Company Bankruptcy Prediction

Loncón Joaquín Danielo

March 24, 2022

Introduction

The objective of this project is to build a **Machine Learning Model** that predicts the **bankruptcy of a company**.

Some of the steps to follow are prepare and pre-process the data before starting to analyze. Use different machine learning algorithms and evaluate their performance. Ensemble the resulting models to have a better prediction. Optimize the model and finally run the model on new data to observe its performance.

The data that is going to be use to achieve this goal is the Bankruptcy data from the *Taiwan Economic Journal* for the years 1999–2009, this data is collected by Deron Liang and Chih-Fong Tsai, from the National Central University, Taiwan. See more about the data in the following link: https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction/metadata.

Prepare

Libraries required

```r
if(!require(dplyr)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(combinat)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(gbm)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(plyr)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(deepnet)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(glmnet)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(Matrix)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(mboost)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(MASS)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(pamr)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")

library(dplyr)
library(caret)
library(combinat)
library(rpart)
library(gbm)
library(plyr)
library(deepnet)
```

```
library(glmnet)
library(Matrix)
library(mboost)
library(MASS)
library(pamr)
library(randomForest)
```

Obtain the data

Download and unzip the data

```
temp <- tempfile()
download.file("https://github.com/joaquinloncon/Capstone/raw/main/Data.zip", destfile = temp)
dataframe <- read.csv(unzip(temp, "data.csv"))
names(dataframe)[1] <- "Bankrupt"
```

Pre-Process

Missing values

No value is missing in the dataframe, we can see this with the following simple code.

```
any(is.na(dataframe))
```

```
## [1] FALSE
```

Data partition for validation

Create a partition of the data to later test the final model.

```
set.seed(1, sample.kind = "Rounding") # just to make the code reproducible

validation_index <- createDataPartition(y = dataframe$Bankrupt, times = 1, p = 0.2, list = FALSE)

validation <- dataframe[validation_index,]
df <- dataframe[-validation_index,]
```

Identifying & Removing Predictors

Useless Predictors

Predictors with close to zero variation are removed.

```
nzv <- nearZeroVar(df)
nzv[!nzv %in% 1] #to avoid delete the variable of interest
```

```
## [1] 86 95
```

```
df <- df[,-nzv[!nzv %in% 1]]
```

Correlated Predictors

Note that there are predictors that provide almost the same information respect others (practically perfect correlated).

```
descrCor <-  cor(df)
sum(abs(descrCor[upper.tri(descrCor)]) > .999)
```

```
## [1] 10
```

Remove the predictors with absolute correlations above 0.75. Note that this value is arbitrary and could be changed.

```
highlyCorDescr <- findCorrelation(descrCor, cutoff = .75)
highlyCorDescr[!highlyCorDescr %in% 1]
```

```
##  [1]  3 86  2  4 20 24 44 23 38 39 18 19 62 91 67 79 56 89  5 41 46 94 65  9  8
## [26]  7 28 74
```

```
df <- df[,-highlyCorDescr[!highlyCorDescr %in% 1]]
```

Variable Format Changes

Make the variable of interest (Bankrupt) a factor and re-level the variable for later interpretation in the models, this makes that the bankruptcy level can be interpreted as $Y = 1$.

```
df$Bankrupt <- factor(df$Bankrupt, labels = c("non_bankruptcy", "bankruptcy"))
df$Bankrupt <- relevel(df$Bankrupt, "bankruptcy")

validation$Bankrupt <- factor(validation$Bankrupt, labels = c("non_bankruptcy", "bankruptcy"))
validation$Bankrupt <- relevel(validation$Bankrupt, "bankruptcy")
```

Train & Test sets

To train the models, test, and optimize, generate an index, with 80% of the data for training the model and 20% for testing (without using the validation set, which is going to be used at the end).

```
train_index <- createDataPartition(y = df$Bankrupt, times = 1, p = 0.8, list = FALSE)

train_set <- df[train_index,]
test_set <- df[-train_index,]
```

Analysis

Simple statistics

Let's start the analysis by looking at some summary statistics. Only a few variables are shown for space reasons

```
summary(df[,1:12])
```

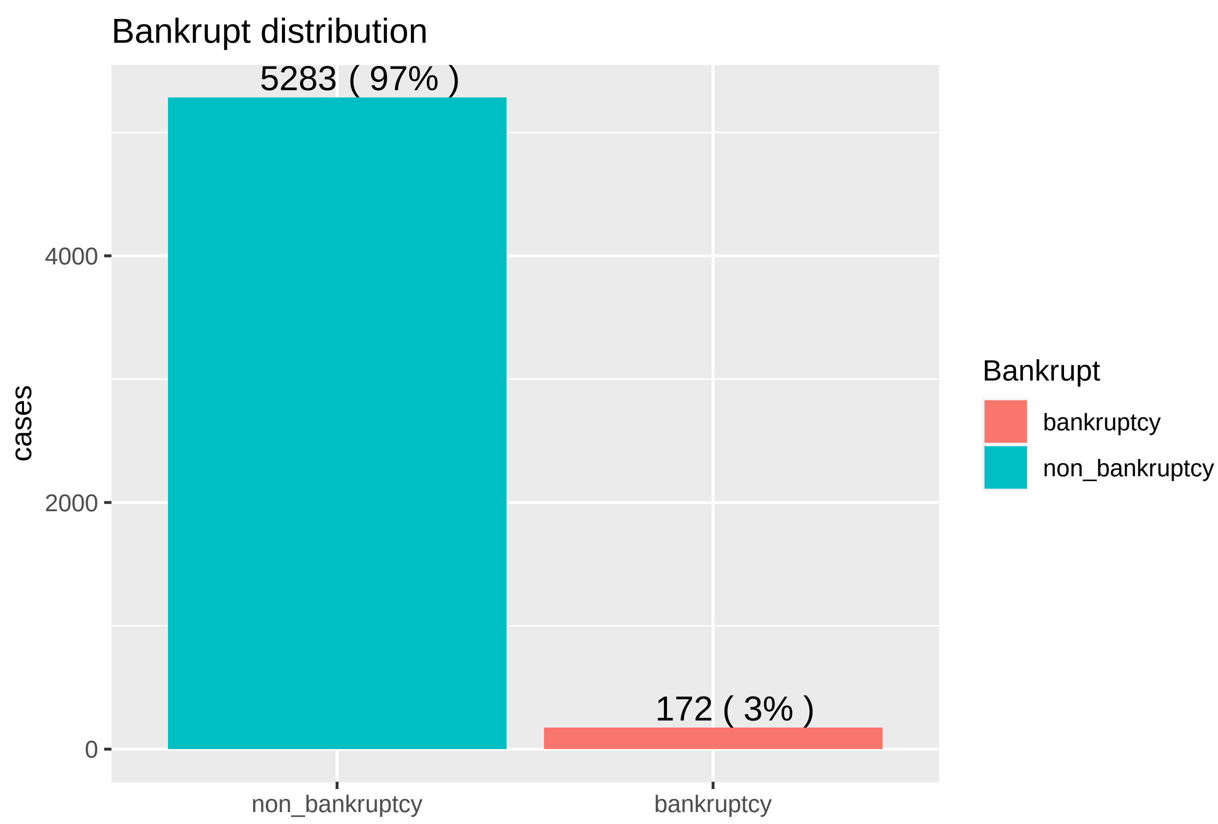
```
##           Bankrupt    Realized.Sales.Gross.Margin
##  bankruptcy    : 172   Min.   :0.0000
##  non_bankruptcy:5283   1st Qu.:0.6005
##                        Median :0.6060
##                        Mean   :0.6080
##                        3rd Qu.:0.6140
##                        Max.   :1.0000
##  Non.industry.income.and.expenditure.revenue
##  Min.   :0.0000
##  1st Qu.:0.3035
##  Median :0.3035
##  Mean   :0.3036
##  3rd Qu.:0.3036
##  Max.   :1.0000
##  Continuous.interest.rate..after.tax. Operating.Expense.Rate
##  Min.   :0.0000                       Min.   :0.000e+00
##  1st Qu.:0.7816                       1st Qu.:0.000e+00
##  Median :0.7816                       Median :0.000e+00
##  Mean   :0.7813                       Mean   :1.975e+09
##  3rd Qu.:0.7817                       3rd Qu.:4.070e+09
##  Max.   :0.8292                       Max.   :9.990e+09
##  Research.and.development.expense.rate Cash.flow.rate
##  Min.   :0.000e+00                      Min.   :0.0000
##  1st Qu.:0.000e+00                      1st Qu.:0.4615
##  Median :4.860e+08                      Median :0.4651
##  Mean   :1.937e+09                      Mean   :0.4674
##  3rd Qu.:3.380e+09                      3rd Qu.:0.4710
##  Max.   :9.980e+09                      Max.   :1.0000
##  Interest.bearing.debt.interest.rate  Tax.rate..A.     Net.Value.Per.Share..B.
##  Min.   :        0                   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:        0                   1st Qu.:0.00000   1st Qu.:0.1737
##  Median :        0                   Median :0.07298   Median :0.1848
##  Mean   : 16912741                   Mean   :0.11490   Mean   :0.1910
##  3rd Qu.:        0                   3rd Qu.:0.20544   3rd Qu.:0.2001
##  Max.   :990000000                   Max.   :0.99970   Max.   :1.0000
##  Cash.Flow.Per.Share Revenue.Per.Share..Yuan.Â..
##  Min.   :0.0000      Min.   :0.000e+00
##  1st Qu.:0.3177      1st Qu.:0.000e+00
##  Median :0.3225      Median :0.000e+00
##  Mean   :0.3234      Mean   :1.107e+06
##  3rd Qu.:0.3287      3rd Qu.:0.000e+00
##  Max.   :1.0000      Max.   :3.020e+09
```

Distributions and Relationships

Bankrupt distribution

This graph shows the high prevalence in the data, only 3% of the companies in the data have a bankruptcy and the other 97% do not.

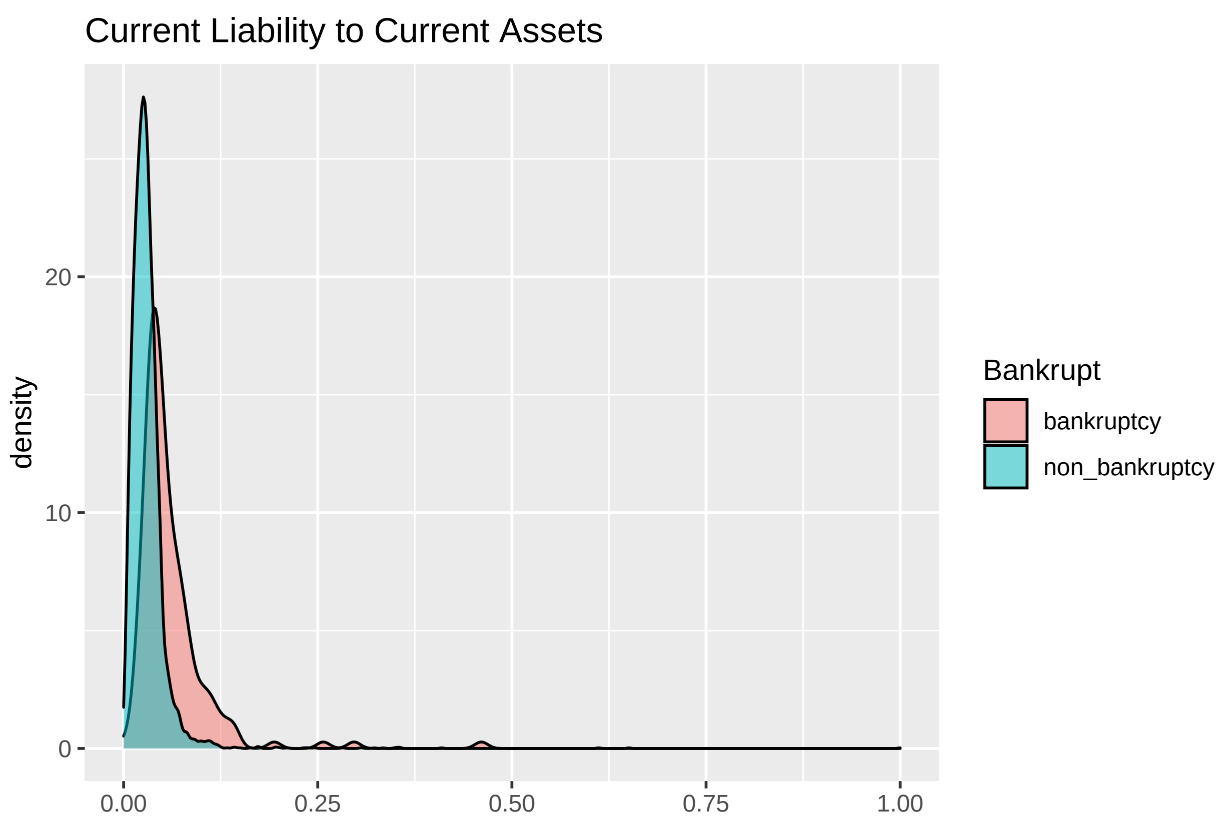
```
df %>%# Change the order just for aesthetics
    mutate(Bankrupt = relevel(df$Bankrupt, "non_bankruptcy"))%>%
    group_by(Bankrupt)%>%#dplyr:: must be added for confusion with plyr package
    dplyr::summarise(n = n())%>%
    mutate(percentage = n/sum(n)*100)%>%
    ggplot(aes(Bankrupt, n, fill=Bankrupt))+
    geom_bar(stat="identity")+
    geom_text(aes(label=n), vjust=-0.3, hjust = 1, size=4.5)+
    geom_text(aes(label=paste0("( ",round(percentage),"% )")), vjust=-0.3, hjust = -0.1, size=4.5)+
    ylab("cases")+
    scale_fill_manual(values = c("bankruptcy" = "#F8766D", "non_bankruptcy" = "#00BFC4"))+
    xlab("")+
    ggtitle("Bankrupt distribution")
```



Other relationships that will not be interpreted are shown below. But there is a marked difference between bankrupt and non-bankrupt companies.

Current Liability to Current Assets

```
df%>%
    group_by(Bankrupt)%>%
    dplyr::summarise(TAGR = Current.Liability.to.Current.Assets)%>%
    ggplot()+
    geom_density(aes(TAGR, group = Bankrupt, fill = Bankrupt), alpha = .5)+
    scale_fill_manual(values = c("bankruptcy" = "#F8766D", "non_bankruptcy" = "#00BFC4"))+
    xlab("")+
    ggtitle("Current Liability to Current Assets")
```

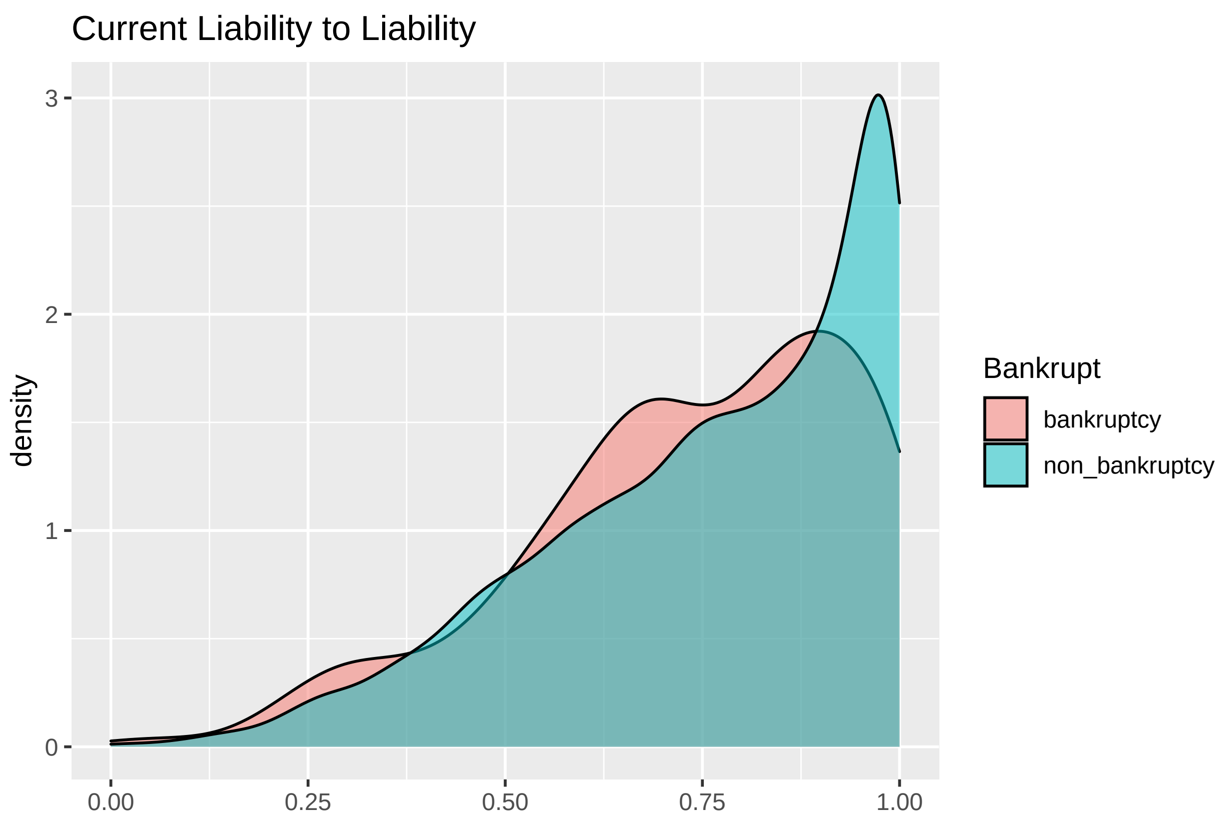


CFO to Assets

```
df%>%
    group_by(Bankrupt)%>%
    dplyr::summarise(CFOA = CFO.to.Assets)%>%
    ggplot()+
    geom_density(aes(CFOA, group = Bankrupt, fill = Bankrupt), alpha = .5)+
    scale_fill_manual(values = c("bankruptcy" = "#F8766D", "non_bankruptcy" = "#00BFC4"))+
    xlab("")+
    ggtitle("CFO to Assets")
```

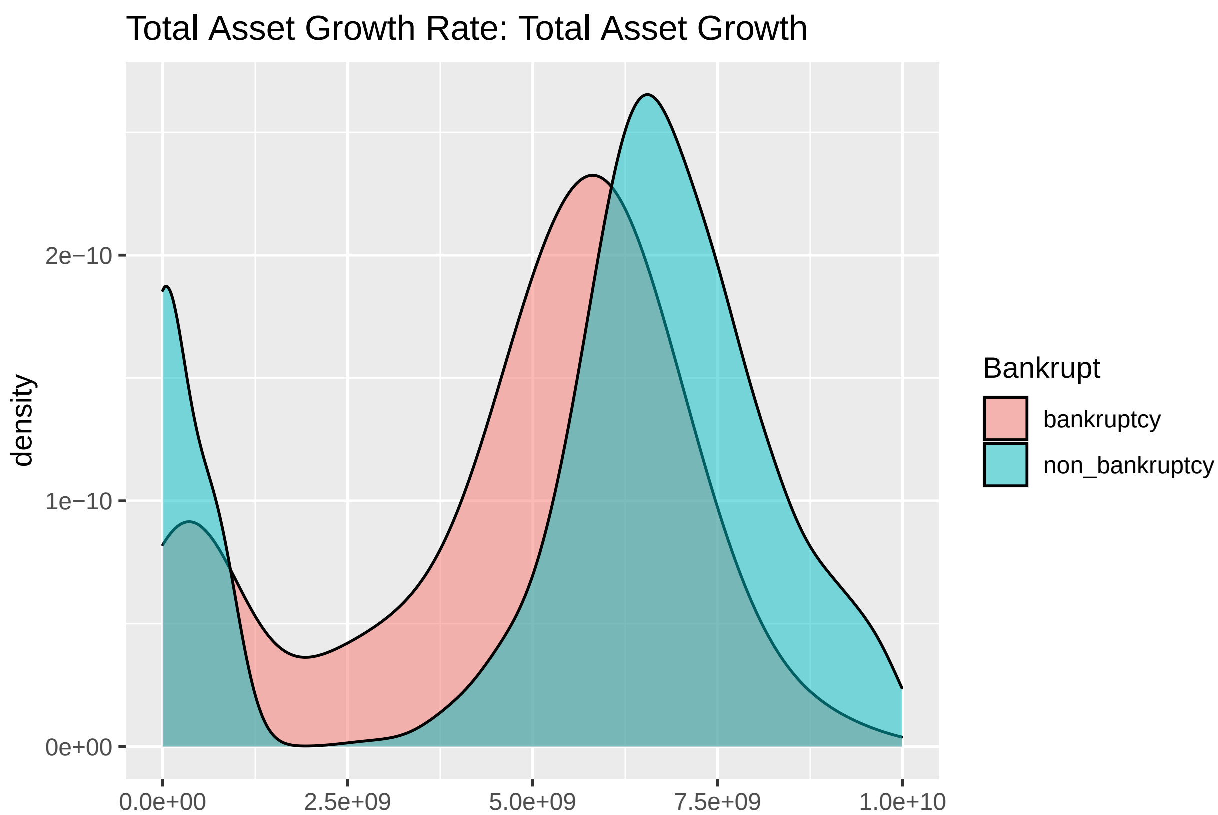
Current Liability to Liability

```
df%>%
    group_by(Bankrupt)%>%
    dplyr::summarise(TAGR = Current.Liability.to.Liability)%>%
    ggplot()+
    geom_density(aes(TAGR, group = Bankrupt, fill = Bankrupt), alpha = .5)+
    scale_fill_manual(values = c("bankruptcy" = "#F8766D", "non_bankruptcy" = "#00BFC4"))+
    xlab("")+
    ggtitle("Current Liability to Liability")
```



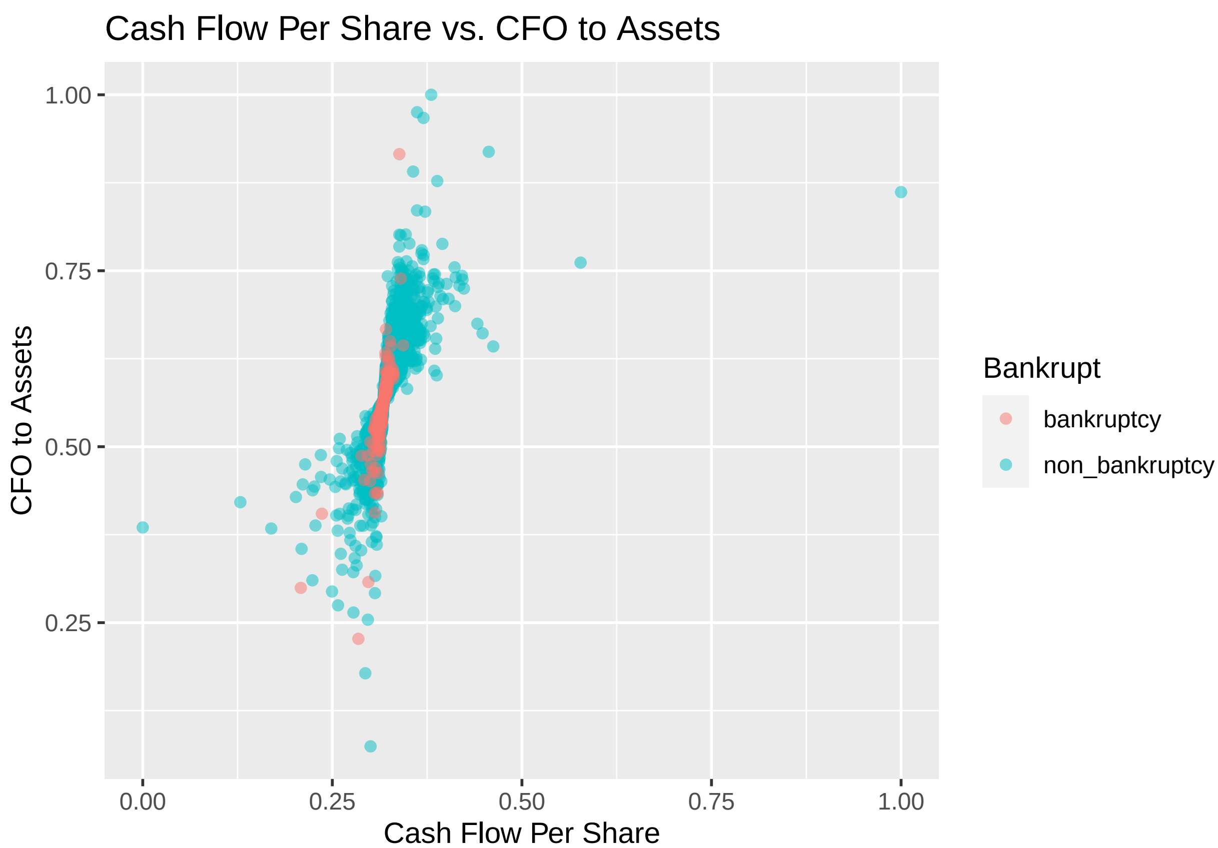
Total Asset Growth Rate: Total Asset Growth

```
df%>%
    group_by(Bankrupt)%>%
    dplyr::summarise(TAGR = Total.Asset.Growth.Rate)%>%
    ggplot()+
    geom_density(aes(TAGR, group = Bankrupt, fill = Bankrupt), alpha = .5)+
    scale_fill_manual(values = c("bankruptcy" = "#F8766D", "non_bankruptcy" = "#00BFC4"))+
    xlab("")+
    ggtitle("Total Asset Growth Rate: Total Asset Growth")
```



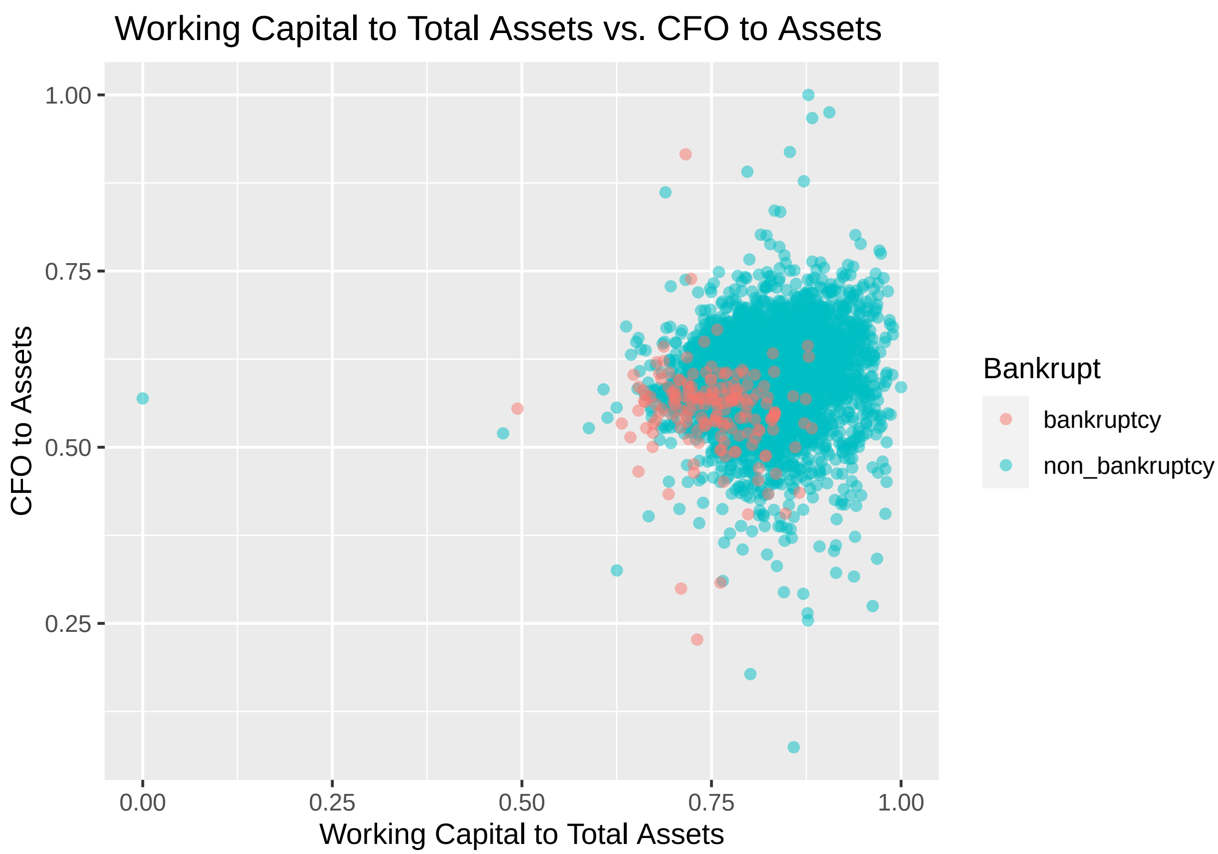
Cash Flow Per Share vs. CFO to Assets

```
df %>%# Change the order just to see better the graph
    mutate(Bankrupt = relevel(df$Bankrupt, "non_bankruptcy"))%>%
    group_by(Bankrupt)%>%
    dplyr::summarise(A =  Cash.Flow.Per.Share, B = CFO.to.Assets)%>%
    ggplot()+
    geom_point(aes(A,B, color = Bankrupt),alpha = .5)+
    scale_color_manual(values = c("bankruptcy" = "#F8766D", "non_bankruptcy" = "#00BFC4"))+
    xlab("Cash Flow Per Share")+
    ylab("CFO to Assets")+
    ggtitle("Cash Flow Per Share vs. CFO to Assets")
```



Working Capital to Total Assets vs. CFO to Assets

```
df %>%
    mutate(Bankrupt = relevel(df$Bankrupt, "non_bankruptcy"))%>%
    group_by(Bankrupt)%>%
    dplyr::summarise(A =  Working.Capital.to.Total.Assets, B = CFO.to.Assets)%>%
    ggplot()+
    geom_point(aes(A,B, color = Bankrupt),alpha = .5)+
    scale_color_manual(values = c("bankruptcy" = "#F8766D", "non_bankruptcy" = "#00BFC4"))+
    xlab("Working Capital to Total Assets")+
    ylab("CFO to Assets")+
    ggtitle(" Working Capital to Total Assets vs. CFO to Assets")
```



Analysis Summary

We see that the companies that go to bankruptcy have different distributions regarding some predictors, in some cases this difference is minimal, and in other is clearer, this differences would help the algorithms to predict the Bankrupt, also we can note that the relations between the predictors could not be linear. In addition to this, we have the problem of having few observation of companies bankrupted compared to those that are not, so this is a problem at the time of create a model because it is easier to predict non-bankruptcy due to the prevalence, which would give us a very high accuracy, but the model would be useless, since can't predict the bankruptcy. The processes and techniques that will be used to solve this problem are explained below.

Model Building

Prevalence Problem

As shown above, there is a high prevalence of one class in the bankruptcy variable. This brings big problems when creating a model, as seen below a classification tree is trained and its performance on the test set is displayed.

```
fit_bad_model <- train(Bankrupt~.,
                      data = train_set,
                      method = "rpart",
                      trControl = trainControl(method = "cv", number = 10))
```

```r
hat_bad_model <- predict(fit_bad_model, newdata = test_set)

confusionMatrix(data = hat_bad_model, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy              0              0
##   non_bankruptcy         34           1056
##
##                Accuracy : 0.9688
##                  95% CI : (0.9567, 0.9783)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 0.5454
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.519e-08
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.96881
##              Prevalence : 0.03119
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : bankruptcy
##
```

The problem is that because bankruptcy is so rare, the model only predicts non-bankruptcy. Thus it has a high accuracy, but as one can see the sensitivity is 0, which is useless since the objective is to predict bankruptcy. As it does not predict $Y = 1$ so the specificity is equal to 1.

Up-Sampling

To solve this problem we can do a technique called up-sampling. this method increases the size of the minority class by sampling with replacement so that the classes in the data will have the same size. Note that by doing this, the observations will balance out, causing there to be no more prevalence.

```r
trainup <- upSample(y=train_set$Bankrupt,
                    x=train_set[,-1],
                    yname = "Bankrupt")

table(trainup$Bankrupt)
```

```
##
##     bankruptcy non_bankruptcy
##           4227           4227
```

Machine Learning Models

Now with the balanced data we are going to train a set of different models ranging from classification trees and random forests to deep neural networks and observe their performances on the test set. Predictors will be centered and scaled by the preProcess option. In addition, we are going to use a 10-fold cross-validation, this is randomly split the observations of our train set into 10 non-overlapping sets, and do the calculation of the Mean Squared Error (MSE) for each of these sets having into account the different parameters of the model that lead us to that MSE, then the model selects the optimal parameters, the ones that minimize the MSE.

Classification Tree 1

```
fit_rpart <- train(Bankrupt ~ .,
                   data = trainup,
                   method = "rpart",
                   trControl = trainControl(method = "cv", number = 10),
                   preProcess = c("center", "scale"))

hat_rpart <- predict(fit_rpart, newdata = test_set)

confusionMatrix(data = hat_rpart, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             22             73
##   non_bankruptcy         12            983
##
##                Accuracy : 0.922
##                  95% CI : (0.9045, 0.9372)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3094
##
##  Mcnemar's Test P-Value : 7.62e-11
##
##             Sensitivity : 0.64706
##             Specificity : 0.93087
##          Pos Pred Value : 0.23158
##          Neg Pred Value : 0.98794
##              Prevalence : 0.03119
##          Detection Rate : 0.02018
##    Detection Prevalence : 0.08716
##       Balanced Accuracy : 0.78897
##
##        'Positive' Class : bankruptcy
##
```

Classification Tree 2

```
fit_rpart2 <- train(Bankrupt ~ .,
                   data = trainup,
                   method = "rpart2",
                   trControl = trainControl(method = "cv", number = 10),
                   preProcess = c("center", "scale"))

hat_rpart2 <- predict(fit_rpart2, newdata = test_set)

confusionMatrix(data = hat_rpart2, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         bankruptcy non_bankruptcy
##    bankruptcy              27            104
##    non_bankruptcy           7            952
##
##                 Accuracy : 0.8982
##                   95% CI : (0.8787, 0.9155)
##      No Information Rate : 0.9688
##      P-Value [Acc > NIR] : 1
##
##                    Kappa : 0.2922
##
##   Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.79412
##              Specificity : 0.90152
##           Pos Pred Value : 0.20611
##           Neg Pred Value : 0.99270
##               Prevalence : 0.03119
##           Detection Rate : 0.02477
##     Detection Prevalence : 0.12018
##        Balanced Accuracy : 0.84782
##
##         'Positive' Class : bankruptcy
##
```

Stochastic Gradient Boosting

```
fit_gbm <- train(Bankrupt ~ .,
                   data = trainup,
                   method = "gbm",
                   trControl = trainControl(method = "cv", number = 10),
                   preProcess = c("center", "scale"),
                   verbose = FALSE)#for omit useless outputs

hat_gbm <- predict(fit_gbm, newdata = test_set)
```

```r
confusionMatrix(data = hat_gbm, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             23             59
##   non_bankruptcy         11            997
##
##                Accuracy : 0.9358
##                  95% CI : (0.9196, 0.9496)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3687
##
##  Mcnemar's Test P-Value : 1.937e-08
##
##             Sensitivity : 0.67647
##             Specificity : 0.94413
##          Pos Pred Value : 0.28049
##          Neg Pred Value : 0.98909
##              Prevalence : 0.03119
##          Detection Rate : 0.02110
##    Detection Prevalence : 0.07523
##       Balanced Accuracy : 0.81030
##
##        'Positive' Class : bankruptcy
##
```

Deep Neural Network

```r
fit_dnn <- train(Bankrupt ~ .,
                   data = trainup,
                   method = "dnn",
                   trControl = trainControl(method = "cv", number = 10),
                   preProcess = c("center", "scale"))

hat_dnn <- predict(fit_dnn, newdata = test_set)

confusionMatrix(data = hat_dnn, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             30            148
##   non_bankruptcy          4            908
##
##                Accuracy : 0.8606
```

```
##                  95% CI : (0.8386, 0.8806)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2434
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.88235
##             Specificity : 0.85985
##          Pos Pred Value : 0.16854
##          Neg Pred Value : 0.99561
##              Prevalence : 0.03119
##          Detection Rate : 0.02752
##    Detection Prevalence : 0.16330
##       Balanced Accuracy : 0.87110
##
##        'Positive' Class : bankruptcy
##
```

Generalized Linear Model

```r
fit_glm <- train(Bankrupt ~ .,
                 data = trainup,
                 method = "glm",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale"))

hat_glm <- predict(fit_glm, newdata = test_set)

confusionMatrix(data = hat_glm, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             27            115
##   non_bankruptcy          7            941
##
##                Accuracy : 0.8881
##                  95% CI : (0.8678, 0.9062)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2701
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.79412
##             Specificity : 0.89110
##          Pos Pred Value : 0.19014
##          Neg Pred Value : 0.99262
```

```
##              Prevalence : 0.03119
##          Detection Rate : 0.02477
##    Detection Prevalence : 0.13028
##       Balanced Accuracy : 0.84261
##
##        'Positive' Class : bankruptcy
##
```

Lasso and Elastic-Net Regularized Generalized Linear Models

```r
#Note: a tuneGrid had to be added due to non-convergence errors
fit_glmnet <- train(Bankrupt ~ .,
                 data = trainup,
                 method = "glmnet",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale"),
                 tuneGrid = expand.grid(lambda = seq(0,.07,len = 1000),
                                        alpha = seq(0,1,len = 10)))

hat_glmnet <- predict(fit_glmnet, newdata = test_set)

confusionMatrix(data = hat_glmnet, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             27            119
##   non_bankruptcy          7            937
##
##                Accuracy : 0.8844
##                  95% CI : (0.8639, 0.9028)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2627
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.79412
##             Specificity : 0.88731
##          Pos Pred Value : 0.18493
##          Neg Pred Value : 0.99258
##              Prevalence : 0.03119
##          Detection Rate : 0.02477
##    Detection Prevalence : 0.13394
##       Balanced Accuracy : 0.84071
##
##        'Positive' Class : bankruptcy
##
```

Generalized linear model by likelihood based boosting

```
fit_glmboost <- train(Bankrupt ~ .,
                    data = trainup,
                    method = "glmboost",
                    trControl = trainControl(method = "cv", number = 10),
                    preProcess = c("center", "scale"))

hat_glmboost <- predict(fit_glmboost, newdata = test_set)

confusionMatrix(data = hat_glmboost, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             31            158
##   non_bankruptcy          3            898
##
##                Accuracy : 0.8523
##                  95% CI : (0.8298, 0.8728)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2377
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.91176
##             Specificity : 0.85038
##          Pos Pred Value : 0.16402
##          Neg Pred Value : 0.99667
##              Prevalence : 0.03119
##          Detection Rate : 0.02844
##    Detection Prevalence : 0.17339
##       Balanced Accuracy : 0.88107
##
##        'Positive' Class : bankruptcy
##
```

Linear Discriminant Analysis

```
fit_lda <- train(Bankrupt ~ .,
                      data = trainup,
                      method = "lda",
                      trControl = trainControl(method = "cv", number = 10),
                      preProcess = c("center", "scale"))

hat_lda <- predict(fit_lda, newdata = test_set)

confusionMatrix(data = hat_lda, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          bankruptcy non_bankruptcy
##   bankruptcy                29            137
##   non_bankruptcy             5            919
##
##                Accuracy : 0.8697
##                  95% CI : (0.8483, 0.8891)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2512
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.85294
##             Specificity : 0.87027
##          Pos Pred Value : 0.17470
##          Neg Pred Value : 0.99459
##              Prevalence : 0.03119
##          Detection Rate : 0.02661
##    Detection Prevalence : 0.15229
##       Balanced Accuracy : 0.86160
##
##        'Positive' Class : bankruptcy
##
```

Nearest Shrunken Centroids

```
fit_pam <- train(Bankrupt ~ .,
                 data = trainup,
                 method = "pam",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale"))

hat_pam <- predict(fit_pam, newdata = test_set)

confusionMatrix(data = hat_pam, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          bankruptcy non_bankruptcy
##   bankruptcy                32            160
##   non_bankruptcy             2            896
##
##                Accuracy : 0.8514
##                  95% CI : (0.8289, 0.872)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 1
##
```

```
##                    Kappa : 0.2431
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.94118
##             Specificity : 0.84848
##          Pos Pred Value : 0.16667
##          Neg Pred Value : 0.99777
##              Prevalence : 0.03119
##          Detection Rate : 0.02936
##    Detection Prevalence : 0.17615
##       Balanced Accuracy : 0.89483
##
##        'Positive' Class : bankruptcy
##
```

Random Forest

```r
fit_rf <- train(Bankrupt ~ .,
                data = trainup,
                method = "rf",
                trControl = trainControl(method = "cv", number = 10),
                preProcess = c("center", "scale"))

hat_rf <- predict(fit_rf, newdata = test_set)

confusionMatrix(data = hat_rf, reference = test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy              4              5
##   non_bankruptcy         30           1051
##
##                Accuracy : 0.9679
##                  95% CI : (0.9556, 0.9775)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 0.6128
##
##                   Kappa : 0.1753
##
##  Mcnemar's Test P-Value : 4.976e-05
##
##             Sensitivity : 0.117647
##             Specificity : 0.995265
##          Pos Pred Value : 0.444444
##          Neg Pred Value : 0.972248
##              Prevalence : 0.031193
##          Detection Rate : 0.003670
##    Detection Prevalence : 0.008257
##       Balanced Accuracy : 0.556456
```

```
##
##        'Positive' Class : bankruptcy
##
```

As can be seen, the use of the up-sampling technique substantially improves the results, although, as it begins to better predict bankruptcies, it reduces the overall accuracy because it has some flaws in predicting non-bankruptcy.

Ensemble

An ensemble is the concept of combining the results of different algorithms, i.e. ensembling different machine learning algorithms into one, this often greatly improve the final results. To achieve this goal we are going to take the vector of conditional probabilities given to each class by every particular algorithm, and then sum all the conditional probabilities given by the algorithms and take an average, so the result would be an average conditional probability for each observation.

Conditional Probabilities

First save the conditional probabilities with the option `type = "prob"` in the `predict` function and then create the ensemble.

```
p_rpart <- predict(fit_rpart, newdata = test_set, type = "prob")
p_rpart2 <- predict(fit_rpart2, newdata = test_set, type = "prob")
p_gbm <- predict(fit_gbm, newdata = test_set, type = "prob")
p_dnn <- predict(fit_dnn, newdata = test_set, type = "prob")
p_glm <- predict(fit_glm, newdata = test_set, type = "prob")
p_glmnet <- predict(fit_glmnet, newdata = test_set, type = "prob")
p_glmboost <- predict(fit_glmboost, newdata = test_set, type = "prob")
p_lda <- predict(fit_lda, newdata = test_set, type = "prob")
p_pam <- predict(fit_pam, newdata = test_set, type = "prob")
p_rf <- predict(fit_rf, newdata = test_set, type = "prob")

p <- (p_rpart + p_rpart2 + p_gbm + p_dnn + p_glm +
          p_glmnet + p_glmboost + p_lda + p_pam + p_rf )/10

head(p)
```

```
##    bankruptcy non_bankruptcy
## 9   0.1088522      0.8911478
## 25  0.4818338      0.5181662
## 27  0.1546637      0.8453363
## 29  0.1323310      0.8676690
## 35  0.1878008      0.8121992
## 39  0.1503327      0.8496673
```

Ensemble Results

Now test the performance of the ensemble.

```
y_pred <- factor(apply(p, 1, which.max))

y_pred <- factor(y_pred, labels = c("bankruptcy", "non_bankruptcy"))

confusionMatrix(y_pred, test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             28             76
##   non_bankruptcy          6            980
##
##                Accuracy : 0.9248
##                  95% CI : (0.9075, 0.9397)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3765
##
##  Mcnemar's Test P-Value : 2.541e-14
##
##             Sensitivity : 0.82353
##             Specificity : 0.92803
##          Pos Pred Value : 0.26923
##          Neg Pred Value : 0.99391
##              Prevalence : 0.03119
##          Detection Rate : 0.02569
##    Detection Prevalence : 0.09541
##       Balanced Accuracy : 0.87578
##
##        'Positive' Class : bankruptcy
##
```

Optimization

Now we are going to test if there was an improvement with the ensemble, or if there is a particular combination of models for the ensemble that stands out from the others for its results.

Required Functions

To make this possible we will first define functions that multiply the conditional probabilities of the algorithm to make the prediction 0 (without taking into account that model) or 1 (taking into account the model).

```
f1 <- function(vector){if(1%in%vector)1 else 0}
f2 <- function(vector){if(2%in%vector)1 else 0}
f3 <- function(vector){if(3%in%vector)1 else 0}
f4 <- function(vector){if(4%in%vector)1 else 0}
f5 <- function(vector){if(5%in%vector)1 else 0}
f6 <- function(vector){if(6%in%vector)1 else 0}
```

```
f7 <- function(vector){if(7%in%vector)1 else 0}
f8 <- function(vector){if(8%in%vector)1 else 0}
f9 <- function(vector){if(9%in%vector)1 else 0}
f10 <- function(vector){if(10%in%vector)1 else 0}
```

Model combinations

And now create a grid with all the possible combinations of the models.

```
grid <- t(combn(c(1:10,rep(NA,9)),10))
grid <- grid[!duplicated(grid),]
grid <- grid[order(as.character(grid[,1])), ]
head(grid)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    2    3    4    5    6    7    8    9    10
## [2,]    1    2    3    4    5    6    7    8    9    NA
## [3,]    1    2    3    4    5    6    7    8   10    NA
## [4,]    1    2    3    4    5    6    7    8   NA    NA
## [5,]    1    2    3    4    5    6    7    9   10    NA
## [6,]    1    2    3    4    5    6    7    9   NA    NA
```

Recalling the objective of this model we want to have high sensitivity (predict bankruptcy), but **how much are we willing to give up in accuracy to achieve this?**

So, lets create two vector, one with the Sensitivity and the other with the Accuracy for each model

```
Sensitivity <- c(confusionMatrix(hat_rpart, test_set$Bankrupt)[["byClass"]][["Sensitivity"]],
                 confusionMatrix(hat_rpart2, test_set$Bankrupt)[["byClass"]][["Sensitivity"]],
                 confusionMatrix(hat_dnn, test_set$Bankrupt)[["byClass"]][["Sensitivity"]],
                 confusionMatrix(hat_gbm, test_set$Bankrupt)[["byClass"]][["Sensitivity"]],
                 confusionMatrix(hat_glm, test_set$Bankrupt)[["byClass"]][["Sensitivity"]],
                 confusionMatrix(hat_glmnet, test_set$Bankrupt)[["byClass"]][["Sensitivity"]],
                 confusionMatrix(hat_glmboost, test_set$Bankrupt)[["byClass"]][["Sensitivity"]],
                 confusionMatrix(hat_lda, test_set$Bankrupt)[["byClass"]][["Sensitivity"]],
                 confusionMatrix(hat_pam, test_set$Bankrupt)[["byClass"]][["Sensitivity"]],
                 confusionMatrix(hat_rf, test_set$Bankrupt)[["byClass"]][["Sensitivity"]])

Accuracy <- c(
    confusionMatrix(hat_rpart, test_set$Bankrupt)[["overall"]][["Accuracy"]],
    confusionMatrix(hat_rpart2, test_set$Bankrupt)[["overall"]][["Accuracy"]],
    confusionMatrix(hat_dnn, test_set$Bankrupt)[["overall"]][["Accuracy"]],
    confusionMatrix(hat_gbm, test_set$Bankrupt)[["overall"]][["Accuracy"]],
    confusionMatrix(hat_glm, test_set$Bankrupt)[["overall"]][["Accuracy"]],
    confusionMatrix(hat_glmnet, test_set$Bankrupt)[["overall"]][["Accuracy"]],
    confusionMatrix(hat_glmboost, test_set$Bankrupt)[["overall"]][["Accuracy"]],
    confusionMatrix(hat_lda, test_set$Bankrupt)[["overall"]][["Accuracy"]],
    confusionMatrix(hat_pam, test_set$Bankrupt)[["overall"]][["Accuracy"]],
    confusionMatrix(hat_rf, test_set$Bankrupt)[["overall"]][["Accuracy"]])
```

Optimization Results

Now we are going to run different combinations of models for the ensemble and keep the best one for us. For this we use the functions multiplying the conditional probabilities of the models, so if the model is in the vector, the function gives a 1 and take that model into account, else give a 0, and the model will be ignored. Note that the vector is a row of the `grid` that contains an unique combination, also, the result of the sum of the models selected are divided by the length of the vector that does not contains NAs (this is the number of models that are taking into account). And finally it is compared to see if the ensemble is better than all the models with respect to Accuracy and Sensitivity.

```
i <- 1:nrow(grid)
optimization <- data.frame(t(sapply(i, function(i){

    vector <- grid[i,]
    p_hat <- (p_rpart*f1(vector) +
                  p_rpart2*f2(vector) +
                  p_gbm*f3(vector) +
                  p_dnn*f4(vector) +
                  p_glm*f5(vector) +
                  p_glmnet*f6(vector) +
                  p_glmboost*f7(vector) +
                  p_lda*f8(vector) +
                  p_pam*f9(vector) +
                  p_rf*f10(vector) )/sum(!is.na(vector))

    y_pred <- factor(apply(p_hat, 1, which.max))

    y_pred <- factor(y_pred, labels = c("bankruptcy", "non_bankruptcy"))


    c(
    all(confusionMatrix(y_pred, test_set$Bankrupt)[["overall"]][["Accuracy"]]>Accuracy),

    all(confusionMatrix(y_pred, test_set$Bankrupt)[["byClass"]][["Sensitivity"]]>Sensitivity)
    )

})))
colnames(optimization) <- c("Accuracy" , "Sensitivity")
head(optimization)
```

```
##   Accuracy Sensitivity
## 1    FALSE       FALSE
## 2    FALSE       FALSE
## 3    FALSE       FALSE
## 4    FALSE       FALSE
## 5    FALSE       FALSE
## 6    FALSE       FALSE
```

See which Ensemble combination was better than all the models in terms of accuracy

```
model_names <- c("rpart", "rpart2", "dnn", "gbm","glm",
                 "glmnet", "glmboost","lda", "pam","rf")

which(optimization$Accuracy)
```

```
## [1] 960
```

```r
max.Acc <- grid[which(optimization$Accuracy),][!is.na(grid[which(optimization$Accuracy),])]

model_names[max.Acc]
```

```
## [1] "gbm" "rf"
```

The problem with this is that we have a very low Sensitivity, thus is useless.

```r
    vector <- max.Acc
    p_hat <- (p_rpart*f1(vector) +
                  p_rpart2*f2(vector) +
                  p_gbm*f3(vector) +
                  p_dnn*f4(vector) +
                  p_glm*f5(vector) +
                  p_glmnet*f6(vector) +
                  p_glmboost*f7(vector) +
                  p_lda*f8(vector) +
                  p_pam*f9(vector) +
                  p_rf*f10(vector) )/sum(!is.na(vector))

    y_pred <- factor(apply(p_hat, 1, which.max))

    y_pred <- factor(y_pred, labels = c("bankruptcy", "non_bankruptcy"))

    confusionMatrix(y_pred, test_set$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction        bankruptcy non_bankruptcy
##   bankruptcy              13             12
##   non_bankruptcy          21           1044
##
##                Accuracy : 0.9697
##                  95% CI : (0.9577, 0.9791)
##     No Information Rate : 0.9688
##     P-Value [Acc > NIR] : 0.4761
##
##                   Kappa : 0.4255
##
##  Mcnemar's Test P-Value : 0.1637
##
##             Sensitivity : 0.38235
##             Specificity : 0.98864
##          Pos Pred Value : 0.52000
##          Neg Pred Value : 0.98028
##              Prevalence : 0.03119
##          Detection Rate : 0.01193
##    Detection Prevalence : 0.02294
##       Balanced Accuracy : 0.68549
##
```

```
##        'Positive' Class : bankruptcy
##
```

In terms of Sensitivity the ensemble is worse than at least one particular model always

```
which(optimization$Sensitivity)
```

```
## integer(0)
```

So lets see which is the model with better Sensitivity

```
model_names[which.max(Sensitivity)]
```

```
## [1] "pam"
```

Be careful, because this does not means that the Ensemble is worst in general, we can expect that the Ensemble of all models is somewhere between maximize the Accuracy and maximize the Sensitivity and not in an extreme, so this could give us a robust prediction.

Final Model

Now that we tried to optimize the model by selecting the better combination we are going to make the following decision: According of how much we are willing to give up in accuracy to improve sensitivity we can use two models, the first is the best particular model in terms of sensitivity, and the second will be the Ensemble, because of the characteristics of this approach the **Ensemble** will be considered as the **Final Model**, but we will also take into account a second option.

The Models will be tested on the validation set **JUST TO SEE** their performance.

Up-Sampling

We use the up-sampling technique again but in the `df`.

```
df_trainup <- upSample(y=df$Bankrupt,
                       x=df[,-1],
                       yname = "Bankrupt")

table(df_trainup$Bankrupt)
```

```
##
##     bankruptcy non_bankruptcy
##           5283           5283
```

Machine Learning Models

And now train the models with all the `df` set

Unique Model (Best Sensitivity)

```r
unique_model_final <- train(Bankrupt ~ .,
                            data = df_trainup,
                            method = model_names[which.max(Sensitivity)],
                            trControl = trainControl(method = "cv", number = 10),
                            preProcess = c("center", "scale"))

hat_unique_model_final <- predict(unique_model_final, newdata = validation)
```

Classification Tree 1

```r
fit_rpart_final <- train(Bankrupt ~ .,
                         data = df_trainup,
                         method = "rpart",
                         trControl = trainControl(method = "cv", number = 10),
                         preProcess = c("center", "scale"))

hat_rpart_final <- predict(fit_rpart_final, newdata = validation)

p_rpart_final <- predict(fit_rpart_final, newdata = validation, type = "prob")

confusionMatrix(data = hat_rpart_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             34            147
##   non_bankruptcy         14           1169
##
##                Accuracy : 0.882
##                  95% CI : (0.8636, 0.8986)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2555
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.70833
##             Specificity : 0.88830
##          Pos Pred Value : 0.18785
##          Neg Pred Value : 0.98817
##              Prevalence : 0.03519
##          Detection Rate : 0.02493
##    Detection Prevalence : 0.13270
##       Balanced Accuracy : 0.79832
##
##        'Positive' Class : bankruptcy
##
```

Classification Tree 2

```
fit_rpart2_final <- train(Bankrupt ~ .,
                    data = df_trainup,
                    method = "rpart2",
                    trControl = trainControl(method = "cv", number = 10),
                    preProcess = c("center", "scale"))

hat_rpart2_final <- predict(fit_rpart2_final, newdata = validation)

p_rpart2_final <- predict(fit_rpart2_final, newdata = validation, type = "prob")

confusionMatrix(data = hat_rpart2_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction        bankruptcy non_bankruptcy
##   bankruptcy              37            187
##   non_bankruptcy          11           1129
##
##                Accuracy : 0.8548
##                  95% CI : (0.835, 0.8731)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2273
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.77083
##             Specificity : 0.85790
##          Pos Pred Value : 0.16518
##          Neg Pred Value : 0.99035
##              Prevalence : 0.03519
##          Detection Rate : 0.02713
##    Detection Prevalence : 0.16422
##       Balanced Accuracy : 0.81437
##
##        'Positive' Class : bankruptcy
##
```

Stochastic Gradient Boosting

```
fit_gbm_final <- train(Bankrupt ~ .,
                 data = df_trainup,
                 method = "gbm",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale"),
                 verbose = FALSE)
```

```r
hat_gbm_final <- predict(fit_gbm_final, newdata = validation)

p_gbm_final <- predict(fit_gbm_final, newdata = validation, type = "prob")

confusionMatrix(data = hat_gbm_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction        bankruptcy non_bankruptcy
##   bankruptcy              33            100
##   non_bankruptcy          15           1216
##
##                Accuracy : 0.9157
##                  95% CI : (0.8997, 0.9299)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.33
##
##  Mcnemar's Test P-Value : 4.762e-15
##
##             Sensitivity : 0.68750
##             Specificity : 0.92401
##          Pos Pred Value : 0.24812
##          Neg Pred Value : 0.98781
##              Prevalence : 0.03519
##          Detection Rate : 0.02419
##    Detection Prevalence : 0.09751
##       Balanced Accuracy : 0.80576
##
##        'Positive' Class : bankruptcy
##
```

Deep Neural Network

```r
fit_dnn_final <- train(Bankrupt ~ .,
                data = df_trainup,
                method = "dnn",
                trControl = trainControl(method = "cv", number = 10),
                preProcess = c("center", "scale"))

hat_dnn_final <- predict(fit_dnn_final, newdata = validation)

p_dnn_final <- predict(fit_dnn_final, newdata = validation, type = "prob")

confusionMatrix(data = hat_dnn_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
```

```
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             37            187
##   non_bankruptcy         11           1129
##
##                Accuracy : 0.8548
##                  95% CI : (0.835, 0.8731)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2273
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.77083
##             Specificity : 0.85790
##          Pos Pred Value : 0.16518
##          Neg Pred Value : 0.99035
##              Prevalence : 0.03519
##          Detection Rate : 0.02713
##    Detection Prevalence : 0.16422
##       Balanced Accuracy : 0.81437
##
##        'Positive' Class : bankruptcy
##
```

Generalized Linear Model

```
fit_glm_final <- train(Bankrupt ~ .,
                 data = df_trainup,
                 method = "glm",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale"))

hat_glm_final <- predict(fit_glm_final, newdata = validation)

p_glm_final  <- predict(fit_glm_final , newdata = validation, type = "prob")

confusionMatrix(data = hat_glm_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             26             83
##   non_bankruptcy         22           1233
##
##                Accuracy : 0.923
##                  95% CI : (0.9076, 0.9366)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2969
```

```
##
##  Mcnemar's Test P-Value : 4.759e-09
##
##             Sensitivity : 0.54167
##             Specificity : 0.93693
##          Pos Pred Value : 0.23853
##          Neg Pred Value : 0.98247
##              Prevalence : 0.03519
##          Detection Rate : 0.01906
##    Detection Prevalence : 0.07991
##       Balanced Accuracy : 0.73930
##
##        'Positive' Class : bankruptcy
##
```

Lasso and Elastic-Net Regularized Generalized Linear Models

```
fit_glmnet_final <- train(Bankrupt ~ .,
                    data = df_trainup,
                    method = "glmnet",
                    trControl = trainControl(method = "cv", number = 10),
                    preProcess = c("center", "scale"),
                    tuneGrid = expand.grid(lambda = seq(0,.07,len = 1000), alpha = seq(0,1,len = 10)))


hat_glmnet_final <- predict(fit_glmnet_final, newdata = validation)

p_glmnet_final <- predict(fit_glmnet_final , newdata = validation, type = "prob")

confusionMatrix(data = hat_glmnet_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         bankruptcy non_bankruptcy
##   bankruptcy               35            204
##   non_bankruptcy           13           1112
##
##                Accuracy : 0.8409
##                  95% CI : (0.8204, 0.8599)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1968
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.72917
##             Specificity : 0.84498
##          Pos Pred Value : 0.14644
##          Neg Pred Value : 0.98844
##              Prevalence : 0.03519
```

```
##          Detection Rate : 0.02566
##    Detection Prevalence : 0.17522
##       Balanced Accuracy : 0.78708
##
##        'Positive' Class : bankruptcy
##
```

Generalized linear model by likelihood based boosting

```
fit_glmboost_final <- train(Bankrupt ~ .,
                      data = df_trainup,
                      method = "glmboost",
                      trControl = trainControl(method = "cv", number = 10),
                      preProcess = c("center", "scale"))

hat_glmboost_final <- predict(fit_glmboost_final, newdata = validation)

p_glmboost_final <- predict(fit_glmboost_final, newdata = validation, type = "prob")

confusionMatrix(data = hat_glmboost_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             39            228
##   non_bankruptcy          9           1088
##
##                Accuracy : 0.8262
##                  95% CI : (0.8051, 0.846)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1999
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.81250
##             Specificity : 0.82675
##          Pos Pred Value : 0.14607
##          Neg Pred Value : 0.99180
##              Prevalence : 0.03519
##          Detection Rate : 0.02859
##    Detection Prevalence : 0.19575
##       Balanced Accuracy : 0.81962
##
##        'Positive' Class : bankruptcy
##
```

Linear Discriminant Analysis

```
fit_lda_final  <- train(Bankrupt ~ .,
                  data = df_trainup,
                  method = "lda",
                  trControl = trainControl(method = "cv", number = 10),
                  preProcess = c("center", "scale"))

hat_lda_final  <- predict(fit_lda_final , newdata = validation)

p_lda_final  <- predict(fit_lda_final, newdata = validation, type = "prob")

confusionMatrix(data = hat_lda_final , reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction        bankruptcy non_bankruptcy
##   bankruptcy              39            214
##   non_bankruptcy           9           1102
##
##                Accuracy : 0.8365
##                  95% CI : (0.8158, 0.8558)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2126
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.81250
##             Specificity : 0.83739
##          Pos Pred Value : 0.15415
##          Neg Pred Value : 0.99190
##              Prevalence : 0.03519
##          Detection Rate : 0.02859
##    Detection Prevalence : 0.18548
##       Balanced Accuracy : 0.82494
##
##        'Positive' Class : bankruptcy
##
```

Nearest Shrunken Centroids

```
fit_pam_final <- train(Bankrupt ~ .,
                  data = df_trainup,
                  method = "pam",
                  trControl = trainControl(method = "cv", number = 10),
                  preProcess = c("center", "scale"))

hat_pam_final <- predict(fit_pam_final, newdata = validation)
```

```r
p_pam_final  <- predict(fit_pam_final, newdata = validation, type = "prob")

confusionMatrix(data = hat_pam_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             39            233
##   non_bankruptcy          9           1083
##
##                Accuracy : 0.8226
##                  95% CI : (0.8013, 0.8425)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1956
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.81250
##             Specificity : 0.82295
##          Pos Pred Value : 0.14338
##          Neg Pred Value : 0.99176
##              Prevalence : 0.03519
##          Detection Rate : 0.02859
##    Detection Prevalence : 0.19941
##       Balanced Accuracy : 0.81772
##
##        'Positive' Class : bankruptcy
##
```

Random Forest

```r
fit_rf_final <- train(Bankrupt ~ .,
                data = df_trainup,
                method = "rf",
                trControl = trainControl(method = "cv", number = 10),
                preProcess = c("center", "scale"))

hat_rf_final <- predict(fit_rf_final, newdata = validation)

p_rf_final  <- predict(fit_rf_final, newdata = validation, type = "prob")

confusionMatrix(data = hat_rf_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
```

```
##   bankruptcy              7             3
##   non_bankruptcy         41          1313
##
##                Accuracy : 0.9677
##                  95% CI : (0.9569, 0.9765)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 0.3095
##
##                   Kappa : 0.2321
##
##  Mcnemar's Test P-Value : 2.434e-08
##
##             Sensitivity : 0.145833
##             Specificity : 0.997720
##          Pos Pred Value : 0.700000
##          Neg Pred Value : 0.969719
##              Prevalence : 0.035191
##          Detection Rate : 0.005132
##    Detection Prevalence : 0.007331
##       Balanced Accuracy : 0.571777
##
##        'Positive' Class : bankruptcy
##
```

Final Ensemble

And now make the final Ensemble.

```
p_final <- (p_rpart_final + p_rpart2_final + p_gbm_final + p_dnn_final +
             p_glm_final + p_glmnet_final  + p_glmboost_final + p_lda_final +
             p_pam_final + p_rf_final)/10

y_final_pred <- factor(apply(p_final, 1, which.max))
y_final_pred <- factor(y_final_pred, labels = c("bankruptcy", "non_bankruptcy"))
```

Results

Unique Model

The results of our unique model, the best in terms of sensitivity was:

```
confusionMatrix(data = hat_unique_model_final, reference = validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             39            233
##   non_bankruptcy          9           1083
##
```

```
##                Accuracy : 0.8226
##                  95% CI : (0.8013, 0.8425)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1956
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.81250
##             Specificity : 0.82295
##          Pos Pred Value : 0.14338
##          Neg Pred Value : 0.99176
##              Prevalence : 0.03519
##          Detection Rate : 0.02859
##    Detection Prevalence : 0.19941
##       Balanced Accuracy : 0.81772
##
##        'Positive' Class : bankruptcy
##
```

Note the low Accuracy but the highest Sensitivity, this model have a lot of false negatives.

Final Ensemble

and the results of our Ensemble, the **Final Model**, are as follows:

```
confusionMatrix(y_final_pred, validation$Bankrupt)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       bankruptcy non_bankruptcy
##   bankruptcy             34            122
##   non_bankruptcy         14           1194
##
##                Accuracy : 0.9003
##                  95% CI : (0.8832, 0.9157)
##     No Information Rate : 0.9648
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2954
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.70833
##             Specificity : 0.90729
##          Pos Pred Value : 0.21795
##          Neg Pred Value : 0.98841
##              Prevalence : 0.03519
##          Detection Rate : 0.02493
##    Detection Prevalence : 0.11437
```

```
##        Balanced Accuracy : 0.80781
##
##         'Positive' Class : bankruptcy
##
```

Let's interpret these results.

The **Accuracy** is the ratio of correct predictions to total predictions made, the accuracy of our final model was 0.9002933, this means that 90.03% of the time we predict classes well, but due to the prevalence of one class in the data we may want to look at other metrics like the **Balanced Accuracy**, this metric is the average between the sensitivity and specificity.

The **Sensitivity** (also called **Recall**) is the number of correct positive predictions, i.e. we predict $\widehat{Y} = 1$ and indeed $Y = 1$, divided by the total number of positives, all the times that $Y = 1$. In our case, the sensitivity represents the number of times we correctly predict a bankruptcy, divided by the total number of bankruptcy cases. So we can correctly predict a bankruptcy 70.83% of the times.

The **Specificity** is the number of correct negative predictions, divided by the total number of negatives, i.e. $\widehat{Y} = 0$ and indeed $Y = 0$ divided by all the times that $Y = 0$. In our case, the sensitivity represents the number of times we correctly predict a non-bankruptcy, divided by the total number of non-bankruptcy cases. Therefore, we are able to correctly predict a non-bankruptcy 90.73% of the times.

If we remember, the **Balanced Accuracy** was the average of this two last metrics ($\frac{Sensitivity+Specificity}{2}$), and we may prefer to use it instead of Accuracy due to the imbalance, because note that if we just predict non-bankruptcy we can achieve a high accuracy due to prevalence. However, if we use Balanced Accuracy we weight our predictive ability for both classes equivalently and is more representative of our predictive power. In our case, we have correct predictions 80.78% of the times by weighting both classes equally (taking into account the prevalence).

Last, but not least, the "Pos Pred Value", the **Precision**, is the number of correct positive predictions, i.e. we predict $\widehat{Y} = 1$ and indeed $Y = 1$, divided by the total number of positive predictions, all the times that we predict $\widehat{Y} = 1$. It is important to say that there is a trade-off between Recall and Precision for a given model, if we want to improve one, the other will get worse. In our case the 21.79% of the times that we predicted bankruptcy, it was indeed a bankruptcy. If we want to think of this in terms of non-bankruptcy we can look at "Neg Pred Value".

Conclusion

By way of conclusion, we saw that depending on how well we want to predict bankruptcy, our models may vary, we might want to use a single model at the cost of our accuracy. However, the best option is our Ensemble model, which takes the conditional probabilities of all the other models to predict better, note that with the Ensemble model we get a high accuracy and high sensitivity, beyond that the sensitivity is less than in the unique model. But in the unique model our accuracy drops a lot, and this is because we fail too much when we want to predict a non-bankruptcy, and because of the prevalence this is a lot of observation that we are predicting wrong and this not happens in the Ensemble model, in this model we get a good accuracy and also a good sensitivity.

Being able to predict the bankruptcy of a company could have several impacts, this could help the company itself to be more careful when making decisions as they may be going the wrong way and could lead to bankruptcy. It can help governments to take into account the companies that are about to go bankrupt and generate their own measurements of the economic impact in the event that this scenario occurs and from this, make decisions on how to intervene or abstain. A final example may be the decision of investors to sell the shares of this companies that presents a high risk in their portfolio.

Finally, for future estimations we should advance in terms of the possible models used to build the ensemble, also we can try different cutoff when we drop predictors by their correlation with each other or even use more preProcess options like BoxCox or YeoJohnson that were not addressed in this project.