**SPAM DETECTION CODE USING PYTHON AND MACHINE LEARNING**

**Imports – Bringing in Libraries**

import seaborn as sns

import matplotlib.pyplot as plt

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 precision\_score, recall\_score, f1\_score

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

* ***pandas:*** Used to handle and process the dataset.
* ***seaborn + matplotlib.pyplot:*** Used to visualize data, especially the confusion matrix.
* ***CountVectorizer:*** Converts text into a format the model can understand (numeric vectors).
* ***MultinomialNB:*** A Naive Bayes model good for text classification.
* ***train\_test\_split:*** Splits the dataset into training and testing.
* ***accuracy\_score, precision\_score, recall\_score, etc.:*** Used to evaluate model performance.

**📁 Step 1: Load Dataset**

df = pd.read\_csv("spam\_ham\_dataset.csv")

print(df.head())

* Reads the CSV file into a DataFrame called ***df.***
* ***df.head()*** prints the first 5 rows to inspect structure.

**🧹Step 2: Data Cleaning / Preprocessing**

df = df[['label', 'text']]

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

* Selects only the ***label (spam/ham) and text (message)*** columns.
* Converts labels into binary:
  + ***ham → 0***
  + ***spam → 1***
* This is important because machine learning models work with numbers, not strings.

**✂️ Step 3: Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['text'], df['label'], test\_size=0.2, random\_state=42)

* Splits the dataset into training (80%) and testing (20%) sets.
* ***X\_train, X\_test:*** message text
* ***y\_train, y\_test:*** label (0 or 1)
* ***random\_state=42:*** ensures reproducibility (same result every run)

**🔢 Step 4: Text Vectorization**

vectorizer = CountVectorizer()

X\_train\_vec = vectorizer.fit\_transform(X\_train)

X\_test\_vec = vectorizer.transform(X\_test)

* ***CountVectorizer*** converts text into a matrix of token counts (bag-of-words).
* ***fit\_transform():*** Learns from and transforms the training data.
* ***transform():*** Transforms test data using the same learned vocabulary.

**🤖 Step 5: Train the Model**

model = MultinomialNB()

model.fit(X\_train\_vec, y\_train)

* Initializes a Naive Bayes classifier.
* Trains it using the vectorized messages and labels.

**🧠 Step 6: Make Prediction**

y\_pred = model.predict(X\_test\_vec)

* The trained model makes predictions on the test data.

**📊 Step 7: Evaluation**

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Precision:", precision\_score(y\_test, y\_pred))

print("Recall:", recall\_score(y\_test, y\_pred))

* **Accuracy**: Overall correct predictions
* **Precision**: Of predicted spam, how many were actually spam (good for reducing false positives)
* **Recall**: Of all actual spam, how many did we detect (good for reducing false negatives)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

* Full report including precision, recall, f1-score, and support for each class (ham/spam)

**📉 Optional: Confusion Matrix (Visualization)**

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',

xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

* Confusion matrix shows:
  + True Positives (Spam predicted as Spam)
  + True Negatives (Ham predicted as Ham)
  + False Positives (Ham predicted as Spam)
  + False Negatives (Spam predicted as Ham)
* Visualized using a heatmap

**✉️ Step 8: Test on a New Sample Message**

sample = ["Urgent! Your bank account is at risk. Click the link to secure it."]

sample\_vec = vectorizer.transform(sample)

print("🚨 Prediction (1=Spam, 0=Ham):", model.predict(sample\_vec)[0])

* This is a manual test case.
* The text is vectorized, then the model predicts if it's spam (1) or ham (0).

**✅ Summary**

| **Step** | **Action** | **Key Tool** |
| --- | --- | --- |
| 1 | Load dataset | pandas |
| 2 | Clean & preprocess | map, rename |
| 3 | Split data | train\_test\_split |
| 4 | Convert text to numbers | CountVectorizer |
| 5 | Train model | MultinomialNB |
| 6 | Make predictions | predict() |
| 7 | Evaluate performance | accuracy\_score, precision\_score, recall\_score |
| 8 | Visualize confusion matrix | seaborn, matplotlib |
| 9 | Test on custom input | vectorizer.transform() |