Title of the project: MOVIE RECOMMENDATION SYSTEM

Objective:1.Enhancing User Experience: By recommending movies that align with users' tastes and preferences, the system aims to enhance user satisfaction and engagement with the platform.

2.Increasing Engagement and Retention: A well-performing recommendation system can increase user engagement by keeping users interested and exploring more content, thus improving platform retention.

3.Optimizing Content Discovery: Users often face difficulty in discovering relevant content from vast libraries. The recommendation system helps in efficiently surfacing movies that users are likely to enjoy but might not have discovered otherwise.

4.Driving Revenue: In platforms with monetization models such as streaming services, effective recommendations can lead to increased consumption and potentially higher revenue through subscriptions or ad views.

5.Personalization: The system aims to tailor recommendations based on individual user behavior, preferences, past interactions, demographics, and contextual factors to deliver a personalized experience.

6.Scalability and Efficiency: Recommendations should be generated in real-time or near real-time, scalable to handle large user bases and diverse content catalogs, and efficient in computation and resource utilization.

Overall, the key objective of a movie recommendation system is to create a seamless and personalized experience that encourages users to discover and consume relevant content, thereby benefiting both users and content providers/ platforms.

```
#IMPORT LIBRARY
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text
import TfidfVectorizer
from sklearn.metrics.pairwise import
cosine_similarity
from sklearn.model_selection import
train_test_split
from sklearn.ensemble import
RandomForestRegressor
from sklearn.metrics import
mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns



#IMPORT DATA
# Assuming 'movies.csv' contains
columns: 'title', 'genre', 'rating'
def import_data(file_path):
    return pd.read_csv(file_path)

# Importing data
file_path = 'movies.csv'  # Replace
with your file path
movies_data = import_data(file_path)
```

```
#DESCRIBE DATA
# Describe the data
def describe_data(data):
    return data.describe()

# Example of data description
data_description =
describe_data(movies_data)
print(data_description)



#DATA VISUALIZATION
# Data visualization
def visualize_data(data):
    sns.histplot(data['rating'],
kde=True)
    plt.title('Distribution of Ratings')
    plt.xlabel('Rating')
    plt.ylabel('Count')
    plt.show()

# Example of data visualization
visualize_data(movies_data)
```

```
#DATA PREPROCESSING
# Data preprocessing
def preprocess_data(data):
    # Drop rows with missing values
    data = data.dropna(subset=['title',
'genre', 'rating'])

    # Convert genres to lowercase
    data['genre'] =
data['genre'].str.lower()

    return data

# Example of data preprocessing
movies_data =
preprocess_data(movies_data)



#DEFINE TARGET VARIABLE AND
FEATURE VARIABLE
# Define target variable
X = movies_data[['title', 'genre']]
y = movies_data['rating']
```

```
#TRAIN TEST SPLIT
# Train-test split
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)



#MODELING
# Modeling (dummy example with
RandomForestRegressor)
tfidf_vectorizer =
TfidfVectorizer(stop_words='english')
X_train_tfidf = tfidf_vectorizer.fit_tran
sform(X_train['title'])

model = RandomForestReg
ressor(n_estimators=100,
random_state=42)
model.fit(X_train_tfidf, y_train)



# Model evaluation
X_test_tfidf = tfidf_vectorizer.transfor
m(X_test['title'])
predictions =
model.predict(X_test_tfidf)
```

```
mse = mean_squared_error(y_test,
predictions)
print(f'Mean Squared Error: {mse}')



# Prediction example
def predict_rating(movie_title, model,
vectorizer):
    movie_tfidf =
vectorizer.transform([movie_title])
    rating = model.predict(movie_tfidf)
    return rating[0]

# Example prediction
movie_title = 'The Dark Knight'
predicted_rating =
predict_rating(movie_title, model,
tfidf_vectorizer)
print(f"Predicted rating for
'{movie_title}': {predicted_rating}")
```

Explanation: Importing Libraries: Libraries such as pandas for data manipulation, numpy for numerical computations, sklearn for machine learning functionalities, and matplotlib/seaborn for data visualization are imported.

Importing Data: The import_data function reads data from a CSV file (movies.csv), assuming it contains columns like 'title', 'genre', and 'rating'.

Describe Data: The describe_data function provides basic statistical information about the dataset using describe() method of pandas DataFrame.

Data Visualization: The visualize_data function uses matplotlib and seaborn to create a histogram showing the distribution of movie ratings.

Data Preprocessing: The preprocess_data function handles missing values, converts genres to lowercase, and performs any other necessary data cleaning steps.

Define Target Variable: The target variable (y) is defined as the 'rating' column from the dataset, while the features (X) include 'title' and 'genre'.

Train-Test Split: The dataset is split into training and testing sets using train_test_split from sklearn.model_selection.

Modeling: A RandomForestRegressor model is trained using movie titles (after converting them to TF-IDF vectors) to predict ratings.

Model Evaluation: The trained model's performance is evaluated using Mean Squared Error (mean_squared_error from sklearn.metrics).

Prediction: The predict_rating function predicts the rating for a given movie title using the trained model and TF-IDF vectorizer.

This example covers the entire pipeline from data import to model evaluation and prediction for a movie recommendation system. Depending on the specific requirements and dataset, you may need to adjust and enhance different parts of the code accordingly.

3:35 pm