**PREDICTION IMDB SCORES**

**project title:** predicted for IMDb scores

**phase 3**: Development

**part 1:**

**Data Preprocessing:**

* Data preprocessing is a crucial step within the statistics analysis and gadget gaining knowledge of pipeline.
* It includes a sequence of strategies and operations finished on uncooked statistics to clean, organize, and transform it right into a layout that is suitable for analysis or device mastering version schooling.
* Data preprocessing goals to enhance the first-class of the records, making it greater reliable and conducive to generating accurate consequences.

Here are some common tasks and techniques involved in data preprocessing:

**Data Cleaning:**

* Handling missing values: Deciding how to deal with missing data, whether by imputing values or removing incomplete records.
* Outlier detection and treatment: Identifying and handling data points that significantly deviate from the norm.

**Noise reduction:**

* Smoothing noisy data through techniques like filtering.

**Data Transformation:**

* **Data normalization:** Scaling numerical features to a standard range (e.g., between 0 and 1) to ensure that they have similar influence in the analysis.
* **Encoding categorical variables:** Converting categorical data into numerical format, such as one-hot encoding or label encoding.
* **Feature engineering:** Creating new features or modifying existing ones to capture more meaningful information from the data.
* **Dimensionality reduction:** Reducing the number of features while retaining essential information, using methods like Principal Component Analysis (PCA).

**Data Integration:**

* **Merging or joining datasets:** Combining data from multiple sources into a single dataset for analysis.

**Aggregation:** Summarizing data at a higher level of granularity, such as aggregating daily sales into monthly totals.

**Data Reduction:**

* **Sampling:** Reducing the size of a large dataset by randomly selecting a representative subset.
* **Binning:** Grouping continuous data into discrete bins to simplify analysis.
* **Filtering:** Selecting a subset of data based on specific criteria.

**Data Standardization:**

* Ensuring that data follows a consistent format and structure.
* Date and time format conversion: Converting date and time data into a uniform format.
* Currency conversion: Converting monetary values into a common currency.

**Data Scaling:**

* Scaling numerical data to a common range to prevent some features from dominating the analysis.

Data preprocessing is an iterative process that may involve several of these steps in various orders, depending on the specific dataset and the analysis goals. Proper data preprocessing is essential for improving the accuracy and effectiveness of machine learning models, as well as for making data more accessible for traditional statistical analysis.

Here is the data preprocessing codes along with the output of the given dataset:

**Importing the libraries:**

Import three basic libraries which are very common in machine learning and will be used every time you train a model

* **NumPy:** it is a library that allows us to work with arrays and as most machine learning models work on arrays NumPy makes it easier
* **matplotlib:** this library helps in plotting graphs and charts, which are very useful while showing the result of your model
* **Pandas:** pandas allowsus to import our dataset and also creates a matrix of features containing the dependent and independent variable.

**Code:**

*#Import libraries*

import numPy as np

import pandas as pd

import matplotlib.pyplo

t as plot

import seaborn as sns

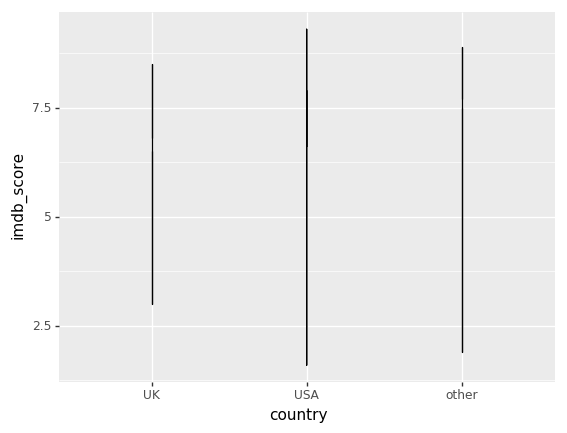
from plotnine import \*

**1.1 Background**

This dataset contains the information about the movies . For a movie to be commercial success , it depends on various factors like director, actors ,critic reviews and viewers reaction. Imdb score is one of the important factor to measure the movie's success.

**1.2 Description of dataset attributes**

Please find the details for the datset attributes:-

1. Color :- Movie is black or coloured
2. Director\_name:- Name of the movie director
3. num\_critic\_for\_reviews :- No of critics for the movie
4. duration:- movie duration in minutes
5. director\_facebook\_likes:-Number of likes for the Director on his Facebook Page
6. actor\_3\_facebook\_likes:- No of likes for the actor 3 on his/her facebook Page
7. actor2\_name:- name of the actor 2
8. actor\_1\_facebook\_likes:- No of likes for the actor 1 on his/her facebook Page
9. gross:- Gross earnings of the movie in Dollars
10. genres:- Film categorization like ‘Animation’, ‘Comedy’, ‘Romance’, ‘Horror’, ‘Sci-Fi’, ‘Action’, ‘Family’
11. actor\_1\_name:- Name of the actor 1
12. movie\_title:-Title of the movie
13. num\_voted\_users:-No of people who voted for the movie
14. cast\_total\_facebook\_likes:- Total facebook like for the movie
15. actor\_3\_name:- Name of the actor 3
16. facenumber\_in\_poster:- No of actors who featured in the movie poster
17. plot\_keywords:-Keywords describing the movie plots
18. movie\_imdb\_link:-Link of the movie link
19. num\_user\_for\_reviews:- Number of users who gave a review
20. language:- Language of the movie
21. country:- Country where movie is produced
22. content\_rating:- Content rating of the movie
23. budget:- Budget of the movie in Dollars
24. title\_year:- The year in which the movie is released
25. actor\_2\_facebook\_likes:- facebook likes for the actor 2
26. imdb\_score:- IMDB score of the movie
27. aspect\_ratio :- Aspect ratio the movie was made in
28. movie\_facebook\_likes:- Total no of facebook likes for the movie
29. **1.3 Case Study**
30. The dataset here gives the massive information about the movies and their IMDB scores respectively. We are going to analyze each and every factors which can influence the imdb ratings so that we can predict better results.The movie with the higher imdb score is more successful as compared to the movies with low imdb score
31. 

**2. Data Preprocessing**

*#Reading the Data*

movie\_df=pd.read\_csv("/kaggle/input/imdb-5000-movie-dataset/movie\_metadata.csv")

*#Displaying the first 10 records*

movie\_df.head(10)

*#Shape of the dataset (no of rows and no of columns)*

movie\_df.shape

Out[4]:

(5043, 28)

*#Displaying the data type of the dataset attributes*

movie\_df.dtypes

Out[5]:

color object

director\_name object

num\_critic\_for\_reviews float64

duration float64

director\_facebook\_likes float64

actor\_3\_facebook\_likes float64

actor\_2\_name object

actor\_1\_facebook\_likes float64

gross float64

genres object

actor\_1\_name object

movie\_title object

num\_voted\_users int64

cast\_total\_facebook\_likes int64

actor\_3\_name object

facenumber\_in\_poster float64

plot\_keywords object

movie\_imdb\_link object

num\_user\_for\_reviews float64

language object

country object

content\_rating object

budget float64

title\_year float64

actor\_2\_facebook\_likes float64

imdb\_score float64

aspect\_ratio float64

movie\_facebook\_likes int64

dtype: object

\*\*We can say we have the datset divided into categorical and numeric columns "

**Categorical Columns**

Color,Director name, actor name,genres,movie\_title,language,country,content\_rating.

**Numerical Columns**

num\_critic\_for\_reviews,duration,director\_facebook\_likes ,actor\_3\_facebook\_likes,actor\_1\_facebook\_likes ,gross,num\_voted\_users,cast\_total\_facebook\_likes,facenumber\_in\_poster,num\_user\_for\_reviews ,budget,title\_year, actor\_2\_facebook\_likes ,imdb\_score,aspect\_ratio,movie\_facebook\_like

Index(['director\_name', 'num\_critic\_for\_reviews', 'duration',

'director\_facebook\_likes', 'actor\_3\_facebook\_likes', 'actor\_2\_name',

'actor\_1\_facebook\_likes', 'gross', 'genres', 'actor\_1\_name',

'movie\_title', 'num\_voted\_users', 'cast\_total\_facebook\_likes',

'actor\_3\_name', 'facenumber\_in\_poster', 'plot\_keywords',

'num\_user\_for\_reviews', 'language', 'country', 'content\_rating',

'budget', 'title\_year', 'actor\_2\_facebook\_likes', 'imdb\_score',

'aspect\_ratio', 'movie\_facebook\_likes'],

dtype='object')

director\_name True

num\_critic\_for\_reviews True

duration True

director\_facebook\_likes True

actor\_3\_facebook\_likes True

actor\_2\_name True

actor\_1\_facebook\_likes True

gross True

genres False

actor\_1\_name True

movie\_title False

num\_voted\_users False

cast\_total\_facebook\_likes False

actor\_3\_name True

facenumber\_in\_poster True

plot\_keywords True

num\_user\_for\_reviews True

language True

country True

content\_rating True

budget True

title\_year True

actor\_2\_facebook\_likes True

imdb\_score False

aspect\_ratio True

movie\_facebook\_likes False

dtype: bool

movie\_df.isna().sum()

director\_name 104

num\_critic\_for\_reviews 50

duration 15

director\_facebook\_likes 104

actor\_3\_facebook\_likes 23

actor\_2\_name 13

actor\_1\_facebook\_likes 7

gross 884

genres 0

actor\_1\_name 7

movie\_title 0

num\_voted\_users 0

cast\_total\_facebook\_likes 0

actor\_3\_name 23

facenumber\_in\_poster 13

plot\_keywords 153

num\_user\_for\_reviews 21

language 12

country 5

content\_rating 303

budget 492

title\_year 108

actor\_2\_facebook\_likes 13

imdb\_score 0

aspect\_ratio 329

movie\_facebook\_likes 0

dtype: int64

movie\_df.dropna(axis=0,subset=['director\_name', 'num\_critic\_for\_reviews','duration','director\_facebook\_likes','actor\_3\_facebook\_likes','actor\_2\_name','actor\_1\_facebook\_likes','actor\_1\_name','actor\_3\_name','facenumber\_in\_poster','num\_user\_for\_reviews','language','country','actor\_2\_facebook\_likes','plot\_keywords'],inplace=True)

movie\_df.shape

(4737, 26)

**We lost only 6% of the data which is acceptable**

*#Replacing the content rating with Value R as it has highest frequency*

movie\_df["content\_rating"].fillna("R", inplace = True)

In [15]:

*#Replacing the aspect\_ratio with the median of the value as the graph is right skewed*

movie\_df["aspect\_ratio"].fillna(movie\_df["aspect\_ratio"].median(),inplace=True)

In [16]:

linkcode

*#We need to replace the value in budget with the median of the value*

movie\_df["budget"].fillna(movie\_df["budget"].median(),inplace=True)

*# We need to replace the value in gross with the median of the value*

movie\_df['gross'].fillna(movie\_df['gross'].median(),inplace=True)

In [18]:

*# Recheck that all the null values are removed*

movie\_df.isna().sum()

Out[18]:

director\_name 0

num\_critic\_for\_reviews 0

director\_facebook\_likes 0

actor\_3\_facebook\_likes 0

actor\_2\_name 0

actor\_1\_facebook\_likes 0

gross 0

genres 0

actor\_1\_name 0

movie\_title 0

num\_voted\_users 0

cast\_total\_facebook\_likes 0

actor\_3\_name 0

facenumber\_in\_poster 0

plot\_keywords 0

num\_user\_for\_reviews 0

language 0

country 0

content\_rating 0

budget 0

title\_year 0

actor\_2\_facebook\_likes 0

imdb\_score 0

aspect\_ratio 0

movie\_facebook\_likes 0

dtype: int64

*#Removing the duplicate values in the datset*

movie\_df.drop\_duplicates(inplace=True)

movie\_df.shape

*Count of the language values*

movie\_df["language"].value\_counts()

English 4405

French 69

Spanish 35

Hindi 25

Mandarin 24

German 18

Japanese 16

Russian 11

Italian 10

Cantonese 10

Portuguese 8

Korean 8

Danish 5

Norwegian 4

Swedish 4

Hebrew 4

Dutch 4

Persian 4

Arabic 3

Thai 3

Indonesian 2

None 2

Aboriginal 2

Dari 2

Zulu 2

Hungarian 1

Mongolian 1

Greek 1

Romanian 1

Bosnian 1

Telugu 1

Maya 1

Polish 1

Filipino 1

Czech 1

Dzongkha 1

Kazakh 1

Vietnamese 1

Icelandic 1

Aramaic 1

Name: language, dtype: int64

*# Graphical presentaion*

plt.figure(figsize=(40,10))

sns.countplot(movie\_df["language"])

plt.show()



*#Most of the values for the languages is english we can drop the english column*

movie\_df.drop('language',axis=1,inplace=True)

In [23]:

linkcode

*#Creating a new column to check the net profit made by the company (Gross-Budget)*

movie\_df["Profit"]=movie\_df['budget'].sub(movie\_df['gross'], axis = 0)

value\_counts=movie\_df["country"].value\_counts()

print(value\_counts)

USA 3568

UK 420

France 149

Canada 107

Germany 96

Australia 53

Spain 32

India 27

China 24

Japan 21

Italy 20

Hong Kong 16

New Zealand 14

South Korea 12

Ireland 11

Denmark 11

Russia 11

Mexico 11

South Africa 8

Brazil 8

Norway 7

Netherlands 5

Sweden 5

Thailand 4

Iran 4

Argentina 4

Czech Republic 3

Switzerland 3

Belgium 3

Israel 3

West Germany 3

Poland 2

Taiwan 2

Iceland 2

Romania 2

Hungary 2

Greece 2

Soviet Union 1

Slovakia 1

Finland 1

Official site 1

Turkey 1

Peru 1

Libya 1

Afghanistan 1

Cambodia 1

Indonesia 1

Nigeria 1

Kyrgyzstan 1

Colombia 1

New Line 1

Philippines 1

Bahamas 1

Bulgaria 1

Georgia 1

Aruba 1

Chile 1

Name: country, dtype: int64

**We can see most of the movies are from USA ,UK and the rest of the countries**

In [26]:

*##get top 2 values of index*

*##get top 2 values of index*

vals = value\_counts[:2].index

print (vals)

movie\_df['country'] = movie\_df.country.where(movie\_df.country.isin(vals), 'other')

Index(['USA', 'UK'], dtype='object')

In [27]:

*#Successfully divided the country into three catogories*

movie\_df["country"].value\_counts()

Out[27]:

USA 3568

other 707

UK 420

Name: country, dtype: int64

In [28]:

linkcode

movie\_df.head(10)

/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord\_cartesian.py:31: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

self.limits = Bunch(xlim=xlim, ylim=ylim)

/opt/conda/lib/python3.6/copy.py:274: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

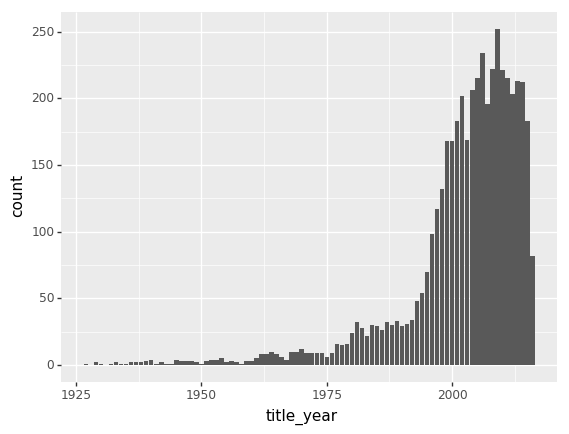
y = func(\*args)

/opt/conda/lib/python3.6/site-packages/plotnine/facets/facet.py:151: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

scales = Bunch()

/opt/conda/lib/python3.6/site-packages/plotnine/facets/layout.py:147: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

return Bunch(x=xsc, y=ysc)



*#Relationship between the imdb score and the profit made by the movie*

ggplot(aes(x='imdb\_score', y='Profit'), data=movie\_df) +\

geom\_line() +\

stat\_smooth(colour='blue', span=1)

/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord\_cartesian.py:31: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

self.limits = Bunch(xlim=xlim, ylim=ylim)

/opt/conda/lib/python3.6/copy.py:274: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

y = func(\*args)

/opt/conda/lib/python3.6/site-packages/plotnine/facets/facet.py:151: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

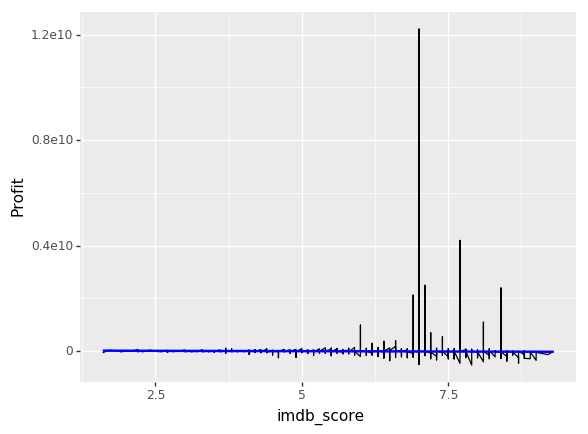
scales = Bunch()

/opt/conda/lib/python3.6/site-packages/plotnine/facets/layout.py:147: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

return Bunch(x=xsc, y=ysc)

/opt/conda/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

return ptp(axis=axis, out=out, \*\*kwargs)



*#Finding the corelation between imdb\_rating with respect to no of facebook likes*

(ggplot(movie\_df)

+ aes(x='imdb\_score', y='movie\_facebook\_likes')

+ geom\_line()

+ labs(title='IMDB\_Score vs. Facebook like for Movies', x='IMDB scores', y='Facebook Likes for movies')

)

/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord\_cartesian.py:31: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

self.limits = Bunch(xlim=xlim, ylim=ylim)

/opt/conda/lib/python3.6/copy.py:274: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

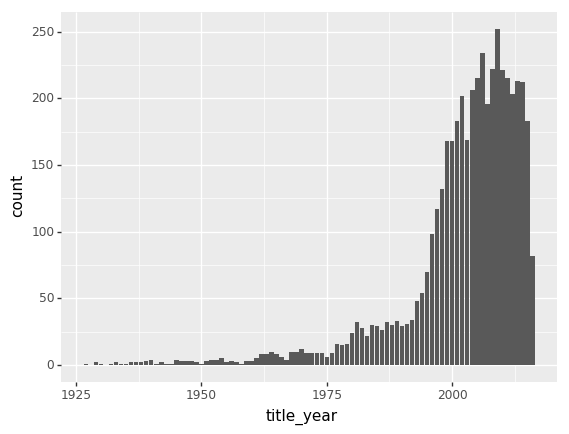
y = func(\*args)

/opt/conda/lib/python3.6/site-packages/plotnine/facets/facet.py:151: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

scales = Bunch()

/opt/conda/lib/python3.6/site-packages/plotnine/facets/layout.py:147: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

return Bunch(x=xsc, y=ysc)



*#Top 20 movies based on the profit they made*

plt.figure(figsize=(10,8))

movie\_df= movie\_df.sort\_values(by ='Profit' , ascending=False)

movie\_df\_new=movie\_df.head(20)

ax=sns.pointplot(movie\_df\_new['Profit'], movie\_df\_new['budget'], hue=movie\_df\_new['movie\_title'])

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=40, ha="right")

plt.tight\_layout()

plt.show()

*#Top 20 movies based on the profit they made*

plt.figure(figsize=(10,8))

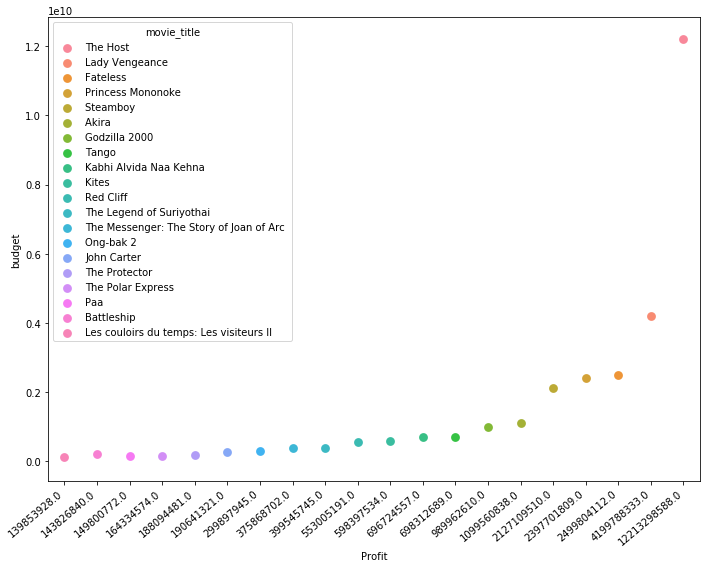
movie\_df= movie\_df.sort\_values(by ='Profit' , ascending=False)

movie\_df\_new=movie\_df.head(20)

ax=sns.pointplot(movie\_df\_new['Profit'], movie\_df\_new['budget'], hue=movie\_df\_new['movie\_title'])

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=40, ha="right")

plt.tight\_layout()



*# Top 20 movies based on the profit percentage*

plt.figure(figsize=(10,8))

movie\_df= movie\_df.sort\_values(by ='Profit\_Percentage' , ascending=False)

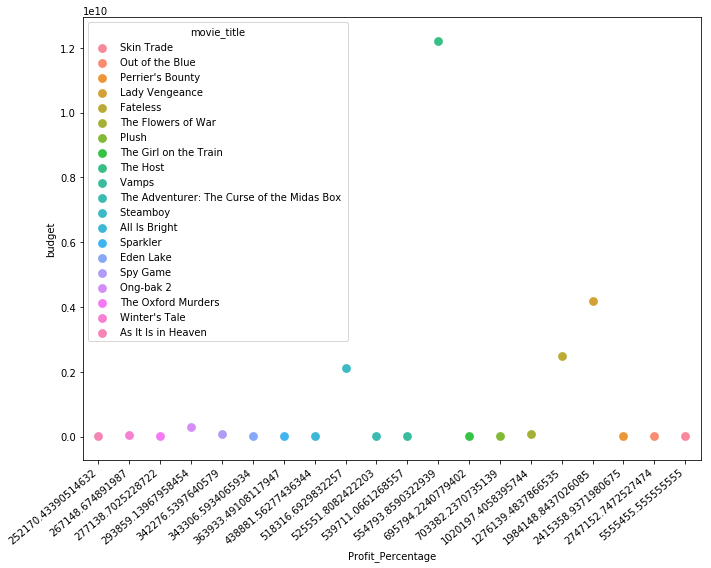
movie\_df\_new=movie\_df.head(20)

ax=sns.pointplot(movie\_df\_new['Profit\_Percentage'], movie\_df\_new['budget'], hue=movie\_df\_new['movie\_title'])

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=40, ha="right")

plt.tight\_layout()

plt.show()



*#Top 20 directors based on the IMDB ratings*

plt.figure(figsize=(10,8))

movie\_df= movie\_df.sort\_values(by ='imdb\_score' , ascending=False)

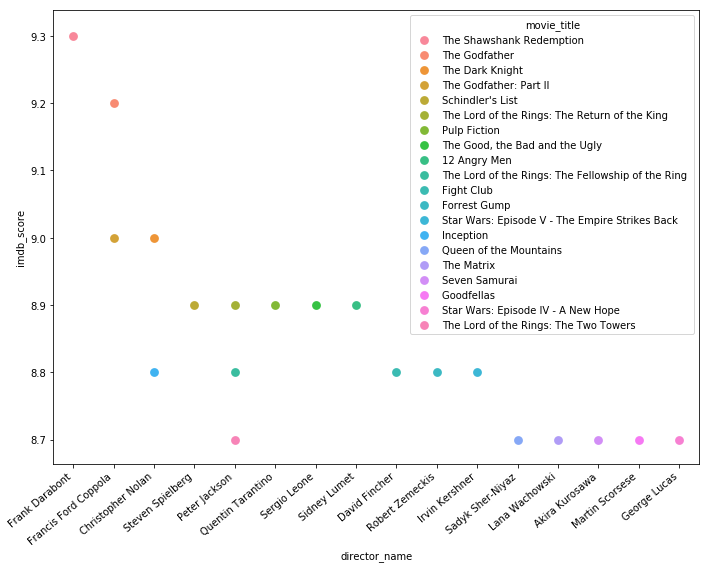
movie\_df\_new=movie\_df.head(20)

ax=sns.pointplot(movie\_df\_new['director\_name'], movie\_df\_new['imdb\_score'], hue=movie\_df\_new['movie\_title'])

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=40, ha="right")

plt.tight\_layout()

plt.show()



*#Commercial success vs critial acclaim*

movie\_df= movie\_df.sort\_values(by ='Profit\_Percentage' , ascending=False)

movie\_df\_new=movie\_df.head(20)

(ggplot(movie\_df\_new)

+ aes(x='imdb\_score', y='gross',color = "content\_rating")

+ geom\_point()

+ geom\_hline(aes(yintercept = 600)) +

geom\_vline(aes(xintercept = 10)) +

xlab("Imdb score") +

ylab("Gross money earned in million dollars") +

ggtitle("Commercial success Vs Critical acclaim") +

annotate("text", x = 8.5, y = 700, label = "High ratings **\n** & High gross"))

/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord\_cartesian.py:31: Mat

/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord\_cartesian.py:31: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

self.limits = Bunch(xlim=xlim, ylim=ylim)

/opt/conda/lib/python3.6/copy.py:274: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

y = func(\*args)

/opt/conda/lib/python3.6/site-packages/plotnine/layer.py:520: MatplotlibDeprecationWarning: isinstance(..., numbers.Number)

return not cbook.iterable(value) and (cbook.is\_numlike(value) or

/opt/conda/lib/python3.6/site-packages/plotnine/facets/facet.py:151: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

scales = Bunch()

/opt/conda/lib/python3.6/site-packages/plotnine/facets/layout.py:147: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

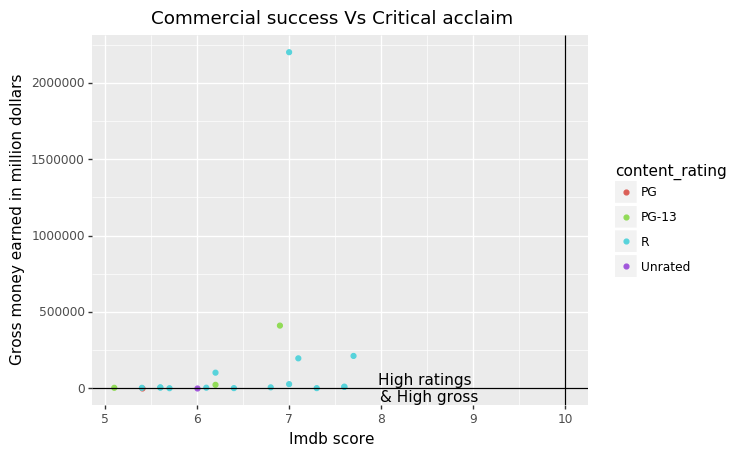
return Bunch(x=xsc, y=ysc)

/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord.py:144: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

y=panel\_params['y\_range'])

/opt/conda/lib/python3.6/site-packages/plotnine/guides/guide\_legend.py:179: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

self.glayers.append(Bunch(geom=geom, data=data, layer=l))



*Top 20 actors of movies based on the commerical success*

plt.figure(figsize=(10,8))

movie\_df= movie\_df.sort\_values(by ='Profit\_Percentage' , ascending=False)

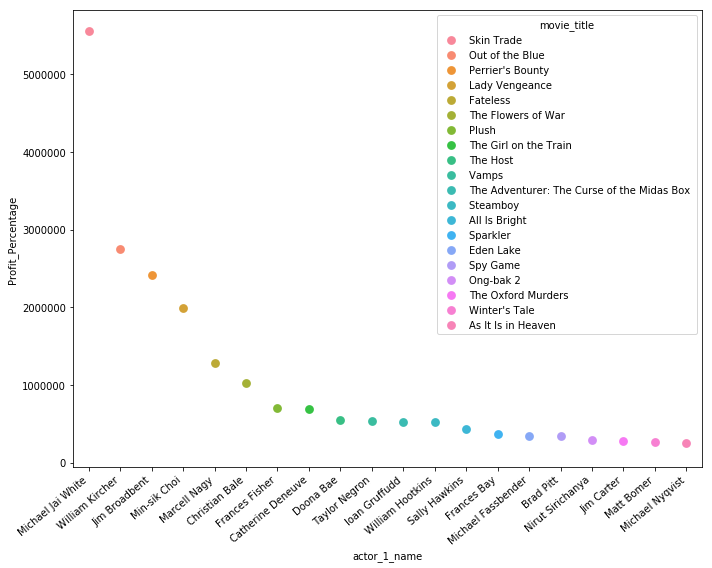
movie\_df\_new=movie\_df.head(20)

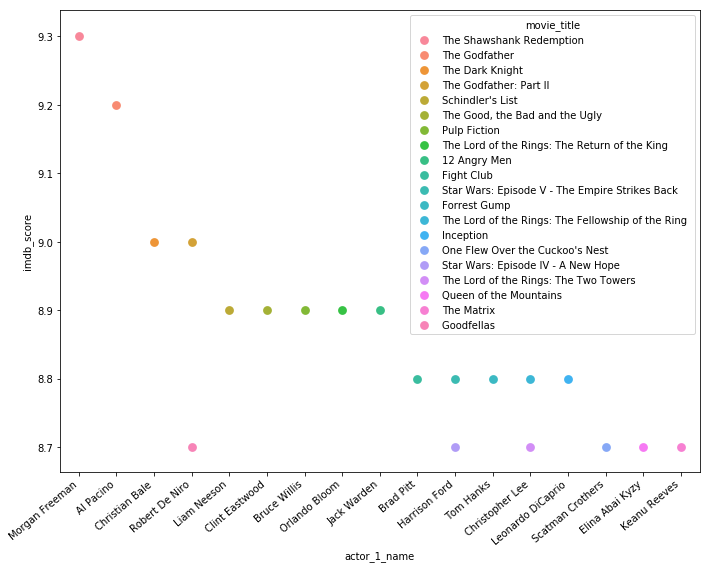
ax=sns.pointplot(movie\_df\_new['actor\_1\_name'], movie\_df\_new['Profit\_Percentage'], hue=movie\_df\_new['movie\_title'])

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=40, ha="right")

plt.tight\_layout()

plt.show()





*# Correlation with heat map*

import matplotli.pyplot as plot

import seaborn as sns

corr = movie\_df.corr()

sns.set\_context("notebook", font\_scale=1.0, rc={"lines.linewidth": 2.5})

plt.figure(figsize=(13,7))

*# create a mask so we only see the correlation values once*

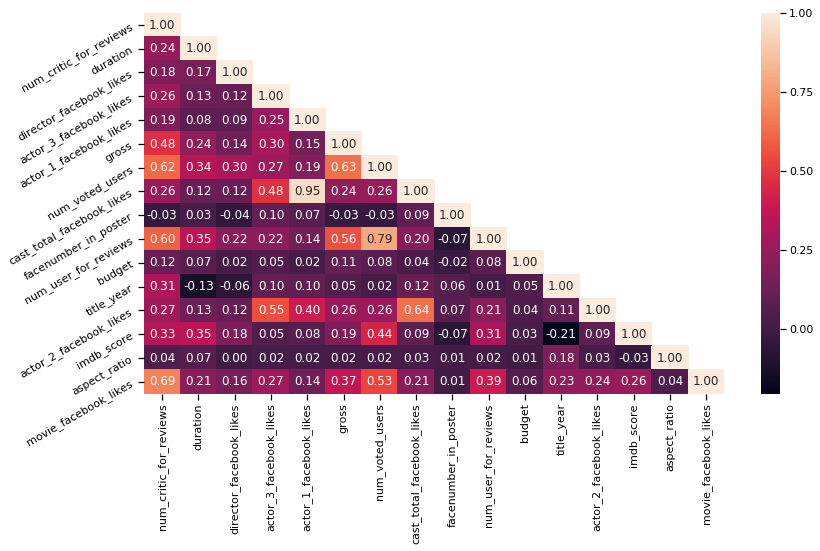
mask = np.zeros\_like(corr)

mask[np.triu\_indices\_from(mask, 1)] = True

a = sns.heatmap(corr,mask=mask, annot=True, fmt='.2f')

rotx = a.set\_xticklabels(a.get\_xticklabels(), rotation=90)

roty = a.set\_yticklabels(a.get\_yticklabels(), rotation=30)



We can see that the cast\_total\_facebook\_likes and actor\_1\_facebook\_like are highly correlated to each other. Both actor2 and actor3 are also somehow correlated to the total. So we want to modify them into two variables: actor\_1\_facebook\_likes and other\_actors\_facebook\_likes.

There are high correlations among num\_voted\_users, num\_user\_for\_reviews and num\_critic\_for\_reviews. We want to keep num\_voted\_users and take the ratio of num\_user\_for\_reviews and num\_critic\_for\_reviews.

*# New Correlation matrix shown in the figure*

import matplotlib.pyplot as plot

import seaborn as sns

corr = movie\_df.corr()

sns.set\_context("notebook", font\_scale=1.0, arc={"lines.linewidth": 2.5})

plt.figure(figsize=(13,7))

*# create a mask so we only see the correlation values once*

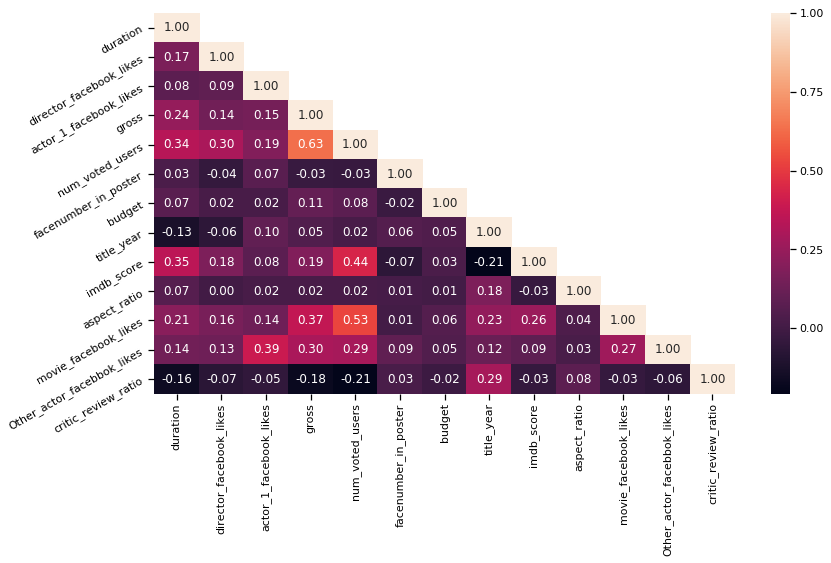
mask = np.zeros\_like(corr)

mask[np.triu\_indices\_from(mask, 1)] = True

a = sns.heatmap(corr,mask=mask, annot=True, fts='.2f')

rotx = a.set\_xticklabels(a.get\_xticklabels(), rotation=90)

roty = a.set\_yticklabels(a.get\_yticklabels(), rotation=30)



from sklearn.metrics import classification\_report

print('Logistic Reports**\n**',classification\_report(y\_test, y\_pred))

print('KNN Reports**\n**',classification\_report(y\_test, knnpred))

print('SVC Reports**\n**',classification\_report(y\_test, svcpred))

print('Naive BayesReports**\n**',classification\_report(y\_test, gaussiannbpred))

print('Decision Tree Reports**\n**',classification\_report(y\_test, dtreepred))

print('Ada Boosting**\n**',classification\_report(y\_test, abcl\_pred))

print('Random Forests Reports**\n**',classification\_report(y\_test, rfcpred))

print('Bagging Clasifier',bgcl.oob\_score\_)

print('Gradient Boosting',classification\_report(y\_test, test\_pred))

print('XGBoosting**\n**',classification\_report(y\_test, xgbprd))

logistic

Reports

precision recall f1-score support

1 0.00 0.00 0.00 46

2 0.50 0.25 0.33 378

3 0.72 0.92 0.81 924

4 0.84 0.52 0.65 61

accuracy 0.69 1409

macro avg 0.52 0.42 0.45 1409

weighted avg 0.64 0.69 0.65 1409

KNN Reports

precision recall f1-score support

1 0.00 0.00 0.00 46

2 0.46 0.41 0.44 378

3 0.73 0.83 0.78 924

4 1.00 0.20 0.33 61

accuracy 0.67 1409

macro avg 0.55 0.36 0.39 1409

weighted avg 0.64 0.67 0.64 1409

SVC Reports

precision recall f1-score support

1 0.14 0.02 0.04 46

2 0.42 0.42 0.42 378

3 0.74 0.79 0.76 924

4 0.57 0.33 0.42 61

accuracy 0.64 1409

macro avg 0.47 0.39 0.41 1409

weighted avg 0.62 0.64 0.63 1409

Naive BayesReports

precision recall f1-score support

1 0.05 0.91 0.09 46

2 0.50 0.00 0.01 378

3 0.71 0.01 0.01 924

4 0.11 0.85 0.19 61

accuracy 0.07 1409

macro avg 0.34 0.44 0.07 1409

weighted avg 0.61 0.07 0.02 1409

Decision Tree Reports

precision recall f1-score support

1 0.11 0.11 0.11 46

2 0.47 0.51 0.49 378

3 0.78 0.76 0.77 924

4 0.77 0.59 0.67 61

accuracy 0.67 1409

macro avg 0.53 0.49 0.51 1409

weighted avg 0.67 0.67 0.67 1409

Ada Boosting

precision recall f1-score support

1 0.17 0.15 0.16 46

2 0.46 0.51 0.48 378

3 0.77 0.75 0.76 924

4 0.65 0.51 0.57 61

accuracy 0.65 1409

macro avg 0.51 0.48 0.49 1409

weighted avg 0.66 0.65 0.66 1409

Random Forests Reports

precision recall f1-score support

1 1.00 0.04 0.08 46

2 0.62 0.47 0.53 378

3 0.77 0.92 0.84 924

4 0.96 0.44 0.61 61

accuracy 0.75 1409

macro avg 0.84 0.47 0.52 1409

weighted avg 0.75 0.75 0.72 1409

Bagging Clasifier 0.7429179978700745

Gradient Boosting precision recall f1-score support

1 0.25 0.02 0.04 46

2 0.60 0.56 0.58 378

3 0.80 0.88 0.84 924

4 0.86 0.49 0.62 61

accuracy 0.75 1409

macro avg 0.63 0.49 0.52 1409

weighted avg 0.73 0.75 0.74 1409

XGBoosting

precision recall f1-score support

1 0.25 0.02 0.04 46

2 0.59 0.52 0.55 378

3 0.79 0.89 0.84 924

4 0.89 0.54 0.67 61

accuracy 0.75 1409

macro avg 0.63 0.49 0.53 1409

weighted avg 0.72 0.75 0.73 1409

opt/conda/lib/python3.6/site-packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.

'precision', 'predicted', average, warn\_for

**10.Conclusion**

The conclusion is that Random Forest Algorithm along with the gradient boosting have the accuracy of 74.5 and 75.5 respectively

**Done by**:

NAME: AVULA MALLA REDDY

NM ID: AU720921244010

JCT COLLEGE OF ENGINEERING