PREDICTING IMDB SCORES

FEATURE ENGINEERING

1.Import Libraries: Import the necessary libraries, including Pandas for data manipulation, scikit-learn for machine learning, NumPy for numerical operations, and Matplotlib for data visualization.
2.Data Loading: Load the dataset from a CSV file named "netflix.csv" with the specified encoding 'latin-1’.
3.Feature Engineering: This is a simple example of feature engineering to capture potential nonlinear relationships between the runtime of movies and their IMDb scores.
4.Data Splitting: Split the data into two sets: the feature set (X) and the target variable (y).
5.Data Splitting (Training and Testing): Further split the data into training and testing sets using **train_test_split**. The testing set will be used to evaluate the model's performance.

6.Model Selection: Choose a machine learning model. In this case, a Linear Regression model is selected to predict IMDb scores based on the features.

7.Model Training: Train the Linear Regression model using the training data (X_train and y_train).

8.Prediction: Use the trained model to make predictions for IMDb scores on the testing data (X_test).

9.Model Evaluation: Calculate the model's performance using two common regression metrics: Mean Squared Error (MSE) and R-squared (R2) score.

10.Printing the Dataset: The program prints the modified dataset (including the new feature "Runtime_squared") to show the dataset with the added feature.

PROGRAM

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import matplotlib.pyplot as plt
file_path = "netflix.csv"
encoding = 'latin-1'  # You can try 'cp1252' or other encodings if needed

# Create a DataFrame from the CSV file with the specified encoding
data = pd.read_csv(file_path, encoding=encoding)
# Create a DataFrame from the dictionary
df = pd.DataFrame(data)

# Feature engineering (example: using Runtime as a feature)
# You can add more features or perform more advanced feature engineering here
df['Runtime_squared'] = df['Runtime'] ** 2
```

```
# Split the data into features (X) and target (y)
X = df[['Runtime', 'Runtime_squared']]
y = df['IMDB Score']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the dataset with the added feature
print(df)
```

OUTPUT

```
                                          Title                  Genre  \
0                                Enter the Anime            Documentary
1                                    Dark Forces               Thriller
2                                        The App  Science fiction/Drama
3                                The Open House         Horror thriller
4                                   Kaali Khuhi                Mystery
..                                          ...                    ...
579        Taylor Swift: Reputation Stadium Tour           Concert Film
580  Winter on Fire: Ukraine's Fight for Freedom            Documentary
581                  Springsteen on Broadway           One-man show
582    Emicida: AmarElo - It's All For Yesterday            Documentary
583     David Attenborough: A Life on Our Planet            Documentary

               Premiere  Runtime  IMDB Score                 Language  \
0        August 5, 2019        58         2.5          English/Japanese
1       August 21, 2020        81         2.6                  Spanish
2     December 26, 2019        79         2.6                  Italian
3      January 19, 2018        94         3.2                  English
4      October 30, 2020        90         3.4                    Hindi
..                  ...       ...         ...                      ...
579  December 31, 2018       125         8.4                  English
580    October 9, 2015        91         8.4  English/Ukranian/Russian
581  December 16, 2018       153         8.5                  English
582   December 8, 2020        89         8.6               Portuguese
583    October 4, 2020        83         9.0                  English
...
582              7921
583              6889

[584 rows x 7 columns]
```

MODEL TRAINING

1.Data Splitting (Features and Target): In this step, you split the dataset into two main parts: the feature set (X) and the target variable (y). This separation is crucial because the model will learn to make predictions based on the features to estimate the target variable.

2.Data Splitting (Training and Testing): The dataset is further divided into training and testing sets using the **train_test_split** function from scikit-learn.

3.Scatter Plot for Model Training: To visualize the training data, a scatter plot is created using Matplotlib. The x-axis represents the "Runtime," and the y-axis represents the "IMDB Score.".

PROGRAM

```
# Print the predicted IMDb scores
print("\nPredicted IMDb Scores:")
print(pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}))
```

OUTPUT

```
Predicted IMDb Scores:
     Actual  Predicted
383     6.7   6.210413
422     6.9   6.299119
90      5.3   6.299119
472     7.1   6.286786
522     7.4   6.414120
..      ...        ...
296     6.4   6.253738
560     7.8   6.727795
167     5.8   6.204947
559     7.7   6.398612
362     6.6   6.359040

[117 rows x 2 columns]
```

PROGRAM FOR MEAN SQUARED ERROR

```
# Print the Mean Squared Error
print(f"\nMean Squared Error: {mse}")
```

OUTPUT

```
Mean Squared Error: 0.9882182648225284
```

SCATTER PLOT FOR MODEL TRAINING

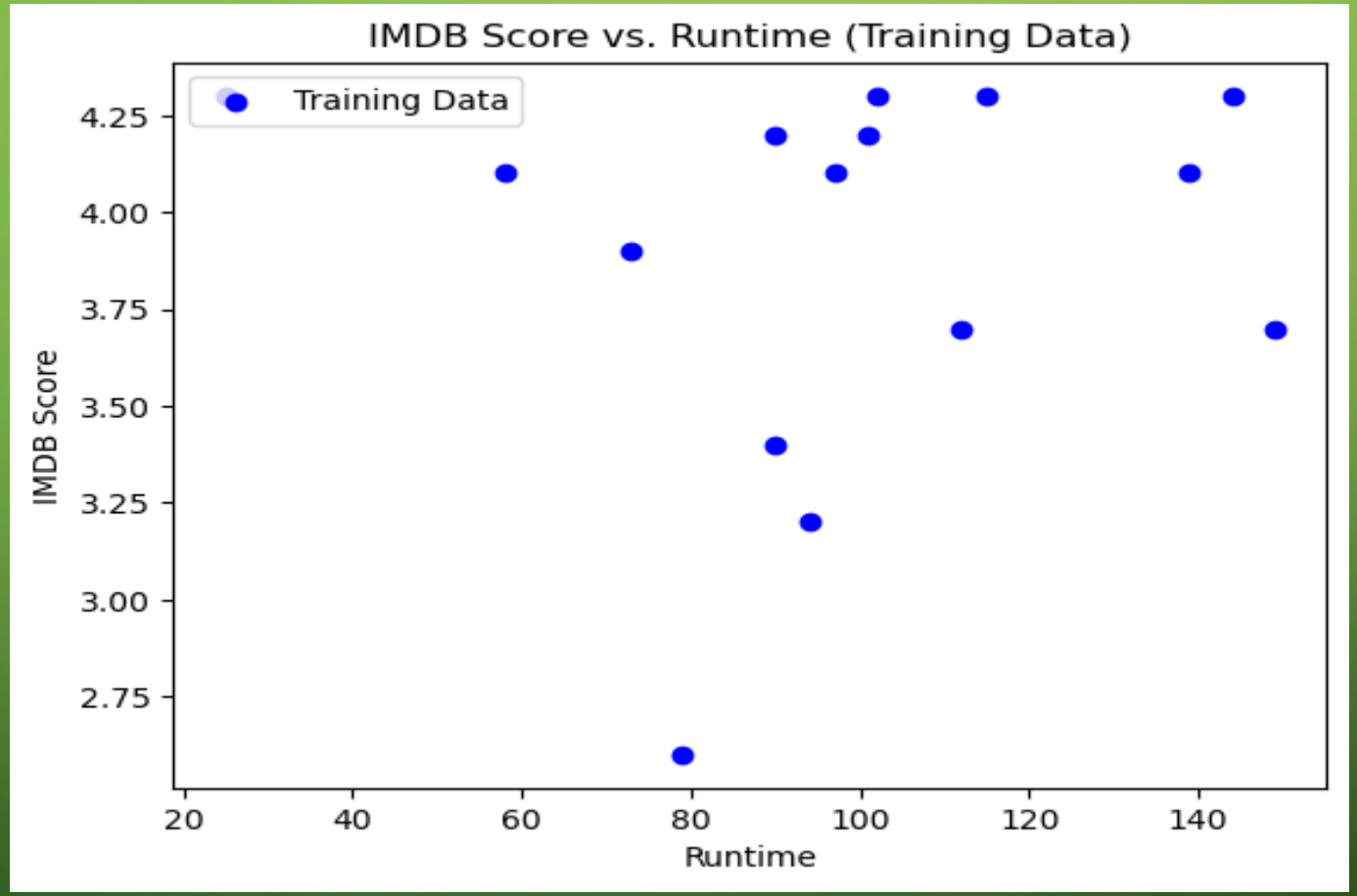
```
# Split the data into features (X) and target (y)
X = df[['Runtime']]
y = df['IMDB Score']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a scatter plot for model training
plt.scatter(X_train, y_train, color='blue', label='Training Data')
plt.xlabel("Runtime")
plt.ylabel("IMDB Score")
plt.legend(loc='upper left')
plt.title("IMDB Score vs. Runtime (Training Data)")

# Show the scatter plot
plt.show()
```

OUTPUT


IMDB Score vs. Runtime (Training Data)
Training Data
IMDB Score
4.25
4.00
3.75
3.50
3.25
3.00
2.75
20
40
60
80
100
120
140
Runtime

EVALUATION

1.Loading the Dataset: The code begins by loading the dataset from a CSV file named "netflix.csv." The dataset contains information about movies, including features like "Genre," "Runtime," and "IMDB Score." The dataset is loaded into a Pandas DataFrame.
2.Feature Engineering: The "Genre" feature is one-hot encoded to convert categorical data into a numerical format suitable for machine learning. The resulting DataFrame (X) contains both the original and one-hot encoded features, while the target variable (y) is "IMDB Score."

3.Data Splitting (Training and Testing): The dataset is split into training and testing sets using the **train_test_split** function. The testing set is used to evaluate the model's performance.

4. **Model Selection:** A Linear Regression model is selected for this task. Linear Regression is a regression algorithm that predicts a continuous target variable based on the features.

5.Model Training: The selected Linear Regression model is trained using the training data (X_train and y_train).

6.Prediction: The trained model is used to make predictions on the test set, resulting in a series of IMDb score predictions (y_pred).

7.Model Evaluation: The performance of the model is evaluated using two common regression metrics:

- **Mean Squared Error (MSE)**
- **R-squared (R2) Score**

8.Print Results: The code prints the computed Mean Squared Error and R-squared score, which help you assess the quality of the model's predictions.

These evaluation metrics provide information about how well the model is performing. A lower MSE and a higher R2 score are indicative of a more accurate model.

PROGRAM

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset (ensure data preprocessing as needed)file_path = "netflix.csv"
encoding = 'latin-1'  # You can try 'cp1252' or other encodings if needed

# Create a DataFrame from the CSV file with the specified encoding
data = pd.read_csv(file_path, encoding=encoding)

# Feature Engineering: Use 'Genre' and 'Runtime' as features
X = data[['Genre', 'Runtime']]
y = data['IMDB Score']

# One-hot encode 'Genre'
X = pd.get_dummies(X, columns=['Genre'], drop_first=True)

# Split the data into a training set and a testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create and train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2) score: {r2}")
```

OUTPUT

Mean Squared Error (MSE): 0.9123928299593 R-squared (R2) score: 0.1209592191852374

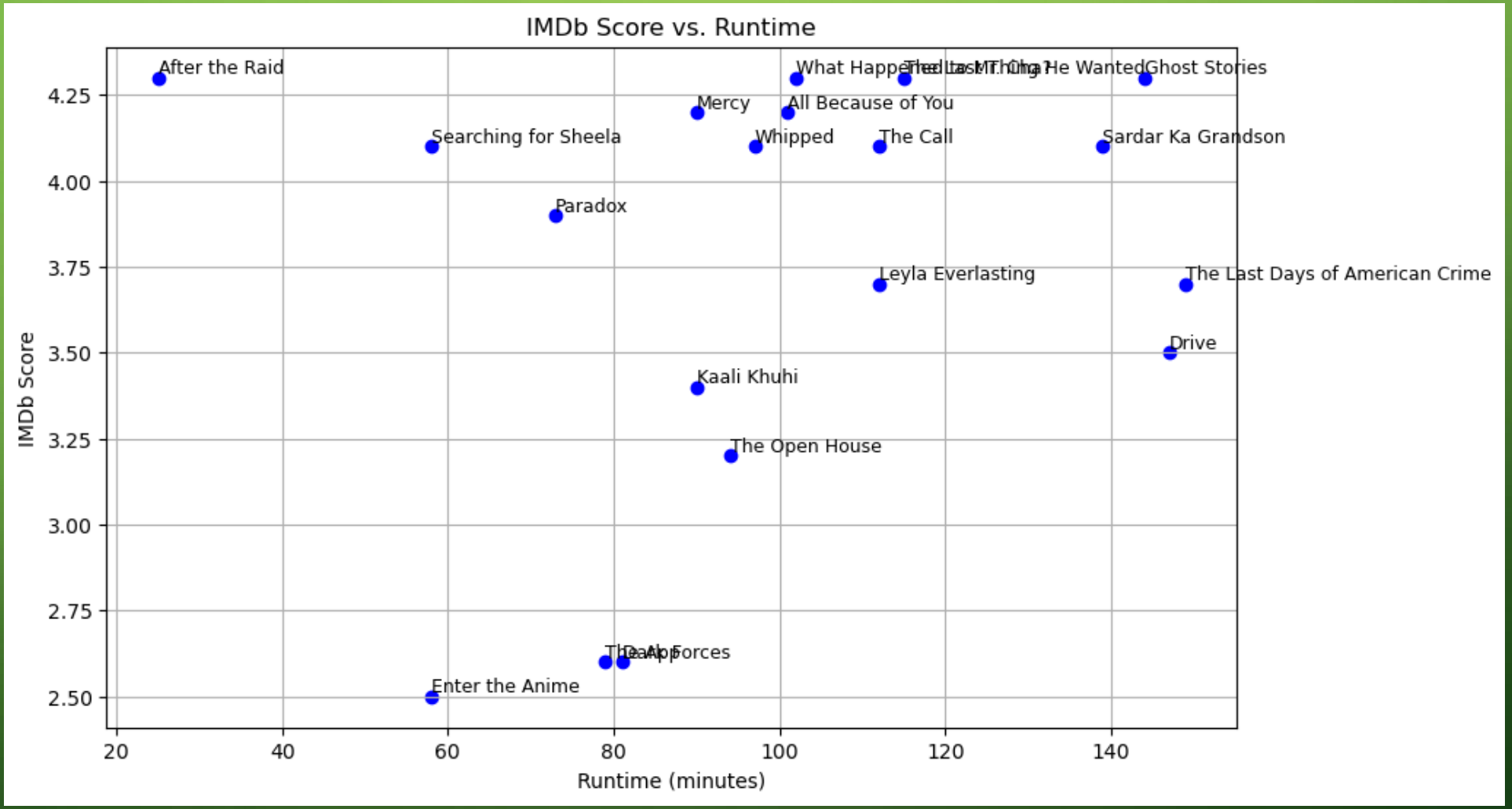
SCATTER PLOT FOR EVALUATION

```
# Create the scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(runtimes, imdb_scores, color='blue', marker='o')
plt.title('IMDb Score vs. Runtime')
plt.xlabel('Runtime (minutes)')
plt.ylabel('IMDb Score')
plt.grid(True)

# Annotate the points with movie titles
for i, title in enumerate(titles):
    plt.annotate(title, (runtimes[i], imdb_scores[i]), fontsize=9, verticalalignment='bottom')

plt.show()
```

OUTPUT


IMDb Score vs. Runtime
IMDb Score
Runtime (minutes)
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2.50
20
40
60
80
100
120
140
After the Raid
Ghost Stories
Mercy
All Because of You
Searching for Sheela
Whipped
The Call
Sardar Ka Grandson
Paradox
Leyla Everlasting
The Last Days of American Crime
Drive
Kaali Khuhi
The Open House
Enter the Anime

CONCLUSION

- Predicting IMDb scores is a challenging task that requires a combination of data science expertise, careful data preprocessing, appropriate model selection, rigorous evaluation, and continuous improvement. When executed effectively, it can provide valuable insights into what makes a movie successful in the eyes of both critics and audiences, contributing to better decision-making in the film industry.