Import the required libaries.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


from scipy.sparse import csr_matrix
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import warnings
from sklearn.neighbors import NearestNeighbors
from tabulate import tabulate
```

Load the datasets into a dataframe

In [2]:
```
movies_df = pd.read_csv("movies.csv")
ratings_df = pd.read_csv("ratings.csv")
tags_df = pd.read_csv("tags.csv")
links_df = pd.read_csv("links.csv")
```

In [3]:
```
movies_df.shape, ratings_df.shape, tags_df.shape,links_df.shape
```

Out[3]:
```
((9742, 3), (100836, 4), (3683, 4), (9742, 3))
```

In [4]:
```
movies_df.head(5)
```

Out[4]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [5]:
```
ratings_df.head(5)
```

Out[5]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [6]:
```
tags_df.head(5)
```

Out[6]:

	userId	movieId	tag	timestamp
0	2	60756	funny	1445714994
1	2	60756	Highly quotable	1445714996

2	2	60756	will ferrell	1445714992
3	2	89774	Boxing story	1445715207
4	2	89774	MMA	1445715200

In [7]:
```
links_df.head(5)
```

Out[7]:

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

Merge the datasets

In [8]:
```
movies_ratings_df = ratings_df.merge(movies_df,on='movieId', how='left')
print(movies_ratings_df.shape)
movies_ratings_df.head(5)
```

```
(100836, 6)
```

Out[8]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	3	4.0	964981247	Grumpier Old Men (1995)	Comedy|Romance
2	1	6	4.0	964982224	Heat (1995)	Action|Crime|Thriller
3	1	47	5.0	964983815	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
4	1	50	5.0	964982931	Usual Suspects, The (1995)	Crime|Mystery|Thriller

In [10]:
```
# Calculate the total number of possible ratings
num_users = movies_ratings_df['userId'].nunique()
num_movies = movies_ratings_df['movieId'].nunique()
total_possible_ratings = num_users*num_movies
print('num_users : ',num_users, ' num_movies : ',num_movies,' total_possible_ratings : '

# Calculate the number of missing ratings
num_actual_ratings = movies_ratings_df.shape[0]
num_missing_ratings = total_possible_ratings - num_actual_ratings
print('num_actual_ratings : ',num_actual_ratings, ' num_missing_ratings : ',num_missing_

# Calculate the sparsity
sparsity = (num_missing_ratings / total_possible_ratings) * 100

print(f"Sparsity of the MovieLens dataset: {sparsity:.2f}%")
```

```
num_users :  610  num_movies :  9724  total_possible_ratings :  5931640
num_actual_ratings :  100836  num_missing_ratings :  5830804
Sparsity of the MovieLens dataset: 98.30%
```

Given the high sparsity, it's important to use a sparse matrix representation (e.g., csr_matrix) to save memory. This will ensure we're not storing unnecessary zero values and can perform operations efficiently.

In [11]:
```
# Map the indices to users and movie ids.
```

```
user_map = dict(zip(np.unique(movies_ratings_df['userId']), list(range(len(movies_rating
movie_map = dict(zip(np.unique(movies_ratings_df['movieId']), list(range(len(movies_rati
```

In [12]:
```
# Create indices for the csr matrix
user_index = [user_map[i] for i in movies_ratings_df['userId']]
movie_index = [movie_map[i] for i in movies_ratings_df['movieId']]
```

In [13]:
```
# Create the csr_matrix
matrix = csr_matrix((movies_ratings_df["rating"], (movie_index, user_index)), shape=(len
```

In [14]:
```
# Map the movies to the movie ids.
movie_titles_mapped = dict(zip(movies_ratings_df['movieId'], movies_ratings_df['title']))
```

Function to identify the best metric for the model

In [15]:
```
def choosing_best_metric(movie_name, total_matches):
    warnings.filterwarnings("ignore")
    metrics_to_try = ['cosine', 'euclidean', 'manhattan', 'correlation']

    # Split the data into training and test sets
    train_data, test_data = train_test_split(movies_ratings_df[['userId','movieId','rati

    # List of metrics to try
    metrics_to_try = ['cosine', 'euclidean', 'manhattan', 'correlation']

    # Iterate over each metric and evaluate the model
    for metric in metrics_to_try:
        # Create Nearest Neighbors model
        model = NearestNeighbors(algorithm='auto', metric=metric)

        # Fit the model on the training data
        model.fit(train_data[['userId', 'movieId']])

        # For each user in the test set, find nearest neighbors and make predictions
        predicted_ratings = []
        for user_id, movie_id, _ in test_data.itertuples(index=False):
            distances, indices = model.kneighbors([[user_id, movie_id]], n_neighbors=5)
            neighbor_ratings = train_data.iloc[indices[0]]['rating']
            predicted_rating = neighbor_ratings.mean() if len(neighbor_ratings) > 0 else
            predicted_ratings.append(predicted_rating)

        # Calculate and print RMSE for the current metric
        rmse = mean_squared_error(test_data['rating'], predicted_ratings, squared=False)
        print(f"Metric: {metric}, RMSE: {rmse:.4f}")
```

In [21]:
```
name = "Underground"
totalMatches = 6
choosing_best_metric(name, totalMatches)
```

```
Metric: cosine, RMSE: 1.1374
Metric: euclidean, RMSE: 1.0701
Metric: manhattan, RMSE: 1.0592
Metric: correlation, RMSE: 1.1292
```

Based on the RMSE values, manhattan has the lowest RMSE of 1.0592. Therefore, manhattan appears to be the best metric to use for your Nearest Neighbors model on the given dataset.

Function to print the recommendations in a user friendly format

In [16]:
```
def print_recommendations(movie_name, sorted_neighbours,total_matches):
    # Print the movie titles and their related accuracy.
```

```
    count = 1
    movie_link = "https://www.themoviedb.org/movie/"
    table_df = pd.DataFrame(sorted_neighbours, columns=["Name","Genre"])
    table_df = table_df.drop("Genre", axis=1)
    table_df["genre"]=""
    table_df["url"]=""

    for index, row in table_df.iterrows():
        if len(table_df) == total_matches:
            break
        else:
            next_movie_id = next((k for k, v in movie_titles_mapped.items() if row.Name.
            genre = movies_df.loc[movies_df['movieId'] == next_movie_id, 'genres'].item(
            table_df["genre"][index] = genre
            tmdb_id = links_df.loc[links_df['movieId'] == next_movie_id, 'tmdbId'].item(
            neighbour_movie_link = movie_link + str(int(tmdb_id))
            table_df["url"][index] = neighbour_movie_link
            count += 1
    #Wrap text that's breaking the table before printing
    table_df['Name']  = table_df['Name'].str.wrap(50)
    table_df['genre']  = table_df['genre'].str.wrap(30)
    print (tabulate(table_df, headers=["Movie Titles","Genre", "URL"],tablefmt='fancy_gr
```

Function to Recommend Movies using the best metric

In [17]:
```
def recommend_movies(movie_name, total_matches):

    # Increment total matches since we'll be removing the same movie
    total_matches += 1

    # Create a variable to hold our neighbors.
    neighbour_ids_with_distance = {}

    # Look up the movie the user entered using "contains" and get the matching movieIds'
    user_movie_id = next((k for k, v in movie_titles_mapped.items() if movie_name.lower(
    # Prepare a vector for the KNN model.
    movie_index_mapped = movie_map[user_movie_id]
    movie_vector = matrix[movie_index_mapped]

    # Set the KNN model and fit it.
    knn = NearestNeighbors(algorithm = 'auto', metric='manhattan')
    #Using "auto" so the algorithm will automatically choose the most appropriate algori
    #based on the input data and other parameters
    knn.fit(matrix)

    # Determine distances for KNN values.
    distances, indices = knn.kneighbors(movie_vector, n_neighbors=total_matches)

    # Loop through the data and flatten the distances.
    for i in range(0,len(distances.flatten())):
        n = indices.flatten()[i]
        neighbour_id = list(filter(lambda x: movie_map[x] == n, movie_map))[0]
        neighbour_ids_with_distance[movie_titles_mapped[neighbour_id]] = distances.flatt

    # Remove the user entered movie title from the list.
    neighbour_ids_with_distance.pop(movie_titles_mapped[user_movie_id], None)

    # Sort the data by accuracy
    sorted_neighbours = sorted(neighbour_ids_with_distance.items(), key=lambda x: x[1],

    actual_matches = total_matches - 1
    print(f"Found {actual_matches} movies related to : {movie_titles_mapped[user_movie_i
    print_recommendations(movie_name, sorted_neighbours, total_matches) #Call function t
```

Testing the functions

In [18]:

```python
name = "Underground"
totalMatches = 10
recommend_movies(name, totalMatches)
```

Found 10 movies related to : Underground (1995)

Movie Titles	Genre	URL
Time of the Gypsies (Dom za vesanje) (1989)	Comedy|Crime|Drama|Fantasy	https://www.themoviedb.org/movie/20123
How to Steal a Million (1966)	Comedy|Crime|Romance	https://www.themoviedb.org/movie/3001
His Secret Life (a.k.a. Ignorant Fairies, The) (Fate ignoranti, Le) (2001)	Drama|Romance	https://www.themoviedb.org/movie/23550
Not One Less (Yi ge dou bu neng shao) (1999)	Drama	https://www.themoviedb.org/movie/36210
I'm Starting From Three (Ricomincio da Tre) (1981)	Comedy	https://www.themoviedb.org/movie/13386
Atalante, L' (1934)	Comedy|Drama|Romance	https://www.themoviedb.org/movie/43904
Hit the Bank (Vabank) (1981)	Comedy|Crime	https://www.themoviedb.org/movie/22257
Nirvana (1997)	Action|Sci-Fi	https://www.themoviedb.org/movie/8765
How to Marry a Millionaire (1953)	Comedy|Drama|Romance	https://www.themoviedb.org/movie/10297
Time Masters (Maîtres du temps, Les) (1982)	Animation|Sci-Fi	https://www.themoviedb.org/movie/22501

In [19]:

```python
name = "Iron Man"
totalMatches = 6

recommend_movies(name, totalMatches)
```

Found 6 movies related to : Iron Man (2008)

Movie Titles	Genre	URL
Iron Man 2 (2010)	Action|Adventure|Sci-Fi|Thriller|IMAX	https://www.themoviedb.org/movie/10138
Avengers, The (2012)	Action|Adventure|Sci-Fi|IMAX	https://www.themoviedb.org/movie/24428
Thor (2011)	Action|Adventure|Drama|Fantasy|IMAX	https://www.themoviedb.org/movie/10195
X-Men: First Class (2011)	Action|Adventure|Sci-Fi|Thriller|War	https://www.themoviedb.org/movie/49538
Iron Man 3 (2013)	Action|Sci-Fi|Thriller|IMAX	https://www.themoviedb.org/movie/68721
Star Trek (2009)	Action|Adventure|Sci-Fi|IMAX	https://www.themoviedb.org/movie/13475

Program to accept user input for movie title and number of recommendations.

In [22]:
```
def main():
    counter = 0
    try:
        print('\033[1m' + '         WELCOME TO THE MOVIE RECOMMENDER APP      ' + '\033[
        # Accept user input for zip code or city
        user_input = input("Enter the movie title you wish to see OR Enter '!' to stop :
        while user_input != '!':
            if user_input != '!':
                if counter > 0:
                    user_input = input("\nEnter a movie title or '!' to stop : ")
                if user_input!="" and user_input!= "!" :
                    number_input = input("\nEnter number of recommendations needed : ")
                    try:
                        number_input = int(number_input)
                        if number_input > 0:
                            recommend_movies(user_input, number_input)
                    except ValueError as val:
                        number_input = 0
                    except RuntimeError as err:
                        print('There was an error processing user input. Please retry.')
            elif user_input == '!':
                print("Hope you enjoy the movies!")
                break
            counter += 1
        print("Bye....Hope you enjoy the movies!")
```

```python
    except RuntimeError as err:
        print('There was an error: ', err, '\nPlease start over.')


if __name__ == '__main__':
    main()
```

```
          WELCOME TO THE MOVIE RECOMMENDER APP
Enter the movie title you wish to see OR Enter '!' to stop : speed

Enter number of recommendations needed : 5
Found 5 movies related to : Speed (1994)
```

Movie Titles	Genre	URL
Die Hard: With a Vengeance (1995)	Action|Crime|Thriller	https://www.themoviedb.org/movie/1572
Mrs. Doubtfire (1993)	Comedy|Drama	https://www.themoviedb.org/movie/788
True Lies (1994)	Action|Adventure|Comedy|Romance|Thriller	https://www.themoviedb.org/movie/36955
Sleepless in Seattle (1993)	Comedy|Drama|Romance	https://www.themoviedb.org/movie/858
Pretty Woman (1990)	Comedy|Romance	https://www.themoviedb.org/movie/114

```
Enter a movie title or '!' to stop : vertical

Enter number of recommendations needed : 6
Found 6 movies related to : Vertical Limit (2000)
```

Movie Titles	Genre	URL
Jimmy Neutron: Boy Genius (2001)	Adventure|Animation|Children|Comedy	https://www.themoviedb.org/movie/12589
Final Destination, The (Final Destination 4) (Final Destination in 3-D, The) (2009)	Horror|Thriller	https://www.themoviedb.org/movie/19912
Return of Jafar, The (1994)	Adventure|Animation|Children|F	http

```
s://www.themoviedb.org/movie/15969 │
│                                         │ antasy|Musical|Romance            │
                                 │
├─────────────────────────────────────────┼───────────────────────────────────┼──────
──────────────────────────────────┤
│ [REC]³ 3 Génesis (2012)                 │ Horror|Thriller                   │ http
s://www.themoviedb.org/movie/80280 │
├─────────────────────────────────────────┼───────────────────────────────────┼──────
──────────────────────────────────┤
│ Skeleton Key, The (2005)                │ Drama|Horror|Mystery|Thriller     │ http
s://www.themoviedb.org/movie/9913  │
├─────────────────────────────────────────┼───────────────────────────────────┼──────
──────────────────────────────────┤
│ [REC]² (2009)                           │ Horror|Thriller                   │ http
s://www.themoviedb.org/movie/10664 │
╘═════════════════════════════════════════╧═══════════════════════════════════╧══════
══════════════════════════════════╛

Enter a movie title or '!' to stop : !
Bye....Hope you enjoy the movies!
```

Reference:

Das, S. (2023b, July 24). Building a movie recommendation system with Machine Learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/