CENTRALESUPELEC

Introduction to Machine Learning

**FINAL REPORT**

*Authors* : *Professor :*

Aurélien Pasteau Fragkiskos Malliaros

Nicolas Gabrion

Marion Gobet

Vincent Bouget



**Abstract**

This report presents an analysis and a comparison of four different algorithms (linear regression, tree and AdaBoost algorithm, neural network and nearest neighbours) which aim is to predict the results of French presidential elections. We used large databases from the French government, giving results of the 2017 elections per candidate and socio-economic data on a local scale, and pre-processed it to select the most pertinent features.

In this report, algorithms are compared so as to find the most performant one. Their strengths and weaknesses are spotlighted, and provide all the elements to decide which method is the most appropriated to predict the results of the election in 2022, and to quantify its reliability.

**Introduction/Motivation**

Our project aims to find a reliable way to predict the results of each candidate to the French presidential elections on a local scale. In fact, precise election results prediction has always been a challenge to solve because of the advantages it can provide to someone achieving it. Also, there are many organisations trying to predict those results by going ask people who they want to vote for, but this only works a short time before the election and it requires to go out and collect data from people one by one, which isn’t efficient and can take a long time.

Here, the goal is to only use more general data about the socio-economic and demographic situation on a local scale, which doesn’t require a long time of data collection since all this data can be found online and is free to use.

With this project, we want to answer the following question: is it possible to obtain a reliable prediction of each candidate’s proportion of votes locally by working with data engineering methods and then applying machine learning algorithms to this open source socio-economic and demographic data?

If the answer to that question is true, one could imagine lots of applications to this tool : one could sell candidates information about their future results in the places where they are not sure whether they are going to win or not, or one could even bet on ranges in which the candidate’s results are going to be and earn money with it. More simply, one could also provide the people with predictive information about the election.

**Problem Definition**

At first, we aimed to predict the result of 2022 presidential election based on the results of the 5 previous election. Indeed, we decided to predict the result of every candidate in every city/town. But as we started working on the project, we realized that our idea was definitely unrealistic because the candidates are different in every election so that was nearly impossible to predict scores as we had to define a political opinion for every candidates in order to know if he is from right or left etc…

Given these thoughts, we decided to focus on the 1st turn of 2017 election and to try to predict the score of every candidate based on socio-economic data. More precisely, we believe that the results are linked to the economic and social situation and that is the relation we are trying to show. In terms geographical splitting, election scores are given per cantons while social and economic data is given by cities/towns and that’s why we have to aggregate data to get the same splitting for data and results.

The goal of this project it to be able to predict the results of some cantons given their social and economic data with algorithms trained on the other part of cantons. However, the ultimate goal could be the prediction of 2022 election. Indeed, if we could collect data for years 2021 and 2022 before 2022 election, we could train our algorithms on 2017 election and then predict the results for 2022.

**Related Work**

[VINCENT]

**Methodology**

1. **Data Collection**

The work first started with the data collection. All the data we used can be found on internet and is free to use (see the references at the end of the report). First, we collected some data about the first turn of the 2017 election[1] with each candidate’s result in every “canton electoral”, which is basically a big city or a zone grouping several villages and towns. We also needed some socio-economic and demographic data, which is why we collected two datasets[2],[3] containing a large number of such data for every town in France. Finally, we also needed to know what “canton” each town belonged to, so we also found a dataset about that[4].

All the data engineering code can be found in the python file *“data\_engineering.py”.*

The first thing we needed to do was to separate the election data concerning each candidate, because at the end, we wanted to train models separately for each candidate. We only chose to work with the main 5 candidates of the 2017 election, because they represent the main 5 political parties or movements in France. At the end of this step, we had 5 datasets (one for each candidate) containing for each canton (described by the ‘Département’ and ‘Code Canton’ fields) the proportion of votes for the candidate during the first turn of the 2017 election.

Unfortunately, the socio-economic and demographic data we had wasn’t gathered in “cantons” but in towns, which is why we had to aggregate all those features into “cantons”. First, we needed to know to which cantons all the towns belonged, so we used a dataset giving us that information. Then we could aggregate the two features datasets into “cantons”, looking precisely for each feature if we had to take the sum, the average or the maximum of the values from the towns inside the “canton”. We decided to fill the NaN values with the average value of the same column’s values, and to normalize between 0 and 1 all the features’ values. Finally, we could join these 2 “canton”-aggregated features datasets to the five candidates’ datasets in order to have five ready-to-use datasets.

Note that in all the above described process, we had several problems because all the datasets don’t describe “cantons” or towns or “Départements” the same way. To make our lifes easier, we used pandas[5], a library allowing to manipulate data in an easier way.

1. **Feature Selection**

In the final 5 datasets, we had approximately 100 features. If we kept them all, our models would overfit, so we had to use a feature selection method. We decided to use a linear filter approach called f\_regression[6]. In *data\_engineering.py*, we coded a function that, given an int k, finds a list with the union of the k best features for each candidate according to that method (we take the union in order to have the same features for every candidate) and returns the 5 datasets with only those features.

The conclusion of all this data\_engineering process is that it allows us to export 5 nice and proper csv files, one for each candidate, with the best selected features. Of course, one can change the number of features one wants to select.

1. **Linear regression**

[MARION]

1. **Neural network**

[NICO]

1. **Nearest neighbours**

[MARION]

1. **Tree and AdaBoost algorithms**

The tree algorithm is not the most relevant algorithm to solve our problem, however it is simple to implement for a bidimensional issue. Thus, it is possible to apply it to solve the question “let , will the percentage of people voting for candidate C in a given town be bigger or smaller than n”.

Obviously, the accuracy heavily depends on the n number (it’s not difficult to predict that less than 100% of the town will vote for the same candidate but far more difficult when n=0.2 for instance!). Indeed, when we apply the precedent algorithm for different values of n, we obtain the following curb with accuracy in function of n:

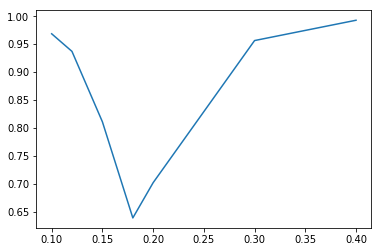


Figure 1: Accuracy=f(n)

Thus, it seems that the critical value of n which is the most difficult value to predict is around 0.18.

Tree algorithm is a weak learner (slightly better than random guessing) thus we decided to implement an AdaBoost algorithm to improve it. As we can see on the following curb, test error is decreasing when over the number of trees used. Thanks to this algorithm we have a first insight of possible results.

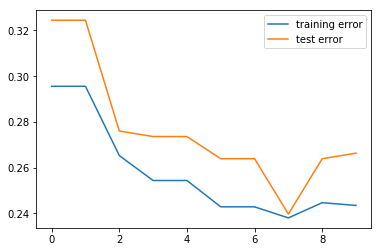


Figure 2: Training and test error for 10 estimators

**Evaluation**

**[explain the lack of precision with the fact that our data doesn’t take into account things like the Macron movement that came as a new and young alternative to the binary system we had before…..]**

**Conclusion**

**[Talk about how we could use predictions of the features values to predict the results in 2022]**

**References**

## [1]: Ministère de l’intérieur. Election présidentielle des 23 avril et 7 mai 2017 - Résultats définitifs du 1er tour. <https://www.data.gouv.fr/fr/datasets/election-presidentielle-des-23-avril-et-7-mai-2017-resultats-definitifs-du-1er-tour-1/>. Visited December 19 2018

**[2]: Stéphane Chauvin. Data INSEE sur les communes.** [**https://www.data.gouv.fr/fr/datasets/data-insee-sur-les-communes/**](https://www.data.gouv.fr/fr/datasets/data-insee-sur-les-communes/)**. Visited December 21 2018**

**[3]: INSEE. Comparateur de territoire.** [**https://www.insee.fr/fr/statistiques/1405599?geo=EPCI-249300088**](https://www.insee.fr/fr/statistiques/1405599?geo=EPCI-249300088)**. Visited December 21 2018**

**[4]: Ministère de l’intérieur. Circonscriptions législatives : Table de correspondance des communes et des cantons pour les élections législatives de 2012 et sa mise à jour pour les élections législatives 2017.** [**https://www.data.gouv.fr/fr/datasets/circonscriptions-legislatives-table-de-correspondance-des-communes-et-des-cantons-pour-les-elections-legislatives-de-2012-et-sa-mise-a-jour-pour-les-elections-legislatives-2017/**](https://www.data.gouv.fr/fr/datasets/circonscriptions-legislatives-table-de-correspondance-des-communes-et-des-cantons-pour-les-elections-legislatives-de-2012-et-sa-mise-a-jour-pour-les-elections-legislatives-2017/)**. Visited December 22 2018**

**[5]: Pandas. Python Data Analysis Library.** [**https://pandas.pydata.org/**](https://pandas.pydata.org/)**. Visited December 19 2018**

**[6]: Scikit Learn. Feature Selection.** [**https://scikit-learn.org/stable/modules/feature\_selection.html**](https://scikit-learn.org/stable/modules/feature_selection.html)**. Visited December 28 2018**