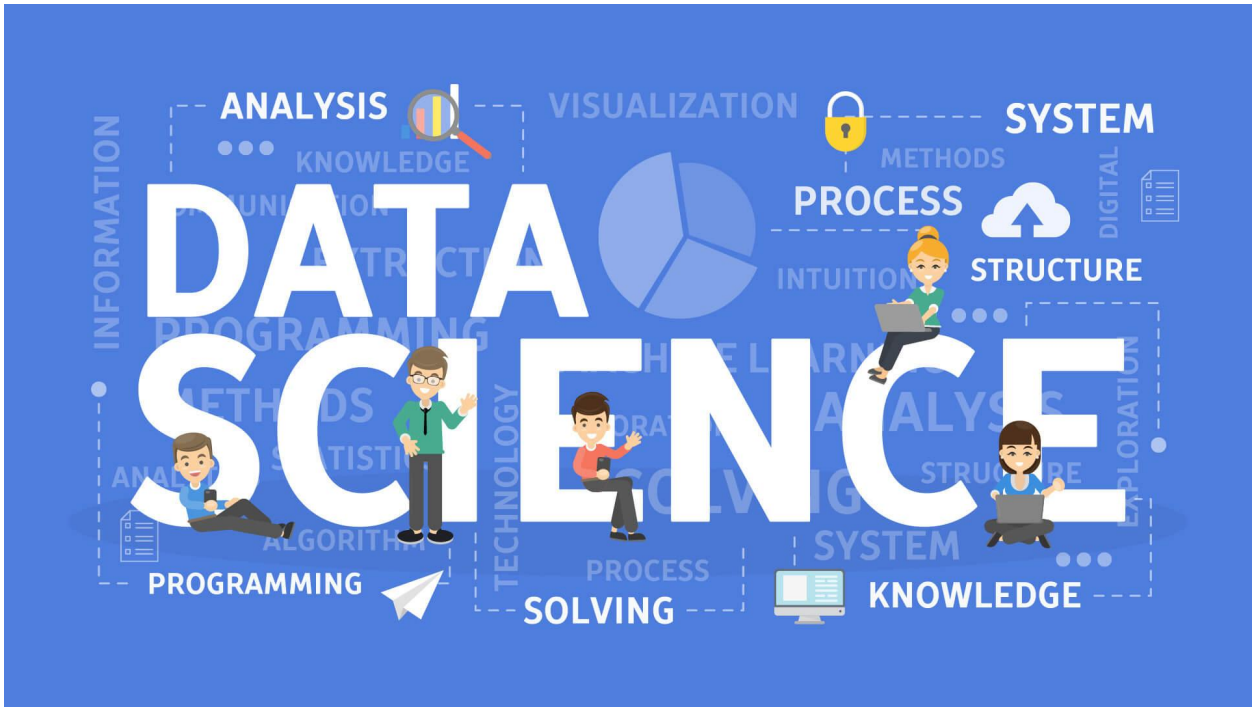
Movie Popularity Analysis

IDS Project Report

Group:- ***Movie Critics***



Team Members

Name	Roll No.
Mayank Roy	18UCS024
Gautam Agrawal	18UCS025
Piyush Priyadarshi	18UCS026
Dev Patel	18UCS197

Introduction

This is our IDS project report, in which we have done a popularity analysis on the hollywood movies released between 2014-2015. We have picked a dataset which contains information about these movies and studied their popularity.

The link for the dataset is as follows:-
https://archive.ics.uci.edu/ml/datasets/CSM+%28Conventional+and+Social+Media+Movies%29+Dataset+2014+and+2015#

We have built our code for the project in Python Language in Spyder IDE.

Code Link:-
https://github.com/gautam2346/IDS-data-preprocessing-and-algos/blob/main/import.pu.py

Objective

Apply different ML Classification Algorithms on the dataset and derive inferences from data using Python.

System Requirements

1. Python3 should be installed in the PC
2. Data Science Packages such as matplotlib, pandas, scikit-learn, numpy etc.
3. The code can be run in Spyder IDE for better understanding and results, if installed. Some of the statistical data was derived using Jupyter Notebook.

About the Data

Our data is based on the Hollywood movies released between 2014 and 2015. The various specifications of this dataset are as follows:-

Data Set Characteristics:	Multivariate	Number of Instances:	217	Area:	Computer
Attribute Characteristics:	Integer	Number of Attributes:	12	Date Donated	2017-10-11
Associated Tasks:	Classification, Regression	Missing Values?	Yes	Number of Web Hits:	33288

The attributes of this dataset are:-

Movie	Year
Rating	Genre
Gross	Budget
Screen	Sequel
Sentiment	Views
Likes	Dislikes
Comments	Aggregate Followers

This is a sample description of our dataset:-

	Movie	Year	Ratings	Genre	Gross	Budget	Screens	Sequel	Sentiment	Views	Likes	Dislikes	Comments	Aggregate Followers
2	13 Sins	2014	6.3	8	9130	4000000	45	1	0	3280543	4632	425	636	1120000
3	22 Jump Street	2014	7.1	1	192000000	50000000	3306	2	2	583289	3465	61	186	12350000
4	3 Days to Kill	2014	6.2	1	30700000	28000000	2872	1	0	304861	328	34	47	483000
5	300: Rise of an Empire	2014	6.3	1	106000000	110000000	3470	2	0	452917	2429	132	590	568000
6	A Haunted House 2	2014	4.7	8	17300000	3500000	2310	2	0	3145573	12163	610	1082	1923800
7	A Long Way Off	2014	4.6	3	29000	500000		1	0	91137	112	7	1	310000
8	A Million Ways to Die in the West	2014	6.1	8	42600000	40000000	3158	1	0	3013011	9595	419	1020	8153000
9	A Most Violent Year	2014	7.1	1	5750000	20000000	818	1	2	1854103	2207	197	593	130655
10	A Walk Among the Tombstones	2014	6.5	10	26000000	28000000	2714	1	3	2213659	2210	419	382	125646
11	About Last Night	2014	6.1	8	48600000	12500000	2253	1	0	5218079	11709	532	770	21697300
12	American Sniper	2014	7.3	1	350000000	58800000	3555	1	4	3927600	13143	573	3134	24300
13	And So It Goes	2014	5.7	8	15200000	30000000	1762	1	0	519327	963	94	70	386400
14	Annabelle	2014	5.4	15	84300000	6500000	3185	1	0	19032902	38810	4382	4392	19420105
15	Annie	2014	5.2	8	85900000	65000000	3116	1	0	930006	5150	707	1484	5130800
16	Atlas Shrugged: Who Is John Galt?	2014	4.4	3	830000	5000000	65	3	0	595194	85	36	39	15112
17	Barefoot	2014	6.6	8	11800	6000000	18	1	2	3915978	6983	247	460	253000
18	Better Living Through Chemistry	2014	6.3	8	72300	5000000	25	1	0	1391527	2479	146	182	1658900

Data Preprocessing

While performing data preprocessing, we did the following:-

1. Imported libraries such as matplotlib, pandas, scikit-learn and numpy.
2. Imported our dataset CSM_2014_2015_dataset.csv in our code using read_csv() command of pandas and stored it in a dataframe.
3. Now a rough view of the dataset was taken using df.head() command.

	Movie	Year	Ratings	Genre	Gross	Budget	Screens	Sequel	Sentiment	Views	Likes	Dislikes	Comments	Aggregate Followers
0	13 Sins	2014.0	6.3	8.0	9130.0	4000000.0	45.0	1.0	0.0	3280543.0	4632.0	425.0	636.0	1120000.0
1	22 Jump Street	2014.0	7.1	1.0	192000000.0	50000000.0	3306.0	2.0	2.0	583289.0	3465.0	61.0	186.0	12350000.0
2	3 Days to Kill	2014.0	6.2	1.0	30700000.0	28000000.0	2872.0	1.0	0.0	304861.0	328.0	34.0	47.0	483000.0
3	300: Rise of an Empire	2014.0	6.3	1.0	106000000.0	110000000.0	3470.0	2.0	0.0	452917.0	2429.0	132.0	590.0	568000.0
4	A Haunted House 2	2014.0	4.7	8.0	17300000.0	3500000.0	2310.0	2.0	0.0	3145573.0	12163.0	610.0	1082.0	1923800.0

4. To check the no. of data in a column and its datatype df.info() is used.

```
In [8]: mydataset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232 entries, 0 to 231
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Movie                231 non-null    object
 1   Year                 231 non-null    float64
 2   Ratings              231 non-null    float64
 3   Genre                231 non-null    float64
 4   Gross                231 non-null    float64
 5   Budget               230 non-null    float64
 6   Screens              221 non-null    float64
 7   Sequel               231 non-null    float64
 8   Sentiment            231 non-null    float64
 9   Views                231 non-null    float64
 10  Likes                231 non-null    float64
 11  Dislikes             231 non-null    float64
 12  Comments             231 non-null    float64
 13  Aggregate Followers  196 non-null    float64
dtypes: float64(13), object(1)
```

5. Now we will check for null objects.

```
In [10]: mydataset.isna().sum()
Out[10]:
Movie                   1
Year                    1
Ratings                 1
Genre                   1
Gross                   1
Budget                  2
Screens                11
Sequel                  1
Sentiment               1
Views                   1
Likes                   1
Dislikes                1
Comments                1
Aggregate Followers    36
dtype: int64
```

6. For better results, we filled the null values with some other values using mydataset.interpolate() command. It fills the null value with the mean of previous non- null value and next non-null value of the same column.
7. We then find the values of the central tendencies of the data.

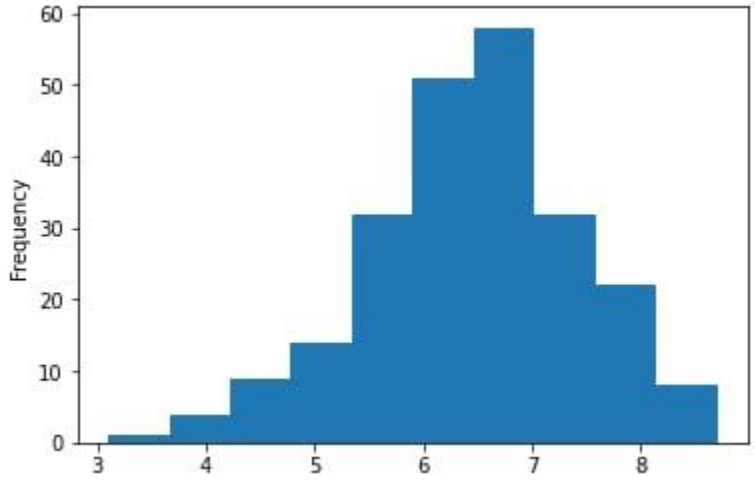
	Year	Ratings	Genre	Gross	Budget	Screens	Sequel	Sentiment	Views	Likes	Dislikes	Comments	Aggregate Followers
count	232.000000	232.000000	232.000000	2.320000e+02	2.320000e+02	232.000000	232.000000	232.000000	2.320000e+02	232.000000	232.000000	232.000000	2.320000e+02
mean	2014.297414	6.432759	5.400862	6.787006e+07	4.758878e+07	2201.495690	1.357759	2.775862	3.699691e+06	12689.900862	677.982759	1820.443966	2.857635e+06
std	0.458109	0.995686	4.180826	8.876046e+07	5.417779e+07	1440.880589	0.965433	7.000416	4.505790e+06	28770.354251	1241.340903	3564.202205	4.571739e+06
min	2014.000000	3.100000	1.000000	2.470000e+03	7.000000e+04	2.000000	1.000000	-38.000000	6.980000e+02	1.000000	0.000000	0.000000	1.066000e+03
25%	2014.000000	5.800000	1.000000	1.035000e+07	8.983195e+06	476.750000	1.000000	0.000000	6.308380e+05	1811.750000	105.750000	249.250000	2.237500e+05
50%	2014.000000	6.500000	3.000000	3.690000e+07	2.750000e+07	2761.500000	1.000000	0.000000	2.401178e+06	6024.500000	344.000000	817.000000	1.188000e+06
75%	2015.000000	7.100000	8.000000	8.932500e+07	6.500000e+07	3323.500000	1.000000	5.250000	5.217030e+06	15195.750000	694.250000	2120.500000	3.191675e+06
max	2015.000000	8.700000	15.000000	6.430000e+08	2.500000e+08	4324.000000	7.000000	29.000000	3.262678e+07	370552.000000	13960.000000	38363.000000	3.103000e+07

Inferences

Here are some of the inferences that we make from the given dataset.

1) Histogram plot (Frequency vs rating)

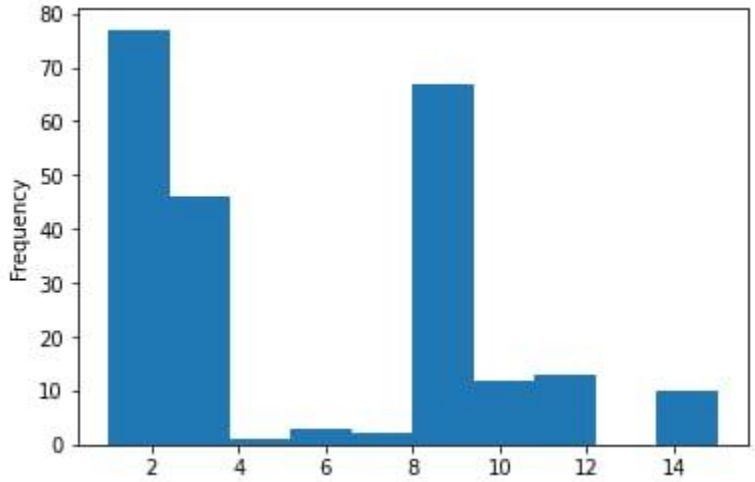
```
In [11]: new_df['Ratings'].plot(kind = 'hist')
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



From the given histogram, we can see that most movies had ratings in the 6-7 range with a gradual decrease in the number of movies on either side of said range. There are hardly any movies rated greater than 9 and less than 4.

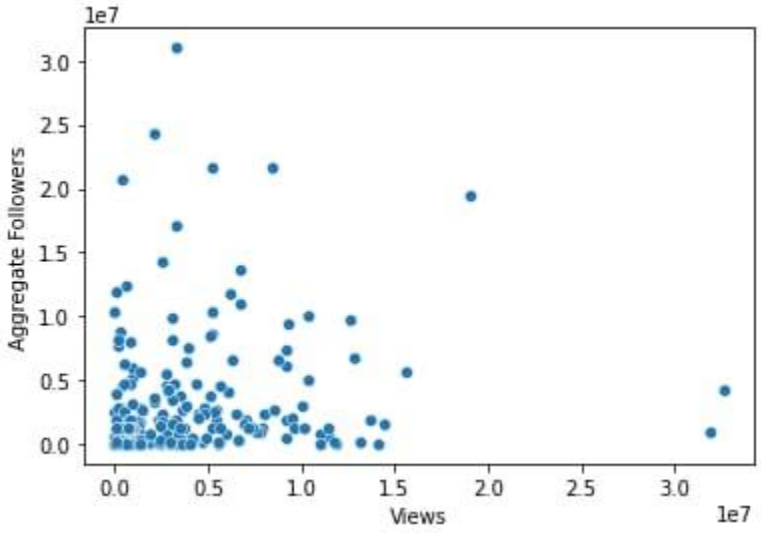
2) Histogram Plot (Frequency vs Genre)

```
In [12]: new_df['Genre'].plot(kind = 'hist')
Out[12]: <AxesSubplot:ylabel='Frequency'>
```



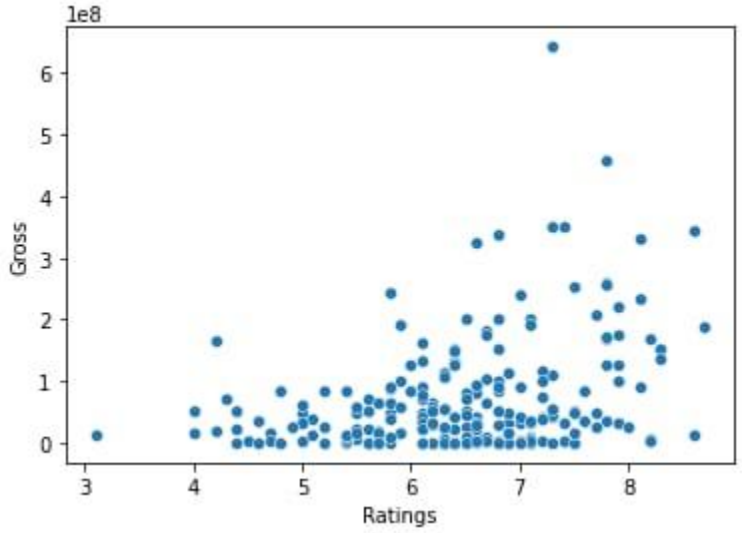
From the above histogram, we can see that the highest number of movies belonged to genre 1 followed by genre 8 and genre 3. Only 2 movies belonged to genre 7, 1 to genre 4 and no movies were released from genre 5, 11, 13 and 14.

3) Scatter plot (Aggregate followers vs Views)



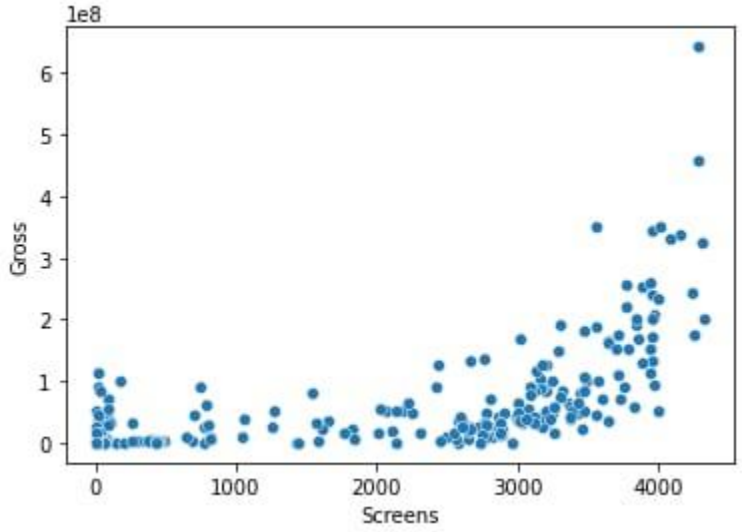
We had expected to see more or less a linear relationship between views and aggregate followers since if a movie has a larger following, then more people will view it and vice versa. But no such relationship was observed as can be seen from the above scatter plot. Some outliers were also observed where some movies with very few followers had lots of views and some movies with a large following had very less views.

4) Scatter plot (Gross vs Ratings)



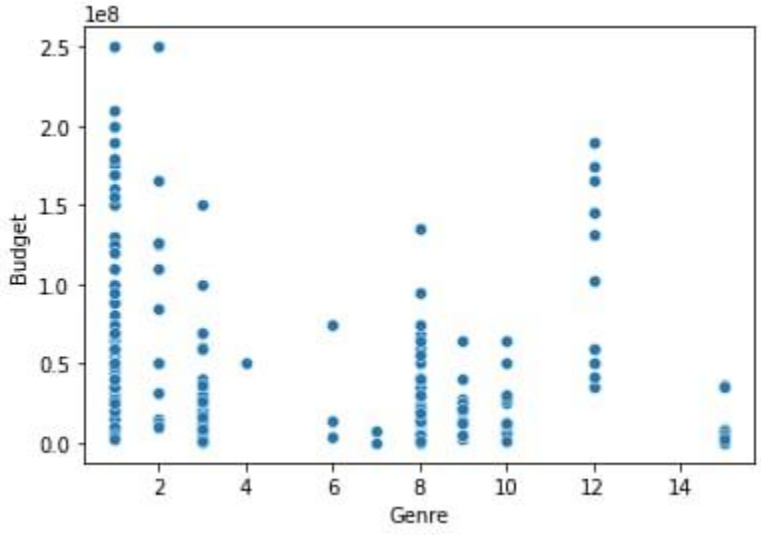
On expected lines, it was found that higher rated movies made more money, generally, than lower rated movies.

5) Scatter plot (Gross vs Screens)



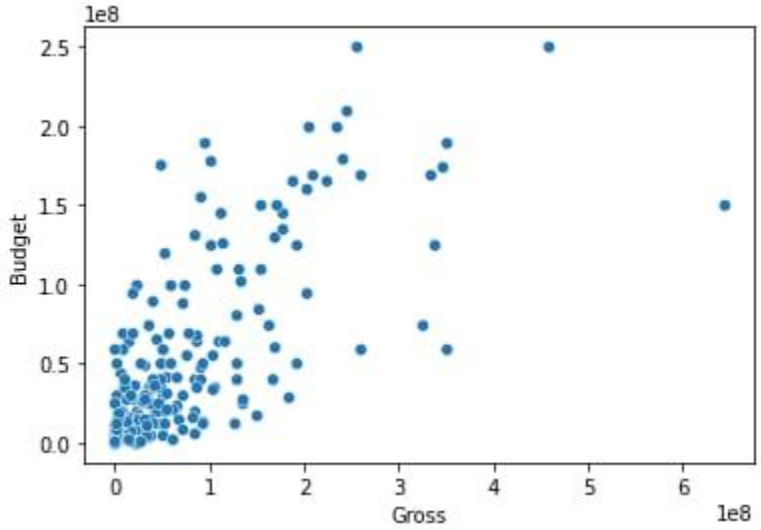
Generally, it was found that revenue generated by the movies increased with the number of screenings they got, but there were indeed some movies which generated a fair bit of revenue even though they got negligible screening, leading us to believe that movies don't just make money from selling tickets but from other activities like selling and renting DVDs and Blurays, streaming on TV, VOD and OTT platforms, selling merchandise, etc.

6) Scatter plot (Budget vs Genre)



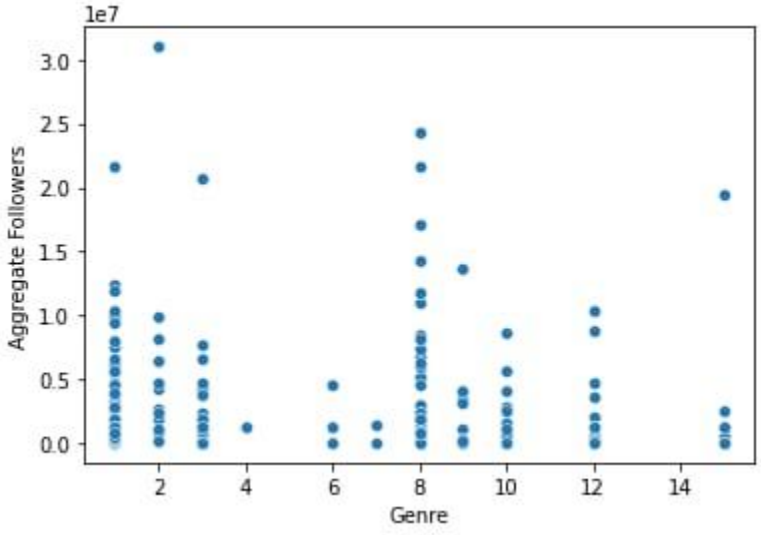
As we can see, movies with the highest budget belong to genre 1 and 2. But on an average, most big budget movies belong to genre 12 while most movies on a tight budget belong to genre 8.

7) Scatter plot (Budget vs Gross)



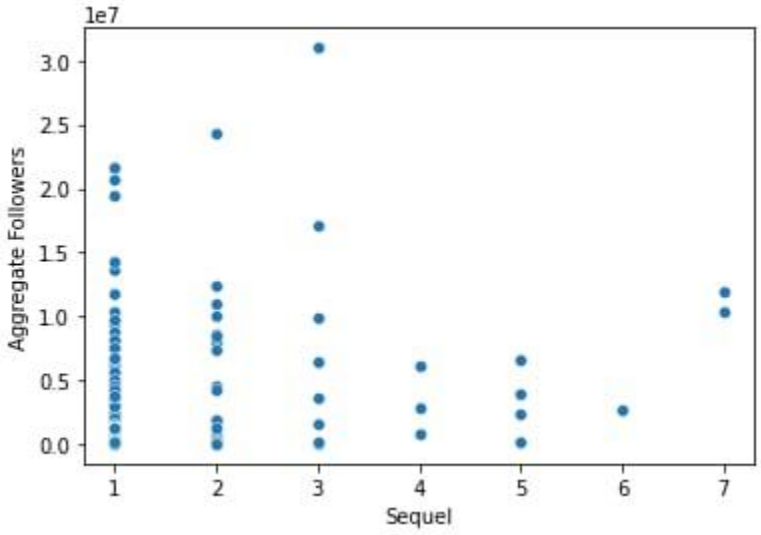
A comparison of budget and gross shows, barring very few outliers, the budget and gross revenue of movies were mostly comparable with most movies making a profit (gross > budget). Another pattern we found that none of the high budget movies incurred a loss. It was only some of the low budget movies that incurred losses.

8) Scatter plot (Aggregate followers vs Genre)



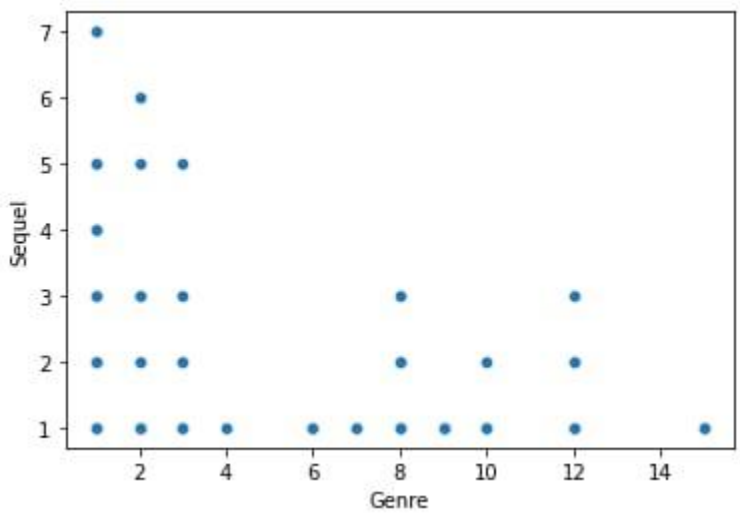
As we can see, the aggregate followers of movies belonging to genre 8 are the highest on an average and that for genre 10 are the least.

9) Scatter plot (Aggregate followers vs Sequel)



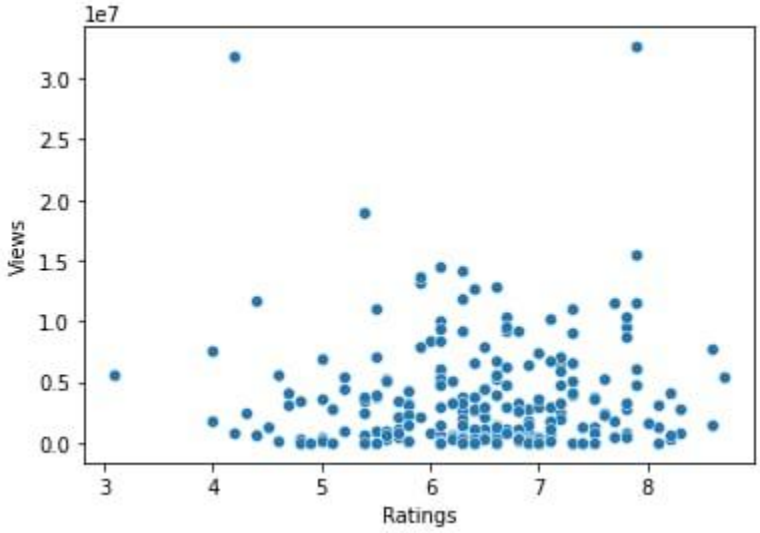
We were hoping to see whether the public followed more standalone movies or franchise movies and whether the increase in the number of sequels had any effect on the following the franchise had. Unfortunately, no obvious pattern could be found between both.

10) Scatter plot (Sequel vs Genre)



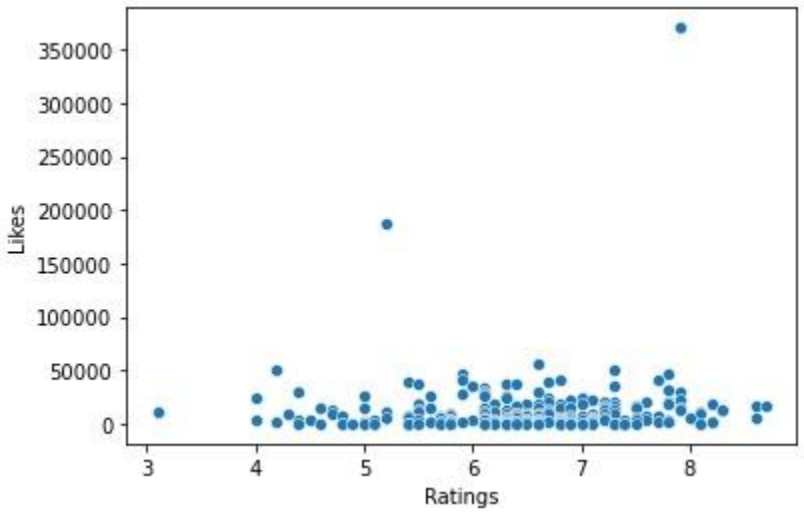
It was found that most franchise films belonged to genre 1 and. Apart from these, genres 3, 8 and 12 had some franchise films. All other genres dealt exclusively with standalone titles.

11) Scatter plot (Views vs Rating)



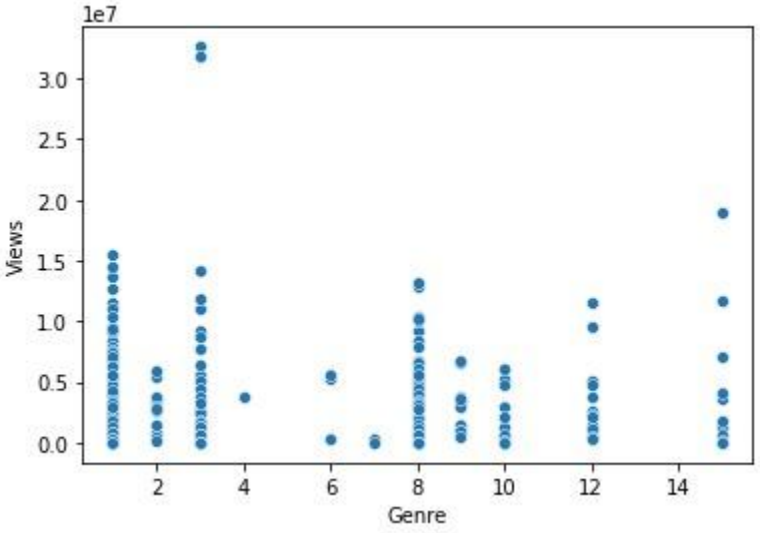
We wanted to see if the rating a movie got affected the viewing habit of people. From the above plot, we found that it had little to no effect and besides a few outliers, people viewed movies from the entire spectrum of ratings with almost equal interest.

12) Scatter plot (Likes vs Rating)



Again, there was no correlation visible between the ratings and likes a movie received with likes uniformly distributed across the entire spectrum of ratings, with the exception of 2 outliers.

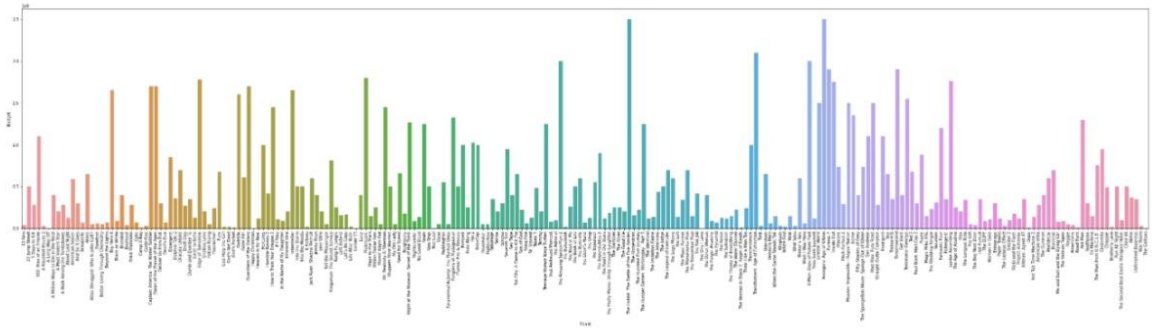
13) Scatter plot (Views vs Genre)



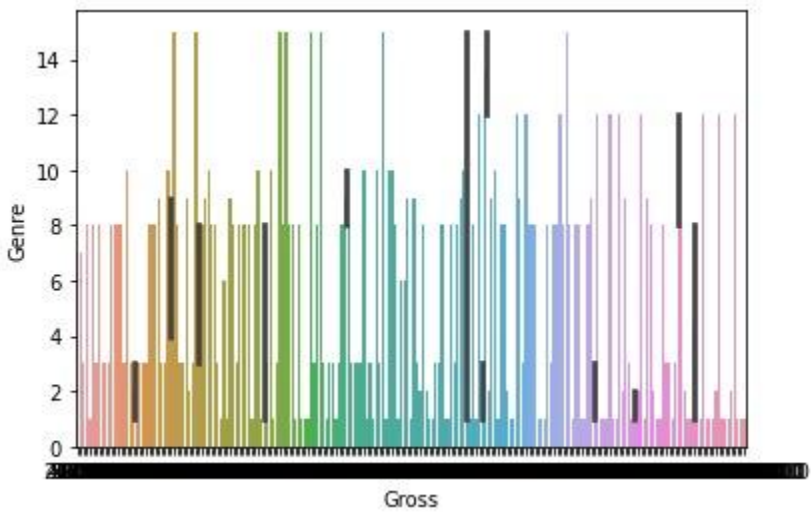
As we can see, movies with the highest number of views belong to genre 8 on an average, followed by genre 3 and 1.

14) Bar graph plot (Movies vs Budget)

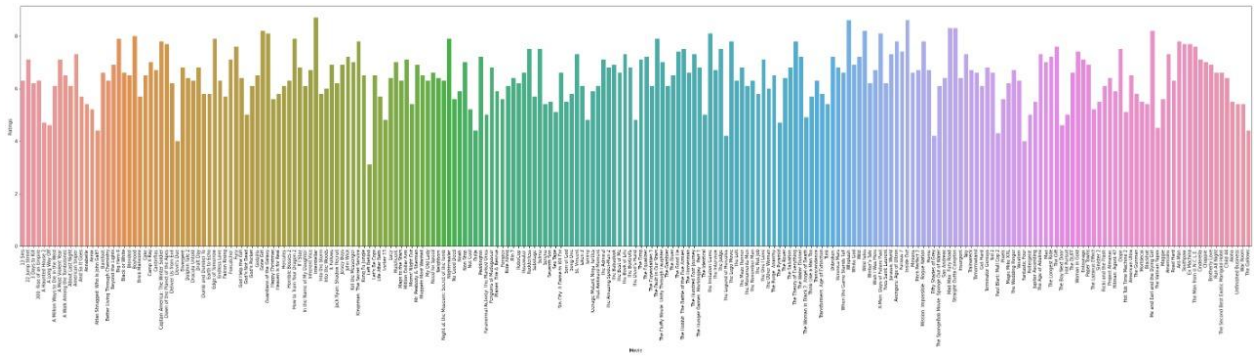
```
[18]: movie = list(new_df['Movie'])
      fig = plt.gcf()
      fig.set_size_inches(50,10)
      ax = sns.barplot(y= new_df['Budget'],x=new_df['Movie'])
      lables = ax.set_xticklabels(movie , rotation = 90)
```



15) Bar graph plot (Genre vs Gross)



16) Bar graph plot (Rating vs Movie)



Model Building

1. Review:- All movies have ratings ranging from 1 to 10. We divided these ratings into 3 different categories: Bad (<5), Average (5-6) and Good (>6).

```
Ratings = new_df["Ratings"].values
Review = []
for num in Ratings:
    if num < 5:
        Review.append("Bad")
    elif num >=7:
        Review.append("Good")
    else:
       Review.append("Average")

new_df['Review'] = Review
```

	Movie	Year	Ratings	Genre	Gross	Budget	Screens	Sequel	Sentiment	Views	Likes	Dislikes	Comments	Aggregate Followers	Review
0	13 Sins	2014.0	6.3	8.0	9130.0	4000000.0	45.0	1.0	0.0	3280543.0	4632.0	425.0	636.0	1120000.0	Average
1	22 Jump Street	2014.0	7.1	1.0	192000000.0	50000000.0	3306.0	2.0	2.0	583289.0	3465.0	61.0	186.0	12350000.0	Good
2	3 Days to Kill	2014.0	6.2	1.0	30700000.0	28000000.0	2872.0	1.0	0.0	304861.0	328.0	34.0	47.0	483000.0	Average
3	300: Rise of an Empire	2014.0	6.3	1.0	106000000.0	110000000.0	3470.0	2.0	0.0	452917.0	2429.0	132.0	590.0	568000.0	Average
4	A Haunted House 2	2014.0	4.7	8.0	17300000.0	3500000.0	2310.0	2.0	0.0	3145573.0	12163.0	610.0	1082.0	1923800.0	Bad

2. Commercial Success:- All movies have a budget and gross earning. We summarised Commercial Success on the basis of difference of budget and gross earning.

```
Gross = new_df['Gross'].values
Budget = new_df['Budget'].values
commercial_success = [0]*len(Gross)
for i in range(0,len(Gross)) :
    num = (Gross[i] - Budget[i])
    if num < 10000000:
        commercial_success[i] = "flop"
    elif num > 17000000:
        commercial_success[i] ="Super hit"
    else:
        commercial_success[i] = "hit"
new_df['commercial_success'] =  commercial_success
```

```
[7]: display(new_df.head())
```

	Movie	Year	Ratings	Genre	Gross	Budget	Screens	Sequel	Sentiment	Views	Likes	Dislikes	Comments	Aggregate Followers	Review	commercial_success
0	13 Sins	2014.0	6.3	8.0	9130.0	4000000.0	45.0	1.0	0.0	3280543.0	4632.0	425.0	636.0	1120000.0	Average	flop
1	22 Jump Street	2014.0	7.1	1.0	192000000.0	50000000.0	3306.0	2.0	2.0	583289.0	3465.0	61.0	186.0	12350000.0	Good	Super hit
2	3 Days to Kill	2014.0	6.2	1.0	30700000.0	28000000.0	2872.0	1.0	0.0	304861.0	328.0	34.0	47.0	483000.0	Average	flop
3	300: Rise of an Empire	2014.0	6.3	1.0	106000000.0	110000000.0	3470.0	2.0	0.0	452917.0	2429.0	132.0	590.0	568000.0	Average	flop
4	A Haunted House 2	2014.0	4.7	8.0	17300000.0	3500000.0	2310.0	2.0	0.0	3145573.0	12163.0	610.0	1082.0	1923800.0	Bad	hit

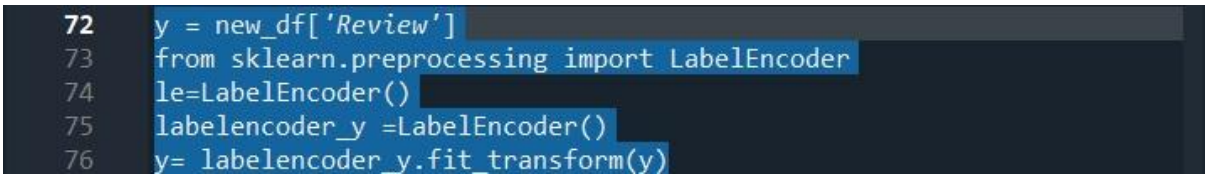
3. <u>Critical Success:-</u> All movies have a Likes-Dislikes figure. The ratio of Likes and Dislikes determines our Critical Success

```
Likes = new_df['Likes'].values
Dislikes = new_df['Dislikes'].values
critical_success = [0]*len(Gross)
for i in range(0,len(Likes)) :
    num = Likes[i] / Dislikes[i]
    if num < 15:
        critical_success[i] = "not supported by public"
    else:
        critical_success[i] = "supported by public"
new_df['critical_success'] =  critical_success
```

```
[9]: display(new_df.head())
```

	Movie	Year	Ratings	Genre	Gross	Budget	Screens	Sequel	Sentiment	Views	Likes	Dislikes	Comments	Aggregate Followers	Review	commercial_success	critical_success
0	13 Sins	2014.0	6.3	8.0	9130.0	4000000.0	45.0	1.0	0.0	3280543.0	4632.0	425.0	636.0	1120000.0	Average	flop	not supported by public
1	22 Jump Street	2014.0	7.1	1.0	192000000.0	50000000.0	3306.0	2.0	2.0	583289.0	3465.0	61.0	186.0	12350000.0	Good	Super hit	supported by public
2	3 Days to Kill	2014.0	6.2	1.0	30700000.0	28000000.0	2872.0	1.0	0.0	304861.0	328.0	34.0	47.0	483000.0	Average	flop	not supported by public
3	300: Rise of an Empire	2014.0	6.3	1.0	106000000.0	110000000.0	3470.0	2.0	0.0	452917.0	2429.0	132.0	590.0	568000.0	Average	flop	supported by public
4	A Haunted House 2	2014.0	4.7	8.0	17300000.0	3500000.0	2310.0	2.0	0.0	3145573.0	12163.0	610.0	1082.0	1923800.0	Bad	hit	supported by public

4. Targeting the Labels:- After creating these new columns, we labelled them with some numericals:
    a) For ‘Review’, 2 is ‘Good’, 1 is ‘Average’ and 0 is ‘Bad’.
    b) For ‘commercial_success’, 2 is for ‘Super hit’, 1 is for ‘hit’, and 0 is for ‘flop’.
    c) For ‘critical_success’, 0 is for ‘not supported by public’ and 1 is for ‘supported by public’.



Training Data and Test Data

1. For ‘Review’
    a) Now we will count the number of good,bad and average instances in the dataframe.

    b) As we can see there are more average movies than good or bad movies.
    c) Now we separate our dataset into two columns x and y where x signifies known attributes and y signifies predicted attributes.

    d) Now we divide the dataset into two parts, where 75% of our dataset is the training part and the rest is for testing part.

```
80
81  from sklearn.model_selection import train_test_split
82  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state = 0)
83
```

2. <u>For commercial_success</u>
    a) Now we will count the number of Super hit, hit and flop instances in the dataframe.

```
In [10]: new_df['commercial_success'].value_counts()
Out[10]:
flop         125
Super hit     95
hit           12
Name: commercial_success, dtype: int64
```

    b) As we can see there are more flop movies than super hit or hit movies.
    c) Now we separate our dataset into two columns x and y where x signifies known attributes and y signifies predicted attributes.

```
175
176  y = new_df['commercial_success']
177  x = new_df[['Gross', 'Views' , 'Likes' , 'Aggregate Followers']]
178
```

    d) Now we divide the dataset into two parts, where 75% of our dataset is the training part and the rest is for testing part.

```
80
81  from sklearn.model_selection import train_test_split
82  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state = 0)
83
```

3. <u>For critical_success</u>
    a) Now we will count the number of 'supported by public' and 'not supported by public' instances in the dataframe.

```
In [12]: new_df['critical_success'].value_counts()
Out[12]:
supported by public        145
not supported by public     87
Name: critical_success, dtype: int64
```

b) As we can see there are more ‘supported by public’ movies than ‘not supported by public’ movies.
c) Now we separate our dataset into two columns x and y where x signifies known attributes and y signifies predicted attributes.

```
273
274
275  y = new_df['critical_success']
276  x = new_df[['Gross', 'Views' , 'Likes' , 'Aggregate Followers']]
277
```

d) Now we divide the dataset into two parts, where 75% of our dataset is the training part and the rest is for testing part.

```
80
81  from sklearn.model_selection import train_test_split
82  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state = 0)
83
```

Algorithms Applied

For our project, we studied the dataset using the following classifiers:-

1. **Logical Regression**:- It is a supervised learning classification algorithm, in which the probability of a target variable is predicted. It simply builds a model between X (dependent variable) and Y (Independent Variable).

   Basic Idea:- We would be given some x attribute values and their corresponding y attribute values as a part of training data. In logical regression, we would build a 2-D plane model, and then use that model to predict the y values for x values given in the testing data

```
83
84  ##logical regression
85  from sklearn.linear_model import LogisticRegression
86  log_reg = LogisticRegression()
87  log_reg.fit(x_train,y_train)
88  predicted = log_reg.predict(x_test)
89
90  from sklearn.metrics import confusion_matrix
91  from sklearn.metrics import accuracy_score
92  from sklearn.metrics import classification_report
93
94  cm=confusion_matrix(y_test , predicted)
95  print(cm)
96  print("Accuracy is "+str(round(accuracy_score(y_test,predicted),4) *100))
97  print("Classification Report\n"+str(classification_report(y_test, predicted)))
98
```

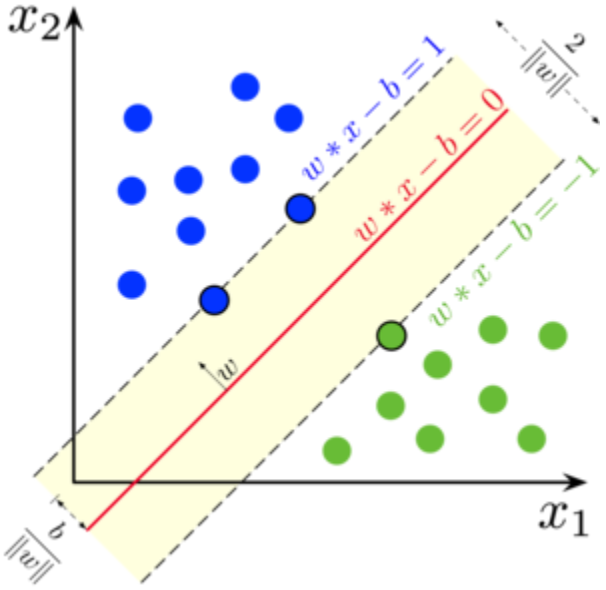
2. **Naive Bayes Classification**:- It is a supervised learning classification algorithm based on Bayes Theorem. It is mainly used in *text classification* that includes a high-dimensional training dataset.

Basic Idea:- Bayes theorem says that P(Y|X) = P(X|Y)P(Y) / P(X). Our given attributes would be mentioned in the X part and we will have to predict the class value which will be Y.

```
100
101 ##naive bayes
102 from sklearn.naive_bayes import GaussianNB
103 classifier=GaussianNB()
104 classifier.fit(x_train,y_train)
105 predicted=classifier.predict(x_test)
106
107 from sklearn.metrics import confusion_matrix
108 from sklearn.metrics import accuracy_score
109 from sklearn.metrics import classification_report
110
111 cm=confusion_matrix(y_test,predicted)
112 print(cm)
113 print("Accuracy is "+str(round(accuracy_score(y_test,predicted),4) *100))
114 print("Classification Report\n"+str(classification_report(y_test, predicted)))
115
```

3. **Support Vector Machine:-** It is a supervised learning algorithm consisting of prediction methods based on the statistical learning framework or VC theory proposed by Vapnik and Chervonenkis.

   Basic Idea:- Suppose some numeric and linearly separable data is given to us which we plot on a plane. Our main goal would be to select the most optimal hyperplane so as to separate the data.



(Note: This diagram is just for an example)

```
118
119  ## SVm
120  from sklearn.svm import SVC
121  classifier=SVC(kernel='rbf',random_state=0,gamma='auto')
122  classifier.fit(x_train,y_train)
123  p=classifier.predict(x_test)
124  from sklearn.metrics import confusion_matrix
125  from sklearn.metrics import accuracy_score
126  from sklearn.metrics import classification_report
127  p=classifier.predict(x_test)
128  cm=confusion_matrix(y_test,p)
129  print(cm)
130  print("Accuracy is "+str(round(accuracy_score(y_test,p),4) *100))
131  print("Classification Report\n"+str(classification_report(y_test, p)))
132
133
```

4. **K-Nearest Neighbour:-** It is a supervised learning algorithm which assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.

   Basic Idea:- We would be given a set of records. If we want to find the class value for a record, then we would find the difference between that record with all the other given records and sort them. After that, we will select the first smallest records on the basis of sorted differences, and then select the class value on the basis of majority voting (for voting, any factor can be chosen, but generally the weight factor= $(1/(\text{difference})^2)$ is taken).

```
135
136  ## KNN
137  from sklearn.neighbors import KNeighborsClassifier
138  classifier=KNeighborsClassifier(n_neighbors=10)
139  classifier.fit(x_train,y_train)
140  p=classifier.predict(x_test)
141  acc=[]
142  for i in range(1,21):
143      classifier1=KNeighborsClassifier(n_neighbors=i)
144      classifier1.fit(x_train,y_train)
145      predicted1=classifier1.predict(x_test)
146      acc.append(accuracy_score(y_test, predicted1)*100)
147  plt.figure(figsize=(7,7))
148  plt.plot(acc)
149  plt.xticks(np.array(range(1,17)))
150  plt.title("Selection of k")
151  plt.xlabel("k for Nearest Neighbour")
152  plt.ylabel("Accuracy")
153  from sklearn.metrics import confusion_matrix
154  from sklearn.metrics import accuracy_score
155  from sklearn.metrics import classification_report
156  cm=confusion_matrix(y_test,p)
157  print(cm)
158  print("Accuracy is "+str(accuracy_score(y_test, p)*100))
159  print("Classification Report\n"+str(classification_report(y_test, p)))
160
161
```

Results

1. **For 'Review'**
    a) Logical Regression:-

```
[[25  0 11]
 [ 5  0  1]
 [ 9  0  7]]
Accuracy is 55.169999999999995
Classification Report
              precision    recall  f1-score   support

           0       0.64      0.69      0.67        36
           1       0.00      0.00      0.00         6
           2       0.37      0.44      0.40        16

    accuracy                           0.55        58
   macro avg       0.34      0.38      0.36        58
weighted avg       0.50      0.55      0.52        58
```

The result is accuracy=56.16%

b) Naive Bayes:-

```
[[14 16  6]
 [ 2  3  1]
 [ 4  8  4]]
Accuracy is 36.21
Classification Report
              precision    recall  f1-score   support

           0       0.70      0.39      0.50        36
           1       0.11      0.50      0.18         6
           2       0.36      0.25      0.30        16

    accuracy                           0.36        58
   macro avg       0.39      0.38      0.33        58
weighted avg       0.55      0.36      0.41        58
```

The result is accuracy=36.21%

c) Support Vector Machine:-

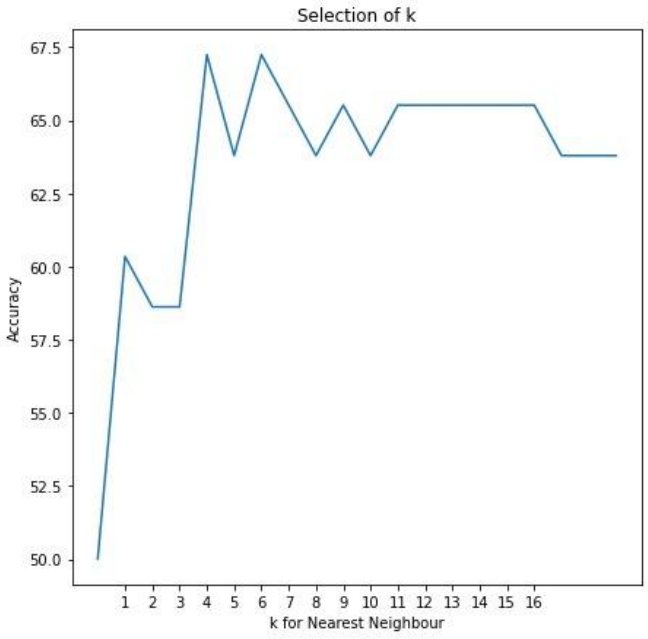
```
[[36  0  0]
 [ 5  1  0]
 [16  0  0]]
Accuracy is 63.79
Classification Report
              precision    recall  f1-score   support

           0       0.63      1.00      0.77        36
           1       1.00      0.17      0.29         6
           2       0.00      0.00      0.00        16

    accuracy                           0.64        58
   macro avg       0.54      0.39      0.35        58
weighted avg       0.50      0.64      0.51        58
```

The result is accuracy=63.79%

d) K-Nearest Neighbour:-




```
[[32  0  4]
 [ 6  0  0]
 [10  0  6]]
Accuracy is 65.51724137931035
Classification Report
              precision    recall  f1-score   support

           0       0.67      0.89      0.76        36
           1       0.00      0.00      0.00         6
           2       0.60      0.38      0.46        16

    accuracy                           0.66        58
   macro avg       0.42      0.42      0.41        58
weighted avg       0.58      0.66      0.60        58
```

The result is accuracy=65.51%

2. For ‘commerical_success’

a) Logical Regression:-

```
Prediction for commercial_success
[[31  8  0]
 [18 31  0]
 [ 4  1  0]]
Accuracy is 66.67
Classification Report
              precision    recall  f1-score   support

           0       0.58      0.79      0.67        39
           1       0.78      0.63      0.70        49
           2       0.00      0.00      0.00         5

    accuracy                           0.67        93
   macro avg       0.45      0.48      0.46        93
weighted avg       0.65      0.67      0.65        93
```

The result is accuracy=66.67%

b) Naive Bayes:-

```
[[19 15  5]
 [ 5 38  6]
 [ 1  1  3]]
Accuracy is 64.52
Classification Report
              precision    recall  f1-score   support

           0       0.76      0.49      0.59        39
           1       0.70      0.78      0.74        49
           2       0.21      0.60      0.32         5

    accuracy                           0.65        93
   macro avg       0.56      0.62      0.55        93
weighted avg       0.70      0.65      0.65        93
```

The result is accuracy=64.52%

c) Support Vector Machine:-

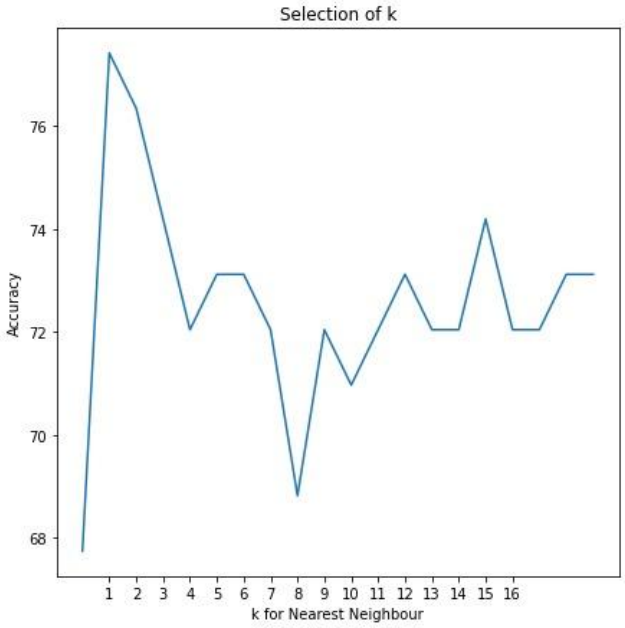
```
[[32  0  4]
 [ 6  0  0]
 [10  0  6]]
Accuracy is 65.51724137931035
Classification Report
              precision    recall  f1-score   support

           0       0.67      0.89      0.76        36
           1       0.00      0.00      0.00         6
           2       0.60      0.38      0.46        16

    accuracy                           0.66        58
   macro avg       0.42      0.42      0.41        58
weighted avg       0.58      0.66      0.60        58
```

The result is accuracy=65.51%

d) K-Nearest Neighbour:-



```
[[29 10  0]
 [11 38  0]
 [ 3  2  0]]
Accuracy is 72.04301075268818
Classification Report
              precision    recall  f1-score   support

           0       0.67      0.74      0.71        39
           1       0.76      0.78      0.77        49
           2       0.00      0.00      0.00         5

    accuracy                           0.72        93
   macro avg       0.48      0.51      0.49        93
weighted avg       0.68      0.72      0.70        93
```

The result is accuracy=72.04%

3. **For 'critical_success'**
    a) Logical Regression:-

```
[[ 8 24]
 [ 9 52]]
Accuracy is 64.52
Classification Report
                        precision    recall  f1-score   support

not supported by public       0.47      0.25      0.33        32
    supported by public       0.68      0.85      0.76        61

              accuracy                           0.65        93
             macro avg       0.58      0.55      0.54        93
          weighted avg       0.61      0.65      0.61        93
```

The result is accuracy=64.52%

b) Naive Bayes:-

```
[[23  9]
 [40 21]]
Accuracy is 47.31
Classification Report
                        precision    recall  f1-score   support

not supported by public       0.37      0.72      0.48        32
    supported by public       0.70      0.34      0.46        61

              accuracy                           0.47        93
             macro avg       0.53      0.53      0.47        93
          weighted avg       0.58      0.47      0.47        93
```

The result is accuracy=47.31%

c) Support Vector Machine:-

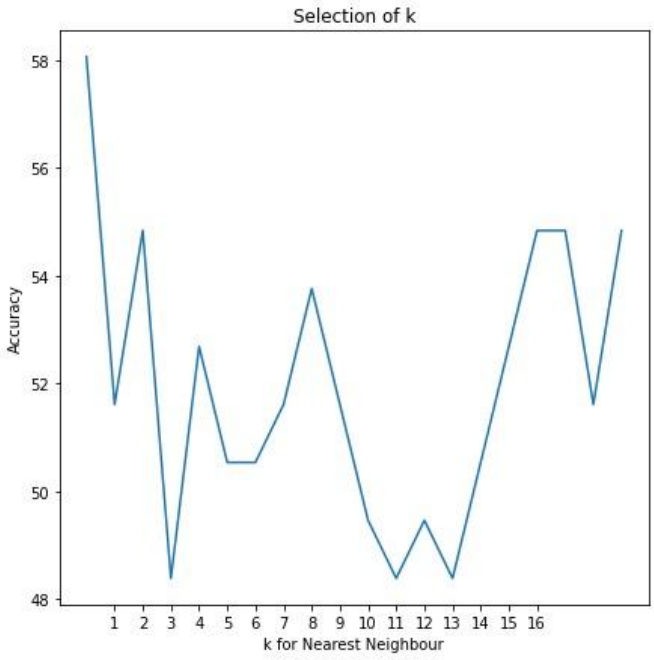
```
[[ 1 31]
 [ 0 61]]
Accuracy is 66.67
Classification Report
                        precision    recall  f1-score   support

not supported by public       1.00      0.03      0.06        32
    supported by public       0.66      1.00      0.80        61

              accuracy                           0.67        93
             macro avg       0.83      0.52      0.43        93
          weighted avg       0.78      0.67      0.54        93
```

The result is accuracy=66.67%

d) K-Nearest Neighbour:-




```
[[12 20]
 [25 36]]
Accuracy is 51.61290322580645
Classification Report
                        precision    recall  f1-score   support

not supported by public       0.32      0.38      0.35        32
    supported by public       0.64      0.59      0.62        61

               accuracy                           0.52        93
              macro avg       0.48      0.48      0.48        93
           weighted avg       0.53      0.52      0.52        93
```

The result is accuracy=51.61%

Conclusion

1. For ‘Review’:-

Algorithm Applied	Result
Logical Regression	55.169%
Naive Bayes Classification	36.21%
Support Vector Machine	63.79%
K-Nearest Neighbour	65.51%

As we can see for the ‘Review’ attribute estimation case, KNN Classifier gives the most accurate results of all four classifiers from the given dataset, and Naive Bayes Classifier gives the least accurate results.

Increasing accuracy:- Naive Bayes<Logical Regression<SVM<KNN

2. For ‘commercial_success’:-

Algorithm Applied	Result
Logical Regression	66.67%
Naive Bayes Classification	64.52%
Support Vector Machine	65.51%
K-Nearest Neighbour	72.04%

As we can see for the commercial_success attribute estimation case, KNN Classifier gives the most accurate results of all four classifiers from the given dataset, and Naive Bayes Classifier gives the least accurate results.

Increasing accuracy:- Naive Bayes<SVM<Logical Regression<KNN

3. For ‘critical_success’:-

Algorithm Applied	Result
Logical Regression	64.52%
Naive Bayes Classification	47.31%
Support Vector Machine	66.67%
K-Nearest Neighbour	51.61%

As we can see for the ‘critical_success’ attribute estimation case, KNN Classifier gives the most accurate results of all four classifiers from the given dataset, and Naive Bayes Classifier gives the least accurate results.

Increasing accuracy:- Naive Bayes<KNN:<Logical Regression<SVM

References:

- scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation (scikit-learn.org)
- An introduction to seaborn — seaborn 0.11.0 documentation (pydata.org)
- Tutorials — Matplotlib 3.1.2 documentation
- Community tutorials — pandas 1.1.4 documentation (pydata.org)
- https://www.youtube.com/watch?v=tVQb1e8Gu94&t=1928s