Database Details

The movie database for this project was taken from **here**

The movie data is split into two separate files: **movies.csv** and **ratings.csv**. The former contains details about the movie under the headings:

- **movieId** : Numeric ID of the selected movie in the database
- **title** : The name of the movie along with the year it was created in
- **genre** : One or multiple genres the selected movie belongs to

The latter, **ratings.csv** contains details about multiple users and how they have rated the movies under the following headings:

- **userId** : The numeric ID of the users who have rated the movies present in the database
- **movieId** : The name of the movie along with the year it was created in
- **rating** : Ratings assigned to the movies by the users ranging from **0.0** to **5.0**
- **timestamp** : Showcases the length (or timestamp) of the given movie

The initial structure of **movies.csv** and **ratings.csv** is shown in the images below:

```
In [2]: moviedatabase = pd.read_csv('movies.csv')
        moviedatabase.head()
```

Out[2]:

	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 [3]: ratingdatabase = pd.read_csv('ratings.csv')
        ratingdatabase.head()
```

Out[3]:

	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 total with both databases included the net data includes **9,472 movies** and **100,836 user ratings** which are all used to compute the recommendations for a selected movie.

Help Document

The following code is written completely in native **python** with the help of a few additional python libraries. Here are all the dependencies and libraries used in the program file:

- **pandas** : Used for database computation and cleaning up given databases. The documentation for the library can be found **here**.
- **scipy** : Used to convert the sparse matric to a non-sparse matrix. The documentation for the library can be found **here**.
- **fuzzywuzzy** : AI python library used to aid search processes in the main recommender functions. The documentation for the library can be found **here**.
- **sklearn** : python library used to implement the KNN algorithm. The documentation for the library can be found **here**.

The entire code is written in just one executable python file, namely **recommendersystem.py**. The file takes no inputs, but outputs a list of 20 recommended movies based on the movie initially provided to the algorithm. The execution steps are as follows:

```
python recommendersystem.py
```

(make sure that all files are in the same directory)

Methods and Techniques

The program **recommendersystem.py** uses the KNN (K Nearest Neighbors) algorithm to find out the similarity between two users and accordingly recommends a list of 20 movies based on the initial movie provided. The code uses the following methods and techniques to achieve the goal:

```
moviedatabase = pd.read_csv('movies.csv', usecols = ['movieId', 'title']) ratingdatabase = pd.read_csv('ratings.csv', usecols = ['userId', 'movieId', 'rating'])
```

This section of code uses **pandas** to read through the .csv files and extract the important information from **movies.csv** and **ratings.csv** to clean up the data and reduce it to a usable form, in our case, **moviedatabase** and **ratingdatabase**.

```
featurematrix = ratingdatabase.pivot(index = 'movieId', columns = 'userId').fillna(0)
sparsefeaturematrix = csr_matrix(featurematrix.values)
```

The following section of the code makes use of **sklearn** to generate a **feature matrix** from the cleaned up databases with **userID** and **movieID** as rows and columns. the generated matrix is stored in the variable **featurematrix**. Aforementioned matrix is a sparse matrix, hence after cleaning it up and getting rid of all null values it is stored in the variable **sparsefeaturematrix**.

```
modelcosine = NearestNeighbors(metric = 'cosine', algorithm = 'brute', n_neighbors = 25)
modelcosine.fit(sparsefeaturematrix)
```

```
def cosinemovierecommender(moviename, data, number):
index = process.extractOne(moviename, moviedatabase[‘title’])[2] print(“Preferred movie:”, moviedatabase[‘title’][index], “Index:” , index)
print(“Generating recommendation list…”) distance, indices = modelcosine.kneighbors(data[index], n_neighbors = number)
for i in indices:
        print(moviedatabase[‘title’][i].where(i != index)) #print(distance)


cosinemovierecommender(‘shawshank redemption’, sparsefeaturematrix, 20)
```

The following section of code makes use of **scipy** and **fuzzywuzzy** to generate a model for the core computation. **modelcosine** generates a data model which when plugged in the function **cosinemovierecommender** outputs a list of movies derived from the algorithm which are the closest to the provided movie (in this case, **Shawshank Redemption**).

For computational purposes, the algorithm uses three different metrics to compute similarity, namely **cosine similarity**, **manhattan distance** and **euclidean distance**. The functions **manhattanmovierecommender** and **euclideanmovierecommender** use the other two metrics to calculate similarity using said metrics.

Results

The even though the program only outputs a list of desired movies, it does a few things to aid the final outcome.

```
moviedatabase = pd.read_csv(‘movies.csv’, usecols = [‘movieId’, ‘title’])
moviedatabase.head()

ratingdatabase = pd.read_csv(‘ratings.csv’, usecols = [‘userId’, ‘movieId’, ‘rating’])
ratingdatabase.head()
```

These two command statements clean and reduce the initial datasets into a database finally used in the calculations.

```
In [4]: moviedatabase = pd.read_csv('movies.csv', usecols = ['movieId', 'title'])
        moviedatabase.head()
```

Out[4]:

	movieId	title
0	1	Toy Story (1995)
1	2	Jumanji (1995)
2	3	Grumpier Old Men (1995)
3	4	Waiting to Exhale (1995)
4	5	Father of the Bride Part II (1995)

```
In [5]: ingdatabase = pd.read_csv('ratings.csv', usecols = ['userId', 'movieId', 'rati
        ingdatabase.head()
```

Out[5]:

	userId	movieId	rating
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0

featurematrix = ratingdatabase.pivot(index = 'movieId', columns = 'userId').fillna(0)
featurematrix.head()

This command makes a matrix using the data in **moviedatabase** and **ratingdatabase**.

```
In [6]: featurematrix = ratingdatabase.pivot(index = 'movieId', columns = 'userId').fillna(0)
        featurematrix.head()
```

Out[6]:

	rating																			
userId	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	...	**601**	**602**	**603**	**604**	**605**	**606**	**607**	**608**	**609**	**610**
movieId																				
1	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	0.0	0.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	0.0	...	0.0	4.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0	0.0
3	4.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 610 columns

The final execution of the command results in a list of recommended movies based on the input movie:

```
total movies in database: 9742
total user ratings in database: 100836
Preferred movie:  Shawshank Redemption, The (1994) Index:  277
Generating recommendation list...
277                                              NaN
314                              Forrest Gump (1994)
257                              Pulp Fiction (1994)
510                 Silence of the Lambs, The (1991)
46                        Usual Suspects, The (1995)
461                          Schindler's List (1993)
2224           Home Alone 2: Lost in New York (1992)
97                                Braveheart (1995)
1938                       Walk on the Moon, A (1999)
123                                Apollo 13 (1995)
43                       Seven (a.k.a. Se7en) (1995)
4791                               Cooler, The (2003)
659                             Godfather, The (1972)
1283                     For Richer or Poorer (1997)
398                               Fugitive, The (1993)
418                               Jurassic Park (1993)
3633                       White Water Summer (1987)
507                Terminator 2: Judgment Day (1991)
224     Star Wars: Episode IV - A New Hope (1977)
3136                         Longest Yard, The (1974)
```