Project Title:

Building a Smarter AI-Powered Spam Classifier

Phase 4 : Development Part 2

Overview:

In this stage, we refine our spam classifier by converting text data into numerical features using TF-IDF. We select the Multinomial Naive Bayes algorithm for model training and then evaluate its performance with accuracy, precision, recall, and F1 score. This step is crucial for improving the classifier's ability to distinguish between spam and non-spam messages.

Feature Engineering:

- ✓ Feature engineering is the process of creating new features or modifying existing ones in a dataset to improve the performance of machine learning models.
- ✓ It involves transforming raw data into a format that the model can understand and use effectively.

Model Training:

- ✓ Model training refers to the process of teaching a machine learning algorithm to make predictions or classifications based on input data.
- ✓ During training, the model learns patterns and relationships within the data.

Evaluation Metrics:

- ✓ Evaluation metrics are quantitative measures used to assess the performance of a model.
- ✓ They provide insights into how well a model is performing by measuring aspects such as accuracy, precision, recall, and F1 score.

Naive Bayes Algorithm:

- ✓ The Naive Bayes algorithm is a probabilistic machine learning algorithm based on Bayes' theorem.
- ✓ It's commonly used for classification tasks, making predictions based on the probability of different outcomes.
- ✓ The "naive" part refers to the simplifying assumption of feature independence.

TF-IDF Vectorizer:

- ✓ The Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer is a technique used to convert text data into numerical vectors.
- ✓ It assigns weights to terms in a document based on their frequency and importance in the context of a corpus.

Program:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score


# Load the preprocessed data from Phase 3

data=pd.read_csv('C:\\Users\\BYAMUNA1\\Desktop\\preprocessed_sms_data.
csv', encoding='latin-1')


# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(data['v2'], data['v1'],
test_size=0.2, random_state=42)


# Print Training and Testing Set Shapes

print(f"Training set shape: {X_train.shape}, {y_train.shape}")

print(f"Testing set shape: {X_test.shape}, {y_test.shape}")


# Create a TF-IDF vectorizer to convert text data to numerical features

tfidf_vectorizer = TfidfVectorizer()

X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

```
# Print TF-IDF Vectorizer Information
print("TF-IDF Vectorizer Information:")
print(f"Vocabulary size: {len(tfidf_vectorizer.vocabulary_)}")
print(f"Number of terms: {X_train_tfidf.shape[1]}")

# Choose a machine learning algorithm (e.g., Multinomial Naive Bayes)
model = MultinomialNB()

# Train the model on the training data
model.fit(X_train_tfidf, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test_tfidf)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='spam')
recall = recall_score(y_test, y_pred, pos_label='spam')
f1 = f1_score(y_test, y_pred, pos_label='spam')

# Print Evaluation Metrics
print("\nEvaluation Metrics:")
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

Program Explanation:

1**. Importing Libraries:** The program starts by importing the necessary libraries: pandas for data manipulation, scikit-learn for machine learning tools, and various modules for specific tasks like text vectorization and performance evaluation.

2**. Loading Preprocessed Data:** It loads the preprocessed data from Phase 3 using pd.read_csv. The encoding='latin-1' parameter is specified to handle any special characters in the text data.

3. **Data Splitting:** The program uses train_test_split to split the data into training and testing sets. This is essential for evaluating the machine learning model's performance. The test_size parameter controls the split ratio.

4. **TF-IDF Vectorization:** A TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is created using TfidfVectorizer. This vectorizer converts the text data into numerical features while giving more weight to important words. It is fitted on the training data and then used to transform both the training and testing data.

5. **Model Selection:** The program selects the machine learning algorithm for the task, in this case, a Multinomial Naive Bayes classifier (MultinomialNB).

6. Model Training: The chosen model is trained using the training data with model.fit(X_train_tfidf, y_train).

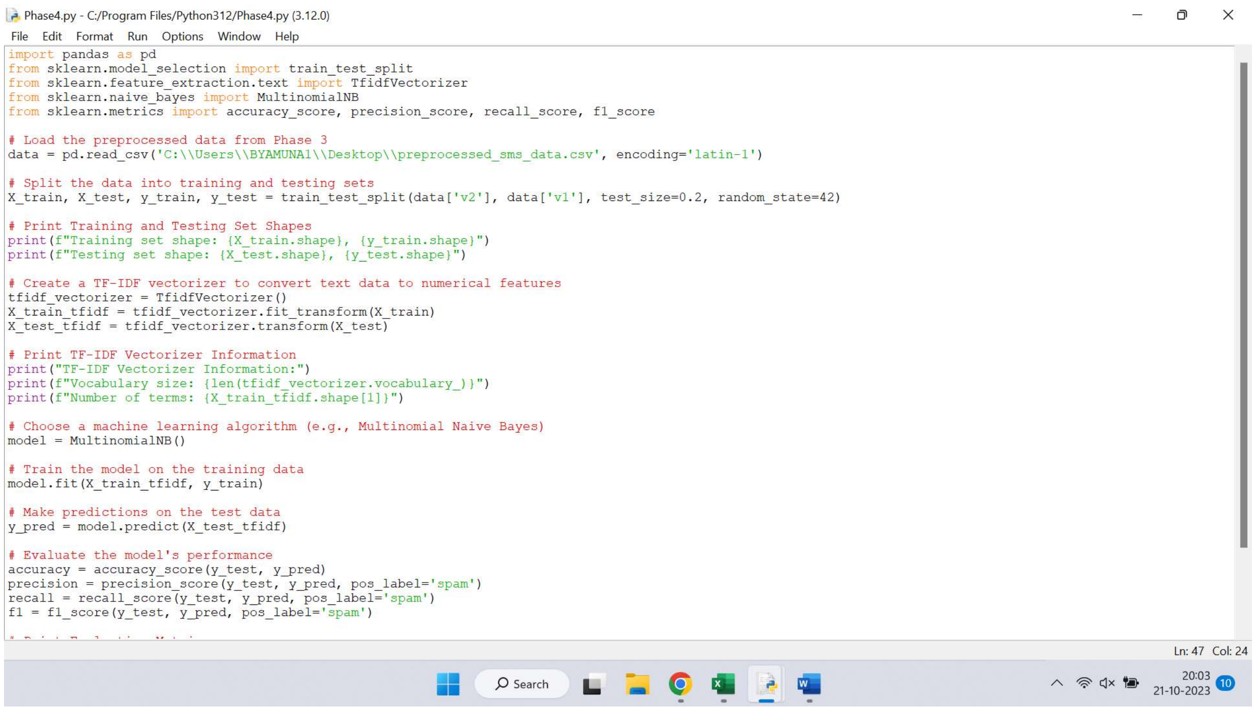
7. **Making Predictions:** The trained model is used to make predictions on the test data with model.predict(X_test_tfidf).

8**. Performance Evaluation:** The program calculates various performance metrics to assess the model's effectiveness:

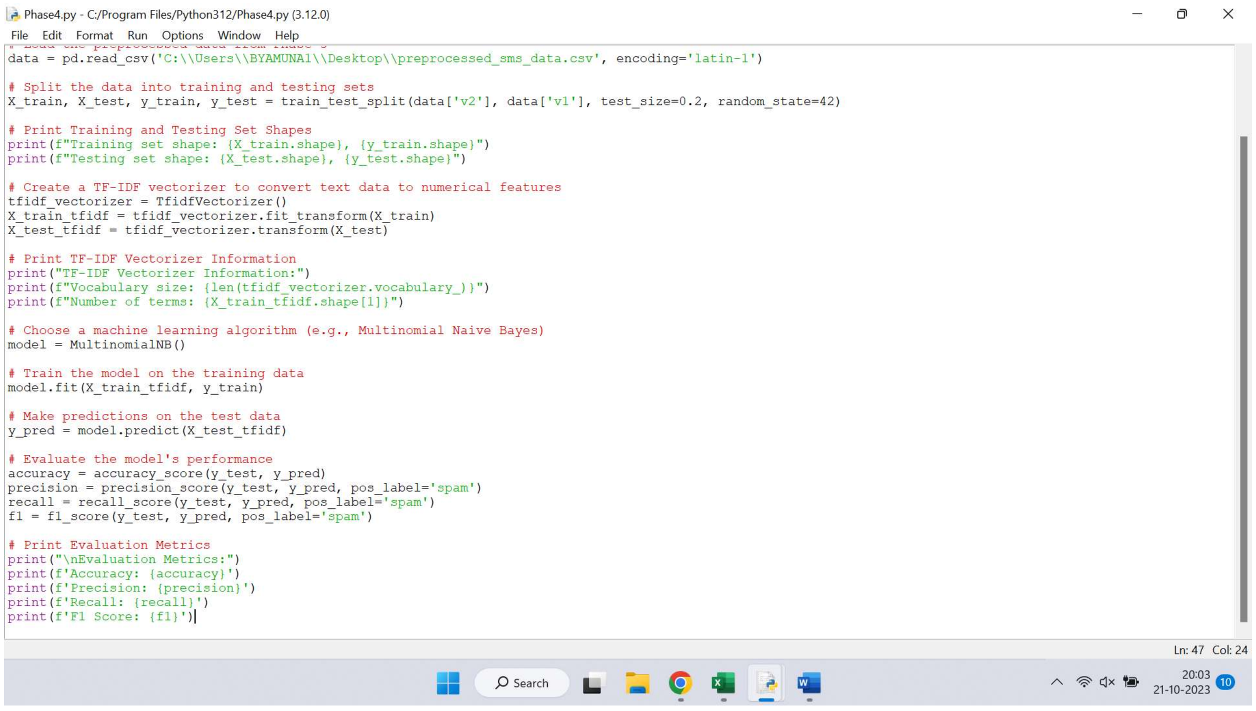
- Accuracy: The proportion of correctly classified instances.
- Precision: The ability of the model not to label as positive a sample that is negative.
- Recall: The ability of the model to find all the positive samples.
- F1 Score: The harmonic mean of precision and recall, which balances the two.

9. **Printing Evaluation Metrics:** The program prints the values of accuracy, precision, recall, and F1 score to assess the model's performance.

Program:


Phase4.py - C:/Program Files/Python312/Phase4.py (3.12.0)
File Edit Format Run Options Window Help
Ln: 47 Col: 24
Search
20:03
21-10-2023

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Load the preprocessed data from Phase 3
data = pd.read_csv('C:\\Users\\BYAMUNA1\\Desktop\\preprocessed_sms_data.csv', encoding='latin-1')

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data['v2'], data['v1'], test_size=0.2, random_state=42)

# Print Training and Testing Set Shapes
print(f"Training set shape: {X_train.shape}, {y_train.shape}")
print(f"Testing set shape: {X_test.shape}, {y_test.shape}")

# Create a TF-IDF vectorizer to convert text data to numerical features
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Print TF-IDF Vectorizer Information
print("TF-IDF Vectorizer Information:")
print(f"Vocabulary size: {len(tfidf_vectorizer.vocabulary_)}")
print(f"Number of terms: {X_train_tfidf.shape[1]}")

# Choose a machine learning algorithm (e.g., Multinomial Naive Bayes)
model = MultinomialNB()

# Train the model on the training data
model.fit(X_train_tfidf, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test_tfidf)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='spam')
recall = recall_score(y_test, y_pred, pos_label='spam')
f1 = f1_score(y_test, y_pred, pos_label='spam')
```


Phase4.py - C:/Program Files/Python312/Phase4.py (3.12.0)
File Edit Format Run Options Window Help
Ln: 47 Col: 24
Search
20:03
21-10-2023

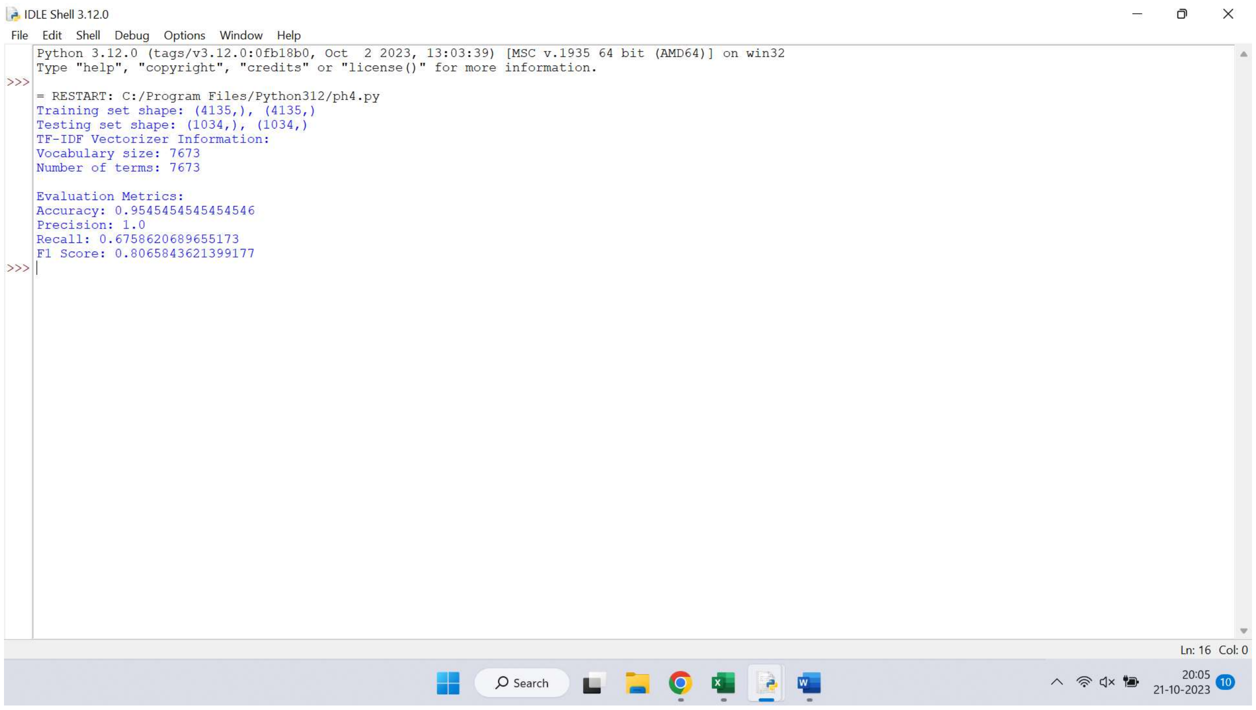
```python
data = pd.read_csv('C:\\Users\\BYAMUNA1\\Desktop\\preprocessed_sms_data.csv', encoding='latin-1')

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data['v2'], data['v1'], test_size=0.2, random_state=42)

# Print Training and Testing Set Shapes
print(f"Training set shape: {X_train.shape}, {y_train.shape}")
print(f"Testing set shape: {X_test.shape}, {y_test.shape}")

# Create a TF-IDF vectorizer to convert text data to numerical features
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Print TF-IDF Vectorizer Information
print("TF-IDF Vectorizer Information:")
print(f"Vocabulary size: {len(tfidf_vectorizer.vocabulary_)}")
print(f"Number of terms: {X_train_tfidf.shape[1]}")

# Choose a machine learning algorithm (e.g., Multinomial Naive Bayes)
model = MultinomialNB()

# Train the model on the training data
model.fit(X_train_tfidf, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test_tfidf)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='spam')
recall = recall_score(y_test, y_pred, pos_label='spam')
f1 = f1_score(y_test, y_pred, pos_label='spam')
```

```python
# Print Evaluation Metrics
print("\nEvaluation Metrics:")
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

Output:

```
Training set shape: (4135,), (4135,)
Testing set shape: (1034,), (1034,)
TF-IDF Vectorizer Information:
Vocabulary size: 7673
Number of terms: 7673

Evaluation Metrics:
Accuracy: 0.9545454545454546
Precision: 1.0
Recall: 0.6758620689655173
F1 Score: 0.8065843621399177
```

Output:



```
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Program Files/Python312/ph4.py
Training set shape: (4135,), (4135,)
Testing set shape: (1034,), (1034,)
TF-IDF Vectorizer Information:
Vocabulary size: 7673
Number of terms: 7673

Evaluation Metrics:
Accuracy: 0.9545454545454546
Precision: 1.0
Recall: 0.6758620689655173
F1 Score: 0.8065843621399177
>>>
```

Conclusion:

During this stage, we took the next steps in building our spam classifier. We selected a machine learning algorithm, trained it using preprocessed SMS data, and carefully assessed its performance. This involved essential procedures like data splitting, TF-IDF vectorization, and utilizing a Multinomial Naive Bayes model. By completing this part, we've made significant progress in creating a dependable AI-powered spam classifier, improving communication security by distinguishing between legitimate and spam messages.