**Movie Recommendation System – Project Report**

Name : Kinjal Patil

Email: patilkinjal628@gmail.com

**1. Project Title**

**Movie Recommendation System using Python (Jupyter Notebook)**

**2. Introduction**

In the era of digital entertainment, movie streaming platforms offer an enormous number of films. Choosing the right movie can be overwhelming for users. This project presents a **Movie Recommendation System** implemented in Python using Jupyter Notebook. The system suggests movies based on a given movie and allows **genre-based filtering** to enhance user experience.

This system uses **content-based filtering**, which recommends movies similar to a given movie based on its title and genre.

**3. Objective**

The main objectives of this project are:

1. To help users discover movies similar to the ones they like.
2. To allow users to filter recommendations based on specific genres.
3. To implement a simple, interactive recommendation system using Python.
4. To demonstrate practical application of **Python, Pandas, and Scikit-learn**.

**4. Tools and Technologies**

* **Programming Language:** Python 3.x
* **IDE / Platform:** Jupyter Notebook
* **Libraries Used:**
  + pandas – For data handling and manipulation
  + scikit-learn – For computing similarity using CountVectorizer and cosine\_similarity
* **Dataset:** movies.csv containing movieId, title, and genres columns

**5. Dataset Description**

The dataset used contains information about movies:

| **Column** | **Description** |
| --- | --- |
| movieId | Unique ID of the movie |
| title | Movie title including release year |
| genres | Genres of the movie (comma-separated) |

**Example entries:**

| **movieId** | **title** | **genres** |
| --- | --- | --- |
| 1 | Toy Story (1995) | Animation,Comedy |
| 2 | Jumanji (1995) | Adventure,Fantasy |
| 3 | The Conjuring (2013) | Horror |

**6. Methodology**

1. **Load Dataset**: Load the movies.csv using Pandas.
2. **Preprocessing**: No additional preprocessing needed, as the title and genre columns are sufficient for content-based recommendation.
3. **Vectorization**: The movie titles are vectorized using CountVectorizer to convert text data into numerical form.
4. **Similarity Calculation**: cosine\_similarity is used to compute similarity between movies based on their titles.
5. **Filtering**: Optional genre-based filtering is implemented to allow recommendations within a specific genre.
6. **Recommendation Function**: A Python function recommend\_movies\_by\_name (or recommend\_movies\_by\_name\_partial) takes a movie name and returns the top N recommended movies, optionally filtered by genre.

**7. Functionalities Implemented**

1. **Movie Recommendation by Name**
   * Users can input a movie name, and the system recommends similar movies.
2. **Top-N Recommendations**
   * Users can specify the number of recommendations they want (top\_n).
3. **Genre-Based Filtering**
   * Users can filter recommendations to only include movies of a specific genre, such as Horror, Sci-Fi, or Animation.
4. **Case-Insensitive and Partial Name Search**
   * The system can find movies even if the user types partial names or uses different casing.
5. **Error Handling**
   * If the movie is not found in the dataset, the system returns an informative message.

**8. Sample Outputs**

**Example 1 – Genre Filtered Recommendation**

print(recommend\_movies\_by\_name('The Conjuring (2013)', top\_n=3, genre\_filter='Horror'))

**Output:**

1. Insidious (2010)

2. Annabelle (2014)

3. Sinister (2012)

**Example 2 – Without Genre Filter**

print(recommend\_movies\_by\_name('Toy Story (1995)', top\_n=5))

**Output:**

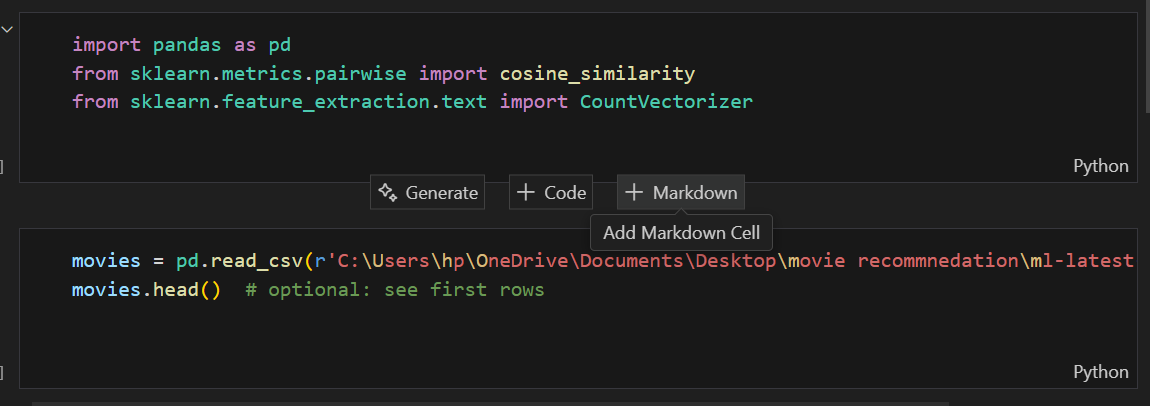
1. Toy Story 2 (1999)

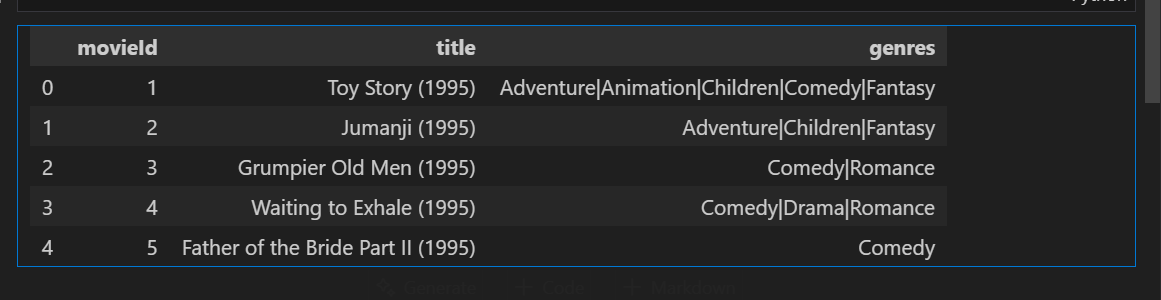
2. A Bug's Life (1998)

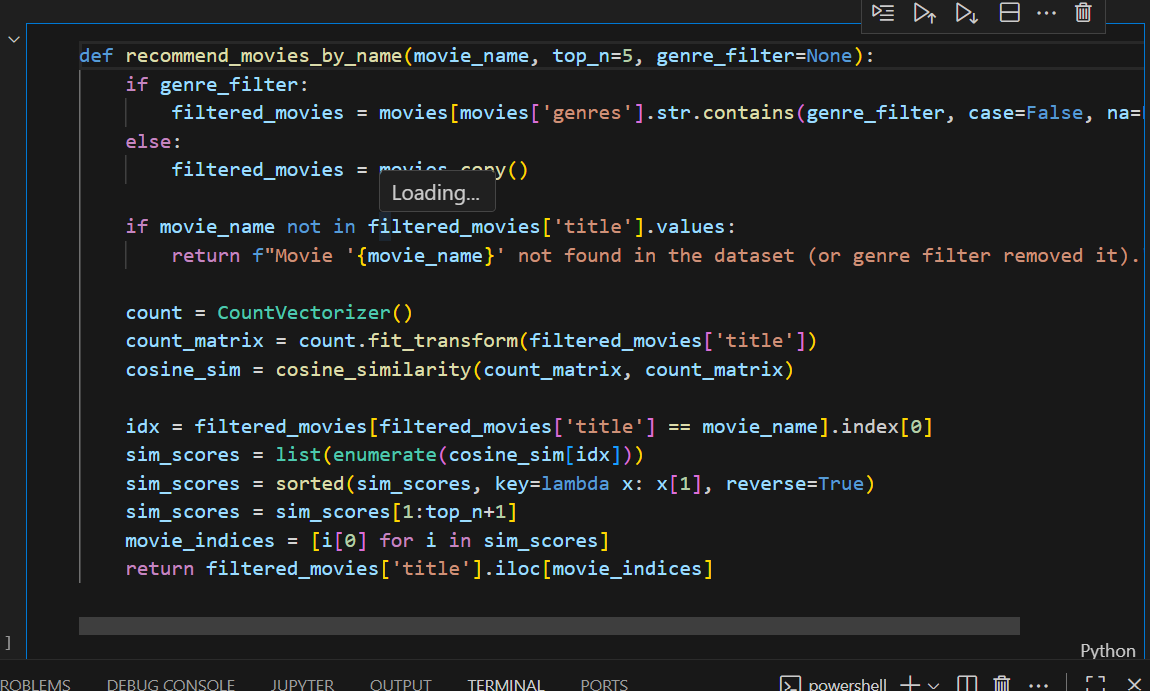
3. Monsters, Inc. (2001)

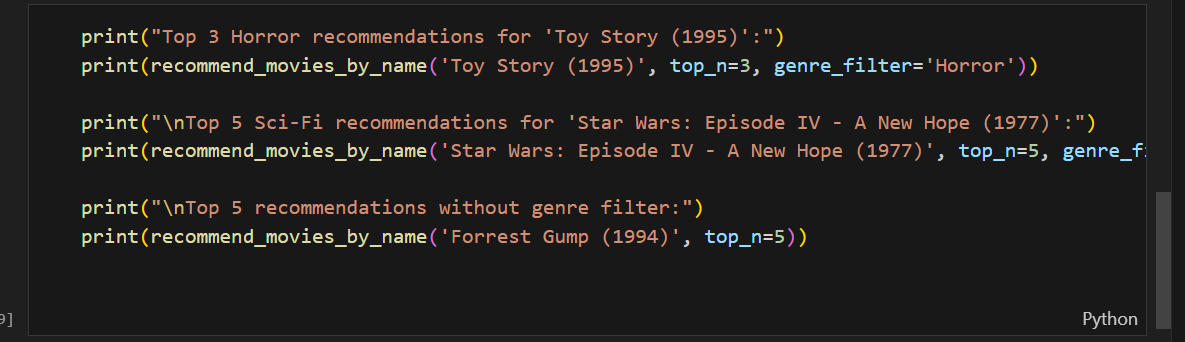
4. Finding Nemo (2003)

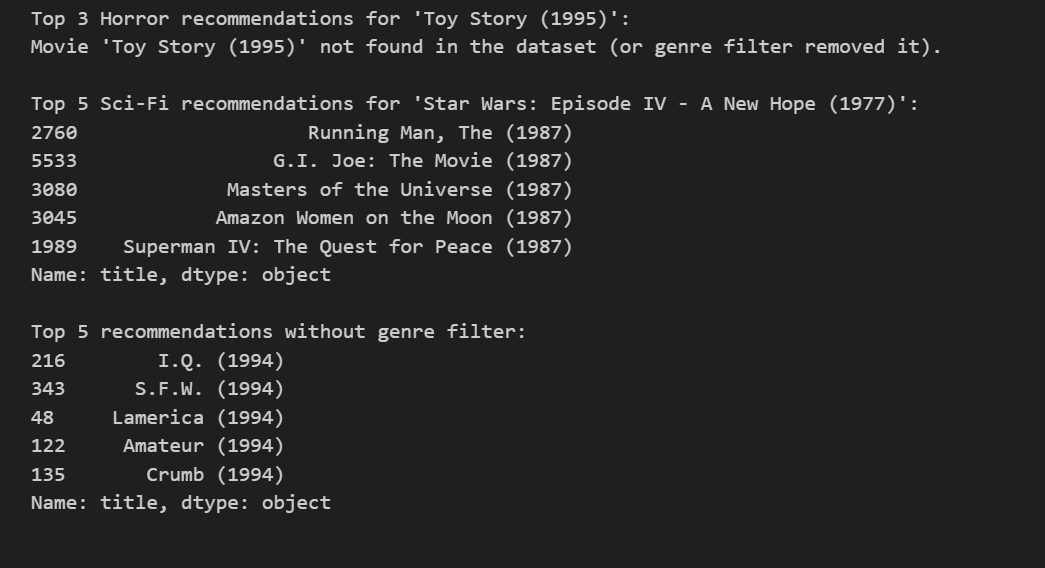
5. Shrek (2001)

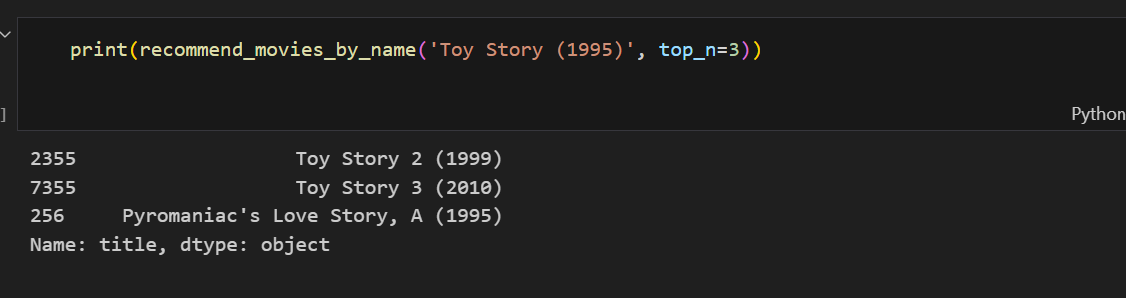












**9. Advantages**

* Helps users discover movies easily.
* Works with **partial movie names**.
* Supports **genre-specific recommendations**.
* Easy to run in Jupyter Notebook with minimal setup.

**10. Limitations**

* Only uses **movie titles** for similarity; using additional features like plot descriptions or actors would improve recommendations.
* Recommendations depend on the dataset used; new movies not in the dataset won’t appear.
* Does not include collaborative filtering; purely content-based.

**11. Future Enhancements**

1. Include **plot descriptions** and **cast information** for more accurate recommendations.
2. Implement a **web interface** using Streamlit or Flask.
3. Add **user ratings** to implement collaborative filtering.
4. Improve recommendation logic with **TF-IDF or embeddings** instead of simple CountVectorizer.

**12. Conclusion**

This Movie Recommendation System demonstrates a basic but functional approach to helping users discover movies based on content similarity and genres. It provides practical exposure to **Python programming, data handling, and machine learning concepts**.

**13. References**

1. Python Documentation – https://docs.python.org/3/
2. Pandas Documentation – https://pandas.pydata.org/
3. Scikit-learn Documentation – https://scikit-learn.org/stable/
4. MovieLens Dataset – https://grouplens.org/datasets/movielens/