



Rapport de mini projet

Fake News Detection Using Machine Learning



Réalisé par :

Ben Hamou Mehdi

Encadré par :

Mr. EL Aroussi EL Mehdi

Sommaire

Chapitre 1 : Théorique

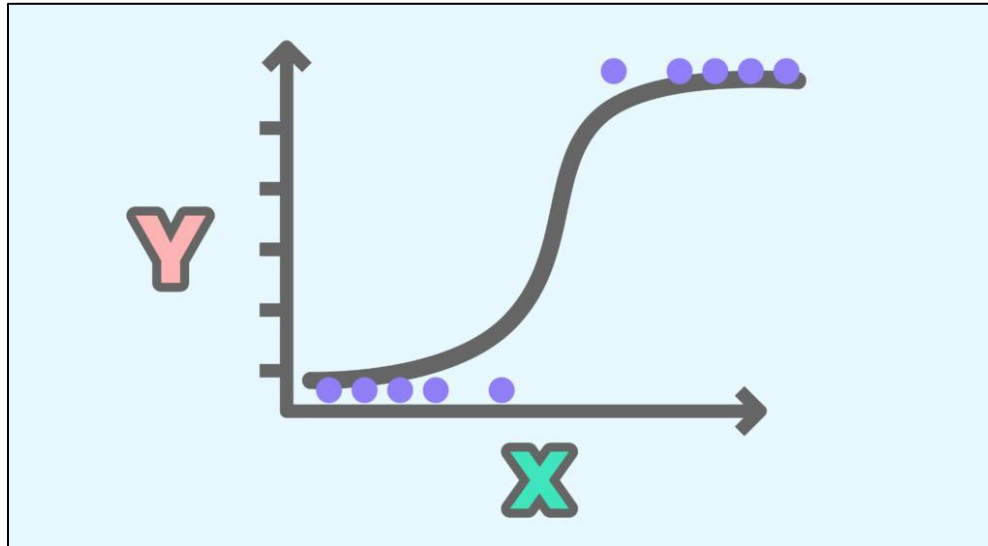
| | |
|--|---|
| 1. Régression Logistique..... | 3 |
| 2. RFC (Random Forest Classifier)..... | 5 |

Chapitre 2 : Pratique

| | |
|--|-----------|
| 1 Analyse exploratoire des données (AED) | 7 |
| 1.1 La recherche de l'ensemble de données | 7 |
| 1.2 Les bibliothèques utilisées..... | 7 |
| 1.3 Visualisation des données et Analyse de répartition | 8 |
| 1.4 Gestion des données manquantes..... | 9 |
| 1.5 Segmentation des données..... | 10 |
| 1.6 Détection des valeurs aberrantes | 11 |
| 1.7 Standardisation des données | 12 |
| 2 Choix de l'algorithme d'apprentissage automatique | 12 |
| 2.1 Sélection des algorithmes | 12 |
| 2.2 Diviser les données..... | 12 |
| 2.3 Entraînement et évaluation | 12 |
| 2.4 Comparaison des performances..... | 14 |
| 2.5 Choix du meilleur modèle..... | 15 |
| 2.6 Évaluation sur l'ensemble de test | 15 |
| 3 Interface API | 16 |
| 4 Index HTML | 18 |
| 5 Java Script..... | 19 |
| 6 CSS | 20 |
| 7 Test du mini projet..... | 22 |

Chapitre 1 : Théorique

1. Régression Logistique



Si vous vous intéressez un tant soit peu au Machine Learning et aux problèmes de classification, vous avez déjà dû avoir affaire au modèle de régression logistique. Et pour cause ! Il s'agit d'un des modèles de Machine Learning les plus simples et interprétables qui existe, prend des données à la fois continues ou discrètes, et les résultats obtenus avec sont loin d'être risibles. Mais que se cache-t'il derrière cette méthode miracle ? Et surtout comment l'utiliser sur Python ? La réponse dans cet article

Qu'est-ce que la régression logistique ?

La régression logistique est un modèle statistique permettant d'étudier les relations entre un ensemble de variables qualitatives X_i et une variable qualitative Y . Il s'agit d'un modèle linéaire généralisé utilisant une fonction logistique comme fonction de lien.

Un modèle de régression logistique permet aussi de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Ce résultat varie toujours entre 0 et 1. Lorsque la valeur prédite est supérieure à un seuil, l'événement est susceptible de se produire, alors que lorsque cette valeur est inférieure au même seuil, il ne l'est pas.

Mathématiquement, comment ça se traduit / ça s'écrit ?

Considérons une entrée $X = x_1 x_2 x_3 \dots x_n$, la régression logistique a pour objectif de trouver une fonction h telle que nous puissions calculer :

$$y = \{1 \text{ si } hX \geq \text{seuil}, 0 \text{ si } hX < \text{seuil}\}$$

On comprend donc qu'on attend de notre fonction h qu'elle soit une probabilité comprise entre 0 et 1, paramétrée par $\theta_1, \theta_2, \dots, \theta_n$ à optimiser, et que le seuil que nous définissons correspond à notre critère de classification, généralement il est pris comme valant 0.5.

La fonction qui remplit le mieux ces conditions est la fonction sigmoïde, définie sur \mathbb{R} à valeurs dans $[0,1]$. Elle s'écrit de la manière suivante :

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Et notre classification dans tout ça ?

La fonction h qui définit la régression logistique s'écrit alors :

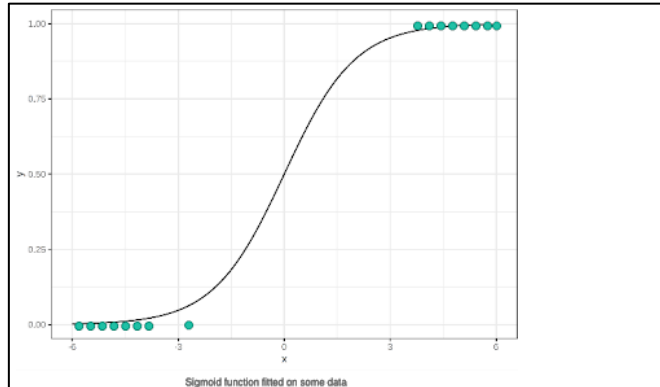
$$\forall (X \in \mathbb{R}^n) \quad h(X) = \sigma(\Theta X)$$

i.e.

$$\forall (X \in \mathbb{R}^n) \quad h(X) = \frac{1}{1 + e^{-\sum_{i=1}^n \theta_i x_i}}$$

Tout le problème de classification par régression logistique apparaît alors comme un simple problème d'optimisation où, à partir de données, nous essayons d'obtenir le meilleur jeu de paramètre Θ permettant à notre courbe sigmoïde de coller au mieux aux données. C'est dans cette étape qu'intervient notre apprentissage automatique.

Une fois cette étape effectuée, voici un aperçu du résultat qu'on peut obtenir:



Il ne reste plus, à partir du seuil défini, qu'à classer les points en fonction de leurs positions par rapport à la régression et notre classification est faite !

La régression logistique en pratique

En Python c'est assez simple, on se sert de la classe `LogisticRegression` du module `sklearn.linear_model` comme un classificateur normal et que l'on entraîne sur des données déjà nettoyées et séparées en ensembles d'entraînement et de test puis le tour est joué !

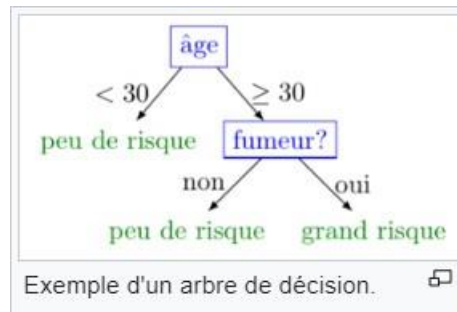
Niveau code, rien de plus basique :

```
1 from sklearn.linear_model import LogisticRegression
2 clf = LogisticRegression()
3 clf.fit(X_train,y_train)
4 y_pred = clf.predict(X_test)
```

2. RFC (Random Forest Classifier)

En intelligence artificielle, plus précisément en apprentissage automatique, les forêts d'arbres décisionnels¹ (ou forêts aléatoires de l'anglais `random forest classifier`) forment une technique d'apprentissage à base d'arbres de décision. Elle apprend plusieurs arbres de décision et de ce fait, elle fait partie des méthodes d'apprentissage ensembliste, c'est-à-dire des méthodes qui utilisent la sagesse des foules². Ils ont été premièrement proposées par Ho en 1995³ et ont été formellement proposées en 2001 par Leo Breiman⁴ et Adele Cutler⁵. Cet algorithme combine les concepts de bagging (méthodes ensemblistes parallèles²) pour la phase de sélection des données, et de sous-espaces aléatoires. L'algorithme des forêts d'arbres décisionnels effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents

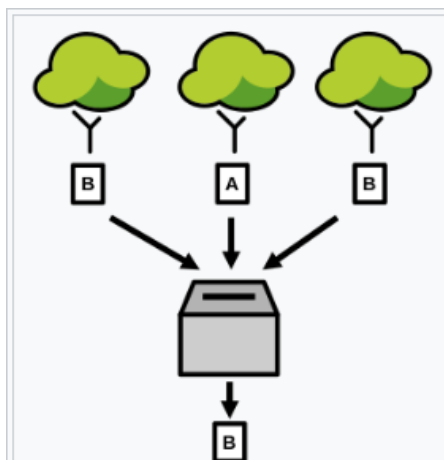
Un arbre décisionnel ou arbre de décision est une structure qui prend la forme d'un arbre, dans laquelle on pose des questions sur des attributs (dans certains ouvrages, on parle plutôt de variables²). La figure donne un exemple d'un arbre de décision. Il y a deux attributs : l'âge et le fait que la personne soit fumeuse. On pose d'abord la question de l'âge. Si on a moins de 30 ans, il y a peu de risque. Si on a plus de 30 ans, on pose la question si la personne est fumeuse. Si non, peu de risque. Si oui, le risque est grand.



Le modèle est constitué de B arbres de décision. Pour prédire la classe d'une observation (par exemple, prédire si une personne fumeuse de 35 ans a un grand risque ou non d'avoir un accident cardiovasculaire) :

On calcule le résultat pour chaque arbre. Si par exemple, on dispose de 3 arbres, on peut imaginer obtenir "peu de risque", "grand risque" et "grand risque".

La prédiction finale de la forêt aléatoire est un simple vote majoritaire (Ensemble learning). Dans l'exemple, il y a deux fois "grand risque" mais qu'un seul "peu de risque". La prédiction finale est donc "grand risque".



Pour prédire avec un modèle de Random Forest, on calcule les prédictions intermédiaires pour chaque arbre, puis on réalise un vote majoritaire.

Chapitre 2 : Pratique

1 Analyse exploratoire des données (AED)

1.1 La recherche de l'ensemble de données

train.csv: A full training dataset with the following attributes:

- **id:** unique id for a news article
- **title:** the title of a news article
- **author:** author of the news article
- **text:** the text of the article; could be incomplete
- **label:** a label that marks the article as potentially unreliable
 - **1:** unreliable
 - **0:** reliable

Nous avons trouvé cette dataset sur kaggle elle contient 5 colonnes l'identifiant le titre de la news l'auteur le texte et le label (0 pour les vraies news et 1 pour les fausses) et 20800 lignes .

1.2 Les bibliothèques utilisées

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
import nltk
nltk.download("stopwords")
```

Nous avons importé les bibliothèques nécessaires (numpy, pandas, sklearn, matplotlib) et d'autres comme :

"re" pour utiliser les expressions régulières.

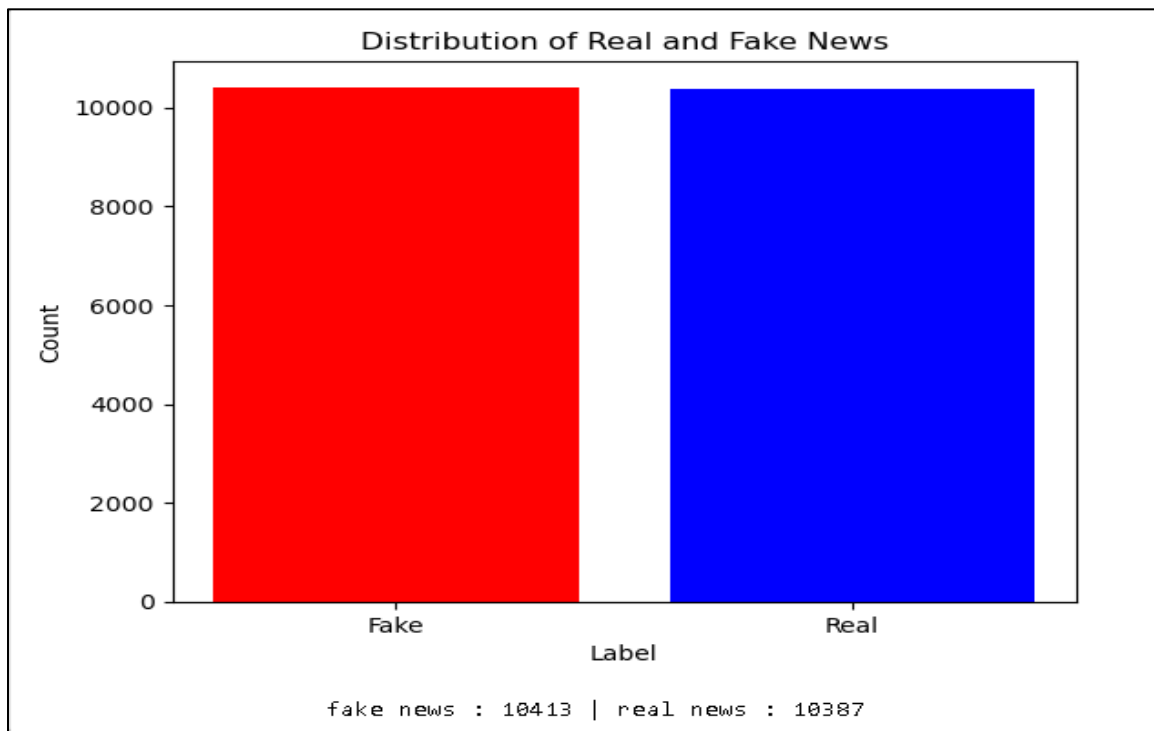
nltk.corpus.stopwords contient des mots qui n'affectent pas notre modèle lors de l'entraînement

Nous avons également Porterstemmer qui ramène le mot à sa forme racine.

tfidfvectorizer attribue une valeur numérique spécifique à chaque mot en fonction de son importance.

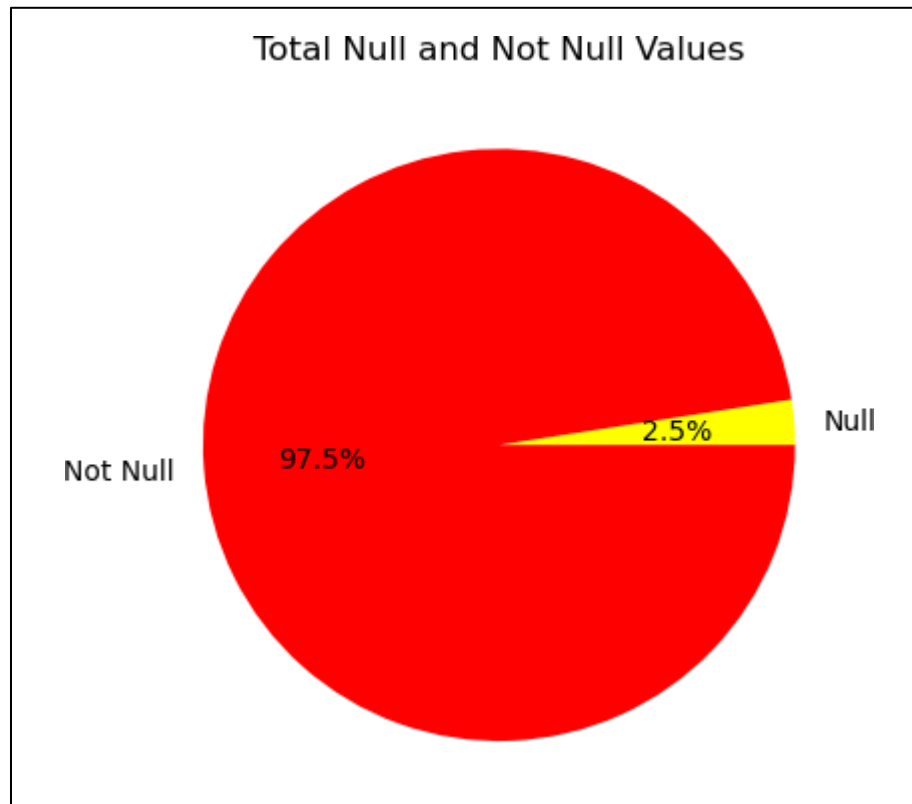
cross_val_score est utilisé pour évaluer le résultat de l'algorithme, et nous l'utiliserons pour déterminer quel algorithme est le meilleur pour notre ensemble de données.

1.3 Visualisation des données et Analyse de répartition



Nous avons compté le nombre de fausses nouvelles (label=1) et le nombre de vraies nouvelles (label=0) dans notre ensemble de données et attribué les valeurs aux variables count_fake et count_real afin de tracer une barre qui facilite la visualisation des données. nous pouvons clairement voir que nous avons presque la même quantité

de fausses (10413) et de vraies nouvelles (10387), donc notre dataset est équilibré, cela sera utile pour entraîner notre modèle.



Nous avons compté le nombre de valeurs nulles et non nulles pour chaque colonne et tracé un diagramme circulaire.

Nous pouvons voir qu'il existe des valeurs nulles 2.5%.

1.4 Gestion des données manquantes

```
In [9]: print("missing values :")
print(data.isna().sum())
```

```
missing values :
id          0
title       558
author      1957
text        39
label       0
dtype: int64
```

```
In [10]: data=data.fillna("")
```

```
In [11]: print("missing values after handling :")
print(data.isna().sum())
```

```
missing values after handling :
id          0
title       0
author      0
text        0
label       0
dtype: int64
```

Notre dataset contient près de 21 000 lignes, donc même supprimer les lignes qui ont des valeurs nulles n'affecterait pas notre modèle lors de l'entraînement, dans notre cas nous avons choisi d'attribuer une chaîne vide aux valeurs manquantes afin d'utiliser tout le dataset.

1.5 Segmentation des données

| id | title | author | text | label | content |
|----|---|--------------------|---|-------|---|
| 0 | House Dem Aide: We Didn't Even See Comey's Let... | Darrell Lucas | House Dem Aide: We Didn't Even See Comey's Let... | 1 | Darrell Lucas House Dem Aide: We Didn't Even S... |
| 1 | FLYNN: Hillary Clinton, Big Woman on Campus - ... | Daniel J. Flynn | Ever get the feeling your life circles the rou... | 0 | Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo... |
| 2 | Why the Truth Might Get You Fired | Consortiumnews.com | Why the Truth Might Get You Fired October 29, ... | 1 | Consortiumnews.com Why the Truth Might Get You... |
| 3 | 15 Civilians Killed In Single US Airstrike Hav... | Jessica Purkiss | Videos 15 Civilians Killed In Single US Aistr... | 1 | Jessica Purkiss 15 Civilians Killed In Single ... |
| 4 | Iranian woman jailed for fictional unpublished... | Howard Portnoy | Print tnAn Iranian woman has been sentenced to... | 1 | Howard Portnoy Iranian woman jailed for fictio... |

En raison de l'énorme quantité de mots que le texte peut contenir, nous ne pouvons pas utiliser cette colonne pour le traitement, cela prendra beaucoup de temps et de ressources. nous avons donc décidé de combiner à la fois la colonne auteur et le titre et nous l'avons nommé content.

```
In [16]: x= data["content"].values
y= data["label"].values
```

Puis nous avons attribué la variable cible à y et la variable prédictive à x.

1.6 Détection des valeurs aberrantes

Nous avons créé une fonction qui traite vos données texte afin d'en obtenir uniquement les données essentielles

```
def stemming(content):  
    stemmed_content = re.sub("[^a-zA-Z]", " ", content)  
    stemmed_content = stemmed_content.lower()  
    stemmed_content = stemmed_content.split()  
    stemmed_content = [port_stem.stem(word) for word in stemmed_content if not word in stopwords.words("English")]  
    stemmed_content = " ".join(stemmed_content)  
    return stemmed_content
```

Nous ne gardons que les caractères, tous les caractères spéciaux et les chiffres seront supprimés

convertir le texte en minuscule

Diviser le texte en mots en utilisant le caractère espace comme séparateur

Ici nous supprimons le suffixe de chaque mot et ajoutons le mot s'il ne s'agit pas d'un stopword dans un tableau.

Enfin nous joignons les mots que contient le tableau et ramenons le séparateur

```
data["content"] = data["content"].apply(stemming)  
x=data["content"].values
```

En utilisant la fonction apply, nous appelons notre fonction stemming pour chaque valeur de la colonne content.

1.7 Standardisation des données

```
In [40]: vectorizer = TfidfVectorizer()  
         vectorizer.fit(x)  
         X = vectorizer.transform(x)
```

Nous avons utilisé le `Tfidf_vectorizer` pour convertir nos données texte en valeurs numériques. il attribue une valeur numérique à chaque mot en fonction du nombre d'apparitions de ce mot. La valeur la plus élevée signifie que le mot est important.

2 Choix de l'algorithme d'apprentissage automatique

2.1 Sélection des algorithms

Nous avons choisi d'utiliser deux algorithmes pouvant être utilisés en classification la régression logistique et la forêt aléatoire.

2.2 Diviser les données

```
In [42]: x_train,x_test,y_train,y_test = train_test_split(X,y,stratify=y,random_state=2)
```

nous avons divisé nos données à l'aide de la fonction `train_test_split` et nous avons attribué `label` à `stratify`, ce qui garantit essentiellement que la distribution de `label` dans les données d'entraînement est similaire distribution de données de test.

2.3 Entraînement et évaluation

L'entrainement de model de régression logistique

```
In [43]: lr = LogisticRegression()
```

```
In [44]: lr.fit(x_train,y_train)
```

```
Out[44]: LogisticRegression  
LogisticRegression()
```

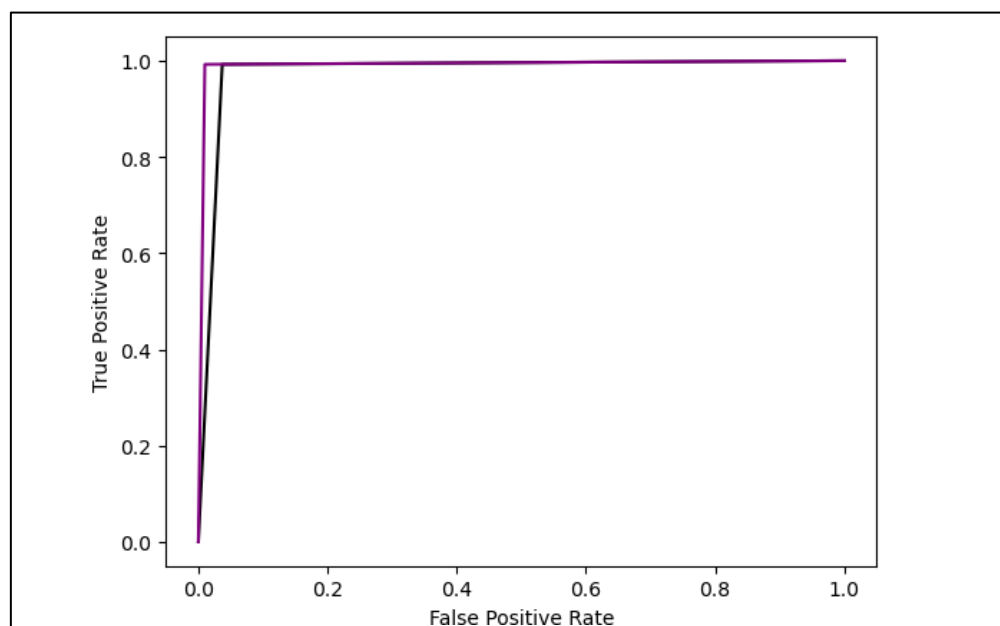
L'entrainement de model de foret aléatoire

```
In [45]: rfc = RandomForestClassifier()
```

```
In [46]: rfc.fit(x_train,y_train)
```

```
Out[46]: RandomForestClassifier  
RandomForestClassifier()
```

Tracer la courbe roc pour les deux algorithmes



Forêt aléatoire

Régression Logistique

Après avoir tracé les courbes, nous pouvons voir qu'elles ont presque le même taux de true positive mais l'algorithme de forêt aléatoire a obtenu moins de false positive.

2.4 Comparaison des performances

```
score_lr_train = accuracy_score(y_train, y_pred_lr_train)
score_lr_test = accuracy_score(y_test, y_pred_lr_test)
score_rfc_train = accuracy_score(y_train, y_pred_rfc_train)
score_rfc_test = accuracy_score(y_test, y_pred_rfc_test)

In [49]: print(f"accuracy_score using logistic regression training : {score_lr_train:.3f} test : {score_lr_test:.3f}")
accuracy_score using logistic regression training : 0.986 test : 0.978

In [50]: print(f"accuracy_score using random forest : {score_rfc_train:.3f} test : {score_rfc_test:.3f}")
accuracy_score using random forest : 1.000 test : 0.993
```

Nous avons utilisé `accuracy_score` pour comparer à la fois les données de test et les données prédites.

La forêt aléatoire a obtenu un meilleur score que la régression logistique.

```
In [51]: cm_lr = confusion_matrix(y_test, y_pred_lr_test)
cm_rfc = confusion_matrix(y_test, y_pred_rfc_test)

In [52]: print("Logistic Regression")
print(cm_lr)
print("Random Forest Classifier")
print(cm_rfc)

Logistic Regression
[[2501  96]
 [ 20 2583]]
Random Forest Classifier
[[2571  26]
 [ 10 2593]]
```

À partir des matrices de confusion, nous pouvons voir que la forêt aléatoire a obtenu moins de valeurs false négative et false positive.

```
In [53]: cv_scores_rfc = cross_val_score(RandomForestClassifier(),X,y,cv=5,scoring='accuracy')
cv_scores_lr = cross_val_score(LogisticRegression(),X,y,cv=5,scoring='accuracy')
avg_score_rfc = cv_scores_rfc.mean()
avg_score_lr = cv_scores_lr.mean()
print("random forest classifier:", avg_score_rfc)
print("logistic regression:", avg_score_lr)

random forest classifier: 0.9926442307692309
logistic regression: 0.9762019230769232
```

Nous avons aussi utilisé le score de validation croisée pour évaluer les performances des deux algorithmes.

L'algorithme de forêt aléatoire a obtenu le meilleur score.

2.5 Choix du meilleur modèle

D'après la comparaison des performances, l'algorithme de forêt aléatoire est le meilleur algorithme pour notre objectif.

2.6 Évaluation sur l'ensemble de test

```
In [54]: def is_real(information):
          prediction=rfc.predict(information)
          return prediction[0]==0

In [55]: is_real(x_test[0])
Out[55]: True

In [56]: print(y_test[0]== 0)
True
```

Nous avons créé une fonction nommée `is_real`, elle renvoie la prédiction de notre modèle, nous lui avons donc donné la première valeur dans nos données de test et elle nous a indiqué qu'elle était vraie et après avoir afficher la sortie des données de test, nous avons découvert que c'était effectivement vraie.

3 L'interface API

```
1 from flask import Flask, request, jsonify
2 from flask_cors import CORS
3 import joblib
4 import signal
5 import sys
6 import re
7 from nltk.corpus import stopwords
8 from nltk.stem.porter import PorterStemmer
```

- Flask, request et jsonify sont utilisés pour créer une application web Flask et manipuler les requêtes HTTP et les réponses JSON.
- CORS est utilisé pour activer CORS (Cross-Origin Resource Sharing), ce qui permet à l'API d'accepter les requêtes depuis n'importe quelle origine.
- joblib est utilisé pour charger le modèle de machine learning et le vectorizer pré-entraînés.
- signal et sys sont utilisés pour gérer les signaux système, ce qui permet de fermer correctement l'application lorsqu'elle est interrompue.
- re est utilisé pour effectuer des opérations de correspondance de motifs (dans ce cas, pour le prétraitement du texte).
- stopwords est utilisé pour obtenir une liste de mots vides à supprimer lors du prétraitement du texte.
- PorterStemmer est utilisé pour effectuer la racinisation (stemming) des mots.

```
1 port_stem = PorterStemmer()
2 def stemming(content):
3     stemmed_content = re.sub("[^a-zA-Z]", " ", content)
4     stemmed_content = stemmed_content.lower()
5     stemmed_content = stemmed_content.split()
6     stemmed_content = [port_stem.stem(word) for word in stemmed_content if not word in stopwords.words("English")]
7     stemmed_content = " ".join(stemmed_content)
8     return stemmed_content
```

Ici, nous initialisons un objet `PorterStemmer` et définissons une fonction `stemming` qui prend du texte en entrée, effectue plusieurs étapes de prétraitement (suppression de la ponctuation, mise en minuscules, racinisation des mots et suppression des mots vides) et renvoie le texte prétraité.


```
1 app = Flask(__name__)
2 CORS(app, resources={r"/*": {"origins": "*"}})
```

Nous créons une instance de l'application Flask et activons CORS pour permettre les requêtes depuis n'importe quelle origine.

```
1 @app.route('/predict', methods=['POST'])
2 def predict():
3     try:
4         data = request.get_json()
5         news = data['author']+" "+data["news"]
6         news = stemming(news)
7         news = vectorizer.transform([news])
8         prediction = model.predict(news)
9         return jsonify({'prediction': int(prediction[0])})
10    except Exception as e:
11        return jsonify({'error': e})
```

Nous définissons une route /predict qui accepte les requêtes POST contenant les données des articles de presse. Les données sont prétraitées à l'aide de la fonction de stemming, puis transformées à l'aide du vectorizer. Enfin, le modèle est utilisé pour prédire la classe de l'article de presse et renvoyer la prédiction au client sous forme JSON. Si une erreur se produit, elle est renvoyée sous forme de réponse JSON.

```
1 signal.signal(signal.SIGINT, signal.SIG_DFL)
```

Nous configurons le gestionnaire de signaux pour capturer SIGINT (par exemple, CTRL+C) et utiliser la gestion par défaut de l'interruption.

```
1 if __name__ == '__main__':
2     app.run(host='localhost', port=5000, debug=True, use_reloader=False)
```

Nous exécutons l'application Flask sur le port 5000 en mode debug. Cette partie du code ne sera exécutée que si le script est exécuté directement (et non importé en tant que module).

4 Index Html

```
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Décteur de Fausses Nouvelles</title>
7      <link rel="stylesheet" href="style.css">
8  </head>
9  <body>
10     <div class="container">
11         <header>
12             <h1>Décteur de Fausses Nouvelles</h1>
13         </header>
14         <main>
15             <form id="newsForm">
16                 <label for="authorInput">Auteur :</label>
17                 <input type="text" id="authorInput" placeholder="Entrez le nom de l'auteur">
18
19                 <label for="newsInput">Titre de nouvelle :</label>
20                 <textarea id="newsInput" rows="4" cols="50" placeholder="Saisissez ou collez votre">
21
22                 <button type="button" onclick="checkNews()">Vérifier les Nouvelles</button>
23             </form>
24             <div id="result"></div>
25         </main>
26     </div>
27     <script src="script.js"></script>
28 </body>
29 </html>
```

Ce code HTML représente une page web qui met en œuvre un "Décteur de Fausses Nouvelles". L'utilisateur peut saisir le nom de l'auteur ainsi que le titre de la nouvelle dans des champs dédiés. Ensuite, en cliquant sur le bouton "Vérifier les Nouvelles", une fonction JavaScript est déclenchée pour analyser les données fournies et évaluer la véracité de la nouvelle. Le résultat de cette analyse est affiché dans la section dédiée à cet effet sur la page.

5 Java Script

```
1  async function checkNews() {
2      const authorInput = document.getElementById("authorInput").value || "";
3      const newsInput = document.getElementById("newsInput").value ;
4
5      if (newsInput.trim() === "") {
6          alert("Veuillez saisir le nom de l'auteur et la nouvelle avant de vérifier.");
7          return;
8      }
9
10     try {
11         const response = await fetch("http://localhost:5000/predict", {
12             method: "POST",
13             headers: {
14                 "Content-Type": "application/json",
15             },
16             body: JSON.stringify({ author: authorInput, news: newsInput }),
17         });
18
19         const result = await response.json();
20         displayResult(result);
21     } catch (error) {
22         console.error("Erreur :", error);
23         alert("Une erreur s'est produite lors du traitement de votre demande.");
24     }
25 }

```

```
26
27 function displayResult(result) {
28     const resultElement = document.getElementById("result");
29
30     if (result.error) {
31         resultElement.innerHTML = "Erreur : " + result.error;
32     } else {
33         if (result.prediction === 1) {
34             resultElement.innerHTML = "Cette nouvelle est fausse.";
35         } else {
36             resultElement.innerHTML = "Cette nouvelle est vraie.";
37         }
38     }
39 }
40

```

Ce code JavaScript implémente une fonction asynchrone `checkNews()` qui envoie une requête HTTP POST à un serveur local, demandant une prédiction sur la véracité d'une nouvelle saisie par l'utilisateur. La réponse est ensuite affichée sur la page HTML. La fonction `displayResult()` est utilisée pour présenter le résultat de la prédiction à l'utilisateur.

6 CSS

```
1  body {
2      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
3      background-color: #eef5f9; /* Light blue-gray background */
4      margin: 0;
5      padding: 0;
6  }
7
8  .container {
9      max-width: 800px;
10     margin: 50px auto;
11     background-color: #ffffff; /* White container background */
12     padding: 30px;
13     border-radius: 8px;
14     box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
15 }
16
17 header {
18     text-align: center;
19     margin-bottom: 30px;
20 }
```

```
21
22 h1 {
23     color: #333; /* Dark gray text color */
24     margin: 0;
25 }
26
27 main {
28     padding: 20px;
29 }
30
31 form {
32     max-width: 400px;
33     margin: 0 auto;
34 }
35
36 label {
37     display: block;
38     margin-bottom: 10px;
39     color: #555; /* Medium gray text color */
40 }
```

```
41
42 input,
43 textarea {
44     width: 100%;
45     padding: 10px;
46     font-size: 16px;
47     border: 1px solid #ccc;
48     border-radius: 4px;
49     margin-bottom: 20px;
50 }
51
52 button {
53     background-color: #00563F; /* Blue button color */
54     color: #fff; /* White text color */
55     padding: 15px;
56     border: none;
57     border-radius: 4px;
58     cursor: pointer;
59     font-size: 16px;
60     width: 100%;
61     transition: background-color 0.3s ease;
62 }
```

```
63
64 button:hover {
65     background-color: #0056b3; /* Darker blue on hover */
66 }
67
68 #result {
69     text-align: center;
70     font-size: 18px;
71     color: #333; /* Dark gray text color */
72 }
```

Ce code CSS fournit une mise en forme esthétique et conviviale pour une page web qui implémente un formulaire de saisie de données. Il utilise une palette de couleurs douces et une disposition équilibrée pour offrir une expérience visuelle agréable aux utilisateurs. Les styles appliqués assurent une présentation claire et lisible des éléments, facilitant ainsi la navigation et l'interaction avec le contenu.

7 Test

Cherchons les news :

News 1 :



Titre de News

Auteur de news

Détecteur de Fausses Nouvelles

Auteur :

Deborah brand

Titre de nouvelle :

POLL: MOST ISRAELIS BELIEVE TRUMP WILL OOD BE FRIENDLY TO ISRAEL

Vérifier les Nouvelles

Cette nouvelle est fausse.

Donc cette nouvelle est fausse, on a trouve que l'auteur de cette nouvelle a une mauvaise reputation

Inconnu, pas de social media, pas de linkedin...+ Deborah Brand C'est un nom d'une Brand de Makeup

25% sont des histoires et news qui sont fake

Access The World's Most Insightful Media Database, Powered by

Deborah Brand

AS SEEN ON
Not enough data

Get in Touch

Preston's Summary ⓘ

Deborah Brand is a journalist for Breitbart.tv, specializing in reporting on international affairs, particularly on issues related to Israel and the Middle East. Her articles often focus on political developments, diplomatic relations, and security concerns in the region, providing a conservative perspective on these topics.

Geo Focus

International

Coverage Attributes:
Beta

Government Announcement: 31%
Evolving Stories: 25%
Legal Policy Regulation: 18%
Breaking News: 15%
Event Coverage: 2%

Themes Covered:

- Politics
- Government & Politics
- Lifestyle
- Entertainment
- General

Most Recent Topics:

Not enough data

News 2 :



Auteur de news

Titre de News

Détecteur de Fausses Nouvelles

Auteur :

Nate Church

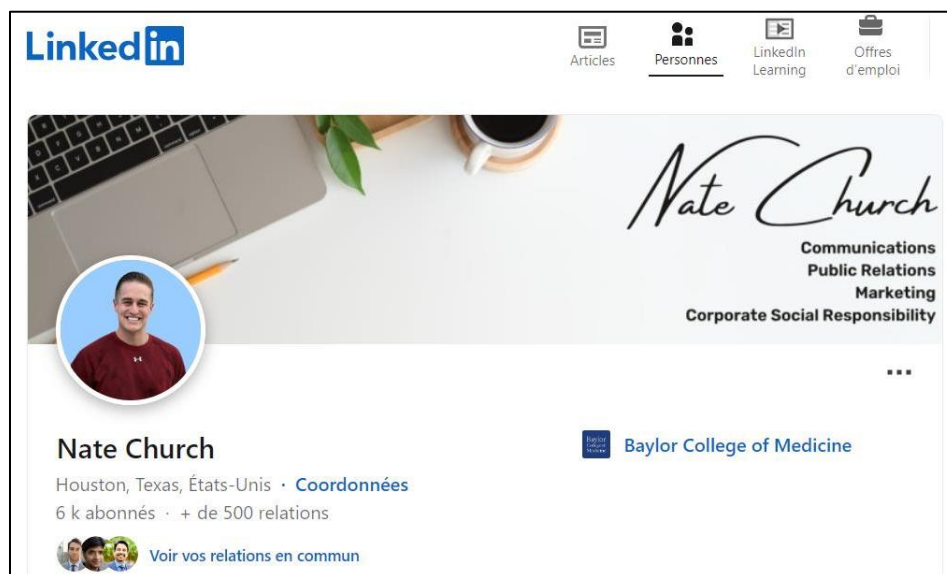
Titre de nouvelle :

BOEING SUITS UP FOR FUTURE OF SPACEFLIGHT
WITH NEW SPACESUIT DESIGN

Vérifier les Nouvelles

Cette nouvelle est vraie.

Donc cette nouvelle est vraie, on a trouvé que l'auteur de cette nouvelle a une Bonne réputation



The image shows a LinkedIn profile for Nate Church. The header includes the LinkedIn logo and navigation links for Articles, Personnes, LinkedIn Learning, and Offres d'emploi. The profile picture is a circular photo of a man in a red shirt. The background banner features a laptop, a pen, and a coffee cup. The name 'Nate Church' is written in a large, stylized script. Below the name, the following skills are listed: Communications, Public Relations, Marketing, and Corporate Social Responsibility. The profile indicates he is located in Houston, Texas, États-Unis, and is associated with Baylor College of Medicine. It also shows he has 6k followers and over 500 connections.

LinkedIn

Articles Personnes LinkedIn Learning Offres d'emploi

Nate Church

Communications
Public Relations
Marketing
Corporate Social Responsibility

Nate Church

Houston, Texas, États-Unis · [Coordonnées](#)

6 k abonnés · + de 500 relations

[Voir vos relations en commun](#)

Baylor College of Medicine