MINOR PROJECT

FAKE NEWS DETECTION

PROJECT REPORT

Jaypee Institute of Information Technology – Noida

BATCH –B11			
S.NO.	**ENROLLMENT NUMBER**	**STUDENT NAME**	**WORK DONE (IN PERCENTAGE OUT OF 100) / CONTRIBUTION OF EACH MEMBER IN GROUP**
1.)	16104013	AJAY KUMAR KUSHWAHA	50
2.)	16104014	NANDINI AGRAWAL	50

For the partial fulfilment for the award of Degree of Bachelor of Technology (I.T.)

ACKNOWLEDGEMENT

This project consumed huge amount of work, research and dedication. Still, implementation would not have been possible if we did not have support of many individuals, therefore we would like to extend our sincere gratitude to all of them.

We are pleased to acknowledge **Gagandeep Kaur** Ma'am for their invaluable guidance and devoting their time and knowledge in the implementation, during the course of this project work.

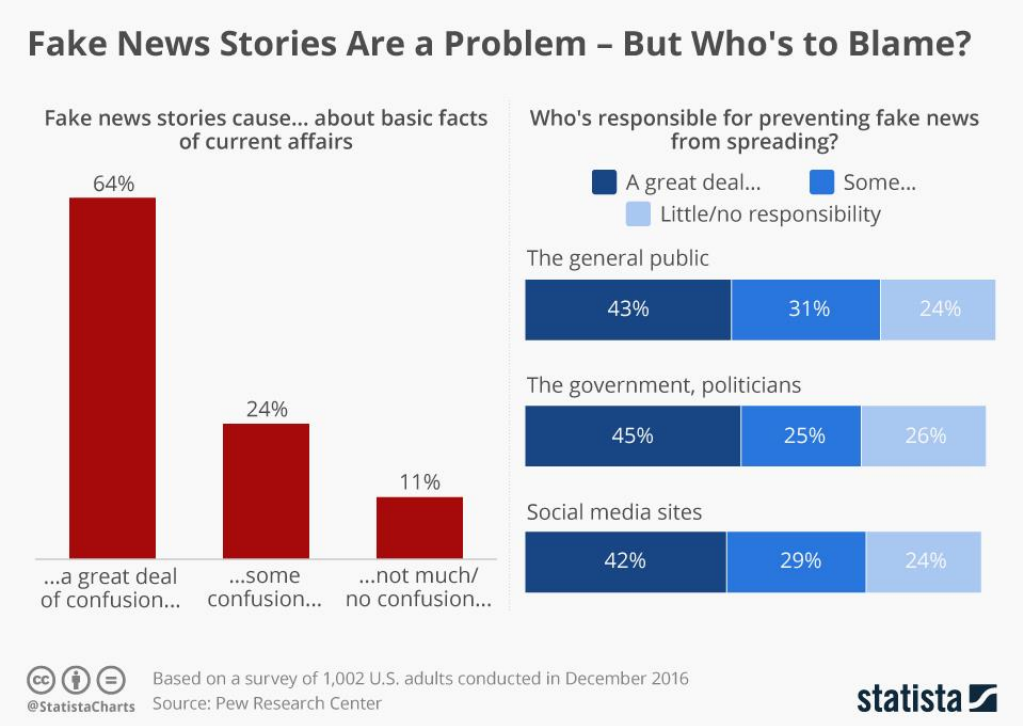
My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped me out with their abilities.

CONTENTS

PROBLEM STATEMENT

As an increasing amount of our lives is spent interacting online over the internet, more and more people tend to seek and consume news from social media, news agency homepages, search engines. On the other hand, it enables the proliferation of “fake news", i.e., low quality news with intentionally false information. Popular social media platforms such as Facebook, twitter have proven to be an effective means of channels for spreading these false news due to their wide reach and the speed in which information is spread.

The plague of fake news not only poses serious threats to the integrity of journalism, but has also created turmoil in the political world. The term ‘fake news’ became common parlance for the issue, particularly to describe factually incorrect and misleading articles published in social media feeds, news blogs and online newspapers, mostly for the purpose of making money through page views. The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Fake news is intentionally written to mislead readers to believe false information, which makes it difficult and nontrivial to detect based on news content. Therefore, the issue of fake new detection is both challenging and crucial. Hence, we decide to take up this challenge and find solution in an efficient way.



PREQUISITE

SOFTWARE USED:-

- **Jupyter Notebook** - The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

- **MongoDb** - MongoDB is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemata.

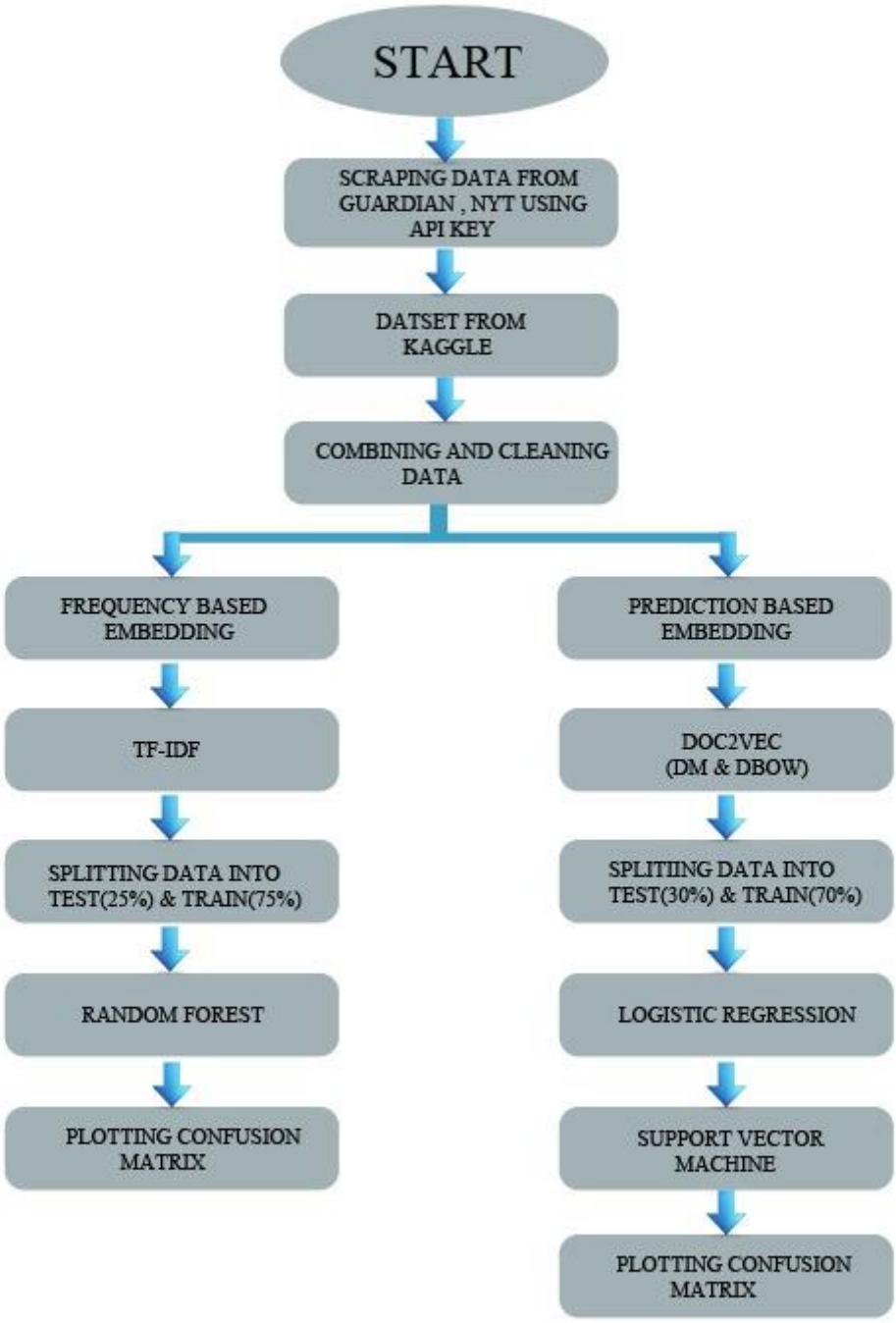
LANGUAGE USED:-

- PYTHON
- MongoDb
- NLP(Natural Language Processing)
- MACHINE LEARNING (concepts)

LIBRARIES USED:-

- BeautifulSoup – a python library to for pulling data out of HTML and XML files. To scrape or extract all the data from a particular website
- pymongo - To establish connection between python code and mongo database. It is used to retrieve the data from MongoDB and store locally.
- json - it is used to retrieve json data from the website
- requests - to get the requested URL for fetching the contents from the URL.
- time and datetime - To get the current date and tim relativedelta- to calculate the absolute year, months, days, time between two given dates.
- pandas - for reading the csv and converting it to dataframe
- numpy - for array computation
- re- for regular expression
- sklearn- scikit learn for shuffling the data, performing cross validation, performance metrics
- seaborn & matplotlib - for plotting graphs
- nltk (Natural Language Toolkit) –suite of libraries and programs for symbolic and statistical natural language processing in python.
- Gensims-python library for topic modelling, document indexing and similarity retrieval for large corpora.
- Tqdm-shows the progress of any iteration of epochs.
- SVM –python library used for multiclass classification

FLOW DIAGRAM



Data Collection and Preparation

The data used for this project was drawn from three different sources. The real news datasets were collected from **New York Times** and **The Guardian News** articles and **Kaggle.**

We used NYT API provided by New York Times, an American newspaper. For collecting data from NYT, we obtained API key from their website, used that API key to get access to their database and then we scraped those articles. We use **BeautifulSoup** and **Pymongo** inbuilt python libraries for scraping and mongodb connectivity, which visits the given URL and collects all the data from that page. Then we cleaned the scraped data by removing unwanted and unnecessary columns and then storing the cleaned data in CSV file we finally got 14 MB data with 3000 rows. We have also calculated the **TF-IDF scores**, which tells the frequency of each term in the entire dataset. We visualized the most tfidf score words were Brett kavanaugh, china, Florence, ryder cup and trump etc.

For **Guardian Dataset**, we used the API key obtained from their website and accessed the dataset of articles. We obtained the data in json format. The cleaning process for this Guardian Dataset involves, reading the json data and putting in a dataframe, extracting headlines and body from the dataframe, removing all unwanted news sections, removing empty spaces and finally storing it in a csv file as done above. At last we got data with 13000 rows.

The fake news dataset is obtained from **Kaggle.com**. This dataset is cleaned and filtered to get only English language, to remove unwanted columns and empty coloumns. The fakeness attribute in the dataset will indicate 0 for real news and 1 for fake news We combine all these fake and real news articles into a single csv file for implementation. The **TF-IDF scores**, which tells the frequency of each term in the entire dataset. We visualized the most tfidf score words were TRUMP, CLINTON, ELECTION, and VIDEO etc

The attributes of the final dataset are:
Id: A unique identifier for the article
Publication date: Article's publication date
Headline: Headline of the article
Body: The textual content of the article
Fakeness: binary 0/1 for real and fake news.

DATA CLEANING

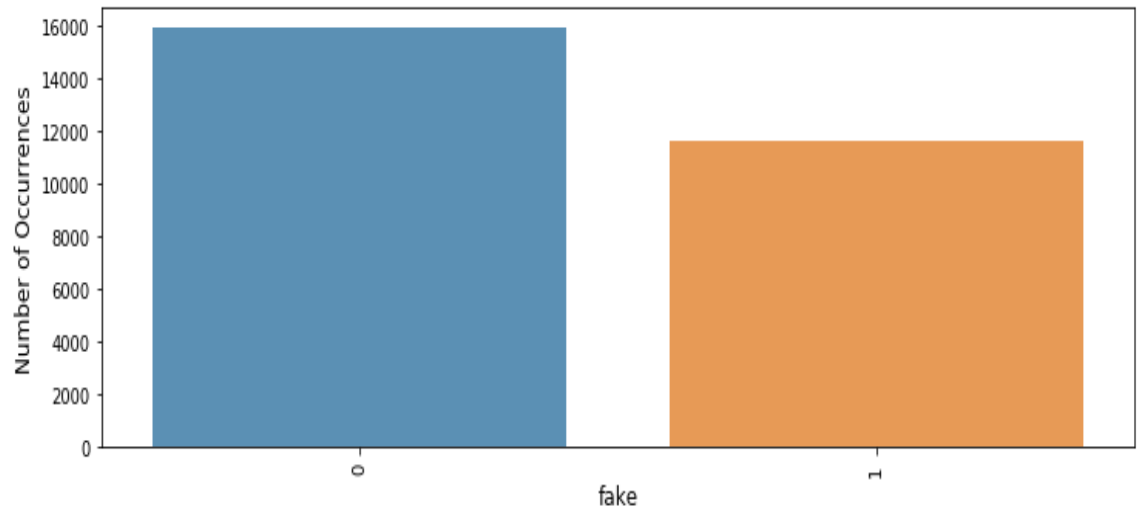
This dataset is cleaned and filtered to get only English language, to remove unwanted columns and NULL values. The fakeness attribute in the dataset will indicate 0 for real news and 1 for fake news.

KAGGLE- Kaggle dataset initially had 12999 rows and 8 columns. Which is cleaned by dropping various columns and rows which ad null values. We only kept columns of interest like author , title, text, language, site_url, uuid. After we had 11677 rows. We further added a new column of 'fakeness' and defined it to 1.

GUARDIAN- We collected all the json articles crapped from the website and combined the whole data into one and converted into csv file. Again we have dropped the unnecessary columns and rows with null values. We initially had 59151 rows and 14 columns out of which we have kept only 'business', 'world news', 'politics', 'US news ' section news. we were left with 13071 rows.

NYT- Data was scrapped using mongodb and was downloaded in the format of json documents and then was converted into csv files. Initially it had 2933 rows and 4 columns. It was observed that data was already very much clean and hence it had only 2900 rows and 4 columns. And then fakeness column added to it as usual.

All the three datasets were then combined and saved into one common csv file named "final.csv" where column names of all the datasets were made common and saved.



ALGORITHMS

Word Embedding:- Word Embeddings are the texts converted into numbers and there may be different numerical representations of the same text. A Word Embedding format generally tries to map a word using a dictionary to a vector, in more simpler terms it finds out the relation between different text written in context.

The different types of word embeddings can be broadly classified into two categories-

1. Frequency based Embedding
2. Prediction based Embedding

Random Forest:-

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks. Random Forest is a supervised learning algorithm. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. Sklearn provides a great tool for this, that measures a features importance by looking at how much the tree nodes, which use that feature, reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results, so that the sum of all importance is equal to 1.

Logistic Regression:-

Logistic regression is named for the function used at the core of the method, the logistic function, also called the sigmoid function. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms. In logistic regression, the dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 (TRUE, success, pregnant, etc.) or 0 (FALSE, failure, non-pregnant, etc.).

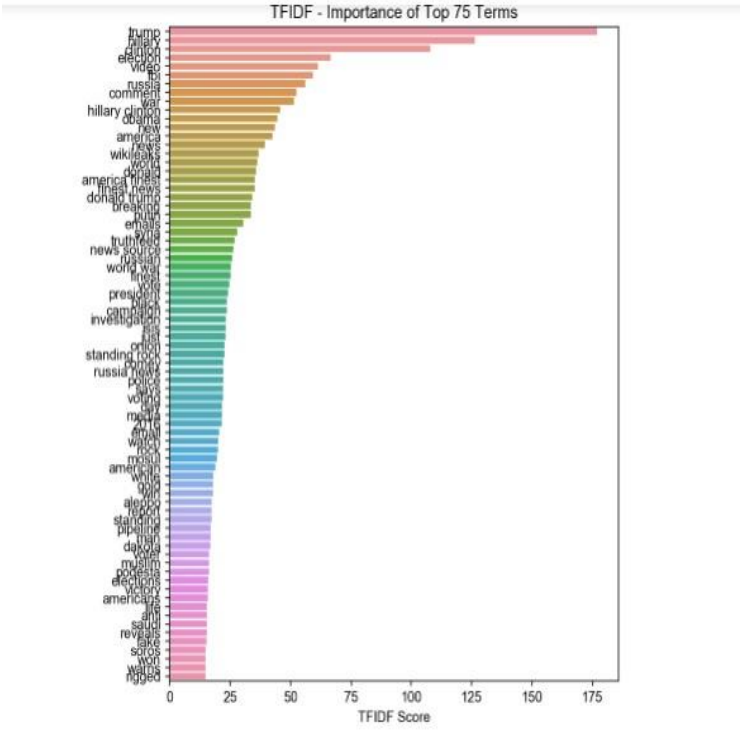
Frequency based Embedding:-

There are generally three types of vectors that we encounter under this category.

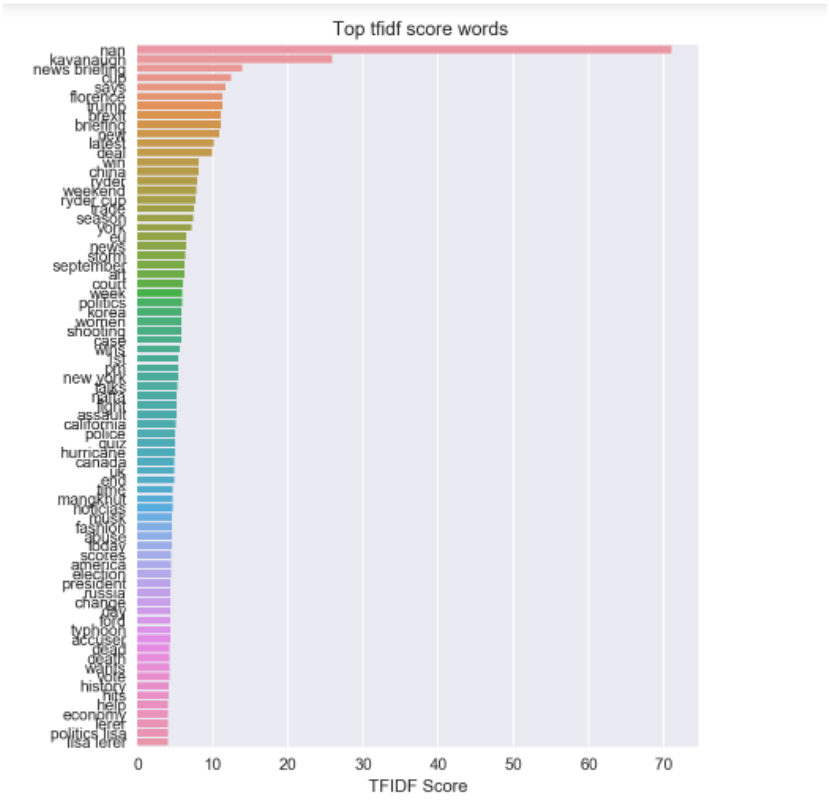
1. Count Vector
2. TF-IDF Vector
3. Co-Occurrence Vector

We have then applied TF-IDF vector for finding weight of each word .TF-IDF was first applied to each individual dataset to get an idea of most weighted words and then to final dataset. It also tells us how important each word is to the document . We have used **n_grams** window range =(1,2) to get additional features along with each word. Tfidf tokenizes our string dataset into corresponding vector by giving each word a score and now it can be trained using any algorithm.

We have split the dataset into training and testing data by 0.75 and 0.25 respectively using train_test_split from sklearn.cross_validation. Then applied Random Forest Classifier for training the dataset . Random Forest is a better algorithm than decision tree. It trains the dataset on various features that were formed by tfidf vectorizer. We then predict values of testing data and their accuracy is tested by using F1 score and accuracy matrix. We get an accuracy of 93.9% on body of our news.



Kaggle TF-IDF SCORE



NYT TF-IDF SCORE

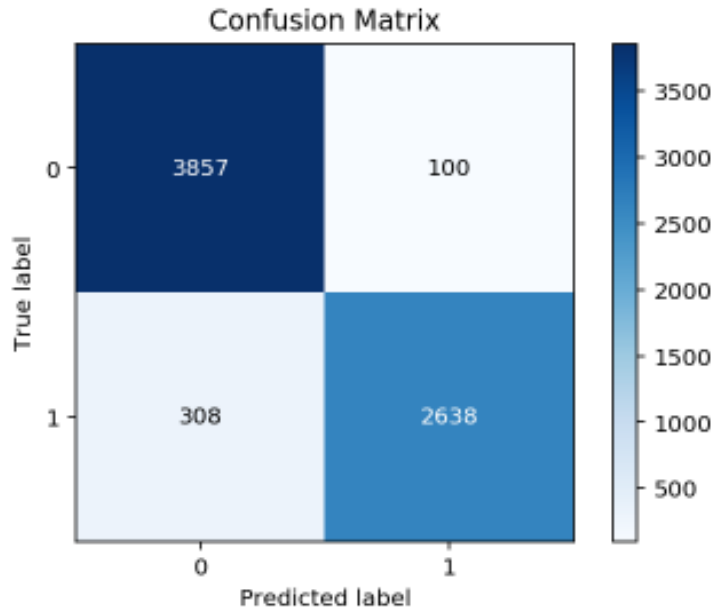
RESULT

Accuracy score and F1 score of body of news:-

```
Random Forest F1 and Accuracy Scores :

F1 score 93.79%
Accuracy score 93.99%
```

Confusion matrix of body



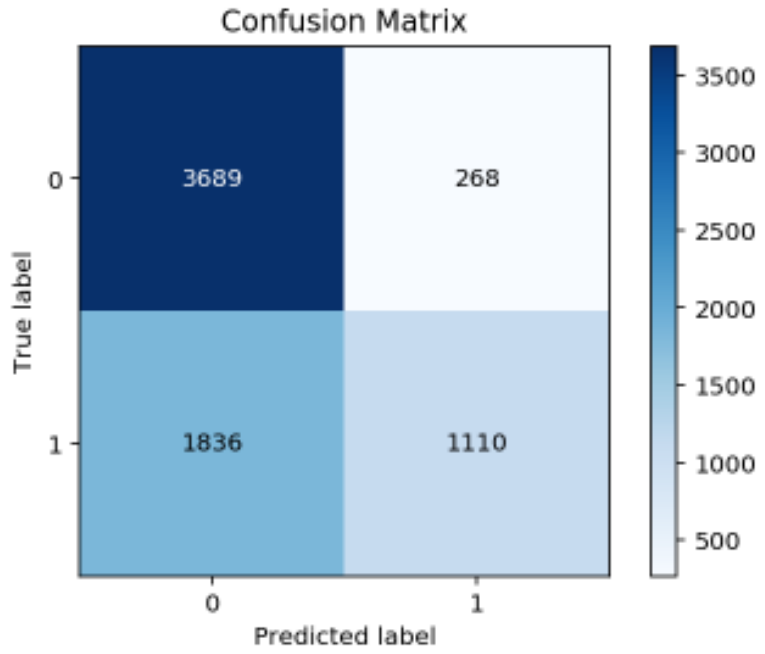
confusion matrix of random forest

Accuracy score and F1 score of headline of news:-

```
Random Forest F1 and Accuracy Scores :

F1 score 64.67%
Accuracy score 69.64%
```

Confusion matrix of headline



confusion matrix of random forest of headlines

Prediction based Embedding:-

These methods were prediction based in the sense that they provided probabilities to the words and proved to be state of the art for tasks like word analogies and word similarities. They were also able to achieve tasks like King -man +woman = Queen, which was considered a result almost incredible.

Our approach for learning paragraph vectors is inspired by the methods for learning the word vectors. The inspiration is that the word vectors are asked to contribute to a prediction task about the next word in the sentence. So despite the fact that the word vectors are initialized randomly, they can eventually capture semantics as an indirect result of the prediction task. We will use this idea in our paragraph vectors in a similar manner. The paragraph vectors are also asked to contribute to the prediction task of the next word given many contexts sampled from the paragraph.

Word2vec is a prediction based algorithm and it's not a single algorithm but a combination of two techniques – CBOW(Continuous bag of words) and Skip-gram model which converts every word into vector. Both of these are shallow neural networks which map word(s) to the target variable which is also a word(s). Both of these techniques learn weights which act as word vector representation.

Doc2vec modifies the word2vec algorithm to unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire document. in the doc2vec architecture, the corresponding algorithms are "distributed memory" Dm) and "distributed bag of words**"** (dbow).The input to Doc2Vec is an iterator is an iterator of labelled Sentence objects. Each such object represents a single sentence, and consists of two simple lists: a list of words and a list of labels. The algorithm then runs through the sentences iterator twice: once to build the vocab, and once to train the model on the input data, learning a vector representation for each word and for each label in the dataset.

Dm defines the training algorithm. If Dm=1 means 'distributed memory' (PV-DM) and Dm =0 means **'**distributed bag of words' (PV-DBOW). Distributed Memory model preserves the word order in a document whereas Distributed Bag of words just uses the bag of words approach, which doesn't preserve any word order.

It was observed that the effect of epochs were as follows :

EPOCHS	ACCURACY(DM)
10	**89.79 %**
30	**92.49%**
50	**94.98 %**

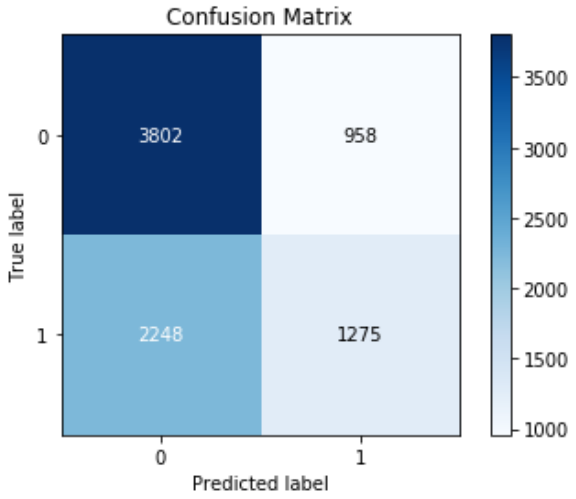
RESULT

Accuracy score and F1 score of DBOW

```
y_train_dbow, X_train_dbow = vec_for_learning(model_dbow, train_tagged)
y_test_dbow, X_test_dbow = vec_for_learning(model_dbow, test_tagged)
logreg = LogisticRegression(n_jobs=1, C=1e5)
logreg.fit(X_train_dbow, y_train_dbow)
y_pred_dbow = logreg.predict(X_test_dbow)
from sklearn.metrics import accuracy_score, f1_score
print('Testing accuracy %s' % accuracy_score(y_test_dbow, y_pred_dbow))
print('Testing F1 score: {}'.format(f1_score(y_test_dbow, y_pred_dbow, average='weighted')))
```

```
Testing accuracy 0.612942170711095
Testing F1 score: 0.5926642075536247
```

Confusion matrix of Distributed Bag of words



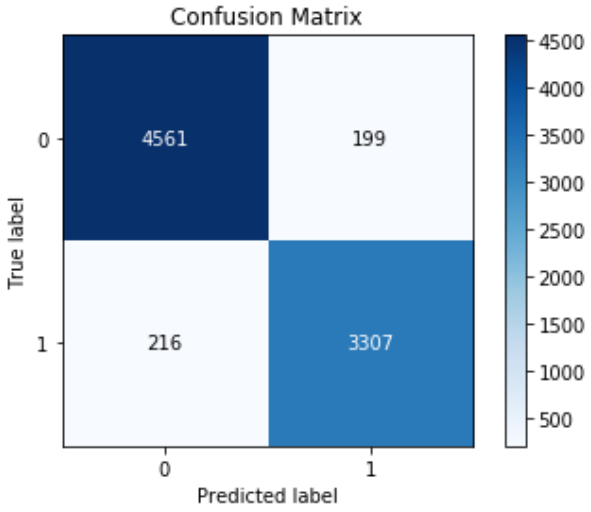
confusion matrix of logistic regression of Doc2vec using distributed bag of words

Accuracy score and F1 score of DM

```
y_train_dm, X_train_dm = vec_for_learning(model_dmm, train_tagged)
y_test_dm, X_test_dm = vec_for_learning(model_dmm, test_tagged)
logreg.fit(X_train_dm, y_train_dm)
y_pred_dm = logreg.predict(X_test_dm)
print('Testing accuracy %s' % accuracy_score(y_test_dm, y_pred_dm))
print('Testing F1 score: {}'.format(f1_score(y_test_dm, y_pred_dm, average='weighted')))
```

```
Testing accuracy 0.9498973801762647
Testing F1 score: 0.9498814470291072
```

Confusion matrix of DM



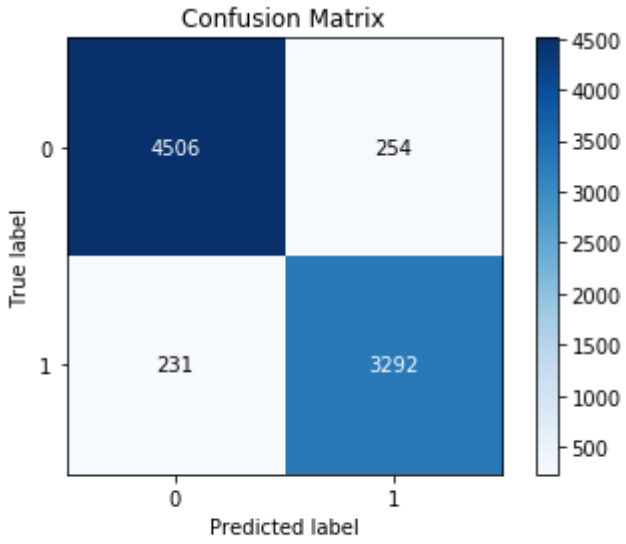
confusion matrix of logistic regression of Doc2vec using distributed memory

Accuracy score and F1 score of DM

```
y_train, X_train = get_vectors(new_model, train_tagged)
y_test, X_test = get_vectors(new_model, test_tagged)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print('Testing accuracy %s' % accuracy_score(y_test, y_pred))
print('Testing F1 score: {}'.format(f1_score(y_test, y_pred, average='weighted')))
```

```
Testing accuracy 0.9414463358686467
Testing F1 score: 0.9414706890767734
```

Confusion matrix of DM



confusion matrix of logistic regression of Doc2vec combining both models

Support Vector Machine:-

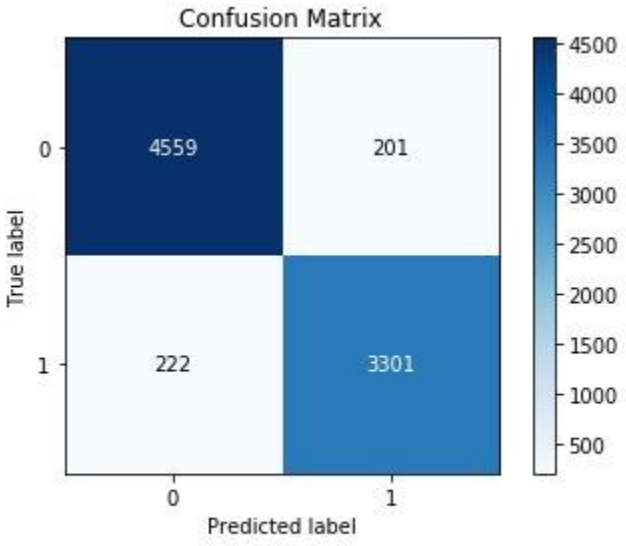
Support Vector Machine is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes.

Accuracy and F1 score of SVM on DM model:-

```
C = 1.0
from sklearn import svm
svc = svm.SVC(kernel='linear', C=C).fit(X_train_dm, y_train_dm)
y_pred= svc.predict(X_test_dm)
print('Testing accuracy %s' % accuracy_score(y_test_dm, y_pred))
print('Testing F1 score: {}'.format(f1_score(y_test_dm, y_pred, average='weighted')))
```

```
Testing accuracy 0.9489315465411083
Testing F1 score: 0.9489114179945066
```

Confusion matrix of SVM:-



confusion matrix of SVM of Doc2vec using distributed memory

ANALYSIS

MODEL	ACCURACY
RANDOM FOREST	94.09 %
LOGISTIC RESGRESSION(DBOW)	61.29 %
LOGISTIC RESGRESSION(DM)	94.98 %
LOGISTIC REGRESSION(DBOW & DM)	94.14 %
SVM	94.89 %

REFRENCES

- https://www.kaggle.com/mrisdal/fake-news/data
- https://open-platform.theguardian.com/
- https://developer.nytimes.com/
- www.codingninjas.com
- www.youtube.com/semicolon

- *RESEARCH PAPER 1-Automatic detection of fake news- Ver´onica P´erez-Rosas1, Bennett Kleinberg2, Alexandra Lefevre1 Rada Mihalcea1*

- *RESEARCH PAPER 2-* GloVe: Global Vectors for Word Representation Jeffrey Pennington, Richard Socher, Christopher D. Manning Computer Science Department, Stanford University, Stanford, CA 94305. This paper was about GloVe . It is a type of word embedding which is very helpful in text classification problems. It was recently developed by Stanford university.

- *RESEARCH PAPER 3-Seperating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter* Svitlana Volkova, Kyle Shaffer, Jin Yea Jang and Nathan Hodas
  Data Sciences and Analytics Group, National Security Directorate
  Pacific Northwest National Laboratory 902 Battelle Blvd, Richland, WA