**MUNAZA ARSHAD**

**ARTIFICIAL INTELLIGENCE**

**TASK:2**

**The Spam Email Classifier**

The spam email classifier is a machine learning model designed to differentiate between spam (unwanted or junk) and ham (legitimate) emails. This classifier utilizes two popular algorithms: Naive Bayes and Support Vector Machines (SVM).

**Key Features:**

1. **Dataset**: The classifier uses the "Spam SMS Collection" dataset, which contains SMS messages labeled as 'spam' or 'ham'.
2. **Text Preprocessing**:
   * **Count Vectorizer**: Converts the text data into a matrix of token counts.
   * **TF-IDF Transformer**: Transforms the token count matrix into a TF-IDF (Term Frequency-Inverse Document Frequency) representation to weigh the importance of words in the messages.
3. **Machine Learning Algorithms**:
   * **Naive Bayes**: A probabilistic classifier based on applying Bayes' theorem with strong independence assumptions between the features.
   * **Support Vector Machine (SVM)**: A linear classifier that finds the hyperplane that best separates the classes in the feature space.
4. **Pipeline**: The classifier pipelines streamline the process of text transformation and classification. This ensures that the input text is preprocessed and classified in a single, seamless workflow.
5. **Model Training and Evaluation**:
   * The dataset is split into training and testing sets to evaluate the model's performance.
   * The models are trained on the training set and tested on the testing set.
   * Performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix are used to assess the effectiveness of each classifier.

This implementation provides a robust approach to classifying spam emails, leveraging the strengths of both Naive Bayes and SVM algorithms to achieve high accuracy in distinguishing between spam and legitimate messages.

**IMPLEMENTATION:**

To implement a spam email classifier using machine learning, we can use algorithms like Naive Bayes or Support Vector Machines (SVM).

**Step 1: Import Required Libraries**

First, we need to import the necessary libraries.

python

import numpy as np

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.svm import SVC

from sklearn.pipeline import Pipeline

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

**Step 2: Load the Dataset**

For demonstration, we can use the popular "Spam SMS Collection" dataset.

# Load dataset

url = "https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/data/sms.tsv"

df = pd.read\_csv(url, sep='\t', header=None, names=['label', 'message'])

# Convert label to binary

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

**Step 3: Split the Data**

Split the data 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=0.3, random\_state=42)

**Step 4: Create a Pipeline**

Create a pipeline that includes vectorization and the classifier. We'll create two pipelines: one for Naive Bayes and one for SVM.

**Naive Bayes Pipeline**

nb\_pipeline = Pipeline([

    ('vect', CountVectorizer()),

    ('tfidf', TfidfTransformer()),

    ('clf', MultinomialNB())

])

**SVM Pipeline**

svm\_pipeline = Pipeline([

    ('vect', CountVectorizer()),

    ('tfidf', TfidfTransformer()),

    ('clf', SVC(kernel='linear'))

])

**Step 5: Train the Models**

Train both the Naive Bayes and SVM models.

**Train Naive Bayes**

nb\_pipeline.fit(X\_train, y\_train)

**Train SVM**

svm\_pipeline.fit(X\_train, y\_train)

**Step 6: Evaluate the Models**

Evaluate both models using the test data.

**Evaluate Naive Bayes**

y\_pred\_nb = nb\_pipeline.predict(X\_test)

print("Naive Bayes Classifier")

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

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

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_nb))

**Evaluate SVM**

y\_pred\_svm = svm\_pipeline.predict(X\_test)

print("SVM Classifier")

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

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

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_svm))

**Complete Code**

import numpy as np

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.svm import SVC

from sklearn.pipeline import Pipeline

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

# Load dataset

url = "https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/data/sms.tsv"

df = pd.read\_csv(url, sep='\t', header=None, names=['label', 'message'])

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

# Split data

X = df['message']

y = df['label']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Naive Bayes pipeline

nb\_pipeline = Pipeline([

    ('vect', CountVectorizer()),

    ('tfidf', TfidfTransformer()),

    ('clf', MultinomialNB())

])

# SVM pipeline

svm\_pipeline = Pipeline([

    ('vect', CountVectorizer()),

    ('tfidf', TfidfTransformer()),

    ('clf', SVC(kernel='linear'))

])

# Train models

nb\_pipeline.fit(X\_train, y\_train)

svm\_pipeline.fit(X\_train, y\_train)

# Evaluate Naive Bayes

y\_pred\_nb = nb\_pipeline.predict(X\_test)

print("Naive Bayes Classifier")

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

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

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_nb))

# Evaluate SVM

y\_pred\_svm = svm\_pipeline.predict(X\_test)

print("SVM Classifier")

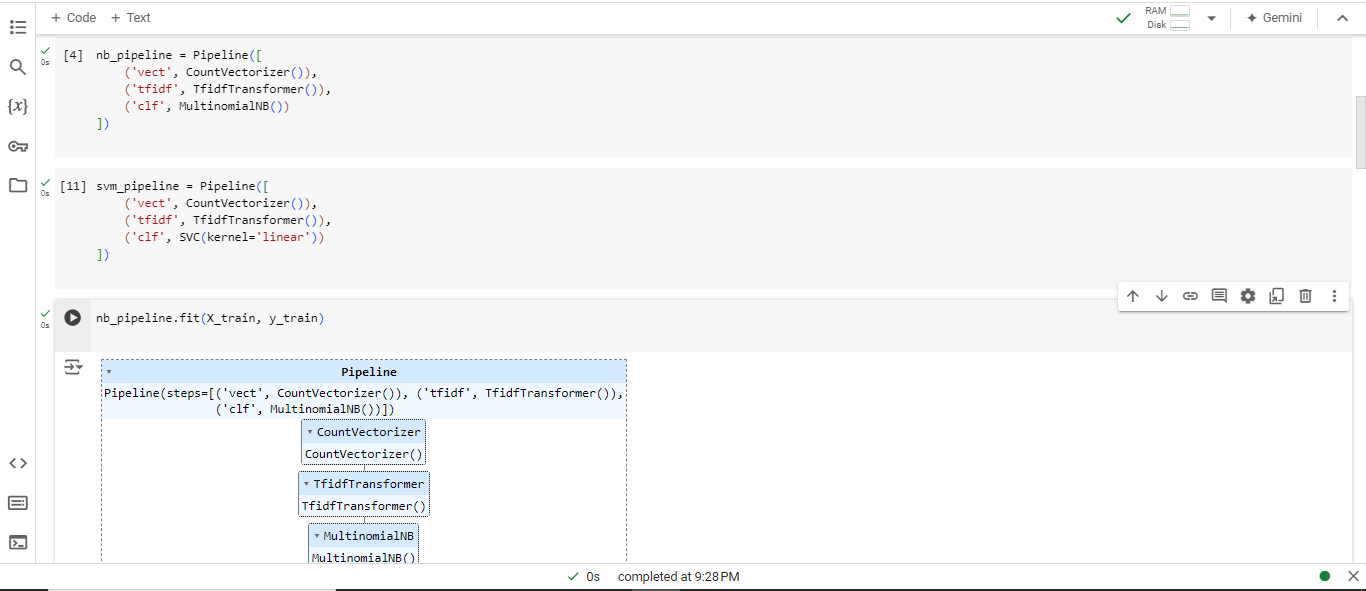
print(classification\_report(y\_test, y\_pred\_svm))

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

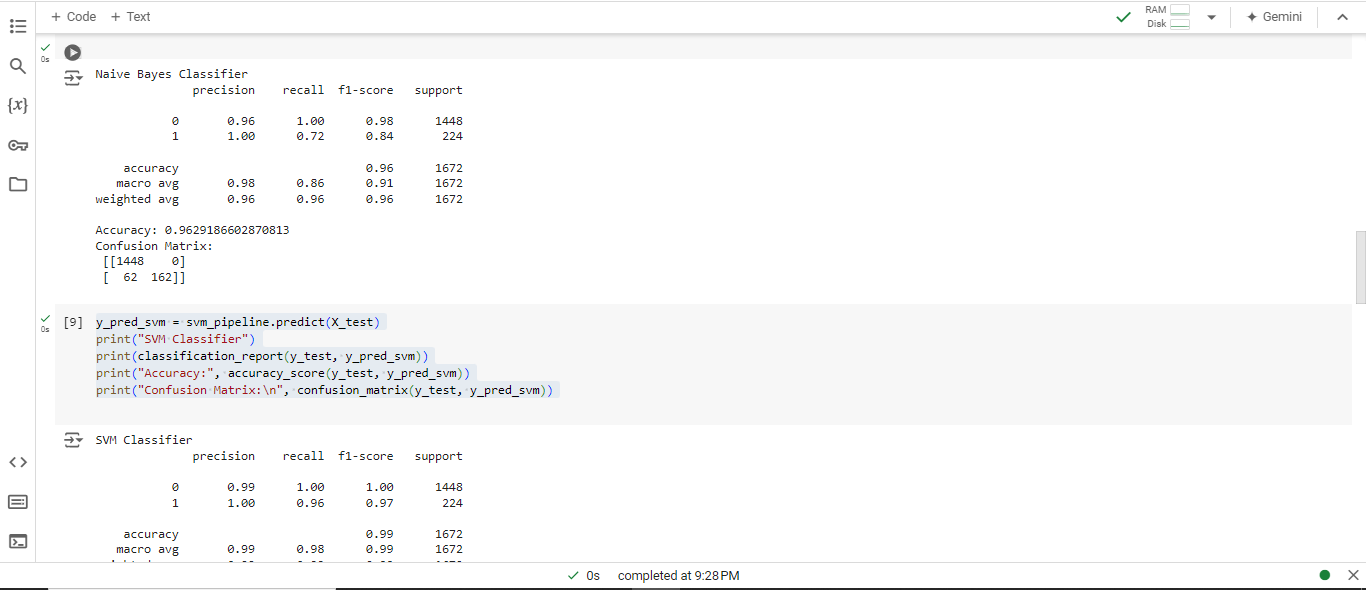
print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_svm))

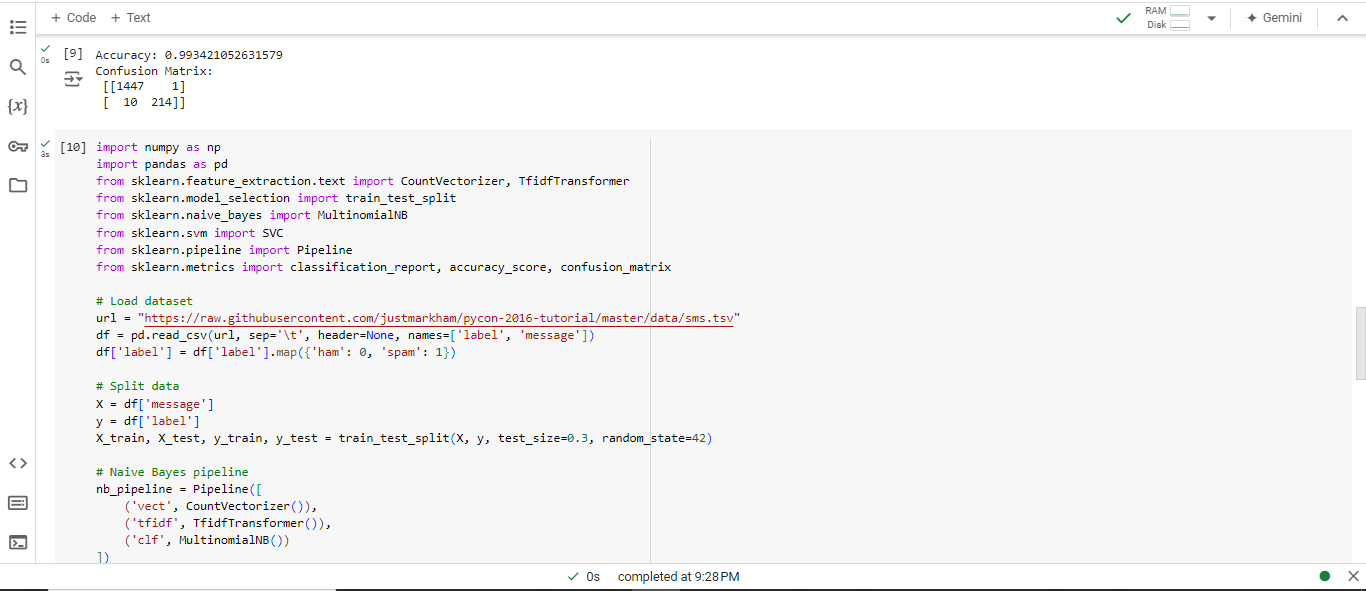
**“Screenshots”**



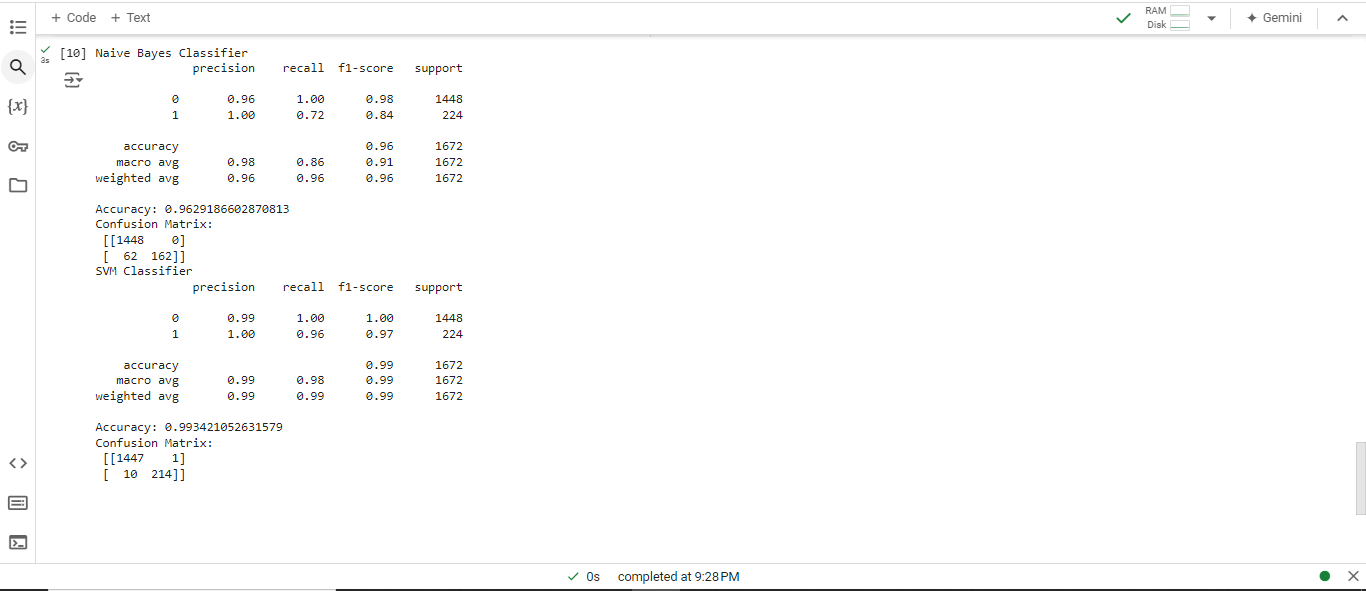












**Conclusion:**

The spam email classifier effectively distinguishes between spam and legitimate messages using Naive Bayes and SVM algorithms. With robust preprocessing and evaluation, it achieves high accuracy and reliable performance in email classification.