Project Title:

Fake News Classification Web app

by Team APACHE

Problem Brief

Fake News A Real Problem - *The plague eating up social media*

The destructive and catastrophic import of fake news can not be overemphasis and utterly underestimated, Though fake news start subtle and goes unnoticeable in the early stages, but when allow to breed, birth violent outcomes which is capable of instigating social, political wars, and capable of causing psychological effect on individuals targeted at, especially today, amid a pandemic, social media platforms are being used to dish out misinformation at lightning speed. One thing we can do is to avoid news altogether which seems nearly impossible or one can utilize tools such as those of machine learning to fight the fatigue of fake new - **This is the intent of this project**

Project Scope And Boundary

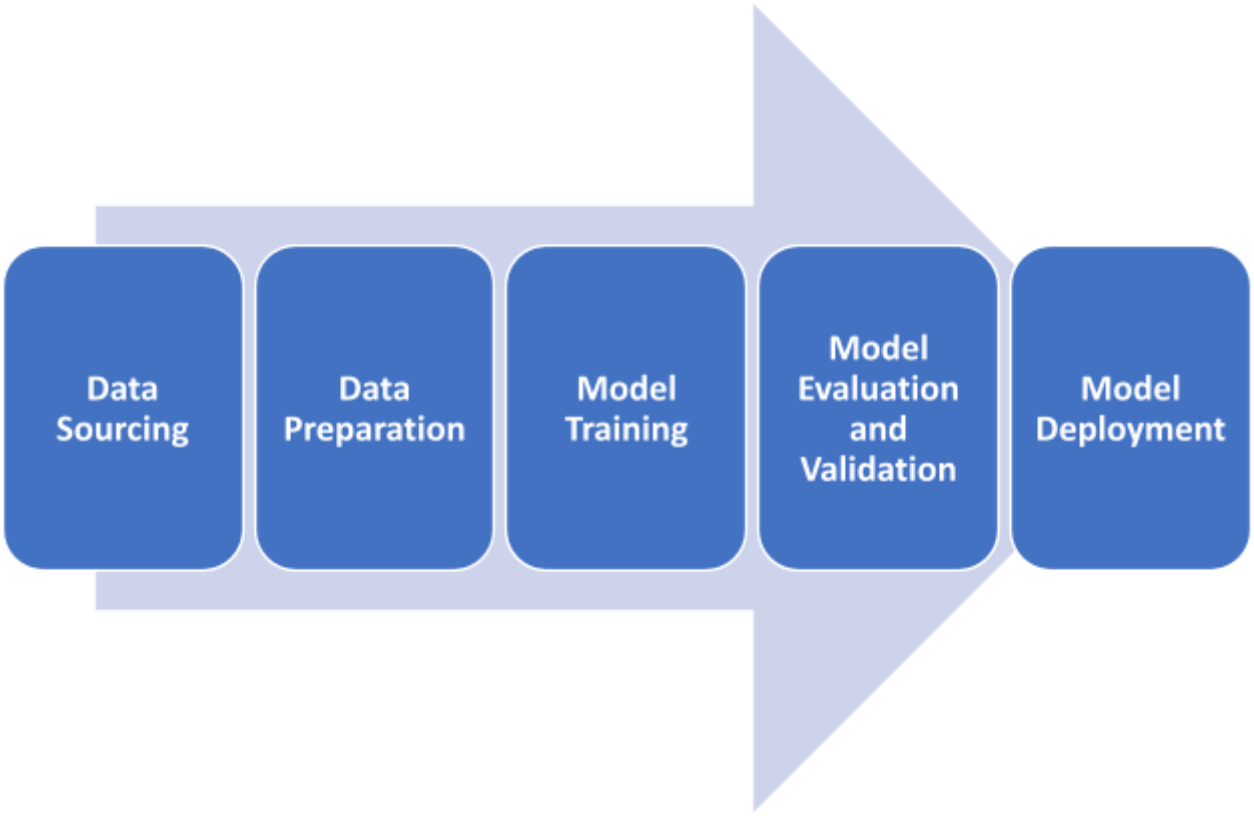
- Kaggle fake news twitter dataset was used for this analysis. link [fake_news_dataset (https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset)](https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset)
- The news niche focus on polical news in the united states
- The news article examined in the dataset is 2 years old.

Project Aim

The objective of this article is to outline end-to-end steps on building and training a machine learning model model to classify fake and true news using the best performing algorithm and deploying this model using Streamlit.

The dataset source is from Kaggle : Fake News dataset from the InClass Prediction Competition.
All notebooks and scripts used can be found on apache GitHub repo [apache-21 (https://github.com/apache-21/fake_news_detection)](https://github.com/apache-21/fake_news_detection). This article will illustrate 7 steps which are outline in the project workflow section.

Project Workflow :



1. Data Source : Data gotten from kaggle.com
2. Data Preprocessing of text
    - a. Exploratory Data Analysis
    - b. Data cleaning and feature engineering
    - c. Visualization
3. Model Selection and Evaluation
4. Data Pipeline
5. Model Deployment
6. Consolidation and Discussion

Data Preprocessing

Library use for project development:

- Pandas for data analysis
- numpy for numerical computation

- matplotlib for visualisation
- spacy for information extraction to perform such as (NER, POS tagging, dependency parsing, word vectors
- nltk for text preprocessing, converting text into numbers for the model.
- Seaborn for visualization
- textblob for text preprocessing, such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation
- re library for to search and find patterns in the tweets
- wordcloud which is a visualization technique for text data wherein each word is picturized with its importance in the context or its frequency.
- pickle to save the model and acess it

Project Methodology

Preparing the dataset

The dataset from Kaggle is provided in 2 CSV files which are already classified between true and fake news. The dataset was loaded using the pandas library however since it is textual data we carried out data cleaning, pre-processing, EDA and model-building operations.

Below is an overview of the dataset

In [6]:

```
fake_news = pd.read_csv('Fake.csv')
fake_news.head()
```

Out[6]:

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017

In [7]:

```
fake_news.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23481 entries, 0 to 23480
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   title    23481 non-null  object
 1   text     23481 non-null  object
 2   subject  23481 non-null  object
 3   date     23481 non-null  object
dtypes: object(4)
memory usage: 733.9+ KB
```

The dataset was examine for missing values, and it is interesting that there are no missing values
we have a dataset of 4 features and 23481 observation

More so, upon examining the dataset, it is observe that there are some words present in the data which are irrelavant to the model, and as result we use regular expression to get such unwanted patterns of text in the data and this is utilize in a function to search and find such words and phrase which is then filter it from the dataset.

The re.sub() function is used to replace occurrences of a particular sub-string with another sub-string.

```
def operate_on_word(text):
    text = re.sub('\w\d\w', '',
      re.sub('\n', '',
        re.sub('[%s]' % re.escape(string.punctuation), '',
            re.sub('<.?>+', '',
              re.sub('https?://\S+|www.\S+', '',
                re.sub("\W", ' ',
                  re.sub('[.?]', '', text.lower()))))))
return text
```

In [7]:

```
# fake_corpus - patterns filter using the helper function
fake_corpus = ' '.join(fake_news.text.apply(operate_on_word))

len(fake_corpus)
```

Out[7]:

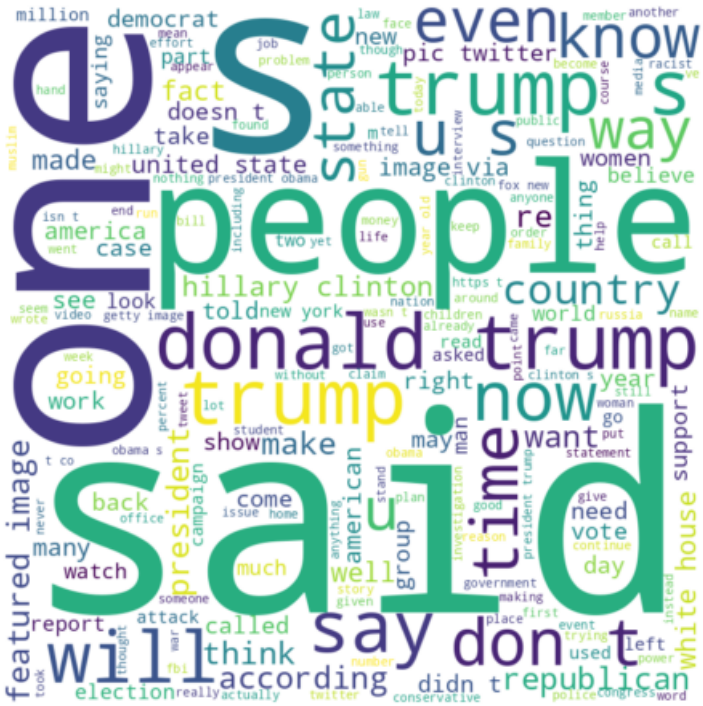
```
59145324
```

Generating the WordCloud from the preprocess dataset

WordCloud : is a visualization technique for text data wherein each word is picturized with its importance in the context or its frequency. However to generate the wordcloud there is a need to define stopwords.

In this entire process of generating a word cloud or processing any text data, we will always have a set of words that is not much of a concern to us. Words that belong to this category of “futile” words include is, was, for, of, it, a, the, etc. As a process of filtering data, we use stopwords to remove useless words.

Below is the WordCloud generated from the preprocess fake news dataset:



Subsequenly The spacy library was utilize for part of speech dependency tagging - This allows us know and understand the different part of speech in the text data and how there interdependency

Futhermore, the content of the true news data was examine

In [34]:

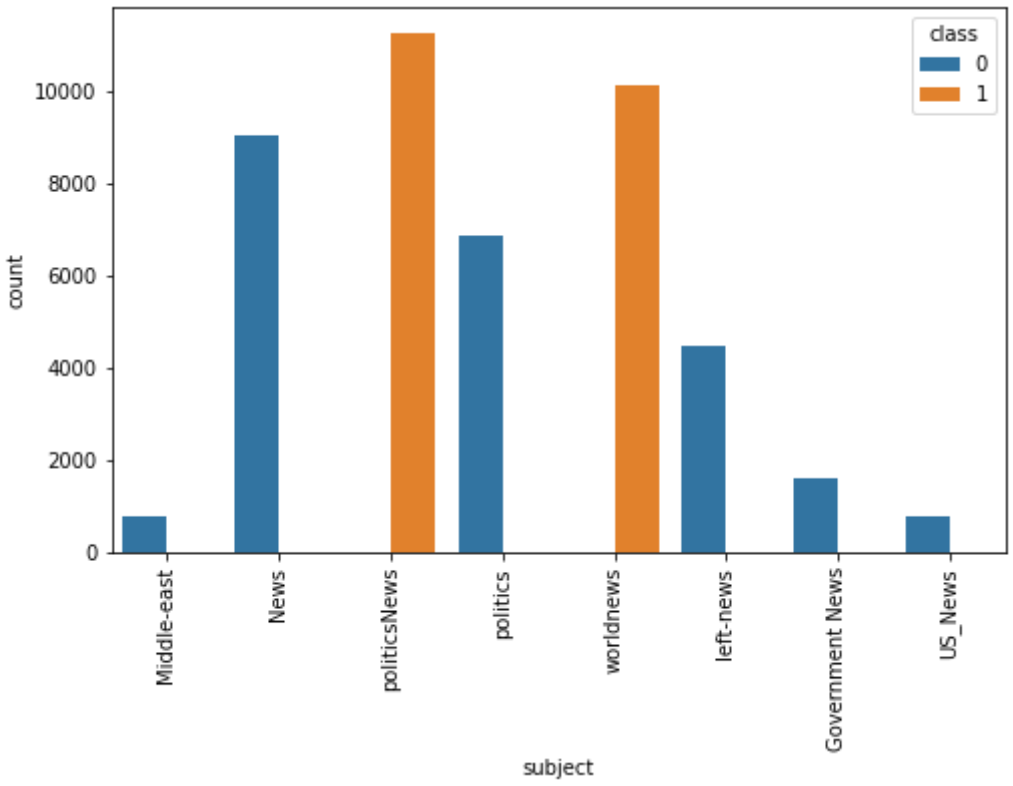
```
genuine_news = pd.read_csv('True.csv')

genuine_news.head()
```

Out[34]:

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

Examining the Class and the Subject of the news Content in the dataset

The fake news data set and genuine news where merge together and the visualization of the class and subject to the category of news in the dataset with the respective frequency is shown below:



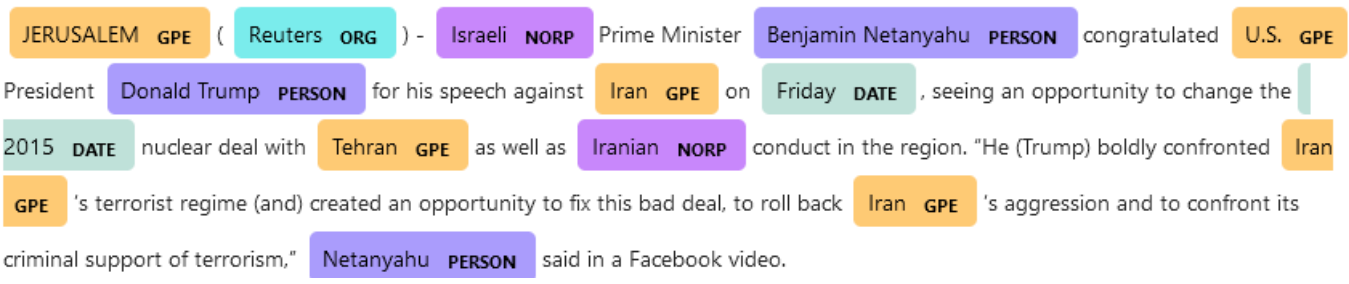
Observation

- the domain subject appears to be heavy on polical news

Performing Name Entity Recognition on the data

Named entity recognition (NER) – also called entity identification or entity extraction – is a natural language processing (NLP) technique that automatically identifies named entities in a text and classifies them into predefined categories.

Name entity recognition was carried out on the text data to extract key names and entities present in the dataset and the below is the visualization



The fake news and genuine dataset were merge together using the pd.concat library and a new dataframe is form (df)

Feature Engineering

Inorder to draw more insight from the dataset new features were engineered such as

- Polarity : which is an output of the textblob which gives the ability of knowing the sentinent in each tweet.
- text_len : gives the length of each text or tweet size
- text_word_count : gives the count of word in the text
- title_len : gives the size of the tweet title

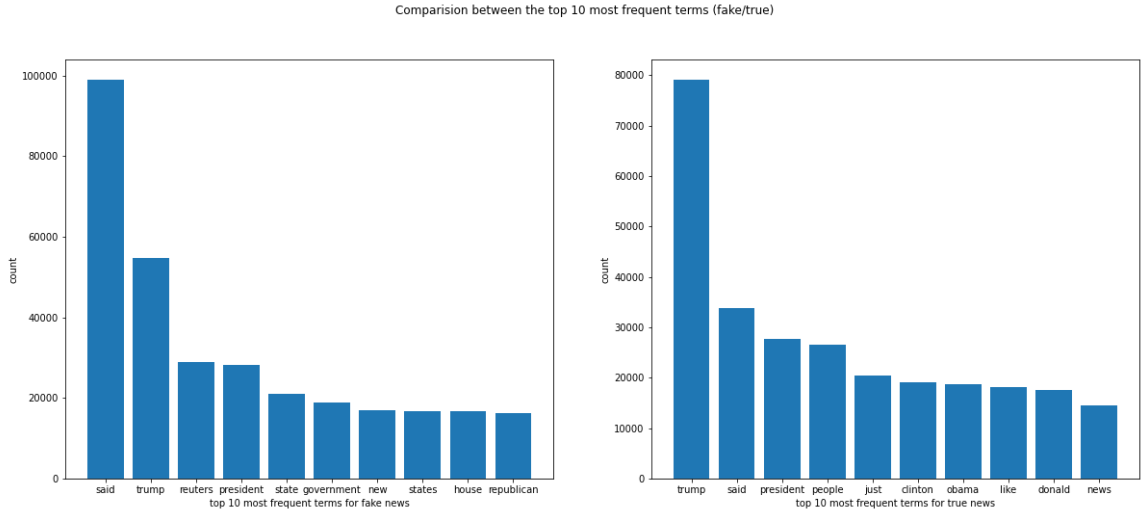
Transforming text data into numerical values

Machine learning algorithm thrive on numerical values hence library such the countvectorizer,bag of words model was use to achieve the numerical transformation.

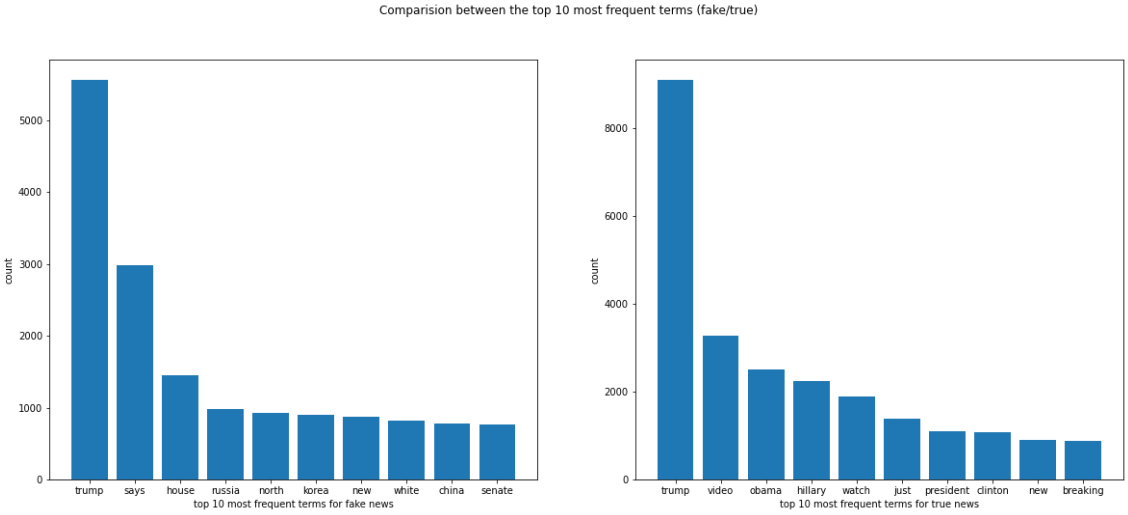
A helper function get_top_n_words is define for this numerical transformation and to also get the top words with visualization

```
def get_top_n_words(corpus, name='text', n=None):
    corpus_fake = corpus[df['genuine'] == 1].astype(str)
    corpus_true = corpus[df['genuine'] == 0].astype(str)

    vec = CountVectorizer(stop_words = 'english').fit(corpus_fake)
    bag_of_words = vec.transform(corpus_fake)
    sum_words = bag_of_words.sum(axis=0)
    words_freq_fake = [(word, sum_words[0, idx]) for word, idx in vec.vocabular
    y_.items()]
    words_freq_fake =sorted(words_freq_fake, key = lambda x: x[1], reverse=Tru
    e)

    vec = CountVectorizer(stop_words = 'english').fit(corpus_true)
    bag_of_words = vec.transform(corpus_true)
    sum_words = bag_of_words.sum(axis=0)
    words_freq_true = [(word, sum_words[0, idx]) for word, idx in vec.vocabular
    y_.items()]
    words_freq_true =sorted(words_freq_true, key = lambda x: x[1], reverse=Tru
    e)

    df_fake = pd.DataFrame(words_freq_fake[:n],columns = ['text', 'count'])
    df_true = pd.DataFrame(words_freq_true[:n],columns = ['text', 'count'])

    fig, (ax1, ax2) = plt.subplots(1,2,figsize=(20,8))
    ax1.bar(df_fake['text'], df_fake['count'])
    ax1.set_xticklabels(df_fake['text'])
    ax1.set(xlabel='top 10 most frequent terms for fake news', ylabel='count')
    ax2.bar(df_true['text'], df_true['count'])
    ax2.set_xticklabels(df_true['text'])
    ax2.set(xlabel='top 10 most frequent terms for true news', ylabel='count')
    plt.suptitle('Comparision between the top 10 most frequent terms (fake/tru
    e)')

    fig.savefig(f'most_freq_{name}.png')
```

visualization for the most frequent text



visualization for the most frequent title in the data



Observation from the above visualization:

Based on the comparison between the top 10 frequent words in titles and news text, we can infer that both fake and true news is dominated by news relating to politics and more specifically, the subject being heavily related to American politics is shared between true and fake news. This would result in the model been biased to classifying news that relates to only American Politics and probably of that time frame. To mitigate this bias more recent data and diverse news data would be needed

Model Selection and Data Pipeline

The dataset was splitted into test and train set and an helper function is created to remove unwanted patterns in the dataset then passed to through a model through another helper function that both transform and preprocess the text into numerical values then make prediction

List of Classifier Models Use

Classical machine learning algorithms were utilize for this classifier, thne a deep learning model lstm was also used.

- Naive bayes - multinomial
- Logistics regression

- Random forest classifier
- Gradient boosting classifier
- Lstm - Deep learning model

Below is the helper function - a data pipeline that take the dataset or tweet then preprocess it and feed it into the model.

data pipeline which entails transformimng the data to numerical data with removal of irrelevant pattern present.

```
def load_classifier(clf, X_train, X_valid, y_train, y_valid):
    pipe_clf = make_pipeline(FunctionTransformer(transform_word),
        TfidfVectorizer(ngram_range=(1, 2), max_features=5000),clf)
    pipe_clf.fit(X_train, y_train)      y_pred = pipe_clf.predict(X_valid)      probas = pipe_clf.predict_proba(X_valid)

   return pipe_clf, y_pred, probas


classifiers = [MultinomialNB(), LogisticRegression(),RandomForestClassifier(), GradientBoostingClassifier()]

model_list, preds_list, probas_list = [], [], []

for clf in classifiers:
    model, pred, probas = load_classifier(clf, X_train_all, X_test, y_train_all, y_test)
    model_list.append(model)
    preds_list.append(pred)
    probas_list.append(probas)
```

Model Evaluation

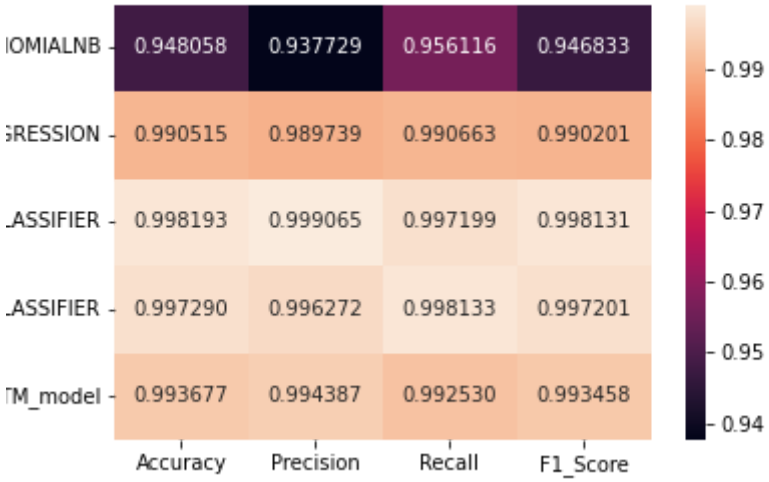
The performance of the model on the validation is examine using the evaluation metrics below:

- confusion matrix
- Accuracy score
- Precision,
- Recall,
- f1score

Training a Deep neural network(LSTM) on the dataset

The dataset was first preprocess for the neural netowrk then train using the lstm models

The models were evalauted and the result shown in the table below:



The random forest seems to outperform other classifer thus, it will be consider as the choice algorithm use in the deloyement phase

However, it will be interesting to check the perfomance of the deep learning model (lstm) on the dataset.

In [27]:

```
Eval_metrics = pd.read_csv('img/all_metrics_df.csv')

Eval_metrics
```

Out[27]:

	Unnamed: 0	Accuracy	Precision	Recall	F1_Score
0	MULTINOMIALNB	0.948058	0.937729	0.956116	0.946833
1	LOGISTICREGRESSION	0.990515	0.989739	0.990663	0.990201
2	RANDOMFORESTCLASSIFIER	0.998193	0.999065	0.997199	0.998131
3	GRADIENTBOOSTINGCLASSIFIER	0.997290	0.996272	0.998133	0.997201
4	LSTM_model	0.993677	0.994387	0.992530	0.993458

Picking the boss model and Saving the model using pickle

The random forest seems to be the boss, hence it is chosen and the model is save using pickle and can then be use for future prediction.

```
model_file_list = []
for model in model_list:
    model_name = model.steps[-1][0]
    filename = f'{model_name}_model.pkl'
    model_file_list.append(filename)
    pickle.dump(model, open(filename, 'wb'))
```

Model Deployement

The fake news classification app is then deploy on the web using streamlit and readily available to end users for use

Fake news Classifiaction source code link : [apache-21 (https://github.com/apache-21/fake_news_detection/blob/main/fake_new_detection_app/fake-news-5.ipynb)](https://github.com/apache-21/fake_news_detection/blob/main/fake_new_detection_app/fake-news-5.ipynb)

In [ ]: