Study of relationship between economic outcomes and political regimes using machine learning models

A Gagliardi

April 2020

1) Introduction

There is the notion in political science and economics that democratic countries are, in general, more prosperous and economically stable than authoritarian ones. After all, most of the highest per capita income countries, like Norway, Denmark, or Switzerland are long-established democracies, while many of the worst-performing ones, like Venezuela, North Korea, or Zimbabwe are harsh dictatorships.

This idea presents a kind of optimistic foresaw for the future. If free democracies are the ones who can achieve the highest levels of economic power, we could expect a more free and democratic international order in the coming decades.

However, since the break of the 2008 financial crisis, there have been serious doubts about the economic soundness of western democracies. Established developed democracies like Japan, Italy, or France are starting to suffer from low productivity growth, economic stagnation, and rising public debt levels.

On the other hand, many rising economic stars from Asia, like China or Singapore, have obtained high levels of economic growth by mixing economic liberalization policies with authoritarian political systems. Moreover, notorious emerging economies like India, Brasil, or Turkey, considered for many as future powerhouses, are now being ruled by leaders with populist and authoritarian traits, even though they are supposed to be established democracies.

This ambiguous relationship between economic success and sociopolitical freedom ought to be closely observed. In this research, we will use data science techniques and advanced machine learning algorithms to determine whether or not degrees of political freedom and macroeconomics outputs are correlated.

For that purpose we compiled a pooled panel data, comparing the macroeconomic performance and political systems of several countries between 1980 and 1989, distinguishing between **Free**, **Non-Free**, and **Partially-Free** countries for every year.

One important limitation is the availability of the data. Not every macroeconomic statistic will be available for every country and every year, and most machine learning algorithms require a totally balanced dataset to properly work. Moreover, authoritarian or semi-authoritarian regimes are less prone to compile and publish relevant economic data, let alone reliable ones. In many cases, those have to be estimated by institutions like the IMF. As a consequence, the sample data could be biased, by having more data concerning free countries compared to the other two types.

The open-source statistical software R, along with a particular set of their packages, will serve as the computational instrument.

2) Data Analysis

2.1) Data compilation, cleaning, and consolidation

The economic data will be extracted from the *World Economic Outlook* (WEO) database, of the International Monetary Fund (IMF). The dataset comprehends 30 economic indicators from 195 countries between 1980 and 2019.

The political data would be taken from the *Freedom in the World* Report Historical Dataset, published by the NGO Freedom House. This dataset compiles the political profile of more than 200 countries around the world for every year since 1973, classifying them as Free (F), Non-Free (NF) or Partially Free (PF).

Both datasets were downloaded directly from its source web pages in xls formats. They were subsequently cleaned to eliminate unnecessary columns and gathered into *tidy* format. Under this format, every individual data point has its own horizontal line in the dataset.

The data from freedom house required a little more processing, given that the downloaded file was not optimized for data analysis purposes. Furthermore, in many years the timeframes of the data did not exactly coincide with the calendar years (i.e spanning from November of one year to November of the subsequent). In consequence, approximations had to be done in order to make the data workable.

```
Political_Data <- Data_FHouse[c(-1,-2),]
seq_data <- seq(1,139,3)
Political_Data <- Political_Data[,seq_data]
seq_years <- c(1, seq(2,137,3))
years <- Data_FHouse[1,seq_years]
colnames(Political_Data) <- years
Political_Data <- Political_Data %>%
rename("1984" = `Nov.1983-Nov.1984`, "1985" = `Nov.1984-Nov.1985`,
    "1986" = `Nov.1985-Nov.1986`,"1987" = `Nov.1986-Nov.1987`,
    "1988" = `Nov.1987-Nov.1988`, "1989" = `Nov.1988-Dec.1989`,
    "1983" = `Aug.1982-Nov.1983`, "Country" = `Year(s) Under Review`,
    "1981"= `Jan.1981-Aug. 1982`) %>%
gather(key = "Year", value = "Grade", - Country)
rm(seq_data,years)
```

Both datasets were consolidated into a single one, through a common variable called “unity”, which combines the year and country of every individual data point (i.e Argentina 1998). The data was subsequently cleaned to remove missing data, blank spaces and to set the data points as a factor or numeric, depending on the case.

```
Economic_Data$Unity <- paste(Economic_Data$Country,Economic_Data$Year)
Political_Data$Unity <- paste(Political_Data$Country,Political_Data$Year)
Consolidated_Data <- left_join(Economic_Data,Political_Data, by = "Unity")

Consolidated_Data$Amount <- str_remove(Consolidated_Data$Amount,"n/a")
Consolidated_Data$Amount <- as.numeric(Consolidated_Data$Amount)
Consolidated_Data$Grade <- str_remove(Consolidated_Data$Grade,"-")
Consolidated_Data$Grade <- as.factor(Consolidated_Data$Grade)
Consolidated_Data <- Consolidated_Data %>% dplyr::filter(Amount != "") %>%
dplyr::filter(Grade != "") %>%
select(-Year.y,-Country.y,-Scale,-Country.x,-Year.x)
```

Once the information is compiled and combined, it is ready to be analyzed.

2.2) Data exploration

2.2.1) Average Values

The next step is to determine which of the more than 30 variables contained in the IMF dataset could be useful to the model. As a first filtering criterion, there were taken out the indicators that come in absolute values, like the GDP of the population. Those indicators are highly dependent on the size of every country, hence they are not generally useful for comparison purposes. As a consequence, only relative indicators, expressed as a percentage of GDP or yearly rates of variation, will be considered.

```
Consolidated_Data <-  Consolidated_Data  %>%
dplyr::filter(Units %in% c("Percent of GDP","Percent change"))
```

The first analysis method will consist of taking the average value of every macroeconomic indicator, discriminating by the political regime. In theory, indicators showing the biggest variation between types of regimes are the ones that could help us to predict them. A new index called **MaxVar** was designed to show the percentage difference between the smaller and bigger average for every indicator among the three groups of countries.

```
Predictor_Analysis <-  Consolidated_Data %>%
group_by(Subject.Descriptor,Grade) %>%
summarize(Amount = mean(Amount)) %>%
spread(key = Grade, value = Amount) %>%
mutate("MaxVar" = (abs((max(F,NF,PF) - min(F,NF,PF))/min(F,NF,PF)*100)))  %>%
arrange(desc(MaxVar))

kable(Predictor_Analysis)
```

Subject.Descriptor	F	NF	PF	MaxVar
General government primary net lending/borrowing	-0.0530079	-3.665453	-0.2672699	98.553852
Inflation, end of period consumer prices	8.6986637	16.662259	14.9178778	91.549640
Inflation, average consumer prices	9.4038974	17.637703	16.2478159	87.557378
General government net lending/borrowing	-2.1977656	-5.296492	-1.9952826	62.328227
General government revenue	35.5768122	25.243813	23.5971345	50.767510
General government total expenditure	37.7797384	30.797100	25.7708713	46.598607
Current account balance	-1.9012209	-2.504050	-3.4678628	45.176006
General government net debt	37.2937951	45.545065	53.8899595	44.501141
Gross domestic product, constant prices	3.0097541	4.136005	3.9581750	37.420027
Volume of exports of goods and services	5.3622349	7.184068	6.3693742	33.975266
Volume of exports of goods	5.7119189	6.952388	6.3334269	21.717207
Gross national savings	20.9887141	19.921136	18.5833079	12.943907
Volume of Imports of goods	5.7860076	6.220988	6.3577657	9.881738
Volume of imports of goods and services	5.8985084	6.393316	6.3534074	8.388694
Total investment	23.0249314	24.336965	23.6404346	5.698317
General government gross debt	55.4738758	57.435796	56.7272913	3.536657

Based on the resulting table, some highlights can be observed.

- The **General Government primary lending/ borrowing**, also known as *primary budget balance* shows the highest MaxVar value. NF countries tend to show an average deficit of 3.66 percentage points (pp) of GDP, compared with the 0.26 pp of PF and 0.053 pp of NF. This indicator shows budget deficits, excluding interest payments.
- However, F countries show a higher rate of both **General government revenue** and **General government total expenditure** over GDP (35.5 pp and 37.7 pp, respectively), compared with NF and PF countries. The latter two show a lesser stream of revenues (25 and 23 pp, respectively) and expenditure (30 pp and 25 pp). Besides having more or less the same capacity to generate revenues, NF countries showed a higher average expenditure than PF, which explains its higher primary deficit.
- As a result, **General Government net debts** are also higher in NF (45.5 pp of GDP) and PF countries (53.88 pp), than in F countries (37.9 pp). It is interesting how NF countries show higher levels of debt than NF, besides its lower primary deficits. This could be evidence that NF countries have in general less access to credit to cover their deficits.
- Inflation measured as **Inflation, average consumer prices**, also shows interesting results. NF and

PF countries tend to show average annual inflation of between 17 and 15 pp, significantly higher than the average of 8.7 pp presented by F countries. This is a result consistent with the higher budget deficit evidenced in previous sections.

- The **Current account balance** (an indicator showing the balance between entering and exiting streams of funds), also has relevant results. F countries tend to have an average deficit of 1.99 pp, compared with 2.5 of NF and 3.46 of PF. Just as in the case of net government debt, PF countries could have more access to foreign currency creditors than NF.

Together, those 6 indicators give us a clear picture of the distinct behavior of three types of countries at study. At first, if the trends shown in the data are clear enough, a decision tree model could result as the most effective algorithm to apply.

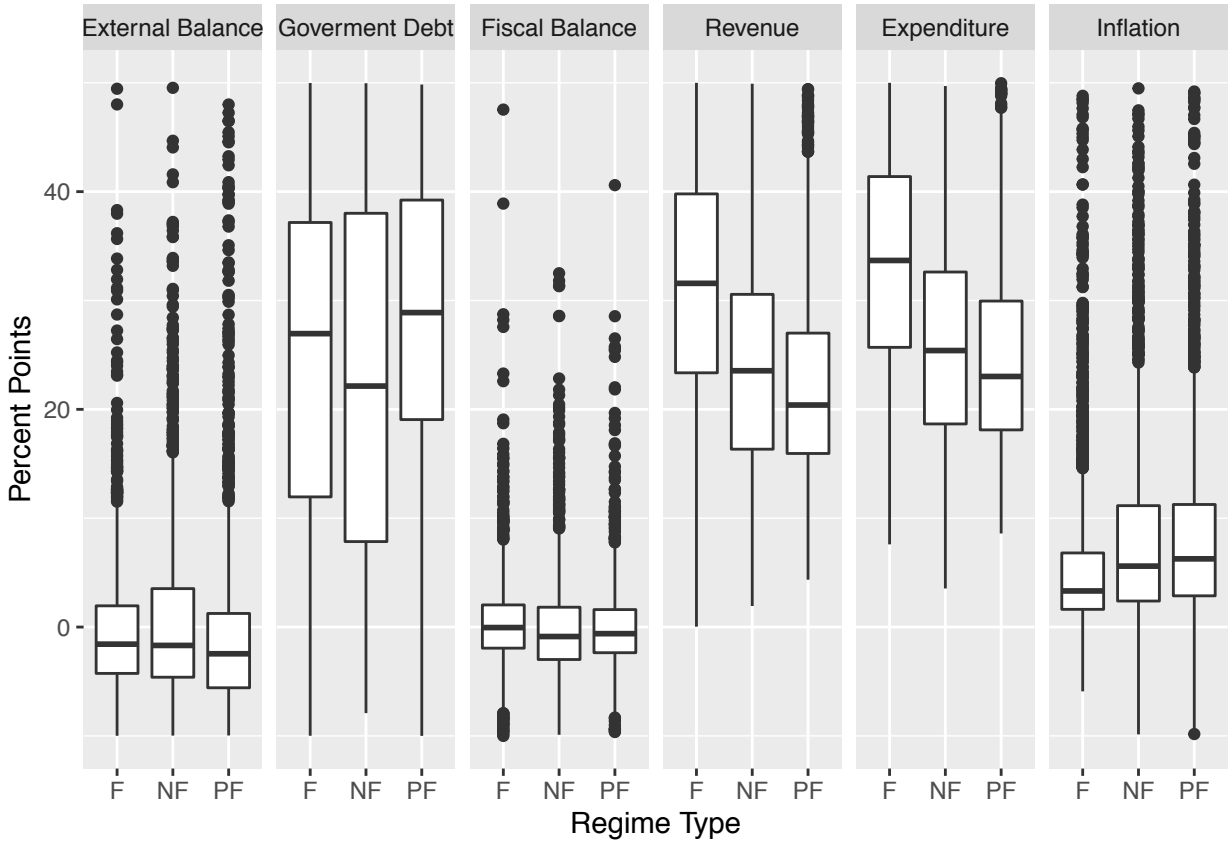
However, a sole analysis based on means may not offer a complete picture. If the dispersion of the indicators is high enough the difference in average values could end up being worthless. It is necessary to apply visualization techniques to gather a more clear picture.

2.2.2) Graphical analysis

2.2.2.1) Boxplot

One of the most commonly used graphical analysis techniques to measure variability is the boxplot, also known as whisker and box graph. The upper and lower borders of every box mark the 75th and the 25th percentile of the variable, while the horizontal line inside the boxes marks the median or 50th percentile. The points above and below the boxes are outlier values.

As the first graphical approximation, we are going to apply a boxplot graph to the six variables previously selected, cutting out the most extreme observation to make the graphs readable.



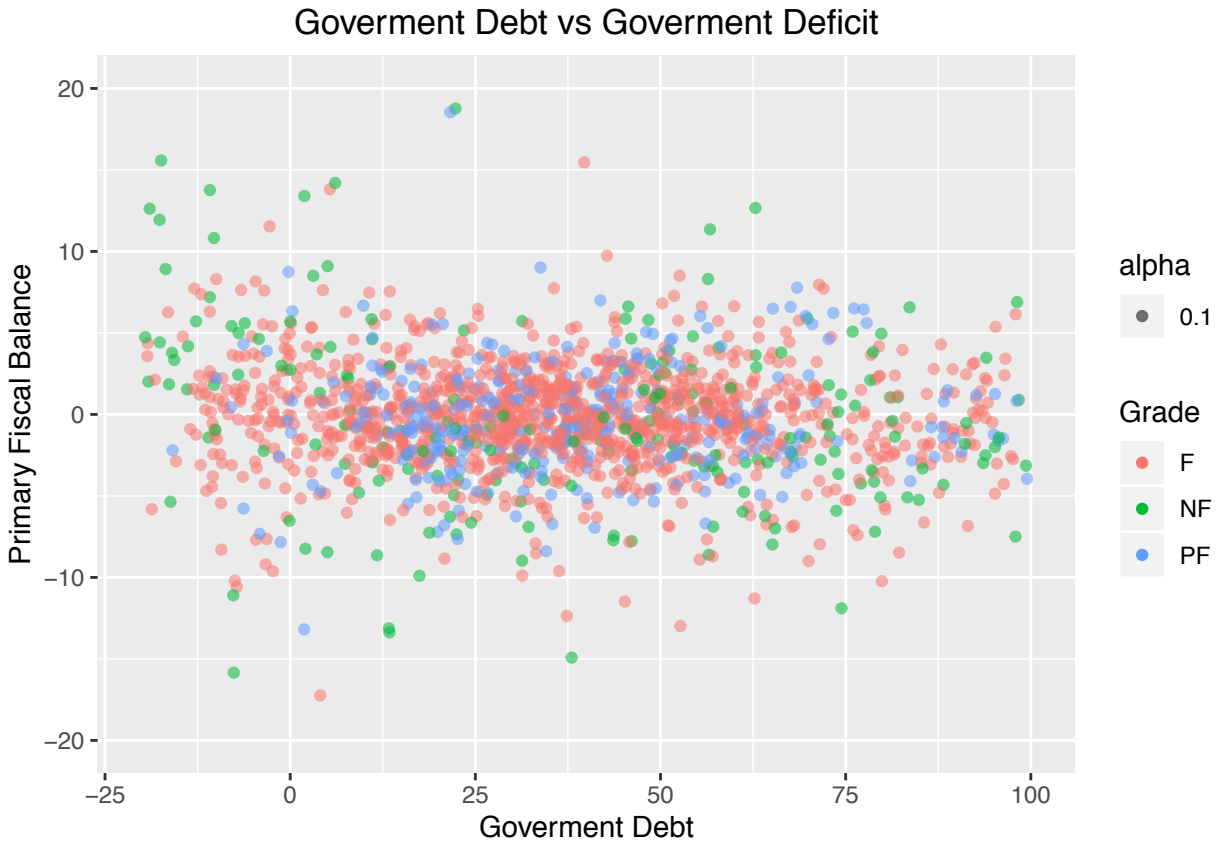
Besides cutting off,It is interesting to see how many outliers data points stand out anyway. At first sight,

macroeconomic indicators tend to be highly volatile, even though the differences between the types of regimes are still visible. The budget variables (Revenue, Expenditure, and deficit) are significantly more stables, containing a bigger proportion of observations inside their respective boxes.

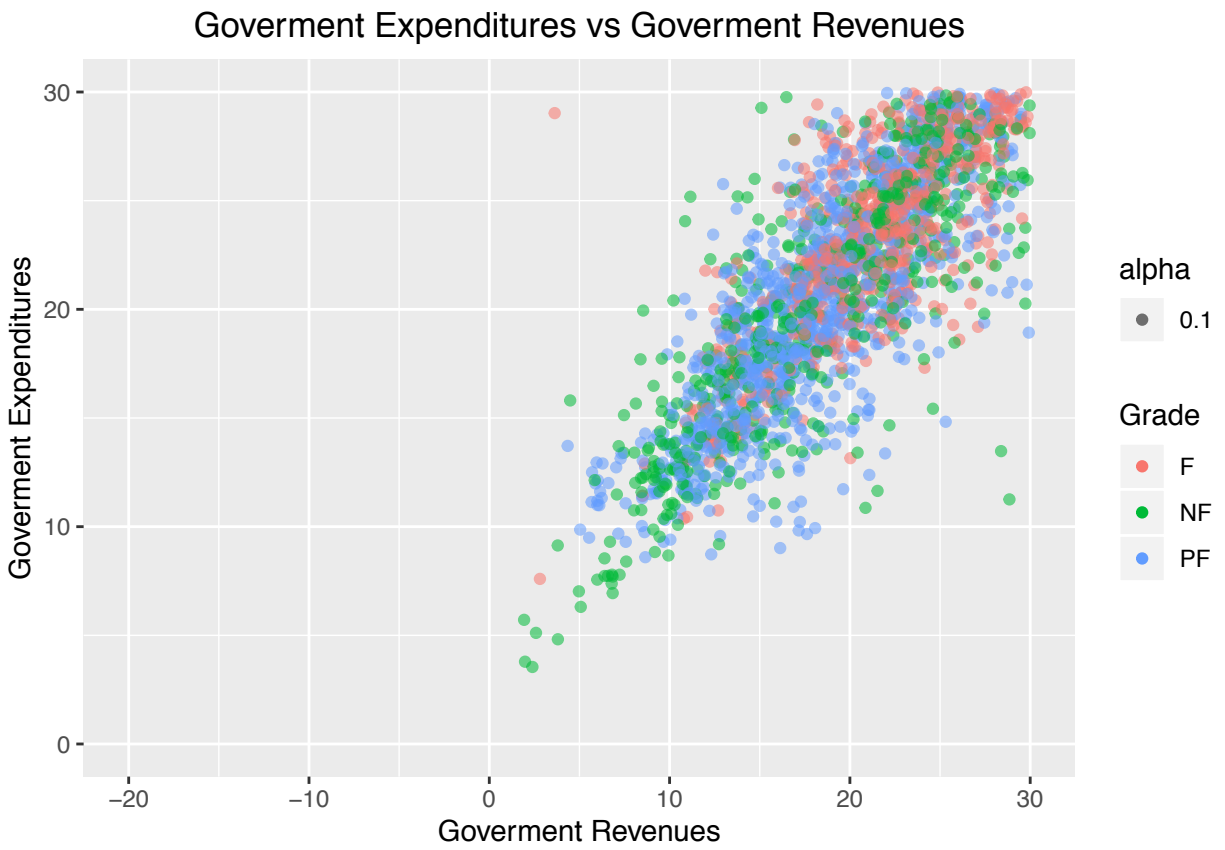
3.2.2.1) Scatter Plots

Another popular way to graph variables is to scatter plot, basically, a bi-dimensional point graph comparing the values of two different variables among a multitude of different individuals.

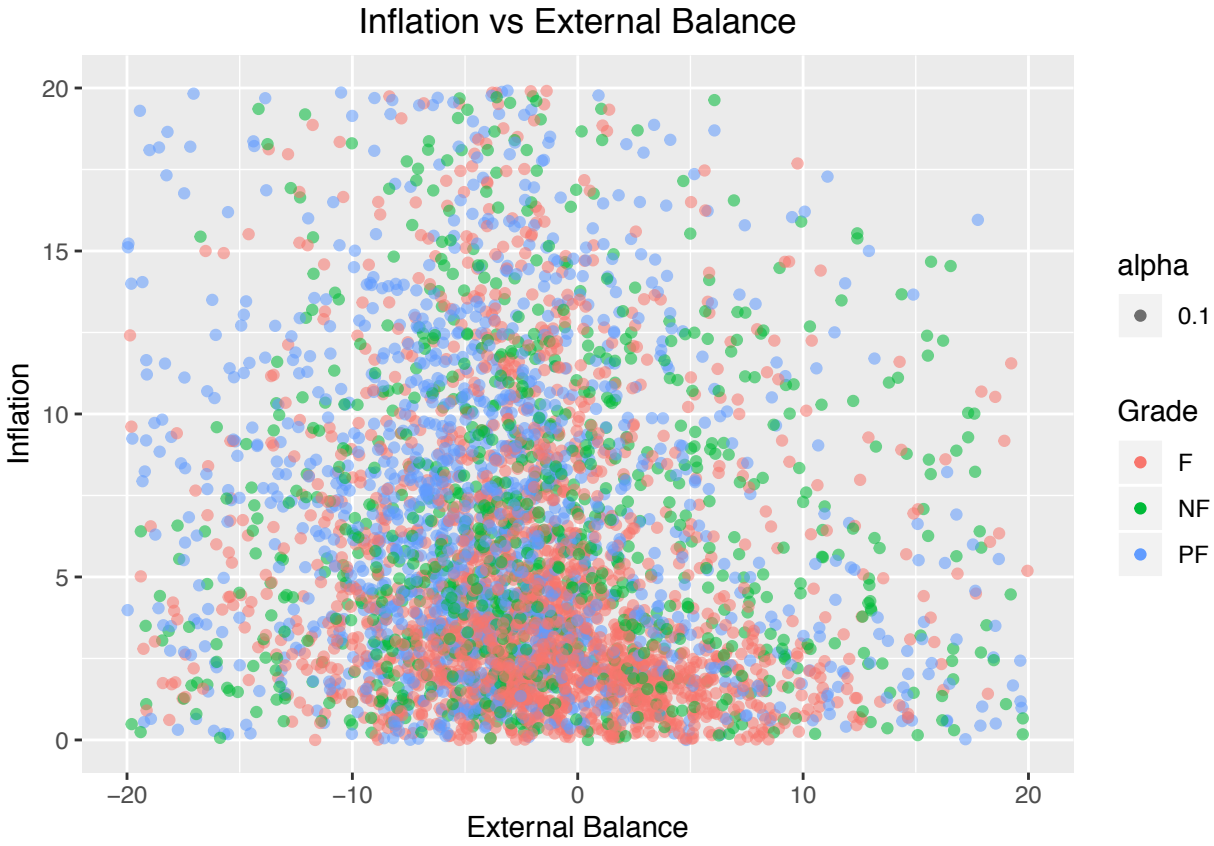
To ease the process of analysis without generating too many graphics, we are going to evaluate the variables by pairs. The first graphs will compare the government's primary deficits and government debts, two highly connected indicators in theory.



Nevertheless, this graph shows no apparent correlation between the two variables. The three types of political regimes spread evenly among both specters of debt and deficit. However, PF countries are apparently less spread than the other two.



A second scatter graph compares the revenues and expenditures of every country. In this case, a positive correlation between the variables is clear, as could be expected. It is also visible how the F countries tend to higher levels of both revenues and expenditures, followed by the PF, and lastly the NF. However, there is a significant amount of juxtaposition between the three. NF countries, in particular, are spread around all the spectrum.



The last comparison, one between the inflation and current account balance, confirms what could be seen in the boxplot graphs. Both indicators are too volatile to show a clear picture. However, some tenuous trends could be observed. Most of the countries tend to have a moderate current account deficit, but F countries are in general more centered on the equilibrium. PF countries have in general worse results in both inflation and current accounts than F ones, while NF countries are more dispersed across both spectrums.

After all the graphical analysis some conclusion could be extracted:

- Free countries have in general better macroeconomics results than the Free and Partially-Free countries. The latter two are more difficult to differentiate.
- Budget related indicators (revenue, expenditure, and deficits) are the more useful ones. Their results are less volatile, and the differences between all three types of countries are clearer.
- Inflation and current account deficits are too volatile to be particularly useful, even though they have to be taken into consideration anyway.
- Debt levels are in general more stables, but the differences in the behavior are not that clear.

3) Machine Learning Modeling

After proceeding with the graphical analysis we can obtain the final dataset. We will call it *Reduced Dataset*. This dataset preserves only the rows (country-year) containing all the selected variables. To get a balanced dataset if a single indicator (i.e inflation) is not available the entire row is discarded. The price to pay is that, as a consequence, the final dataset is significantly smaller than the previously studied, and many of their statistical properties may end up being modified.

```
Reduced_Data <- Consolidated_Data_spread %>%
rename("External" = `Current account balance`,
```

```
       "Inflation" = `Inflation, average consumer prices`,
    "Debt" = `General government net debt`,
    "Deficit" = `General government primary net lending/borrowing`,
    "Revenue" = `General government revenue`,
    "Expenditure" = `General government total expenditure`) %>%
drop_na("External","Debt","Deficit") %>%
  select(-Unity) %>% dplyr::filter(Grade != "")
Reduced_Data$Grade <- as.factor(as.character(Reduced_Data$Grade))
```

Now that the data final dataset is compiled, it is time to find out which machine learning (ML) model has a better performance at predicting the outcome variable. ML algorithms have the distinct ability to train themselves. They can detect patterns in the data, and automatically create models based on those patterns, with little or none human intervention.

Before running any kind of procedure concerning ML, the first step is to split the data between a *training set* (the one used to develop the algorithms) and a *test set* (the one used to test the performance of the algorithm). In this case, the train set will comprehend 80% of the *Reduced_Data* dataset proportion, and the remaining 20% will go to the test set.

```
set.seed(1)
test_index <- createDataPartition(Reduced_Data$Grade,times = 1, p = 0.2, list = FALSE)
train_set <- Reduced_Data[-test_index,]
test_set <- Reduced_Data[test_index,]
```

Among all the machine learning models available, three widely proven ones have been selected for the task: the **Classification Tree**, the **K-Nearest Neighbors** and the **Random Forest**. We will see how well each of them can carry out the task.

To execute those models, we will be using an R package called "caret". This package is designed to automatically apply a wide variety of ML algorithms over a dataset, creating models of predictions for any selected variable.

3.1) Classification Tree

The classification tree is one of the most popular ML models. It consists of a mapping of binary decisions, leading to determining the class of a certain object. Every "branch" of the tree raises a question about a particular characteristic of the object, and the "leaves" are two possible binary answers to that question. Every answer should lead to a new question being raised until enough questions are answered and it's possible to determine the true class of the object.

The "train" function of the caret package can automatically create a predictive model for every ML algorithm selected, based on the training dataset. Besides its autonomous nature, every ML algorithm has one or more parameters that should be optimized or tuned to obtain the best results. Fortunately, the train function of the package automatically optimizes the algorithm through a process called cross-validation. There is a default range of different values for the tunable parameter in every algorithm, and the package automatically chooses the value that shows the best accuracy.

```
set.seed(1)
train_rpart   <-  train(Grade ~ .,data = train_set, method = "rpart")
```

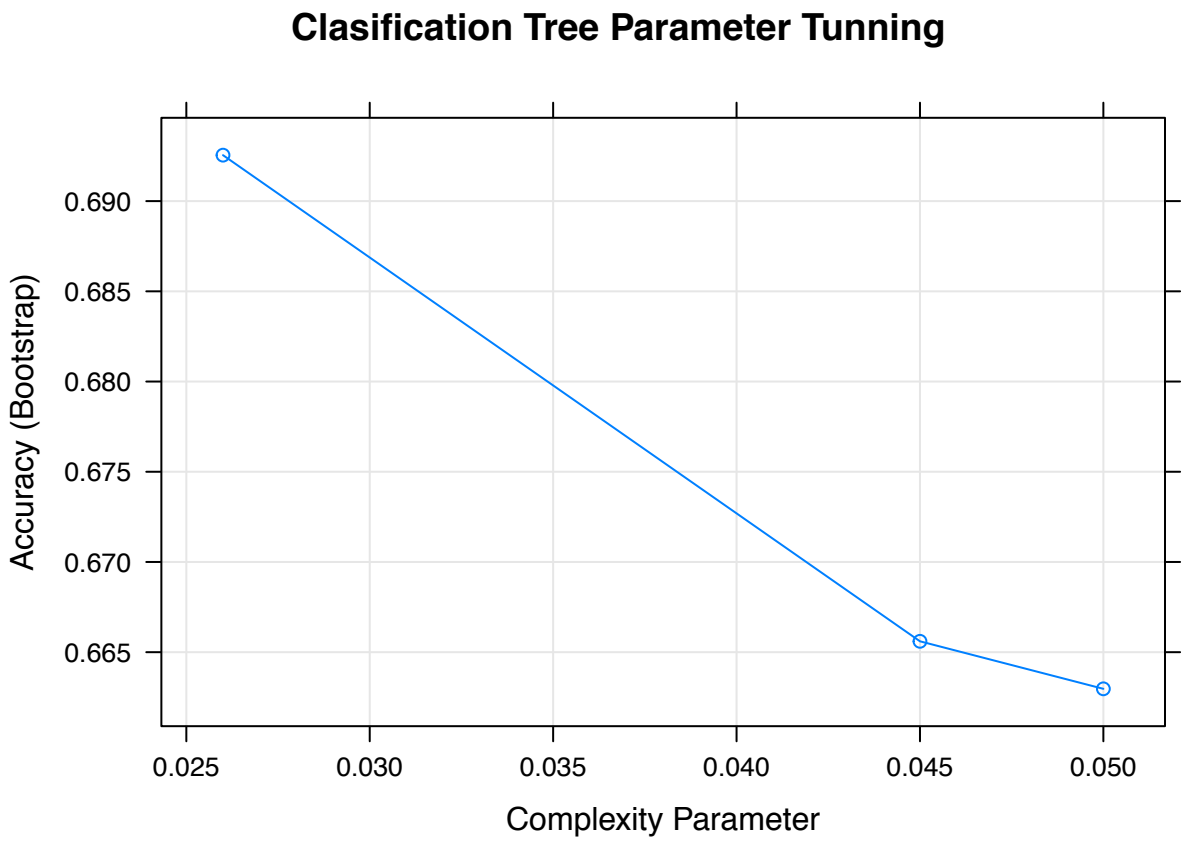
In this case, the parameter to be tuned is the *cp*, or *complexity parameter*. This parameter sets the minimal predictive improvement necessary to create a new branch in the tree. The default range for it is between 0.040 and 0.070.

However, sometimes the best possible tuning value for a particular parameter may be outside the default range. As a consequence, It's necessary to check if the best performance tuning value of a parameter is at the

extremes of the default range, which could be a sign of the existence of a better tuning value outside the range.

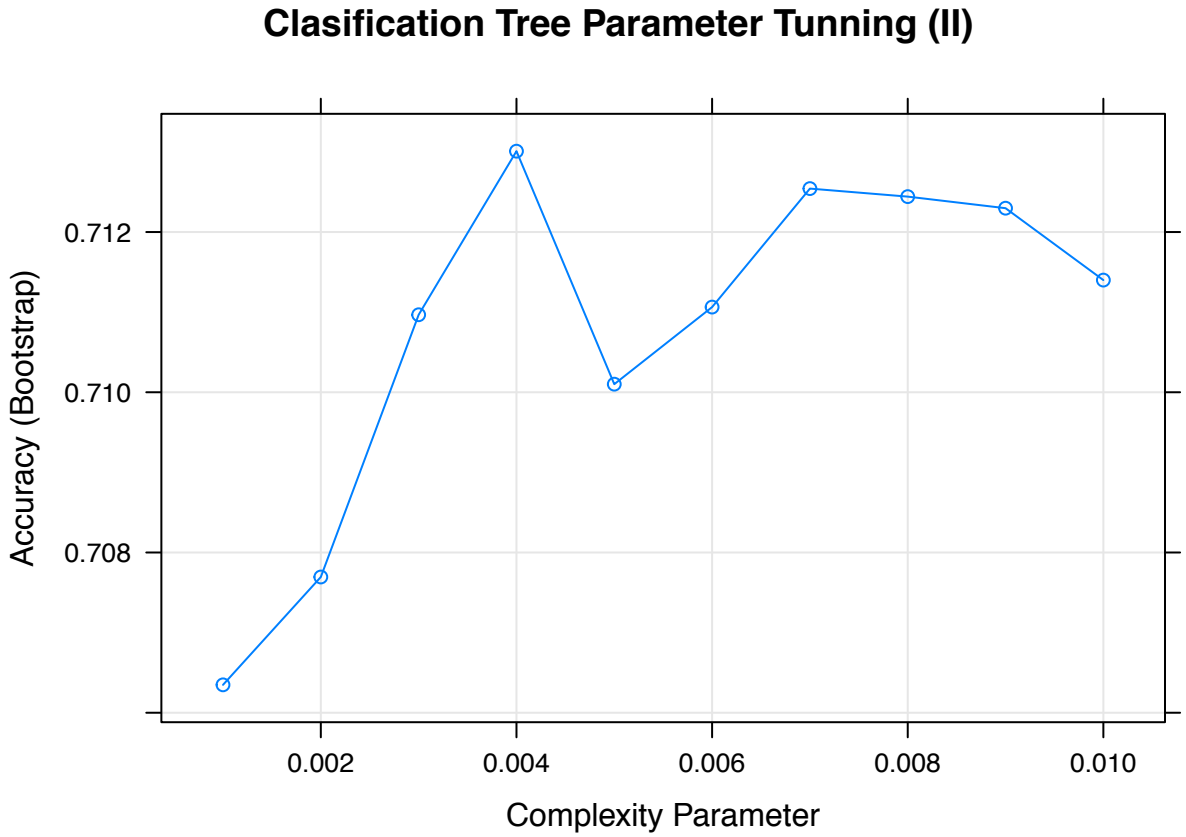
By plotting the output object resulting from every round of training, we can see the accuracy level from every level of the tuning parameter.

```
plot(train_rpart, main = "Clasification Tree Parameter Tunning")
```



In the case of our own regression, it is remarkable that the minimum value in the cp default range (0.040) is the one showing the best accuracy (0.685). It's possible that an even smaller cp value could result in an even better accuracy score. To evaluate this, we proceeded to repeat the training process, but changing the so-called *tune grid* parameter to evaluate the performance of values lower than 0.01

```
set.seed(1)
train_rpart_tuned   <-  train(Grade ~ .,data = train_set, method = "rpart",
                              tuneGrid = data.frame(cp = seq(0.001, 0.01, 0.001)))
plot(train_rpart_tuned, main = "Clasification Tree Parameter Tunning (II)")
```



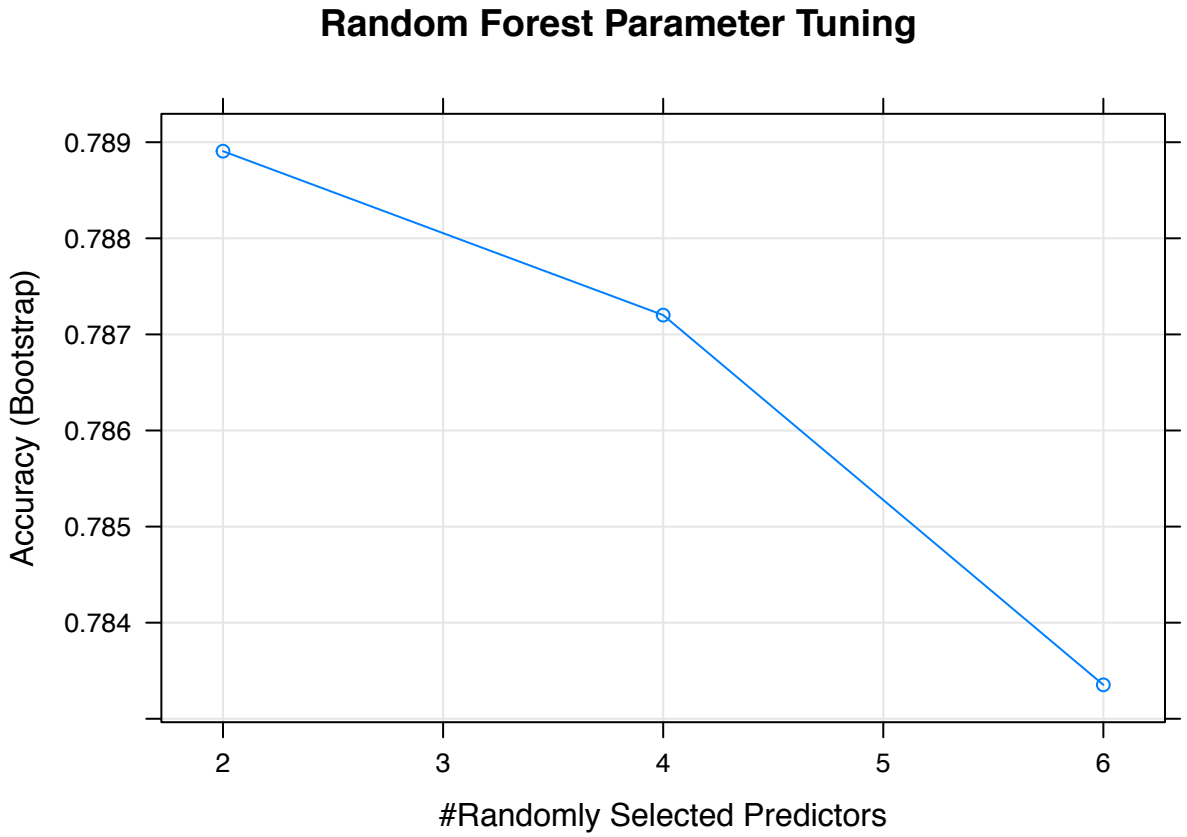
The new training round shows significantly better accuracy coming from a considerably lower cp value (0.005). This is the configuration we'll now use through the test dataset, to properly test its effectiveness at prediction.

3.2) Random Forest

Another frequently used algorithm is the so-called *Random Forest*. This algorithm creates a multitude of classification trees (a "forest") and ensembles their results into a single classification model. It holds several advantages over the single tree model; among them are a lower variance and more consistent results, at the expense of some loss interpretability and longer calculation times.

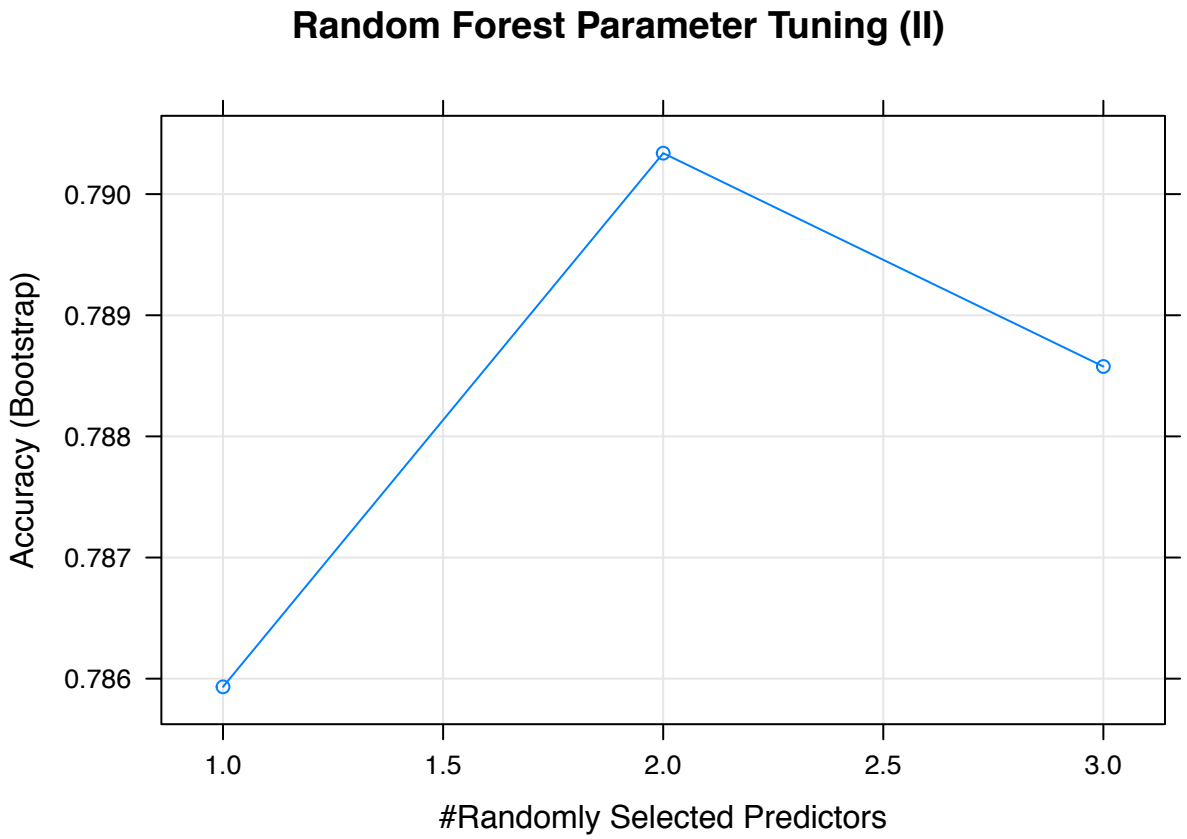
Like the previous method, the Random Forest can be applied and trained by the caret package, under the algorithm "rf", but has to pass through the same verification of tuning variables. In this case, the tuning variable is the number of variables randomly sampled as candidates at each split, called mtry.

```
set.seed(1)
train_rf   <-  train(Grade ~ ., data = train_set, method = "rf")
plot(train_rf, main = "Random Forest Parameter Tuning")
```



A first look at the trained model shows us maximization of performance at a low level of the predictor (2), which suggests the chance of a better performance by picking a single predictor. A new training round will be held just taking the values between 1 and 3 for the tuning variable.

```
set.seed(1)
train_rf_tuned   <- train(Grade ~ ., data = train_set, method = "rf",
                          tuneGrid = data.frame(mtry = seq(1, 3, 1)))
plot(train_rf_tuned, main = "Random Forest Parameter Tuning (II)")
```



Somehow contradicting the first estimation, the second optimization round finds the 3.0 as the optimal level for the mtry variable. To avoid an overlarge analysis process, that level will be selected for the final model.

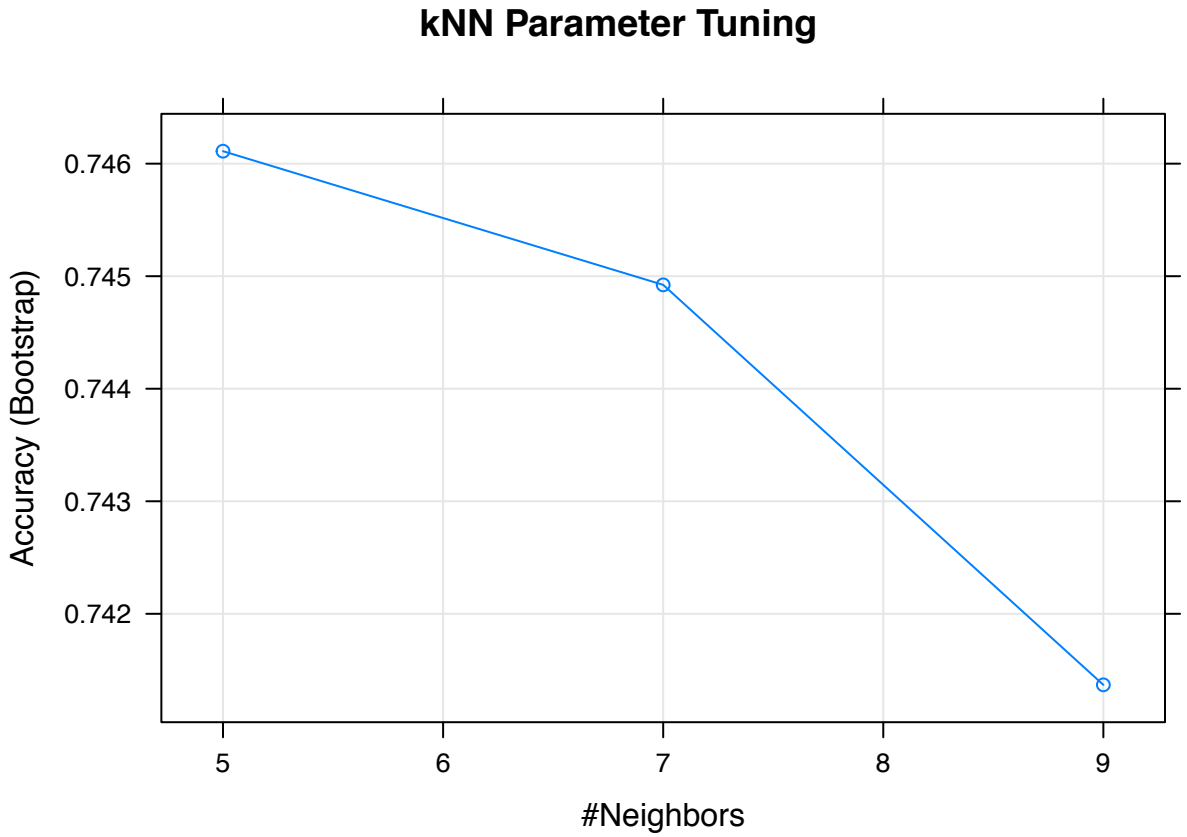
3.3) K- Nearest Neighbor (kNN)

One of the simplest and better-known ML models is the K - Nearest Neighbor (kNN). This model classifies the individuals based on the class of its closest "neighbors", or other individuals whose features are quantitatively similar. The smaller the neighborhood, the bigger weight would have on the final "vote" to classify the nature of the considered individual.

In this case, the algorithm should classify every country based on the nature of the countries with similar macro economical profiles.

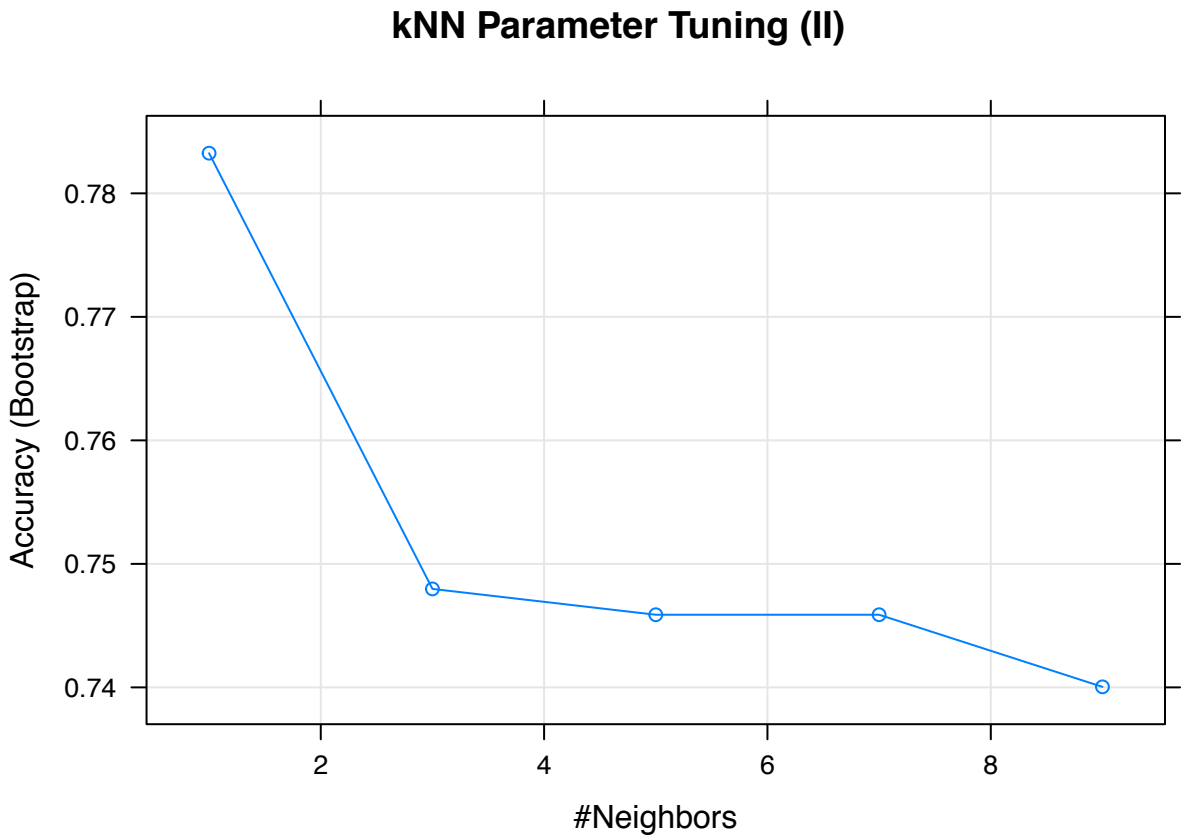
The main tuning parameter of kNN models is k, the number of neighbors considered to vote. A low K indicates that only the closest neighbors will be considered. Following the standard procedure, the optimal k will be automatically selected by the train function of the caret package.

```
set.seed(1)
train_knn <- train( Grade ~ ., data = train_set, method = "knn")
plot(train_knn, main = "kNN Parameter Tuning")
```



Just as in the previous cases, the preliminary results indicate better performance for the model at the lowest tuning values. Therefore, a new train round would have to be executed considering lower levels of k.

```
set.seed(1)
train_knn_tuned  <- train( Grade ~ ., data = train_set,
                           method = "knn",tuneGrid = data.frame(k = seq(1, 10, 2)))
plot(train_knn_tuned, main = "kNN Parameter Tuning (II)")
```



The new round of training indicates an optimal neighborhood of just one individual, continuing the trend of low optimal tuning values. The overall accuracy of this third model (0.81) is even better than the second one (0.79), but only by a small margin.

4) Results

`Disclaimer: Due to unknown reasons, the final results given by knirt could be different from the ones given by the RMD and R codes. The text in the pdf document will be based on the latter.`

4.1) Clasification Tree

```
predict_rpart <-  predict(train_rpart_tuned, test_set, type = "raw" )
Results_rpart <-  confusionMatrix(predict_rpart,test_set$Grade)
Overall_rpart <-  Results_rpart$overall[["Accuracy"]]
Table_rpart   <-  Results_rpart$table
Overall_rpart
```

```
## [1] 0.7506849
```

The overall accuracy of this free test was 0.72602. A little more than double the expected level of accuracy of pure chance in a situation of three possible outputs. It's the first benchmark to compare against the other models.

```
grid.arrange(grobs = list(tableGrob(round(Results_rpart$table, digits = 3))))
```

	F	NF	PF
F	207	18	23
NF	11	21	6
PF	21	12	46

The adjacent table is the *Confusion Matrix*. It tabulates every combination of prediction (vertical axis) and the actual value (horizontal axis). For instance, the three numbers of the first column (204, 16 and 19) indicate that of the 239 cases classified as F in the original dataset, the model predicted as such a total of 204, while 16 cases were (wrongly) predicted as NF and 19 as PF.

Those numbers mean that the *sensitivity* of the model is 85.35% (204/239) for free countries. The sensitivity indicates the proportion of individuals in a given class that are actually predicted as such by the model.

On the flip side, the first row of the table indicates a total of 16 NF cases and 31 PF cases wrongly predicted as F by the model. In this case, the relevant indicator is the *specificity*, or the capacity of the model to **not** predict an individual as a member of a class it does not belong to. It is measured as the ratio between the number of cases rightly predicted as negatives (true negatives) and the sum of those true negatives and the false positives. In this case, the specificity for F countries is just 0.6269.

```
confusion_rpart <- Results_rpart$byClass[,c("Sensitivity","Specificity","Balanced Accuracy")]
grid.arrange(grobs =list(tableGrob(round(confusion_rpart, digits = 3))))
```

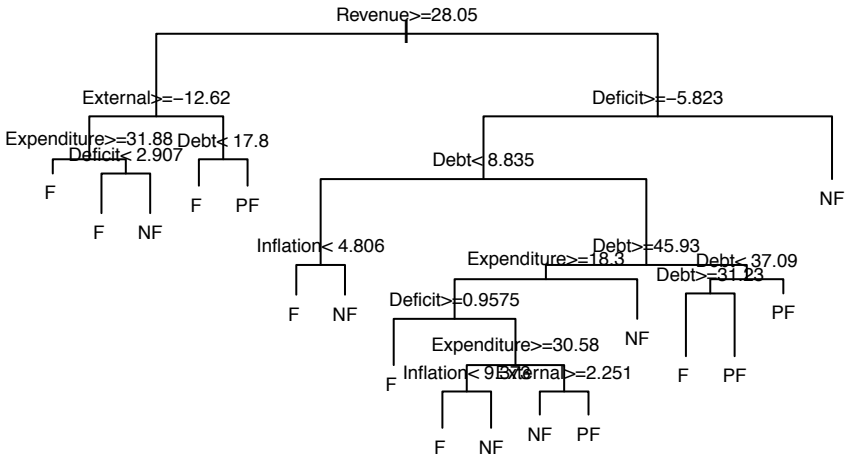
	Sensitivity	Specificity	Balanced Accuracy
Class: F	0.866	0.675	0.77
Class: NF	0.412	0.946	0.679
Class: PF	0.613	0.886	0.75

This first model clearly shows a bias towards predicting F cases, probably motivated by the high proportion of this kind of countries in the sample. Only half of the NF and PF cases are predicted as such.

However, one of the main advantages of the classification tree method is its high interpretability. It's possible to graphically reflect the actual logical process of selection in a treemap. Even though they are usually difficult to intrepretative, it is possible to extract some important highlights from them.

```
set.seed(1)
fit_rpart <- rpart(Grade ~ ., data = train_set)
plot(fit_rpart, margin = 0.1, main = "Grafical Representation of the Clasification Tree")
text(fit_rpart, cex = 0.55)
```

Grafical Representation of the Clasification Tree



4.2) Random Forest

```
set.seed(1)
predict_rf <-  predict(train_rf_tuned, test_set, type = "raw" )
Results_rf <- confusionMatrix(predict_rf,test_set$Grade)
Table_rf   <-  Results_rf$table
Overall_rf <- Results_rf$overall[["Accuracy"]]
Overall_rf
```

```
## [1] 0.8520548
```

The Random Forest model shows an important improvement over the previous method. The 0.72 accuracies climbed to 0.7917.

```
grid.arrange(grobs = list(Table_rf,Derivatives_rf),
             top = text_grob("Random Forest Confusion Matrix", size = 15))
```

Random Forest Confusion Matrix

	F	NF	PF
F	226	8	20
NF	5	32	2
PF	8	11	53

	Sensitivity	Specificity	Balanced Accuracy
Class: F	0.946	0.778	0.862
Class: NF	0.627	0.978	0.803
Class: PF	0.707	0.934	0.821

The confusion matrix also shows an improvement in PF sensitivity from 0.46 to 0.61, even though the NF sensitivity remains almost the same. The F specificity also improves from 0.62 to 0.71. The new model has not only a better overall result but a more balanced performance among the three classes.

4.3) K- Nearest Neighbor (kNN)

```
set.seed(1)
predict_knn <-  predict(train_knn_tuned, test_set, type = "raw" )
Results_knn <-  confusionMatrix(predict_knn,test_set$Grade)
Table_knn   <-  Results_knn$table
Overall_knn <- Results_knn$overall[["Accuracy"]]
Overall_knn
```

```
## [1] 0.8383562
```

The overall accuracy improved slightly over the last model, from 0.7917 to 0.8109. Even though is the best result, the rate of improvement is smaller.

```
grid.arrange(grobs = list(Table_knn,Derivatives_knn),
             top = text_grob("kNN Confusion Matrix", size = 15))
```

kNN Confusion Matrix

	F	NF	PF
F	220	10	13
NF	10	28	4
PF	9	13	58

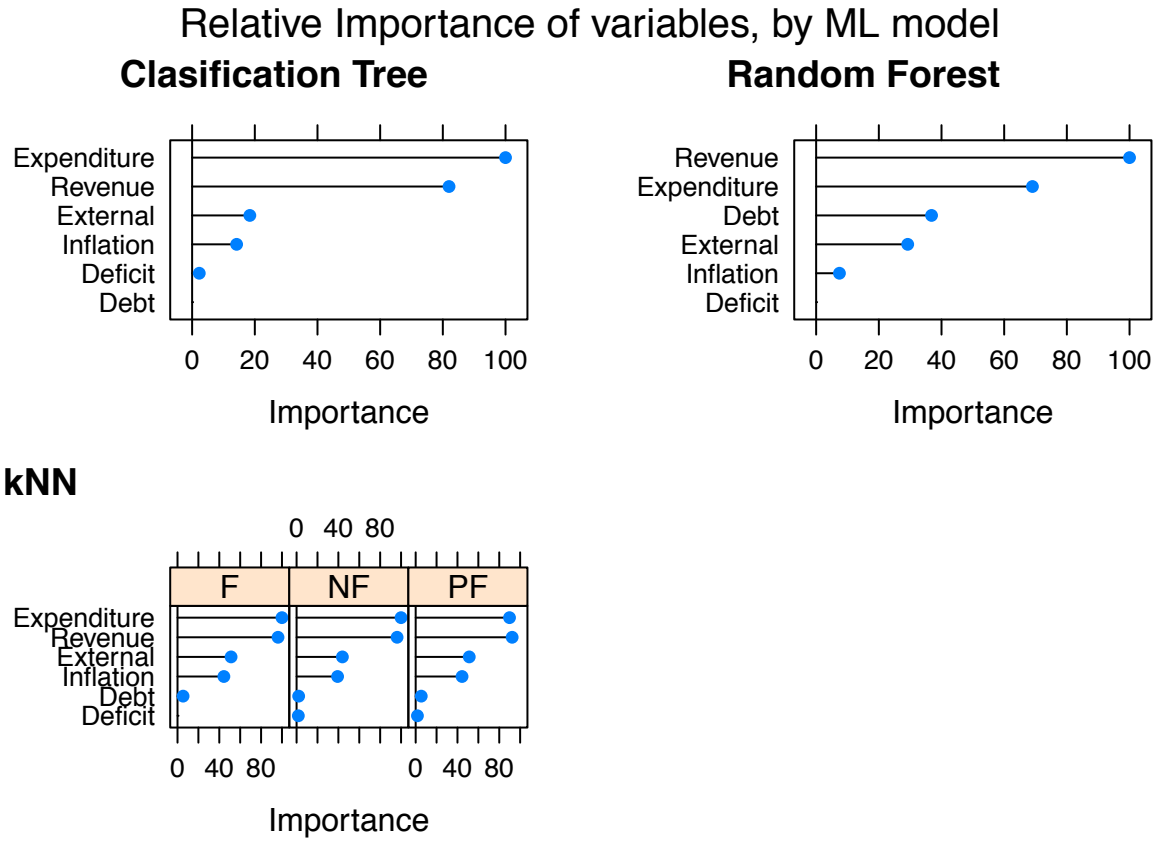
	Sensitivity	Specificity
Class: F	0.921	0.817
Class: NF	0.549	0.955
Class: PF	0.773	0.924

The confusion matrix of this third model shows no significant improvement over the second. The sensitivity and specificity for the three classes are quite similar in both cases. Given that both methodologies are essentially different, we could be reaching the limits of the chosen reduced dataset's predictive value.

4.4) Effective relevance of variables

Once the models are set, there is still some new information that can be extracted from the trained algorithm. One example is to measure the relevance of each variable in every model. The caret function *VarImp* allows us to find out.

```
var_rpart <- plot(varImp(train_rpart_tuned))
var_knn   <- plot(varImp(train_knn_tuned))
var_rf    <- plot(varImp(train_rf_tuned))
Variable_chart <- ggarrange(var_rpart,var_rf,var_knn,
  labels = c("Clasification Tree","Random Forest","kNN"), hjust = -0.5)
annotate_figure(Variable_chart,
  top = text_grob("Relative Importance of variables, by ML model", size = 15))
```



After measuring the variable importance in every one of the three models, some conclusions could be taken:

- Expenditures and Revenues are the most important and consistent variables in the model.
- Fiscal Balance, Government Debt and Inflation are the least important variables.

These results greatly coincide with the conclusions emanating from the graphical analysis. Given the consistency of the results, some final conclusions could be made.

5) Conclusion

This work is the first approach to a topic that is both theoretically and technically quite complex, with somehow mixed results. The best overall accuracy could barely pass the 80% threshold. It's not a bad result, considering that, with three output categories, the guessing rate approximately 33%. But they are still far from being reliable models.

Nevertheless, some important key conclusions could be extracted from the whole analysis process:

- In general, free countries showed a better economic performance and a more consistent behavior than the non-free or semi-free countries, even though all categories demonstrated high variability in their results. The idea that democracies have better economic performance still holds.
- The proportion of government's revenues and expenditures are clearly the best indicators to predict the degree of freedom of a society. Free countries tend to have higher rates of both indicators over GDP, suggesting that the citizens of these countries are more willing to pay taxes and the government more interest in providing public goods and services.
- Current account balances also showed some relevance, even though way lower than the previously mentioned variables.

- On the other hand, inflation, fiscal deficits, and debt levels demonstrated to have very weak predictive measures in practice, even though fiscal deficits are clearly related to the two most important variables (Revenue and Expenditures).
- Non-free countries are the most difficult to predict group. Their economic behavior is apparently the most variable, at least when comparing to the other two.

There are also a few important recommendations to apply in subsequent iterations

- Due to the need for a completely balanced dataset, a different selection of variables could lead to a different sample of data. Some of the least important variables of this model dropped, in exchange for a broader dataset to work with.
- There is a wide range of other ML algorithms to apply; now we are just applying three of most frequently used.
- Different training criteria could compensate for the over-representation of free countries in the data, leading to more balanced results.