Projet 5 : Catégorisez automatiquement des questions

Maby Antoine
Octobre 2022

Sommaire

Chapitre 1

Introduction

Stack Overflow est un site questions-réponses liée au développement informatique. Après avoir posÃ© une question, il est nécessaire de rentrer plusieurs tags de façon à retrouver facilement la question par la suite. Cette opération peut être compliquÃ© pour les nouveaux utilisateurs. Notre objectif est alors d'automatiser cette gestion des tags en proposant un algorithme de machine learning pour aider les nouveaux utilisateurs.

Dans un premier temps, nous allons devoir récupérer les données grâce au site mis en place par Stack Overflow. Ce site est Stack Exchange. Nous aurons également comme problématique de déterminer la qualité des posts pour réduire notre étude seulement aux bons posts. Ensuite, nous procéderons à la partie nettoyage de nos données pour les rendre utilisables dans un algorithme de machine learning. Dans le même temps, nous ferons une exploration de données. Enfin, nous finirons par tester plusieurs modèles de machine learning pour procéder à notre suggestion des tags. Nous testerons des modèles supervisés mais également un modèle non supervisé

Ce projet est l'occasion de mettre en place les techniques de traitements de données textuelles. Nous n'avons utilisé que des données numériques pour le moment et c'est une première approche. Celui-ci sera également utilisé pour avoir une première utilisation de la gestion de répertoire à l'aide d'un logiciel de gestion de versions (GitHub). Finalement, nous mettrons en place une API pour mettre en œuvre notre modèle d'automatisation de tags.

Chapitre 2

Récupération des données

Notre projet concerne la suggestion de tags pour les nouveaux utilisateurs du site Stack Overflow. Pour cela, il est nécessaire de récupérer des posts sur celui-ci. Stack Overflow propose un outil d'export de données Stack Exchange qui recense un grand nombre de données authentiques de la plateforme d'entraide. Le nombre de posts sur la plateforme étant très important, dans un premier temps, nous effectuerons une analyse exploratoire pour déterminer la qualité des posts pour les récupérer sur plusieurs années sans surcharger notre base de données finale. Pour cela, nous nous concentrerons sur un maximum de posts de l'année 2020.

Ainsi, notre base de données est composée de 100 000 posts répartis tels que 50 000 sur les 6 premiers mois et 50 000 sur les 6 derniers. Regardons les distributions de certaines variables pour suggérer des bornes inférieures qui nous permettront de déterminer la qualité des posts. Les variables analysées sont les suivantes : Nombre de vues, Nombre de réponses, Nombres de commentaires et le score de la publication.

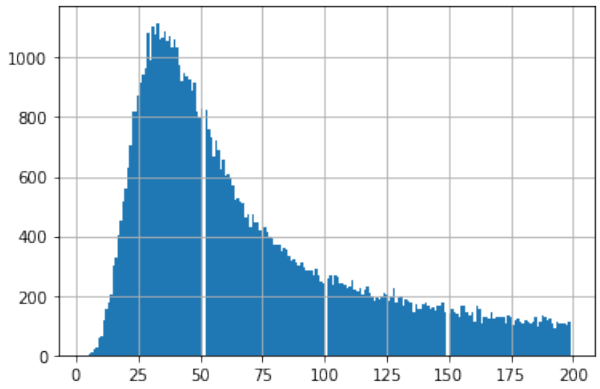

FIGURE 2.1 – Distribution du Nombre de vues par posts

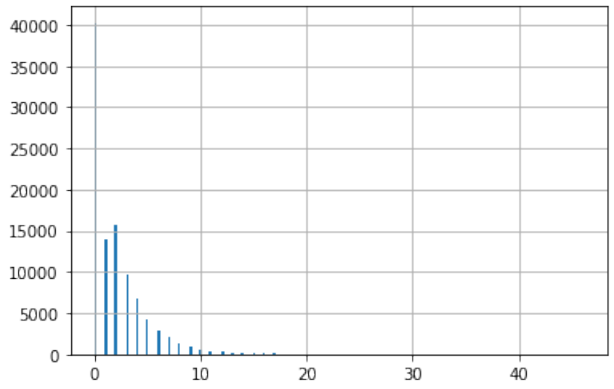

FIGURE 2.2 – Distribution du Nombre de commentaires par posts

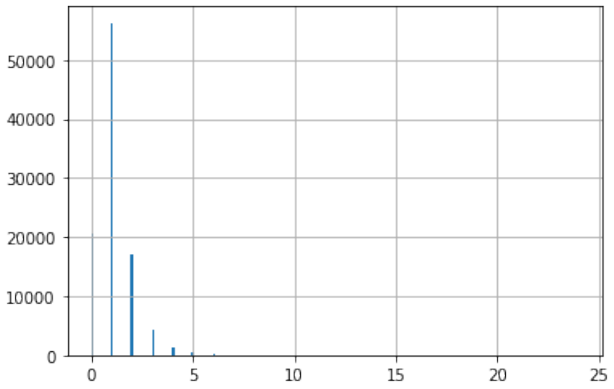

FIGURE 2.3 – Distribution du Nombre de Réponses par posts

Comme nous l'observons sur la figure (2.1), le maximum de la distribution du nombre de vues est de 35 vues. Ainsi, nous décidons de prendre au moins 35 vues pour considérer le post. Sur la figure (2.2), la distributions du nombre de commentaires est maximum pour 0 commentaire. Mais pour réduire un maximum le nombre de publications par année, nous prendrons un minimum de 4 commentaires. Sur la figure (2.3), le maximum se situe pour 1 réponse, nous posons alors que les posts doivent avoir au moins 3 réponses pour le considérer comme de qualité. Ceux-ci auront également une note minimal de 3. Toutes ces bornes inférieures ont pour but de ne considérer que les posts intéressants mais aussi pour réduire un maximum le nombre de données. Dans le but d'avoir une data-base d'une taille suffisamment grande mais sans une explosion des temps de calculs. Comme notre étude porte sur les tags, tous les posts retenus ont au moins 1 tag.

Nous avons nos bornes inférieures pour déterminer la qualité de nos posts. Nous récupérons maintenant tous les posts depuis la création de Stack Overflow. Notre data set final est composé de plus de 130 000 posts. Nous pouvons maintenant passer à la partie exploration et nettoyage de notre corpus.

Chapitre 3

Exploration et Nettoyage du corpus

Avant de nettoyer notre corpus pour l'utiliser, regardons un peu plus précisément les distributions de certaines variables pour mieux connaître notre base de données.

3.1 Exploration

Dans un premier temps, grâce aux bornes déterminant la qualité des posts, nous avons été capable de récupérer des posts depuis la création de Stack Overflow, analysons la répartions de nos donées en fonction des dates de publication.

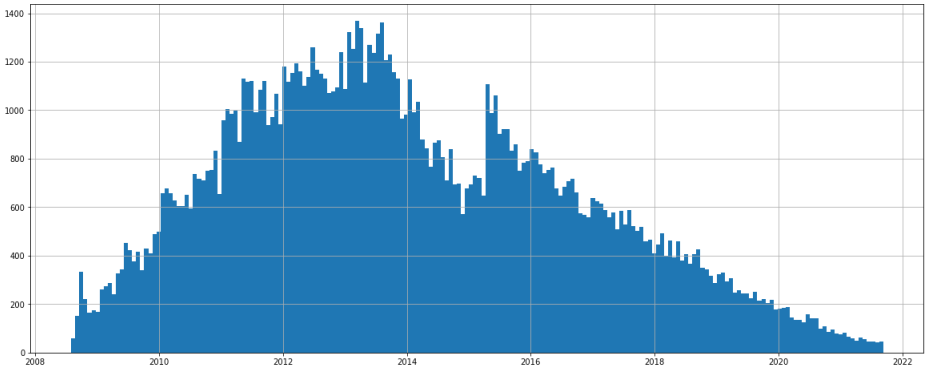

FIGURE 3.1 – Distribution du Nombre de posts en fonction de la date de publication

Il semble que le nombre de posts diminue constamment depuis 2014. Pour les dernières années, cela peut être expliquÃ© par le manque de temps pour que ceux-ci rentrent dans nos critères.

Notre projet porte sur les tags, il est intéressant d'analyser les plus employés dans notre data-base. Celle-ci en comporte plus de 18 000 différents. Nous nous limiterons aux plus importants pour réduire nos temps de calculs. Nous garderons les 50 tags les plus utilisés. L'histogramme (3.2) suivant représente leurs récurrences dans notre data set.

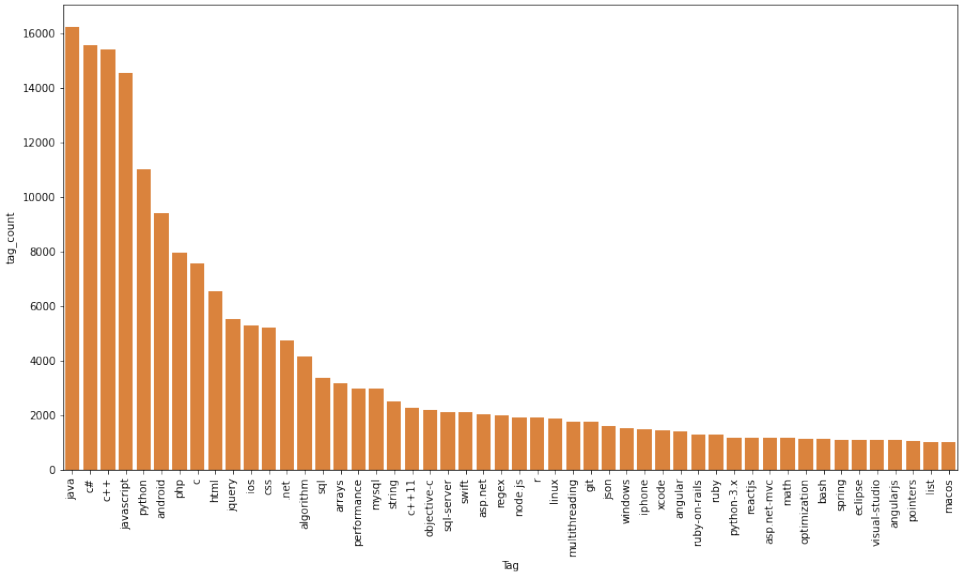

FIGURE 3.2 – 5O Tags les plus utilisés

Maintenant, regardons le nombre de tags par post. Nous pouvons remarquer que certains posts n'ont ainsi plus de tags. Ceux-ci seront supprimés car notre étude se base sur les tags. Nous ne pourrions pas comparé les résultats de nos algorithmes à la réalité.

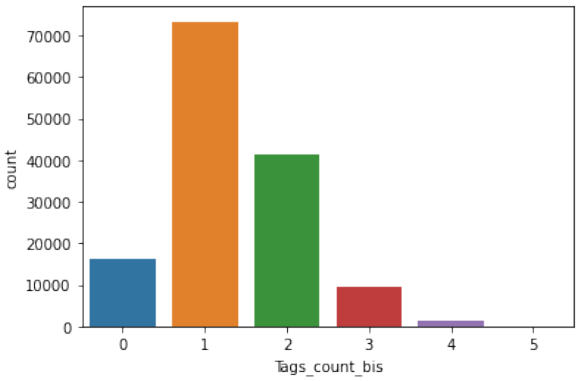

FIGURE 3.3 – 5O Tags les plus utilisés

3.2 Nettoyage du corpus

Maintenant que nous avons exploré les variables, focalisons nous sur le corpus. Deux variables composeront notre corpus final. Celles-ci sont le corps et le titre du post. Regardons la distribution des longueurs pour ces deux variables.

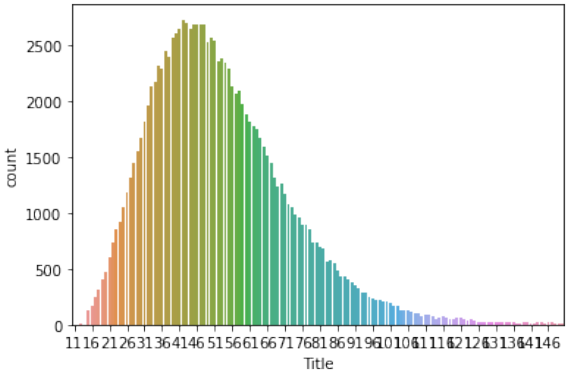

FIGURE 3.4 – Distribution de la longueur des titres

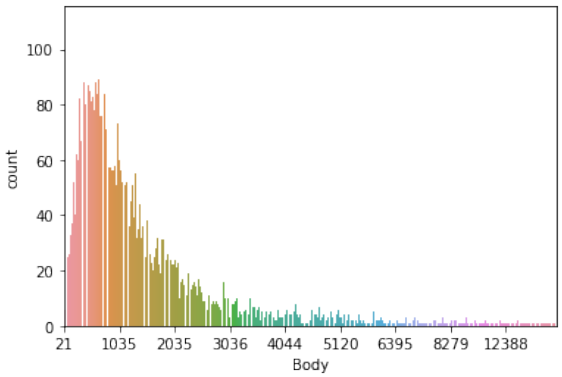

FIGURE 3.5 – Distribution de la longueur des posts

Sur la figure (3.4), la longueur des posts peut être très longue. La majorité ont des longueurs inférieures à 5 000 mais certains les dépassent très largement. Pour ne pas compliquer nos modèles, limitons la longueur des posts à 5000. Pour les titres, le pic de la distribution se trouve aux alentours de 46 caractères. Les plus longs se trouvent autour de 150 caractères. Vu la distribution sur la figure (3.5), il ne semble pas nécessaire de limiter la longueur des titres

Passons à la partie nettoyage de notre corpus. Après la récupération sur Stack Exchange, celui-ci est composé de beaucoup de termes non utiles, comme, par exemple, les balises html . Focalisons nous sur la suppression de tous ces contenus. Les différentes opérations effectuées sur le corpus sont :

— Suppression des bannières HTML
— Mettre tout le texte en minuscules
— Supprimer les caractères Unicode
— Suppression des espaces supplémentaires
— Suppression de la ponctuation
— Suppression des liens
— Supprimer les nombres
— Supprimer les Stopwords

Nous effectuerons également deux opérations. La première est de Tockeniser toutes les phrases de notre corpus. C'est à dire decouper les phrase en mots et dans la création d'une liste. Nous supprimerons également les Stop-words. Ce sont les mots dont le sens est vide et qui ne sont pas nécessaire dans notre étude. La deuxième est de Lemmatiser. Cette méthode consiste à prendre la racine des mots.

Regardons l'évolution de nos distribution après toutes ces opérations.

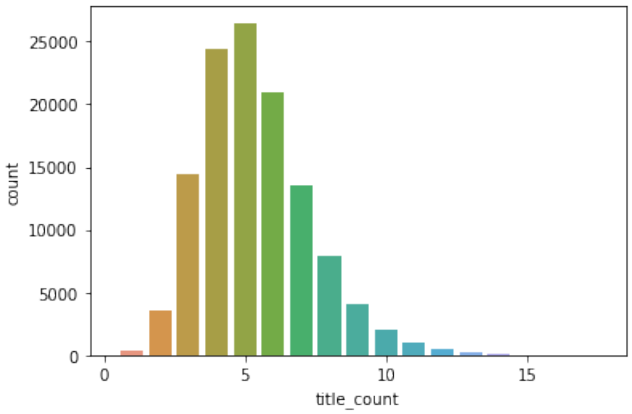

FIGURE 3.6 – Distribution de la longueur des Titres

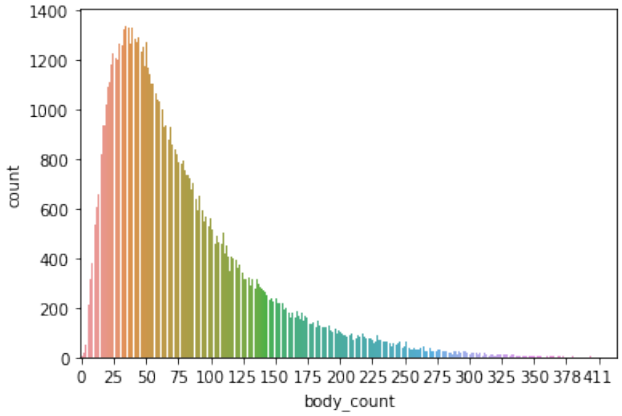

FIGURE 3.7 – Distribution de la longueur des posts

Après avoir nettoyé notre corpus, passons à la partie modélisation de la prédiction des tags

Chapitre 4

Modélisation

Dans cette partie, nous effectuerons deux types de modélisation différentes. L'une sera supervisée, tel que le Random Forrest Classifier par exemple. Mais également une méthode non supervisée. Celle-ci nécessitera de déterminer une méthode de prédiction de tags à partir des résultats obtenus.

4.1 Modèle non surpervisé

Commençons par la méthode non supervisée. C'est Latent Dirichlet Allocation. Que nous appellerons LDA dans la suite de notre projet. C'est un modèle probabiliste qui permet d'obtenir des clusters de posts. Pour cela, elle est basée sur plusieurs hypothèses. Chaque document du corpus est un ensemble de mots sans ordre. Chaque document aborde un certain nombre de thèmes dans différentes proportions qui lui sont propres. Chaque mot possède une distribution associée à chaque thème. On peut ainsi représenter chaque thème par une probabilité sur chaque mot.

Ainsi, la première étape est de déterminer le bon nombre de Topics. Pour cela, nous utiliserons la librairie gensim. Dans celle-ci, nous analyserons le Cohérence Score. C'est ce score qui va nous permettre de trouver le bon nombre de Topics. Nous réalisons une première routine entre 10 et 90 Topics et ensuite une deuxième pour trouver le bon nombre plus précisément.

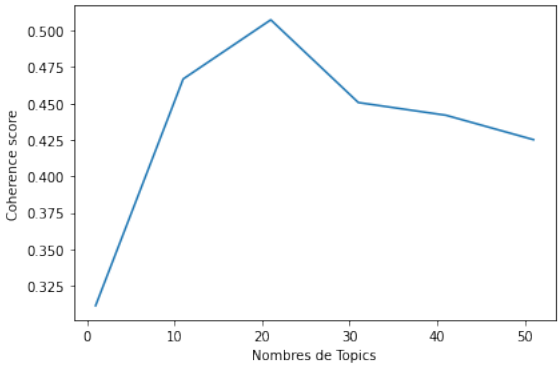

FIGURE 4.1 – Tracer du score de cohérence en fonction du nombre de Topics

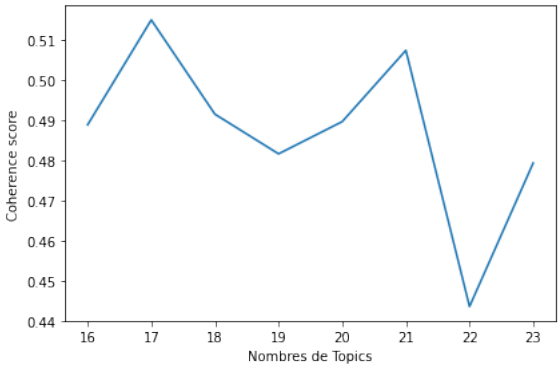

FIGURE 4.2 – Tracer du score de cohérence en fonction du nombre de Topics

Grâce à ces deux routines, nous obtenons que le score de cohérence optimal est de 17 Topics. Appliquons la LDA avec 17 Topics et regardons s'ils sont cohérents.

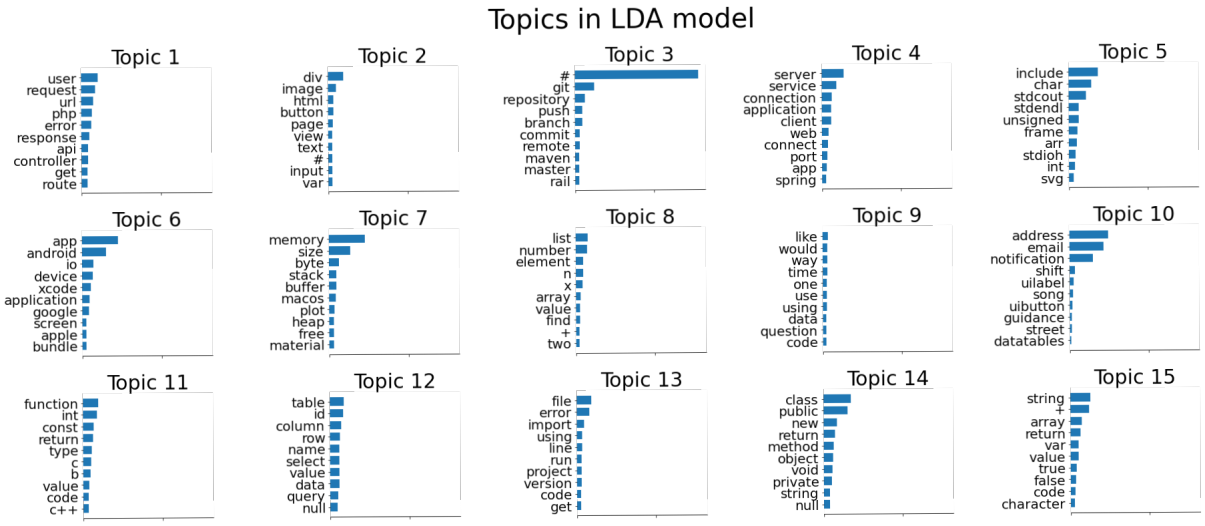

FIGURE 4.3 – Exemple de mots composant les 15 premiers Topics

Comme nous pouvons le voir sur la figure (4.3), certains Topics semblent cohérents. Comme le Topic 6 avec les mots app, application et Android. Le Topic 10 avec adresse, mail, notification. Passons à la méthode que nous utiliserons pour associer des tags à nos posts. La méthode LDA permet pour chaque post d'obtenir le Topics auquel il appartient. Nous sélectionnons les 100 mots avec les plus grandes probabilités pour chaque Topic et nous prenons l'intersection avec nos 50 Tags. Ainsi, nous aurons limité les mots des Topics à nos Tags. Nommés tags intermédiaires. Nous prendrons ensuite l'intersection des tags intermédiaires avec le corps de chaque post. Et ce sont les mots obtenus à partir de cette deuxième intersection qui seront nos prédictions de Tags pour chaque post. Nous pouvons maintenant passer aux performances de ce modèle.

	LDA
F1	0.24
Recall	0.18
Précision	0.51
Accuracy Moyenne	0.19
Accuracy Faible	0.28
Accuracy Forte	0.13

Nous choisissons de nous focaliser sur 6 métriques différentes. Le F1, Recall et Précision sont des scores déjà implantés dans sklearn. Mais, pour avoir une meilleure représentation, des accuracy ont été crée pour mieux rendre compte des performances du modèle. Comme nos modèles sont multi task, il semblait important de pouvoir voir les performances en fonction de cela. L'accuracy faible représente si au moins un parmi l'ensemble des tags réels est prédit. L'accuracy moyenne est la moyenne du nombre de tags prédits exacts sur le nombre de tags réels. Et l'accuracy forte représente si l'ensemble des tags réels ont été prédits.

Les performances de la LDA ne semblent pas très importantes mais cela pouvait être attendu. En effet, le modèle semble ne pas séparer convenablement les groupes. On peut voir que certain Topic avait des mots en commun. Passons maintenant aux modèles supervisés.

4.2 Modèle supervisé

Dans cette partie, nous nous focaliserons sur les modèles supervisés. Dans un premier temps, parlons des pre-processing effectués sur les tokens de notre corpus. Pour la variable X, composée du titre et des corps des posts, est transformée à l'aide de la routine sklearn Term-Frequency - Inverse Document Frequency. Cette routine va nous permettre de sélectionner les mots les plus importants et en supprimant ceux qui apparaissent trop souvent. Le but étant de sélectionner les mots les plus signifiants par post. Pour la variable Y, les tags, nous appliquerons un MultiLabelBinariser qui va créer une matrice avec 50 colonnes pour les tags et en ligne le nombre de posts. Pour chaque post, les tags de celui-ci seront représentés par un 1 dans la matrice. Maintenant les variables transformées, nous pouvons passer à la partie modélisation.

4.2.1 Logistic Regression

Pour la Logistic Regression, nous appliquerons un OneVsRestClassier en plus. Grâce à cette routine, nous aurons un modèle multi task. Ceci nous permettra de prédire plusieurs tags. Nous appliquerons une gridsearchcv pour trouver les meilleurs hyper-paramètres. Pour raccourcir le temps de calculs, nous utiliserons seulement 10 000 posts. Une fois les meilleurs paramètres trouvés, nous rappliquons une gridsearchcv avec toutes les données et tester notre modèle sur plusieurs set de trains et de validations. Une fois notre modèle sélectionné, nous obtenons les performances suivantes :

	LDA	Logistic
F1	0.24	0.65
Recall	0.18	0.57
Précision	0.51	0.76
Accuracy Moyenne	0.19	0.66
Accuracy Faible	0.28	0.77
Accuracy Forte	0.13	0.55

4.2.2 Random Forrest Classifier

Pour le Random Forrest Classifier, nous appliquerons un OneVsRestClassier en plus. Les performances sont meilleures. Nous appliquerons une gridsearchcv pour trouver les meilleurs hyper-paramètres. Pour raccourcir le temps de calculs, nous n'utiliserons seulement 10 000 posts. Une fois les meilleurs paramètres trouvés, nous rappliquons une gridsearchcv avec toutes les données et tester notre modèle sur plusieurs set de trains et de validations. Une fois notre modèle sélectionné, nous obtenons les performances suivantes :

	LDA	Logistic	RandomForrest
F1	0.24	0.65	0.65
Recall	0.18	0.57	0.79
Précision	0.51	0.76	0.56
Accuracy Moyenne	0.19	0.66	0.90
Accuracy Faible	0.28	0.77	0.92
Accuracy Forte	0.13	0.55	0.86

4.2.3 Multi-layer Perceptron classifier

Pour le Multi-layer Perceptron classifier, nous appliquerons directement une gridsearchcv pour trouver les meilleurs hyper-paramètres. Pour raccourcir le temps de calculs, nous n'utiliserons seulement 10 000 posts. Une fois les meilleurs paramètres trouvés, nous rappliquons une gridsearchcv avec toutes les données et tester notre modèle sur plusieurs set de trains et de validations. Une fois notre modèle sélectionné, nous obtenons les performances suivantes :

	LDA	Logistic	RandomForrest	MLP
F1	0.24	0.65	0.65	0.64
Recall	0.18	0.57	0.79	0.58
Précision	0.51	0.76	0.56	0.74
Accuracy Moyenne	0.19	0.66	0.90	0.73
Accuracy Faible	0.28	0.77	0.92	0.85
Accuracy Forte	0.13	0.55	0.86	0.60

4.3 Choix du modèle

Avec tous les scores pour chacun de nos modèles, nous pouvons sélectionner le modèle optimal. Selon, les différents tableaux, le meilleur semble être Random Forrest. Pour avoir une meilleur vision sur les scores, représentons les sur un graphique. Sur la figure (4.4), le Random Forrest semble toujours être le meilleur modèle.

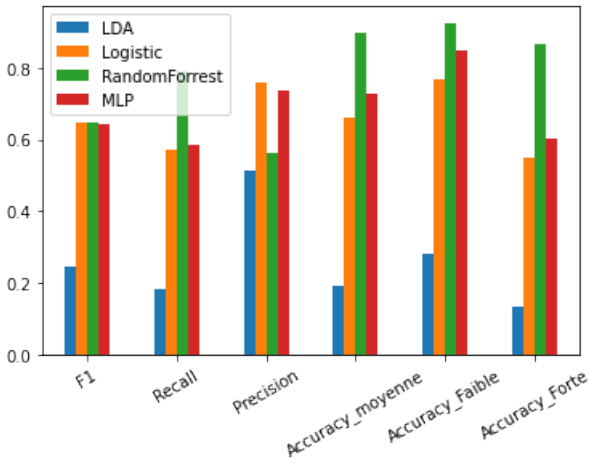

FIGURE 4.4 – Histogramme des différents scores en fonction des modèles sur les sets de train

Pour avoir un petit complément sur les performances des modèles, nous sélectionnons aléatoirement des posts pour faire un set de tests. Ceux-ci n'ont pas été utilisé pour l'entraînement des algorithmes. Nous regroupons les résultats dans le graphique (4.5). Celui-ci nous permet de confirmer que le meilleurs pour la prédiction de tags est le modèle Random Forrest.

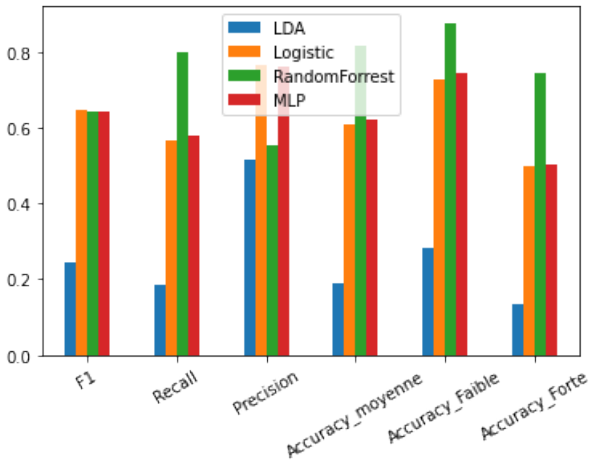

FIGURE 4.5 – Histogramme des différents scores en fonction des modèles sur les sets de tests

Chapitre 5

Conclusion

Après avoir appliqué nos différents modèles sur les données recupérées. Il semble que le modèle avec les meilleures performances soit le modèle Random Forrest Classifier. Celui-ci a les meilleurs score déjà implantés dans Sklearn mais surtout sur ceux que nous avons crées et dont nous maîtrisons les calculs. Celui-ci semble avoir de trés bonnes performances. Même sur des données de tests, les performances sont plus proche de 1. A la fin de nos Notebook, il y a une classe qui permet de retourner les prédictions de notre algorithme pour tester celui-ci.

Dans ce projet, nous avons eu pour la première fois affaire à des données textuelles. Celle-ci, nous ont permis d'apprendre les méthodes pour traiter et nettoyer notre corpus. Ce projet nous a également permis, d'apprendre le SQL pour pouvoir récupérer les données sur le site Stack Exchange.

Nous aurons appris à nous servir de GitHub pour pouvoir poster nos codes dans le temps. Et avoir en mémoire nos différentes versions si nous avions eu un problème à un moment donné et que nous voulions revenir en arrière.

Ce projet aura été également l'occasion de créer une API. Celle-ci permet de tester notre algorithme pour toutes questions. Cette API a été réalisé avec Fast API et Heroku pour pouvoir la mettre en ligne. Un répertoire git avec cette API est aussi disponible.