RV UNIVERSITY
School of Computer Science and Engineering
Bengaluru – 560059

Machine Learning For Cyber Security

VII Semester

Student Name:	**Hussaina Mustafa**
USN:	**1RVU22BSC035**
Academic Year	**2025 - 2026**

SOURCE CODE:

1. Data Preprocessing & Feature Extraction

```python
import pandas as pd
import re
import string
import pickle
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

# -------------------------
# Load dataset
# -------------------------
df = pd.read_csv("processed_data.csv")

# Clean text function
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'\d+', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = " ".join(text.split())
    return text

# Apply cleaning
df['clean_text'] = df['message'].apply(clean_text)

# Features and labels
X = df['clean_text']
y = df['label'].astype(int)

# Convert text → numerical features
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X_tfidf = vectorizer.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_tfidf, y, test_size=0.2, random_state=42, stratify=y
)
```

2. Model Training & Evaluation

```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# -------------------------
# Define Models
# -------------------------
models = {
    "Naive Bayes": MultinomialNB(),
    "Logistic Regression": LogisticRegression(max_iter=2000),
    "SVM": LinearSVC(),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "KNN": KNeighborsClassifier(n_neighbors=5),
    # "Gradient Boosting": GradientBoostingClassifier(n_estimators=100, random_state=42)
}
results = {}

# -------------------------
# Train & Evaluate
# -------------------------
for name, model in models.items():
    print(f"\nTraining {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    results[name] = {"Accuracy": acc, "Precision": prec, "Recall": rec, "F1": f1}

    print(f"{name} - Accuracy: {acc:.4f}, Precision: {prec:.4f}, Recall: {rec:.4f}, F1: {f1:.4f}")
```

```python
# -------------------------
# Pick Best Model
# -------------------------
best_model_name = max(results, key=lambda x: results[x]['F1'])
best_model = models[best_model_name]

print("\nBest Model:", best_model_name, results[best_model_name])

# Save best model and vectorizer
pickle.dump(best_model, open("spam_model.pkl", "wb"))
pickle.dump(vectorizer, open("vectorizer.pkl", "wb"))
# Save model info
model_info = {
    "best_model_name": best_model_name,
    "dataset_shape": df.shape,
    "evaluation": results[best_model_name]  # metrics for best model
}

pickle.dump(model_info, open("model_info.pkl", "wb"))
```

3. Streamlit App

```python
import streamlit as st
import PyPDF2
import pickle
import re
import string
import pandas as pd
import plotly.express as px

# --------------------------
# Load trained model, vectorizer, and model info
# --------------------------
model = pickle.load(open("spam_model.pkl", "rb"))
vectorizer = pickle.load(open("vectorizer.pkl", "rb"))
model_info = pickle.load(open("model_info.pkl", "rb"))

# --------------------------
# Text cleaning function
# --------------------------
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'\d+', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = " ".join(text.split())
    return text

# --------------------------
# Streamlit UI
# --------------------------
st.set_page_config(page_title="Email Spam Detector", page_icon="📧", layout="wide")
st.title("📧 Email Spam Detector")
st.write("Upload one or more PDF emails, and the model will classify them as SPAM or NOT SPAM.")
```

4. Screenshots of the working web page

a. Metrics of each model

```
Training Naive Bayes...
Naive Bayes - Accuracy: 0.9686, Precision: 0.9951, Recall: 0.9575, F1: 0.9759

Training Logistic Regression...
Logistic Regression - Accuracy: 0.9920, Precision: 0.9916, Recall: 0.9965, F1: 0.9940

Training SVM...
SVM - Accuracy: 0.9938, Precision: 0.9938, Recall: 0.9968, F1: 0.9953

Training Random Forest...
Random Forest - Accuracy: 0.9940, Precision: 0.9948, Recall: 0.9962, F1: 0.9955

Training KNN...
KNN - Accuracy: 0.8933, Precision: 0.8622, Recall: 0.9993, F1: 0.9257

Best Model: Random Forest {'Accuracy': 0.9940334128878282, 'Precision': 0.9948279291824149, 'Recall': 0.9962151394422311, 'F1': 0.9955210510600179}
```

b. UI

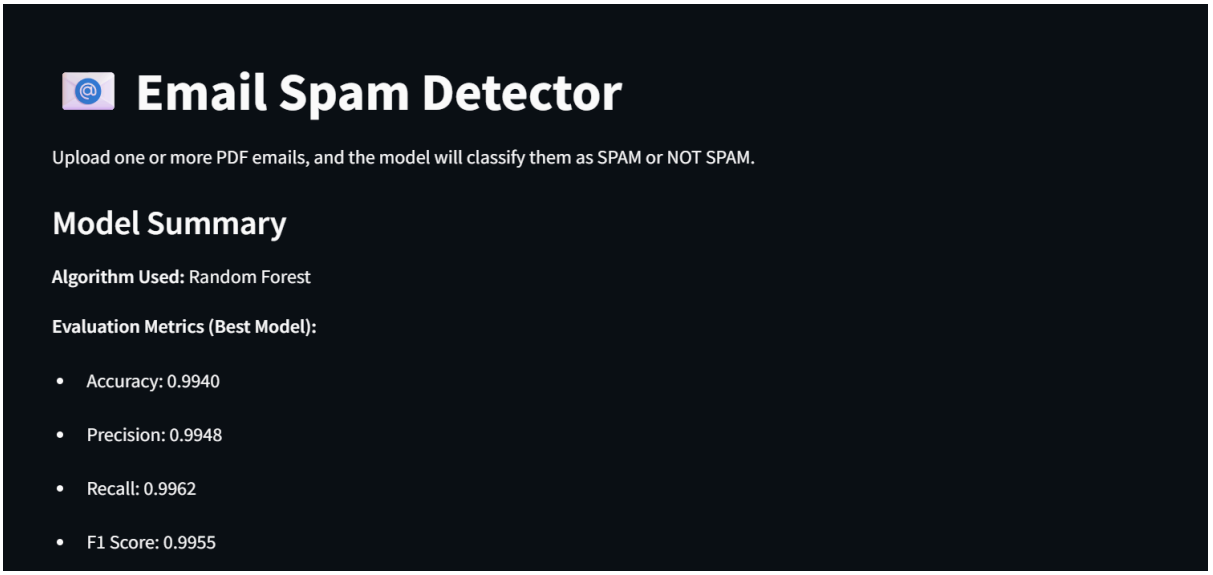

Fig: Shows the summary of the best model chosen

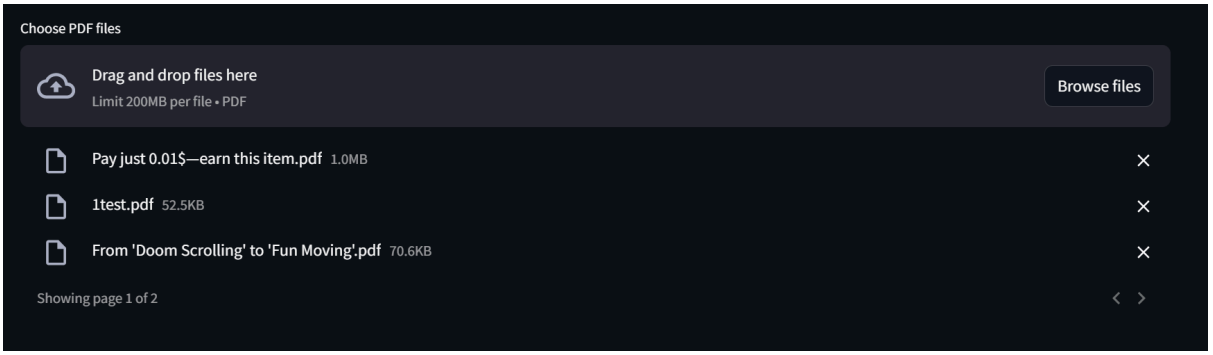

Fig**:** Takes multiple/one pdf email as input

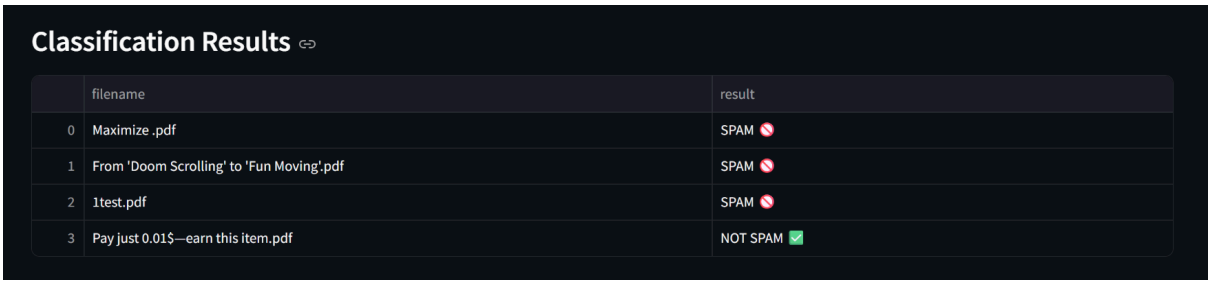

Fig: Classifies emails into Spam or Not spam

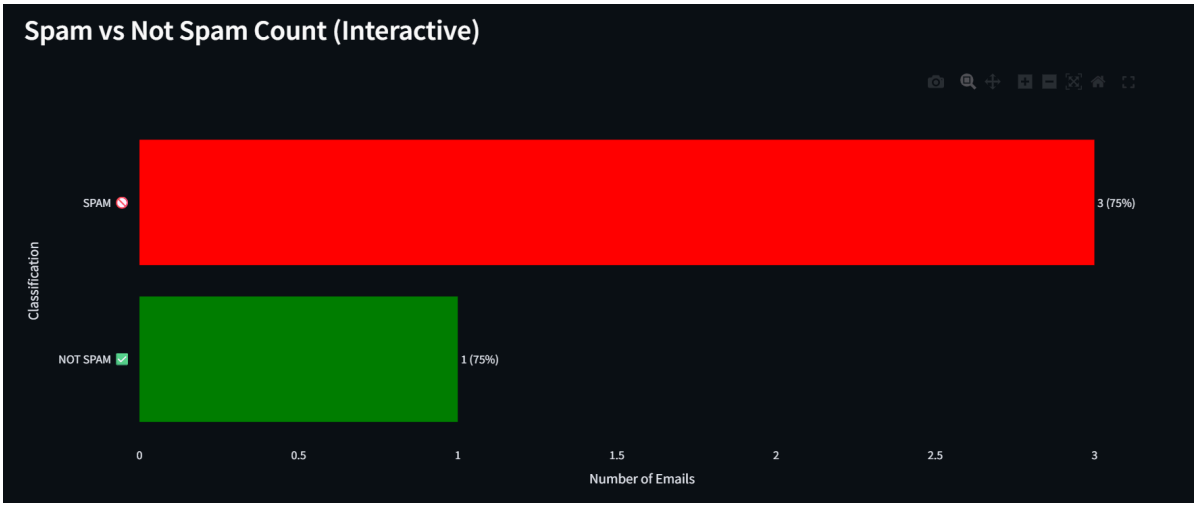

Fig: Interactive graph showing no.of emails and types of classification, includes features: Download plot as png, zoom in, zoom out, pan, autoscale, reset axes, full screen