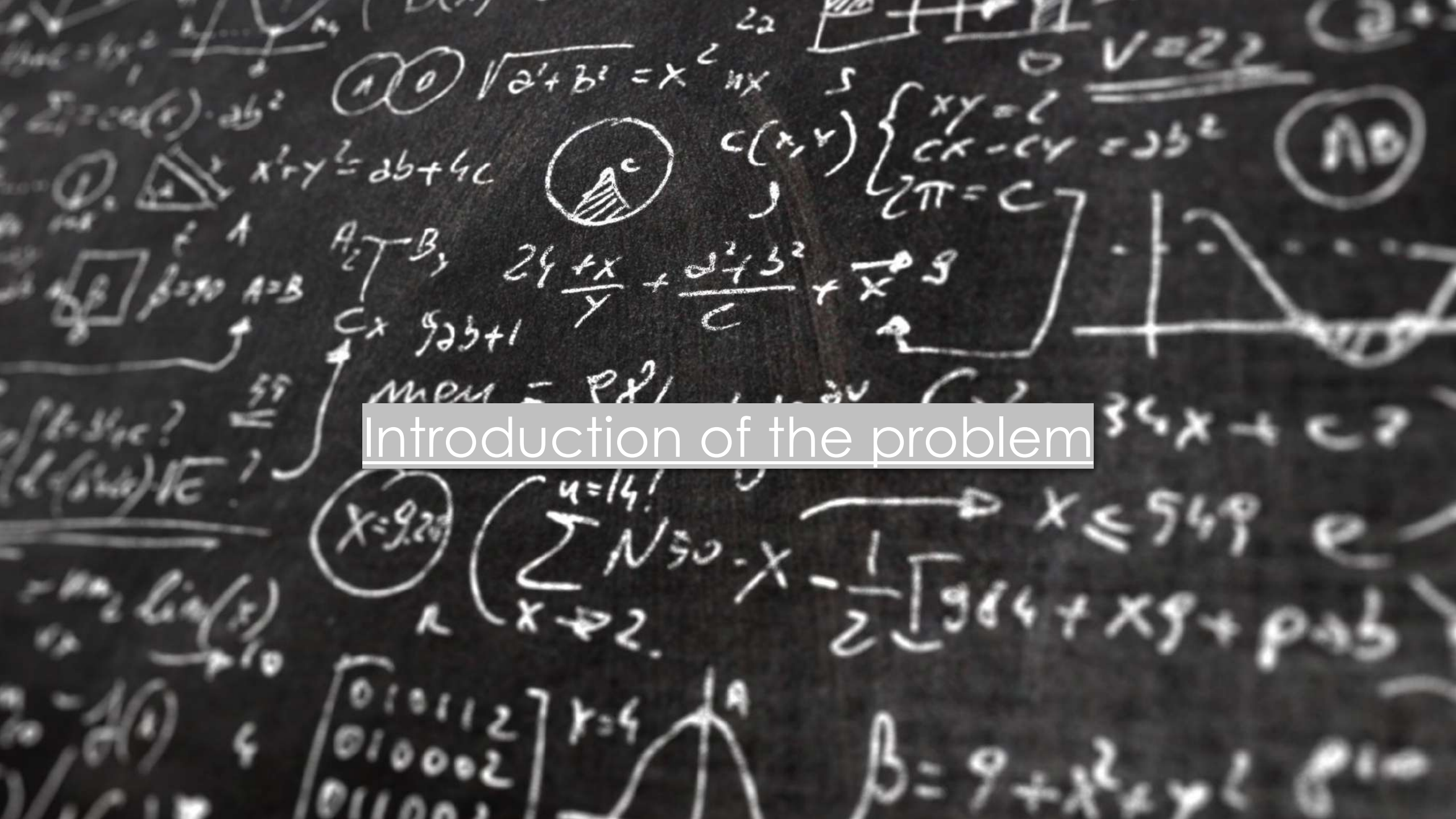
python *for data analysis*

Bilal Selmi & Yanis Fadili – DIA5

Table of contents

You can click on the icons to access directly to the section you want.



Introduction of the problem

The dataset 📧

- For this data science project we used the **spam dataset.**
- This dataset contains informations about a big number of **received e-mails**, and we're going to analyze it in order to make predictions on it to determine if a mail is a spam or not.
- The dataset contains **4601 rows and 58 columns.**

Data exploration 🔍

```
Entrée [9]: data = pd.read_csv(r"C:\Users\SELMI Bilal\Desktop\Cours (7)\Python for data analysis\spambase.data",header=None)

Entrée [10]: data.columns = a

Entrée [11]: data

Out[11]:
```

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet	word_freq_order
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31
...	...	...	...	...	...	...	...	...	...
4596	0.31	0.00	0.62	0.0	0.00	0.31	0.00	0.00	0.00
4597	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00
4598	0.30	0.00	0.30	0.0	0.00	0.00	0.00	0.00	0.00
4599	0.96	0.00	0.00	0.0	0.32	0.00	0.00	0.00	0.00
4600	0.00	0.00	0.65	0.0	0.00	0.00	0.00	0.00	0.00

4601 rows × 58 columns

- So, we began to ask us relevant questions about the dataset in order to understand how it could help us to answer to the problem. To do this, a rigorous data exploration after the importation had to be done.
- First of all, as the dataset is a .csv file, we had to import it using the read_csv/read_table commands, and see if the data was « clean ». In our case, the table was clean, so we were able to work on it without problem.
- Then, we did a very important data exploration has been carried out. It represents the first part of the data analysis, before the visualisation. So, we printed a part of the complete table in the variable *data* to understand the data, we checked the columns' types stocked in the variable *names*, and we tried to see which columns were relevant to prepare the prediction part.

→ At first view we saw that the frequencies of some words, some adresses and some other fields could have an influence on the relevance of an e-mail, and consequently if it's a spam or not. But we deepened the analysis with a datavisualisation to see exactly which frequencies are the most influent in the dataset.

Data visualization

Importing librairies

```
Entrée [1]: import pandas as pd
            import numpy as np
            import seaborn as sns
            import matplotlib as plt
            import matplotlib.pyplot as plt
```

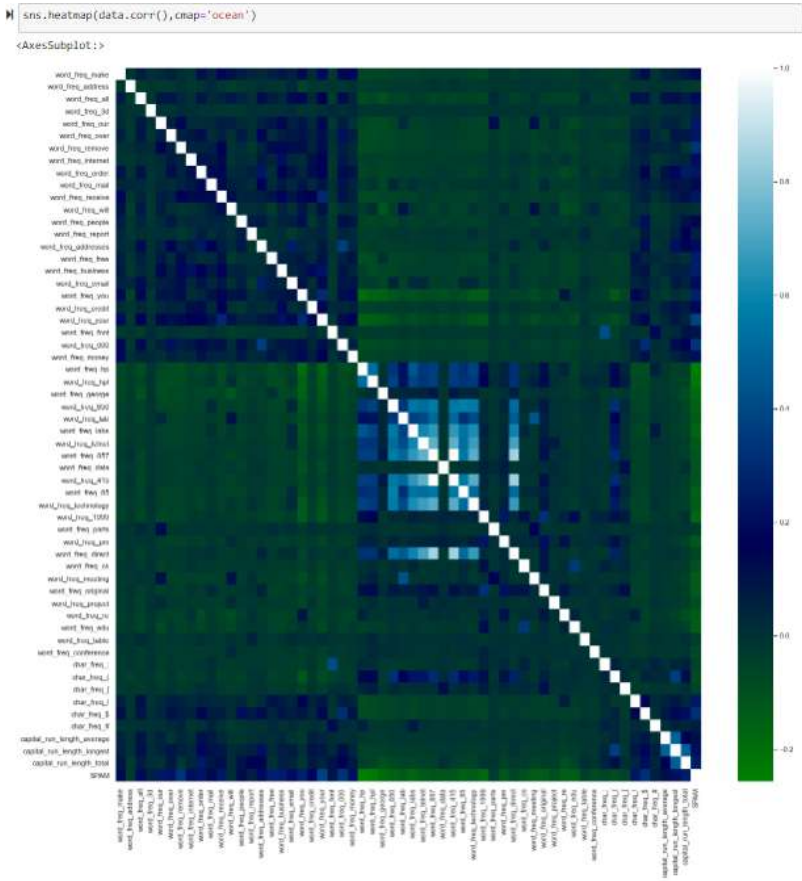
For the second part of the data analysis, we did a visulization of miscellaneous variables in or order to emphasize various aspects of our data set – by showing different plots, charts, and graphs.

Beforehand, it was necessary to import the libraries **matplotlib**, **pandas** & **seaborn.** We imported them at the first cell of the notebook.

First look : the heatmap 🔥

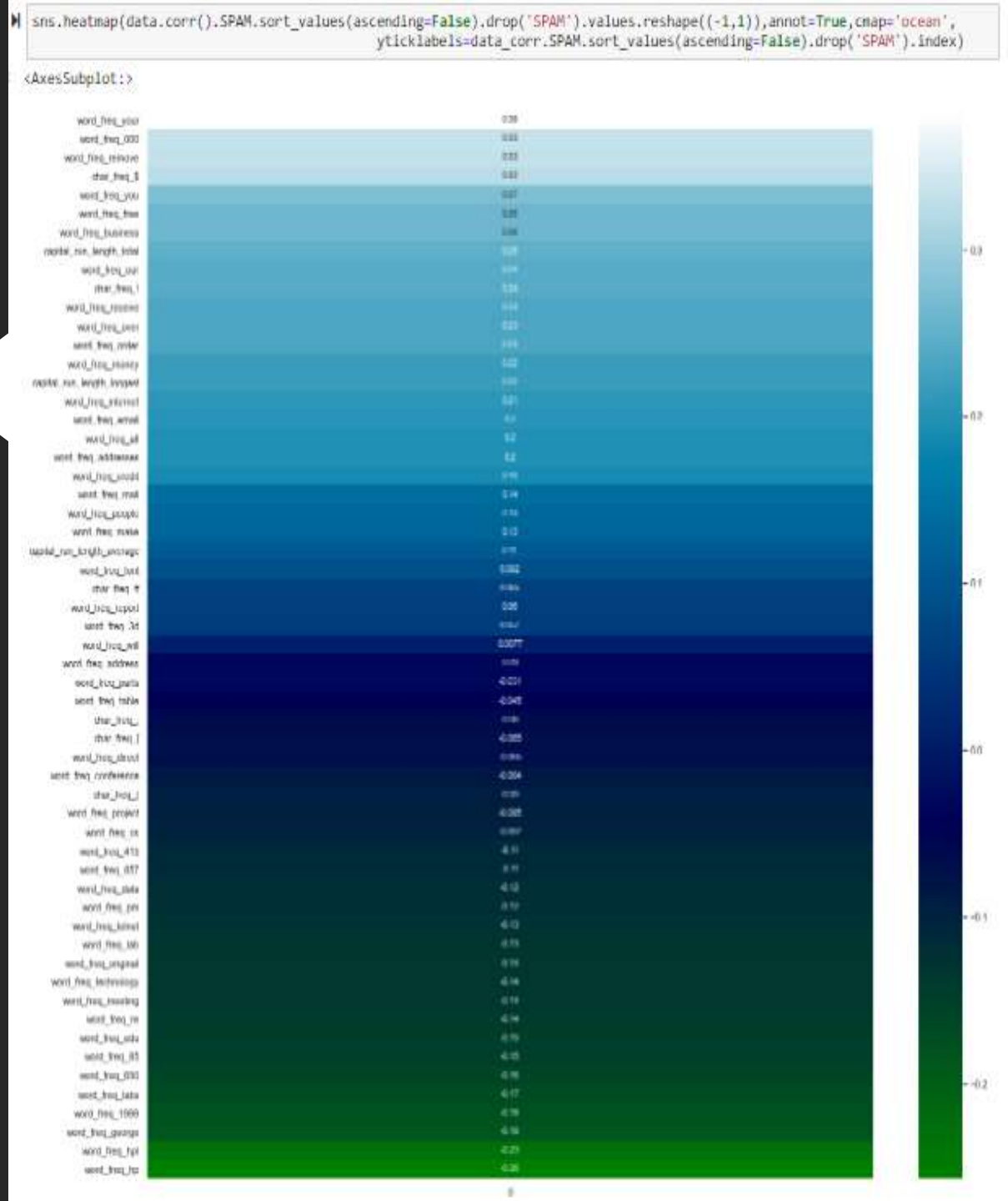
The heatmap is very interessant visualization to have a general idea of your variables and the correlation between them. By seeing this plot we can directly know which links are strong between 2 variables according to the color. Take a look →

Here we can that the correlation coefficient of the variables linked by a blue point is very high (about 1). So we can assume that these variables are enough proportional.



Correlation of the output (spam or not) 📲

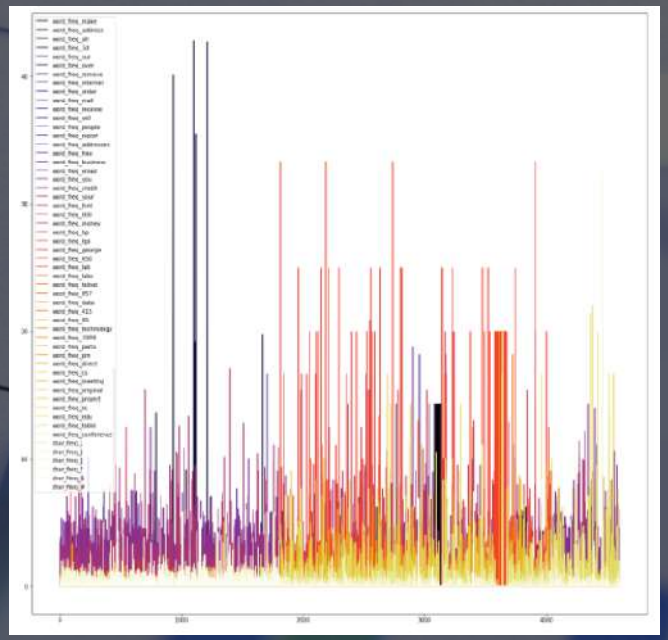
- We plotted the correlation plot at the right, because it is more interpretable than the previous which represent the correlations of all variables although this one represent the correlation between all variables and just the target, which is if the mail is spam or not.
- This plot will be very useful to model our data in order to do the data modelization.

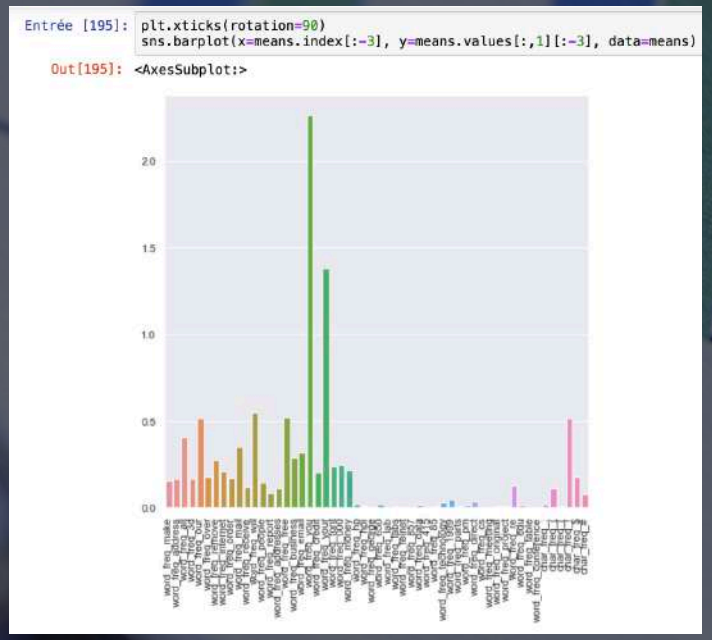


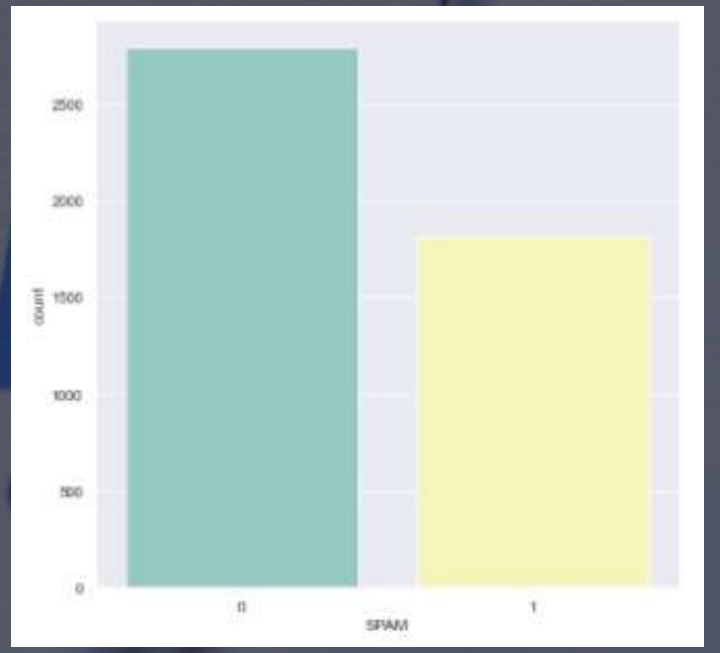
Bar charts

Bar charts (or "sticks" in some cases) are very efficient for the **quantitative** variables, so plot a few of them to see have a better idea of some variables.

Here you can see the representation of the mean value of each variable, the proportion of spam for each variable and the count of spams.



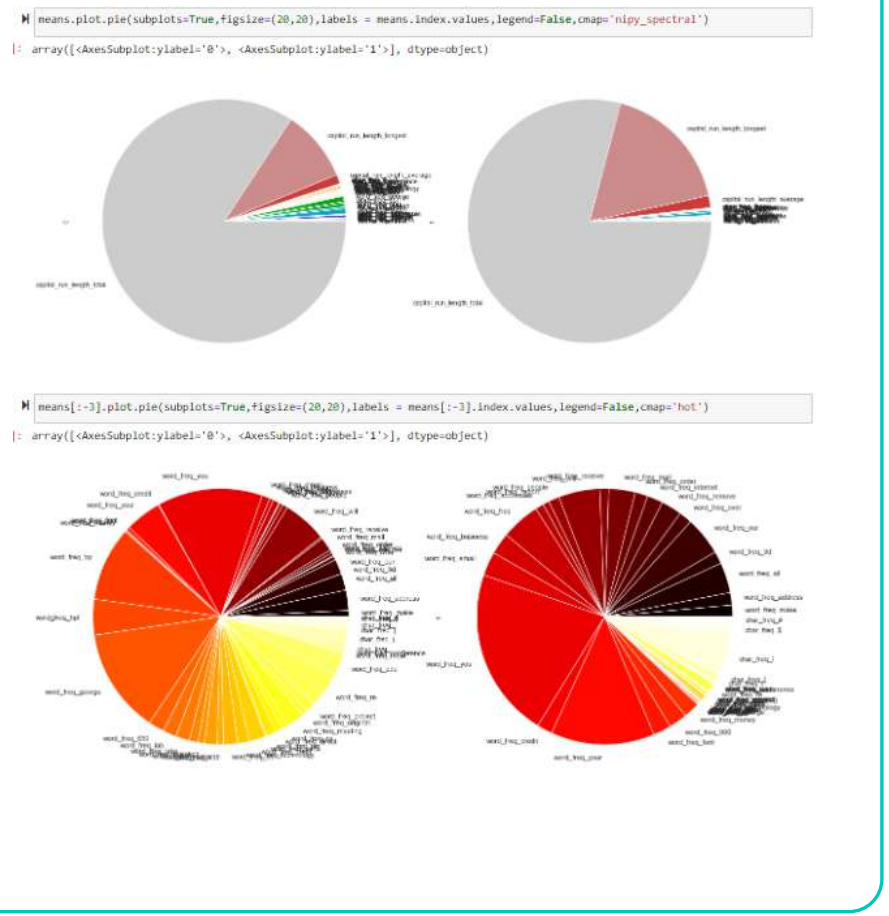


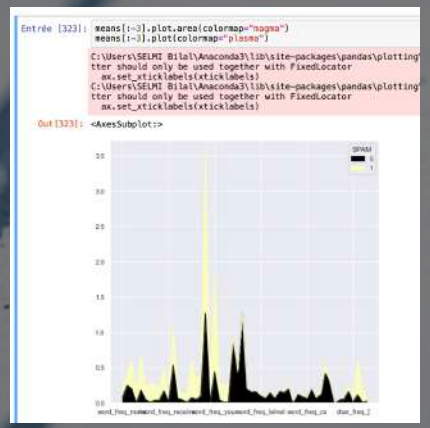


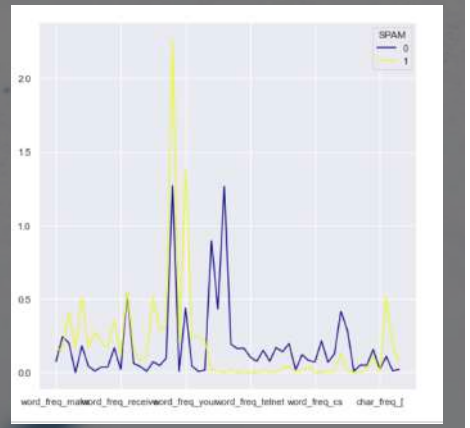
Pie charts 🍰

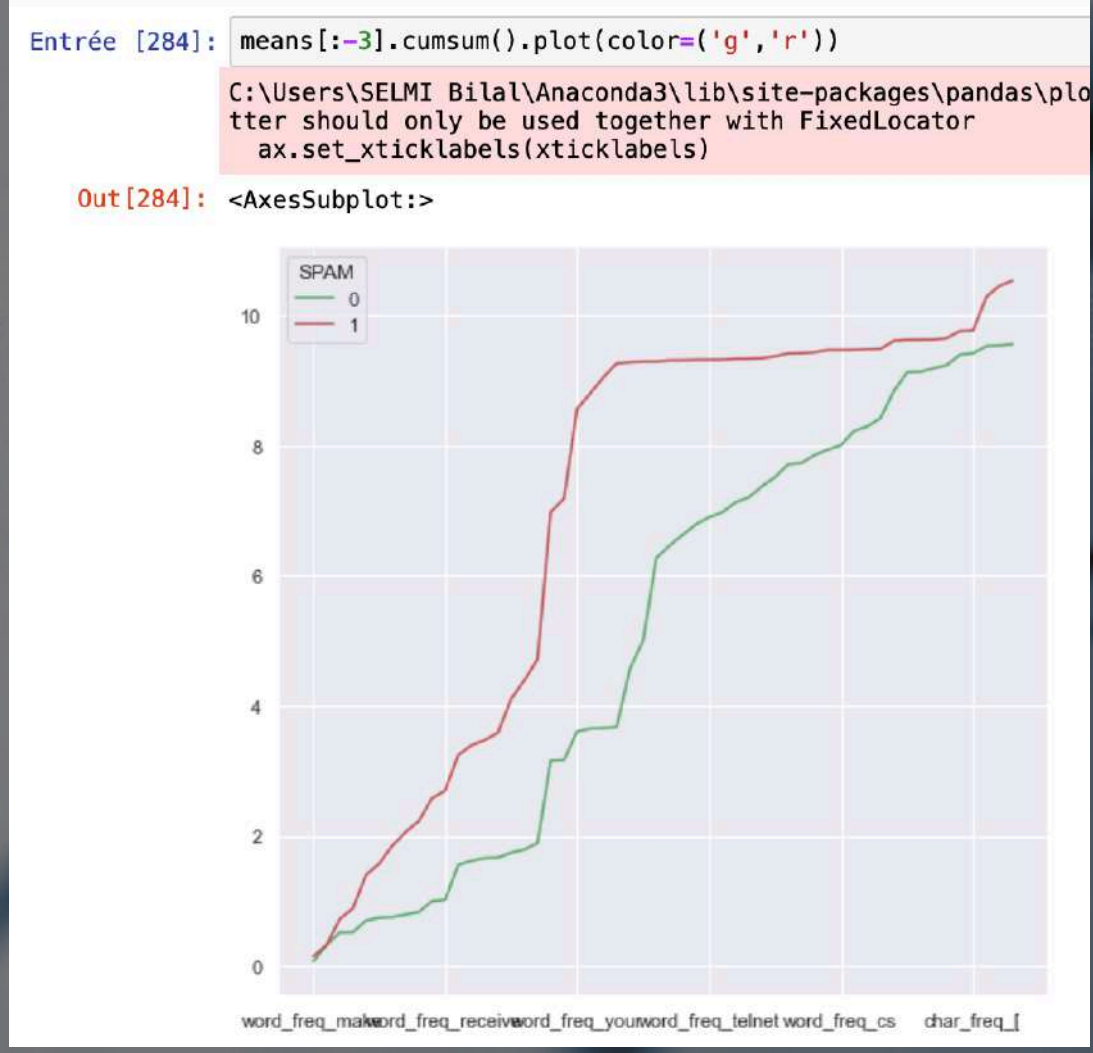
Pie charts are very useful for quantitative variables. It allows us to see which variable(s) takes a biggest(s) part on the dataset. For example the frequency of a word like “you” on an email is more important than a one like “3d” to determine if this email is a spam one or not. You can find above some pie charts we plotted to do these conclusions.

From the pie charts of the top to those of the bottom, we excluded the variable which was too dominant.









Graphs 📈

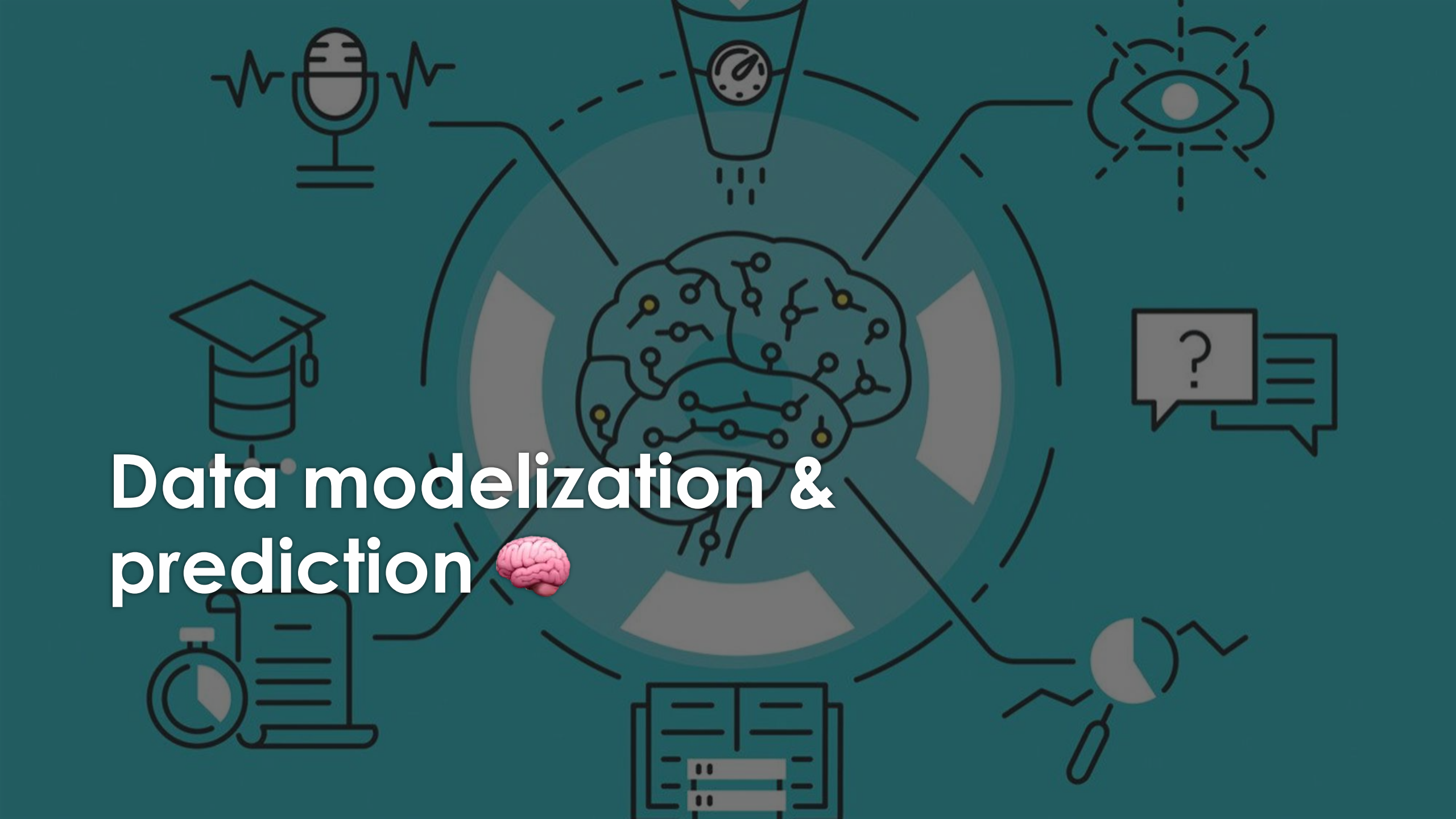
A graph is pertienent to compare quantitive informations and qualitative aspects. We plotted few graphs to compare the frequencies of different words like : make, you, receive, tel, etc in a spam and a non-spam email or to evaluate the autocorrelation of some variables.

Data analysis conclusion 🤔

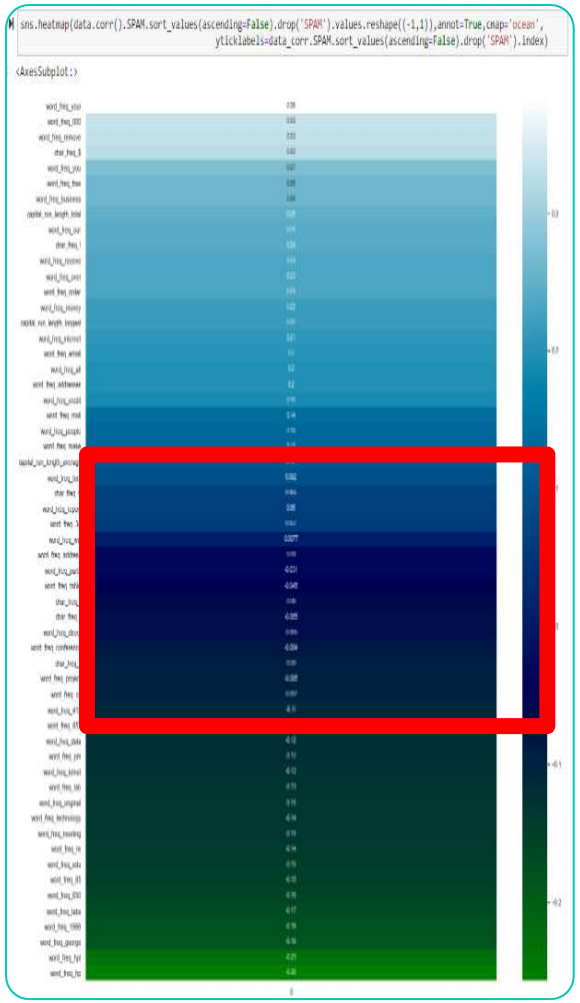
After the realization of this 2-parts analysis, we had a better idea of our data and the parameters which are more to do a mail a spam one.

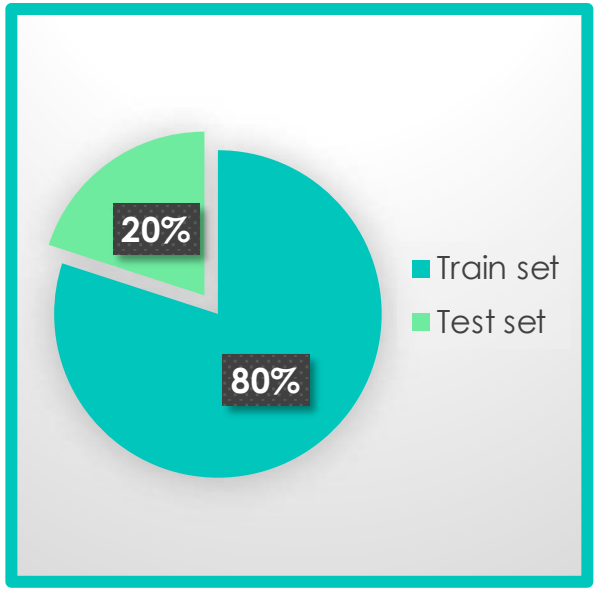
So, we start the second part of this project : the machine learning. We wanted to make prediction of some parameters by using different inteeligent methods to predict if, based on previous database, if an email will be a spam or not…

Let’s discover it !



Data modelization & prediction 🧠

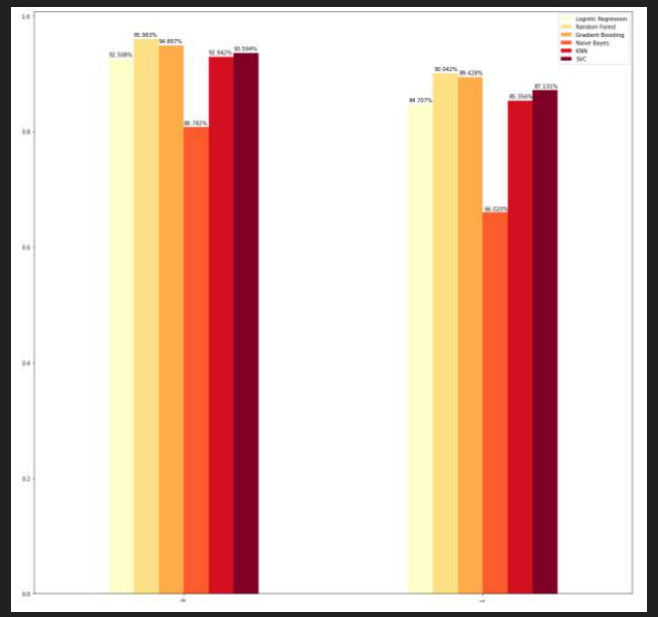


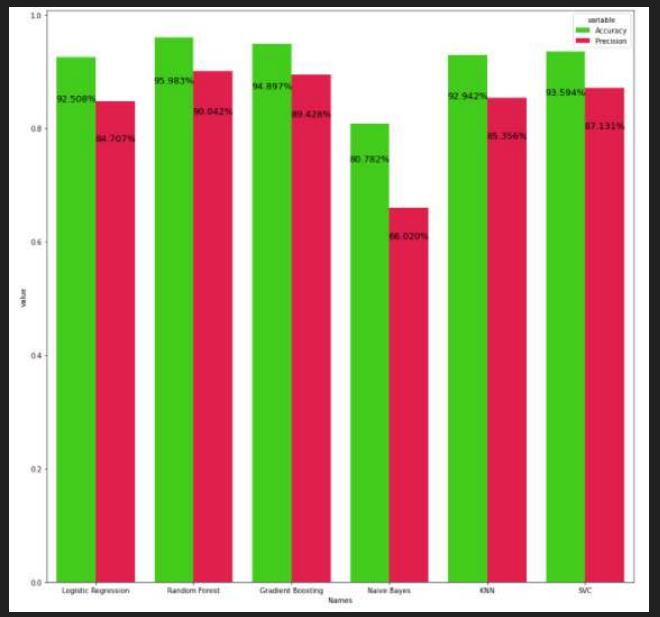



```
my_seed = 704043
from sklearn.model_selection import train_test_split
rand.seed( my_seed )
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.80)
```

Splitting the data

- For each model, we are going to train it on 4 forms of data : all the data, the whole data scaled, the data just with the selected featured & the data scaled just with the selected featured.
- So we had 4 datasets and for each of them we had in a train set & a test set in order to fit our models.
- Then, we did a preprocessing : we excluded the variables which have a correlation under -0.12 and upper 0.12.





▼ **List of Models**

- 1. Logistic Regression
- 2 . Random Forest
- 3 . Gradient Boosting
- 4 . Naive Bayes
- 5 . KNN
- 6 . SVC

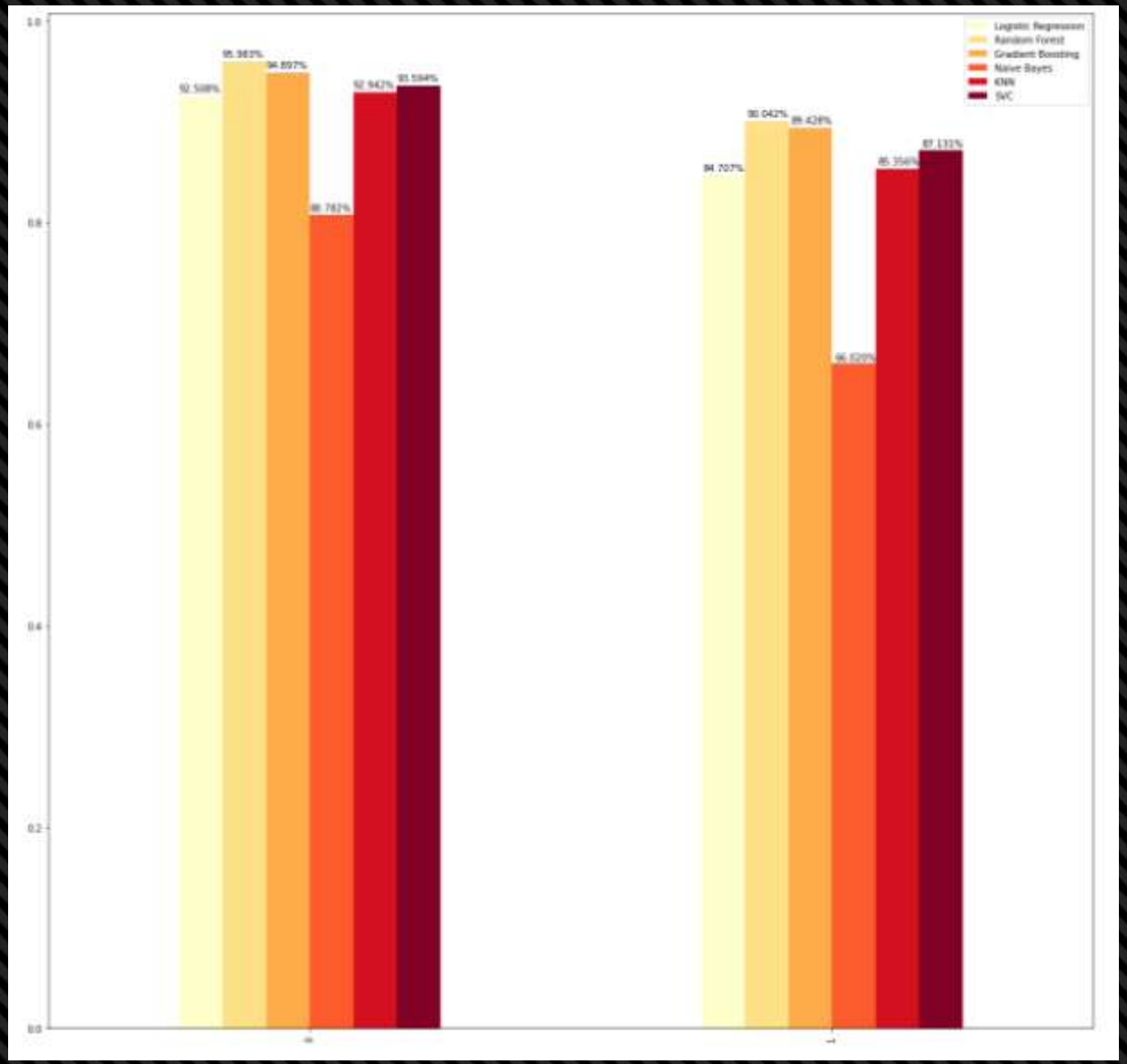
Best model 🏆

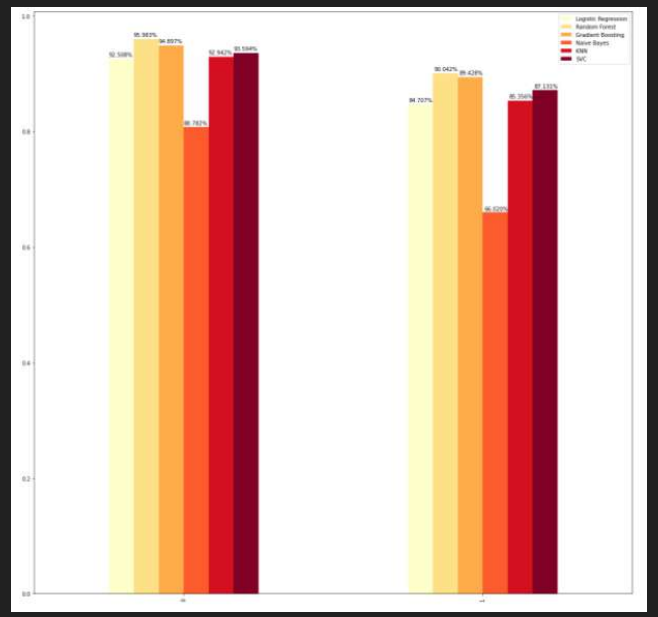
Best results from all Models :

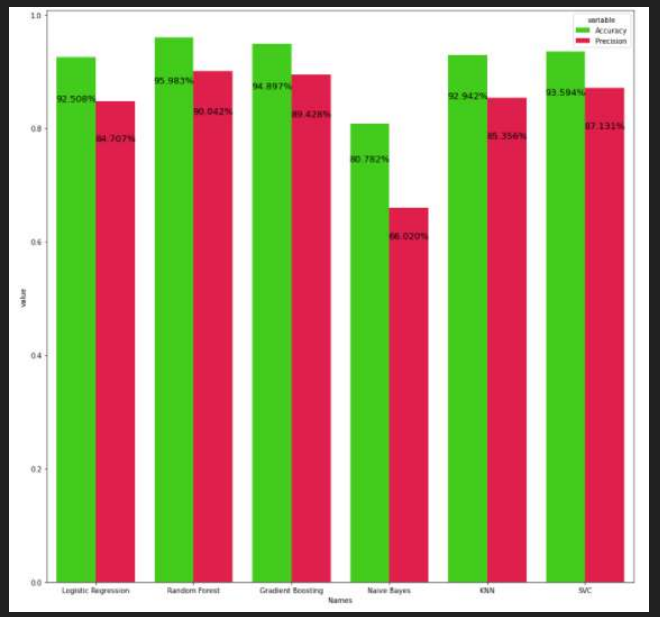
Accuracy of 95.98262757871878 %

Precision of 91.6194502545261 %

- We worked on 6 machine learning models : logistic regression, random forest, gradient boosting, naive Bayes, KNN, SVC.
- As you can see on the plots below, the most efficient predictive model of the 6 we chose in terms of accuracy and precision is the **random forest model**. You can click on the plots to see it more in depth.


Logistic Regression
Random Forest
Gradient Boosting
Naive Bayes
KNN
SVC
92.508%
95.363%
94.597%
88.782%
92.947%
93.594%
84.707%
90.042%
89.428%
86.020%
86.356%
87.131%
1.0
0.8
0.6
0.4
0.2
0.0



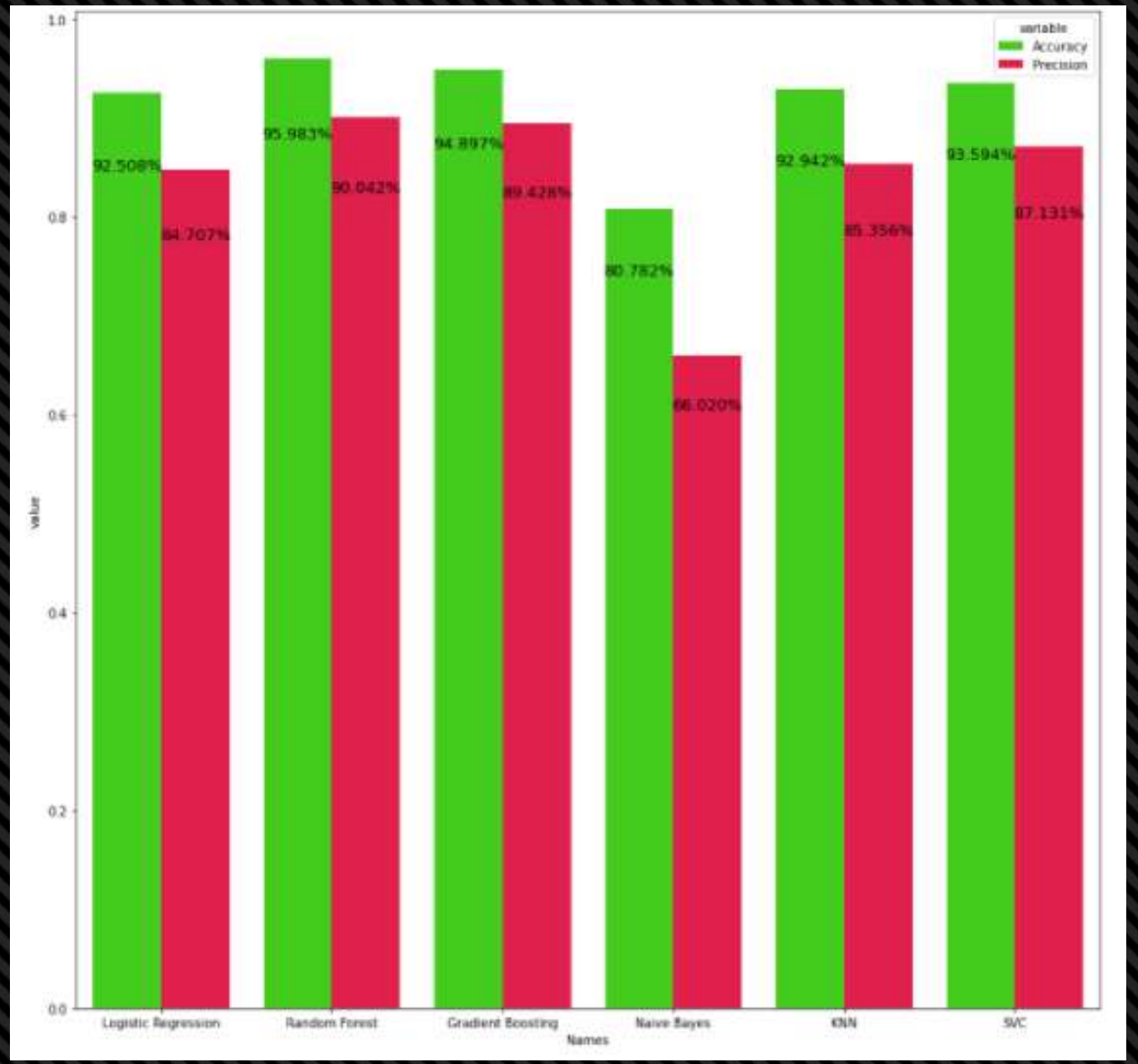


▼ **List of Models**

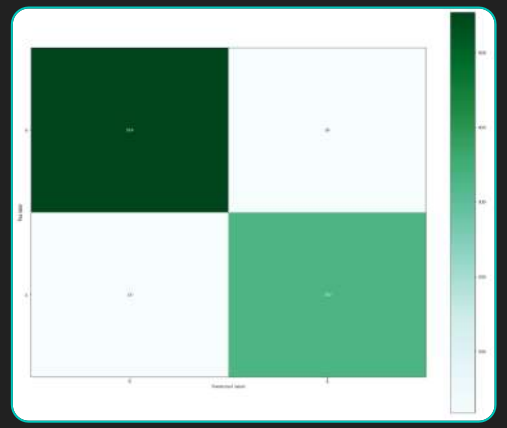
- 1. Logistic Regression
- 2 . Random Forest
- 3 . Gradient Boosting
- 4 . Naive Bayes
- 5 . KNN
- 6 . SVC

Best model 🏆

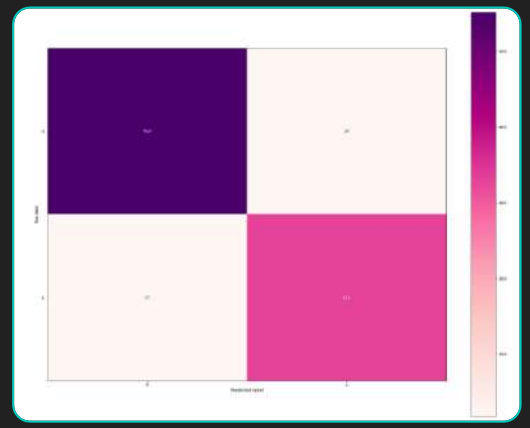
- We worked on 6 machine learning models : logistic regression, random forest, gradient boosting, naive Bayes, KNN, SVC.
- As you can see on the plots below, the most efficient predictive model of the 6 we chose in terms of accuracy and precision is the **random forest model**. You can click on the plots to see it more in depth.


variable
Accuracy
Precision
1.0
0.8
0.6
0.4
0.2
0.0
value
92.508%
84.707%
95.983%
90.042%
94.897%
89.428%
80.782%
66.020%
92.942%
85.356%
93.594%
87.131%
Logistic Regression
Random Forest
Gradient Boosting
Naive Bayes
KNN
SVC
Names

Optimization with features engineering 🎲



gridsearch



- We used **gridsearch** which is a library that allows us to optimize the hyper parameters of the model contained in a list in order to obtain the best accuracy & the best precision.
- You see on the right the best theorical parameters to give the best accuracy and precision for the random forest model.
- Unfortunately, the accuracy and the precision have not really increased, so kept the original model.

```
best_model.score(x_test, y_test)
```

```
0.9587404994571118
```

```
precision(y_test,best_model.predict(x_test))
```

```
0.9128629128673005
```

```
best_model.get_params()
```

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': 20,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 250,
 'n_jobs': None,
 'oob_score': True,
 'random_state': 1,
 'verbose': 0,
 'warm_start': True}
```

Importance of the features 👍🏼

Here you can see that some feature don’t really have a big influence on our model, so we removed them.

Once again, this choice was not conclusive.

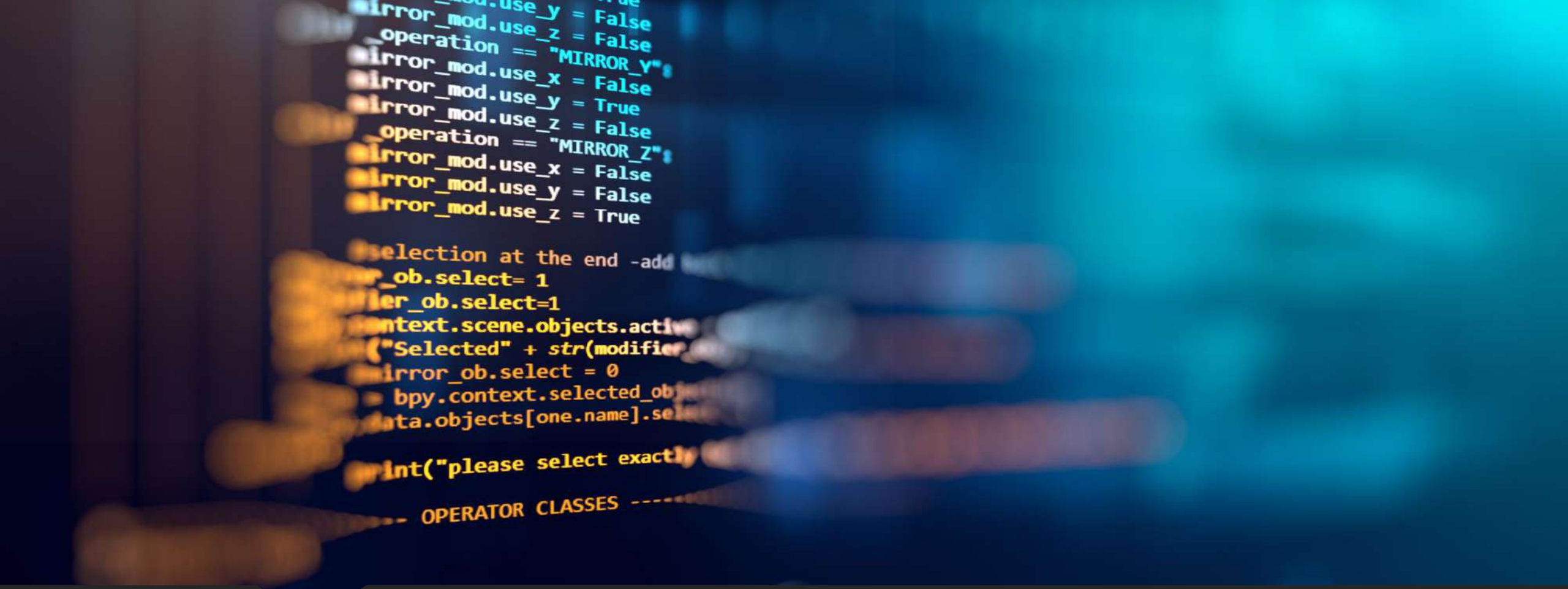
Model saving

- Finally, we decided to keep the first random forest model because it had the best results.
- So we saved it in a file using the library joblib to export it for the API.

Entrée [223]:
```
from joblib import dump, load
RF.fit(x_train,y_train)

dump(RF, 'model_saved.joblib')
```

Out[223]:
```
['model_saved.joblib']
```



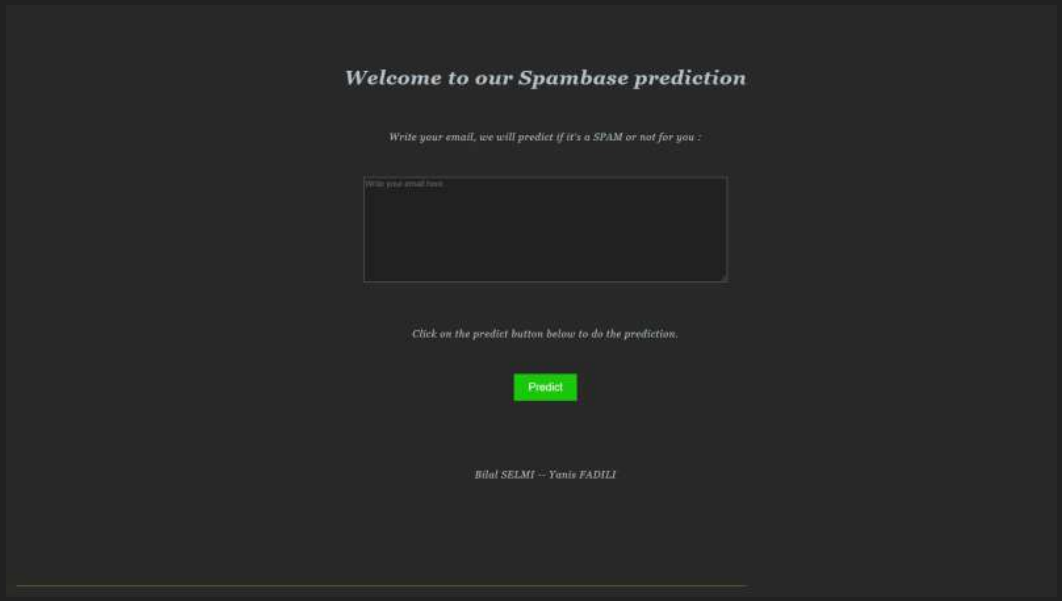
Transformation into a Flask API 💻

Interface

HTML
CSS

- We developed a web application in HTML & CSS based on our Python project to see if an email is a spam or not.
- We can distinguish 2 interfaces : one where you must enter an email and another one where you can see the prediction.

1



2

IF IT'S A SPAM

IF IT'S NOT A SPAM

Link with the Python project 🐍

- In the API, we wrote a Python file in Visual Studio Code to predicted if the entered mail is a spam or not.
- We created a function Frequencies in which we stored the email content into a string variable, and we calculate the frequency of each word present in the email. Then the function matches the email's words with the words of each columns of the spam dataset in order to apply the prediction on it. Finally, we store the prediction in a binary variable : 0 if the email is not a spam, 1 if it is.

```
14  def Frequencies(text):
15
16      variables = ['make', 'address', 'all', '3d', 'our',
17                   'over', 'remove', 'internet', 'order',
18                   'mail', 'receive', 'will', 'people', 'report',
19                   'addresses', 'free', 'business', 'email', 'you',
20                   'credit', 'your', 'font', '000', 'money', 'hp', 'hpl',
21                   'george', '650', 'lab', 'labs', 'telnet', '857', 'data',
22                   '415', '85', 'technology', '1999', 'parts', 'pm', 'direct',
23                   'cs', 'meeting', 'original', 'project', 're', 'edu', 'table',
24                   'conference', ';', '(', '[', '!', '$', '#',
25                   'capital_run_length_average', 'capital_run_length_longest', 'capital_run_length_total']
26
27      values100 = []
28      words = re.findall(r"[\w']+", text)
29      words_count = collections.Counter(words)
30
31
```

Conclusion

We are right now able to know in an efficient way if a mail is spam or not. We took so much pleasure by doing this project, which allows us to improve our machine learning skills and our Python coding abilities.

Thank you for your attention !

Here's our contact details :

- Bilal SELMI – *bilal.selmi@edu.devinci.fr*
- Yanis FADILI – *yanis.fadili@edu.devinci.fr*

The link of the GitHub repository in where you can find the project :

https://github.com/bilalselmi7/SPAM_Prediction