# finance

*Mingjie Ye*

# Introduction

In this project, I will build binary classification models that classify the clients according to the credit quality (ability to repay a credit loan in according to the contractual terms). We will build Logistic Regression,Lasso (least absolute shrinkage and selection operator) Regression,Decision Tree,Random Forest and LDA (Linear discriminant analysis) models and compare them accordinig to the cost and accuracy.

# Setup and review the dataset

Let's check the dataset if there is NA value.

```
colSums(is.na(data))
```

```
##                     KEY                  target       NON_CURRENT_ASSETS
##                       0                       0                        0
##    tangible_fixed_assets          CURRENT_ASSETS cash_other_liquid_assets
##                       0                       0                        0
##                  STOCKS debtors_other_short_term             TOTAL_ASSETS
##                       0                       0                        0
##               net_worth  NON_CURRENT_LIABILITIES      CURRENT_LIABILITIES
##                       0                       0                        0
##               SUPPLIERS        TOTAL_LIABILITIES OTHER_OPERATION_EXPENSES
##                       0                       0                        0
##         ORDINARY_EBITDA      Non_Ordinary_Result     NET_OPERATION_RESULT
##                       0                       0                        0
##       FINANCIAL_INCOMES       FINANCIAL_EXPENSES        FINANCIAL_RESULTS
##                       0                       0                        0
##    BENEFITS_BEFORE_TAXES            COMPANIES_TAX        BANK_INDEBTEDNESS
##                       0                       0                        0
##     GROSS_FINANCIAL_DEBT               Net_result       CLIENT_COLLECTING
##                       0                       0                        0
##      REVENUES_VARIATION          Exercise_result                   Income
##                       0                       0                        0
##  other_operation_incomes       personnel_expenses  depreciation_provision
##                       0                       0                        0
##            ROLLING_FUND
##                       0
```

Change the target to factor and check the class of each attributes. I found the classes of some columns are character, but I hope they are numeric so that we can do some analysis. So I change them to numeric.

```
data$target<-as.factor(data$target)
sapply(data, class)
```

```
##                       KEY                    target        NON_CURRENT_ASSETS
##                 "integer"                  "factor"               "character"
##     tangible_fixed_assets            CURRENT_ASSETS cash_other_liquid_assets
##               "character"               "character"               "character"
##                    STOCKS debtors_other_short_term              TOTAL_ASSETS
##               "character"               "character"               "character"
##                 net_worth  NON_CURRENT_LIABILITIES       CURRENT_LIABILITIES
##               "character"               "character"               "character"
##                 SUPPLIERS         TOTAL_LIABILITIES OTHER_OPERATION_EXPENSES
##               "character"               "character"               "character"
##           ORDINARY_EBITDA      Non_Ordinary_Result      NET_OPERATION_RESULT
##               "character"               "character"               "character"
##         FINANCIAL_INCOMES        FINANCIAL_EXPENSES         FINANCIAL_RESULTS
##               "character"               "character"               "character"
##     BENEFITS_BEFORE_TAXES             COMPANIES_TAX         BANK_INDEBTEDNESS
##               "character"               "character"               "character"
##      GROSS_FINANCIAL_DEBT                Net_result         CLIENT_COLLECTING
##               "character"               "character"               "character"
##        REVENUES_VARIATION           Exercise_result                    Income
##               "character"               "character"               "character"
##   other_operation_incomes        personnel_expenses    depreciation_provision
##               "character"               "character"               "character"
##              ROLLING_FUND
##               "character"
```

```
data_num <- as.data.frame(apply(data, 2, as.numeric))
```

```
## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion
```

```
## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion

## Warning in apply(data, 2, as.numeric): NAs introduced by coercion
```

However, we found there are many NAs. When I check the original csv dataset, I found there are ‘,’ in some cells which should be ‘.’. So, I replace these ‘,’ with ‘.’ and create and use a new csv file called ‘new_Balances_final.csv’. Besides, I also replace three targets of rows whose target is 2 with 1 because it should be a binary target.

```
data<-fread('new_Balances_final.csv')
data$target<-as.factor(data$target)
sapply(data, class)
```

```
##                      KEY                   target       NON_CURRENT_ASSETS
##                "integer"                 "factor"                "numeric"
##    tangible_fixed_assets           CURRENT_ASSETS cash_other_liquid_assets
##                "numeric"                "numeric"                "numeric"
##                   STOCKS debtors_other_short_term             TOTAL_ASSETS
##                "numeric"                "numeric"                "numeric"
##                net_worth  NON_CURRENT_LIABILITIES      CURRENT_LIABILITIES
##                "numeric"                "numeric"                "numeric"
##                SUPPLIERS        TOTAL_LIABILITIES OTHER_OPERATION_EXPENSES
##                "numeric"                "numeric"                "numeric"
##          ORDINARY_EBITDA      Non_Ordinary_Result     NET_OPERATION_RESULT
##                "numeric"                "numeric"                "numeric"
##        FINANCIAL_INCOMES       FINANCIAL_EXPENSES        FINANCIAL_RESULTS
##                "numeric"                "numeric"                "numeric"
##    BENEFITS_BEFORE_TAXES            COMPANIES_TAX        BANK_INDEBTEDNESS
##                "numeric"                "numeric"                "numeric"
##     GROSS_FINANCIAL_DEBT               Net_result        CLIENT_COLLECTING
##                "numeric"                "numeric"                "numeric"
##       REVENUES_VARIATION          Exercise_result                   Income
##                "numeric"                "numeric"                "numeric"
##  other_operation_incomes       personnel_expenses   depreciation_provision
##                "numeric"                "numeric"                "numeric"
##             ROLLING_FUND
##                "numeric"
```

# Preparing the trainset and testset

I divide the dataset into trainset (80%) and testset (20%)

```
set.seed(1)
indexes<-sample(nrow(data),0.8*nrow(data),replace = F)
train<-data[indexes,2:34]
test<-data[-indexes,2:34]
dim(train)
```

```
## [1] 9584   33
```

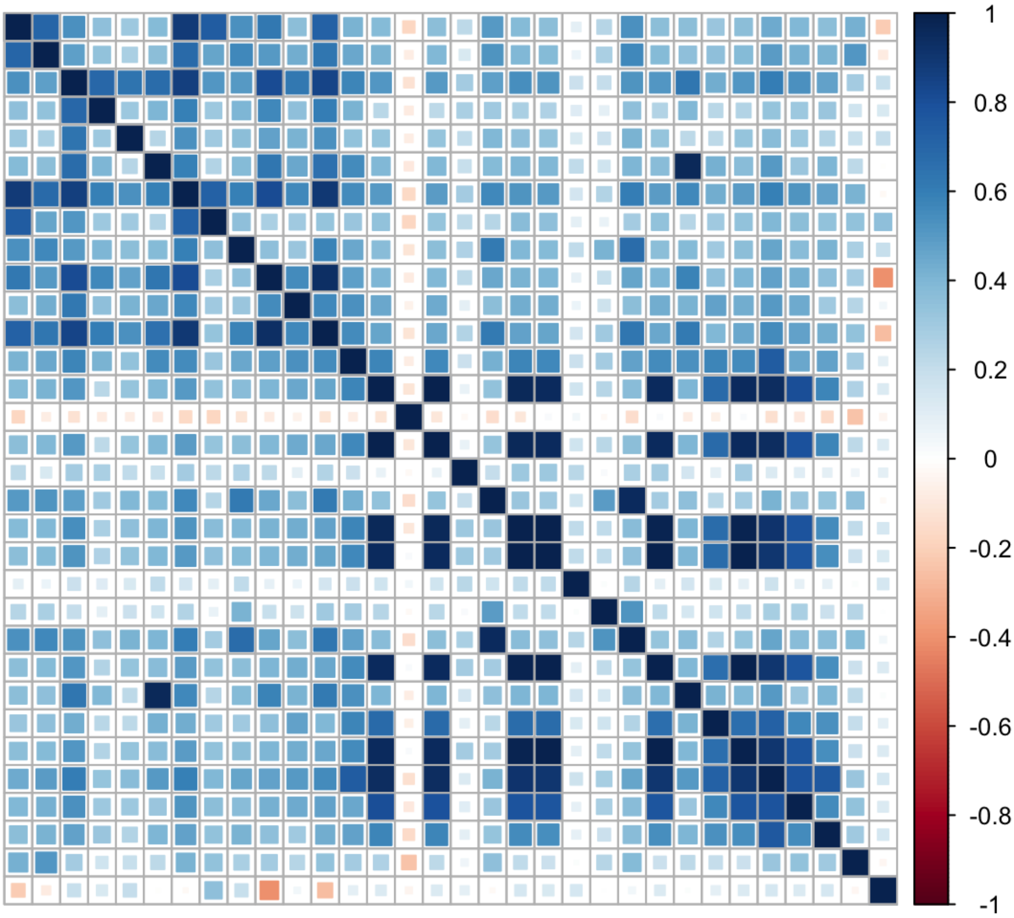
```
dim(test)
```

```
## [1] 2396   33
```

# Analysis

Let's the the correlationship of these variables.

```
cormat<-cor(train[,-'target'])
corrplot(cormat,method='square', tl.pos = FALSE)
```

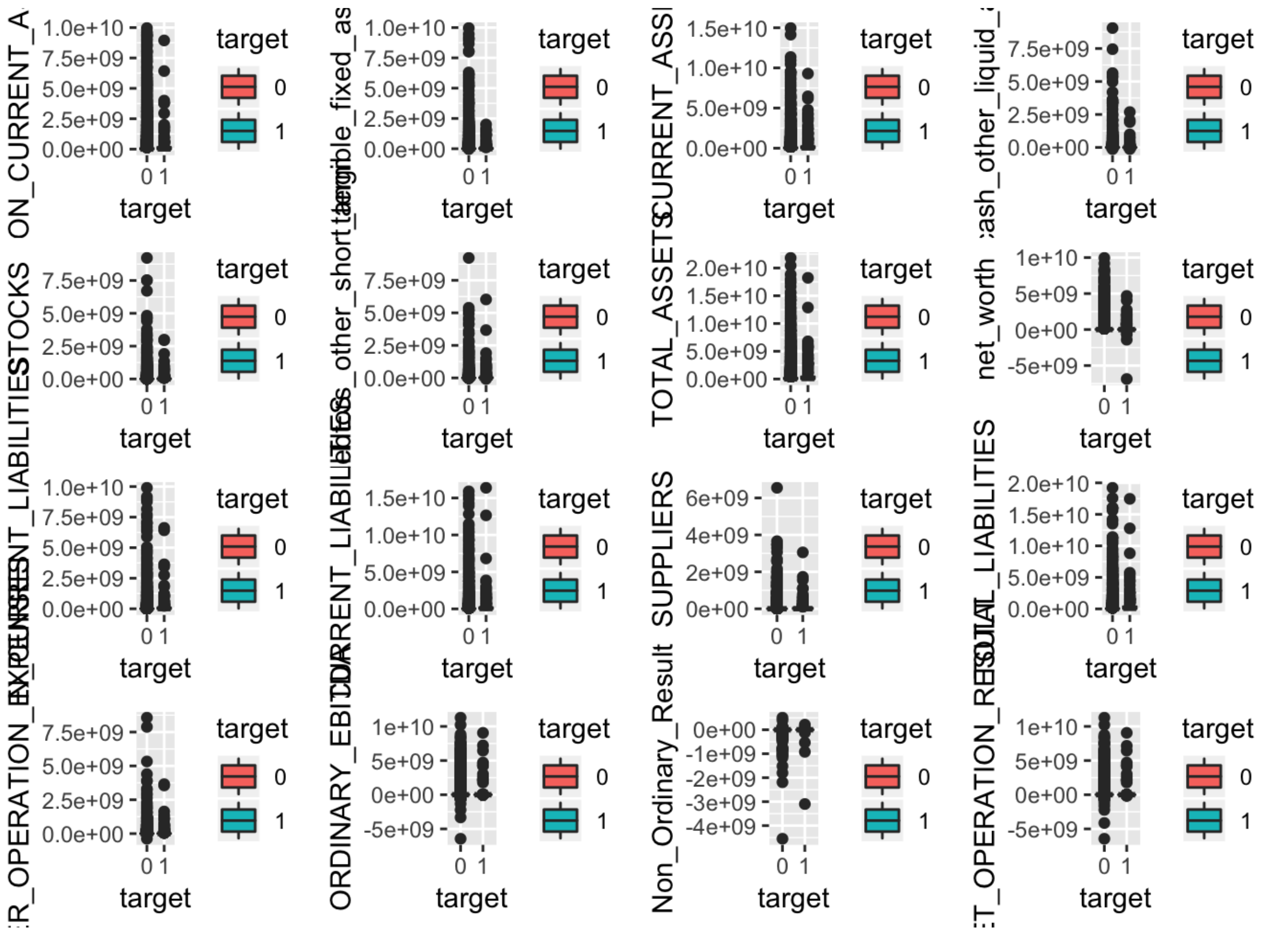



Then, let's check the distributions of each variables with target.

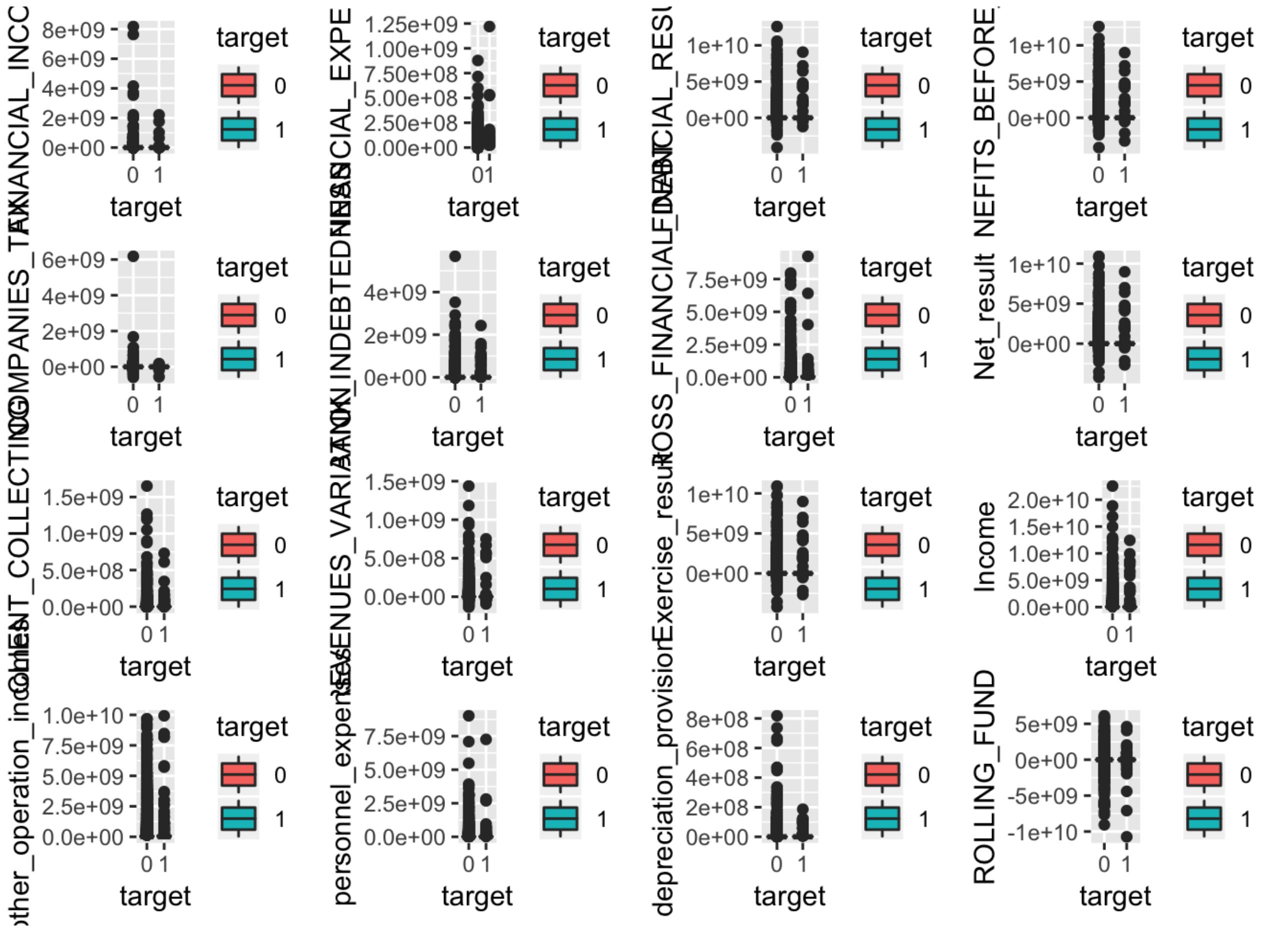
```
p1<-ggplot(data = train,aes(x = target,y = NON_CURRENT_ASSETS,fill=target))+geo
m_boxplot()
p2<-ggplot(data = train,aes(x = target,y = tangible_fixed_assets,fill=target))+
geom_boxplot()
p3<-ggplot(data = train,aes(x = target,y = CURRENT_ASSETS,fill=target))+geom_bo
xplot()
p4<-ggplot(data = train,aes(x = target,y = cash_other_liquid_assets,fill=targe
t))+geom_boxplot()
p5<-ggplot(data = train,aes(x = target,y = STOCKS,fill=target))+geom_boxplot()
p6<-ggplot(data = train,aes(x = target,y = debtors_other_short_term,fill=targe
t))+geom_boxplot()
p7<-ggplot(data = train,aes(x = target,y = TOTAL_ASSETS,fill=target))+geom_boxp
lot()
p8<-ggplot(data = train,aes(x = target,y = net_worth,fill=target))+geom_boxplot
()
p9<-ggplot(data = train,aes(x = target,y = NON_CURRENT_LIABILITIES,fill=targe
t))+geom_boxplot()
p10<-ggplot(data = train,aes(x = target,y = CURRENT_LIABILITIES,fill=target))+g
eom_boxplot()
p11<-ggplot(data = train,aes(x = target,y = SUPPLIERS,fill=target))+geom_boxplo
t()
p12<-ggplot(data = train,aes(x = target,y = TOTAL_LIABILITIES,fill=target))+geo
m_boxplot()
p13<-ggplot(data = train,aes(x = target,y = OTHER_OPERATION_EXPENSES,fill=targe
t))+geom_boxplot()
p14<-ggplot(data = train,aes(x = target,y = ORDINARY_EBITDA,fill=target))+geom_
boxplot()
p15<-ggplot(data = train,aes(x = target,y = Non_Ordinary_Result,fill=target))+g
eom_boxplot()
p16<-ggplot(data = train,aes(x = target,y = NET_OPERATION_RESULT,fill=target))+
geom_boxplot()
p17<-ggplot(data = train,aes(x = target,y = FINANCIAL_INCOMES,fill=target))+geo
m_boxplot()
p18<-ggplot(data = train,aes(x = target,y = FINANCIAL_EXPENSES,fill=target))+ge
om_boxplot()
p19<-ggplot(data = train,aes(x = target,y = FINANCIAL_RESULTS,fill=target))+geo
m_boxplot()
p20<-ggplot(data = train,aes(x = target,y = BENEFITS_BEFORE_TAXES,fill=target))
+geom_boxplot()
p21<-ggplot(data = train,aes(x = target,y = COMPANIES_TAX,fill=target))+geom_bo
xplot()
p22<-ggplot(data = train,aes(x = target,y = BANK_INDEBTEDNESS,fill=target))+geo
m_boxplot()
p23<-ggplot(data = train,aes(x = target,y = GROSS_FINANCIAL_DEBT,fill=target))+
geom_boxplot()
p24<-ggplot(data = train,aes(x = target,y = Net_result,fill=target))+geom_boxpl
ot()
p25<-ggplot(data = train,aes(x = target,y = CLIENT_COLLECTING,fill=target))+geo
m_boxplot()
p26<-ggplot(data = train,aes(x = target,y = REVENUES_VARIATION,fill=target))+ge
om_boxplot()
p27<-ggplot(data = train,aes(x = target,y = Exercise_result,fill=target))+geom_
boxplot()
p28<-ggplot(data = train,aes(x = target,y = Income,fill=target))+geom_boxplot()
p29<-ggplot(data = train,aes(x = target,y = other_operation_incomes,fill=targe
t))+geom_boxplot()
p30<-ggplot(data = train,aes(x = target,y = personnel_expenses,fill=target))+ge
```

```
om_boxplot()
p31<-ggplot(data = train,aes(x = target,y = depreciation_provision,fill=targe
t))+geom_boxplot()
p32<-ggplot(data = train,aes(x = target,y = ROLLING_FUND,fill=target))+geom_box
plot()

grid.arrange(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,nrow=4)
```



```
grid.arrange(p17,p18,p19,p20,p21,p22,p23,p24,p25,p26,p27,p28,p29,p30,p31,p32,nr
ow=4)
```




# Regularization

Before building our models, I regularize the dataset in order to overcome overfitting.

```
train<-regularize(train)
test<-regularize(test)
```

# Model 1: Logistic regression

The first model is logistic regression. I calculate the AUC with the fun.auc() provided. Then show the ROC graph. The AUC is 0.562.

```
full.log.probit<-glm(data = train,target~.,family = binomial(link=probit))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(full.log.probit)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = probit), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
##   -8.49     0.00     0.00     0.00     8.49
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error    z value Pr(>|z|)
## (Intercept)              -6.567e+14  7.260e+05 -904578833   <2e-16 ***
## NON_CURRENT_ASSETS       -5.841e+14  4.988e+07  -11709167   <2e-16 ***
## tangible_fixed_assets    -7.787e+03  2.739e-03   -2842845   <2e-16 ***
## CURRENT_ASSETS           -7.413e+14  5.555e+07  -13343908   <2e-16 ***
## cash_other_liquid_assets -1.271e+05  4.505e-03  -28202924   <2e-16 ***
## STOCKS                    1.401e+05  4.245e-03   32993031   <2e-16 ***
## debtors_other_short_term  2.212e+06  1.388e-02  159331644   <2e-16 ***
## TOTAL_ASSETS              5.841e+14  4.988e+07   11709183   <2e-16 ***
## net_worth                -8.302e+08  1.237e+01  -67100960   <2e-16 ***
## NON_CURRENT_LIABILITIES   1.471e+05  3.125e-03   47087719   <2e-16 ***
## CURRENT_LIABILITIES       1.572e+14  3.152e+07    4988145   <2e-16 ***
## SUPPLIERS                -3.742e+05  6.213e-03  -60229595   <2e-16 ***
## TOTAL_LIABILITIES        -8.305e+08  1.237e+01  -67122308   <2e-16 ***
## OTHER_OPERATION_EXPENSES -7.896e+14  2.739e+07  -28827897   <2e-16 ***
## ORDINARY_EBITDA           9.757e+14  5.309e+07   18377969   <2e-16 ***
## Non_Ordinary_Result       2.349e+14  4.095e+07    5736042   <2e-16 ***
## NET_OPERATION_RESULT     -1.888e+15  4.535e+07  -41632901   <2e-16 ***
## FINANCIAL_INCOMES        -1.228e+14  1.341e+06  -91523992   <2e-16 ***
## FINANCIAL_EXPENSES        1.228e+14  1.341e+06   91524018   <2e-16 ***
## FINANCIAL_RESULTS         3.576e+14  4.095e+07    8733955   <2e-16 ***
## BENEFITS_BEFORE_TAXES     2.886e+15  7.780e+07   37091701   <2e-16 ***
## COMPANIES_TAX            -3.121e+15  7.708e+07  -40483847   <2e-16 ***
## BANK_INDEBTEDNESS        -3.764e+05  6.034e-03  -62387992   <2e-16 ***
## GROSS_FINANCIAL_DEBT     -3.657e+06  9.304e-03 -393092262   <2e-16 ***
## Net_result               -3.121e+15  7.708e+07  -40483847   <2e-16 ***
## CLIENT_COLLECTING        -1.176e+07  7.063e-02 -166537296   <2e-16 ***
## REVENUES_VARIATION        1.130e+06  2.337e-02   48361134   <2e-16 ***
## Exercise_result                  NA         NA         NA       NA
## Income                    7.896e+14  2.739e+07   28827897   <2e-16 ***
## other_operation_incomes   1.735e+05  1.946e-03   89168555   <2e-16 ***
## personnel_expenses       -7.896e+14  2.739e+07  -28827897   <2e-16 ***
## depreciation_provision   -1.765e+15  4.537e+07  -38912853   <2e-16 ***
## ROLLING_FUND              1.572e+14  3.152e+07    4988145   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance:  1943.6  on 9583  degrees of freedom
## Residual deviance: 13840.8  on 9552  degrees of freedom
## AIC: 13905
##
## Number of Fisher Scoring iterations: 23
```

```
full.log.probit.prediction<-predict(full.log.probit,type = "response")
fun.auc(full.log.probit.prediction,train$target)
```

```
##        AUC x.Sens x.Spec SS_min_dif SS_max_sum Min_Err Min_Err_Cut
## 9450 0.562  0.375  0.666          1          1    0.02           1
```

```
pROC_obj <- roc(train$target,full.log.probit.prediction,
                smoothed = TRUE,
                # arguments for ci
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                # arguments for plot
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE)
```

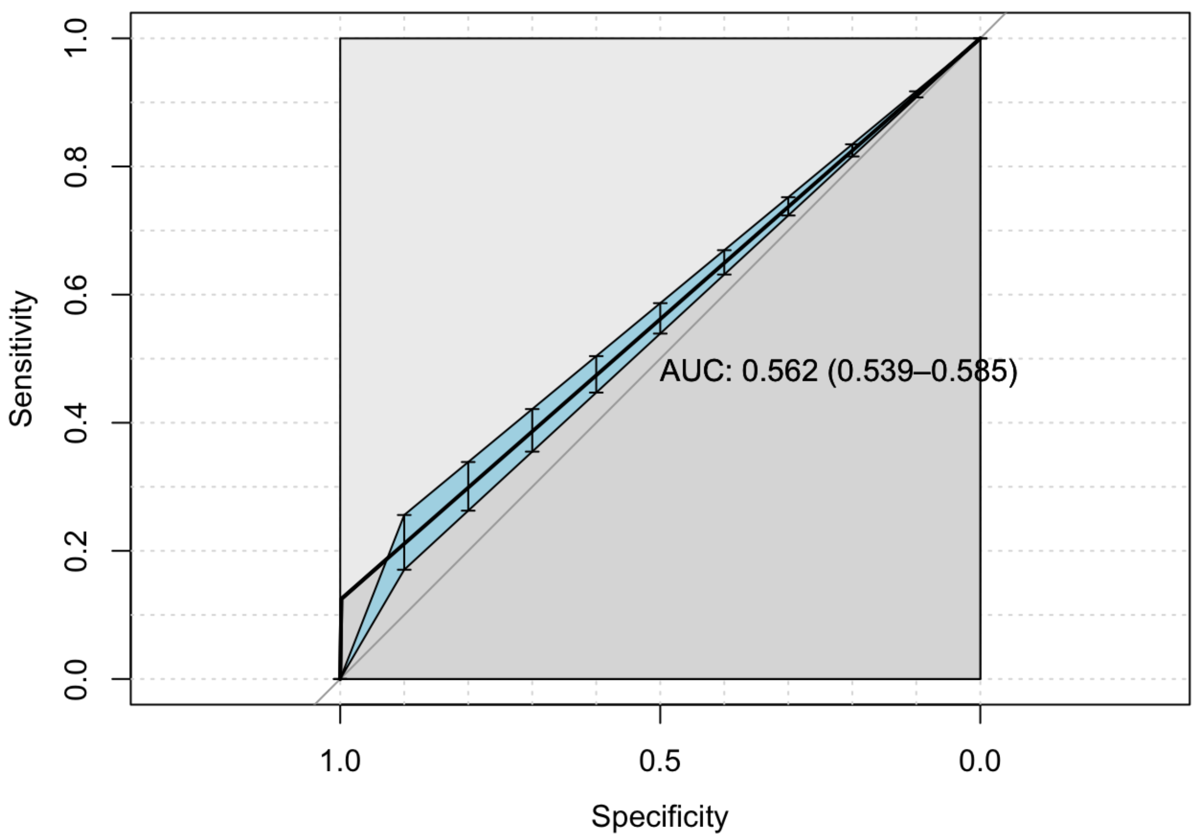
```
## Setting levels: control = 0, case = 1
```

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

```
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col="lightblue")
```

```
## Warning in plot.ci.se(sens.ci, type = "shape", col = "lightblue"): Low
## definition shape.
```

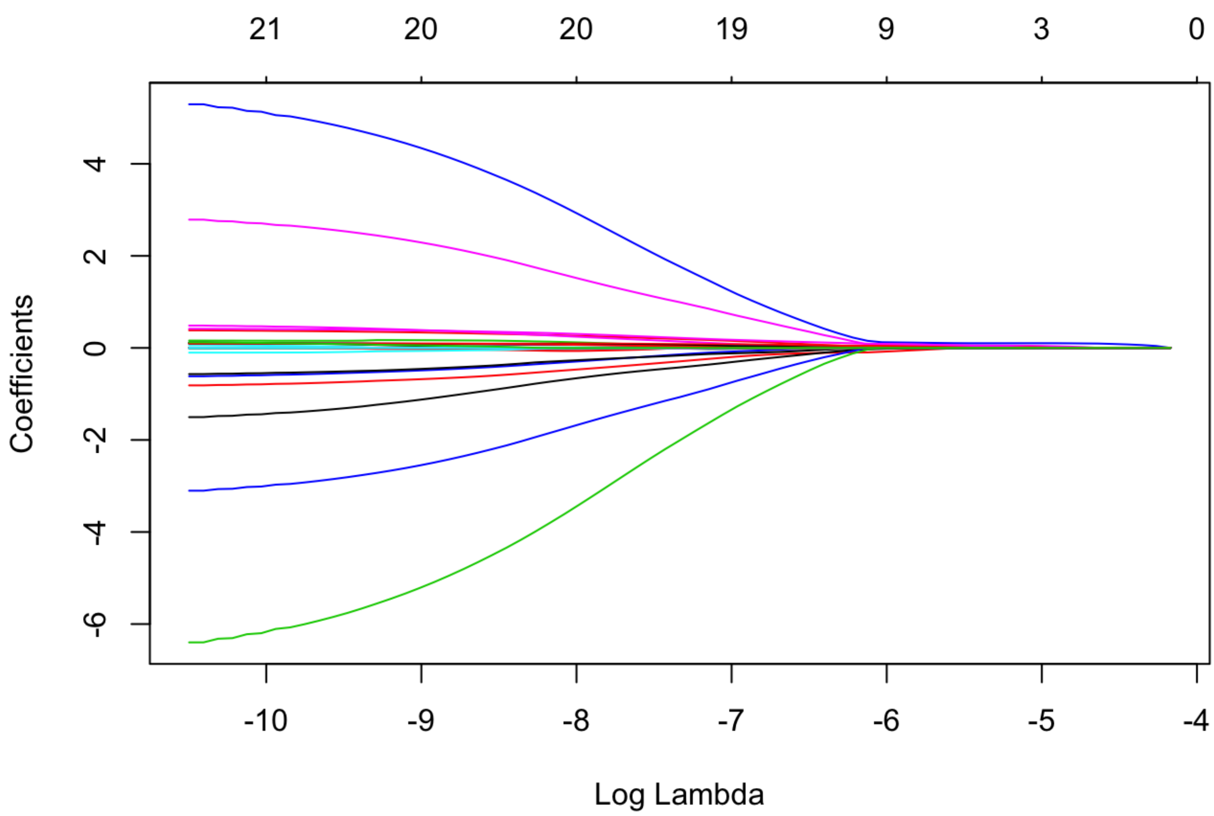
```
plot(sens.ci, type="bars")
```



# Model 2: Lasso (least absolute shrinkage and selection operator) Regression

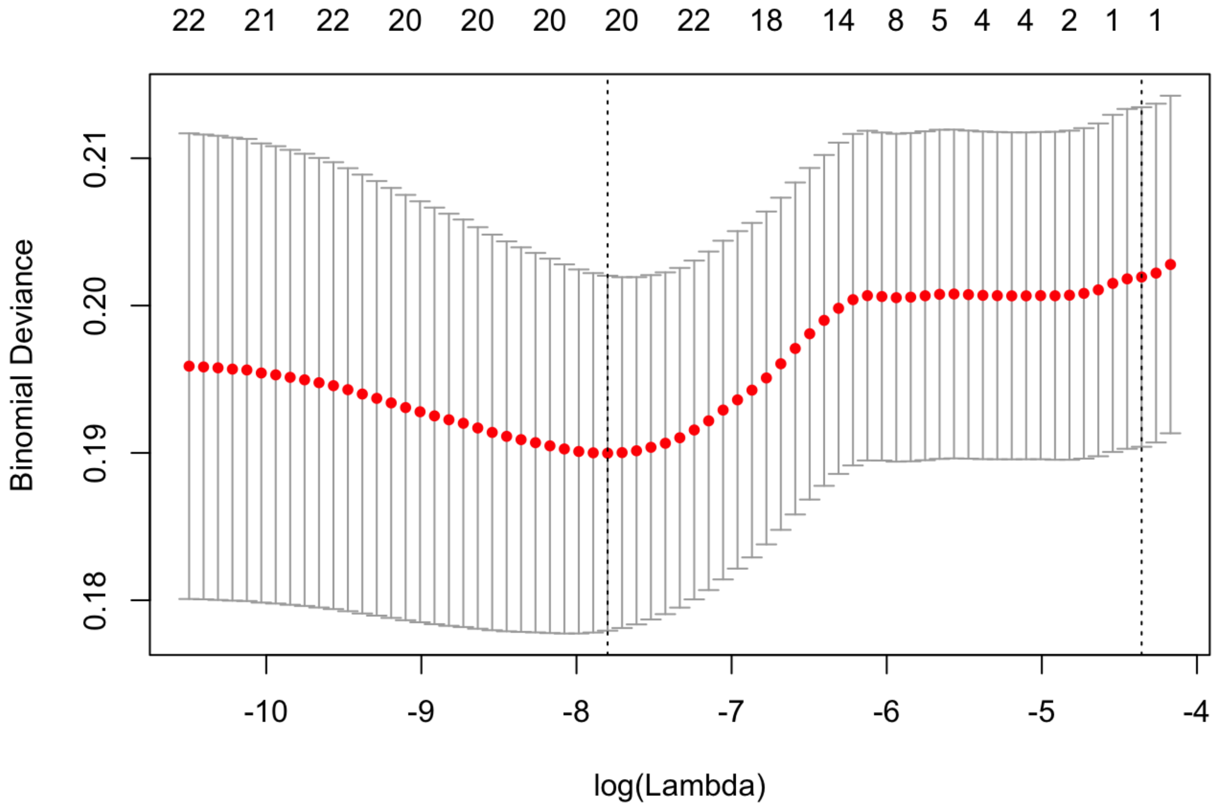
I want to select variables that are most important.

```
# Model 2:LASSO
X<-scale(train[,-'target'])
X<-as.matrix(X)
Y<- as.matrix(train[,'target'])
lasso.fit<- glmnet(x=X, y=Y, family = "binomial", alpha = 1)
plot(lasso.fit, xvar = "lambda")
```




We want to choose the optimum value of lambda using Cross Validation.

```
cv.lasso<- cv.glmnet(x=X, y=Y,family = "binomial", alpha = 1, nfolds = 10)
plot(cv.lasso)
```

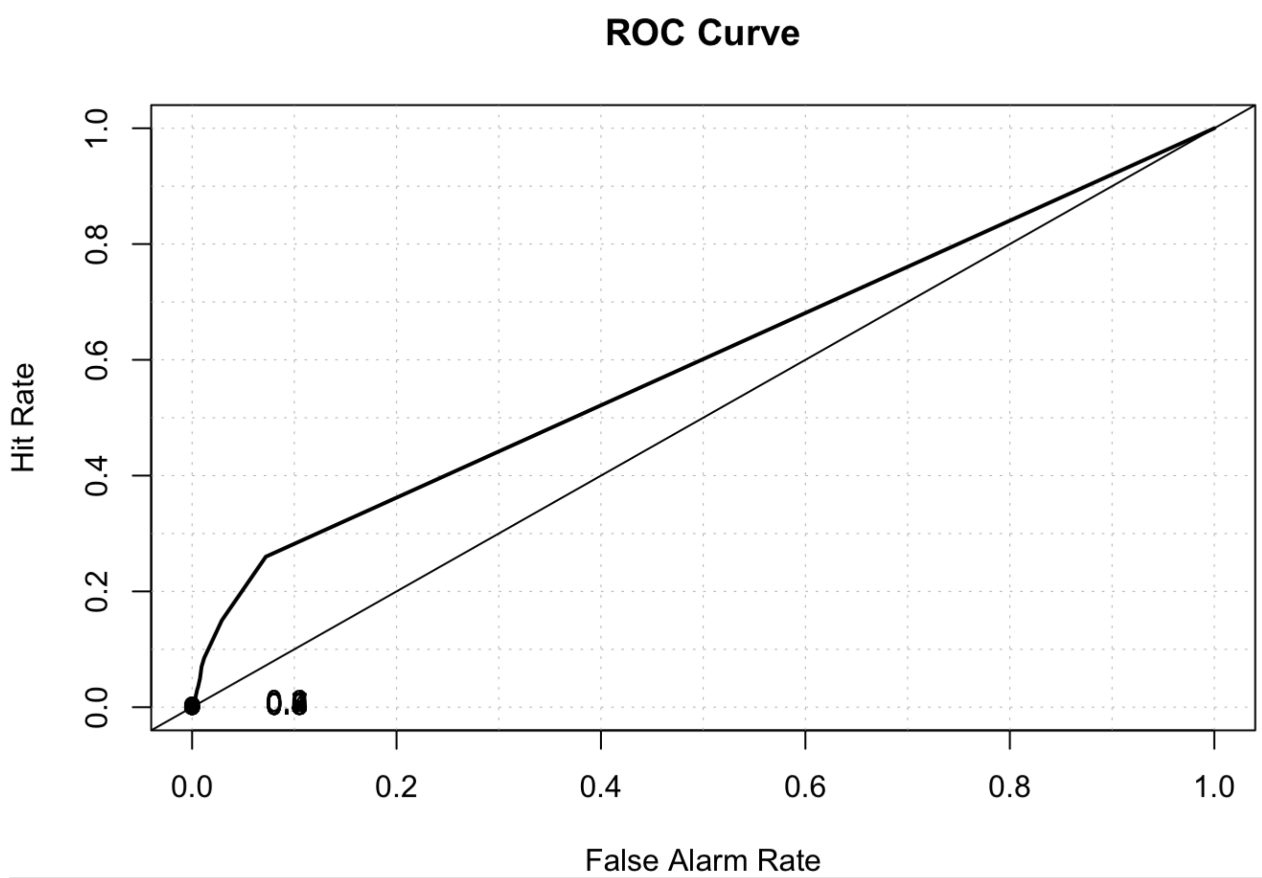



```
# cv.lasso$lambda.min
# cv.lasso$lambda.1se

# Choose cv.lasso$lambda.1se
coef(lasso.fit, s=cv.lasso$lambda.1se)
```

```
## 33 x 1 sparse Matrix of class "dgCMatrix"
##                                    1
## (Intercept)              -3.85127982
## NON_CURRENT_ASSETS         .
## tangible_fixed_assets      .
## CURRENT_ASSETS             .
## cash_other_liquid_assets   .
## STOCKS                     .
## debtors_other_short_term   .
## TOTAL_ASSETS               .
## net_worth                  .
## NON_CURRENT_LIABILITIES    .
## CURRENT_LIABILITIES        .
## SUPPLIERS                  .
## TOTAL_LIABILITIES          .
## OTHER_OPERATION_EXPENSES   .
## ORDINARY_EBITDA            .
## Non_Ordinary_Result        .
## NET_OPERATION_RESULT       .
## FINANCIAL_INCOMES          .
## FINANCIAL_EXPENSES         0.06245771
## FINANCIAL_RESULTS          .
## BENEFITS_BEFORE_TAXES      .
## COMPANIES_TAX              .
## BANK_INDEBTEDNESS          .
## GROSS_FINANCIAL_DEBT       .
## Net_result                 .
## CLIENT_COLLECTING          .
## REVENUES_VARIATION         .
## Exercise_result            .
## Income                     .
## other_operation_incomes    .
## personnel_expenses         .
## depreciation_provision     .
## ROLLING_FUND               .
```

```
## Predictions using, s=cv.lasso$lambda.1se
pred.lasso<- predict(lasso.fit, newx = X, s=cv.lasso$lambda.1se,type = 'respons
e')
roc.plot(x = train$target == "1", pred = pred.lasso,thresholds = thresh)$roc.vo
l
```



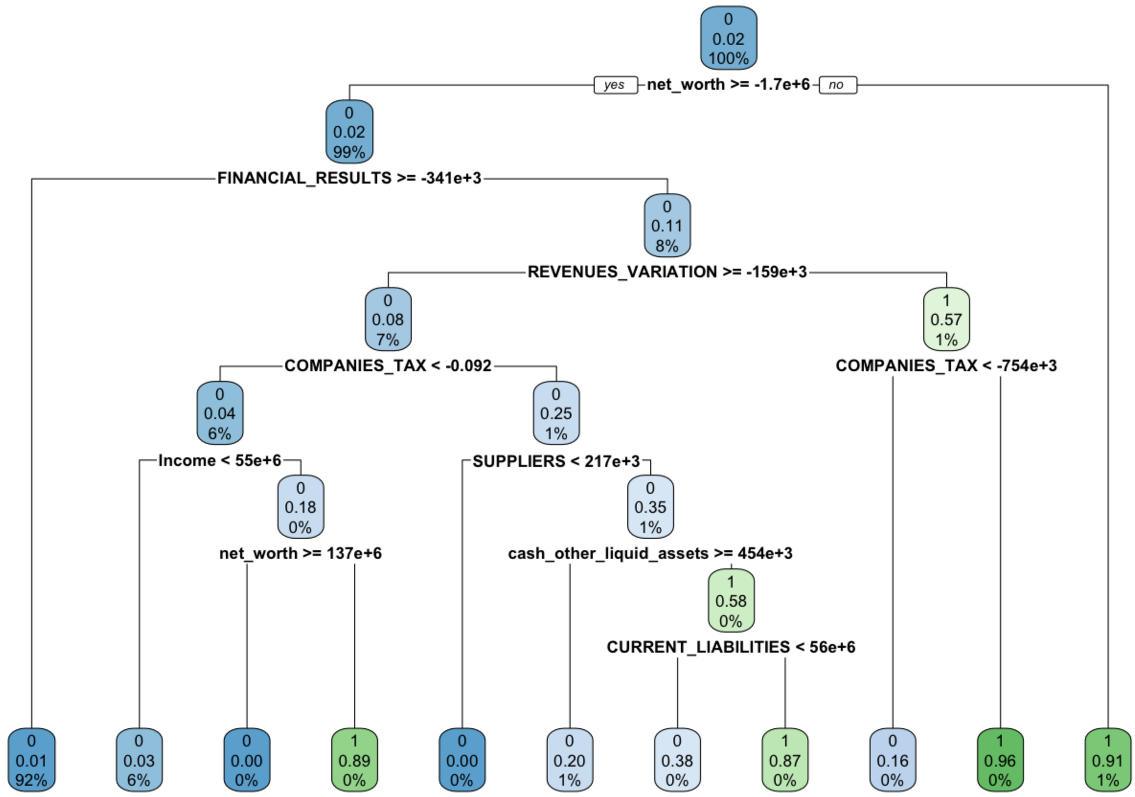

| Model | Area | p.value | binorm.area |
|---|---|---|---|
| <fctr> | <dbl> | <dbl> | <dbl> |
| Model 1 | 0.6892991 | 9.842435e-21 | NA |

1 row

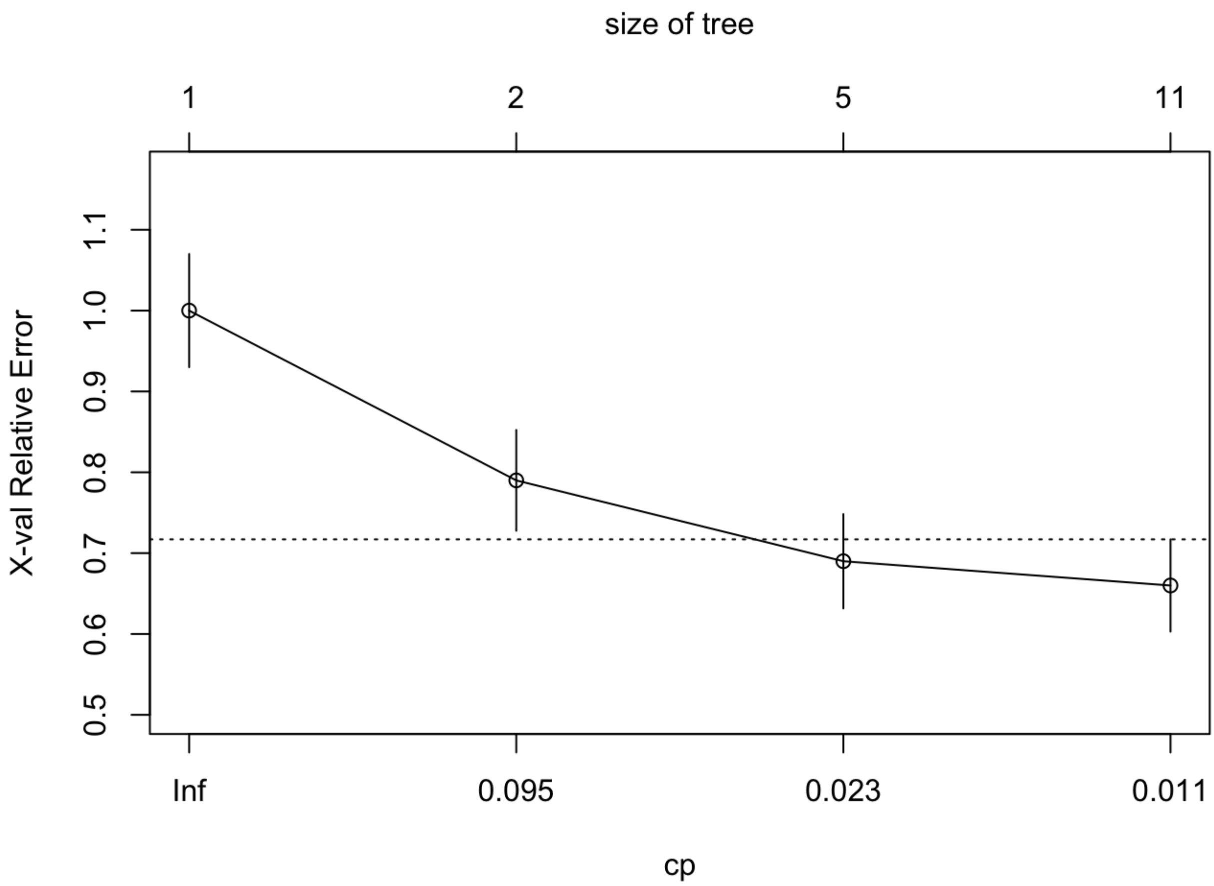
Finally, we got AUC with 0.6892991.

# Model 3: Classification Tree

```
full.rpart<-rpart(data = train,target~.,method = 'class')
rpart.plot(full.rpart)
```



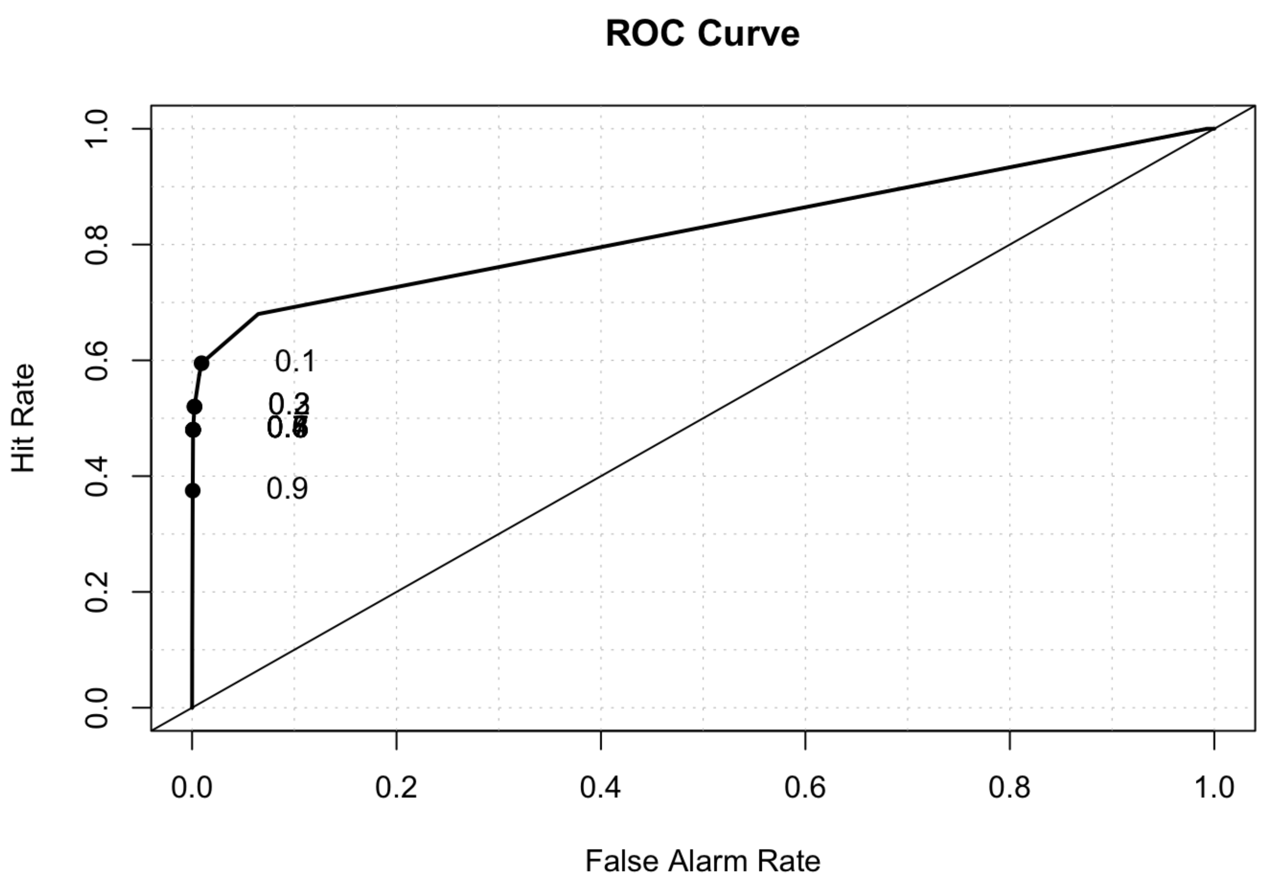

```
plotcp(full.rpart)
```




```
printcp(full.rpart)
```

```
##
## Classification tree:
## rpart(formula = target ~ ., data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] cash_other_liquid_assets COMPANIES_TAX
## [3] CURRENT_LIABILITIES      FINANCIAL_RESULTS
## [5] Income                   net_worth
## [7] REVENUES_VARIATION       SUPPLIERS
##
## Root node error: 200/9584 = 0.020868
##
## n= 9584
##
##      CP nsplit rel error xerror     xstd
## 1 0.225      0     1.000   1.00 0.069969
## 2 0.040      1     0.775   0.79 0.062329
## 3 0.013      4     0.655   0.69 0.058312
## 4 0.010     10     0.565   0.66 0.057049
```

```
rpart.prediction<-predict(full.rpart,type = 'prob')
roc.plot(x = train$target == "1", pred = rpart.prediction[,2],thresholds = thre
sh)$roc.vol
```



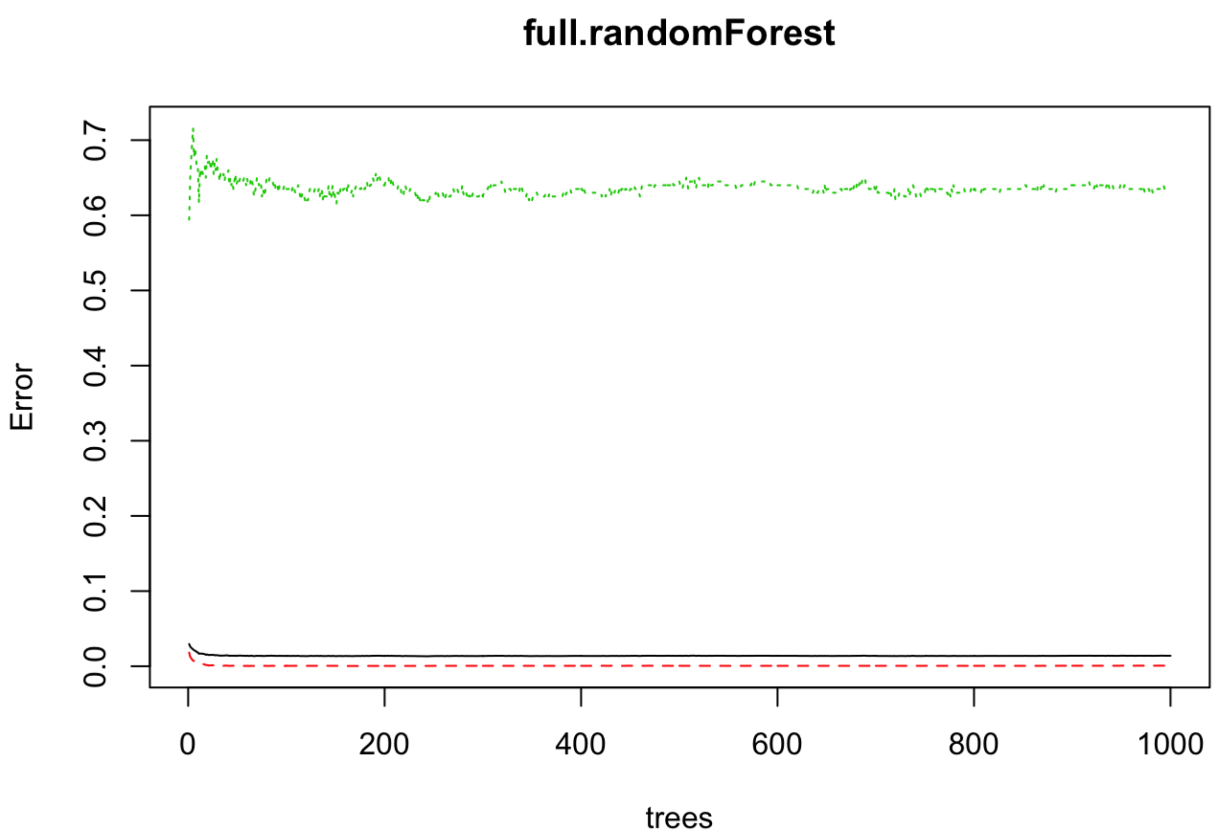

| Model | Area | p.value | binorm.area |
|---|---|---|---|
| <fctr> | <dbl> | <dbl> | <dbl> |
| Model 1 | 0.8270479 | 2.877032e-236 | NA |

1 row

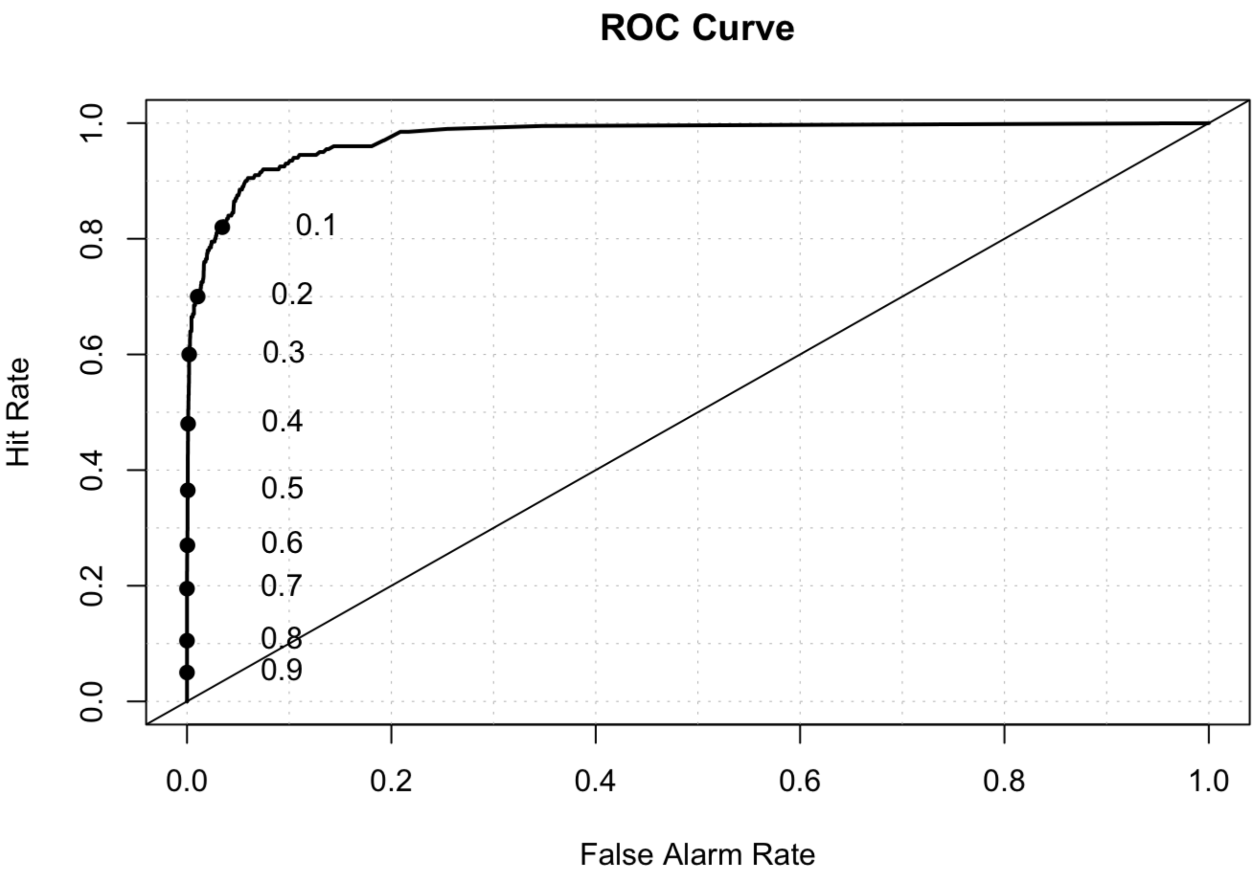
The AUC is 0.8270479.

# Model 4: Random forest

```
full.randomForest<-randomForest(data=train,target~.,ntree=1000)
plot(full.randomForest)
```




```
rf.predicted<-predict(full.randomForest,type = 'prob')
roc.plot(x = train$target == "1", pred = rf.predicted[,2],thresholds = thresh)
$roc.vol
```




| Model <fctr> | Area <dbl> | p.value <dbl> | binorm.area <dbl> |
|---|---|---|---|
| Model 1 | 0.9765175 | 1.92025e-159 | *NA* |

1 row

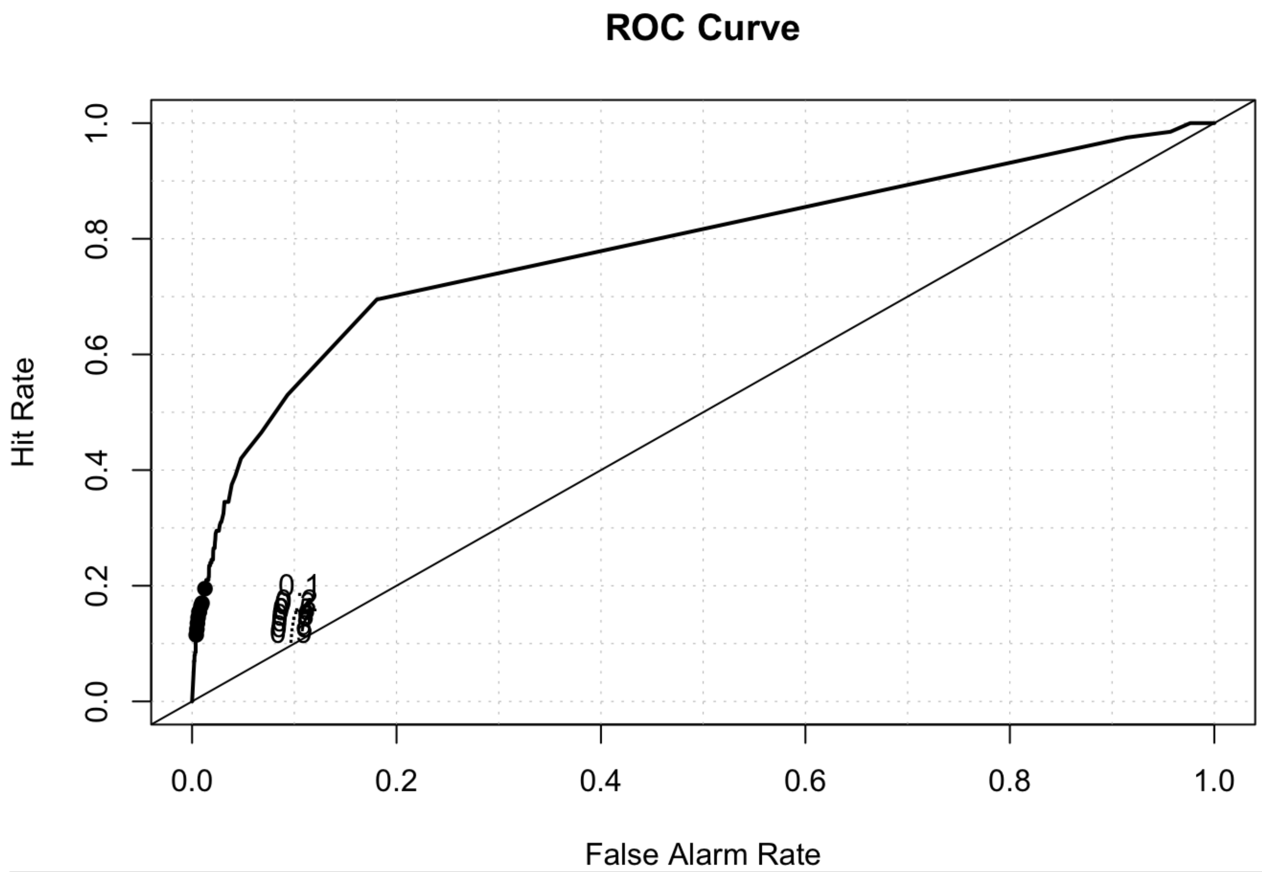
The AUC is 0.9765175.

# Model 5: Linear Discriminant Analysis

```
model.lda<-lda(data=train,target~.)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda.predicted<-predict(model.lda)$posterior[,2]
roc.plot(x=train$target=="1",pred=lda.predicted,thresholds = thresh)$roc.vol
```




| **Model** <fctr> | **Area** <dbl> | **p.value** <dbl> | **binorm.area** <dbl> |
|---|---:|---:|---:|
| Model 1 | 0.8274339 | 4.965233e-57 | *NA* |

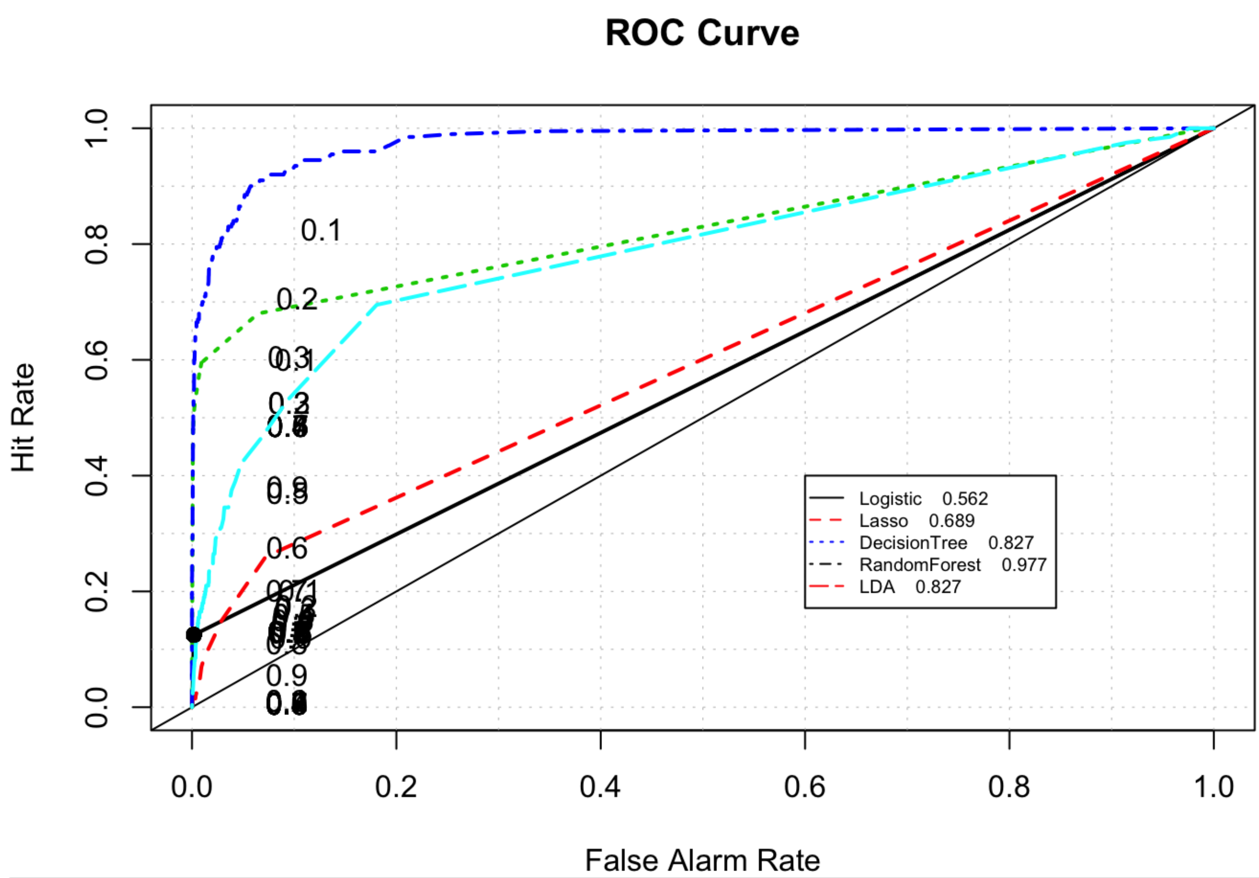
1 row

The AUC is 0.8274339.

# Comparision

When we compare the results of the five models, we can find that random forest is the best with the highest AUC of 0.9765175.

```
roc.plot(x=train$target=="1",pred=cbind(full.log.probit.prediction,pred.lasso,
                                        rpart.prediction[,2],rf.predicted[,2],ld
a.predicted),legend = T,
         leg.text = c("Logistic","Lasso","DecisionTree",
                      "RandomForest","LDA"),thresholds = thresh)$roc.vol
```




| Model <fctr> | Area <dbl> | p.value <dbl> | binorm.area <dbl> |
|---|---|---|---|
| Model 1 | 0.5615942 | 1.994126e-150 | NA |
| Model 2 | 0.6892991 | 9.842435e-21 | NA |
| Model 3 | 0.8270479 | 2.877032e-236 | NA |
| Model 4 | 0.9765175 | 1.920250e-159 | NA |
| Model 5 | 0.8274339 | 4.965233e-57 | NA |

5 rows

Then, we compare them with testset. The result is similar: Random Forest has the best peformance.

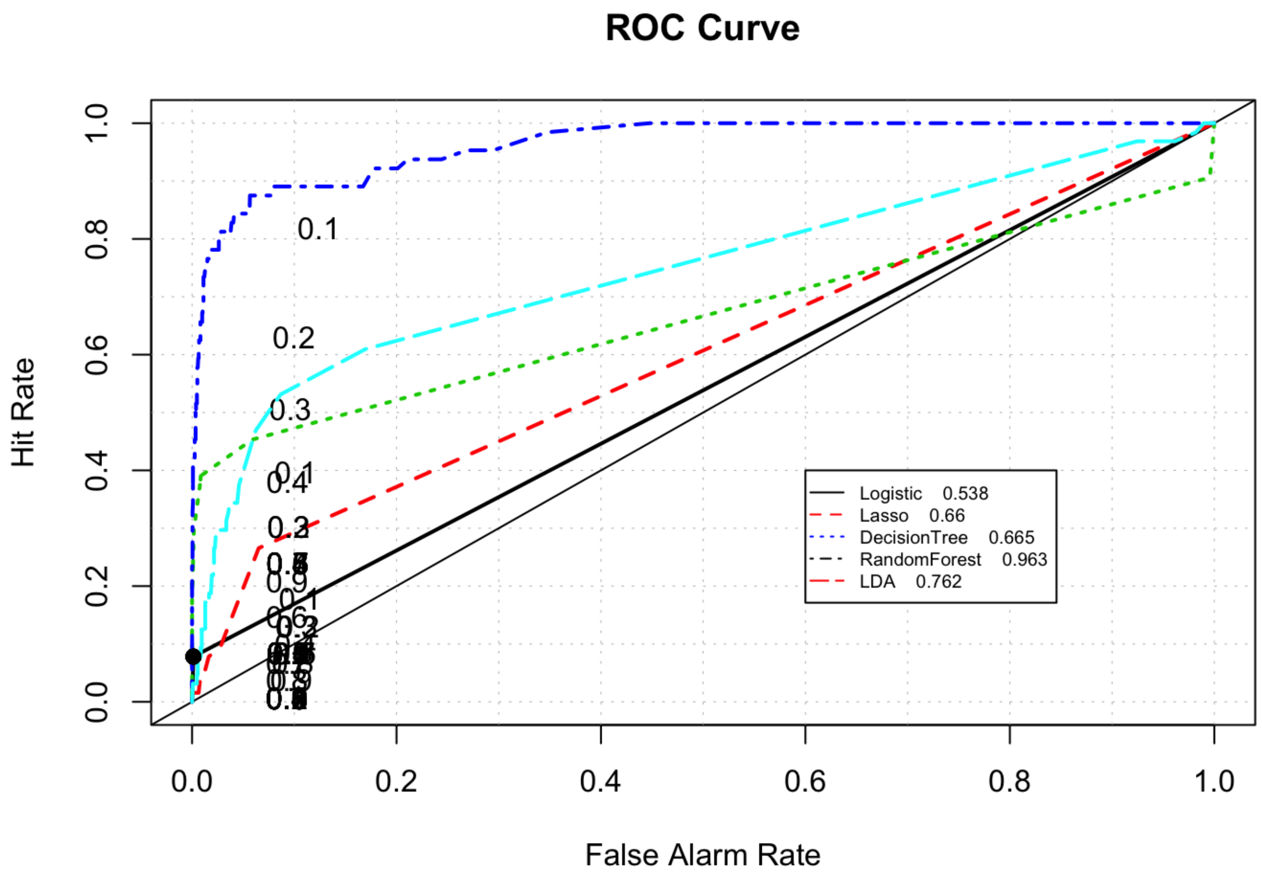
```
X<-scale(test[,-'target'])
X<-as.matrix(X)

logit.test.pred<-predict(full.log.probit,test,type = 'response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
lasso.test.pred<-predict(lasso.fit, newx = X, s=cv.lasso$lambda.1se,type = 'res
ponse')
rpart.test.pred<-predict(full.rpart,test,type = 'prob')
rf.test.pred<-predict(full.randomForest,test,type = 'prob')[,2]
lda.test.pred<-predict(model.lda,test)$posterior[,2]

roc.plot(x=test$target=="1",pred=cbind(logit.test.pred,lasso.test.pred,
                                      rpart.test.pred[,2],rf.test.pred,
                                      lda.test.pred),legend = T,
         leg.text = c("Logistic","Lasso","DecisionTree",
                      "RandomForest","LDA"),thresholds = thresh)$roc.vol
```




| Model <fctr> | Area <dbl> | p.value <dbl> | binorm.area <dbl> |
|---|---|---|---|
| Model 1 | 0.5384193 | 3.901219e-26 | NA |
| Model 2 | 0.6604611 | 4.701152e-06 | NA |
| Model 3 | 0.6648498 | 4.979949e-23 | NA |
| Model 4 | 0.9632189 | 1.543355e-43 | NA |
| Model 5 | 0.7621141 | 3.898907e-13 | NA |

5 rows

Show the comparision results in table in order:

```
models<-c("Logistic Reg","Lasso Reg","DecisionTree","RandomForest","LDA")
TrainAuc<-c(cost2(train$target,as.numeric(full.log.probit.prediction)),cost2(tr
ain$target,as.numeric(pred.lasso)),
            cost2(train$target,as.numeric(rpart.prediction[,2])),cost2(train$ta
rget,as.numeric(rf.predicted[,2])),
            cost2(train$target,as.numeric(lda.predicted)))
```

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

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

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

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

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

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

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

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

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

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

```
TestAuc<-c(cost2(test$target,as.numeric(logit.test.pred)),cost2(test$target,as.
numeric(lasso.test.pred)),
           cost2(test$target,as.numeric(rpart.test.pred[,2])),cost2(test$targe
t,as.numeric(rf.test.pred)),
           cost2(test$target,as.numeric(lda.test.pred)))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

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

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

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

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

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

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

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

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

```
results<-as.data.frame(cbind(models,TrainAuc,TestAuc))
results<-results%>%arrange(desc(TestAuc))
datatable(results)
```

Show 10 entries Search:

| | models | TrainAuc | TestAuc |
|---|---|---|---|
| 1 | RandomForest | 0.97651747655584 | 0.963218937607204 |
| 2 | LDA | 0.827433930093777 | 0.762114065180103 |
| 3 | DecisionTree | 0.827047900682012 | 0.664849780231561 |
| 4 | Lasso Reg | 0.689299072890026 | 0.66046111170669 |
| 5 | Logistic Reg | 0.561594202898551 | 0.538419275300172 |

Showing 1 to 5 of 5 entries Previous 1 Next

# Find optimum cutoff probability for minimizing the cost function

Finally, I want to find optimum cutoff probability for minimizing the cost function or maximizing the accuracy function. Here, I maximize the accuracy function and pursue the highest accuracy which is 0.9828881.

```
probs<-seq(0,1,0.001)
cost<-NULL
for (i in 1:1000)
{
  cutoff<-probs[i]
  predicted<-ifelse(rf.predicted[,2]>cutoff,1,0)
  cost[i]<-accuracy(train$target,predicted)
}
plot(1:1000,cost)
cutoffProb<-probs[which(cost==max(cost))]
cutoffProb
```

```
## [1] 0.300 0.301 0.305 0.306 0.307 0.308
```

```
predicted<-ifelse(rf.test.pred>cutoffProb,1,0)
```

```
## Warning in rf.test.pred > cutoffProb: longer object length is not a
## multiple of shorter object length
```

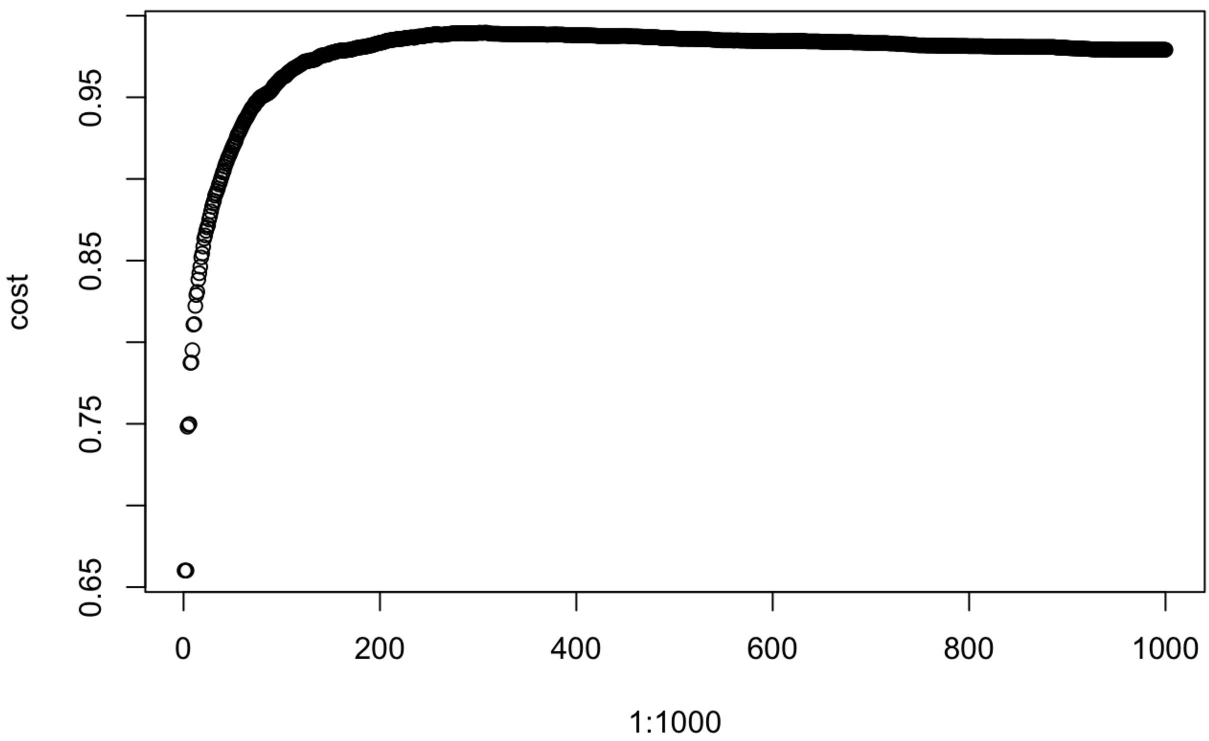
```
cm<-confusionMatrix(as.factor(predicted),test$target)
cm[2]
```

```
## $table
##           Reference
## Prediction    0    1
##          0 2324   33
##          1    8   31
```

```
cm[3]$overall[1]
```

```
##  Accuracy
## 0.9828881
```

```
# the highest accuracy
probs<-seq(0,1,0.001)
cost<-NULL
for (i in 1:1000)
{
  cutoff<-probs[i]
  predicted<-ifelse(rf.predicted[,2]>cutoff,1,0)
  cost[i]<-accuracy(train$target,predicted)
}
plot(1:1000,cost)
```




```
cutoffProb<-probs[which(cost==max(cost))]
cutoffProb
```

```
## [1] 0.300 0.301 0.305 0.306 0.307 0.308
```

```
predicted<-ifelse(rf.test.pred>cutoffProb,1,0)
```

```
## Warning in rf.test.pred > cutoffProb: longer object length is not a
## multiple of shorter object length
```

```
cm<-confusionMatrix(as.factor(predicted),test$target)
cm[2]
```

```
## $table
##           Reference
## Prediction    0    1
##          0 2324   33
##          1    8   31
```

```
cm[3]$overall[1]
```

```
##  Accuracy
## 0.9828881
```

```
# the highest accuracy
```

# Conclusion

In conclusion, compared with the five models, I finally recommend choosinv random forest model and the accuracy is 0.9828881 which is very high and satisfying.