# Assignment 6

## Group Members :

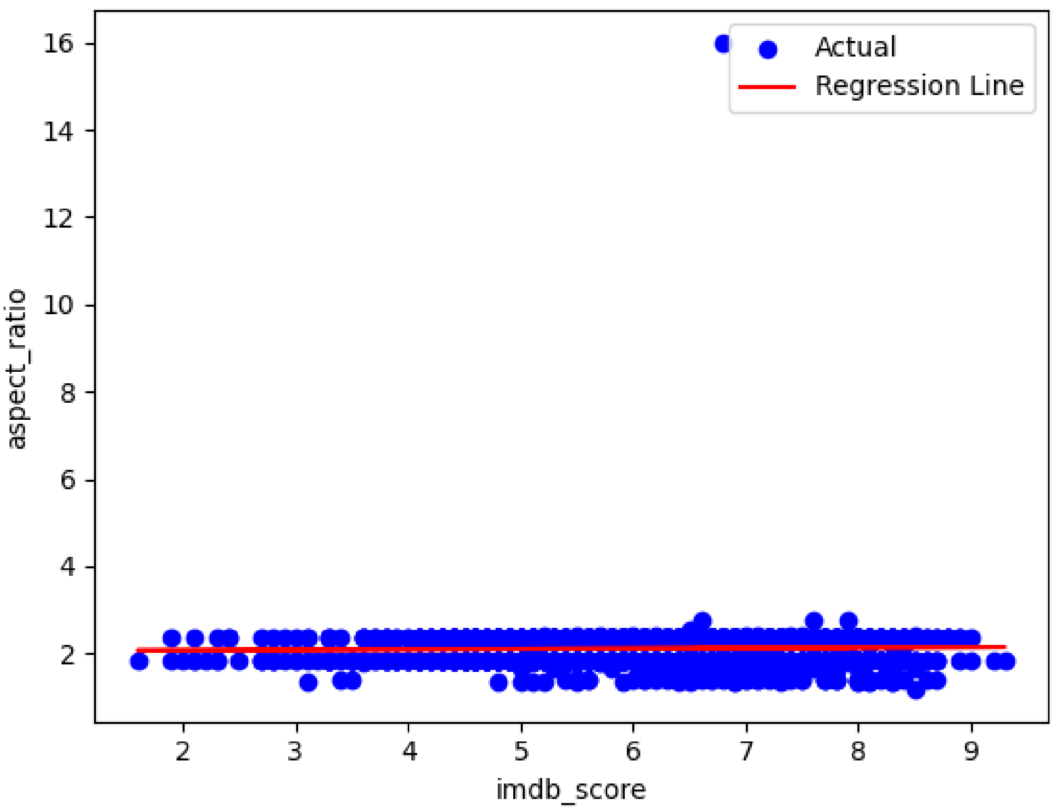
206 Snehal Bankar<br>
207 Ajay Bhapkar<br>
214 Sayali Deshmukh

```
#Linear regration

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model import LinearRegression
df1=pd.read_csv("/content/sample_data/MOVIES DATASET1.csv")
data = df1.dropna()
print(data)
# Extract the columns for linear regression
X = data['imdb_score'].values.reshape(-1, 1) # Input feature
y = data['aspect_ratio'].values # Target variable
# Create and fit the linear regression model
model = LinearRegression()
model.fit(X, y)
# Predict the target variable
y_pred = model.predict(X)
# Plot the data points and the regression line
plt.scatter(X, y, color='blue', label='Actual')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.xlabel('imdb_score')
plt.ylabel('aspect_ratio')
plt.legend()
plt.show()
```

```
color      director_name  num_critic_for_reviews  duration  \
0     Color      James Cameron                   723.0     178.0
1     Color     Gore Verbinski                   302.0     169.0
2     Color         Sam Mendes                   602.0     148.0
3     Color  Christopher Nolan                   813.0     164.0
5     Color     Andrew Stanton                   462.0     132.0
...     ...                ...                     ...       ...
```

```
5026  Color    Olivier Assayas                    81.0     110.0
5027  Color       Jafar Panahi                    64.0      90.0
5033  Color      Shane Carruth                   143.0      77.0
5035  Color   Robert Rodriguez                    56.0      81.0
5042  Color           Jon Gunn                    43.0      90.0

      director_facebook_likes  actor_3_facebook_likes       actor_2_name  \
0                         0.0                   855.0   Joel David Moore   
1                       563.0                  1000.0      Orlando Bloom   
2                         0.0                   161.0       Rory Kinnear   
3                     22000.0                 23000.0     Christian Bale   
5                       475.0                   530.0    Samantha Morton   
...                       ...                     ...                ...   
5026                    107.0                    45.0     Béatrice Dalle   
5027                    397.0                     0.0  Nargess Mamizadeh   
5033                    291.0                     8.0     David Sullivan   
5035                      0.0                     6.0    Peter Marquardt   
5042                     16.0                    16.0   Brian Herzlinger   

      actor_1_facebook_likes        gross  \
0                     1000.0  760505847.0   
1                    40000.0  309404152.0   
2                    11000.0  200074175.0   
3                    27000.0  448130642.0   
5                      640.0   73058679.0   
...                      ...          ...   
5026                   576.0     136007.0   
5027                     5.0     673780.0   
5033                   291.0     424760.0   
5035                   121.0    2040920.0   
5042                    86.0      85222.0   

                                   genres  ... num_user_for_reviews language  
\
0         Action|Adventure|Fantasy|Sci-Fi  ...               3054.0  English   
1                Action|Adventure|Fantasy  ...               1238.0  English   
2               Action|Adventure|Thriller  ...                994.0  English   
3                         Action|Thriller  ...               2701.0  English   
5                 Action|Adventure|Sci-Fi  ...                738.0  English   
...                                   ...  ...                  ...      ...   
5026                  Drama|Music|Romance  ...                 39.0   French   
5027                                Drama  ...                 26.0  Persian   
5033                Drama|Sci-Fi|Thriller  ...                371.0  English   
5035  Action|Crime|Drama|Romance|Thriller  ...                130.0  Spanish   
5042                          Documentary  ...                 84.0  English   

      country  content_rating       budget  title_year actor_2_facebook_likes  
\
0         USA           PG-13  237000000.0      2009.0                  936.0   
1         USA           PG-13  300000000.0      2007.0                 5000.0   
2          UK           PG-13  245000000.0      2015.0                  393.0   
3         USA           PG-13  250000000.0      2012.0                23000.0   
5         USA           PG-13  263700000.0      2012.0                  632.0   
...       ...             ...          ...         ...                    ...   
5026   France               R       4500.0      2004.0                  133.0   
5027     Iran       Not Rated      10000.0      2000.0                    0.0   
5033      USA           PG-13       7000.0      2004.0                   45.0   
```

```
5035      USA                 R       7000.0      1992.0                   20.0
5042      USA                PG       1100.0      2004.0                   23.0

     imdb_score  aspect_ratio movie_facebook_likes
0           7.9          1.78                33000
1           7.1          2.35                    0
2           6.8          2.35                85000
3           8.5          2.35               164000
5           6.6          2.35                24000
...         ...           ...                  ...
5026        6.9          2.35                  171
5027        7.5          1.85                  697
5033        7.0          1.85                19000
5035        6.9          1.37                    0
5042        6.6          1.85                  456

[3756 rows x 28 columns]
```



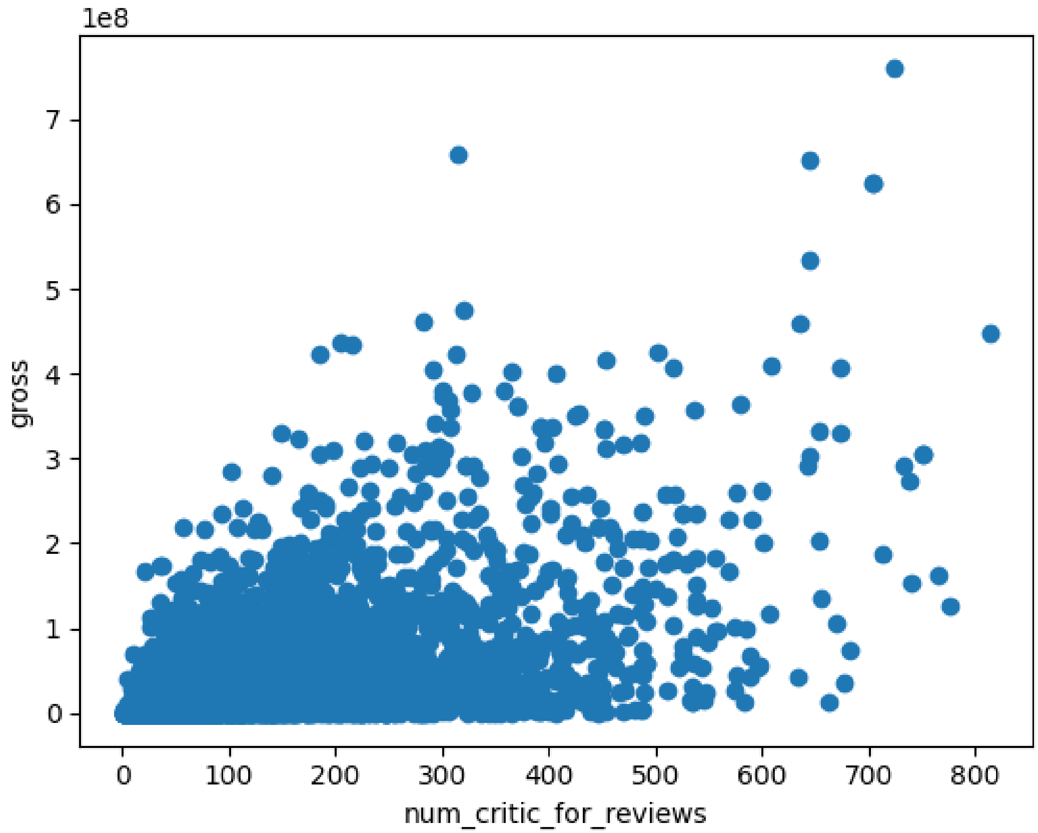
```
#Knn
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.axes as ax
from sklearn.metrics import classification_report,\
    confusion_matrix
df = pd.read_csv('/content/sample_data/MOVIES DATASET1.csv')
# Drop the missing values
df = df.dropna()
X=df['num_critic_for_reviews']
df=df.dropna()
Y=df['duration']
X=np.array(df['num_critic_for_reviews']).reshape(-1,1)
Y=np.array(df['duration']).reshape(-1,1)
X_train, X_test,y_train, y_test = train_test_split(X,Y,test_size=0.30)

from sklearn.metrics import classification_report,\
    confusion_matrix

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)

# Predictions and Evaluations
# Let's evaluate our KNN model !
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

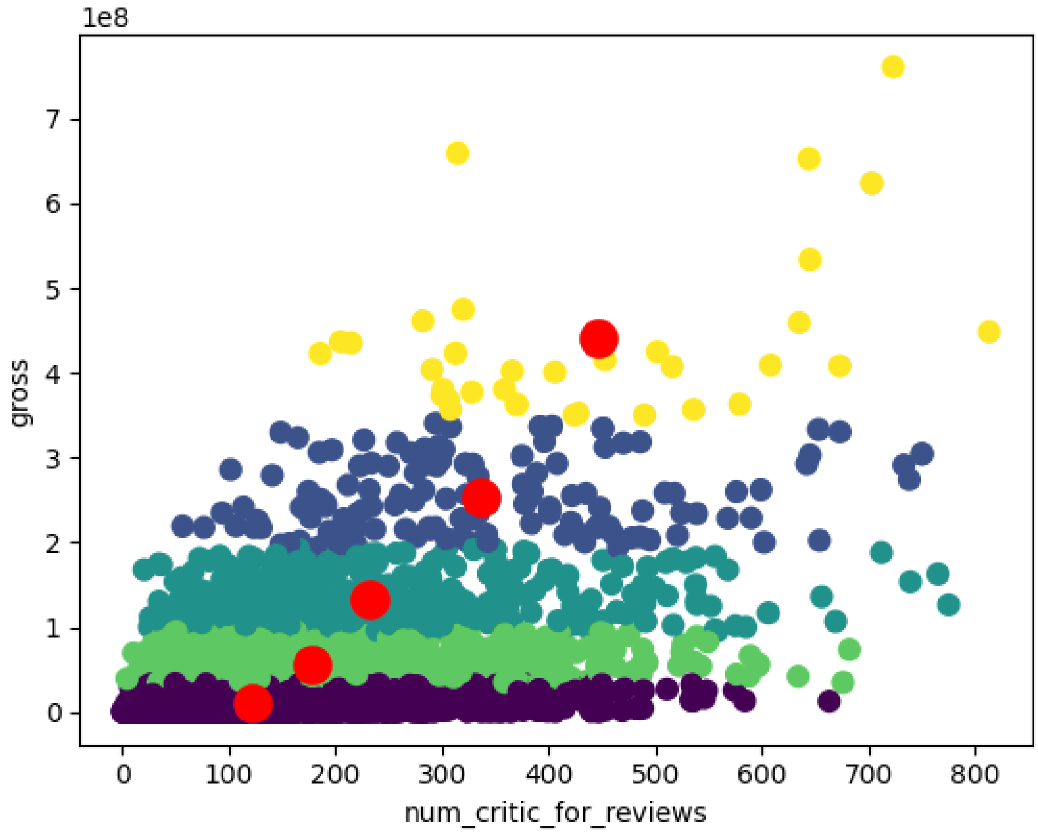
```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
              precision    recall  f1-score   support

        45.0       0.00      0.00      0.00         0
        63.0       0.00      0.00      0.00         1
        66.0       0.00      0.00      0.00         1
        69.0       0.00      0.00      0.00         1
        72.0       0.00      0.00      0.00         1
        74.0       0.00      0.00      0.00         1
        75.0       0.00      0.00      0.00         3
        76.0       0.00      0.00      0.00         1
        77.0       0.00      0.00      0.00         2
        78.0       0.00      0.00      0.00         2
        79.0       0.00      0.00      0.00         0
        80.0       0.00      0.00      0.00         6
        81.0       0.00      0.00      0.00         4
        82.0       0.00      0.00      0.00         9
        83.0       0.00      0.00      0.00         7
        84.0       0.00      0.00      0.00         8
```

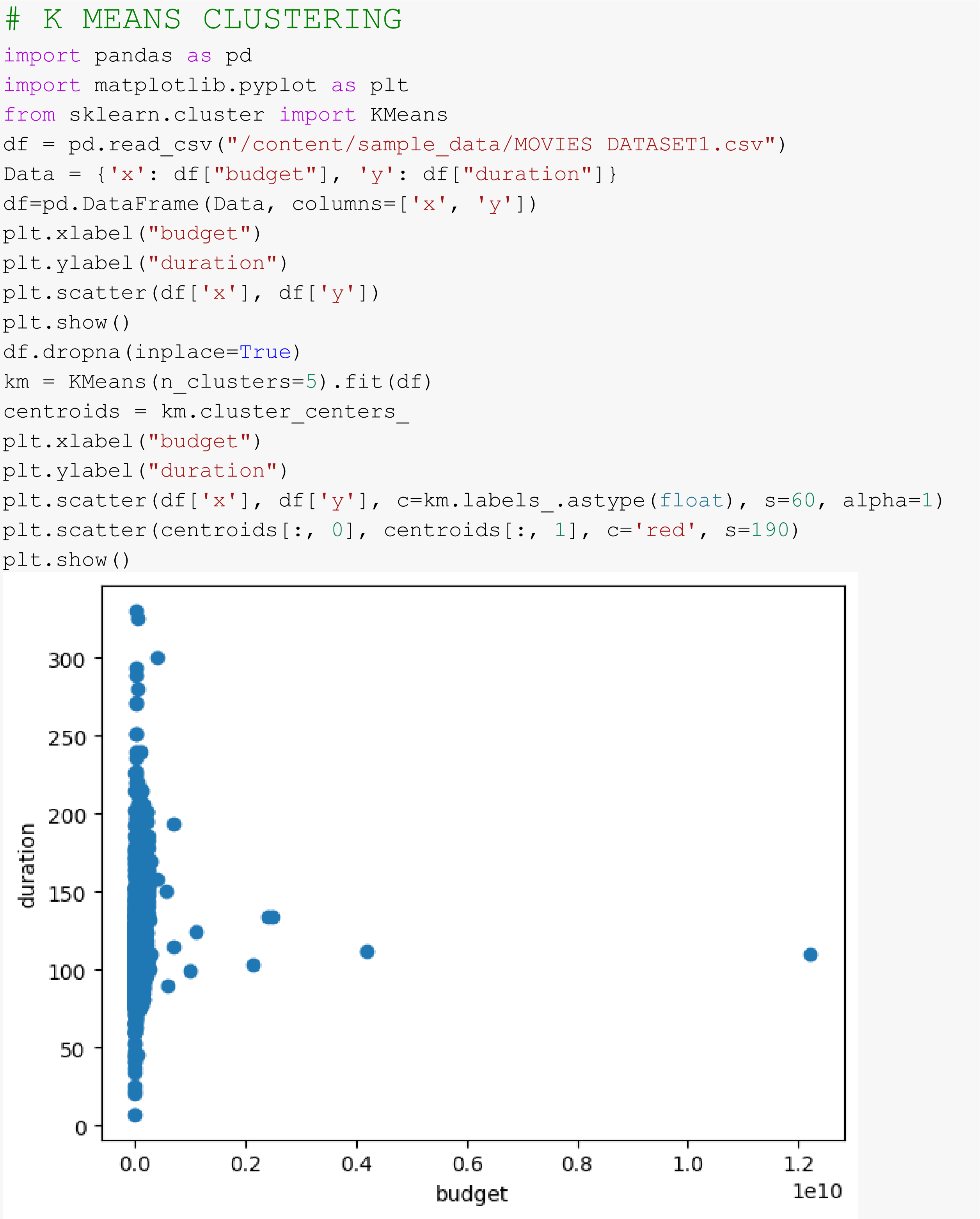
|  |  |  |  |  |
|---|---|---|---|---|
| 85.0 | 0.00 | 0.00 | 0.00 | 13 |
| 86.0 | 0.00 | 0.00 | 0.00 | 8 |
| 87.0 | 0.00 | 0.00 | 0.00 | 12 |
| 88.0 | 0.00 | 0.00 | 0.00 | 21 |
| 89.0 | 0.07 | 0.06 | 0.06 | 17 |
| 90.0 | 0.05 | 0.03 | 0.04 | 29 |
| 91.0 | 0.00 | 0.00 | 0.00 | 27 |
| 92.0 | 0.00 | 0.00 | 0.00 | 21 |
| 93.0 | 0.03 | 0.03 | 0.03 | 29 |
| 94.0 | 0.05 | 0.04 | 0.05 | 23 |
| 95.0 | 0.05 | 0.06 | 0.05 | 32 |
| 96.0 | 0.08 | 0.07 | 0.07 | 30 |
| 97.0 | 0.00 | 0.00 | 0.00 | 33 |
| 98.0 | 0.03 | 0.04 | 0.03 | 28 |
| 99.0 | 0.04 | 0.04 | 0.04 | 23 |
| 100.0 | 0.05 | 0.03 | 0.04 | 35 |
| 101.0 | 0.08 | 0.11 | 0.09 | 36 |
| 102.0 | 0.00 | 0.00 | 0.00 | 23 |
| 103.0 | 0.00 | 0.00 | 0.00 | 23 |
| 104.0 | 0.04 | 0.05 | 0.04 | 22 |
| 105.0 | 0.00 | 0.00 | 0.00 | 19 |
| 106.0 | 0.04 | 0.04 | 0.04 | 24 |
| 107.0 | 0.05 | 0.08 | 0.06 | 24 |
| 108.0 | 0.06 | 0.03 | 0.04 | 29 |
| 109.0 | 0.00 | 0.00 | 0.00 | 25 |
| 110.0 | 0.04 | 0.04 | 0.04 | 25 |
| 111.0 | 0.00 | 0.00 | 0.00 | 18 |
| 112.0 | 0.06 | 0.04 | 0.04 | 27 |
| 113.0 | 0.05 | 0.05 | 0.05 | 21 |
| 114.0 | 0.03 | 0.06 | 0.04 | 18 |
| 115.0 | 0.05 | 0.06 | 0.05 | 18 |
| 116.0 | 0.05 | 0.07 | 0.06 | 14 |
| 117.0 | 0.00 | 0.00 | 0.00 | 14 |
| 118.0 | 0.08 | 0.09 | 0.09 | 22 |
| 119.0 | 0.05 | 0.07 | 0.06 | 14 |
| 120.0 | 0.04 | 0.08 | 0.05 | 13 |
| 121.0 | 0.00 | 0.00 | 0.00 | 16 |
| 122.0 | 0.06 | 0.08 | 0.07 | 13 |
| 123.0 | 0.00 | 0.00 | 0.00 | 22 |
| 124.0 | 0.00 | 0.00 | 0.00 | 18 |
| 125.0 | 0.00 | 0.00 | 0.00 | 12 |
| 126.0 | 0.00 | 0.00 | 0.00 | 2 |
| 127.0 | 0.00 | 0.00 | 0.00 | 10 |
| 128.0 | 0.00 | 0.00 | 0.00 | 12 |
| 129.0 | 0.00 | 0.00 | 0.00 | 13 |
| 130.0 | 0.20 | 0.11 | 0.14 | 9 |
| 131.0 | 0.00 | 0.00 | 0.00 | 11 |
| 132.0 | 0.00 | 0.00 | 0.00 | 4 |
| 133.0 | 0.00 | 0.00 | 0.00 | 6 |
| 134.0 | 0.00 | 0.00 | 0.00 | 10 |
| 135.0 | 0.00 | 0.00 | 0.00 | 10 |
| 136.0 | 0.00 | 0.00 | 0.00 | 9 |
| 137.0 | 0.00 | 0.00 | 0.00 | 4 |
| 138.0 | 0.00 | 0.00 | 0.00 | 1 |

```
       139.0       0.00      0.00      0.00         5
       140.0       0.00      0.00      0.00         4
       141.0       0.00      0.00      0.00         4
       142.0       0.00      0.00      0.00         6
       143.0       0.25      0.33      0.29         3
       144.0       0.33      0.25      0.29         4
       145.0       0.00      0.00      0.00         2
       146.0       0.00      0.00      0.00         4
       147.0       0.00      0.00      0.00         1
       148.0       0.00      0.00      0.00         2
       150.0       0.00      0.00      0.00         8
       152.0       0.00      0.00      0.00         0
       153.0       0.00      0.00      0.00         3
       154.0       0.00      0.00      0.00         5
       156.0       0.00      0.00      0.00         0
       158.0       0.00      0.00      0.00         3
       160.0       0.00      0.00      0.00         0
       161.0       0.00      0.00      0.00         2
       162.0       0.00      0.00      0.00         0
       163.0       0.00      0.00      0.00         0
       164.0       0.00      0.00      0.00         1
       165.0       0.00      0.00      0.00         1
       167.0       0.00      0.00      0.00         0
       169.0       0.00      0.00      0.00         3
       170.0       0.00      0.00      0.00         1
       171.0       0.00      0.00      0.00         2
       172.0       0.00      0.00      0.00         3
       173.0       0.00      0.00      0.00         0
       174.0       0.00      0.00      0.00         0
       176.0       0.00      0.00      0.00         2
       178.0       0.00      0.00      0.00         5
       183.0       0.00      0.00      0.00         1
       184.0       0.00      0.00      0.00         1
       186.0       0.00      0.00      0.00         1
       188.0       0.00      0.00      0.00         1
       189.0       0.00      0.00      0.00         0
       193.0       0.00      0.00      0.00         1
       194.0       0.00      0.00      0.00         0
       195.0       0.00      0.00      0.00         0
       196.0       0.00      0.00      0.00         1
       197.0       0.00      0.00      0.00         1
       201.0       0.25      1.00      0.40         1
       202.0       0.00      0.00      0.00         1
       212.0       0.00      0.00      0.00         2
       216.0       0.00      0.00      0.00         0
       226.0       0.00      0.00      0.00         1
       227.0       0.00      0.00      0.00         0
       240.0       0.00      0.00      0.00         1
       280.0       0.00      0.00      0.00         0
       293.0       0.00      0.00      0.00         1

    accuracy       0.67      0.67      0.67      1127
   macro avg       0.02      0.03      0.02      1127
weighted avg       0.03      0.03      0.03      1127
```

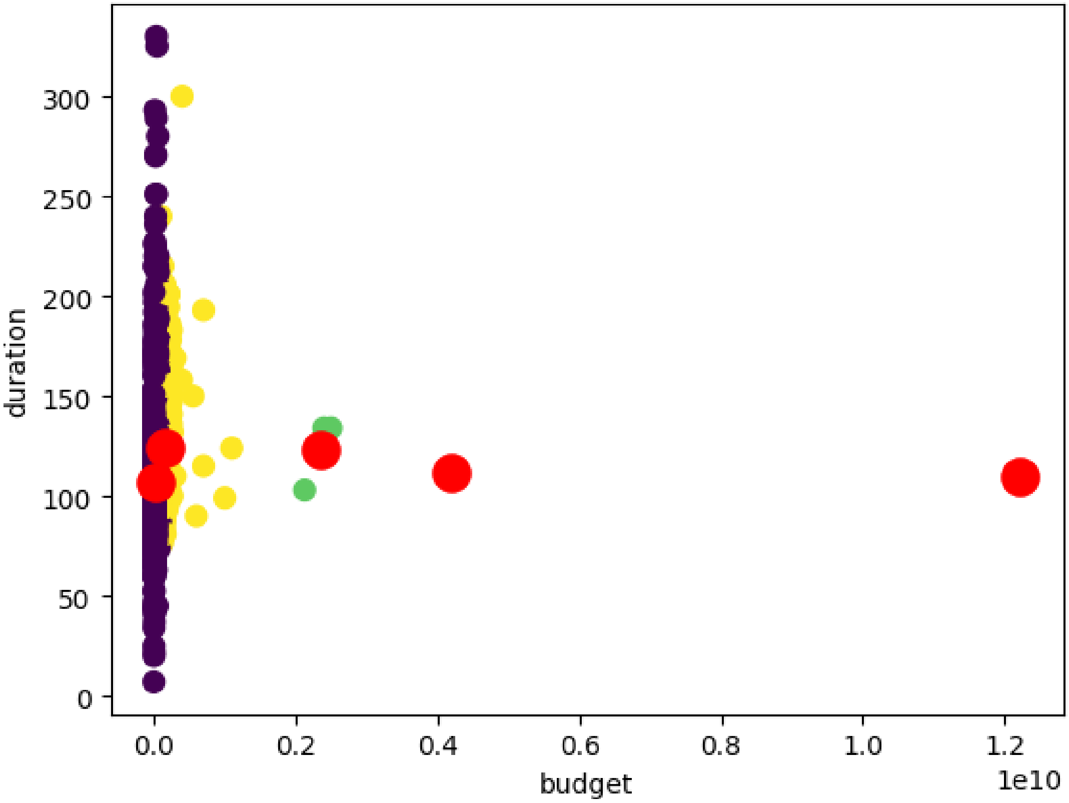
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

df = pd.read_csv("/content/sample_data/MOVIES DATASET1.csv")
Data = {'x': df["num_critic_for_reviews"], 'y': df["gross"]}
df=pd.DataFrame(Data, columns=['x', 'y'])

plt.xlabel("num_critic_for_reviews")

plt.ylabel("gross")

plt.scatter(df['x'], df['y'])
plt.show()
df.dropna(inplace=True)

km = KMeans(n_clusters=5).fit(df)
centroids = km.cluster_centers_

plt.xlabel("num_critic_for_reviews")
plt.ylabel("gross")
plt.scatter(df['x'], df['y'], c=km.labels_.astype(float), s=60, alpha=1)
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=190)
plt.show()
```


1e8
gross
0
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700
800
num_critic_for_reviews


1e8
gross
0
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700
800
num_critic_for_reviews



```
# K MEANS CLUSTERING
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
df = pd.read_csv("/content/sample_data/MOVIES DATASET1.csv")
Data = {'x': df["budget"], 'y': df["duration"]}
df=pd.DataFrame(Data, columns=['x', 'y'])
plt.xlabel("budget")
plt.ylabel("duration")
plt.scatter(df['x'], df['y'])
plt.show()
df.dropna(inplace=True)
km = KMeans(n_clusters=5).fit(df)
centroids = km.cluster_centers_
plt.xlabel("budget")
plt.ylabel("duration")
plt.scatter(df['x'], df['y'], c=km.labels_.astype(float), s=60, alpha=1)
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=190)
plt.show()
```


duration
0
50
100
150
200
250
300
0.0
0.2
0.4
0.6
0.8
1.0
1.2
budget
1e10