Model Development Phase Template

Date	1 November 2024
Team ID	SWTID1726834817
Project Title	Fake News Analysis in social media using NLP
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

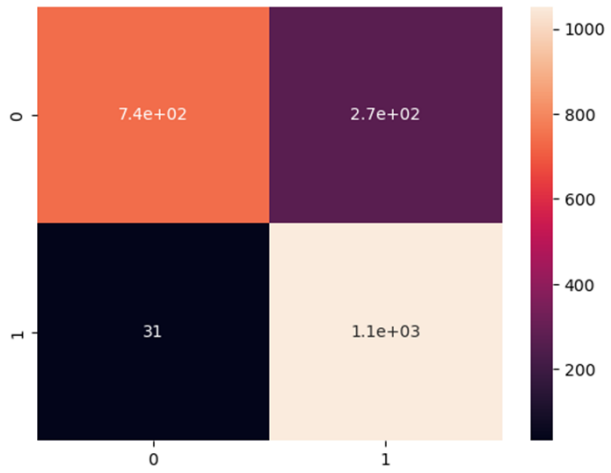
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import pickle
```

```
df = pd.read_csv('news.csv')  # Replace with your dataset path

X = df['text']  # Features (text data)
y = df['label']  # Labels (FAKE or REAL)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=53)

count_vectorizer = CountVectorizer(stop_words='english')
count_train = count_vectorizer.fit_transform(X_train)  # Fit and transform
training data
count_test = count_vectorizer.transform(X_test)  # Transform test data

tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
tfidf_train = tfidf_vectorizer.fit_transform(X_train)  # Fit and transform
training data
tfidf_test = tfidf_vectorizer.transform(X_test)  # Transform test data

nb_classifier = MultinomialNB()
nb_classifier.fit(count_train, y_train)  # Fit the model
pred = nb_classifier.predict(count_test)  # Predict on test data

nb_classifier.fit(tfidf_train, y_train)  # Fit the model
pred = nb_classifier.predict(tfidf_test)  # Predict on test data

score = accuracy_score(y_test, pred)  # Calculate accuracy
print(score)
```

```
0.8565279770444764
```

```
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])  # Confusion
matrix
sns.heatmap(cm, annot=True)  # Visualizing the confusion matrix
```

```
<Axes: >
```




```
alphas = np.arange(0, 1, 0.1)
for alpha in alphas:
    nb_classifier = MultinomialNB(alpha=alpha)  # Create model with different
alpha
    nb_classifier.fit(tfidf_train, y_train)  # Fit the model
    pred = nb_classifier.predict(tfidf_test)  # Predict
    score = accuracy_score(y_test, pred)  # Calculate accuracy
    print('Alpha:', alpha, 'Score:', score)  # Print results

C:\Users\DELL\anaconda3\Lib\site-packages\sklearn\naive_bayes.py:890:
RuntimeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(

Alpha: 0.0 Score: 0.6150167384026781
Alpha: 0.1 Score: 0.8976566236250598
Alpha: 0.2 Score: 0.8938307030129125
Alpha: 0.30000000000000004 Score: 0.8900047824007652
Alpha: 0.4 Score: 0.8857006217120995
Alpha: 0.5 Score: 0.8842659014825442
Alpha: 0.6000000000000001 Score: 0.874701099952176
Alpha: 0.7000000000000001 Score: 0.8703969392635102
Alpha: 0.8 Score: 0.8660927785748446
Alpha: 0.9 Score: 0.8589191774270684

with open('fake_news_model.pkl', 'wb') as handle:
    pickle.dump(nb_classifier, handle)  # Save the trained model

with open('vectorizer.pkl', 'wb') as handle:
    pickle.dump(tfidf_vectorizer, handle)  # Save the vectorizer
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix

Random Forest	`# Display the confusion matrix`
	`rf_cm = confusion_matrix(y_test, rf_pred, labels=['FAKE', 'REAL'])`
	`print("Confusion Matrix:\n", rf_cm)`
	
	`# Calculate the accuracy score`
	`rf_score = accuracy_score(y_test, rf_pred)`
	`print("Random Forest Model Accuracy:", rf_score)`
		Accuracy: 0.9076896652319465	Confusion Matrix:
			[[931 77]
			[116 967]]

Decision Tree	```dt_score = accuracy_score(y_test, dt_pred)
	print("Decision Tree Model Accuracy:", dt_score)
	
	# Display the confusion matrix
	dt_cm = confusion_matrix(y_test, dt_pred, labels=['FAKE', 'REAL'])
	print("Confusion Matrix:\n", dt_cm)```
		Accuracy: 0.8106169296987088	Confusion Matrix:
			[[810 198]
			[198 885]]

KNN	```knn_score = accuracy_score(y_test, knn_pred)
	print("KNN Model Accuracy:", knn_score)
	
	knn_param_grid = {
	'n_neighbors': [3, 5, 7, 9, 11],
	'weights': ['uniform', 'distance'],
	'metric': ['euclidean', 'manhattan']
	}
	
	# Initialize the KNN classifier
	knn_classifier = KNeighborsClassifier()
	
	# Display the confusion matrix
	knn_cm = confusion_matrix(y_test, knn_pred, labels=['FAKE', 'REAL'])
	print("Confusion Matrix:\n", knn_cm)```
		Accuracy: 0.5442372070779531	Confusion Matrix:
			[[1006 2]
			[951 132]]

Gradient Boosting	```# Display the confusion matrix
	gb_cm = confusion_matrix(y_test, gb_pred, labels=['FAKE', 'REAL'])
	print("Confusion Matrix:\n", gb_cm)
	
	# Calculate the accuracy score
	gb_score = accuracy_score(y_test, gb_pred)
	print("Gradient Boosting Model Accuracy:", gb_score)```
		Accuracy: 0.8919177427068389	Confusion Matrix:
			[[927 81]
			[145 938]]