/*Practical No : 05,
Name: Dhiraj Vijay Barwal ,
Roll No : 08,
Class: TECSD,
Batch: T1 */

Prediction of email spam or ham

```
# Import necessary libraries
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Sample data (replace this with your own email dataset)
data = {
    'email': [
        'Congratulations! You have won a prize.',
        'Hello, I would like to schedule a meeting.',
        'Limited time offer, claim your free gift now!',
        'Hi, just wanted to follow up on our last conversation.',
        'Exclusive deal just for you, sign up now!',
        'Reminder: Your appointment is tomorrow.',
    ],
    'label': ['spam', 'ham', 'spam', 'ham', 'spam', 'ham']  # spam or ham labels
}

# Convert the data into a pandas DataFrame
df = pd.DataFrame(data)

# Define the feature (email text) and target (label: spam or ham)
X = df['email']
y = df['label']

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Convert email text into numerical features using CountVectorizer (Bag of Words model)
vectorizer = CountVectorizer(stop_words='english')  # You can use TF-IDF as well (TfidfVectorizer)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Initialize the Naive Bayes classifier
model = MultinomialNB()

# Train the model with the training data
model.fit(X_train_vec, y_train)

# Predict the labels (spam/ham) for the test data
y_pred = model.predict(X_test_vec)

# Print the predicted values
print("Predicted labels: ", y_pred)

# Evaluate the model's performance
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
# Example: Predicting a new email
new_email = ["Free money now! Claim your prize!"]
new_email_vec = vectorizer.transform(new_email)
prediction = model.predict(new_email_vec)
print(f"The new email is predicted as: {prediction[0]}")
```

OUTPUT :

```
Predicted labels:  ['ham']
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

         ham       1.00      1.00      1.00         1
        spam       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

The new email is predicted as: spam