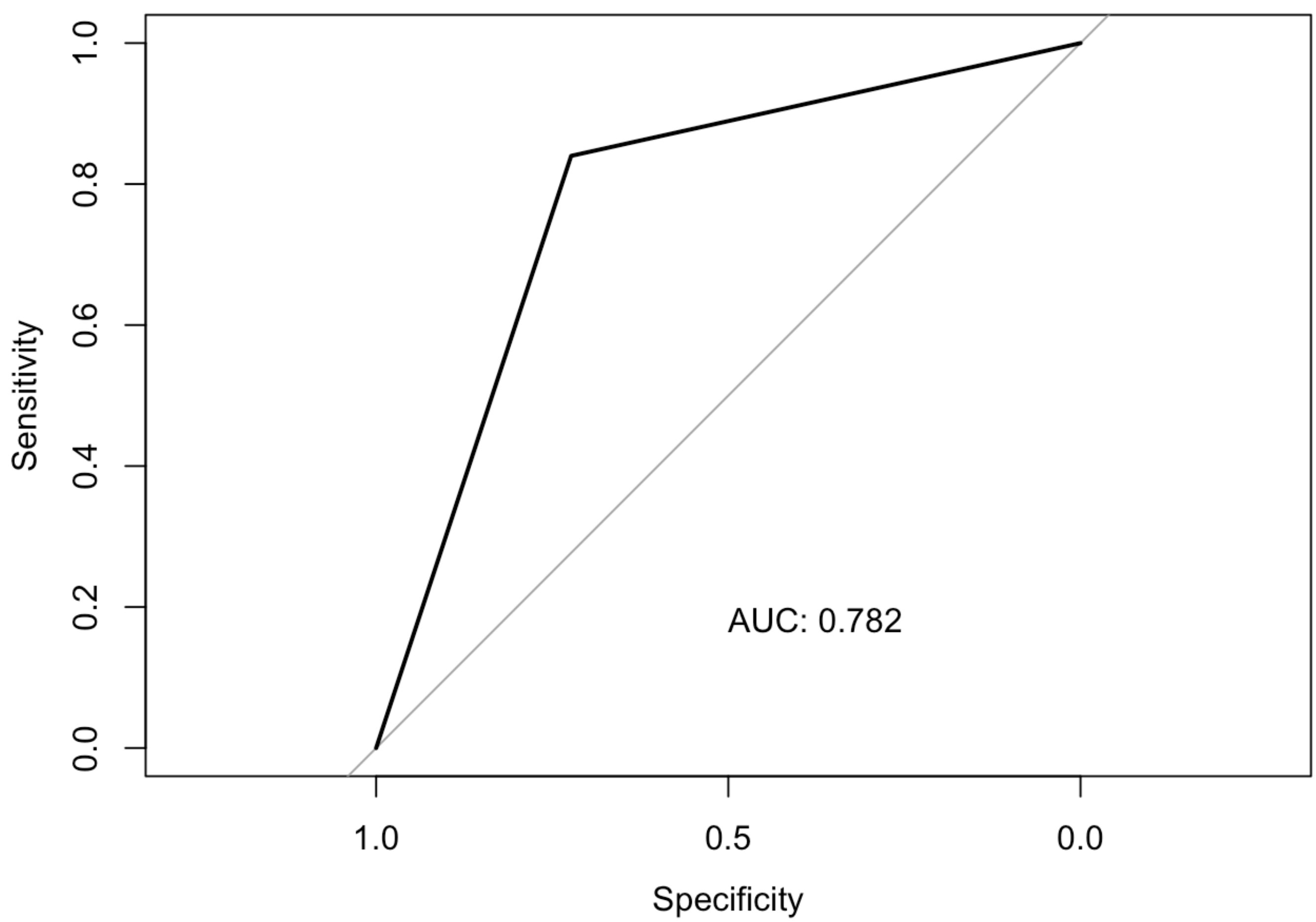
note: roc() function expects predicted class probabilities, not class labels

```
response=as.numeric(newBank.test$y)-1
predictor=as.numeric(predictions4)-1
rocobj3 <- roc(response, predictor)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(rocobj3,print.auc=TRUE,print.auc.x=0.5,print.auc.y=0.2)
```




```
aucScore3=auc(rocobj3)
print(aucScore3)
```

```
## Area under the curve: 0.7815
```

## calculate precision and accuracy

```
truePositive4=confusion4$table[2,2]
falsePositive4=confusion4$table[1,2]
precision4=truePositive4/(truePositive4+falsePositive4)
accuracy4=confusion4$overall["Accuracy"]
cat("Precision measures how often the model correctly predicts that customers will su
bscribe the term deposit.")
```

```
## Precision measures how often the model correctly predicts that customers will subs
cribe the term deposit.
```

```
cat("\n")
```

```
cat("The precision score using cross validation with the selected feature is"
,round(precision4*100,2), "%.")
```

```
## The precision score using cross validation with the selected feature is 84 %.
```

```
cat("\n")
```

```
cat("\n")
```

```
cat("Accuracy measures how often the model correctly predicts, regardless of it is ab
out predicting no subscirbe or subscribe the term deposit.")
```

```
## Accuracy measures how often the model correctly predicts, regardless of it is abou
t predicting no subscirbe or subscribe the term deposit.
```

```
cat("\n")
```

```
cat("The accuracy score using cross validation with the selected feature is"
,round(accuracy4*100,2), "%")
```

```
## The accuracy score using cross validation with the selected feature is 78.12 %
```

# Next, fit the model using Naive Bayes.

The only problem of using Naive Bayes is that it is assume that features are conditionally independent.

```
naiveBayesFit=train(y ~ ., data = newBank.train , method = "naive_bayes", trControl =
fitControl2)
predictions5=predict(naiveBayesFit,newdata = newBank.test)
```

# confusion matrix with selected features from forward selection

```
confusion5=confusionMatrix(data=predictions5,newBank.test$y)
confusion5
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 392 220
##          1  63 230
##
##                Accuracy : 0.6873
##                  95% CI : (0.656, 0.7174)
##     No Information Rate : 0.5028
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3734
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8615
##             Specificity : 0.5111
##          Pos Pred Value : 0.6405
##          Neg Pred Value : 0.7850
##              Prevalence : 0.5028
##          Detection Rate : 0.4331
##    Detection Prevalence : 0.6762
##       Balanced Accuracy : 0.6863
##
##        'Positive' Class : 0
##
```

# ROC curve with selected features from forward selection

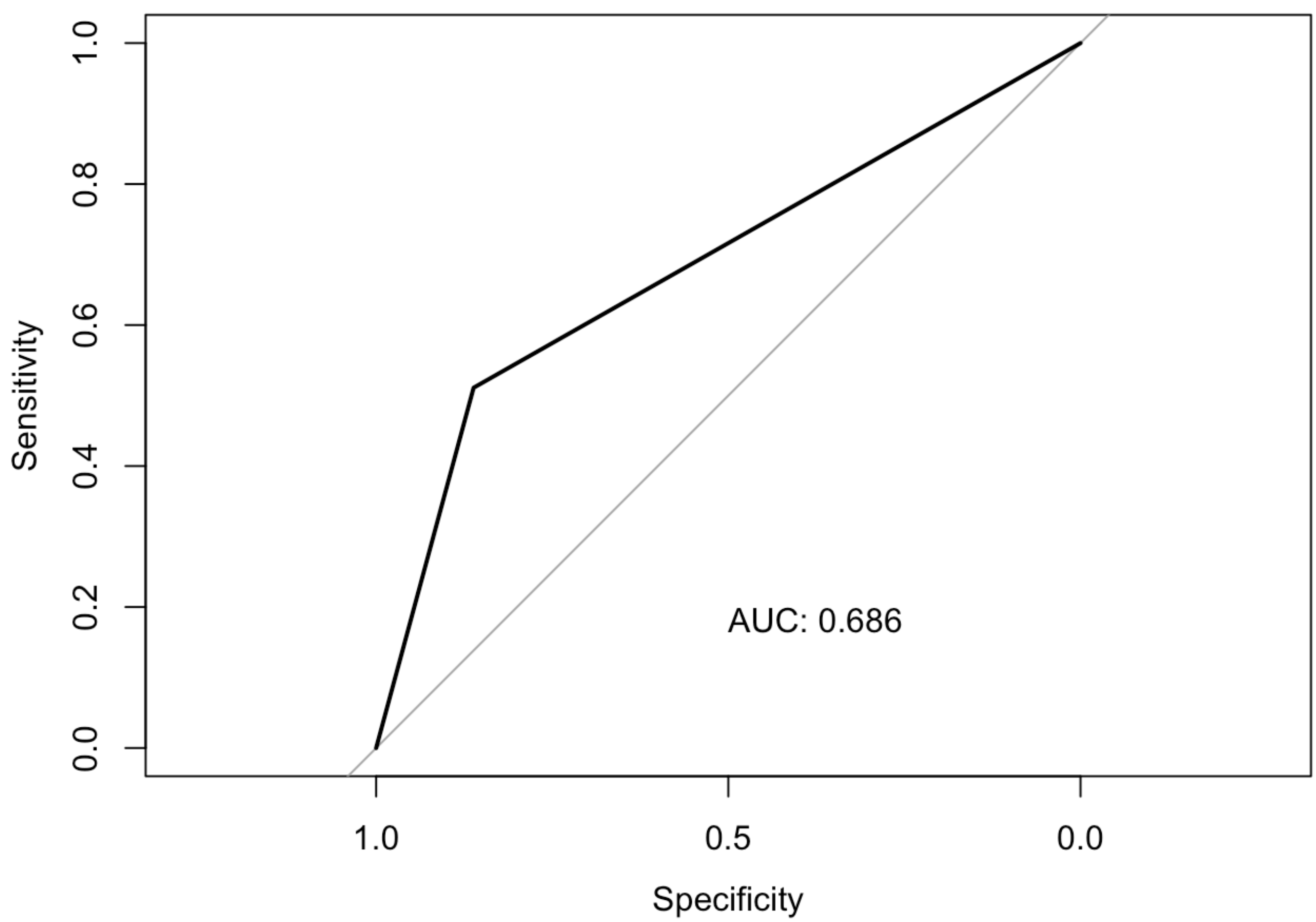
note: roc() function expects predicted class probabilities, not class labels

```
response2=as.numeric(newBank.test$y)-1
predictor2=as.numeric(predictions5)-1
rocobj4 <- roc(response2, predictor2)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(rocobj4,print.auc=TRUE,print.auc.x=0.5,print.auc.y=0.2)
```




```
aucScore4=auc(rocobj4)
print(aucScore4)
```

```
## Area under the curve: 0.6863
```

# calculate precision and accuracy

```
truePositive5=confusion5$table[2,2]
falsePositive5=confusion5$table[1,2]
precision5=truePositive5/(truePositive5+falsePositive5)
accuracy5=confusion5$overall["Accuracy"]
cat("Precision measures how often the model correctly predicts that customers will su
bscribe the term deposit.")
```

```
## Precision measures how often the model correctly predicts that customers will subs
cribe the term deposit.
```

```
cat("\n")
```

```
cat("The precision score using cross validation with the selected feature is"
,round(precision5*100,2), "%.")
```

```
## The precision score using cross validation with the selected feature is 51.11 %.
```

```
cat("\n")
```

```
cat("\n")
```

```
cat("Accuracy measures how often the model correctly predicts, regardless of it is ab
out predicting no subscirbe or subscribe the term deposit.")
```

```
## Accuracy measures how often the model correctly predicts, regardless of it is abou
t predicting no subscirbe or subscribe the term deposit.
```

```
cat("\n")
```

```
cat("The accuracy score using cross validation with the selected feature is"
,round(accuracy5*100,2), "%")
```

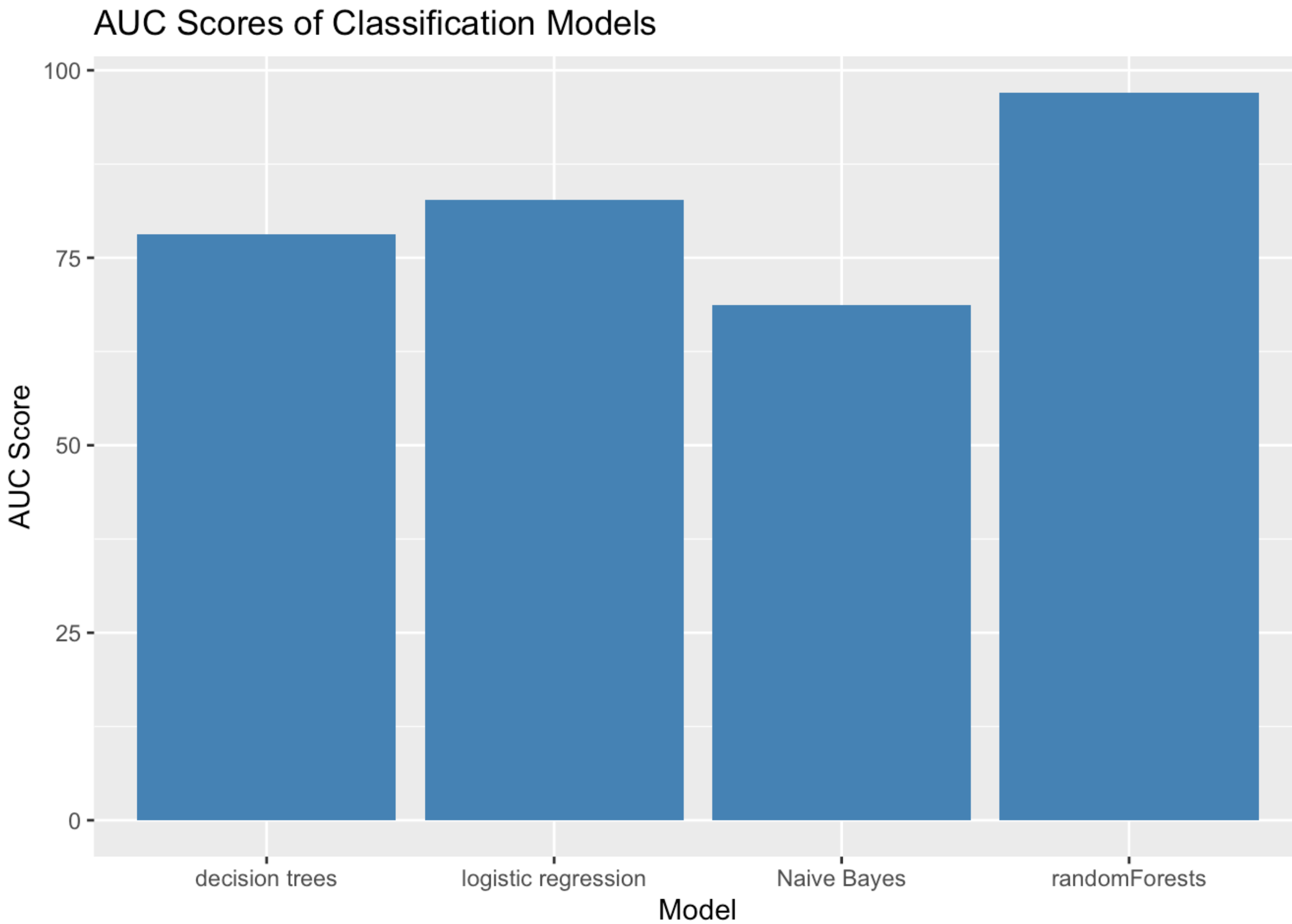
```
## The accuracy score using cross validation with the selected feature is 68.73 %
```

# Create a data frame containing the model names and corresponding AUC, accuracy, and precision scores.

```
modelScores=data.frame(Model=c("logistic regression", "randomForests", "decision tree
s","Naive Bayes"),
                       Precision=c(round(precision2*100,2),round(precision3*100,2),ro
und(precision4*100 ,2),round(precision5*100,2)),
                       Accuracy=c(round(accuracy2 *100,2),round(accuracy3 *100,2),rou
nd(accuracy4*100,2),round(accuracy5*100,2)),
                   AUC=c(round(aucScore1 *100 ,2),round(aucScore2 * 100,2),round(aucS
core3 * 100 ,2),round(aucScore4*100,2)))
```

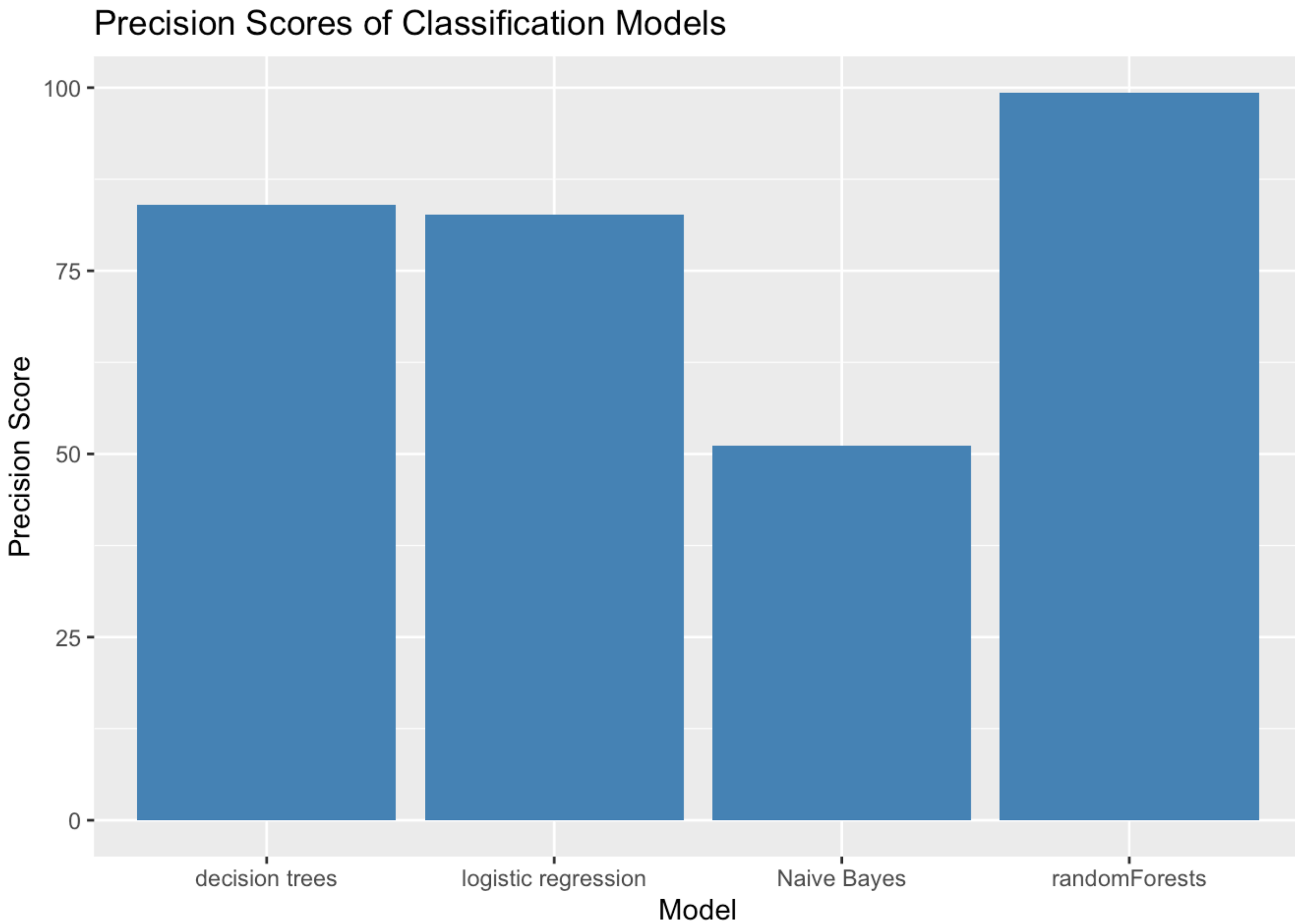
# Create a bar graph showing the AUC scores of logistic regression, random Forest, and decision trees

```
ggplot(modelScores,aes(x=Model,y=AUC))+
  geom_bar(stat="identity",fill="steelblue")+
  labs(title = "AUC Scores of Classification Models",
       x= "Model",y="AUC Score")
```



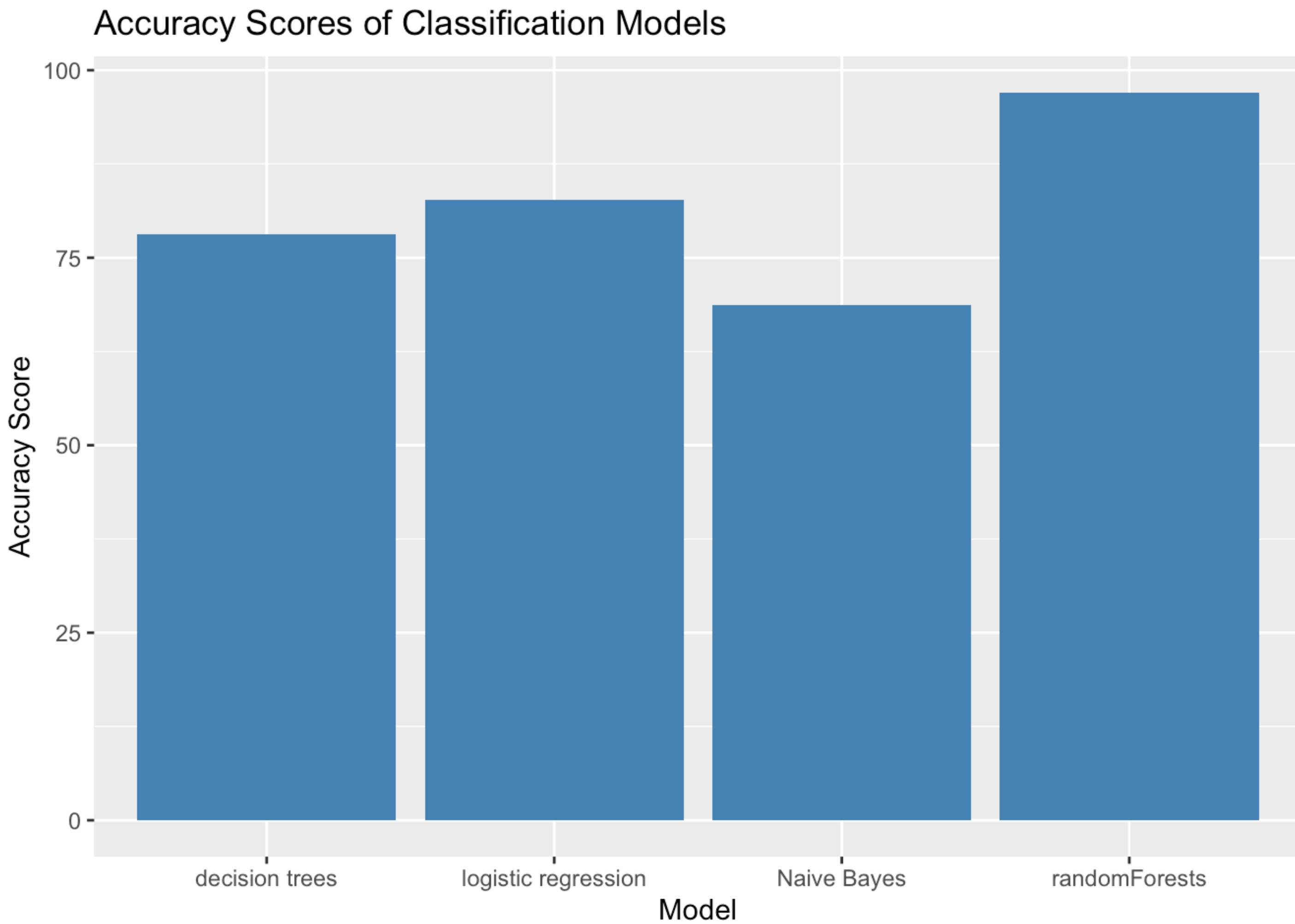

# Create a bar graph showing the Precision scores of logistic regression, random Forest, and decision trees

```
ggplot(modelScores,aes(x=Model,y=Precision))+
  geom_bar(stat="identity",fill="steelblue")+
  labs(title = "Precision Scores of Classification Models",
       x= "Model",y="Precision Score")
```




## Create a bar graph showing the Accuracy scores of logistic regression, random Forest, and decision trees

```
ggplot(modelScores,aes(x=Model,y=Accuracy))+
  geom_bar(stat="identity",fill="steelblue")+
  labs(title = "Accuracy Scores of Classification Models",
       x= "Model",y="Accuracy Score")
```


Accuracy Scores of Classification Models
100
75
50
25
0
Accuracy Score
decision trees
logistic regression
Naive Bayes
randomForests
Model