HARPY AEROSPACE INTERNSHIP

AIoT Zone Recommendation System

Prepared by :

Surya K S
2022510022
Madras Institute of Technology
B. Tech AI and DS

04/07/24

Outputs

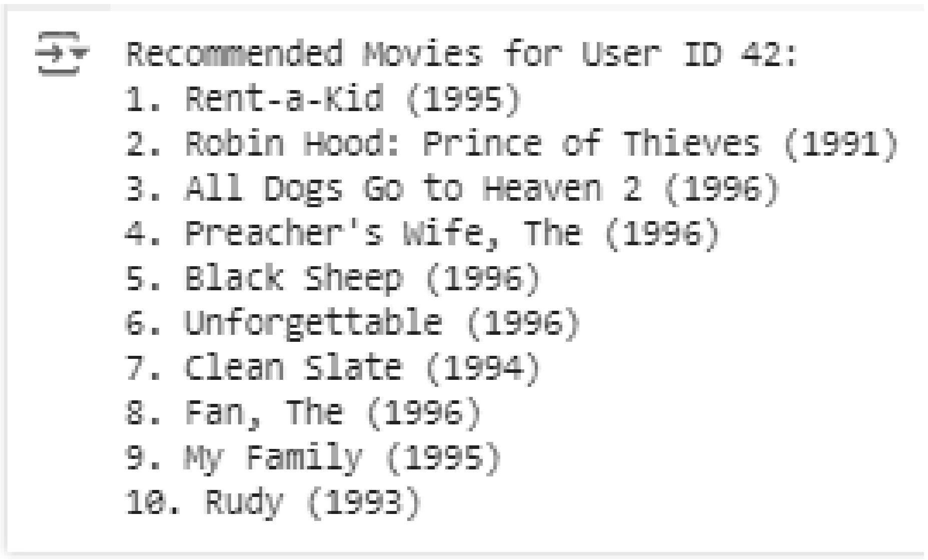
Recommendation System 1

```
def get_movie_recommendations(user_id, model, movie_titles, top_k=10):
   user_embedding = model.user_model(tf.constant([user_id]))
   movie_embeddings = model.movie_model(tf.constant(movie_titles))

   scores = tf.matmul(user_embedding, movie_embeddings,
transpose_b=True).numpy().flatten()
   top_indices = np.argsort(scores)[-top_k:][::-1]
   return [movie_titles[i] for i in top_indices]

# Example: Get recommendations for a specific user ID
user_id_example = "42"  # Replace with a valid user ID from your data
recommended_movies = get_movie_recommendations(user_id_example, model,
unique_movie_titles)

# Print the recommendations
print(f"Recommended Movies for User ID {user_id_example}:")
for idx, movie in enumerate(recommended_movies, 1):
   print(f"{idx}. {movie}")
```



```
Recommended Movies for User ID 42:
1. Rent-a-Kid (1995)
2. Robin Hood: Prince of Thieves (1991)
3. All Dogs Go to Heaven 2 (1996)
4. Preacher's Wife, The (1996)
5. Black Sheep (1996)
6. Unforgettable (1996)
7. Clean Slate (1994)
8. Fan, The (1996)
9. My Family (1995)
10. Rudy (1993)
```

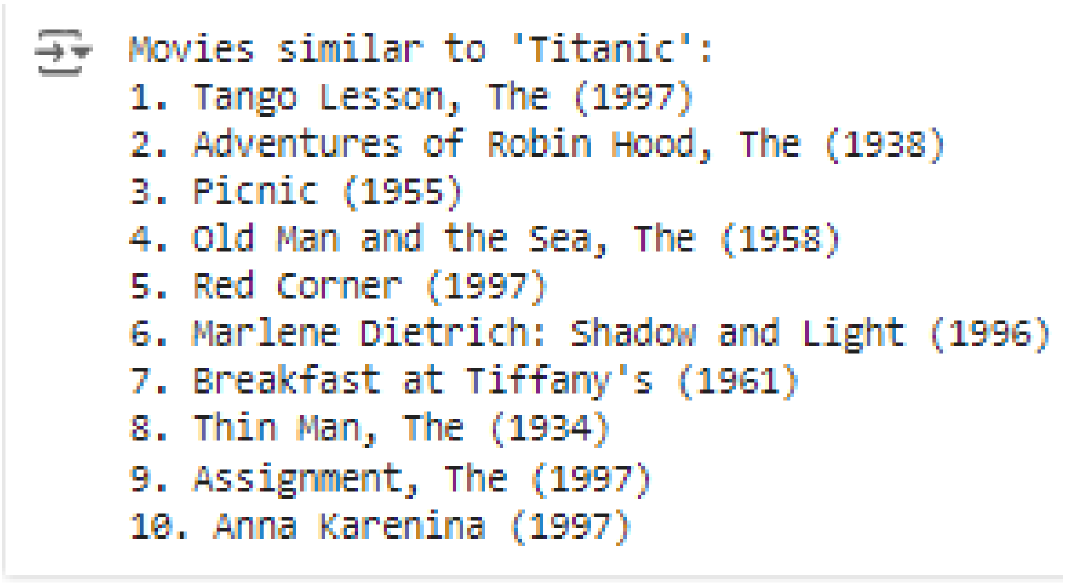
Defines a function to recommend movies for a given user ID, based on the trained model. It computes similarity scores between the user and movies, returning the top-k recommendations.Provides movie recommendations for a specific user ID, printing the top recommendations based on the trained model.

```
def recommend_similar_movies(movie_title, model, movie_titles, top_k=10):
  movie_embedding = model.movie_model(tf.constant([movie_title]))
  all_movie_embeddings = model.movie_model(tf.constant(movie_titles))

  scores = tf.matmul(movie_embedding, all_movie_embeddings,
transpose_b=True).numpy().flatten()
  top_indices = np.argsort(scores)[-top_k:][::-1]
  return [movie_titles[i] for i in top_indices]

# Example: Recommend movies similar to "Toy Story (1995)"
movie_example = "Titanic" # Replace with a valid movie title from your data
similar_movies = recommend_similar_movies(movie_example, model,
unique_movie_titles)

# Print similar movies
print(f"Movies similar to '{movie_example}':")
for idx, movie in enumerate(similar_movies, 1):
  print(f"{idx}. {movie}")
```



```
Movies similar to 'Titanic':
1. Tango Lesson, The (1997)
2. Adventures of Robin Hood, The (1938)
3. Picnic (1955)
4. Old Man and the Sea, The (1958)
5. Red Corner (1997)
6. Marlene Dietrich: Shadow and Light (1996)
7. Breakfast at Tiffany's (1961)
8. Thin Man, The (1934)
9. Assignment, The (1997)
10. Anna Karenina (1997)
```

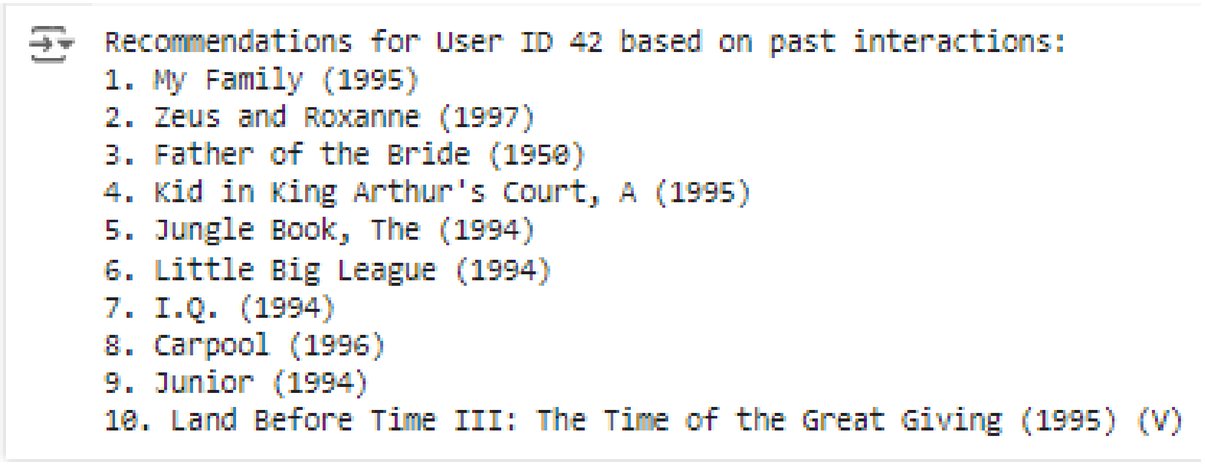
Defines a function to recommend movies similar to a given movie title, using the model to compute similarity scores and returning the top-k similar movies.Provides movie recommendations similar to a specified movie title, printing the top similar movies based on the model.

```
def recommend_based_on_preferences(user_id, ratings_data, model, movie_titles, top_k=10):
  rated_movies = [rating['movie_title'].numpy().decode('utf-8') for rating in ratings_data if rating['user_id'].numpy().decode('utf-8') == user_id]
  movie_candidates = [movie for movie in movie_titles if movie not in rated_movies]

  recommended_movies = get_movie_recommendations(user_id, model, movie_candidates, top_k)
  return recommended_movies

# Example: Recommend movies for a user based on past interactions
user_id_preferences_example = "42" # Replace with a valid user ID
preferences_based_recommendations = recommend_based_on_preferences(user_id_preferences_example, ratings, model, unique_movie_titles)

# Print the recommendations based on user preferences
print(f"Recommendations for User ID {user_id_preferences_example} based on past interactions:")
for idx, movie in enumerate(preferences_based_recommendations, 1):
  print(f"{idx}. {movie}")
```



```
Recommendations for User ID 42 based on past interactions:
1. My Family (1995)
2. Zeus and Roxanne (1997)
3. Father of the Bride (1950)
4. Kid in King Arthur's Court, A (1995)
5. Jungle Book, The (1994)
6. Little Big League (1994)
7. I.Q. (1994)
8. Carpool (1996)
9. Junior (1994)
10. Land Before Time III: The Time of the Great Giving (1995) (V)
```

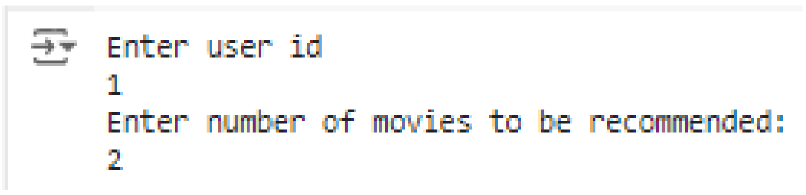
Defines a function to recommend movies for a user based on their past interactions and preferences, excluding movies they have already rated.Provides movie recommendations for a user based on their past interactions, listing movies that the user is likely to enjoy based on their previous ratings.

Outputs

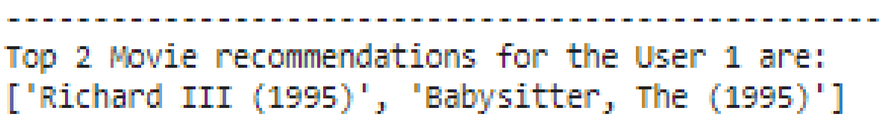
Recommendation System 2

def recommender_system(user_id, model, n_movies):

```
 print("")
 print("Movie seen by the User:")
 pprint(list(refined_dataset[refined_dataset['user id'] == user_id]['movie title']))
 print("")
 encoded_user_id = user_enc.transform([user_id])
 seen_movies = list(refined_dataset[refined_dataset['user id'] == user_id]['movie'])
 unseen_movies = [i for i in range(min(refined_dataset['movie']),
max(refined_dataset['movie'])+1) if i not in seen_movies]
 model_input = [np.asarray(list(encoded_user_id)*len(unseen_movies)),
np.asarray(unseen_movies)]
 predicted_ratings = model.predict(model_input)
 predicted_ratings = np.max(predicted_ratings, axis=1)
 sorted_index = np.argsort(predicted_ratings)[::-1]
 recommended_movies = item_enc.inverse_transform(sorted_index)
 print("-------------------------------------------------------------------------------")
 print("Top "+str(n_movies)+" Movie recommendations for the User "+str(user_id)+
" are:")
 pprint(list(recommended_movies[:n_movies]))
print("Enter user id")
user_id= int(input())
print("Enter number of movies to be recommended:")
n_movies = int(input())
recommender_system(user_id,model,n_movies)
```



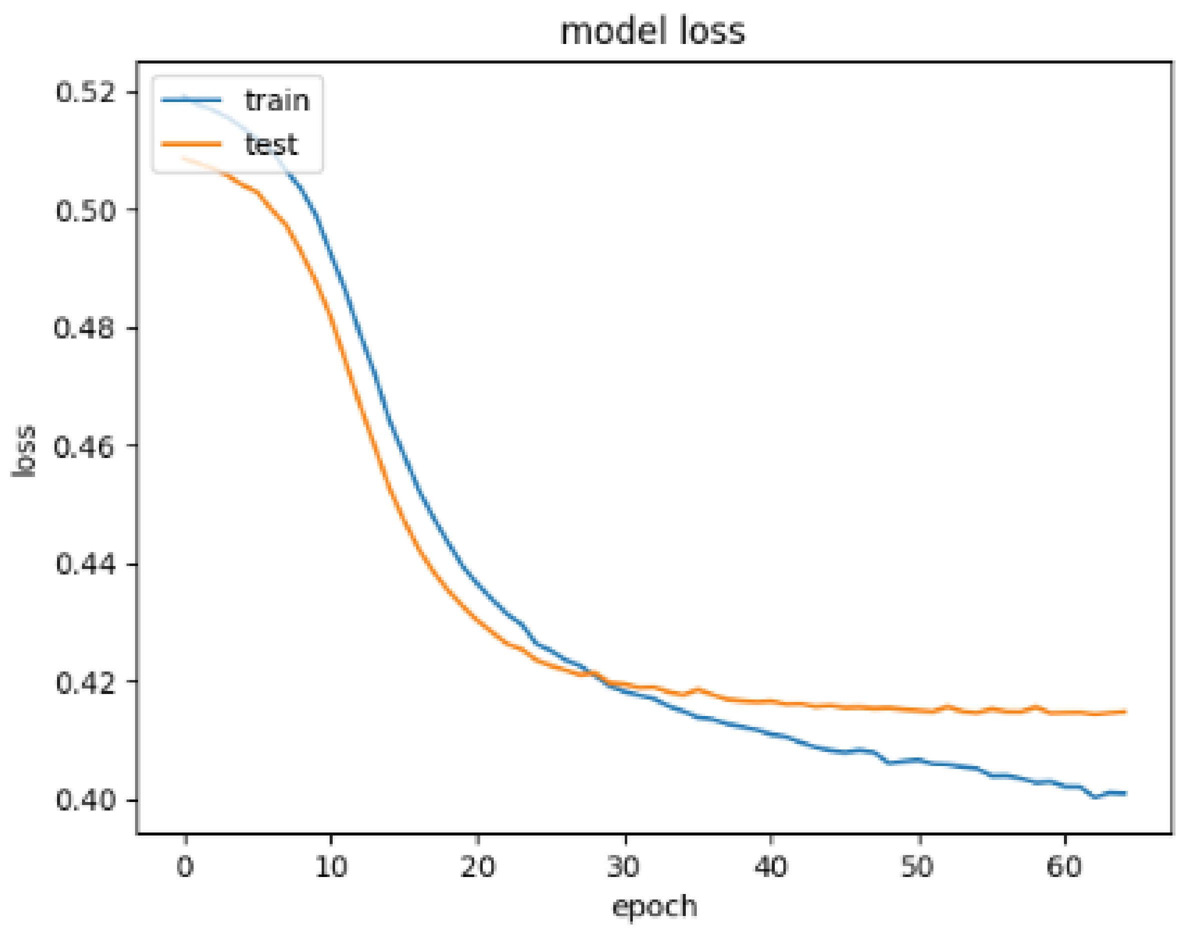
```
Enter user id
1
Enter number of movies to be recommended:
2
```



```
--------------------------------------------------------------
Top 2 Movie recommendations for the User 1 are:
['Richard III (1995)', 'Babysitter, The (1995)']
```

Defines a function to recommend movies to a specific user based on their viewing history and predicted ratings.

Model Loss



Trained the model on X_train_array and y_train data, with specified batch size, epochs, validation data, and callbacks for learning rate reduction.

Outputs

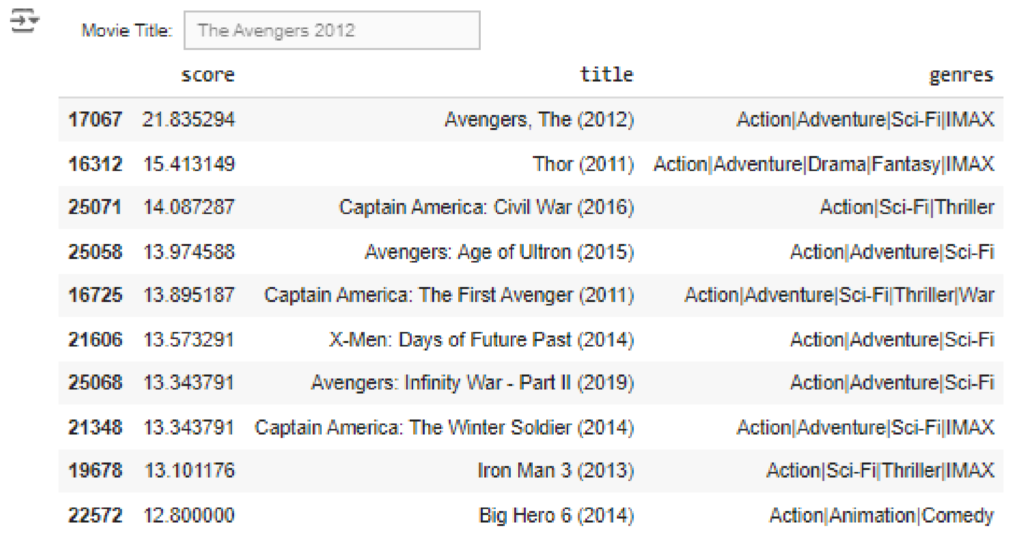
Recommendation System 3

```
def find_similar_movies(movie_id):

  similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] > 4)]
["userId"].unique()
  similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) &
(ratings["rating"] > 4)]["movieId"]
  similar_user_recs = similar_user_recs.value_counts() / len(similar_users)

  similar_user_recs = similar_user_recs[similar_user_recs > .10]
  all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) &
(ratings["rating"] > 4)]
  all_user_recs = all_users["movieId"].value_counts() /
len(all_users["userId"].unique())
  rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
  rec_percentages.columns = ["similar", "all"]

  rec_percentages["score"] = rec_percentages["similar"] / rec_percentages["all"]
  rec_percentages = rec_percentages.sort_values("score", ascending=False)
  return rec_percentages.head(10).merge(movies, left_index=True,
right_on="movieId")[["score", "title", "genres"]]
```

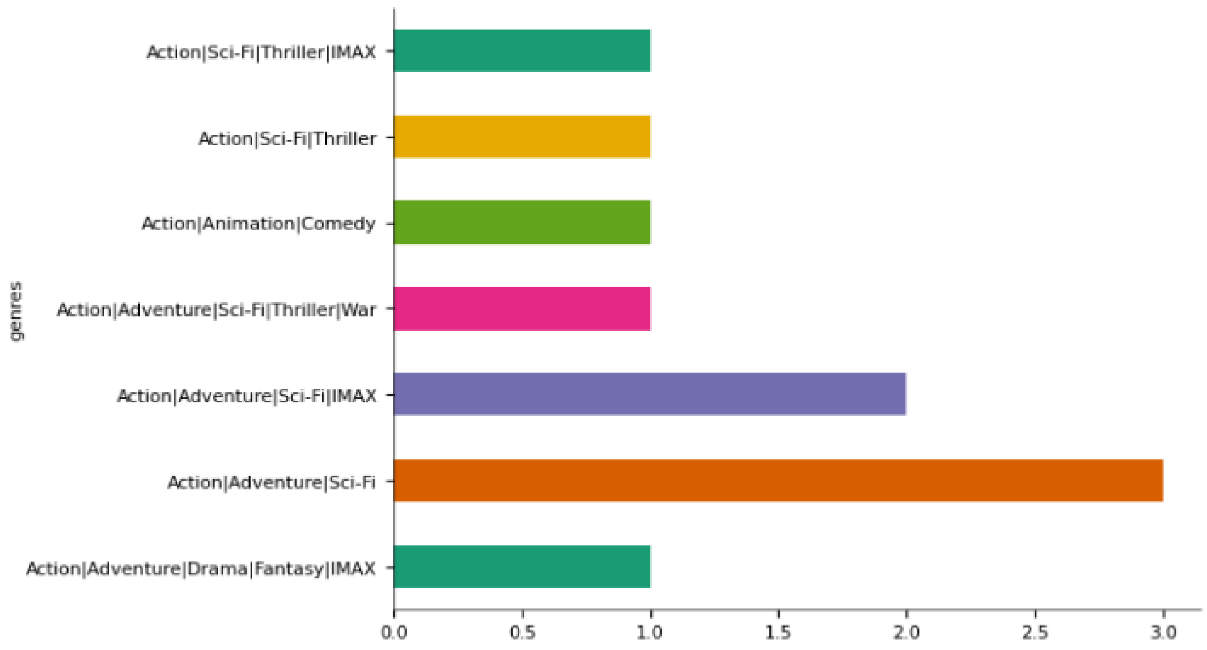

Movie Title: The Avengers 2012

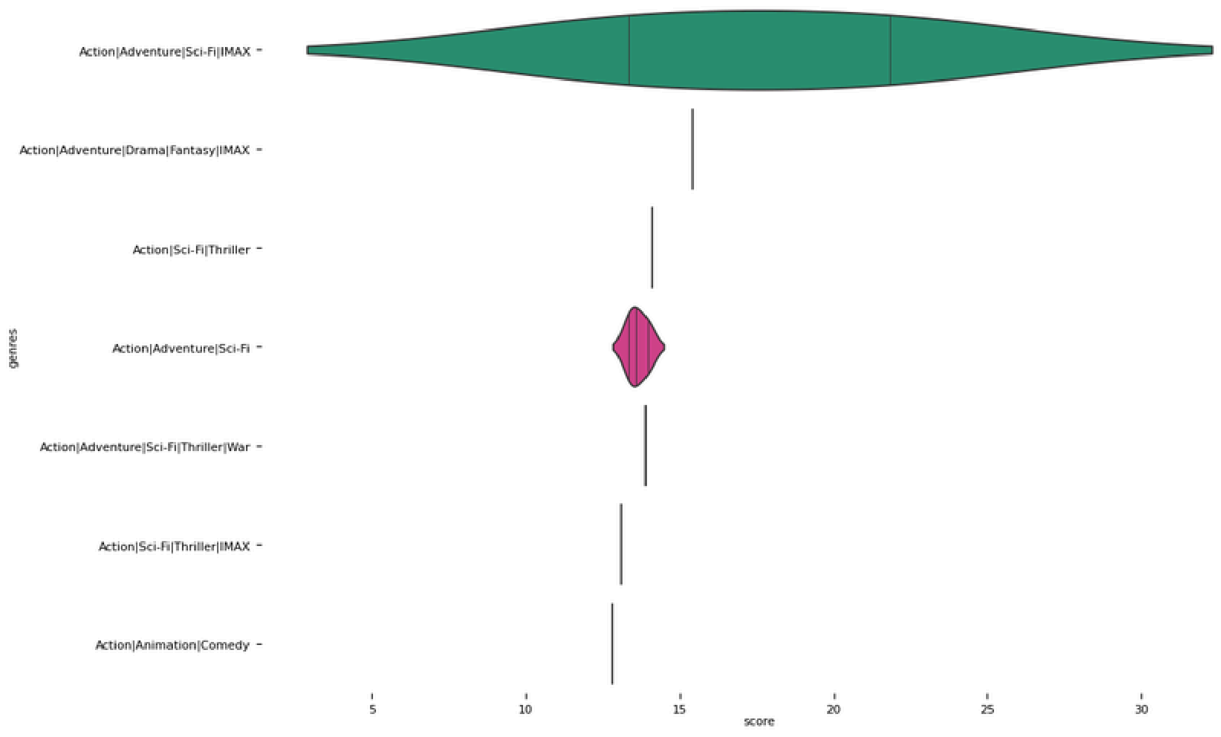
	score	title	genres
17067	21.835294	Avengers, The (2012)	Action|Adventure|Sci-Fi|IMAX
16312	15.413149	Thor (2011)	Action|Adventure|Drama|Fantasy|IMAX
25071	14.087287	Captain America: Civil War (2016)	Action|Sci-Fi|Thriller
25058	13.974588	Avengers: Age of Ultron (2015)	Action|Adventure|Sci-Fi
16725	13.895187	Captain America: The First Avenger (2011)	Action|Adventure|Sci-Fi|Thriller|War
21606	13.573291	X-Men: Days of Future Past (2014)	Action|Adventure|Sci-Fi
25068	13.343791	Avengers: Infinity War - Part II (2019)	Action|Adventure|Sci-Fi
21348	13.343791	Captain America: The Winter Soldier (2014)	Action|Adventure|Sci-Fi|IMAX
19678	13.101176	Iron Man 3 (2013)	Action|Sci-Fi|Thriller|IMAX
22572	12.800000	Big Hero 6 (2014)	Action|Animation|Comedy

Finds movies similar to a specified movie (movie_id) based on user ratings. Filters users who rated the specified movie highly (rating > 4) and identifies other movies highly rated by these users. Calculates a score based on the proportion of similar ratings to all ratings for each recommended movie.

Visualizations

Categorical distributions





Thank You

Github Link