**Problem Statement**

​Develop a model to detect if a given news article is a fake one or legitimate(real) one.

**Dataset location**

<https://www.kaggle.com/c/fake-news/data>

**About dataset**

The dataset consists of 3 CSV files: "train.csv", "test.csv" and "submit.csv".  
"train.csv" contains following columns

* "**id**" of article
* "**Title**" of the article
* "**Text**" of the article
* "**Author**" of the article
* "**Label**" of article
  + **1**: fake
  + **0**: real

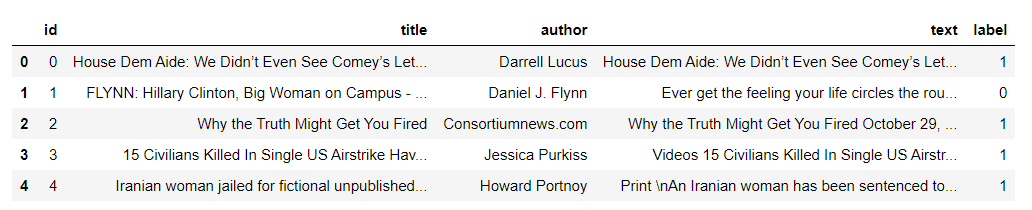
"test.csv" contains the same attributes as "train.csv", but without the "label" attribute. Prediction is to be performed on these articles.

"submit.csv" contains sample submission format. For every article in "test.csv" file, submission CSV file should have two columns: `id` and `label`. The `id` column should refer to a row in the "test.csv" file, and the `label` column

**Solution Plan:**

From the problem statement we can clearly understand this is a Binary classification problem.

1. Data Analysis:



1. Selecting features to work with:

If we look at the data closely we have 4 feature columns and ‘Label’ is the dependent variable column.

We can exclude the column ‘id’ as it has just serial number for each article and does not provide a lot of meaning.

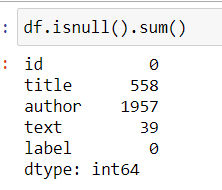
“title”, “author”, “text” -> These columns are textual columns and all 3 columns can be used for building a model, but first we need to transform the text-based data into numeric format.

We can either choose to work with all 3 columns by merging the text from all 3 columns into 1 column and then applying transformation on the data.

For sake of simplicity I am just using the text column.

1. Check for Null Values:

Upon checking for null values, we can see dataset has few null values:



We cannot use any imputation technique as this is all text data and filling in missing title or author name would be a new problem statement in its own

If we calculate, we have less than 2% of the data missing and computing Author name and Title would not be a good choice, we can simply drop these rows.

1. Check for Outliers:

Outliers in this case could be considered as an article which is very small in length. i.e. 10-20 words.

We can get the check for maximum and minimum words present across all articles and find if minimum is less than 10-20 words, then we need to find these articles and decide the next action.

For the sake of simplicity I have skipped this step, however this is one of the important step of pre-processing data.

1. Data Pre-processing:

As we have textual data we need to convert this data into numerical form before building a model on it.

Below are the common operations that needs to be performed:

1. Apply Word Tokenize to break the sentence into list of words.

2. Remove Punctuation's (special characters) and Apostrophe

3. Remove stopwords(i.e. words like a, the, is, etc.) from the 'text' column for each row

4. Changing all words to lower case (so that algorithm does not consider the same word with different casing as different words.)

5. Apply Stemming (Alternatively we can apply Lemmatisation)

Additionally, we can also consider converting Numbers to words.

Upon applying the following we will now have a group of words for each article.

1. Extracting Feature

Next step is to convert the bag of words to numeric form.

Following are the options to encode the text data to numeric data

1. **One-Hot encoding** - Not good for the purpose, as the number of occurrences and context of occurrence will be lost in this case. (information loss)

2. **Count Vector Encoding**- We will have the advantage of having a number of occurrences

3. **TF-IDF** - We will have the count for the occurrence of the word in each article vs the total occurrence of the word.

4. **Co-occurrence** - Building a co-occurrence matrix to see the words occurring together.

5. **Word embedding**- This is great when we want our machine to understand the context

We can use Count Vector in this case as we just need to check the occurrence of the words in the document, we not need the model to be complex.

Now upon applying count vector encoding, we have numerical representation of the given textual data.

1. Split the data

Though we have a separate set of training data, but for building and evaluating the model we do need to split the data.

1. Evaluation Matrix

For the given problem statement, we can consider both precision and re-call as evaluating metrics. As we want the model to predict the fake article more accurately.

1. Model Selection

As we know this is a Binary classification problem, we can select any of the following algorithm to build our model.

* Logistic Regression
* k-Nearest Neighbors
* Decision Trees
* Support Vector Machine
* Naive Bayes

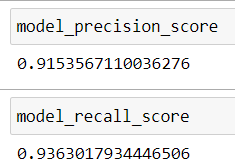
We can use GridSerachCV to first find the best hyper-parameter for which we have good precision and recall score.

Once we have the hyper-parameter, we can build the model on the training set and then use the test set we had created to evaluate the model.

Another way is to take advantage of VotingClassifier which can internally train data on all the models and then based on a voting system take the final decision on the prediction/output.

I have used Voting Classifier with “LogisticRegression” , “Support Vector Classifier”, “KNeighborsClassifier” to build the model.

1. Model Score



1. Predicating Category for Test Data

The given Test Data is in same format as the train data, so we need to apply all the pre-processing on the data as we did to the train set.

After pre-processing and transforming the data, we can call the model predict method to get the prediction and save the result.