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.

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

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

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

```
library(mice)
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"),]
```

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.

```
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",
                "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")
# no int_paid due to multicollinearity issue (no prediction from MICE)
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. This dataset is of the same size of the dataset used in the main analysis omitting the missing observations.

```
set.seed(2020)
data_size <- sample(seq_len(nrow(data_lagged)), size = 92819*0.1)
omitted <- data_lagged[data_size,]
```

For the data imputation we use the with multiple imputations using the CART (Friedman et al., 1984) Fully Conditional Specification (FCS) developed by (Van Buuren et al., 2010) and implemented in the `R` package `MICE`.

```
imputed_data <- mice(omitted, m = 1, maxit = 1, method = 'cart', seed = 500)
```

```
##
##  iter imp variable
##   1   1  shareholders_funds  added_value  cash_flow  ebitda  fin_rev  liquidity_ratio  total_assets
```

```
## Warning: Number of logged events: 37
```

```
data_imp <- complete(imputed_data, 1)
data_imp <- as.data.frame(cbind(omitted$failure, data_imp))
index <- sample(seq_len(nrow(data_imp)), size = nrow(data_imp)*0.9)
train <- data_imp[index,]
test <- data_imp[-index,]
```

Logit

Run a Logistic regression analysis.

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

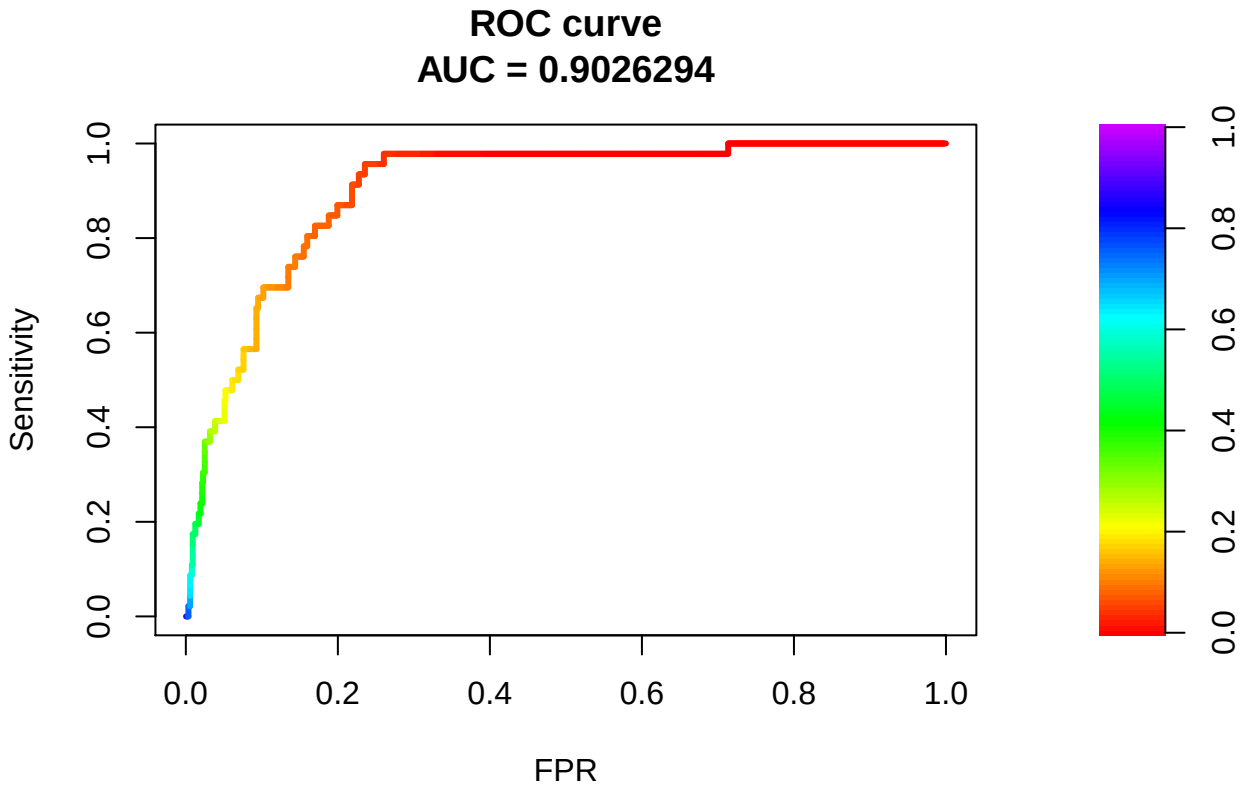
```
##    user  system elapsed
##    1.33    0.02    1.50
```

Depict the performance measures by running the following chunks.

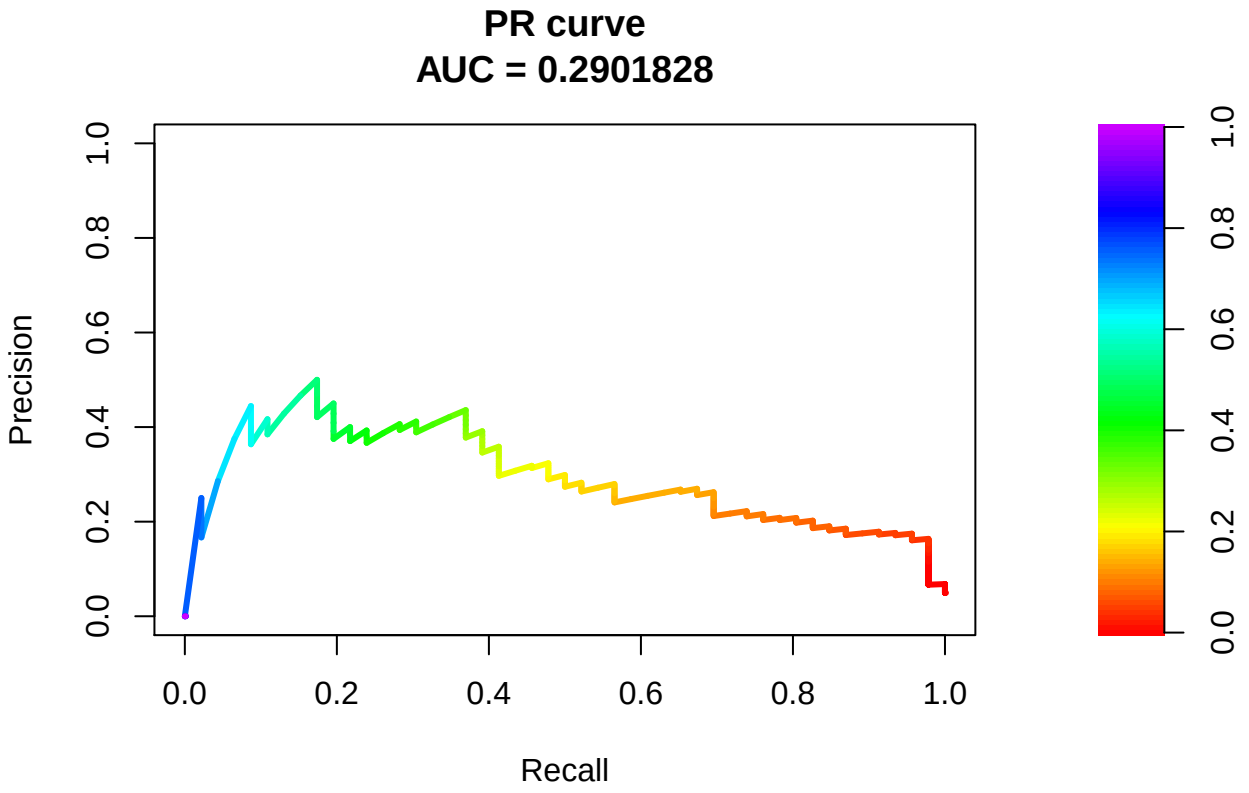
```
fitted.prob.logit <- predict(logit, newdata=test, type='response')
```

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

```
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 0.9026294 0.2901828 0.2539683 0.7144608 0.07022326
```

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))
})
```

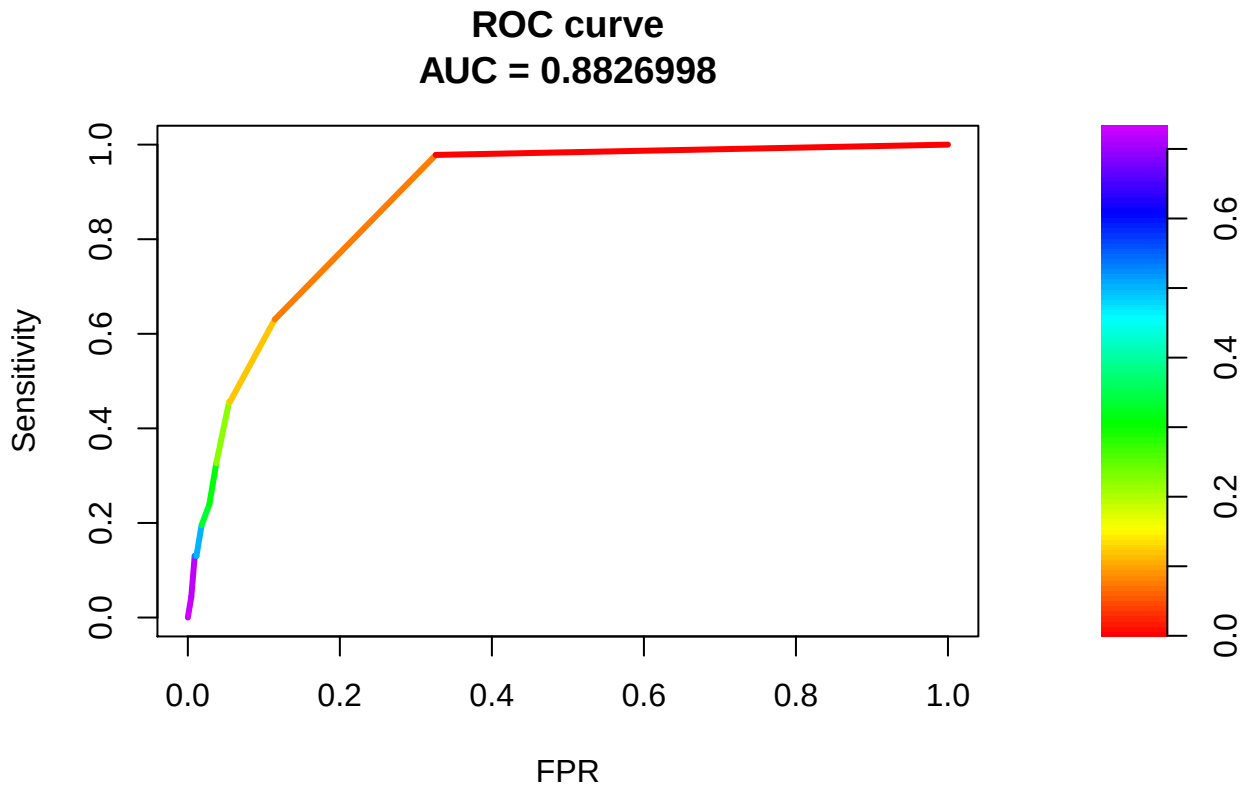
```
##    user  system elapsed
##   18.67    0.06   20.65
```

Depict the performance measures by running the following chunks.

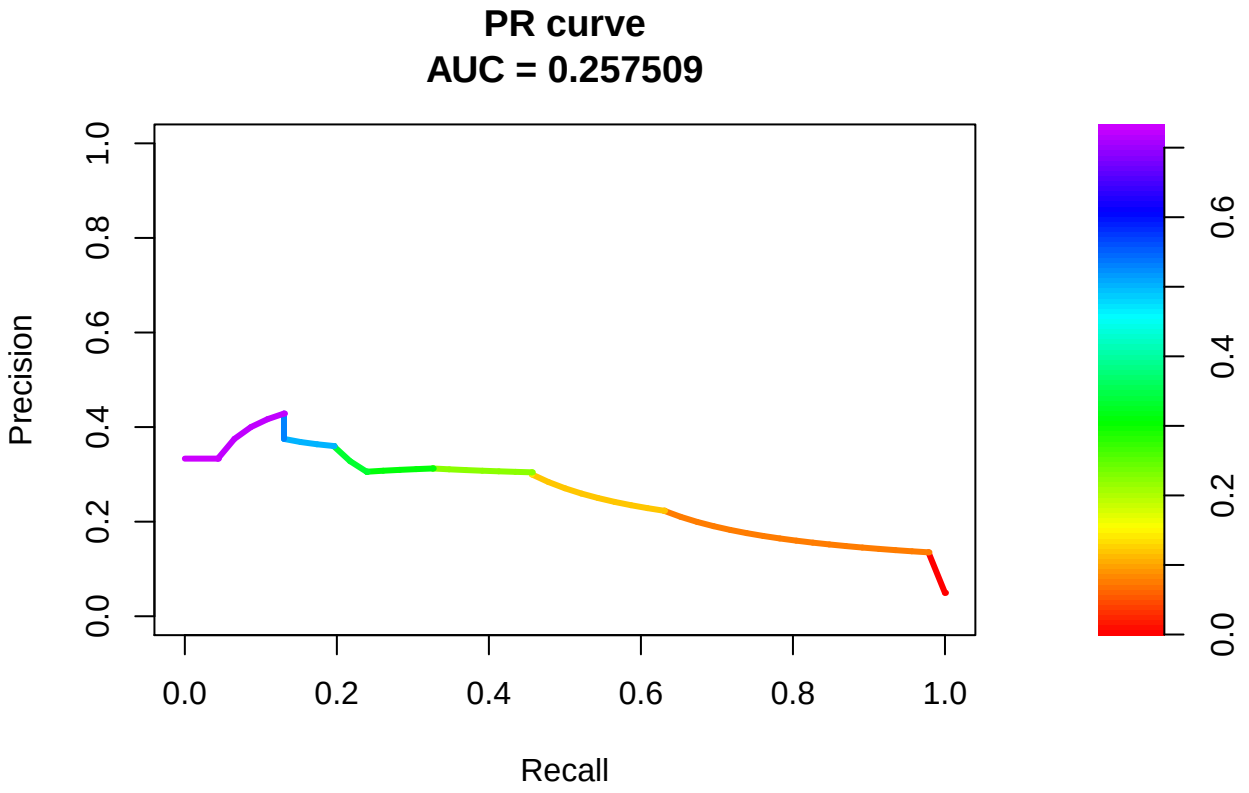
```
fitted.results.tree <- as.matrix(unlist(predict(c.tree,
                                  newdata = test, type='prob')))
fitted.prob.tree <- fitted.results.tree[seq_along(fitted.results.tree) %%2 == 0]
```

```
#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 0.8826998 0.257509 0.1935484 0.6655942 0.03944777
```

Random Forest

Run a Random Forest analysis.

```r
set.seed(2020)

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

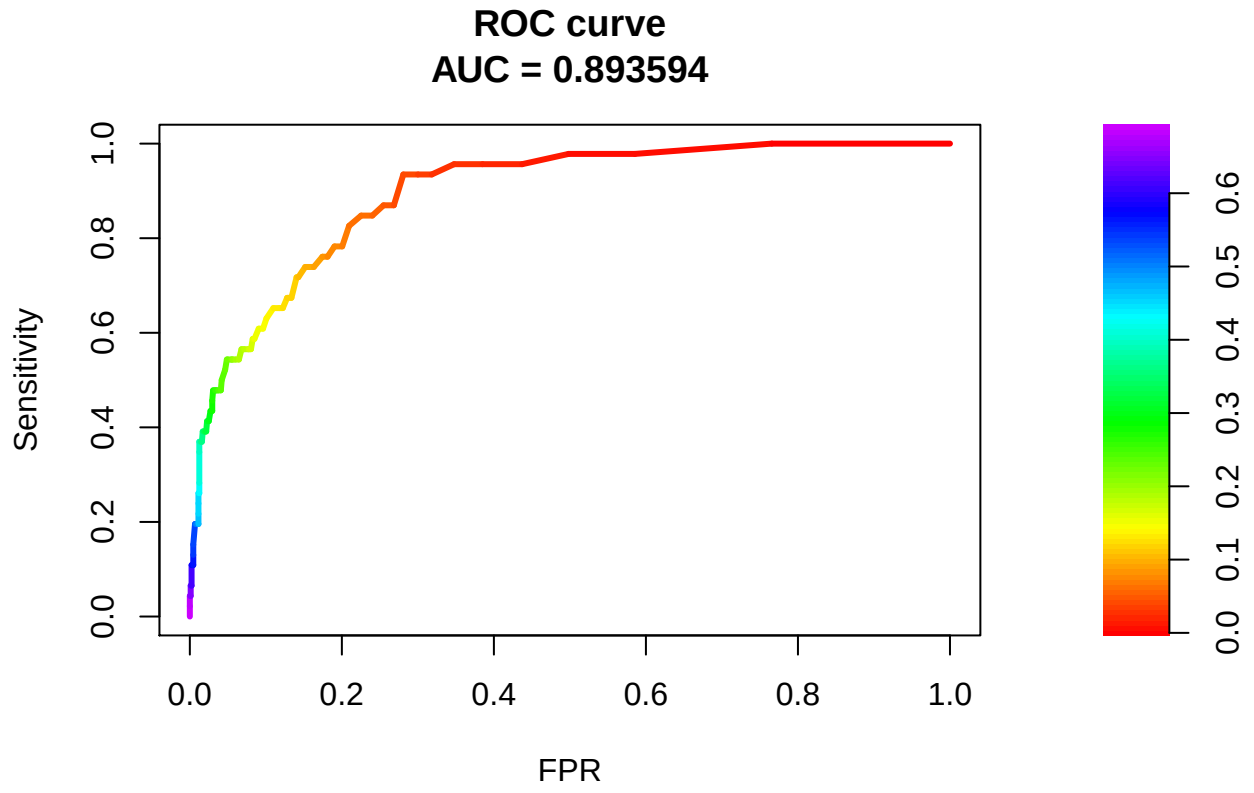
```
##    user  system elapsed
##    9.23    0.05   10.00
```

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]
```

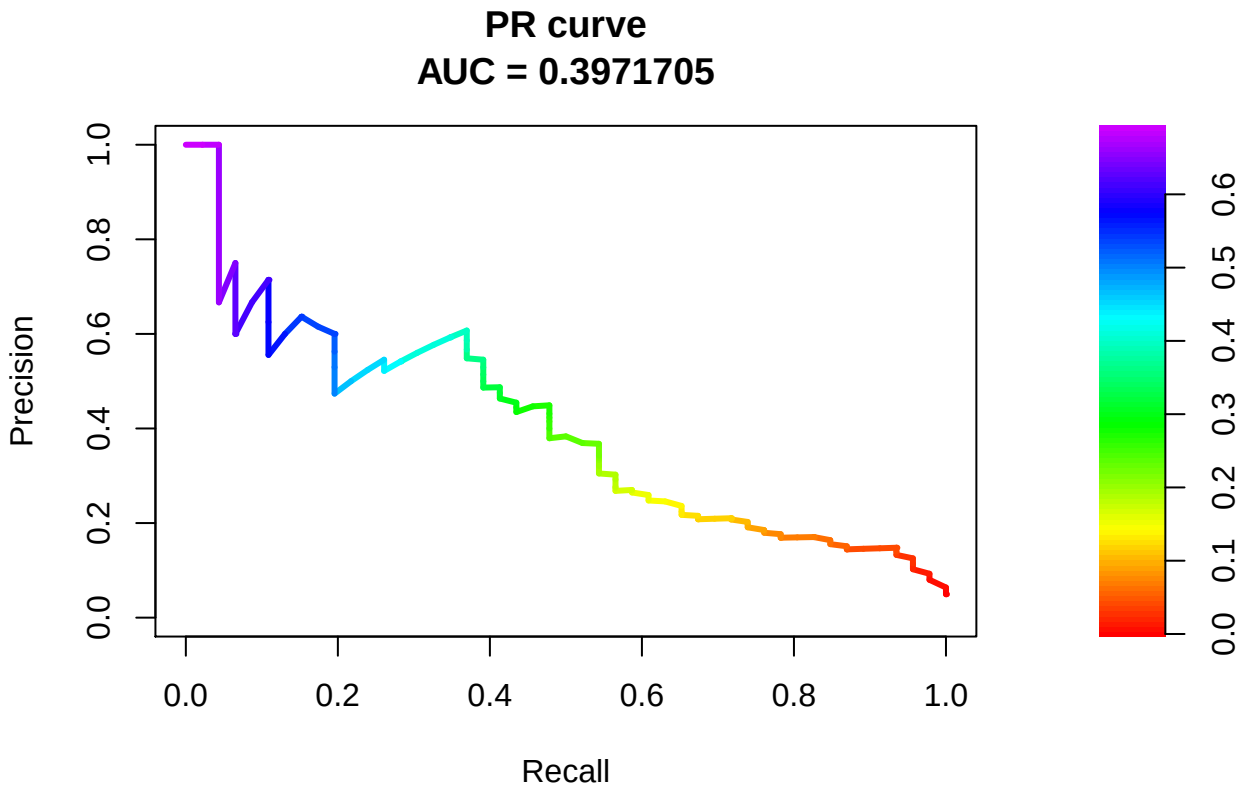
```r
#Roc
fg.rf<-fitted.prob.rf[test$failure==1]
bg.rf<-fitted.prob.rf[test$failure==0]
```

```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 0.893594 0.3971705 0.2857143 0.7444208 0.091214
```

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.

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

```
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())
})
```

```
## Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = object$xlevels) :
##   factor area has new levels
```

```
##    user  system elapsed
##  434.70    5.86  477.24
```

```
coef(SL)
```

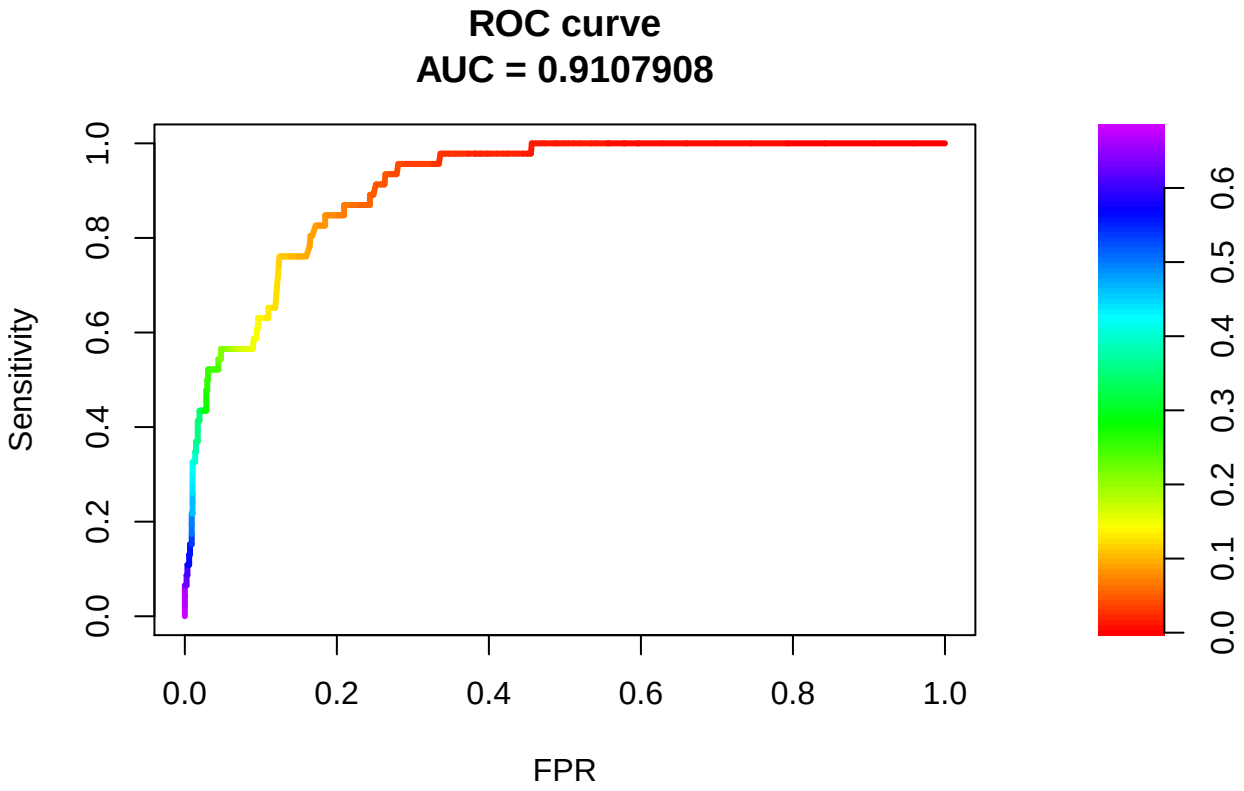
```
##          SL.glm_All SL.randomForest_All   SL.rpartPrune_All
##          0.00000000          0.96255974          0.03744026
```

Depict the performance measures by running the following chunks.

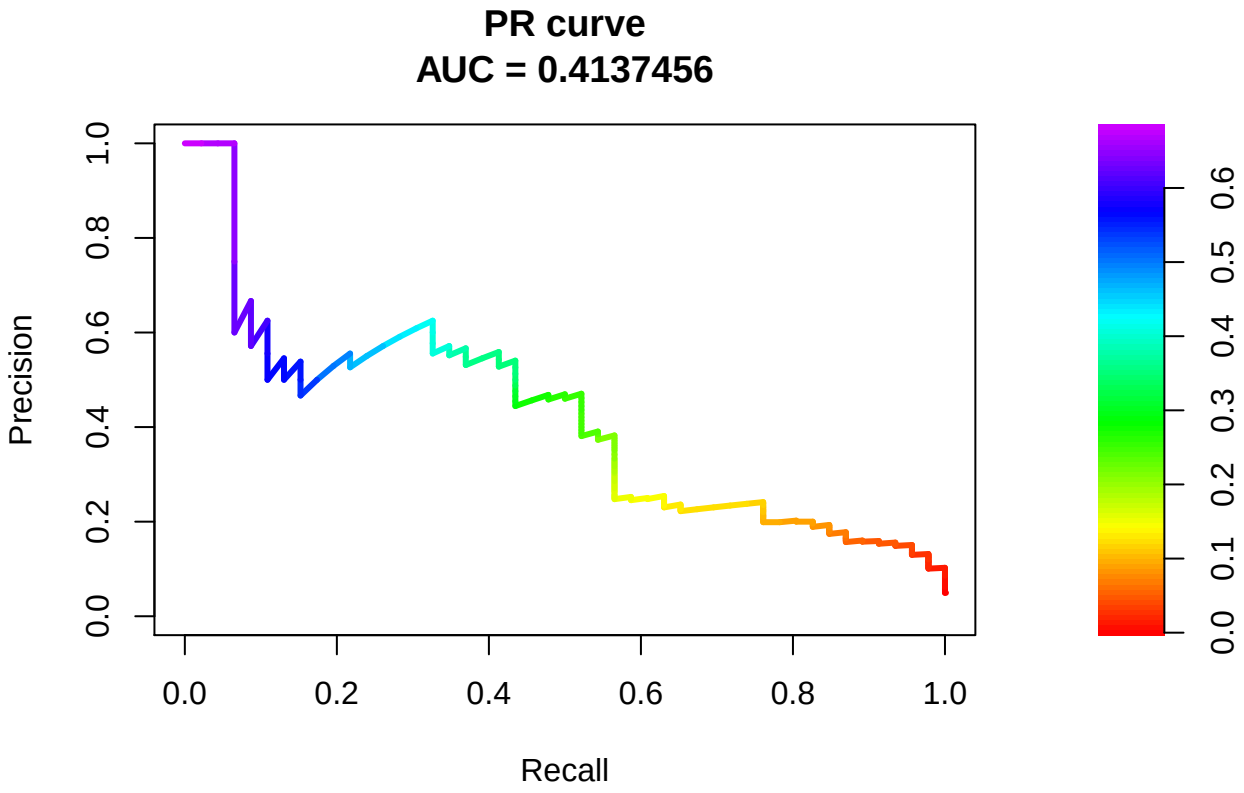
```
test$X <-(test[predictors])
sl.fitted <- predict.SuperLearner(SL, test$X,
                                  X = train$X,
                                  Y = train$failure,
                                  onlySL = TRUE)
```

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

```
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 0.9107908 0.4137456 0.2857143 0.7444208 0.091214
```

BART-mia

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

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

```
train <- omitted[index,]
test <- omitted[-index,]
train$X <- as.data.frame(train[predictors])
test$X <- as.data.frame(test[predictors])
```

Run the analysis.

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

```
## bartMachine initializing with 50 trees...
## bartMachine vars checked...
## bartMachine java init...
## bartMachine factors created...
## bartMachine before preprocess...
## bartMachine after preprocess... 67 total features...
## bartMachine sigsq estimated...
## bartMachine training data finalized...
## Now building bartMachine for classification ...Covariate importance prior ON. Missing data feature O
## evaluating in sample data...done
```

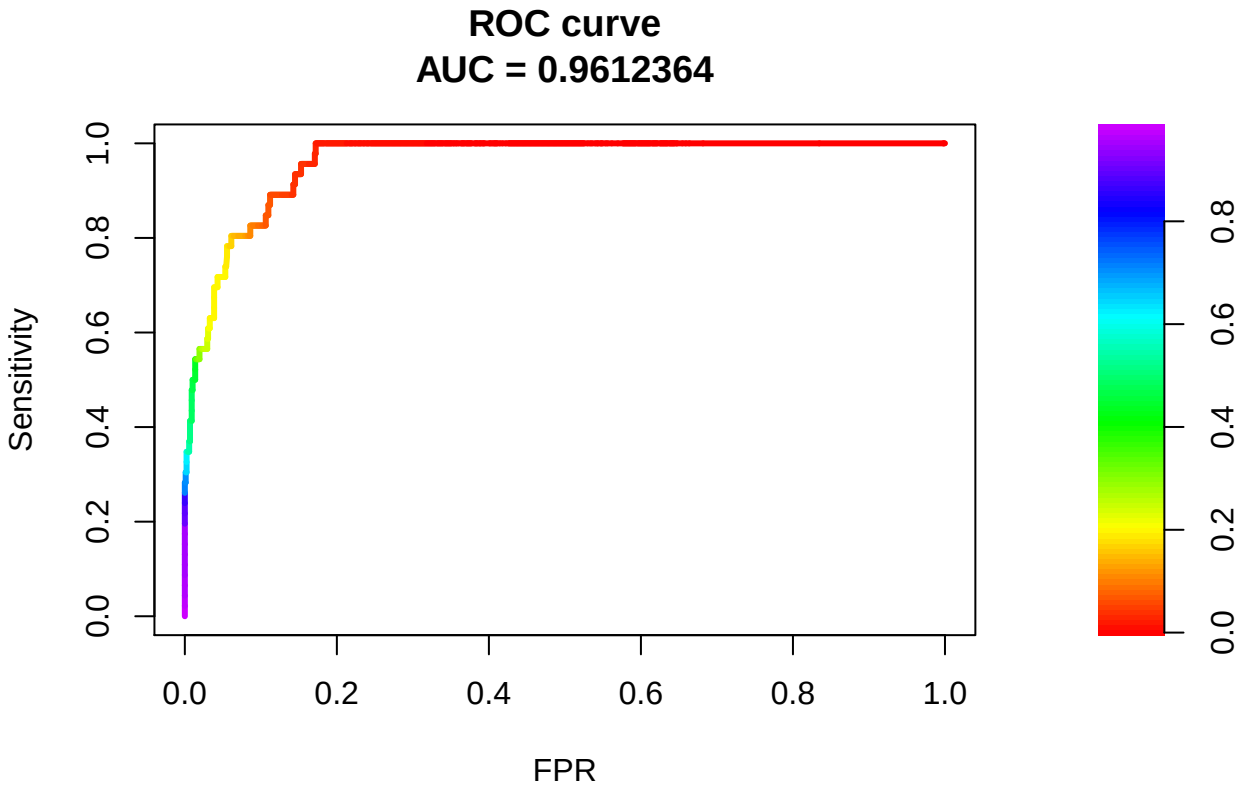
```
##    user  system elapsed
##   91.94    5.31   98.64
```

Depict the performance measures by running the following chunks.

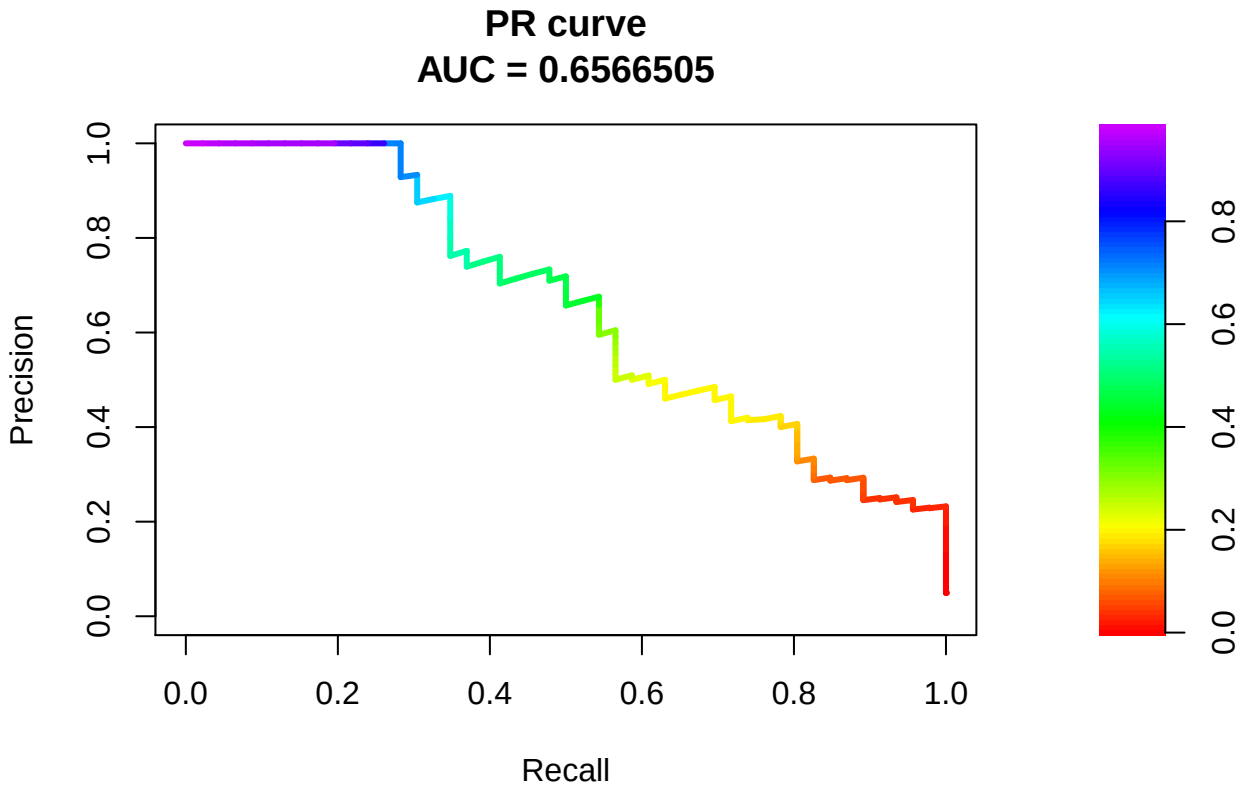
```
fitted.results.bart <- 1- round(predict(bart_machine,
                                        test$X,
                                        type='prob'), 6)
```

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

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




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




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

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

```
bart_fit <- as.data.frame(cbind(roc_bart$auc,
                                pr_bart$auc.integral,
                                f1_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_results
```

```
##          AUC        PR  f1-score      BACC   Rsquared
## 1  0.9026294 0.2901828 0.2539683 0.7144608 0.07022326
```

```
## 11 0.8826998 0.2575090 0.1935484 0.6655942 0.03944777
## 12 0.8935940 0.3971705 0.2857143 0.7444208 0.09121400
## 13 0.9107908 0.4137456 0.2857143 0.7444208 0.09121400
## 14 0.9612364 0.6566505 0.5205479 0.8368851 0.27219202
```