Master 1 Informatique Parcours Ico

Rapport de projet Machine Learning

HAI822I
Classification d'assertions selon leur valeurs de véracité

Groupe de travail :
Aicha Kimouche
Hanane Benarab
Hamza Raihane
Safae Terroufi

Année Universitaire :2021/2022

Sommaire

1 Introduction

Ce projet est consacré au module Machine Learning, il a pour but de proposer un modèle d'apprentissage automatique supervisé pour la prédiction des fake news dans le cadre du Fact-checking.

Nous disposons des données labellisé extraites sur le site ClaimsKG qui fournit des données sur les affirmations, des métadonnées (telles que leur site de publication), les entités impliquées (annotées à l'aide de techniques NLP) et certaines évaluations de vérité normalisées.

Dans ce rapport, nous aborderons les grandes étapes qui nous ont permis de proposer un modèle capable de détecter la véracité d'une claims ("True", "False", "Mixture"), en commençons par le prétraitement, ensuite la classification et finalement l'optimisation.

2 Conception et Implementation du système

Le système prend en entrée un fichier CSV de données extraites du site web ClaimKG et les transforme en une base utilisable par la phase d'apprentissage. Cette transformation est appelée pré-traitement.

2.1 Pré-traitements des données

Le pré-traitement des données brute, est une étape primordiale dans le traitement de langage naturel, la qualité des données doit être vérifiée avant l'usage d'algorithmes d'apprentissage automatique.

Nous avons effectué les étapes suivantes pour le prétraitement des données :

1-Nettoyage : Qui consiste à éliminer les mots vides tel que "a", "about", am", "you", "are", "it"...ext et les caractères spéciaux tel que ! ? : ; , ...et toute information non utile.

2-Tokenisation : C'est un moyen de séparer un morceau de texte en unités plus petites appelées jetons. Il consiste à identifier les unités de textes élémentaires qui peuvent être des mots, mais aussi des lettres, des syllabes, des phrases, ou des séquences de ces éléments.

3-Stemming : Qui consiste à réduire un mot dans sa forme « racine ». Le but du stemming est de regrouper de nombreuses variantes d'un mot comme un seul et même mot.

4-Lemmatisation du texte : Il s'agit de la réduction de mots et considère le vocabulaire complet d'une langue pour appliquer une analyse morphologique aux mots, visant à supprimer uniquement les fins flexionnelles et à renvoyer la forme de base ou de dictionnaire d'un mot.

Afin d'y appliquer des algorithmes d'apprentissage automatique, nous devons convertir les fichiers texte en vecteurs de caractéristiques numériques. Pour ce faire, nous avons utilisé le modèle bag-of-word qui compte le nombre de fois où chaque mot apparaît dans chaque texte et enfin à l'aide de TF-ID pour obtenir les fréquences pondérées.

En visualisant les données du DataSet, nous nous somme rendu compte qu'il y'a un déséquilibre au niveau de nos classes comme le démontre la figure suivante :

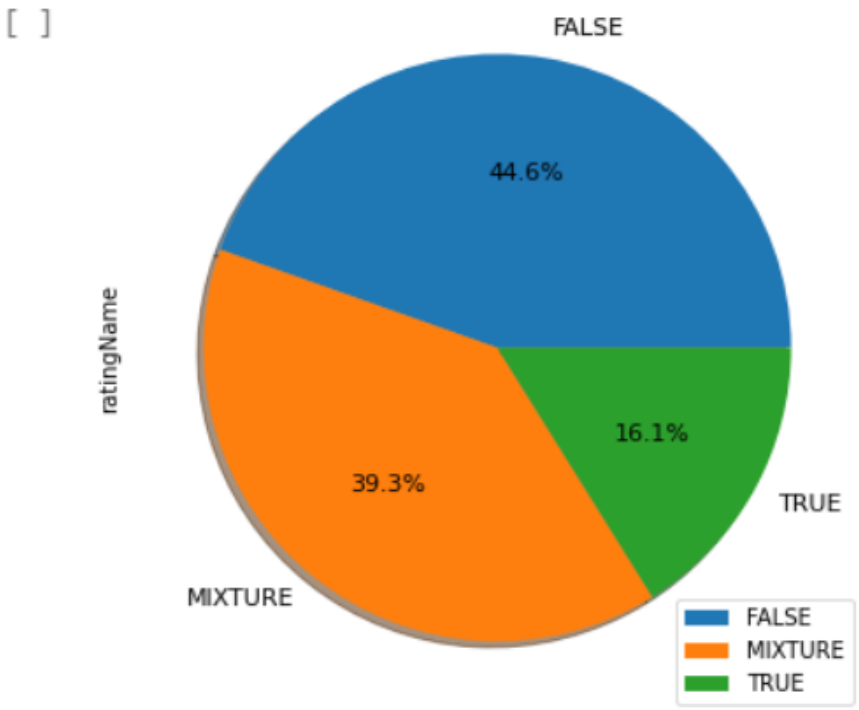

FIGURE 1 – Visualisation des données du dataSet

Cela entraînera effectivement une mauvaise précision au niveau des modèles qu'on va construire, mais il existe des solutions afin d'y remédier. Nous allons démontrer l'importance d'avoir des données équilibrées pour obtenir une précision optimale au niveau des modèles dans la prochaine section.

2.2 Classification :

Après le prétraitement, La base pré-traitée est subdivisée en deux parties ; une pour l'entraînement et l'autre pour le test. Le module d'entraînement utilise la base d'entraînement et un algorithme d'apprentissage pour fournir un modèle de décision qui est appliqué sur la base de test.

2.3 Premier classifieur

Pour la classification, nous avons utilisé en premier Logistic Regression pour observer la matrice de confusion et l'impact de déséquilibre des classes au niveau de la précision.

→ **True vs False Vs Mixture :**

→ **Logistic Regression :**

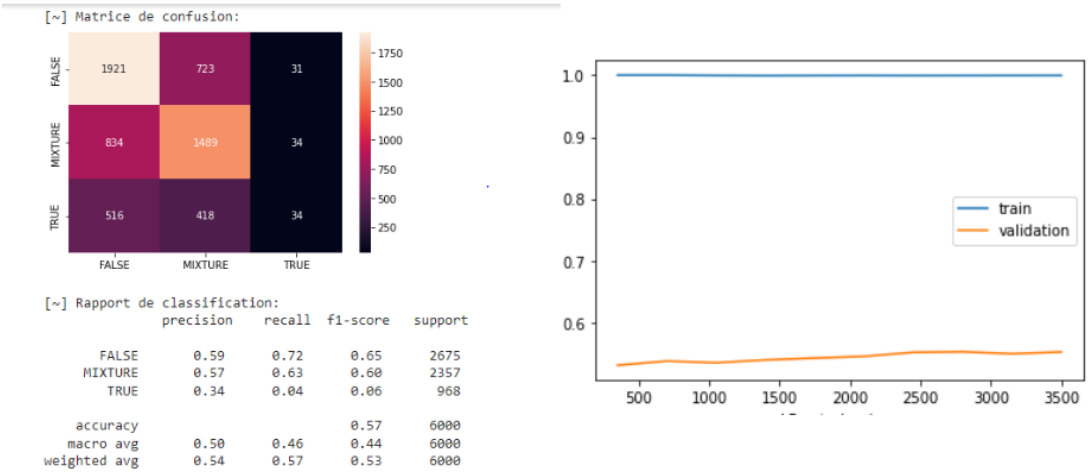

FIGURE 2 – Matrice de confusion + Graphe de l'evolution du train et validation score

On voit clairement au niveau des résultats que le classifier arrive à classer les False ainsi que les Mixture avec plus de 60%de précision, tandis que pour

la classe true seulement 4% cela s'explique en raison du manque de données pour la classe True.

On observe également dans la figure 3, un phénomène d'Overfitting.

2.4 Sur échantillonnage

Pour pallier le problème de déséquilibre au niveau des classes, nous utiliserons le suréchantillonnage (Up sampling), une technique qui nous permettra d'avoir un équilibre au niveau des classes ce qui signifie qu'on rajoutera des classes True pour s'adapter à la classe False.

→True Vs False :

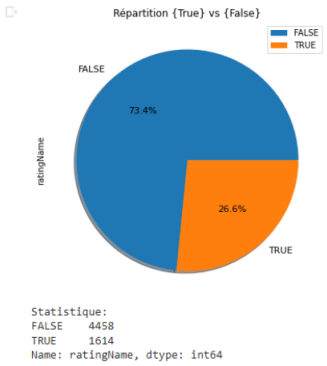

FIGURE 3 – Répartition des classes avant le UpSimpling

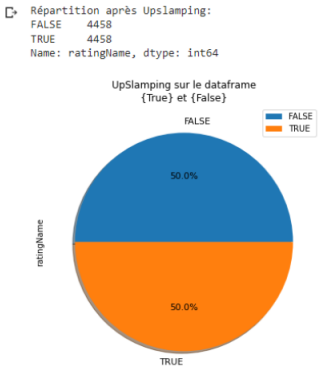

FIGURE 4 – Répartition des classes aprés le UpSimpling

On voit que le nombre de la classe True est passé de 1614 à 4458, qui sera donc égale a la classe False.

→**True Vs False Vs Mixture :**

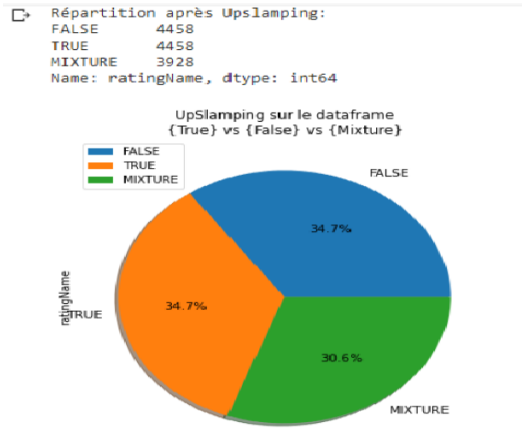

FIGURE 5 – Répartition de True vs False Vs Mixture aprés Upsampling

Appliquons maintenant le classifieur :

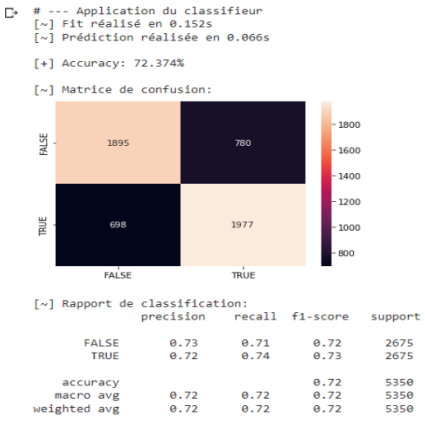

FIGURE 6 – Matrice de Confusion clf logistic Regression

Nous constatons une amélioration au niveau de l'accuracy et également au niveau de la précision des deux classes.

3 Optimisation

3.1 Sélection des features

Après avoir effectué différents essaies afin d'optimiser nos résultats en choisissant les bons features, nous avons pu obtenir le profits de chaque ajout d'une données additionnelle

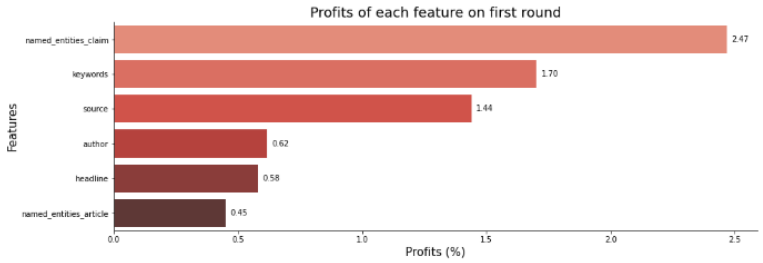

FIGURE 7 – Profits de chaque ajout d'une données additionnelle

Toutes les données sembles utiles à ajouter. Sauf named-entities-article qui est très faible et author qui ont un profit trés faible.

Nous avons eu le même constat en effectuant True Vs False Vs Mixture.

3.2 GridSearch

Afin d'optimiser nos résultats en choisissant les bon hyper-paramètres qui s'adaptent le plus à nos classifiers pour avoir une meilleure performance pour notre modèle, nous avons appliqué le GridSearch.

→Logistic Regression :

```
[+] Accuracy: 76.243%
[+] Meilleurs paramètres:  {'clf__C': 2, 'clf__max_iter': 1000, 'clf__solver': 'liblinear'}
```

→K-nearest neighbors

```
[+] Accuracy: 61.570%
[+] Meilleurs paramètres:  {'clf__leaf_size': 2, 'clf__n_neighbors': 5, 'clf__weights': 'distance'}
```

→Arbre de décision :

```
[+] Accuracy: 61.570%
[+] Meilleurs paramètres:  {'clf__leaf_size': 2, 'clf__n_neighbors': 5, 'clf__weights': 'distance'}
```

Après avoir effectué l'optimisation, le meilleur classifier est RandomForest avec une accuracy de 82.551 %, il suit Logistic Regression avec 76,24 %.

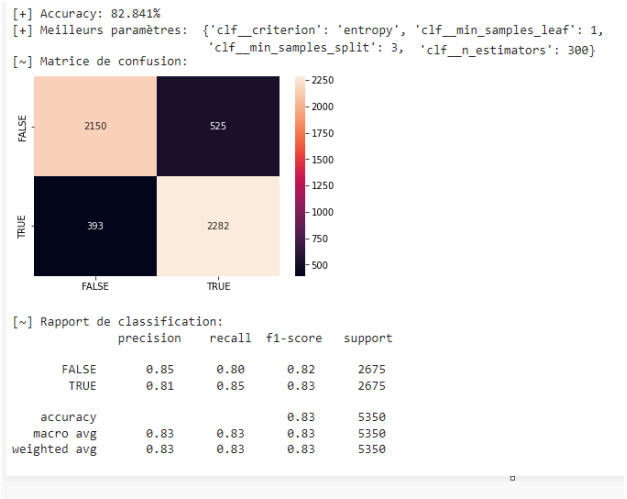

FIGURE 8 – Matrice de confusion du classifier RandomForest

4 Conclusion

Grâce à ce projet, nous avons pu mettre en œuvre un modèle qui prédit la véracité des claims dans le cadre du Fact Checking, nous avons pu constater à travers nos résultats l'importance d'avoir des classes équilibrées pour avoir des résultats satisfaisants. Nous avons pu démontrer que grâce à l'optimisation (Avec GridSearch qui nous a permit de choisir les hyper-parametres les mieux adapté a notre classifier, la sélection des Features qui nous a permit de voir l'impact de l'ajout des données additionelles sur le modéle), on arrive à améliorer la précision de notre modéle.

Finalement, Après avoir effectué le prétraitement, la classification et l'optimisation nous avons pu obtenir un modèle avec une précision de 82.551% avec le classifier RandomForest.