## Purpose and Expectations

The Purpose and Expectation: The Purpose of this project is to successfully determine whether a given news is fake or not. In response to Facebook coming under fire for inefficiently deploying its AI Bots to filter out fake news, which, many felt was a tool to sensor out any beliefs contrary to their own, it has become a challenge to filter out the news articles by classifying them into one of two categories: Fake or Real. We also are aiming to further quantify the “Real-ness” or “Fake-ness” of a given article and visualize the concluded data to easily represent our model.

Just like all Data Science Projects, this too, can be coded in a more efficient manner. But as a beginner project and the fact that we’ll be using NLP for the first time, we’ll be expecting a minimum of 85% success rate amongst the Test Sets. If possible, even reach to the 90% mark.

Also, we’ll be checking the accuracy of the model not just by a binary matrix, but also by using Linear Regression on the “Realness/Fakeness of an Article” Data Set to determine by exactly how much is our system is accurate/inaccurate by.

## Method of Approach:

To do so, we’ll be using two approaches, in both approaches, we’ll employ a Naive Bayes classifier in order to create a model to classify an article as fake or real based on its words and phrases.

<space for figure>

The figure describes how we’ll be using two different algorithms, namely :  
 1. Bag-of-Words Algorithm using CountVectorizer to extract features from the tokenized words from the articles.

2. Use of Linear or SVM (Support Vector Machine) Model using TF-IDF Vectorizer to extract features from the tokenized words from the articles.

We use the MultinomialNB of the Naive Bayes model to compare the above two approaches. This classifier is suitable for classification problems having discrete features, like word counts.

**The Data Set**

The dataset we are going to use was released by Kaggle, comprising of articles published during the 2016 US Presidential elections. It contains equal number of fake and real articles, making the null accuracy of the dataset 50%.

The final dataset contains nearly 15000 articles, with their headlines and bodies separated out, and labeled as well (fake or real).

**Libraries**

1.Numpy

2.Scikit-learn

3. NLTK

4. Pandas

**Building the Classifiers**

We remove the **stopwords** from the text before processing it further. Stopwords are commonly used words like ‘is’, ‘an’, ‘the’, ‘of’ which hold little to no significance, and will not contribute to any further operations. We use nltk.corpus for this, and set the language to “english”.

Countvectorizer and Tf-idf Vectorizer are both used for feature extraction.

**Countvectorizer:**

It takes each text, and creates a column for each word that exists on the

corpus and sets the number of times that that word repeats on the column for a given text.

**Tfidfvectorizer**

Tf-idf stands for *"*Term Frequency- Inverse Document Frequency”.

\text{tf-idf(t,d)}=\text{tf(t,d)} \times \text{idf(t)}

\text{idf}(t) = log{\frac{1 + n_d}{1+\text{df}(d,t)}} + 1

It highlights words that are frequent in a document, but not across documents.

**Comparing Models**

One way is to use confusion matrices. The correct labels are shown on the diagonals, while other cells show the incorrect predictions (false positives or false negatives).

Another way is to use an ROC curve.