Lecture 4: Predictive Modeling

Falco J. Bargagli Stoffi

11/06/2020

In some scenarios we may be interested in building a statistical model to predict an outcome. In this case, for instance, we may want to use different models to predict the location of a firm. The Compustat data entail US and Canadian enterprises. In the next chunks of code I will build five different models (logistic regression, CART, Conditional Inference Tree, Random Forest, Bayesian Additive Regression Trees) to predict the location of the firm.

Moreover, I will provide details on the most widely used performance measures in the case of a classification problem.

```r
library(readxl)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(bartMachine)
```

```
## Loading required package: rJava

## Loading required package: bartMachineJARs

## Loading required package: car

## Loading required package: carData

## Loading required package: missForest

## Loading required package: foreach

## Loading required package: itertools

## Loading required package: iterators

## Welcome to bartMachine v1.2.3! You have 0.48GB memory available.
##
## If you run out of memory, restart R, and use e.g.
## 'options(java.parameters = "-Xmx5g")' for 5GB of RAM before you call
```

```
## 'library(bartMachine)'.
```

```r
library(PRROC)
library(rpart)
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

##
## Attaching package: 'modeltools'

## The following object is masked from 'package:car':
##
##     Predict

## The following object is masked from 'package:rJava':
##
##     clone

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

First things first, let's upload the Compustat data and perform a naive trimming of the data, excluding all the missing observations.

```r
data <- read_excel("G:\\Il mio Drive\\Econometrics Lab\\Data\\Compustat Data.xlsx")
data <- data[, !names(data) %in% c("Interest Expense - Total (Financial Services)",
                            "Net Interest Income", "Nonperforming Assets - Total")]
data_clean <- na.omit(data)
```

Before running the analyses, I restict the set of predictors to the following variables and I create a dummy variable that assumes value 1 if the firm is located in the US and 0 if it is located in Canada.

```r
myvariables <- c("ISO Currency Code",
                 "Assets - Total", "Average Short-Term Borrowings",
                 "Current Assets - Total", "Long-Term Debt Due in One Year",
                 "Debt in Current Liabilities - Total", "Employees",
                 "Earnings Before Interest and Taxes", "Liabilities - Total",
                 "Net Income (Loss)", "In Process R&D Expense",
                 "GIC Sectors", "Standard Industry Classification Code")
data_prediction <- data_clean[myvariables]
data_prediction$iso_code <- ifelse(data_prediction$`ISO Currency Code`=="USD", 0, 1)
data_prediction <- data_prediction[, !names(data_prediction) %in% c("ISO Currency Code")]
```

In order to check how good are the five models, I randomly split the data into two disjoint sets: a training set that I will use to build the model and a test set that I will use to validate the quality of the model's prediction.

```
set.seed(123)
index <- sample(seq_len(nrow(data_prediction)),
                    size = nrow(data_prediction)*0.5)

train <- data_prediction[index,]
test <- data_prediction[-index,]
```

Moreover, I am renaming the variables in the dataset and constructing the "formula" that I will use for all the predictive models that I will run.

```
colnames(train) <- c("assets", "short_term_borrow",
                     "current_assets", "debt",
                     "debt_liabilities", "employees",
                     "EBIT", "liabilities",
                     "net_income", "r_d",
                     "gic", "SICC", "iso_code")
colnames(test) <- c("assets", "short_term_borrow",
                     "current_assets", "debt",
                     "debt_liabilities", "employees",
                     "EBIT", "liabilities",
                     "net_income", "r_d",
                     "gic", "SICC", "iso_code")
predictors <- c("assets", "short_term_borrow",
                     "current_assets", "debt",
                     "debt_liabilities", "employees",
                     "EBIT", "liabilities",
                     "net_income", "r_d",
                     "gic", "SICC")
formula <- as.formula(paste("as.factor(iso_code) ~",
                            paste(predictors, collapse="+")))
formula
```

```
## as.factor(iso_code) ~ assets + short_term_borrow + current_assets +
##     debt + debt_liabilities + employees + EBIT + liabilities +
##     net_income + r_d + gic + SICC
```

Logistic Regression

The first model that I run is a logistic regression with the inclusion of all the covariates.

```
logit<-glm(formula, data= train, family=binomial(link='logit'))
```

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

```
summary(logit)
```

```
##
## Call:
## glm(formula = formula, family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4832  -0.4930  -0.3027  -0.1754   3.1015
##
```

```
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)    
## (Intercept)        8.663e-01  1.508e-01   5.743 9.29e-09 ***
## assets             4.599e-04  3.459e-04   1.330 0.183633    
## short_term_borrow -1.684e-02  9.810e-03  -1.716 0.086105 .  
## current_assets    -2.555e-03  7.378e-04  -3.463 0.000535 ***
## debt              -6.835e-03  6.994e-03  -0.977 0.328398    
## debt_liabilities   1.073e-02  6.779e-03   1.582 0.113545    
## employees         -4.996e-02  3.530e-02  -1.415 0.156980    
## EBIT               2.414e-03  2.280e-03   1.059 0.289682    
## liabilities       -8.082e-04  6.295e-04  -1.284 0.199163    
## net_income        -4.237e-04  2.352e-03  -0.180 0.857055    
## r_d                1.391e+02  5.892e+03   0.024 0.981162    
## gic               -4.632e-02  6.250e-03  -7.411 1.26e-13 ***
## SICC              -3.329e-04  3.724e-05  -8.938  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 2140.2  on 2869  degrees of freedom
## Residual deviance: 1669.4  on 2857  degrees of freedom
## AIC: 1695.4
## 
## Number of Fisher Scoring iterations: 25
```

To get the accuracy of the model I first get the predicted probabilities, then impute the values for the outcome variable.

```r
# Accurancy from cv
fitted.results.logit <- predict(logit, newdata = test, type='response')
fitted.logit <- ifelse(fitted.results.logit >= 0.5, 1, 0)
head(fitted.logit)
```

```
## 1 2 3 4 5 6 
## 0 0 0 0 0 0
```

Once I get the predicted (or fitted values) for this model, I can evaluate its performance using a number of different performance measures. Below the functions to compute the F-1 Score and the Balanced Accuracy.

```r
## F1- Score
# predicted: vector of predicted values
# expected: vector of observed value
# positive.class: class of binary predictions we are mostly interested in (e.g., "1", "0")

f1_score <- function(predicted, expected, positive.class) {

  # Generate Confusion Matrix
  c.matrix = as.matrix(table(expected, predicted))

  # Compute Precision
  precision <- diag(c.matrix) / colSums(c.matrix)

  # Compute Recall
  recall <- diag(c.matrix) / rowSums(c.matrix)
```

```r
  # Compute F-1 Score
  f1 <-  ifelse(precision + recall == 0, 0, 2*precision*recall/(precision + recall))

  # Extract F1-score for the pre-defined "positive class"
  f1 <- f1[positive.class]

  # Assuming that F1 is zero when it's not possible compute it
  f1[is.na(f1)] <- 0

  # Return F1-score
  return(f1)
}

## Balanced Accuracy (BACC)
# predicted: vector of predicted values
# expected: vector of observed value

balanced_accuracy <- function(predicted, expected) {

  # Generate Confusion Matrix
  c.matrix = as.matrix(table(predicted, expected))

  # First Row Generation
  first.row <- c.matrix[1,1] / (c.matrix[1,1] + c.matrix[1,2])

  # Second Row Generation
  second.row <- c.matrix[2,2] / (c.matrix[2,1] + c.matrix[2,2])

  # # "Balanced" proportion correct (you can use different weighting if needed)
  acc <- (first.row + second.row)/2

  # Return Balanced Accuracy
  return(acc)
}
```

```r
# RMSE
caret::postResample(fitted.logit, test$iso_code)
```

```
##       RMSE   Rsquared        MAE
## 0.36476662 0.02285701 0.13305468
```

```r
# For good predictive model the MAE and RMSE values should be low


# Confusion Matrix
confusionMatrix(data = as.factor(fitted.logit),
                reference = as.factor(test$iso_code))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2473  376
##          1    6   16
##
```

```
##                Accuracy : 0.8669
##                  95% CI : (0.854, 0.8792)
##     No Information Rate : 0.8635
##     P-Value [Acc > NIR] : 0.3045
##
##                   Kappa : 0.0637
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99758
##             Specificity : 0.04082
##          Pos Pred Value : 0.86802
##          Neg Pred Value : 0.72727
##              Prevalence : 0.86346
##          Detection Rate : 0.86137
##    Detection Prevalence : 0.99234
##       Balanced Accuracy : 0.51920
##
##        'Positive' Class : 0
##
```

```r
# Balanced Accuracy
balanced_accuracy_logit<-balanced_accuracy(fitted.logit, test$iso_code)
balanced_accuracy_logit
```

```
## [1] 0.7976483
```

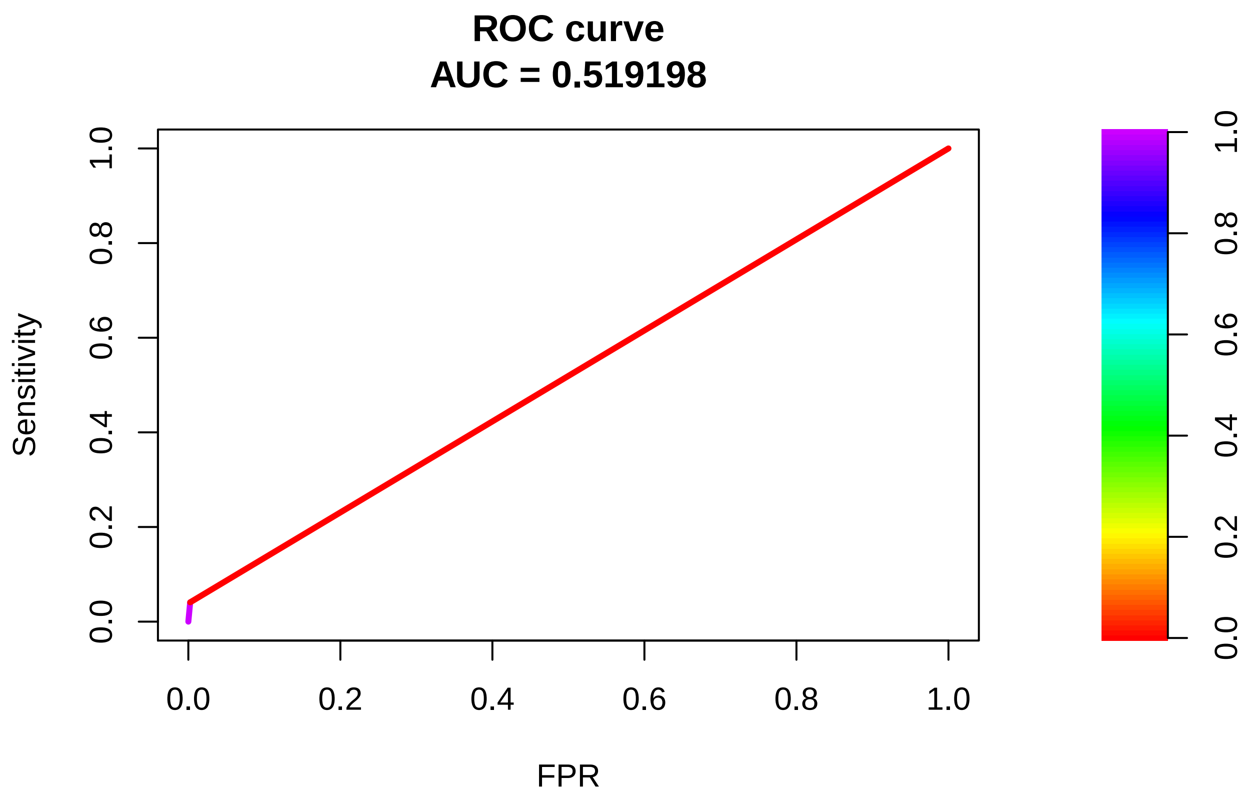
```r
# F1-Score
f1_logit_1 <- f1_score(fitted.logit,
                     test$iso_code,
                     positive.class="1")
f1_logit_1
```

```
##          1
## 0.07729469
```

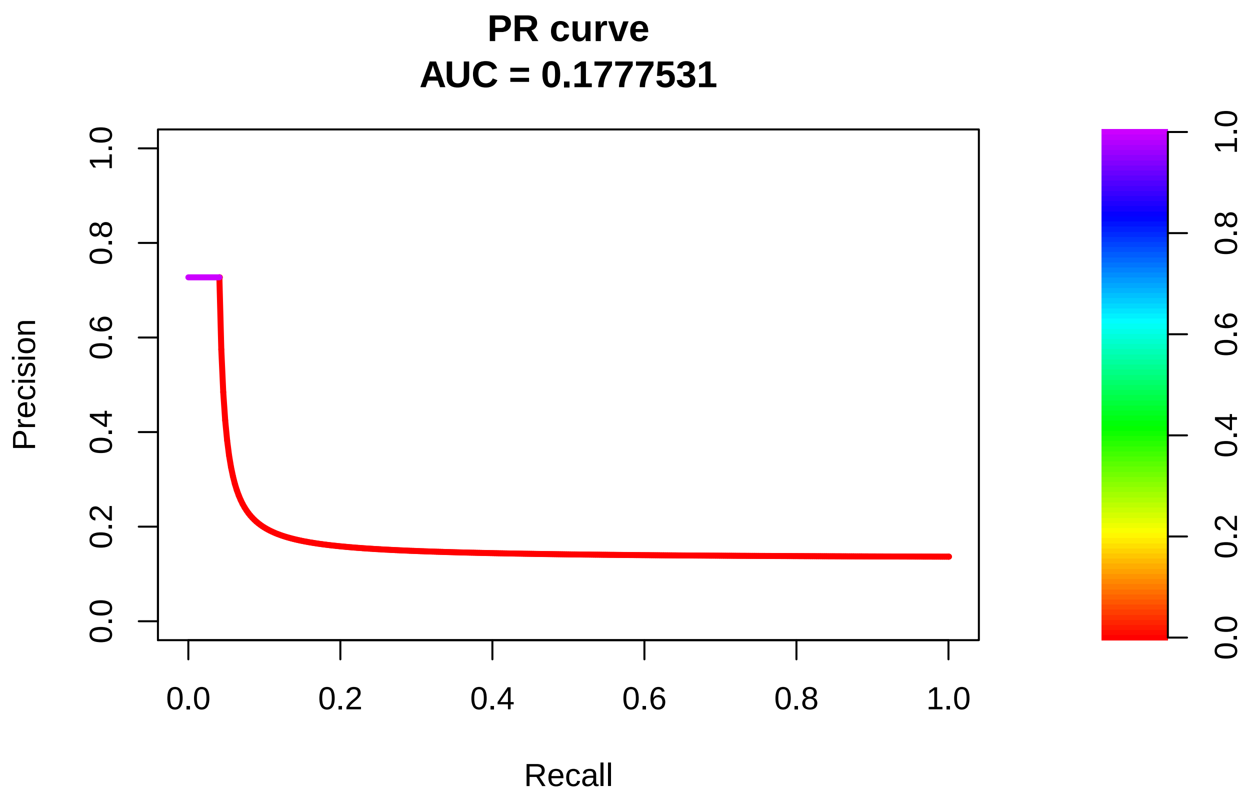
```r
f1_logit_0 <- f1_score(fitted.logit,
                     test$iso_code,
                     positive.class="0")
f1_logit_0
```

```
##         0
## 0.9283033
```

```r
# ROC Curve and PR- Curve
fg.logit <- fitted.logit[test$iso_code==1]
bg.logit <- fitted.logit[test$iso_code==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)
```



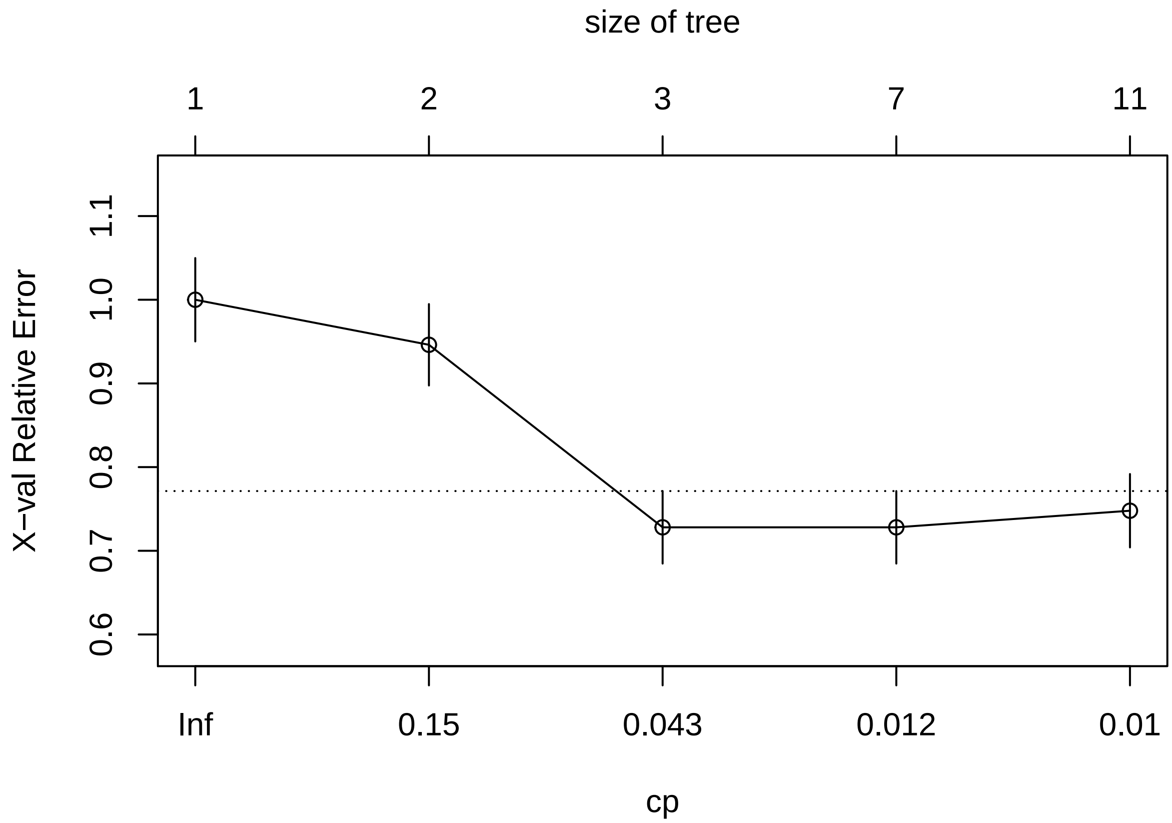
Classification and Regression Tree

The second model that I run is a classification and regression tree from the "rpart" package in R.

```
rpart <- rpart(formula, data=train, method="class")
printcp(rpart) # display the results
```

```
##
## Classification tree:
## rpart(formula = formula, data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] assets            current_assets   debt_liabilities EBIT
## [5] gic               liabilities      SICC
##
## Root node error: 353/2870 = 0.123
##
## n= 2870
##
##         CP nsplit rel error  xerror     xstd
## 1 0.158640      0   1.00000 1.00000 0.049844
## 2 0.133144      1   0.84136 0.94618 0.048667
## 3 0.014164      2   0.70822 0.72805 0.043333
## 4 0.010387      6   0.65156 0.72805 0.043333
## 5 0.010000     10   0.60907 0.74788 0.043860
```

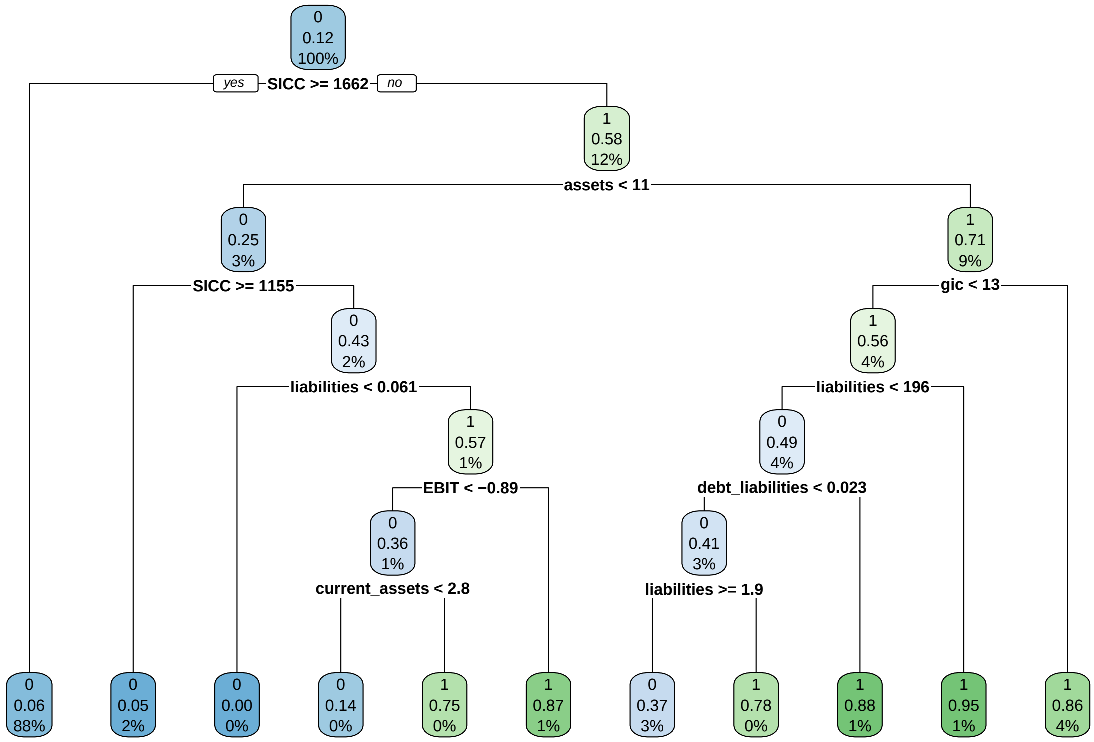
```r
plotcp(rpart) # visualize cross-validation results
```




```r
# summary(rpart) # detailed summary of splits
```

You can depict the classification tree by using the "plot()" function.

```r
#Plot tree
rpart.plot::rpart.plot(rpart)
```



To get the accuracy of the model I first get the predicted probabilities, then impute the values for the outcome variable.

```
fitted.results.rpart <- predict(rpart, newdata=test,type='prob')
fitted.rpart <- ifelse(fitted.results.rpart[,2] >= 0.5, 1, 0)
```

Below, I depict the predictive performance of the model.

```
# RMSE
caret::postResample(fitted.rpart, test$iso_code)
```

```
##      RMSE  Rsquared       MAE
## 0.3183684 0.2371798 0.1013584
```

```
# For good predictive model the MAE and RMSE values should be low


# Confusion Matrix
confusionMatrix(data = as.factor(fitted.rpart),
                reference = as.factor(test$iso_code))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2436  248
##          1   43  144
##
##                Accuracy : 0.8986
```

```
##                  95% CI : (0.887, 0.9094)
##     No Information Rate : 0.8635
##     P-Value [Acc > NIR] : 6.802e-09
##
##                   Kappa : 0.4488
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9827
##             Specificity : 0.3673
##          Pos Pred Value : 0.9076
##          Neg Pred Value : 0.7701
##              Prevalence : 0.8635
##          Detection Rate : 0.8485
##    Detection Prevalence : 0.9349
##       Balanced Accuracy : 0.6750
##
##        'Positive' Class : 0
##
```

```r
# Balanced Accuracy
balanced_accuracy_rpart<-balanced_accuracy(fitted.rpart, test$iso_code)
balanced_accuracy_rpart
```

```
## [1] 0.838827
```

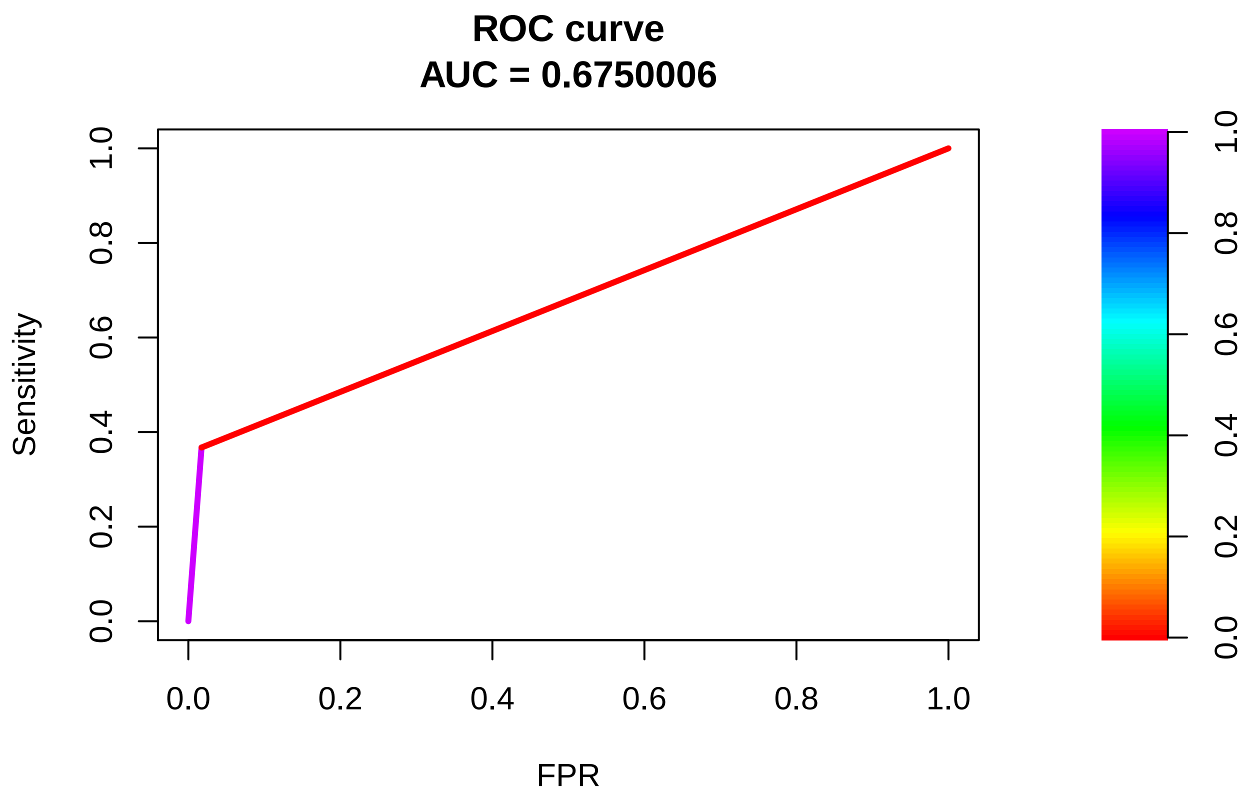
```r
# F1-Score
f1_rpart_1 <- f1_score(fitted.rpart,
                       test$iso_code,
                       positive.class="1")
f1_rpart_1
```

```
##         1
## 0.4974093
```

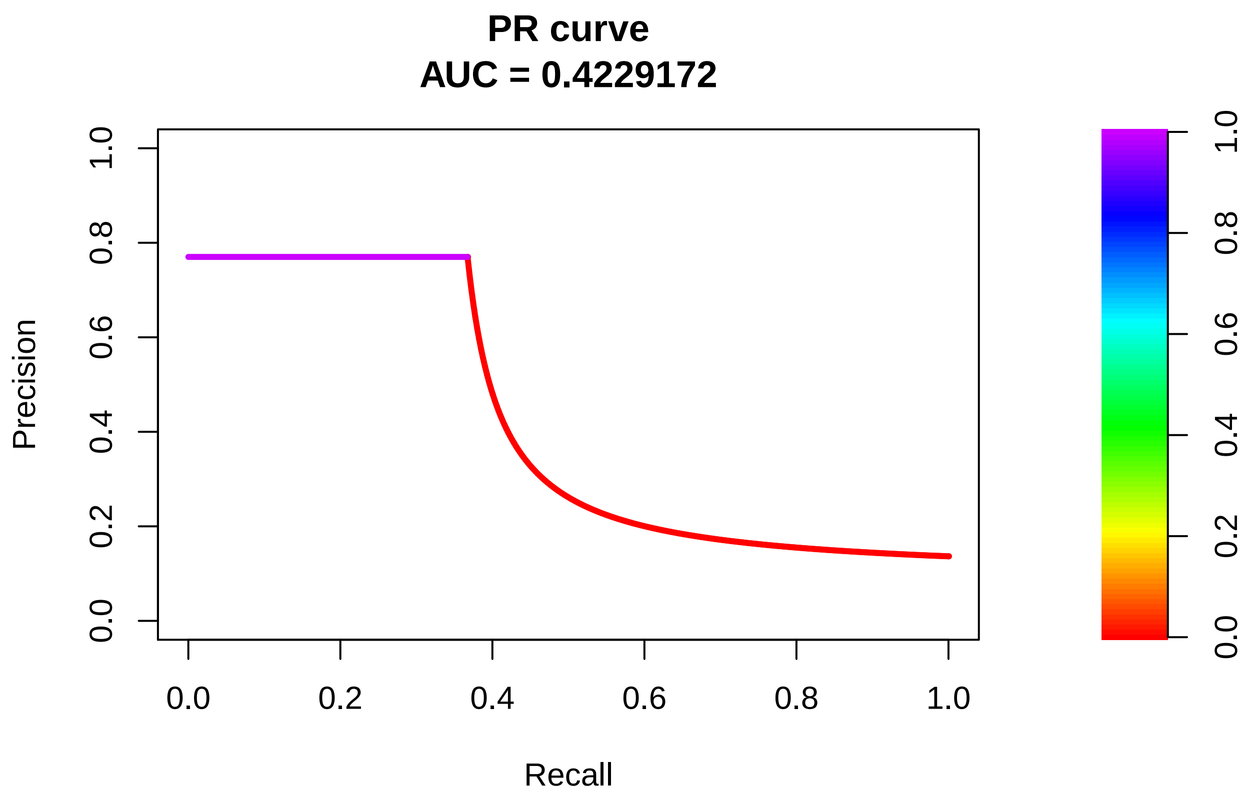
```r
f1_rpart_0 <- f1_score(fitted.rpart,
                       test$iso_code,
                       positive.class="0")
f1_rpart_0
```

```
##         0
## 0.9436374
```

```r
# ROC Curve and PR- Curve
fg.rpart <- fitted.rpart[test$iso_code==1]
bg.rpart <- fitted.rpart[test$iso_code==0]
roc_rpart <- roc.curve(scores.class0 = fg.rpart,
                       scores.class1 = bg.rpart,
                       curve = T)
plot(roc_rpart)
```




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



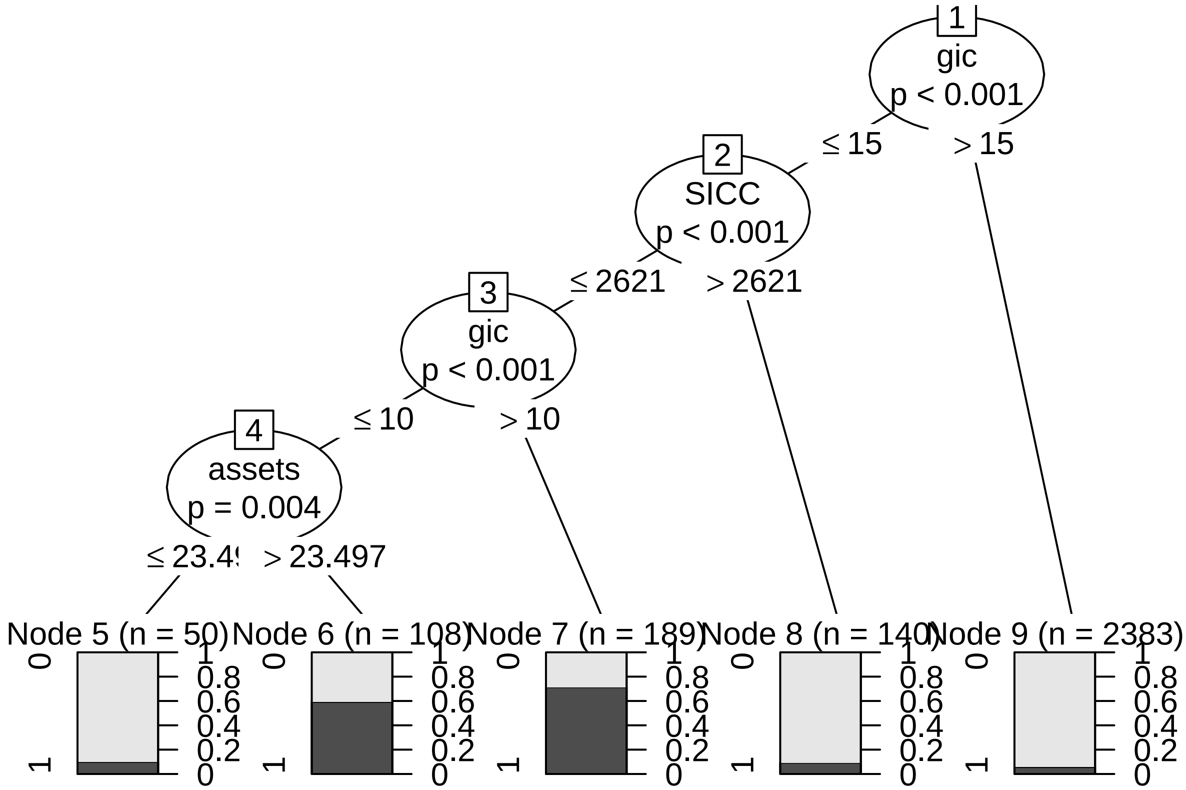
Conditional Inference Tree

One potential drawback of the classification and regression trees

```
c.tree <- ctree(formula, data=train,
               control = ctree_control(testtype = "MonteCarlo",
               mincriterion = 0.99, nresample = 1000))
```

You can plot the tree by running the following chunk of code.

```
plot(c.tree, gp = gpar(fontsize = 6))
```



To get the accuracy of the model I first get the predicted probabilities, then impute the values for the outcome variable.

```
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]
fitted.tree <- ifelse(fitted.prob.tree >= 0.5, 1, 0)
```

Below, I depict the predictive performance of the model.

```
# RMSE
caret::postResample(fitted.tree, test$iso_code)
```

```
##      RMSE  Rsquared       MAE
## 0.3194605 0.2586794 0.1020550
```

```
# For good predictive model the MAE and RMSE values should be low


# Confusion Matrix
confusionMatrix(data = as.factor(fitted.tree),
               reference = as.factor(test$iso_code))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2399  213
##          1   80  179
```

```
##
##                Accuracy : 0.8979
##                  95% CI : (0.8863, 0.9088)
##     No Information Rate : 0.8635
##     P-Value [Acc > NIR] : 1.347e-08
##
##                   Kappa : 0.4951
##
##  Mcnemar's Test P-Value : 1.243e-14
##
##             Sensitivity : 0.9677
##             Specificity : 0.4566
##          Pos Pred Value : 0.9185
##          Neg Pred Value : 0.6911
##              Prevalence : 0.8635
##          Detection Rate : 0.8356
##    Detection Prevalence : 0.9098
##       Balanced Accuracy : 0.7122
##
##        'Positive' Class : 0
##
```

```r
# Balanced Accuracy
balanced_accuracy_tree<-balanced_accuracy(fitted.tree, test$iso_code)
balanced_accuracy_tree
```

```
## [1] 0.8047865
```

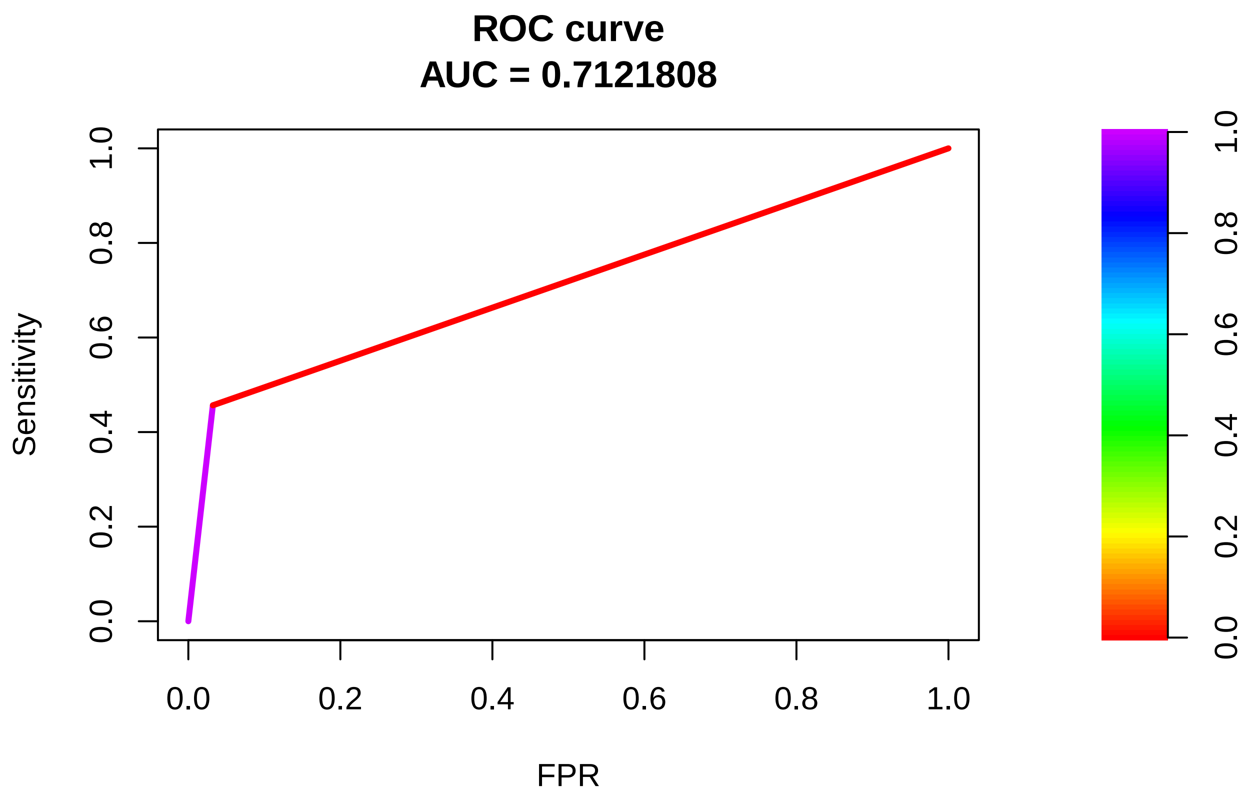
```r
# F1-Score
f1_tree_1 <- f1_score(fitted.tree,
                       test$iso_code,
                       positive.class="1")
f1_tree_1
```

```
##         1
## 0.5499232
```

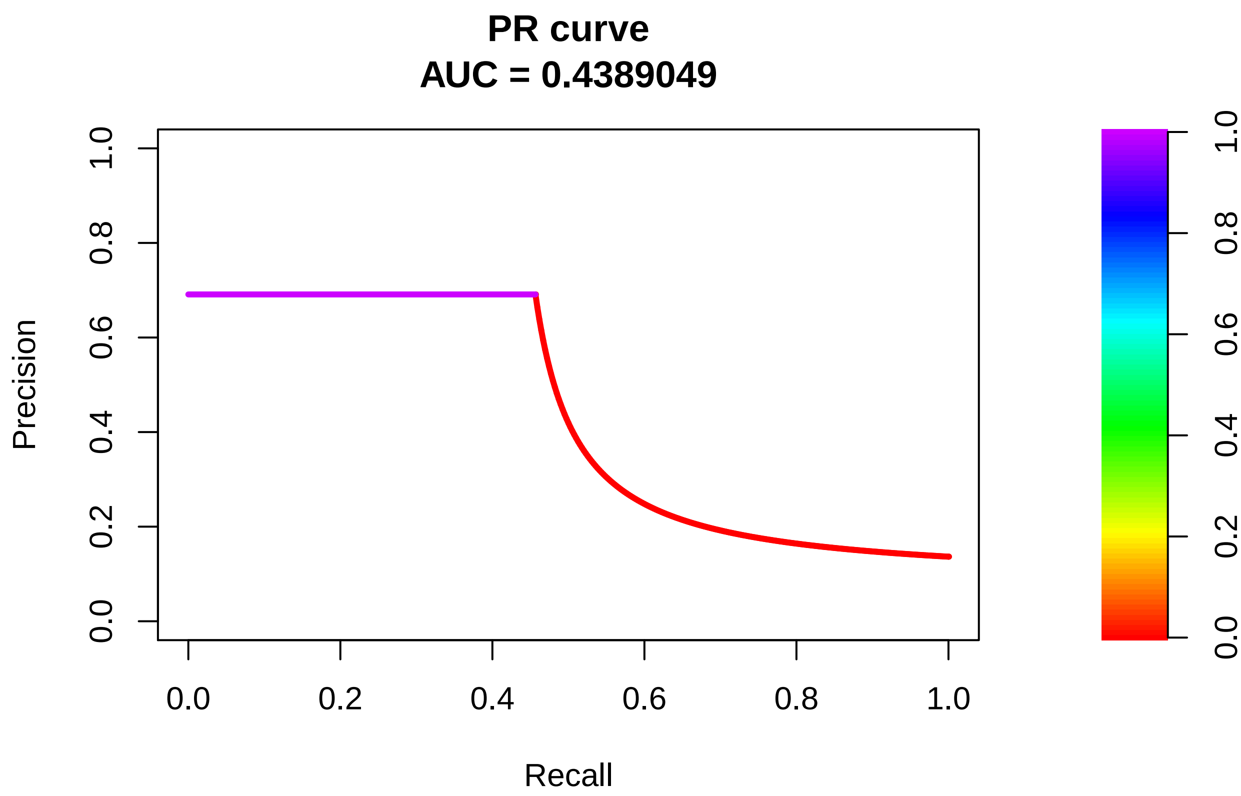
```r
f1_tree_0 <- f1_score(fitted.tree,
                       test$iso_code,
                       positive.class="0")
f1_tree_0
```

```
##         0
## 0.9424475
```

```r
# ROC Curve and PR- Curve
fg.tree <- fitted.tree[test$iso_code==1]
bg.tree <- fitted.tree[test$iso_code==0]
roc_tree <- roc.curve(scores.class0 = fg.tree,
                       scores.class1 = bg.tree,
                       curve = T)
plot(roc_tree)
```




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



Random Forest

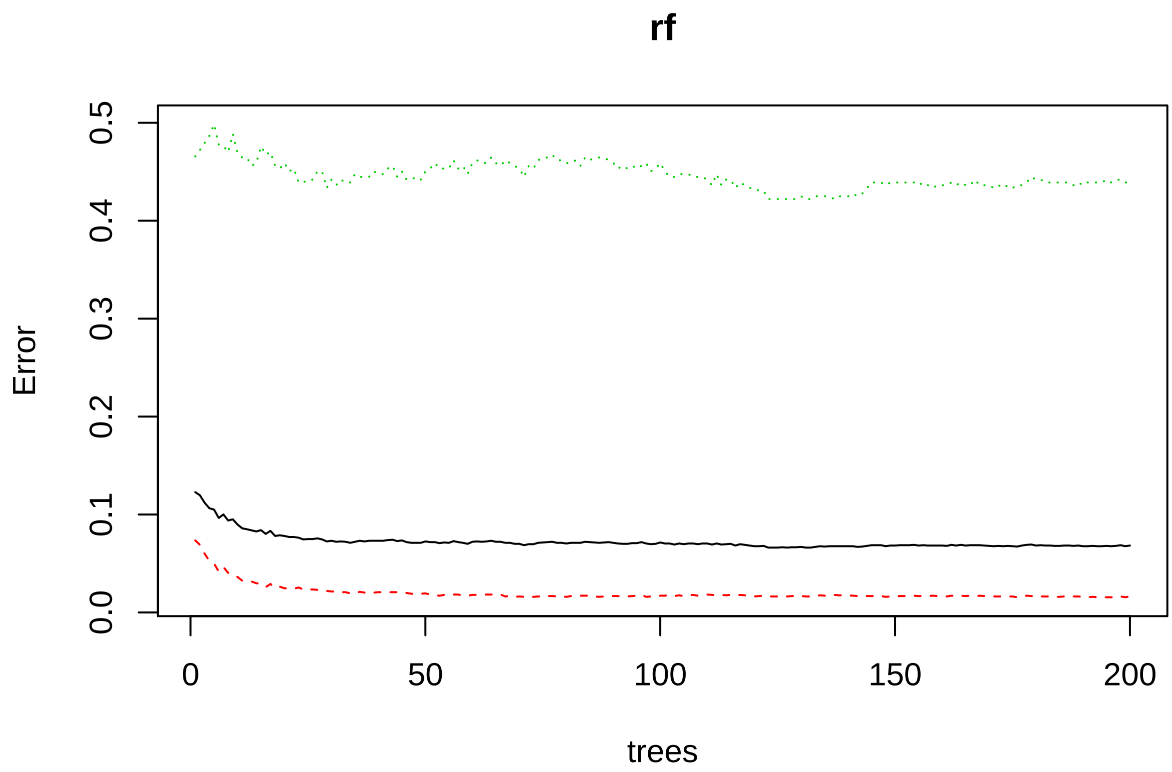
The last model that I build is a random forest from the "randomForest" package.

```
set.seed(133234)
rf <- randomForest(formula, data=train, importance=TRUE, ntree=200)
```

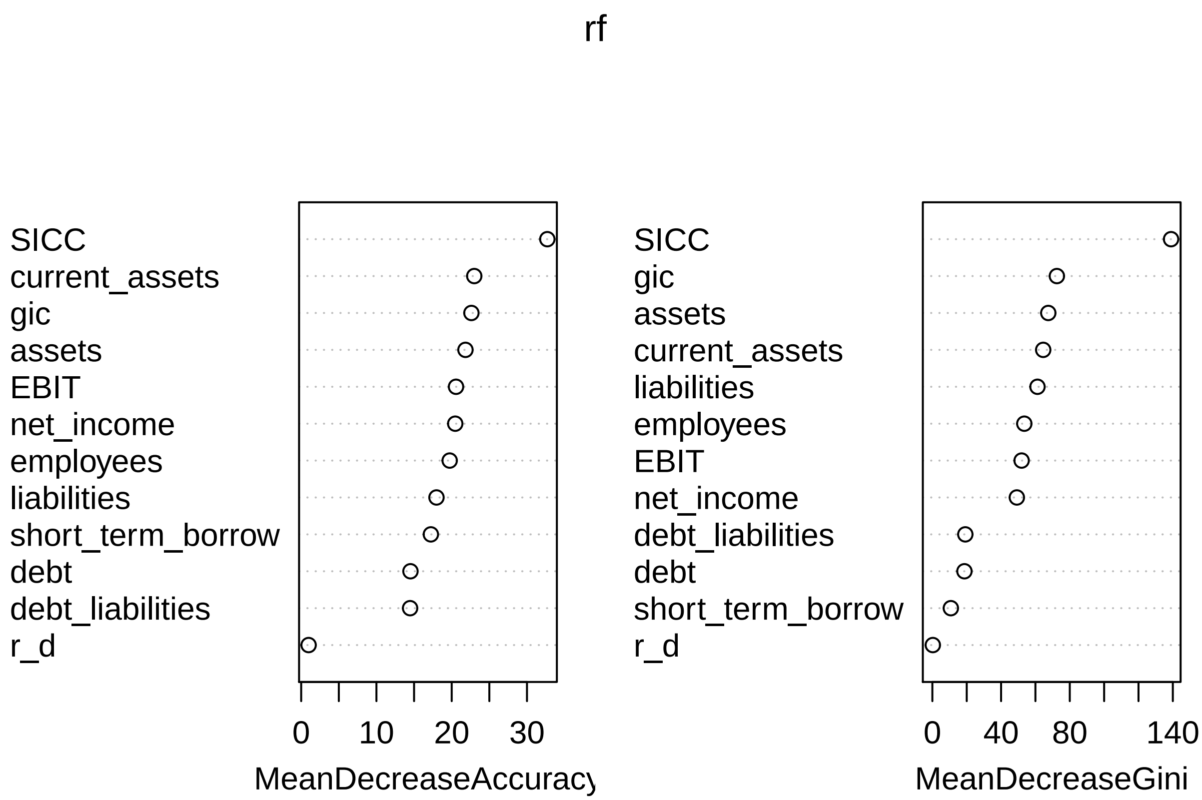
```
print(rf)
```

```
##
## Call:
##  randomForest(formula = formula, data = train, importance = TRUE,      ntree = 200)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 6.83%
## Confusion matrix:
##      0   1 class.error
## 0 2476  41  0.01628923
## 1  155 198  0.43909348
```

```
plot(rf)
```




```
varImpPlot(rf)
```



To get the accuracy of the model I first get the predicted probabilities, then impute the values for the outcome variable.

```
fitted.rf <- predict(rf, test)
fitted.rf <- as.numeric(matrix(fitted.rf))
```

Below, I depict the predictive performance of the model.

```
# RMSE
caret::postResample(fitted.rf, test$iso_code)
```

```
##       RMSE   Rsquared        MAE
## 0.28852427 0.36034498 0.08324626
```

```
# For good predictive model the MAE and RMSE values should be low


# Confusion Matrix
confusionMatrix(data = as.factor(fitted.rf),
                reference = as.factor(test$iso_code))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2435  195
##          1   44  197
##
##                Accuracy : 0.9168
```

```
##                  95% CI : (0.906, 0.9266)
##     No Information Rate : 0.8635
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5786
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9823
##             Specificity : 0.5026
##          Pos Pred Value : 0.9259
##          Neg Pred Value : 0.8174
##              Prevalence : 0.8635
##          Detection Rate : 0.8481
##    Detection Prevalence : 0.9161
##       Balanced Accuracy : 0.7424
##
##        'Positive' Class : 0
##
```

```r
# Balanced Accuracy
balanced_accuracy_rf<-balanced_accuracy(fitted.rf, test$iso_code)
balanced_accuracy_rf
```

```
## [1] 0.8716414
```

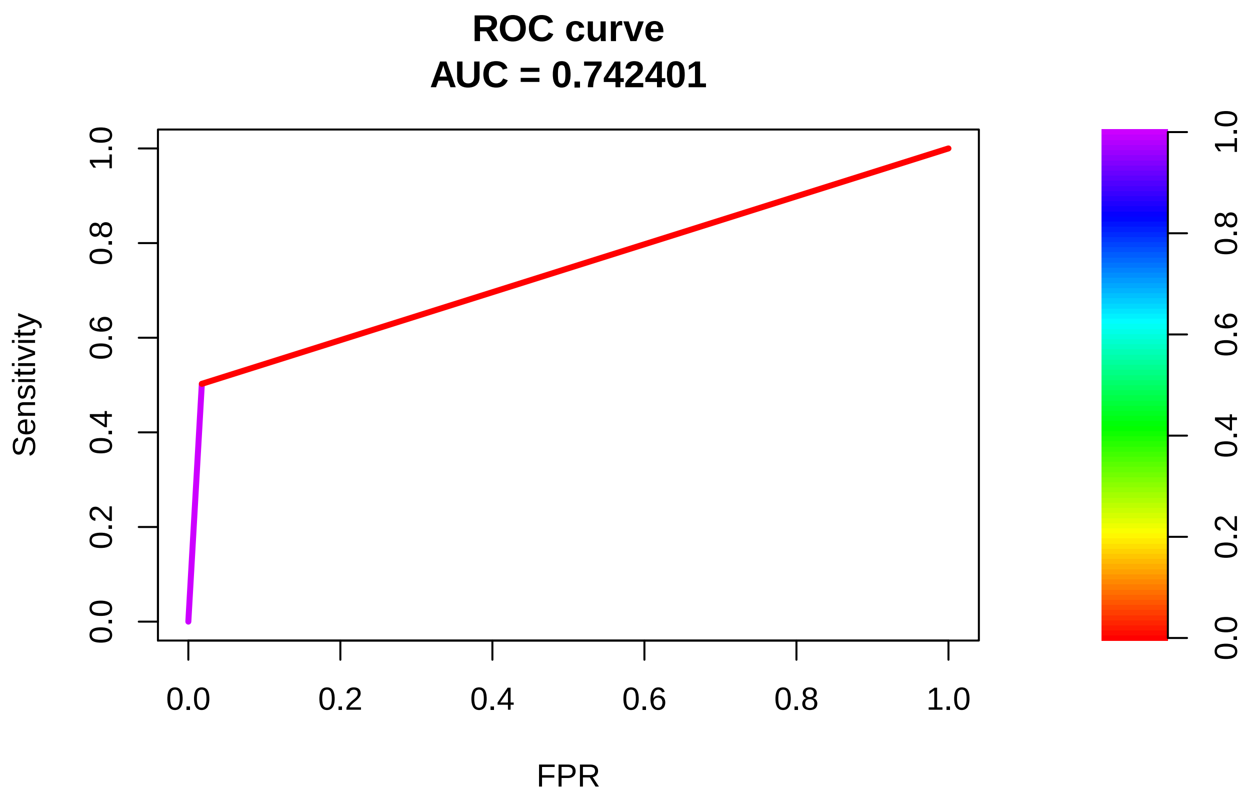
```r
# F1-Score
f1_rf_1 <- f1_score(fitted.rf,
                       test$iso_code,
                       positive.class="1")
f1_rf_1
```

```
##         1
## 0.6224329
```

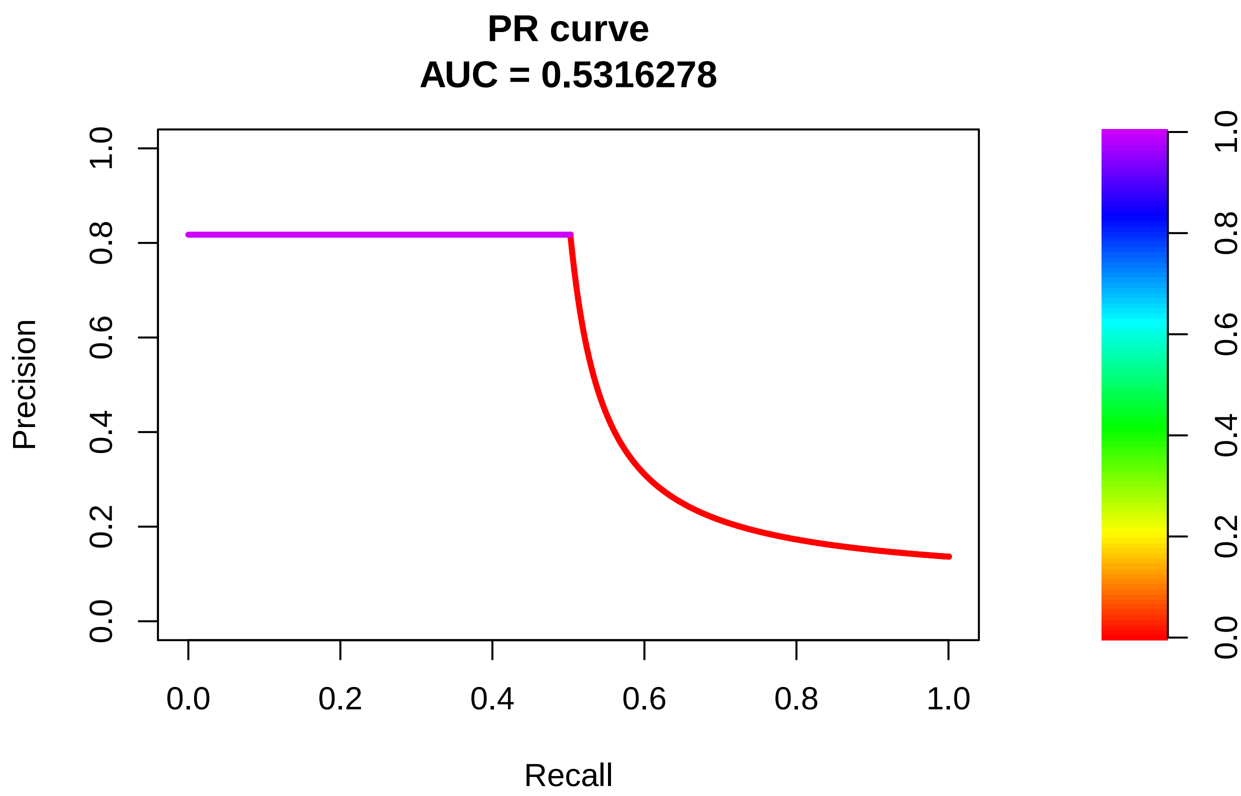
```r
f1_rf_0 <- f1_score(fitted.rf,
                       test$iso_code,
                       positive.class="0")
f1_rf_0
```

```
##         0
## 0.9532198
```

```r
# ROC Curve and PR- Curve
fg.rf <- fitted.rf[test$iso_code==1]
bg.rf <- fitted.rf[test$iso_code==0]
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)
```



Bayesian Forest (Bayesian Additive Regression Trees)

```
set.seed(133234)
bart_machine <- bartMachine(X = as.data.frame(train[predictors]),
                            y = as.factor(train$iso_code),
                            use_missing_data=FALSE)
```

```
## bartMachine initializing with 50 trees...
## bartMachine vars checked...
## bartMachine java init...
## bartMachine factors created...
## bartMachine before preprocess...
## bartMachine after preprocess... 13 total features...
## bartMachine sigsq estimated...
## bartMachine training data finalized...
## Now building bartMachine for classification ...
## evaluating in sample data...done
```

To get the accuracy of the model I first get the predicted probabilities, then impute the values for the outcome variable.

```
fitted.prob.bart <- 1- round(predict(bart_machine,
                                     as.data.frame(test[predictors]),
                                     type='prob'), 6)
fitted.bart <- ifelse(fitted.prob.bart> 0.5, 1, 0)
```

```r
# RMSE
caret::postResample(fitted.bart, test$iso_code)
```

```
##      RMSE  Rsquared       MAE
## 0.3128503 0.2634878 0.0978753
```

```r
# For good predictive model the MAE and RMSE values should be low


# Confusion Matrix
confusionMatrix(data = as.factor(fitted.bart),
                reference = as.factor(test$iso_code))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2430  232
##          1   49  160
##
##                Accuracy : 0.9021
##                  95% CI : (0.8907, 0.9128)
##     No Information Rate : 0.8635
##     P-Value [Acc > NIR] : 1.781e-10
##
##                   Kappa : 0.4834
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9802
##             Specificity : 0.4082
##          Pos Pred Value : 0.9128
##          Neg Pred Value : 0.7656
##              Prevalence : 0.8635
##          Detection Rate : 0.8464
##    Detection Prevalence : 0.9272
##       Balanced Accuracy : 0.6942
##
##        'Positive' Class : 0
##
```

```r
# Balanced Accuracy
balanced_accuracy_bart<-balanced_accuracy(fitted.bart, test$iso_code)
balanced_accuracy_bart
```

```
## [1] 0.8391989
```

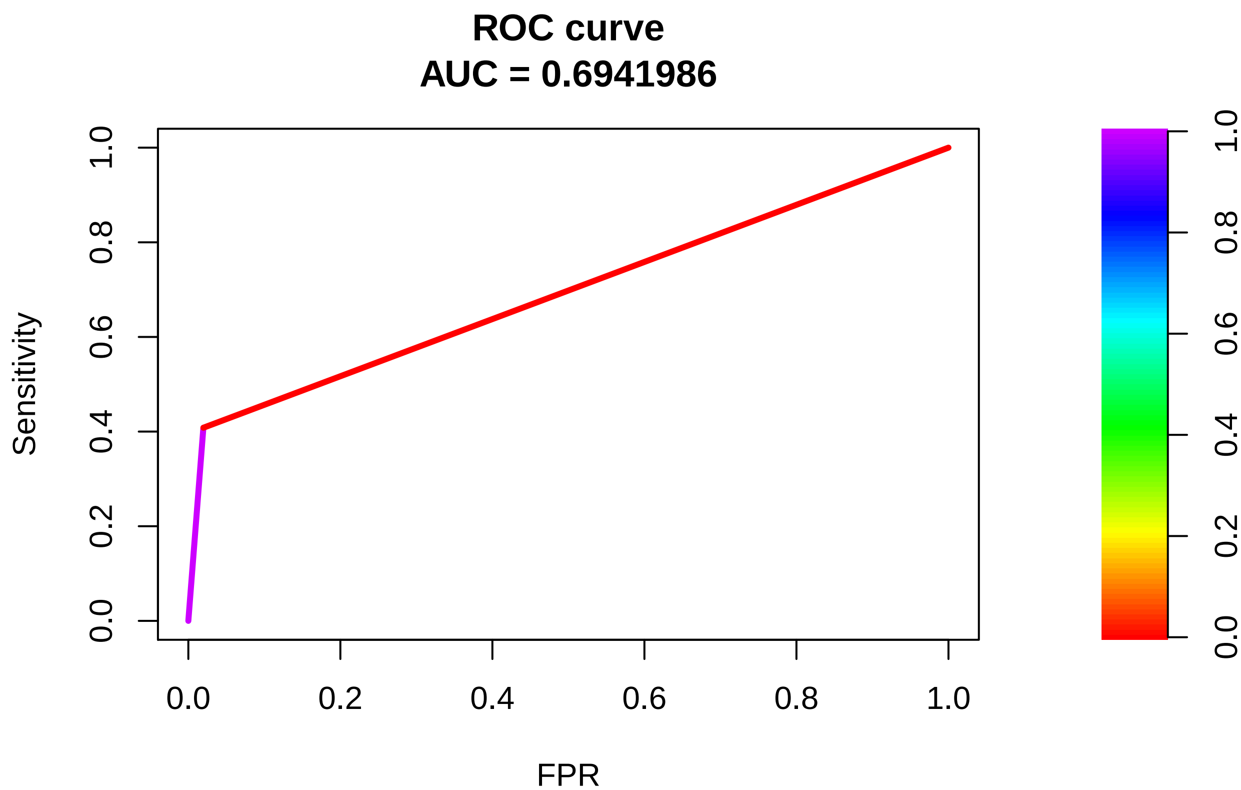
```r
# F1-Score
f1_bart_1 <- f1_score(fitted.bart,
                       test$iso_code,
                       positive.class="1")
f1_bart_1
```

```
##         1
## 0.5324459
```

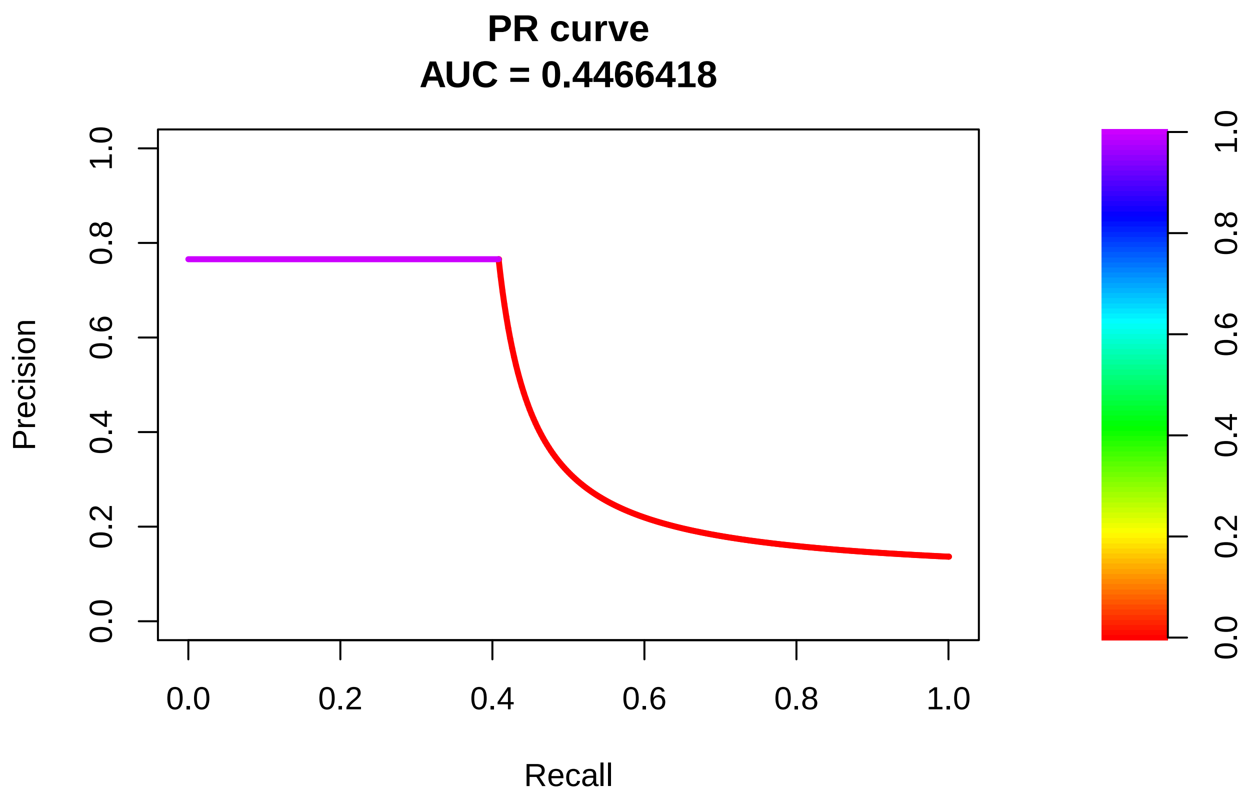
```r
f1_bart_0 <- f1_score(fitted.bart,
                      test$iso_code,
                      positive.class="0")
f1_bart_0
```

```
##         0
## 0.9453414
```

```r
# ROC Curve and PR- Curve
fg.bart <- fitted.bart[test$iso_code==1]
bg.bart <- fitted.bart[test$iso_code==0]
roc_bart <- roc.curve(scores.class0 = fg.bart,
                      scores.class1 = bg.bart,
                      curve = T)
plot(roc_bart)
```



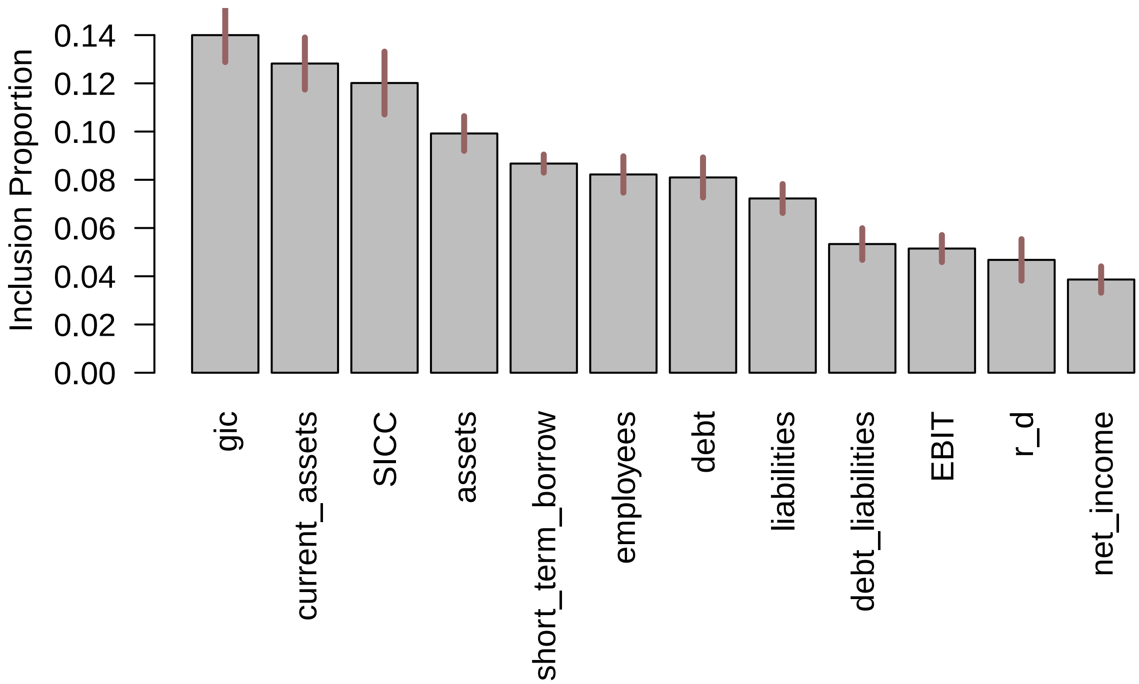

```r
pr_bart <- pr.curve(scores.class0 = fg.bart,
                    scores.class1 = bg.bart,
                    curve = T)
plot(pr_bart)
```



The package bartMachine provides tools for investigation of variables' importance and variables' selection

```
investigate_var_importance(bart_machine, num_replicates_for_avg = 20)
```

```
## ....................
```

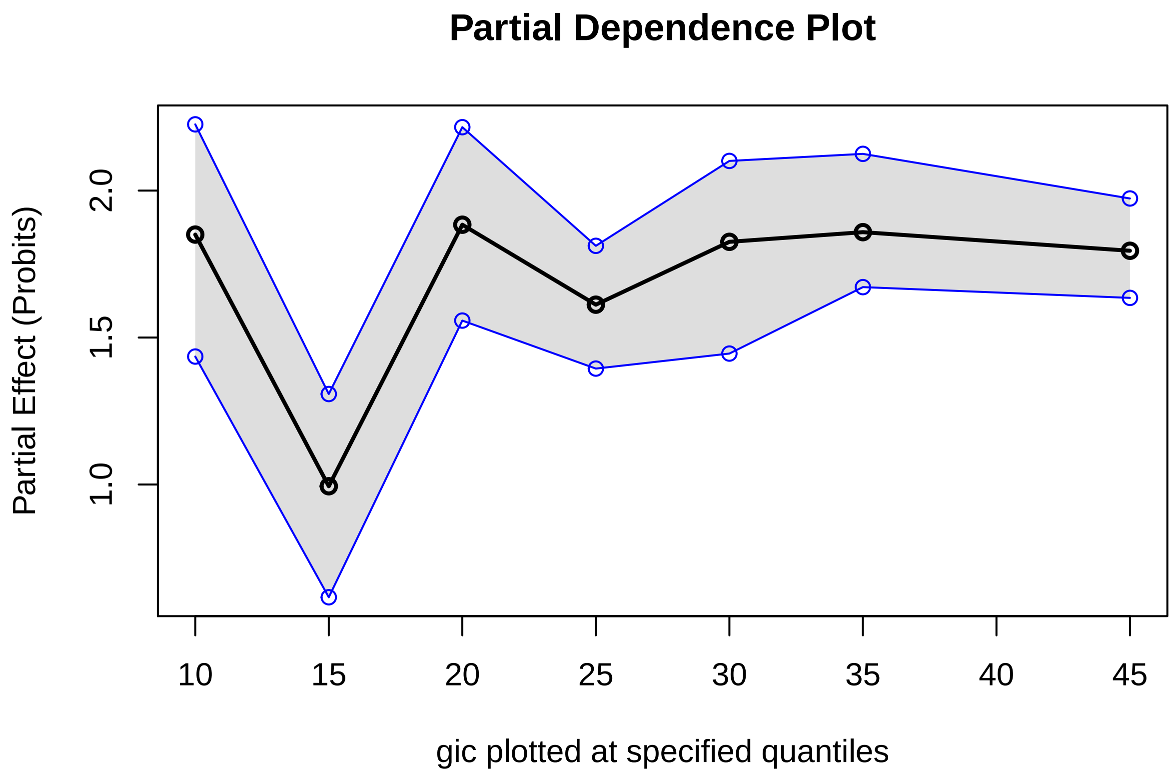



```r
vs <- var_selection_by_permute(bart_machine,
                               num_permute_samples = 10)
```

As well as for partial dependency plots.

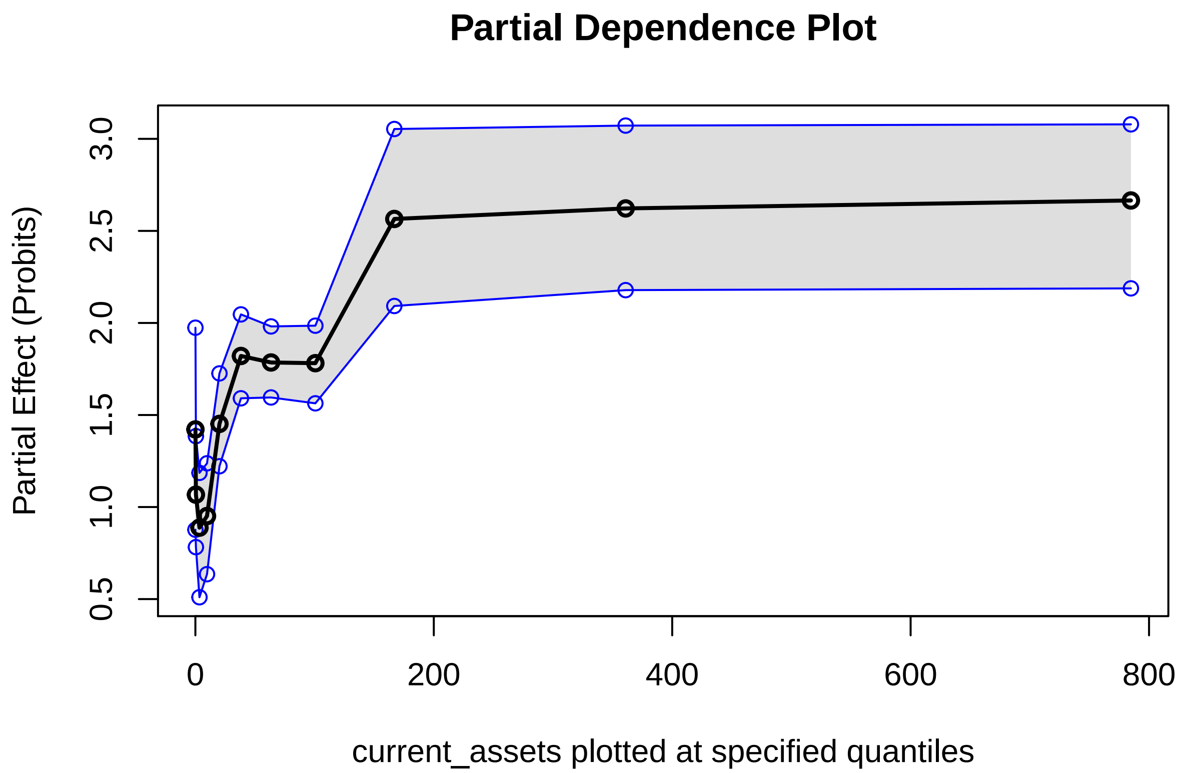
```r
pd_plot(bart_machine, j = "gic")
```

```
## .......
```




```
pd_plot(bart_machine, j = "current_assets")
```

```
## ...........
```



We can incorporate information on the best predictors in the model by using a different set of priors.

```
predictors
```

```
##  [1] "assets"            "short_term_borrow" "current_assets"
##  [4] "debt"              "debt_liabilities"  "employees"
##  [7] "EBIT"              "liabilities"       "net_income"
## [10] "r_d"               "gic"               "SICC"
```

```
prior <- c(rep(1, times = 10), rep(2, times = 2))
```

```
set.seed(133234)
bart_prior <- bartMachine(X = as.data.frame(train[predictors]),
                          cov_prior_vec = prior,
                          y = as.factor(train$iso_code),
                          use_missing_data=FALSE)
```

```
## bartMachine initializing with 50 trees...
## bartMachine vars checked...
## bartMachine java init...
## bartMachine factors created...
## bartMachine before preprocess...
## bartMachine after preprocess... 13 total features...
## bartMachine sigsq estimated...
## bartMachine training data finalized...
## Now building bartMachine for classification ...Covariate importance prior ON.
## evaluating in sample data...done
```

```r
fitted.prob.bart <- 1- round(predict(bart_prior,
                              as.data.frame(test[predictors]),
                              type='prob'), 6)
fitted.prior <- ifelse(fitted.prob.bart> 0.5, 1, 0)
```

Evaluate the performance of this new model.

```r
# RMSE
caret::postResample(fitted.prior, test$iso_code)
```

```
##      RMSE  Rsquared       MAE
## 0.3072331 0.2839116 0.0943922
```

```r
# For good predictive model the MAE and RMSE values should be low


# Confusion Matrix
confusionMatrix(data = as.factor(fitted.prior),
                reference = as.factor(test$iso_code))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2435  227
##          1   44  165
##
##                Accuracy : 0.9056
##                  95% CI : (0.8943, 0.9161)
##     No Information Rate : 0.8635
##     P-Value [Acc > NIR] : 3.168e-12
##
##                   Kappa : 0.5018
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9823
##             Specificity : 0.4209
##          Pos Pred Value : 0.9147
##          Neg Pred Value : 0.7895
##              Prevalence : 0.8635
##          Detection Rate : 0.8481
##    Detection Prevalence : 0.9272
##       Balanced Accuracy : 0.7016
##
##        'Positive' Class : 0
##
```

```r
# Balanced Accuracy
balanced_accuracy_prior<-balanced_accuracy(fitted.prior, test$iso_code)
balanced_accuracy_prior
```

```
## [1] 0.8520997
```

```r
# F1-Score
f1_prior_1 <- f1_score(fitted.prior,
                       test$iso_code,
```

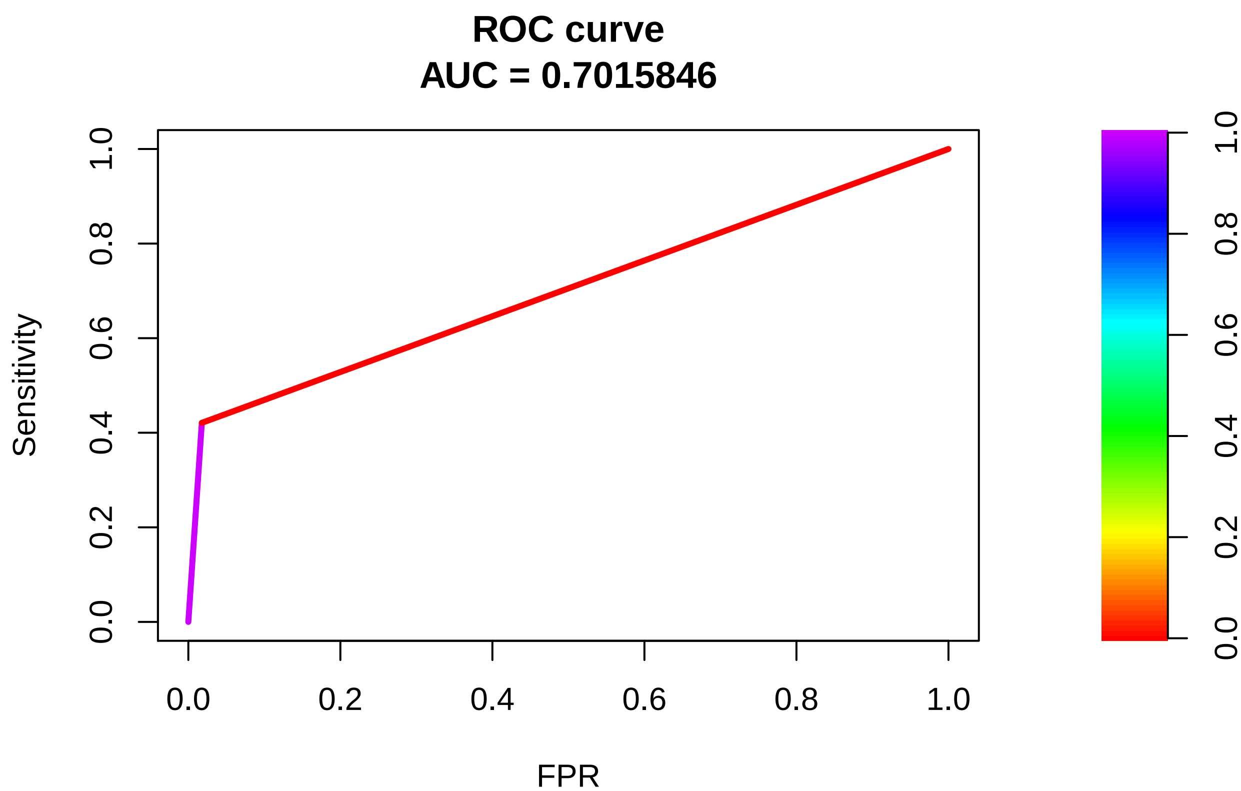
```
                              positive.class="1")
f1_prior_1
```

```
##         1
## 0.5490849
```

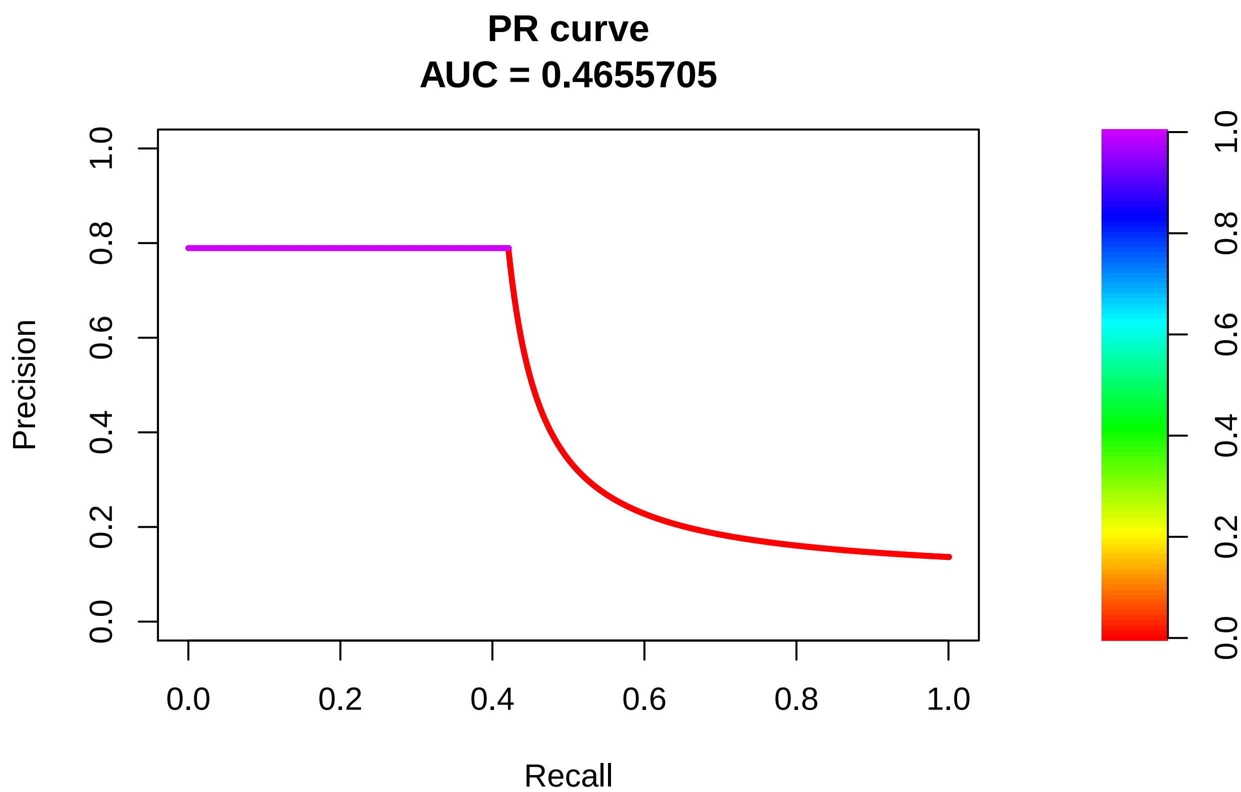
```
f1_prior_0 <- f1_score(fitted.prior,
                       test$iso_code,
                       positive.class="0")
f1_prior_0
```

```
##         0
## 0.9472865
```

```
# ROC Curve and PR- Curve
fg.prior <- fitted.prior[test$iso_code==1]
bg.prior <- fitted.prior[test$iso_code==0]
roc_prior <- roc.curve(scores.class0 = fg.prior,
                       scores.class1 = bg.prior,
                       curve = T)
plot(roc_prior)
```




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


PR curve
AUC = 0.4655705
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0