Phase 5: Project Documentation & Submission

Documentation

Problem Statement:

The project aims to build a spam classifier for SMS messages using a machine learning approach. The primary objective is to create a model that can accurately distinguish between spam and non-spam (ham) messages, thus helping users filter out unwanted content.

Design Thinking Process:

1. **Understanding the Problem**: We started by understanding the problem of SMS spam and its impact on users. The goal was to develop a solution to reduce the annoyance caused by spam messages.

2. **Data Collection**: We obtained a labeled dataset of SMS messages containing both spam and non-spam messages from Kaggle.

3. **Data Preprocessing**: We performed data preprocessing tasks such as converting labels to binary values, text cleaning, and splitting the dataset into training and testing sets.

4. **Feature Extraction**: We used CountVectorizer to convert the text messages into numerical features.

5. **Model Selection and Training**: We selected the Multinomial Naive Bayes algorithm for the initial model and trained it on the training data.

6. **Evaluation**: We evaluated the model's performance using metrics like accuracy, precision, recall, and the F1-score.

Dataset Description:

The dataset used for this project is sourced from Kaggle and contains a collection of SMS messages labeled as spam or non-spam (ham). It consists of two columns: 'label' and 'message'. The 'label' column contains binary labels (1 for spam and 0 for ham), while the 'message' column contains the text of the SMS messages.

Choice of Machine Learning Algorithm:

We chose the Multinomial Naive Bayes algorithm for the initial spam classifier due to its simplicity and effectiveness for text classification tasks. Other algorithms, such as Logistic Regression or Support Vector Machines, can also be explored for comparison.

Model Training and Evaluation Metrics:

The selected model was trained using the training dataset. We evaluated the model's performance using the following metrics:

- Accuracy: To measure the proportion of correctly classified messages.

- Precision: To assess the ratio of true positive predictions to the total positive predictions.

- Recall: To measure the ratio of true positive predictions to the total actual positive cases.

- F1-Score: To balance precision and recall, providing a single performance metric.

Innovative Techniques:

This project mainly focuses on traditional machine learning techniques for spam classification. However, innovative approaches could include using deep learning models, natural language processing techniques, or ensemble methods to improve performance.

Submission

To submit the project, follow these steps:

1. Compile all code files, including data preprocessing, model training, and evaluation steps.

2. Create a well-structured README file that includes the following information:

- Instructions on how to run the code.
- A list of dependencies and installation instructions.
- A brief project overview and the problem statement.
- Dataset source and description.
- Choice of machine learning algorithm and why it was selected.
- Results and evaluation metrics.
- Any additional notes or innovative techniques used.

3. Include the dataset source in your README and provide a brief description of the dataset.

4. Share your project on platforms like GitHub or a personal portfolio website for others to access and review.

By following these steps, you'll ensure that your project is well-documented and ready for submission, allowing others to understand and potentially build upon your work.

FULL PYTHON SCRIPT :

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix

# Load the dataset
df = pd.read_csv('spam.csv', encoding='latin-1')

# Data preprocessing
df['label'] = df['label'].map({'spam': 1, 'ham': 0})

# Split the dataset into training and testing sets
X = df['message']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=42)

# Feature extraction
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

```
# Train the model (Multinomial Naive Bayes)
clf = MultinomialNB()
clf.fit(X_train, y_train)

# Evaluate the model
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print('Classification Report:')
print(report)
print('Confusion Matrix:')
print(confusion)
```