FAKE NEWS PROJECT REPORT

Submitted by:

NEETAL TIWARI

Business Problem Framing

Fake News has become one of the major problems in the existing society. Fake News has high potential to change opinions, facts and can be the most dangerous weapon in influencing society. The proposed project uses NLP techniques for detecting the 'fake news', that is, misleading news stories which come from the non-reputable sources. By building a model based on a Decision Tree Classifier algorithm, the fake news can be detected. The data science community has responded by taking actions against the problem. It is impossible to determine a news as real or fake accurately. So, the proposed project uses the datasets that are trained using count vectorizer method for the detection of fake news and its accuracy will be tested using machine learning algorithms.

Data Sources and their formats

We can see in fake project dataset 5 columns are there.

The data set includes:

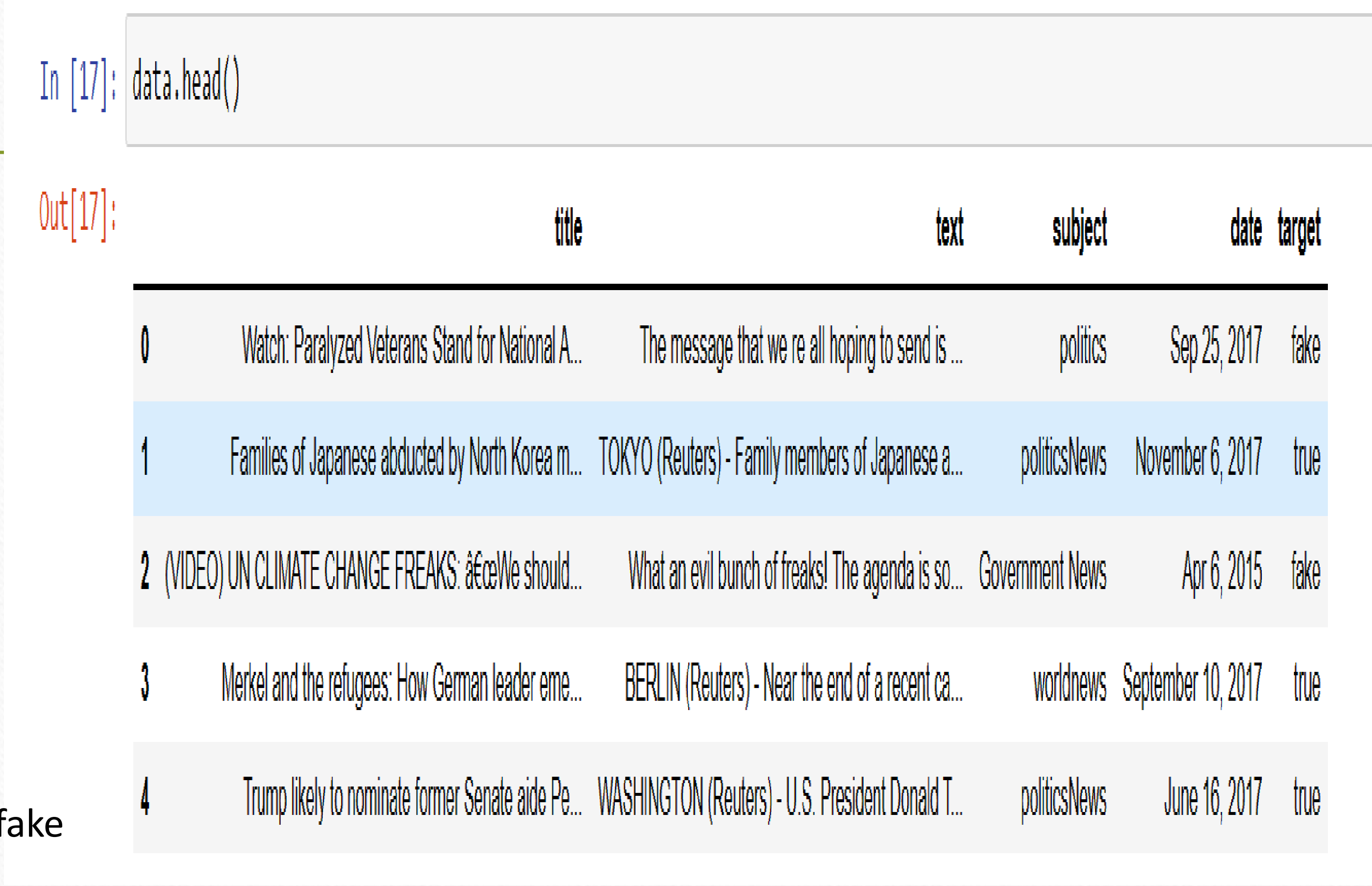
title: The Title of a news article

text: The text of the title

subject: Subject of the news article

date: Date of the News article

target: A label that marks the article is fake or true



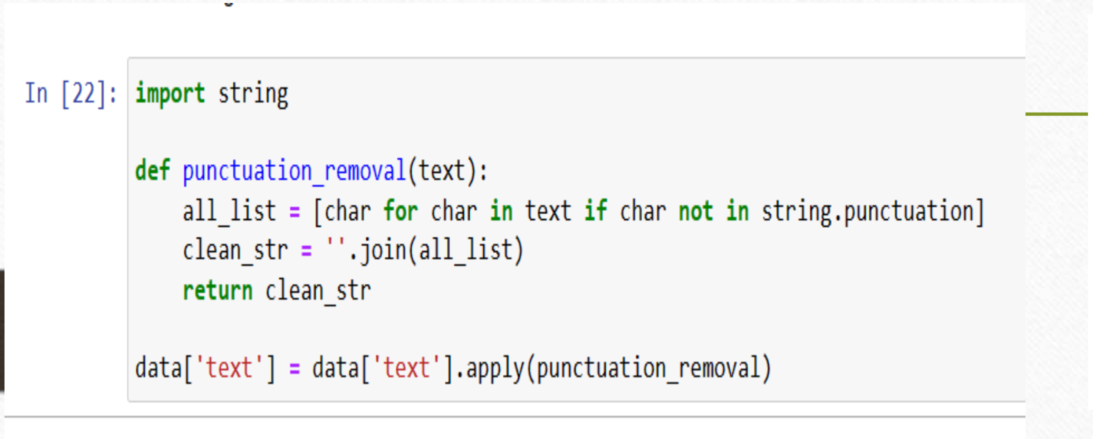
```
In [17]: data.head()
```

Out[17]:

	title	text	subject	date	target
0	Watch: Paralyzed Veterans Stand for National A...	The message that we re all hoping to send is ...	politics	Sep 25, 2017	fake
1	Families of Japanese abducted by North Korea m...	TOKYO (Reuters) - Family members of Japanese a...	politicsNews	November 6, 2017	true
2	(VIDEO) UN CLIMATE CHANGE FREAKS: â€œWe should...	What an evil bunch of freaks! The agenda is so...	Government News	Apr 6, 2015	fake
3	Merkel and the refugees: How German leader eme...	BERLIN (Reuters) - Near the end of a recent ca...	worldnews	September 10, 2017	true
4	Trump likely to nominate former Senate aide Pe...	WASHINGTON (Reuters) - U.S. President Donald T...	politicsNews	June 16, 2017	true

Data Pre-processing Done

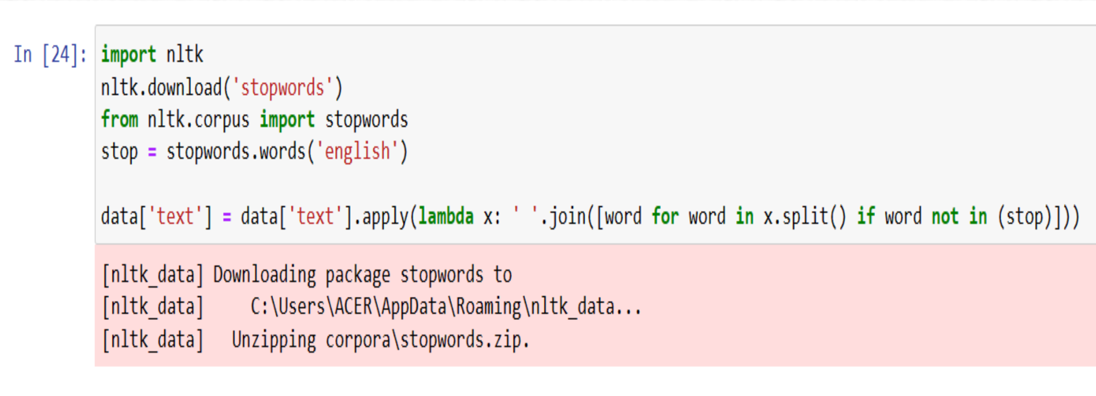
Removing punctuation



```
In [22]: import string

         def punctuation_removal(text):
             all_list = [char for char in text if char not in string.punctuation]
             clean_str = ''.join(all_list)
             return clean_str

         data['text'] = data['text'].apply(punctuation_removal)
```

The punctuation removal process will help to treat each text equally. For example, the word data and data! are treated equally after the process of removal of punctuations.

STOP WORD REMOVAL

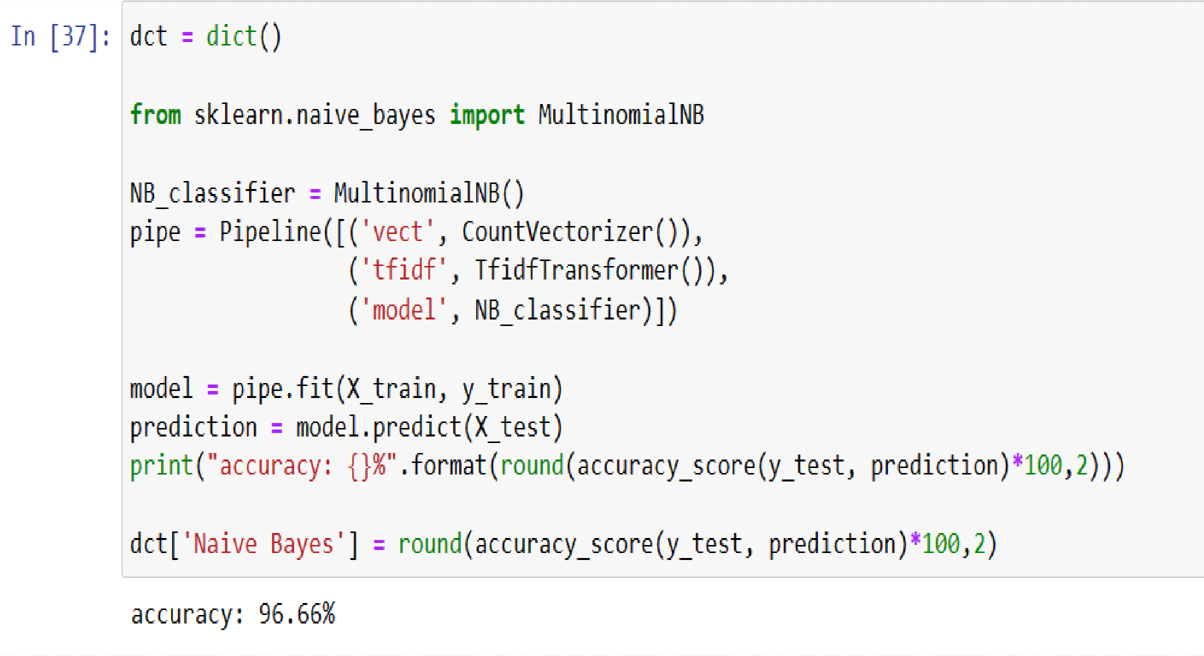


```
In [24]: import nltk
         nltk.download('stopwords')
         from nltk.corpus import stopwords
         stop = stopwords.words('english')

         data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))

         [nltk_data] Downloading package stopwords to
         [nltk_data]     C:\Users\ACER\AppData\Roaming\nltk_data...
         [nltk_data]   Unzipping corpora\stopwords.zip.
```

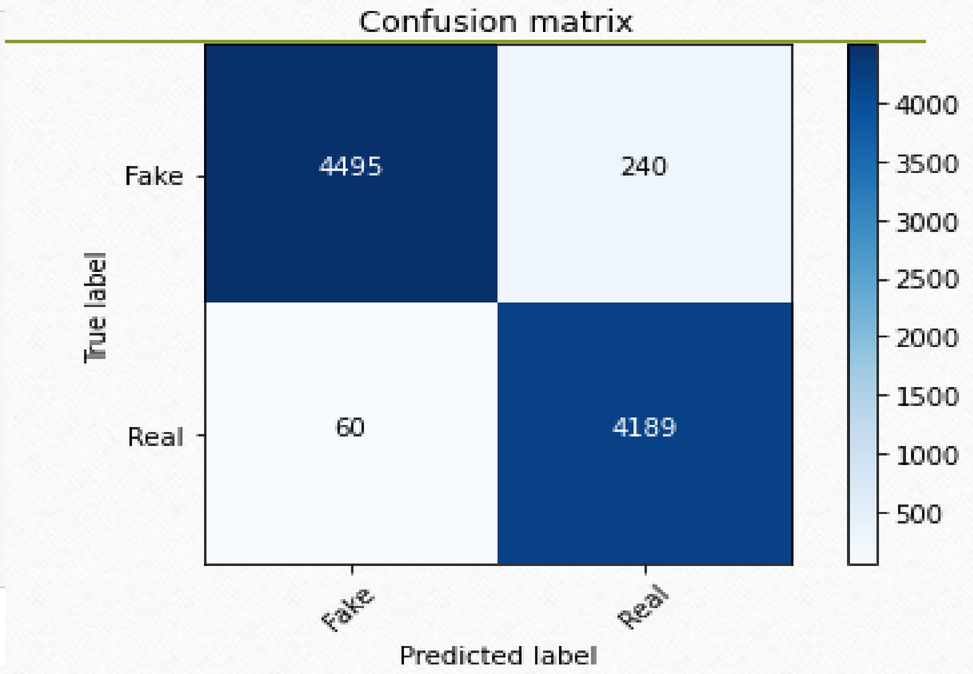
A Stop Word is a commonly used word in any natural language such as “a, an, the, for, is, was, which, are, were, from, do, with, and, so, very, that, this, no, yourselves etc....”. These Stop Words will have a very high frequency and so these should be eliminated while calculating the term frequency so that the other important things are given priority. Stop word removal is such a Pre-processing step which removes these stop words and thereby helping in the further steps and also reducing some processing time because the size of the document decreases tremendously.

Model/s Development and Evaluation

For data analysis I had used five algorithms such as Naive Bayes, Logistic regression, DecisionTreeClassifier, RandomForestClassifier, SVM

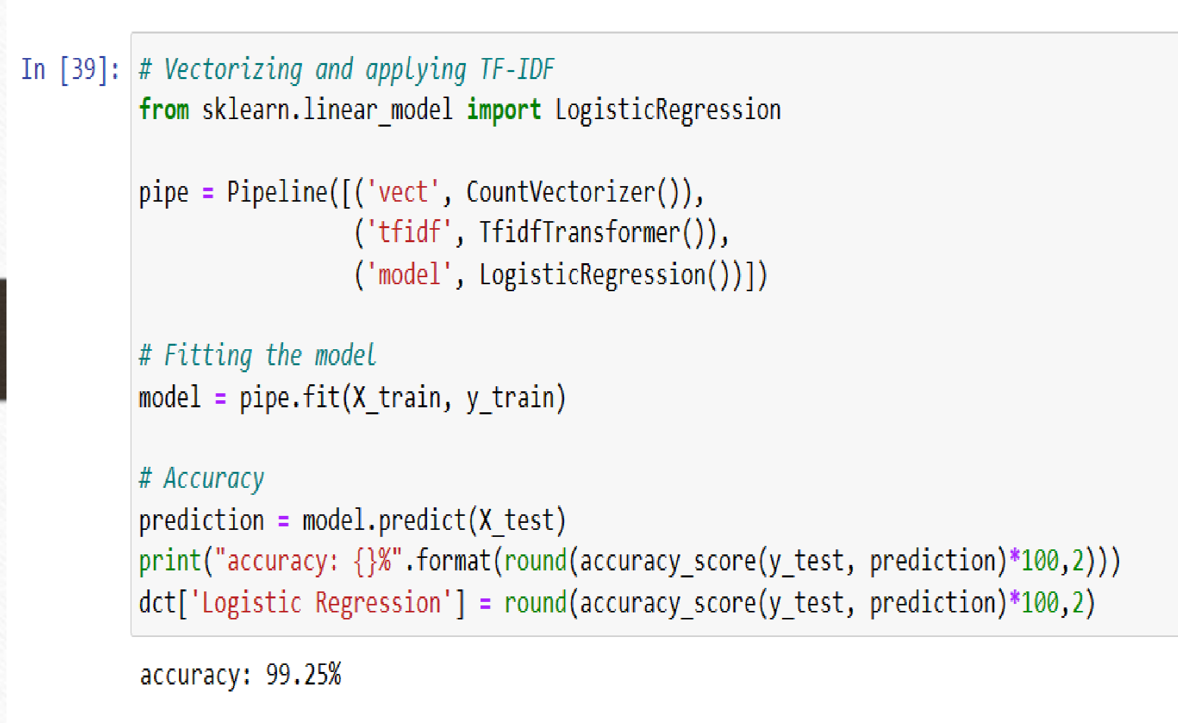


```
In [37]: dct = dict()

         from sklearn.naive_bayes import MultinomialNB

         NB_classifier = MultinomialNB()
         pipe = Pipeline([('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer()),
                          ('model', NB_classifier)])

         model = pipe.fit(X_train, y_train)
         prediction = model.predict(X_test)
         print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))

         dct['Naive Bayes'] = round(accuracy_score(y_test, prediction)*100,2)

         accuracy: 96.66%
```

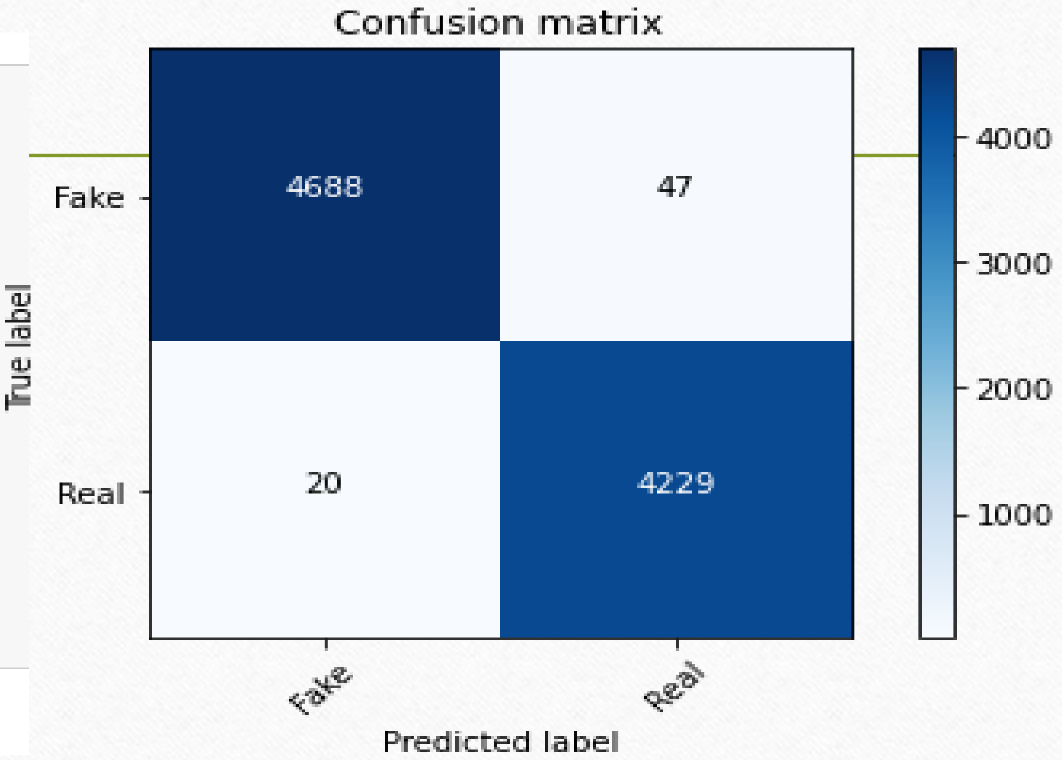


Accuracy score of NAIVE BAYES is 96.66%

Model/s Development and Evaluation

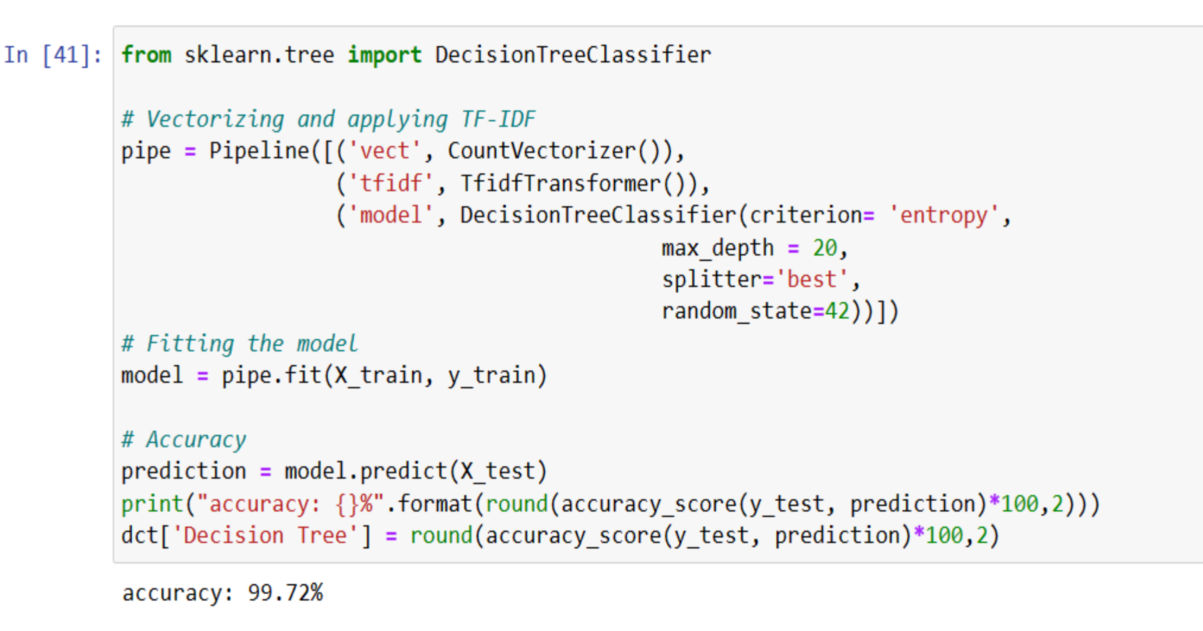


```
In [39]: # Vectorizing and applying TF-IDF
         from sklearn.linear_model import LogisticRegression

         pipe = Pipeline([('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer()),
                          ('model', LogisticRegression())])

         # Fitting the model
         model = pipe.fit(X_train, y_train)

         # Accuracy
         prediction = model.predict(X_test)
         print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
         dct['Logistic Regression'] = round(accuracy_score(y_test, prediction)*100,2)

         accuracy: 99.25%
```

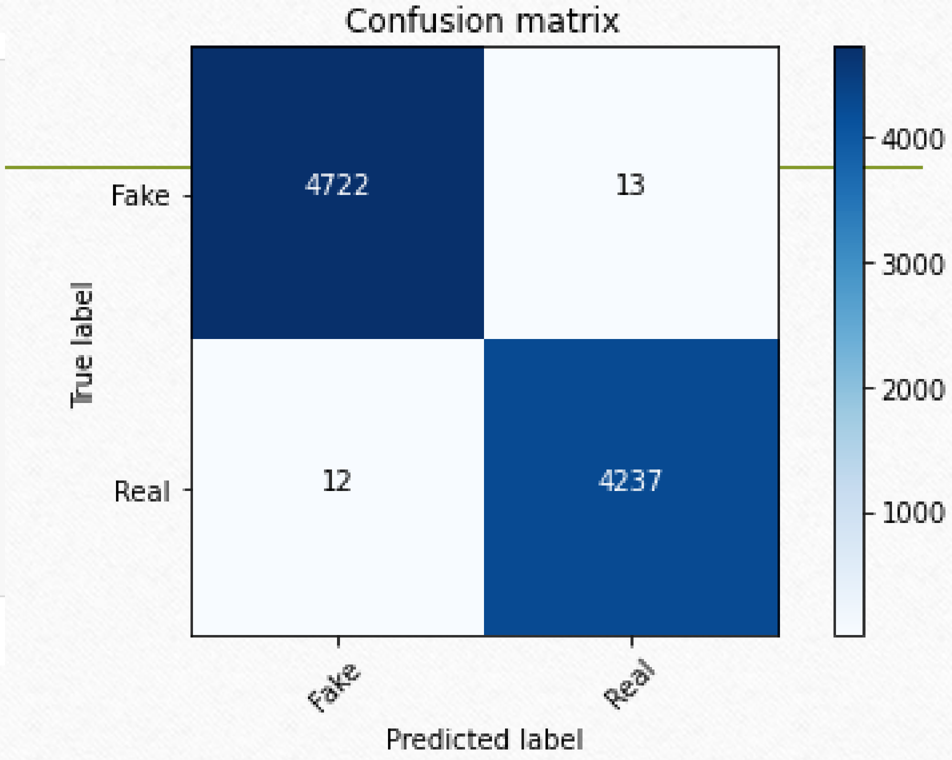


Accuracy score of Logistic regression is 99.25%

Model/s Development and Evaluation

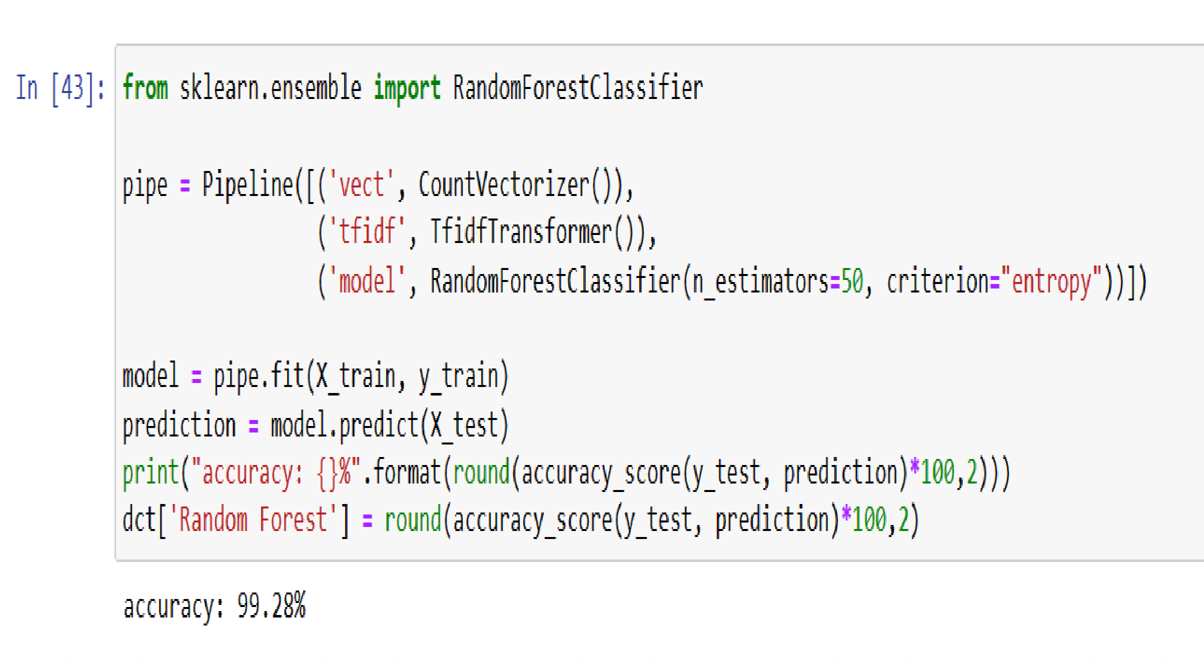


```
In [41]: from sklearn.tree import DecisionTreeClassifier

         # Vectorizing and applying TF-IDF
         pipe = Pipeline([('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer()),
                          ('model', DecisionTreeClassifier(criterion= 'entropy',
                                                 max_depth = 20,
                                                 splitter='best',
                                                 random_state=42))])
         # Fitting the model
         model = pipe.fit(X_train, y_train)

         # Accuracy
         prediction = model.predict(X_test)
         print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
         dct['Decision Tree'] = round(accuracy_score(y_test, prediction)*100,2)

         accuracy: 99.72%
```

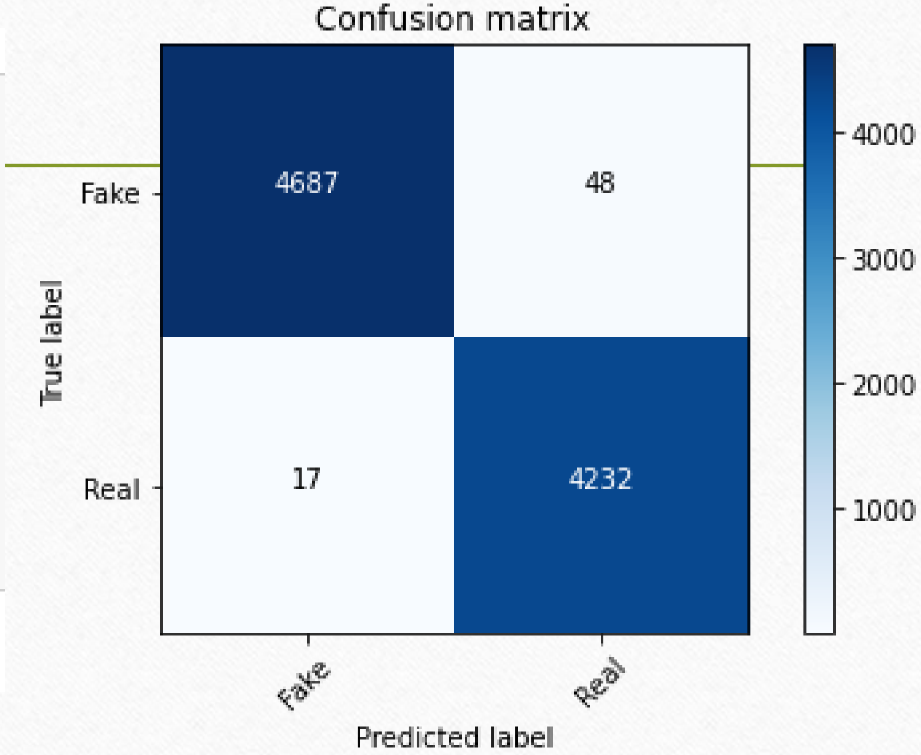


Accuracy score of Decision tree classifier is 99.72%

Model/s Development and Evaluation

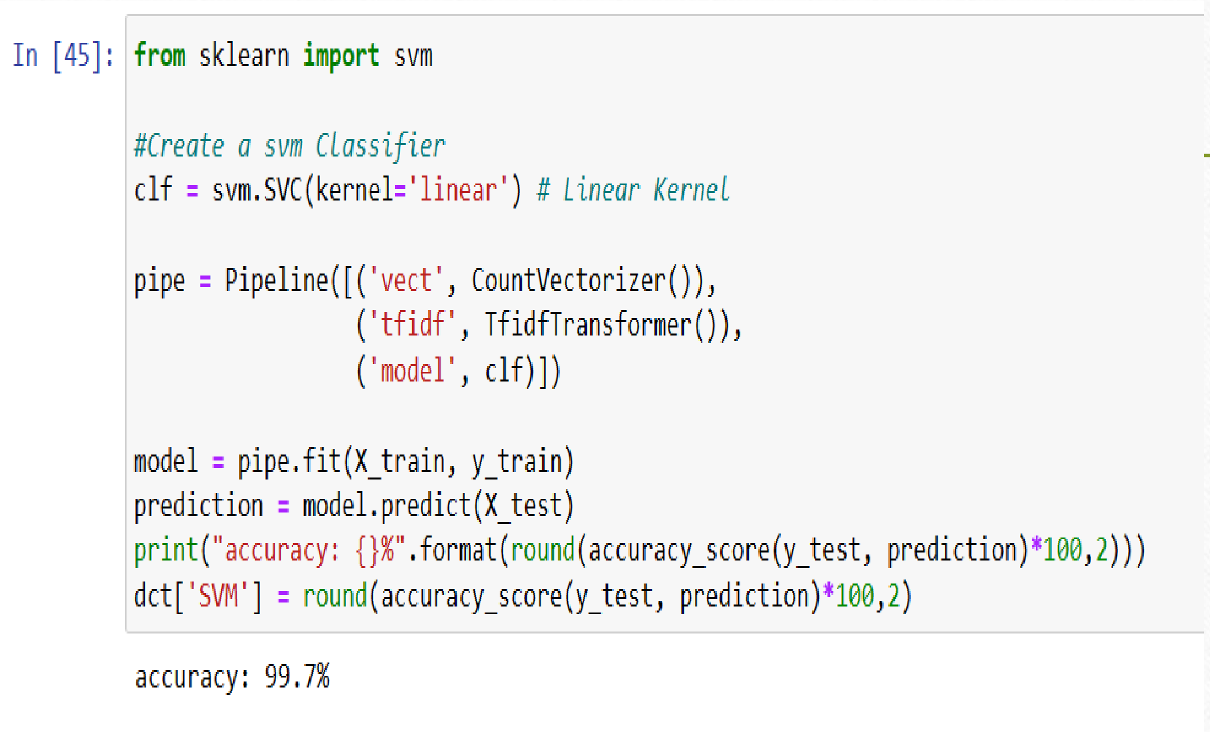


```
In [43]: from sklearn.ensemble import RandomForestClassifier

         pipe = Pipeline([('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer()),
                          ('model', RandomForestClassifier(n_estimators=50, criterion="entropy"))])

         model = pipe.fit(X_train, y_train)
         prediction = model.predict(X_test)
         print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
         dct['Random Forest'] = round(accuracy_score(y_test, prediction)*100,2)

         accuracy: 99.28%
```

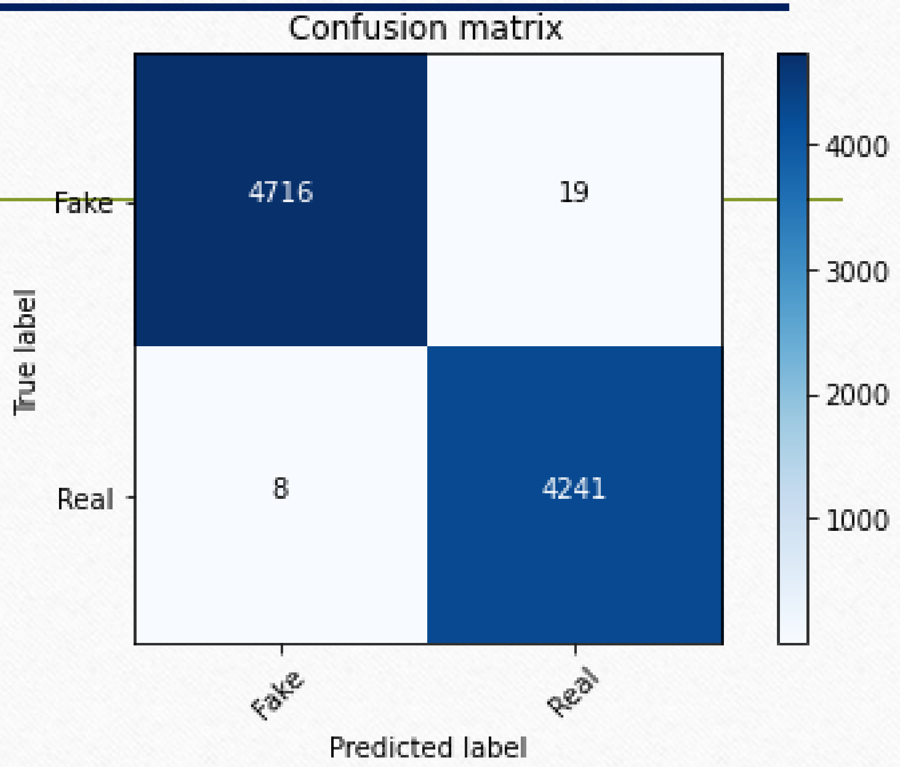


Accuracy score of Random Forest classifier is 99.28%

Model/s Development and Evaluation

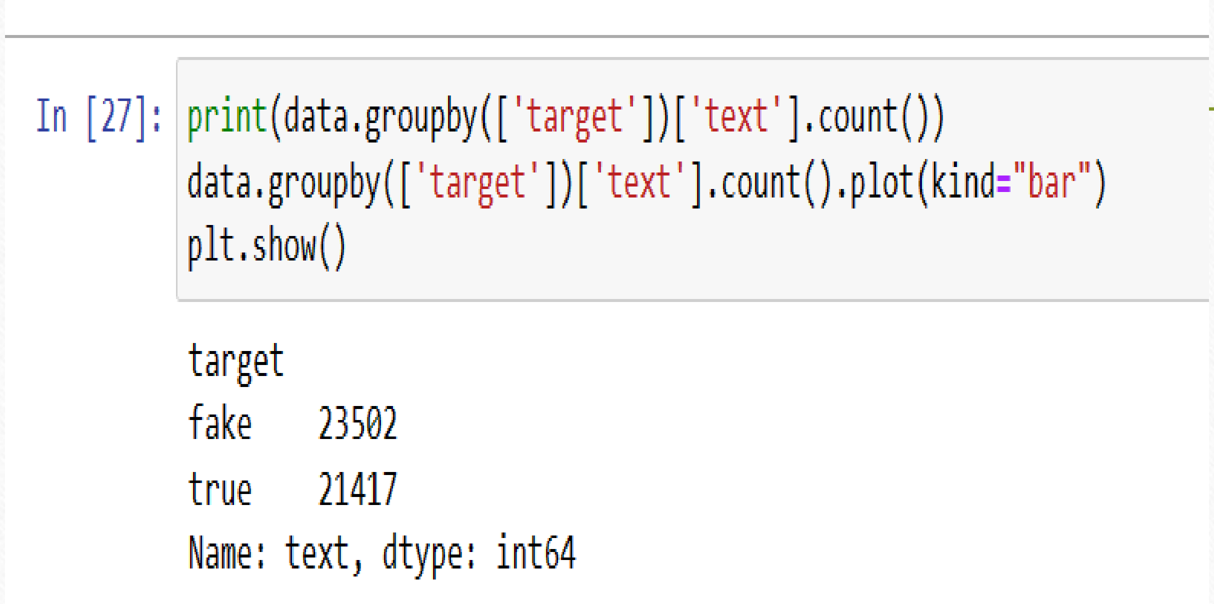


```
In [45]: from sklearn import svm

         #Create a svm Classifier
         clf = svm.SVC(kernel='linear') # Linear Kernel

         pipe = Pipeline([('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer()),
                          ('model', clf)])

         model = pipe.fit(X_train, y_train)
         prediction = model.predict(X_test)
         print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
         dct['SVM'] = round(accuracy_score(y_test, prediction)*100,2)

         accuracy: 99.7%
```

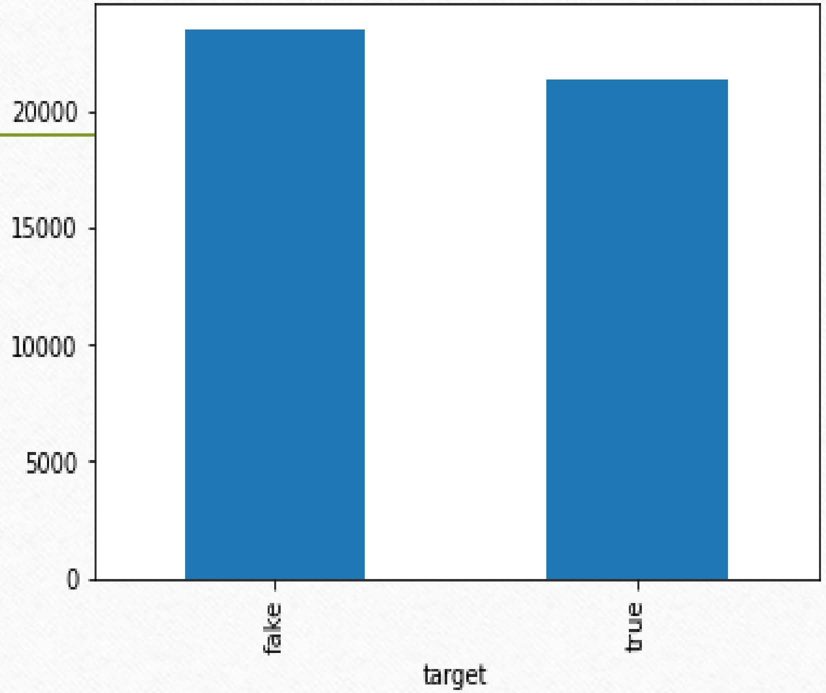


Accuracy Score of SVM is 99.7%

Visualizations

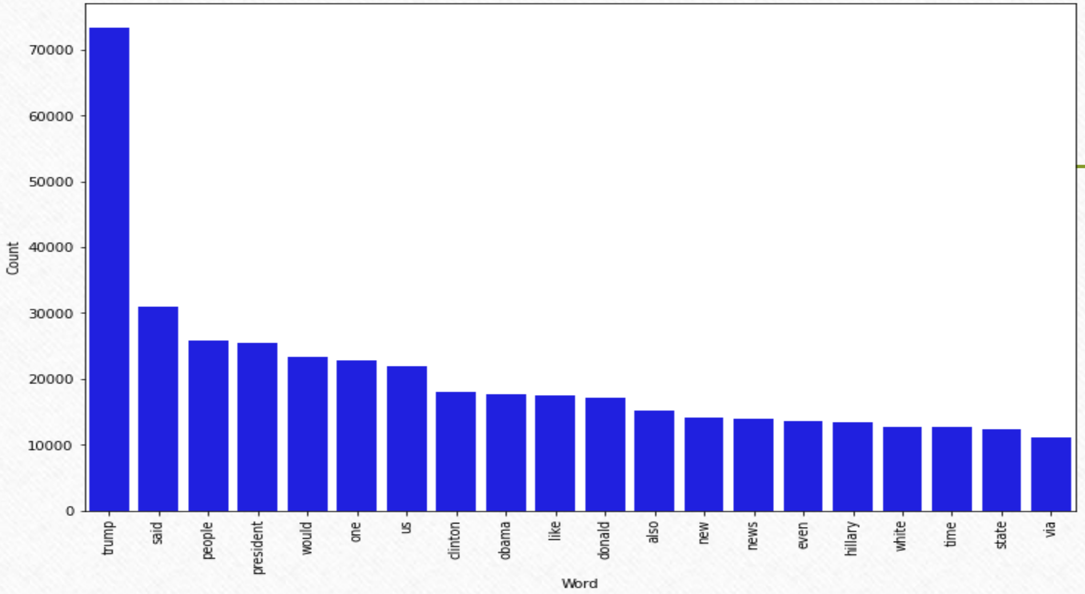


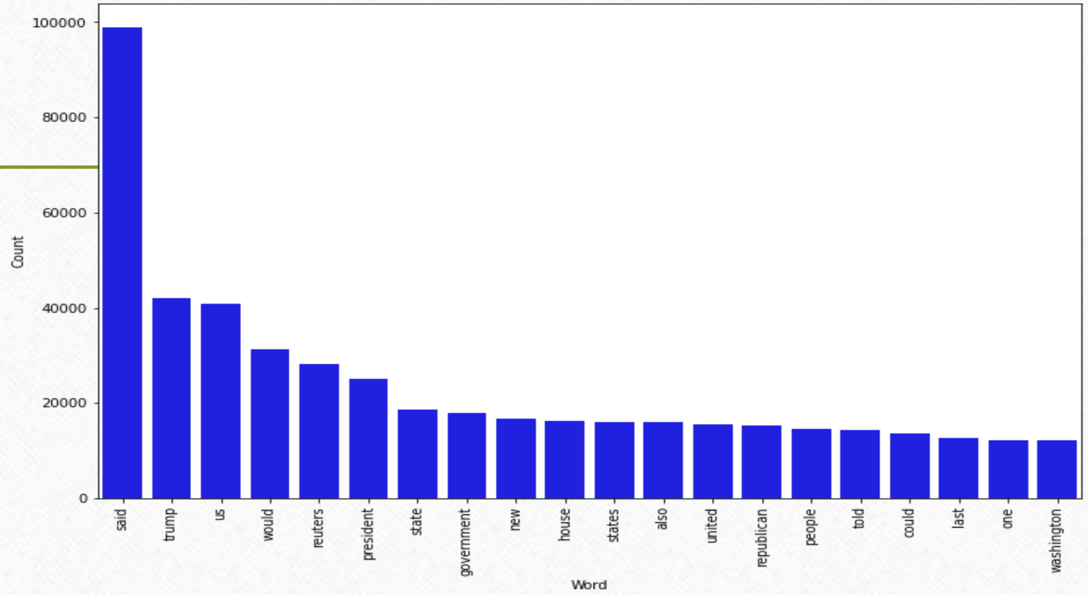
```
In [27]: print(data.groupby(['target'])['text'].count())
         data.groupby(['target'])['text'].count().plot(kind="bar")
         plt.show()

         target
         fake    23502
         true    21417
         Name: text, dtype: int64
```



fake news is 23502 and true news are 21417.

Visualizations





These all are most frequent words in fake news. trump word is highly used in fake news.

These all are most frequent words in true news. said word is highly used in true news.

CONCLUSION

Our project can ring the initial alert for fake news. The model produces worse results if the article is written cleverly, without any denationalization. This is a very complex problem but we tried to address it as much as we could. We believe the interface provides an easier way for the average person to check the authenticity of a news. Projects like this one with more advanced features should be integrated on social media to prevent the spread of fake news.