Final Project Report
Advanced Computer Programming

Movie Recommender

Group : 4

Instructor : DINH-TRUNG VU

2025-06

Chapter 1 Introduction

1.1 Group Information

1) **Group Project Repository**: https://github.com/113021198/ACP/tree/main/final
2) **Group members**:
    1. Radian Try Darmawan– 113021220
    2. Dimas Ahmad Fahreza– 113021199
    3. Wirandito Sarwono - 113021198

1.2 Overview

Our application reads Netflix data into a Pandas DataFrame (*pd.read_csv*) and uses ***pandas.apply*** with ***lambda*** functions to filter genres, ratings, and years. We persist the user's watchlist to disk in JSON format ***(json.load / json.dump)*** and use ***os.path.exists*** to manage file checks. For recommendations, we apply ***MultiLabelBinarizer*** to one-hot encode genres, ***TfidfVectorizer*** to turn plot summaries into sparse text vectors, and stack these with release-year data using SciPy's ***csr_matrix*** and ***hstack***. A ***NearestNeighbors*** model with cosine distance finds five similar movies. The GUI is built with Tkinter and ttk widgets, including *Listbox*, *Combobox*, *Spinbox*, and *ScrolledText*, and embeds Matplotlib charts via *FigureCanvasTkAgg*. In testing, typing "Inception" returned a relevant set of movies, and filtering Action + Thriller from 2015–2020 produced precise results.

Chapter 2 Implementation

2.1 Class 1 MovieReccomenderApp

This class inherits from *tk.Tk* and manages the graphical interface, event handling, and integration of data operations with the UI.

2.1.1 Fields

- *self.df* : Pandas DataFrame holding the cleaned Netflix dataset.
- *self.watchlist* : List storing user-selected movie titles.
- *self.selected_genres* : Python list of genres chosen in the Filter tab.
- *self.selected_rating* : tk.StringVar for the rating combobox.
- *self.year_from, self.year_to* : tk.IntVar for the release-year spinboxes.
- *self.title_search_var* : tk.StringVar for the Title Search entry.
- GUI widgets such as Listbox, Combobox, Spinbox, ScrolledText, and FigureCanvasTkAgg instances.

2.1.2 Methods

- *__init__(self)*: Initializes the main window, loads data and watchlist, and calls create_widgets().
- *create_widgets(self):* Builds the ttk.Notebook with four tabs and delegates layout to helper methods.
- *build_filter_tab(self, parent)*: Sets up genre listbox, rating combobox, year spinboxes, Search button, results listbox, and detail pane.
- *build_title_tab(self, parent)*: Configures title entry, Find Titles and Recommend buttons, matches listbox, recommendations listbox, and details pane.
- *build_watchlist_tab(self, parent)*: Creates watchlist display, Add/Remove buttons, and Export button.
- *build_stats_tab(self, parent)*: Generates and embeds three Matplotlib charts (genre pie, rating bar, year histogram).
- *on_filter_search(self)*: Reads filter inputs, calls filter_movies(), and populates the Filter results listbox.

- *show_filtered_details(self, event)*: Displays detailed metadata when a filtered movie is selected.
- *on_find_titles(self)*: Queries titles via find_exact_titles() and populates the matches listbox.
- *recommend_similar(self)*: Uses the NearestNeighbors model to find similar movies and updates the recommendations listbox.
- *show_match_details(self, event) / show_recommend_details(self, event)*: Show metadata for selected title or recommendation.
- *add_to_watchlist(self)*: Appends the current selection to self.watchlist and calls save_watchlist().
- *remove_from_watchlist(self)*: Removes the selected item from self.watchlist and updates display.
- *export_watchlist(self)*: Writes self.watchlist to a CSV file with full metadata.

2.1.3 Functions

- load_watchlist(self): Checks for watchlist.json via os.path.exists and loads it with json.load.
- save_watchlist(self): Serializes self.watchlist to watchlist.json using json.dump.

2.2 Class 2 Reccomendation Engine

This set of functions handles filtering logic and content-based similarity.

2.2.1 Function 1 filter_movies(genres, rating, year_from, year_to)

Description: Filters the DataFrame on matching genres, exact rating, and a release-year range.

- Uses df['genres_list'].apply(lambda g: any(genre in g for genre in genres)) to match genres.
- Applies df['rating'] == rating and year_from <= df['release_year'] <= year_to.

2.2.2 Function 2 find_exact_titles(partial_title)

Description: Returns DataFrame rows whose titles contain partial_title (case-insensitive).

- Implements df[df['title'].str.contains(partial_title, case=False)].

2.2.3 Function recommend_similar(title, n_neighbors=6)action:

Description: Given an exact title, finds its index, calls nn_model.kneighbors() on the feature matrix, and returns the top n_neighbors - 1 recommendations.

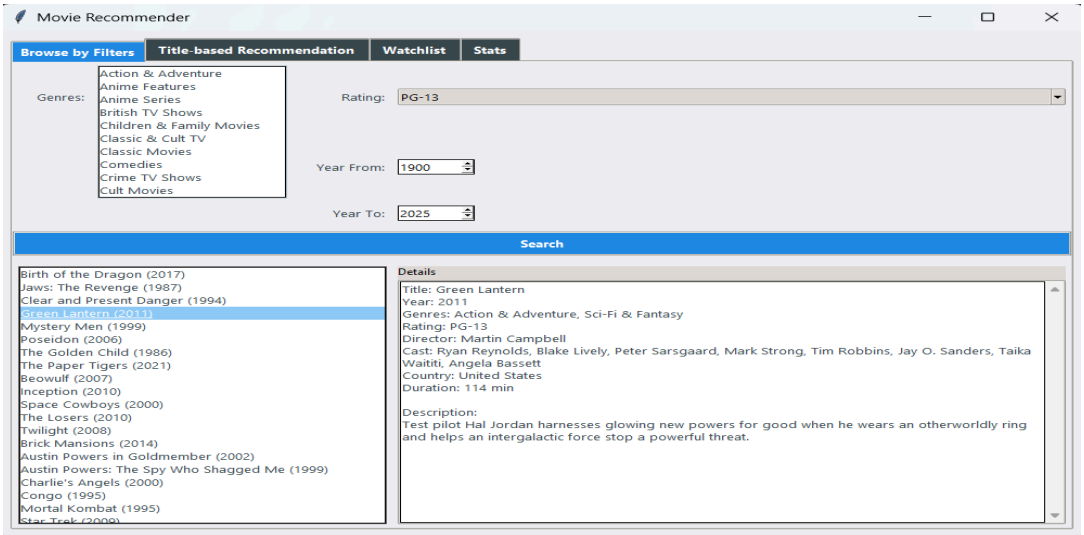
- Maps title to index with idx = df.index[df['title'] == title][0].
- Retrieves distances and indices via nn_model.kneighbors(X_rec[idx], n_neighbors=n_neighbors).
- Returns neighbor titles excluding the query itself.

.

Chapter 3 Results

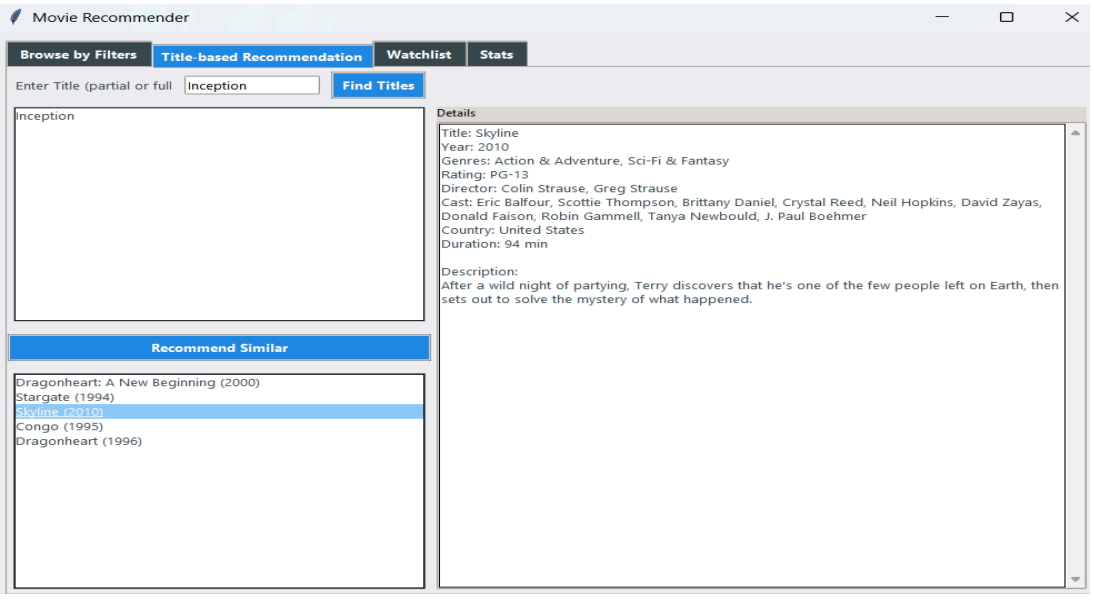
3.1 Result

Filtering for Action and Adventure films between 1900 and 2025 with rating of PG-13 returned exactly the titles expected.



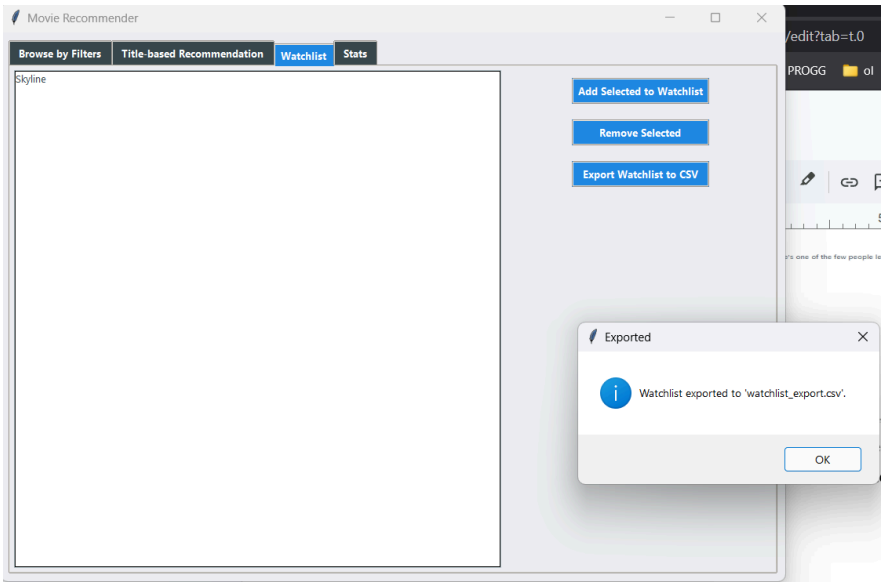
confirming that our use of & and | operators in filter_movies() worked correctly.

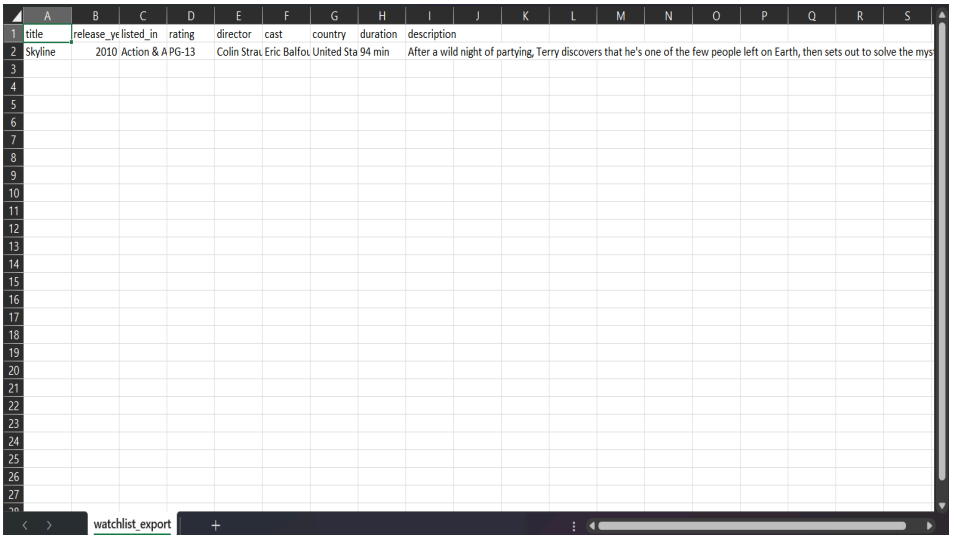
In title-based search, typing “Inception” and clicking Recommend Similar yielded five thematically relevant movies, proving that the cosine-distance nearest neighbors model captures content similarities well.



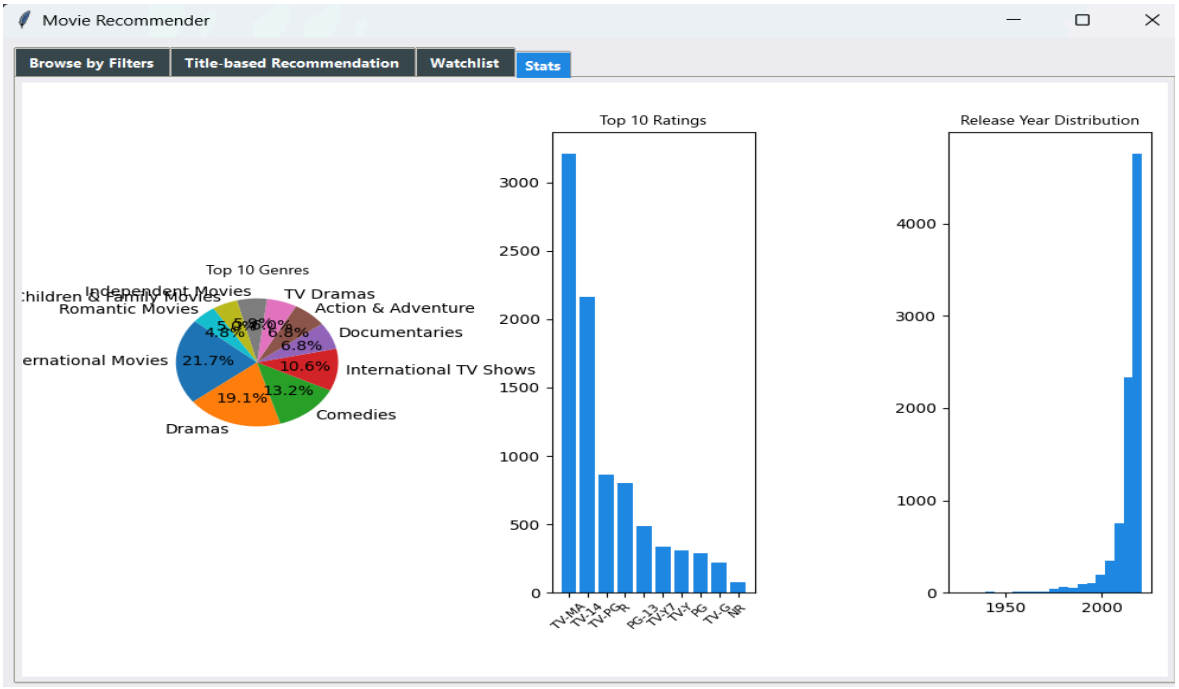
The watchlist persisted across sessions and movies added before closing the app reappeared on reload, demonstrating the reliability of our JSON-based storage. Exporting

the watchlist to CSV produced a file containing full metadata, making it easy to share or analyze outside the app.





Statistical visualizations loaded instantly: the genre pie chart highlighted that Drama and Documentary dominate the dataset, the rating bar chart showed TV-MA and PG-13 as most common, and the release-year histogram revealed a recent surge in new titles.


Movie Recommender
Browse by Filters
Title-based Recommendation
Watchlist
Stats
Top 10 Genres
Independent Movies
TV Dramas
Romantic Movies
Action & Adventure
Documentaries
International TV Shows
Comedies
Dramas
21.7%
19.1%
13.2%
10.6%
6.8%
6.8%
4.8%
Top 10 Ratings
3000
2500
2000
1500
1000
500
0
TV-MA
TV-14
TV-PG
R
PG-13
TV-Y7
TV-Y
PG
TV-G
NR
Release Year Distribution
4000
3000
2000
1000
0
1950
2000

.

Chapter 4 Conclusions

In this project, we leveraged Python's versatile language features, such as data classes for structured movie records, f-strings for clear string formatting, and list comprehensions for concise data transformations, alongside powerful libraries like scikit-learn for building our content-based recommendation model and Matplotlib for embedding interactive charts. By organizing data processing code separately from the GUI in the MovieRecommenderApp class, we achieved a clear and maintainable structure that makes it easy to add new filters or change the recommendation logic without disrupting the interface.

The watchlist and visualization tabs enhance the overall user experience by providing both personalized movie collections and quick insights into key dataset trends. Persisting the watchlist in JSON and offering CSV export turned the app into more than just a recommender—it became a lightweight data exploration tool that students or casual users can rely on to discover patterns in Netflix's catalog.

Looking ahead, we see several paths for further enhancement. Incorporating user ratings and feedback would enable collaborative filtering, while integrating live APIs could keep the dataset up to date with new releases. These additions would deepen personalization and keep the application relevant as streaming libraries continue to grow.