Capstone Project: Bankrupt Prediction

Luis Octavio Gaytán Valejo

INTRODUCTION

The creditors (a person, company or a government) lends money to another companies, named debtors, expecting that the debtors will be able to repay the money along with the interest. However, even if debtors are willing to, some of them won't fulfill their commitments. If this situation continues and the debtors cannot fulfill with the rest of their commitments, the companies can be declared bankrupt and all their assets might be valued and used to repay a portion of the outstading debt. This implies that the creditors will recover only a fraction of their money, which will provoke them losses.

In order to avoid situations like the one described before, investors and banks uses credit analysis to measure the ability of the debtors to meet its obligations. Some of the tools used in credit analysis are credit analysis ratios, this ratios are calculated using information extracted from the financial statements of each company. Some of the most used ratios are EBITDA (Earnings Before Interest, Taxes, Depreciation and, Amortization), Gross Profit Margin, Debt to Assets Ratio and Interest Coverage Ratio.

The objective of this work is to implement a machine learning algorithm that, by using different credit analysis ratios, predicts if a company will be bankrupt. The data used for this work comes from the "Company Bankruptcy Prediction" dataset, which is publicly available on Kaggle, and was created originally by Deron Liang and Chih-Fong Tsai from the National Central University, Taiwan. According to the description provided on Kaggle, the data were collected from the Taiwan Economic Journal for the years 1999 to 2009, and company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange. In the dataset exists a variable that indicates if a company is in bankrptcy. If a company is bankrupt, the value of the variable will be 1; but if not, the value will be 0.

SECTION 1: DOWNLOADING PACKAGES AND DATASET

The necessary packages for this work will be `tidyverse`, `dplyr`, `lubridate`, `purrr`, `caret`, `smotefamily`, `glmnet`, `RCurl`, `mltools` and, `knitr`. With the following code we will install and load them.

Even when the dataset was obtained on Kaggle, it will be downloaded from a repository at github and stored in an object named `dat`.

```
dl <- tempfile()
download.file("https://raw.githubusercontent.com/lgaytanv/Company_Bankruptcy_Prediction/main/Company_Ba

dat <- read.csv("Company_Bankruptcy_Prediction.csv")
```

As we can observe, the dataset is composed by 6819 rows and 96 columns.

```
kable(data.frame(Rows = dim(dat)[1], Columns = dim(dat)[2]),"simple", align = "c")
```

Rows	Columns
6819	96

The first column of the datasets is a binary variable where the value is 1 if a company is bankrupt, and 0 otherwise. The rest of the columns are different credit analysis ratios that were calculated using the financial statements of each company.

```
kable(data.frame(Column.Names.Example = head(colnames(dat),10L)),"simple")
```

Column.Names.Example
Bankrupt.
ROA.C..before.interest.and.depreciation.before.interest
ROA.A..before.interest.and. . . after.tax
ROA.B..before.interest.and.depreciation.after.tax
Operating.Gross.Margin
Realized.Sales.Gross.Margin
Operating.Profit.Rate
Pre.tax.net.Interest.Rate
After.tax.net.Interest.Rate
Non.industry.income.and.expenditure.revenue

SECTION 2: DATASET PREPARATION - DEALING WITH MULTI-COLLINEARITY

First, we will remove the ratios of the dataset whose values are always the same because those ratios won't help to develop our model.

```
unused.ratio <- which(apply(dat,2,function(x){ all(x == x[1])}))
kable(names(unused.ratio),"simple", col.names = "Ratio", align = "c")
```

Ratio
Net.Income.Flag

```
dat <- dat[,-unused.ratio]
```

Now, given that many of the ratios that are used in credit analysis are calculated using another ratios this generates the problem of multicollinearity. Multicollinearity is a situation where the variables that are used as predictors are highly intercorrelated. This can lead to several problems while developing our model.

In order to solve this problem, we will develop a two step process to only keep the ratios that are significant and doesn't have the problem of multicollinearity.

- Step 1: With the following code we will eliminate the ratios whose correlation with the bankruptcy variable is less than the median of the absolute value of the correlation of between bankruptcy variable and all the ratios. Also, to easily handle the correlations we will change the names of the variables. The bankruptcy variable will be named y, while the ratios will be named as $x_1,\ x_2,\ \ldots\ ,\ x_n$.

```
col_index <- which(abs(cor(dat)[,1]) > median(abs(cor(dat)[,1])))

dat <- dat[,col_index]

column_names <- colnames(dat)

colnames(dat) <- c("y",paste("x",1:(dim(dat)[2]-1),sep = ""))
```

- Step 2: Next, we will eliminate the variables whose intercorrelation is high. As a rule of thumb, a correlation is high when its absolute value is equal or higher than 0.7. Using the next piece of code we can eliminate the variables which meet the described criteria.

```
correlation <- cor(dat)

correlation[upper.tri(correlation, diag=TRUE)] <- NA

correl_frame <- reshape2::melt(correlation, na.rm=TRUE, value.name="cor")

xs <- correl_frame %>% filter(abs(cor) >= 0.7) %>% select(Var1) %>% unlist() %>% table()

names_xs <- names(xs)[which(xs != 0)]

xs_index <- which(colnames(dat) %in% names_xs)

dat <- dat[,-xs_index]
```

As we can observe, our model is comprised by 21 ratios whose names are presented in the next table:

```
variable.index <- colnames(dat)[-1] %>% str_remove("x") %>% as.numeric()

variable.index <- variable.index + 1

kable(data.frame(Varieble.Number = seq(1,length(variable.index), by = 1), Variable = colnames(dat)[-1],
```

Varieble.Number	Variable	Ratio.Name
1	x1	ROA.C..before.interest.and.depreciation.before.interest
2	x4	Operating.Gross.Margin
3	x6	Cash.flow.rate
4	x7	Tax.rate..A.
5	x8	Net.Value.Per.Share..B.
6	x12	Cash.Flow.Per.Share
7	x15	Debt.ratio..
8	x17	Borrowing.dependency
9	x18	Contingent.liabilities.Net.worth
10	x22	Total.Asset.Turnover
11	x23	Fixed.Assets.Turnover.Frequency
12	x24	Operating.profit.per.person
13	x25	Working.Capital.to.Total.Assets
14	x26	Quick.Assets.Total.Assets
15	x27	Cash.Total.Assets
16	x28	Cash.Current.Liability

Varieble.Number	Variable	Ratio.Name
17	x34	Total.expense.Assets
18	x35	Fixed.Assets.to.Assets
19	x38	Cash.Flow.to.Total.Assets
20	x40	Current.Liability.to.Current.Assets
21	x41	Liability.Assets.Flag
22	x46	Equity.to.Liability

SECTION 3: DATA PARTITION AND BALANCING TRAIN SET

We will proceed to make partitions to create the train and test sets. However, we can see that the percentage of bankrupcies that exists in our dataset is quite low, around 3%.

```
kable(data.frame(Bankruptcy = sum(dat$y), Bankruptcy.As.Percentage = mean(dat$y)*100),"simple",align =
```

Bankruptcy	Bankruptcy.As.Percentage
220	3.226279

Given the few observations of bankruptcies, to create the train set we will use a bigger proportion of the data than the proportion used to the test set. The train set will have 90% of the data, while the test set will have the remaining 10%.

```
test_index <- createDataPartition(dat[,1],times = 1, p = 0.10, list = FALSE)
test_set <- dat[test_index,]
train_set <- dat[-test_index,]

test_x <- test_set[,-1]
test_y <- test_set[,1] %>% as.factor()

train_x <- train_set[,-1]
train_y <- train_set[,1] %>% as.factor()
```

Nevertheless, if we create models using the current train set we might get misleading results. With the next code, we will create 3 models with the train set: GLM, LDA and QDA models. Later, we will compute the accuracy for each model.

```
glm.model.unbalanced <- caret::train(train_x,train_y, method = "glm")

glm.prediction.unbalanced <- predict(glm.model.unbalanced,test_x)

glm.table.unbalanced <- confusionMatrix(glm.prediction.unbalanced,as.factor(test_y))$table

glm.accuracy.unbalanced <- confusionMatrix(glm.prediction.unbalanced,test_y)$overall["Accuracy"]


lda.model.unbalanced <- caret::train(train_x,train_y, method = "lda")

lda.prediction.unbalanced <- predict(lda.model.unbalanced,test_x)
```

```
lda.table.unbalanced <- confusionMatrix(lda.prediction.unbalanced,test_y)$table

lda.accuracy.unblanced <- confusionMatrix(lda.prediction.unbalanced,test_y)$overall["Accuracy"]


qda.model.unbalanced <- caret::train(train_x,train_y, method = "qda")

qda.prediction.unbalanced <- predict(qda.model.unbalanced,test_x)

qda.table.unbalanced <- confusionMatrix(qda.prediction.unbalanced,test_y)$table

qda.accuracy.unbalanced <- confusionMatrix(qda.prediction.unbalanced,test_y)$overall["Accuracy"]

kable(data.frame(GLM.Model = glm.accuracy.unbalanced, LDA.Model = lda.accuracy.unblanced, QDA.Model = q
```

	GLM.Model	LDA.Model	QDA.Model
Accuracy	0.9765396	0.973607	0.9721408

We can observe that the value of accuracy in the three models is high; however, given that there are few companies with bankruptcies in proportion with the companies that don't we can say that our dataset is unbalanced. We call unbalanced to a dataset when the classification that we want to predict is a minority among the data. By checking the confussion matrix of the three models we can observe that the high accuracies are due to the fact that, most of the times, the algorithm only predicts 0 values.

```
glm.table.unbalanced
```

```
##            Reference
## Prediction   0   1
##          0 665  15
##          1   1   1
```

```
lda.table.unbalanced
```

```
##            Reference
## Prediction   0   1
##          0 662  14
##          1   4   2
```

```
qda.table.unbalanced
```

```
##            Reference
## Prediction   0   1
##          0 663  16
##          1   3   0
```

When this happens, using the accuracy to evaluate the model won't be useful and we need to use another metrics to do it.

3.1 Metrics for unbalanced datasets - G-Mean and MCC.

In this work we will use two metrics, the geometric mean between Sensitivity and Specificity or G-Mean, and the Matthew's Correlation Coefficient or MCC:

- G-Mean: is a metric that measures the balance between classification performances on both the majority and minority classes. A value close to 1 means that our model has a good performance; on the other hand, if the G-Mean of our model is close to 0, our model isn't performing well.
- Matthew's Correlation Coefficient: it measures the prediction of binary clasification algorithm. If MCC = -1, the prediction is totally contrary to the observations; if MCC = 0, the prediction is as good as random prediction; and, if MCC = 1, all the predictions are equal to the observations.

3.2 Balancing Train Set - SMOTE

The Synthetic Minority Oversampling Technique (SMOTE), is a technique to deal with unbalanced datasets. Using a k-nearest neighbors, SMOTE creates synthetic data. This procedure is repeated until the minority class has a similar proportion as the majority class.

With the following code we will create our balanced train set using SMOTE:

```
newData <- SMOTE(train_x,train_y)

newData$data[,dim(newData$data)[2]] <- newData$data[,dim(newData$data)[2]] %>% as.factor()

balanced_train_x <- newData$data[,-dim(newData$data)[2]]

balanced_train_y <- newData$data[,dim(newData$data)[2]]
```

We can easily check that our train set now has almost the same bankruptcies as non-bankrutcies.

```
kable(table(balanced_train_y),"simple",align = "c")
```

balanced_train_y	Freq
0	5933
1	5916

SECTION 4: MODELS CREATION

In this section we will create 5 models using the `caret` package, and an `ensemble` model using the predictions of the first 5 models. The models that we use are `glm`, `ridge`, `elastic`, `lda` and `qda`.

4.1 GLM Model

```
glm.model <- caret::train(balanced_train_x,balanced_train_y, method = "glm")

glm.prediction <- predict(glm.model,test_x)

glm.confusion <- confusionMatrix(glm.prediction,as.factor(test_y))
```

4.2 Ridge Model

```
lambda.parameters <- seq(0,5,by = 0.5)

ridge.model <- caret::train(balanced_train_x,balanced_train_y, method = "glmnet", tuneGrid = expand.gri

ridge.prediction <- predict(ridge.model,test_x)

ridge.confusion <- confusionMatrix(ridge.prediction,as.factor(test_y))
```

4.3 Elastic Model

```
elastic.model <- caret::train(balanced_train_x,balanced_train_y, method = "glmnet", trControl = trainCo

elastic.prediction <- predict(elastic.model,test_x)

elastic.confusion <- confusionMatrix(elastic.prediction,as.factor(test_y))
```

4.4 LDA Model

```
lda.model <- caret::train(balanced_train_x,balanced_train_y, method = "lda")

lda.prediction <- predict(lda.model,test_x)

lda.confusion <- confusionMatrix(lda.prediction,as.factor(test_y))
```

4.5 QDA Model

```
qda.model <- caret::train(balanced_train_x,balanced_train_y, method = "qda")

qda.prediction <- predict(qda.model,test_x)

qda.confusion <- confusionMatrix(qda.prediction,test_y)
```

4.6 Ensemble Model

```
models.matrix <- data.frame(glm.prediction,ridge.prediction,elastic.prediction,lda.prediction,qda.predi

models.matrix <- apply(models.matrix,2,function(x)ifelse(x == 1,1,0))

ensemble.prediction <- ifelse(rowMeans(models.matrix) > 0.5,1,0)

ensemble.prediction <- ensemble.prediction %>% as.factor()

ensemble.confusion <- confusionMatrix(ensemble.prediction,test_y)
```

4.7 Models Results

In the next piece of code we will present the sensitivy and specificity obtained for each model.

```
confusion.matrix <- list(glm.confusion,ridge.confusion,elastic.confusion,lda.confusion,qda.confusion,ens

methods <- c("glm", "ridge", "elastic", "lda", "qda" , "ensemble")

models.sensitivities <- lapply(confusion.matrix,function(x)x$byClass["Sensitivity"]) %>% unlist()

names(models.sensitivities) <- methods

models.sensitivities
```

```
##       glm     ridge   elastic       lda       qda  ensemble
## 0.8768769 0.8693694 0.8768769 0.8513514 0.9954955 0.8843844
```

```
models.specificities <- lapply(confusion.matrix,function(x)x$byClass["Specificity"]) %>% unlist()

names(models.specificities) <- methods

models.specificities
```

```
##      glm    ridge  elastic      lda      qda ensemble
##     0.75     0.75     0.75     0.75     0.00     0.75
```

The qda model is the one that has de highest sensitivity, on the other hand, the glm has the highest specificity.

Now, when we have highly unbalanced datasets is convenient to examine the confusion matrix of the model that we create. The confusion matrix for each model are the following:

```
glm.confusion$table
```

```
##           Reference
## Prediction   0   1
##          0 584   4
##          1  82  12
```

```
ridge.confusion$table
```

```
##           Reference
## Prediction   0   1
##          0 579   4
##          1  87  12
```

```
elastic.confusion$table
```

```
##           Reference
## Prediction   0   1
##          0 584   4
##          1  82  12
```

```
lda.confusion$table
```

```
##            Reference
## Prediction   0   1
##          0 567   4
##          1  99  12
```

```
qda.confusion$table
```

```
##            Reference
## Prediction   0   1
##          0 663  16
##          1   3   0
```

```
ensemble.confusion$table
```

```
##            Reference
## Prediction   0   1
##          0 589   4
##          1  77  12
```

As we can observe on the confusion matrix of each model, 5 of the 6 models tend to predict many false positives. Despite this result we need to keep in mind that the bankruptcy of company implies severe losses for its creditors, that's why many financial institution tend to be conservative and deny a credit if they consider that the solicitor has a considerable risk of defaulting. Nevertheless, given that financial institutions obtain a high portion of its revenues by the interests of loans, they try to give as many credits as possible but only to debtors that are within a certain level of default risk.

Considering the reasons presented, we need a metric that balances the results of sensitivity and specificity, that is the G-Mean. In the following code we present the for the G-Mean and MCC obtained for each model.

```
Gmean <- lapply(confusion.matrix, function(x){
  sqrt(x$byClass[1] * x$byClass[2])
}) %>% unlist

kable(data.frame(methods,Gmean),"simple",align = "c")
```

methods	Gmean
glm	0.8109609
ridge	0.8074819
elastic	0.8109609
lda	0.7990704
qda	0.0000000
ensemble	0.8144251

```
predictions <- list(glm.prediction,ridge.prediction,elastic.prediction,lda.prediction,qda.prediction,ens

mccs <- lapply(predictions, function(x){
  mcc(x,test_y)
```

```
}) %>% unlist

kable(data.frame(methods,mccs),"simple",align = "c")
```

methods	mccs
glm	0.2752496
ridge	0.2661307
elastic	0.2752496
lda	0.2465732
qda	-0.0103026
ensemble	0.2850541

Finally, we observe that according to the G-Mean, the best model is ensemble; and according to the MCC, the best model is ensemble.

SECTION 5: CONCLUSION

The results of this works shows that linear models, like `glm` or `lda`, have a good performance on data like financial ratios; however, models like `qda` don't perform well on this kind of data. One limitation in this work is the high unbalance of classes, even when we try to balance the dataset with SMOTE, there are many false positives in the predictions.

For future works, it's suggested to collect more data and try to fit another models like decision tree algorithms or penalized models