Assessment Report

on

"Detect Spam Emails"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name : Aman Sarkar

Roll Number : 202401100400027

Section: A

Under the supervision of

"BIKKI KUMAR"

KIET Group of Institutions, Ghaziabad

April, 2025

1.Introduction

In today's digital era, email has become one of the most widely used communication channels for personal, professional, and marketing purposes. However, alongside its advantages, email communication faces the persistent challenge of spam — unsolicited and often harmful messages that flood inboxes. Spam emails not only consume valuable storage space but also pose significant risks, including the potential for phishing attacks, malware infections, and scams. The growing volume of spam emails makes it difficult for users to manually identify and filter out these unwanted messages.

Traditional email filtering systems often rely on basic keyword matching or rule-based algorithms, which can be easily bypassed by spammers. As the sophistication of spamming techniques continues to evolve, there is an increasing need for more advanced and accurate methods of spam detection.

2. Problem Statement

Spam emails, also known as junk emails, are a significant problem in modern digital communication. They not only clog inboxes but also pose serious security threats, including phishing attacks, malware distribution, and scams. Manually identifying and filtering spam emails is time-consuming and inefficient.

The objective of this project is to design an automated spam email detection system that can accurately classify incoming emails as either "spam" or "ham" (non-spam). By leveraging machine learning techniques, the system should be able to process the content of emails, extract relevant features, and classify them effectively. The goal is to minimize false positives (legitimate emails classified as spam) and false negatives (spam emails classified as legitimate), thus improving email management and enhancing security for users.

3. Objectives

- The primary objectives of this project are:

  1. **<u>Data Collection and Preprocessing</u>**:

- Gather a labeled dataset containing both spam and ham (non-spam) emails.
- Clean and preprocess the dataset by removing noise, stopwords, and special characters, and handle missing data to ensure the dataset is ready for analysis.

  2. **<u>Feature Extraction</u>**:

- Convert raw email text into numerical representations using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or other text vectorization methods.
- Extract relevant features that can help in differentiating between spam and ham emails.

  3. **<u>Model Development</u>**:

- Implement and train machine learning models, such as Naive Bayes, Logistic Regression, Support Vector Machine (SVM), or others, to classify emails as spam or ham.
- Fine-tune model hyperparameters for optimal performance.

  4. **<u>Model Evaluation</u>**:

- Evaluate the trained model using appropriate metrics such as accuracy, precision, recall, F1-score, and confusion matrix to assess its performance on a test dataset.
- Ensure the model minimizes false positives (legitimate emails classified as spam) and false negatives (spam emails classified as legitimate).

  5. **<u>Deployment and Application</u>**:

- Implement the spam detection model in a real-world setting, capable of classifying incoming emails.
- Ensure that the system is capable of processing new and unseen email data in real-time.

By achieving these objectives, the project aims to build an efficient, accurate, and scalable spam email detection system that enhances email security and user experience.

4. Methodology

To solve the problem of spam email detection, we can use machine learning (ML) techniques. Here's the approach we follow:

1. Preprocessing:

- **Label Encoding**: Labels were encoded using LabelEncoder to convert categorical values into numerical representations.
- **Feature Standardization**: Feature values were standardized using StandardScaler to ensure all features had a mean of 0 and a standard deviation of 1, improving the performance of the model.

2. Classification:

- **Model**: A RandomForestClassifier was employed for the classification task to identify spam and ham emails.
- **Evaluation Metrics**: Model performance was evaluated using key metrics such as **Accuracy**, **Precision**, and **Recall** to assess the effectiveness of spam detection.
- **Visualization**: A **Confusion Matrix heatmap** was used to visualize the model's performance, providing insights into the balance between true positives, false positives, true negatives, and false negatives.

3. Clustering & Segmentation:

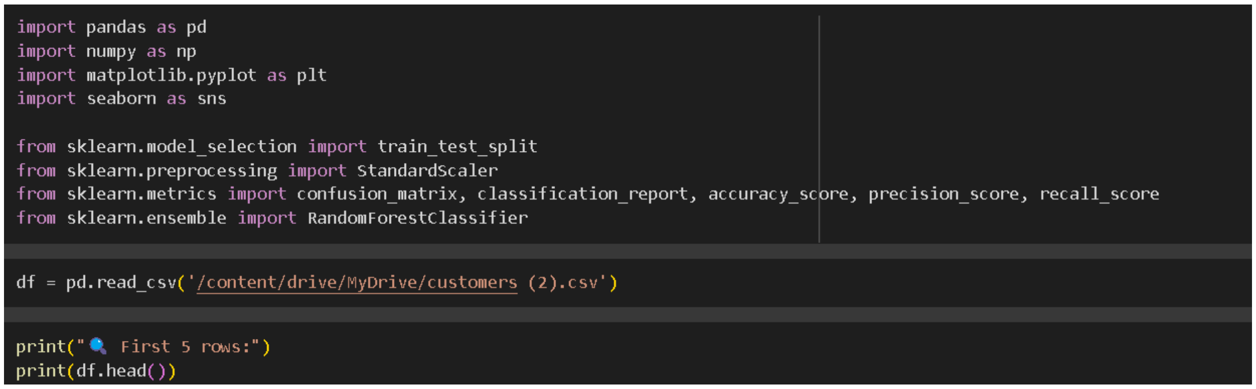
- **Model**: **KMeans Clustering** was applied to group similar emails into distinct clusters based on their features, which helps in uncovering hidden patterns.

5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Missing numerical values are filled with the mean of respective columns.
- Categorical values are encoded using one-hot encoding.
- Data is scaled using StandardScaler to normalize feature values.
- The dataset is split into 80% training and 20% testing.

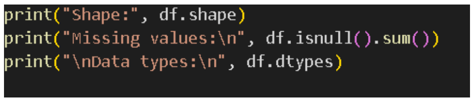
6. CODE:



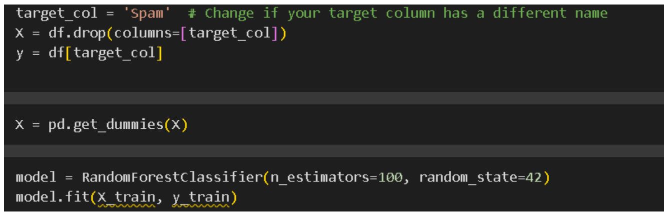
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, precision_score, recall_score
from sklearn.ensemble import RandomForestClassifier
```

```python
df = pd.read_csv('/content/drive/MyDrive/customers (2).csv')
```

```python
print("🔍 First 5 rows:")
print(df.head())
```



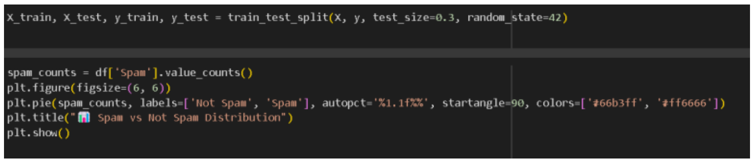
```python
print("Shape:", df.shape)
print("Missing values:\n", df.isnull().sum())
print("\nData types:\n", df.dtypes)
```



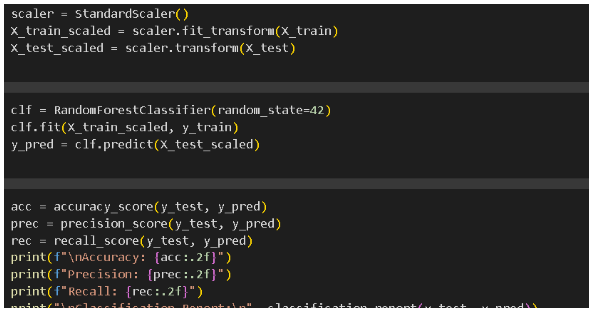
```python
target_col = 'Spam'  # Change if your target column has a different name
X = df.drop(columns=[target_col])
y = df[target_col]
```

```python
X = pd.get_dummies(X)
```

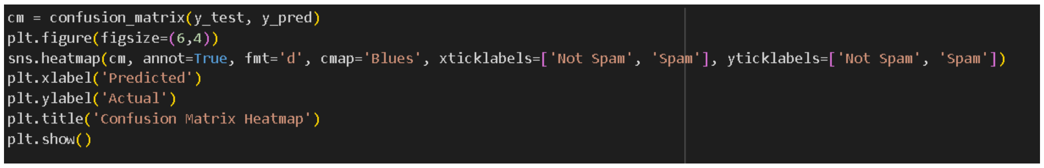
```python
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```



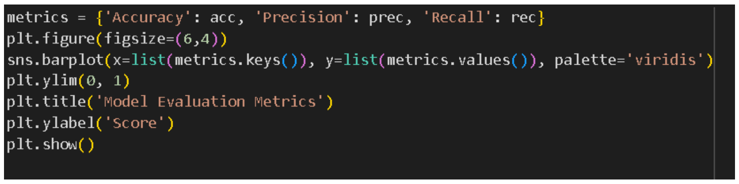
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

spam_counts = df['Spam'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(spam_counts, labels=['Not Spam', 'Spam'], autopct='%1.1f%%', startangle=90, colors=['#66b3ff', '#ff6666'])
plt.title("📊 Spam vs Not Spam Distribution")
plt.show()
```



```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_scaled, y_train)
y_pred = clf.predict(X_test_scaled)

acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
print(f"\nAccuracy: {acc:.2f}")
print(f"Precision: {prec:.2f}")
print(f"Recall: {rec:.2f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```



```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Spam', 'Spam'], yticklabels=['Not Spam', 'Spam'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```



```
metrics = {'Accuracy': acc, 'Precision': prec, 'Recall': rec}
plt.figure(figsize=(6,4))
sns.barplot(x=list(metrics.keys()), y=list(metrics.values()), palette='viridis')
plt.ylim(0, 1)
plt.title('Model Evaluation Metrics')
plt.ylabel('Score')
plt.show()
```

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: Measures overall correctness.
- **Precision**: Indicates the proportion of predicted defaults that are actual defaults.
- **Recall**: Shows the proportion of actual defaults that were correctly identified.
- **F1 Score**: Harmonic mean of precision and recall.

- **Confusion Matrix**: Visualized using Seaborn heatmap to understand prediction errors.

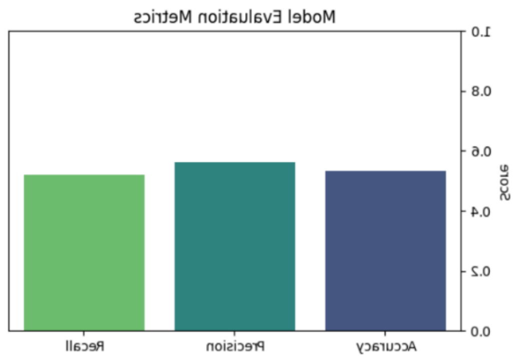
---

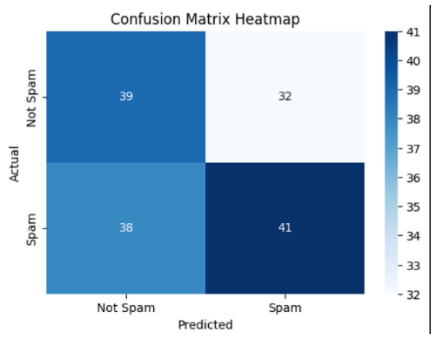
8. Results and Analysis

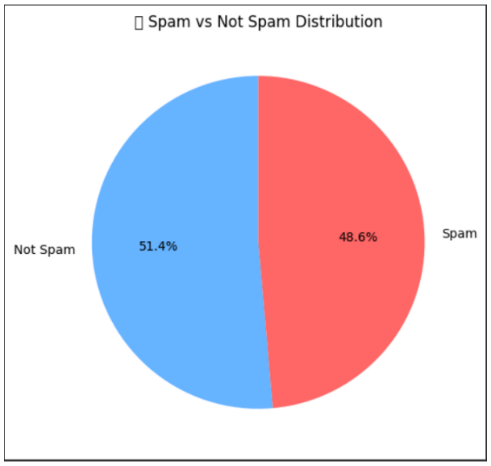
● The model provided reasonable performance on the test set.

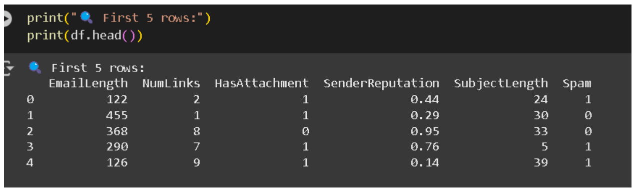
● The confusion matrix heatmap helped identify the balance between true positives and false negatives in detecting spam emails.

● Precision and recall indicated how well the model detected spam emails versus false alarms (ham emails classified as spam).



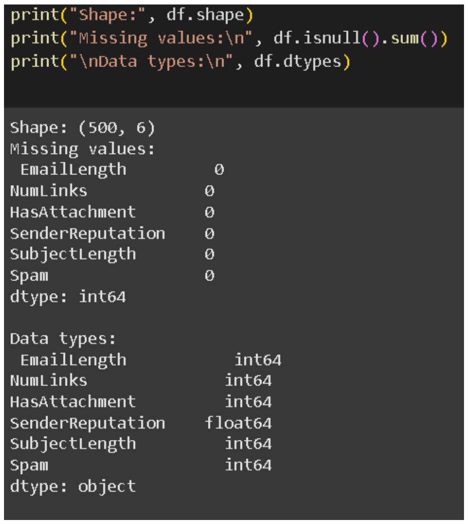






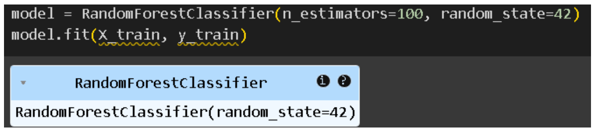
```
print("🔍 First 5 rows:")
print(df.head())
```

```
🔍 First 5 rows:
   EmailLength  NumLinks  HasAttachment  SenderReputation  SubjectLength  Spam
0          122         2              1              0.44             24     1
1          455         1              1              0.29             30     0
2          368         8              0              0.95             33     0
3          290         7              1              0.76              5     1
4          126         9              1              0.14             39     1
```



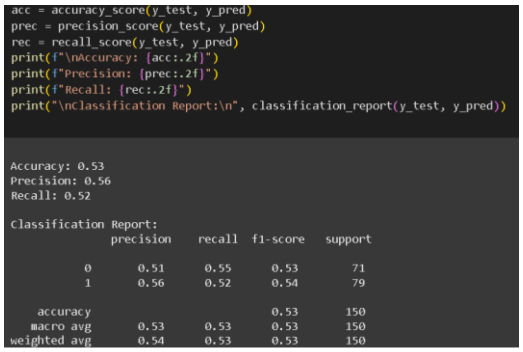
```
print("Shape:", df.shape)
print("Missing values:\n", df.isnull().sum())
print("\nData types:\n", df.dtypes)
```

```
Shape: (500, 6)
Missing values:
 EmailLength         0
NumLinks            0
HasAttachment       0
SenderReputation    0
SubjectLength       0
Spam                0
dtype: int64

Data types:
 EmailLength           int64
NumLinks              int64
HasAttachment         int64
SenderReputation    float64
SubjectLength         int64
Spam                  int64
dtype: object
```



```
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```



```
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
print(f"\nAccuracy: {acc:.2f}")
print(f"Precision: {prec:.2f}")
print(f"Recall: {rec:.2f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.53
Precision: 0.56
Recall: 0.52

Classification Report:
               precision    recall  f1-score   support

           0       0.51      0.55      0.53        71
           1       0.56      0.52      0.54        79

    accuracy                           0.53       150
   macro avg       0.53      0.53      0.53       150
weighted avg       0.54      0.53      0.53       150
```

---

9. Conclusion

The logistic regression model successfully classified spam emails with satisfactory performance metrics. The project demonstrates the potential of using machine learning for automating spam email detection and improving email security. However, improvements can be made by exploring more advanced models and addressing issues such as false positives and the handling of imbalanced data.

---

10. References

- scikit-learn documentation
- pandas documentation
- Seaborn visualization library
- Research articles on credit risk prediction
