PDF v. ICR

Falco J. Bargagli Stoffi

2/12/2018

Why not just using ICR?

In this section of the Appendix we check what is the comparative advantage of using our Machine-Learning-based definition of Persistently Distressed Firms (PDF) rather than other commonly used deterministic indicators (i.e., Interest Covarage Ratio [ICR], negative added value).

The first and more obvious advantage is that the PDF indicator can be constructed even in the presence of missing data. As we saw in the Section of the paper about missing data, there are significant missingness patterns in the two variables that compose the ICR (EBIT and Interest Paid).

The second advantage is that, while ICR entails a deterministic definition of zombies, PDF represents a probabilistic based definition that can be tuned to embody different "likelihoods" of being zombie.

Beside these clear advantages, there are more subtle benefits that can be obtained by using PDF instead of ICR. To highlight these advantages we focus on a particular dimension of our PDF definition. To be a PDF a firm needs to be at high risk of failure (but not failing) for three consecutive years. Our PDF definition focuses on those firms that "are predicted to fail, but are surviving" (following the ML literature we could call them *false positives*). Hence, a good indicator of "*zombieness*" should be able to discriminate between *false positives* and *true positives*.

In this spirit, we develop a two stage algorithm to highlight which are the best predictors of *false positives*. In the first stage, we train a Logit model to predict the probability of failure:

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

and to obtain the fitted values $\hat{y}_i$. In the second stage, we fit a LASSO just on those observations with $\hat{y}_i = 1$ (*positives*) to get the most important predictors (namely, those with non-zero coefficient) for the *false positives*.

In the following code chuncks the implementation of this "two-stage" algorithm.

Load Packages and Data

In this chunk we load the packages and the data used for the analysis, and we split the overall sample in a *training* and a *test* set.

```
options(java.parameters = "-Xmx50g")
library(rJava)
library(bartMachine)
library(haven)
library(plyr)
library(dplyr)
library(PRROC)
library(caret)
library(glmnet)

#Upload dati
setwd("G:\\Il mio Drive\\Research\\Italian Firms\\Zombie Hunting New Data")
```

```
#LOAD DATA

data <- read_dta("data_lagged_10_07.dta")

#OMIITED DATA
myvariables <- c("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 <- na.omit(data[myvariables])
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_cons", "fin_cons100", "inv",
                "ICR_t", "time", "real_SA", "shareholders_funds", "NEG_VA",
                "ICR_failure", "profitability", "misallocated_fixed",
                "interest_diff")

### Define samples
set.seed(123)
train_sample <- sample(seq_len(nrow(dati)), size = nrow(dati)*0.5)
train <- as.data.frame(dati[train_sample,])
test <- as.data.frame(dati[-train_sample,])
```

In this chunck we contruct the LOGIT model, we get the predicted probabilities and the confusion matrix. We use as a threshold 0.3 (namely, $\hat{p}_i(Y_i = 1|X_i = x) > 0.3 \rightarrow \hat{y}_i = 1$), however this threshold can be moved up to 0.5 without any meaninful change.

```
log = glm(formula = train$failure ~ ., family = binomial, data = train)
prob_pred = predict(log, type = 'response', newdata = test)
prediction <- as.numeric(prob_pred > 0.3) # change up to 0.5
cmlog=table(test$failure,prediction)
cmlog
```

```
##    prediction
##         0     1
##   0 63392   326
##   1  1830   301
```

In this chunck we use a LASSO model to select the variables with the highest predictive power for *false positives*.

```
test$prediction <- prediction
positive <- test[which(test$prediction==1),]
positive$iso <- as.numeric(as.factor(positive$iso))
positive$control <- as.numeric(as.factor(positive$control))
```

```
x<-as.matrix(positive[, 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_cons", "fin_cons100", "inv",
                          "ICR_t",  "time", "real_SA", "shareholders_funds", "NEG_VA",
                          "ICR_failure", "profitability", "misallocated_fixed",
                          "interest_diff")])
y <- as.numeric(positive$prediction==1 & positive$failure==0)

mod <- cv.glmnet(x , y, alpha=1)
as.matrix(coef(mod, mod$lambda.1se))
```

```
##                                1
## (Intercept)           0.493922108
## iso                  -0.013582090
## tfp_acf               0.012983613
## Number_of_patents     0.000000000
## Number_of_trademarks  0.000000000
## consdummy             0.087160526
## control               0.003484858
## nace_2                0.000000000
## fin_rev               0.000000000
## int_paid              0.000000000
## ebitda                0.000000000
## cash_flow             0.000000000
## depr                  0.000000000
## revenue               0.000000000
## total_assets          0.000000000
## long_term_debt        0.000000000
## employees             0.000000000
## added_value           0.000000000
## materials             0.000000000
## wage_bill             0.000000000
## loans                 0.000000000
## int_fixed_assets      0.000000000
## fixed_assets          0.000000000
## current_liabilities   0.000000000
## liquidity_ratio       0.000000000
## solvency_ratio        0.000000000
## current_assets        0.000000000
## fin_expenses          0.000000000
## net_income            0.000000000
## fin_cons              0.000000000
## fin_cons100           0.000000000
## inv                   0.000000000
## ICR_t                 0.000000000
## time                  0.000000000
## real_SA               0.000000000
## shareholders_funds    0.000000000
## NEG_VA                0.000000000
## ICR_failure           0.000000000
```

```
## profitability          0.000000000
## misallocated_fixed    -0.014444096
## interest_diff          0.000000000
```

```
row.names(as.matrix(coef(mod, mod$lambda.1se)))[which(as.matrix(coef(mod, mod$lambda.1se))!=0)]
```

```
## [1] "(Intercept)"        "iso"                "tfp_acf"
## [4] "consdummy"          "control"            "misallocated_fixed"
```

The indicator that seems to have the best disciminative power are the productivity level, the consolidated variable and the zombie indicator from Schivardi, Sette and Tabellini (2017) (*misallocated_fixed*). Hence, ICR is not among these indicators meaning that ICR alone can't catch the component of "resistence" among highly distressed firms that makes them zombies.

Exclusion of ICR from the predictors

A second central check, to understand how important is the contribute of ICR to the predicted failure probability, is to evaluate the performance of the model when we rule out its contribute as a predictor. This is particularly interesting if we focus just on the subsample of observations that have an ICR bigger than one (namely, are not defined as zombie firms).

In the following chunks we subset the data in a way that our favourite ML algorithm (BART) is trained on samples where the number of units with $Y_i = 1$ and $Y_i = 0$ is the same. This trick is used to improve the predicive power of the model.

```
attach(dati)
predictors <- dati[predictors]
output <- as.numeric(failure == 1)
detach(dati)

model1<-data.frame(predictors, output)
colnames(model1)[length(model1)]<-"output"
model1_0<-subset(model1,model1$output==0)
model1_1<-subset(model1,model1$output==1)
```

In this chunck we define a function for the BART algorithm.

```
### Define functions
bart <- function(train1, test1, output){
  train1 <- as.data.frame(train1)
  test1 <- as.data.frame(test1)
  train1$predictors <- train1[predictors]
  test1$predictors <- test1[predictors]
  bart_machine<-bartMachine(train1$predictors,as.factor(train1$output))
  fitted.results.bart <- 1- round(predict(bart_machine, test1$predictors,
                                          type='prob'), 6)
  #fitted.results.bart <-predict(bart_machine, test1$predictors,  type='class')
  prediction <- as.numeric(fitted.results.bart > 0.5)
  cmsl=table(test1$output,prediction)
  res_all<-(cmsl[1,1]+cmsl[2,2])/(length(test1$output))
  res_1<-cmsl[2,2]/(cmsl[2,1]+cmsl[2,2])
  res_0<-cmsl[1,1]/(cmsl[1,1]+cmsl[1,2])
  res<-cbind(res_all,res_1,res_0)
}
```

And in this last chunck we fit the models (namely, one with all predictors and one just on the subset of observations where ICR is more than one [*ICR_failure*==0]) on two bootstraped samples from the data.

```
### Vector of predictors
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_cons", "fin_cons100", "inv",
                "ICR_t", "time", "real_SA", "shareholders_funds", "NEG_VA",
                "ICR_failure", "profitability", "misallocated_fixed",
                "interest_diff")

### Create matrix to save bootstrapped results (N = B)
B=2
number_of_models=3
results<-matrix(data=NA, nrow = B, ncol = number_of_models*2)

system.time({
  for (i in (1:B)) {
    ### Start loop
    set.seed(123 + i) #setting seed for reproducible results

    # Repeatedly (B) draw subsamples
    model1_0sub <- model1_0[sample(seq_len(nrow(model1_0)),
                                   size = nrow(model1_1),
                                   replace = TRUE),]
    model1_full <- rbind(model1_0sub,model1_1)

    # split into training and test (easier to handle)
    train_ind <- sample(seq_len(nrow(model1_full)),
                        size = nrow(model1_full)*0.5)

    train1 <- model1_full[train_ind,]
    test1 <- model1_full[-train_ind,]

    #BART ALL
    results[i,1:3]<-bart(train1,test1, output)

    #BART ICR = 0
    model1_full <- model1_full[which(model1_full$ICR_failure==0),]
    train_ind <- sample(seq_len(nrow(model1_full)),
                        size = nrow(model1_full)*0.5)

    train1 <- model1_full[train_ind,]
    test1 <- model1_full[-train_ind,]

    results[i,4:6]<-bart(train1,test1, output)

    # LOADING STATUS
    print(i/B)
  }
```

```
})
```

As we can see from the outcome of this chunck the overall predictions for the two models are not signficantly different at a significance level of 0.95.

```
summary(results)
```

```
##        V1               V2               V3               V4
##  Min.   :0.8022   Min.   :0.8521   Min.   :0.7526   Min.   :0.7694
##  1st Qu.:0.8044   1st Qu.:0.8522   1st Qu.:0.7569   1st Qu.:0.7731
##  Median :0.8066   Median :0.8524   Median :0.7613   Median :0.7769
##  Mean   :0.8066   Mean   :0.8524   Mean   :0.7613   Mean   :0.7769
##  3rd Qu.:0.8088   3rd Qu.:0.8525   3rd Qu.:0.7656   3rd Qu.:0.7807
##  Max.   :0.8111   Max.   :0.8527   Max.   :0.7699   Max.   :0.7844
##        V5               V6
##  Min.   :0.6988   Min.   :0.8124
##  1st Qu.:0.7053   1st Qu.:0.8146
##  Median :0.7118   Median :0.8167
##  Mean   :0.7118   Mean   :0.8167
##  3rd Qu.:0.7183   3rd Qu.:0.8188
##  Max.   :0.7248   Max.   :0.8210
```

```
t.test(results[,1], results[,4])
```

```
##
##  Welch Two Sample t-test
##
## data:  results[, 1] and results[, 4]
## t = 3.4103, df = 1.6209, p-value = 0.1014
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.01757694  0.07704034
## sample estimates:
## mean of x mean of y
## 0.8066331 0.7769014
```