```
import pandas as pd
from sklearn.neighbors import NearestNeighbors
from sklearn.model_selection import train_test_split
import numpy as np


# Load in the dataset
data = pd.read_csv("rating.csv")

# Split data into training (70%) and testing (30%) sets
trainingData, testingData = train_test_split(data, test_size=0.3, random_state=42)


# Load in movie dataset
movieData = pd.read_csv("movie.csv")

# Create dictionary
movieDict = movieData.set_index('movieId')[['title', 'genres']].to_dict(orient='index')
```

```
{1: {'title': 'Toy Story (1995)', 'genres': 'Adventure|Animation|Children|Comedy|Fantasy'}, 2: {'title': 'Jumanji (1995)', 'genres': 'Adventure|Childre
```

```
# Create user-item matrices for both training and test sets
trainingMatrix = trainingData.pivot(index='userId', columns='movieId', values='rating').fillna(0)
testingMatrix = testingData.pivot(index='userId', columns='movieId', values='rating').fillna(0)


# Fit the NearestNeighbors model on the training data
knn = NearestNeighbors(metric='cosine', algorithm='brute')
knn.fit(trainingMatrix)
```

```
NearestNeighbors
NearestNeighbors(algorithm='brute', metric='cosine')
```

```
# Evaluate the model by predicting ratings for users in the test set
def recommendMovies(userId, n_recommendations=5):
    # Ensure the user exists in the test set
    if userId not in testingMatrix.index:
        return f"User {userId} not found."

    # Select the target user
    targetUser = trainingMatrix.loc[[userId]] if userId in trainingMatrix.index else None
    if targetUser is None:
        return f"User {userId} not found."

    # Find similar users to the target user
    distances, indices = knn.kneighbors(targetUser, n_neighbors=3)
    similarUsers = indices.flatten()[1:]  # Exclude the target user themselves

    # Aggregate ratings from similar users for movies the target user hasn't rated
    similarUsersRatings = trainingMatrix.iloc[similarUsers]
    unratedMovies = trainingMatrix.loc[userId] == 0.0
    recommendedMovies = similarUsersRatings.mean(axis=0)[unratedMovies]
    recommendedMovies = recommendedMovies.sort_values(ascending=False)

    return recommendedMovies.head(n_recommendations)


# Example: Recommend movies for a user in the test set
userId = 1  # Replace with an actual user ID from your test set
recommendedMovies = recommendMovies(userId)

# Convert series to list of pairs (movieId & rating)
moviePairs = list(recommendedMovies.items())

print(f"Recommended movies for user {userId}:\n")
for index in range(len(moviePairs)) :
    print(f"{movieDict[moviePairs[index][0]]['title']:>25} : {movieDict[moviePairs[index][0]]['genres']}")
```

```
 Recommended movies for user 1:

            Braveheart (1995) : Action|Drama|War
      Schindler's List (1993) : Drama|War
             King Kong (1933) : Action|Adventure|Fantasy|Horror
       Green Mile, The (1999) : Crime|Drama
             Gladiator (2000) : Action|Adventure|Drama
```