PRML MAJOR PROJECT

Nakkina Vinay (B21AI023)
Sakam Sai Santhosh (B21AI035)
Geda Durga Vara Praveen (B21CS031)

Using K-Means Algorithm

- Importing the necessary libraries required for our project
- Using the read_csv function getting the dataset values for movies, ratings, tags, links

	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

- Next we will use info to find that there are any null values or anything useless values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34208 entries, 0 to 34207
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  34208 non-null  int64
 1   title    34208 non-null  object
 2   genres   34208 non-null  object
dtypes: int64(1), object(2)
memory usage: 801.9+ KB
```

- We should import the dataset of ratings to the colab file.

	userId	movieId	rating	timestamp
0	1	169	2.5	1204927694
1	1	2471	3.0	1204927438
2	1	48516	5.0	1204927435
3	2	2571	3.5	1436165433
4	2	109487	4.0	1436165496

- Next we will use info to find that there are any null values or anything useless values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22884377 entries, 0 to 22884376
Data columns (total 4 columns):
 #   Column     Dtype  
---  ------     -----  
 0   userId     int64  
 1   movieId    int64  
 2   rating     float64
 3   timestamp  int64  
dtypes: float64(1), int64(3)
memory usage: 698.4 MB
```

- Next we filter the rating files which are greater than 4.2 and less than 4.2 to filter this huge dataset
- Next we select only the rows in a pandas ratings DataFrame, where the value in the "movieId" column appears more than 10000 times in the DataFrame, based on the count of unique values.
- Now we find shapes of movies and ratings

```
(67, 3)    (1233738, 4)
```

- Randomly deleting the rows to reduce the data. Since google collab is crashing for huge data
- After doing all this we finally end with the shape of ratings as (246748,4) and movies shape as (67,3)
- We should import the dataset of tags to the colab file.

	userId	movieId	tag	timestamp
0	19	2324	bittersweet	1428651158
1	19	2324	holocaust	1428651112
2	19	2324	World War II	1428651118
3	23	7075	hilarious	1378675786
4	23	7075	Underrated	1378675786

- Next we will use info to find that there are any null values or anything useless values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 586994 entries, 0 to 586993
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   userId     586994 non-null  int64 
 1   movieId    586994 non-null  int64 
 2   tag        586978 non-null  object
 3   timestamp  586994 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 17.9+ MB
```

- We should import the dataset of links to the colab file.

	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

- Next we will use info to find that there are any null values or anything useless values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34208 entries, 0 to 34207
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  34208 non-null  int64
 1   imdbId   34208 non-null  int64
 2   tmdbId   33912 non-null  float64
dtypes: float64(1), int64(2)
memory usage: 801.9 KB
```

- Now we merge movie and ratings on movieID by how='inner'

movieId	title	genres	userId	rating	timestamp
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	23	5.0	1378675311
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	99	4.5	1226089249
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	216	5.0	866570221
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	310	5.0	846940105
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	322	5.0	974730227

- Now the number of ratings and movies in two dataframes, and the shape and number of unique values in another dataframe.

```
The dataset contains:  246748  ratings of  67  movies.
movieId          67
title            67
genres           47
userId       104193
rating            2
timestamp    242801
dtype: int64
```

- Now we find the no of unique users and unique movies

```
number of unique user:
104193
number of unique movies:
67
```

- Now we drop the duplicates from this dataset.

```
        movieId              title  \
0             1   Toy Story (1995)
1             1   Toy Story (1995)
2             1   Toy Story (1995)
3             1   Toy Story (1995)
4             1   Toy Story (1995)
...         ...                ...
246743    79132   Inception (2010)
246744    79132   Inception (2010)
246745    79132   Inception (2010)
246746    79132   Inception (2010)
246747    79132   Inception (2010)

                                             genres  userId  rating  \
0           Adventure|Animation|Children|Comedy|Fantasy      23     5.0
1           Adventure|Animation|Children|Comedy|Fantasy      99     4.5
2           Adventure|Animation|Children|Comedy|Fantasy     216     5.0
3           Adventure|Animation|Children|Comedy|Fantasy     310     5.0
4           Adventure|Animation|Children|Comedy|Fantasy     322     5.0
...                                                 ...     ...     ...
246743  Action|Crime|Drama|Mystery|Sci-Fi|Thriller|IMAX  247203     5.0
246744  Action|Crime|Drama|Mystery|Sci-Fi|Thriller|IMAX  247224     5.0
246745  Action|Crime|Drama|Mystery|Sci-Fi|Thriller|IMAX  247287     5.0
246746  Action|Crime|Drama|Mystery|Sci-Fi|Thriller|IMAX  247353     5.0
246747  Action|Crime|Drama|Mystery|Sci-Fi|Thriller|IMAX  247575     5.0

         timestamp
0       1378675311
1       1226089249
2        866570221
3        846940105
4        974730227
...            ...
246743  1295166717
246744  1301954015
246745  1340319983
246746  1305776937
246747  1438024755

[246748 rows x 6 columns]
```

- Then we use describe to find mean ,count, std ,min and percentile of features.

	movieId	userId	rating	timestamp
count	246748.000000	246748.000000	246748.000000	2.467480e+05
mean	3069.776529	123663.191280	4.857889	1.131183e+09
std	10171.288088	71423.889788	0.225522	1.924316e+08
min	1.000000	4.000000	4.500000	8.232041e+08
25%	356.000000	61431.000000	4.500000	9.655479e+08
50%	1136.000000	123318.000000	5.000000	1.118756e+09
75%	2324.000000	185472.000000	5.000000	1.287338e+09
max	79132.000000	247753.000000	5.000000	1.454051e+09

- Now we extract genres from a column in a Pandas DataFrame, create binary columns for each unique genre, and map each movie to its corresponding genres. The final DataFrame drops unnecessary columns.

	movieId	userId	rating	timestamp	Adventure	Mystery	Crime	Action	Comedy	Thriller	Drama	Horror
0	1	23	5.0	1378675311	1	0	0	0	1	0	0	0
1	1	99	4.5	1226089249	1	0	0	0	1	0	0	0
2	1	216	5.0	866570221	1	0	0	0	1	0	0	0
3	1	310	5.0	846940105	1	0	0	0	1	0	0	0
4	1	322	5.0	974730227	1	0	0	0	1	0	0	0

- Computing the average rating for each movie in the dataset, and storing it in the 'a' variable as a Pandas Series, where the movie IDs are the index and the mean rating values are the values in the Series.

```
movieId
1        4.880529
32       4.845227
47       4.831122
50       4.866745
110      4.902096
           ...
6874     4.747391
7153     4.813108
7361     4.766813
58559    4.782436
79132    4.781811
Name: rating, Length: 67, dtype: float64
```

- Series of movie ratings in descending order and assigns the result to a new variable called `sorted_ratings_wise_movie`.

```
movieId
590      4.919485
457      4.916435
150      4.911475
110      4.902096
260      4.901322
           ...
79132    4.781811
4306     4.772087
2329     4.769423
7361     4.766813
6874     4.747391
Name: rating, Length: 67, dtype: float64
```

- Here defines a function get_genre_ratings that calculates the average rating for each user by genre in a Pandas DataFrame. The function takes as inputs three Pandas DataFrames, a list of genres, and a list of column names. The function returns a new DataFrame called genre_ratings, which contains the average rating for each genre by user. The genre_ratings DataFrame is then displayed using the head() method.

```
        rating  rating  rating
14         5.0     NaN     5.0
15         4.5    4.50     NaN
17         5.0    5.00     5.0
39         4.5    4.83     5.0
43         5.0     NaN     5.0
...        ...     ...     ...
247704     NaN     NaN     5.0
247708     NaN     NaN     5.0
247722     NaN     NaN     5.0
247729     NaN     NaN     5.0
247746     NaN     NaN     5.0

[69323 rows x 3 columns]
```

	avg_romance_rating	avg_scifi_rating	avg_comedy_rating
14	5.0	NaN	5.0
15	4.5	4.50	NaN
17	5.0	5.00	5.0
39	4.5	4.83	5.0
43	5.0	NaN	5.0

- We define a function called bias_genre_rating_dataset that creates a biased dataset based on user-defined score limits for two genres. The function returns a new Pandas DataFrame called biased_dataset, which is displayed using the head() method. The function also outputs the number of records in the biased_dataset.

```
Number of records:  2
```

	index	avg_romance_rating	avg_scifi_rating	avg_comedy_rating
0	14	5.0	NaN	5.0
1	15	4.5	4.5	NaN

- We select three columns from a Pandas DataFrame and assigns them to a NumPy array. The code also generates a list of possible k-values for clustering based on the length of the array. The output of the len(X) function call is also displayed which is 2
- Now we merge ratings and movies on the "movieId" column, create a pivot table user_movie_ratings, and print the dimensions of the resulting DataFrame. The iloc method is then used to display a subset of the first six rows and first ten columns of the user_movie_ratings DataFrame.

```
dataset dimensions:  (104193, 67)

Subset example:
```

title	2001: A Space Odyssey (1968)	Aladdin (1992)	Alien (1979)	Aliens (1986)	Amelie (Fabuleux destin d'Amélie Poulain, Le) (2001)	American Beauty (1999)	American History X (1998)	Apocalypse Now (1979)	Apollo 13 (1995)	Back to the Future (1985)
userId										
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

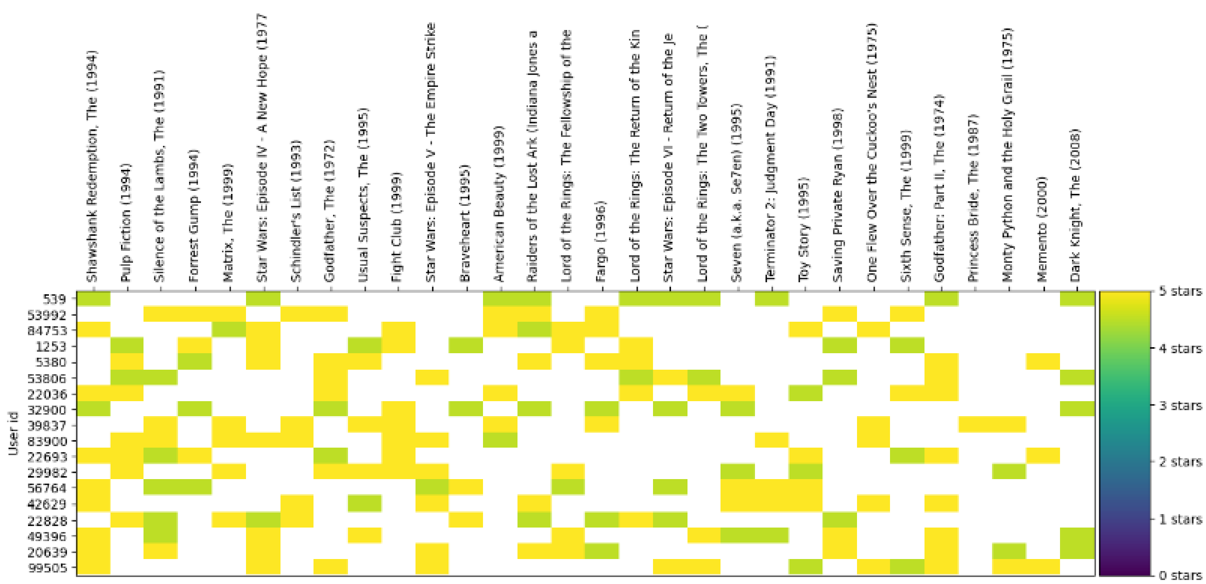
- function sort_by_rating_density that takes in a user-movie ratings matrix, the number of movies to consider, and the number of users to consider. The function returns a new DataFrame most_rated_movies, which is the result of calling two helper functions get_most_rated_movies and get_users_who_rate_the_most on the input DataFrame.
- a function get_most_rated_movies that takes in a user-movie ratings matrix and the maximum number of movies to consider. The function first prints the number of ratings each movie has received, appends a row to the input matrix with the counts of ratings each user has given, and sorts the matrix based on the count of ratings. The function then selects the top max_number_of_movies most rated movies and returns them in a new DataFrame called most_rated_movies.
- Applied function sort_by_rating_density() to select most rated movies and top rating users. Selected 30 movies and 18 users. Output displays dataset dimensions of selected movies and users.

```
title
2001: A Space Odyssey (1968)                              2346
Aladdin (1992)                                            2107
Alien (1979)                                              2729
Aliens (1986)                                             2230
Amelie (Fabuleux destin d'Amélie Poulain, Le) (2001)      3100
                                                          ...
Terminator, The (1984)                                    2186
Toy Story (1995)                                          3821
Trainspotting (1996)                                      2065
Twelve Monkeys (a.k.a. 12 Monkeys) (1995)                 3153
Usual Suspects, The (1995)                                5951
```

title	Shawshank Redemption, The (1994)	Pulp Fiction (1994)	Silence of the Lambs, The (1991)	Forrest Gump (1994)	Matrix, The (1999)	Star Wars: Episode IV - A New Hope (1977)	Schindler's List (1993)	Godfather, The (1972)	Usual Suspects, The (1995)	Fight Club (1999)	...	Terminator 2: Judgment Day (1991)	Toy Story (1995)	Saving Private Ryan (1998)	One Flew Over the Cuckoo's Nest (1975)	Sixth Sense, The (1999)	Godfather: Part II, The (1974)	Princess Bride, The (1987)	Monty Python and the Holy Grail (1975)	Memento (2000)	Dark Knight, The (2008)
99505	5.0	NaN	NaN	NaN	NaN	5.0	NaN	5.0	NaN	NaN	...	NaN	4.5	NaN	5.0	4.5	5.0	NaN	5.0	5.0	NaN
20639	5.0	NaN	5.0	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	5.0	NaN	NaN	5.0	NaN	4.5	NaN	4.5
49396	5.0	NaN	4.5	NaN	NaN	5.0	NaN	NaN	5.0	NaN	...	4.5	NaN	5.0	NaN	NaN	5.0	NaN	NaN	NaN	4.5
22828	NaN	5.0	4.5	NaN	5.0	4.5	5.0	NaN	NaN	NaN	...	NaN	NaN	4.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN
42629	5.0	NaN	NaN	NaN	NaN	NaN	5.0	NaN	4.5	NaN	...	5.0	5.0	NaN	5.0	NaN	5.0	NaN	NaN	NaN	NaN

5 rows × 30 columns

- The draw_movies_heatmap function takes a dataframe of movie ratings by users, most_rated_movies_users_selection, and plots a heatmap visualization of the data. The function allows for axis labels to be displayed or not and includes a color bar legend indicating the rating scale. The output is a visualization of the data.



- get_most_rated_movies() function is defined for selecting a limited number of movies based on the number of ratings received. In this case, it selects the 1000 most rated movies from the user_movie_ratings dataframe. This function is used to create a subset of the original data that focuses on the most rated movies.

```
title
2001: A Space Odyssey (1968)                              2346
Aladdin (1992)                                            2107
Alien (1979)                                              2729
Aliens (1986)                                             2230
Amelie (Fabuleux destin d'Amélie Poulain, Le) (2001)      3100
                                                          ...
Terminator, The (1984)                                    2186
Toy Story (1995)                                          3821
Trainspotting (1996)                                      2065
Twelve Monkeys (a.k.a. 12 Monkeys) (1995)                 3153
Usual Suspects, The (1995)                                5951
Length: 67, dtype: int64
```

- The function sparse_clustering_errors calculates the sum of mean squared errors of a KMeans clustering model for a given number of clusters and data, indicating how well the model fits the data.
- Draw_movie_clusters function loops over all unique cluster IDs and selects at most max_users users and max_movies movies to plot a heatmap.
- Bias_genre_rating_dataset function takes a DataFrame 'genre_ratings', and two score limits 'score_limit_1' and 'score_limit_2' as inputs. It filters out the rows from the 'genre_ratings' DataFrame where the average romance rating is less than 'score_limit_1 - 0.2' AND the average scifi rating is greater than 'score_limit_2', OR where the average scifi rating is less than 'score_limit_1' AND the average romance rating is greater than 'score_limit_2'. The filtered rows are concatenated with the first 300 rows of 'genre_ratings' DataFrame and the first two rows of 'genre_ratings' DataFrame. Finally, the resulting DataFrame is converted to a regular DataFrame with a default index and returned.
- Sort_by_rating_density function returns the top-rated movies that have been rated by the most number of users.
- This clustering the most rated movies using KMeans with 20 clusters, and then plotting the clusters in heatmaps. The heat maps show the ratings of a subset of the users in the cluster (up to `max_users` users) for a subset of the movies (up to `max_movies` movies).
- The heatmaps are plotted and are in more number so it is not possible to paste them here
- And soon.

	Shawshank Redemption, The (1994)	Matrix, The (1999)	Pulp Fiction (1994)	Fight Club (1999)	Godfather, The (1972)	Usual Suspects, The (1995)	Lord of the Rings: The Fellowship of the Ring, The (2001)	Lord of the Rings: The Return of the King, The (2003)	Memento (2000)	Silence of the Lambs, The (1991)	...	Lion King, The (1994)	Schindler's List (1993)	Aliens (1986)	Fugitive, The (1993)	Apollo 13 (1995)	Jurassic Park (1993)	Taxi Driver (1976)	Dances with Wolves (1990)	Terminator, The (1984)	Aladdin (1992)
179	4.5	4.5				4.5					...	4.5						4.5			
223	4.5										...					5.0		5.0			
1766	4.5	5.0	5.0							4.5	...							5.0	4.5		
2027	4.5	4.5	5.0				4.5	5.0			...					5.0				5.0	
733	4.5				5.0					5.0	...						5.0		4.5	4.5	4.5

5 rows × 67 columns

- The cluster's top 20 recommendations based on the mean ratings for the movie `"Usual Suspects, The (1995)"` which we proovided are

```
Shawshank Redemption, The (1994)                                                    4.500000
Matrix, The (1999)                                                                  4.821429
Pulp Fiction (1994)                                                                 4.666667
Fight Club (1999)                                                                   4.909091
Godfather, The (1972)                                                               4.928571
Usual Suspects, The (1995)                                                          4.650000
Lord of the Rings: The Fellowship of the Ring, The (2001)                           4.687500
Lord of the Rings: The Return of the King, The (2003)                               4.916667
Memento (2000)                                                                      4.833333
Silence of the Lambs, The (1991)                                                    4.735294
Forrest Gump (1994)                                                                 4.750000
Lord of the Rings: The Two Towers, The (2002)                                       4.791667
American Beauty (1999)                                                              4.818182
Amelie (Fabuleux destin d'Amélie Poulain, Le) (2001)                                4.714286
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)      4.738095
Star Wars: Episode V - The Empire Strikes Back (1980)                               4.833333
Dark Knight, The (2008)                                                             4.642857
Seven (a.k.a. Se7en) (1995)                                                         4.678571
American History X (1998)                                                           4.805556
One Flew Over the Cuckoo's Nest (1975)                                              4.909091
dtype: float64
```

- The recommendation system also recommends movies to a particular user based on their past ratings and the ratings of other users. Here we specified the user-id as 4 and sorts the average ratings in descending order and selects the top 20 highest-rated movies to recommend to the user.

```
Star Wars: Episode IV - A New Hope (1977)                                         5.000000
Schindler's List (1993)                                                           5.000000
Godfather, The (1972)                                                             4.928571
Casablanca (1942)                                                                 4.928571
Lord of the Rings: The Return of the King, The (2003)                             4.916667
One Flew Over the Cuckoo's Nest (1975)                                            4.909091
Fight Club (1999)                                                                 4.909091
Blade Runner (1982)                                                               4.906250
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)       4.857143
Toy Story (1995)                                                                  4.833333
Memento (2000)                                                                    4.833333
Terminator 2: Judgment Day (1991)                                                 4.833333
Star Wars: Episode V - The Empire Strikes Back (1980)                             4.833333
Matrix, The (1999)                                                                4.821429
American Beauty (1999)                                                            4.818182
Kill Bill: Vol. 1 (2003)                                                          4.818182
American History X (1998)                                                         4.805556
Lord of the Rings: The Two Towers, The (2002)                                     4.791667
Twelve Monkeys (a.k.a. 12 Monkeys) (1995)                                         4.791667
Godfather: Part II, The (1974)                                                    4.791667
Name: 0, dtype: float64
```

Using K-Nearest-Neighbours Algorithm

- First resets the index of the movies dataframe to default sequential integers and updates the dataframe in place. The new index will have values ranging from 0 to the number of rows in the dataframe minus one

	index	movieId	title	genres
0	0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	31	32	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller
2	46	47	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
3	49	50	Usual Suspects, The (1995)	Crime|Mystery|Thriller
4	108	110	Braveheart (1995)	Action|Drama|War
...	...	...	...	...
62	6765	6874	Kill Bill: Vol. 1 (2003)	Action|Crime|Thriller
63	7042	7153	Lord of the Rings: The Return of the King, The...	Action|Adventure|Drama|Fantasy
64	7250	7361	Eternal Sunshine of the Spotless Mind (2004)	Drama|Romance|Sci-Fi
65	12540	58559	Dark Knight, The (2008)	Action|Crime|Drama|IMAX
66	15562	79132	Inception (2010)	Action|Crime|Drama|Mystery|Sci-Fi|Thriller|IMAX

67 rows × 4 columns

- Now creating a Pandas Series `movie_indices` with the index set to the `movieId` column of the `movies` DataFrame and the values set to the corresponding row indices of the `movies` DataFrame. This is often used to quickly find the index of a movie given its `movieId`
- Merging the `ratings` and `movies` dataframes on the `movieId` column, creating a new dataframe `ratings_title` with the movie titles added. It then creates a pivot table `user_movie_ratings` with the rows representing movie IDs, the columns representing user IDs, and the values representing the ratings given by each user to each movie. Finally, printing the dimensions of the pivot table and displays the first 6 rows and 10 columns as an example subset.

```
dataset dimensions:  (67, 104193)

Subset example:
```

userId	4	6	7	11	13	14	15	16	17	19
movieId										
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	NaN	NaN	NaN	NaN	NaN	NaN	4.5	NaN	NaN	NaN
47	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
110	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
111	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN

- Converting the `user_movie_ratings` DataFrame into a sparse matrix format using `csr_matrix` function from the `scipy.sparse` module. The `csr_matrix` function converts the DataFrame into a Compressed Sparse Row matrix which can be used as an input for the `NearestNeighbors` algorithm.
- Storing csr_matrix(user_movie_ratings) in csr_data and printing it
- Computing an imputer with a mean strategy and fitting the imputer to the csr_data and rating. This helps for the missing values to be imputed with the mean of the non-missing values in the respective columns.
- Printing movie_indices

```
movieId
1         0
32        1
47        2
50        3
110       4
         ..
6874     62
7153     63
7361     64
58559    65
79132    66
Length: 67, dtype: int64
```

- Now, creating a K-Nearest Neighbors (KNN) model with cosine similarity as the distance metric, brute force algorithm, and 20 nearest neighbors.
- Filtering the `movies` DataFrame for movies with the title "Usual Suspects, The (1995)" and retrieves its index.
- Uses the KNN model to find the 20 nearest neighbors (excluding itself) to the movie at the retrieved index.
- Prints the titles of the recommended movies.

```
19    Dances with Wolves (1990)
Name: title, dtype: object
17    Aladdin (1992)
Name: title, dtype: object
13    Fugitive, The (1993)
Name: title, dtype: object
6    Apollo 13 (1995)
Name: title, dtype: object
46    L.A. Confidential (1997)
Name: title, dtype: object
5    Taxi Driver (1976)
Name: title, dtype: object
12    Lion King, The (1994)
Name: title, dtype: object
14    Jurassic Park (1993)
Name: title, dtype: object
49    Life Is Beautiful (La Vita è bella) (1997)
Name: title, dtype: object
26    2001: A Space Odyssey (1968)
Name: title, dtype: object
34    Aliens (1986)
Name: title, dtype: object
43    Groundhog Day (1993)
Name: title, dtype: object
41    Terminator, The (1984)
Name: title, dtype: object
25    Casablanca (1942)
Name: title, dtype: object
23    Trainspotting (1996)
Name: title, dtype: object
42    Shining, The (1980)
Name: title, dtype: object
27    Die Hard (1988)
Name: title, dtype: object
55    Being John Malkovich (1999)
Name: title, dtype: object
36    Apocalypse Now (1979)
Name: title, dtype: object
```

Using Cosaine Similarity Algorithm

- Printing the movies dataset

	index	movieId	title	genres
0	0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	31	32	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller
2	46	47	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
3	49	50	Usual Suspects, The (1995)	Crime|Mystery|Thriller
4	108	110	Braveheart (1995)	Action|Drama|War
...	...	...	...	...
62	6765	6874	Kill Bill: Vol. 1 (2003)	Action|Crime|Thriller
63	7042	7153	Lord of the Rings: The Return of the King, The...	Action|Adventure|Drama|Fantasy
64	7250	7361	Eternal Sunshine of the Spotless Mind (2004)	Drama|Romance|Sci-Fi
65	12540	58559	Dark Knight, The (2008)	Action|Crime|Drama|IMAX
66	15562	79132	Inception (2010)	Action|Crime|Drama|Mystery|Sci-Fi|Thriller|IMAX

67 rows × 4 columns

- Copying the ratings dataset values and movies dataset values into ratings_df and movies_df
- Importing CountVectorizer from sklearn library
- Implementing two functions for converting the genres feature into list form by splitting each genre value one function splits “|” separated values and the other separates comma separated values
- Clean_data function is used to convert all strings to lower case and strip names of space
- Implementing two functions create_soup and find_genre that creates string mixtures of preferences
- set_score_1 function adds a new column \`score_1\` to a pandas DataFrame \`movies_df\`, which represents the result of the first algorithm

used to calculate the similarity between movies. This function first extracts the similarity scores from the `similarity` column of `movies_df`. It then sorts the movies based on these similarity scores. For each movie in `movies_df`, the function assigns the corresponding similarity score from `sim_scores` to the `score_1` column. This is done by using the `loc` function to access the appropriate row in `movies_df`, and then setting the `score_1` value equal to the corresponding similarity score in `sim_scores`.

- The `run_algorithm_1()` function uses a string `user_soup` to represent user preferences for movie genres. It then creates a subset of movies (`movies_1`) that only includes movies with user-preferred genres. The function then creates a count matrix of the word occurrences in the `soup` column of `movies_1` using `CountVectorizer()`. It calculates the cosine similarity between the user preferences and each movie in `movies_1`, adds the similarity scores to a new column `similarity` in the `movies_df` dataframe, and sorts the dataframe by the similarity scores. Finally, it assigns the movie indices to `sim_indices` and `movie_indices` variables for later use.

- The recommend function takes a movie name and a number 'n' as inputs and recommends 'n' number of similar movies based on two algorithms. The first algorithm considers the user's preferred genres and the second algorithm considers the similarity in movie plots. It calculates the similarity scores for each algorithm, adds them up, and sorts the movies based on the total score. It returns a list of 'n' recommended movies.

	similarity	score
67	1.118034	1.118034
14	1.118034	1.118034
51	0.944272	0.944272
41	0.944272	0.944272
37	0.944272	0.944272
..	...	...
40	0.000000	0.000000
50	0.000000	0.000000
49	0.000000	0.000000
42	0.000000	0.000000
38	0.000000	0.000000

- Now applying two functions `to_list()` and `clean_data()` to the "genres" column of the "movies_df" dataframe to convert the string of genres to a list of genres, and clean any unwanted characters. Then, it creates a new column called "soup" by applying the `create_soup()` function to each row of the dataframe. The `create_soup()` function combines the genres and the title of each movie into a string that represents the movie's content. The resulting "soup" column contains a mixture of movie titles and genres which will be used as the input for the content-based recommendation algorithm.

- Calling the `recommend` function with the movie name "Usual Suspects, The (1995)" and a number `n=20` as input. The output will be a pandas Series object containing the titles of the recommended movies.

```
14                              Jurassic Park (1993)
51                              Matrix, The (1999)
41                              Terminator, The (1984)
37    Star Wars: Episode VI - Return of the Jedi (1983)
31    Star Wars: Episode V - The Empire Strikes Back...
16                              Blade Runner (1982)
7             Star Wars: Episode IV - A New Hope (1977)
34                              Aliens (1986)
66                              Inception (2010)
18                 Terminator 2: Judgment Day (1991)
44                         Back to the Future (1985)
1             Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
26                       2001: A Space Odyssey (1968)
35                         Clockwork Orange, A (1971)
33    Raiders of the Lost Ark (Indiana Jones and the...
45            Indiana Jones and the Last Crusade (1989)
62                        Kill Bill: Vol. 1 (2003)
39                              Alien (1979)
56                              Gladiator (2000)
Name: title, dtype: object
```

Contributions

- **Nakkina Vinay -** Implemented the K-Means algorithm and Cosine Similarity algorithm and contributed in Report making
- **Sakam Sai Santhosh -** Implemented the K-Means algorithm and Cosine Similarity algorithm and contributed in Report making
- **Geda Durga Vara Praveen -** Implemented KNN algorithm and contributed in Report making