Projet Python for data analysis

Nathan BOON & Marius HOUSTON

Problème:

Le problème consiste à déterminer si un mail est un spam ou non en s'appuyant sur la fréquence d'apparition d'un mot, …..

L'objectif d'un spam est varié: publicité pour des produits ou un site, générer de l'argent…

C'est un problème de classification binaire.

C'est un problème bien concret qui répond à un besoin, en effet nous recevons de plus en plus de mail ainsi il est donc très intéressant et important de pouvoir mettre en place des modèles permettant de détecter les spams.

Attributs du dataset

Le Dataset contenant des mails spams et non spams ont été récupéré d'un seul utilisateur ayant trié lui même les mails.

Composition du dataset:

4601 lignes (mails)

58 colonnes (attributs)

- 48 attributs sont des attributs qui affichent le pourcentage d'apparition d'un certain mot dans un mail: word_freq_WORD de type FLOAT
- 6 attributs sont des attributs qui affichent le pourcentage d'apparition d'un certain caractère dans un mail: char_freq_CHAR de type FLOAT
- 1 attribut pour la longueur moyenne de lettres capitales: capital_run_length_average de type FLOAT
- 1 attribut pour la plus longue séquence de lettres capitales: capital_run_length_longest de type INT
- 1 attribut pour la somme des longueurs des séquences de lettres capitales: capital_run_length_total de type INT
- 1 attribut target prenant la valeur 1 si mail spam sinon 0 de type INT

Vérification du dataset

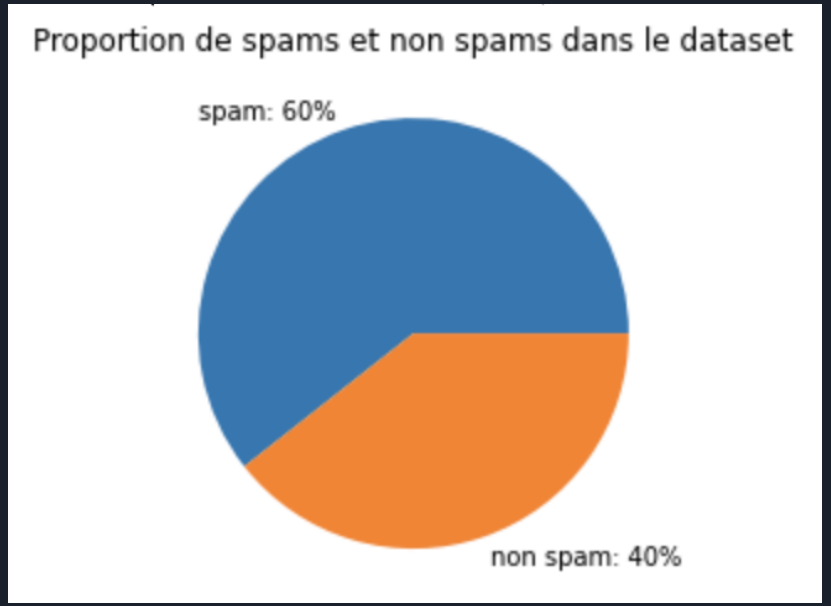
- Avant de faire de la visualisation de données et l'implémentation de modèles, nous avons vérifié qu'il ne manquait pas de données ou qu'il n'y avait pas de données indisponibles susceptible de fausser nos modèles et nos visualisations.

- En appliquant les fonctions `isna().sum()` et `isnull().sum()` nous nous sommes rendu comptes que toute la dataset était correctement remplie.

- Si nous nous étions rendus compte que des données étaient manquantes, nous aurions modifier la dataset afin de la rendre exploitable.

Le dataset

- Le dataset contient 40% de mails spams et 60% de mails non spams



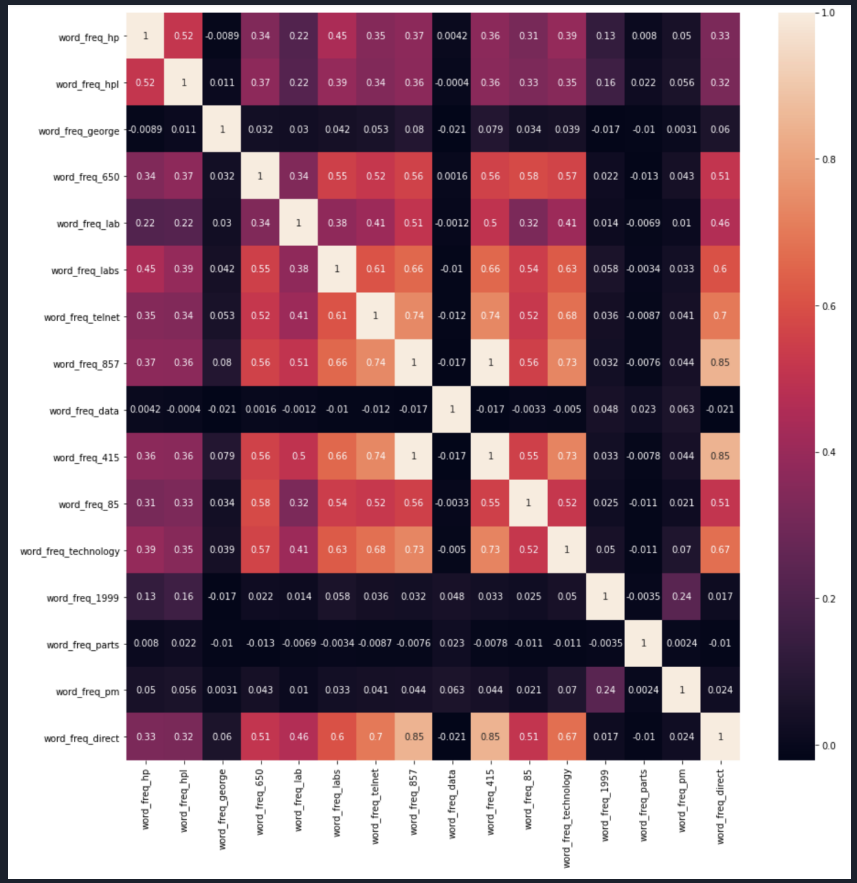
Data visualisation: matrice de corrélation

Afin d'afficher les différentes corrélations qui existent entre les attributs de notre dataset, nous avons affiché une matrice de corrélation:

Dans un premier nous avions réalisé une matrice de corrélation sur l'ensemble des attributs cependant celle-ci n'était pas lisible (disponible dans le notebook) mais nous a permis d'effectuer une matrice de corrélation sur des attributs intéressants.

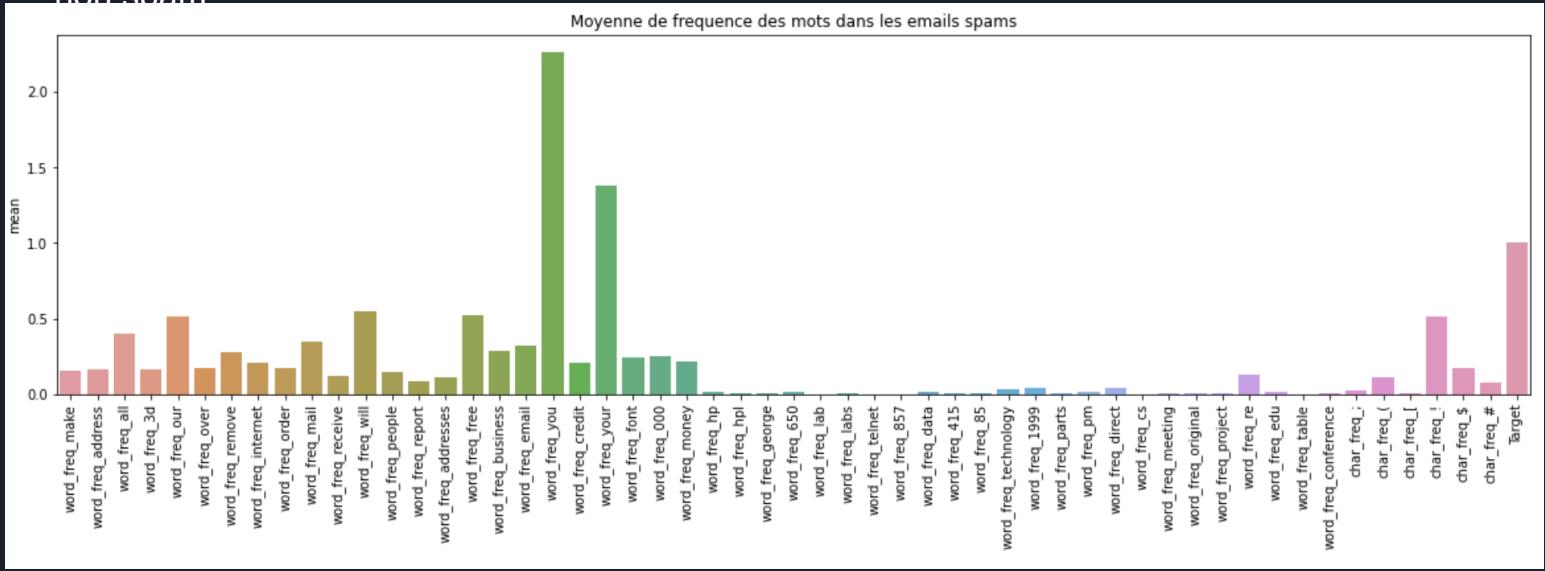
On observe une forte corrélation entre:

- word_freq_857 et word_freq_direct
- word_freq_415 et word_freq_direct



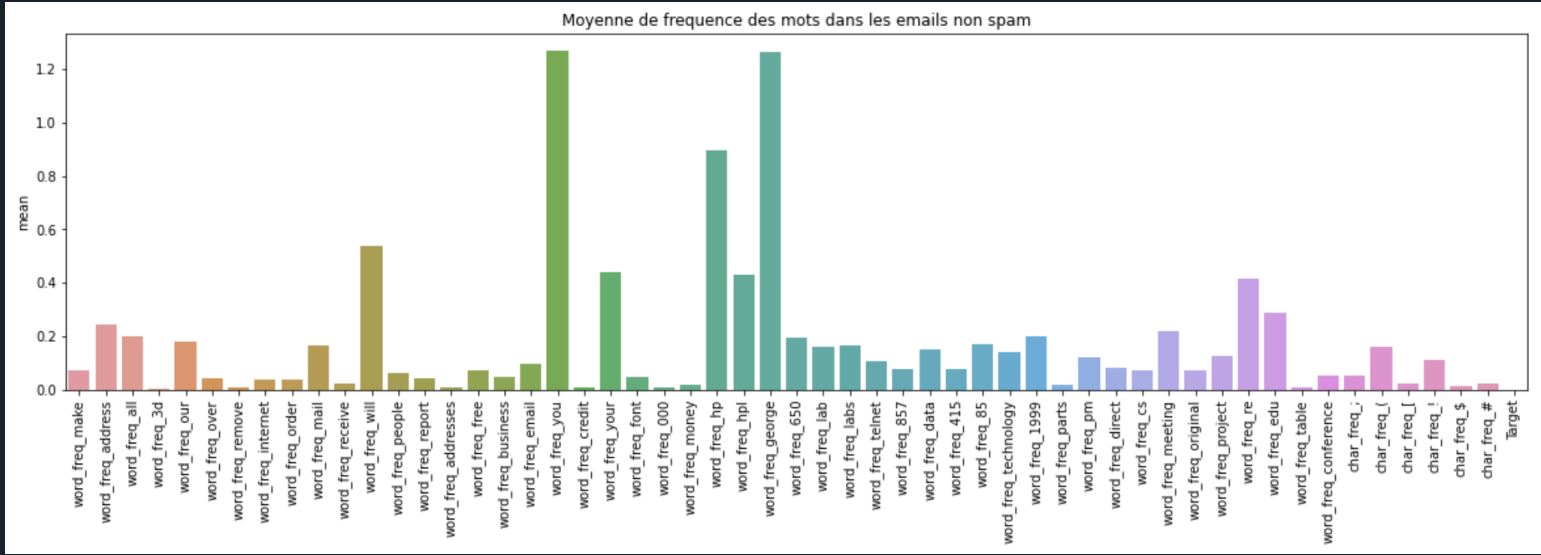
Data visualisation: fréquence des mots pour spam

Etant donné que la fréquence des mots sélectionnés est important pour prédire si un mail est un spam ou non, nous avons affiché moyenne d'apparition d'un mot dans un mail spam et non spam:



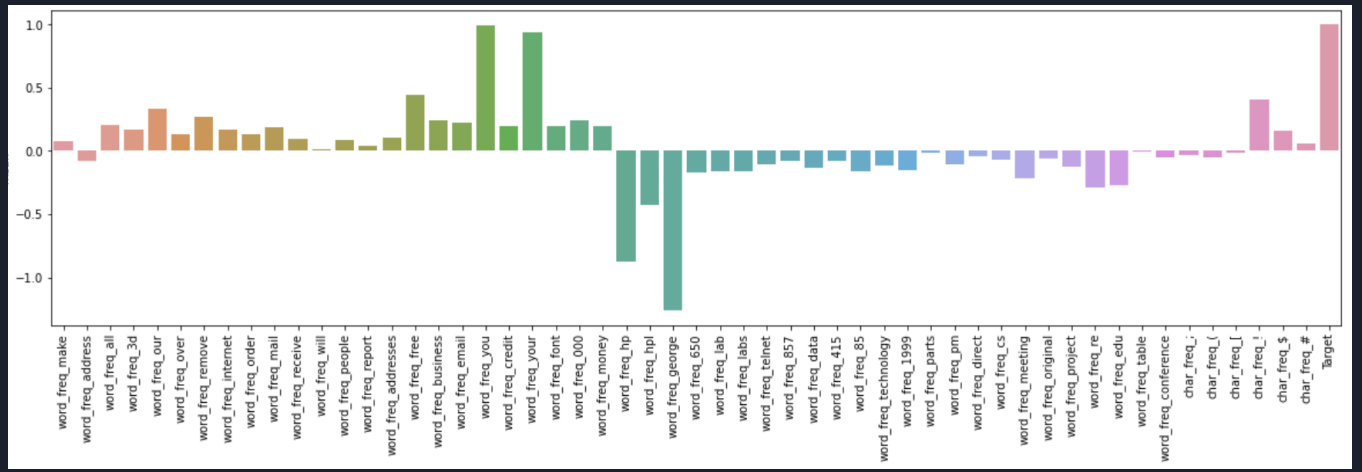
Data visualisation: fréquence des mots pour non spam

Cette visualisation nous permet de voir quel sont les mots qui apparaissent beaucoup plus dans un mail spam que dans un mail non spam: “3D”, “000”, “remove”, apparaissent respectivement 185, 29 et 34 fois de plus dans les mails spams.



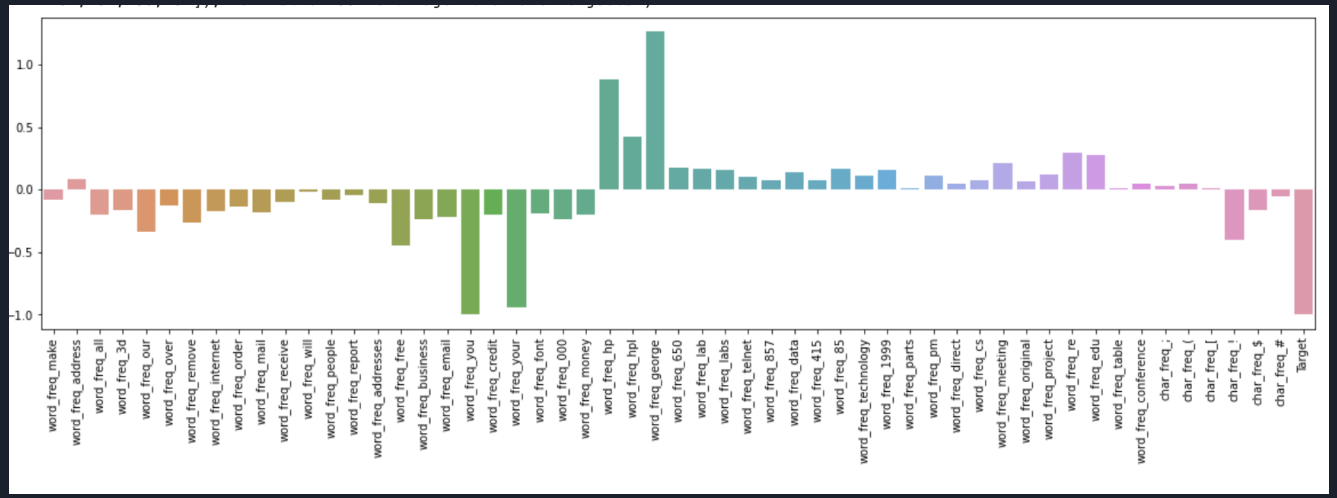
Data visualisation: fréquence des mots pour exclusif aux spams

Nous avons dans la visualisation que des mots tel que “you”, “your”... sont présent de nombreuses fois dans les mails spams mais également non spam ainsi cette visualisation nous permet de voir les fréquences moyennes des mots dans les spams. On soustrait au fréquence moyenne de chaque mot dans les spams les fréquence moyenne des mots dans les non spams. On remarque que les mots free, remove et “!” sont des mots propres au spams en plus de mots courant comme “you”.



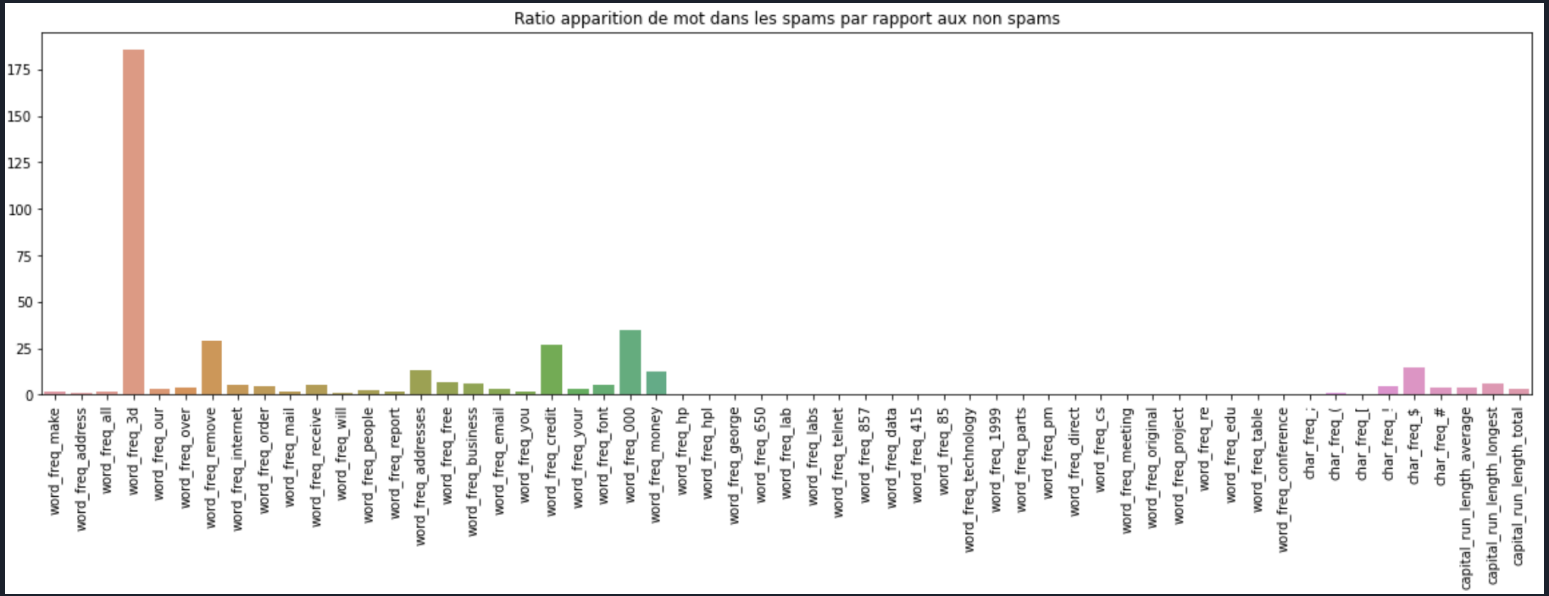
Data visualisation: fréquence des mots pour exclusif aux non spams

Cette visualisation nous permet de voir les fréquences moyennes des mots dans les non spams. On soustrait au fréquence moyenne de chaque mot dans les non spams les fréquence moyenne des mots dans les spams. On remarque que les mots hp, hpl et george sont des mots propres au non spams.



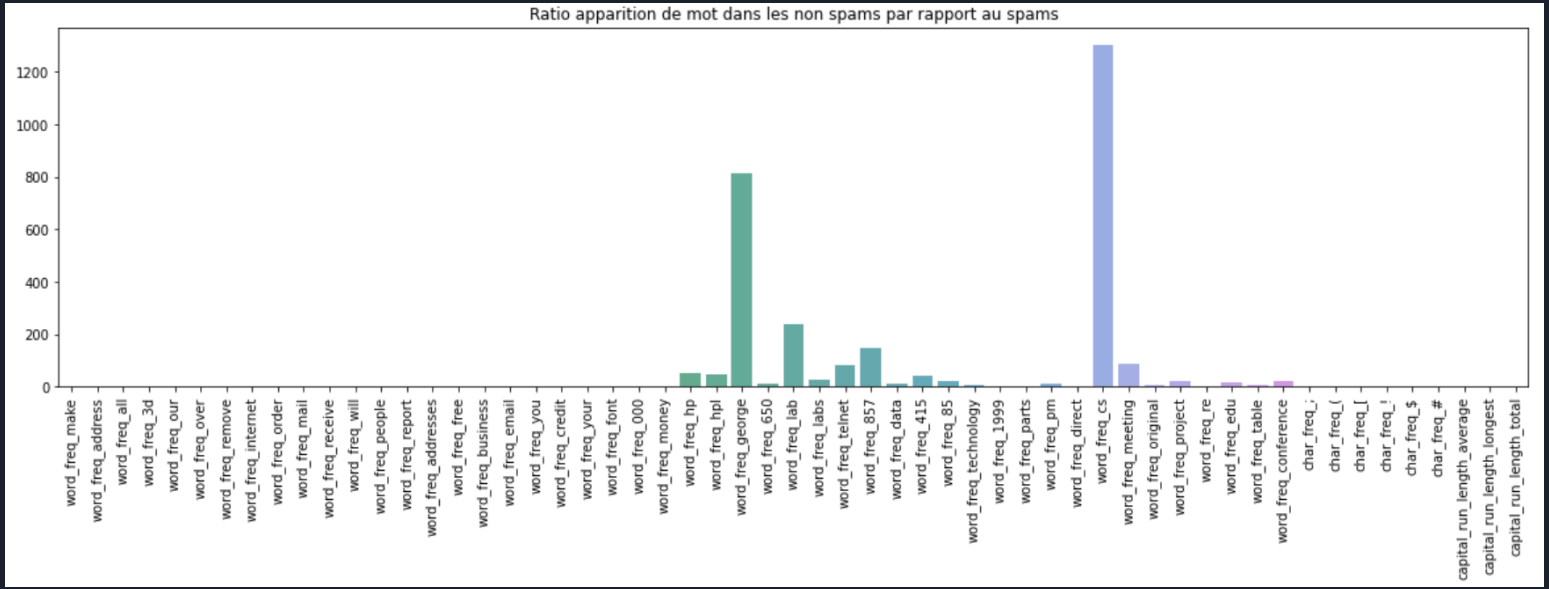
Data visualisation: Ratio des apparitions des mots spam/non spam

Cette visualisation nous permet de voir quel sont les mots qui apparaissent beaucoup plus dans un mail spam que dans un mail non spam: “3D”, “000”, “remove”, apparaissent respectivement 185, 29 et 34 fois de plus dans les mails spams.



Data visualisation: Ratio des apparitions des mots non spam/spam

Cette visualisation nous permet de voir quel sont les mots qui apparaissent beaucoup plus dans un mail non spam que dans un mail spam: “cs”, “george”, “lab”, “857”, “meeting” apparaissent respectivement 1305, 815, 237, 149 et 88 fois de plus dans les mails non



Ajout de colonne à la dataframe

Dans le problème que nous avons et avec le dataset, nous sommes en présence de données qui ne sont pas brutes. En effet, cette dernière a été constitué à la suite d'un choix arbitraire d'une personne jugeant si le mail était un spam.

Ainsi et étant donné que ce filtrage est propre à la personne, il est impossible d'ajouter de nouvelles colonnes issues de données brutes.

Si l'objectif avait été d'effectuer une filtrage personnalisé des mails en spam ou non spam nous aurions ainsi fait le choix de mots et d'attributs différents nous poussant à créer de nouvelles colonnes.

Implémentation des modèles: préparation des datas

Avant de commencer l'implémentation de différents modèles et après s'être assuré que le dataset était bien complet, il nous faut diviser le dataset en train et test afin dans un premier temps d'entraîner nos modèles et par la suite de les tester pour vérifier leur précisions et retenir le meilleur.

Par défaut nous avon diviser le train et test en proportion 70/30.

Implémentation des modèles: KNeighborsClassifier

Dans un premier temps nous avons implémenté un modèle de classification des voisins les plus proches (KNN) sur l’ensemble de nos attributs.

Précision: 80,21%

Implémentation des modèles: Clustering

Notre second modèle de classification a été un système de clustering basé sur algorithme nommé k-means.

Il consiste à diviser les éléments en un nombre de groupe fixé (en l'occurrence deux spams ou deux non spams).

Précision : 65,76%

Implémentation des modèles: Regression linéaire

Finalement nous avons opté pour une regression linéaire qui sans amélioration nous obtenait une précision de 92,71%.

Une précision aussi élévée sans ‘feature engineering’ s’explique par le fait que les données que nous utilisons ont déjà été choisi par précaution par l’utilisateur.

En effet, la donnée brute, à savoir les mails ne nous été pas donné. Ainsi, les données que nous avons ont déjà subi du feature engineering.

Précision: 93,26%

Amélioration des modèles:

Dans l’optique d’améliorer la précision de nos modèles, nous avons fait appel à la méthode classique du moindre carré.

C’est un modèle de régression qui nous sert à déterminer dans quelle mesure chaque colonne impacte les chance qu’un mail soit un spam ou non en ce concentrant sur ce qu’on appelle: le p_value.

Nous avons enlevé les colonnes avec les p_value les plus élevés, puis en testant petit à petit nous arrivons de meilleurs résultats.

Modèles sur un subset de colonnes

Ainsi nous avons retesté nos différents modèles en supprimant de la dataframe les colonnes dont la p_value était supérieur à 0,05.

Dans nos trois modèles nous observons des améliorations de précisions.

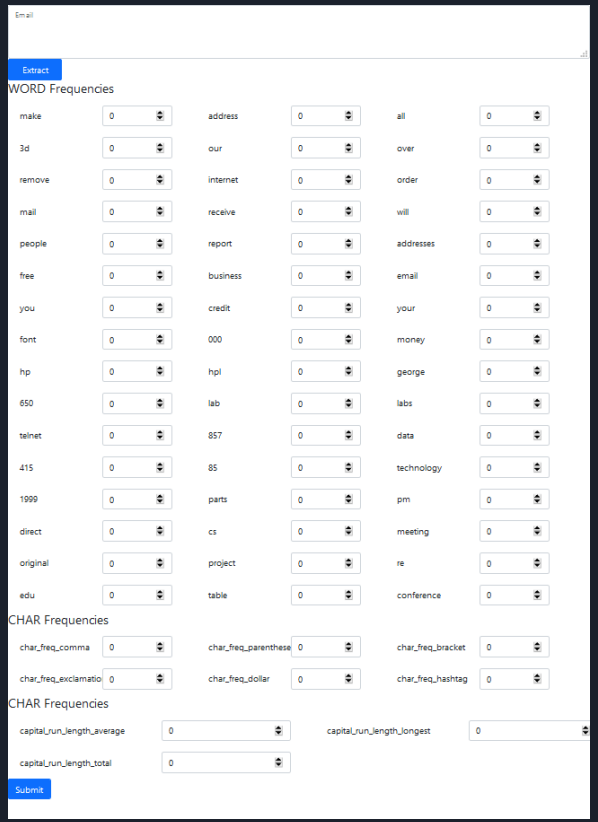
Interface graphique:

Afin d'exploiter au mieux notre API, nous avons souhaité vous donner une interface graphique (page html) dans le but de la tester.

Cette page consiste d'un form qui interroge l'api et affichera le résultats de notre prédiction.

Pour faciliter l'utilisation de la page web nous avons écrit un script JS/Jquery qui remplira les champs numérique à partir de votre mail dès l'appuie du bouton extract.

Libre à vous de remplir les champs numériques qui constitue l'ensemble des données nécessaires ou de les générés avec l'option "Extract"



Conclusion

	KNN	Clustering	Regression linéaire
Modèle base (%)	80,21	65,76	93,26
Données normalisées (%)	91,19	38,15	93,15
Subset p_value>0,05 (%)	84,13	65,97	100