# Robustness Checks

*Falco J. Bargagli Stoffi*

*27/10/2018*

## Introduction

The idea of this section of the appendix is to build a "sensitivity analysis" for the predictions obtained from the BART algorithm (which is the ML algorithm that seems to perform better in terms of predictive ability). Since these predictions are the foundations of our identification strategy, it is important to check whether or not they are stable. The stability of predictions is checked with respect to the unit level predicted probabilities of failure:

$$f_{BART}(x) = \hat{p}_i(Y_i = 1|X_i = x). \tag{1}$$

The "robustness" of the predictions is tested with respect to two dimensions: (i) the inclusion of a new "important" predictor uncorrelated with the other predictors but strongly correlated with the outcome; (ii) changes in the training sample used to build the BART model.

These robustness checks are done in two ways:

1. generating a new predictor (a "confounder" $R_i$) and checking if (and how) the inclusion of it in the model changes the predicted probabilities of failure: namely, what is the distance between $\hat{p}_i(Y_i = 1|X_i = x)$ and $\hat{p}_i(Y_i = 1|X_i = x, R_i = r)$?
2. sub-sampling with replacement from the same population and checking the stability of the unit level predictions $\hat{p}_i(Y_i = 1|X_i = x)$ (using for the analysis the observations that are common to all the different sub-samples).

Let's now see in detail how do we implement these robustness checks in the statistical software R.

### Load Data

First we upload the set of variables that we will need for the analysis.

```
myvariables <- c("id", "iso", "tfp_acf",  "Number_of_patents",
                 "Number_of_trademarks","consdummy", "control",
                 "failure", "nace_2","fin_rev", "int_paid",
                 "ebitda", "cash_flow", "depr", "revenue",
                 "total_assets", "long_term_debt", "employees",
                 "added_value", "materials", "wage_bill", "loans" ,
                 "int_fixed_assets", "fixed_assets", "current_liabilities",
                 "liquidity_ratio", "solvency_ratio", "current_assets",
                 "fin_expenses", "net_income", "fin_cons", "fin_cons100",
                 "inv",  "ICR_t", "time", "real_SA", "shareholders_funds",
                 "NEG_VA", "ICR_failure", "profitability",
                 "misallocated_fixed", "interest_diff")
#capital_intensity, labour_product, retained_earnings, firm_value excluded: highly missing
dati <- dati[myvariables]
```

### Work with a subsample

We work with a subsample of the entire population. This is due to two reasons:

1. to reduce the computational time when we will run the BART algorithm on different bootstrap samples;
2. to increase the instability of the model.

It is self-evident that the larger is the sample used for the training, the bigger is the stability of the algorithm itself. In this case we use a subsample because we want to see if the overall stability is reproduced even in subsamples of the training population. The results that we report here are than a lower bound of the results that we could get using the entire population of observations.

The sample is chosen to have a size of 1% of the population. This is done following the advices of Varian (2014).

```
set.seed(123)
dati<-as.data.frame(dati[,1:length(dati)])
dati <- dati[sample(1:nrow(dati), size = nrow(dati)*0.01, replace = FALSE ),]
```

## Generating a new variable strongly correlated with the outcome

The following function returns a data frame of two variables which correlate with a population correlation of $\rho$. If desired, one of both variables can be fixed to an existing variable by specifying $X_i$. The function is made to build an $R_i$ that has a high correlation to $Y_i$. In this case we chose $R_i$ to be a normally distributed variable (but we could also use any other distribution).

We generate a variable $R_i$ that is correlated with $Y_i$ but results to be uncorrelated with all the other predictors $X_i$. We set the correlation between $R_i$ and $Y_i$ to be the same as the correlation of the best predictor of in the BART algorithm and the most correlated variable. In this case both the variables (i.e., the best predictor and the most correlated variable) are the same, namely *negative added value*.

```
set.seed(123)
getBiCop <- function(n, rho, mar.fun=rnorm, x = NULL) {
     if (!is.null(x)) {X1 <- x} else {X1 <- mar.fun(n)}
     if (!is.null(x) & length(x) != n)
     warning("Variable x does not have the same length as n!")

     C <- matrix(rho, nrow = 2, ncol = 2)
     diag(C) <- 1

     C <- chol(C)

     X2 <- mar.fun(n)
     X <- cbind(X1,X2)

     # induce correlation (does not change X1)
     df <- X %*% C

     ## if desired: check results
     #all.equal(X1,X[,1])
     #cor(X)

     return(df)
}
```

The following are the correlations between (i) $Y_i$ and $R_i$; (ii) $Y_i$ and *negative added value*; (iii) $R_i$ and *negative added value*.

Morover, in the plot is shown the density of $R_i$.

```r
cor(omit$failure, omit$robust)
```

```
## [1] 0.1556155
```

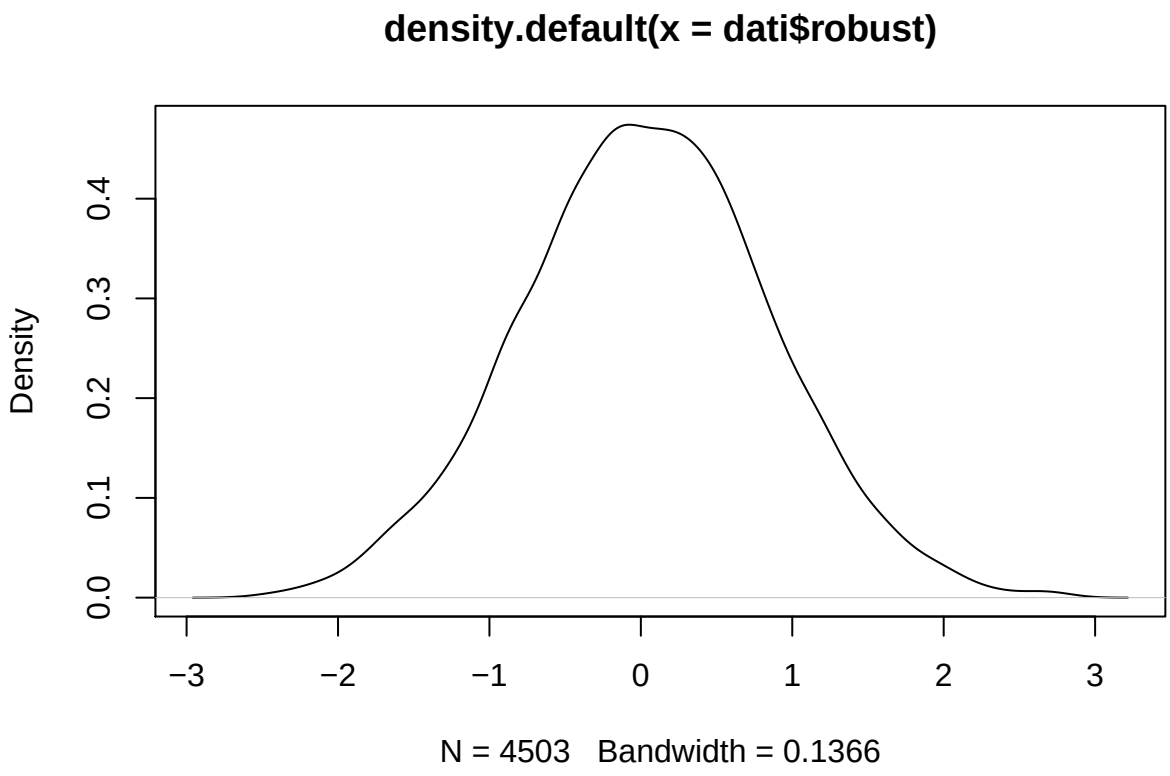
```r
cor(omit$failure, omit$NEG_VA)
```

```
## [1] 0.1669955
```

```r
cor(omit$robust, omit$NEG_VA)
```

```
## [1] 0.02747978
```

```r
plot(density(dati$robust))
```




### BART with all the predictors

Let's now build a first BART model with all the predictors.

```r
predictors <- c("iso", "tfp_acf", "Number_of_patents",
                "Number_of_trademarks", "consdummy",
                "control","nace_2", "fin_rev", "int_paid",
                "ebitda", "cash_flow", "depr", "revenue",
                "total_assets", "long_term_debt", "employees",
                "added_value", "materials", "wage_bill", "loans" ,
                "int_fixed_assets","fixed_assets", "current_liabilities",
                "liquidity_ratio",  "solvency_ratio", "current_assets",
                "fin_expenses", "net_income",  "fin_cons100" , "inv" ,
```

```
                "real_SA", "shareholders_funds" , "NEG_VA" ,"ICR_failure" ,
                "profitability" ,  "misallocated_fixed" , "interest_diff")
# , "+ capital_intensity", "labour_product"
dati$X <-as.data.frame(dati[predictors])
```

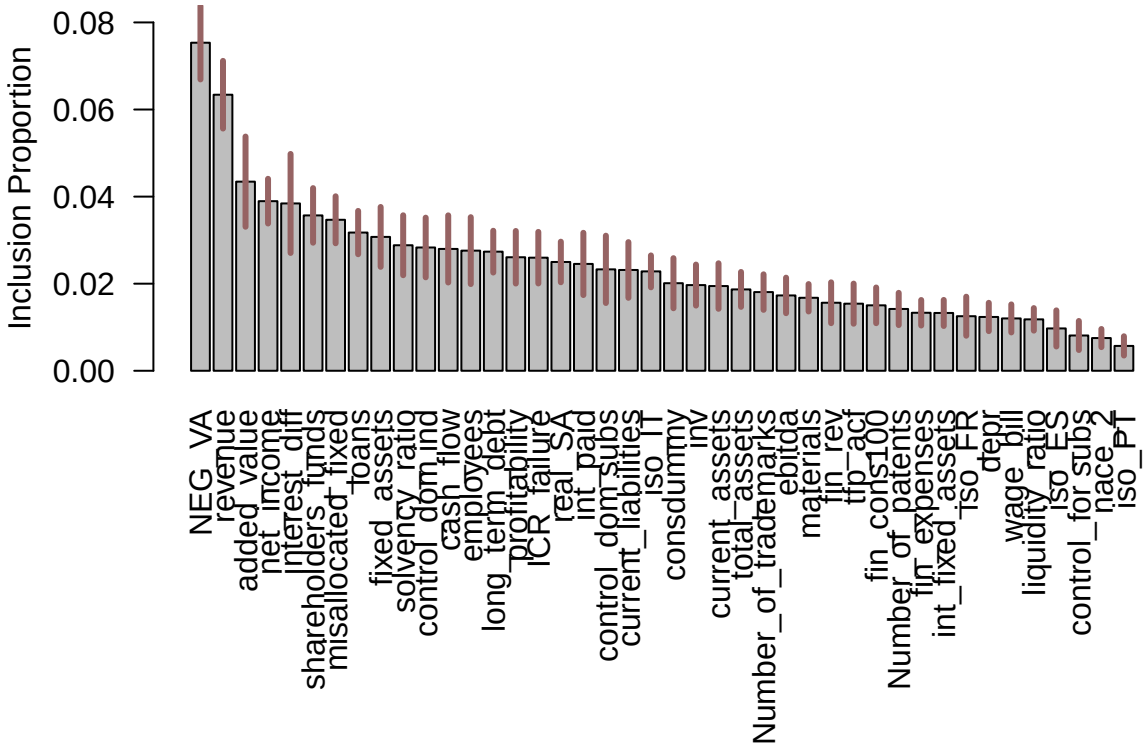
```
set.seed(123)
system.time({
bart_predictors<-bartMachine(dati$X,as.factor(dati$failure), use_missing_data=TRUE)
})

dati$fitted.results.bart.predictors <- 1- round(predict(bart_predictors, dati$X,
                                                        type='prob'), 6)
dati$fitted.results.bart.predictors <- as.numeric(dati$fitted.results.bart.predictors)
predictions <- as.data.frame(cbind(dati$id, dati$fitted.results.bart.predictors))
```

*Negative Added Value* results to be the best predictor in the model.

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

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




Generate the matrix of predictors with the new predictor that we generated before $R_i$ (*robust*).

```
robust <- c("iso", "tfp_acf", "Number_of_patents", "Number_of_trademarks",
  "consdummy", "control","nace_2", "fin_rev", "int_paid", "ebitda", "cash_flow",
  "depr", "revenue", "total_assets", "long_term_debt", "employees", "added_value",
  "materials", "wage_bill", "loans" , "int_fixed_assets","fixed_assets",
  "current_liabilities",  "liquidity_ratio",  "solvency_ratio", "current_assets",
```

```
  "fin_expenses", "net_income",  "fin_cons100" , "inv" , "real_SA", "shareholders_funds" ,
  "NEG_VA" ,"ICR_failure" , "profitability" ,  "misallocated_fixed" , "interest_diff" ,
  "robust")
```

### Create the BART function

We now create a BART function that will run on different boostraped samples.

```
bart <- function(sample) {
  bart_machine<- bartMachine(X, Y,
  use_missing_data=TRUE)
  sample$fitted.results.bart <- 1- round(predict(bart_machine, X,  type='prob'), 6)
  res <- cbind(sample$id, sample$fitted.results.bart)
  return(res)
}
```

## Robustness check: what happens when we introduce a new "strong" predictor? (1)

Let's first see how many times the predictions of the first BART (without $R_i$) are contained in the confindence intervals of the predictions build for $\hat{p}(Y_i = 1|X_i = x, R_i = r)$.

```
### Create matrix to save bootstrapped results (N = B)
B=6
results<-matrix(data=NA, nrow = nrow(dati)*0.95, ncol = B)
id<-matrix(data=NA, nrow = nrow(dati)*0.95, ncol = B)
merge <- matrix(data=NA, nrow = nrow(dati)*0.95, ncol = B*2)

#prob1 <- rep(1/nrow(dati), nrow(dati))
#prob1

### Start loop
set.seed(123)
for (i in (1:B)) {
  # Repeatedly (B) draw subsamples
  sample <- dati[sample(1:nrow(dati), nrow(dati)*0.95, replace = FALSE),]

  X <-as.data.frame(sample[robust])
  Y <- as.factor(sample$failure)

  # split into training and test
  #train_ind <- sample(seq_len(nrow(dati)), size = nrow(dati)*0.9)
  #train <- dati[train_ind,]
  #test <-  dati[-train_ind,]

  #BART
  merge[, ((i*2)-1):(i*2)] <- bart(sample)
  id[,i]  <- merge[,((i*2)-1)]
  results[,i] <- merge[,(i*2)]
  #merge[, ((i*2)-1):(i*2)] <- cbind(id[,i], results[,i])
}
```

```r
# The following code is for the predictions of the "Robust model"
length(unique(dati$id))
length(unique(id[,1])) #which are the unique IDs?

# Overlapping observations between the different samples
# Namely, obs that are drawn in all the samples
bootstrap <- as.data.frame(id[,1][id[,1] %in% id[,2]])

for (i in (3:(B))) {
  bootstrap <- as.matrix(bootstrap[,1][bootstrap[,1] %in% id[,i]])
}

bootstrap <- as.data.frame(bootstrap)
names(bootstrap) <- c("id") # IDs of overlapping obs

data_merge <- as.data.frame(merge)
names(data_merge) <- c("id", "values",
                       "id", "values",
                       "id", "values",
                       "id", "values",
                       "id", "values",
                       "id", "values")

# Obtaining the predicted probabilities just for the overlapping IDs
total <- merge(bootstrap,data_merge[,1:2], by= c("id"))

for (i in (2:(B))) {
  total <- merge(total,data_merge[,((i*2)-1):(i*2)], by= c("id"))
}
total <- as.matrix(total)

# Computing means and SDs of overlapping probabilities
tot <- matrix(data=NA, nrow = nrow(total), ncol = (ncol(total)-1))
tot[,1:(ncol(total)-1)]<-as.matrix(sapply(total[,2:ncol(total)], as.numeric))
mean_sd<-cbind(rowMeans(tot[,1:ncol(tot)]), rowSds(tot[,1:ncol(tot)]))

mean_sd <- as.data.frame(cbind(total[,1], mean_sd[,1:2]))
names(mean_sd) <- c("id", "mean", "sd")
names(predictions) <- c("id", "values")

# Getting a matrix with means and sd of predicted probabilites
# from the "robust" model and the point estimates from the "general" model
final <- as.data.frame(merge(mean_sd, predictions, by= c("id")))
fin <- matrix(data=NA, nrow = nrow(final), ncol = 3)
fin[,1:3]<- as.matrix(sapply(as.matrix(final[,2:4]), as.numeric))
fin <- as.data.frame(fin)
names(fin) <- c("mean", "sd", "value")
```

The proportion of inclusion of $\hat{p}(Y_i|X_i = x)$ in the confidence intervals of $\hat{p}(Y_i|X_i = x, R_i = r)$ is the following.

```r
#Change the t-value accordingly to B and alpha
length(which(fin$value <= (fin$mean + 4.03*(fin$sd/sqrt(B))) &
fin$value >= (fin$mean - 4.03*(fin$sd/sqrt(B))) ))/nrow(fin)
```

```
## [1] 0.5408654
```

# Robustness check: what happens when we introduce a new "strong" predictor? (2)

## Check of CI overlap

This time we perform a bootstrap with replacement also for the general model (the one without $R_i$) and we will see how large is the overlap between the confidence intervals of the predictions of $\hat{p}_i(Y_i = 1|X_i = x)$ and $\hat{p}_i(Y_i = 1|X_i = x, R_i = r)$.

## Robust Model

```
### Create matrix to save bootstrapped results (N = B)
B=4
results<-matrix(data=NA, nrow = nrow(dati), ncol = B)
id<-matrix(data=NA, nrow = nrow(dati), ncol = B)
merge <- matrix(data=NA, nrow = nrow(dati), ncol = B*2)

#prob1 <- rep(1/nrow(dati), nrow(dati))
#prob1

### Start loop for ROBUST

for (i in (1:B)) {
  set.seed(123 + i)
  # Repeatedly (B) draw subsamples
  sample <- dati[sample(1:nrow(dati), nrow(dati), replace = TRUE),]

  X <-as.data.frame(sample[robust])
  Y <- as.factor(sample$failure)

  # split into training and test
  #train_ind <- sample(seq_len(nrow(dati)), size = nrow(dati)*0.9)
  #train <- dati[train_ind,]
  #test <-  dati[-train_ind,]

  #BART
  merge[, ((i*2)-1):(i*2)] <- bart(sample)
  id[,i]  <- merge[,((i*2)-1)]
  results[,i] <- merge[,(i*2)]
  #merge[, ((i*2)-1):(i*2)] <- cbind(id[,i], results[,i])
}
```

```
# The comments are the same of the previous chunk

# The following code is for the model "robust"
length(unique(dati$id))
length(unique(id[,1]))
bootstrap <- as.data.frame(id[,1][id[,1] %in% id[,2]])

for (i in (3:(B))) {
  bootstrap <- as.matrix(bootstrap[,1][bootstrap[,1] %in% id[,i]])
}
```

```
bootstrap <- as.data.frame(bootstrap)
names(bootstrap) <- c("id")

data_merge <- as.data.frame(merge)
names(data_merge) <- c("id", "values",
                       "id", "values",
                       "id", "values",
                       "id", "values") #change accordingly

total <- merge(bootstrap,data_merge[,1:2], by= c("id"))

for (i in (2:(B))) {
  total <- merge(total,data_merge[,((i*2)-1):(i*2)], by= c("id"))
}

total <- as.matrix(total)

library(matrixStats)
tot <- matrix(data=NA, nrow = nrow(total), ncol = (ncol(total)-1))
tot[,1:(ncol(total)-1)]<-as.matrix(sapply(total[,2:ncol(total)], as.numeric))
mean_sd<-cbind(rowMeans(tot[,1:ncol(tot)]), rowSds(tot[,1:ncol(tot)]))

mean_sd <- as.data.frame(cbind(total[,1], mean_sd[,1:2]))
names(mean_sd) <- c("id", "mean", "sd")
```

```
# Matrix for the Mean and SD of the predicted probabilities
length(unique(mean_sd$id))
```

```
## [1] 738
```

```
mean_sd <- subset(mean_sd, !duplicated(mean_sd$id))
head(mean_sd)
```

```
##             id                  mean                    sd
## 1  ESA01016120 3.45000000000207e-05 5.82208439192385e-05
## 3  ESA08240095  0.00620349999999998  0.00844694285920456
## 19 ESA08258451              0.09862   0.0508179995867606
## 23 ESA20521498 0.000157249999999998 0.000198813773835351
## 27 ESA28034254           0.18913375    0.061974100915221
## 28 ESA95209037  0.00232874999999999  0.00182647591005195
```

## General model

```
results_model<-matrix(data=NA, nrow = nrow(dati), ncol = B)
id_model<-matrix(data=NA, nrow = nrow(dati), ncol = B)
merge_model <- matrix(data=NA, nrow = nrow(dati), ncol = B*2)

# Start Loop

for (i in (1:B)) {
  set.seed(123 + i)
  # Repeatedly (B) draw subsamples
  sample <- dati[sample(1:nrow(dati), nrow(dati), replace = TRUE),]
```

```
  X <-as.data.frame(sample[predictors])
  Y <- as.factor(sample$failure)

  # split into training and test
  #train_ind <- sample(seq_len(nrow(dati)), size = nrow(dati)*0.9)
  #train <- dati[train_ind,]
  #test <-  dati[-train_ind,]

  #BART
  merge_model[, ((i*2)-1):(i*2)] <- bart(sample)
  id_model[,i]  <- merge_model[,((i*2)-1)]
  results_model[,i] <- merge_model[,(i*2)]
  #merge[, ((i*2)-1):(i*2)] <- cbind(id[,i], results[,i])
}
```

```
# The comments are the same of the previous chunk

# The following code is for the model "general"
length(unique(id_model[,1]))
bootstrap_model <- as.data.frame(id_model[,1][id_model[,1] %in% id_model[,2]])

for (i in (3:(B))) {
  bootstrap_model <- as.matrix(bootstrap_model[,1][bootstrap_model[,1] %in% id_model[,i]])
}

bootstrap_model <- as.data.frame(bootstrap_model)
names(bootstrap_model) <- c("id")

data_merge_model <- as.data.frame(merge_model)
names(data_merge_model) <- c("id", "values",
                        "id", "values",
                        "id", "values",
                        "id", "values") #change accordingly

total_model <- merge(bootstrap_model,data_merge_model[,1:2], by= c("id"))

for (i in (2:(B))) {
  total_model <- merge(total_model,data_merge_model[,((i*2)-1):(i*2)], by= c("id"))
}

total_model <- as.matrix(total_model)

tot_model <- matrix(data=NA, nrow = nrow(total_model), ncol = (ncol(total_model)-1))
tot_model[,1:(ncol(total_model)-1)]<-as.matrix(sapply(total_model[,2:ncol(total_model)], as.numeric))
mean_sd_model<-cbind(rowMeans(tot_model[,1:ncol(tot_model)]), rowSds(tot_model[,1:ncol(tot_model)]))

mean_sd_model <- as.data.frame(cbind(total_model[,1], mean_sd_model[,1:2]))
names(mean_sd_model) <- c("id", "mean_model", "sd_model")
```

```
# Matrix for the Mean and SD of the predicted probabilities
length(unique(mean_sd_model$id))
```

```
## [1] 738
```

```r
mean_sd_model <- subset(mean_sd_model, !duplicated(mean_sd_model$id))
head(mean_sd_model)
```

```
##             id           mean_model             sd_model
## 1  ESA01016120 4.02499999999917e-05 4.99090840094771e-05
## 3  ESA08240095  0.00155075000000002  0.00193876461954515
## 19 ESA08258451           0.06926025    0.036177451940631
## 23 ESA20521498 4.70000000000192e-05 7.03088424974617e-05
## 27 ESA28034254             0.124761   0.0257599197203718
## 28 ESA95209037           0.01977425   0.0197731076528198
```

### Merging

The overlap between the confidence itervals of the predicted probabilities between the *general* model (without $R_i$) and the *robust* model (with $R_i$) is particularly high. The overlap is reported below.

```r
final <- as.data.frame(merge(mean_sd, mean_sd_model, by= c("id")))

fin <- matrix(data=NA, nrow = nrow(final), ncol = ncol(final)-1)
fin[,1:ncol(fin)]<- as.matrix(sapply(as.matrix(final[,2:ncol(final)]), as.numeric))
fin <- as.data.frame(fin)
names(fin) <- c("mean", "sd", "mean_model", "sd_model")

#Change the t-value accordingly to B and alpha

# CI overlap
length(which((fin$mean_model - 4.03*(fin$sd_model/sqrt(B))) <= (fin$mean + 4.03*(fin$sd/sqrt(B))) &  (fi
```

```
## [1] 0.9837398
```

## Standardized Difference in Means

Moreover, the standardized difference in the means between $\hat{p}(Y_i = 1|X_i = x)$ and $\hat{p}(Y_i = 1|X_i = x, R_i = r)$ is not significant in all the cases.

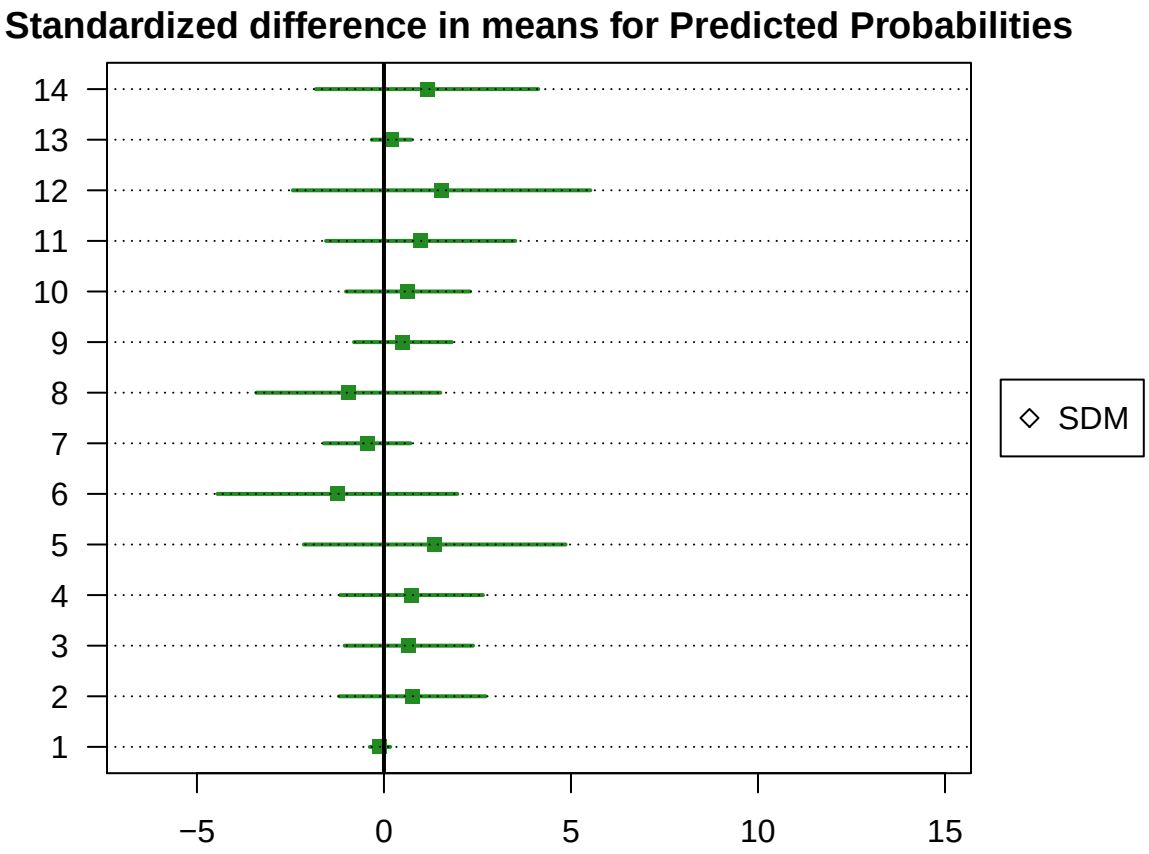
```r
# Standardized difference in means
diff.means <- fin$mean - fin$mean_model
standard.diff.means <-  (fin$mean - fin$mean_model)/sqrt((fin$sd^2 + fin$sd_model^2)/2)

# 99% CI (t-student distribution)
x0 <- standard.diff.means - 2.58 * # change accordingly to sample size
  (fin$mean - fin$mean_model)/sqrt((fin$sd^2 + fin$sd_model^2)/2) #check dimension
x1 <-standard.diff.means  + 2.58 * # change accordingly to sample size
  (fin$mean - fin$mean_model)/sqrt((fin$sd^2 + fin$sd_model^2)/2) #check dimension

(length(which(x0 > 0 & x1 < 0)) + length(which(x0 < 0 & x1 > 0)))/ length(x0)
```

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

This can be seen from the plot of the Standardized difference in means for the probabilities predicted by the two models.




Moreover, we run multiple t-tests for the predicted values in the general model and in the robust model. Below we can see the proportion of observations that are exceeding the standard significativity levels of 0.01 and 0.05.

```
test <- as.data.frame(cbind(total[,1], tot))
names(test) <- c("id", "v1", "v2", "v3", "v4")
test <- subset(test, !duplicated(test$id))

test_model <- as.data.frame(cbind(total_model[,1], tot_model))
names(test_model) <- c("id", "m1", "m2", "m3", "m4")
test_model <- subset(test_model, !duplicated(test_model$id))

testing <- as.data.frame(merge(test, test_model, by= c("id")))
dim(testing)
```

```
## [1] 738   9
```

```
test <- matrix(data = NA, ncol = (ncol(testing) -5), nrow = nrow(testing))
test[, 1:4] <- as.matrix(sapply(as.matrix(testing[,2:5]), as.numeric))
test_model <- matrix(data = NA, ncol = (ncol(testing) -5), nrow = nrow(testing))
test_model[,1:4] <- as.matrix(sapply(as.matrix(testing[,6:9]), as.numeric))

pvalue <- matrix(data = NA, ncol = 1, nrow = nrow(test))
N = nrow(test)
for (i in (1:N)) {
    pvalue[i,] <- t.test(test[i, ], test_model[i,])$p.value
}
```

```r
length(which(pvalue[,1]<0.01))/N
```

```
## [1] 0.02168022
```

```r
length(which(pvalue[,1]<0.05))/N
```

```
## [1] 0.09620596
```

## Robustness check: what happens when we change the sample?

We propose to use 10 different bootstrap samples (change B to 10 and run the "function" chunks). What we are interested in is the variance of the different sampling predictions.

```r
B = 10
results_model<-matrix(data=NA, nrow = nrow(dati), ncol = B)
id_model<-matrix(data=NA, nrow = nrow(dati), ncol = B)
merge_model <- matrix(data=NA, nrow = nrow(dati), ncol = B*2)

for (i in (1:B)) {
  set.seed(123 + i)
  # Repeatedly (B) draw subsamples
  sample <- dati[sample(1:nrow(dati), nrow(dati), replace = TRUE),]

  X <-as.data.frame(sample[predictors])
  Y <- as.factor(sample$failure)

  # split into training and test
  #train_ind <- sample(seq_len(nrow(dati)), size = nrow(dati)*0.9)
  #train <- dati[train_ind,]
  #test <-  dati[-train_ind,]

  #BART
  merge_model[, ((i*2)-1):(i*2)] <- bart(sample)
  id_model[,i]  <- merge_model[,((i*2)-1)]
  results_model[,i] <- merge_model[,(i*2)]
  #merge[, ((i*2)-1):(i*2)] <- cbind(id[,i], results[,i])
}
```

```r
length(unique(id_model[,1]))
```

```
## [1] 2861
```

```r
bootstrap_model <- as.data.frame(id_model[,1][id_model[,1] %in% id_model[,2]])

for (i in (3:(B))) {
  bootstrap_model <- as.matrix(bootstrap_model[,1][bootstrap_model[,1] %in% id_model[,i]])
}

bootstrap_model <- as.data.frame(bootstrap_model)
names(bootstrap_model) <- c("id")

data_merge_model <- as.data.frame(merge_model)
names(data_merge_model) <- c("id", "values",
                        "id", "values",
```

```
                        "id", "values",
                        "id", "values",
                        "id", "values",
                        "id", "values",
                        "id", "values",
                        "id", "values",
                        "id", "values",
                        "id", "values") #change accordingly

total_model <- merge(bootstrap_model,data_merge_model[,1:2], by= c("id"))

for (i in (2:(B))) {
  total_model <- merge(total_model,data_merge_model[,((i*2)-1):(i*2)], by= c("id"))
}

total_model <- as.matrix(total_model)

tot_model <- matrix(data=NA, nrow = nrow(total_model), ncol = (ncol(total_model)-1))
tot_model[,1:(ncol(total_model)-1)]<-as.matrix(sapply(total_model[,2:ncol(total_model)], as.numeric))
mean_sd_model<-cbind(rowMeans(tot_model[,1:ncol(tot_model)]), rowSds(tot_model[,1:ncol(tot_model)]))

mean_sd_model <- as.data.frame(cbind(total_model[,1], mean_sd_model[,1:2], tot_model[,1:10]))
names(mean_sd_model) <- c("id", "mean_model", "sd_model")
```

```
length(unique(mean_sd_model$id))
```

```
## [1] 55
```

```
mean_sd_model <- subset(mean_sd_model, !duplicated(mean_sd_model$id))
head(mean_sd_model)
```

```
##               id           mean_model             sd_model
## 1    ESB18895953            0.0163832   0.0159229836735038
## 325  ESB58667130  0.00478550000000001  0.00336707222804753
## 1093 ESB62072228            0.0178882   0.0158386749151142
## 1237 FR321212904            0.0098803  0.00825784782360257
## 1285 FR350914685 8.54999999999717e-05 0.000127953333507022
## 1381 FR379249014 7.89999999999957e-05 0.000168312539969864
##                        NA                NA.1                NA.2
## 1                 0.03074             0.024629   0.0427070000000001
## 325  0.000951999999999953  0.00660300000000003  0.00525500000000001
## 1093             0.044649             0.038564             0.023599
## 1237  0.00217699999999998             0.004826             0.011505
## 1285 0.000400999999999985 5.99999999995049e-06 2.59999999999705e-05
## 1381  1.6000000000016e-05 1.99999999994649e-06 9.00000000003676e-06
##                      NA.3                NA.4                 NA.5
## 1                0.011892 0.00740499999999999  0.00274799999999997
## 325   0.00124500000000005 0.00724000000000002             0.009544
## 1093  0.00570300000000001  0.0224490000000001  0.00561100000000003
## 1237             0.024123 0.00417100000000004  0.00160000000000005
## 1285 6.99999999997925e-06 7.9999999999969e-05 5.99999999995049e-06
## 1381  1.2000000000012e-05 1.2000000000012e-05 0.000122999999999984
##                      NA.6                NA.7                 NA.8
## 1                0.003193  0.00217999999999996  0.00132399999999999
## 325   0.00150099999999997 0.000734000000000012  0.00834900000000005
```

```
## 1093  0.00138099999999997  0.00409700000000002  0.00427999999999995
## 1237              0.024101              0.009494              0.011182
## 1285 0.000202000000000035 9.99999999995449e-06 9.99999999995449e-06
## 1381 0.000546999999999964   8.000000000008e-06 1.49999999999872e-05
##                      NA.9
## 1                0.037014
## 325   0.00643199999999999
## 1093             0.028549
## 1237  0.00562399999999996
## 1285 0.000106999999999968
## 1381 4.59999999999905e-05
```

## Outliers detection (1): Check observations that are larger than 2SDs from the mean

The percentage of outliers detected with this method is shown below.

```
values <- matrix(NA, nrow = nrow(mean_sd_model), ncol = ncol(mean_sd_model) - 3)
values[, 1:10] <- as.numeric(as.matrix(mean_sd_model[,4:ncol(mean_sd_model)]))

rep.col<-function(x,n){
   matrix(rep(x,each=n), ncol=n, byrow=TRUE)
}

lower_bound <- rep.col(as.matrix(as.numeric(as.matrix(mean_sd_model$mean_model))
- 2 * as.numeric(as.matrix(mean_sd_model$sd_model))), 10)


upper_bound <- rep.col(as.matrix(as.numeric(as.matrix(mean_sd_model$mean_model))
+ 2 * as.numeric(as.matrix(mean_sd_model$sd_model))), 10)

length(which( values <  lower_bound |    values >=  upper_bound))/ (nrow(mean_sd_model)*(ncol(mean_sd_m
```

```
## [1] 0.05272727
```

```
# The denominator is the product of the dimensions of the "values" matrix:
# dim(mean_sd_model[, 4: ncol(mean_sd_model)])
```

## Outliers detection (2): Use the R function “Boxplot” to find outliers

The percentage of outlier detected in this manner is shown below.

```
value <- t(values)

outlier_values <- matrix(NA, ncol = 3, nrow = ncol(value))
for (i in (1:ncol(value))) {
      if (length(boxplot.stats(value[,i])$out) == 0)     {outlier_values[i,1] <- NA}
      else  {outlier_values[i,1] <- boxplot.stats(value[,i])$out[1]}

      if (length(boxplot.stats(value[,i])$out) == 0)     {outlier_values[i,2] <- NA}
      else  {outlier_values[i,2] <- boxplot.stats(value[,i])$out[2]}

      if (length(boxplot.stats(value[,i])$out) == 0)     {outlier_values[i,3] <- NA}
```

```
      else  {outlier_values[i,3] <- boxplot.stats(value[,i])$out[3]}

      #outlier_values[, i] <- boxplot.stats(value[, i])$out  # outlier values
}

length(which(!is.na(outlier_values)))/ (nrow(mean_sd_model)*(ncol(mean_sd_model) - 3))
```

```
## [1] 0.06181818
```

We can visualize the outliers using the boxplots.