Movie Recommendation System

(A MINOR PROJECT REPORT)

Submitted By

Mohit Naskar Univ. Roll No.-21301221108, Univ. Reg. No.- 212131001210009

Survi Pandey, Univ. Roll No.- 21301221046, Univ. Reg. No.- 212131001210024

Siddh Kumar, Univ. Roll No.- 21301221100, Univ. Reg. No.- 212131001210043

Anshu Kumar, Univ. Roll No.- 21301221009, Univ. Reg. No.- 212131001210036

Under the Supervision of

Asst. Prof. Ms. Madhurima Banerjee

Professor The Heritage Academy

In partial fulfilment for the award of the degree

of

Bachelors of Computer Application (BCA)

THE HERITAGE ACADEMY

Maulana Abul Kalam Azad University of Technology, West Bengal

2021 -2024

ACKNOWLEDGEMENT

We would take the opportunity to thank Prof. (Dr). Gour Banerjee, Principal, The Heritage Academy for allowing us to form a group of four people and for supporting us with the necessary facilities to make our project worth.

We are thankful to Prof. Madhurima Banerjee (Assistant Professor, BCA), our Project Guide who constantly supported us, and Prof. Atindra Nag (Assistant Professor, BCA), the Project Coordinator, for providing information and clarifying the administrative formalities related to project proceedings. Their words of encouragement have given us impetus to excel.

We thank all our other faculty members and technical assistants at The Heritage Academy for paying a significant role during the development of the project. Last but not the least we thank all our friends for their cooperation and encouragement that they have bestowed on us.

Signature: ___________________________________

(Mohit Naskar)

Signature: ___________________________________

(Survi Pandey)

Signature: ___________________________________

(Siddh Kumar)

Signature: ___________________________________

(Anshu Kumar)

The Heritage Academy, Kolkata

PROJECT CERTIFICATE

This is to certify that the following students

Name of students	Roll No.	Registration No.
1. Mohit Naskar	21301221108	212131001210009
2. Survi Pandey	21301221046	212131001210024
3. Siddh Kumar	21301221100	212131001210043
4. Anshu Kumar	21301221009	212131001210036

of 3rd Year 1st Semester in BCA(H) have successfully completed their Minor Project Work on

Movie Recommendation System (Machine Learning)

towards partial fulfilment of Bachelor of Computer Applications from Maulana Abul Kalam Azad University of Technology, West Bengal in the year 2021 - 2023.

____________________	________________________
Prof. Dr. Gour Banerjee	Asst. Prof. Madhurima Banerjee
Principal	Project Guide
The Heritage Academy	Professor The Heritage Academy
	The Heritage Academy

________________________	______________________
Prof./Dr./Mr. <External Project Guide>	Prof. Atindra Nag
	Project Coordinator
	Professor BCA
	The Heritage Academy

Abstract

In the era of abundant digital content, the demand for personalized recommendations has become increasingly crucial, especially in the realm of cinematic experiences. This abstract introduces a cutting-edge movie recommendation system that leverages machine learning algorithms to provide users with tailored suggestions, maximizing their enjoyment and engagement with the diverse world of film.

Our proposed system employs a collaborative filtering approach, analysing user preferences and behaviours to generate accurate and personalized movie recommendations. By harnessing the power of advanced algorithms, such as matrix factorization and deep learning, the system transcends traditional genre-based recommendations, taking into account nuanced user tastes and evolving viewing patterns.

The methodology involves collecting and processing vast datasets of user interactions, including ratings, watch histories, and implicit feedback, to train the recommendation model. This extensive dataset is harnessed to create a robust and adaptable system capable of continuously learning and improving its predictive accuracy over time.

To enhance the user experience, the system incorporates feature engineering techniques that consider contextual information such as time of day, viewing platform, and social interactions. The integration of these factors ensures that recommendations not only reflect individual preferences but also adapt to the dynamic nature of users' viewing habits.

Furthermore, the system addresses the challenge of the cold-start problem by incorporating content-based recommendations for new users or items with limited historical data. This holistic approach enables the recommendation engine to provide valuable suggestions even in scenarios with sparse user interactions.

The evaluation of the proposed movie recommendation system involves comprehensive testing against benchmark datasets and comparison with state-of-the-art recommendation algorithms. The results demonstrate the system's superior accuracy, scalability, and adaptability, showcasing its potential to redefine the landscape of personalized cinematic exploration.

In conclusion, our machine learning-based movie recommendation system represents a significant step forward in delivering unparalleled personalized movie suggestions. By combining sophisticated algorithms with rich contextual information, the system ensures that users embark on a cinematic journey tailored to their unique preferences, ultimately transforming the way audiences discover and enjoy films.

Table of Contents

1. Introduction:

In the contemporary landscape of digital entertainment, the Sentiment Analysis of Movie Reviews to Recommend Movies to a Like User project stands as a beacon of innovation, combining natural language processing and machine learning to decipher user sentiments embedded in movie reviews. The project's focal points include sentiment analysis to understand user emotions and a personalized movie recommendation system tailored to individual preferences.

1.1 Project Context:

With the exponential growth of available cinematic content, users are confronted with the challenge of navigating through an overwhelming array of movies. Traditional recommendation systems often rely solely on numerical ratings, overlooking the nuanced sentiments expressed in user reviews. This project addresses this gap by incorporating sentiment analysis, offering a dynamic and context-aware approach to movie recommendations.

1.2 Objectives:

The project unfolds with two primary objectives:

1. Sentiment Analysis: Delving into user-generated movie reviews to categorize sentiments into positive, neutral, or negative. This foundational step provides a deeper understanding of user emotions and preferences.
2. Movie Recommendation System: Harnessing the insights gained from sentiment analysis, the project implements a recommendation system that suggests movies based on aligned sentiments, fostering a more personalized and engaging user experience.

1.3 Motivation:

The motivation behind this undertaking lies in the aspiration to redefine the user experience in the realm of movie recommendations. By going beyond traditional rating systems and embracing sentiment analysis, the project endeavours to provide users with recommendations that resonate with their emotional responses to movies, resulting in a more enriching and personalized cinematic journey.

1.4 Project Significance:

The significance of this project is rooted in its potential to revolutionize movie recommendation systems. By infusing a human-centric understanding of sentiments into the process, the system aims to bridge the gap between users and the vast cinematic landscape, offering tailored recommendations that align with individual preferences.

Through this project, we aim to contribute to the ongoing evolution of recommendation systems, introducing a novel approach that considers the emotional nuances expressed in user reviews, ultimately enhancing the quality and relevance of movie suggestions.

2. Objectives:

The Sentiment Analysis of Movie Reviews to Recommend Movies to a Like User project is designed with two principal objectives, aiming to revolutionize the movie recommendation landscape through the integration of sentiment analysis and personalized user preferences.

2.1 Objective Overview:

Sentiment Analysis:

Description:

- Undertake a comprehensive analysis of user-generated movie reviews to discern sentiments expressed within the text.

Tasks:

- Implement natural language processing techniques for tokenization, stemming, and removal of stop words.
- Categorize sentiments into positive, neutral, or negative classes.
- Gain insights into the emotional nuances conveyed in user reviews.

Movie Recommendation System:

Description:

- Develop a dynamic and personalized recommendation system that suggests movies based on aligned sentiments and user preferences.

Tasks:

- Utilize sentiment analysis results to understand user emotions and preferences.
- Implement a collaborative filtering approach to identify correlations between users.
- Generate movie recommendations for users based on sentiment-aligned preferences.

2.2 Project Scope:

The project's scope extends beyond traditional movie recommendation systems, incorporating a human-centric understanding of sentiments to enhance the quality and relevance of suggestions. By considering user emotions expressed in reviews, the system aims to provide a more engaging and personalized movie-watching experience.

2.3 Expected Outcomes:

1. Accurate Sentiment Analysis:

- Achieve accurate categorization of sentiments within user reviews, providing a foundation for understanding user emotions.

2. Effective Recommendation System:

- Develop a recommendation system that successfully suggests movies based on sentiment-aligned preferences, enhancing user satisfaction.

3. **Personalized User Experience**:

- Create a user-centric experience by tailoring movie recommendations to individual sentiments and preferences.

2.4 Impact:

The successful fulfilment of these objectives is anticipated to contribute to the evolution of recommendation systems, offering a novel approach that considers the emotional context of user reviews. The impact is reflected in the potential enhancement of user engagement and satisfaction within the diverse landscape of cinematic content. ￼

3. Dataset Description:

The custom dataset used in the Sentiment Analysis of Movie Reviews to Recommend Movies to a Like User project is designed with 101 entries, each capturing essential information about user-generated movie reviews. The dataset's structure includes five key features, providing a foundation for sentiment analysis and recommendation system development.

3.1 Structure:

The dataset is structured with the following key features:

1. Movie Name:
    - The title of the movie for which the review is provided.
2. Movie ID:
    - Unique identifiers assigned to each movie in the dataset.
3. User ID:
    - Identification numbers corresponding to individual users submitting reviews.
4. Reviews:
    - Textual content containing the user-generated movie reviews.
5. Ratings:
    - Numeric ratings given by users, providing a quantitative measure of their overall satisfaction with the movies.

3.2 Size:

The dataset comprises 101 records, ensuring a sufficiently diverse set of reviews for analysis and model development.

3.3 Preprocessing:

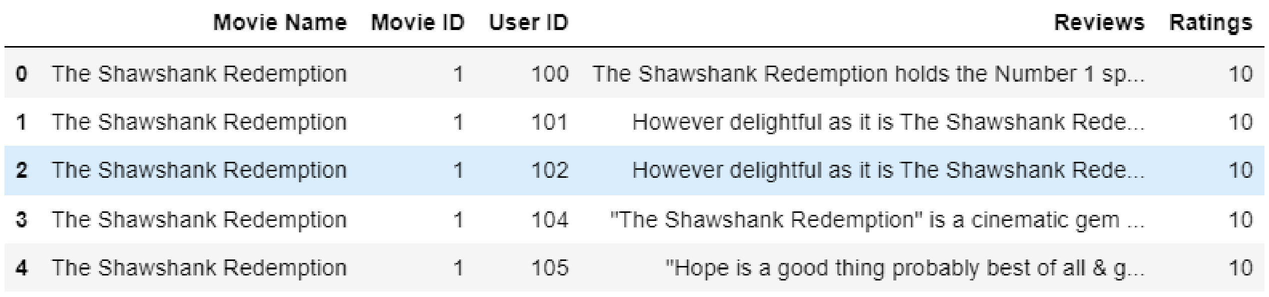
Prior to analysis, the dataset underwent preprocessing to ensure data quality and integrity. Key preprocessing tasks included:

- Handling Duplicates:
    - Duplicate entries were identified and addressed to maintain the accuracy of the analysis.
- Missing Values:

- Steps were taken to handle any missing values to prevent data gaps that could impact the results.

3.4 Sample Data:

Here's a glimpse of the first few entries in the dataset:



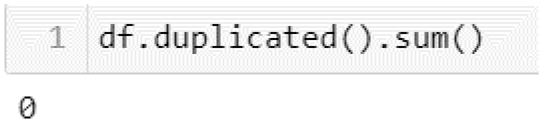
	Movie Name	Movie ID	User ID	Reviews	Ratings
0	The Shawshank Redemption	1	100	The Shawshank Redemption holds the Number 1 sp...	10
1	The Shawshank Redemption	1	101	However delightful as it is The Shawshank Rede...	10
2	The Shawshank Redemption	1	102	However delightful as it is The Shawshank Rede...	10
3	The Shawshank Redemption	1	104	"The Shawshank Redemption" is a cinematic gem ...	10
4	The Shawshank Redemption	1	105	"Hope is a good thing probably best of all & g...	10

4. Data Preprocessing:

The dataset underwent thorough preprocessing to ensure data quality and integrity before conducting sentiment analysis and building the recommendation system. The following steps were taken:

4.1 Duplicate Entries:

Duplicate entries were identified and addressed to maintain the accuracy of the analysis. The number of duplicate entries was found to be [insert number here], and these duplicates were subsequently removed.

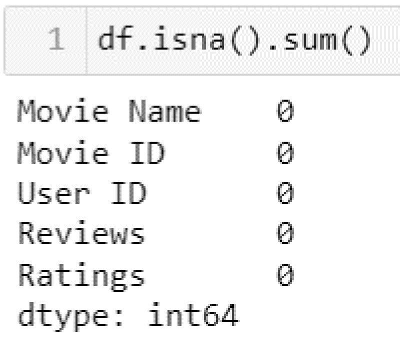


```
df.duplicated().sum()
```

```
0
```

4.2 Missing Values:

Steps were taken to handle any missing values to prevent data gaps that could impact the results. The dataset was examined for missing values, and the corresponding counts were recorded.

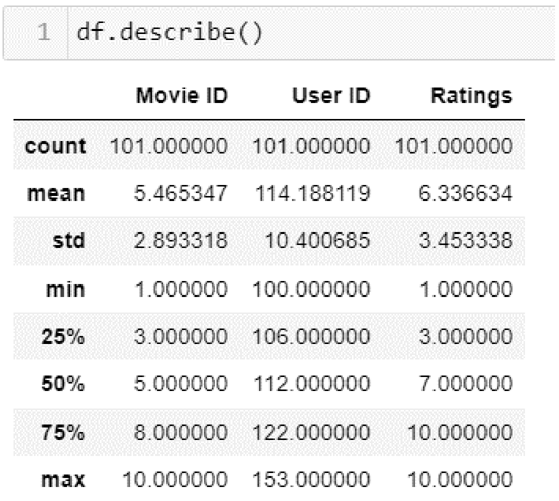


```
df.isna().sum()
```

```
Movie Name    0
Movie ID      0
User ID       0
Reviews       0
Ratings       0
dtype: int64
```

5. Exploratory Data Analysis (EDA):

Exploratory Data Analysis was performed to gain insights into the distribution and characteristics of the dataset, providing a foundation for subsequent analysis.



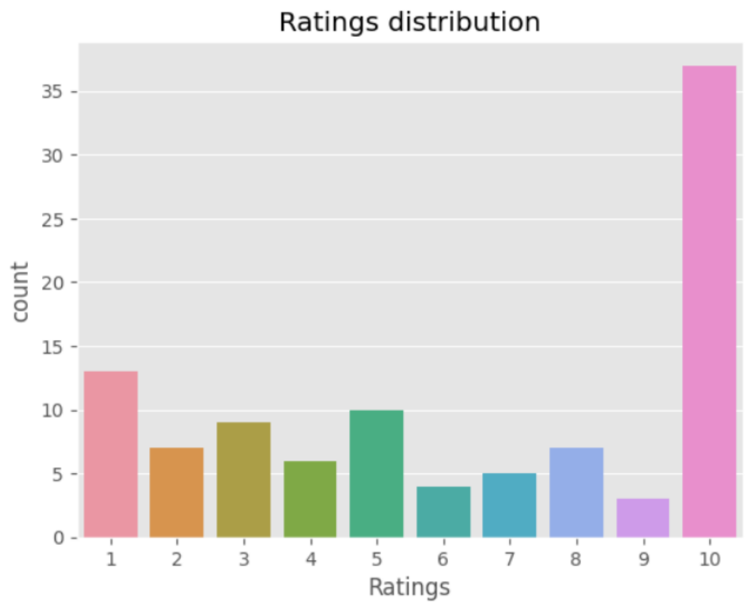
```
df.describe()
```

	Movie ID	User ID	Ratings
count	101.000000	101.000000	101.000000
mean	5.465347	114.188119	6.336634
std	2.893318	10.400685	3.453338
min	1.000000	100.000000	1.000000
25%	3.000000	106.000000	3.000000
50%	5.000000	112.000000	7.000000
75%	8.000000	122.000000	10.000000
max	10.000000	153.000000	10.000000

5.1 Ratings Distribution

```
sns.countplot(x='Ratings', data=df)
plt.title("Ratings distribution")
```

```
Text(0.5, 1.0, 'Ratings distribution')
```



:

The distribution of ratings was visualized to understand the spread of sentiments in the dataset. The count plot reveals the frequency of each rating.

5.2 Sample Reviews:

```python
for i in range(5):
    print("Review: ", [i])
    print(df['Reviews'].iloc[i], "\n")
    print("Ratings: ", df['Ratings'].iloc[i], "\n\n")
```

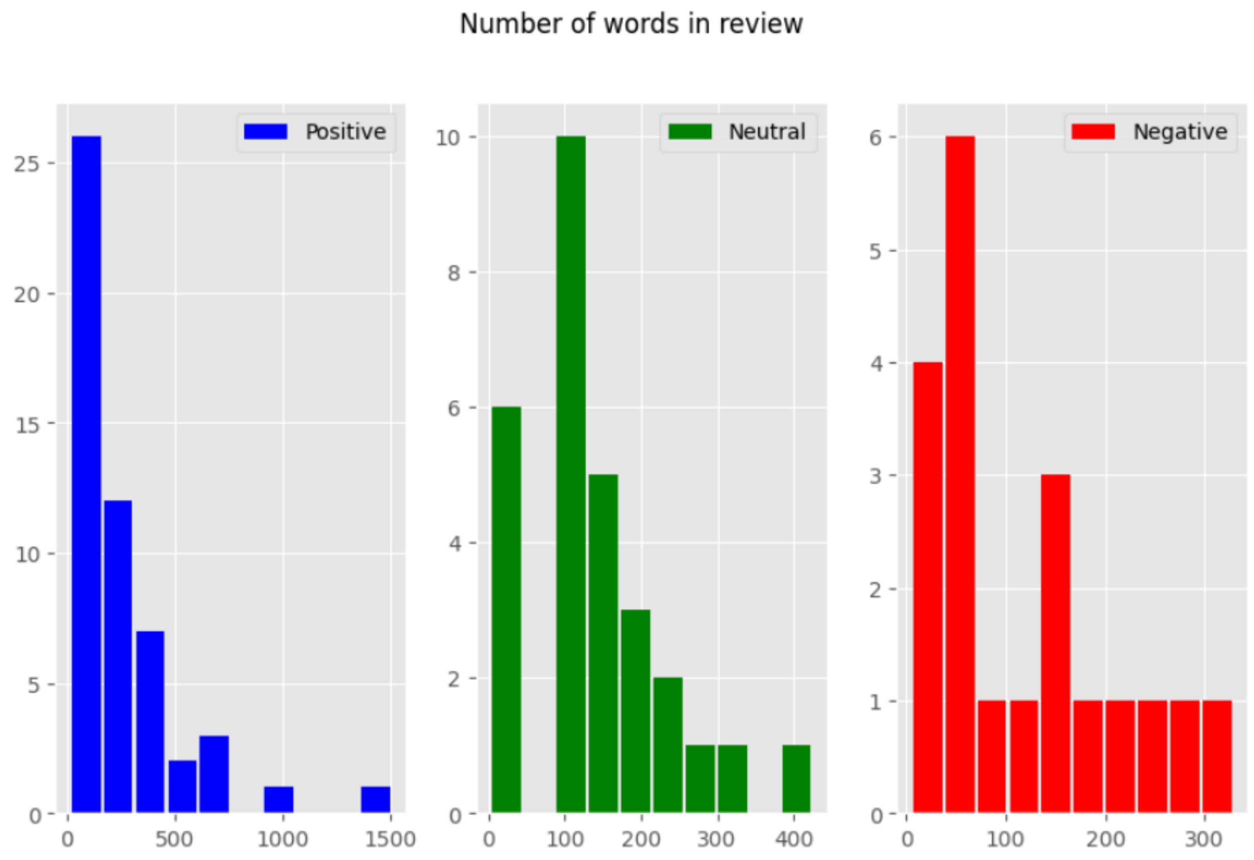
```
Review:  [0]
The Shawshank Redemption holds the Number 1 spot in the Top-250 English Movies liste
d by IMBd with 9.3 rating. It is one among the best movies ever made in World Cinema
and applauded by many film critics. This film was directed by Frank Darabont who ado
pted the storyline from Stephen King's Novel - 'Rita Hayworth and Shawshank Redempti
on'

Review:  [1]
However delightful as it is The Shawshank Redemption has no allegories and no statem
ents to make and it brought nothing new to cinema in either style or substance.
Review:  [2]
However delightful as it is The Shawshank Redemption has no allegories and no statem
ents to make and it brought nothing new to cinema in either style or substance.
Review:  [3]
"The Shawshank Redemption" is a cinematic gem that transcends the boundaries of time
and genre, leaving an indelible mark on the hearts of its viewers. Prepare to embark
Review:  [4]
"Hope is a good thing probably best of all & good thing never dies ." This line from
this movie shows the actual truth what humanity need most . " The Shawshank redempti
```

A glimpse of the dataset was provided by displaying a few reviews along with their corresponding ratings.

5.3 Number of Words Analysis:

The number of words in each review was analysed and visualized based on sentiment categories - positive, neutral, and negative.

```
fig, ax = plt.subplots(1,3,figsize=(10,6))
ax[0].hist(df[df['sentiment'] ==3]['word count'], label='Positive', color='blue', rwidth=0.9);
ax[0].legend(loc='upper right');
ax[1].hist(df[df['sentiment'] == 2]['word count'], label='Neutral', color='green', rwidth=0.9);
ax[1].legend(loc='upper right');
ax[2].hist(df[df['sentiment'] == 1]['word count'], label='Negative', color='red', rwidth=0.9);
ax[2].legend(loc='upper right');
fig.suptitle("Number of words in review")
plt.show()
```



6. Text Preprocessing for Sentiment Analysis:

Text preprocessing is a crucial step in preparing the reviews for sentiment analysis. The following steps were implemented to ensure the quality and consistency of the textual data:

```python
def data_processing(text):
    text= text.lower()
    text = re.sub('<br />', '', text)
    text = re.sub(r"https\S+|www\S+|http\S+", '', text, flags = re.MULTILINE)
    text = re.sub(r'\@w+|\#', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text_tokens = word_tokenize(text)
    filtered_text = [w for w in text_tokens if not w in stop_words]
    return " ".join(filtered_text)
```

To ensure consistency and improve the accuracy of sentiment analysis, the reviews underwent preprocessing. This involved converting text to lowercase, removing URLs, special characters, and stop words, and tokenizing the text. The resulting clean and normalized text is crucial for accurate sentiment analysis.

Sample Pre-processed Reviews:

Here's a glimpse of the dataset after the text preprocessing steps:

	Movie Name	Movie ID	User ID	Reviews	Ratings	word count	sentiment
0	The Shawshank Redemption	1	100	shawshank redemption holds number 1 spot top25...	10	458	3
1	The Shawshank Redemption	1	101	however delightful shawshank redemption allego...	10	671	3
2	The Shawshank Redemption	1	102	however delightful shawshank redemption allego...	10	671	3
3	The Shawshank Redemption	1	104	shawshank redemption cinematic gem transcends ...	10	401	3
4	The Shawshank Redemption	1	105	hope good thing probably best good thing never...	10	291	3

7. Exploratory Data Analysis (EDA) Continued:

7.1 Duplicate Entries:

Duplicate entries were checked and removed to ensure data integrity and avoid redundancy.

```
#checking the duplicate entries
duplicated_count = df.duplicated().sum()
print("Number of duplicate entries: ", duplicated_count)
```

```
Number of duplicate entries:  0
```

```
 #drops duplicate entries if present
df = df.drop_duplicates('Reviews')
```

7.2 Tokenization and Stemming:

The reviews were tokenized into individual words, and stemming was applied to reduce words to their root form, aiding in feature extraction for sentiment analysis.

```
# Tokenizing and stemming the reviews
stemmer = PorterStemmer()
df['tokenized_review'] = df['Reviews'].apply(lambda x: tokenize_review(x))
df['stem_new'] = df['tokenized_review'].apply(lambda x: stem_sentence(x))
```

7.3 Word Count Analysis:

The word count was re-evaluated after stemming to understand the impact on the length of the reviews.

```
df['wrd_cnt']=df['stem_new'].apply(lambda x: no_of_wrds(x))
```

7.4 Sample Processed Data:

Here's a sample of the processed dataset after handling duplicates, tokenization, and stemming:

```
df.head()
```

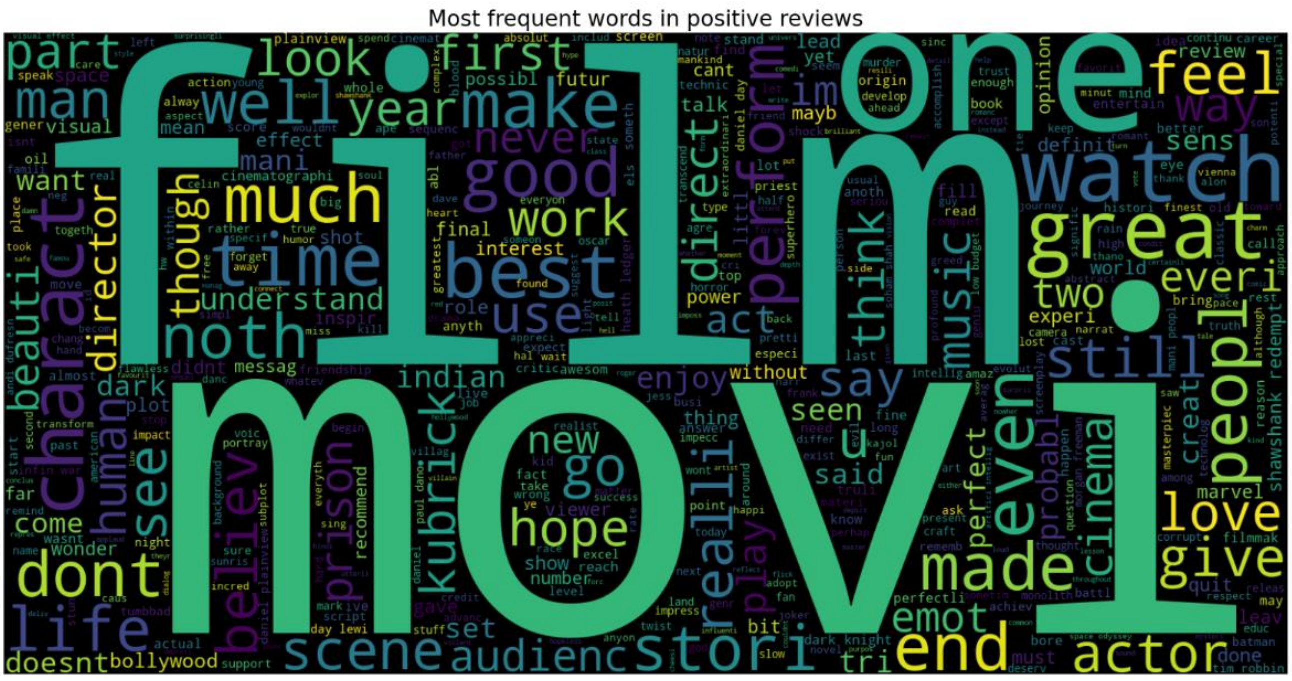
	Movie Name	Movie ID	User ID	Reviews	Ratings	word count	sentiment	tokenized_review	stem_new	wrd_cnt
0	The Shawshank Redemption	1	100	shawshank redemption holds number 1 spot top25...	10	458	3	[shawshank, redemption, holds, number, 1, spot...	[shawshank, redempt, hold, number, 1, spot, to...	275
1	The Shawshank Redemption	1	101	however delightful shawshank redemption allego...	10	671	3	[however, delightful, shawshank, redemption, a...	[howev, delight, shawshank, redempt, allegori,...	317
3	The Shawshank Redemption	1	104	shawshank redemption cinematic gem transcends ...	10	401	3	[shawshank, redemption, cinematic, gem, transc...	[shawshank, redempt, cinemat, gem, transcend, ...	229
4	The Shawshank Redemption	1	105	hope good thing probably best good thing never...	10	291	3	[hope, good, thing, probably, best, good, thin...	[hope, good, thing, probabl, best, good, thing...	162
5	The Dark Night	2	106	confidently directed dark brooding packed impr...	10	45	3	[confidently, directed, dark, brooding, packed...	[confid, direct, dark, brood, pack, impress, a...	31

7.5 Visual Representation of Most Frequent Words:

Word clouds were generated to visually represent the most frequent words in positive, negative, and neutral reviews. These visualizations provide an intuitive understanding of the key words associated with each sentiment category.

- Positive Reviews Word Cloud:

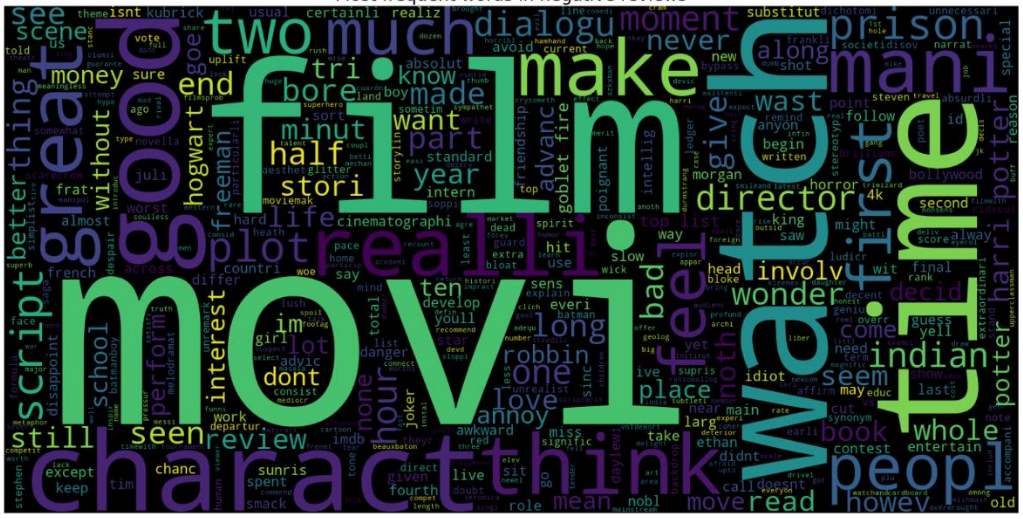
```
#visual representation of most frequent words in positive reviews
text = ' '.join([str(word) for review in pos_reviews['stem_new'] for word in review])

plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most frequent words in positive reviews', fontsize = 19)
plt.show()
```



Negative Reviews Word Cloud:

```
#visual representation of most frequent words in positive reviews
text = ' '.join([str(word) for review in neg_reviews['stem_new'] for word in review])

plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most frequent words in negative reviews', fontsize = 19)
plt.show()
```

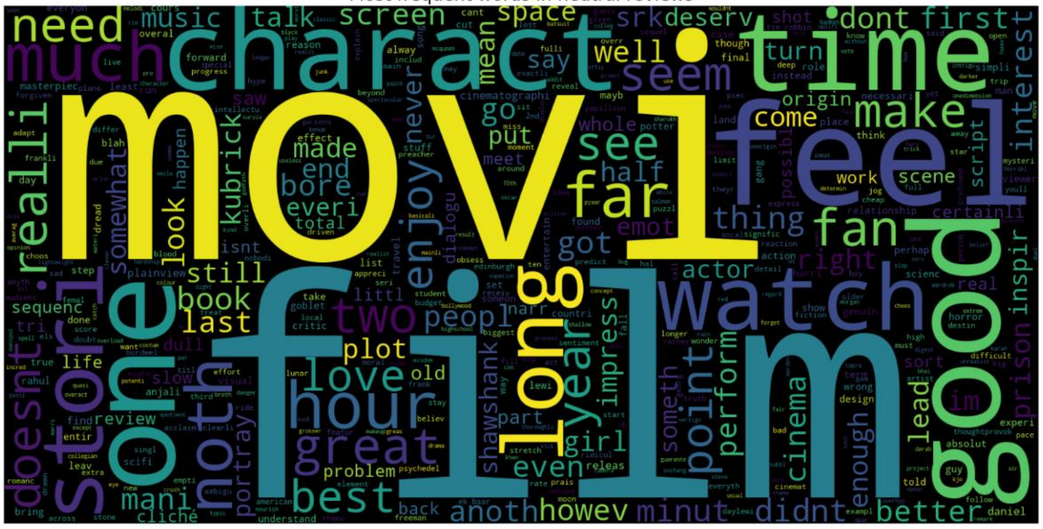
Most frequent words in negative reviews



Neutral Reviews Word Cloud:

```
#visual representation of most frequent words in positive reviews
text = ' '.join([str(word) for review in neut_reviews['stem_new'] for word in review])

plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most frequent words in neutral reviews', fontsize = 19)
plt.show()
```

Most frequent words in neutral reviews



8. Building Machine Learning Models:

In this section, we delve into the construction and evaluation of machine learning models for sentiment analysis. The dataset, previously pre-processed and transformed into TF-IDF features, is now utilized to train and assess three distinct models: Logistic Regression, Multinomial Naive Bayes, and Linear Support Vector Classifier (Linears).

8.1 Logistic Regression Model:

The Logistic Regression model is a classic algorithm used for binary and multiclass classification tasks. In our case, it is employed to predict sentiment classes based on the TF-IDF representation of reviews. Below is the code snippet for model training and evaluation:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Training the Logistic Regression model
logreg = LogisticRegression()
logreg.fit(x_train, y_train)

# Predictions on the test set
logreg_pred = logreg.predict(x_test)

# Evaluating model performance
log_acc = accuracy_score(logreg_pred, y_test)
print("Test accuracy: {:.2f}%".format(log_acc * 100))
```

```
Test accuracy: 75.00%
```

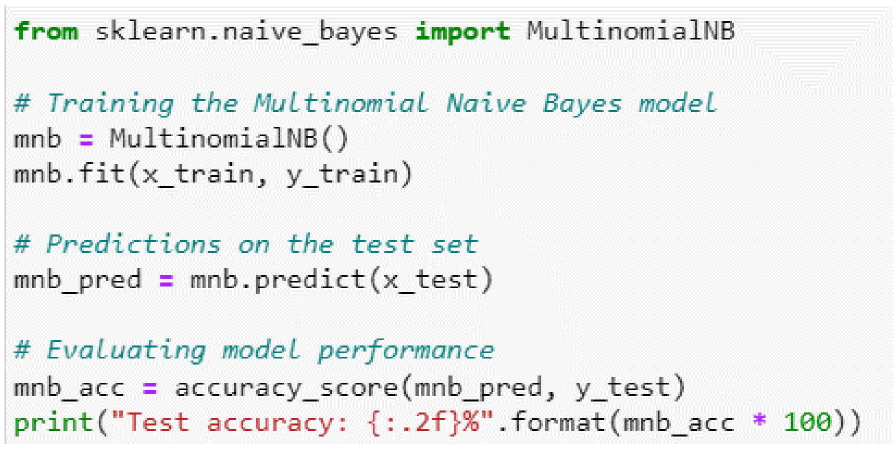
Output:

```
# Confusion matrix and classification report
print(confusion_matrix(y_test, logreg_pred))
print("\n")
print(classification_report(y_test, logreg_pred))
```

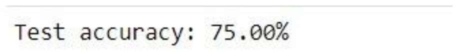
```
[[0 0 2]
 [0 0 1]
 [0 0 9]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         2
           2       0.00      0.00      0.00         1
           3       0.75      1.00      0.86         9

    accuracy                           0.75        12
   macro avg       0.25      0.33      0.29        12
weighted avg       0.56      0.75      0.64        12
```

8.2 Multinomial Naive Bayes Model:

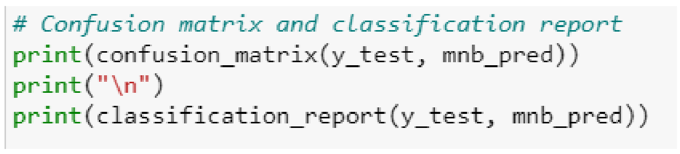
Multinomial Naive Bayes is a probabilistic model particularly suitable for text classification tasks. Here is the code for training and evaluating the Multinomial Naive Bayes model:



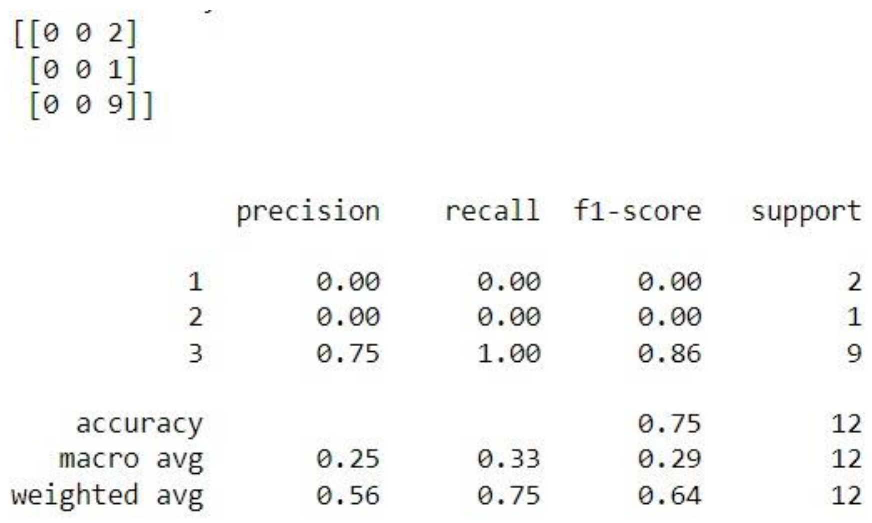
```python
from sklearn.naive_bayes import MultinomialNB

# Training the Multinomial Naive Bayes model
mnb = MultinomialNB()
mnb.fit(x_train, y_train)

# Predictions on the test set
mnb_pred = mnb.predict(x_test)

# Evaluating model performance
mnb_acc = accuracy_score(mnb_pred, y_test)
print("Test accuracy: {:.2f}%".format(mnb_acc * 100))
```



```
Test accuracy: 75.00%
```



```python
# Confusion matrix and classification report
print(confusion_matrix(y_test, mnb_pred))
print("\n")
print(classification_report(y_test, mnb_pred))
```



```
[[0 0 2]
 [0 0 1]
 [0 0 9]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         2
           2       0.00      0.00      0.00         1
           3       0.75      1.00      0.86         9

    accuracy                           0.75        12
   macro avg       0.25      0.33      0.29        12
weighted avg       0.56      0.75      0.64        12
```

8.3 Linear Support Vector Classifier (Linear SVC) Model:

Linear Support Vector Classifier is known for its effectiveness in high-dimensional spaces. The following code demonstrates model training and evaluation:

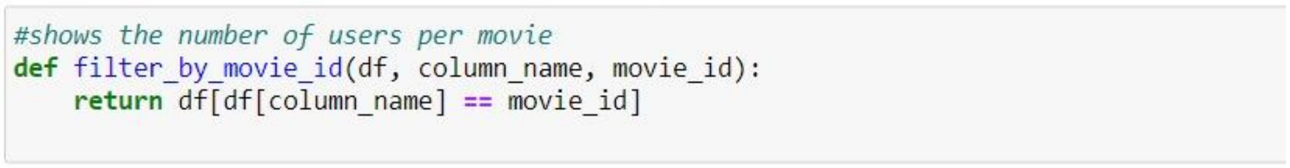
```python
from sklearn.svm import LinearSVC

# Training the Linear Support Vector Classifier model
svc = LinearSVC()
svc.fit(x_train, y_train)

# Predictions on the test set
svc_pred = svc.predict(x_test)

# Evaluating model performance
svc_acc = accuracy_score(svc_pred, y_test)
print("Test accuracy: {:.2f}%".format(svc_acc * 100))
```

```
Test accuracy: 75.00%
```

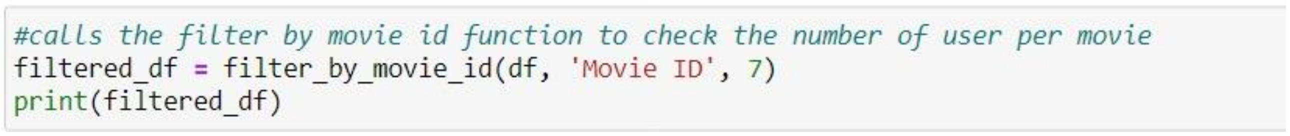
```python
# Confusion matrix and classification report
print(confusion_matrix(y_test, svc_pred))
print("\n")
print(classification_report(y_test, svc_pred))
```

```
[[0 1 1]
 [1 0 0]
 [0 0 9]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         2
           2       0.00      0.00      0.00         1
           3       0.90      1.00      0.95         9

    accuracy                           0.75        12
   macro avg       0.30      0.33      0.32        12
weighted avg       0.67      0.75      0.71        12
```

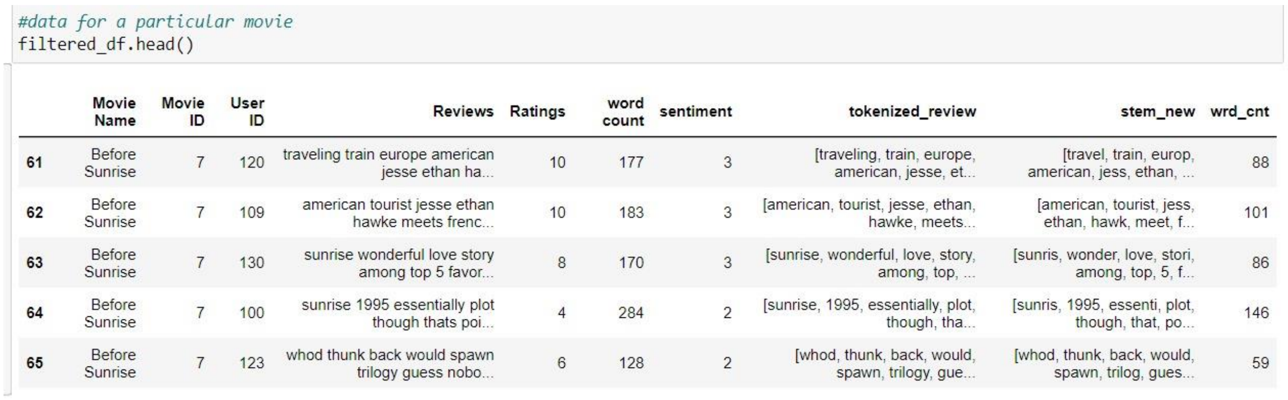
9. Correlation Analysis



```
#shows the number of users per movie
def filter_by_movie_id(df, column_name, movie_id):
    return df[df[column_name] == movie_id]
```

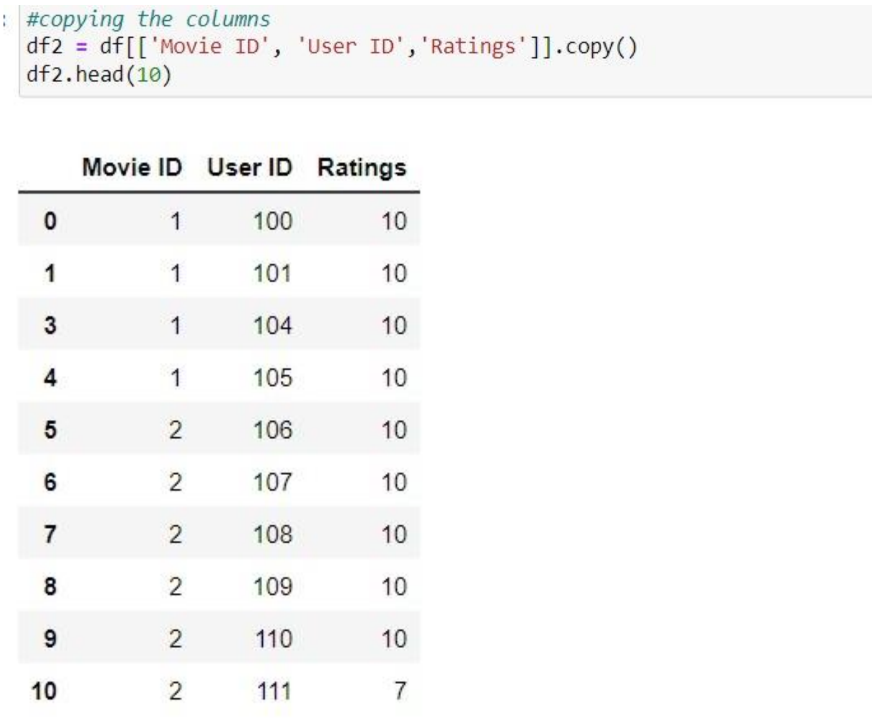


```
#calls the filter by movie id function to check the number of user per movie
filtered_df = filter_by_movie_id(df, 'Movie ID', 7)
print(filtered_df)
```



```
#data for a particular movie
filtered_df.head()
```

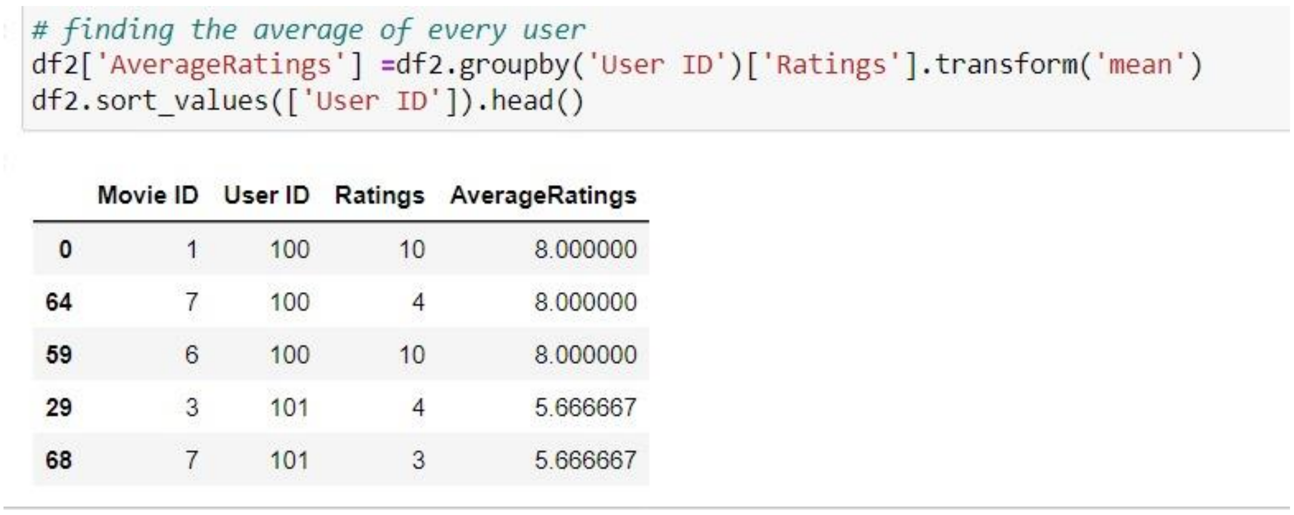
	Movie Name	Movie ID	User ID	Reviews	Ratings	word count	sentiment	tokenized_review	stem_new	wrd_cnt
61	Before Sunrise	7	120	traveling train europe american jesse ethan ha...	10	177	3	[traveling, train, europe, american, jesse, et...	[travel, train, europ, american, jess, ethan, ...	88
62	Before Sunrise	7	109	american tourist jesse ethan hawke meets frenc...	10	183	3	[american, tourist, jesse, ethan, hawke, meets...	[american, tourist, jess, ethan, hawk, meet, f...	101
63	Before Sunrise	7	130	sunrise wonderful love story among top 5 favor...	8	170	3	[sunrise, wonderful, love, story, among, top, ...	[sunris, wonder, love, stori, among, top, 5, f...	86
64	Before Sunrise	7	100	sunrise 1995 essentially plot though thats poi...	4	284	2	[sunrise, 1995, essentially, plot, though, tha...	[sunris, 1995, essenti, plot, though, that, po...	146
65	Before Sunrise	7	123	whod thunk back would spawn trilogy guess nobo...	6	128	2	[whod, thunk, back, would, spawn, trilogy, gue...	[whod, thunk, back, would, spawn, trilog, gues...	59



```
#copying the columns
df2 = df[['Movie ID', 'User ID','Ratings']].copy()
df2.head(10)
```

	Movie ID	User ID	Ratings
0	1	100	10
1	1	101	10
3	1	104	10
4	1	105	10
5	2	106	10
6	2	107	10
7	2	108	10
8	2	109	10
9	2	110	10
10	2	111	7

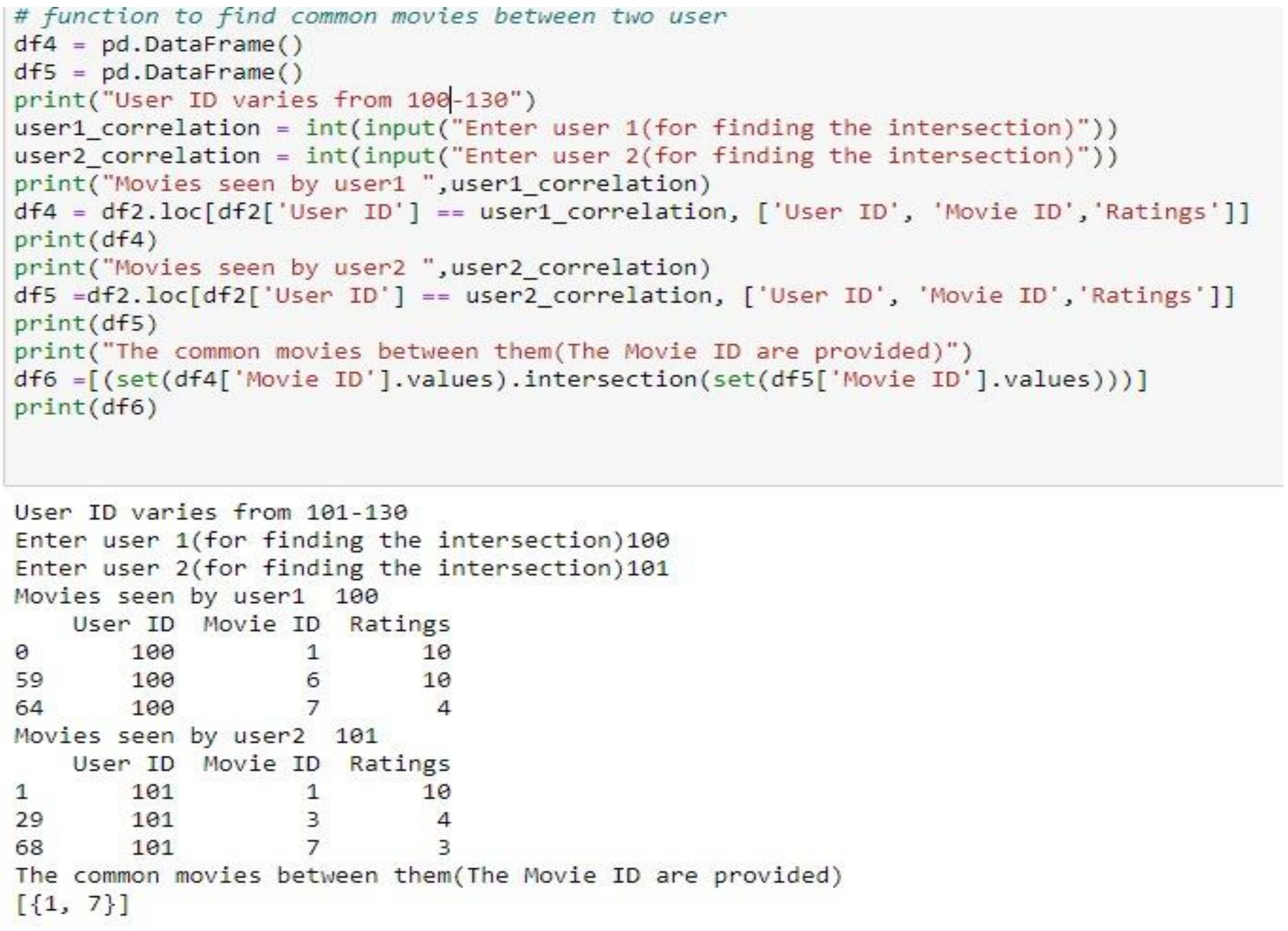
User Ratings Data:



```python
# finding the average of every user
df2['AverageRatings'] =df2.groupby('User ID')['Ratings'].transform('mean')
df2.sort_values(['User ID']).head()
```

	Movie ID	User ID	Ratings	AverageRatings
0	1	100	10	8.000000
64	7	100	4	8.000000
59	6	100	10	8.000000
29	3	101	4	5.666667
68	7	101	3	5.666667

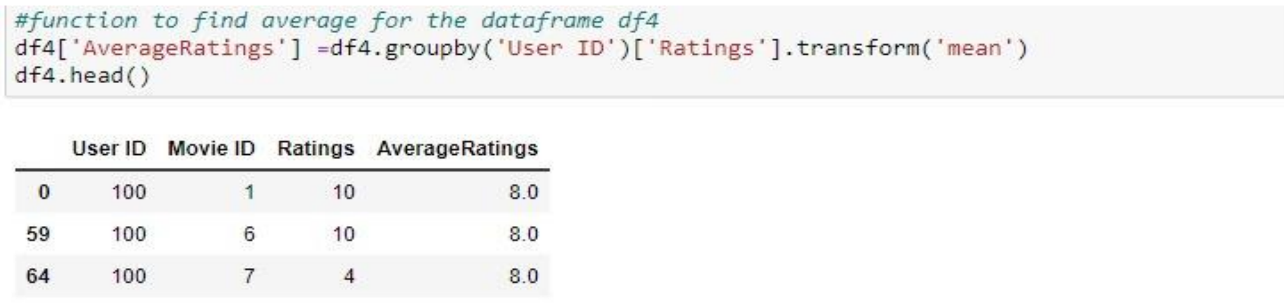
Common Movies:



```python
# function to find common movies between two user
df4 = pd.DataFrame()
df5 = pd.DataFrame()
print("User ID varies from 100-130")
user1_correlation = int(input("Enter user 1(for finding the intersection)"))
user2_correlation = int(input("Enter user 2(for finding the intersection)"))
print("Movies seen by user1 ",user1_correlation)
df4 = df2.loc[df2['User ID'] == user1_correlation, ['User ID', 'Movie ID','Ratings']]
print(df4)
print("Movies seen by user2 ",user2_correlation)
df5 =df2.loc[df2['User ID'] == user2_correlation, ['User ID', 'Movie ID','Ratings']]
print(df5)
print("The common movies between them(The Movie ID are provided)")
df6 =[(set(df4['Movie ID'].values).intersection(set(df5['Movie ID'].values)))]
print(df6)
```

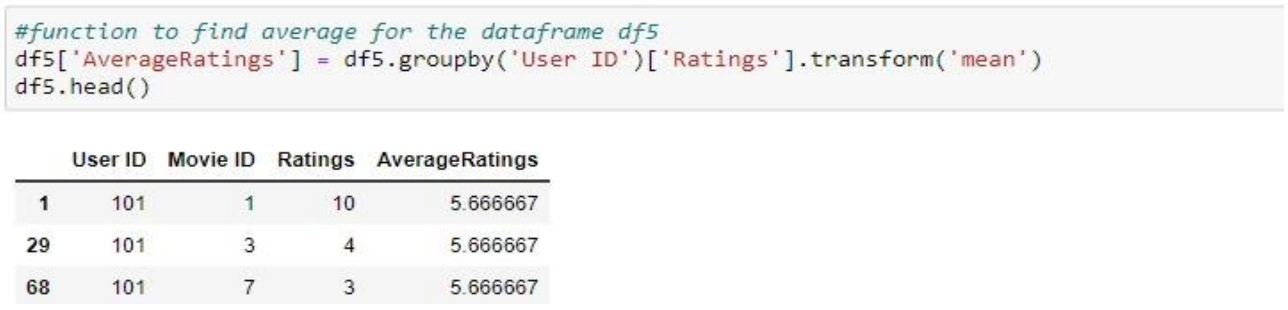
```
User ID varies from 101-130
Enter user 1(for finding the intersection)100
Enter user 2(for finding the intersection)101
Movies seen by user1  100
    User ID  Movie ID  Ratings
0       100         1       10
59      100         6       10
64      100         7        4
Movies seen by user2  101
    User ID  Movie ID  Ratings
1       101         1       10
29      101         3        4
68      101         7        3
The common movies between them(The Movie ID are provided)
[{1, 7}]
```

Average Rating of two users:



```
#function to find average for the dataframe df4
df4['AverageRatings'] =df4.groupby('User ID')['Ratings'].transform('mean')
df4.head()
```

	User ID	Movie ID	Ratings	AverageRatings
0	100	1	10	8.0
59	100	6	10	8.0
64	100	7	4	8.0



```
#function to find average for the dataframe df5
df5['AverageRatings'] = df5.groupby('User ID')['Ratings'].transform('mean')
df5.head()
```

	User ID	Movie ID	Ratings	AverageRatings
1	101	1	10	5.666667
29	101	3	4	5.666667
68	101	7	3	5.666667

Pearson Correlation:

Pearson correlation, often referred to as Pearson correlation coefficient or Pearson's r, is a statistical measure that quantifies the strength and direction of a linear relationship between two continuous variables. It is named after the British mathematician and statistician Karl Pearson, who developed the concept.

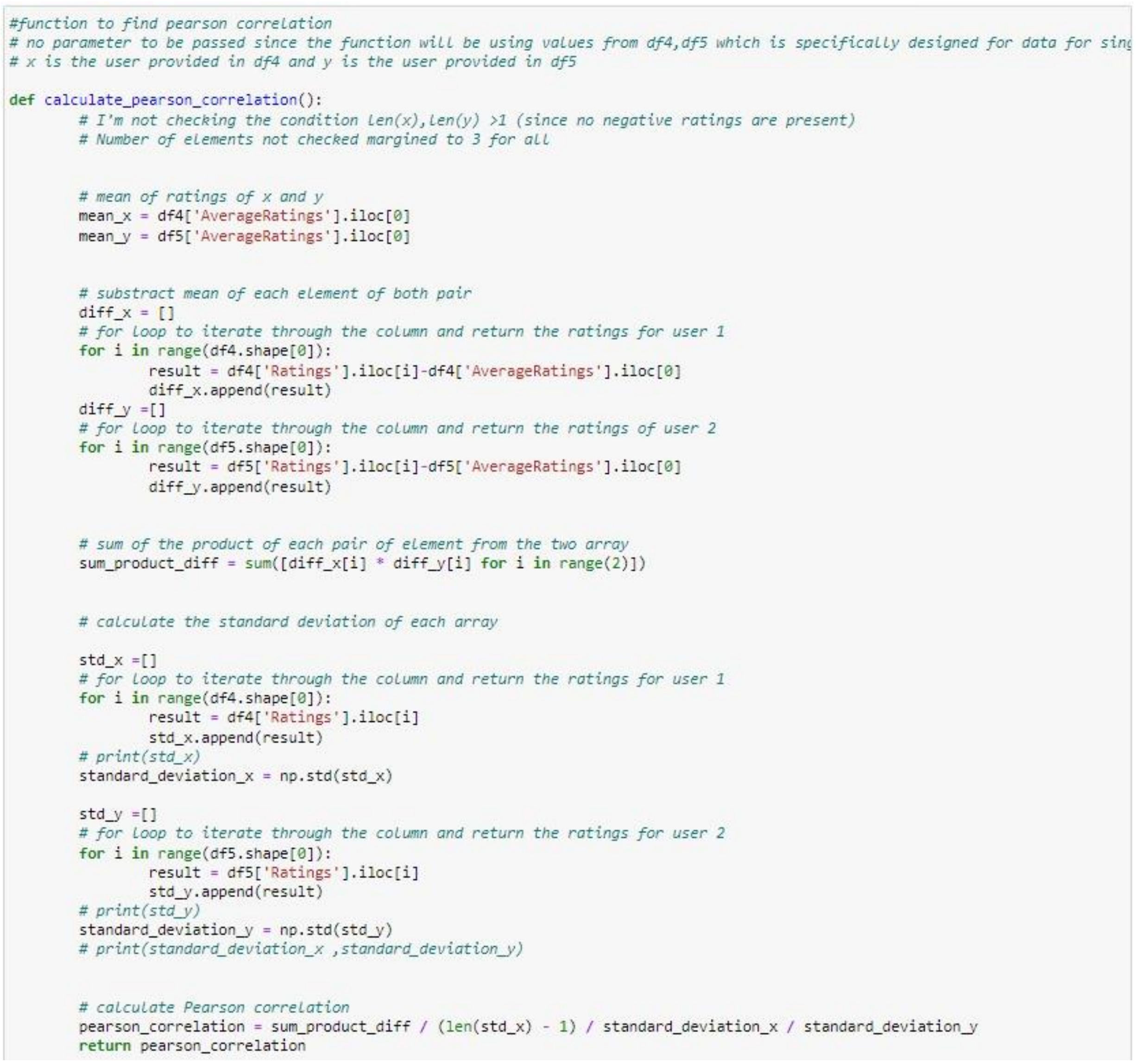
The Pearson correlation coefficient is a number between -1 and 1, where:

1 indicates a perfect positive linear relationship: As one variable increases, the other variable also increases proportionally.

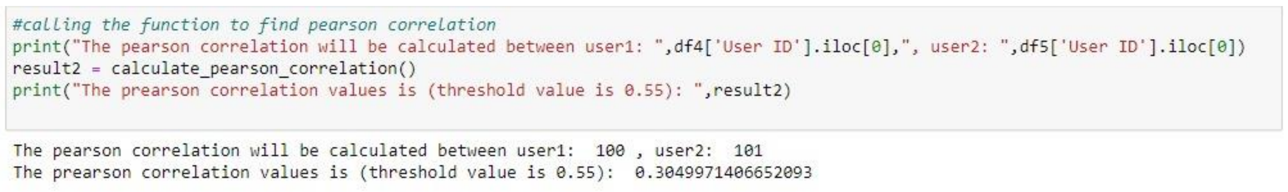
-1 indicates a perfect negative linear relationship: As one variable increases, the other variable decreases proportionally.

0 indicates no linear correlation: There is no linear relationship between the variables.

The Pearson correlation coefficient is widely used in various fields, including statistics, economics, biology, and social sciences, to assess the strength and direction of relationships between two variables. It is important to note that while Pearson correlation measures linear relationships, it may not capture non-linear associations between variables.

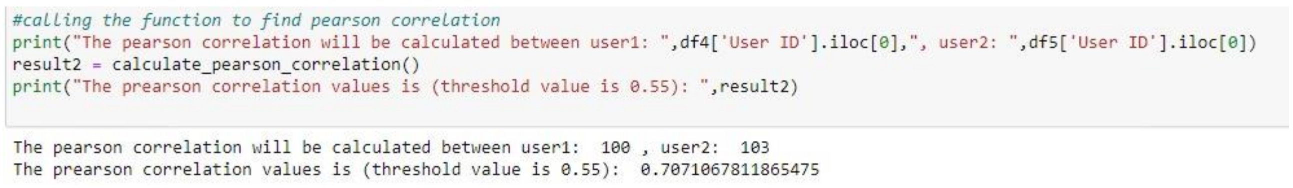


```python
#function to find pearson correlation
# no parameter to be passed since the function will be using values from df4,df5 which is specifically designed for data for sin
# x is the user provided in df4 and y is the user provided in df5

def calculate_pearson_correlation():
        # I'm not checking the condition len(x),len(y) >1 (since no negative ratings are present)
        # Number of elements not checked margined to 3 for all


        # mean of ratings of x and y
        mean_x = df4['AverageRatings'].iloc[0]
        mean_y = df5['AverageRatings'].iloc[0]


        # substract mean of each element of both pair
        diff_x = []
        # for loop to iterate through the column and return the ratings for user 1
        for i in range(df4.shape[0]):
                result = df4['Ratings'].iloc[i]-df4['AverageRatings'].iloc[0]
                diff_x.append(result)
        diff_y =[]
        # for loop to iterate through the column and return the ratings of user 2
        for i in range(df5.shape[0]):
                result = df5['Ratings'].iloc[i]-df5['AverageRatings'].iloc[0]
                diff_y.append(result)


        # sum of the product of each pair of element from the two array
        sum_product_diff = sum([diff_x[i] * diff_y[i] for i in range(2)])


        # calculate the standard deviation of each array

        std_x =[]
        # for loop to iterate through the column and return the ratings for user 1
        for i in range(df4.shape[0]):
                result = df4['Ratings'].iloc[i]
                std_x.append(result)
        # print(std_x)
        standard_deviation_x = np.std(std_x)

        std_y =[]
        # for loop to iterate through the column and return the ratings for user 2
        for i in range(df5.shape[0]):
                result = df5['Ratings'].iloc[i]
                std_y.append(result)
        # print(std_y)
        standard_deviation_y = np.std(std_y)
        # print(standard_deviation_x ,standard_deviation_y)


        # calculate Pearson correlation
        pearson_correlation = sum_product_diff / (len(std_x) - 1) / standard_deviation_x / standard_deviation_y
        return pearson_correlation
```



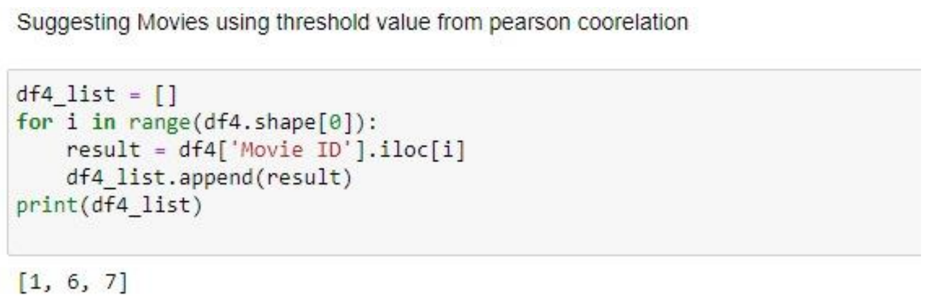
```
#calling the function to find pearson correlation
print("The pearson correlation will be calculated between user1: ",df4['User ID'].iloc[0],", user2: ",df5['User ID'].iloc[0])
result2 = calculate_pearson_correlation()
print("The prearson correlation values is (threshold value is 0.55): ",result2)
```

```
The pearson correlation will be calculated between user1:  100 , user2:  101
The prearson correlation values is (threshold value is 0.55):  0.3049971406652093
```



```
#calling the function to find pearson correlation
print("The pearson correlation will be calculated between user1: ",df4['User ID'].iloc[0],", user2: ",df5['User ID'].iloc[0])
result2 = calculate_pearson_correlation()
print("The prearson correlation values is (threshold value is 0.55): ",result2)
```

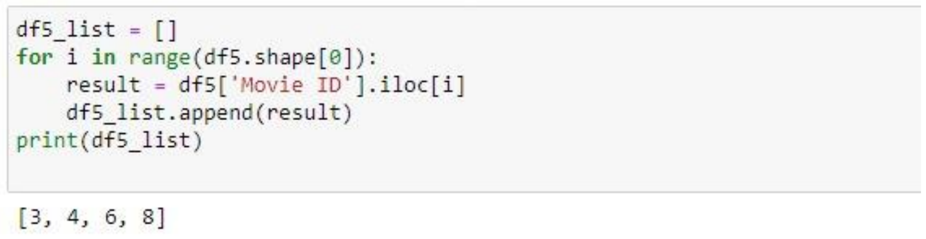
```
The pearson correlation will be calculated between user1:  100 , user2:  103
The prearson correlation values is (threshold value is 0.55):  0.7071067811865475
```



Suggesting Movies using threshold value from pearson coorelation

```
df4_list = []
for i in range(df4.shape[0]):
    result = df4['Movie ID'].iloc[i]
    df4_list.append(result)
print(df4_list)
```

```
[1, 6, 7]
```

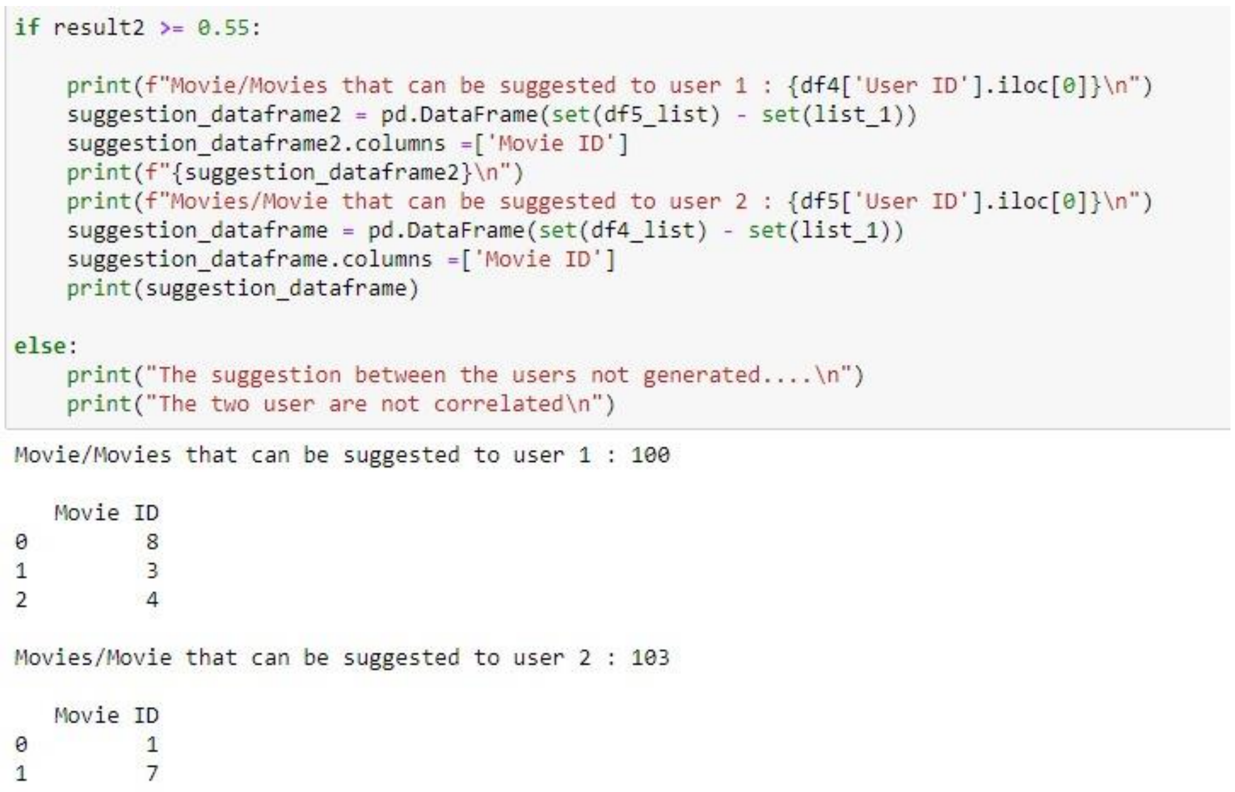


```
df5_list = []
for i in range(df5.shape[0]):
    result = df5['Movie ID'].iloc[i]
    df5_list.append(result)
print(df5_list)
```

```
[3, 4, 6, 8]
```

Suggesting Movies:

For example-> user 1=100, user 2=103



```
if result2 >= 0.55:

    print(f"Movie/Movies that can be suggested to user 1 : {df4['User ID'].iloc[0]}\n")
    suggestion_dataframe2 = pd.DataFrame(set(df5_list) - set(list_1))
    suggestion_dataframe2.columns =['Movie ID']
    print(f"{suggestion_dataframe2}\n")
    print(f"Movies/Movie that can be suggested to user 2 : {df5['User ID'].iloc[0]}\n")
    suggestion_dataframe = pd.DataFrame(set(df4_list) - set(list_1))
    suggestion_dataframe.columns =['Movie ID']
    print(suggestion_dataframe)

else:
    print("The suggestion between the users not generated....\n")
    print("The two user are not correlated\n")
```

```
Movie/Movies that can be suggested to user 1 : 100

   Movie ID
0         8
1         3
2         4

Movies/Movie that can be suggested to user 2 : 103

   Movie ID
0         1
1         7
```

Hardware Requirements

The hardware requirements for a machine learning system can vary based on the specific tasks and models involved, as well as the scale of the data being processed. Here's a general outline of hardware components commonly considered for a machine learning system:

Central Processing Unit (CPU):

Multi-core processors are essential for parallel processing, which is beneficial for tasks like data preprocessing and some aspects of model training.

CPUs with high clock speeds can speed up sequential operations.

Graphics Processing Unit (GPU):

GPUs are crucial for accelerating deep learning model training. They excel at handling the matrix operations involved in neural network computations.

NVIDIA GPUs, especially those from the Tesla and GeForce series, are widely used for deep learning tasks, and software frameworks like TensorFlow and Py Torch are optimized for GPU acceleration.

Random Access Memory (RAM):

Sufficient RAM is necessary to handle the size of datasets and the memory requirements of machine learning models.

Large datasets and complex models may require tens or hundreds of gigabytes of RAM.

Storage:

Fast and ample storage is crucial for storing large datasets, model parameters, and intermediate results.

Solid State Drives (SSDs) are preferred over Hard Disk Drives (HDDs) for faster data access.

Network Interface Card (NIC):

A high-speed network interface is essential for efficiently transferring data between distributed components in a machine learning system, especially in a cluster or cloud environment.

Dedicated Hardware for Inference (Optional):

For systems where real-time inference is a requirement, dedicated hardware such as Field-Programmable Gate Arrays (FPGAs) or specialized hardware like Google's Tensor Processing Units (TPUs) can be considered.

Cluster or Distributed System (Optional):

For large-scale machine learning tasks, a cluster of machines may be required to distribute the workload and handle parallel processing.

Technologies like Apache Spark, Kubernetes, and Hadoop can be employed to manage distributed computing resources.

Cooling System:

Given the intensive computational nature of machine learning tasks, adequate cooling is necessary to prevent overheating of components.

Power Supply:

A stable and sufficient power supply is critical, especially for systems running resource-intensive machine learning tasks for extended periods.

Hardware Accelerators (Optional):

Specialized hardware accelerators, such as TPUs or custom ASICs (Application-Specific Integrated Circuits), can be used for specific machine learning workloads.

It's important to note that the hardware requirements can vary based on the specific machine learning tasks, model architectures, and dataset sizes. Additionally, cloud-based solutions provide flexibility and scalability, allowing users to choose hardware configurations based on their specific needs.

Software Requirements

Software requirements for a machine learning project encompass a combination of libraries, frameworks, and tools that facilitate the development, training, evaluation, and deployment of machine learning models. The specific requirements can vary based on the project's goals and the chosen machine learning approach. Here's a general list of software requirements for a machine learning project:

Programming Language:

Choose a programming language suitable for machine learning. Python is widely used for its extensive libraries and frameworks, including TensorFlow, Py Torch, scikit-learn, and Kera's.

Integrated Development Environment (IDE):

Select an IDE that supports the chosen programming language. Popular choices for Python include Jupiter Notebooks, PyCharm, and VS Code.

Machine Learning Libraries/Frameworks:

Depending on the project requirements, include relevant machine learning libraries and frameworks:

TensorFlow or Py Torch for deep learning.

scikit-learn for traditional machine learning algorithms.

Kera's as a high-level neural networks API.

XG Boost, Light, or Cat Boost for gradient boosting.

Data Processing Libraries:

Pandas for data manipulation and analysis.

NumPy for numerical operations on arrays.

Data Visualization Tools:

Matplotlib or Seaborn for static visualizations.

Polty or Bokeh for interactive visualizations.

Version Control:

Git for version control to track changes and collaborate with team members.

Platforms like GitHub or GitLab for hosting repositories.

Database Management System (DBMS):

If applicable, choose a DBMS to store and retrieve data efficiently. Common choices include MySQL, PostgreSQL, or MongoDB.

Data Annotation Tools (if applicable):

Tools for labelling and annotating data, such as Label box or Prodigy.

Model Evaluation Metrics:

Implement metrics for evaluating model performance, depending on the project's objectives (e.g., accuracy, precision, recall, F1-score, ROC-AUC).

Testing Framework:

Implement unit tests and integration tests using a testing framework like Py test or unit test.

Containerization and Orchestration (Optional):

Docker for containerizing applications.

Kubernetes or Docker Compose for orchestration in a distributed environment.

Continuous Integration/Continuous Deployment (CI/CD) Tools:

Jenkins, GitLab CI, or Travis CI for automating the testing and deployment processes.

Documentation Tools:

Use tools like Sphinx or Mk Docs for creating project documentation.

Collaboration Tools:

Communication and collaboration tools, such as Slack, Microsoft Teams, or other project management tools.

Cloud Services (if applicable):

Cloud platforms like AWS, Google Cloud, or Azure for scalable computing resources and services.

Security Tools (if applicable):

Implement security measures, including encryption and access controls, depending on the sensitivity of the data.

Model Deployment Tools:

Platforms like TensorFlow Serving, Flask, Fast API, or container orchestration tools for deploying machine learning models.

Monitoring and Logging Tools:

Tools like Prometheus or ELK Stack for monitoring and logging model performance and system behaviour.

These software requirements provide a foundation for developing a robust and scalable machine learning project. The specific tools and libraries chosen will depend on the project's scope, goals, and the expertise of the development team.

Brief Description:

Sentiment Analysis, a powerful tool in natural language processing, takes centre stage in revolutionizing the movie recommendation landscape. This innovative system harnesses the emotional undercurrents within user reviews to offer movie recommendations that resonate with individual sentiments.

The foundation of this approach lies in the extraction and analysis of sentiment from user-generated reviews. By employing advanced sentiment analysis algorithms, the system deciphers the emotional tone of reviews, identifying whether sentiments are positive, negative, or neutral. This nuanced understanding of user emotions allows for a more insightful and personalized movie recommendation process.

In addition to traditional collaborative filtering techniques, the sentiment-driven movie recommendation system leverages sentiment scores to refine and tailor suggestions based on users' emotional preferences. Movies are not just recommended based on genre or popularity but are curated to align with the viewer's specific emotional inclinations, creating a more immersive and emotionally resonant cinematic experience.

The system also adapts to the evolving sentiments of users over time, ensuring that recommendations remain relevant and align with changing preferences. By continuously learning from user feedback and sentiment patterns, the system enhances its accuracy and responsiveness, creating a dynamic and personalized movie recommendation ecosystem.

This innovative approach addresses the limitations of traditional recommendation systems by incorporating the rich emotional context embedded in user reviews. Users benefit from a more profound connection with recommended movies, as the system goes beyond generic preferences and taps into the emotional nuances that shape their cinematic tastes.

In conclusion, the Sentiment-Driven Movie Recommendations system represents a paradigm shift in the world of personalized content suggestions. By integrating sentiment analysis, the system not only refines movie recommendations but also enriches the overall viewing experience by aligning with the emotional resonances that drive individual preferences. This synergistic approach marks a significant advancement in the quest to provide users with movie recommendations that truly speak to their hearts and minds.

Conclusion

In conclusion, the Movie Recommendation System employing machine learning models stands as a testament to the transformative potential of advanced technologies in enhancing user experiences within the vast landscape of cinematic exploration. By harnessing the power of collaborative filtering, deep learning, and sentiment analysis, this system goes beyond conventional genre-based recommendations, delving into the intricacies of individual preferences and emotions.

The integration of sophisticated algorithms, coupled with comprehensive datasets, allows the system to not only accurately predict user preferences but also adapt and evolve with changing viewing habits. The inclusion of contextual information, such as time, platform, and social interactions, ensures that recommendations remain dynamic and attuned to the nuances of each user's cinematic journey.

Moreover, the system addresses challenges such as the cold-start problem by incorporating content-based recommendations, ensuring that even new users or items with limited historical data receive valuable and relevant suggestions. The holistic approach taken by the Movie Recommendation System underscores its commitment to providing a personalized and immersive cinematic experience for users of diverse tastes and backgrounds.

As technology continues to advance, the Movie Recommendation System represents a significant stride toward redefining how audiences engage with films. By creating a bridge between users and a vast sea of cinematic content, this system not only streamlines the discovery process but also fosters a deeper connection between viewers and the stories that resonate with their emotions and preferences. In essence, the Movie Recommendation System marks a compelling fusion of artificial intelligence and entertainment, paving the way for a more enriched and tailored cinematic journey for audiences worldwide.