1.TITLE PAGE

TITLE: PHISHING EMAIL DETECTION USING MACHINE LEARNING AND PY TK

SUBMITTED BY: KUSHAGRA GHADIGAONKAR AND ANAND KHANDARE

INSTITUTION: DIGI SURAKSHA

DATE: 12 MAY 2025

2. INTRODUCTION

- **Phishing** is a cybercrime where attackers impersonate trustworthy entities to steal sensitive data.
- Manual identification is time-consuming and error-prone.
- This project provides a **real-time email classifier** using **Machine Learning** with a GUI made in **Tkinter** (**PyTk**).

3.OBJECTIVE

- Build a classifier to detect phishing emails.
- Use NLP and Machine Learning (Naive Bayes, TF-IDF).
- Develop a **Tkinter-based desktop application.**
- Provide user-friendly predictions with confidence scores.

4.PROBLEMS

- Email users receive hundreds of messages daily.
- Most users can't spot well-crafted phishing content.
- Traditional spam filters often fail to detect **targeted phishing** emails.
- Lack of tools for **on-demand verification** by users.

5.SOLUTIONS

- A **Python-based GUI application** that:
- Uses **Machine Learning** (**Naive Bayes**) to classify emails.
- Cleans and analyzes email content using **TF-IDF** vectorization.
- Provides a **confidence score** for the prediction.
- Is lightweight, offline, and user-friendly using **Tkinter**.

6.REAL WORLD USE CASE

- **Corporate Email Gateways**
- Detect phishing before user opens the email.
- **Personal Cybersecurity Tools**
- Help individuals verify suspicious messages.
- **Educational Tools**
- Teach users how phishing content looks.
- **Add-on for Email Clients**
- Could be integrated into Gmail, Outlook, Thunderbird

7.FUTURE ENHANCEMENTS

- Planned Upgrades:
- Use **deep learning models** (**LSTM, BERT**) for more accuracy.
- Analyze **email headers, sender metadata**, and URLs.
- Support for **real-time email scraping** from inbox.
- Deploy as a **web app or browser extension.**

8. DATASET OVERVIEW

- CSV file (`phishing.csv`) with
  - `EmailText`: raw email content.
  - `Label`: 0 = Legitimate, 1 = Phishing.
- Cleaned using:
  - Lowercasing.
  - Punctuation removal.
- Split into **training** (**80%**) and **testing** (**20%**) data.

9. GUI OVERVIEW (TKINTER)

- Desktop app built using `tkinter`.
- Allows user to paste email text.
- Clicks "🧠 **Analyze Email**" to classify.
- Shows:
- Prediction: **Phishing** ⚠️ or **Legitimate** ✅
- Confidence %
- Character count

10. GUI FEATURES

- Clean modern layout .
- Real-time feedback without lag.
- Labels:

  1.Input email box

  2.Prediction result

  3.Character counter

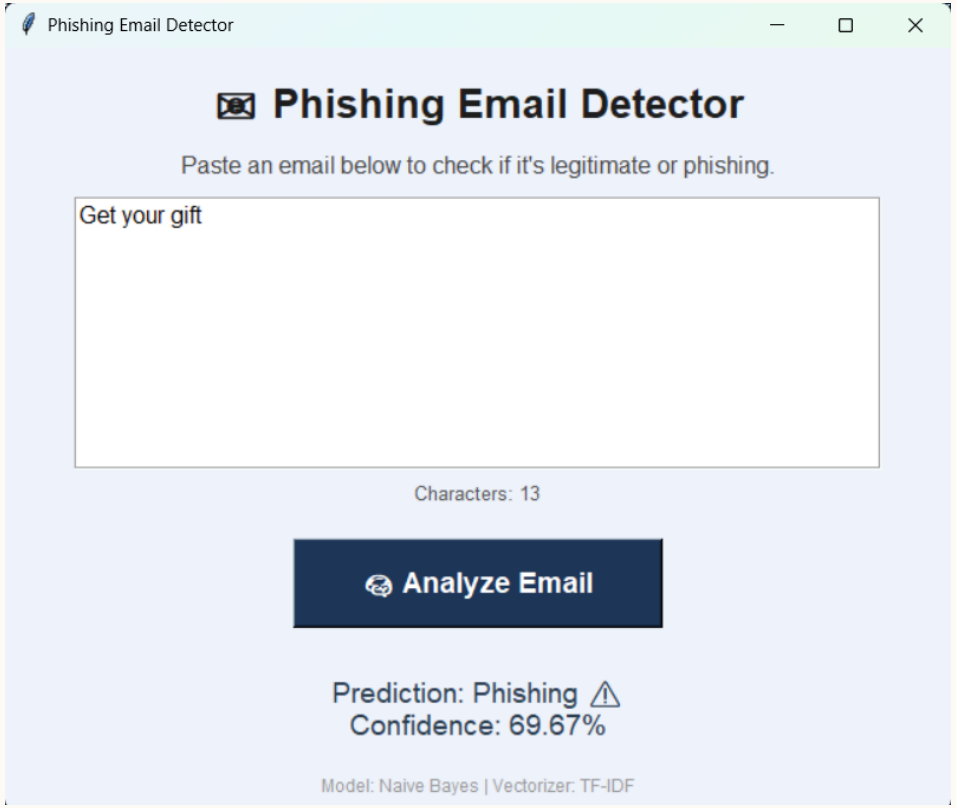
  4.Footer info: model + vectorizer

11.CODE TOOL BREAKDOWN


```python
import pandas as pd
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score
import tkinter as tk
from tkinter import messagebox

# Load dataset
df = pd.read_csv("phishing.csv", encoding='utf-8')
df.columns = df.columns.str.strip()
df.dropna(inplace=True)

# Text preprocessing
def clean_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text

df['EmailText'] = df['EmailText'].astype(str).apply(clean_text)
X = df['EmailText']
y = df['Label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

vectorizer = TfidfVectorizer()
X_train_vectors = vectorizer.fit_transform(X_train)
X_test_vectors = vectorizer.transform(X_test)

model = MultinomialNB()
model.fit(X_train_vectors, y_train)

y_pred = model.predict(X_test_vectors)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nReport:\n", classification_report(y_test, y_pred))

# GUI Functions
def predict_email():
```

```python
# GUI Functions
def predict_email():
    email = email_input.get("1.0", "end-1c").strip()
    char_count_label.config(text=f"Characters: {len(email)}")

    if email == '':
        messagebox.showwarning("Input Error", "Please enter email text to classify.")
        result_label.config(text="Prediction: –", fg="#2c3e50")
        return

    email_clean = clean_text(email)
    email_vector = vectorizer.transform([email_clean])
    result = model.predict(email_vector)
    prob = model.predict_proba(email_vector).max()

    prediction = "Phishing ⚠" if result[0] == 1 else "Legitimate ☑"
    result_label.config(text=f"Prediction: {prediction}\nConfidence: {round(prob*100, 2)}%", fg="#2c3e50")

# GUI Setup
root = tk.Tk()
root.title("Phishing Email Detector")
root.geometry("640x520")
root.config(bg="#edf2fb")

# Title
title_label = tk.Label(root, text="📧 Phishing Email Detector", font=("Helvetica", 20, "bold"), bg="#edf2fb", fg="#1c1c1c")
title_label.pack(pady=(20, 10))

# Instruction
instruction = tk.Label(root, text="Paste an email below to check if it's legitimate or phishing.", font=("Helvetica", 12), bg="#edf2fb", fg="#444
instruction.pack()

# Input Box
email_input = tk.Text(root, height=10, width=60, font=("Helvetica", 12), wrap="word", bd=2, relief="groove")
email_input.pack(pady=(10, 5))
```

12. DEMO (SCREENSHOTS)




```
Accuracy: 0.6666666666666666

Report:
              precision    recall  f1-score   support

           0       1.00      0.33      0.50         3
           1       0.60      1.00      0.75         3

    accuracy                           0.67         6
   macro avg       0.80      0.67      0.62         6
weighted avg       0.80      0.67      0.62         6
```