Machine Learning Analysis with Lagged Predictors

Falco J. Bargagli-Stoffi, Massimo Riccaboni, Armando Rungi

23/2/2020

Introduction

This `R Markdown` file reproduces the lagged machine learning analysis for the paper *"Machine learning for zombie hunting. Firms' failures, financial constraints, and misallocation"* by Falco J. Bargagli-Stoffi (IMT School for Advanced Studies/KU Leuven), Massimo Riccaboni (IMT School for Advanced Studies) and Armando Rungi (IMT School for Advanced Studies).

R Markdown

This is an `R Markdown` document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using `R Markdown` see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded `R` code chunks within the document. You can embed an R code chunks like the following.

Packages Upload

The following packages and functions are the ones used for the analyses performed in the `R` code. The `functions.R` file contains the functions `F1_score`, `balanced_accuracy`, `model_compare` and `DtD` that were developed to reproduce the following analyses.

```r
rm(list=ls()) # to clean the memeory
memory.limit(size=1000000)
```

```
## [1] 1e+06
```

```r
options(java.parameters = "-Xmx15000m")
library(rJava)
library(bartMachine)
library(haven)
library(plyr)
library(dplyr)
library(PRROC)
library(rpart)
library(party)
library(caret)
library(Amelia)
library(PresenceAbsence)
library(devtools)
library(SuperLearner)
library(Metrics)
library(pROC)
library(Hmisc)
source('functions.R')
```

Data Upload

In the following chunks of code we upload the data, we initialize the main variables used in the analysis and we restrict the sample to the Italian firms.

```
data <- read_dta("analysis_data_indicators.dta")
```

```
names(data)[names(data) == 'GUO___BvD_ID_number'] <- 'guo'
data$control <- ifelse(data$guo=="", 0, 1)
data$nace <- as.factor(data$nace)
data$area <- as.factor(data$area)
levels(data$nace) <- floor(as.numeric(levels(data$nace))/100)
data_italy <- data[which(data$iso=="IT"),]
```

Explorative Data Analysis (EDA)

Run the following chunks of code to produce Tables 1 in the paper.

```
table_1 <- table(status_aggregated)
table_1
```

```
## status_aggregated
##         Active     Bankruptcy      Dissolved In Liquidation
##         287587           1533           8540           7221
```

```
prop.table(table_1)
```

```
## status_aggregated
##         Active     Bankruptcy      Dissolved In Liquidation
##    0.943276229    0.005028191    0.028010929    0.023684651
```

Run the following chunk of code to produce Table 8 in the paper.

```
table(data$iso)
```

```
##
##     ES     FR     IT     PT
##  86979  74821 304906  41166
```

```
table_2 <- table(data$iso, data$failure)
table_2
```

```
##
##           0      1
##   ES  83639   3340
##   FR  70191   4630
##   IT 287587  17319
##   PT  38248   2918
```

```
prop.table(table_2, 1)
```

```
##
##               0          1
##   ES 0.96159993 0.03840007
##   FR 0.93811898 0.06188102
##   IT 0.94319889 0.05680111
##   PT 0.92911626 0.07088374
```

Run the following chunks of code to produce Table 9 in the paper.

```
icr_italy <- table(data_italy$ICR_failure)
prop.table(icr_italy)
```

```
##
##         0         1
## 0.7830209 0.2169791
```

```
icr_spain <- table(data_spain$ICR_failure)
prop.table(icr_spain)
```

```
##
##         0         1
## 0.7733839 0.2266161
```

```
icr_france <- table(data_france$ICR_failure)
prop.table(icr_france)
```

```
##
##         0         1
## 0.7355627 0.2644373
```

```
icr_portugal <- table(data_portugal$ICR_failure)
prop.table(icr_portugal)
```

```
##
##         0         1
## 0.7596578 0.2403422
```

```
neg_va_italy <- table(data_italy$NEG_VA)
prop.table(neg_va_italy)
```

```
##
##          0          1
## 0.96555555 0.03444445
```

```
neg_va_spain <- table(data_spain$NEG_VA)
prop.table(neg_va_spain)
```

```
##
##         0         1
## 0.9429404 0.0570596
```

```
neg_va_france <- table(data_france$NEG_VA)
prop.table(neg_va_france)
```

```
##
##          0          1
## 0.98668644 0.01331356
```

```
neg_va_portugal <- table(data_portugal$NEG_VA)
prop.table(neg_va_portugal)
```

```
##
##          0          1
## 0.96941531 0.03058469
```

```
interest_diff_italy <- table(data_italy$interest_diff)
prop.table(interest_diff_italy)
```

```
##
##         0         1
## 0.5763131 0.4236869
```

```
interest_diff_spain <- table(data_spain$interest_diff)
prop.table(interest_diff_spain)
```

```
##
##         0         1
## 0.7468279 0.2531721
```

```
interest_diff_france <- table(data_france$interest_diff)
prop.table(interest_diff_france)
```

```
##
##         0         1
## 0.7394369 0.2605631
```

```
interest_diff_portugal <- table(data_portugal$interest_diff)
prop.table(interest_diff_portugal)
```

```
##
##         0         1
## 0.7437028 0.2562972
```

```
profitability_italy <- table(data_italy$profitability)
prop.table(profitability_italy)
```

```
##
##          0          1
## 0.96418232 0.03581768
```

```
profitability_spain <- table(data_spain$profitability)
prop.table(profitability_spain)
```

```
##
##          0          1
## 0.98663231 0.01336769
```

```
profitability_france <- table(data_france$profitability)
prop.table(profitability_france)
```

```
##
##          0          1
## 0.96267853 0.03732147
```

```
profitability_portugal <- table(data_portugal$profitability)
prop.table(profitability_portugal)
```

```
##
##          0          1
## 0.93905444 0.06094556
```

```
misallocated_fixed_italy <- table(data_italy$misallocated_fixed)
prop.table(misallocated_fixed_italy)
```

```
##
##         0         1
## 0.8794937 0.1205063
```

```r
misallocated_fixed_spain <- table(data_spain$misallocated_fixed)
prop.table(misallocated_fixed_spain)
```

```
##
##          0          1
## 0.97075506 0.02924494
```

```r
misallocated_fixed_france <- table(data_france$misallocated_fixed)
prop.table(misallocated_fixed_france)
```

```
##
##         0         1
## 0.8888808 0.1111192
```

```r
misallocated_fixed_portugal <- table(data_portugal$misallocated_fixed)
prop.table(misallocated_fixed_portugal)
```

```
##
##         0         1
## 0.7938577 0.2061423
```

Run the following code to explore the missingness patterns in the variables.

```r
# Missingness in the variables
sapply(data_italy,function(x) sum(is.na(x)))
```

Exclude the highly missing variables: *labour_product, retained_earnings, firm_value, tax_payables, pension_payables, pension_tax_debts* (above 200,000 missing: +65% missing).

Run the following code to reproduce the "missingness maps" in Figures 1 and 2 of the paper.

```r
raw_variables <- c("failure", "iso", "control", "Number_of_patents",
                   "Number_of_trademarks", "conscode", "nace",
                   "wage_bill","shareholders_funds", "added_value",
                   "cash_flow", "ebitda", "fin_rev", "liquidity_ratio",
                   "total_assets", "depr", "long_term_debt", "employees",
                   "materials", "loans", "fixed_assets", "tax",
                   "current_liabilities", "current_assets",
                   "fin_expenses", "int_paid", "solvency_ratio",
                   "net_income", "revenue", "int_fixed_assets")
raw_data_missing <- data_italy[raw_variables]
set.seed(2020)
sample_1000 <- sample(nrow(raw_data_missing),
                      1000, replace = FALSE)
pdf("missing_variables.pdf")
missmap(raw_data_missing[sample_1000,],
        main = "Missing values vs Observed")
```

```
## Warning in if (class(obj) == "amelia") {: la condizione la lunghezza > 1 e
## solo il promo elemento verrÃ  utilizzato

## Warning: Unknown or uninitialised column: 'arguments'.

## Warning: Unknown or uninitialised column: 'arguments'.

## Warning: Unknown or uninitialised column: 'imputations'.
```

```r
dev.off()
```

```
## pdf
##   2
```

```
indicators <- c("consdummy", "capital_intensity", "fin_cons100",
                "inv", "ICR_failure", "interest_diff", "NEG_VA",
                "real_SA", "Z_score", "misallocated_fixed",
                "profitability", "area", "tfp_acf", "dummy_patents",
                "dummy_trademark", "financial_sustainability",
                "liquidity_return")
indicators_missing <- data_italy[indicators]
pdf("missing_indicators.pdf")
missmap(indicators_missing[sample_1000,],
        main = "Missing values vs Observed")
```

```
## Warning in if (class(obj) == "amelia") {: la condizione la lunghezza > 1 e
## solo il promo elemento verrÃ  utilizzato

## Warning: Unknown or uninitialised column: 'arguments'.

## Warning: Unknown or uninitialised column: 'arguments'.

## Warning: Unknown or uninitialised column: 'imputations'.
```

```
dev.off()
```

```
## pdf
##   2
```

Machine Learning Analysis

Data Inizialization

Select the lagged variables.

```
lagged_variables <- c("failure", "iso", "control", "nace",
                      "shareholders_funds", "added_value",
                      "cash_flow", "ebitda", "fin_rev",
                      "liquidity_ratio", "total_assets",
                      "depr", "long_term_debt", "employees",
                      "materials", "loans", "wage_bill",
                      "tfp_acf", "fixed_assets", "tax",
                      "current_liabilities", "current_assets",
                      "fin_expenses", "int_paid",
                      "solvency_ratio", "net_income",
                      "revenue", "consdummy", "capital_intensity",
                      "fin_cons100", "inv", "ICR_failure",
                      "interest_diff", "NEG_VA", "real_SA",
                      "Z_score", "misallocated_fixed",
                      "profitability", "area", "dummy_patents",
                      "dummy_trademark","financial_sustainability",
                      "liquidity_return", "int_fixed_assets")
data_lagged <- data_italy[lagged_variables]
```

Select the predictors.

```r
predictors <- c("control", "nace", "shareholders_funds",
                "added_value", "cash_flow", "ebitda",
                "fin_rev", "liquidity_ratio", "total_assets",
                "depr", "long_term_debt", "employees",
                "materials", "loans", "wage_bill", "tfp_acf",
                "fixed_assets", "tax", "current_liabilities",
                "current_assets", "fin_expenses", "int_paid",
                "solvency_ratio", "net_income", "revenue",
                "consdummy", "capital_intensity", "fin_cons100",
                "inv", "ICR_failure", "interest_diff", "NEG_VA",
                "real_SA", "misallocated_fixed", "profitability",
                "area", "dummy_patents", "dummy_trademark",
                "financial_sustainability", "liquidity_return",
                "int_fixed_assets")
formula <- as.formula(paste("as.factor(failure) ~",
                            paste(predictors, collapse="+")))
```

Create training and testing sets by assigning 90% of the observations to the training set and 10% to the testing set.

```r
omitted <- na.omit(data_lagged)
set.seed(2020)
index <- sample(seq_len(nrow(omitted)), size = nrow(omitted)*0.9)
train <- omitted[index,]
test <- omitted[-index,]
```

Logit

Run a Logistic regression analysis.

```r
system.time({
logit <- glm(formula, data= train,  na.action = "na.omit",
             family=binomial(link='logit'))
})
```

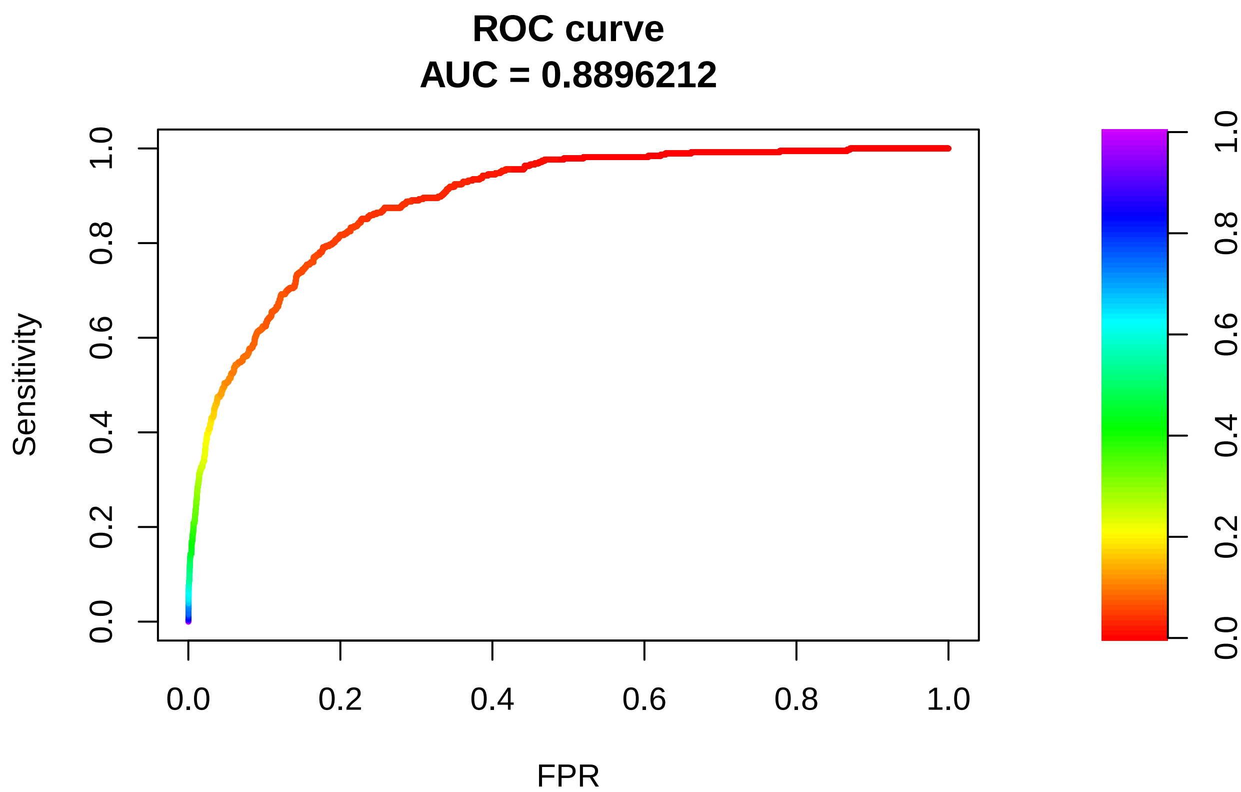
```
##    user  system elapsed
##    6.97    0.32    9.56
```

Depict the performance measures by running the following chunks.

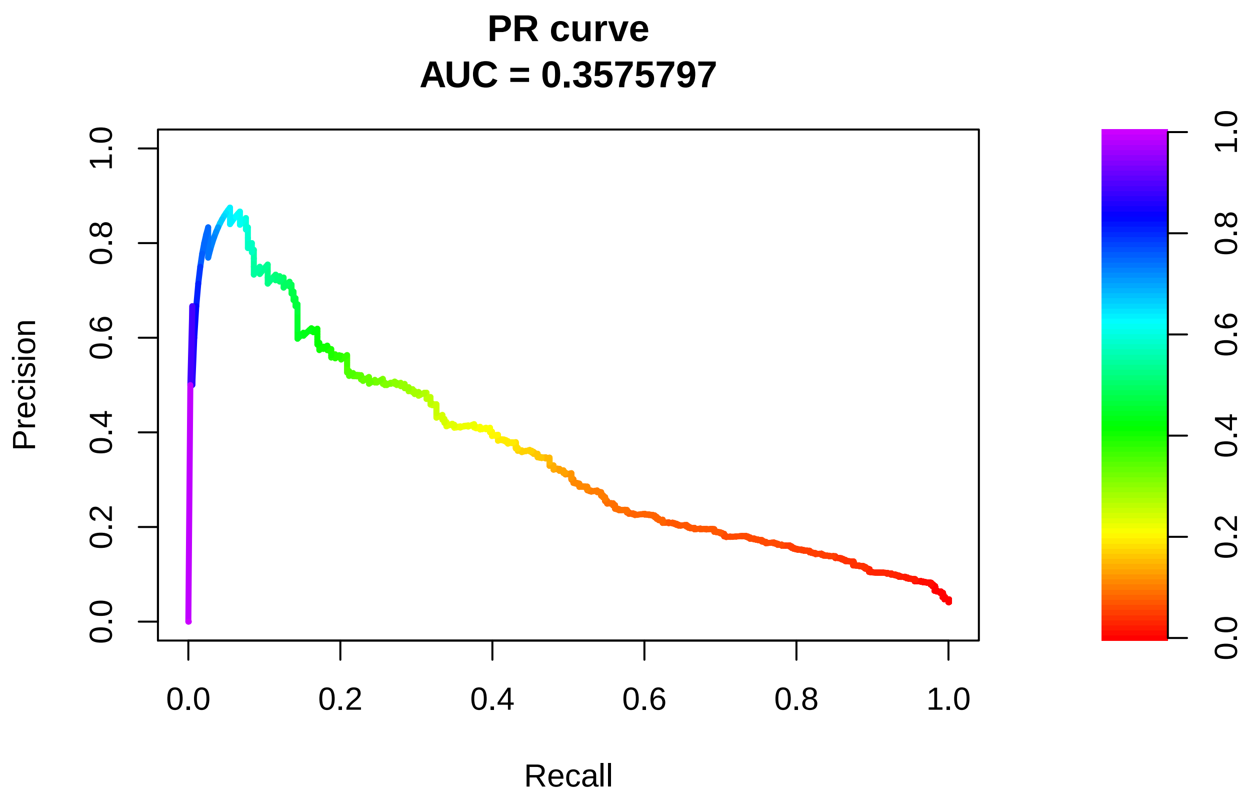
```r
fitted.prob.logit <- predict(logit, newdata=test, type='response')
```

```r
fitted.logit <- as.numeric(fitted.prob.logit)
fg.logit <- fitted.logit[test$failure==1]
bg.logit <- fitted.logit[test$failure==0]
```

```r
roc_logit <- roc.curve(scores.class0 = fg.logit,
                       scores.class1 = bg.logit,
                       curve = T)
plot(roc_logit)
```




```
pr_logit <- pr.curve(scores.class0 = fg.logit,
                     scores.class1 = bg.logit,
                     curve = T)
plot(pr_logit)
```




```
fitted.logit <- ifelse(fitted.prob.logit>0.5,1,0)
f1_logit <- f1_score(fitted.logit,
                     test$failure,
                     positive.class="1")
```

```
balanced_accuracy_logit<-balanced_accuracy(fitted.logit, test$failure)
accuracy_logit <- as.data.frame(rbind(postResample(fitted.logit,
                                                   test$failure)))
```

```
logit_fit <- as.data.frame(cbind(roc_logit$auc,
                                 pr_logit$auc.integral,
                                 f1_logit,
                                 balanced_accuracy_logit,
                                 accuracy_logit$Rsquared))
colnames(logit_fit) <- c("AUC", "PR", "f1-score", "BACC", "Rsquared")
logit_fit
```

```
##         AUC        PR  f1-score      BACC   Rsquared
## 1 0.8896212 0.3575797 0.2098214 0.8433113 0.08287033
```

Classification Tree

Run a classification tree analysis.

```
set.seed(2020)
system.time({
c.tree <- ctree(formula, data=train,
```

```
                 control = ctree_control(testtype = "MonteCarlo",
                 mincriterion = 0.90, nresample = 1000))
})
```

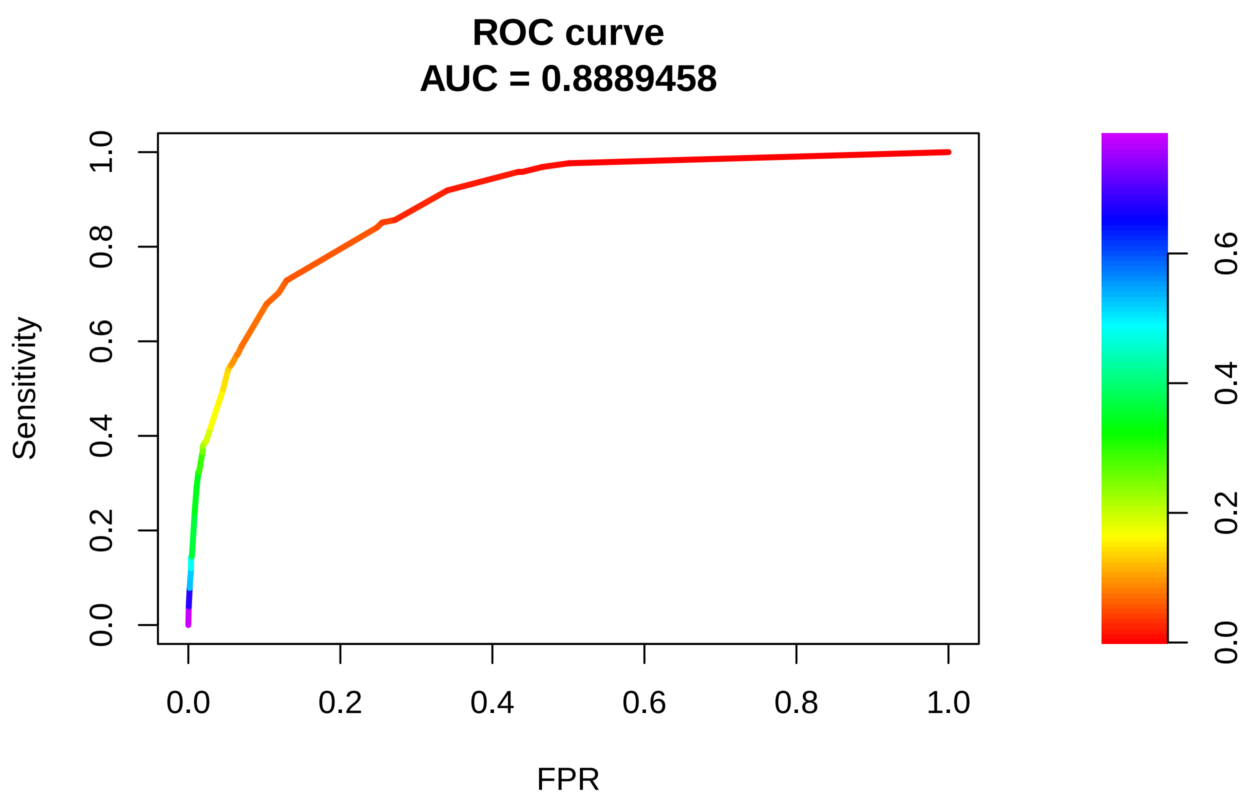
```
##    user  system elapsed
##  402.23    0.85  411.26
```

Depict the performance measures by running the following chunks.

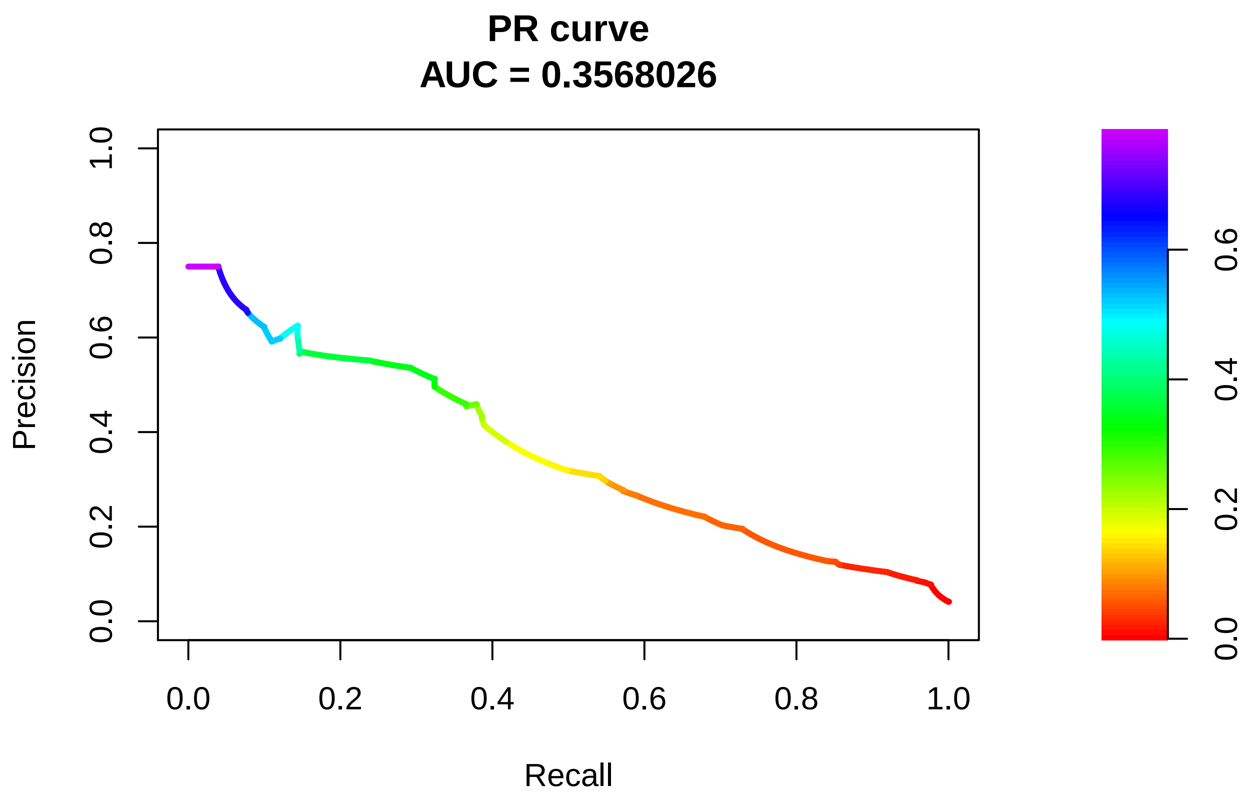
```
fitted.results.tree <- as.matrix(unlist(predict(c.tree,
                                  newdata = test, type='prob')))
fitted.prob.tree <- fitted.results.tree[seq_along(fitted.results.tree) %%2 == 0]
```

```
#Roc
fg.tree<-fitted.prob.tree[test$failure==1]
bg.tree<-fitted.prob.tree[test$failure==0]
```

```
roc_ctree <- roc.curve(scores.class0 = fg.tree,
                       scores.class1 = bg.tree,
                       curve = T)
plot(roc_ctree)
```




```
pr_ctree <- pr.curve(scores.class0 = fg.tree,
                     scores.class1 = bg.tree,
                     curve = T)
plot(pr_ctree)
```




```
fitted.ctree <- predict(c.tree,
                        newdata = test,
                        type='response')
f1_ctree <- f1_score(fitted.ctree,
                     test$failure,
                     positive.class="1")
```

```
balanced_accuracy_ctree <- balanced_accuracy(fitted.ctree, test$failure)
accuracy_ctree <- as.data.frame(rbind(postResample(as.double(fitted.ctree), test$failure)))
```

```
ctree_fit <- as.data.frame(cbind(roc_ctree$auc,
                                 pr_ctree$auc.integral,
                                 f1_ctree,
                                 balanced_accuracy_ctree,
                                 accuracy_ctree$Rsquared))
colnames(ctree_fit) <- c("AUC", "PR", "f1-score", "BACC", "Rsquared")
ctree_fit
```

```
##         AUC        PR f1-score     BACC   Rsquared
## 1 0.8889458 0.3568026      0.2 0.780396 0.06540008
```

Random Forest

Run a Random Forest analysis.

```r
set.seed(2020)

system.time({
rf <- randomForest(formula, data=train,
                   importance = FALSE,
                   ntree=200)
})
```

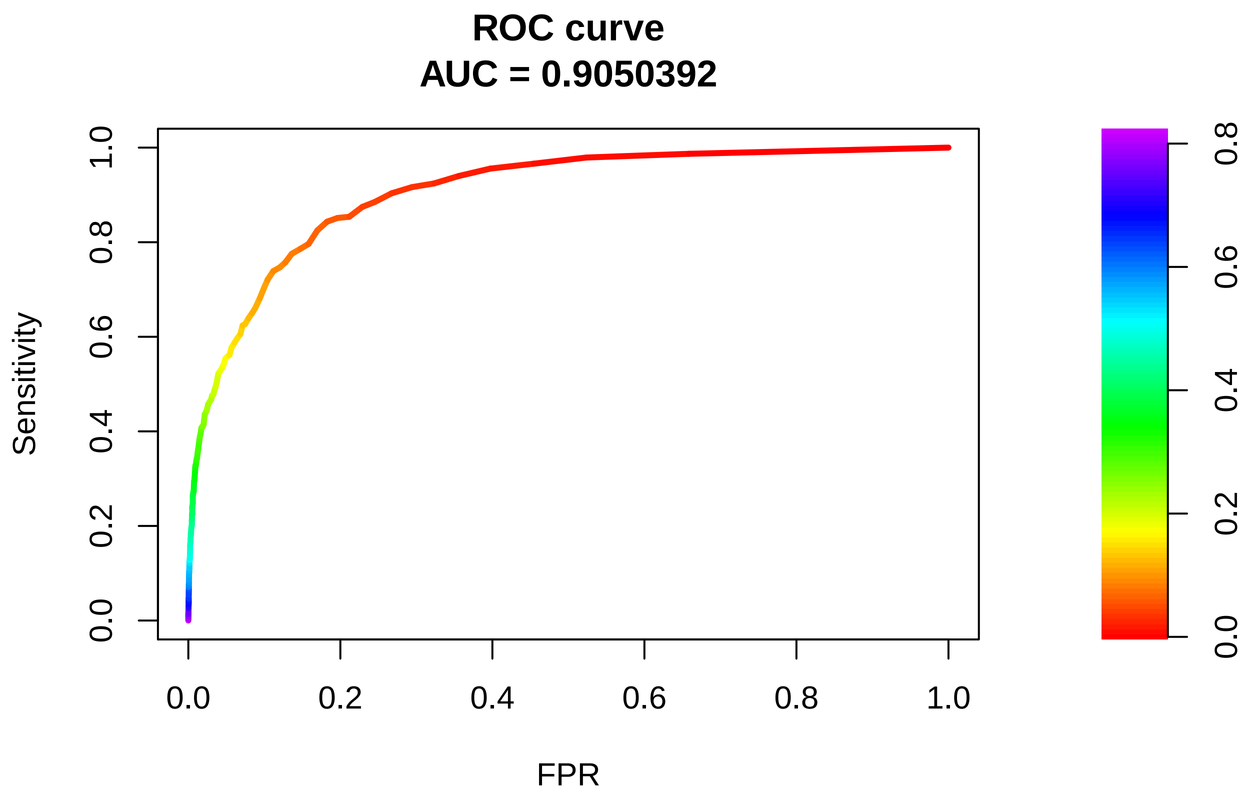
```
##    user  system elapsed
##  204.77    0.83  208.34
```

Depict the performance measures by running the following chunks.

```r
# Fitted results Random Forest
fitted.prob.rf <- predict(rf, newdata=test, type='prob')
fitted.prob.rf <- fitted.prob.rf[,2]
```

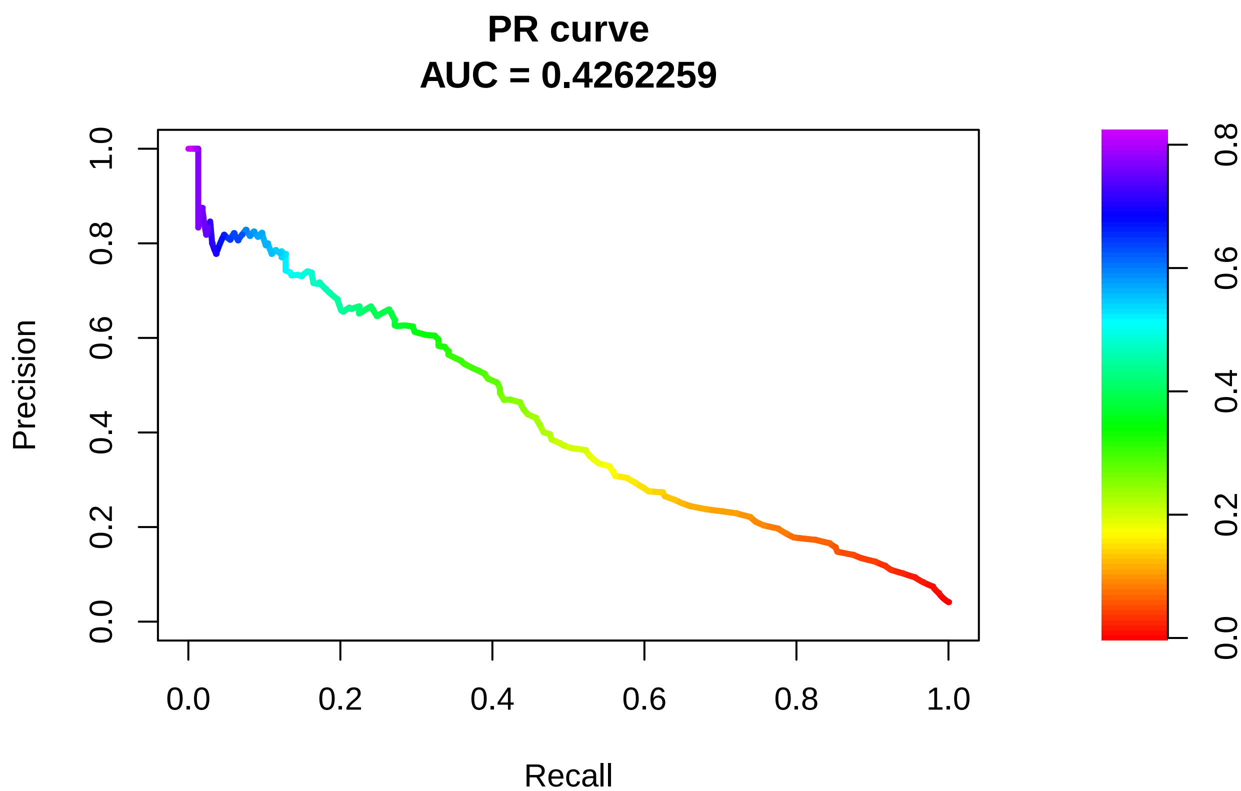
```r
#Roc
fg.rf<-fitted.prob.rf[test$failure==1]
bg.rf<-fitted.prob.rf[test$failure==0]
```

```r
roc_rf <- roc.curve(scores.class0 = fg.rf,
                    scores.class1 = bg.rf,
                    curve = T)
plot(roc_rf)
```




```r
pr_rf<-pr.curve(scores.class0 = fg.rf,
                scores.class1 = bg.rf,
```

```
                 curve = T)
plot(pr_rf)
```




```
fitted.rf <- ifelse(fitted.prob.rf > 0.5, 1, 0)
f1_rf <- f1_score(fitted.rf,
                  test$failure,
                  positive.class="1")
```

```
balanced_accuracy_rf <- balanced_accuracy(fitted.rf, test$failure)
accuracy_rf <- as.data.frame(rbind(postResample(as.double(fitted.rf),
                                                test$failure)))
```

```
rf_fit <- as.data.frame(cbind(roc_rf$auc,
                              pr_rf$auc.integral,
                              f1_rf, balanced_accuracy_rf,
                              accuracy_rf$Rsquared))
colnames(rf_fit) <- c("AUC", "PR", "f1-score", "BACC", "Rsquared")
rf_fit
```

```
##         AUC        PR  f1-score      BACC   Rsquared
## 1 0.9050392 0.4262259 0.2256637 0.8515472 0.09220139
```

Super Learner

Run a super learner analysis, by using an ensemble method with three learners: a logistic regression, a classification tree and a random forest.

```r
SL.library <- c("SL.glm",  "SL.randomForest", "SL.rpartPrune")
```

```r
train$X <-(train[predictors])
set.seed(123)
system.time({
  SL <- SuperLearner(Y=train$failure, X=train$X,
                     SL.library = SL.library,
                     verbose = FALSE,
                     method = "method.NNLS",
                     family = binomial())
})
```

```
##     user   system  elapsed
## 11308.47   385.36 12721.22
```

```r
coef(SL)
```

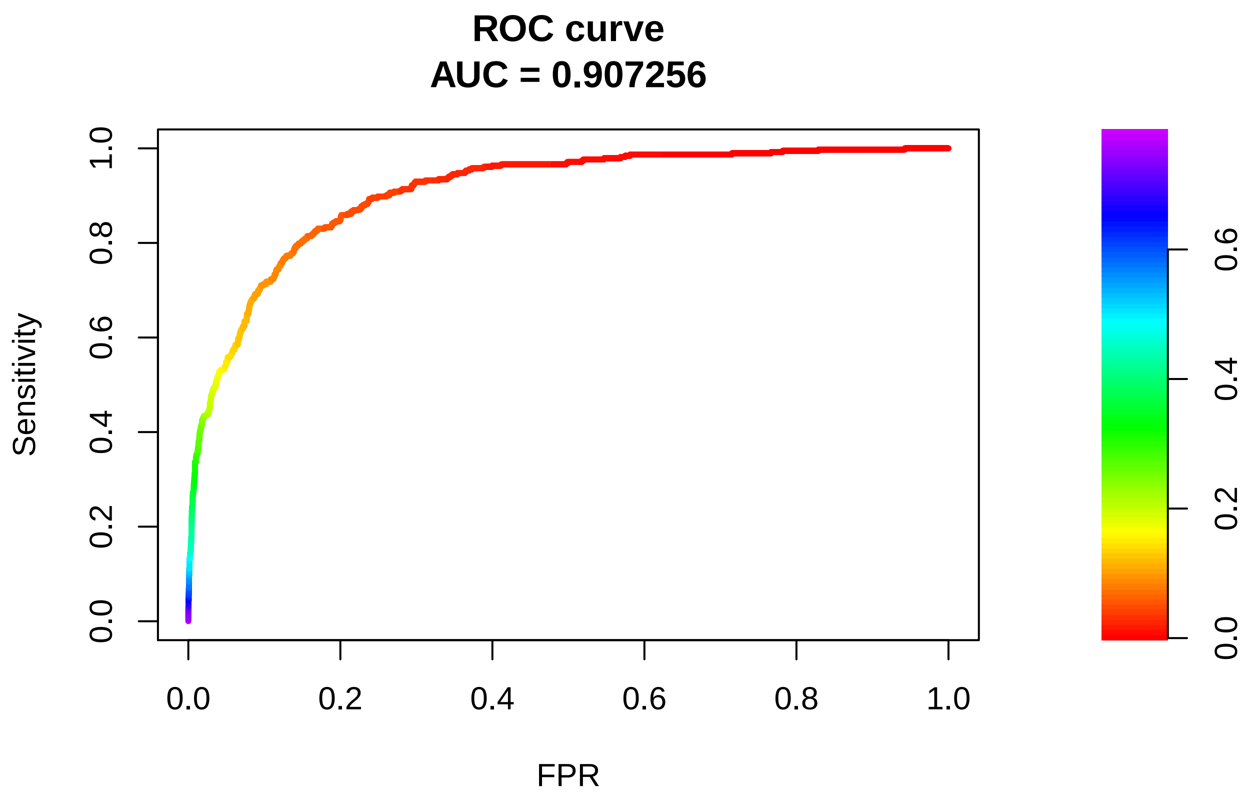
```
##          SL.glm_All SL.randomForest_All   SL.rpartPrune_All
##          0.18903816          0.78628692          0.02467493
```

Depict the performance measures by running the following chunks.

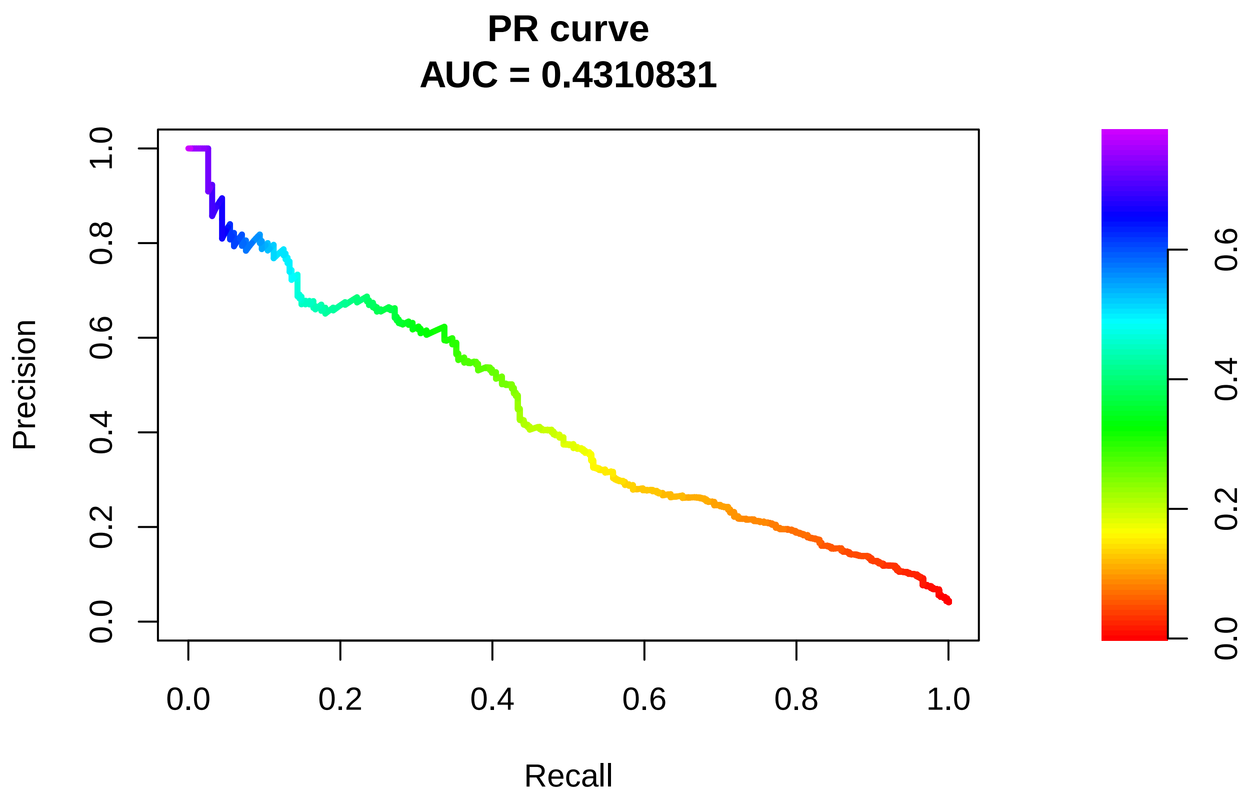
```r
test$X <-(test[predictors])
sl.fitted <- predict.SuperLearner(SL, test$X,
                                  X = train$X,
                                  Y = train$failure,
                                  onlySL = TRUE)
```

```r
#Roc
fg.sl<-sl.fitted$pred[test$failure==1]
bg.sl<-sl.fitted$pred[test$failure==0]
```

```r
roc_sl<-roc.curve(scores.class0 = fg.sl,
                  scores.class1 = bg.sl,
                  curve = T)
plot(roc_sl)
```




```
pr_sl<-pr.curve(scores.class0 = fg.sl,
                scores.class1 = bg.sl,
                curve = T)
plot(pr_sl)
```




```
fitted.sl <- ifelse(sl.fitted$pred > 0.5, 1, 0)
f1_sl <- f1_score(fitted.sl,
                  test$failure,
                  positive.class="1")
```

```
balanced_accuracy_sl <- balanced_accuracy(fitted.sl, test$failure)
accuracy_sl <- as.data.frame(rbind(postResample(as.double(fitted.sl),
                                                 test$failure)))
```

```
sl_fit <- as.data.frame(cbind(roc_sl$auc,
                              pr_sl$auc.integral,
                              f1_sl,
                              balanced_accuracy_sl,
                              accuracy_sl$Rsquared))
colnames(sl_fit) <- c("AUC", "PR", "f1-score", "BACC", "Rsquared")
sl_fit
```

```
##        AUC        PR  f1-score      BACC  Rsquared
## 1 0.907256 0.4310831 0.2232143 0.8665509 0.0944695
```

BART-mia

Run a Bayesian Additive Regression Tree analysis by using the overall data sample (no need to omit the observations with missing values).

```r
set.seed(2020)
sample <- sample(seq_len(nrow(data_italy)),
                 size = nrow(omitted),
                 replace=FALSE)
data_italy_bart <- data_italy[sample,]
```

Select the same number of observations as in the previous models for the training and testing samples.

```r
set.seed(2020)
train_sample <- sample(seq_len(nrow(data_italy_bart)),
                       size = nrow(data_italy_bart )*0.9, replace=FALSE)
train_bart <- data_italy_bart[train_sample,]
test_bart <- data_italy_bart[-train_sample,]
train_bart$X <- as.data.frame(train_bart[predictors])
test_bart$X <- as.data.frame(test_bart[predictors])
```

Run the analysis.

```r
system.time({
bart_machine<-bartMachine(train_bart$X,
                          as.factor(train_bart$failure),
                          use_missing_data=TRUE)
})
```

Depict the performance measures by running the following chunks.

```r
fitted.results.bart <- 1- round(predict(bart_machine,
                                        test_bart$X,
                                        type='prob'), 6)
```

```r
#Roc
fg.bart<-fitted.results.bart[test_bart$failure==1]
bg.bart<-fitted.results.bart[test_bart$failure==0]
```

```r
roc_bart<-roc.curve(scores.class0 = fg.bart,
                    scores.class1 = bg.bart,
                    curve = T)
plot(roc_bart)
```

```r
pr_bart<-pr.curve(scores.class0 = fg.bart,
                  scores.class1 = bg.bart,
                  curve = T)
plot(pr_bart)
```

```r
#Get Accurancy
fitted.bart <- ifelse(fitted.results.bart> 0.5, 1, 0)
f1_bart <- f1_score(fitted.bart,
                    test_bart$failure,
                    positive.class="1")
```

```r
balanced_accuracy_bart <- balanced_accuracy(fitted.bart, test_bart$failure)
accuracy_bart <- as.data.frame(rbind(postResample(as.double(fitted.bart),
                                                  test_bart$failure)))
```

```r
bart_fit <- as.data.frame(cbind(roc_bart$auc,
                                pr_bart$auc.integral,
                                f1_bart,
```

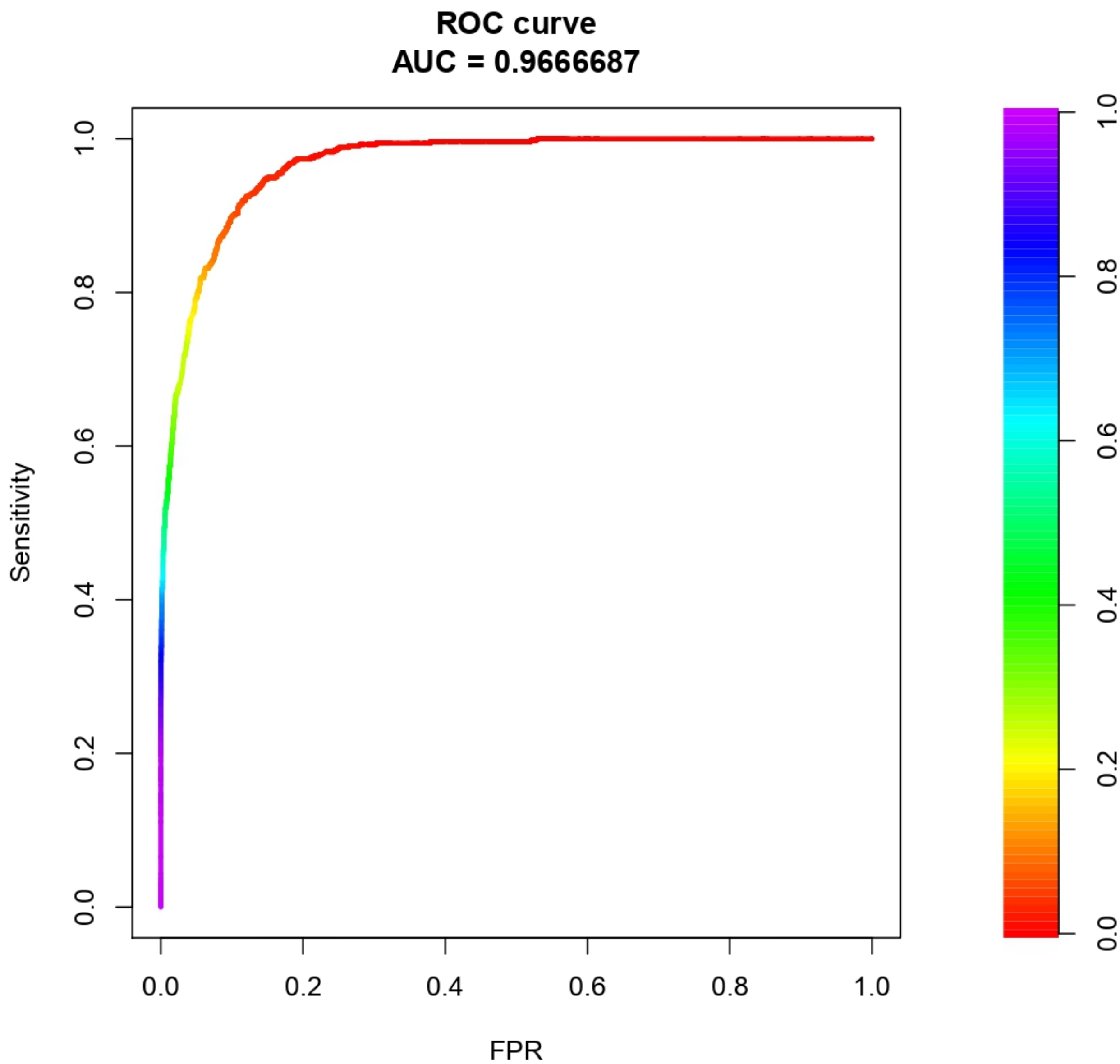


Figure 1: Area under the ROC curve, BART

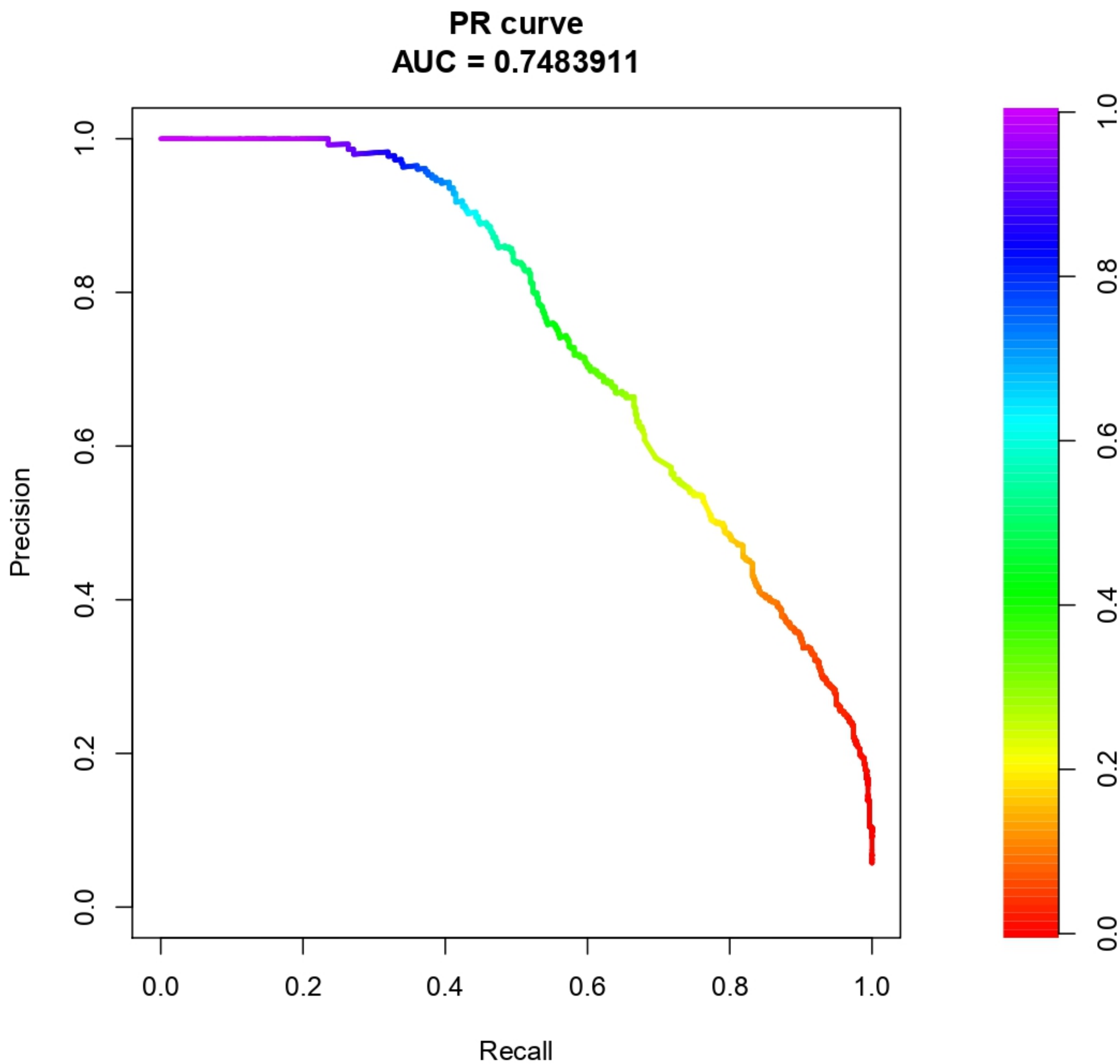


Figure 2: Area under the PR curve, BART

```
                                balanced_accuracy_bart,
                                accuracy_bart$Rsquared))
colnames(bart_fit) <- c("AUC", "PR", "f1-score", "BACC", "Rsquared")
```

Save Results

```
model_results <- rbind(logit_fit, ctree_fit, rf_fit, sl_fit, bart_fit)
write.csv(model_results, file = "model_results.csv")
```

Model Comparison

Here, we run an empirical horse race where we define two competitors (benchmark or "usual methods") and we compare them with our preferred BART methodology. Natural candidates are "default probability predictors" (credit-ratings type of measures) such as:

1. Altman Z-score;
2. Distance-to-Default.

As these measures do not provide direct predictions for failed firms, we create a series of dummy variables in the following way:

$$Z - dummy_i = \begin{cases} 1 & \text{if } Z - score_i \leq q, \\ 0 & \text{otherwise} \end{cases}$$

and

$$DtD - dummy_i = \begin{cases} 1 & \text{if } DtD_i \leq q, \\ 0 & \text{otherwise} \end{cases}$$

where q is the 1st to 10th percentile distribution of the Z-score and the DtD measures, respectively.

By doing so, we assume to be predicted as "failed", those observations with values on the left tails of the Z and DtD measures.

We create these variables on the testing set and then we compare their performance, in terms of precision and false discovery rate (FDR), with the one of BART.

Z-score

```
precision_bart <- as.data.frame(t(model_compare(fitted.bart,
                                  test_bart$failure, "1")))
colnames(precision_bart) <- c("Precision BART", "FDR BART")
write.csv(precision_bart, "precision_and_fdr_bart.csv")
```

```
seq <- seq(1, 10, 1)
precision_zscore <- matrix(NA, ncol = 2, nrow = length(seq))
for(i in seq) {
  failed_zscore <- ifelse(test$Z_score <= quantile(test$Z_score, i/100),
                                                          1, 0)
  precision_zscore[i, c(1:2)] <- t(model_compare(failed_zscore,
                                               test$failure, "1"))
}
```

Merton's Distance-to-Default (DtD)

The average equity volatility in the considered time series is 27.9120 (Bank of Italy), while for the average risk free interest rate we use the long term government bond yields" EMU (Eurostat) that has a value of 4.2937.

```
dtd_score <- DtD(mcap = test$shareholders_funds, debt = test$long_term_debt, vol = 27.9120, r = 4.2937)
```

```
seq <- seq(1, 10, 1)
precision_dtd_score <- matrix(NA, ncol = 2, nrow = length(seq))
for(i in seq) {
  failed_dtd_score <- ifelse(dtd_score <= quantile(dtd_score, i/100), 1, 0)
  precision_dtd_score[i, c(1:2)] <- t(model_compare(failed_dtd_score,
                                                    test$failure, "1"))
}
```

```
precision_models <- as.data.frame(cbind(precision_dtd_score, precision_zscore))
colnames(precision_models) <- c("Precision DtD",
                                "FDR DtD",
                                "Precision Z-score",
                                "FDR Z-score")
write.csv(precision_models, "precision_and_fdr_scores.csv")
```