**Problem Requirements: -**

Perform the following tasks on the Kaggle Fake News data-

1. Perform Exploratory data analysis on the data
2. Perform necessary text pre-processing steps – tokenization, cleaning, stemming, lemmatization, etc. (make sure to explain each step and observations, if any)
3. Train a text classification model on 80% of data–
4. Test the model on 20% of data-
5. Perform unsupervised topic modelling on the fake news data (text + title)

**Hypothesis Metrics: -**

1. Article Text
2. Polarity of the News
3. Number of Sentence in a news
4. Topic/Subject
5. EmoRatio: -Number of negative words / Number of positive words

**Planning: -**

1. **Pre-processing steps: -**

Before representing the data using n-gram and vector-based model, the data need to be subjected to certain refinements like stop-word removal, tokenization, a lower casing, sentence segmentation, and punctuation removal. This will help us reduce the size of actual data by removing the irrelevant information that exists in the data. We created a generic processing function to remove punctuation and non-letter characters for each document; then we lowered the letter case in the document. In addition, an n-gram word-based tokenizer was created to slice the text based on the length of n.

**a. Tokenization: -** Word Tokenization

**b. Cleaning: -** **Lowercase, punctuation removal, stop words removal, special characters removal**

Stop words are insignificant words in a language that will create noise when used as features in text classification. These are words commonly used a lot in sentences to help connect thought or to assist in the sentence structure. Articles, prepositions and conjunctions and some pronouns are considered stop words. We removed common words such as, a, about, an, are, as, at, be, by, for, from, how, in, is, of, on, or, that, the, these, this, too, was, what, when, where, who, will, etc. Those words were removed from each document, and the processed documents were stored and passed on to the next step.

**c. Lemmatization/Stemming: -**

After tokenizing the data, the next step is to transform the tokens into a standard form. Stemming simply is changing the words into their original form and decreasing the number of word types or classes in the data. For example, the words “Running”, “Ran” and “Runner” will be reduced to the word “run.” We use stemming to make classification faster and efficient. Furthermore, we use Porter stemmer, which is the most used stemming algorithms due to its accuracy.

**d. Data Merging**

We will merge the fake and true news into one to prepare the Ad for the text prediction

**e. TDF -IDF**

**Term Frequency** is an approach that utilizes the counts of words appearing in the documents to figure out the similarity between documents. Each document is represented by an equal length vector that contains the words counts. Next, each vector is normalized in a way that the sum of its elements will add to one. Each word count is then converted into the probability of such word existing in the documents. For example, if a word is in a certain document it will be represented as one, and if it is not in the document, it will be set to zero. Thus, each document is represented by groups of words. The **Term Frequency-Inverted Document Frequency (TF-IDF)** is a weighting metric often used in information retrieval and natural language processing. It is statistical metric used to measure how important a term is to a document in a dataset. A term importance increases with the number of times a word appears in the document, however, this is counteracted by the frequency of the word in the corpus. One of the main characteristics of IDF is it weights down the term frequency while scaling up the rare ones. For example, words such as “the” and “then” often appear in the text, and if we only use TF, terms such as these will dominate the frequency count. However, using IDF scales down the impact of these terms

**2. EDA: -**

**Performed the below EDA for both Fake and Real News**

**a. Distribution of word length: -** To find out news having greater number of words.

**b. Word frequency for unigram, bigram: -** To find out what are the common topics for both Fake and not fake news and the different topics as well

**c. Word Cloud: -** To find out words having more importance in fake and non-Fake news

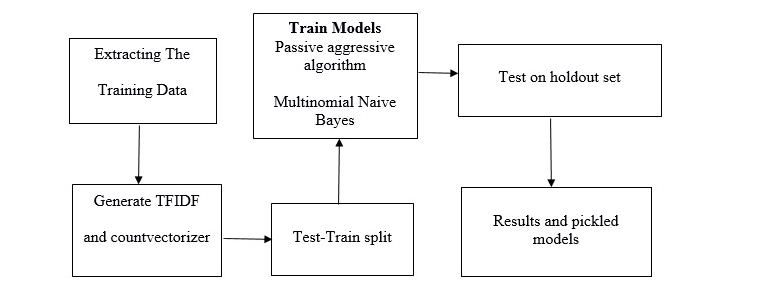
**d. TDF-IDF Representation: -**

Aim to find out the high TDF-IDF weightage bigrams for both fake and true news.

**e. Sentiment Analysis: -** Aims to understand and analyse the characteristics of fake news especially in relation to sentiments, for the automatic detection of fake news and rumours.

1. **Classification Process: -**

**Figure 1**



Classification Process **Figure 1** is a diagrammatic representation of the classification process. It will start with pre-processing the data set, by removing unnecessary characters and words from the data. N-gram features are extracted, and a features matrix is formed representing the documents involved. The last step in the classification process is to train the classifier. We will investigate different classifiers to predict the class of the documents. We will investigate specifically different machine learning algorithms, SVM, Passive aggressive algorithm and Multinomial Naïve Bayes. We will implement these classifiers from the Python Natural Language Toolkit (NLTK). We will split the dataset into training and testing sets. For instance, in the experiments presented subsequently, we use 5-fold cross validation, so in each validation around 80% of the dataset is used for training and 20% for testing.

**2. Feature Engineering**

The next step is the feature engineering step. In this step, raw text data will be transformed into feature vectors and new features will be created using the existing dataset. We will implement the following different ideas in order to obtain relevant features from our dataset.

**2.1 Count Vectors as features**

**2.2 TF-IDF Vectors as features**

* Word level
* N-Gram level
* Character level

**2.3 Text / NLP based features**

**2.1 Count Vectors as features**

Count Vector is a matrix notation of the dataset in which every row represents a document from the corpus, every column represents a term from the corpus, and every cell represents the frequency count of a particular term in a particular document.

**2.2 TF-IDF Vectors as features**

TF-IDF score represents the relative importance of a term in the document and the entire corpus. TF-IDF score is composed by two terms: the first computes the normalized Term Frequency (TF), the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

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

IDF(t) = log\_e(Total number of documents / Number of documents with term t in it)

TF-IDF Vectors can be generated at different levels of input tokens (words, characters, n-grams)

a. Word Level TF-IDF : Matrix representing tf-idf scores of every term in different documents

b. N-gram Level TF-IDF : N-grams are the combination of N terms together. This Matrix representing tf-idf scores of N-grams

c. Character Level TF-IDF : Matrix representing tf-idf scores of character level n-grams in the corpus

**2.3 Text / NLP based features**

A number of extra text based features can also be created which sometimes are helpful for improving text classification models. Some examples are:

Word Count of the documents – total number of words in the documents

Character Count of the documents – total number of characters in the documents

**4. Perform unsupervised topic modelling on the fake news data (text title): -** uncovering hidden structure in a collection of texts.

**Steps: -**

1. Load Cleaned data
2. Preparing data for LDA analysis
3. LDA model training
4. Analysing LDA model results