Robustness checks

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

25/2/2020

Introduction

This `R Markdown` file reproduces the robustness check analyses 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(haven)
library(BART)
library(ggplot2)
source('sensitivity.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"),]
```

Sensitivity of Predictions Analysis

Here, we perform the "sensitivity analysis" for the predictions obtained from the BART algorithm. Since these predictions are the foundations of our paper, it is important to check whether or not they are stable. The following sensitivity of predictions analysis comes from a recent paper by Bargagli-Stoffi et al. (2020).

The stability of predictions is checked with respect to the unit level predicted financial literacy scores that we get from the original model:

$$f_(x) = \hat{Y}(X_i = x). \tag{1}$$

The "robustness" of the predictions is tested with respect to the inclusion of new predictors uncorrelated with varying correlations with the outcome.

In particular, this check is performed by generating a new predictor, a "confounder" R_i, orthogonal to the set of predictors $\mathbf{X}$ and with a correlation with the outcome that varies between 0.1 and 0.4, and checking how the inclusion of this additional predictor affects on the original model. The new model is the following:

$$f_(x, r) = \hat{Y}(X_i = x, R_i = r). \tag{2}$$

This check is done with respect to the following dimensions:

1. percentual decrease in the bias of $f_(x, r)$ wrt $f_(x)$;
2. percentual decrease in the RMSE of $f_(x, r)$ wrt $f_(x)$;
3. percentual increase in the R^2 of $f_(x, r)$ wrt $f_(x)$;
4. statistical difference between $\hat{Y}(x, r)$ and $\hat{Y}(x)$ at a significance level of α.

We argue that if these differences are not wide, there is room to think that the original model is capturing much of the signal in the data. In this case, the original model is stable and the addition of a new important predictor (i.e., the best predictors have roughly correlation 0.5 with the output) is not inducing substantial changes in the accuracy measures (i.e., bias, RMSE, R^2) and, more importantly, is not affecting in a sensible way the predicted values.

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

Data Inizialization

Select the lagged variables and the predictors.

```
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]

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")
```

Here we will use the same observations used in the main analysis.

```
omitted <- as.data.frame(na.omit(data_lagged))
set.seed(2020)
index <- sample(seq_len(nrow(omitted)), size = nrow(omitted)*0.9)
train_bart <- omitted[index,]
test_bart <- omitted[-index,]
train_bart$X <- as.data.frame(train_bart[predictors])
test_bart$X <- as.data.frame(test_bart[predictors])
y_train <- train_bart$failure
```

```
set.seed(2019)
sensitivity <- sensitivity_bart(x_train = train_bart$X,
                     y_train = y_train,
                     x_test = test_bart$X,
                     nburn = 500,
                     nsamp = 1000,
                     alpha = 0.1)
```

```
sensitivity
```

```
## $`Proportion of statistically different PPVs`
## [1] 0.000000e+00 7.182446e-05 4.189760e-04 4.429175e-04
##
## $`RMSE original model`
## [1] 0.1734016
##
## $`RMSE synthetic model`
## [1] 0.1710408 0.1633554 0.1480988 0.1269585
##
## $`Rsquared original model`
## [1] 0.213108
##
## $`Rsquared synthetic model`
## [1] 0.2344523 0.3016732 0.4259994 0.5781519
```

Testing Differences in RMSE and R squared

In order to test if the predictive performance of the synthetic model is better than the one of the original model we construct the condidence intervals for both the RMSE and R^2 of the synthetic model and we check if they overlap the values for the original model. In the following we depict the 99% confidence intervals

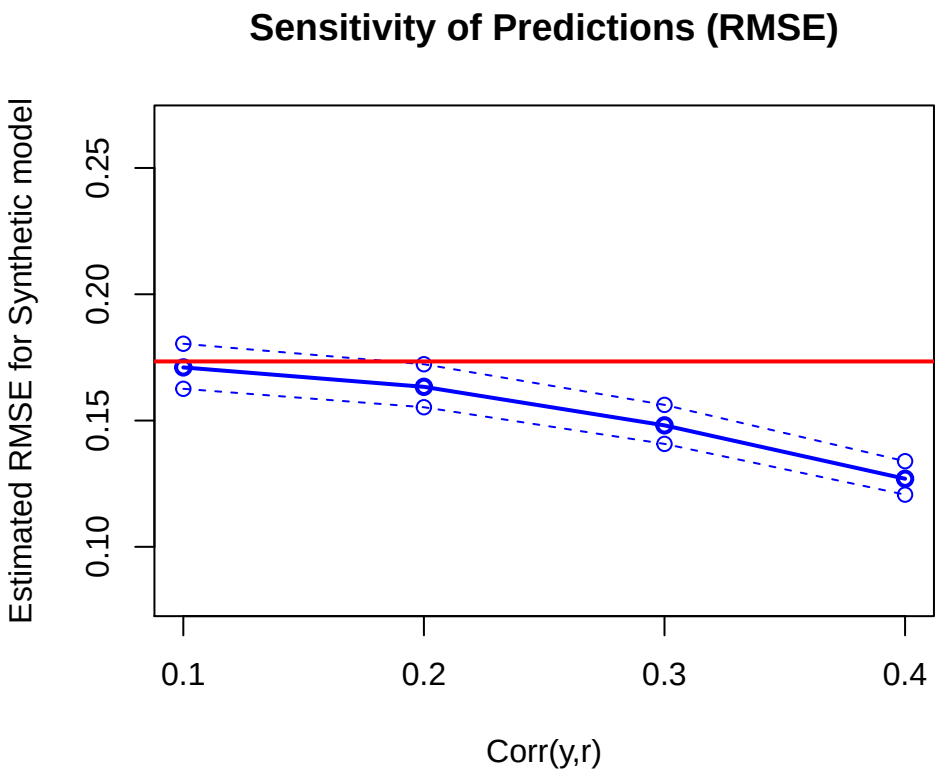
RMSE

In the following chunk of code we construct confidence intervals for the $RMSE$. The standard error of the $RMSE$ can be derived as follows:

$$se_{RMSE} = sqrt\frac{n}{\chi^2_{1-\alpha,n}} \cdot RMSE. \tag{3}$$

Source: https://stats.stackexchange.com/questions/78079/confidence-interval-of-rmse.

```
par(mar=c(5.1, 4.1, 4.1, 8.1), xpd=TRUE)
plot(sensitivity$`RMSE synthetic model`,
     main = "Sensitivity of Predictions (RMSE)",
     xlab = "Corr(y,r)",
     ylab = "Estimated RMSE for Synthetic model",
     xaxt='n',
     type = "o",
     col = "blue",
     lwd = 2,
     ylim=c(min(sensitivity$`RMSE original model`*sqrt(1000/qchisq(0.05,df = 1000)))-0.1,
            max(sensitivity$`RMSE original model`*sqrt(1000/qchisq(0.95,df = 1000)))+0.1))
par(xpd = FALSE)
lines(sensitivity$`RMSE synthetic model`*sqrt(1000/qchisq(0.99,df = 1000)), col = "blue", lty=2, type =
lines(sensitivity$`RMSE synthetic model`*sqrt(1000/qchisq(0.01,df = 1000)), col = "blue", lty=2, type =
abline(h= sensitivity$`RMSE original model`, col = "red", lwd = 2)
axis(1, at=1:(length(sensitivity$`RMSE synthetic model`)), labels=c(seq(0.1,0.4,0.1)))
```



R squared

In the following chunk of code we construct confidence intervals for the R^2. The standard error of the R^2 can be derived as follows:

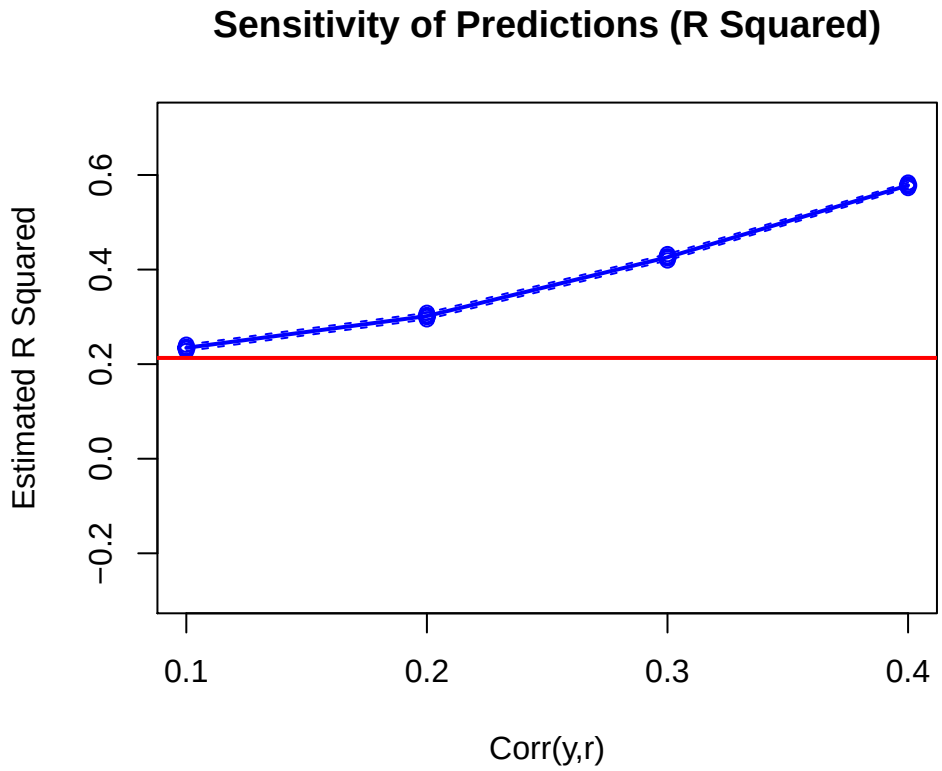
$$se_{R^2} = \sqrt{\frac{4R^2(1-R^2)^2(n-k-1)^2}{(n^2-1)(n+3)}} \tag{4}$$

where k is the number of predictors. See Cohen et al. (2003), Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences, p. 88 for details.

```
n = nrow(train_bart$X)
k = ncol(train_bart$X)
r2 = sensitivity$`Rsquared original model`
ub_new <- lb_new <- c()
for(j in (1:length(sensitivity$`Rsquared synthetic model`))){
  r2_new = sensitivity$`Rsquared synthetic model`[j]
  se_r2_new <- sqrt((4*r2_new*(1-r2_new)^2*(n-k-1)^2)/((n^2-1)*(n+3)))
  ub_new[j] = r2_new + 2.58*se_r2_new # Change Z-score for difference alpha levels
  lb_new[j] = r2_new - 2.58*se_r2_new # Change Z-score for difference alpha levels
}

par(mar=c(5.1, 4.1, 4.1, 8.1), xpd=TRUE)
plot(sensitivity$`Rsquared synthetic model`,
     main = "Sensitivity of Predictions (R Squared)",
     xlab = "Corr(y,r)",
     ylab = "Estimated R Squared",
```

```
     xaxt='n',
     type = "o",
     col = "blue",
     lwd = 2,
     ylim=c(r2-0.5,r2+0.5))
par(xpd = FALSE)
lines(lb_new, col = "blue", lty=2, type = "o")
lines(ub_new, col = "blue", lty=2, type = "o")
abline(h= r2, col = "red", lwd = 2)
axis(1, at=1:(length(sensitivity$`Rsquared synthetic model`)),
     labels=c(seq(0.1,0.4,0.1)))
```



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.

Standardized differences in mean and their standard deviations from Cohen (1988, p.44).

```
# Standardized difference in means
diff.means <- ppd_mean - new_ppd_mean
standard.diff.means <-  (diff.means)/sqrt((ppd_sd^2 + new_ppd_sd^2)/2)

# 99% CI (t-student distribution)
x0 <- standard.diff.means - 2.58 *
```

```
  sqrt((ppd_sd^2 + new_ppd_sd^2)/2)
x1 <-standard.diff.means  + 2.58 *
  sqrt((ppd_sd^2 + new_ppd_sd^2)/2)
```

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