Contents

Abstract

This Project comes up with the applications of NLP (Natural Language Processing) techniques for detecting the 'fake news, that is, misleading news stories that comes from the non-reputable sources. Only by building a model based on a count vectorizer (using word tallies) or a (Term Frequency Inverse Document Frequency) tfidf matrix, (word tallies relative to how often they're used in other articles in your dataset) can only get you so far. But these models do not consider the important qualities like word ordering and context. It is very possible that two articles that are similar in their word count will be completely different in their meaning. The data science community has responded by taking actions against the problem.

Introduction

Motivation and Overview

Fake news denotes a type of yellow press which intentionally presents misinformation or hoaxes spreading through both traditional print news media and recent online social media. Fake news has been existing from a long time. In recent years, due to the booming developments of online social networks, fake news for various commercial and political purposes has been appearing in large numbers and widespread in the online world. With deceptive words, online social network users can get infected by this online fake news easily, which has brought about tremendous effects on the offline society already. This project helps in resolving this problem by detecting whether the news is fake or not.

Objective

The main objective is to detect the fake news, which is a classic text classification problem with a straight forward proposition. It is needed to build a model that can differentiate between “Real” news and “Fake” news.

Problem Statement

Study the fake news detection problem in online social networks. Based on various types of heterogeneous information sources, including both textual contents/profile/descriptions and the authorship and article subject relationships among them, we aim at identifying fake news from the online social networks simultaneously. Develop a machine learning program to identify when a news source may be producing fake news based on content acquired. We aim to use a corpus of labelled real and fake new articles to build a classifier that can make decisions about information based on the content from the corpus. The model will focus on identifying fake news.

Requirements

Hardware:

- Computer/Laptop
- 8GB RAM

Software:

- Operating System(Windows)
- Jupyter
- Anaconda Navigator

Implementation

---

1. Dataset Description

- train.csv: A full training dataset with the following attributes:
  - id
  - title
  - text
  - label
- test.csv: A testing training dataset with all the same attributes at train.csv without the label.

2. Dependencies Needed

- python 3.6+
- numpy
- pandas
- matplotlib
- scikitplot
- sklearn

3. Operations

- Train and test dataset.
- Data Preprocessing
  - Perform various text cleaning steps (remove all non-alphanumeric characters, delete stop words, delete missing rows, etc.)

- **Dataset**

```
In [4]: df.head(10)
```

Out[4]:

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL
5	6903	Tehran, USA	\nI'm not an immigrant, but my grandparents ...	FAKE
6	7341	Girl Horrified At What She Watches Boyfriend D...	Share This Baylee Luciani (left), Screenshot o...	FAKE
7	95	'Britain's Schindler' Dies at 106	A Czech stockbroker who saved more than 650 Je...	REAL
8	4869	Fact check: Trump and Clinton at the 'commande...	Hillary Clinton and Donald Trump made some ina...	REAL
9	2909	Iran reportedly makes new push for uranium con...	Iranian negotiators reportedly have made a las...	REAL

i) Tf-idf

Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining.

- **TF: Term Frequency**, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length as away of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

- **IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

  IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

```
In [13]: 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)
```

ii) HashVectorizer

Convert a collection of text documents to a matrix of token occurrences. It turns a collection of text documents into a scipy.sparse matrix holding token occurrence counts (or binary occurrence information), possibly normalized as token frequencies if norm='l1' or projected on the Euclidean unit sphere if norm='l2'. This text vectorizer implementation uses the hashing trick to find the token string name to feature integer index mapping.

This strategy has several advantages:

It is very low memory scalable to large datasets as there is no need to store a vocabulary dictionary in memory.

It is fast to pickle and un-pickle as it holds no state besides the constructor parameters.

It can be used in a streaming (partial fit) or parallel pipeline as there is no state computed during fit.

```
In [14]: hash_vectorizer = HashingVectorizer(stop_words='english', non_negative=True)
         hash_train = hash_vectorizer.fit_transform(X_train)
         hash_test = hash_vectorizer.transform(X_test)
```

iii) CountVectorizer

CountVectorizer is used to transform a corpora of text to a vector of term / token counts. It also provides the capability to preprocess your text data prior to generating the vector representation making it a highly flexible feature representation module for text.

It provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

You can use it as follows:

1)Create an instance of the CountVectorizer class.

2)Call the fit() function in order to learn a vocabulary from one or more documents.

3)Call the transform() function on one or more documents as needed to encode each as a vector.

```
In [12]: count_vectorizer = CountVectorizer(stop_words='english')
         count_train = count_vectorizer.fit_transform(X_train)
         count_test = count_vectorizer.transform(X_test)
```

4. Algorithms Used

i) Multinomial Naive Bayes

MultinomialNB implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i \mid y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class y.

ii) Passive-Aggressive Classifier

Passive Aggressive Algorithm is a family of online learning algorithms proposed by Crammer at al. The Passive Aggressive (PA) algorithm is perfect for classifying massive streams of data. “Classification” is one in which there are instances in R^n and labels. The goal is to find a hyperplane that separates the positive instances from the negative instances, that is, we would like that yt(w) * xt will have a large positive value. By “large” we mean at least epsilon e we see in figure.The instance-label pair is denoted by z.

iii) Linear SVM

Linear SVM is the newest extremely fast machine learning (data mining) algorithm for solving multiclass classification problems from ultra large data sets that implements an original proprietary version of a cutting plane algorithm for designing a linear support vector machine. LinearSVM is a linearly scalable routine meaning that it creates an SVM model in a CPU time which scales linearly with the size of the training data set. Our comparisons with other known SVM models clearly show its superior performance when high accuracy is required. We would highly appreciate if you may share LinearSVM performance on your data sets with us.

Features:

•Efficiency in dealing with extra large data sets (say, several millions training data pairs)

•Solution of multiclass classification problems with any number of classes

•Working with high dimensional data (thousands of features, attributes) in both sparse and dense format

•No need for expensive computing resources (personal computer is a standard platform)

•Ideal for contemporary applications in digital advertisement, e-commerce, web page categorization, text classification, bioinformatics, proteomics, banking services and many other areas.

iv) SGD Classifier

SGD is indeed a technique that is used to find the minima of a function. SGDClassifier is a linear classifier (by default in sklearn it is a linear SVM) that uses SGD for training (that is, looking for the minima of the loss using SGD). According to the documentation:

SGDClassifier is a Linear classifiers (SVM, logistic regression, a.o.) with SGD training.

This estimator implements regularized linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). SGD allows minibatch (online/out-of-core) learning, see the partial_fit method. For best results using the default learning rate schedule, the data should have zero mean and unit variance.

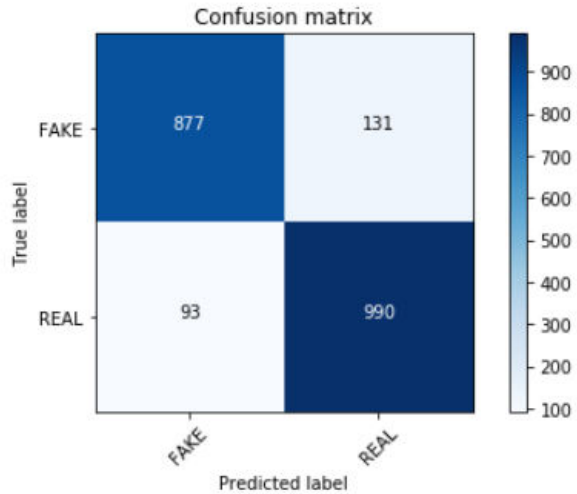
This implementation works with data represented as dense or sparse arrays of floating point values for the features. The model it fits can be controlled with the loss parameter; by default, it fits a linear support vector machine (SVM).

5. Result

i) Accuracy and Confusion Matrix of 'Multinomial Naive Bayes' using :

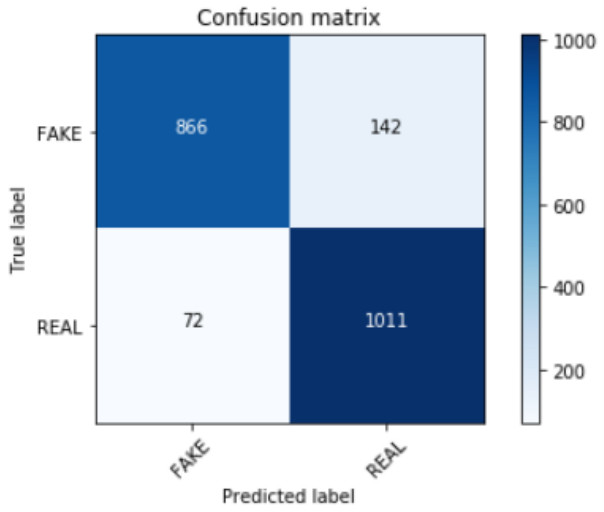
a) CountVectorizer

```
accuracy:   89.287 %
Confusion matrix, without normalization
```



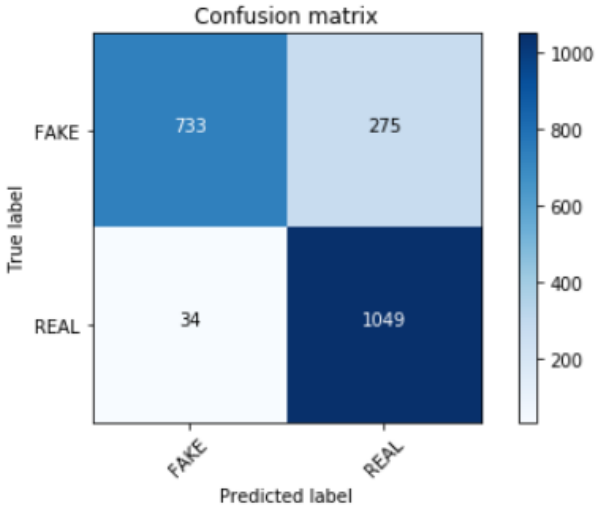
b) Tf-idf

```
accuracy:   89.766 %
Confusion matrix, without normalization
```



c) HashVectorizer

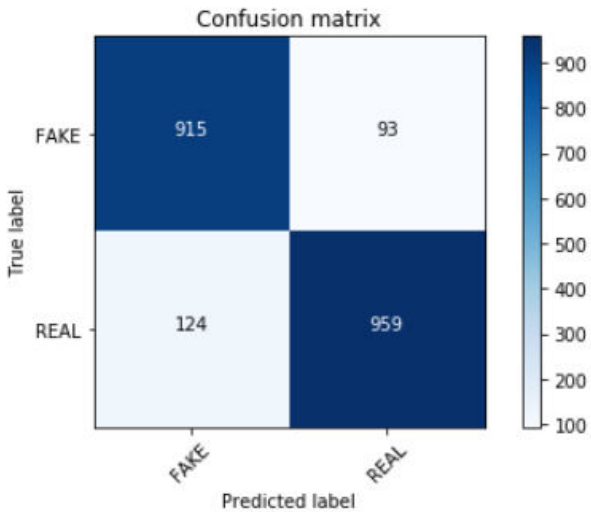
```
accuracy:   85.222 %
Confusion matrix, without normalization
```



ii) Accuracy and Confusion Matrix of 'Passive-Aggressive Classifier' using :

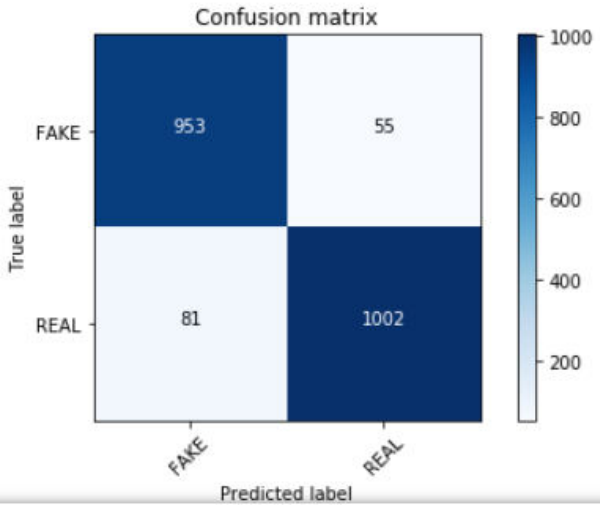
a) CountVectorizer

```
accuracy:   89.622 %
Confusion matrix, without normalization
```



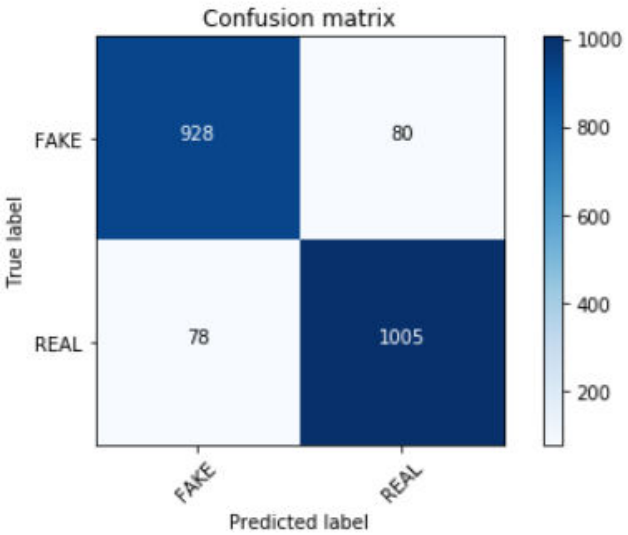
b) **Tf-idf**

```
accuracy:   93.496 %
Confusion matrix, without normalization
```



c) **HashVectorizer**

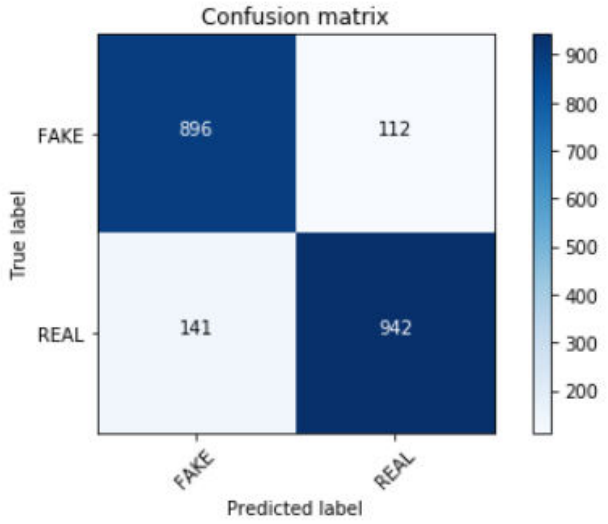
```
accuracy:   92.444 %
Confusion matrix, without normalization
```



iii) Accuracy and Confusion Matrix of 'Linear SVM' using :

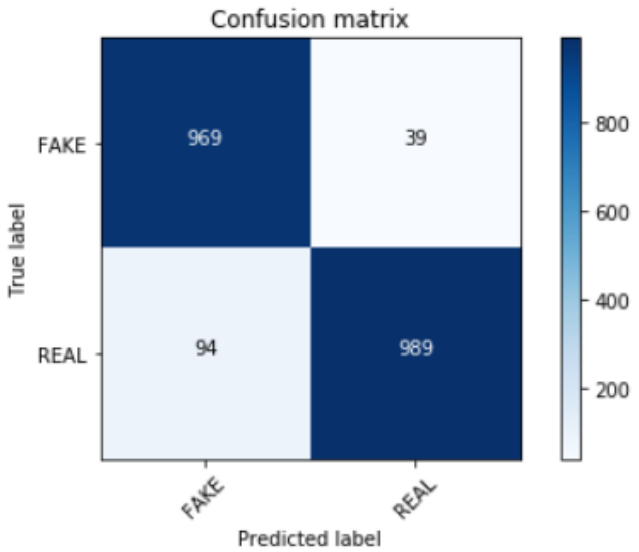
a) **CountVectorizer**

```
accuracy:   87.901 %
Confusion matrix, without normalization
```



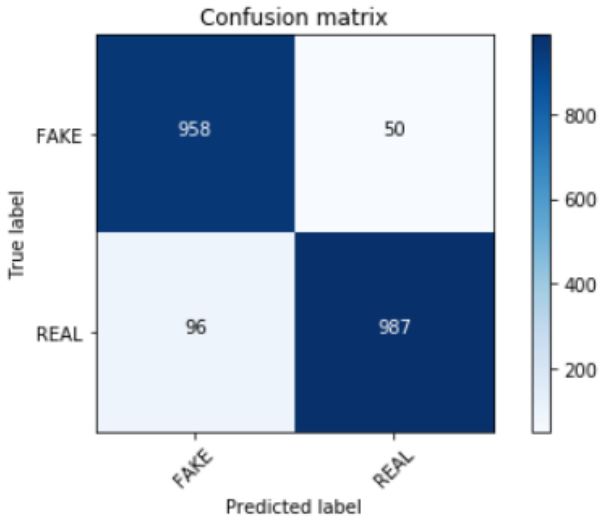
b) **Tf-idf**

```
accuracy:   93.639 %
Confusion matrix, without normalization
```



c) HashVectorizer

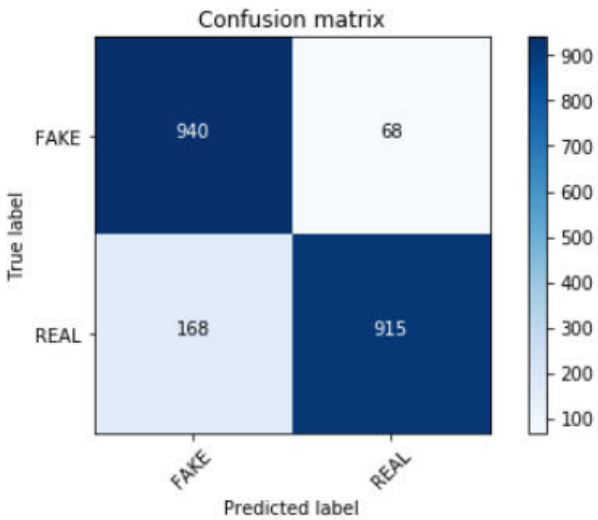
```
accuracy:   93.018 %
Confusion matrix, without normalization
```



iv) Accuracy and Confusion Matrix of 'SGD Classifier' using :

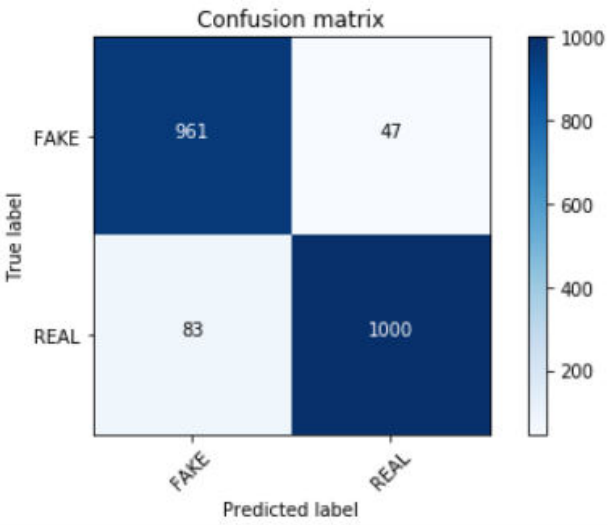
a) CountVectorizer

```
accuracy:   88.714 %
Confusion matrix, without normalization
```



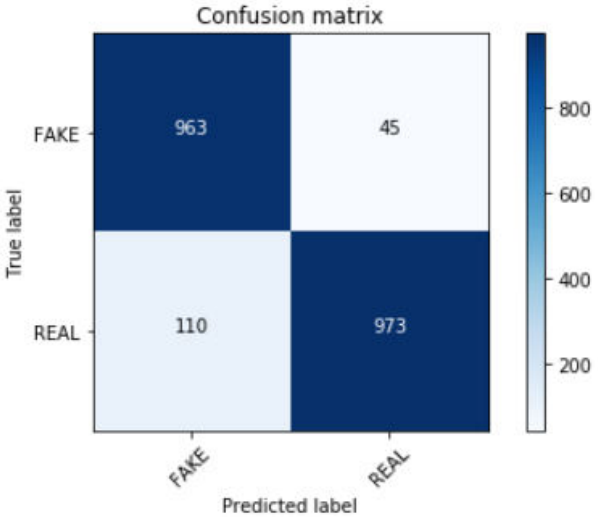
b) **Tf-idf**

```
accuracy:   93.783 %
Confusion matrix, without normalization
```



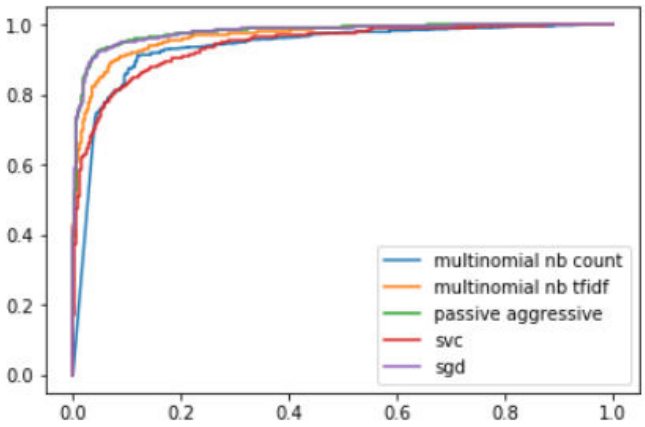
c) **HashVectorizer**

```
accuracy:   92.587 %
Confusion matrix, without normalization
```



v) **ROC Curve**

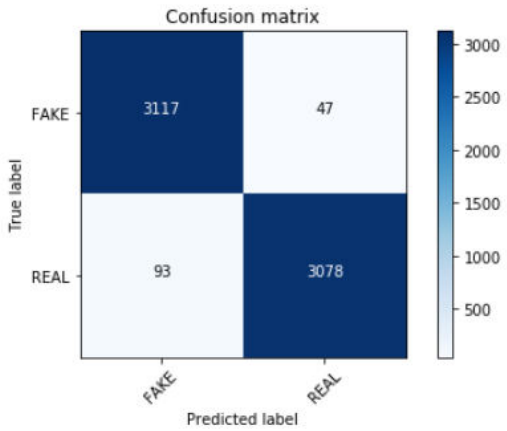
```
Out[123]: <matplotlib.legend.Legend at 0x18ba2b9da20>
```



vi) Best Fit Model

```
In [125]: predicted_value = sgd_tfidf_clf.predict(tfidf_test_for_other)
          score = metrics.accuracy_score(y_val, predicted_value)
          score=score*100
          print("accuracy:   %0.3f" % score,"%")
          cm = metrics.confusion_matrix(y_val, predicted_value, labels=['FAKE', 'REAL'])
          plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])

accuracy:   97.790 %
Confusion matrix, without normalization
```



vii) Implemented Model

```
In [162]: check=input("enter your news")
          ser=[]
          ser.append(check)
          count_for_other = count_vectorizer.transform(ser)
          tfidf_for_other = tfidf_vectorizer.transform(ser)
          hash_for_other = hash_vectorizer.transform(ser)
          predicted_by_tfifd = sgd_tfidf_clf.predict(count_for_other)

          enetr your newsprint(predicted_by_tfifd)

          C:\Users\anike\Anaconda3\lib\site-packages\sklearn\feature_extraction\hashing.py:94: DeprecationWarning: the option non_negativ
          e=True has been deprecated in 0.19 and will be removed in version 0.21.
            " in version 0.21.", DeprecationWarning)

In [163]: print(predicted_by_tfifd)

          ['FAKE']
```

Future Prospect

As this project helps in detecting the fake news, it can control the spread of fake information or messages circulated all over the social media and will save many people or sources to from being misguided. By its help, only the news that are real will be circulated.

References

https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

https://pathmind.com/wiki/word2vec

https://machinelearningmastery.com/gentle-introduction-bag-words-model/count vectorizer

https://towardsdatascience.com/natural-language-processing-count-ectorization-with-scikit-learn-e7804269bb5e

Datasets from Kaggle