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


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


def predict_spam():
    message = text_entry.get("1.0", tk.END).strip()
    if message == "":
        messagebox.showwarning("Input Error", "Please enter a message!")
        return
    vector = vectorizer.transform([message])
    prediction = model.predict(vector)[0]
    messagebox.showinfo("Result", f"The message is: {prediction.upper()}")


root = tk.Tk()
root.title("Spam Detection")
root.geometry("400x300")
tk.Label(root, text="Enter your message:", font=("Arial", 14)).pack(pady=10)
text_entry = tk.Text(root, height=5, width=40)
text_entry.pack(pady=5)
tk.Button(root, text="Check Spam", command=predict_spam, font=("Arial", 12)).pack(pady=20)

root.mainloop()
```

[ ]:

```
    'Message': [
        'Congratulations! You won a free ticket',
        'Call me later',
        'You have been selected for a $1000 gift card',
        'Hello, how are you?',
        'Win a free iPhone now',
        'Meeting at 5 PM',
        'Get cheap loans now',
        'Are you coming today?'
    ],
    'Label': ['spam', 'ham', 'spam', 'ham', 'spam', 'ham', 'spam', 'ham']
}

df = pd.DataFrame(data)


X = df['Message']
y = df['Label']

vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2, random_state=42)


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


def predict_spam():
    message = text_entry.get("1.0", tk.END).strip()
    if message == "":
        messagebox.showwarning("Input Error", "Please enter a message!")
        return
    vector = vectorizer.transform([message])
    prediction = model.predict(vector)[0]
    messagebox.showinfo("Result", f"The message is: {prediction.upper()}")
```

```
    prediction = model.predict(vector)[0]
    if prediction == 1:
        messagebox.showinfo("Result", "The message is: SPAM")
    else:
        messagebox.showinfo("Result", "The message is: HAM (Not Spam)")


root = tk.Tk()
root.title("Spam Detection")
root.geometry("500x350")

tk.Label(root, text="Enter your message to check spam:", font=("Arial", 14)).pack(pady=10)
text_entry = tk.Text(root, height=7, width=50)
text_entry.pack(pady=10)
tk.Button(root, text="Check Spam", command=predict_spam, font=("Arial", 12), bg="lightblue").pack(pady=20)

root.mainloop()
```

[ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
import tkinter as tk
from tkinter import messagebox


data = {
    'Message': [
        'Congratulations! You won a free ticket',
        'Call me later',
        'You have been selected for a $1000 gift card',
        'Hello, how are you?'
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
import tkinter as tk
from tkinter import messagebox
from tkinter import filedialog

try:
    df = pd.read_csv('spam.csv', encoding='latin-1')[['v1', 'v2']]
except FileNotFoundError:
    messagebox.showerror("File Error", "spam.csv file not found in the folder!")
    exit()

df = df.rename(columns={'v1': 'Label', 'v2': 'Message'})
df['Label'] = df['Label'].map({'ham': 0, 'spam': 1})  # 0 = ham, 1 = spam


X = df['Message']
y = df['Label']

vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2, random_state=42)

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

def predict_spam():
    message = text_entry.get("1.0", tk.END).strip()
    if message == "":
        messagebox.showwarning("Input Error", "Please enter a message!")
        return
    vector = vectorizer.transform([message])
    prediction = model.predict(vector)[0]
    if prediction == 1:
        messagebox.showinfo("Result", "The message is: SPAM")
```

```
        with out:
            out.clear_output()
            print("Error:", e)

train_btn.on_click(train_model)


def predict_new(btn):
    global model
    if model is None:
        with out:
            out.clear_output()
            print("Please train the model first!")
        return
    try:
        input_values = [float(x) for x in input_text.value.split(',')]
        prediction = model.predict([input_values])
        with out:
            out.clear_output()
            print(f"The predicted risk class is: {prediction[0]}")
    except Exception as e:
        with out:
            out.clear_output()
            print("Error:", e)

predict_btn.on_click(predict_new)

VBox([upload, train_btn, HBox([input_text, predict_btn]), out])
```

```
            plt.figure(figsize=(6,5))
            sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
            plt.xlabel("Predicted")
            plt.ylabel("Actual")
            plt.title("Confusion Matrix")
            plt.show()


            plt.figure(figsize=(8,5))
            importances = model.feature_importances_
            feature_names = X.columns
            sns.barplot(x=importances, y=feature_names)
            plt.title("Feature Importance")
            plt.show()

    except Exception as e:
        with out:
            out.clear_output()
            print("Error:", e)

train_btn.on_click(train_model)


def predict_new(btn):
    global model
    if model is None:
        with out:
            out.clear_output()
            print("Please train the model first!")
        return
    try:
        input_values = [float(x) for x in input_text.value.split(',')]
        prediction = model.predict([input_values])
        with out:
            out.clear_output()
            print(f"The predicted risk class is: {prediction[0]}")
```

```python
def train_model(btn):
    global model, df
    if df is None:
        with out:
            out.clear_output()
            print("Please upload a dataset first!")
        return
    try:
        X = df.drop('Risk', axis=1)
        y = df['Risk']


        for col in X.select_dtypes(include='object').columns:
            X[col] = LabelEncoder().fit_transform(X[col])
        if y.dtype == 'object':
            y = LabelEncoder().fit_transform(y)


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


        model = RandomForestClassifier()
        model.fit(X_train, y_train)


        y_pred = model.predict(X_test)
        report = classification_report(y_test, y_pred)
        cm = confusion_matrix(y_test, y_pred)


        with out:
            out.clear_output()
            print("Model Trained Successfully!\n")
            print("Classification Report:\n", report)


            plt.figure(figsize=(6,5))
            sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
            plt.xlabel("Predicted")
```

[ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from ipywidgets import Button, FileUpload, VBox, HBox, Text, Output


%matplotlib inline


df = None
model = None


upload = FileUpload(accept=".csv", multiple=False)
train_btn = Button(description="Train Model", button_style='success')
predict_btn = Button(description="Predict", button_style='info')
input_text = Text(description="New Input (comma sep):")
out = Output()


def load_dataset(change):
    global df
    if upload.value:
        for name, file_info in upload.value.items():
            df = pd.read_csv(file_info['metadata']['name'])
            with out:
                out.clear_output()
                print("Dataset Loaded Successfully!")
                display(df.head())

upload.observe(load_dataset, names='value')


def train_model(btn):
```

```python
import pytesseract
from PIL import Image, ImageTk
import cv2
import tkinter as tk
from tkinter import filedialog, messagebox


pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

img_path = None

def run_gui():
    global img_path

    def select_image():
        global img_path
        img_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg *.png *.jpeg")])
        if img_path:
            image = Image.open(img_path)
            image = image.resize((250, 250))
            img_display = ImageTk.PhotoImage(image)
            lbl_image.config(image=img_display)
            lbl_image.image = img_display
            txt_output.delete(1.0, tk.END)

    def extract_text():
        if not img_path:
            messagebox.showerror("Error", "Please select an image first!")
            return
        image = cv2.imread(img_path)
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        _, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)
        text = pytesseract.image_to_string(thresh)
        txt_output.delete(1.0, tk.END)
        txt_output.insert(tk.END, text)

    root = tk.Tk()
    root.title("Handwritten to Text Converter")
```

```
root = tk.Tk()
root.title("Handwritten to Text Converter")


lbl_image = tk.Label(root)
lbl_image.pack(pady=10)

btn_select = tk.Button(root, text="Select Image", command=select_image)
btn_select.pack(pady=5)

btn_extract = tk.Button(root, text="Extract Text", command=extract_text)
btn_extract.pack(pady=5)


txt_output = tk.Text(root, height=10, width=50)
txt_output.pack(pady=10)


img_path = None

root.mainloop()
```

```
Requirement already satisfied: Pillow>=8.0.0 in c:\users\ayush\appdata\local\programs\python\python313\lib\site-packages (fro
Requirement already satisfied: opencv-python in c:\users\ayush\appdata\local\programs\python\python313\lib\site-packages (4.1
Requirement already satisfied: numpy<2.3.0,>=2 in c:\users\ayush\appdata\local\programs\python\python313\lib\site-packages (f
6)
Requirement already satisfied: pillow in c:\users\ayush\appdata\local\programs\python\python313\lib\site-packages (10.4.0)
```

```
!pip install pytesseract
!pip install opencv-python
!pip install pillow
```

```
import pytesseract
from PIL import Image, ImageTk
import cv2
import tkinter as tk
from tkinter import filedialog, messagebox
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe
def select_image():
    global img_path
    img_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg *.png *.jpeg")])
    if img_path:
        image = Image.open(img_path)
        image = image.resize((250, 250))
        img_display = ImageTk.PhotoImage(image)
        lbl_image.config(image=img_display)
        lbl_image.image = img_display
        txt_output.delete(1.0, tk.END)
def extract_text():
    if not img_path:
        messagebox.showerror("Error", "Please select an image first!")
        return
    image = cv2.imread(img_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)


    text = pytesseract.image_to_string(thresh)
    txt_output.delete(1.0, tk.END)
    txt_output.insert(tk.END, text)
```

```
Collecting pytesseract
  Downloading pytesseract-0.3.13-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: packaging>=21.3 in c:\users\ayush\appdata\local\programs\python\python313\lib\site-packages (from pytesseract) (25.0)
Requirement already satisfied: Pillow>=8.0.0 in c:\users\ayush\appdata\local\programs\python\python313\lib\site-packages (from pytesseract) (10.4.0)
Downloading pytesseract-0.3.13-py3-none-any.whl (14 kB)
Installing collected packages: pytesseract
Successfully installed pytesseract-0.3.13
Collecting opencv-python
  Downloading opencv_python-4.12.0.88-cp37-abi3-win_amd64.whl.metadata (19 kB)
Collecting numpy<2.3.0,>=2 (from opencv-python)
  Downloading numpy-2.2.6-cp313-cp313-win_amd64.whl.metadata (60 kB)
Downloading opencv_python-4.12.0.88-cp37-abi3-win_amd64.whl (39.0 MB)
   ---------------------------------------- 0.0/39.0 MB ? eta -:--:--
   ---------------------------------------- 0.3/39.0 MB ? eta -:--:--
   ---------------------------------------- 0.3/39.0 MB ? eta -:--:--
    --------------------------------------- 0.8/39.0 MB 1.7 MB/s eta 0:00:23
   - -------------------------------------- 1.6/39.0 MB 2.3 MB/s eta 0:00:17
   -- ------------------------------------- 2.1/39.0 MB 2.3 MB/s eta 0:00:16
   -- ------------------------------------- 2.9/39.0 MB 2.6 MB/s eta 0:00:14
   --- ------------------------------------ 3.4/39.0 MB 2.7 MB/s eta 0:00:14
   ---- ----------------------------------- 4.2/39.0 MB 2.7 MB/s eta 0:00:13
   ---- ----------------------------------- 4.7/39.0 MB 2.8 MB/s eta 0:00:13
   ----- ---------------------------------- 5.2/39.0 MB 2.7 MB/s eta 0:00:13
   ------ --------------------------------- 6.3/39.0 MB 2.9 MB/s eta 0:00:12
   ------- -------------------------------- 7.1/39.0 MB 3.0 MB/s eta 0:00:11
   ------- -------------------------------- 7.6/39.0 MB 3.0 MB/s eta 0:00:11
   -------- ------------------------------- 8.1/39.0 MB 2.9 MB/s eta 0:00:11
   -------- ------------------------------- 8.7/39.0 MB 3.0 MB/s eta 0:00:11
   --------- ------------------------------ 9.7/39.0 MB 3.1 MB/s eta 0:00:10
   ---------- ----------------------------- 10.7/39.0 MB 3.2 MB/s eta 0:00:09
   ---------- ----------------------------- 11.5/39.0 MB 3.2 MB/s eta 0:00:09
   ----------- ---------------------------- 12.3/39.0 MB 3.3 MB/s eta 0:00:09
   ------------ --------------------------- 12.8/39.0 MB 3.2 MB/s eta 0:00:09
   ------------ --------------------------- 13.4/39.0 MB 3.2 MB/s eta 0:00:09
   ------------- -------------------------- 13.9/39.0 MB 3.2 MB/s eta 0:00:08
   -------------- ------------------------- 14.7/39.0 MB 3.2 MB/s eta 0:00:08
   -------------- ------------------------- 15.5/39.0 MB 3.2 MB/s eta 0:00:08
   --------------- ------------------------ 15.7/39.0 MB 3.2 MB/s eta 0:00:08
   --------------- ------------------------ 16.3/39.0 MB 3.1 MB/s eta 0:00:08
   ---------------- ----------------------- 17.0/39.0 MB 3.1 MB/s eta 0:00:08
   ---------------- ----------------------- 17.3/39.0 MB 3.1 MB/s eta 0:00:08
```

```
!pip install pytesseract
!pip install opencv-python
!pip install pillow



import pytesseract
from PIL import Image
import cv2


pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'


image_path = 'handwritten_sample.jpg'
image = cv2.imread(image_path)


gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)


_, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)

text = pytesseract.image_to_string(thresh)


print("Recognized Text:")
print(text)
```