import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.pipeline import make\_pipeline

import random

tasks = pd.DataFrame(columns=['description', 'priority'])

try:

    tasks = pd.read\_csv('tasks.csv')

except FileNotFoundError:

    pass

tasks.dropna(inplace=True)

vectorizer = CountVectorizer()

clf = MultinomialNB()

model = make\_pipeline(vectorizer, clf)

if not tasks.empty:

    model.fit(tasks['description'], tasks['priority'])

def add\_task(description, priority):

    global tasks  # Declare tasks as a global variable

    new\_task = pd.DataFrame({'description': [description], 'priority': [priority]})

    tasks = pd.concat([tasks, new\_task], ignore\_index=True)

    tasks.dropna(inplace=True)  # Drop any NaNs after adding a new task

    save\_tasks()

def remove\_task(description):

    global tasks  # Declare tasks as a global variable

    tasks = tasks[tasks['description'] != description]

    save\_tasks()

def list\_tasks():

    if tasks.empty:

        print("No tasks available.")

    else:

        print(tasks)

def recommend\_task():

    if not tasks.empty:

        # Get high-priority tasks

        high\_priority\_tasks = tasks[tasks['priority'] == 'High']

        if not high\_priority\_tasks.empty:

            # Choose a random high-priority task

            random\_task = random.choice(high\_priority\_tasks['description'].values)

            print(f"Recommended task: {random\_task} - Priority: High")

        else:

            print("No high-priority tasks available for recommendation.")

    else:

        print("No tasks available for recommendations.")

# Main menu

while True:

    print("\nTask Management App")

    print("1. Add Task")

    print("2. Remove Task")

    print("3. List Tasks")

    print("4. Recommend Task")

    print("5. Exit")

    choice = input("Select an option: ")

    if choice == "1":

        description = input("Enter task description: ")

        priority = input("Enter task priority (Low/Medium/High): ").capitalize()

        add\_task(description, priority)

        print("Task added successfully.")

    elif choice == "2":

        description = input("Enter task description to remove: ")

        remove\_task(description)

        print("Task removed successfully.")

    elif choice == "3":

        list\_tasks()

    elif choice == "4":

        recommend\_task()

    elif choice == "5":

        print("Goodbye!")

        break

    else:

        print("Invalid option. Please select a valid option.")

OUTPUT:

Task Management App

1. Add Task

2. Remove Task

3. List Tasks

4. Recommend Task

5. Exit

Task added successfully.

Task Management App

1. Add Task

2. Remove Task

3. List Tasks

4. Recommend Task

5. Exit

Task added successfully.

Task Management App

1. Add Task

2. Remove Task

3. List Tasks

4. Recommend Task

5. Exit

Task removed successfully.

...

3. List Tasks

4. Recommend Task

5. Exit

Goodbye!

*Output is truncated. View as a* [*scrollable element*](command:cellOutput.enableScrolling?3f63abde-9604-4e5c-99c9-29c1d85a74e8) *or open in a* [*text editor*](command:workbench.action.openLargeOutput?3f63abde-9604-4e5c-99c9-29c1d85a74e8)*. Adjust cell output* [*settings*](command:workbench.action.openSettings?%5B%22%40tag%3AnotebookOutputLayout%22%5D)*...*