

RAPPORT DE PROJET

MACHINE LEARNING - HAI817

Classification d'assertions selon leur valeurs de véracité

automatic fact-checking

Temouden Kaotar : 21717901

Lehouaoui Sara : 21807688

M1 IASD

Année 2021/2022

Sommaire

1	Introduction	2
2	Exploration des données	3
3	Ingénierie des données	4
3.1	Pré-traitements des données textuelles et vécotorisation	4
3.2	Utilisation des pipelines	4
4	Les tâches de classification	5
4.1	Classification binaire	5
4.1.1	TRUE vs FALSE	5
4.1.2	TRUE or FALSE vs MIXTURE	7
4.2	Classification multiclasse : TRUE vs FALSE vs MIXTURE	9
5	Conclusion	11

Partie 1

Introduction

Les fake news est un phénomène qui a un impact significatif sur notre vie sociale. On retrouve souvent ces fausses assertions et rumeurs sur les réseaux sociaux (Facebook, Twitter). La détection des fake news est un problème complexe, étant donné que les humains ont tendance à croire ces fausses assertions. Notre but dans ce projet sera de proposer une méthode automatique utilisant des modèles d'apprentissage supervisés, en traitant le problème comme une classification.

Partie 2

Exploration des données

Pour ce projet on a utilisé le jeu de données ClaimsKG. Ce dataset contient des assertions (son texte et son titre) et un label qui donne la véracité de celles-ci :

- **TRUE**
- **FALSE**
- **MIXTURE** lorsque les sources sont divisées sur la véracité de cette affirmation
- **OTHER** lorsqu'on n'a pas d'informations sur la véracité de l'assertion.

D'autres méta-données existent comme l'auteur, la date, l'article, les mots-clés ...
On a pu récupérer 10000 assertions dont la répartition des labels est la suivante :

- **TRUE** : 1285
- **FALSE** : 3552
- **MIXTURE** : 3144
- **OTHER** : 2019

Afin d'avoir un échantillon équilibré et un temps d'exécution raisonnable lors des différents tests, on a décidé de travailler avec 300 assertions des classes True/False et 600 de la classe Mixture.

Pour visualiser la répartition des différentes classes (TRUE, dans l'espace 2D nous avons appliqué une PCA sur la vectorisation TF-IDF du champs **text** :

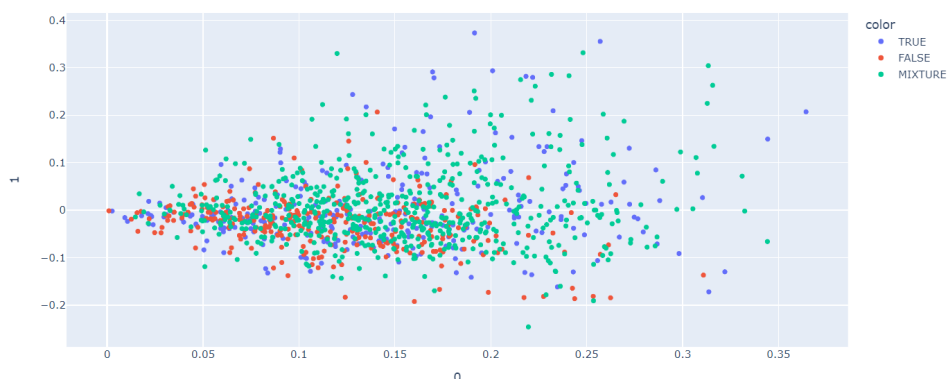


FIGURE 2.1 –

On remarque que la projection sur les deux axes ne permet pas de séparer les différentes classes. D'où l'intérêt d'une analyse plus approfondie exploitant les différentes features.

Partie 3

Ingénierie des données

Pour la suite du projet et après une discussion avec le prof, on a décidé de n'exploiter que les champs **text** et **headline** puisqu'ils sont riches et définissent bien l'assertion. On ne s'est pas aux données catégorielles pour simplifier le modèle.

3.1 Pré-traitements des données textuelles et vectorisation

Pour utiliser ces données sous forme de texte pour notre tâche de classification, il a fallu :

- Appliquer des prétraitements NLP pour nettoyer le texte et le normaliser. Pour cela, on a repris les différents traitements vus en cours/tp et on a utilisé les estimateurs/transformer pour encapsuler cette brique de pré-traitement.
- Générer des features à partir des différents éléments texte. Pour cela on a utilisé la vectorisation TF-IDF qui va transformer notre texte en un vecteur (numérique) qui sera utilisé ensuite pour la modélisation.

3.2 Utilisation des pipelines

On a utilisé les pipelines pour tester plusieurs configurations : la combinaison de différents traitements avec la vectorisation TF-IDF.

Parmi les pipelines créés, on cite les exemples suivants :

- **TEXTALL** utiliser uniquement le champs "text" et application de tous les prétraitements (lowercase=True, removestopwords=True, removedigit=True, getstemmer=True), avec TF-IDF limitée à 1000 features.
- **HEADLINE_NoSTEMMER** utiliser uniquement le titre de l'assertion, et ne pas appliquer le stemming, avec TF-IDF limitée à 1000 features.
- **HEADLINEALLFEATURES** utiliser uniquement le titre de l'assertion, application de tous les prétraitements (lowercase=True, removestopwords=True, removedigit=True, getstemmer=True) avec TF-IDF non limitée.
- **HEADLINE_TEXT** combiner les features calculées à partir des pipelines **textAll** et **headlineAll**.

Partie 4

Les tâches de classification

On a testé principalement trois classifieurs : **SVC** , **RandomForestClassifier** et **LogisticRegression**.

Pour évaluer la performance et la robustesse de nos modèles, on a effectué une validation croisée (type k-folds), en utilisant les métriques suivantes :

- **ACCURACY** : mesurer le taux de prédictions correctes sur l'ensemble des individus.
- **F1 SCORE** : permet de résumer les valeurs de précision et du recall en une seule métrique. Il est particulièrement utilisé pour les problèmes utilisant des données déséquilibrées comme la détection de fraudes ou la prédiction d'incidents graves.

4.1 Classification binaire


On s'est intéressé dans la classification binaire à deux tâches :

- **TRUE vs FALSE**
- **TRUE or FALSE vs MIXTURE**

4.1.1 TRUE vs FALSE

Dans cette tâche, on a utilisé la variable cible `is_fake` qui est égale à 1 quand le label de l'assertion est FALSE et 0 sinon.

La figure 4.1 ci-dessous est un tableau récapitulatif des résultats obtenus :



	model	f1_score	acc_score
16	RF_headlineAllFeatures	0.683622	0.605000
7	RF_text_NoStemmer	0.638629	0.586667
3	SVC_TextAllFeatures	0.638319	0.611667
4	RF_TextAllFeatures	0.635544	0.585000
10	RF_text_Basique	0.629075	0.595000
15	SVC_headlineAllFeatures	0.628499	0.595000
19	RF_headline_NoStemmer	0.626335	0.598333
20	LR_headline_NoStemmer	0.622631	0.606667
9	SVC_text_Basique	0.622009	0.625000
11	LR_text_Basique	0.619289	0.616667
23	LR_headline_Basique	0.610432	0.616667
18	SVC_headline_NoStemmer	0.610301	0.591667
5	LR_TextAllFeatures	0.604827	0.598333
17	LR_headlineAllFeatures	0.603198	0.593333
8	LR_text_NoStemmer	0.599246	0.603333
2	LR_textAll	0.596816	0.600000




FIGURE 4.1 –

Si on effectue une analyse par f1 score, on remarque que le classifieur **RandomForest** est le plus performant (le plus présent dans le top 5).

En particulier, le modèle **headlineAllFeatures** utilisant le titre de l’assertion donne le meilleur résultat.

Ce-ci est cohérent car le titre résume mieux l’information de l’assertion, on remarque aussi que l’utilisation de la totalité des features TF-IDF permet d’avoir un boost significatif.

En ce qui concerne l’analyse de l’importance des features, on a utilisé la méthode *feature_importances_* du classifieur **RandomForest** .

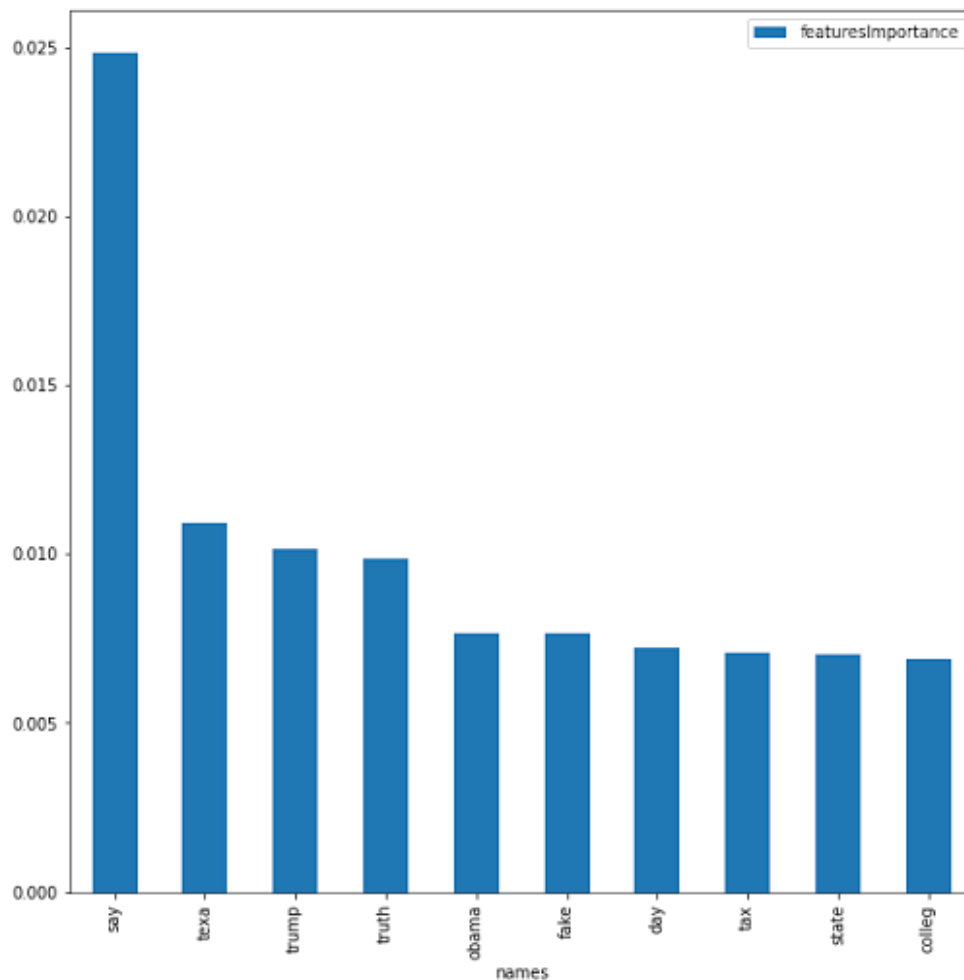



FIGURE 4.2 – Features importance analysis graph

Cette analyse permet de conclure quant aux termes qui assistent le plus dans sa discrimination : on remarque la présence du terme “trump” qui est potentiellement associé aux fake news. Le terme “say” a également une grande importance, on peut dire que les assertions qui contiennent des citations sont plus simples à discriminer.

4.1.2 TRUE or FALSE vs MIXTURE

Dans cette tâche, on a utilisé la variable cible `is_mixture` qui est égale à 1 quand le label de l’assertion est MIXTURE et 0 sinon.

La figure 4.3 ci-dessous est un tableau récapitulatif des résultats obtenus :



	model	f1_score	acc_score
24	SVC_headline_text	0.668404	0.659167
26	LR_headline_text	0.647468	0.646667
12	SVC_headlineAll	0.645106	0.632500
6	SVC_text_NoStemmer	0.643515	0.632500
9	SVC_text_Basique	0.630382	0.615833
18	SVC_headline_NoStemmer	0.630291	0.614167
25	RF_headline_text	0.628177	0.620000
13	RF_headlineAll	0.627248	0.635833
0	SVC_textAll	0.626240	0.622500
15	SVC_headlineAllFeatures	0.624408	0.640833
3	SVC_TextAllFeatures	0.623494	0.640000
11	LR_text_Basique	0.622550	0.611667
5	LR_TextAllFeatures	0.622330	0.631667
14	LR_headlineAll	0.622208	0.626667
20	LR_headline_NoStemmer	0.621009	0.626667
17	LR_headlineAllFeatures	0.619538	0.632500
2	LR_textAll	0.617855	0.620000




FIGURE 4.3 –

Dans ce cas, on peut conclure à partir d'une analyse par f1 score, que le classifieur **SVC** est le plus performant (présent 4 fois dans le top 5 meilleurs résultats) ce qui est raisonnable sachant qu'avec SVC on obtient souvent de bons résultats sur les données textuelles. En particulier, le modèle SVC_headline_text utilisant une combinaison du **text** et du titre de l'assertion **headline** donne le meilleur résultat, ce qui est logique puisqu'on utilise le maximum d'informations.

En ce qui concerne l'analyse de l'importance des features, la figure 4.4 résume les résultats obtenus :

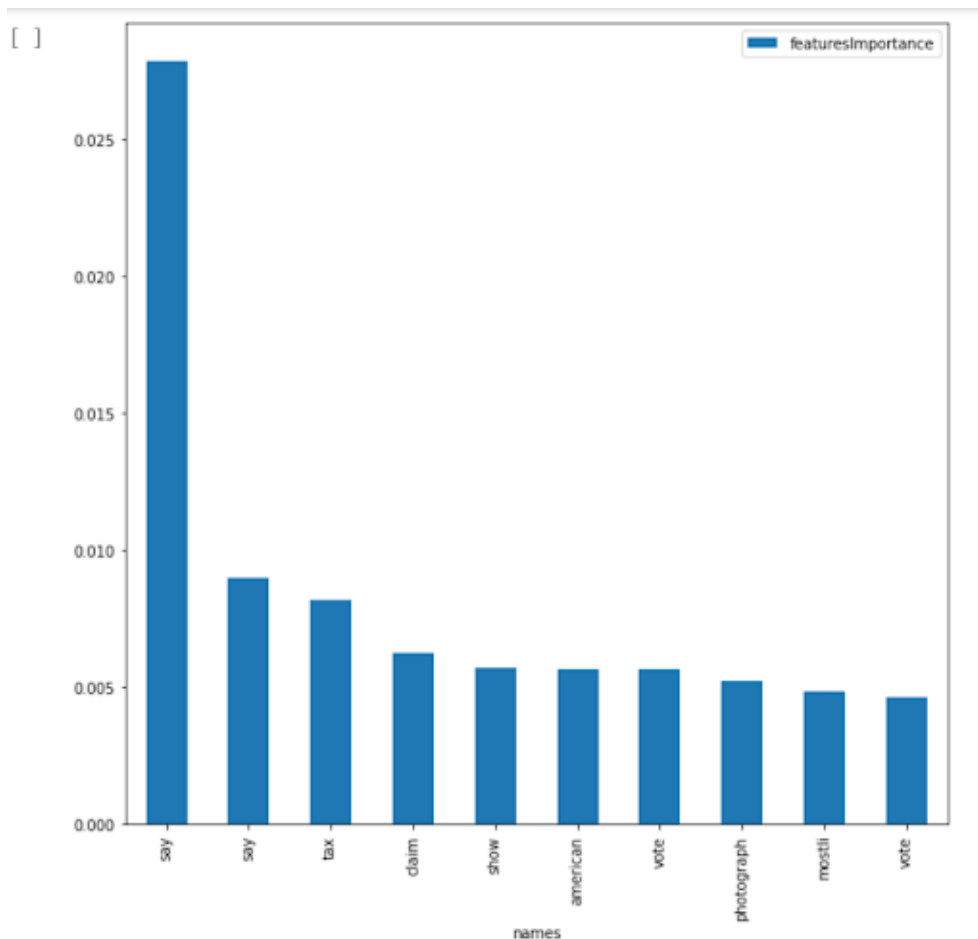


FIGURE 4.4 – Features importance analysis graph

On remarque que la présence du terme “say” a une grande importance dans la prise de décision du modèle.

4.2 Classification multiclasse : TRUE vs FALSE vs MIXTURE

Pour cette tâche, on a utilisé la variable **ratingName** comme variable cible. En ce qui concerne les modèles testés, on a ajouté l’algorithme **KNN** (*k-nearest neighbors classifier*) souvent utilisé pour le clustering ainsi que le classifieur **Naïve Bayes**.

Pour les métriques d’évaluation, on a utilisé la pondération du score f1 : **f1_weighted**

Les figures 4.5 et 4.6 ci-dessous récapitulent les résultats obtenus :

	model	f1_score	acc_score
30	SVC_headline_Basique	0.458303	0.462222
29	RF_headline_Basique	0.457109	0.456667
34	SVC_headline_text	0.447504	0.448889
18	SVC_headlineAll	0.443782	0.446667
6	SVC_TextAllFeatures	0.443402	0.451111
14	SVC_text_Basique	0.441992	0.443333
26	SVC_headline_NoStemmer	0.441366	0.446667
25	RF_headline_NoStemmer	0.437015	0.455556
22	SVC_headlineAllFeatures	0.435753	0.444444
17	RF_headlineAll	0.435325	0.444444
28	GNB_headline_Basique	0.434790	0.437778
2	SVC_textAll	0.428826	0.430000
33	RF_headline_text	0.428166	0.415556
32	GNB_headline_text	0.424363	0.424444
10	SVC_text_NoStemmer	0.417783	0.417778
24	GNB_headline_NoStemmer	0.416880	0.418889
21	RF_headlineAllFeatures	0.414784	0.462222
9	RF text NoStemmer	0.414569	0.428889

FIGURE 4.5 –

13	RF_text_Basique	0.404297	0.403333
16	GNB_headlineAll	0.401955	0.404444
5	RF_TextAllFeatures	0.401162	0.420000
0	GNB_textAll	0.391565	0.395556
8	GNB_text_NoStemmer	0.387480	0.388889
12	GNB_text_Basique	0.383399	0.385556
1	RF_textAll	0.381805	0.432222
4	GNB_TextAllFeatures	0.379206	0.384444
23	KNN_headlineAllFeatures	0.379113	0.381111
20	GNB_headlineAllFeatures	0.374724	0.376667
7	KNN_TextAllFeatures	0.354446	0.356667
15	KNN_text_Basique	0.306916	0.366667
35	KNN_headline_text	0.276132	0.338889
3	KNN_textAll	0.216716	0.335556
31	KNN_headline_Basique	0.210341	0.346667

FIGURE 4.6 –

D’abord, on remarque que les scores sont dégradés par rapport à la *classification binaire* (c’est souvent le cas). Le modèle **SVC** est le plus performant et robuste. En contrepartie, le modèle **KNN** donne des résultats décevants.

Partie 5

Conclusion

Ce projet nous a permis de tester différents modèles de classification pour la détection des fake news en utilisant une analyse statistique TF-IDF du texte.

Les modèles les plus performants sont **SVC** et le modèle de forêts aléatoires **Random forest** avec une accuracy de 60-65% pour la classification binaire, et 48% pour le multiclass.

Ces performances peuvent être améliorées en explorant les pistes suivantes :

- **UNE INGÉNIERIE DE DONNÉES PLUS APPROFONDIE** jouer sur les paramètres de TF-IDF, et tester d'autres vécotorisations en utilisant des modèles de *deep learning* type *BERT* .
- **TESTER DES ARCHITECTURES DE RÉSEAUX DE NEURONES** avec des concepts comme *LSTM* par exemple.