EDS 6346- Data Mining for Engineers

Movie Recommendation System Using R

Group- 4

Introduction:

The "Movie Recommendation System using R" project is an innovative endeavor that aims to create a personalized recommendation engine for suggesting movies to users based on their preferences and viewing behavior. This project utilizes the MovieLens dataset, which is widely recognized for its comprehensive information on movie ratings. The dataset includes 105,339 ratings given to 10,329 unique movies, making it an ideal resource for implementing and testing recommendation algorithms. By focusing on real-world data, the project bridges the gap between theoretical concepts and practical applications in data science and machine learning.

We designed four recommendation methods to provide personalized movie suggestions. It employs Content-Based Filtering using nearest neighbours to recommend movies based on their features, such as genre and style. User-Based Collaborative Filtering with KMeans clustering is used to group users with similar preferences and suggest movies based on the behavior of like-minded users. Item-Based Collaborative Filtering, leveraging a cosine similarity matrix, identifies similar movies based on user ratings, ensuring recommendations are tailored to the user's specific tastes. Lastly, a Hybrid Approach combines KMeans clustering with nearest neighbours to blend the strengths of both user and item-based methods, improving recommendation relevance and accuracy.

R, a powerful programming language for statistical computing and data visualization, forms the backbone of this project. The implementation leverages several R libraries, including recommenderlab, ggplot2, data.table, FNN, dplyr and reshape2, etc. These libraries enable

efficient data manipulation, visualization, and the construction of sophisticated recommendation models.

The project not only focuses on building the recommendation system but also emphasizes the importance of data preprocessing and analysis using the MovieLens dataset. This involves cleaning and organizing the movies.csv and ratings.csv files to ensure that the recommendations are based on accurate and relevant information. The dataset is merged, and relevant features are extracted to improve the quality of the recommendations. Additionally, data visualizations are used to explore user behavior, movie trends, and rating distributions, which help in understanding user preferences and movie popularity. These insights inform the development of more effective algorithms for generating personalized movie recommendations.

Through this project, I demonstrate a strong grasp of key data science concepts, including machine learning, collaborative filtering, and data visualization. By working with the MovieLens 20M dataset and employing advanced tools, the project serves as a hands-on application of these skills, highlighting my ability to tackle real-world challenges in recommendation systems. Additionally, the insights gained from this project can be transferred to other domains that rely on personalized recommendations, such as e-commerce, streaming platforms, and online learning systems.

Dataset Overview:

The MovieLens 20M dataset offered a useful and varied dataset for developing and evaluating movie recommendation systems. It showcased user interactions with movies through ratings and tags, making it an invaluable tool for both research and practical implementations. Its structured format and comprehensive coverage of user preferences provided significant insights into recommendation system design.

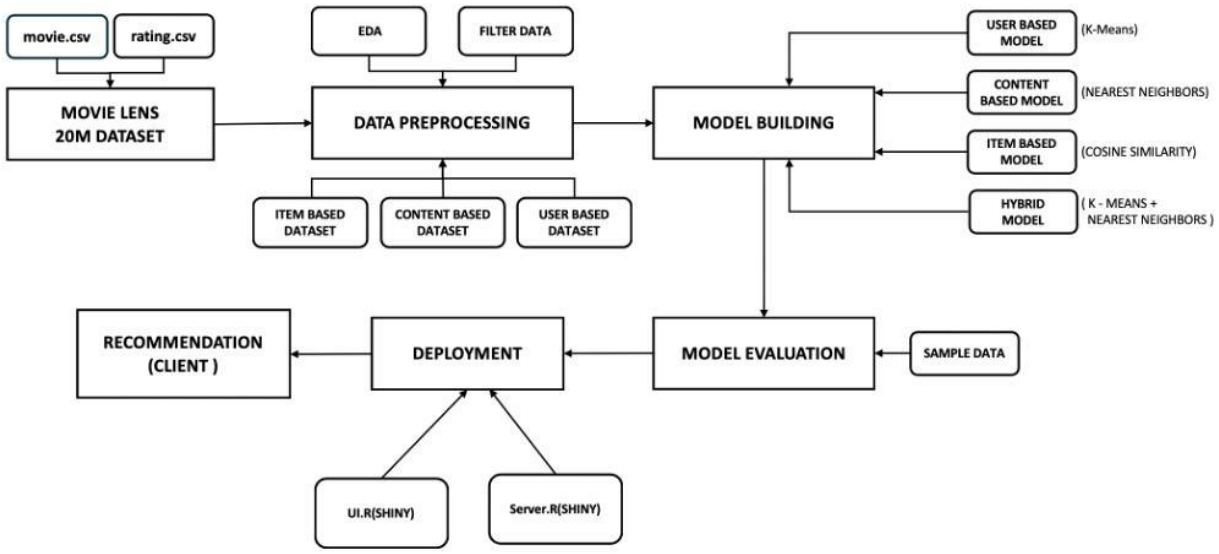
Dataset - https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset

Detailed Characteristics of the Dataset:

1. **File Structure**:
    - The dataset included multiple CSV files:
        - **Ratings**: Contained user ID, movie ID, rating, and timestamp.
        - **Movies**: Included movie ID, title, and genres.
        - **Tags**: Provided user ID, movie ID, tag, and timestamp.
        - **Links**: Mapped movie IDs to external databases such as IMDb and TMDb.
2. **User Profiles**:
    - Each user had a unique identifier, anonymized to protect privacy.
    - Users displayed diverse engagement levels, with some providing only a few ratings and others contributing thousands.
    - This variance allowed developers to study long-tail effects in recommendation systems.
3. **Movie Metadata**:
    - Movie titles came with genre labels (e.g., Action, Comedy, Drama), enabling content-based filtering.

- The presence of external database links allowed integration with rich movie metadata for enhanced analysis.
- Timestamps spanned several years, capturing long-term trends in user preferences and movie popularity.
- The dataset allowed experiments with temporal recommendation models like time-aware collaborative filtering.

System Design:



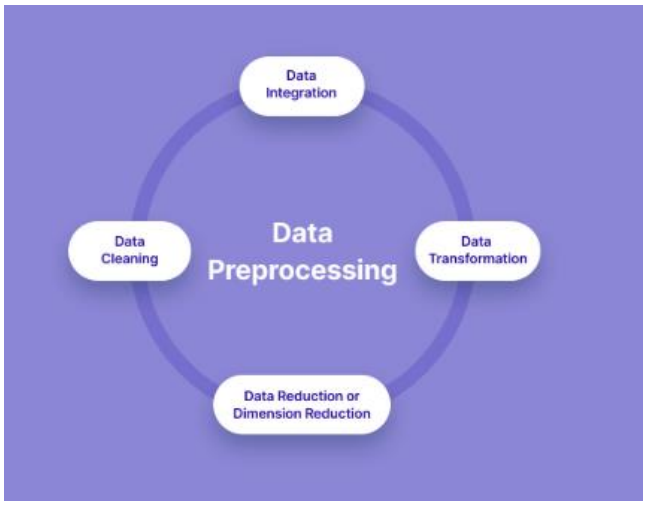
Data Preprocessing:

In this project, the data preprocessing phase is crucial to preparing the MovieLens 20M dataset for effective movie recommendation system development. The dataset consists of two primary files, movies.csv and ratings.csv, which are merged in three different ways to cater to distinct recommendation approaches. The preprocessing process involves several key steps to ensure data quality and relevance for building accurate and personalized recommendation models.

First, we perform **null checking** to identify and handle any missing values in the dataset. This is followed by **duplicate checking** to ensure that there are no repeated records that could distort the

analysis. Next, we apply **standard scaling** to normalize numerical features, ensuring that they are on a comparable scale and improving the performance of machine learning algorithms. We also conduct **one-hot encoding** on the genres feature, which is a categorical attribute, to transform it into a format that can be used by machine learning models.

Throughout this process, we use **R tools** for visualization, which helps us analyze various aspects of the dataset, such as user behavior, movie trends, and rating distributions. These visualizations provide insights that inform the design of more effective algorithms and enhance the overall understanding of the dataset.



Here, we are performing one-hot encoding on the movies.csv file on the genres feature.

```
movies_onehot <- movies_genres %>%
  mutate(value = 1) %>%
  pivot_wider(names_from = genres, values_from = value, values_fill = 0)

# Preview the one-hot encoded dataset
head(movies_onehot)
```

```
## # A tibble: 6 × 22
##   movieId title Adventure Animation Children Comedy Fantasy Romance Drama Action
##     <int> <chr>     <dbl>     <dbl>    <dbl>  <dbl>   <dbl>   <dbl> <dbl>  <dbl>
## 1       1 Toy …         1         1        1      1       1       0     0      0
## 2       2 Juma…         1         0        1      0       1       0     0      0
## 3       3 Grum…         0         0        0      1       0       1     0      0
## 4       4 Wait…         0         0        0      1       0       1     1      0
## 5       5 Fath…         0         0        0      1       0       0     0      0
## 6       6 Heat…         0         0        0      0       0       0     0      1
## # ℹ 12 more variables: Crime <dbl>, Thriller <dbl>, Horror <dbl>,
## #   Mystery <dbl>, `Sci-Fi` <dbl>, IMAX <dbl>, Documentary <dbl>, War <dbl>,
## #   Musical <dbl>, Western <dbl>, `Film-Noir` <dbl>, `(no genres listed)` <dbl>
```

The dataset is merged in three distinct ways to support different recommendation techniques. For the **content-based filtering** approach, we focus on movie attributes such as genre, director, and actors. In this case, we remove user data and merge only the movies.csv and ratings.csv files to build a model based solely on the content of the movies themselves. This approach recommends movies similar to those a user has previously rated highly, relying on the intrinsic features of the items.

	movieId	title	(no genres listed)	Action	Adventure	Animation	Children
0	1	Toy Story (1995)	0	0	1	1	
1	2	Jumanji (1995)	0	0	1	0	

…….

IMAX	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western	rating
0	0	0	0	0	0	0	0	3.921239561
0	0	0	0	0	0	0	0	3.211976802

For the **item-based filtering** and **hybrid recommendation approaches**, the merged dataset includes user information. In item-based filtering, the goal is to identify movies that are similar based on user ratings, using techniques such as cosine similarity. The hybrid approach combines the strengths of multiple methods, integrating both user and movie data to generate more personalized and relevant recommendations. By including user data, these approaches leverage the interactions between users and movies, making the recommendations more tailored to individual preferences.

Finally, for **user-based filtering**, the dataset is further refined by focusing on data from 2,000 users, after merging the movies.csv and ratings.csv files. This approach groups users based on their rating behaviours and recommends movies based on the preferences of similar users. By selecting a subset of users, we ensure that the model is focused on a manageable, yet diverse group of individuals, which is essential for generating meaningful recommendations.

```r
# Transform genres into binary columns
movie_features <- movies %>%
  separate_rows(genres, sep = "\\|") %>%
  mutate(value = 1) %>%
  pivot_wider(names_from = genres, values_from = value, values_fill = 0)

# Merge with average ratings
movie_features <- merge(movie_features, movie_avg_ratings, by = "movieId")

# View the feature matrix
head(movie_features)
```

	movieId <int>	title.x <chr>	Adventure <dbl>	Animation <dbl>	Children <dbl>	Com… <dbl>	Fantasy <dbl>	Roma… <dbl>
1	1	Toy Story (1995)	1	1	1	1	1	0
2	2	Jumanji (1995)	1	0	1	0	1	0
3	3	Grumpier Old Men (1995)	0	0	0	1	0	1
4	4	Waiting to Exhale (1995)	0	0	0	1	0	1
5	5	Father of the Bride Part II (1995)	0	0	0	1	0	0
6	6	Heat (1995)	0	0	0	0	0	0

6 rows | 1-9 of 25 columns

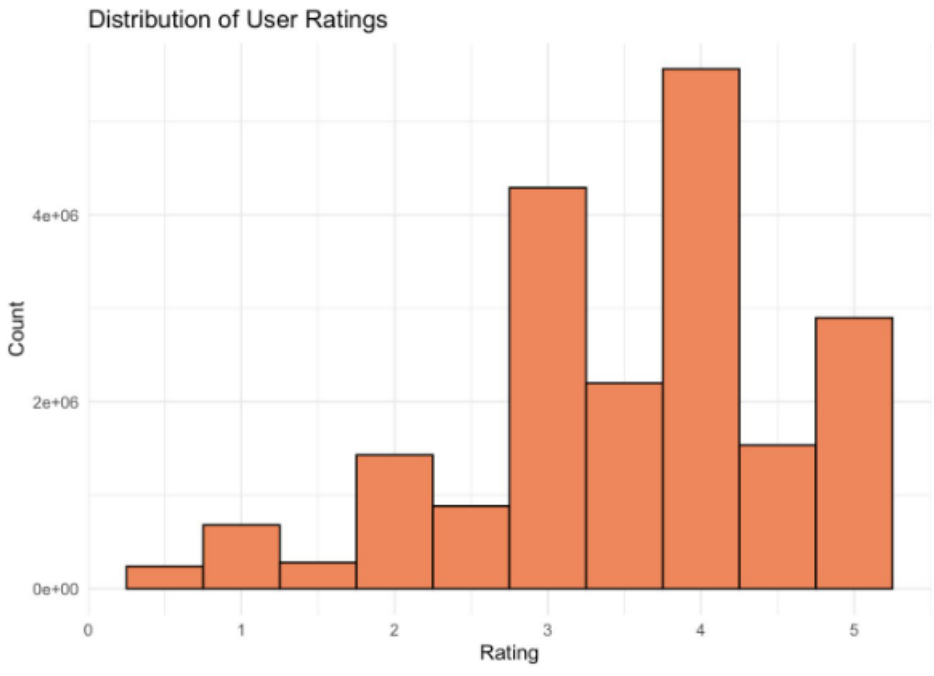
Hide

These preprocessing steps ensure that the MovieLens dataset is clean, well-organized, and ready for use in building various recommendation models. Through this comprehensive preprocessing pipeline, we prepare the dataset for content-based, item-based, hybrid, and user-based

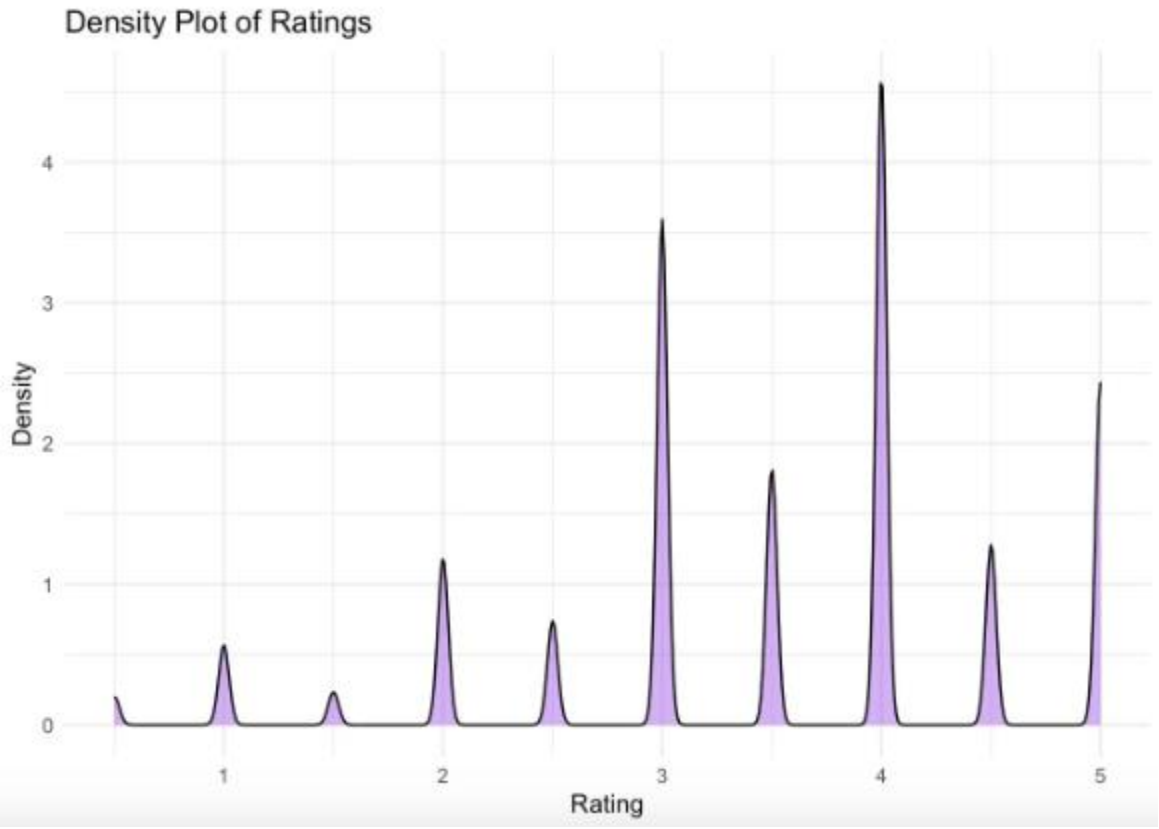
recommendation systems, each offering unique ways of personalizing movie recommendations for users.

Exploratory Data Analysis:

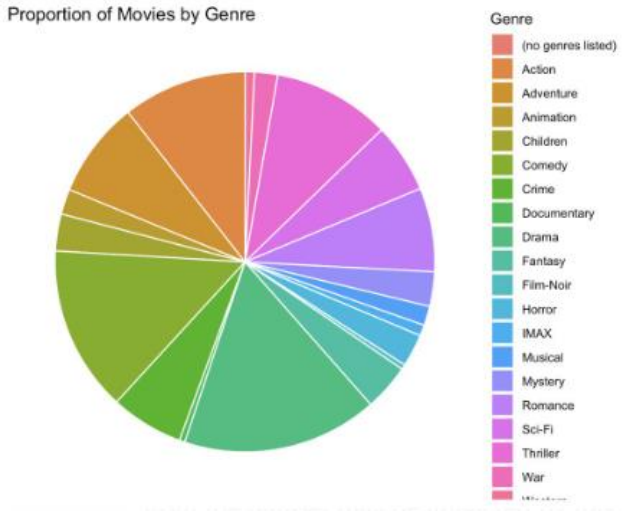
In the Exploratory Data Analysis (EDA) phase of this project, we aimed to gain a deeper understanding of the MovieLens dataset and its features, which include movie information, user ratings, and genres.



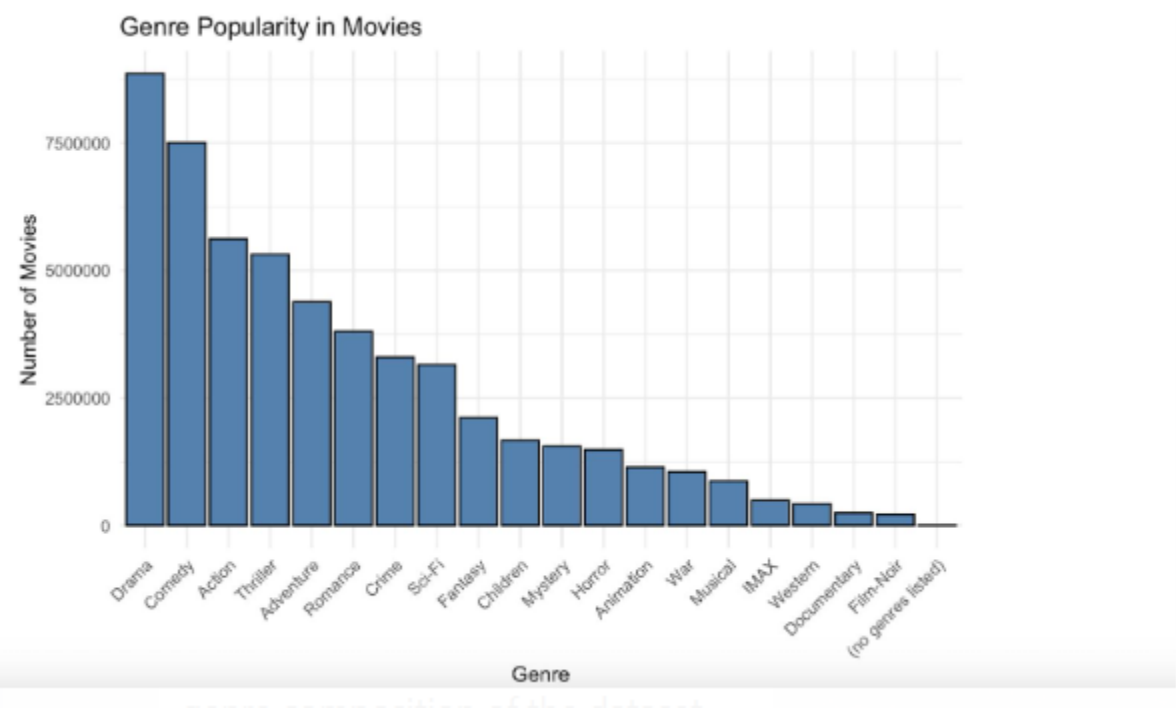
The histogram indicates a propensity toward average to good assessments, with the majority of user ratings centered between 3 and 4. Less often occurring ratings of 1 and 2 indicate less severely negative viewpoints. A comparatively optimistic bias in the dataset is shown by ratings of 5, which are noteworthy as well although somewhat less common than 4.



These are the most commonly given ratings, as seen by the density map, which shows clear peaks at ratings of 3, 4, and 5. The abrupt peaks indicate a strong bias toward average to high ratings, with a high concentration of user ratings around these ranges. There are far less unfavorable evaluations in the sample, as indicated by the lower frequency of ratings like 1 and 2.



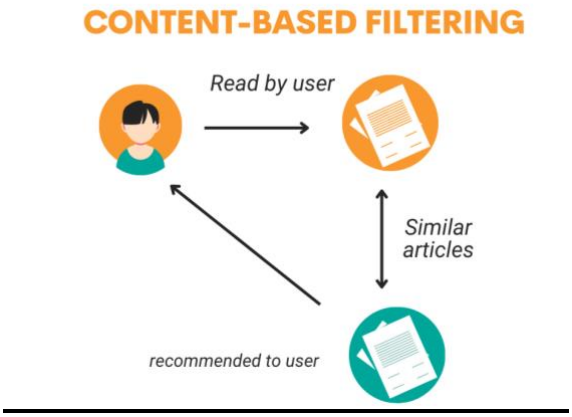
The percentage of films by genre in the dataset is displayed in the pie chart. The dataset is dominated by drama, comedy, and action, as evidenced by its highest shares. There is an imbalance in the representation of genres, as evidenced by the smaller percentages of less popular genres including Western, IMAX, and film noir. The range of suggestions or insights obtained from the dataset may be impacted by this distribution.



The distribution of films by genre is seen in the bar plot. The most popular genres are action, comedy, and drama, with drama having the biggest number. On the other hand, genres such as Western, Documentary, and Film-Noir are noticeably underrepresented, suggesting an imbalance in the genre composition of the dataset. Diversity in analysis or recommendations may be impacted by this bias.

Model Building:

Model-1: Content Based Filtering



In this method, we used a merged dataset obtained after the data preprocessing step, which resulted in features like genres and ratings, to develop a content-based filtering system for movie recommendations. The core of our approach was a Nearest Neighbors algorithm, implemented using Euclidean distance to measure the similarity between movies based on their numerical attributes. This algorithm was designed to identify the second-closest movie to a given input, ensuring that the recommendations were both accurate and relevant by excluding the input movie itself.

Nearest Neighbors approach.

```
train_matrix <- as.matrix(data[, -c(1, 2)])
train_structure <- knn.index(train_matrix)
save(train_structure, file = "knn_structure.RData")
```

+ Code

```
get_second_closest_neighbors <- function(data, input_movie, train_matrix, k = 2) {
  filtered_movies <- data[data$movieId %in% input_movie, ]
  test_matrix <- as.matrix(filtered_movies[, -c(1, 2)])
  knn_result <- get.knnx(data = train_matrix, query = test_matrix, k = k)
  return(knn_result$nn.index[, 2])
}
```

```
load("knn_structure.RData")
input_movie <- c(1,22, 34, 40)
res <- get_second_closest_neighbors(data, input_movie, train_matrix, k = 2)
```

```
cat("The movie recommendation ids are:", res, "\n")
```

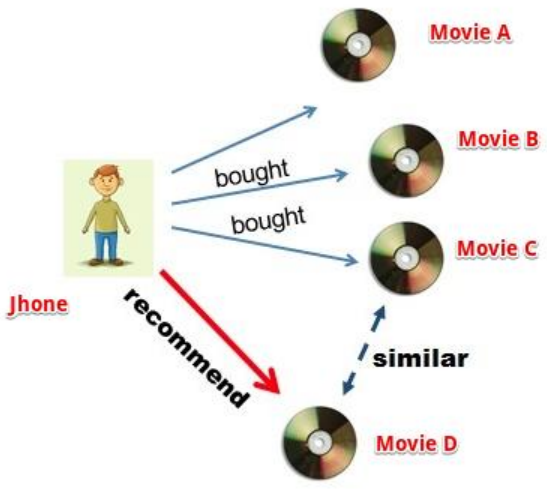
```
The movie recommendation ids are: 4791 20826 8077 3095
```

This approach on content-based filtering proved highly effective as it directly utilized content features to identify movies most similar to the input. The results were displayed as a list of recommended movie IDs, providing a clear and practical output that could be easily interpreted and applied.

One of the key advantages of this content-based filtering method is its independence from user interaction data, making it especially beneficial for addressing the cold-start problem for new users. Furthermore, the model offers interpretable recommendations, as each suggestion is based on measurable attributes such as genres and ratings, allowing for transparency in understanding why specific movies were recommended.

Overall, nearest neighbors approach in content-based filtering performed well, demonstrating its capability to provide precise and meaningful recommendations compared to other approaches on this filtering.

Model-2: Item-Based Collaborative Filtering



In this project, we developed a movie recommendation engine using Item-Based Collaborative Filtering, applied to the MovieLens dataset. The process began with data preprocessing, where

movie genres were extracted and one-hot encoded into a binary matrix, enabling movies to be categorized by their genres. The user rating data was reshaped into a user-item rating matrix, which was then converted into a sparse matrix for efficient computation.

```
number_of_items_sorted <- sort(number_of_items, decreasing = TRUE)
number_of_items_top <- head(number_of_items_sorted, n = 4)
table_top <- data.frame(as.integer(names(number_of_items_top)),
                        number_of_items_top)
for(i in 1:4) {
  table_top[i,1] <- as.character(subset(movie_data,
                                        movie_data$movieId == table_top[i,1])$title)
}

colnames(table_top) <- c("Movie Title", "No. of Items")
head(table_top)
```

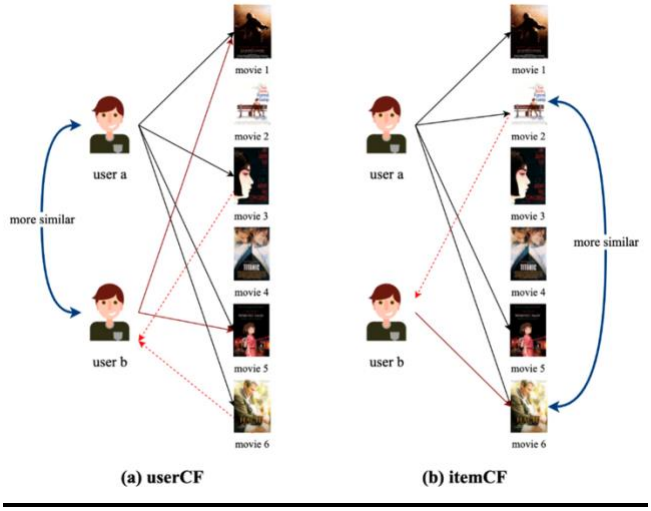
	Movie Title <chr>	No. of Items <fctr>
25	Leaving Las Vegas (1995)	16
16	Casino (1995)	11
904	Rear Window (1954)	11
150	Apollo 13 (1995)	10

4 rows

The Item-Based Collaborative Filtering model, implemented using the recommenderlab library, calculates movie similarity using cosine similarity between items. This approach ensures that the system recommends movies that are similar to those the user has rated highly, improving the relevance of the suggestions. Additionally, various visualizations, such as heatmaps and bar plots, were employed to explore the dataset and understand user behavior, including identifying the most viewed movies and examining the distribution of ratings.

The recommendation system generates personalized movie suggestions for each user by identifying movies that share similar characteristics to those the user has rated positively. This method ensures highly relevant recommendations based on the content users have previously interacted with. The model's ability to leverage collaborative filtering techniques makes it a robust solution for delivering accurate and meaningful movie recommendations to users, offering a strong foundation for building recommendation systems.

Model-3: User-Based Collobarative Filtering



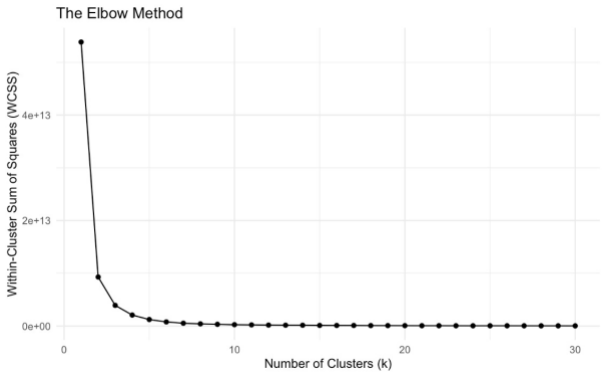
In this method, we used a processed dataset created after a thorough data preprocessing step, including handling missing values and normalizing ratings. This ensured the data was suitable for clustering and collaborative filtering tasks.

The core of our approach involved applying K-Means Clustering to group users into clusters based on their interaction patterns. This step helped identify groups of users with similar preferences, reducing the computational cost by narrowing down similarity calculations to users within the same cluster.

Within each cluster, user similarity was computed using Pearson Correlation. Recommendations were generated by aggregating ratings from users in the same cluster, weighted by their similarity scores. This localized filtering approach allowed us to provide more precise and personalized recommendations.

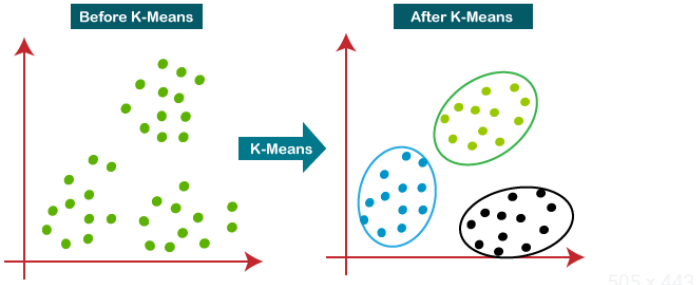
One of the key advantages of using K-Means in user-based filtering is its ability to handle large datasets efficiently while maintaining high-quality recommendations. By clustering users with similar behaviors, the approach minimizes noise from unrelated user data and focuses on relevant interactions.

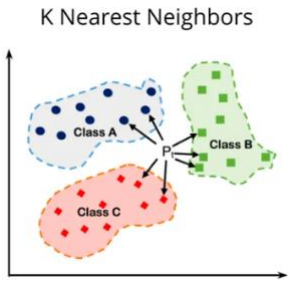
Overall, the K-Means clustering approach in user-based filtering performed well. It enhanced the scalability and accuracy of the recommendation system, providing an efficient framework for personalized suggestions while reducing computational complexity.



Model 4: Hybrid model

To build a hybrid movie recommendation system combining K-Means Clustering and Nearest Neighbors (KNN) for personalized recommendations.





505 × 443

<u>Grouping Movies (K-Means Clustering):</u>

- Movies are grouped into clusters based on how similar they are in terms of genres (like Action, Comedy) and average ratings.
- For example, movies with similar genres and ratings might be grouped into the same cluster.

<u>Finding Similar Movies (KNN):</u>

- When a user provides a movie title, the system:
    1. Finds which cluster that movie belongs to.
    2. Looks for the most similar movies within the same cluster.

- The system uses the distance between the input movie and others to measure similarity.

  Euclidean distance =$\sqrt{(x2 - x1)^2} + (\mathrm{y2} - \mathrm{y1})^2$

Advantage:

***** Clustering reduces the number of movies to search through by focusing only on similar groups.

* KNN ensures precise recommendations by finding the closest matches within the group.

Implementation

1) Features were scaled to normalize values, allowing fair comparison during clustering and similarity computations.

2) **Clustering with K-Means**: Group movies into clusters based on genres and ratings.

   Method: Applied K-Means clustering with k=10, using **Euclidean distance** for similarity.

   Outcome: Each movie was assigned a cluster label, reducing the search space for recommendations.

3) **Nearest Neighbor Recommendations**: Recommend movies similar to the user's input movie within the same cluster.

   Method:

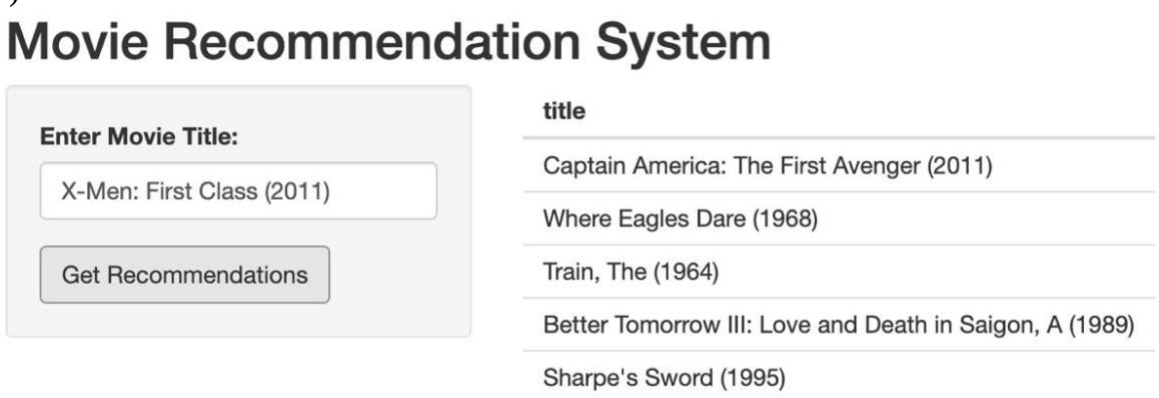
   - Identified the input movie's cluster.
   -
   - Used **KNN** within the cluster to find k nearest movies using **Euclidean distance**.

4) **Saving Model Parameters**:

- Saved the clustered dataset, scaled features, and K-Means model as RDS files.
- Enabled efficient loading during deployment, avoiding redundant computations.

Accuracy, Output and Analysis:

1)



Clustering: "X-Men: First Class (2011)" was grouped into a cluster with action, sci-fi, and adventure movies based on genres and ratings.

Nearest Neighbors: Movies from the same cluster were ranked by similarity using Euclidean distance.

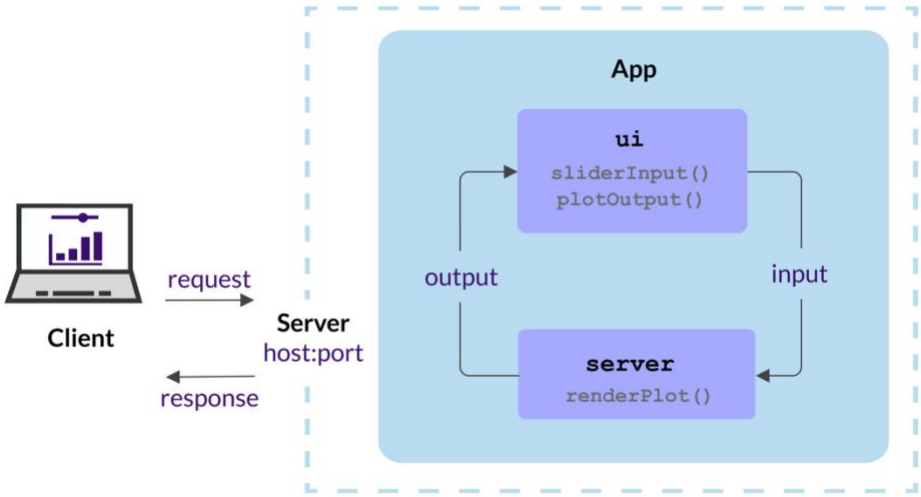
Why the model recommended it:

- Captain America: The First Avenger (2011): Superhero action and adventure, similar genre and release year.
- Where Eagles Dare (1968): Action-adventure themes with high ratings.
- The Train (1964): Tactical missions and historical action.
- A Better Tomorrow III (1989): Action-drama with intense war themes.
- Sharpe's Sword (1995): Action-war, resonating with teamwork and combat themes.

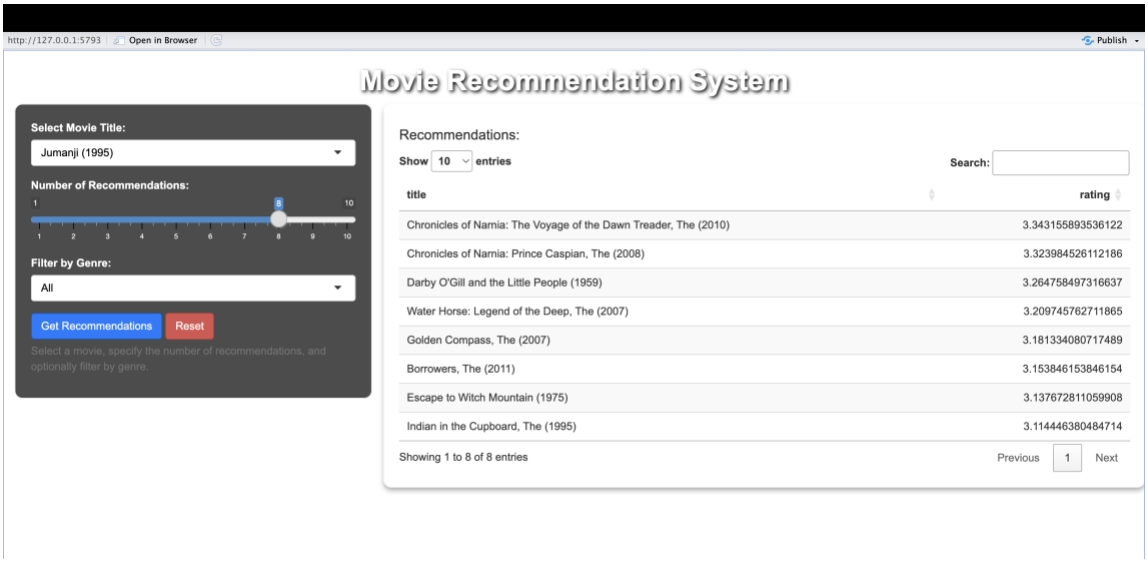
Deployment using Shiny:

For this project, we used the Shiny framework to create a web-based movie recommendation system.

1. **Interactive Input**: Users enter a movie title to receive real-time recommendations.
2. **Dynamic Display**: Recommendations are shown instantly in a clean, responsive table.
3. **Backend Integration**: Pre-trained K-Means and KNN models with saved parameters ensure efficient runtime performance.
4. **User-Friendly UI**: Intuitive interface with real-time responses.
5. **Optimized Performance**: Pre-loaded models and Shiny's reactive features minimize computational overhead, ensuring fast interactions.



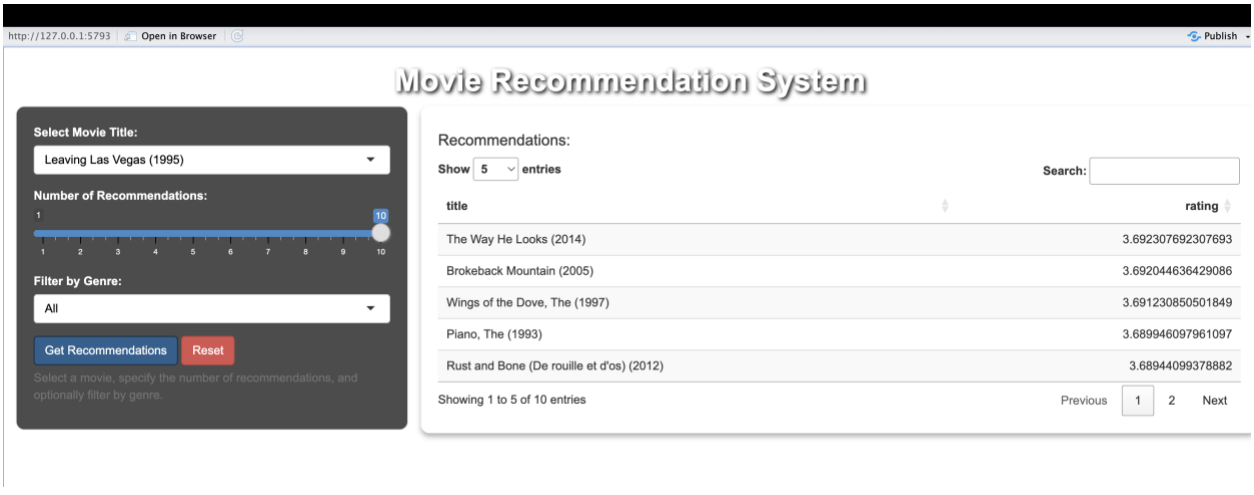
Implementation:



Filter by genre:

Movies were placed recommended using single genre.

Movies were placed on top with respective to IMDB rating.



Limitations:

1. **UI Constraints:** The dropdowns and filtering in the Shiny app become sluggish with large datasets, reducing responsiveness.
2. **Memory Issues:** The current system lacks memory-efficient data structures, leading to out-of-memory (OOM) errors with large datasets.
3. **Cluster Limitations:** Clustering works well for small datasets but becomes computationally intensive as cluster size grows.
4. **Scalability Constraints:** The system struggles to handle user-item matrices exceeding 140,000 entries, causing delays and bottlenecks.
5. **Resource Exhaustion:** Tasks like clustering and nearest-neighbor searches demand significant memory and processing power, often exceeding the capacity of standard machines.
6. **No GPU Support:** The system relies on CPU-only processing, making tasks like clustering and matrix operations time-consuming and unsuitable for real-time recommendations.

Project Contributions:

PSID	Name	Subgroup(Task)	Role
2339727	Ashish Darshi	Algorithm	user-based collaborative filtering, content based , Hybrid model
2316063	Vasista Tummala	Algorithm	user-based collaborative filtering
2315346	Praneeth Puppala	Algorithm	Item based cf
2313512	Lakshmi Rohith Reddy Annapureddy	Algorithm	Hybrid Model, UI
2315340	Saran teja mallela	Preprocessing	Algorithm, matrix factorization
2315355	Tanmai Veerapaneni	Preprocessing	Data Collection and Preprocessing, Algorithm
2311912	Keerthi Yadav Eeraboina	Preprocessing	Report,ppt, Algorithm,UI
2313014	Prajith sai Macha	UI	Report,ppt, Algorithm,UI
2313039	Nikhil Tanneeru	UI	Report,ppt, Algorithm,UI
2333658	Saladi Venkat Akshay	UI	Report,ppt, Algorithm,UI

GitHub Link:

https://github.com/RohithAnnapureddy26/Movie-Recommendation-Sysytem-using-R

Demo Video:
https://drive.google.com/file/d/1yntvbi3fPlEl5NXx5YgGYd2WC9SKesUB/view?usp=sharing

References:

1. Resnick, P., & Varian, H. R. (1997). Recommender Systems. *Communications of the ACM*, 40(3), 56–58.
2. Harper, F. M., & Konstan, J. A. (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4), 1–19.
3. Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4), 331–370.
4. Aggarwal, C. C. (2016). Content-Based Recommender Systems. *Recommender Systems: The Textbook*, Springer, 139–166.
5. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8), 30–37.
6. Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery*, 5(1–2), 115–153.
7. Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Computing Surveys (CSUR)*, 52(1), 1–38.
8. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-Based Collaborative Filtering Recommendation Algorithms. *Proceedings of the 10th International Conference on World Wide Web*, 285–295.
9. Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749.

10. Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. *Recommender Systems Handbook*, Springer, 1–35.
11. Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets. *2008 Eighth IEEE International Conference on Data Mining*, 263–272.
12. Bell, R. M., & Koren, Y. (2007). Lessons from the Netflix Prize Challenge. *ACM SIGKDD Explorations Newsletter*, 9(2), 75–79.
13. Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems (TOIS)*, 22(1), 5–53.
14. Bennett, J., & Lanning, S. (2007). The Netflix Prize. *Proceedings of KDD Cup and Workshop*, 35(4), 3.
15. Gunawardana, A., & Shani, G. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *Journal of Machine Learning Research*, 10, 2935–2962.