Spam and Ham Filtering Project

Project Overview:

The Spam and Ham Filtering project aims to develop a machine learning model that can accurately classify incoming messages as either "spam" or "ham" (non-spam). By utilizing the Multinomial Naive Bayes algorithm and text processing techniques, this project focuses on creating an efficient and effective system for filtering unwanted and potentially harmful messages.

Project Steps:

Data Preprocessing:

The provided dataset is analyzed, which contains messages labeled as either "spam" or "ham." Descriptive statistics are generated to gain insights into the distribution of spam and ham messages.

Data Transformation:

A new column, "spam," is created in the dataset by replacing the "spam" and "ham" labels with numerical values (1 for "spam" and 0 for "ham"). The updated dataset is displayed to verify the changes.

Data Splitting:

The dataset is split into training and testing sets, with 75% used for training and 25% for evaluation. The messages are separated into input features (X) and the corresponding spam labels (y).

Feature Extraction:

The CountVectorizer is initialized to transform the text data into a matrix of token counts. The CountVectorizer is fit on the training data and transforms both the training and testing data.

Model Training:

A Multinomial Naive Bayes classifier is instantiated. The classifier is trained using the training data to learn patterns and relationships between the input features and the spam labels.

Spam and Ham Prediction:

New email messages are created and transformed into token count matrices using the fitted CountVectorizer. The trained model predicts the class labels (spam or ham) for the new email messages.

Model Evaluation:

The model's performance is evaluated using a classification report. The classification report provides precision, recall, F1-score, and support metrics for both the ham and spam classes.

The Spam and Ham Filtering project leverages the Multinomial Naive Bayes algorithm to accurately classify messages as spam or ham. By preprocessing the data, transforming text into numerical features, and training the model, it becomes possible to predict the spam or ham label for new messages. The evaluation metrics from the classification report help assess the model's effectiveness in distinguishing between spam and ham messages.

This project provides a foundation for developing a robust spam filtering system that can be implemented in various applications to reduce the impact of unwanted and potentially harmful messages on users' digital experiences.

```
import pandas as pd
import numpy as np
```

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

```
data = pd.read_csv("spam.csv")
data.head()
```

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
data.groupby('Category').describe()
```

	Message			
	count	unique	top	freq
Category				
ham	4825	4516	Sorry, I'll call later	30
spam	747	641	Please call our customer service representativ...	4

```
data['spam'] = data['Category'].replace({'spam':1, "ham": 0})
```

```
data.head()
```

	Category	Message	spam
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

```
X = data['Message'].values
y = data["spam"].values
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.25)
```

```
cv = CountVectorizer()
X_train = cv.fit_transform(X_train)
X_test = cv.transform(X_test)
```

```
X_train.toarray()
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
email = ["Hi Tien. It is great to meet you!"]
email = cv.transform(email)
```

```
model.predict(email)
```

```
array([0])
```

```
# 0 means ham
```

```
email1 = ["Do you want to get a free money.Click to it! reward reward"]
email1 = cv.transform(email1)
model.predict(email1)
```

```
array([1])
```

```
from sklearn.metrics import classification_report
```

```
y_pred = model.predict(X_test)
```

```
print(classification_report(y_pred, y_test))
```

```
                   precision    recall  f1-score   support

               0       1.00      0.98      0.99      1232
               1       0.87      0.98      0.92       161

        accuracy                           0.98      1393
       macro avg       0.93      0.98      0.95      1393
    weighted avg       0.98      0.98      0.98      1393
```

✓ 0s completed at 3:09 PM