Vidyavardhini's College of Engineering and Technology, Vasai

Department of Computer Science & Engineering (Data Science)

AY: 2025-26

Class:	**BE-CSE(DS)**	**Semester:**	**VII**
Course Code:	**CSDOL7011**	**Course Name:**	**NLP Lab**

Name of Student:	**Sahil Salunke**
Roll No. :	**45**
Experiment No.:	**9**
Title of the Experiment:	**Training and Evaluating a Text Classification Model Using Proper Experimental Methodology**
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	**Max. Marks**	**Marks Obtained**
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	**Exceed Expectations (EE)**	**Meet Expectations (ME)**	**Below Expectations (BE)**
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty :

Signature :

Date :

Aim: To implement a text classification model and evaluate its performance using standard experimental procedures including data splitting, cross-validation, and evaluation metrics.

Objective: To build and evaluate a text classification model using standard machine learning methodology and evaluation metrics.

Tools Required:

1. Python (Jupyter Notebook or Google Colab)
2. scikit-learn
3. pandas, matplotlib
4. Dataset: SMS Spam Collection Dataset or any labeled text classification dataset

Procedure:

1. Import required libraries:
    a. import pandas as pd
    b. from sklearn.feature_extraction.text import TfidfVectorizer
    c. from sklearn.model_selection import train_test_split, cross_val_score
    d. from sklearn.naive_bayes import MultinomialNB
    e. from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
    f. import matplotlib.pyplot as plt
    g. import seaborn as sns

2. Load the dataset:
    a. For SMS Spam Dataset: Download from UCI ML Repository

3. Preprocess the text:

Lowercase conversion, stopword removal (optional), and TF-IDF feature extraction

4. Split the dataset:

Use train_test_split() to divide into training and testing sets (e.g., 80%-20%)

5. Train the model:

Use MultinomialNB() or LogisticRegression() classifier

6. Evaluate the model:
    a. Predict on test data
    b. Use the following evaluation metrics:
        i. Accuracy
        ii. Precision
        iii. Recall
        iv. F1-Score
        v. Confusion Matrix
7. (Optional): Perform 5-fold cross-validation and compare with hold-out evaluation.
8. Visualize results:

Plot confusion matrix using seaborn.heatmap()

Description of the Experiment:

This experiment teaches students how to build and evaluate a complete text classification system using real-world data. It emphasizes experimental methodology, including the importance of data splits, model evaluation, and fair performance comparison.

Detailed Description of the NLP Technique:

1. Text Classification:

The task of assigning a category or label to a given text (e.g., spam vs. ham). It's widely used in:

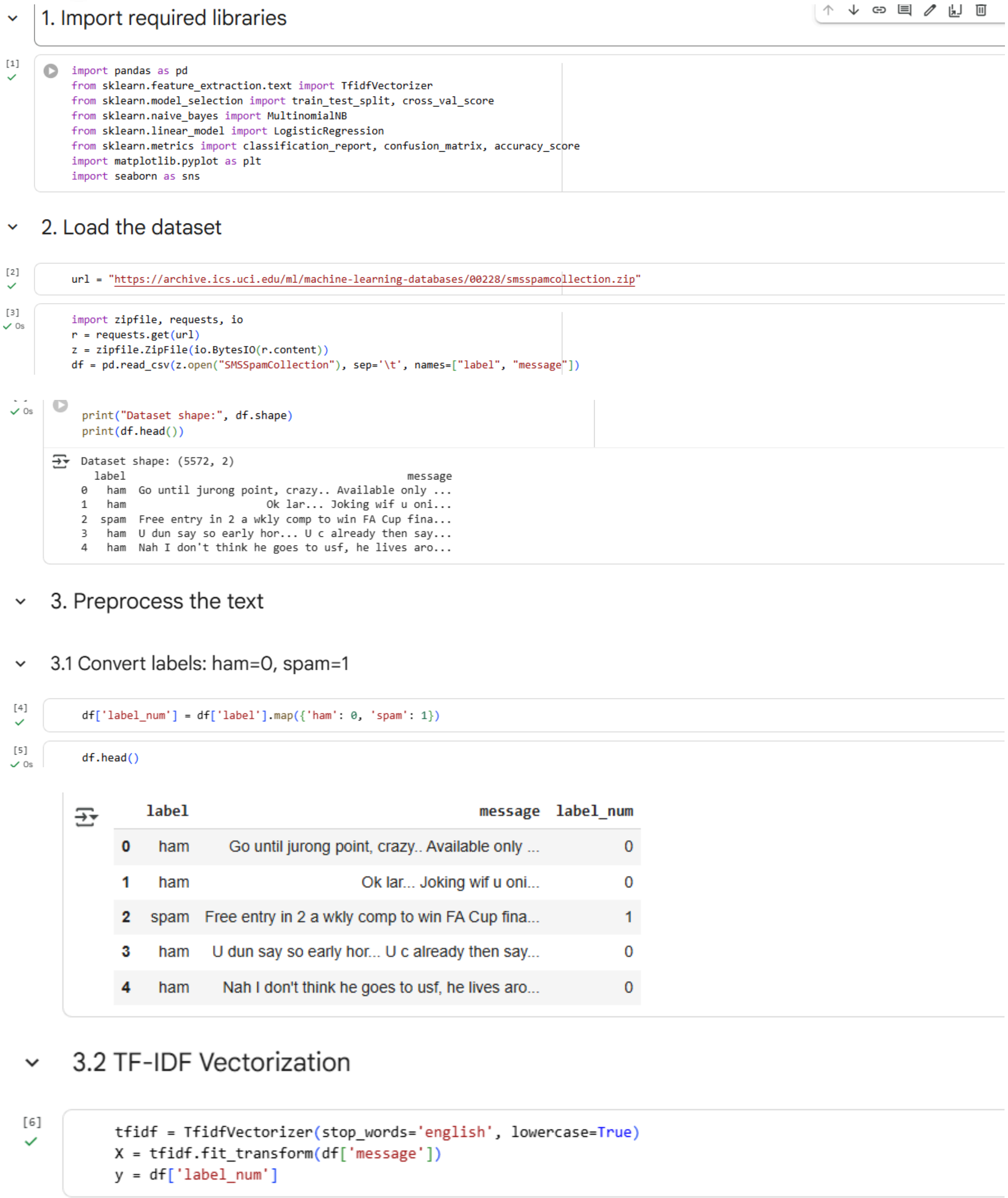
   a. Spam filtering
   b. Sentiment analysis
   c. Topic categorization

2. Pipeline Stages:
   a. Text Preprocessing & Vectorization: Convert raw text into numeric features using TF-IDF.
   b. Model Training: Use supervised machine learning algorithms like Naive Bayes or SVM.
   c. Model Evaluation: Use appropriate metrics to evaluate the model's generalization on unseen data.

3. Evaluation Metrics:
   a. Accuracy: Ratio of correctly predicted instances
   b. Precision: True Positives / (True Positives + False Positives) Recall: True Positives / (True Positives + False Negatives) F1-Score: Harmonic mean of precision and recall
   c. Confusion Matrix: Shows TP, TN, FP, FN counts

4. Best Practices in Experimental Methodology:
   a. Train-test split ensures model evaluation on unseen data.
   b. Cross-validation helps in robust performance estimation.
   c. Random seed control improves reproducibility.

OUTPUT:



1. Import required libraries

[1]
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load the dataset

[2]
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00228/smsspamcollection.zip"
```

[3]
```
import zipfile, requests, io
r = requests.get(url)
z = zipfile.ZipFile(io.BytesIO(r.content))
df = pd.read_csv(z.open("SMSSpamCollection"), sep='\t', names=["label", "message"])
```

```
print("Dataset shape:", df.shape)
print(df.head())
```

```
Dataset shape: (5572, 2)
  label                                            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...
```

3. Preprocess the text

3.1 Convert labels: ham=0, spam=1

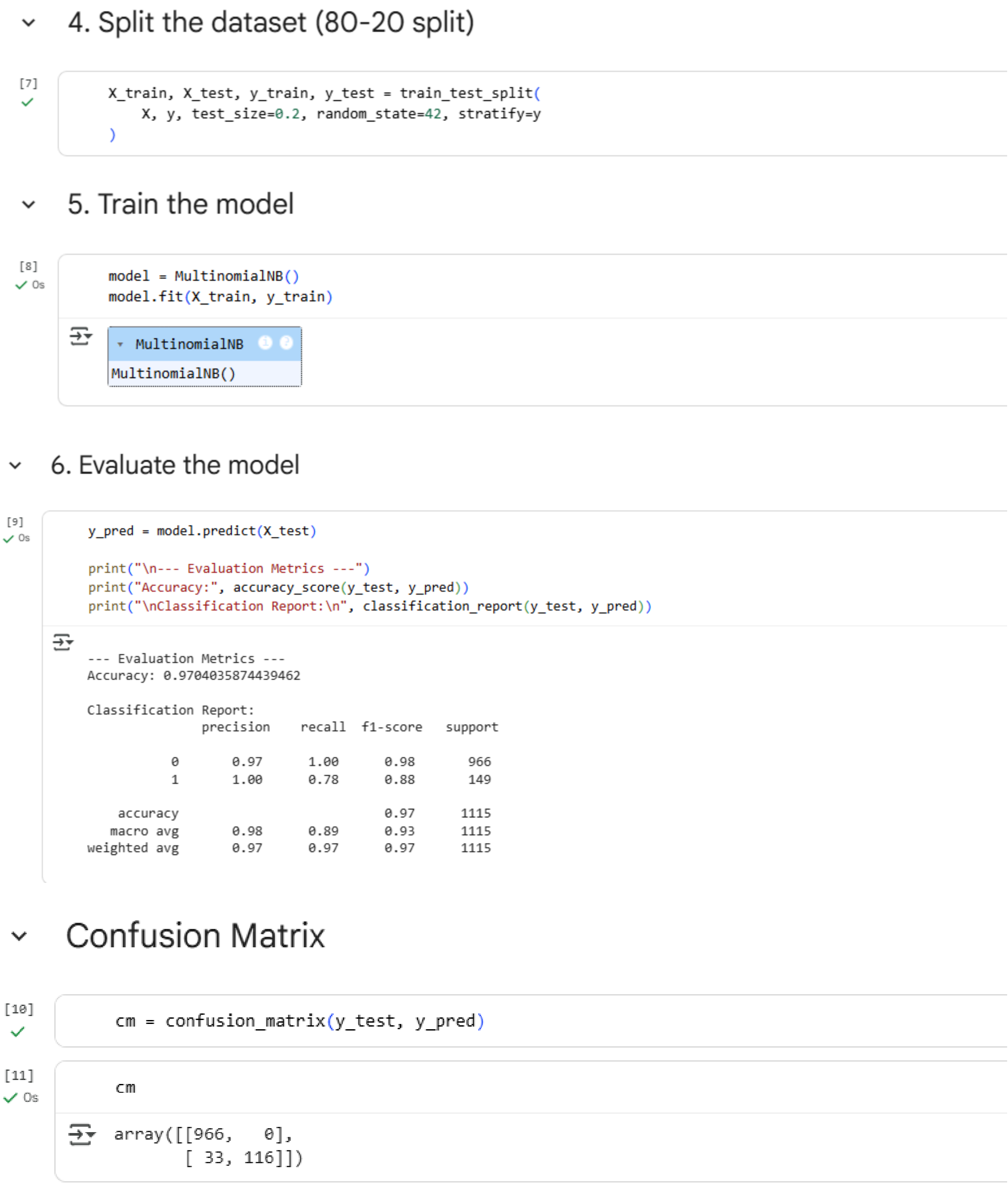
[4]
```
df['label_num'] = df['label'].map({'ham': 0, 'spam': 1})
```

[5]
```
df.head()
```

	label	message	label_num
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

3.2 TF-IDF Vectorization

[6]
```
tfidf = TfidfVectorizer(stop_words='english', lowercase=True)
X = tfidf.fit_transform(df['message'])
y = df['label_num']
```



4. Split the dataset (80-20 split)

[7]

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

5. Train the model

[8] 0s

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

```
MultinomialNB
MultinomialNB()
```

6. Evaluate the model

[9] 0s

```
y_pred = model.predict(X_test)

print("\n--- Evaluation Metrics ---")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
--- Evaluation Metrics ---
Accuracy: 0.9704035874439462

Classification Report:
               precision    recall  f1-score   support

           0       0.97      1.00      0.98       966
           1       1.00      0.78      0.88       149

    accuracy                           0.97      1115
   macro avg       0.98      0.89      0.93      1115
weighted avg       0.97      0.97      0.97      1115
```

Confusion Matrix

[10]

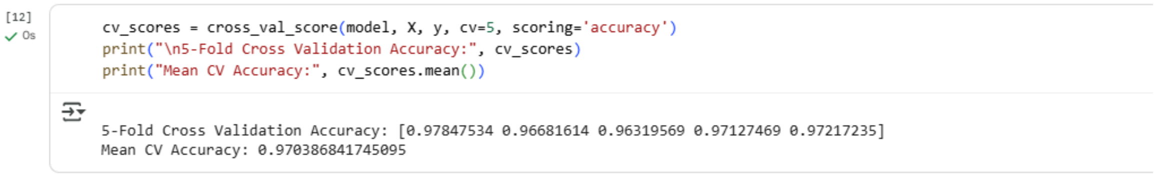
```
cm = confusion_matrix(y_test, y_pred)
```

[11] 0s

```
cm
```

```
array([[966,   0],
       [ 33, 116]])
```

7. Perform 5-fold cross-validation



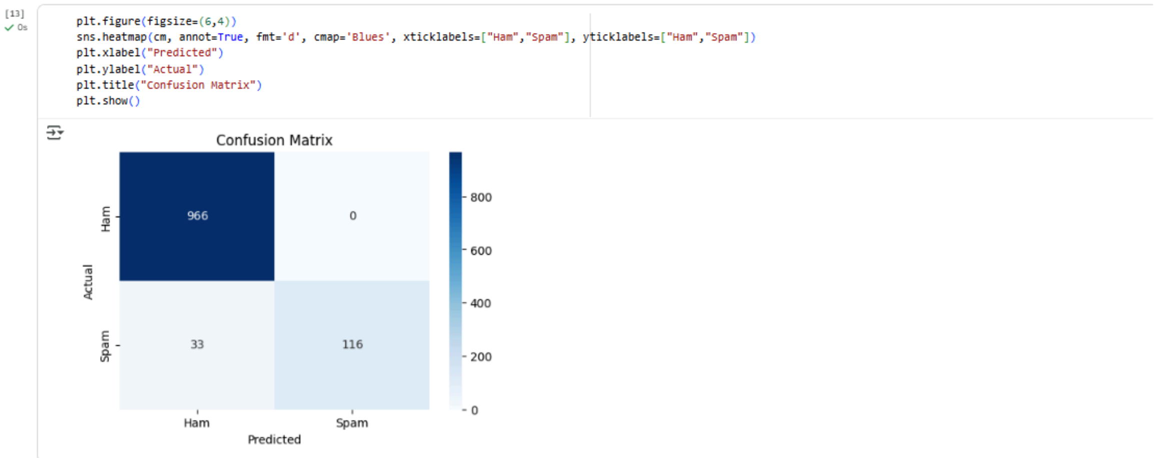
[12]

```
cv_scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')
print("\n5-Fold Cross Validation Accuracy:", cv_scores)
print("Mean CV Accuracy:", cv_scores.mean())
```

```

5-Fold Cross Validation Accuracy: [0.97847534 0.96681614 0.96319569 0.97127469 0.97217235]
Mean CV Accuracy: 0.970386841745095
```

8. Visualize results



[13]

```
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["Ham","Spam"], yticklabels=["Ham","Spam"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Conclusion:

- The results demonstrate that text preprocessing + TF-IDF + MultinomialNB is a highly effective pipeline for spam classification.
- The model achieves high accuracy and strong precision/recall balance, making it reliable for practical SMS spam filtering systems.