FAKE NEWS DETECTION USING NLP

PHASE 4

COMBINING DATA

- Before merging our two separate datasets containing true and fake news, we introduced a unifying column called "news_authenticity" in both data frames. This new column serves as an indicator, clearly marking whether each piece of news is classified as true or fake. By creating this consistent label across the datasets, we're now prepared to combine and analyze the data as a single, cohesive unit for subsequent feature extraction and analysis. This step ensures that we can effectively distinguish the authenticity of news articles throughout the combined dataset.

```
true_data['news_authenticity']=1
fake_data['news_authenticity']=0
```

```
fkn_dataset=pd.concat([fake_data, true_data], ignore_index=True)
```

FEATURE EXTRACTION

- Feature extraction is a crucial step in natural language processing (NLP) when working with text data. It involves converting text data into numerical features or representations that machine learning models can use for analysis and prediction.
- TF-IDF (Term Frequency-Inverse Document Frequency):
- Term Frequency (TF): This part of TF-IDF measures how frequently a term (word) occurs in a document. It is calculated by counting the number of times a term appears in a document. The idea is that more frequent terms are often more important.
- Inverse Document Frequency (IDF): This part of TF-IDF measures how unique or rare a term is across a collection of documents (a corpus). It is calculated by dividing the total number of documents by the number of documents containing the term.

```
from sklearn.feature_extraction.text import TfidfVectorizer
fkn_dataset['news1'] = fkn_dataset['news'].apply(lambda tokens: ' '.join(tokens))
tfidf_vectorizer = TfidfVectorizer(max_features=500)
X_tfidf = tfidf_vectorizer.fit_transform(fkn_dataset['news1'])
```

SPLITTING DATASET

- In this phase, I divided the entire dataset into two subsets, one for training and the other for testing. The data was further split into an 80% portion designated for training and a 20% portion allocated for testing.

```
from sklearn.model_selection import train_test_split
x=X_tfidf
y=fkn_dataset['news_authenticity']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

MODEL TRAINING

LOGISTIC REGRESSION

Logistic regression is a suitable choice for this project due to its simplicity, interpretability, and effectiveness in binary classification tasks. It is a linear model that works well when there is a clear separation between classes, making it particularly useful in distinguishing between genuine and fake news.

```
from sklearn.linear_model import LogisticRegression
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
```

RANDOM FOREST

Random Forest, on the other hand, is a versatile ensemble learning algorithm known for its robustness and ability to handle complex datasets. In the context of fake news detection, it excels at capturing non-linear relationships and interactions among features.

```
from sklearn.ensemble import RandomForestClassifier
random_forest_model = RandomForestClassifier()
random_forest_model.fit(X_train, y_train)
```

MODEL EVALUATION

Logistic regression

REPORT

```
#testing LR
logistic_predictions = logistic_model.predict(X_test)
```

```
#report
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
print(classification_report(y_test, logistic_predictions))
```

	precision	recall	f1-score	support
0	0.99	0.98	0.99	4723
1	0.98	0.98	0.98	4255
accuracy			0.98	8978
macro avg	0.98	0.98	0.98	8978
weighted avg	0.98	0.98	0.98	8978

Random forest

REPORT

```
random_forest_predictions = random_forest_model.predict(X_test)
```

```
print(classification_report(y_test, random_forest_predictions))
```

The Random Forest technique consistently outperforms Logistic Regression, delivering better results.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	4723
1	1.00	1.00	1.00	4255
accuracy			1.00	8978
macro avg	1.00	1.00	1.00	8978
weighted avg	1.00	1.00	1.00	8978

TESTING

```
In [11]: y_test[:10]

Out[11]: 13735    0
         34895    1
         18405    0
         18285    0
         5199     0
         11906    0
         1784     0
         4506     0
         35626    1
         18156    0
         Name: news_authenticity, dtype: int64

In [35]: logistic_predictions[:10]

Out[35]: array([0, 1, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)

In [12]: random_forest_predictions[:10]

Out[12]: array([0, 1, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```