```r
#install.packages("PreProcess")
#install.packages("rattle")
library(corrplot)
library(dplyr)
library(rpart)
library(randomForest)
library(caret)
library(CORElearn)
library(PreProcess)
library(MASS)
library(rattle)
library(pROC)
```

1 Exploration

```r
test <- read.table(file="test.csv", sep=",", header=TRUE)
train <- read.table(file="train.csv", sep=",", header=TRUE)

set.seed(100)
```

Target feature is biodegradability.

```r
summary(test)
```

```
##        V1              V2              V3              V4         
##  Min.   :2.000   Min.   :1.135   Min.   :0.000   Min.   :0.00000  
##  1st Qu.:4.414   1st Qu.:2.494   1st Qu.:0.000   1st Qu.:0.00000  
##  Median :4.807   Median :3.039   Median :0.000   Median :0.00000  
##  Mean   :4.751   Mean   :3.130   Mean   :0.622   Mean   :0.08612  
##  3rd Qu.:5.188   3rd Qu.:3.555   3rd Qu.:1.000   3rd Qu.:0.00000  
##  Max.   :6.253   Max.   :9.178   Max.   :8.000   Max.   :3.00000  
##        V5               V6                V7               V8       
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.00  
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.:29.40  
##  Median : 0.000   Median : 0.0000   Median : 0.000   Median :34.20  
##  Mean   : 1.115   Mean   : 0.3397   Mean   : 1.555   Mean   :35.57  
##  3rd Qu.: 1.000   3rd Qu.: 0.0000   3rd Qu.: 3.000   3rd Qu.:41.20  
##  Max.   :16.000   Max.   :12.0000   Max.   :14.000   Max.   :60.00  
##        V9             V10             V11              V12         
##  Min.   :0.000   Min.   : 0.00   Min.   : 0.000   Min.   :-4.2790  
##  1st Qu.:0.000   1st Qu.: 0.00   1st Qu.: 0.000   1st Qu.:-0.1320  
##  Median :1.000   Median : 2.00   Median : 0.000   Median : 0.0000  
##  Mean   :1.512   Mean   : 1.88   Mean   : 1.478   Mean   :-0.2048  
##  3rd Qu.:2.000   3rd Qu.: 3.00   3rd Qu.: 2.000   3rd Qu.: 0.0000  
##  Max.   :9.000   Max.   :11.00   Max.   :20.000   Max.   : 2.7530  
##       V13             V14             V15              V16        
##  Min.   :1.544   Min.   :0.000   Min.   : 4.174   Min.   : 0.000  
##  1st Qu.:3.074   1st Qu.:0.881   1st Qu.: 9.459   1st Qu.: 0.000  
##  Median :3.414   Median :1.258   Median : 9.956   Median : 2.000  
##  Mean   :3.444   Mean   :1.402   Mean   : 9.892   Mean   : 3.522  
##  3rd Qu.:3.851   3rd Qu.:1.761   3rd Qu.:10.525   3rd Qu.: 6.000  
##  Max.   :5.069   Max.   :4.044   Max.   :12.421   Max.   :18.000  
##       V17             V18             V19               V20         
##  Min.   :0.959   Min.   :1.082   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.983   1st Qu.:1.121   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :1.003   Median :1.138   Median :0.00000   Median :0.00000  
##  Mean   :1.016   Mean   :1.137   Mean   :0.02392   Mean   :0.08134  
##  3rd Qu.:1.027   3rd Qu.:1.146   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.311   Max.   :1.377   Max.   :2.00000   Max.   :3.00000  
##       V21               V22             V23             V24         
##  Min.   :0.00000   Min.   :0.863   Min.   : 0.00   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:1.170   1st Qu.: 0.00   1st Qu.:0.00000  
##  Median :0.00000   Median :1.224   Median : 1.00   Median :0.00000  
##  Mean   :0.04785   Mean   :1.221   Mean   : 1.12   Mean   :0.04785  
##  3rd Qu.:0.00000   3rd Qu.:1.289   3rd Qu.: 1.00   3rd Qu.:0.00000  
##  Max.   :3.00000   Max.   :1.532   Max.   :10.00   Max.   :1.00000  
##       V25              V26               V27             V28           
##  Min.   :0.0000   Min.   :0.00000   Min.   :1.000   Min.   :-1.099000  
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:2.048   1st Qu.:-0.008000  
##  Median :0.0000   Median :0.00000   Median :2.242   Median : 0.000000  
##  Mean   :0.1148   Mean   :0.05742   Mean   :2.199   Mean   :-0.001407  
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:2.351   3rd Qu.: 0.005000  
##  Max.   :1.0000   Max.   :3.00000   Max.   :2.789   Max.   : 1.073000  
##       V29               V30              V31              V32         
##  Min.   :0.00000   Min.   : 0.000   Min.   : 0.444   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.: 0.000   1st Qu.: 1.481   1st Qu.:0.00000  
##  Median :0.00000   Median : 0.000   Median : 2.195   Median :0.00000  
##  Mean   :0.03349   Mean   : 9.075   Mean   : 2.806   Mean   :0.09569  
##  3rd Qu.:0.00000   3rd Qu.:11.846   3rd Qu.: 3.193   3rd Qu.:0.00000  
```

```
## Max.   :1.00000   Max.   :67.469   Max.   :12.745   Max.   :4.00000
##      V33               V34              V35           V36
## Min.   : 0.0000   Min.   : 0.000   Min.   :0.0   Min.   : 2.267
## 1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.:0.0   1st Qu.: 3.401
## Median : 0.0000   Median : 0.000   Median :1.0   Median : 3.694
## Mean   : 0.8038   Mean   : 1.411   Mean   :1.1   Mean   : 3.903
## 3rd Qu.: 1.0000   3rd Qu.: 2.000   3rd Qu.:2.0   3rd Qu.: 3.991
## Max.   :12.0000   Max.   :18.000   Max.   :6.0   Max.   :10.355
##      V37             V38              V39              V40
## Min.   :1.576   Min.   :0.0000   Min.   : 4.917   Min.   :0.00000
## 1st Qu.:2.146   1st Qu.:0.0000   1st Qu.: 7.872   1st Qu.:0.00000
## Median :2.469   Median :0.0000   Median : 8.464   Median :0.00000
## Mean   :2.629   Mean   :0.7464   Mean   : 8.574   Mean   :0.01914
## 3rd Qu.:2.967   3rd Qu.:1.0000   3rd Qu.: 9.017   3rd Qu.:0.00000
## Max.   :5.825   Max.   :6.0000   Max.   :14.030   Max.   :2.00000
##      V41              Class
## Min.   : 0.0000   Min.   :1.000
## 1st Qu.: 0.0000   1st Qu.:1.000
## Median : 0.0000   Median :1.000
## Mean   : 0.7895   Mean   :1.354
## 3rd Qu.: 0.0000   3rd Qu.:2.000
## Max.   :27.0000   Max.   :2.000
```

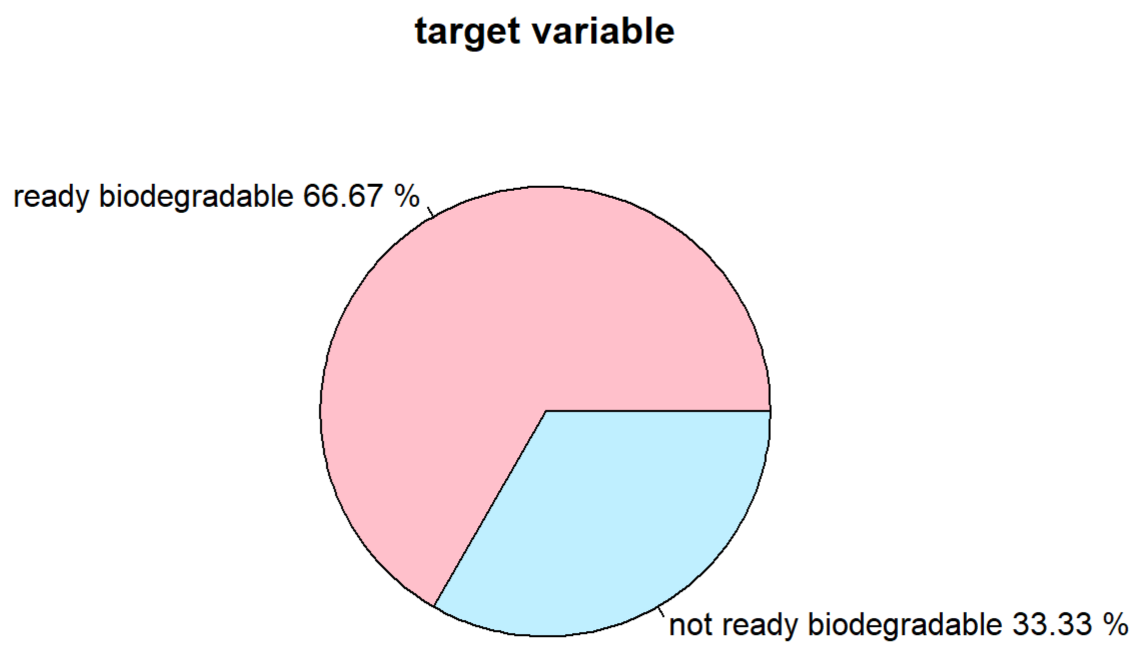
How balanced is the target variable?

Mean of target variable is 1.337 that is how we know there are more not ready biodegradable chemicals in our training dataset than ready biodegradable chemicals.

```
table(train$Class)
```

```
##
##   1   2
## 564 282
```

```
table_class <- (table(train$Class))
piepercent <- paste(round(100*table_class/sum(table_class), 2), "%")
names(table_class) <- c("ready biodegradable", "not ready biodegradable")
labels <- paste(names(table_class), piepercent)
pie(table_class, labels = labels, main = "target variable", col=c("pink", "lightblue1"))
```



Are there any missing values present? If there are, choose a strategy that takes this into account.

Instances with missing values only consist 7% of data and we decided that is insignificant and we removed them.

```
is.na(train)
```

```
## 738   5.475   0   0   0   0 3.347 2.048   0  8.242   0   0     1
## 742   1.542   0   0   0   0 3.232 3.000   1  7.151   0   0     1
## 752   7.261   0   0   0   0 3.663    NA   0  9.972   0   0     1
## 765   1.164   0   0   0   2 6.874 2.750   0 12.258   0   2     1
## 771   4.692   0   0   1   0 4.029 2.131   3  8.883   0   0     1
## 809   3.141   0   8   0   0 6.954    NA   0 13.369   0   6     1
## 810   1.951   0   1   2   2 3.708 2.773   1  8.322   0   0     1
## 829   1.511   0   1   0   0 4.706    NA   0  9.538   0   0     1
## 841   3.685   0   0   2   1 2.957    NA   1  7.078   0   0     2
## 844   2.094   0   0   0   0 3.058 2.800   0  6.889   0   0     2
## 846   3.667   0   0   0   0 3.402 2.985   0  8.054   0   0     2
## 856   3.779   0   0   0   2 3.755    NA   0  8.711   2   0     2
## 882   1.833   1   0   0   1 3.478    NA   0  7.975   0   0     2
## 899   1.740   1   0   0   0 3.454 1.817   0  7.920   0   0     2
## 938   1.740   0   1   0   0 3.881    NA   0  8.620   0   1     1
## 942   2.451   0   2   6   0 3.728 2.356   2  8.668   0   0     1
## 954   2.693   0   0   0   0 3.760    NA   0  8.680   0   0     1
## 956   1.075   0   1   0   0 4.009 2.101   0  8.805   0   1     1
## 957   1.339   0   1   0   0 3.880    NA   0  8.525   0   0     1
## 984   2.542   0   4   8   1 4.748 3.086   2  9.947   0   2     1
## 1005  3.430   0   5   2   0 4.016 2.781   1  9.395   0   2     1
## 1008  1.140   0   2   0   1 3.872 2.681   0  8.473   0   1     1
## 1046  1.544   0   2   0   1 3.865 2.352   0  8.506   0   0     1
```

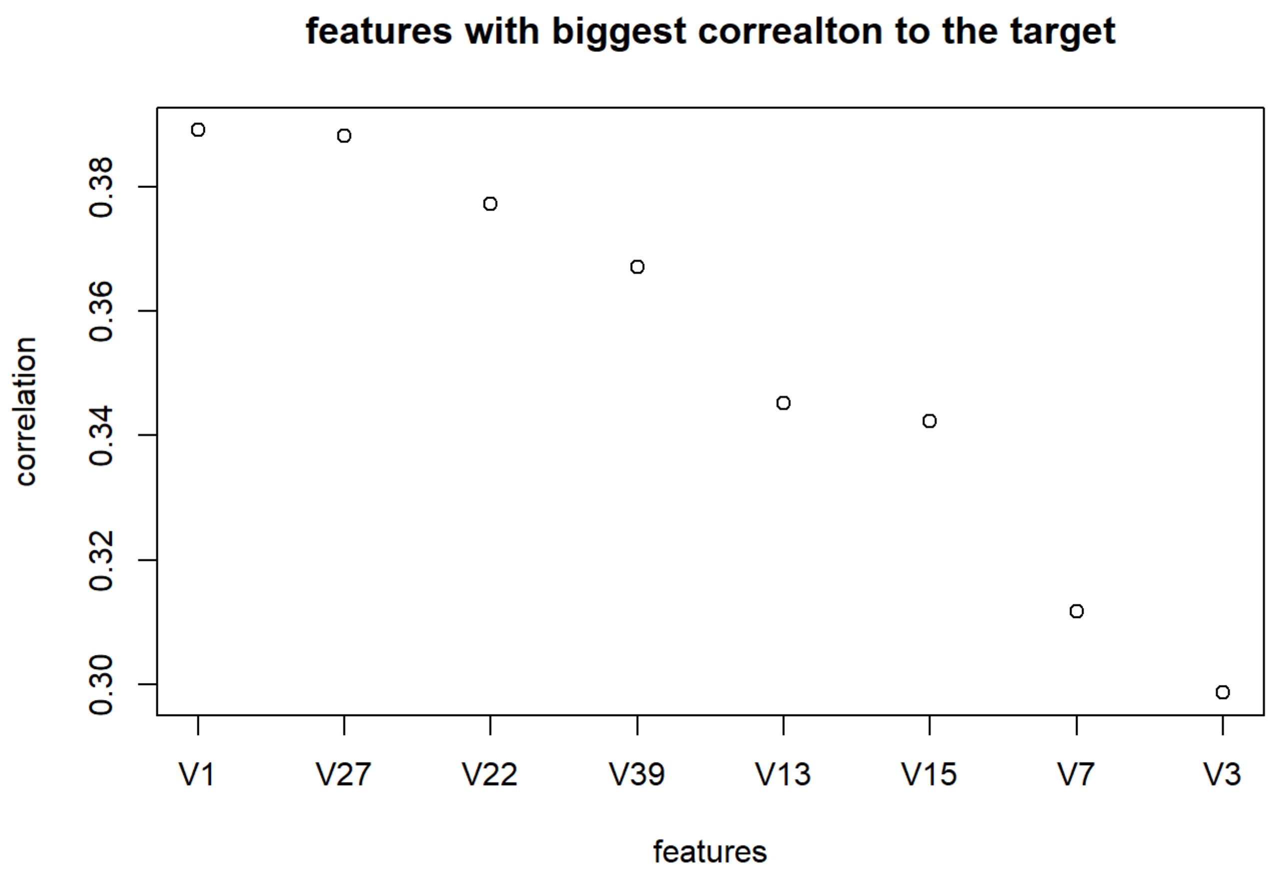
```
# 81/1055

#remove instances where one of values is missing
train <- na.omit(train)
#test <- na.omit(test)
```

First we normalized our data

```
process <- preProcess(train, method=c("range"))
train <- predict(process, train)
test <- predict(process, test)
```

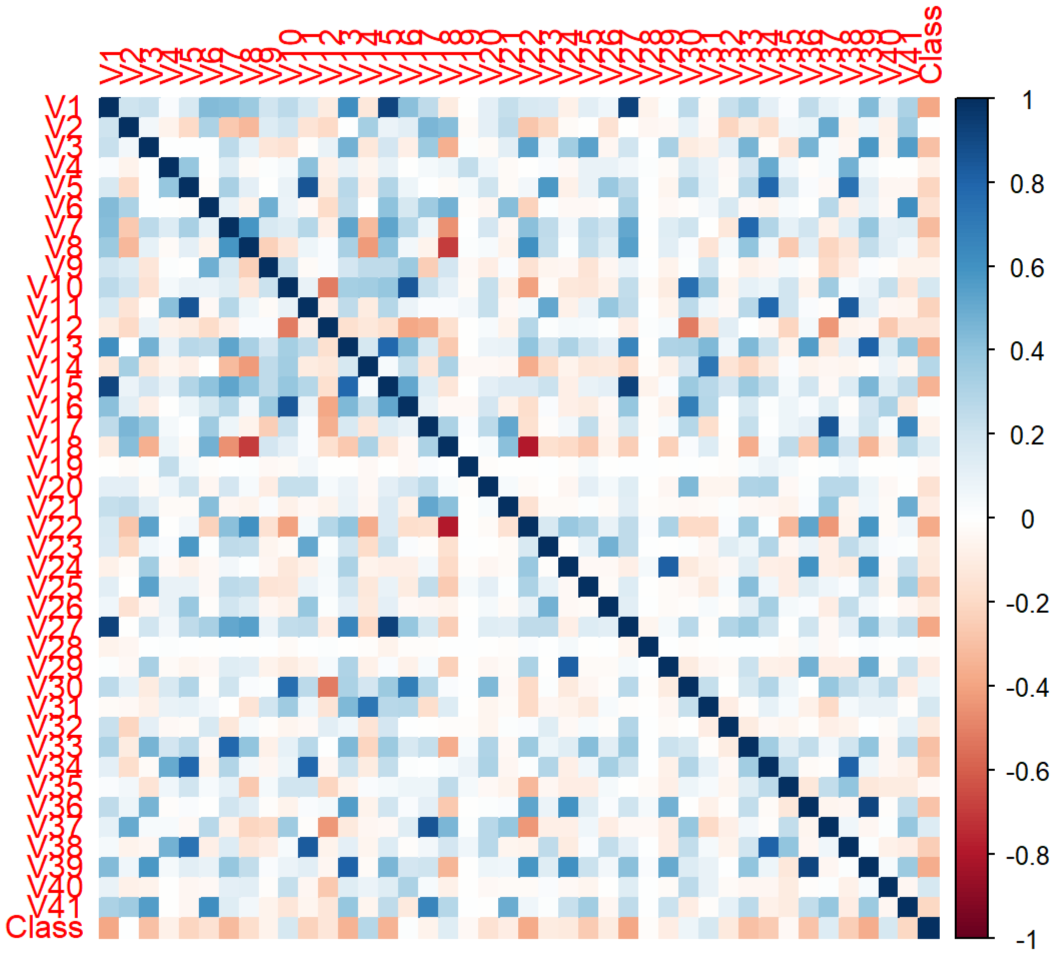
Most of your data is of the numeric type. Can you identify, by adopting exploratory analysis, whether some features are directly related to the target? What about feature pairs?

```
target_corr = abs(cor(train[,names(train)])[names(train)[42],])
target_corr <- sort(target_corr, decreasing = TRUE)
target_corr_top8 <- target_corr[2: 9]


plot(target_corr_top8, xlab="features", ylab = "correlation", main="features with biggest cor
realton to the target", xaxt = "n")
axis(1, at = c(1,2,3,4,5,6,7,8), labels = names(target_corr_top8))
```



Correlation matrix

```
correlation_matrix <- cor(train)

corrplot(correlation_matrix, method = 'color')
```

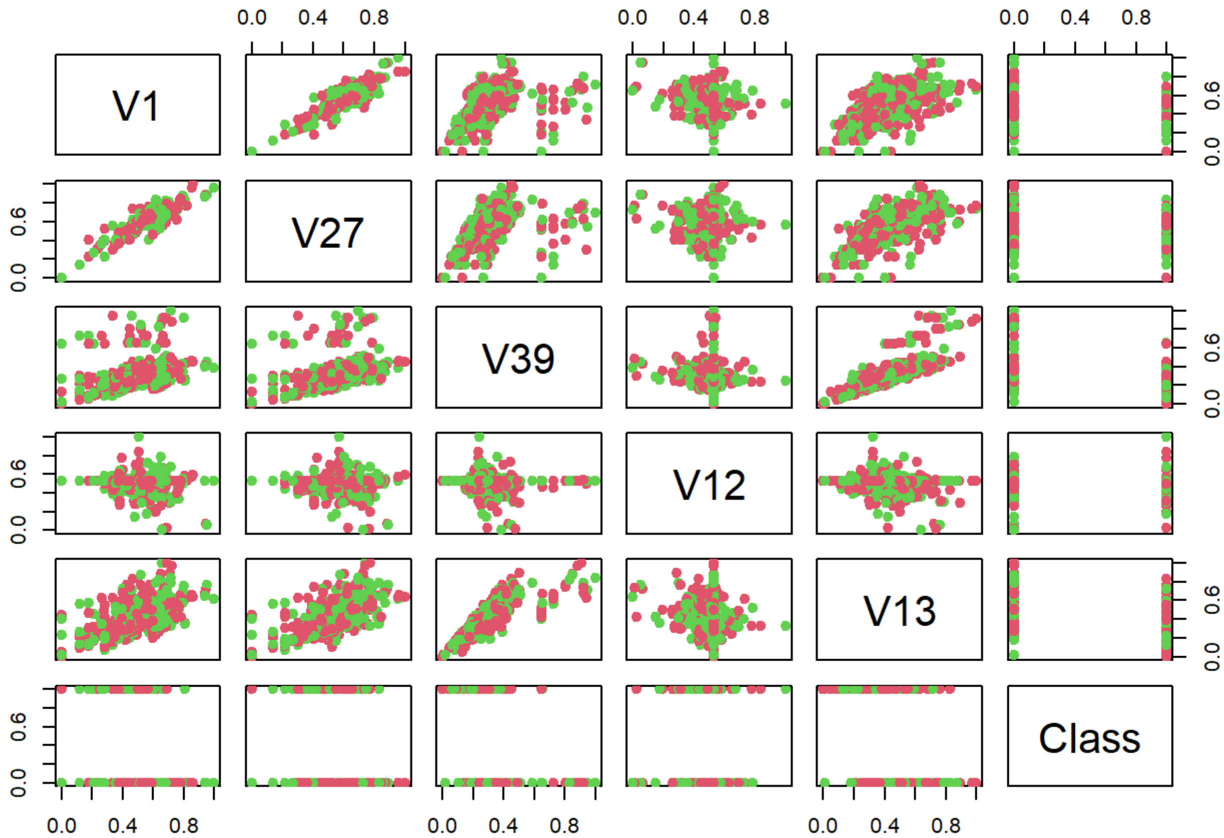


```
findCorrelation(correlation_matrix, names=TRUE, cutoff=0.9)
```

```
## [1] "V15" "V27" "V39"
```

Scatter matrix

```
colors = c("#00AFBB", "#E7B800")
pairs(train[c("V1", "V27", "V39", "V12", "V13", "Class")], pch=19, col=c(2, 3))
```

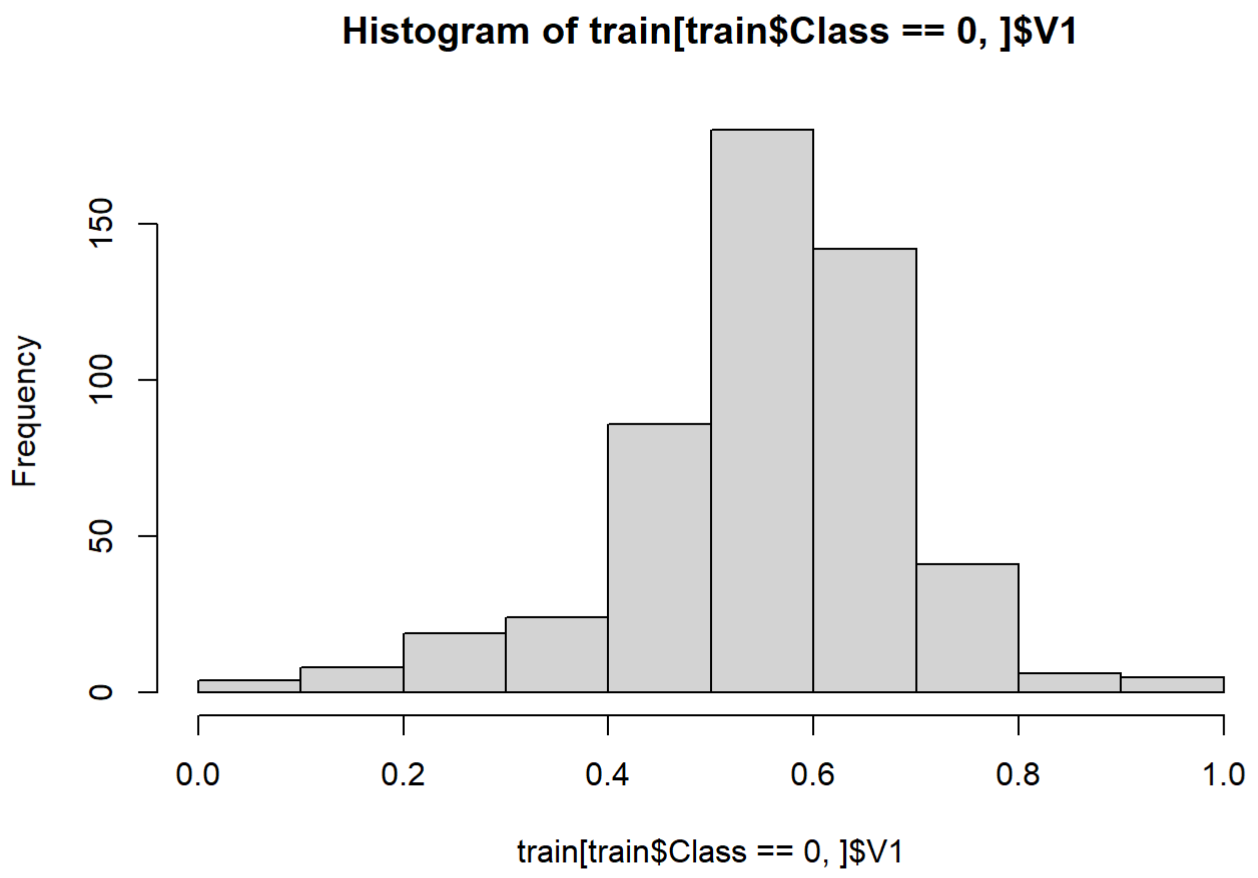



```
train1 = train[train$Class == 0,]
train2 = train[train$Class == 1,]
median1 <- apply(train1, 2, median)
median2 <- apply(train2, 2, median)
diff <- sort(abs(median1 - median2))
print(diff)
```

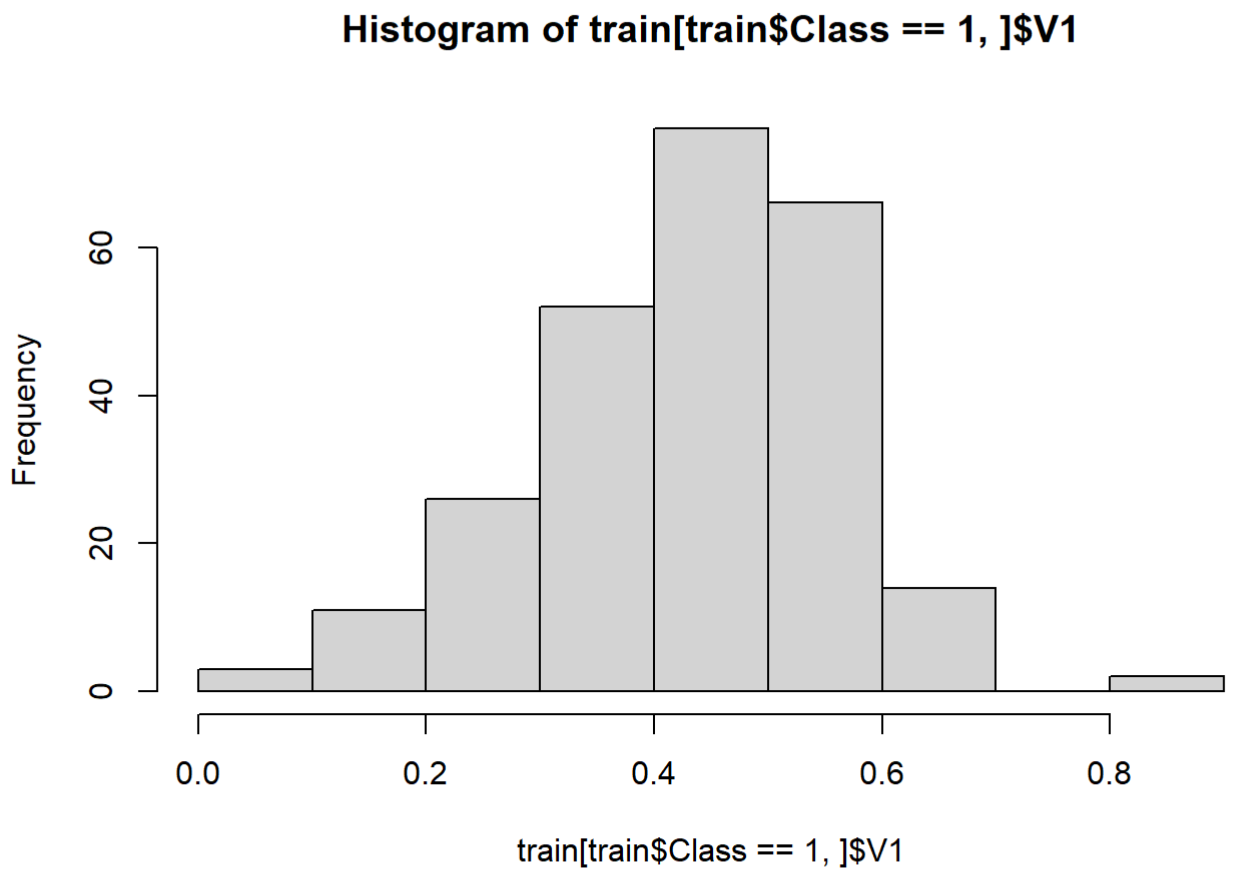
```
##          V3          V4          V5          V6          V9         V11
## 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
##         V12         V16         V19         V20         V21         V24
## 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
##         V25         V26         V28         V29         V32         V33
## 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
##         V34         V35         V38         V40         V41          V2
## 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.002038091
##         V17         V23         V31         V18         V37         V36
## 0.004335260 0.006802721 0.009081260 0.037900875 0.042260098 0.042779089
##         V14         V39         V10         V13         V15         V22
## 0.074593632 0.075907963 0.083333333 0.097433211 0.101080632 0.102288022
##          V7         V27          V1          V8         V30       Class
## 0.111111111 0.124221453 0.124570938 0.129844961 0.143886914 1.000000000
```

Distribution

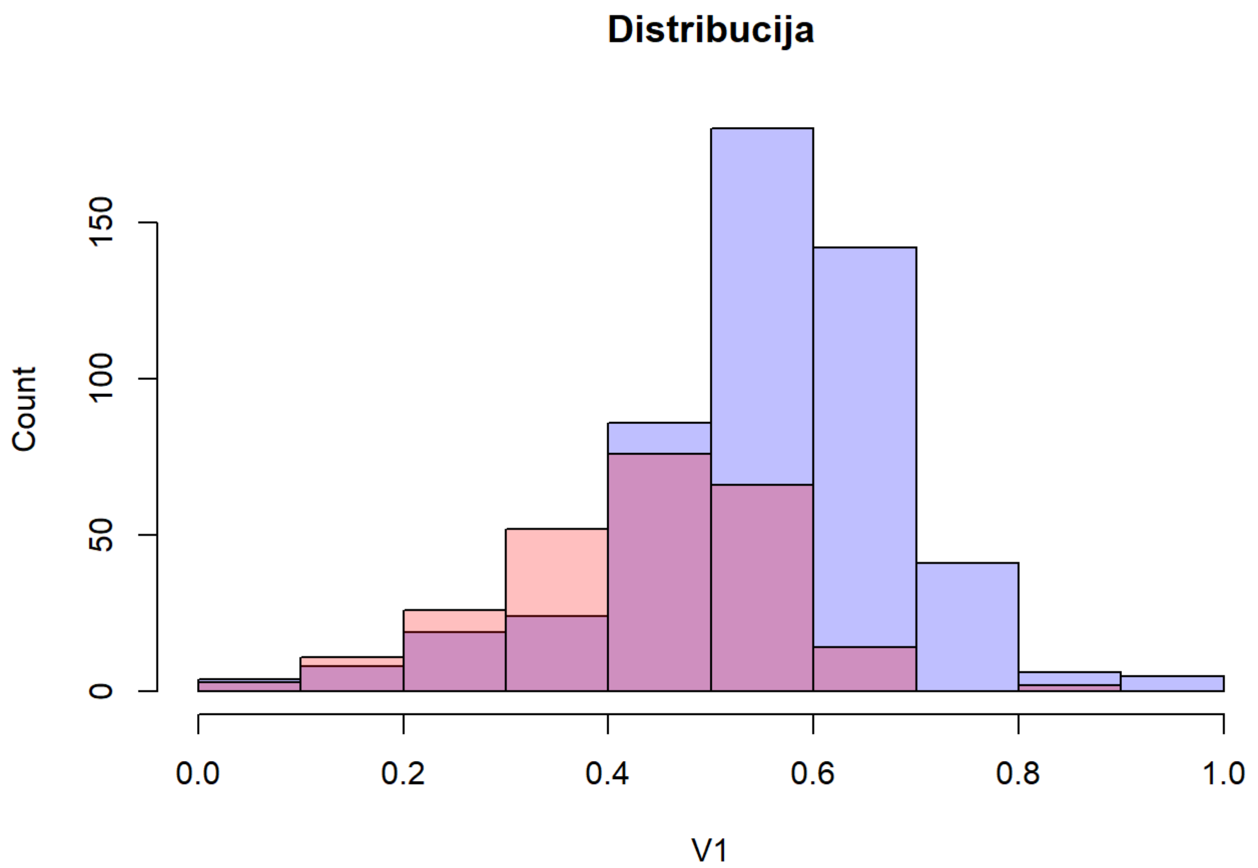
```
p1 <- hist(train[train$Class == 0,]$V1)
```




```
p2 <- hist(train[train$Class == 1,]$V1)                    # centered at 4
```

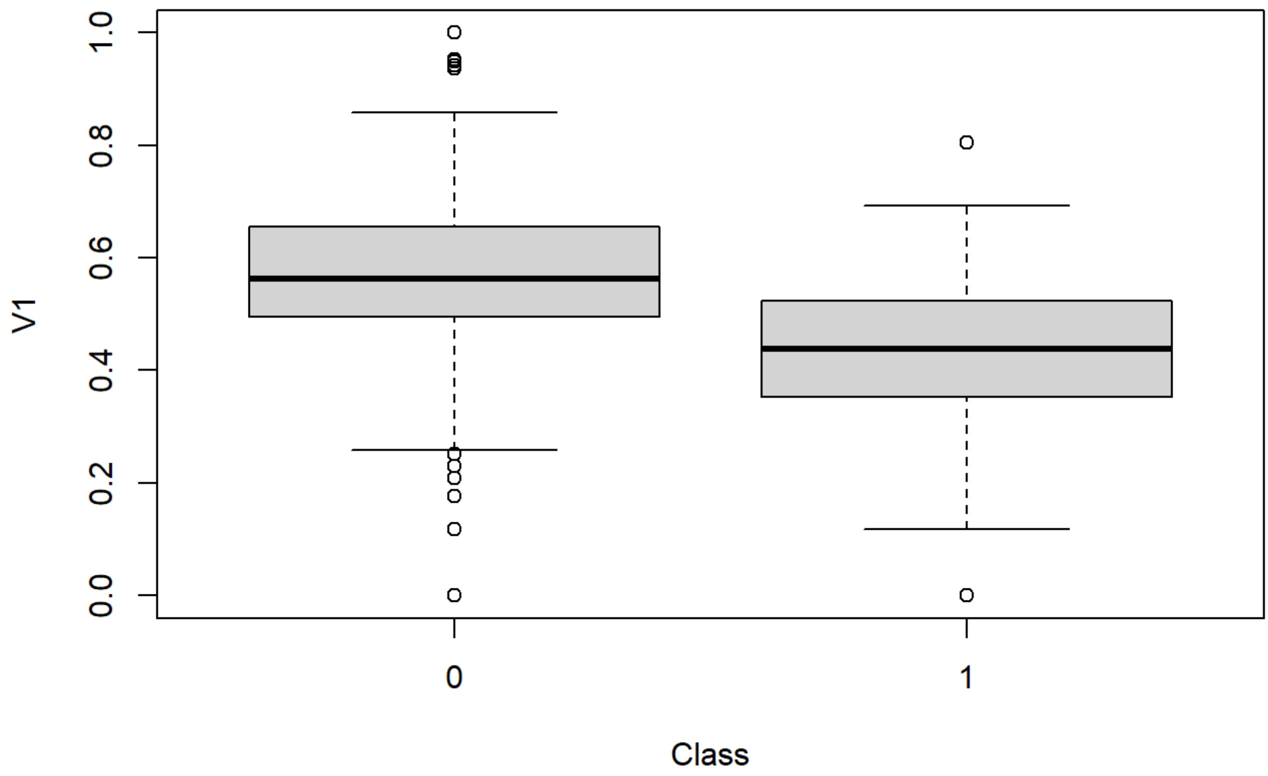


```
plot( p1, col=rgb(0,0,1,1/4), xlab = "V1", ylab = "Count", main="Distribucija")  # first hist
ogram
plot( p2, col=rgb(1,0,0,1/4), add=T)
```



Box plot for feature V1

```
boxplot(V1 ~ Class, train)
```



2 Modeling

Make target variable non numeric

```
train[train$Class == 0,]$Class <- "C0"
train[train$Class == 1,]$Class <- "C1"
test[test$Class == 0,]$Class <- "C0"
test[test$Class == 1,]$Class <- "C1"

train$Class <- as.character(train$Class)
train$Class <- as.factor(train$Class)

test$Class <- as.character(test$Class)
test$Class <- as.factor(test$Class)
```

Try to construct new features from existing ones.

We looked at attribute information and saw 3 attributes that looked similar:

V5: F04[C-N]: Frequency of C-N at topological distance 4 V11: F03[C-N]: Frequency of C-N at topological distance 3 V34: F02[C-N]: Frequency of C-N at topological distance 2

We decided to add a new feature that combines these three features and we added the sum of V5, V11 and V34.

```
train <- train %>% mutate(V42 = V5 + V11 + V34)
test <- test %>% mutate(V42 = V5 + V11 + V34)
```

feature selection

We used two feature selection methods ReliefFequalK and Information gain and from both we created subset of 15 best features.

```
feature_selection1 <- sort(attrEval(Class ~ ., train, "ReliefFequalK"), decreasing = TRUE)
selected_features1 <- c(names(head(feature_selection1, 15)))
selected_features1 <- append(selected_features1, "Class")
subset1 = train[selected_features1]
```

```
feature_selection2 <- sort(attrEval(Class ~ ., train, "InfGain"), decreasing = TRUE)
selected_features2 <- c(names(head(feature_selection2, 15)))
selected_features2 <- append(selected_features2, "Class")
subset2 = train[selected_features2]
```

majority classifier

```
majority.class <- names(which.max(table(train$Class)))
majority.class
```

```
## [1] "C0"
```

```
sum(train$Class == majority.class) / length(train$Class)
```

```
## [1] 0.6732026
```

random classifier

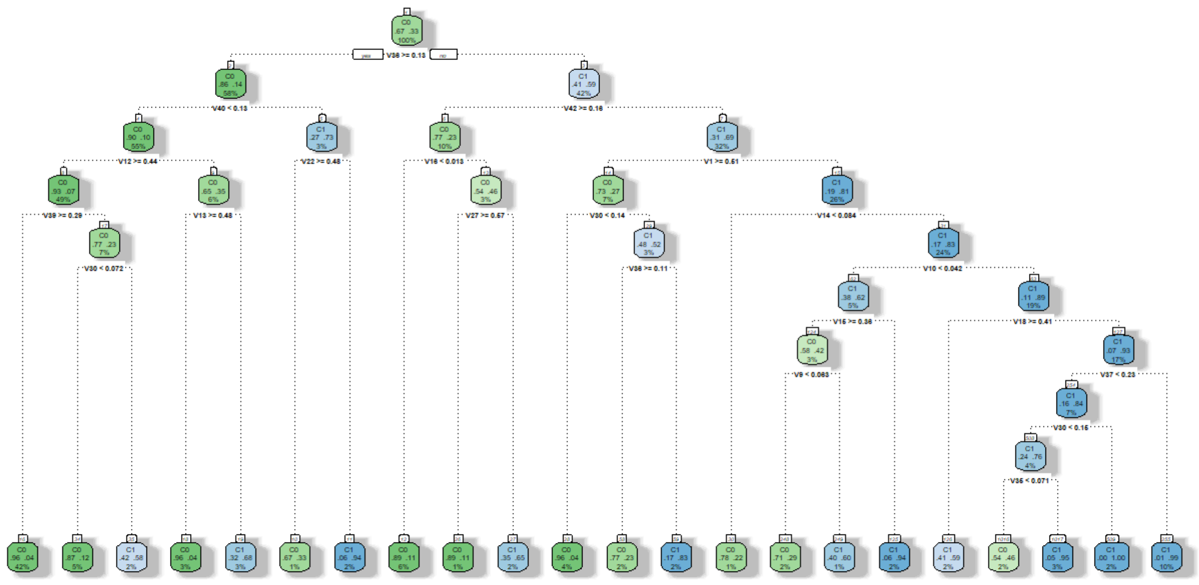
```
# Generate random predictions with equal probability for each class
predictions <- sample(c("C0", "C1"), size = nrow(train), replace = TRUE, prob = c(0.5, 0.5))
# Calculate the accuracy of the random classifier
accuracy <- mean(predictions == train$Class)
accuracy
```

```
## [1] 0.5150327
```

decision tree

Decision tree seemed like a good place to start. It is simple but insightful.

```
train_control <- trainControl(method = "repeatedcv", number = 5, repeats=10, classProbs=TRUE,
savePredictions = TRUE, summaryFunction = prSummary)

# Train the model using the train function and the tuneGrid argument
train.dt <- train(Class ~ ., data = train, tuneLength = 50,
          method = "rpart", metric = "AUC", trControl = train_control)
train.dt.score <- train.dt$results[1,]

#train.dt.roc <- roc(train$Class, predict(train.dt, train, type="prob")[,2])
fancyRpartPlot(train.dt$finalModel)
```



Rattle 2023-jan.-08 23:40:07 nikac

feature selection with decision tree

We used decision tree for out thid feature selection and also generated subset with 15 top features.

```
importance <- varImp(train.dt$finalModel, scale=FALSE)
importance_sorted <- arrange(importance, desc(Overall))

selected_features3 <- c(row.names(head(importance_sorted, 15)))
selected_features3 <- append(selected_features3, "Class")
subset3 = train[selected_features3]
```

decision tree with subsets

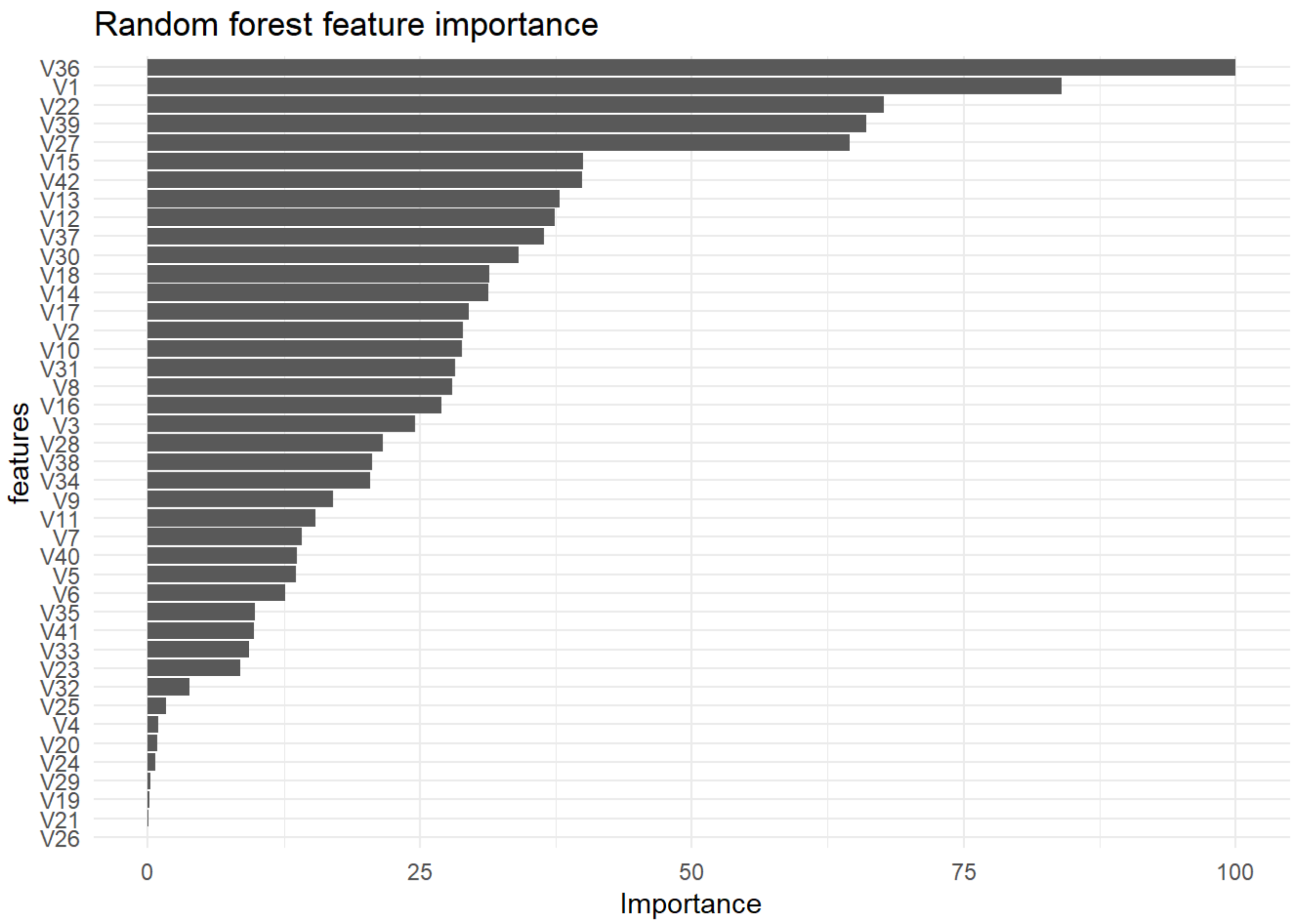
```
train.dtS3 <- train(Class ~ ., data = subset3, tuneLength = 50, method = "rpart", metric = "A
UC", trControl = train_control)
train.dtS3.score <- train.dtS3$results[1,]

#ReliefFequalK
train.dtS1 <- train(Class ~ ., data = subset1, tuneLength = 50, method = "rpart", metric = "A
UC", trControl = train_control)
train.dtS1.score <- train.dtS1$results[1,]

#infGain
train.dtS2 <- train(Class ~ ., data = subset2, tuneLength = 50, method = "rpart", metric = "A
UC", trControl = train_control)
train.dtS2.score <- train.dtS2$results[1,]
```

Random forest

```
set.seed(100)
tuneGrid <- expand.grid(.mtry = c(sqrt(ncol(train))))

trControl = trainControl(method='repeatedcv', number = 5, repeats = 10)
train.rf <- train(Class ~ ., data = train,method="rf",trControl=train_control,
                  metric='AUC', tuneGrid=tuneGrid)
train.rf.score <- train.rf$results[1,]
```

```
var_imp <- varImp(train.rf)
## Create a plot of variable importance
var_imp %>%
        ggplot(aes(x=reorder(variables, importance), y=importance)) +
        geom_bar(stat='identity') +
        coord_flip() +
        xlab('features') +
        labs(title='Random forest feature importance') +
        theme_minimal()
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

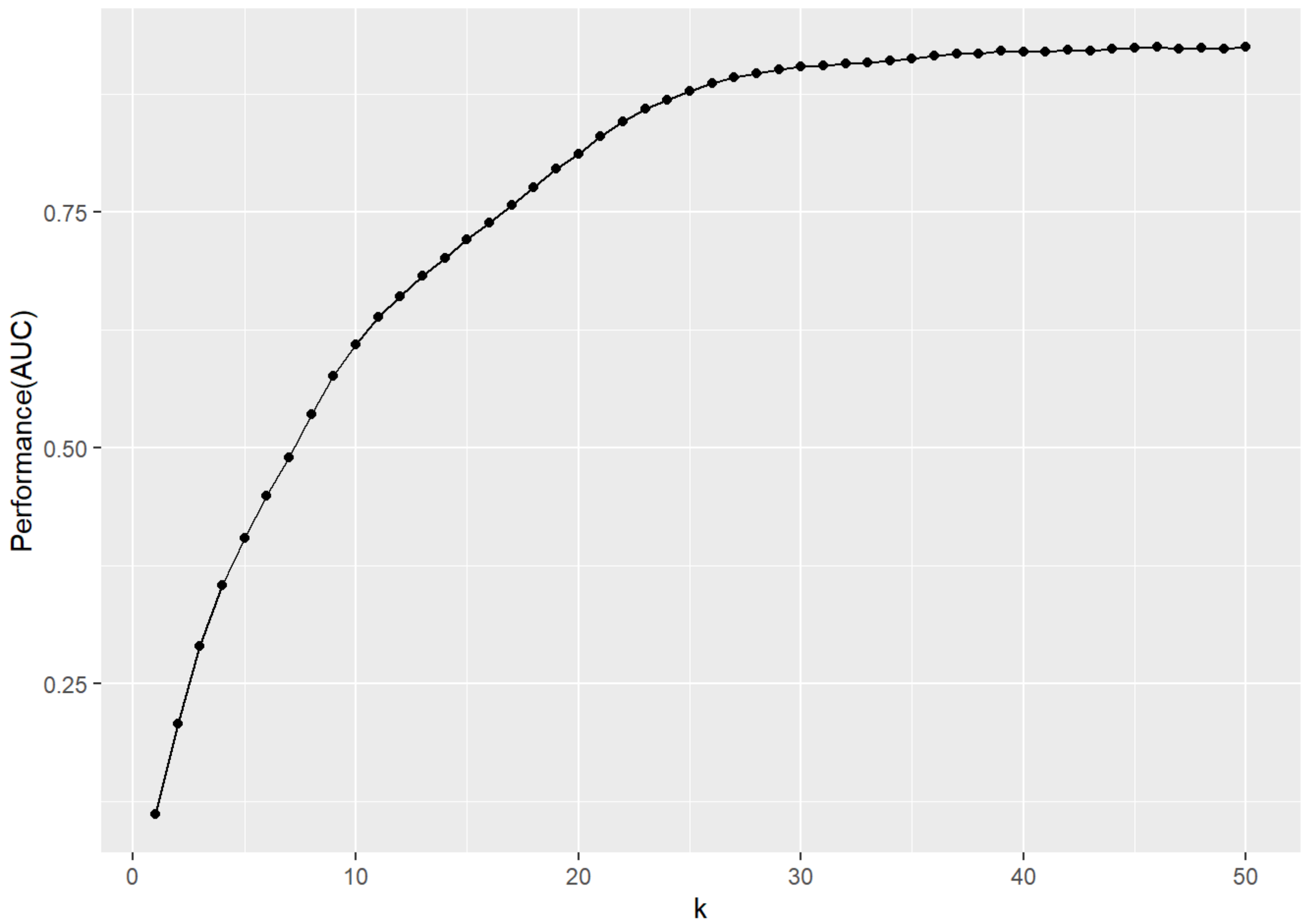


random forest with subsets

```
#ReliefFequalK
train.rfS1 <- train(Class ~ ., data = subset1, method = "rf", metric = "AUC", trControl = tra
in_control, tuneGrid=tuneGrid)
train.rfS1.score <- train.rfS1$results[1,]

#infGain
train.rfS2 <- train(Class ~ ., data = subset2, method = "rf", metric = "AUC", trControl = tra
in_control, tuneGrid=tuneGrid)
train.rfS2.score <- train.rfS2$results[1,]

train.rfS3 <- train(Class ~ ., data = subset3, method = "rf", metric = "AUC", trControl = tra
in_control, tuneGrid=tuneGrid)
train.rfS3.score <- train.rfS3$results[1,]
```

KNN

```
knitr::opts_chunk$set(warning = FALSE)

train$Class = factor(train$Class)
bigK = 50
grid = expand.grid(k = c(1:bigK))

train.knn <- train(Class ~., method= "knn", data = train,
                    trControl = train_control, metric="AUC",
                     tuneGrid = grid, preProcess = c("scale", "center"))

best_k = which.max(train.knn$results$AUC)
train.knn.score <- train.knn$results[best_k,]

qplot(1:bigK, train.knn$results$AUC, xlab = "k", ylab = "Performance(AUC)", geom = c("point",
"line"))
```

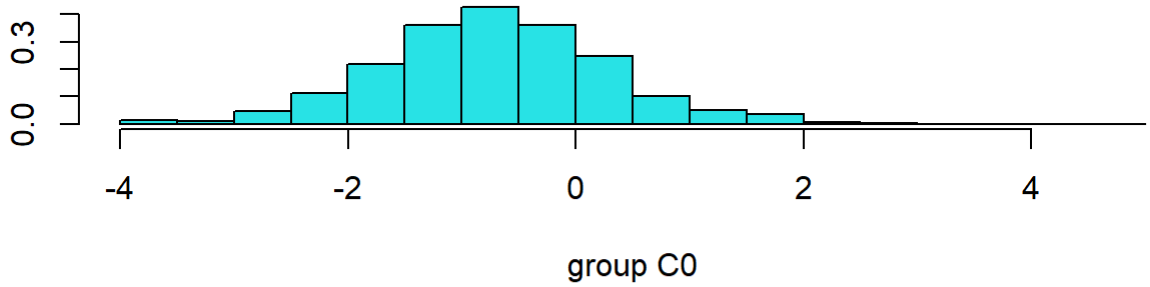


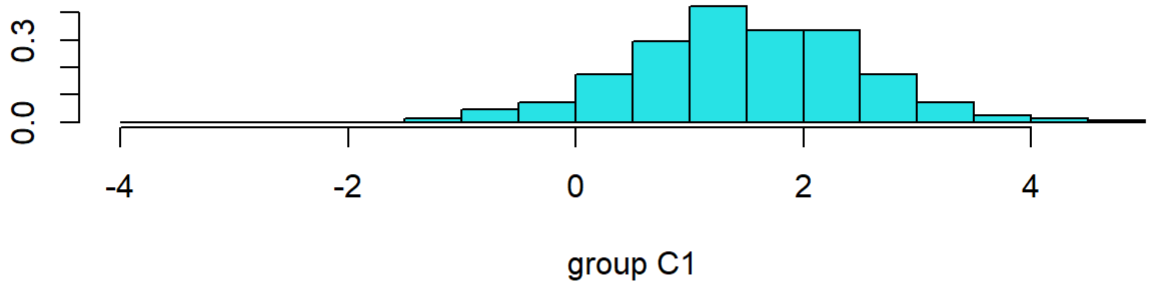
KNN with subsets

```
#ReliefFequalK
train.knnS1 <- train(Class ~ ., data = subset1, method = "knn", metric = "AUC", trControl = t
rain_control, tuneGrid=grid, preProcess = c("scale", "center"))
best_k = which.max(train.knnS1$results$AUC)
train.knnS1.score <- train.knnS1$results[best_k,]

#infGain
train.knnS2 <- train(Class ~ ., data = subset2, method = "knn", metric = "AUC", trControl = t
rain_control, tuneGrid=grid, preProcess = c("scale", "center"))
best_k = which.max(train.knnS2$results$AUC)
train.knnS2.score <- train.knnS2$results[best_k,]

train.knnS3 <- train(Class ~ ., data = subset3, method = "knn", metric = "AUC", trControl = t
rain_control, tuneGrid=grid, preProcess = c("scale", "center"))
best_k = which.max(train.knnS3$results$AUC)
train.knnS3.score <- train.knnS3$results[best_k,]
```

LDA classifier

```
train.lda <- lda(Class~., train)
train.lda.values <- predict(train.lda, train)
ldahist(train.lda.values$x[,1], g=train$Class)
```






```
train.lda <-  train(Class ~., method= "lda", data = train,
                    trControl = train_control, metric="AUC",
                     preProcess = c("scale", "center"))
train.lda.score <- train.lda$results[1,]
```

XGBoost

```
tune_grid <-  expand.grid(max_depth = c(3, 5, 7),
                        nrounds = (1:10)*50,    # number of trees
                        # default values below
                        eta = 0.3,
                        gamma = 0,
                        subsample = 1,
                        min_child_weight = 1,
                        colsample_bytree = 0.6)



train.xgboost <- train(Class ~ . , data = train, method = "xgbTree",
                trControl=train_control,
                tuneGrid = tune_grid,
                tuneLength = 10,
                verbosity = 0)
best_k = which.max(train.xgboost$results$AUC)
train.xgboost.score <- train.xgboost$results[best_k,]
```

XGBoost with subsets

```
#ReliefFequalK
train.xgboostS1 <- train(Class ~ ., data = subset1, method = "xgbTree", metric = "AUC", trCon
trol = train_control, tuneGrid=tune_grid, tuneLength = 10,verbosity = 0)
train.xgboostS1.score <- train.xgboostS1$results[1,]

#infGain
train.xgboostS2 <- train(Class ~ ., data = subset2, method = "xgbTree", metric = "AUC", trCon
trol = train_control, tuneGrid=tune_grid, tuneLength = 10,verbosity = 0)
train.xgboostS2.score <- train.xgboostS2$results[1,]

train.xgboostS3 <- train(Class ~ ., data = subset3, method = "xgbTree", metric = "AUC", trCon
trol = train_control, tuneGrid=tune_grid, tuneLength = 10,verbosity = 0)
train.xgboostS3.score <- train.xgboostS3$results[1,]
```

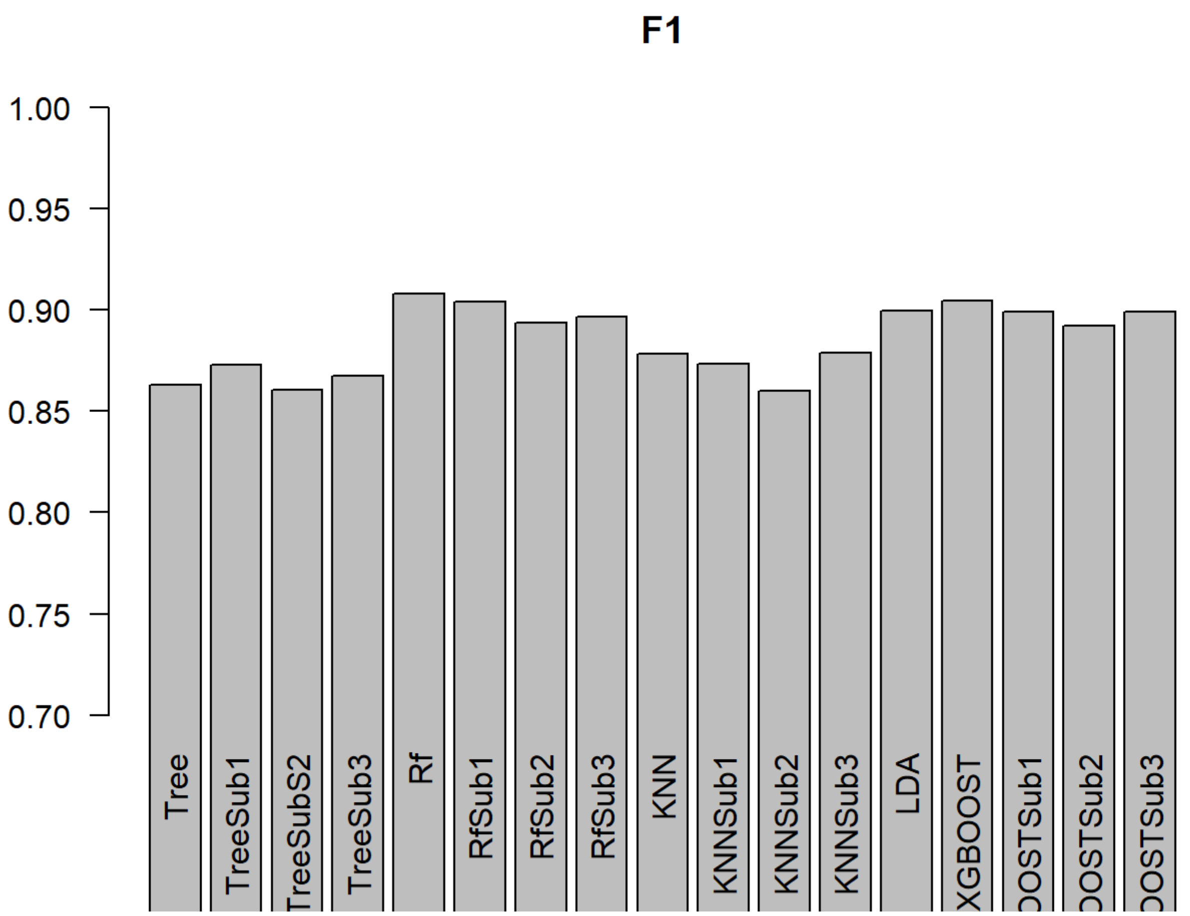
Evaluation

```
#tree
score <- train.dt.score
tree <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.dtS1.score
tree_S1 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.dtS2.score
tree_S2 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.dtS3.score
tree_S3 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

#random forest
score <- train.rf.score
rf <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.rfS1.score
rf_S1 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.rfS2.score
rf_S2 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.rfS3.score
rf_S3 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

#KNN
score <- train.knn.score
knn <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.knnS1.score
knn_S1 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.knnS2.score
knn_S2 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.knnS3.score
knn_S3 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

#XGBOOST
score <- train.xgboost.score
xgboost <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])
```

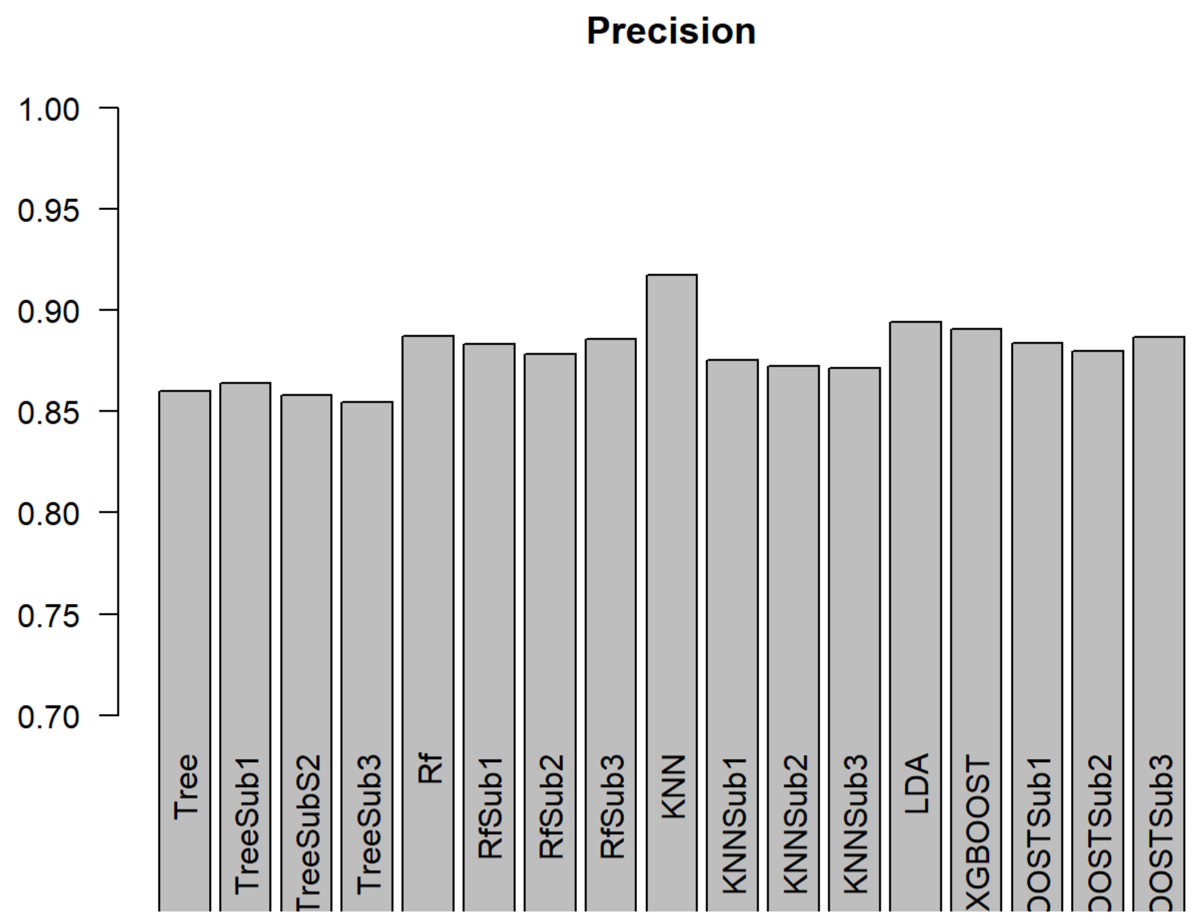
```
score <- train.xgboostS1.score
xgboost_S1 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"F
SD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.xgboostS2.score
xgboost_S2 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"F
SD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

score <- train.xgboostS3.score
xgboost_S3 <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"F
SD"],score[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])
#LDA
score <- train.lda.score
lda <- c(score[1,"F"],score[1,"Precision"],score[1,"Recall"],score[1,"AUC"],score[1,"FSD"],sc
ore[1,"PrecisionSD"], score[1,"RecallSD"], score[1,"AUCSD"])

x <- data.frame(row.names=c("F1", "PREC", "RECALL", "AUC", "F1STD", "PRECSTD", "RECALLSTD",
"AUCSTD"),
                Tree = tree, TreeSub1 = tree_S1, TreeSubS2 = tree_S2, TreeSub3 = tree_S3,
                Rf =  rf, RfSub1 = rf_S1, RfSub2 = rf_S2, RfSub3 = rf_S3,
                KNN = knn, KNNSub1 = knn_S1, KNNSub2 = knn_S2, KNNSub3 = knn_S3,
                LDA=lda,
                XGBOOST=xgboost, XGBOOSTSub1 = xgboost_S1, XGBOOSTSub2 = xgboost_S2, XGBOOSTS
ub3 = xgboost_S3 )

print(x)
```

```
##                 Tree   TreeSub1  TreeSubS2   TreeSub3         Rf     RfSub1
## F1         0.86301419 0.87307838 0.86077182 0.86757678 0.90809499 0.90417334
## PREC       0.85982084 0.86418515 0.85805992 0.85464456 0.88734154 0.88311638
## RECALL     0.86776699 0.88407767 0.86524272 0.88252427 0.93029126 0.92679612
## AUC        0.77279658 0.66545738 0.78842347 0.73739667 0.94274165 0.93012908
## F1STD      0.01927420 0.02132139 0.02158199 0.02423953 0.01786690 0.01500080
## PRECSTD    0.02526941 0.02879442 0.03153893 0.02747210 0.02276773 0.02165253
## RECALLSTD  0.03853111 0.04172015 0.03816744 0.04316329 0.02259430 0.02114470
## AUCSTD     0.16723878 0.19753172 0.13568492 0.17582759 0.02207009 0.02610762
##               RfSub2     RfSub3        KNN    KNNSub1    KNNSub2    KNNSub3
## F1         0.89348619 0.89677091 0.87829755 0.87329474 0.85995908 0.87859689
## PREC       0.87816076 0.88560286 0.91755225 0.87516516 0.87220201 0.87163009
## RECALL     0.91009709 0.90893204 0.84271845 0.87242718 0.84893204 0.88679612
## AUC        0.91578761 0.91763882 0.92515542 0.90445785 0.88657024 0.91426012
## F1STD      0.01849342 0.01902336 0.01991816 0.02325776 0.02345112 0.01766270
## PRECSTD    0.02658333 0.02323892 0.01905535 0.02377999 0.02601519 0.02748820
## RECALLSTD  0.02495068 0.02848444 0.02849187 0.03742281 0.03443245 0.02906737
## AUCSTD     0.03288027 0.03216470 0.02213207 0.03369592 0.03286742 0.02579334
##                  LDA    XGBOOST XGBOOSTSub1 XGBOOSTSub2 XGBOOSTSub3
## F1         0.89948837 0.90472478  0.89909081  0.89227384  0.89915836
## PREC       0.89437989 0.89054495  0.88357039  0.87992827  0.88651328
## RECALL     0.90518488 0.92000000  0.91592233  0.90582524  0.91262136
## AUC        0.94672285 0.94593095  0.94287742  0.93536282  0.94396890
## F1STD      0.02025204 0.01691955  0.01701521  0.02092448  0.01875119
## PRECSTD    0.02405164 0.02325809  0.02451236  0.02664059  0.02227251
## RECALLSTD  0.02672015 0.02430630  0.02516563  0.03052893  0.02473309
## AUCSTD     0.01120483 0.01872740  0.01573857  0.01915467  0.01423875
```

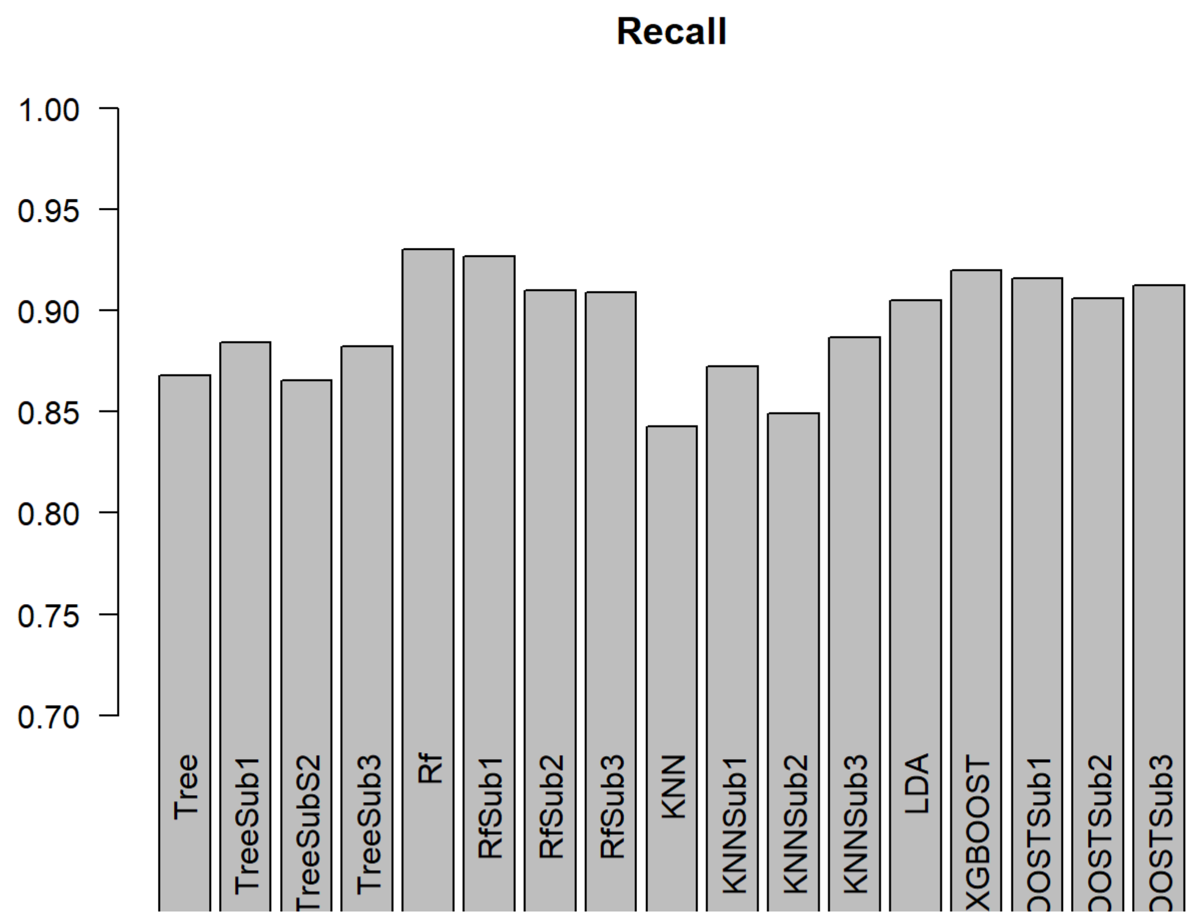
```
barplot(unlist(x["F1",]), names.arg=colnames(x), main="F1", ylim=c(0.7, 1),las = 2, cex.names
= 1)
```



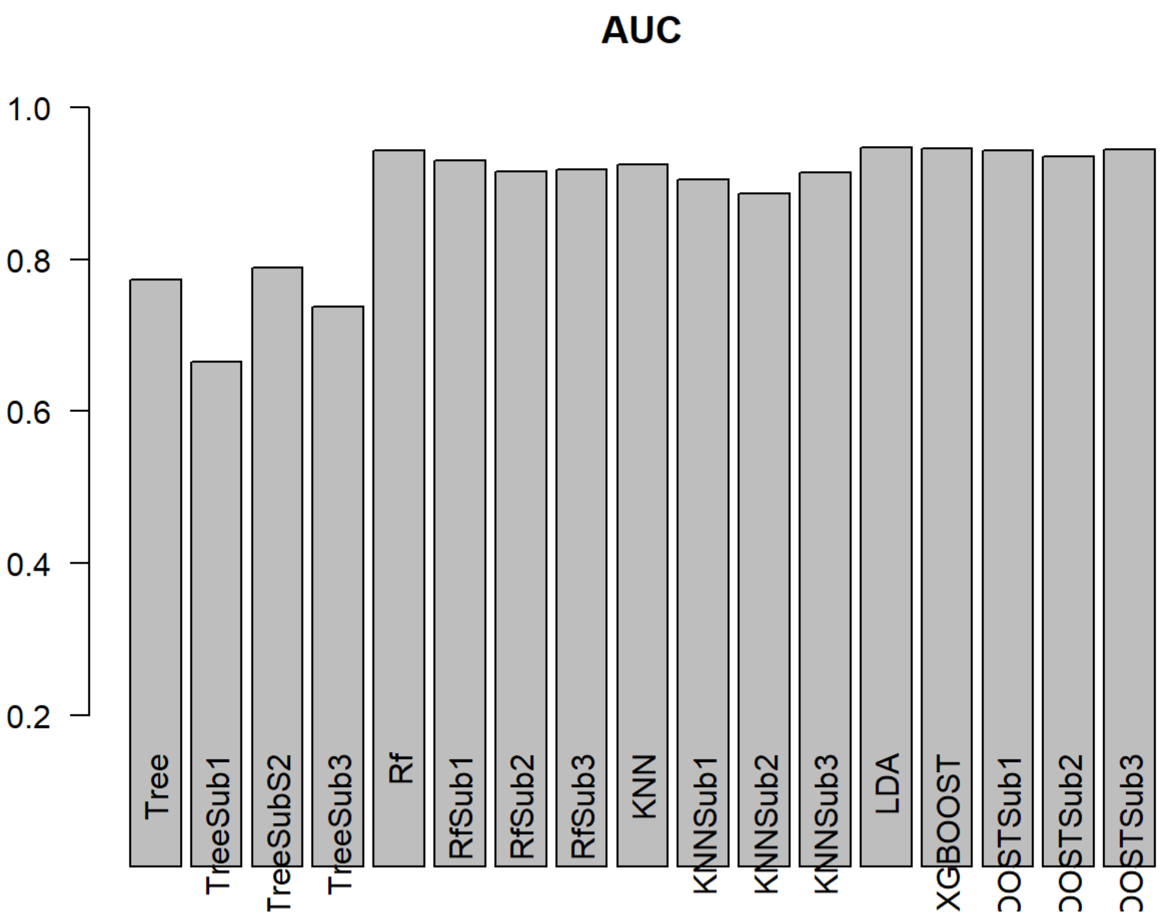

```
barplot(unlist(x["PREC",]), names.arg=colnames(x), main="Precision",  ylim=c(0.7, 1),las = 2,
cex.names = 1)
```




```
barplot(unlist(x["RECALL",]), names.arg=colnames(x), main="Recall",  ylim=c(0.7, 1),las = 2,
 cex.names = 1)
```



```
barplot(unlist(x["AUC",]), names.arg=colnames(x), main="AUC",  ylim=c(0.2, 1),las = 2, cex.na
mes = 1)
```

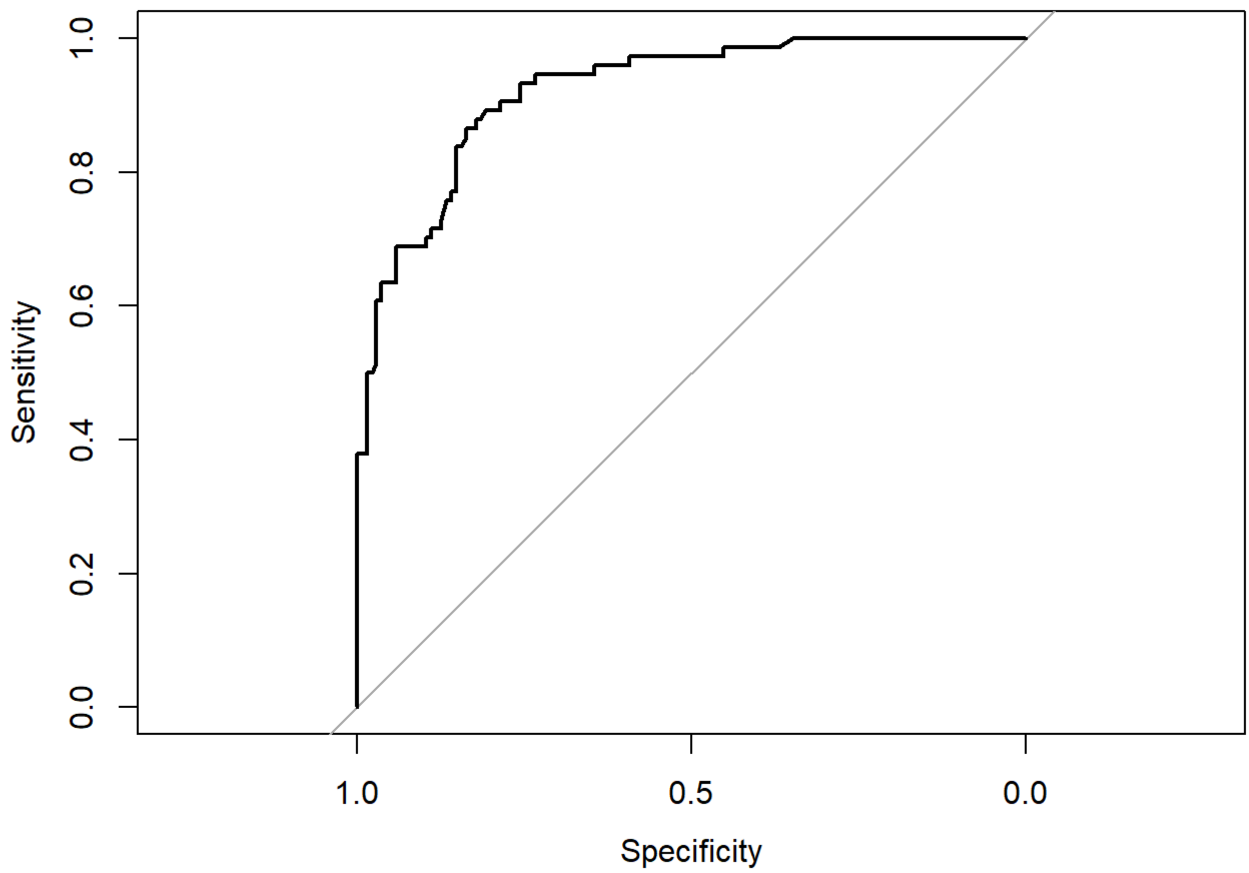


First we compared AUC scores: Random Forest and XGBOOST performed similar. Then we checked F1 score and random forest had slight advantage so we choose it for out final model.

```
p <- predict(train.rf, test)
confusion_matrix <- confusionMatrix(test$Class, p)
p <- predict(train.rf, test, type="prob")
r <- roc(test$Class, p[,1], plot=TRUE)
```

```
## Setting levels: control = C0, case = C1
```

```
## Setting direction: controls > cases
```




```
AUC <- r$auc
precision <- confusion_matrix$table[2,2] / (confusion_matrix$table[2,2] + confusion_matrix$ta
ble[2,1])
recall <- confusion_matrix$table[2,2] / (confusion_matrix$table[2,2] + confusion_matrix$table
[1,2])
f1_score <- confusion_matrix$F1
AUC
```

```
## Area under the curve: 0.922
```

```
precision
```

```
## [1] 0.7567568
```

```
recall
```

```
## [1] 0.7466667
```

```
f1_score
```

```
## NULL
```