import tkinter as tk

from tkinter import ttk, messagebox

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import linear\_kernel

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from PIL import Image, ImageTk

# Load the dataset

books\_data = pd.read\_csv('C:\\Users\\Nadir\\Downloads\\books.csv')

# Preprocess the data

books\_data['title'] = books\_data['title'].fillna('')

books\_data['authors'] = books\_data['authors'].fillna('')

books\_data['Genres'] = books\_data['Genres'].fillna('')

# Combine relevant information into a single column

books\_data['combined\_features'] = books\_data['title'] + ' ' + books\_data['authors'] + ' ' + books\_data['Genres']

# Use TfidfVectorizer to convert the combined features into numerical data

tfidf\_vectorizer = TfidfVectorizer(stop\_words='english')

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(books\_data['combined\_features'])

# Compute the cosine similarity between books

cosine\_sim = linear\_kernel(tfidf\_matrix, tfidf\_matrix)

# Initialize liked column with 0

books\_data['liked'] = 0

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(tfidf\_matrix, books\_data['liked'], test\_size=0.2, random\_state=42)

# Train a k-Nearest Neighbors classifier

knn\_classifier = KNeighborsClassifier(n\_neighbors=5)

knn\_classifier.fit(X\_train, y\_train)

def predict\_like\_knn(book\_title, author\_name, genre):

input\_features = tfidf\_vectorizer.transform([f'{book\_title} {author\_name} {genre}'])

prediction = knn\_classifier.predict(input\_features)

return prediction[0]

# Function to get book recommendations

def get\_recommendations\_knn(book\_title, author\_name, genre, cosine\_sim\_matrix, data):

if author\_name:

# Filter data based on the specified author

filtered\_data = data[data['authors'].str.contains(author\_name, case=False)]

if filtered\_data.empty:

return pd.DataFrame()

else:

# If no author is specified, consider all books

filtered\_data = data

if book\_title:

# If book title is specified, further filter based on title

filtered\_data = filtered\_data[filtered\_data['title'].str.contains(book\_title, case=False)]

if genre:

# If genre is specified, further filter based on genre

genres\_list = genre.split(', ')

for genre in genres\_list:

filtered\_data = filtered\_data[filtered\_data['Genres'].str.contains(genre, case=False)]

if filtered\_data.empty:

return pd.DataFrame()

# Get the pairwise similarity scores of all books

sim\_scores = cosine\_sim\_matrix[filtered\_data.index]

# Sort the books based on the similarity scores

rankings = pd.Series(sim\_scores.sum(axis=1), name='rankings', index=filtered\_data.index)

rankings = rankings.sort\_values(ascending=False)

# Return the top 10 most similar books along with additional information

recommendations = data.loc[rankings.index[:10], ['id', 'original\_publication\_year', 'average\_rating', 'title', 'authors', 'Genres']]

return recommendations

# GUI classes

class BookRecommendationGUI:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Book Recommendation System")

self.root.geometry("600x470")

bg\_image = Image.open("bg3.jpeg")

bg\_image = bg\_image.resize((600, 470))

self.bg\_photo = ImageTk.PhotoImage(bg\_image)

self.bg\_label = ttk.Label(root, image=self.bg\_photo)

self.bg\_label.place(relwidth=1, relheight=1)

# self.root.configure(bg="#E6E6FA")

self.title\_label = ttk.Label(root, text="Book Recommendation System",

font=('Helvetica', 24, 'bold'), background="#E6E6FA", foreground="#191970")

self.title\_label.pack(pady=20)

self.book\_label = ttk.Label(root, text="Enter Book Title:", font=('Helvetica', 14), background="#E6E6FA")

self.book\_entry = ttk.Entry(root, width=40, font=('Helvetica', 12))

self.book\_label.pack(pady=10)

self.book\_entry.pack(pady=10)

self.author\_label = ttk.Label(root, text="Enter Author Name:", font=('Helvetica', 14), background="#E6E6FA")

self.author\_entry = ttk.Entry(root, width=40, font=('Helvetica', 12))

self.author\_label.pack(pady=10)

self.author\_entry.pack(pady=10)

self.genre\_label = ttk.Label(root, text="Enter Genres (comma-separated):", font=('Helvetica', 14),

background="#E6E6FA")

self.genre\_entry = ttk.Entry(root, width=40, font=('Helvetica', 12))

self.genre\_label.pack(pady=10)

self.genre\_entry.pack(pady=10)

self.next\_button = ttk.Button(root, text="Recommend Books", command=self.open\_recommendation\_gui,

style="Accent.TButton")

self.next\_button.pack(pady=20)

style = ttk.Style()

style.configure("TButton", padding=10, font=('Helvetica', 14))

style.configure("Accent.TButton", background="#8A2BE2", foreground="#0d0d0d", padding=10,

font=('Helvetica', 14, 'bold'))

def open\_recommendation\_gui(self):

book\_title = self.book\_entry.get()

author\_name = self.author\_entry.get()

genre = self.genre\_entry.get()

if book\_title or author\_name or genre:

self.root.destroy()

BookRecommendationResultGUI(book\_title, author\_name, genre)

else:

messagebox.showinfo("Error", "Please enter a book title, author name, or genre.")

class BookRecommendationResultGUI:

def \_\_init\_\_(self, book\_title, author\_name, genre):

self.book\_title = book\_title

self.author\_name = author\_name

self.genre = genre

self.root = tk.Tk()

self.root.title("Book Recommendations")

self.root.geometry("900x600")

self.root.configure(bg="#E6E6FA")

self.result\_label = ttk.Label(self.root, text="Recommendations:", font=('Helvetica', 24, 'bold'),

background="#E6E6FA", foreground="#191970")

self.result\_label.pack(pady=20)

columns = ("Book Title", "Original PublicationYear", "Average Rating", "Authors", "Genres")

self.result\_treeview = ttk.Treeview(self.root, columns=columns, show="headings", height=20)

for col in columns:

self.result\_treeview.heading(col, text=col)

self.result\_treeview.column(col, width=150)

self.show\_recommendations()

vertical\_scrollbar = ttk.Scrollbar(self.root, orient="vertical", command=self.result\_treeview.yview)

vertical\_scrollbar.pack(side="right", fill="y")

horizontal\_scrollbar = ttk.Scrollbar(self.root, orient="horizontal", command=self.result\_treeview.xview)

horizontal\_scrollbar.pack(side="bottom", fill="x")

self.result\_treeview.configure(yscrollcommand=vertical\_scrollbar.set, xscrollcommand=horizontal\_scrollbar.set)

self.result\_treeview.pack(pady=20, ipadx=10, ipady=10, fill=tk.BOTH, expand=True)

def show\_recommendations(self):

recommendations = get\_recommendations\_knn(self.book\_title, self.author\_name, self.genre, cosine\_sim, books\_data)

if not recommendations.empty:

for \_, row in recommendations.iterrows():

self.result\_treeview.insert("", "end", values=(

row["title"], row["original\_publication\_year"], row["average\_rating"], row["authors"],

row["Genres"]))

else:

messagebox.showinfo("No Recommendations",

"No book recommendations found for the given title, author name, and genre.")

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

root.resizable(False, False)

app = BookRecommendationGUI(root)

root.mainloop()