Spam Dataset Analyzis

Antoine CELLIER – Pawel PRESNAL – Guillaume ZHU

The dataset

Our dataset contains many attributes whose purpose is to describe an e-mail content in order to determine if this is a spam or not.

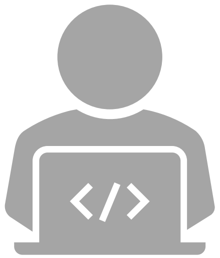
In order to do so the dataset has 57 columns :

48 attributes of type word_freq_WORD which describe the percentage of words in the e-mail that match WORD	6 attributes of type char_freq_CHAR which describe the percentage of characters in the e-mail that match CHAR	3 attribute of type capital_run_length_... Sum/Longest/Average of length of uninterrupted sequences of capital letters	1 attribute that is equal to 1 if the e-mail is a spam and 0 if not

Ins and Outs

Ins :

- The file contains a lot of individuals. It allows to have an accurate training and testing for our model.
- The dataset also has many attributes. Thanks to this we will be able to test and find which attributes show a correlation with an email being a spam



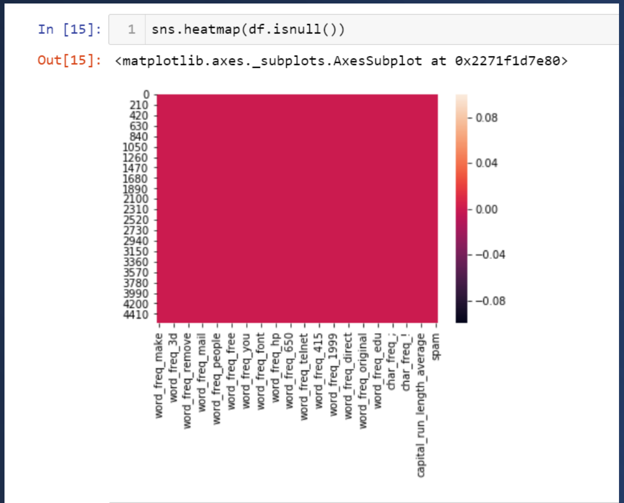
Outs :

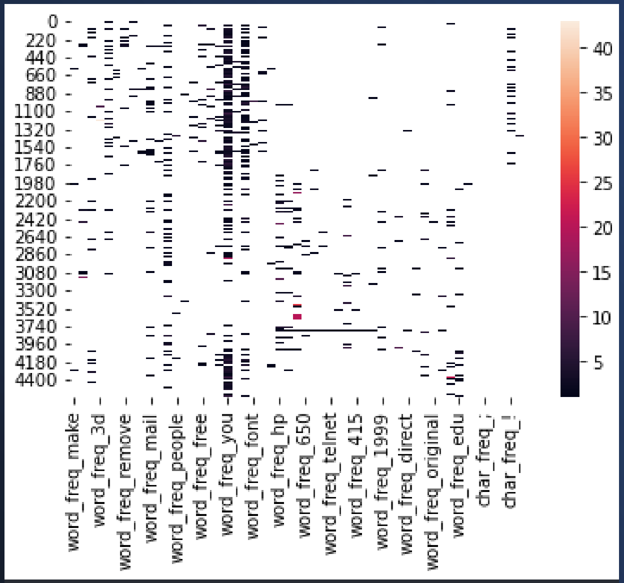
- It would have been interesting to have total number of words for each e-mail.
- We have only 3 types of columns : word frequency, character frequency and Capital letter datas.
- We have no connection with the real data

How we worked on this data-set : pre-processing

We started working on the data-set by studying the content and format of each columns. To do so we used basics functions (shape/describe/dtypes) and the web description of the data-set.

Then we checked if all data in the set was relevant using seaborn.heatmap





How we worked on this data-set : pre-processing

- Then we created some sub-set in order to facilitate our analyze :

  - df_spam : it contains every columns for each line identified as a spam. This means the column « spam » in this dataset will always equal 1
  - df_non_spam : equals df-df_spam

  - df_freq : contains every lines for every « frequencies » columns
  - Df_non_freq : equals df-df_freq

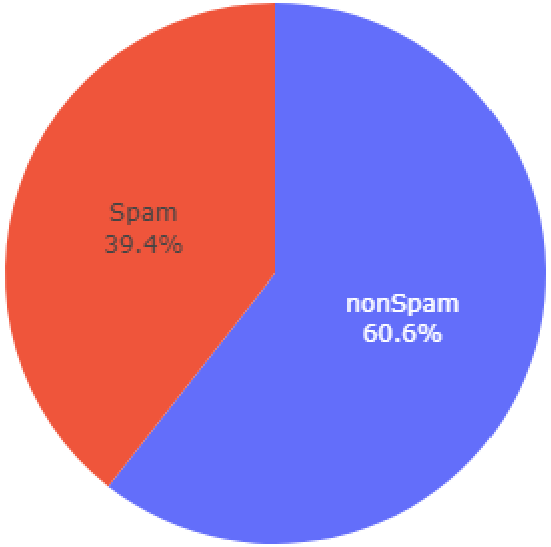
How we worked on this data-set : visualisation

- Following we start making visualisation in order to analyze the spam data base :
  - First, we checked the proportion of spam and not-spam e-mail in our data base.
  - Second, we studied correlations between columns using a correlation matrix
  - Finally, we used many graphs to identify which attributes had the best chances of being usefull to deduct if an e-mail is a spam or not.

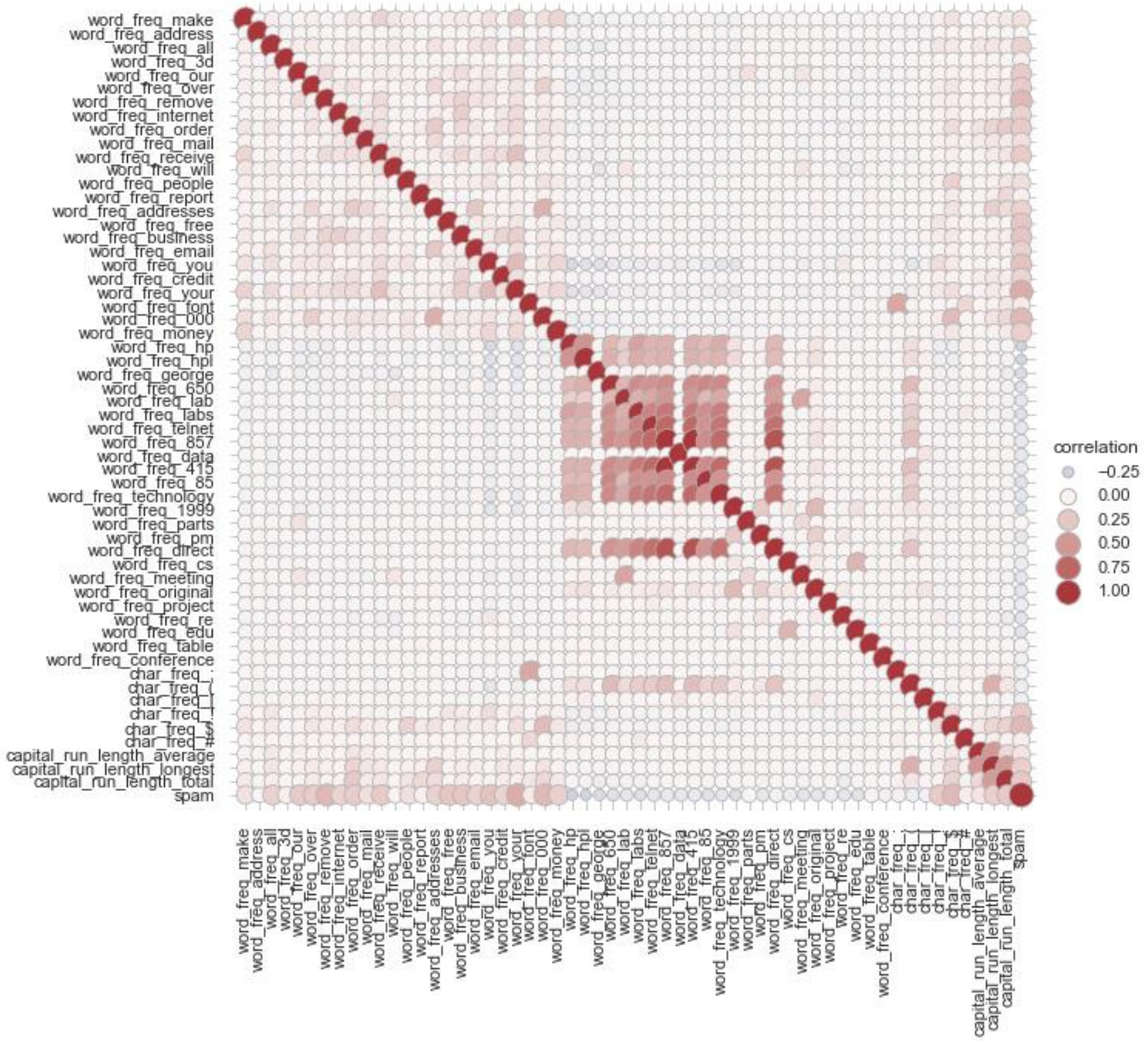
Proportion of spam in the dataset

```
fig=plt.figure()
pie=px.pie(df.spam.value_counts(),
           values=df.spam.value_counts(),
           names=["Non Spam","Spam"],
           title='Proportion of spams')
pie.update_traces(textposition='inside', textinfo='percent+label')
fig.savefig('Proportion_figure.png')
```

```
<Figure size 640x480 with 0 Axes>
```

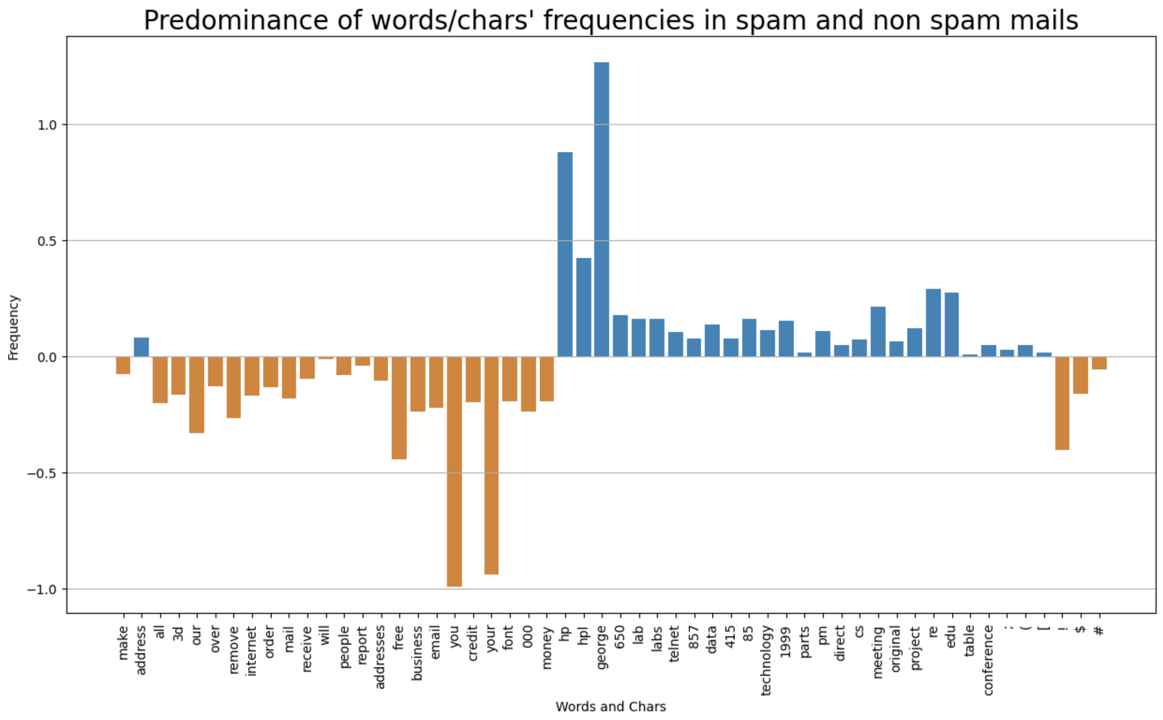


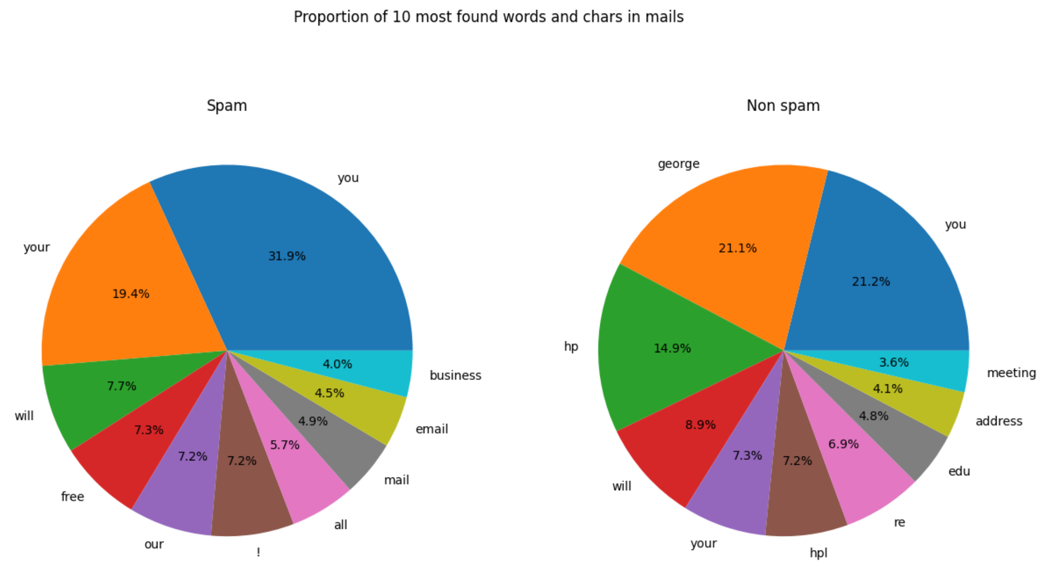
Some graphs to identify the more relevant attributes -Matrix of correlation-



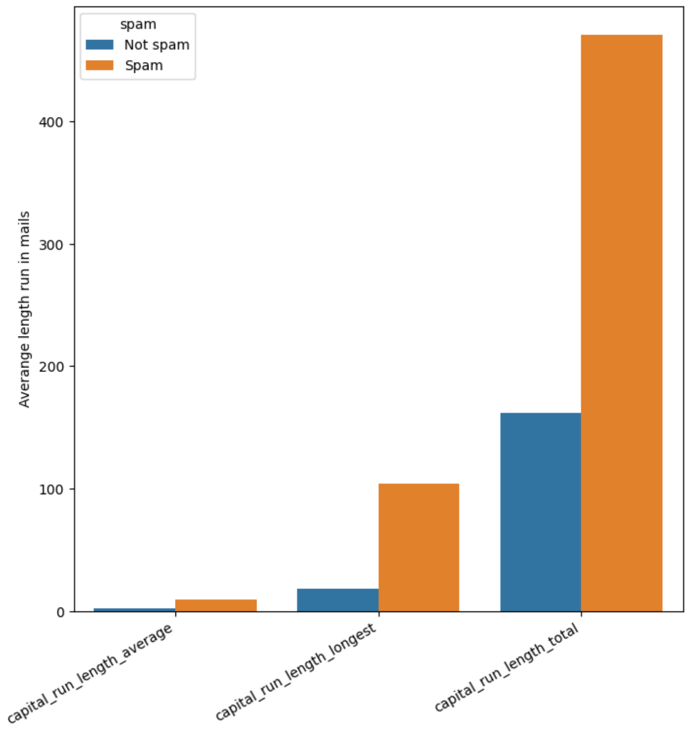
ESILV ENGINEERING SCHOOL DE VINCI PARIS

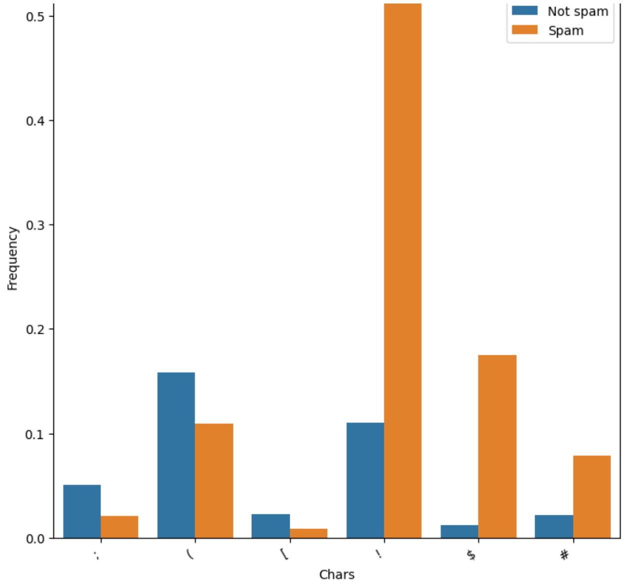
Some graphs to identify the more relevant attributes -Analysis-





Some graphs to identify the more relevant attributes -Analysis-





ESILV
ENGINEERING SCHOOL DE VINCI PARIS

How we worked on this data-set : modeling

- We started working on a model to predict if an e-mail is a spam or not by doing the standardization of the dataset. In addition we created training and test set.

In [ ]:
```
from sklearn.model_selection import train_test_split

X = np.array(df.drop("spam",axis=1))
y = np.array(df["spam"])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state=123)
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
#scaler.fit(X_train)
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

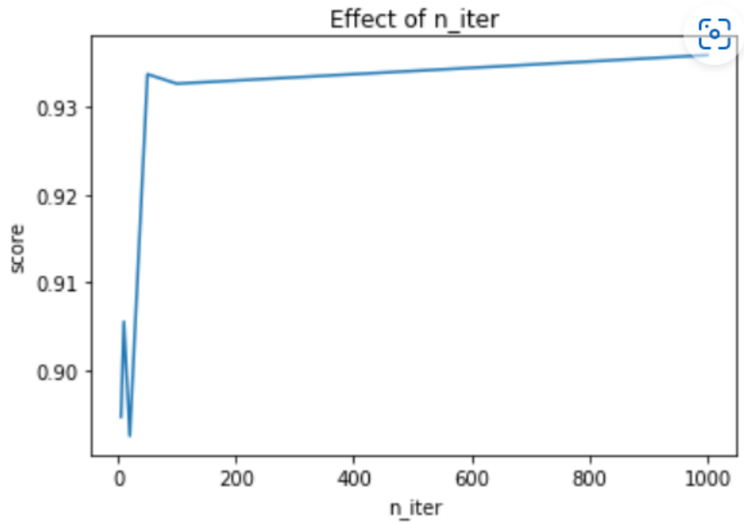

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

How we worked on this data-set : modeling

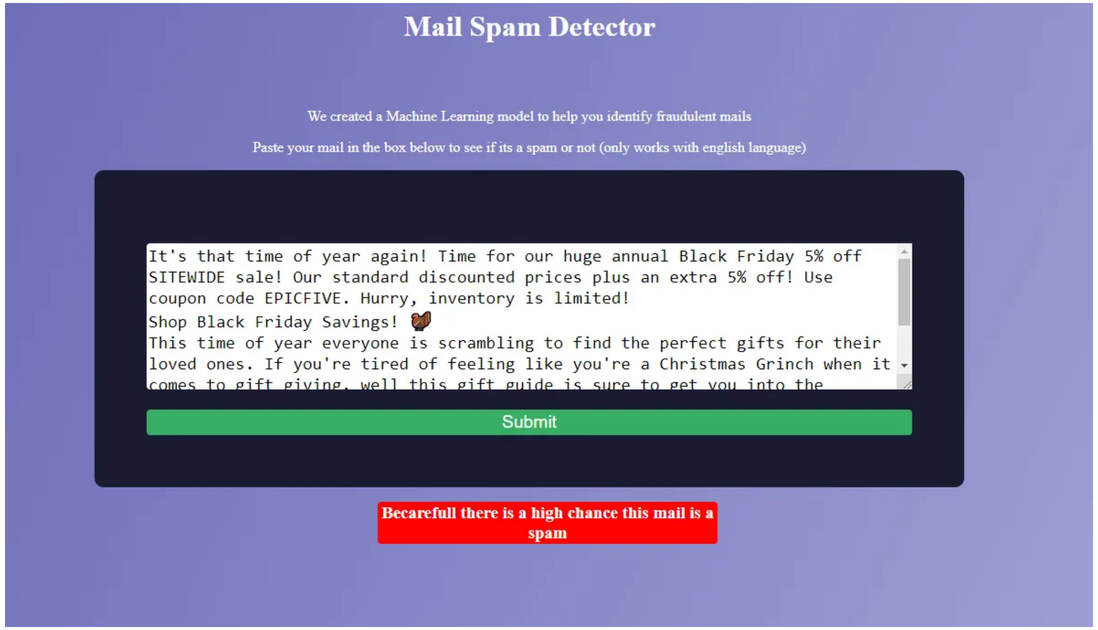
- Once this was done we started trying different model : Logistic Regression, Linear SVM and finally Random Forest
- The purpose of trying diffferents models was to be able to compare those models in order to pick the one which has the result.
- To compare them we also plotted their score
- Logistic Regression : 0.935
- Linear SVM : 0.93
- Random Forest : 0.95

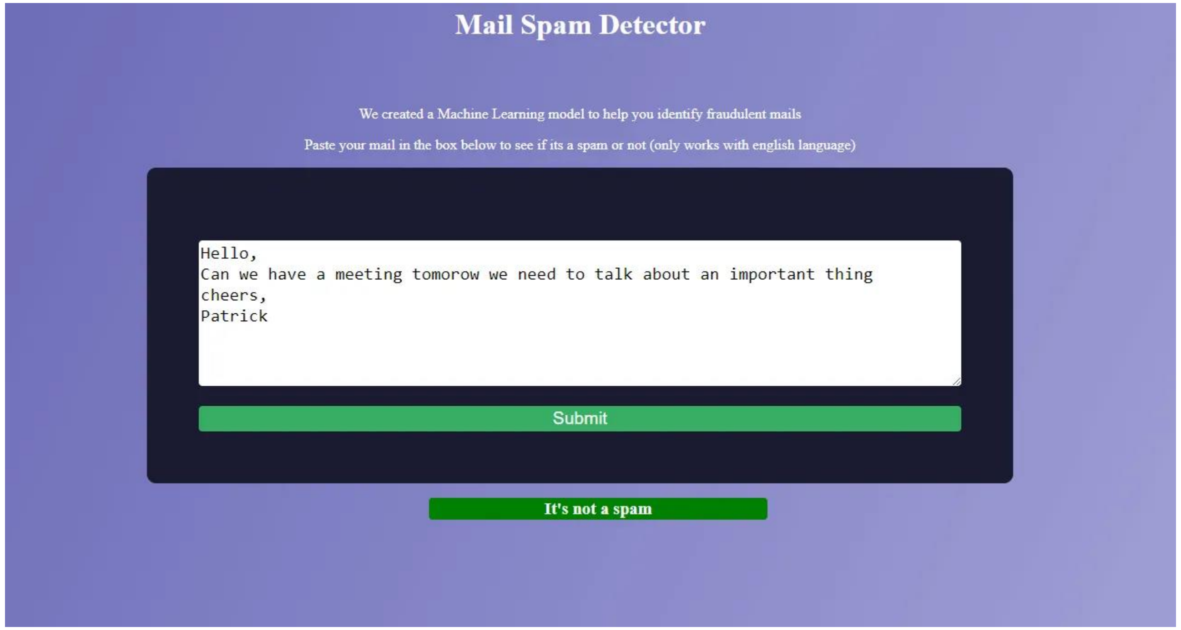
Result for the Logistic Regression :

Out[16]: [<matplotlib.lines.Line2D at 0x21d0e6f3860>]



How we worked on this data-set : Django





Merci !