In this report, I will provide an overview of the entire project, highlighting key stages and decisions made along the way.

Data Analysis

I initiated the analysis with the main dataset containing movie ratings. The first step involved removing the "Timestamp" column due to its overwhelming number of distinct values. Subsequently, attention shifted to the movie dataset, where I identified and addressed missing values in the "VideoReleaseDate" column. Notably, I encountered a peculiar case of a movie lacking a title and genre but had received ratings. Opting for data integrity, I removed this anomalous entry.

Another challenge emerged with two movies lacking links. While I swiftly identified the first, the second presented complications—a movie with a similar name but distinct release date and incongruent genres. I chose to retain this movie while flagging it as lacking a link. Duplicates in the movie dataset were also addressed by removing redundant entries, necessitating the corresponding adjustments in the main dataset with ratings.

The user dataset analysis proved relatively straightforward, primarily focusing on the prevalence of numerous distinct zip codes. Despite the option to convert these codes into city names or coordinates, I decided against it, recognizing that movie recommendations typically transcend geographical considerations. Finally, I identified and removed movies that had never been rated.

In summary, the dataset now encompasses comprehensive information about users and their rated movies.

All details can be found in the first notebook.

Model implementation

Opting for simplicity, I implemented the Nearest Neighbor algorithm, circumventing the need for model training. Categorical data underwent conversion to scalars using LabelEncoder. I constructed a similarity matrix for users and their rated movies, incorporating user information. To prevent model overfitting, I standardized the entire dataset using StandardScaler. For user convenience, I implemented a class outside the model to automatically handle data conversion before inputting it into the model. Thus, to receive movie suggestions, users need only provide information about themselves and the movies they've rated. You can find the implementation in this file

Model fine-tuning

Fine-tuning the model involved leveraging Bayesian optimization to optimize hyperparameters, specifically the number of neighbors and leaf size in the KNN algorithm. This process resulted in the identification of optimal hyperparameter values such as $k = 16$ and $leaf\_size = 64$ showing up to 80% of accuracy. All details you can find in the second notebook.

Model Evaluation

For evaluation, I generated test data where the model earned a point by suggesting a liked movie, excluding one that the user had already watched and rated positively. I employed F1 score and accuracy score as evaluation metrics, emphasizing them as the primary metrics discussed in the university. You can check the evaluation by running this script

Results

The resulting model stands as an effective recommender system, offering movie suggestions based on user information. With an impressive 60% accuracy according to the metrics, I anticipate further improvements with the incorporation of additional data on modern movies and contemporary user preferences worldwide.