Movie Recommendation Application Report

Project 1: Personalized Product Recommendation System

Microsoft Data Engineer Project   
CLS CAI1\_AIS4\_S4e

**Submitted by/ Group 2:**

* **Nour Omar**
* **Fady Hany**
* **Mohab Saber**
* **Reem Ahmed**
* **Nada Mostafa**

**Submitted to: Prof/ Ibrahim Eldeskoki**

# Introduction:

This project focuses on developing a **movie recommendation system** whichis designed to enhance the movie-watching experience by providing personalized movie suggestions using the **TMDB Movies Dataset** based on user preferences. This system utilizes data processing techniques and **cosine similarity** calculations to recommend movies that are like a user-selected title. Built with Python, the application employs the **Streamlit** framework for user interaction, along with **Pandas** for data manipulation and **Scikit-Learn** for modelling.

# Technologies Used: The following technologies were utilized in the development of the Movie Recommendation Application:

1. Python
2. Streamlit
3. Pandas
4. Scikit-Learn
5. Pickle: A Python module used for serializing and de-serializing Python objects. This is used to save and load the movie dataset and the similarity matrix, ensuring efficient data handling.
6. Requests
7. TMDB API date information on the movies being recommended

# Objective:

The primary objective of this application is to offer users movie recommendations that align with their tastes, improving their viewing experience. By utilizing a collaborative filtering approach, the system analyses the features of movies to suggest similar titles, leveraging a vast dataset to ensure diverse and relevant suggestions.

# Dataset

The project uses the **TMDB Movies Dataset**, which contains metadata on movies, including:

* **id**: Unique identifier for each movie.
* **title**: The title of the movie.
* **overview**: A brief plot summary or description of the movie.
* **genre**: The genre(s) of the movie, such as Action, Drama, Comedy, etc.

The dataset is preprocessed to create a new column, **tags**, by concatenating the **overview** and **genre** fields, which is then used to compute movie similarities.

# Work Package 1: ETL Pipeline Development:

## Extracting and loading Data:

The application begins by loading the necessary data files, which include a list of movies and a similarity matrix. This is accomplished using the `pandas` library:

**1. Initial Data Exploration**

* Loaded the TMDB dataset using pandas.
* Displayed a few rows of the dataset using movies.head(10) to understand its structure.
* Used movies.describe() to summarize numerical data (if any).
* Used movies.info() to display detailed information about each column.
* A screenshot of a computer

  Description automatically generatedChecked for missing values using movies.isnull().sum().

A screenshot of a computer

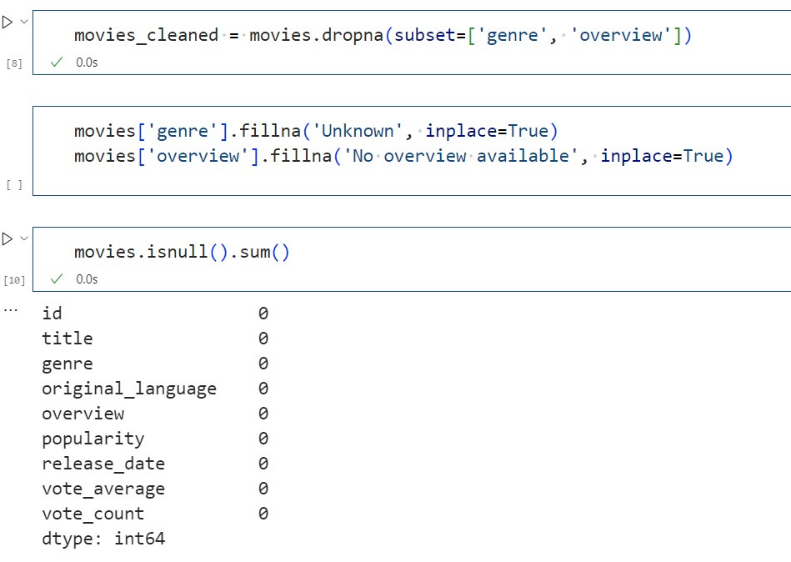
Description automatically generated

A screenshot of a computer program

Description automatically generated

# Work Package 2: Data Preparation and Transformation

## Data Cleansing and Transformation:

A screenshot of a computer

Description automatically generatedOnce loaded, the movie data is cleaned to remove inconsistencies. This step ensures that the dataset is ready for analysis, enabling accurate recommendations using the same library:

**2. Preprocessing**

* **Selecting Relevant Columns**:
  + We selected the columns **id**, **title**, **overview**, and **genre** as these are essential for the recommendation process.
* **Combining Text Data**:
  + Created a new **tags** column by concatenating the **overview** and **genre** columns.
* **Lowercasing Titles**:
  + Converted all movie titles to lowercase to avoid case sensitivity when searching for movies.
* **Dropping Unnecessary Columns**:
  + After creating the **tags** column, the **overview** column was dropped as it was no longer needed separately.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

# Work Package 3: Recommendation Engine Development

## Model Selection:

Use a **content-based filtering** model, applying **cosine similarity** to recommend movies based on their plot descriptions and genres.

## Model Building:

Implement the recommendation engine using Python libraries, specifically **scikit-learn** for vectorizing text and calculating similarity.

**3. Text Vectorization**

* Applied **CountVectorizer** from the sklearn.feature\_extraction.text module to convert the **tags** column into a matrix of token counts (text features).
* A screenshot of a computer program

  Description automatically generatedLimited the maximum number of features to 10,000 to reduce dimensionality.
* Removed common English stop words (such as "the", "and", "is") to focus on important words related to movie descriptions and genres.

**4. Cosine Similarity Calculation**

* Computed **cosine similarity** using cosine\_similarity from sklearn.metrics.pairwise on the text features generated by CountVectorizer.
* Cosine similarity measures the similarity between two movies based on their descriptions, regardless of their length, and outputs a similarity score between 0 and 1.

A screenshot of a computer

Description automatically generated

**5. Recommendation System**

* Implemented a *recommend* function that:
  + Takes a movie title as input.
  + Searches for the movie in the dataset.
  + Retrieves the **top k most similar movies** by sorting their cosine similarity scores in descending order.
  + A screenshot of a computer program

    Description automatically generatedPrints the titles of the recommended movies.

## Model Evaluation:

In this section, we tried various evaluation techniques to assess the quality and relevance of the movie recommendations:

**1. Hit Rate Evaluation**

* **Definition**: The Hit Rate measures the proportion of recommended movies that match the genre of the input movie.
* **Process**:
  + For each input movie, we compare the genres of the recommended movies with the genre of the input movie.
  + **Formula**:   
    A black and white text

    Description automatically generated
* **Results**:
  + We tested this metric for several movies, such as "Iron Man" and "The Godfather", with **k=5** recommendations.
  + A screenshot of a computer code

    Description automatically generated**Iron Man** achieved a **Hit Rate of 0.6**, meaning 60% of the recommended movies shared the same genre.

**2. Mean Absolute Error (MAE)**

* **Definition**: MAE measures the average absolute error between the predicted similarity score and a predefined relevance score. In this case, since we don't have user ratings, we approximated relevance based on genre similarity.
* **Process**:
  + For each recommendation, we assigned a relevance score of 1 if the recommended movie shared the genre of the input movie and 0 otherwise.
  + The **MAE** was calculated based on the difference between predicted similarity scores and these binary relevance labels.
* **Results**:
  + A screenshot of a computer code

    Description automatically generatedFor "Iron Man", the MAE was **0.2**, indicating low average error in genre relevance for the top 5 recommendations.

**3. Root Mean Squared Error (RMSE)**

* **Definition**: RMSE penalizes larger errors more heavily than MAE. It gives a better sense of how large the errors are in terms of predicting genre relevance.
* **Process**:
  + Similar to MAE, we calculated RMSE based on the difference between predicted similarity scores and binary relevance labels.
* **Results**:
  + For "Iron Man", the RMSE was **0.28**, showing that the error was reasonably low for the top recommendations.

A computer code with numbers and symbols

Description automatically generated

**4. Precision, Recall, and F1-Score**

* **Precision**: Measures the proportion of relevant recommendations among all the recommended movies.
* **Recall**: Measures the proportion of relevant recommendations out of all relevant movies in the dataset.
* **F1-Score**: A balance between precision and recall, providing a single score to evaluate the system's performance.
* **Results**:
  + For "Iron Man", the evaluation gave:
    - **Precision**: **0.67**, indicating that 67% of the recommendations were relevant.
    - **Recall**: **0.60**, meaning 60% of the relevant movies were successfully recommended.
    - **F1-Score**: **0.63**, reflecting a balance between precision and recall.

A screenshot of a computer program

Description automatically generated

**5. Precision@K Evaluation**

* **Definition**: Precision@K evaluates the relevance of the top **k** recommendations. It checks how many of the top **k** movies are relevant (based on genre).
* **Process**: Calculated how many of the top **k** recommended movies matched the input movie's genre.
* **Results**:
  + A screenshot of a computer program

    Description automatically generatedFor **k=5** recommendations for "Iron Man", the **Precision@K** was **0.8**, meaning 80% of the top 5 recommendations were relevant.

A screenshot of a computer

Description automatically generated**7. Data Saving with Pickle**

* After building the similarity matrix, the processed data (new\_data) and similarity matrix (similarity) were saved using the pickle library for future use, so the model does not need to be recomputed each time.

# Work Package 4: API Deployment and Visualization

## API Development:

1. **User Interface Development**: Build an interactive user interface using **Streamlit** where users can:

* Search for a movie from a dropdown menu.
* View recommended movies based on cosine similarity.
* A screen shot of a computer code

  Description automatically generatedFilter recommendations by release year, genre, and more.

1. **Personalized User Interaction**: Allow users to add movies to their favorites list and display their favorite movies dynamically.

A screen shot of a computer program

Description automatically generated

1. **Reporting Dashboard**:

Provide users with additional information on recommended movies, such as:

* A screen shot of a computer program

  Description automatically generated**Genres**, **Release Year**, **Ratings**, and **Movie Overview**.

*The whole code of our project is available in this link:*

# Conclusion

This project successfully developed a **movie recommendation system** that recommends similar movies based on content (description and genre) and evaluates the recommendation quality using a variety of metrics. The system was deployed using **Streamlit**, providing users with an easy-to-use interface for exploring recommendations and viewing movie details.

Using multiple evaluation techniques, including **Precision@K**, **Hit Rate**, **MAE**, **RMSE**, **Precision**, **Recall**, and **F1-score**, the project offers a well-rounded assessment of the recommendation engine's performance.

Future improvements can include integrating collaborative filtering and additional features such as user ratings and personalized recommendations based on user behavior.