**IEMS5726 Data Science in Practice (Fall 2022)**

**Course Project**

**Student Name:** DAI Yayuan

**Student ID:** 1155182964

**Project Number:** 3

**Project Title:** Fake News Detection

**Problem definition:**

In the face of many complex social network news, it is difficult to distinguish the true and false news. Fake news items may contain false or exaggerated claims, and may end up being viralized by algorithms, and users may end up in a filter bubble.Therefore, based on this background, this project proposes a **machine learning** method to detect and distinguish the true and false news.

In Part 1, **TfidfVectorizer** and **PassiveAggressive Classifier** of the sklearn library are mainly involved. Among them, **TfidfVectorizer** is mainly used to vectorize text, only after the text is vectorized can the text be trained, and **PassiveAggressive Classifier** is a text classifier. In this project, news text can be classified after entering marked data and training it.

**Data source:**

The dataset that will be used for this python project is **[news.csv](https://drive.google.com/file/d/1er9NJTLUA3qnRuyhfzuN0XUsoIC4a-_q/view)**. It has a shape of **6335×4**. The **first column** identifies the news, the **second** and **third** are the title and text, and the **fourth column** has labels denoting whether the news is REAL or FAKE. The dataset takes up 29.2MB of space.

**Download URL:** <https://drive.google.com/file/d/1er9NJTLUA3qnRuyhfzuN0XUsoIC4a-_q/view>

**Data preprocessing:**

In Part 1, data preprocessing only includes **text vectorization**.

Usually, text vectorization is a import part of NLP processes, because text as an unstructured data, must be aligned for vectorization before being aligned for processing.

In this project, we use **TfidfVectorizer** to vectorize text. It is based on **TF-IDF,**

its main idea is that if a word or phrase appears with a high frequency TF in one article and rarely appears in other articles, it is considered that the word or phrase has good classification ability and is suitable for classification. TF-IDF is a statistical method to evaluate the importance of a word to a file set or one of the files in a corpus.

The following method is used to vectorize the text data：

tfidf\_vectorizer = TfidfVectorizer(stop\_words='english', max\_df=0.7)  
tfidf\_train = tfidf\_vectorizer.fit\_transform(x\_train)  
tfidf\_test = tfidf\_vectorizer.transform(x\_test)

**Data analysis:**

In Part 1, **Passive Aggressive Classifier** is selected, and the passive attack algorithm is based on **online learning**.

The principle of the algorithm for **model adjustment** is as follows:

* When the sample classification is correct and the model's prediction of possibility is accurate (the degree is greater than one), the model will not be adjusted (passive is shown here)
* When the sample classification is correct, but the model's prediction of possibility is biased (not accurate), the model makes slight adjustment

When the sample classification is wrong, the model makes a large adjustment (reflecting a strong "aggressiveness")

For the selection of metrics, only the **accuracy score** is selected. After getting the prediction result Y of the model for X, the accuracy score is compared with the true value of Y to calculate the accuracy of the model classification. In this project, the classification accuracy of passive AggressiveClassifier has reached more than **92%**.



**Conclusions and discussions:**

In this project, after quantifying the text and training the model with the passive aggregative classifier, the news was divided into **'FAKE'** and **'REAL'**. Finally, the classification results were compared with the true value, and the quasi group rate of classification reached more than **92%**.

**Extensions of the course project in part 2:**

1. **Apply more NLP skills on data preprocessing**

On the basis of the Part 1 source code, the data **preprocessing part is added.**

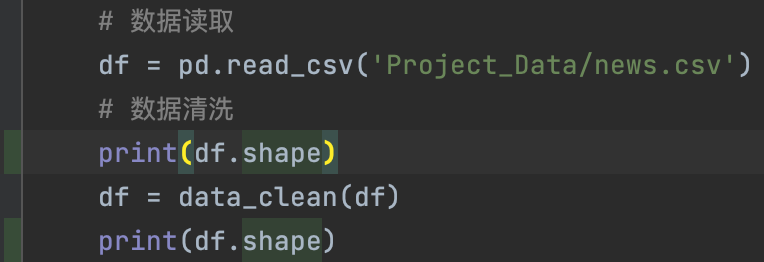
* **Delete Empty Columns.**

df = df.dropna()

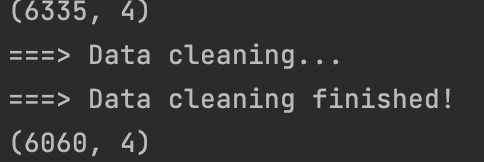
* **Eliminate duplicate news.**

df.drop\_duplicates(subset=['text'], keep='first', inplace=True)

After eliminating duplicate news, this data shape reduced from **6335 × 4** to **6060 × 4.**



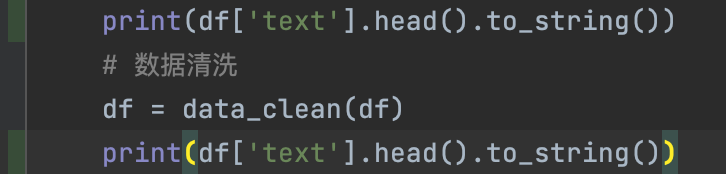
**Terminal Print:**



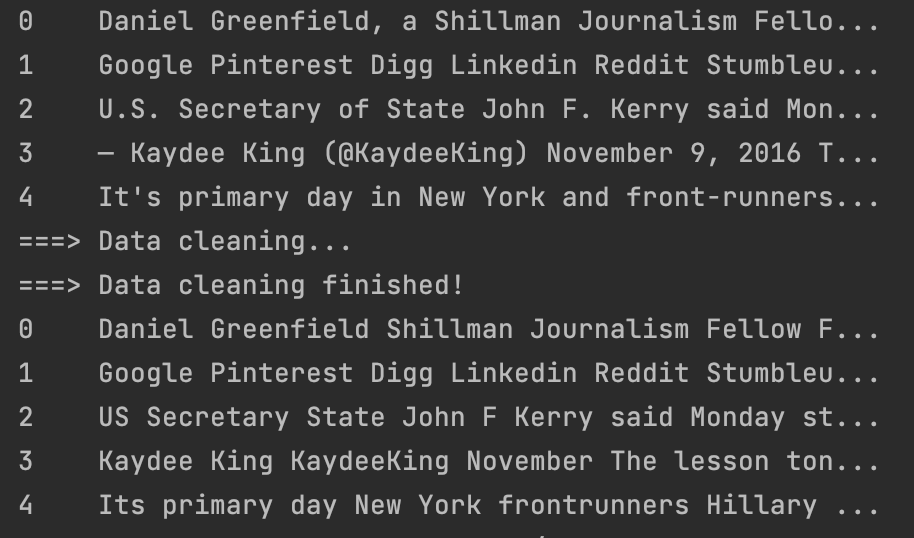
* **Remove punctuation, number, stopwords.**

Use **RE** library to remove punctuation, number, stopwords.

p = re.compile(r'[^\w\s]+')  
df['text'] = [p.sub('', x) for x in df['text'].tolist()]  
p = re.compile(r'[^\D]+')  
df['text'] = [p.sub('', x) for x in df['text'].tolist()]  
stop\_word = stopwords.words('english')  
df['text'] = df['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop\_word)]))



**Terminal Print:**



1. **Perform comparison for multiple machine learning algorithms**

The following categories of classifiers are selected for training, and the accuracy of the classified results is compared.

* **Random Forest Classifier**

RandomForestClassifier(n\_estimators=50, criterion="entropy")

* **Passive Aggressive Classifier**

PassiveAggressiveClassifier(max\_iter=50)

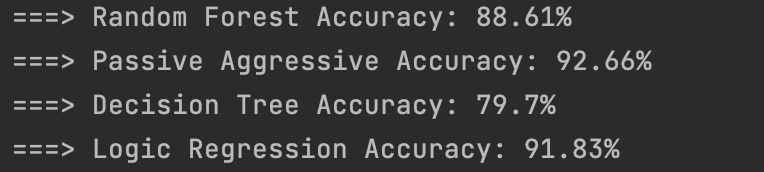
* **Decision Tree Classifier**

DecisionTreeClassifier(criterion='entropy', max\_depth=20, splitter='best', random\_state=42)

* **Logic Regression Classifier**

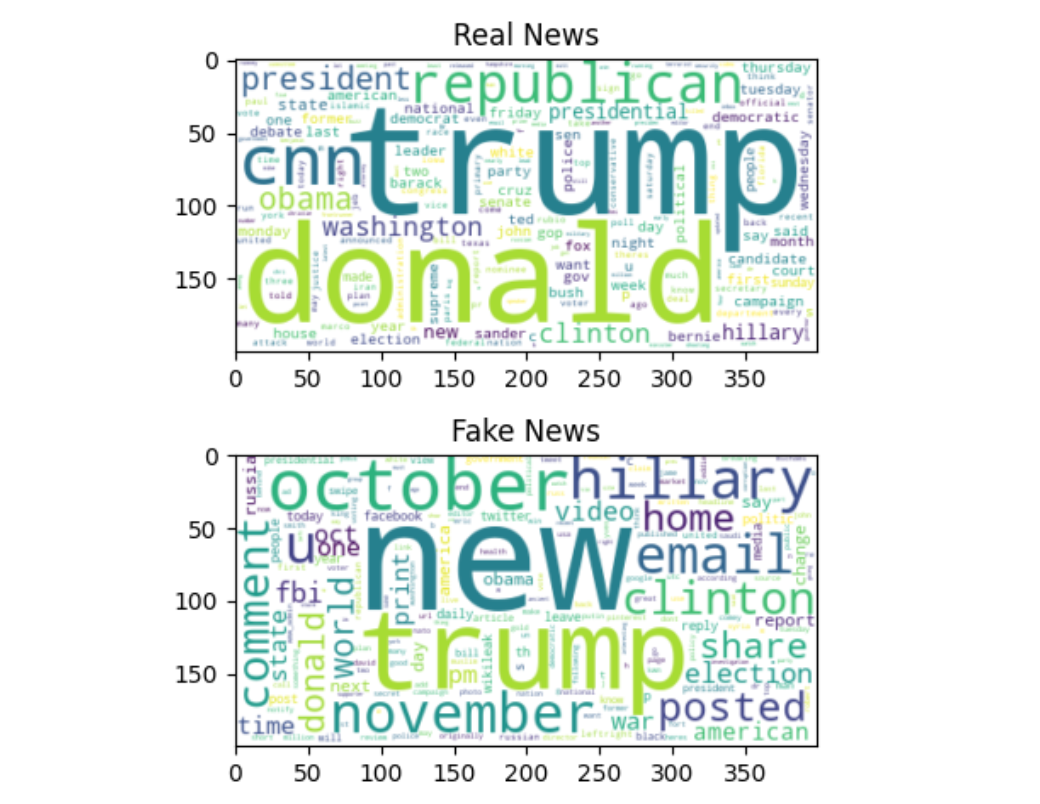
LogisticRegression()

By using some metrics, we get the accuracy of different classification models to identify false news. The specific comparison pictures are shown in the visualization section. The accuracy results are shown in the figure below.

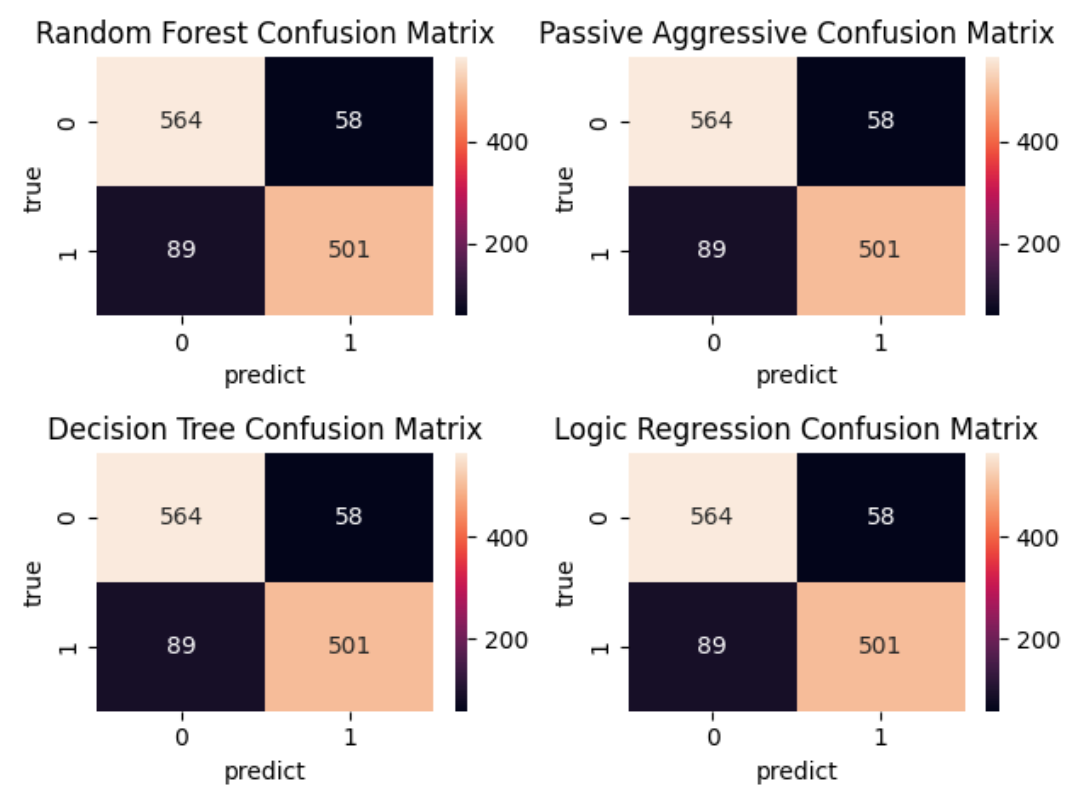


1. **Perform some visualizations for the analysis**

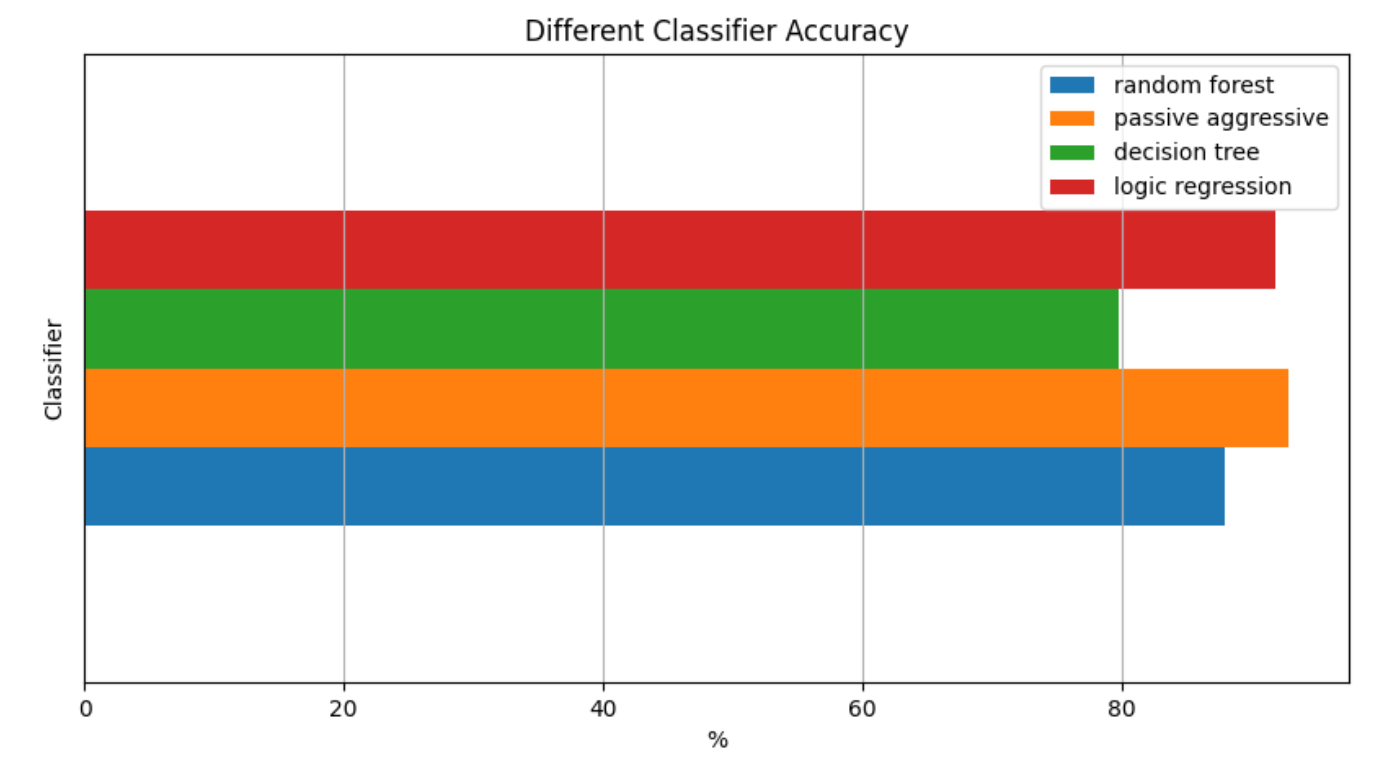
According to the **marked data**, after the data is divided into **real** news and **fake** news, using **WordCloud** library **to generate word clouds** of fake news and real news respectively for reference.



Four different classifiers are used to classify the real and fake news, and their **confusion matrices** and **model prediction accuracy** are obtained, and then use **seaborn** and **matpoltlib** libraries to draw comparison chart.



**Confusion Matrices**



**Classifier Accuracy**