ONLINE SOCIAL NETWORK ANALYSIS

PROJECT II - SOCIAL MEDIA DATA ANALYSIS

PROJECT REPORT : FAKE NEWS CLASSIFICATION

Submitted by:
Rahul Kolli
Kruthi Kanukuntla

I. INTRODUCTION

This project focuses on the detection of fake news/misinformation that are widely spread through social media. The primary objective is to classify a news article B as agreed, disagreed, or unrelated based on its title and a previously identified fake news article A. The data provided includes a training set, a test set, and a sample submission file for formatting predictions. The training set can be divided into training and validation subsets to train and evaluate with various machine learning algorithms like decision trees, random forests, naïve and bayes.

Once the model is trained and evaluated, it can be used to make predictions on the test set, and the output can be saved in the same format as the sample submission file. The final submission file should contain the predicted labels for the test set. The rampant spread of fake news and misinformation on social media can have detrimental effects on individuals and society at large. Therefore, this project's significance lies in its potential to mitigate the negative impact of fake news and misinformation by accurately identifying and classifying them.

II. SYSTEM MODULES

The steps to be followed are :

1. Importing the Libraries
2. Importing the Dataset
3. Pre-processing the Dataset
4. Model Training
5. Model Evaluation
6. Model Prediction

A. IMPORTING THE LIBRARIES:

The libraries we will be using are :

- Pandas: To load dataset.
- SkLearn: For train-test split and to import the modules for model evaluation.
- Matplotlib ,Seaborn, tabulate: For data visualization.
- Re: For regular expressions during data pre-processing stage.
- Nltk: To support classification task.

Importing Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

import nltk
from nltk.corpus import stopwords
from nltk.corpus import wordnet as wn
from nltk.stem.wordnet import WordNetLemmatizer

from tabulate import tabulate
```

B. IMPORTING THE DATASET:

We import the training dataset from the csv files using pandas into a dataframe and use this dataframe further for data preprocessing and passing into the model.

Importing Dataset

```
In [2]: # Loading the data
        fakenews_train_dataset = pd.read_csv("train.csv")
        fakenews_train_dataset.head()
```

Out[2]:

	id	tid1	tid2	title1_en	title2_en	label
0	195611	0	1	There are two new old-age insurance benefits f...	Police disprove "bird's nest congress each per...	unrelated
1	191474	2	3	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outstrips Hong Kong? Shenzhen S...	unrelated
2	25300	2	4	"If you do not come to Shenzhen, sooner or lat...	The GDP overtopped Hong Kong? Shenzhen clarifi...	unrelated
3	123757	2	8	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP overtakes Hong Kong? Bureau of ...	unrelated
4	141761	2	11	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outpaces Hong Kong? Defending R...	unrelated

C. PRE-PROCESSING THE DATASET

Data Cleaning is one of the a crucial steps in the Data Pre-processing phase . Data quality issues such as duplication, mislabelling, irrelevant symbols, URLs, and inaccuracies can significantly impact the accuracy of the results produced by any algorithm, even if the outputs may appear to be correct. Therefore, it is important to handle these issues appropriately to ensure that our model produces accurate results and ensure that businesses can make informed decisions based on accurate and reliable analysis of data.

- **REMOVING MISSING DATA AND UNWANTED COLUMNS**

We first check for null values in each column.

Data Prepocessing

```
In [5]: fakenews_train_dataset.dtypes
```

```
Out[5]: id            int64
        tid1          int64
        tid2          int64
        title1_en    object
        title2_en    object
        label        object
        dtype: object
```

```
In [4]: fakenews_train_dataset.isnull().sum()
```

```
Out[4]: id           0
        tid1         0
        tid2         0
        title1_en    0
        title2_en    0
        label        0
        dtype: int64
```

After analysing the dataset, we can conclude that there are no null values and the data is complete. Therefore, there is no need to handle missing values. Additionally, we have determined that the columns 'tid1' and 'tid2' are not relevant in identifying fake news or misinformation, and thus we have decided to remove them from the dataset.

```
In [8]: cols = ['tid1', 'tid2']
        fakenews_train_dataset = fakenews_train_dataset.drop(cols, axis=1)
        fakenews_train_dataset.head()
```

Out[8]:

	id	title1_en	title2_en	label
0	195611	There are two new old-age insurance benefits f...	Police disprove "bird's nest congress each per...	unrelated
1	191474	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outstrips Hong Kong? Shenzhen S...	unrelated
2	25300	"If you do not come to Shenzhen, sooner or lat...	The GDP overtopped Hong Kong? Shenzhen clarifi...	unrelated
3	123757	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP overtakes Hong Kong? Bureau of ...	unrelated
4	141761	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outpaces Hong Kong? Defending R...	unrelated

- **TEXT CLEANING**

Text cleaning is the process of preparing raw text for machines to understand human language. Clean text refers to human language that has been rearranged into a machine-readable format. Text cleaning typically involves using Python code to eliminate stopwords, remove non-ASCII characters, and reduce complex words to their root form using techniques like lemmatization or stemming.

We need to perform the two most basic text cleaning techniques on this query:

1. **Normalizing Text:**

In order to make the text analysis process easier for machines, we need to perform text normalization techniques:

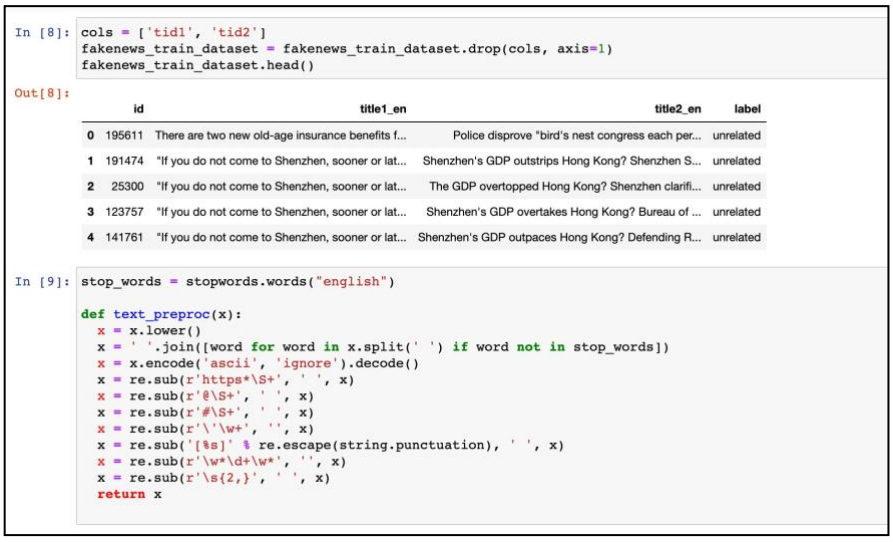
- that convert words to their basic or root form(Lemmatization)
- removing special characters
- removing html tags
- removing punctuations
- standardize the letter case to lower case
- @mentions, URLs, and emojis are also converted into Unicode

This step ensures that the text is uniform and standardized, and we can reduce the complexity of the text while retaining the important information.

For example, we can convert "running", "runs", "ran" to "run", which is the root form of the word. Similar, "Amazon" and "AMAZON" to "amazon" to ensure consistency. Additionally, we can remove special characters like "@", "#" and punctuations like ".", ",", "?" which may not provide much value for text analysis.

2. **Removing Stopwords:**

After removing punctuations and converting the text to lowercase, we still may have some words that do not add any meaning to the text and can be safely removed without affecting the context of the text. These words are known as "stopwords". To eliminate these words, we can use pre-existing lists of stopwords for various languages, including English. By removing these words, we can simplify the text and improve the efficiency of our analysis.



In [8]:
```
cols = ['tid1', 'tid2']
fakenews_train_dataset = fakenews_train_dataset.drop(cols, axis=1)
fakenews_train_dataset.head()
```

Out[8]:

	id	title1_en	title2_en	label
0	195611	There are two new old-age insurance benefits f...	Police disprove "bird's nest congress each per...	unrelated
1	191474	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outstrips Hong Kong? Shenzhen S...	unrelated
2	25300	"If you do not come to Shenzhen, sooner or lat...	The GDP overtopped Hong Kong? Shenzhen clarifi...	unrelated
3	123757	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP overtakes Hong Kong? Bureau of ...	unrelated
4	141761	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outpaces Hong Kong? Defending R...	unrelated

In [9]:
```
stop_words = stopwords.words("english")

def text_preproc(x):
  x = x.lower()
  x = ' '.join([word for word in x.split(' ') if word not in stop_words])
  x = x.encode('ascii', 'ignore').decode()
  x = re.sub(r'https*\S+', ' ', x)
  x = re.sub(r'@\S+', ' ', x)
  x = re.sub(r'#\S+', ' ', x)
  x = re.sub(r'\'\w+', '', x)
  x = re.sub('[%s]' % re.escape(string.punctuation), ' ', x)
  x = re.sub(r'\w*\d+\w*', '', x)
  x = re.sub(r'\s{2,}', ' ', x)
  return x
```

D. MODEL TRAINING

After loading the training data, we split it into training and validation sets. We split the training dataset into 80% training and 20% validation sets. We use 'title1_en' and 'title2_en' columns as the input variables and 'label' column as the output variable while training the data.

Vectorization is an essential step in natural language processing that involves converting textual data into numerical vectors. Once the text has been pre-processed and cleaned, vectorization can significantly enhance the efficiency and accuracy of machine learning models. There are various techniques available for vectorization, each with its own advantages and drawbacks. We use the Tfidf Vectorizer from scikit-learn to convert the text data into a matrix of numerical features.

```
In [10]: # Concatinating title 1 and title 2 and passing it as input variable.Making lable the output variable
         X = fakenews_train_dataset['title1_en']+fakenews_train_dataset['title2_en']
         y = fakenews_train_dataset['label']

         # Spliting the data into 80 and 20
         X_train,X_test,y_train,y_test= train_test_split(X,y, train_size=0.80,random_state=1)
         print(X_train.shape,y_train.shape,X_test.shape,y_test.shape)

         (205153,) (205153,) (51289,) (51289,)

In [11]: vectorizer = TfidfVectorizer()
         train_vectors = vectorizer.fit_transform(X_train)
         test_vectors = vectorizer.transform(X_test)

In [12]: model_nb = MultinomialNB()
         model_nb.fit(train_vectors, y_train)

         val_predictions_nb = model_nb.predict(test_vectors)

In [13]: model_rf = RandomForestClassifier()
         model_rf.fit(train_vectors, y_train)

         val_predictions_rf = model_rf.predict(test_vectors)

In [14]: model = DecisionTreeClassifier(random_state=42)
         model.fit(train_vectors, y_train)

         val_predictions_dt = model.predict(test_vectors)

In [15]: model_knn = KNeighborsClassifier(n_neighbors=5)
         model_knn.fit(train_vectors, y_train)

         val_predictions_knn = model_knn.predict(test_vectors)
```

After vectorization we use machine learning algorithms to train the model. The algorithms that we used are:

1. Multinomial Naive Bayes
2. Decision trees
3. Random forests
4. K-nearest neighbours

These four algorithms are commonly used for text classification tasks. Each algorithm has its own strengths and weaknesses, and the best algorithm to use depends on the specific task, performance of the model and the characteristics of the data.

- **Multinomial Naive Bayes** is a good choice for text classification tasks because it is computationally efficient and works well for high-dimensional data. It assumes that the features of the data are independent, which is a reasonable assumption for text data. This assumption makes it computationally efficient and easy to interpret, but it can be violated if the data contains complex interactions between the features.

- **Decision Trees and Random Forests** are also well-suited for text classification tasks. They can handle both categorical and continuous data, can easily handle missing data, and can handle non-linear relationships between features. Decision trees are also interpretable, which can be useful for understanding the decision-making process of the model. However, decision trees can be sensitive to overfitting, which is when the model learns the training data too well and does not generalize well to new data. But whereas, random forests can be used to reduce the risk of overfitting and improve the generalization performance of the model.
- **K-Nearest Neighbours** is another algorithm that can be used for text classification, particularly for small datasets with few features. It is simple to implement and can handle both categorical and continuous data, as well as missing data. However, it may not handle high-dimensional data well and can be computationally expensive.

E. Model Evaluation

Model evaluation is the process of assessing how well a machine learning model performs on new data. It involves measuring various metrics that indicate how well the model is able to predict the target variable. The goal of model evaluation is to determine the generalization performance of the model and identify best algorithm/model for predicting test data.

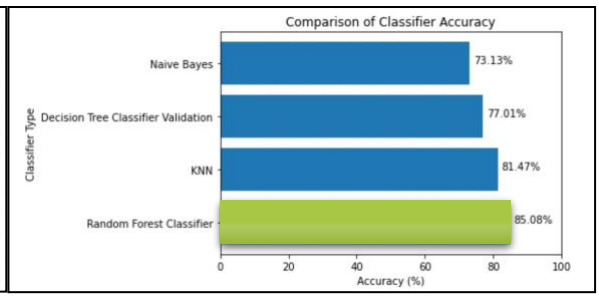
There are several metrics that can be used to evaluate the performance of a machine learning model, depending on the type of problem and the nature of the data. Here are some common metrics for classification problems:

- **Accuracy** is the proportion of correct predictions out of the total number of predictions. It is a simple and intuitive metric, but it can be misleading if the data is imbalanced.
- **Precision** is the proportion of true positives (correctly predicted positive instances) out of the total number of predicted positive instances. It is a measure of how accurate the model is at predicting positive instances.
- **Recall** is the proportion of true positives out of the total number of actual positive instances. It is a measure of how complete the model is at predicting positive instances.
- **F1 Score** is the harmonic mean of precision and recall, which gives equal weight to both metrics. It is a more comprehensive metric than accuracy, as it takes into account both the accuracy and completeness of the model.

It is also important to note that no single metric can provide a complete picture of the performance of a machine learning model. It is often helpful to look at multiple metrics to get a better understanding of the model's performance.

Accuracies for selected classifier models:

Classifier Type	Accuracy(%)
Decision Tree Classifier Validation	77.01
Random Forest Classifier	85.08
Naive Bayes	73.13
KNN	81.47

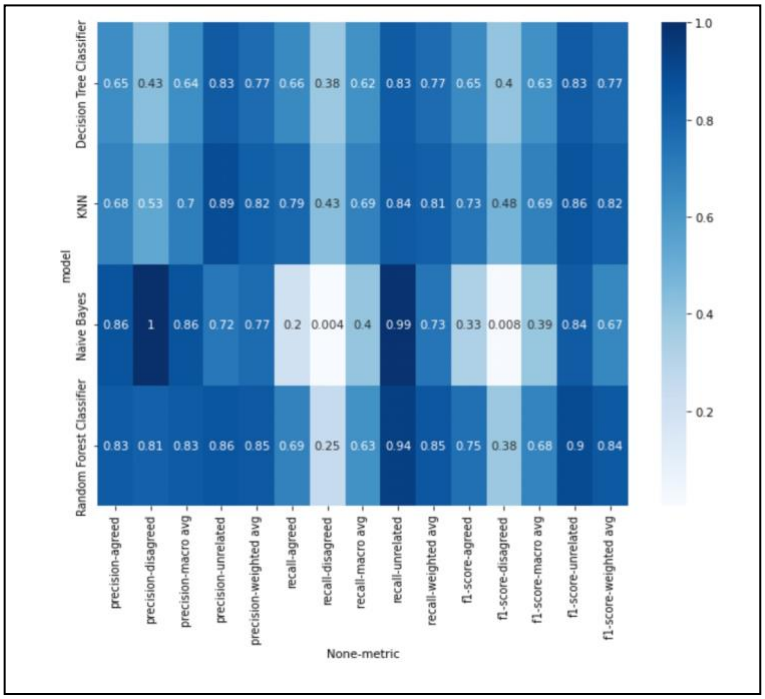


In our case the **best accuracy** is calculated for **Random Forest Classifier.**

It is crucial to select appropriate metrics for evaluating the performance of a machine learning model based on the specific problem and dataset. In situations where the data is imbalanced, accuracy may not be the only metric that can accurately assess the performance of a model. This is because a model can achieve high accuracy by simply predicting the majority class, which may not be useful in identifying patterns in the minority class.

Therefore, in such cases, it is important to use additional metrics such as precision, recall, and F1-score in conjunction with accuracy to select the best model. These metrics provide a more comprehensive understanding of the model's performance by taking into account the model's ability to correctly identify both the majority and minority classes. By using multiple metrics, we can better evaluate the model's performance and ensure that we select the best model for our specific problem and dataset.

We used a heatmap to visually compare the precision, recall and F1-score of each model selected. This visually helps to quickly compare the performance of the models across multiple classes or categories.



Based on the heatmap analysis, we can observe that Naïve Bayes has higher precision values compared to the other models. However, when we consider both precision and recall together, we can see that Random Forest performs better than Naïve Bayes. This means that Random Forest has a better balance between correctly identifying positive cases (precision) and capturing all positive cases (recall) compared to Naïve Bayes.

Therefore, it can be concluded that Random Forest is a better performing model for the given problem, taking into account both precision and recall metrics.

F. Model Prediction

Model prediction is the process of using a machine learning model to make predictions on new data. The model is trained on a dataset of historical data, and then it is used to make predictions on new data. The predictions can be used to improve decision-making in a variety of applications.

In a classification model, the output will be a discrete value that represents the predicted class for the target variable.

Before making predictions on new data, it is important to ensure that the input data is pre-processed in the same way as the data used to train the model.

```
# load the test data
test_df = pd.read_csv('test.csv')
test_df['title1_en'] = test_df.title1_en.apply(text_preproc)
test_df['title2_en'] = test_df.title2_en.apply(text_preproc)
test_df.head()
```

	id	tid1	tid2	title1_en	title2_en
0	256442	100672	100673	great coat brother zhu zhu wen mandarin love s...	lin xinsheng birth hard milking huo jianhua se...
1	256443	162269	162270	nasa reveals fact ufo wreckage found moon	ufo found yuancun jiaocheng county shanxi shoc...
2	256444	157826	157854	hollow tomato loaded hormone	li chenfan bingbing home photo netizen called ...
3	256445	109579	74076	ange pavilion geoshui accurate matrimony match...	master one eight character presumption marriag...
4	256446	15068	15085	year old bus bus blow year old child rumor rum...	joe johnson disgruntled timing order myth

After comparing the performance of all the selected models, we found that Random Forest outperformed the others. Therefore, we used Random Forest to predict the label column in the test dataset. To make the prediction, we used both titles columns in the dataset as input variables, just as we did in the training dataset. Once we made the prediction, we wrote the predicted values along with their respective IDs into a submission CSV file.

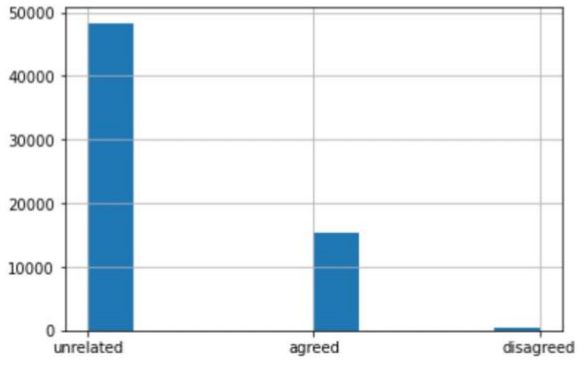
```
test_texts = test_df['title1_en'] + test_df['title2_en']
test_vectors = vectorizer.transform(test_texts)

# make predictions on the test set
test_predictions = model_rf.predict(test_vectors)

# save the predictions to a file in the required format
submission_df = pd.DataFrame({'id': test_df['id'], 'label': test_predictions})
submission_df.to_csv('submission.csv', index=False)
```

The images below provide the output file of the model's predictions on the test data and a graphical analysis of the label column. The output file shows the predicted values of the model for each instance in the test dataset. The graphical analysis of the label column displays the distribution of the predicted values across the different categories.

	A	B
1	id	label
2	256442	unrelated
3	256443	unrelated
4	256444	unrelated
5	256445	unrelated
6	256446	unrelated
7	256447	unrelated
8	256448	unrelated
9	256449	unrelated
10	256450	unrelated
11	256451	unrelated
12	256452	agreed
13	256453	agreed
14	256454	agreed
15	256455	unrelated
16	256456	unrelated
17	256457	unrelated
18	256458	unrelated
19	256459	unrelated

```
submission_df['label'].hist()
submission_df['label'].value_counts()
```

```
unrelated    48340
agreed       15267
disagreed      503
Name: label, dtype: int64
```



III. CONCLUSION :

This project is addressing the issue of fake news and misinformation. By developing a classification model that can accurately identify and classify fake news articles, we can minimize the negative impact of these articles on society. After testing multiple models over the training data, and analysis on the metrics on them we finalized Random Forest as the best model with **85.08%** accuracy and over all best performance of F1-score for prediction and used it for predicting 'label' column on test dataset.

The results obtained from this project can serve as a foundation for future work in the field of fake news detection, ultimately leading to a safer and more informed society.

IV. REFERENCES:

[1] https://heartbeat.comet.ml/machine-learning-for-classifying-social-media-ads-f9be75910ec3
[2] https://www.geeksforgeeks.org/fake-news-detection-using-machine-learning/?ref=rp#
[3] https://towardsdatascience.com/cleaning-text-data-with-python-b69b47b97b76]
[4] https://monkeylearn.com/blog/text-cleaning/
[5] https://heartbeat.comet.ml/vectorization-in-machine-learning-2e3bdce7dbe
[6] https://www.geeksforgeeks.org/fake-news-detection-model-using-tensorflow-in-python/
[7] https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
[8] https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234
[9] NumPy. (2021). NumPy: A fundamental package for scientific computing with Python. Retrieved from https://numpy.org/
[10] McKinney, W., & others. (2011). pandas: a foundational Python library for data analysis and statistics. Python for High Performance and Scientific Computing, 14, 1-9. Retrieved from https://pandas.pydata.org/docs/
[11] Scikit-learn. (2021). Scikit-learn: Machine learning in Python. Retrieved from https://scikit-learn.org/stable/
[12] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Retrieved from https://matplotlib.org/
[13] Waskom, M. (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021. Retrieved from https://joss.theoj.org/papers/10.21105/joss.03021
[14] Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace. Retrieved from https://www.python.org/doc/3/library/re.html
[15] Bird, S., Loper, E., & Klein, E. (2009). Natural language processing with Python. O'Reilly Media, Inc. Retrieved from https://www.nltk.org/book/