pyda-usecase-23msp3068-23msp3074

November 3, 2023

1 Top IMDb Movies Analysis and Prediction

PYTHON FOR DATA ANALYTICS [CS6102]

By * Gurucharan S (23MSP3074) * Dharshani A (23MSP3068)

AIM: To analyze and visualize the data from the IMDB Top 250 movies, establish database connectivity, and implement linear regression for predicting movie ratings.

DESCRIPTION: The main goal of this project is to gain insights from the IMDb top 250 movies,preprocess the data for analysis, visualize the data patterns, establish a connection with an SQLite database to store and retrieve movie data, and implement a linear regression model to predict movie ratings.

Source of the dataset: IMDB (as inferred from the filename "imdbTop250.csv").

Dataset description: The dataset contains various columns such as title, rating, score, and runtime of the top 250 movies on IMDB.

[1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sqlite3
import shutil

from tabulate import tabulate
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from google.colab import drive
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[1], line 10
      7 import shutil
      9 from tabulate import tabulate
---> 10 from sklearn.model_selection import train_test_split
     11 from sklearn.linear_model import LinearRegression
```

```
     12 from sklearn import metrics

File ~\anaconda3\Lib\site-packages\sklearn\model_selection\__init__.py:3
      1 import typing
----> 3 from ._plot import LearningCurveDisplay, ValidationCurveDisplay
      4 from ._search import GridSearchCV, ParameterGrid, ParameterSampler,␣
↪RandomizedSearchCV
      5 from ._split import (
      6     BaseCrossValidator,
      7     BaseShuffleSplit,
   (…)
     24     train_test_split,
     25 )

File ~\anaconda3\Lib\site-packages\sklearn\model_selection\_plot.py:7
      5 from ..utils import check_matplotlib_support
      6 from ..utils._plotting import _interval_max_min_ratio,␣
↪_validate_score_name
----> 7 from ._validation import learning_curve, validation_curve
     10 class _BaseCurveDisplay:
     11     def _plot_curve(
     12         self,
     13         x_data,
   (…)
     23         errorbar_kw=None,
     24     ):

File ~\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:29
     27 from ..base import clone, is_classifier
     28 from ..exceptions import FitFailedWarning
---> 29 from ..metrics import check_scoring, get_scorer_names
     30 from ..metrics._scorer import _check_multimetric_scoring,␣
↪_MultimetricScorer
     31 from ..preprocessing import LabelEncoder

File ~\anaconda3\Lib\site-packages\sklearn\metrics\__init__.py:7
      1 """
      2 The :mod:`sklearn.metrics` module includes score functions, performance
↪metrics
      3 and pairwise metrics and distance computations.
      4 """
----> 7 from . import cluster
      8 from ._classification import (
      9     accuracy_score,
     10     balanced_accuracy_score,
   (…)
     27     zero_one_loss,
     28 )
```

```
     29 from ._dist_metrics import DistanceMetric

File ~\anaconda3\Lib\site-packages\sklearn\metrics\cluster\__init__.py:9
      1 """
      2 The :mod:`sklearn.metrics.cluster` submodule contains evaluation metric
 ↪for
      3 cluster analysis results. There are two forms of evaluation:
   (…)
      6 - unsupervised, which does not and measures the 'quality' of the model␣
 ↪itself.
      7 """
      8 from ._bicluster import consensus_score
----> 9 from ._supervised import (
     10     adjusted_mutual_info_score,
     11     adjusted_rand_score,
     12     completeness_score,
     13     contingency_matrix,
     14     entropy,
     15     expected_mutual_information,
     16     fowlkes_mallows_score,
     17     homogeneity_completeness_v_measure,
     18     homogeneity_score,
     19     mutual_info_score,
     20     normalized_mutual_info_score,
     21     pair_confusion_matrix,
     22     rand_score,
     23     v_measure_score,
     24 )
     25 from ._unsupervised import (
     26     calinski_harabasz_score,
     27     davies_bouldin_score,
     28     silhouette_samples,
     29     silhouette_score,
     30 )
     32 __all__ = [
     33     "adjusted_mutual_info_score",
     34     "normalized_mutual_info_score",
   (…)
     51     "consensus_score",
     52 ]

File ~\anaconda3\Lib\site-packages\sklearn\metrics\cluster\_supervised.py:29
     27 from ...utils.multiclass import type_of_target
     28 from ...utils.validation import check_array, check_consistent_length
---> 29 from ._expected_mutual_info_fast import expected_mutual_information
     32 def check_clusterings(labels_true, labels_pred):
     33     """Check that the labels arrays are 1D and of same dimension.
     34
```

```
     35     Parameters
   (…)
     41         The predicted labels.
     42     """

File <frozen importlib._bootstrap>:405, in parent(self)

KeyboardInterrupt:
```

```
[ ]: # Mount Google Drive
drive.mount('/content/drive')
src = '/content/drive/MyDrive/Colab Notebooks/imdbTop250.csv'
dst = '/content/drive/My Drive/Colab Notebooks/imdbTop250m.csv'

# Copy file from source location to destination location
shutil.copy(src, dst)
```

Data Pre-Processing: The data preprocessing involves reading the dataset, slicing it to get the first 50 movies, and handling missing values by dropping rows that have any.

Exploratory Data Analysis(Visualization): The exploratory data analysis comprises several visual representations:

- Boxplot to identify outliers in votes.
- Scatter plot for comparing votes against runtime.
- Bar plot to show top 20 movie ratings.
- Histogram for comparing movie rankings with their runtime.

Storing Data in Database: The processed data is stored in an SQLite database named "usecase.db". The movie details are saved under the table "TopIMDBMovies". The database further allows for querying and extraction of movies based on different criteria related to their ratings. An SQLite database named "usecase.db" is created. The data from the dataset (columns 1 to 10) is stored in a table named "TopIMDBMovies" in the SQLite database. Various SQL queries are executed to retrieve and analyze data. For example:

- Movies with a rating of less than 8.
- The count of movies with a rating greater than 8.
- The movie with the highest rating.
- The movie with the lowest rating.

Building Predictive model and testing: Linear regression is a statistical method that models the relationship between a dependent variable and one or more independent variables. In predictive analysis, it is used to forecast values based on known data, allowing for the prediction of outcomes based on input variables.

In the below code linear regression is being applied to predict movie "Rating" based on its "Score". By splitting the data into training and testing sets, the model is trained on a subset of the data and then evaluated on unseen data. This approach, commonly used in predictive analysis, helps in understanding how well the model will generalize to new, previously unseen data.

```
[ ]: # Define a class for the use case
class Usecase:
    @staticmethod
    def dataPreprocessing():
        # Read the CSV file into a DataFrame
        df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/imdbTop250m.
 ↪csv')
        df = df.iloc[:51]  # Filter the first 50 rows
        df = df.dropna()  # Drop rows with missing values
        return df

    df1 = dataPreprocessing()


# Data Visualization code snippets

    @staticmethod
    def visualization_outliers(df):
        # Visualize outliers using a boxplot
        plt.figure(figsize=(10, 6))
        plt.xlabel("Total votes")
        plt.title("Identifying Outliers")
        sns.boxplot(y="Votes", data=df)
        plt.show()

    @staticmethod
    def visualization_scatterplot(df):
        # Visualize data using a scatterplot
        plt.figure(figsize=(10, 6))
        plt.xlabel("Total Votes")
        plt.ylabel("Runtime")
        plt.title("Comparison of Total votes and Runtime")
        sns.scatterplot(x="Votes", y="RunTime", data=df)
        plt.show()

    @staticmethod
    def visualization_barplot(df):
        # Visualize top 10 movies and their ratings using a barplot
        df_subset = df.iloc[0:10]
        plt.figure(figsize=(10, 6))
        plt.title("Movies and their Ratings")
        sns.barplot(x="Title", y="Rating", data=df_subset)
        plt.xticks(rotation=90)  # Rotate x-axis labels for clarity
        plt.show()

    @staticmethod
    def visualization_histogram(df):
```

```
        # Visualize data using a histogram
        plt.figure(figsize=(10, 6))
        plt.xlabel("Ranking")
        plt.ylabel("Runtime")
        plt.title("Comparison of Total votes and Runtime")
        sns.histplot(data=df, x="Ranking", y="RunTime")
        plt.show()


# Database Connectivity code snippets

    @staticmethod
    def database_connectivity(df1):
        # Extract columns 1 to 10 from the df1 DataFrame
        df2 = df1.iloc[:, 1:11]

        # Create a connection to the SQLite database
        conn = sqlite3.connect('usecase.db')
        cursor = conn.cursor()

        # Write the data to the database
        df2.to_sql(name="TopIMDBMovies", con=conn, if_exists='replace',␣
 ↪index=False)

        # Query to retrieve records where RATING is less than 8
        select_query = "SELECT * FROM TopIMDBMovies WHERE RATING < 8"

        # Query to count the number of movies with a RATING greater than 8
        count_query = "SELECT COUNT(TITLE) AS BEST_MOVIES FROM TopIMDBMovies␣
 ↪WHERE RATING > 8"

        # Query to display the movie with the highest rating
        highest_query = "SELECT * FROM TopIMDBMovies WHERE Rating IN (SELECT␣
 ↪MAX(Rating) FROM TopIMDBMovies )"

        # Query to display the movie with the minimum rating
        lowest_query = "SELECT * FROM TopIMDBMovies WHERE Rating IN (SELECT␣
 ↪MIN(Rating) FROM TopIMDBMovies )"

        # Execute the queries and fetch results
        cursor.execute(select_query)
        results_low_rating = cursor.fetchall()

        cursor.execute(count_query)
        result_best_movies = cursor.fetchall()

        cursor.execute(highest_query)
```

```python
        result_highest_rating = cursor.fetchall()

        cursor.execute(lowest_query)
        result_lowest_rating = cursor.fetchall()

        # Get column names (headers) for the first query
        headers_low_rating = [description[0] for description in conn.
↪execute(select_query).description]

        # Get column names (headers) for the second query
        headers_best_movies = [description[0] for description in conn.
↪execute(count_query).description]

        # Get column names (headers) for the third query
        headers_highest_rating = [description[0] for description in conn.
↪execute(highest_query).description]

        # Get column names (headers) for the fourth query
        headers_lowest_rating = [description[0] for description in conn.
↪execute(lowest_query).description]

        # Print results in tabular form using tabulate
        print("Movies with Rating < 8:")
        print(tabulate(results_low_rating, headers_low_rating, tablefmt="grid"))
        print("\n")

        print("Number of Best Movies (Rating > 8):")
        print(tabulate(result_best_movies, headers_best_movies,␣
↪tablefmt="grid"))
        print("\n")

        print("Movie with the Highest Rating:")
        print(tabulate(result_highest_rating, headers_highest_rating,␣
↪tablefmt="grid"))
        print("\n")

        print("Movie with the Lowest Rating:")
        print(tabulate(result_lowest_rating, headers_lowest_rating,␣
↪tablefmt="grid"))

        # Commit and close the connection
        conn.commit()
        cursor.close()
        conn.close()
```

```
# Data preparation for linear regression and rating prediction code snippet

    @staticmethod
    def dataset_linearregression(df):
      # Prepare data for linear regression
        y = df["Rating"]
        x = df[["Score"]]
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.
 ↪3, random_state=10)

# Create an instance of the Usecase class
usecase_instance = Usecase()

# Perform data preprocessing
df1 = usecase_instance.dataPreprocessing()
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
usecase_instance.df1
```

[ ]:
```
     Ranking  IMDByear           IMDBlink  \
0          1      1996  /title/tt0076759/
1          2      1996  /title/tt0111161/
2          3      1996  /title/tt0117951/
3          4      1996  /title/tt0114814/
6          7      1996  /title/tt0117731/
7          8      1996  /title/tt0034583/
8          9      1996  /title/tt0083658/
9         10      1996  /title/tt0108052/
10        11      1996  /title/tt0033467/
11        12      1996  /title/tt0110912/
12        13      1996  /title/tt0057012/
13        14      1996  /title/tt0068646/
14        15      1996  /title/tt0116209/
15        16      1996  /title/tt0112573/
17        18      1996  /title/tt0116905/
18        19      1996  /title/tt0114709/
19        20      1996  /title/tt0073486/
21        22      1996  /title/tt0116282/
22        23      1996  /title/tt0110877/
23        24      1996  /title/tt0111495/
24        25      1996  /title/tt0047478/
25        26      1996  /title/tt0080684/
```

27	28	1996	/title/tt0110413/
28	29	1996	/title/tt0093779/
29	30	1996	/title/tt0114388/
30	31	1996	/title/tt0062622/
31	32	1996	/title/tt0105236/
32	33	1996	/title/tt0112682/
33	34	1996	/title/tt0082971/
34	35	1996	/title/tt0117887/
35	36	1996	/title/tt0109445/
36	37	1996	/title/tt0088846/
37	38	1996	/title/tt0071853/
38	39	1996	/title/tt0047396/
39	40	1996	/title/tt0114369/
40	41	1996	/title/tt0090605/
41	42	1996	/title/tt0066921/
43	44	1996	/title/tt0094336/
44	45	1996	/title/tt0060196/
45	46	1996	/title/tt0114746/
47	48	1996	/title/tt0112431/
49	50	1996	/title/tt0112818/

	Title	Date	RunTime	\
0	Star Wars: Episode IV - A New Hope	1977	121	
1	The Shawshank Redemption	1994	142	
2	Trainspotting	1996	93	
3	The Usual Suspects	1995	106	
6	Star Trek: First Contact	1996	111	
7	Casablanca	1942	102	
8	Blade Runner	1982	117	
9	Schindler's List	1993	195	
10	Citizen Kane	1941	119	
11	Pulp Fiction	1994	154	
12	Dr. Strangelove or: How I Learned to Stop Worr…	1964	95	
13	The Godfather	1972	175	
14	The English Patient	1996	162	
15	Braveheart	1995	178	
17	Lone Star	1996	135	
18	Toy Story	1995	81	
19	One Flew Over the Cuckoo's Nest	1975	133	
21	Fargo	1996	98	
22	The Postman	1994	108	
23	Three Colors: Red	1994	99	
24	Seven Samurai	1954	207	
25	Star Wars: Episode V - The Empire Strikes Back	1980	124	
27	Léon: The Professional	1994	110	
28	The Princess Bride	1987	98	
29	Sense and Sensibility	1995	136	

30	2001: A Space Odyssey	1968	149
31	Reservoir Dogs	1992	99
32	The City of Lost Children	1995	112
33	Indiana Jones and the Raiders of the Lost Ark	1981	115
34	That Thing You Do!	1996	108
35	Clerks	1994	92
36	Brazil	1985	132
37	Monty Python and the Holy Grail	1975	91
38	Rear Window	1954	112
39	Se7en	1995	127
40	Aliens	1986	137
41	A Clockwork Orange	1971	136
43	Withnail & I	1987	107
44	The Good, the Bad and the Ugly	1966	178
45	12 Monkeys	1995	129
47	Babe	1995	91
49	Dead Man Walking	1995	122

	Genre	Rating	Score	Votes	Gross	\
0	Action, Adventure, Fantasy	8.6	90.0	1299781	322.74	
1	Drama	9.3	80.0	2529673	28.34	
2	Drama	8.1	83.0	665213	16.50	
3	Crime, Drama, Mystery	8.5	77.0	1045626	23.34	
6	Action, Adventure, Drama	7.6	71.0	122819	92.00	
7	Drama, Romance, War	8.5	100.0	551575	1.02	
8	Action, Drama, Sci-Fi	8.1	84.0	736925	32.87	
9	Biography, Drama, History	8.9	94.0	1292510	96.90	
10	Drama, Mystery	8.3	100.0	428750	1.59	
11	Crime, Drama	8.9	94.0	1948662	107.93	
12	Comedy, War	8.4	97.0	474011	0.28	
13	Crime, Drama	9.2	100.0	1741574	134.97	
14	Drama, Romance, War	7.4	87.0	186242	78.65	
15	Biography, Drama, History	8.3	68.0	1003006	75.60	
17	Drama, Mystery, Western	7.4	78.0	29329	13.27	
18	Animation, Adventure, Comedy	8.3	95.0	945624	191.80	
19	Drama	8.7	84.0	969223	112.00	
21	Crime, Thriller	8.1	85.0	654107	24.61	
22	Biography, Comedy, Drama	7.7	81.0	35664	21.85	
23	Drama, Mystery, Romance	8.1	100.0	100082	4.04	
24	Action, Drama	8.6	98.0	334350	0.27	
25	Action, Adventure, Fantasy	8.7	82.0	1228288	290.48	
27	Action, Crime, Drama	8.5	64.0	1105424	19.50	
28	Adventure, Family, Fantasy	8.1	77.0	416207	30.86	
29	Drama, Romance	7.7	84.0	111580	43.18	
30	Adventure, Sci-Fi	8.3	84.0	641401	56.95	
31	Crime, Drama, Thriller	8.3	79.0	974876	2.83	
32	Drama, Fantasy, Sci-Fi	7.5	73.0	67358	1.51	

33	Action, Adventure	8.4	85.0	931142	248.16
34	Comedy, Drama, Music	6.9	71.0	67061	25.81
35	Comedy	7.7	70.0	218279	3.15
36	Drama, Sci-Fi	7.9	84.0	196892	9.93
37	Adventure, Comedy, Fantasy	8.2	91.0	525003	1.23
38	Mystery, Thriller	8.5	100.0	473590	36.76
39	Crime, Drama, Mystery	8.6	65.0	1552226	100.13
40	Action, Adventure, Sci-Fi	8.3	84.0	690005	85.16
41	Crime, Sci-Fi	8.3	77.0	798504	6.21
43	Comedy, Drama	7.6	84.0	42901	1.54
44	Adventure, Western	8.8	90.0	731123	6.10
45	Mystery, Sci-Fi, Thriller	8.0	74.0	602534	57.14
47	Comedy, Drama, Family	6.8	83.0	122545	66.60
49	Crime, Drama	7.5	80.0	92996	39.39

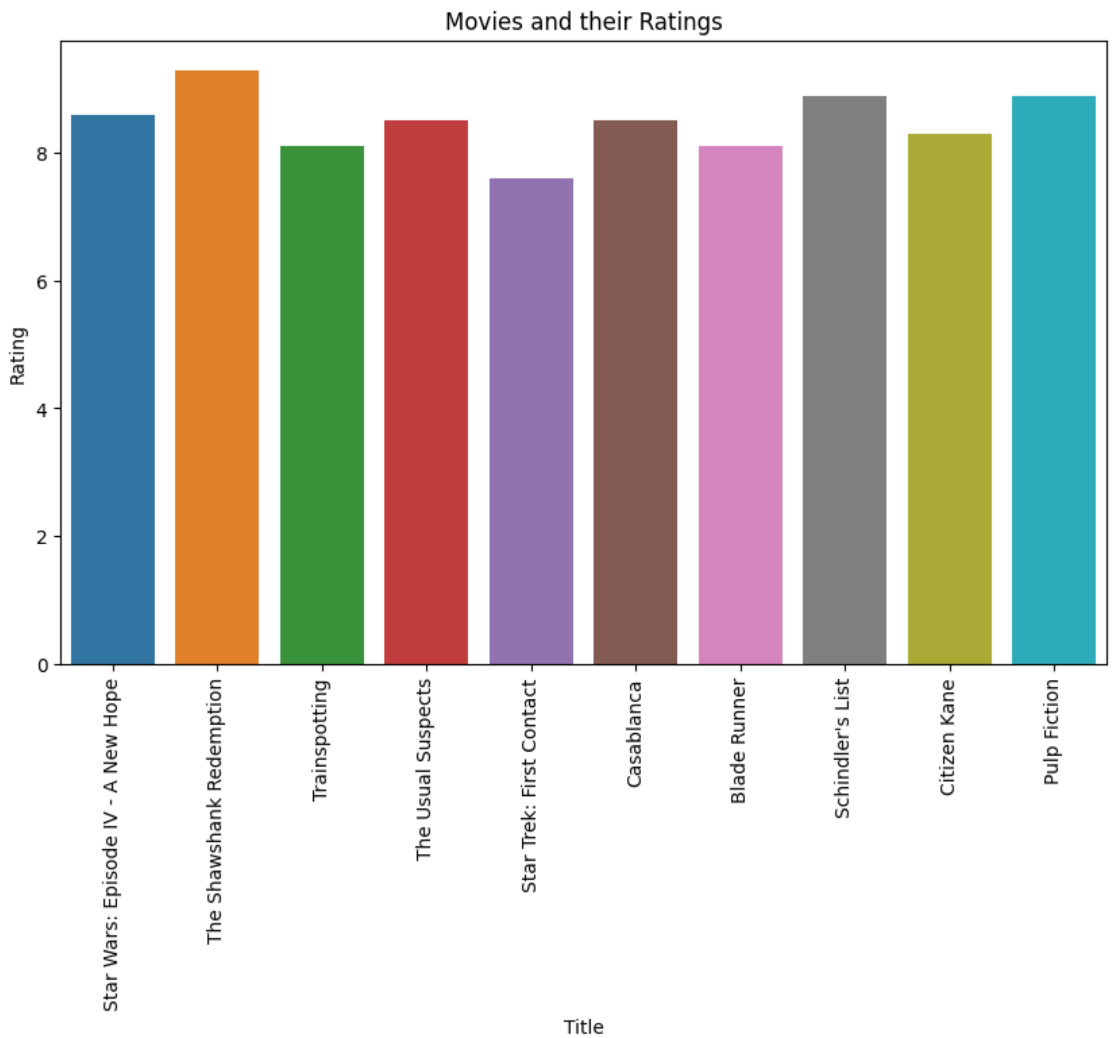
	Director	Cast1 \
0	George Lucas	Mark Hamill
1	Frank Darabont	Tim Robbins
2	Danny Boyle	Ewan McGregor
3	Bryan Singer	Kevin Spacey
6	Jonathan Frakes	Patrick Stewart
7	Michael Curtiz	Humphrey Bogart
8	Ridley Scott	Harrison Ford
9	Steven Spielberg	Liam Neeson
10	Orson Welles	Orson Welles
11	Quentin Tarantino	John Travolta
12	Stanley Kubrick	Peter Sellers
13	Francis Ford Coppola	Marlon Brando
14	Anthony Minghella	Ralph Fiennes
15	Mel Gibson	Mel Gibson
17	John Sayles	Chris Cooper
18	John Lasseter	Tom Hanks
19	Milos Forman	Jack Nicholson
21	Joel Coen, Ethan Coen	William H. Macy
22	Michael Radford, Massimo Troisi	Massimo Troisi
23	Krzysztof Kieslowski	Irène Jacob
24	Akira Kurosawa	Toshirô Mifune
25	Irvin Kershner	Mark Hamill
27	Luc Besson	Jean Reno
28	Rob Reiner	Cary Elwes
29	Ang Lee	Emma Thompson
30	Stanley Kubrick	Keir Dullea
31	Quentin Tarantino	Harvey Keitel
32	Marc Caro, Jean-Pierre Jeunet	Ron Perlman
33	Steven Spielberg	Harrison Ford
34	Tom Hanks	Tom Hanks
35	Kevin Smith	Brian O'Halloran

36	Terry Gilliam	Jonathan Pryce
37	Terry Gilliam, Terry Jones	Graham Chapman
38	Alfred Hitchcock	James Stewart
39	David Fincher	Morgan Freeman
40	James Cameron	Sigourney Weaver
41	Stanley Kubrick	Malcolm McDowell
43	Bruce Robinson	Richard E. Grant
44	Sergio Leone	Clint Eastwood
45	Terry Gilliam	Bruce Willis
47	Chris Noonan	James Cromwell
49	Tim Robbins	Susan Sarandon

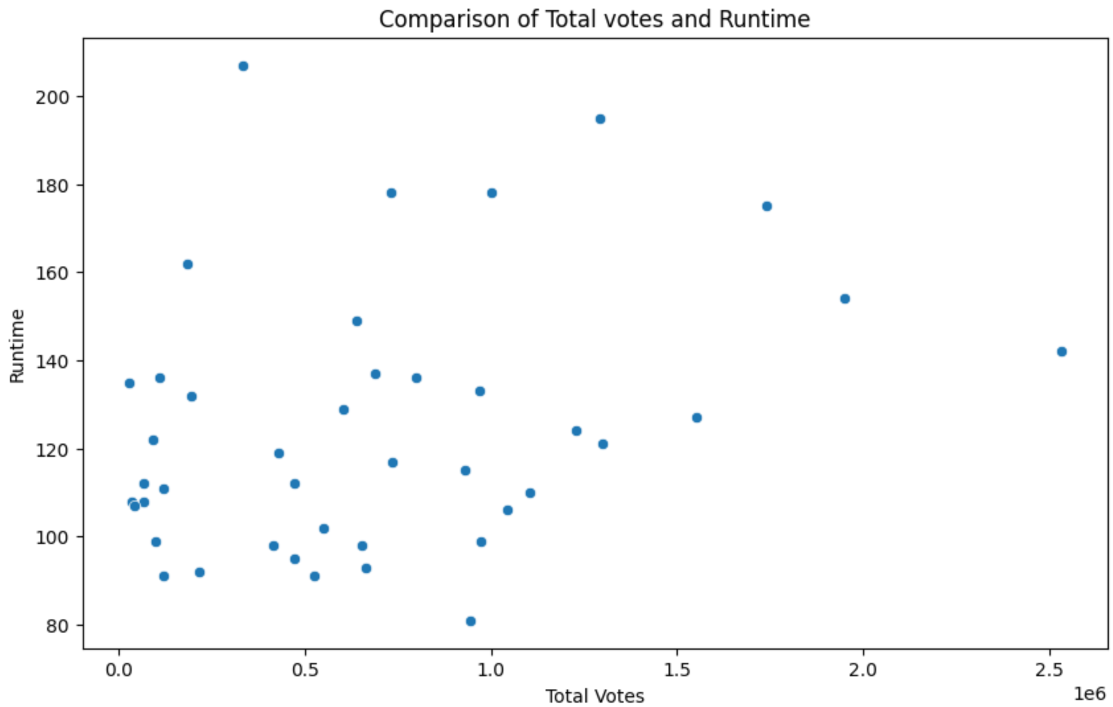
	Cast2	Cast3	Cast4
0	Harrison Ford	Carrie Fisher	Alec Guinness
1	Morgan Freeman	Bob Gunton	William Sadler
2	Ewen Bremner	Jonny Lee Miller	Kevin McKidd
3	Gabriel Byrne	Chazz Palminteri	Stephen Baldwin
6	Jonathan Frakes	Brent Spiner	LeVar Burton
7	Ingrid Bergman	Paul Henreid	Claude Rains
8	Rutger Hauer	Sean Young	Edward James Olmos
9	Ralph Fiennes	Ben Kingsley	Caroline Goodall
10	Joseph Cotten	Dorothy Comingore	Agnes Moorehead
11	Uma Thurman	Samuel L. Jackson	Bruce Willis
12	George C. Scott	Sterling Hayden	Keenan Wynn
13	Al Pacino	James Caan	Diane Keaton
14	Juliette Binoche	Willem Dafoe	Kristin Scott Thomas
15	Sophie Marceau	Patrick McGoohan	Angus Macfadyen
17	Elizabeth Peña	Stephen Mendillo	Stephen J. Lang
18	Tim Allen	Don Rickles	Jim Varney
19	Louise Fletcher	Michael Berryman	Peter Brocco
21	Frances McDormand	Steve Buscemi	Peter Stormare
22	Philippe Noiret	Maria Grazia Cucinotta	Renato Scarpa
23	Jean-Louis Trintignant	Frédérique Feder	Jean-Pierre Lorit
24	Takashi Shimura	Keiko Tsushima	Yukiko Shimazaki
25	Harrison Ford	Carrie Fisher	Billy Dee Williams
27	Gary Oldman	Natalie Portman	Danny Aiello
28	Mandy Patinkin	Robin Wright	Chris Sarandon
29	Kate Winslet	James Fleet	Tom Wilkinson
30	Gary Lockwood	William Sylvester	Daniel Richter
31	Tim Roth	Michael Madsen	Chris Penn
32	Daniel Emilfork	Judith Vittet	Dominique Pinon
33	Karen Allen	Paul Freeman	John Rhys-Davies
34	Liv Tyler	Charlize Theron	Tom Everett Scott
35	Jeff Anderson	Marilyn Ghigliotti	Lisa Spoonauer
36	Kim Greist	Robert De Niro	Katherine Helmond
37	John Cleese	Eric Idle	Terry Gilliam
38	Grace Kelly	Wendell Corey	Thelma Ritter

```
39             Brad Pitt            Kevin Spacey   Andrew Kevin Walker
40         Michael Biehn             Carrie Henn           Paul Reiser
41         Patrick Magee           Michael Bates         Warren Clarke
43           Paul McGann      Richard Griffiths           Ralph Brown
44           Eli Wallach           Lee Van Cleef          Aldo Giuffrè
45       Madeleine Stowe               Brad Pitt         Joseph Melito
47       Magda Szubanski    Christine Cavanaugh     Miriam Margolyes
49             Sean Penn          Robert Prosky       Raymond J. Barry
```

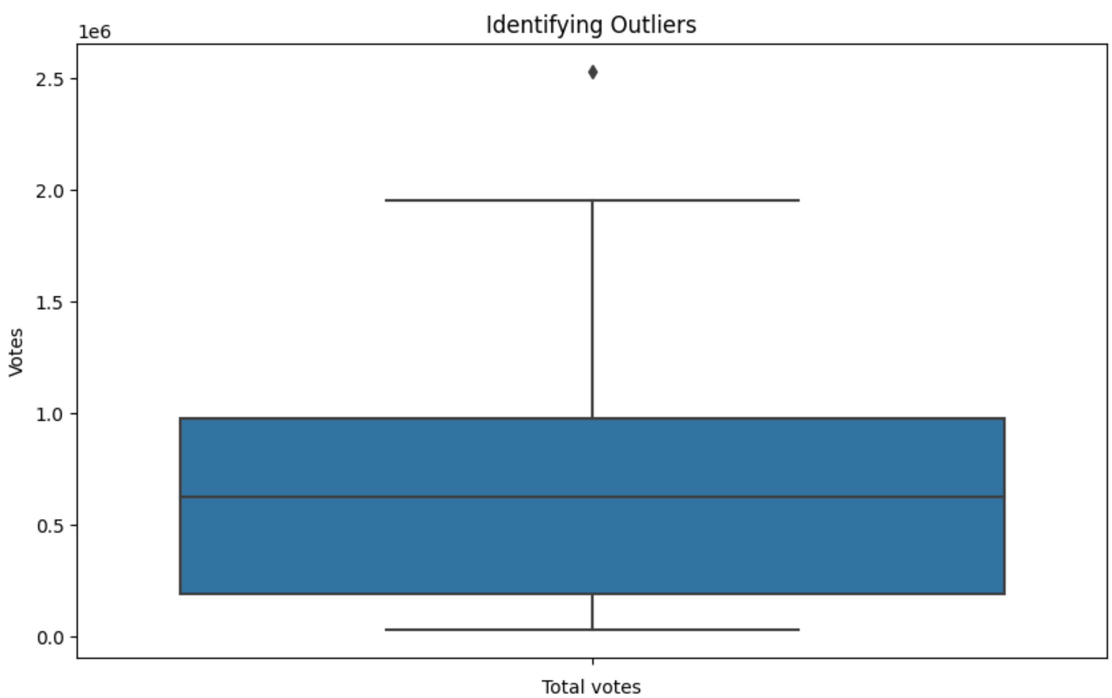
[ ]:
```
usecase_instance.visualization_barplot(df1)
```



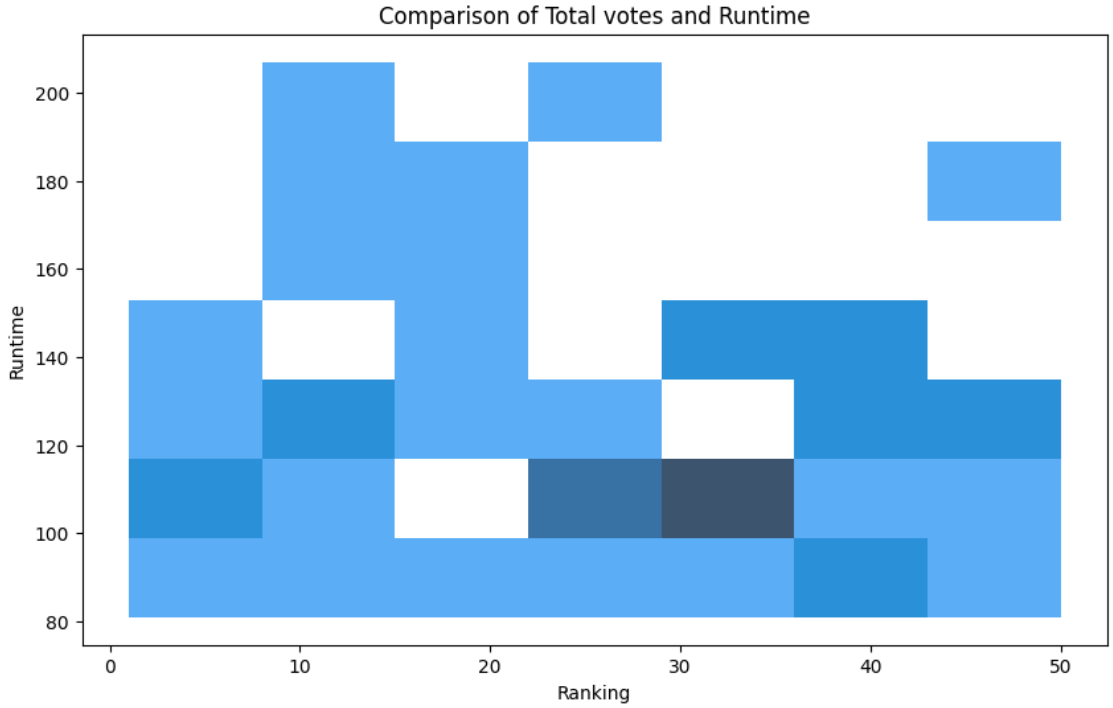
[ ]:
```
usecase_instance.visualization_scatterplot(df1)
```



[ ]:
```
usecase_instance.visualization_outliers(df1)
```



[ ]:
```
usecase_instance.visualization_histogram(df1)
```



[ ]:
```
# Call database connectivity function
usecase_instance.database_connectivity(df1)
```

```
Movies with Rating < 8:
+------------+-------------------+---------------------------+--------+---------
--+-------------------------+----------+---------+---------+---------+
|   IMDByear | IMDBlink          | Title                     |   Date |
RunTime | Genre                   |   Rating |   Score |   Votes |   Gross |
+============+===================+===========================+========+=========
==+=========================+==========+=========+=========+=========+
|       1996 | /title/tt0117731/ | Star Trek: First Contact  |   1996 |
111 | Action, Adventure, Drama |      7.6 |      71 |  122819 |   92    |
+------------+-------------------+---------------------------+--------+---------
--+-------------------------+----------+---------+---------+---------+
|       1996 | /title/tt0116209/ | The English Patient       |   1996 |
162 | Drama, Romance, War      |      7.4 |      87 |  186242 |   78.65 |
+------------+-------------------+---------------------------+--------+---------
--+-------------------------+----------+---------+---------+---------+
|       1996 | /title/tt0116905/ | Lone Star                 |   1996 |
135 | Drama, Mystery, Western  |      7.4 |      78 |   29329 |   13.27 |
+------------+-------------------+---------------------------+--------+---------
--+-------------------------+----------+---------+---------+---------+
```

```
|       1996 | /title/tt0110877/ | The Postman                |   1994 |
108 | Biography, Comedy, Drama |      7.7 |      81 |   35664 |   21.85 |
+------------+-------------------+----------------------------+--------+---------
--+-------------------------+----------+---------+----------+---------+
|       1996 | /title/tt0114388/ | Sense and Sensibility      |   1995 |
136 | Drama, Romance           |      7.7 |      84 |  111580 |   43.18 |
+------------+-------------------+----------------------------+--------+---------
--+-------------------------+----------+---------+----------+---------+
|       1996 | /title/tt0112682/ | The City of Lost Children |   1995 |
112 | Drama, Fantasy, Sci-Fi   |      7.5 |      73 |   67358 |    1.51 |
+------------+-------------------+----------------------------+--------+---------
--+-------------------------+----------+---------+----------+---------+
|       1996 | /title/tt0117887/ | That Thing You Do!         |   1996 |
108 | Comedy, Drama, Music     |      6.9 |      71 |   67061 |   25.81 |
+------------+-------------------+----------------------------+--------+---------
--+-------------------------+----------+---------+----------+---------+
|       1996 | /title/tt0109445/ | Clerks                     |   1994 |
92 | Comedy                   |      7.7 |      70 |  218279 |    3.15 |
+------------+-------------------+----------------------------+--------+---------
--+-------------------------+----------+---------+----------+---------+
|       1996 | /title/tt0088846/ | Brazil                     |   1985 |
132 | Drama, Sci-Fi            |      7.9 |      84 |  196892 |    9.93 |
+------------+-------------------+----------------------------+--------+---------
--+-------------------------+----------+---------+----------+---------+
|       1996 | /title/tt0094336/ | Withnail & I               |   1987 |
107 | Comedy, Drama            |      7.6 |      84 |   42901 |    1.54 |
+------------+-------------------+----------------------------+--------+---------
--+-------------------------+----------+---------+----------+---------+
|       1996 | /title/tt0112431/ | Babe                       |   1995 |
91 | Comedy, Drama, Family    |      6.8 |      83 |  122545 |   66.6  |
+------------+-------------------+----------------------------+--------+---------
--+-------------------------+----------+---------+----------+---------+
|       1996 | /title/tt0112818/ | Dead Man Walking           |   1995 |
122 | Crime, Drama             |      7.5 |      80 |   92996 |   39.39 |
+------------+-------------------+----------------------------+--------+---------
--+-------------------------+----------+---------+----------+---------+


Number of Best Movies (Rating > 8):
+---------------+
|   BEST_MOVIES |
+===============+
|            29 |
+---------------+


Movie with the Highest Rating:
+------------+-------------------+---------------------------+--------+----------
```

```
-+----------+----------+---------+---------+---------+
|   IMDByear | IMDBlink           | Title                          |    Date |    RunTime
| Genre    |    Rating |    Score |    Votes |    Gross |
+============+===================+=========================+========+==========
=+=========+=========+=========+========+=========+
|       1996 | /title/tt0111161/ | The Shawshank Redemption |    1994 |        142
| Drama    |       9.3 |       80 | 2529673 |    28.34 |
+------------+-------------------+-------------------------+--------+----------
-+----------+----------+---------+---------+---------+


Movie with the Lowest Rating:
+------------+-------------------+---------+--------+-----------+---------------
--------+----------+---------+---------+---------+
|   IMDByear | IMDBlink           | Title    |    Date |    RunTime | Genre
|   Rating |    Score |    Votes |    Gross |
+============+===================+=========+========+===========+===============
=======+==========+=========+=========+=========+
|       1996 | /title/tt0112431/ | Babe     |    1995 |         91 | Comedy, Drama,
Family |       6.8 |       83 |  122545 |     66.6 |
+------------+-------------------+---------+--------+-----------+---------------
--------+----------+---------+---------+---------+
```

[ ]:
```python
#Setting value for x and y
df2 = usecase_instance.df1
y= df2["Rating"]
x= df2[["Score"]]
x_train,x_test, y_train,y_test = train_test_split(x,y, test_size=0.3,␣
 ↪random_state = 10)
x_train
```

[ ]:

	Score
3	77.0
47	83.0
7	100.0
15	68.0
17	78.0
29	84.0
39	65.0
1	80.0
43	84.0
13	100.0
45	74.0
32	73.0
19	84.0
23	100.0
6	71.0

```
27     64.0
38    100.0
12     97.0
44     90.0
49     80.0
10    100.0
40     84.0
34     71.0
30     84.0
33     85.0
0      90.0
18     95.0
41     77.0
11     94.0
```

[ ]:
```
slr= LinearRegression()
slr.fit(x_train,y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
#Print model coefficients
print('Intercept:',slr.intercept_)
print('Coefficient:', slr.coef_)
```

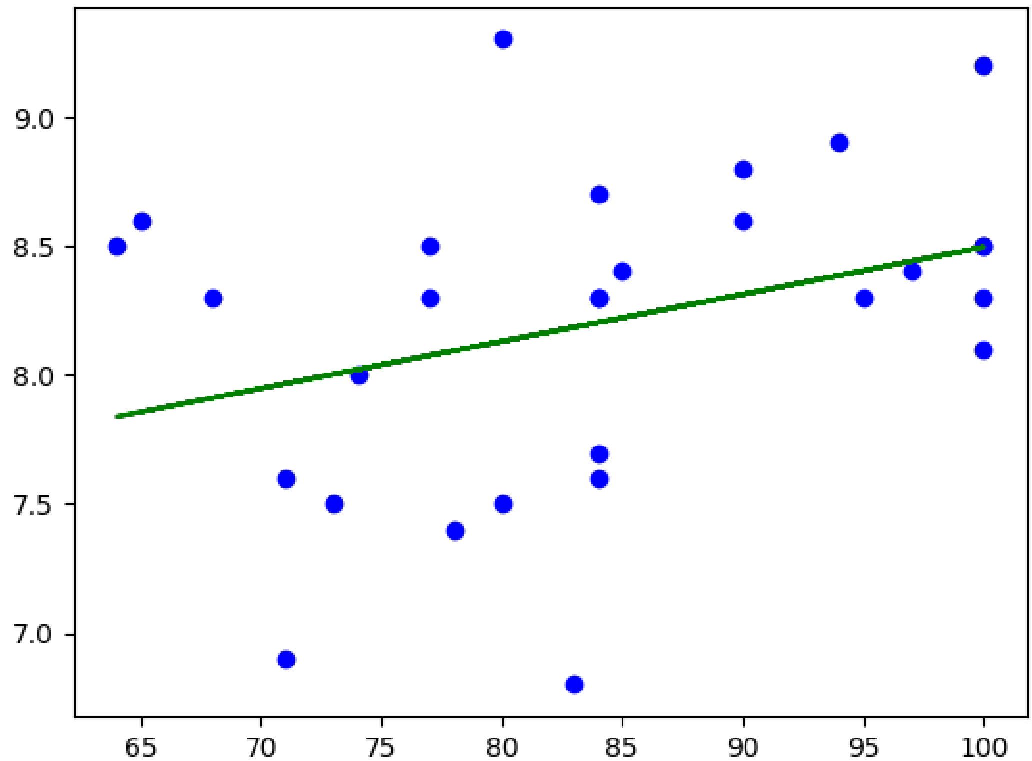
```
Intercept: 6.672150220621699
Coefficient: [0.0182186]
```

[ ]:
```
'''A linear regression model is implemented to predict movie ratings based on␣
↪their scores.
The dataset is split into training and testing sets using a 70:30 split ratio.
↪'''
```

[ ]:
```
'A linear regression model is implemented to predict movie ratings based on
their scores.\nThe dataset is split into training and testing sets using a 70:30
split ratio.'
```

[ ]:
```
#LINE OF Best Fit

# Assuming x_train and y_train are your data
x_train = np.array(x_train).reshape(-1, 1)  # Reshaping if x_train is a 1D array

# Create a linear regression model
model = LinearRegression()

# Fit the model to the data
model.fit(x_train, y_train)
```

```
# Predict y values for the x_train data
y_pred = model.predict(x_train)

# Plot the original data points
plt.scatter(x_train, y_train, color='blue')

# Plot the line of best fit
plt.plot(x_train, y_pred, color='green')

plt.show()
```



[ ]:
```
y_pred_slr = slr.predict(x_test)
print('Prediction result:{}'.format(y_pred_slr))
```

```
Prediction result:[7.94745246 8.1660757  8.11141989 8.38469895 8.20251291
8.07498269
 8.18429431 8.22073151 8.1478571  8.45757336 8.25716872 8.20251291
 8.33004314]
```

[ ]:
```
#actual value and predicted value
slr_diff = pd.DataFrame({'Actual value': y_test,'Predicted Value': y_pred_slr})
slr_diff
```

[ ]:
```
    Actual value  Predicted Value
35           7.7         7.947452
25           8.7         8.166076
31           8.3         8.111420
9            8.9         8.384699
36           7.9         8.202513
28           8.1         8.074983
2            8.1         8.184294
21           8.1         8.220732
22           7.7         8.147857
24           8.6         8.457573
14           7.4         8.257169
8            8.1         8.202513
37           8.2         8.330043
```

[ ]:
```
#predict for any value
slr.predict([[13]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

[ ]:
```
array([6.90899207])
```

[ ]:
```
#R squared value
from sklearn.metrics import accuracy_score
print('R squared value: {:2f}'.format(slr.score(x,y)*100))
```

```
R squared value: 12.534178
```

[ ]:
```
# Calculate the Mean Absolute Error (MAE) between the actual and predicted␣
 ↪values
MeanAbsErr = metrics.mean_absolute_error(y_test, y_pred_slr)

# Calculate the Mean Squared Error (MSE) between the actual and predicted values
MeanSquErr = metrics.mean_squared_error(y_test, y_pred_slr)

# Calculate the Root Mean Squared Error (RMSE) between the actual and predicted␣
 ↪values
RootMeanSqErr = np.sqrt(metrics.mean_squared_error(y_test, y_pred_slr))

# Display the calculated error metrics with precision up to three decimal places
print('Absolute Mean error:', round(MeanAbsErr, 3))
print('Mean Square error:', round(MeanSquErr, 3))
print('Root Mean Square error:', round(RootMeanSqErr, 3))
```

```
Absolute Mean error: 0.284
Mean Square error: 0.134
```

```
Root Mean Square error: 0.366
```

CONCLUSION: The data from the IMDB Top 250 movies was successfully analyzed and visualized. Database operations were performed to store and query the data. A linear regression model was initiated for predicting movie ratings based on their scores.

Overall, the project gives a comprehensive overview of how to handle, process, and analyze data from a real-world dataset. The combination of data visualization, database operations, and the beginning of predictive modeling offers a solid foundation for further enhancements and deeper analyses.