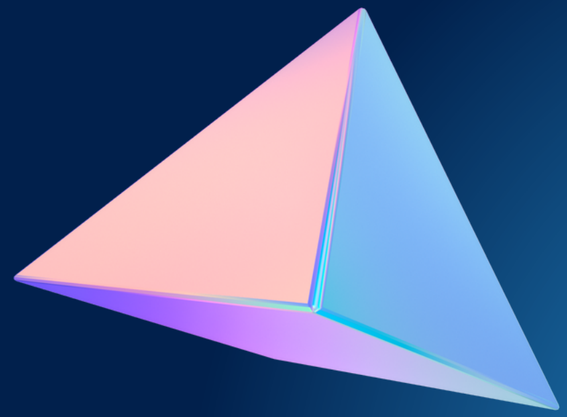
SC1015 Mini Project Detection of MetaScore in IMDB using DS concepts

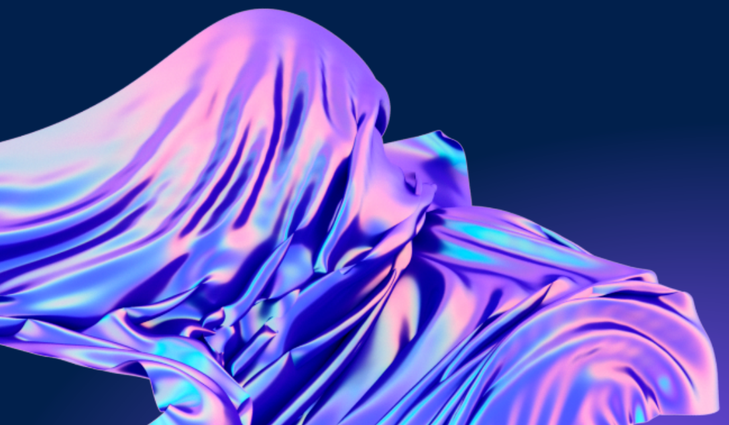
Argel
Shuhao
Jie Hao

Motivation



Meta score is a crucial measure of critical acclaim for movies

Understanding the relationship between these factors and Meta score can provide valuable insights for movie producers, studios, and other stakeholders in the film industry



IMDB rating and number of votes are strong predictors of Meta score, with higher ratings and more votes typically resulting in a higher score

Considering and optimizing these factors can help increase a movie's chances of receiving a higher Meta score and thus, better critical reception

Problem Statement

The problem statement aims to explore the relationships between five factors (IMDB rating, runtime, gross earnings, certificate, and number of votes) and the Meta score of movies.

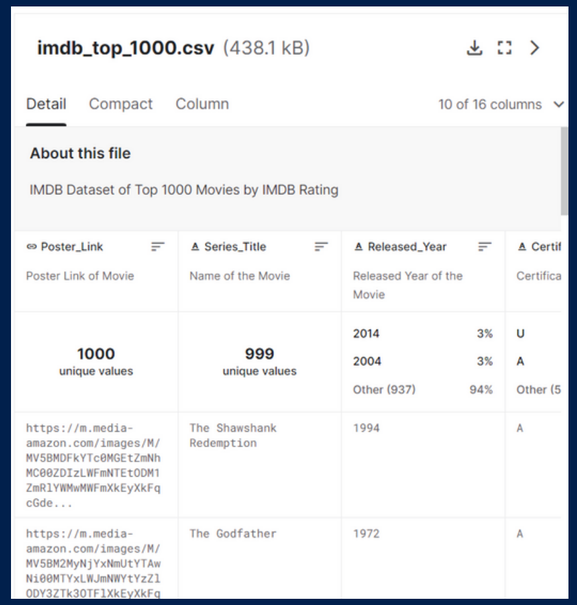
To achieve the above objective, the approach would involve performing exploratory data analysis.

Feature engineering is a technique that can be used to create new features from the existing data, which can help in improving the performance of machine learning models. This technique could also be applied to this problem.

The proposed approach involves using machine learning techniques such as regression or classification to build a predictive model that can determine the Meta score based on the five factors.

Sample Collection

Extracted Data Set from Kaggle.com



imdb_top_1000.csv (438.1 kB)

Detail | Compact | Column | 10 of 16 columns

About this file

IMDB Dataset of Top 1000 Movies by IMDB Rating

Poster_Link	Series_Title	Released_Year	Certif
Poster Link of Movie	Name of the Movie	Released Year of the Movie	Certifica
1000 unique values	999 unique values	2014 3% 2004 3% Other (937) 94%	U A Other (5
https://m.media-amazon.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkEyXkFqcGde...	The Shawshank Redemption	1994	A
https://m.media-amazon.com/images/M/MV5BM2MyNjYxNmUtYTAwNi00MTYxLWJmNWYtYzZlODY3ZTk3OTFlXkEyXkFq	The Godfather	1972	A

Analysing the Data Set using Panda

```
odata = pd.read_csv('imdb_top_1000.csv')
```

Check the vital statistics of the dataset using the **type** and **shape** attributes.

```
print("Data type : ", type(odata))
print("Data dims (train): ", odata.shape)
```

```
Data type :  <class 'pandas.core.frame.DataFrame'>
Data dims (train):  (1000, 16)
```

Dataset has 1000 movies with 16 columns.

Check the variables (and their types) in the dataset using the dtypes attribute.

Exploratory Analysis and Data Preparation

Preliminary Exploration

Analysing the Data Set using Panda

In [4]:
```
print("IMDB Dataset:")
odata.dtypes
```

```
IMDB Dataset:
```

Out[4]:
```
Poster_Link       object
Series_Title      object
Released_Year     object
Certificate       object
Runtime           object
Genre             object
IMDB_Rating      float64
Overview          object
Meta_score       float64
Director          object
Star1             object
Star2             object
Star3             object
Star4             object
No_of_Votes        int64
Gross             object
dtype: object
```

Data Preparation and Cleaning

Data Inspection

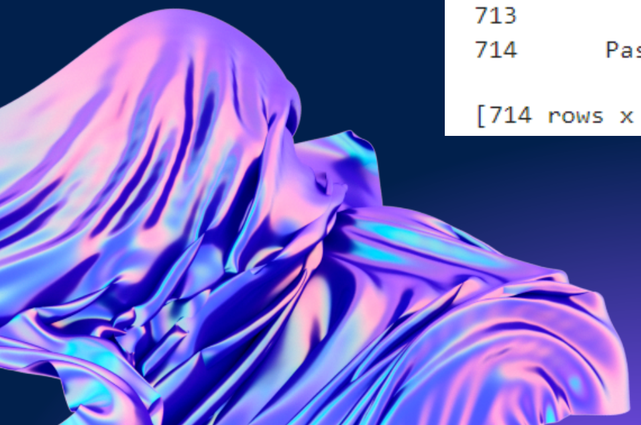
```
df = odata.drop('Poster_Link', axis=1)
df = df.drop('Series_Title', axis=1)
df = df.drop('Released_Year', axis=1)
df = df.drop('Overview', axis=1)
df = df.drop('Genre', axis=1)
df = df.drop('Star1', axis=1)
df = df.drop('Star2', axis=1)
df = df.drop('Star3', axis=1)
df = df.drop('Star4', axis=1)
df = df.drop('Director', axis=1)

print(df)
```

```
    Certificate  Runtime  IMDB_Rating  Meta_score  No_of_Votes        Gross
0             A  142 min          9.3        80.0      2343110   28,341,469
1             A  175 min          9.2       100.0      1620367  134,966,411
2            UA  152 min          9.0        84.0      2303232  534,858,444
3             A  202 min          9.0        90.0      1129952   57,300,000
4             U   96 min          9.0        96.0       689845    4,360,000
..          ...      ...          ...         ...          ...          ...
995           A  115 min          7.6        76.0       166544          NaN
996           G  201 min          7.6        84.0        34075          NaN
997      Passed  118 min          7.6        85.0        43374   30,500,000
998         NaN   97 min          7.6        78.0        26471          NaN
999         NaN   86 min          7.6        93.0        51853          NaN

[1000 rows x 6 columns]
```

Cleaning

```
]: df = df.dropna()
   df.index = np.arange(1, len(df) + 1)
   df['Runtime'] = df['Runtime'].str.replace('min', '')
   df['Gross'] = df['Gross'].str.replace(',', '')
   print(df)
```

```
    Certificate Runtime  IMDB_Rating  Meta_score  No_of_Votes      Gross
1             A     142          9.3        80.0      2343110   28341469
2             A     175          9.2       100.0      1620367  134966411
3            UA     152          9.0        84.0      2303232  534858444
4             A     202          9.0        90.0      1129952   57300000
5             U      96          9.0        96.0       689845    4360000
..          ...     ...          ...         ...          ...        ...
710          PG     157          7.6        77.0        30144     696690
711          GP     144          7.6        50.0        45338    1378435
712           U      78          7.6        65.0       166409  141843612
713           U      87          7.6        96.0        40351   13780024
714      Passed     118          7.6        85.0        43374   30500000

[714 rows x 6 columns]
```

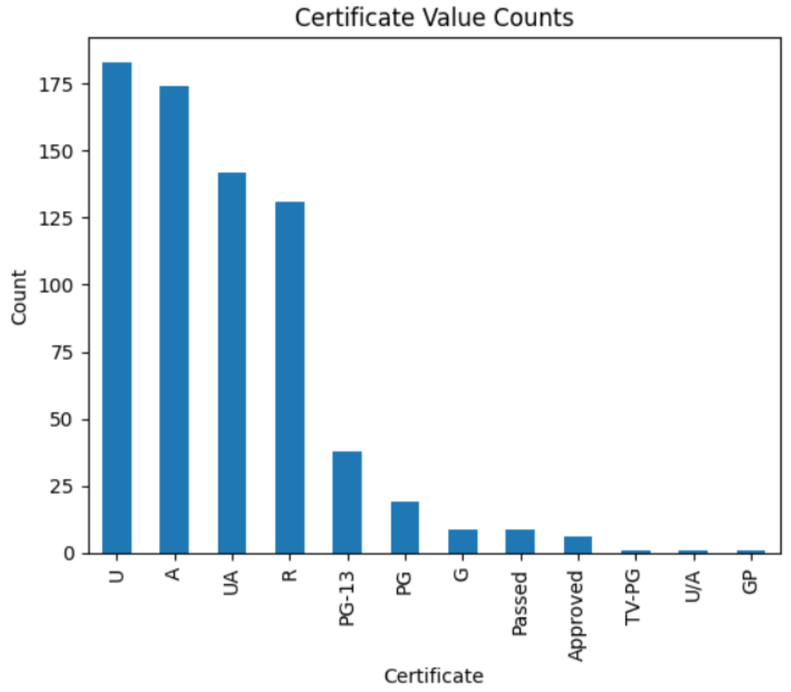


Cleaning

```
]: df["Certificate"].value_counts()
```

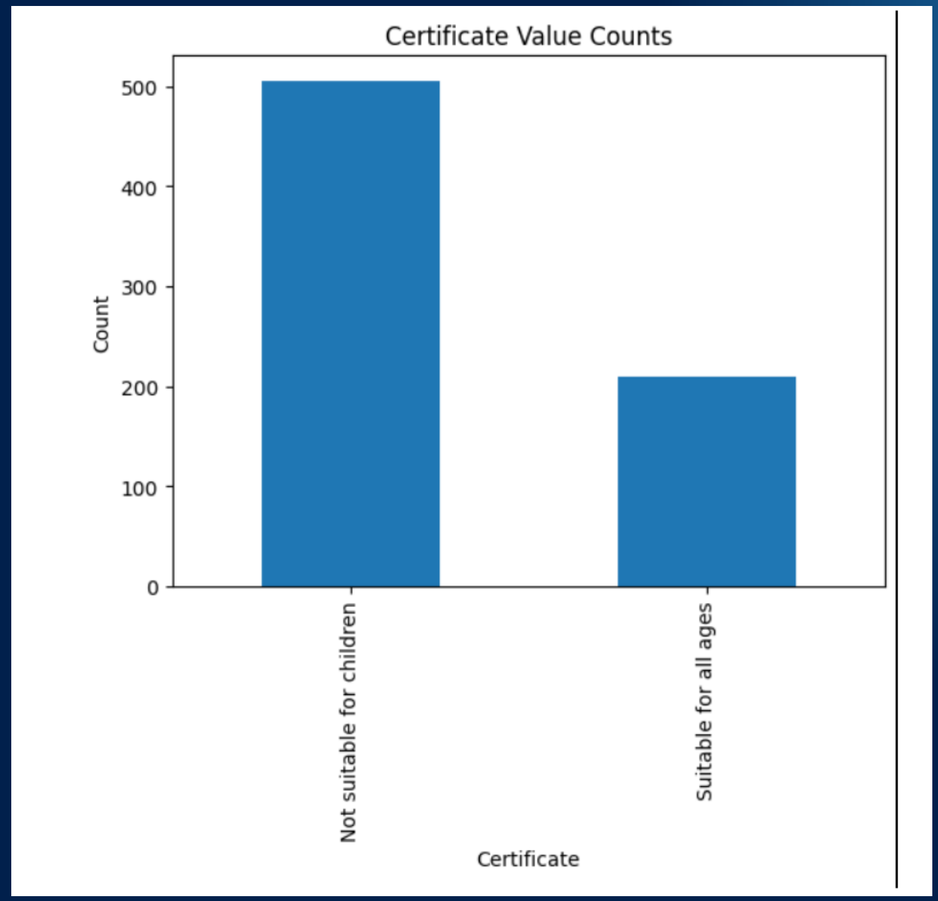
```
]: U            183
   A            174
   UA           142
   R            131
   PG-13         38
   PG            19
   G              9
   Passed         9
   Approved       6
   TV-PG          1
   U/A            1
   GP             1
   Name: Certificate, dtype: int64
```

```
df["Certificate"].value_counts().plot(kind='bar')
plt.title('Certificate Value Counts')
plt.xlabel('Certificate')
plt.ylabel('Count')
plt.show()
```



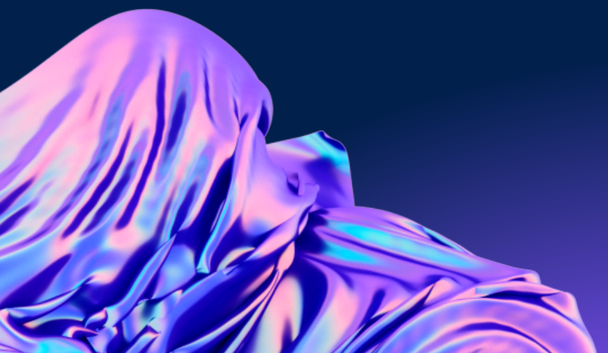
Cleaning

```python
def reclassify_certification(cert):
    if cert in ['U', 'G', 'Approved', 'Passed', 'TV-PG', 'GP']:
        return 'Suitable for all ages'
    elif cert in ['A', 'R', 'PG-13']:
        return 'Not suitable for children'
    elif cert in ['UA', 'PG', 'U/A']:
        return 'Not suitable for children'
    else:
        return None

# Apply reclassification to 'Certificate' column
df['Certificate'] = df['Certificate'].apply(reclassify_certification)


df["Certificate"].value_counts().plot(kind='bar')
plt.title('Certificate Value Counts')
plt.xlabel('Certificate')
plt.ylabel('Count')
plt.show()
```



Analysis

Reclassfication



Reclassification helps to reduce class imbalance without upscaling or downscaling.

7]:
```
print(df)
```

```
                  Certificate Runtime  IMDB_Rating  Meta_score  No_of_Votes  \
1    Not suitable for children    142          9.3        80.0      2343110
2    Not suitable for children    175          9.2       100.0      1620367
3    Not suitable for children    152          9.0        84.0      2303232
4    Not suitable for children    202          9.0        90.0      1129952
5        Suitable for all ages     96          9.0        96.0       689845
..                         ...    ...          ...         ...          ...
710  Not suitable for children    157          7.6        77.0        30144
711      Suitable for all ages    144          7.6        50.0        45338
712      Suitable for all ages     78          7.6        65.0       166409
713      Suitable for all ages     87          7.6        96.0        40351
714      Suitable for all ages    118          7.6        85.0        43374

         Gross
1     28341469
2    134966411
3    534858444
4     57300000
5      4360000
..         ...
710     696690
711    1378435
712  141843612
713   13780024
714   30500000

[714 rows x 6 columns]
```

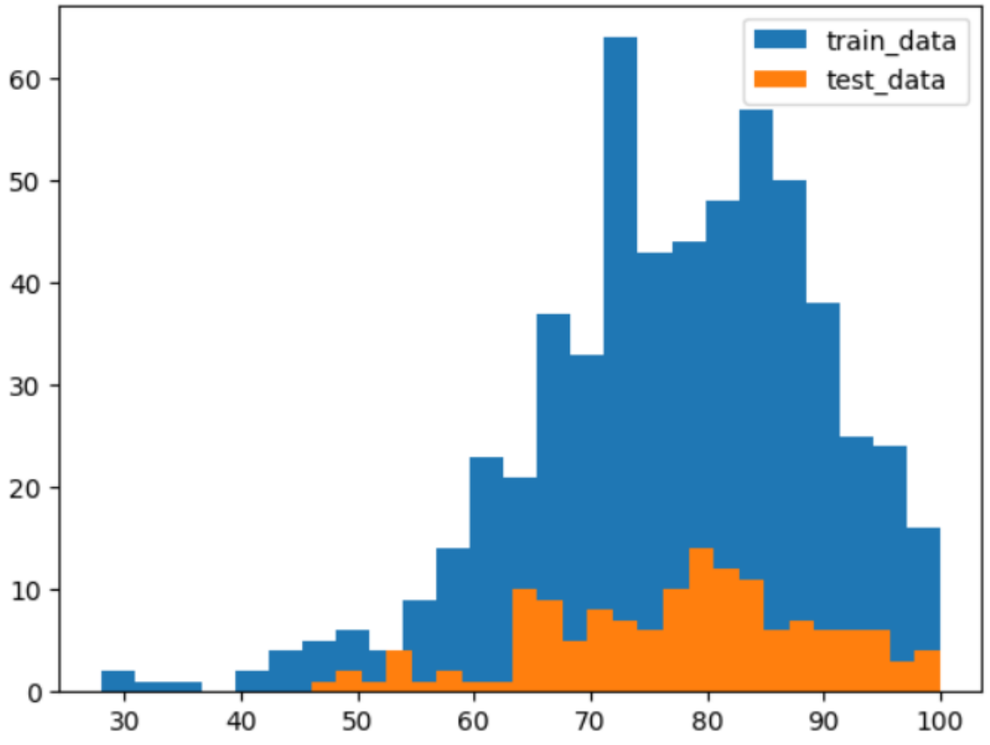
Now we split the data into train and test sets, and display the size of the train and test sets

8]:
```
train_data, test_data = train_test_split(df, test_size=0.2, random_state=42)

print(f"Training data size: {len(train_data)}")
print(f"Test data size: {len(test_data)}")
```

```
Training data size: 571
Test data size: 143
```

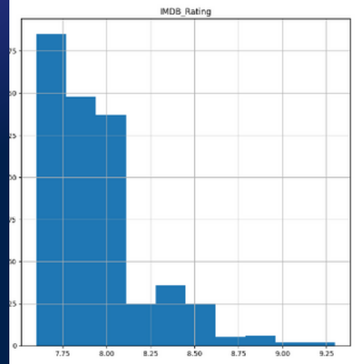
Using Histograms

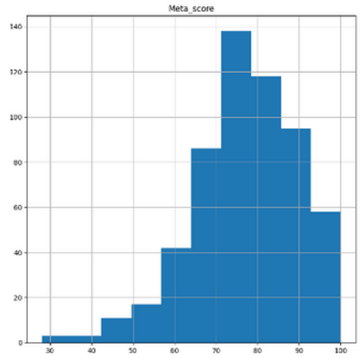
```
#distribution of METASCORE for test and train, accurate and similar
# Extract the 'metascore' column from the train and test data
train_metascore = train_data['Meta_score']
test_metascore = test_data['Meta_score']

plt.hist(train_metascore, bins=25, label="train_data")
plt.hist(test_metascore, bins=25, label="test_data")
plt.legend()
plt.show()
```

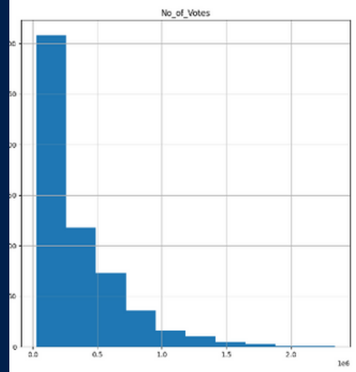


Using Histograms

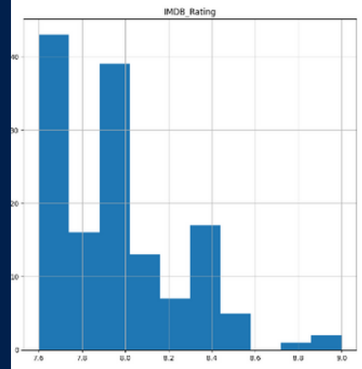
```
train_data.hist(figsize=(20,20))
plt.show()
```

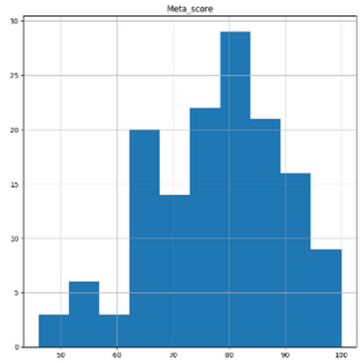


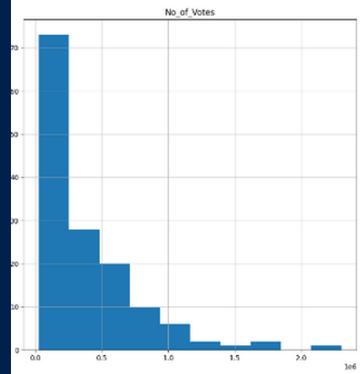




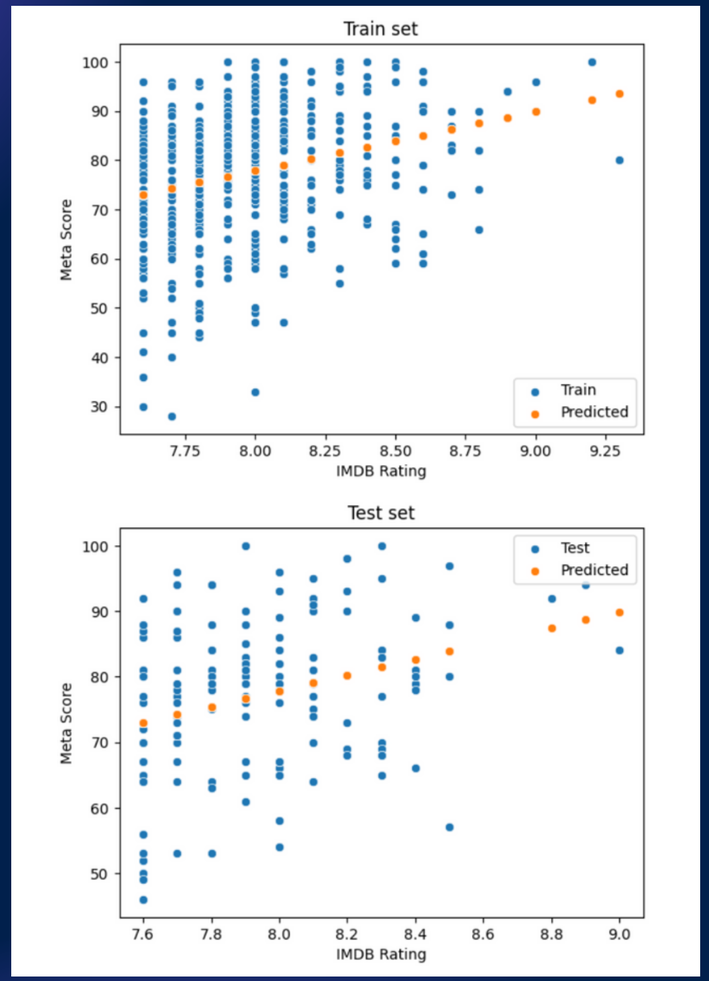

```
test_data.hist(figsize=(20,20))
plt.show()
```







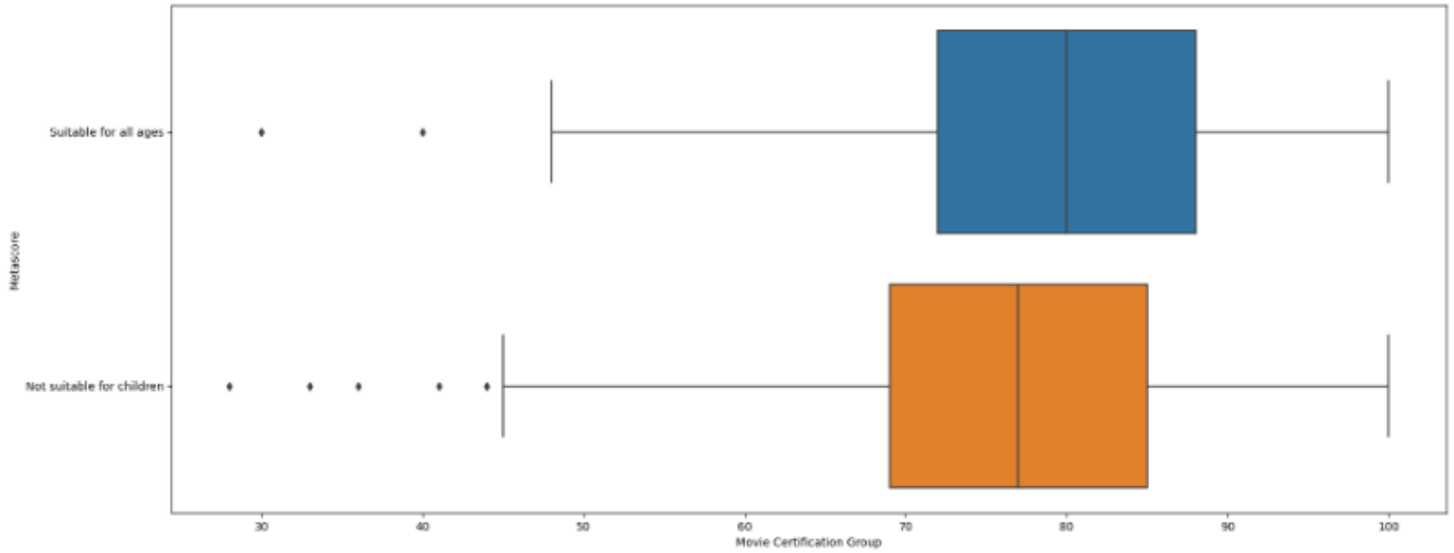
Scatter Plot



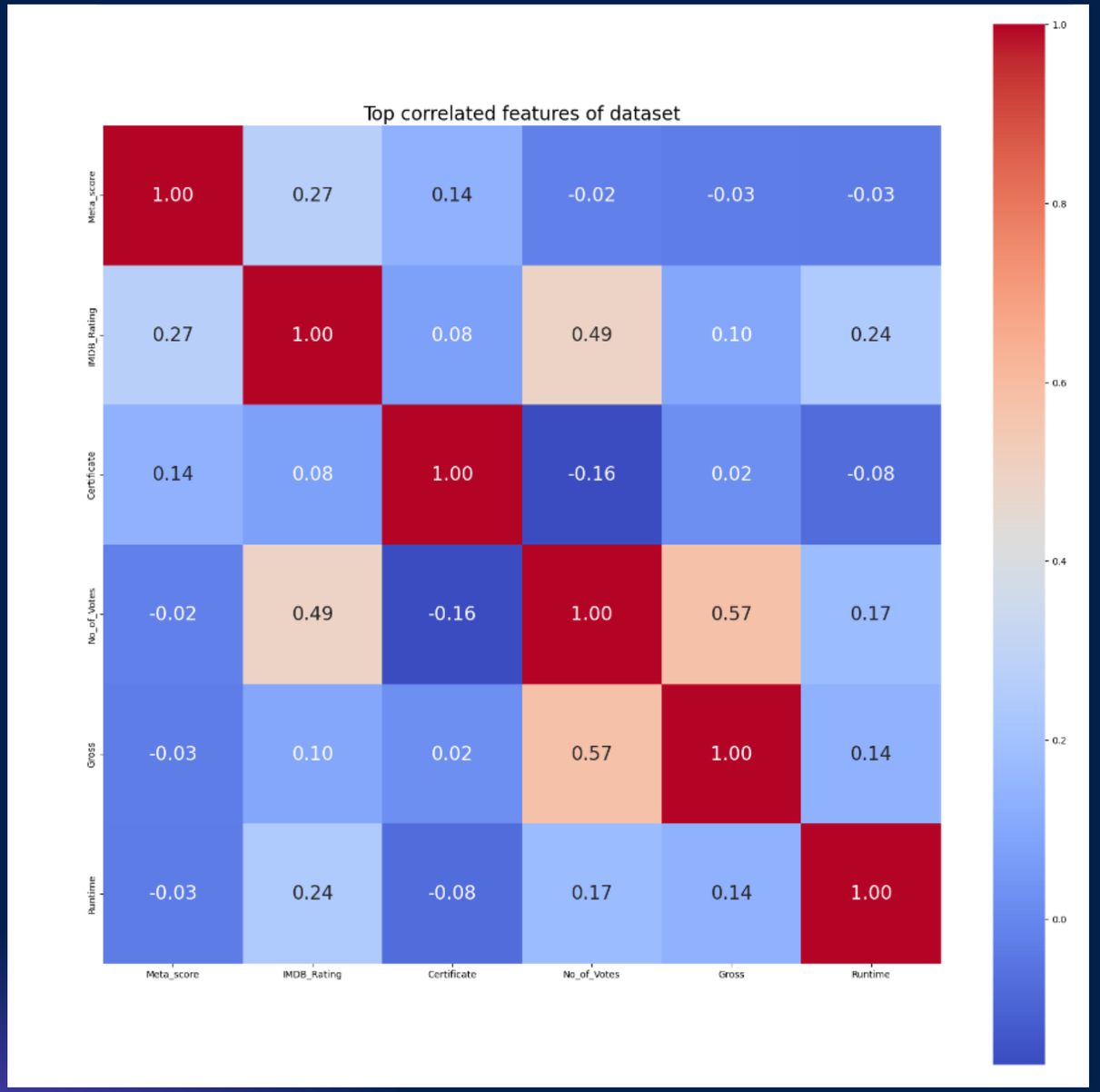

```
Goodness of fit (R-squared) for train set: 0.0791136271172117
Goodness of fit (R-squared) for test set: 0.07656892402803817
Classification accuracy for train set: 0.9667250437828371
Classification accuracy for test set: 0.9790209790209791
Confusion matrix for train set:
[[  0  19]
 [  0 552]]
Confusion matrix for test set:
[[  0   3]
 [  0 140]]
Classification report for train set:
              precision    recall  f1-score   support

       False       1.00      0.00      0.00        19
        True       0.97      1.00      0.98       552

    accuracy                           0.97       571
   macro avg       0.98      0.50      0.49       571
weighted avg       0.97      0.97      0.95       571

Classification report for test set:
              precision    recall  f1-score   support

       False       1.00      0.00      0.00         3
        True       0.98      1.00      0.99       140

    accuracy                           0.98       143
   macro avg       0.99      0.50      0.49       143
weighted avg       0.98      0.98      0.97       143
```

Box Plot

```
]:
order = ['Suitable for all ages', 'Not suitable for children']

# Create box plot using Seaborn
plt.figure(figsize=(20, 8))
sns.boxplot(data=df, y='Certificate', x='Meta_score', order=order, orient='h')
plt.xlabel('Movie Certification Group')
plt.ylabel('Metascore')
plt.show()
```



Results


Top correlated features of dataset
Meta_score
IMDB_Rating
Certificate
No_of_Votes
Gross
Runtime
1.00 0.27 0.14 -0.02 -0.03 -0.03
0.27 1.00 0.08 0.49 0.10 0.24
0.14 0.08 1.00 -0.16 0.02 -0.08
-0.02 0.49 -0.16 1.00 0.57 0.17
-0.03 0.10 0.02 0.57 1.00 0.14
-0.03 0.24 -0.08 0.17 0.14 1.00
1.0
0.8
0.6
0.4
0.2
0.0

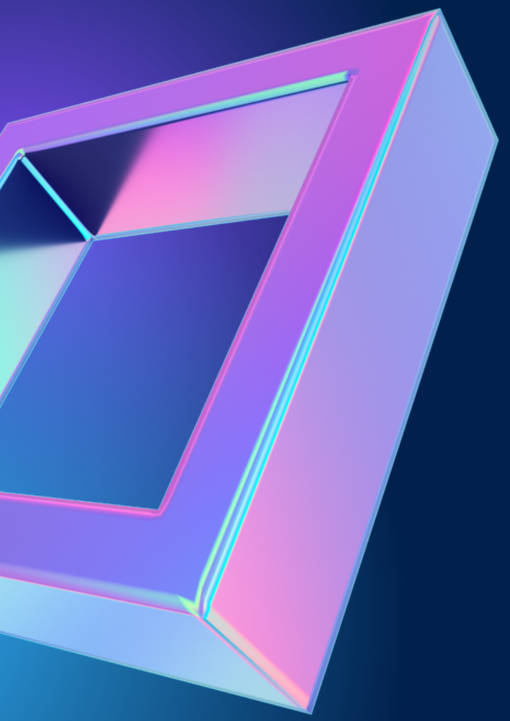
Reccomandation

IMDB rating is a reliable predictor for Meta score due to their high correlation.

Filmmakers should focus on improving the IMDB rating of their movies to achieve higher Meta scores.

Other factors such as Budget and Box Office can significantly impact a movie's success in terms of Meta score.

Filmmakers should consider these factors when creating movies to increase their chances of achieving a higher Meta score.



Thank You