ILLINOIS INSTITUTE OF TECHNOLOGY

PROJECT REPORT

CSP 579 Online Social Network Analysis

May 26th, 2023

Fake News Classification

Team Members

Jasleen Kaur (A20495420, jbhatia@hawk.iit.edu)

Pranjal Naik (A20489131, pnaik13@hawk.iit.edu)

INDEX

I. ABSTRACT

These days, news about current events and particular areas of interest is broadcast on radio and television as well as written in newspapers and on social media worldwide. Given the rapid expansion of online material, it is getting harder to distinguish between genuine and fake information. Fake news has consequently spread like wildfire, making it extremely difficult for makers of such information to verify it to prevent public misinformation. The process of fact-checking rumors and claims needs to be put in place, especially for those that receive thousands of views and likes before being disputed and discredited by reliable sources. In this project, we will show an approach for fake news classification by using Logistic Regression.

II. OVERVIEW

In today's fast-paced digital world, the extensive spread of information has given rise to a significant challenge: the prevalence of fake news. This phenomenon not only undermines the credibility of media outlets but also has the potential to distort public opinion, and impact decision-making. It could also lead to the compromise of the integrity of democratic processes. The 'Fake News Classifier' project aims to tackle this critical issue and help create a more reliable information landscape for all. In this project, we have used Logistic regression which is machine learning algorithm to classify the news data the news to one of the three categories agreed or disagreed or unrelated. This algorithm is simpler and interpretable that can be more suitable for smaller datasets. Firstly, we pre-process and clean a data set. Secondly, we use TF-IDF vectorizer to convert data into vectors. Third, we split the train data to train the model and classifier and evaluate it performance with the validation data. Finally, we use the trained model to classify new data set and predict the results.

III. DATASET DESCRIPTION

As per the dataset provided, we have 3 CSV files:

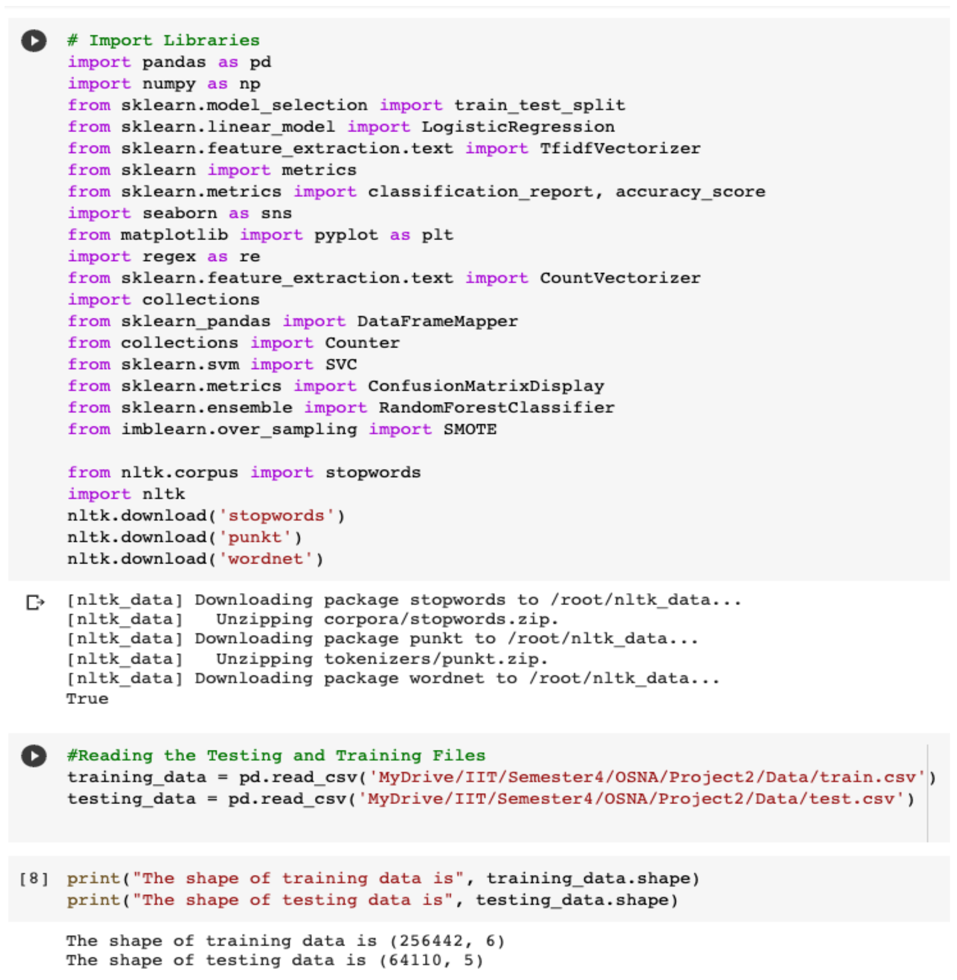
- *train.csv*: contains the training data,
- *test.csv*: contains the test data,
- *submission.csv*: Expected submission layout

With columns as –

- *id*: the unique id of news pair.
- *label*: indicates the relation between the news pair - agreed/disagreed/unrelated.

The distribution of the label in the dataset is as follows –

- *Unrelated* - The number of records is 175598.
- *Agreed* - The number of records is 74238.
- *Disagreed* - The number of records is 6606.



```
# Import Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import metrics
from sklearn.metrics import classification_report, accuracy_score
import seaborn as sns
from matplotlib import pyplot as plt
import regex as re
from sklearn.feature_extraction.text import CountVectorizer
import collections
from sklearn_pandas import DataFrameMapper
from collections import Counter
from sklearn.svm import SVC
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE

from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
#Reading the Testing and Training Files
training_data = pd.read_csv('MyDrive/IIT/Semester4/OSNA/Project2/Data/train.csv')
testing_data = pd.read_csv('MyDrive/IIT/Semester4/OSNA/Project2/Data/test.csv')
```

```
[8] print("The shape of training data is", training_data.shape)
    print("The shape of testing data is", testing_data.shape)
```

```
The shape of training data is (256442, 6)
The shape of testing data is (64110, 5)
```

IV. TOOLS

- *Programming Language* - Python
- *Applications/Framework* – Google Colab
- *Libraries* – Sklearn, Numpy, pandas, seaborn, matplotlib,regex,collections, sklearn_pandas,nltk,imbanced-learn,imblearn.
- *Development* – GitHub

V. DATA PREPROCESSING

Data Preprocessing is a critical step in building an accurate and reliable machine learning model for fake news classification. By carefully cleaning, transforming, and balancing the data, we were able to create a dataset that provided a representative sample of the data and helped to improve the performance of our machine learning models.

5.1 Data Source and Collection

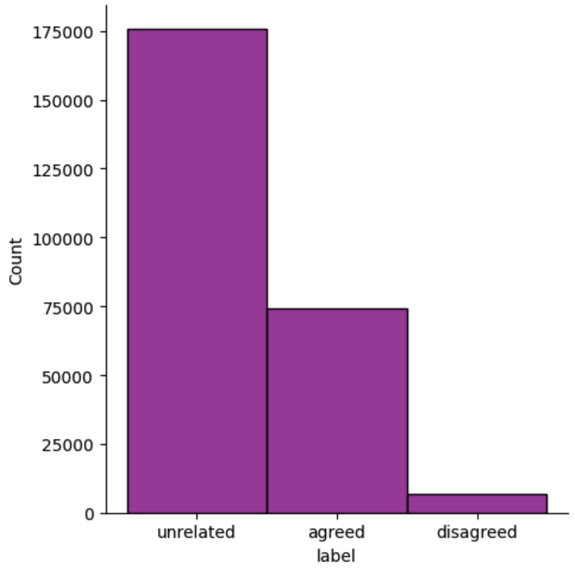
The data used for this project is given by the professor. The training and testing data given separately, and the label features are visualized for training data mentioned below-

```
[12] data_per_class = Counter(training_data['label'])
     print("Records as per the label feature: ")
     for i, label in enumerate(data_per_class):
         print(f"{label} : {data_per_class[label]} ({data_per_class[label] / training_data.shape[0]*100:.3f})%")
```

```
Records as per the label feature:
unrelated : 175598 (68.475)%
agreed : 74238 (28.949)%
disagreed : 6606 (2.576)%
```

```
sns.displot(training_data, x='label',color='purple')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdba51d6b50>
```

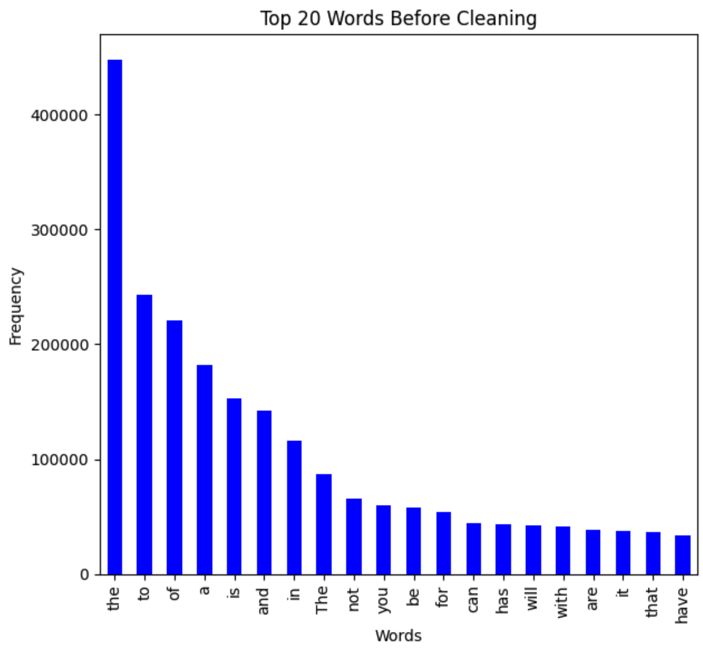


As we can see from the data distribution , the classes are quite uneven, making it exceedingly unlikely that any classification strategy will be effective or generalize effectively.

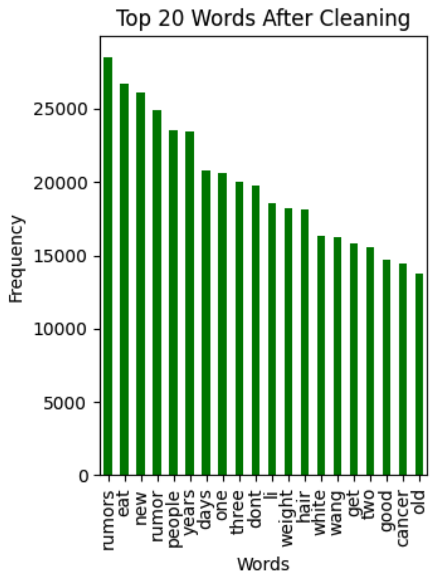
5.2 Data Cleaning

Data cleaning is an essential step in any machine learning-based classification task, as the quality of the input data directly affects the performance of the models. In this report, we describe our data cleaning process for a fake news classification task and discuss the impact of the data cleaning on the model performance. The data was cleaned to remove any irrelevant or duplicate data. We further cleaned the data by removing the stop words, special characters, numbers, and punctuation. After performing the data cleaning steps, we analyzed the cleaned dataset and found that the noise levels were significantly reduced.
The below mentioned bar chart shows the Top 20 most frequent words in the text data before and after the data cleaning.

```python
# Calculate word frequencies before cleaning
word_freq_before = pd.Series(' '.join(training_data['title1_en']+" "+training_data['title2_en']).split()).value_counts()
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
word_freq_before[:20].plot(kind='bar', color='blue')
plt.title('Top 20 Words Before Cleaning')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

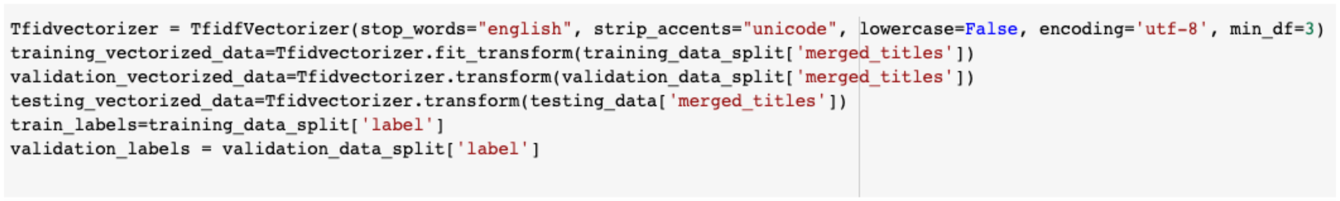


```python
# Calculate word frequencies after cleaning
word_freq_after = pd.Series(' '.join(training_data['merged_titles']).split()).value_counts()

plt.subplot(1,2,2)
word_freq_after[:20].plot(kind='bar', color='green')
plt.title('Top 20 Words After Cleaning')
plt.xlabel('Words')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```

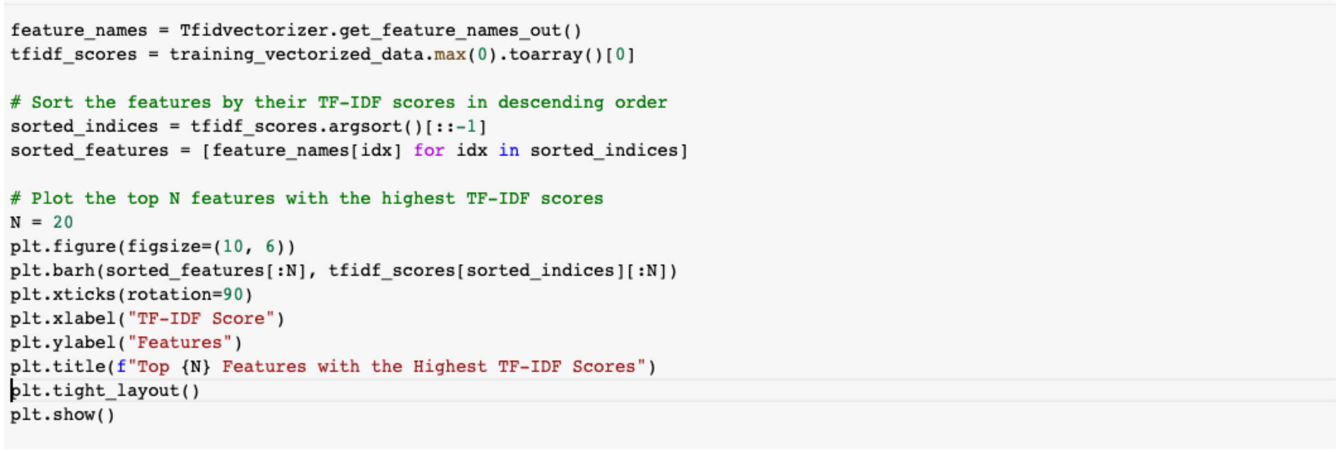


5.3 Data Transformation

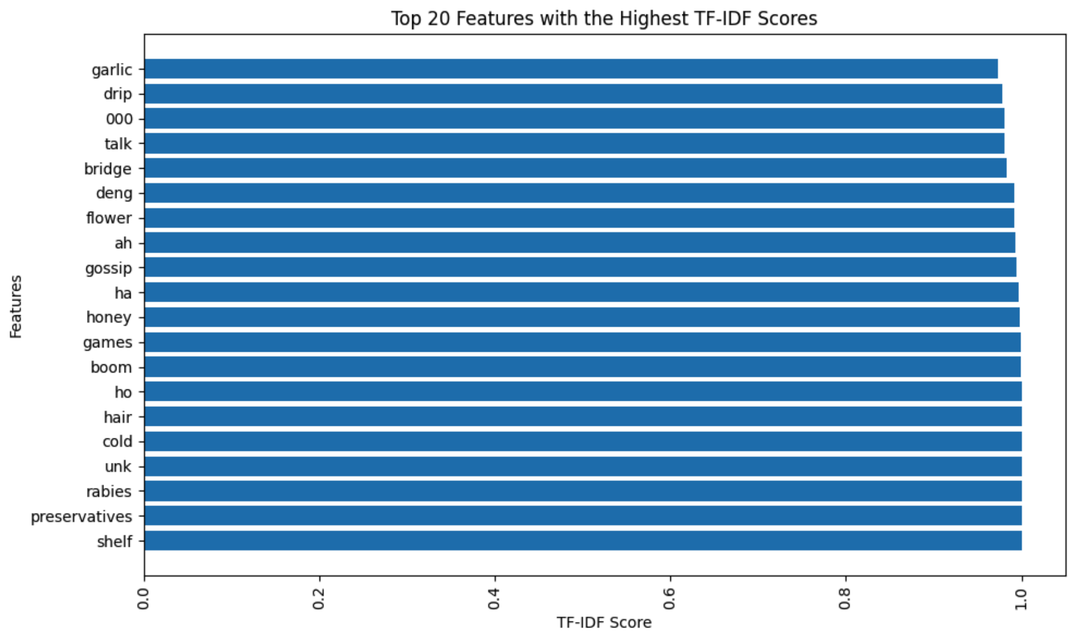
The text data was transformed by performing TF-IDF on the above datasets to transforms text to feature vectors that can be used as input to estimator vocabulary a dictionary that converts each token (word) to a feature index in the matrix, each unique token gets a feature index. It is used to determine the most important features or words that distinguish one category of documents from another. The TF-IDF score is assigned to each term in a document to measure of how important and relevant that term is to that document compared to the rest of the corpus. The below mentioned bar chart shows the top 30 features with the highest TF-IDF scores.



```
Tfidvectorizer = TfidfVectorizer(stop_words="english", strip_accents="unicode", lowercase=False, encoding='utf-8', min_df=3)
training_vectorized_data=Tfidvectorizer.fit_transform(training_data_split['merged_titles'])
validation_vectorized_data=Tfidvectorizer.transform(validation_data_split['merged_titles'])
testing_vectorized_data=Tfidvectorizer.transform(testing_data['merged_titles'])
train_labels=training_data_split['label']
validation_labels = validation_data_split['label']
```



```
feature_names = Tfidvectorizer.get_feature_names_out()
tfidf_scores = training_vectorized_data.max(0).toarray()[0]

# Sort the features by their TF-IDF scores in descending order
sorted_indices = tfidf_scores.argsort()[::-1]
sorted_features = [feature_names[idx] for idx in sorted_indices]

# Plot the top N features with the highest TF-IDF scores
N = 20
plt.figure(figsize=(10, 6))
plt.barh(sorted_features[:N], tfidf_scores[sorted_indices][:N])
plt.xticks(rotation=90)
plt.xlabel("TF-IDF Score")
plt.ylabel("Features")
plt.title(f"Top {N} Features with the Highest TF-IDF Scores")
plt.tight_layout()
plt.show()
```

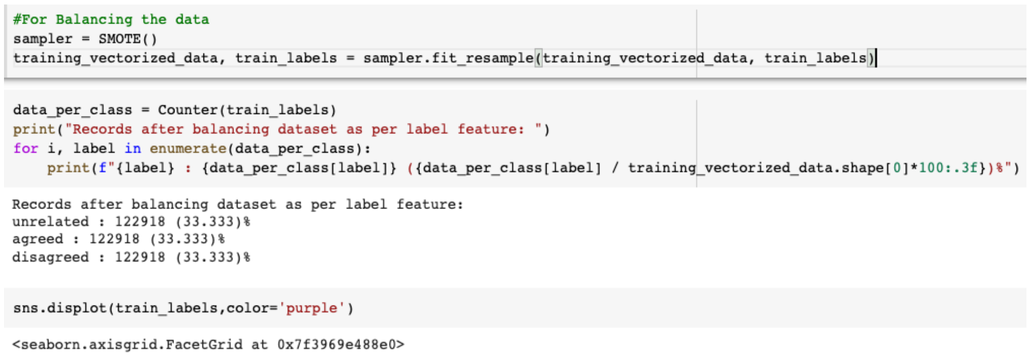


5.4 Data Augmentation

There are various methods to balance out the dataset. Most popular are –

- Resampling the existing data
- Generate new text similar to the existing text.

To augment the data, synthetic data was generated using the SMOTE algorithm to balance the class distribution in the dataset. These synthetic samples can be saved along with the original data to use in future analysis or modelling.



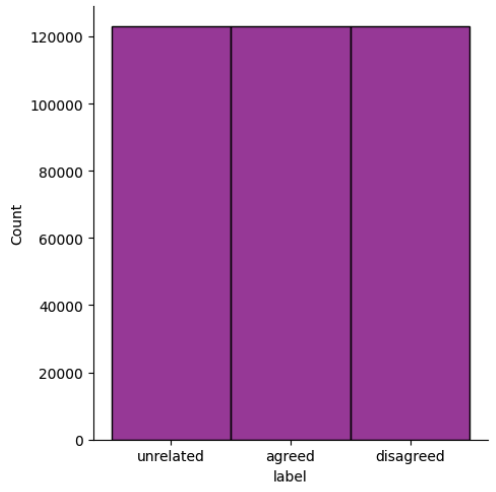
```
#For Balancing the data
sampler = SMOTE()
training_vectorized_data, train_labels = sampler.fit_resample(training_vectorized_data, train_labels)
```

```
data_per_class = Counter(train_labels)
print("Records after balancing dataset as per label feature: ")
for i, label in enumerate(data_per_class):
    print(f"{label} : {data_per_class[label]} ({data_per_class[label] / training_vectorized_data.shape[0]*100:.3f})%")
```

```
Records after balancing dataset as per label feature:
unrelated : 122918 (33.333)%
agreed : 122918 (33.333)%
disagreed : 122918 (33.333)%
```

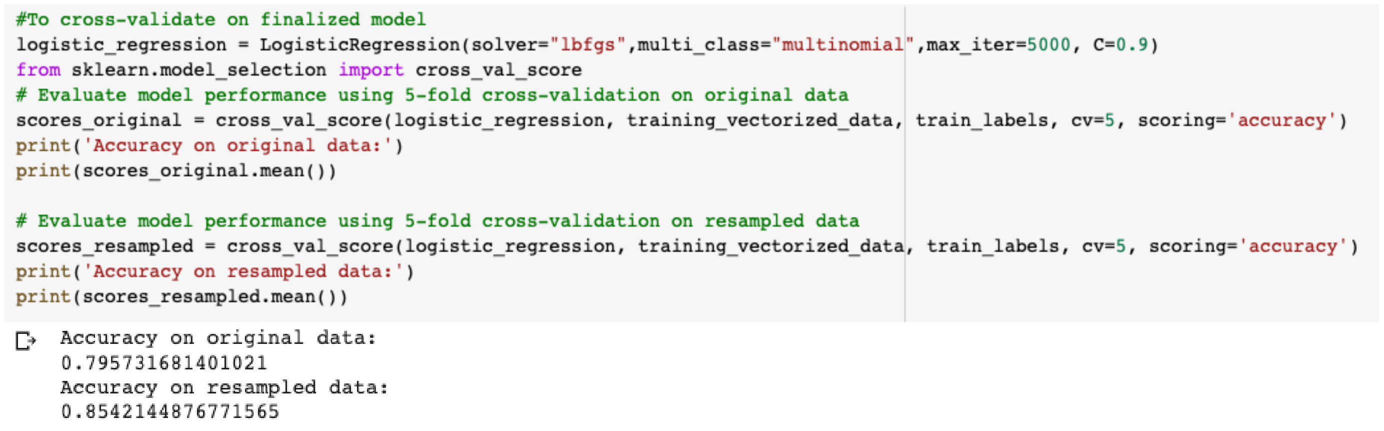
```
sns.displot(train_labels,color='purple')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f3969e488e0>
```



5.5 Resampled Data Evaluation

By comparing the performance of the model on both the original and resampled datasets using cross-validation, we can determine whether oversampling the minority class using SMOTE has improved the model's performance on imbalanced datasets. If the accuracy on the resampled data is significantly higher than that on the original data, it suggests that oversampling has improved the model's ability to predict the minority class. We use the **cross_val_score()** function from scikit-learn to evaluate the performance of the logistic regression model using 5-fold cross-validation. We pass the model, the features, the target variable, the number of folds (cv=5), and the scoring metric ('accuracy') as arguments. We then print the mean accuracy score of the model on both the original and resampled datasets.



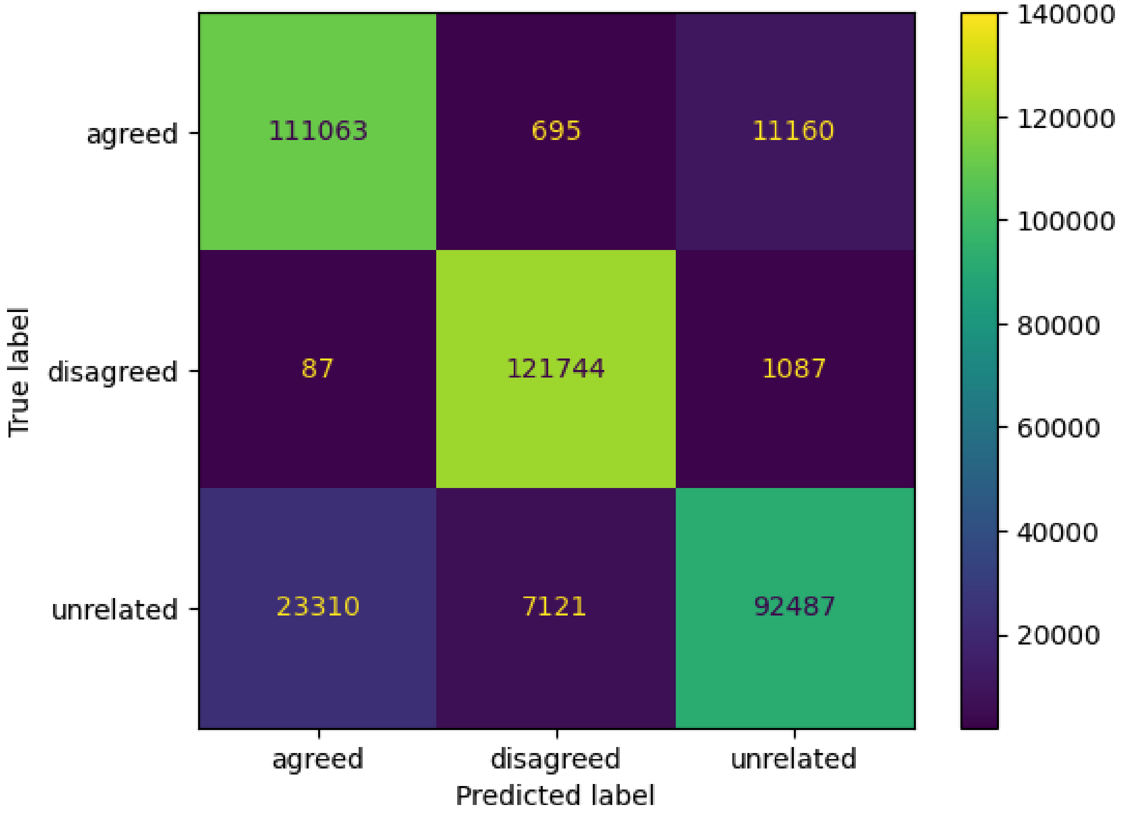
```
#To cross-validate on finalized model
logistic_regression = LogisticRegression(solver="lbfgs",multi_class="multinomial",max_iter=5000, C=0.9)
from sklearn.model_selection import cross_val_score
# Evaluate model performance using 5-fold cross-validation on original data
scores_original = cross_val_score(logistic_regression, training_vectorized_data, train_labels, cv=5, scoring='accuracy')
print('Accuracy on original data:')
print(scores_original.mean())

# Evaluate model performance using 5-fold cross-validation on resampled data
scores_resampled = cross_val_score(logistic_regression, training_vectorized_data, train_labels, cv=5, scoring='accuracy')
print('Accuracy on resampled data:')
print(scores_resampled.mean())
```

```
Accuracy on original data:
0.795731681401021
Accuracy on resampled data:
0.8542144876771565
```

VI. DATA MODELLING

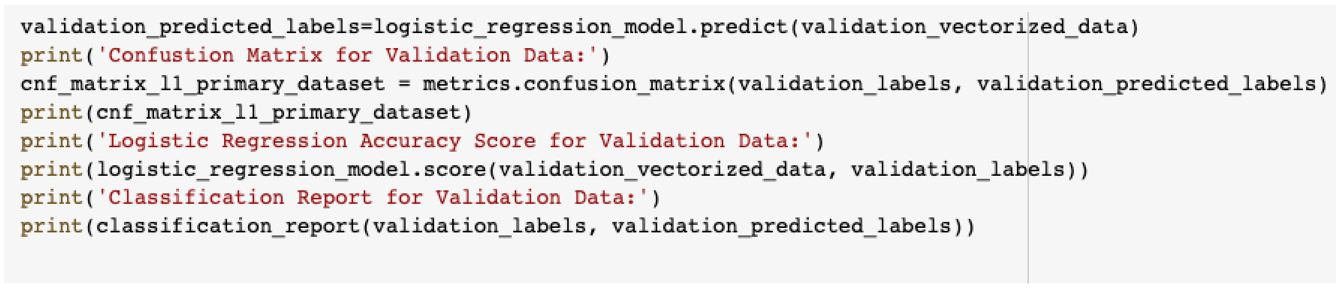
Data Modeling in software engineering is the process of simplifying the diagram or data model of a software system by applying certain formal techniques. Here, we used simple Logistic Regression model built with title1_en and title2_en combined. These combined titles are vectorized using the TF-IDF vectorizer as shown in above steps and passed to the logistic regression model. The training data is split into training and validation data into 7:3 ratio. The model is trained on 70% of the preprocessed data and tested on the remaining 30%. The model was trained using the “ **LogisticRegression**” function where the hyperparameters classifier is “**Multinomial**” from the python “Sklearn” Library.

```
Logistic Regression Accuracy Score:
0.8821436513231042
```

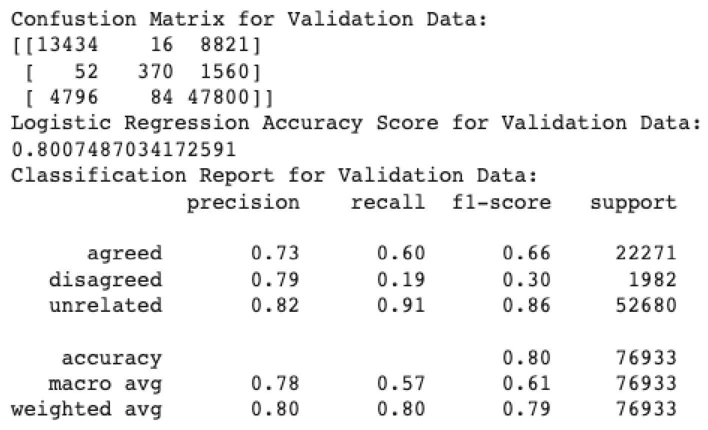


VII. MODEL EVALUATION

The validation data used for the model consisted 30% of the training data before training the model. The validation data is preprocessed in same way as the training and testing data. The logistic model achieved the accuracy of 80% on validation data. The classification and confusion matrix for validation data is mentioned below-



```
validation_predicted_labels=logistic_regression_model.predict(validation_vectorized_data)
print('Confustion Matrix for Validation Data:')
cnf_matrix_l1_primary_dataset = metrics.confusion_matrix(validation_labels, validation_predicted_labels)
print(cnf_matrix_l1_primary_dataset)
print('Logistic Regression Accuracy Score for Validation Data:')
print(logistic_regression_model.score(validation_vectorized_data, validation_labels))
print('Classification Report for Validation Data:')
print(classification_report(validation_labels, validation_predicted_labels))
```



```
Confustion Matrix for Validation Data:
[[13434    16  8821]
 [   52   370  1560]
 [ 4796    84 47800]]
Logistic Regression Accuracy Score for Validation Data:
0.8007487034172591
Classification Report for Validation Data:
              precision    recall  f1-score   support

      agreed       0.73      0.60      0.66     22271
   disagreed       0.79      0.19      0.30      1982
   unrelated       0.82      0.91      0.86     52680

    accuracy                           0.80     76933
   macro avg       0.78      0.57      0.61     76933
weighted avg       0.80      0.80      0.79     76933
```

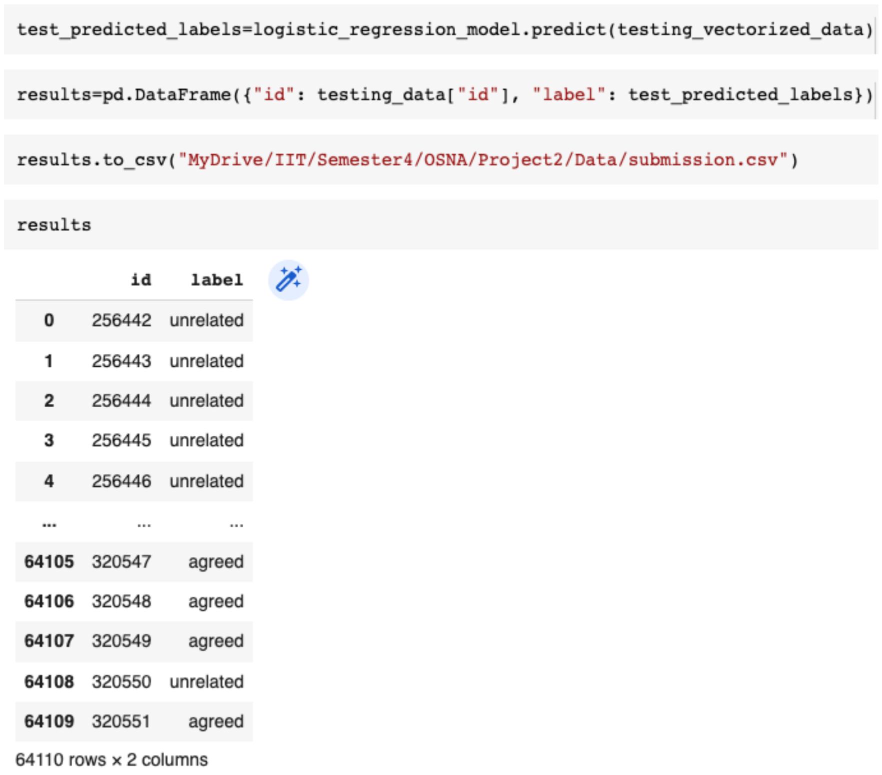
VIII. OBSERVATIONS AND CONCLUSION

In this report, we presented our work on fake news classification using logistic regression and discussed potential future directions. Our results showed that the logistic regression model could effectively distinguish fake news from real news and identified important features for the classification task. The accuracy achieved on training data is 88% and predicted results on test data which is stored in “**submission.csv**” file using ”**to_csv**” function.

Here are our observations:

We observed that it is a simple and easy-to-understand algorithm that is widely used for classification tasks. It can be implemented relatively easily and quickly, making it a good choice for smaller datasets. It is less prone to overfitting on small datasets than more complex models and can still produce good results. Logistic regression is computationally efficient, meaning that it can be trained quickly even on large datasets. This makes it a good choice for real-time applications where fast predictions are required. Here, the training time for given dataset is 88.5710 seconds. However, it is important to carefully consider the potential issues and limitations of the method. Firstly, Fake news classification can be a subjective and biased task, as what one person considers to be fake news may differ from another person's opinion. This can make it challenging to develop a model that accurately captures the true nature of fake news. Careful labeling and annotation of the data, and considering the perspectives of multiple annotators, can help to mitigate this issue. Secondly, Fake news classification may involve complex, non-linear relationships between the input features and the outcome variable.

Logistic regression assumes a linear relationship and may not be able to capture these non-linear relationships. More flexible models such as decision trees or neural networks may be better suited to this task.



```
test_predicted_labels=logistic_regression_model.predict(testing_vectorized_data)
```

```
results=pd.DataFrame({"id": testing_data["id"], "label": test_predicted_labels})
```

```
results.to_csv("MyDrive/IIT/Semester4/OSNA/Project2/Data/submission.csv")
```

```
results
```

```
            id      label
    0   256442  unrelated
    1   256443  unrelated
    2   256444  unrelated
    3   256445  unrelated
    4   256446  unrelated
  ...      ...        ...
64105   320547     agreed
64106   320548     agreed
64107   320549     agreed
64108   320550  unrelated
64109   320551     agreed
64110 rows × 2 columns
```

IX. FUTURE WORKS

Although our logistic regression model performed well, there are several potential areas for future work. First, we could investigate the impact of different preprocessing techniques on the model performance, such as using word embeddings instead of TF-IDF. Second, we could explore more advanced machine learning models, such as neural networks or ensemble methods, and compare their performance with logistic regression. Third, we could extend the dataset to include news articles from different languages and domains, and evaluate the model's cross-lingual and cross-domain generalizability. Fourth, we could address the issue of class imbalance in the dataset, as there were more real news articles than fake news articles. Finally, we could deploy the model as a web-based application or a browser extension, which could help users verify the authenticity of news articles they encounter online.

X. REFERENCES

[1] Fake news websites. (n.d.) Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Fake_news_website. Accessed Feb. 6, 2017.

[2] Naive Bayes classifier. (n.d.) Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_classifier. Accessed Feb. 6, 2017.

[3] Kai Shu, Suhang Wang, Huan Liu, “Understanding User Profiles on Social Media for Fake News Detection,” 2018, MIPR.

[4] Conroy, N. J., Rubin, V. L., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. Proceedings of the Association for Information Science and Technology, 52(1), 1-4.

[5] Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., & Choi, Y. (2017). Truth of varying shades: Analyzing language in fake news and political fact-checking. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 2931-2937).

Project Link:

https://github.com/Jasleen-bots21/CSP_579_Fake_News_Classification