อธิบายการทำงานของโค้ด

1. การนำเข้าไลบรารี:
   - โค้ดนำเข้าไลบรารีต่างๆ ที่จำเป็น เช่น tkinter สำหรับ GUI, pandas สำหรับการจัดการข้อมูล, TfidfVectorizer สำหรับการแปลงข้อความเป็นเวกเตอร์, และ MultinomialNB สำหรับการสร้างโมเดล Naive Bayes
2. ฟังก์ชัน train_spam_filter() สำหรับการฝึกสอนโมเดล:
   - โหลดชุดข้อมูล SMS Spam Collection ผ่าน URL
   - ทำการแปลงข้อความในคอลัมน์ v1 โดยเปลี่ยนค่าข้อความ "spam" เป็น 1 และ "ham" เป็น 0
   - แบ่งข้อมูลออกเป็นชุดฝึกสอน (80%) และชุดทดสอบ (20%) ด้วยฟังก์ชัน train_test_split()
   - ใช้ TF-IDF Vectorizer ในการแปลงข้อความให้เป็นเวกเตอร์เชิงตัวเลข โดยคำนวณความถี่ของคำในข้อความ
   - สร้างโมเดล Naive Bayes ด้วย MultinomialNB() และฝึกสอนโมเดลด้วยข้อมูลฝึกสอน
   - ทดสอบความแม่นยำของโมเดลด้วยข้อมูลทดสอบ และแสดงผลค่าความแม่นยำ (Accuracy)
3. ฟังก์ชัน scan_spam() สำหรับการตรวจสอบข้อความสแปม:
   - ดึงข้อความจากช่อง message_entry ที่ผู้ใช้ป้อนใน GUI
   - แปลงข้อความที่ป้อนเป็นเวกเตอร์ด้วย TF-IDF ที่ถูกฝึกสอนแล้ว
   - ใช้โมเดลที่ฝึกสอนแล้วทำการทำนายว่าข้อความนั้นเป็นสแปมหรือไม่ โดยการใช้ model.predict()
   - แสดงผลลัพธ์ใน result_text โดยจะแจ้งว่าข้อความเป็นสแปมหรือไม่
4. ฟังก์ชัน clear_results() สำหรับการล้างผลลัพธ์:
   - ล้างข้อความทั้งหมดที่แสดงใน result_text เพื่อเตรียมสำหรับการตรวจสอบข้อความใหม่
5. ฟังก์ชัน show_about() สำหรับการแสดงข้อมูลโปรแกรม:
   - เมื่อผู้ใช้เลือกเมนู "About" ระบบจะแสดงหน้าต่างป๊อปอัพที่มีข้อมูลเกี่ยวกับโปรแกรมกรองสแปม
6. สร้าง GUI ด้วย Tkinter:
   - สร้างหน้าต่างหลักของโปรแกรมด้วย Tk()
   - มีการสร้างกรอบข้อความสำหรับให้ผู้ใช้ป้อนข้อความที่ต้องการตรวจสอบ
   - เพิ่มปุ่ม "ตรวจสอบข้อความ" สำหรับสั่งให้ระบบตรวจสอบว่าเป็นสแปมหรือไม่
   - เพิ่มกรอบแสดงผลลัพธ์ และปุ่ม "ล้างผลลัพธ์" เพื่อให้ผู้ใช้สามารถลบผลลัพธ์เดิมได้
   - เพิ่มเมนู "Help" ซึ่งมีตัวเลือก "About" เพื่อแสดงข้อมูลของโปรแกรม

สรุปขั้นตอนการทำงาน:

1. ผู้ใช้ป้อนข้อความในช่องที่กำหนด
2. เมื่อกดปุ่ม "ตรวจสอบข้อความ" ระบบจะใช้โมเดล Naive Bayes ที่ฝึกสอนแล้วทำการทำนายว่าข้อความเป็นสแปมหรือไม่
3. ผลลัพธ์จะแสดงในหน้าต่าง GUI ว่าข้อความนั้นเป็นสแปมหรือไม่
4. ผู้ใช้สามารถกดปุ่ม "ล้างผลลัพธ์" เพื่อล้างข้อความที่แสดงผล

---

1. การนำเข้าไลบรารีที่จำเป็น:

```
import tkinter as tk

from tkinter import messagebox

import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, classification_report
```

ไลบรารีที่ใช้ในการสร้าง GUI และสร้างโมเดลกรองสแปม:

---

2. ฟังก์ชันสำหรับการประมวลผลและฝึกสอนโมเดล train_spam_filter()

```
def train_spam_filter():
    # โหลดข้อมูล (ใช้ชุดข้อมูล SMS Spam Collection)
    url = "https://raw.githubusercontent.com/mohitgupta-omg/Kaggle-SMS-spam-collection-dataset-/master/spam.csv"
    data = pd.read_csv(url, encoding='latin-1')

    # แปลงข้อความสแปมเป็น 1 และไม่เป็นสแปมเป็น 0
    data['v1'] = data['v1'].map({'spam': 1, 'ham': 0})

    # แยกข้อมูลเป็นชุดฝึกสอนและชุดทดสอบ
    X_train, X_test, y_train, y_test = train_test_split(data['v2'], data['v1'], test_size=0.2, random_state=42)

    # ทำการแปลงข้อความเป็นเวกเตอร์ด้วย TF-IDF
    vectorizer = TfidfVectorizer(stop_words='english')
    X_train_tfidf = vectorizer.fit_transform(X_train)
    X_test_tfidf = vectorizer.transform(X_test)

    # สร้างโมเดล Naive Bayes
    model = MultinomialNB()
    model.fit(X_train_tfidf, y_train)

    # ทดสอบโมเดล
    y_pred = model.predict(X_test_tfidf)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.4f}")
```

```
    return model, vectorizer
```

- โหลดข้อมูลจาก URL
- แปลงข้อความให้เป็นสแปม (1) หรือไม่เป็นสแปม (0)
- แยกข้อมูลฝึกสอนและข้อมูลทดสอบ
- แปลงข้อความเป็นเวกเตอร์ด้วย TF-IDF
- สร้างโมเดล Naive Bayes และทำการทดสอบ

---

3. ฝึกสอนโมเดล (เรียกใช้ฟังก์ชัน train_spam_filter())

```
# ฝึกสอนโมเดล

model, vectorizer = train_spam_filter()
```

- ฝึกสอนโมเดลกรองสแปมโดยเรียกฟังก์ชัน train_spam_filter() เพื่อฝึกโมเดลและคืนค่ากลับเป็น model และ vectorizer

---

4. ฟังก์ชันสำหรับการสแกนข้อความสแปม scan_spam()

```
def scan_spam():
    message = message_entry.get("1.0", tk.END).strip()

    if not message:
        result_text.insert(tk.END, "กรุณาใส่ข้อความที่ต้องการตรวจสอบ\n")
        return

    # แปลงข้อความเป็น TF-IDF เวกเตอร์
    message_tfidf = vectorizer.transform([message])

    # ทำนายว่าข้อความเป็นสแปมหรือไม่
    prediction = model.predict(message_tfidf)

    # แสดงผลลัพธ์
    if prediction == 1:
        result_text.insert(tk.END, "ข้อความนี้เป็นสแปม\n")
    else:
        result_text.insert(tk.END, "ข้อความนี้ไม่เป็นสแปม\n")
```

- อ่านข้อมูลจากผู้ใช้ใน message_entry
- แปลงข้อความให้เป็นเวกเตอร์ด้วย TF-IDF
- ใช้โมเดลที่ฝึกสอนแล้วทำการทำนายว่าข้อความเป็นสแปมหรือไม่
- แสดงผลลัพธ์ในหน้าต่าง GUI

---

5. ฟังก์ชันสำหรับการล้างผลลัพธ์ clear_results()

```
def clear_results():
    result_text.delete("1.0", tk.END)
```

- ล้างข้อความทั้งหมดในกรอบผลลัพธ์เพื่อเตรียมสำหรับการตรวจสอบครั้งใหม่

---

6. ฟังก์ชันแสดงข้อมูลโปรแกรม show_about()

```
def show_about():
    messagebox.showinfo("About", "โปรแกรมนี้เป็นเครื่องมือสำหรับการกรองข้อความสแปม")
```

- แสดงข้อความเกี่ยวกับโปรแกรมเมื่อผู้ใช้กดเมนู "About"

---

7. การสร้าง GUI ด้วย Tkinter

```
root = tk.Tk()

root.title("Spam Filter Tool")

root.geometry("600x400")


# กรอบข้อความ

message_frame = tk.Frame(root)

message_frame.pack(pady=10)


message_label = tk.Label(message_frame, text="ใส่ข้อความที่ต้องการตรวจสอบ:")

message_label.pack()


message_entry = tk.Text(message_frame, height=5, width=50)

message_entry.pack()


scan_spam_button = tk.Button(message_frame, text="ตรวจสอบข้อความ",
command=scan_spam)

scan_spam_button.pack(pady=5)


# แสดงผลลัพธ์การสแกน

result_frame = tk.Frame(root)

result_frame.pack(pady=10)


result_label = tk.Label(result_frame, text="ผลการสแกน:")

result_label.pack()


result_text = tk.Text(result_frame, height=10, width=70)
```

```
result_text.pack()


# ปุ่มล้างผลลัพธ์

clear_button = tk.Button(root, text="ล้างผลลัพธ์", command=clear_results)

clear_button.pack(pady=5)


# เมนู About

menu = tk.Menu(root)

root.config(menu=menu)


help_menu = tk.Menu(menu)

menu.add_cascade(label="Help", menu=help_menu)

help_menu.add_command(label="About", command=show_about)


root.mainloop()
```

- สร้างอินเทอร์เฟซหลักของโปรแกรมด้วย Tkinter
- มีส่วนให้ผู้ใช้ป้อนข้อความและปุ่มสำหรับตรวจสอบว่าข้อความเป็นสแปมหรือไม่
- มีกรอบแสดงผลลัพธ์การตรวจสอบ และปุ่มล้างผลลัพธ์
- เพิ่มเมนู "Help" ที่มีข้อมูลเกี่ยวกับโปรแกรม

สรุป:

- ขั้นตอนการทำงานเริ่มต้นด้วยการฝึกสอนโมเดลโดยใช้ข้อมูล SMS Spam Collection จากนั้นผู้ใช้สามารถป้อนข้อความใน GUI เพื่อทำการตรวจสอบว่าเป็นสแปมหรือไม่ ผลลัพธ์จะแสดงในหน้าต่าง GUI