FLIP ROBO

FAKE NEWS DECTECTION PROJECT

Submitted by:

ASHOK KUMAR SHARMA

ACKNOWLEDGMENT

I would like to express my deep sense of gratitude to my SME (Subject Matter Expert) **Ms. Rashi Mathur** as well as **Flip Robo Technologies** who gave me the golden opportunity to do project on **Fake News Detection**, which also helped me in doing lots of research and I came to know about so many new things.

I am very much thankful to **Mr. Shankargouda Tegginmani, Trainer (DataTrained)**, for their valuable guidance, keen interest and encouragement at various stages of my training period which eventually helped me a lot in doing this project.

I also acknowledge with thanks for suggestion and timely guidance, which I have received from my SME Ms. Rashi Mathur during this project, which immensely helped me in the evaluation of my ideas on the project.

ASHOK KUMAR SHARMA

INTRODUCTION

- Business Problem Framing

  The authenticity of Information has become a longstanding issue affecting businesses and society, both for printed and digital media. On social networks, the reach and effects of information spread occur at such a fast pace and so amplified that distorted, inaccurate, or false information acquires a tremendous potential to cause real-world impacts, within minutes, for millions of users. Recently, several public concerns about this problem and some approaches to mitigate the problem were expressed. This project is intended to build a model which can predict whether the news is fake or not.

- ## Conceptual Background of the Domain Problem

  Fake news is false or misleading information presented as news. It often has the aim of damaging the reputation of a person or entity, or making money through advertising revenue. However, the term does not have a fixed definition, and has been applied more broadly to include any type of false information, including unintentional and unconscious mechanisms, and also by high-profile individuals to apply to any news unfavourable to his/her personal perspectives.

- ## Review of Literature

  1. What is Fake News?

     Fake news is false or misleading information presented as news. It often has the aim of damaging the reputation of a person or entity, or making money through advertising revenue.

- ## Motivation for the Problem Undertaken

  This model can be used by the print media, news media, social media and search engine websites to detect a news article as Fake or Not Fake. This model can classify fake news so that it can be controlled and restricted from spreading around the world.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

For checking datatypes and null values, pandas.DataFrame.info() and pandas.Series.isnull().sum() method has been used. To drop the null values pandas.DataFrame.dropna() method has been used. To replace and remove the certain terms and punctuations, pandas.Series.str.replace() method with regular expression has been used. To get rid of stop words, nltk.corpus.stopwords() method has been used.

- ## Data Sources and their formats

The dataset is in the form of .CSV (Comma Seperated Value) format and consists of 6 columns (5 features and 1 label) with 20800 number of records as explained below:

- `id`: Unique id for each news article.
- `headline`: It is the title of the news.
- `news`: It contains the full text of the news article.
- `Unnamed0`: It is a serial number.
- `written_by`: It represent the author of the news article.
- `label`: It tells whether the news is fake(1) or not fake(0).

- ## Data Pre-processing Done

The following pre-processing pipeline is required to perform model prediction:

1. **Load dataset**
2. **Drop column** Unnamed: 0 and id
3. **Treating Null Values:** Replace null values with ' ' (single space) and merging feature headline, written_by and news to new column **text_feature** and then dropp columns headline, written_by and news
4. **Convert text_feature to lower case** and replace '\n' with single space.
5. **Keep only text data i.e., a-z' and 0-9** and remove other data from text_feature.
6. **Remove stop words and punctuations**
7. **Covert text_feature to vectors** using TfidfVectorizer
8. **Load the serialized model**
9. **Predict values** by passing the vectors of text_feature.

- Data Inputs- Logic- Output Relationships

Input	Logic (algorithm)	Output
`text_feature`	MultinomialNB SGDClassifier HistGradientBoostingClassifier	0 (Not Fake) OR 1 (Fake)

There is 1 input variable needs to be provided to the logic to get the output i.e. 0-NotFake or 1-Fake. Logic highlighted in green i.e. HistGradientBoostingClassifier is the best performing algorithm among all other logics on this dataset.

- Hardware and Software Requirements and Tools Used

During this project, following set of hardware is being used:

RAM: 8 GB

PAGE_FILE: 90GB on SSD

CPU: AMD A8 Quad Core 2.2 Ghz

GPU: AMD Redon R5 Graphics

and the following software and tools is being used:

a. Python
b. Jupyter Notebook
c. Anaconda

With following libraries and packages:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- nltk
- wordcloud
- sys
- tqdm.notebook

- timeit
- sklearn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  To solve this problem following steps are used:

  1. Load dataset
  2. Drop column Unnamed: 0 and id
  3. Treating Null Values: Replace null values with ' ' (single space) and merging feature headline, written_by and news to new column text_feature and then dropp columns headline, written_by and news
  4. Convert text_feature to lower case and replace '\n' with single space.
  5. Keep only text data i.e., a-z’ and 0-9 and remove other data from text_feature.
  6. Remove stop words and punctuations
  7. Covert text_feature to vectors using TfidfVectorizer
  8. Separate Input and Output Variables.
  9. Train & Test the Model by supplying Input and Output Variables.

- Testing of Identified Approaches (Algorithms)

  Following are the list of algorithms used for training and testing:

  1. MultinomialNB
  2. SGDClassifier
  3. HistGradientBoostingClassifier

- Run and Evaluate selected models

  A total of 3 algorithm has been used on this dataset for training testing purpose, these are MultinomialNB, SGDClassifier and HistGradientBoostingClassifer. To perform training and testing operation(s) following functions has been defined for which codes are as follows:

```
#importing required libraries
from sklearn.model_selection import train_test_split, GridSearchCV, cross_v
al_score
from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
import tqdm.notebook as tqdm
import sys, timeit
from IPython.display import display

#convert news text to vectors using TfidfVectorizer
tfidf = TfidfVectorizer(max_features=8000)
features = tfidf.fit_transform(df_news.text_feature).toarray()

#Input variable
X = features
print("Feature's Shape: ",X.shape)

#Output variable
Y = df_news.label
print("Target's Shape: ",Y.shape)

#function to get best random state
def get_best_random_state(model,X,Y,t_size=0.25,rs_range=range(1,301,50)):
    best_rstate = 0
    best_accuracy_score = 0
    random_state_message = "\r"

    for i in tqdm.tqdm(rs_range,desc=f"Best_Random_State => {model}"):
        X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=t
_size,random_state=i)
        model.fit(X_train, Y_train)
        y_pred = model.predict(X_test)
        a_score = accuracy_score(Y_test,y_pred)
        if a_score > best_accuracy_score:
            best_accuracy_score = a_score
            best_rstate = i

        random_state_message += f"[{i}: {round(a_score*100,2)}]<--->"
        sys.stdout.write(random_state_message)

    sys.stdout.write(f"\n\nBest Random State: {best_rstate} found with Accuracy: {best_accuracy_score}")
    return best_rstate, best_accuracy_score
#End of function



#function to get best cv score
def get_best_cv(model,X_train,Y_train,parameters,cv_range=range(5,25,5)):
    best_cv_score = 0
    best_cv = 0

    cv_message = "\r"
    for i in tqdm.tqdm(cv_range,desc=f"Best_CV => {model}"):
        gscv = GridSearchCV(model,parameters)
        gscv.fit(X_train,Y_train)

        cv_score = cross_val_score(gscv.best_estimator_,X_train,Y_train,cv=i).mean()

        if cv_score > best_cv_score:
            best_cv_score = cv_score
            best_cv = i

        cv_message += f"[{i}:{round(cv_score*100,2)}]<--->"
        sys.stdout.write(cv_message)

    sys.stdout.write(f"\n\nBest CV: {best_cv} found with Cross Val Score: {best_cv_score}")

    return best_cv, best_cv_score
#End of function

#function to build models
def build_models(models,X,Y,t_size=0.25,rs_range=range(1,301,50),cv_range=range(5,25,5)):
    for i in tqdm.tqdm(models,desc="Building Models"):
        sys.stdout.write("\n=======================================================================================\n")
        sys.stdout.write(f"Current Model in Progress: {i} ")
        sys.stdout.write("\n=======================================================================================\n")

        #start time
        start_time = timeit.default_timer()

        #Find the best random state
        best_random_state, best_accuracy_score = get_best_random_state(models[i]['name'],X,Y,t_size,rs_range)
        sys.stdout.write("\n")

        #Spliting train and test data using train_test_split method with best random state value
        X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=t_size,random_state=best_random_state)

        #Find the best CV
        best_cv, best_cv_score = get_best_cv(models[i]['name'],X_train,Y_train,models[i]['parameters'],cv_range)
        sys.stdout.write("\n\nBuilding Model...")

        #Training the model using best CV
        gscv = GridSearchCV(models[i]['name'],models[i]['parameters'],cv=best_cv)
        gscv.fit(X_train,Y_train)

        #Testing model
        y_pred = gscv.best_estimator_.predict(X_test)

        #Recording model performance
        model_accuracy_score = accuracy_score(Y_test,y_pred)
        model_classification_report = classification_report(Y_test,y_pred)
        model_log_loss = log_loss(Y_test,y_pred)

        #end time
        end_time = timeit.default_timer()
        sys.stdout.write(f"Completed in [{end_time-start_time} sec.]")

        #storing model specifications
        models[i]['initial_accuracy_score'] = best_accuracy_score
        models[i]['best_random_state'] = best_random_state
        models[i]['x_train'] = X_train
        models[i]['x_test'] = X_test
        models[i]['y_train'] = Y_train
        models[i]['y_test'] = Y_test
        models[i]['best_cv'] = best_cv
        models[i]['best_cv_score'] = best_cv_score
        models[i]['gscv'] = gscv
        models[i]['y_predict'] = y_pred
        models[i]['final_accuracy'] = model_accuracy_score
        models[i]['log_loss'] = model_log_loss
        models[i]['classification_report'] = model_classification_report
        models[i]['build_time'] = f"{end_time - start_time} (in sec.)"

        sys.stdout.write("\n===============================================
========================================\n\n\n")

    return models
#End of function

#function to display model performance
def display_performance(models):
    model_names = []
    model_initial_score = []
    model_cross_val_score = []
    model_log_loss_score = []
    model_final_score = []
    model_build_time = []
    for i in models:
        model_names.append(i)
        model_initial_score.append(models[i]['initial_accuracy_score'])
        model_cross_val_score.append(models[i]['best_cv_score'])
        model_log_loss_score.append(models[i]['log_loss'])
        model_final_score.append(models[i]['final_accuracy'])
        model_build_time.append(models[i]['build_time'])

    model_df = pd.DataFrame({
        "Name": model_names,
        "Initial Score": model_initial_score,
        "Cross Val Score": model_cross_val_score,
        "Log Loss": model_log_loss_score,
        "Final Score": model_final_score,
        "Build Time": model_build_time,
    })

    model_df['Difference (Final Score - Cross Val Score)'] = model_df['Fina
l Score'] - model_df['Cross Val Score']
    display(model_df)

    for i in models:
        print("========================================================")
        print(f"for model: {i}")
        print("========================================================")
        print("CLASSIFICATION REPORT")
        print(models[i]['classification_report'])
        print("=======================================================\n\n"
)

    return
#End of function

#List of models for training & testing
models = {
    "MultinomialNB":{
        "name": MultinomialNB(),
        "parameters":{
            "alpha": [1.0]
        }
    },
    "SGDClassifier":{
        "name": SGDClassifier(),
        "parameters":{
            "loss":['hinge','modified_huber'],
            "alpha":[0.001,0.0001,0.00001],
            "n_jobs":[-1],
            "learning_rate":['optimal'],
            "max_iter":[100]
        }
    },
    "HistGradientBoostingClassifier":{
        "name": HistGradientBoostingClassifier(),
        "parameters":{
            "loss": ['binary_crossentropy'],
            "l2_regularization": [0,1.0]
        }
    }
}

#building models
build_model = build_models(models,X,Y,rs_range=[38,40,42,44],cv_range=[5,7,
9])
```

```
Building Models: 100% ██████████████████████████████ 3/3 [7:51:42<00:00, 12737.05s/it]

===========================================================================
============
Current Model in Progress: MultinomialNB
===========================================================================
============
Best_Random_State => MultinomialNB(): 100% ██████████████████████████████ 4/4
[00:15<00:00, 3.56s/it]
[38: 89.94]<--->[40: 90.81]<--->[42: 90.4]<--->[44: 90.15]<--->

Best Random State: 40 found with Accuracy: 0.9080769230769231
Best_CV => MultinomialNB(): 100% ██████████████████████████████ 3/3 [01:10<00:00,
24.28s/it]
[5:90.06]<--->[7:89.99]<--->[9:90.07]<--->

Best CV: 9 found with Cross Val Score: 0.9007059959405734

Building Model...Completed in [105.37263130000065 sec.]
===========================================================================
============




===========================================================================
============
Current Model in Progress: SGDClassifier
===========================================================================
============
Best_Random_State => SGDClassifier(): 100% ██████████████████████████████ 4/4
[00:42<00:00, 10.19s/it]
[38: 96.69]<--->[40: 96.73]<--->[42: 97.08]<--->[44: 96.56]<--->

Best Random State: 42 found with Accuracy: 0.9707692307692307
Best_CV => SGDClassifier(): 100% ██████████████████████████████ 3/3 [12:41<00:00,
256.81s/it]
[5:95.94]<--->[7:96.53]<--->[9:96.54]<--->

Best CV: 9 found with Cross Val Score: 0.9653841321479695

Building Model...Completed in [1180.5848260999992 sec.]
===========================================================================
============




===========================================================================
============
Current Model in Progress: HistGradientBoostingClassifier
===========================================================================
============
Best_Random_State => HistGradientBoostingClassifier(): 100% ████████████████████ 4/4
[23:42<00:00, 355.32s/it]
[38: 97.92]<--->[40: 98.12]<--->[42: 97.85]<--->[44: 97.75]<--->

Best Random State: 40 found with Accuracy: 0.9811538461538462
```

Best_CV => HistGradientBoostingClassifier(): 100% ████████████████████ 3/3 [6:02:34<00:00, 7848.92s/it]

```
[5:97.69]<--->[7:97.67]<--->[9:97.62]<--->

Best CV: 5 found with Cross Val Score: 0.976923076923077

Building Model...Completed in [27015.6240779 sec.]
===========================================================================
============
```

	Name	Initial Score	Cross Val Score	Log Loss	Final Score	Build Time	Difference (Final Score - Cross Val Score)
0	MultinomialNB	0.908077	0.900706	3.174932	0.908077	105.37263130000065 (in sec.)	0.007371
1	SGDClassifier	0.970769	0.965384	1.135799	0.967115	1180.5848260999992 (in sec.)	0.001731
2	HistGradientBoostingClassifier	0.981154	0.976923	0.677499	0.980385	27015.6240779 (in sec.)	0.003462

```
========================================================
for model: MultinomialNB
========================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      2575
           1       0.94      0.87      0.91      2625

    accuracy                           0.91      5200
   macro avg       0.91      0.91      0.91      5200
weighted avg       0.91      0.91      0.91      5200

=======================================================


========================================================
for model: SGDClassifier
========================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      2585
           1       0.99      0.95      0.97      2615

    accuracy                           0.97      5200
   macro avg       0.97      0.97      0.97      5200
weighted avg       0.97      0.97      0.97      5200

=======================================================


========================================================
for model: HistGradientBoostingClassifier
========================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      2575
           1       0.98      0.98      0.98      2625
    accuracy                           0.98      5200
   macro avg       0.98      0.98      0.98      5200
weighted avg       0.98      0.98      0.98      5200

=======================================================
```

From the above model comparison it is clear that **HistGradientBoostingClassifier** performs better with **Accuracy Score: 98.03%** and **Log Loss: 0.67** than other models. Therefore, proceeding with **HistGradientBoostingClassifier**.

• Key Metrics for success in solving problem under consideration

To find out best performing model following metrices are used:

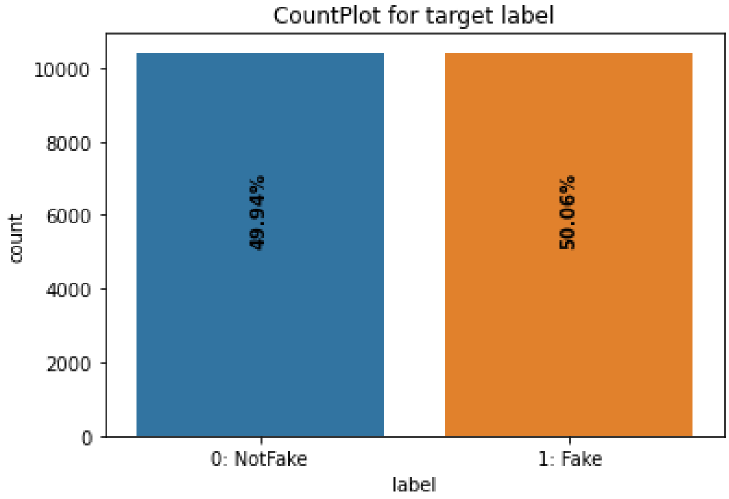
1. Accuracy Score: It is used to check the model performance score between 0.0 to 1.0
2. Log Loss: Log Loss is the negative average of the log of corrected predicted probabilities for each instance.
3. Classification Report: A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False.

• Visualizations

To better understand the data, following types of visualizations have been used: 1. Univariate.

1. Univariate Analysis: Univariate analysis is the simplest form of data analysis where the data being analysed contains only one variable. In this project, distribution plot, count plot and box plot has been used.

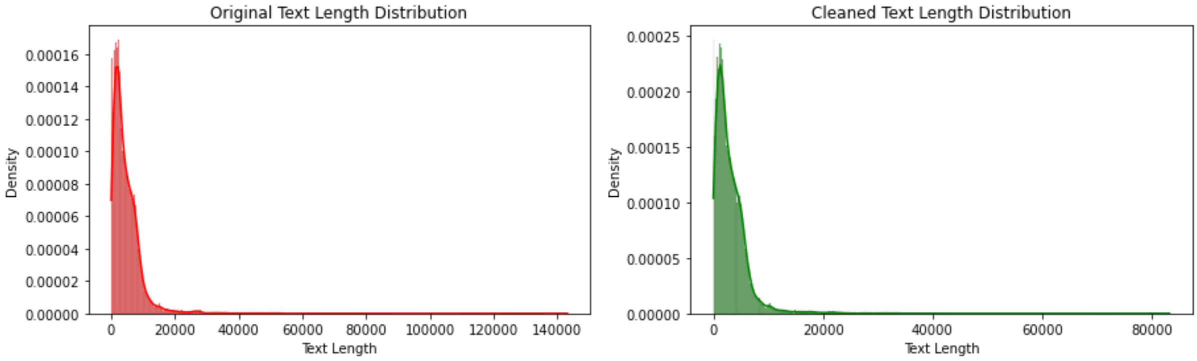
Count Plot (countplot):



Remarks:

- There are almost equal number of records available for fake (1) and Not Fake (0) label.
- **Dataset is balanced.**

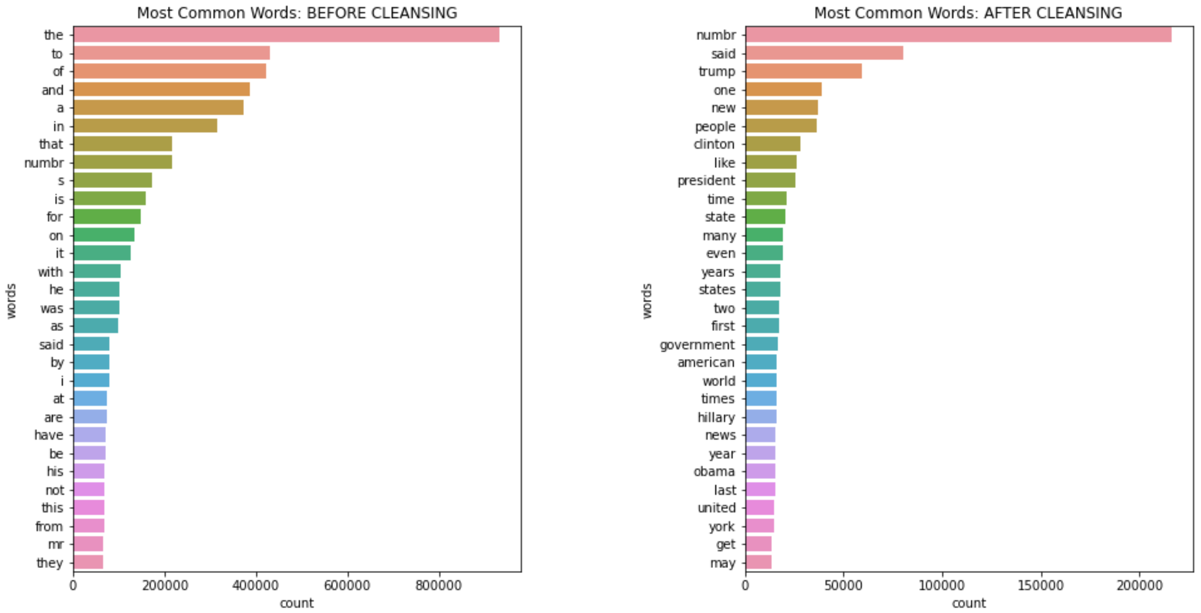
Distribution Plot (distplot):



Remarks:

- From the above dipiction, it is clear that the **length of text reduced by greater amount** after cleansing of text data.
- Also, the range of **text length reduced from 0-140000 to 0-80000**.

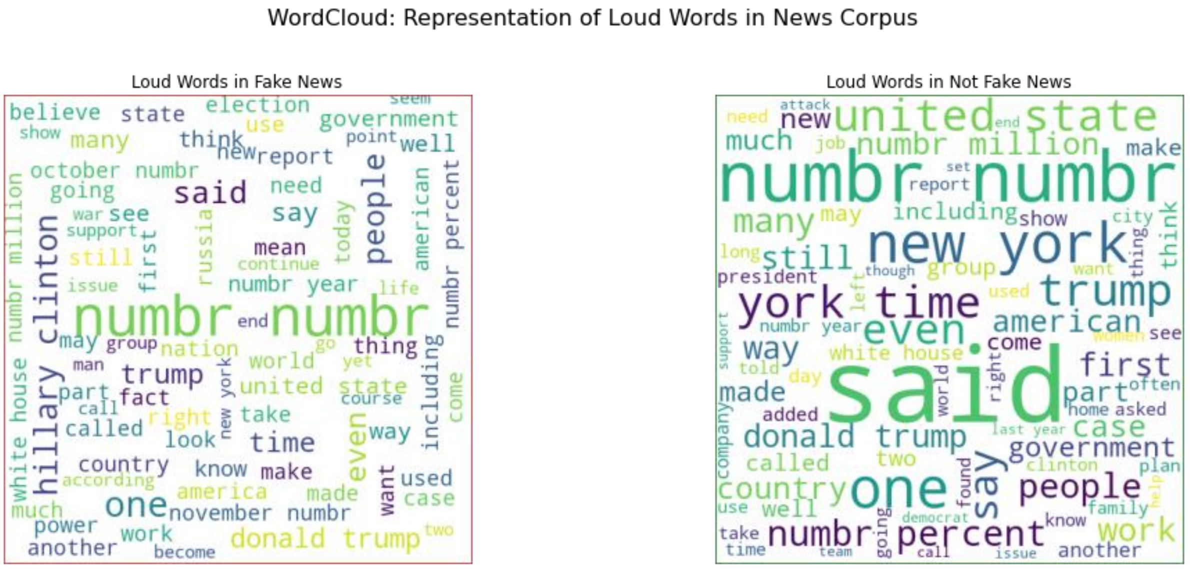
Bar Plot (barplot):



Remarks:

- From above dipiction, it is clear that the most common words before cleansing were the, to, of, and, a, etc. while after cleansing most common words are numbr, trump, one, new, people, etc.

Displaying with WordCloud (wordcloud):



Remarks:

- Loud words in Fake news are numbers, hillary clinton, donald trump, america, united state, people, war, etc.

- Loud words in Not Fake news are new york, trump, united state, support, help, democrat etc.

- Interpretation of the Results

  Starting with count plot, it was found that dataset is balanced with almost equal number of records for fake news as well as not fake news. Moving further with distribution plot for news text length, it was found that after cleansing of news text, the length of news text reduced by a greater amount. Also with the help of barplot for most common words, it was found that, before cleansing, the most common words are mostly stop words i.e. the, and, of, a, etc. and after cleansing, the most common words are like, numbr, said, clinton, people, president, time, state etc. With the help of word cloud, it was found that the loud words in fake news are hillary clinton, donald trump, america, united state, war etc. while in not fake news are new york, trump, united state, support, help, demorat etc.

CONCLUSION

- Key Findings and Conclusions of the Study

  HistGradientBoostingClassifier performs better with **Accuracy Score: 98.03%** and **Log Loss: 0.67** than other models. Therefore, proceeding with **HistGradientBoostingClassifier**.

- Learning Outcomes of the Study in respect of Data Science

  During the data analysis, I have replaced the null values with single white space and then merge the feature headline, news, written_by to make a single feature text_feature and then dropped the feature headline, news and written_by. But these null values can also be removed by dropping the entire row and proceed further without merging the features into single feature which might impact the model performance either in positive or negative way. As of now, I am finishing this project with my current approach which gives the **final accuracy score of 98.03% and log loss: 0.67** and this can be further improved by training with more specific data.

- Limitations of this work and Scope for Future Work

Current model is limited to news data but this can further be improved for other sectors of fake document detection by training the model accordingly. The overall score can also be improved further by training the model with more specific data.