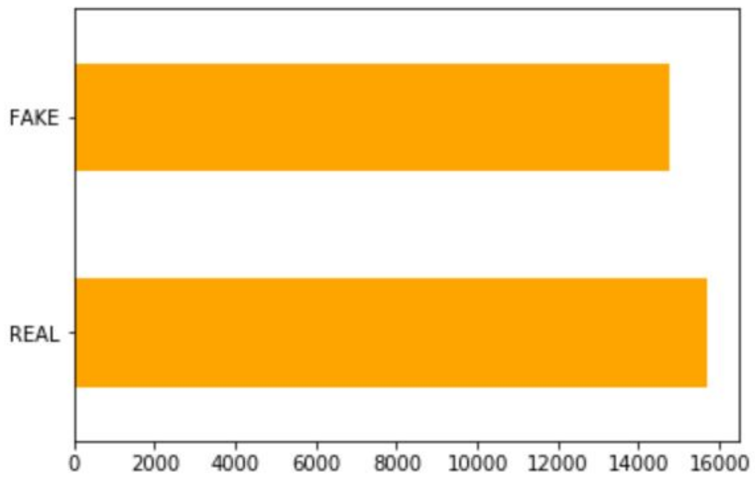
Classification of Fake and Real News Articles

1. **Use Case -**
   In today's age of quick information, it is important to take a step back and verify the information and news we are processing. Authenticity of news articles is essential as the spread of rumours may lead to disastrous results and unnecessary chaos.
   The main objective of this project is to create a model which can take in a news article's headline and body and classify it to be Fake or Real news.

2. **Data Exploration and Visualization -**
   To build this supervised ML model, we have taken the datasets from Kaggle.
   Four datasets have been used to gain enough data points so as to build an efficient model.
   We will be using 2 main fields from the datasets - Headline and Body, and 1 target field - Label ('FAKE', 'REAL').
   We have 31,163 data points in total.



3. **Data Quality Assessment -**
   The 31,163 data points collected from 4 datasets were assessed. Various NULL values and duplicate data points were identified. Some extra columns were identified. Once these have been cleared or dealt with in the dataset, we will have good quality data to work on our application.

4. **Feature Engineering -**
   Data integration was performed as part of this step where four datasets of similar type were appended to get the required number of data points. Relevant columns from the dataset were selected before appending them. The final dataset was identified to have duplicate values and NULL points. These were removed and the final data count is as follows-
   REAL = 15677
   FAKE = 14753
   Total = 30430

5. **Platform and Framework -**
   We are building the application in a Python 3.6 Jupyter Notebook on IBM Watson.
   This is because the free cloud tier of IBM Watson gives us enough capability to build applications using the open-source Jupyter Notebooks. Python Libraries can be used easily to build and run ML and Deep Learning Models. We have used libraries like Pandas, nltk, sklearn, keras matplotlib etc. TF-IDF algorithm has been used to convert text to vectors as part of NLP. Tensorflow backend has been used to run the application.

6. **Algorithms -**
   After the data cleaning and getting feature vectors, a total of 3 algorithms were implemented to get various accuracies and model performances-

   1) Naive Bayes:
   It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. It has been popularly known to work for text classifiers with feature vectors as word frequencies. Since the use case is based on a text classifier, this algorithm was chosen.
   An accuracy of 92% was obtained on the testing data which comprised 33% of the total dataset.

   2) Gradient boosting Classifier:
   Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Gradient boosting is an example of boosting ensemble model. Boosting is an ensemble technique in which the predictors are not made independently, but sequentially. Accuracy achieved is 85.1%

3) Neural Network (Deep Learning):
The architecture of the Deep Neural Network developed consists of 1 Embedding layer, dropout layers for regularization, dense layers, 1 Conv layer, 1 max pooling layer to average out max vectors and LSTM layers. ReLu activation has been used and Sigmoid after the last dense layer for classification. Adam optimizer has been used. The dataset has been split into training and testing data in a ratio of 7:3. We used Keras in TensorFlow and ran the model for 7 epochs.
The accuracy gradually changed and max was 80.97%.

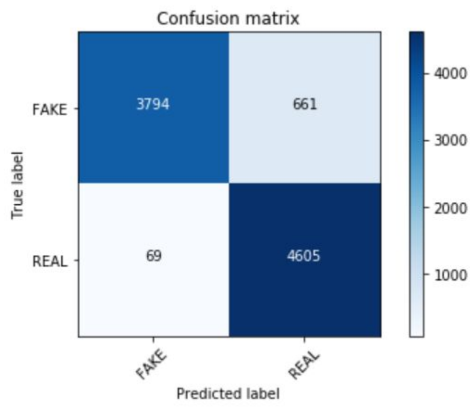
4) TFIDF Vectorizer (NLP):
TF-IDF is a method used to represent text in a format which can be easily processed by the machine learning algorithms. The importance of a word is proportional to the number of times the word appears in the document but inversely proportional to the number of times the word appears in the corpus. The weights are composed of 2 terms:
a) TF - Term Frequency - Calculate frequency of words
b) IDF - Inverse Document Frequency - Denotes importance of word in the document.

7. Model Performance Indicator -

We have assessed the performance of our models using various indicators. We have used Confusion Matrix to see the True Positive, True Negative, False Positive and False Negative values after the model has run.
We used the Accuracy indicator to see how well the model does on Training and Testing data. It gives us a good idea as to how well the model is working on unseen data in terms of percentage.
F1 Score and Precision has also been calculated for the performance parameter of the models.

Confusion Matrix for Naive Bayes Algorithm.

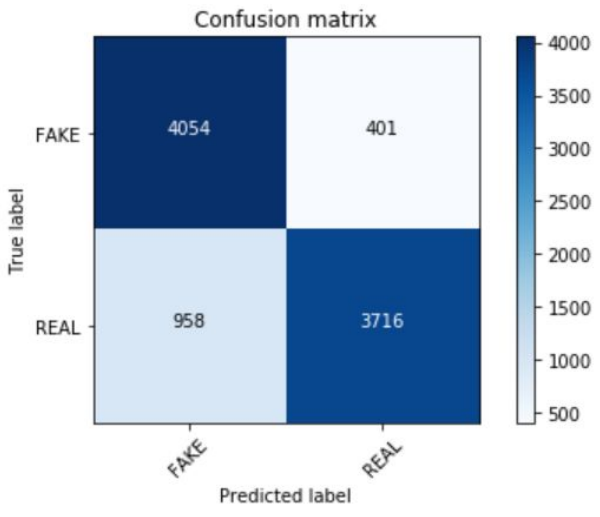
```
Confusion matrix
```



F1 score, Precision and Recall for Naive Bayes' Algorithm

```
F1 score with MultinomialNB Classifier  0.919
Precision with MultinomialNB Classifier   0.928
Recall with MultinomialNB Classifier   0.918
```

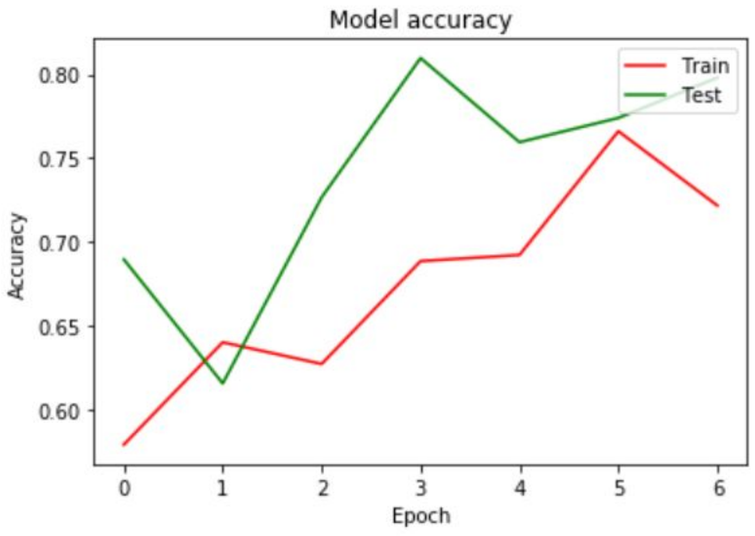
Confusion Matrix for Gradient Boosting Classifier.

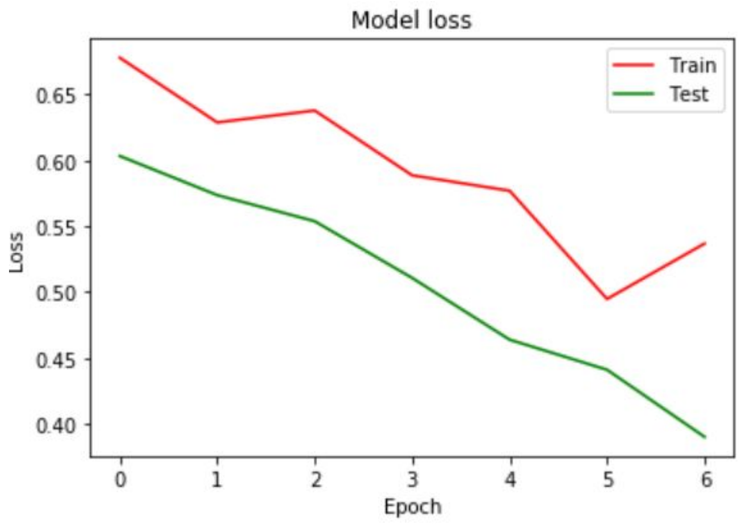


F1 score, Precision and Recall for Gradient Boosting Classifier.

```
F1 score with MultinomialNB Classifier  0.851
Precision with MultinomialNB Classifier   0.856
Recall with MultinomialNB Classifier   0.853
```

Plot of Accuracy and Loss of Deep Learning Model over epochs -


Model accuracy
Train
Test
Accuracy
Epoch


Model loss
Train
Test
Loss
Epoch