**Fake News Detection using NLP and Machine Learning in Python**

Step by Step guide for fake news detection using machine learning, natural language processing in python

In this post, we will be discussing fake news detection using machine learning and will start to understand what is fake news, and what is the source of its generation?

**What is Fake news?**

Fake news is a form of news that consists of falsified information or hoaxes to deceive users for clickbait. Clickbait is the technique for grabbing the attention of users with flashy headlines which makes them click the link with the purpose of generating revenue by showing different advertisements. Today, with the increasing usage of social media, spreading fake news over the internet, increases manifold and the major source of spreading fake news is online news portals which makes it really difficult to distinguish between real and fake news. In this case study, we will discuss how we can detect fake news from news headlines using natural language processing (NLP) and machine learning-based techniques. The full code used in this post is available in my Github repo.

**About Data**

The dataset used in this case study is the ISOT Fake News Dataset. The dataset contains two types of articles fake and real news. This dataset was collected from real-world sources; the truthful articles were obtained by crawling articles from Reuters.com (News website). As for the fake news articles, they were collected from different sources. The fake news articles were collected from unreliable websites that were flagged by Politifact (a fact-checking organization in the USA) and Wikipedia. The dataset contains different types of articles on different topics, however, the majority of articles focus on political and world news topics.

The dataset consists of two CSV files. The first file named contains more than 12,600 articles from reuter.com. The second file named contains more than 12,600 articles from different fake news outlet resources. Each article contains the following information:

Article title (News Headline),

Text,

Label (REAL or FAKE)

Exploratory data analysis

In this case study, we have extracted interesting patterns from the news headline text using NLP and perform exploratory data analysis to provide useful insights about real and fake news headlines.

**Snapshot of dataset**

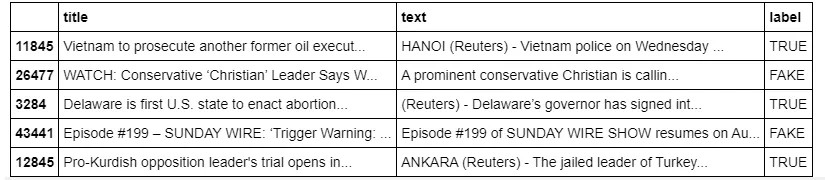
Figure 1 shows the top 5 entries of the actual dataset used in the case study.

Figure 1. Snapshot of the actual dataset used in the case study

Distribution of Fake news

Firstly, we check the distribution of fake and true news in the dataset by plotting the bar graph as shown in Figure 2.

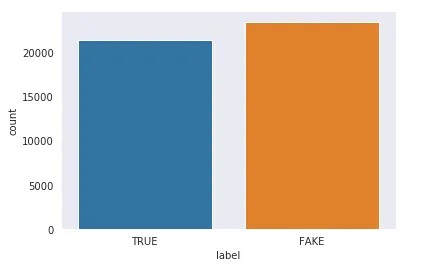


Figure 2. Distribution of Real and Fake news in the dataset

As we have seen from the above figure, the dataset is balanced having 21417 true news and 23481 fake news.

**Distribution of the number of characters**

Next, we have checked the distribution of the number of characters in the fake and true news titles. From the below figure it is evident that the average number of characters is higher in case of fake news in comparison to true news.

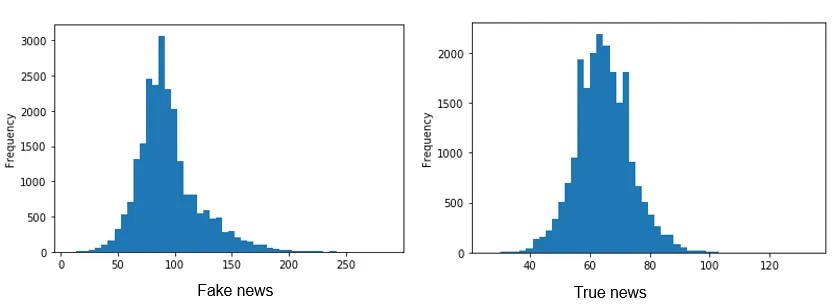


Figure 3. Distribution of the number of characters in fake news (left) and true news (right)

Distribution of unique words

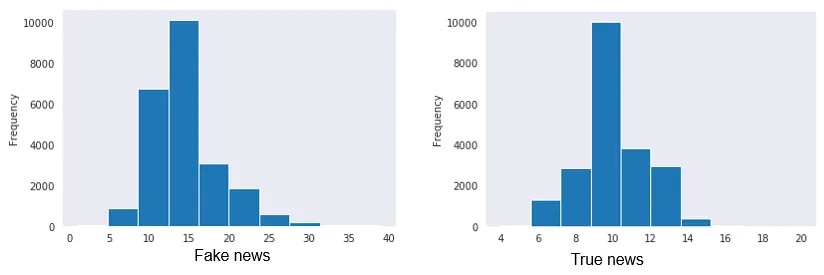
Further, we have checked the distribution of unique words used in both fake news and true news titles. As per the figure, it is observed that fake news consists of more unique words in comparison to true news as its main objective is to deceive users with the use of attention-grabbing words in the headlines.

Figure 4. distribution of unique words used in fake news (left) and true news (right)

**Distribution of special characters**

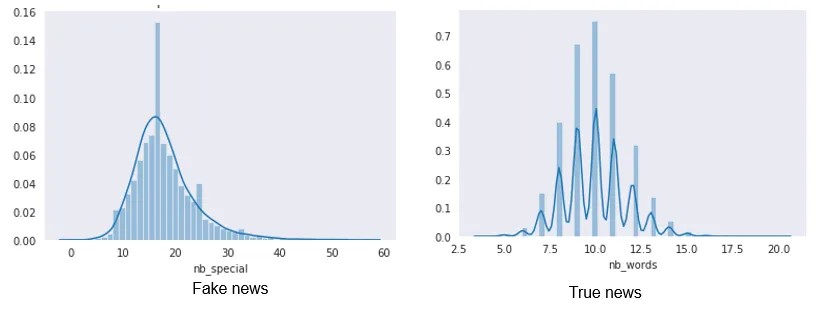
In the end, we have checked the distribution of special characters used in both fake and true news and found that more special characters are used in fake news in comparison to true news.

Figure 5. distribution of special characters in fake news (left) and true news (right)

Word Cloud to plot the most frequent words

So, next, we will plot the most frequent words in fake news and real news using the word cloud. Word cloud is a technique for visualizing most frequent words in a text corpus where the size of the words represents their frequency. For plotting word cloud we have used word cloud python library.

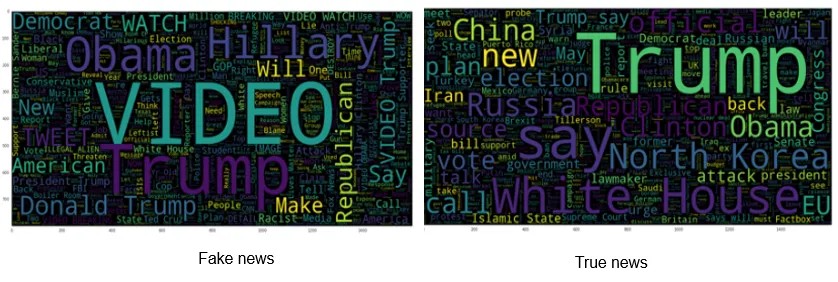


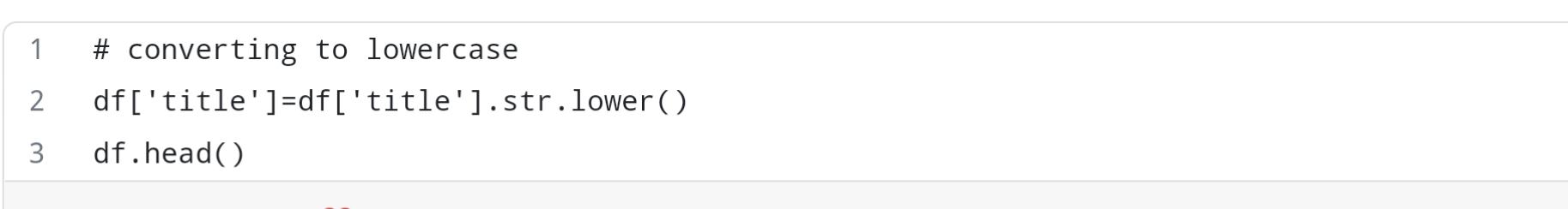
Figure 6. Word cloud representation of fake news and real news

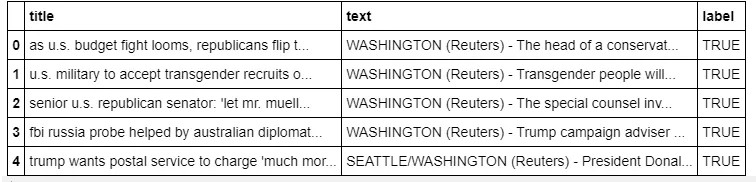
As we can see from the above figure, most frequent words in fake news are Video, Obama, Hillary, Trump and Republican whereas Real news comprises Trump, White House, North Korea, China, etc.

**Text pre-processing**

After analyzing the data, we move towards text pre-processing before building machine learning models. The text pre-processing consists of the following steps:

Step 1: Lower casing

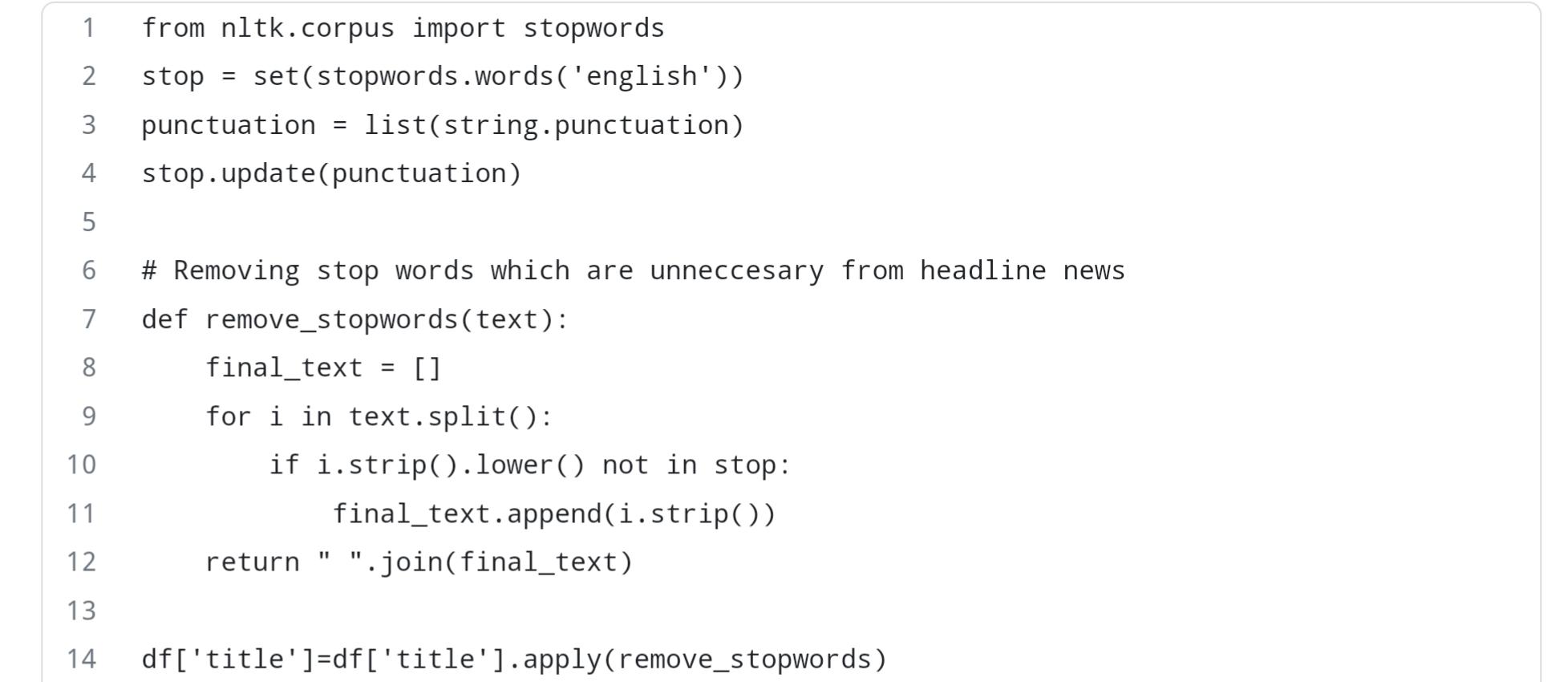
In the lower casing, we convert all the words in the text in lower case as words VIDEO and video are the same in contextual meaning but it represents different words in vector space resulting in more dimensions.



So, as we can see from the above data frame, all the text in the title column is now converted to lower case. Here df is the pandas data frame using which we have loaded the dataset.

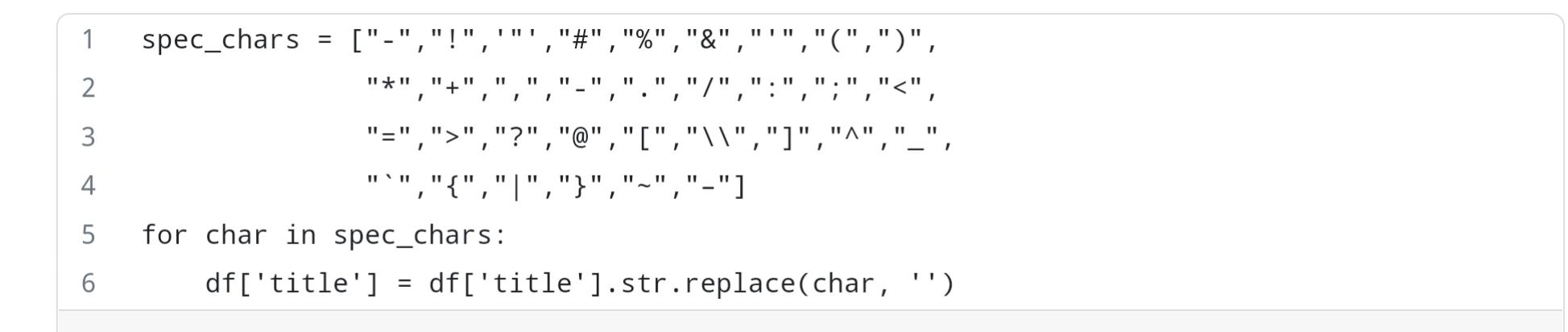
Step 2: Stop word removal

Stop words are the most common words used in regular conversation and it does not add significant value in the conversation. The examples of stop words are a, an, the, he, she, it, etc. So In text classification tasks, we use to remove such stop words by importing stop words defined in nltk corpus. The python code for removing stop words are shown below:



Step 3: Special character removal

In this step, we remove all types of special characters in the news title. The special characters are also not significant for text classification and it only increases the total dimension in vector space, so we filter them out before building the machine learning model. The code for filtering out special characters is shown below.



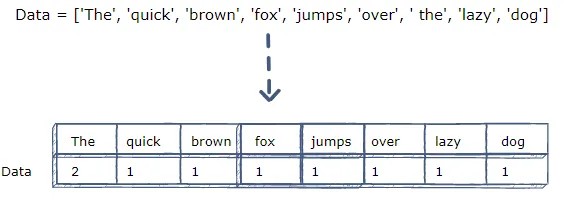
It is important to note that, stemming and lemmatization are considered important steps but for this case study we have not performed any of them either, but you are free to try them and see how the final result changes.

Train Test Split

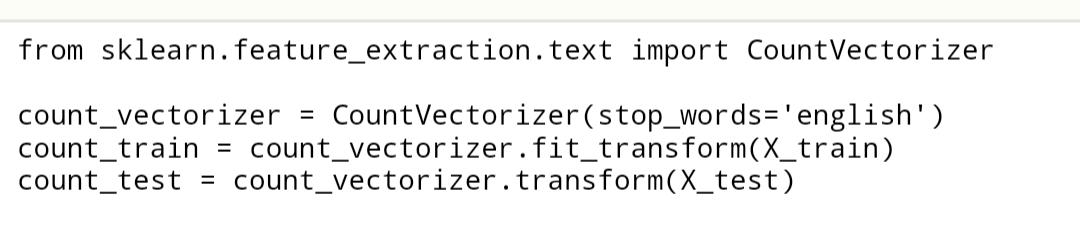
In this step, we split the data into train and test set in the ratio of 75:25 i.e., 75% of the data used in training the model and rest 25% used for testing the model. The code for splitting data is shown below.

**Model building**

First, we have built our baseline model with a count vectorizer. Count vectorizer converts the text document to a vector of token counts. The actual working of the count vectorizer is shown in the below figure.



So, as we can see from the below figure, the sentence “The quick brown fox jumps over the lazy dog” is converted into a frequency table in which column represents tokens in the sentence and rows represents the corresponding frequency of those tokens.



So, the code for applying the count vectorizer to the text document is shown below. Code vectorizer is defined in sklearn python library

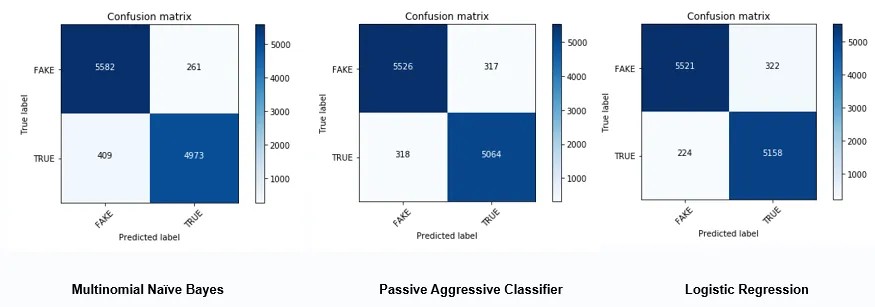
Next, with count vectorizer, we have used three machine-learning algorithms Multinomial Naïve Bayes which generally performs better in case of text classification, Passive Aggressive classifier, and Logistic regression algorithms. The results of these algorithms in terms of the confusion matrix are below.

Figure 7. Figure showing the confusion matrix of Multinomial Naïve Bayes (Left), Passive-aggressive classifier (Middle) and Logistic regression (Right) using count vectorizer

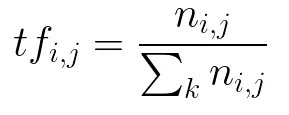
As we can see from Figure 7, the highest true positives and true negatives detected by the logistic regression algorithm while the second best is a passive-aggressive classifier.

To improve the performance of our machine learning algorithms we have used the TFIDF vectorizer. TFIDF vectorizer converts the text document into a matrix of TFIDF features. Now lest see what is TFIDF ? and why it performs better than count vectorizer?

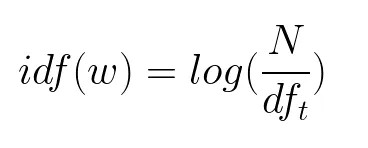
Now, let’s break down TFIDF into TF i.e., Term Frequency and IDF i.e., Inverse Document Frequency.

Term Frequency (TF)

Term Frequency represents the number of times a word appears in a document divided by the total number of words in the document. The formulae of Term frequency is mathematically shown as below.

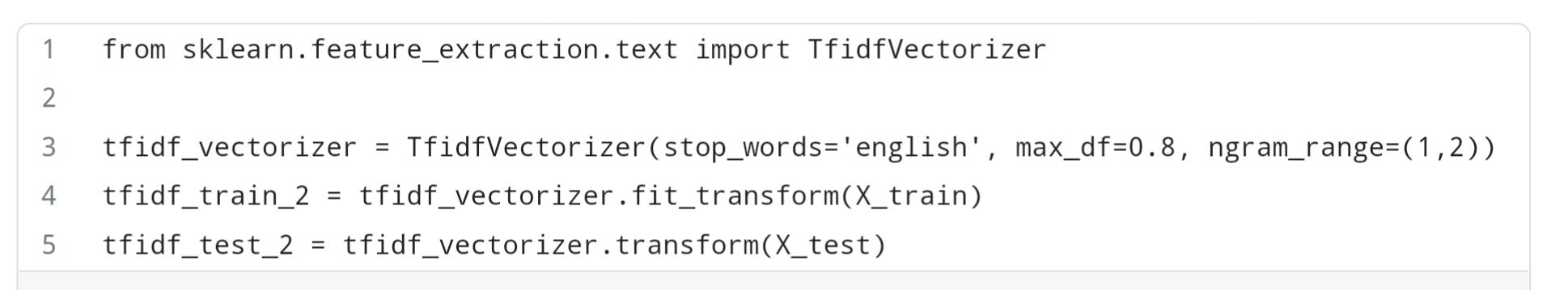


Inverse Document Frequency (IDF)

It represents the log of the number of documents divided by the number of documents containing the word w. Inverse data frequency used to weight the rare words across all documents in the corpus.

The reason why TFIDF performs better than count vectorizer is that TFIDF gives higher weight to rare words across all the documents whereas count vectorizer gives importance to common words which are of very little importance in text classification.

The code for Implementing TFIDF is defined in the form of a library that is defined under sklearn.



TFIDF Hyperparameters

Now let’s talk about its hyper-parameter first one is stopwords which is defined for making aware that stopwords used in the text are of English language, max\_df is used for removing terms that appear too frequently, In our case, we have taken its value as 0.8 which means remove those words which appear in more than 80% of the documents and the last hyper-parameter is ngram\_range which is set as (1,2) i.e., it will allow both unigrams and bigrams.

Next, with the TFIDF vectorizer, we have used the same machine-learning algorithms i.e., Multinomial Naïve Bayes, Passive Aggressive classifier, and Logistic regression algorithms. The results of these algorithms in terms of the confusion matrix is shown below.

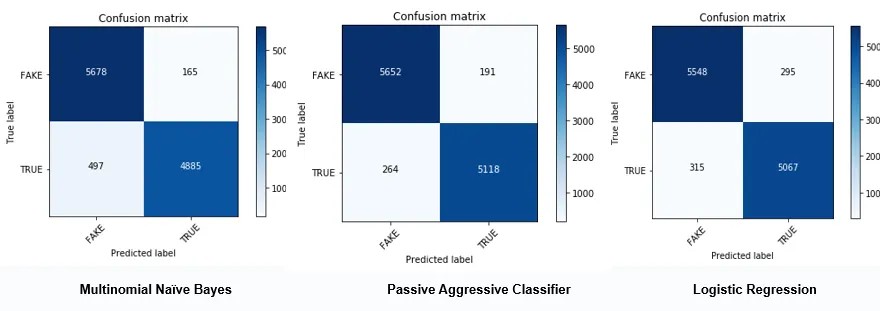
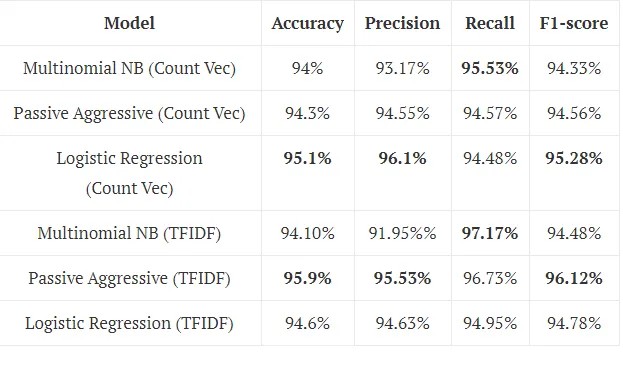


Figure 8. Figure showing the confusion matrix of Multinomial Naïve Bayes (Left), Passive-aggressive classifier (Middle) and Logistic regression (Right) using TFIDF vectorizer

**Result Analysis**

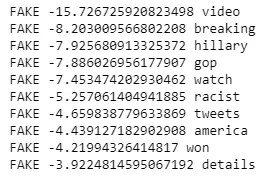
Now let’s see detailed results of both count vectorizer and TFIDF vectorizer and compare which one performs better. Now, let’s compare both the techniques and find which one performs better.

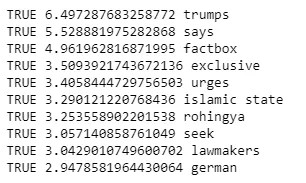
**Findings and discussion**

As per the above Table, Passive Aggressive classifier with TFIDF outperforms others in terms of accuracy, precision, and F1-score whereas the highest recall achieved by Multinomial Naïve Bayes which is really surprising. The important thing to notice that after applying TFIDF significant improvement observed in the performance of all three algorithms in comparison to the count vectorizer. In the case of count vectorizer, Logistic Regression outperforms others in all the performance measures except recall.

**Feature Importance**

Now, let’s see what are the most informative features in predicting Fake and True news with the TFIDF vectorizer. The top 10 key features in predicting fake news are shown below.

The top 10 key features in predicting true news are shown below.



As we can see from the above figures, the first column is the label the second column consists of coefficient and the last one are the top contributing features. From the figure, it is evident that top features in the case of Fake News are the lowest negative value whereas in the case of true news highest positive value.

In most of the top fake and true news, only unigrams are present while in case of true news one bigram feature is also present i.e., Islamic state.