**Predicting the Popularity of Movies with Machine Learning Methods**

Zeynep Köse 2014510056  
Engineering Faculty *Department of Computer Engineering Dokuz Eylül University*Izmir,Turkey  
zeynep.kose@ceng.deu.edu.tr

1. **INTRODUCTION**

This article is about predicting the popularity of movies. A number of features such as cast, genre, budget, production house, and rating affect a movie's popularity. Twitter, YouTube etc. Social media are the main platforms where people can share their views on movies. Traditional features and social media features are two types of features to be used to estimate the popularity of the movie. The estimate of popularity can be measured in terms of Ratings. (Represented by a positive numeric number less than 10 or a label).

1. **DATASET**
2. **Description**

Data Name: CSM (Conventional and Social Media Movies) Dataset 2014 and 2015 Data Set

Data Link: <https://archive.ics.uci.edu/ml/datasets/CSM+%28Conventional+and+Social+Media+Movies%29+Dataset+2014+and+2015>

The dataset retrieved information about movies from diverse sources including movies web site, i.e. IMDB, generic web resource i.e. Wikipedia, and social media including YouTube and Twitter. Beyond that, it also used sentiment analysis libraries to get the sentiment score for different movies. The total dataset contains twelve features and can be split in to two sub-dataset, the conventional features and social media features.

1. Conventional Features: Conventional Features contain six features in total and those features are typically available on movie resource websites, such as IMDB.

• Genre: There are 19 different types of genre in the dataset, such as Action, Adventure and Drama etc. They were already mapped on to integer value from 1-19 and in our project, they are treated as factor variables to represent different genre.

• Sequel: This variable in integer represents whether the movie is sequel or individual. 1 shows that movie is first release; other n larger than 1 shows that movie is 2nd. e.g. Pirates of Caribbean: Dead Man’s Chest is 2nd in sequel, therefore it is assigned the value of 2.

• Ratings: The value of Ratings ranges between 1 to 10 with 1 being lowest and 10 the highest. These values are collected from IMDB.

• Gross Income, Budget and Number of Screens: Gross world-wide income and Budget for each movie is collected from IMDB. The unit of gross income and budget is USD and they are already converted into USD if they are represented in other currencies. Number of screens on which movie was initially launched in US is also considered.

1. Social Media Features: Social Media Features also contains six features and those features are collected for each movie.

• Aggregate Actor Followers: Number of followers of actors in one movie on twitter is used. Only the top 3 in cast are considered.

• Number of Views and Comments: Those variables represent the number of views and comments of trailer of movies on YouTube.

• Number of likes and dislikes: Number of Likes and Dislikes of trailers on YouTube are considered.

• Sentiment Score: A signed integer value is used to represent sentiment score. 0 represents neutral sentiment; “+”sign shows the positive sentiment and the value shows the magnitude; “–”sign shows negative sentiment and the value shows the magnitude. The sentiment score is calculated through analysing the sentiments of tweets about one movie.

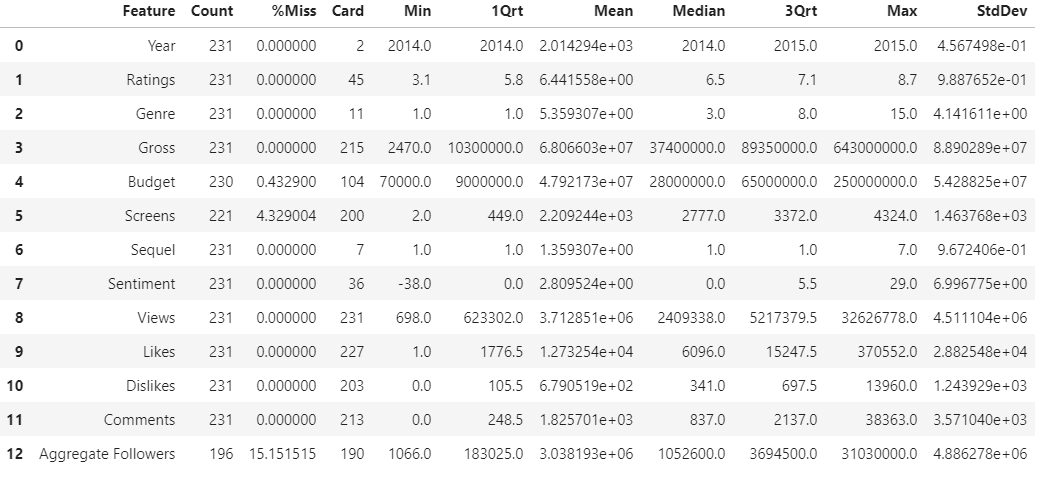
1. **Plots**

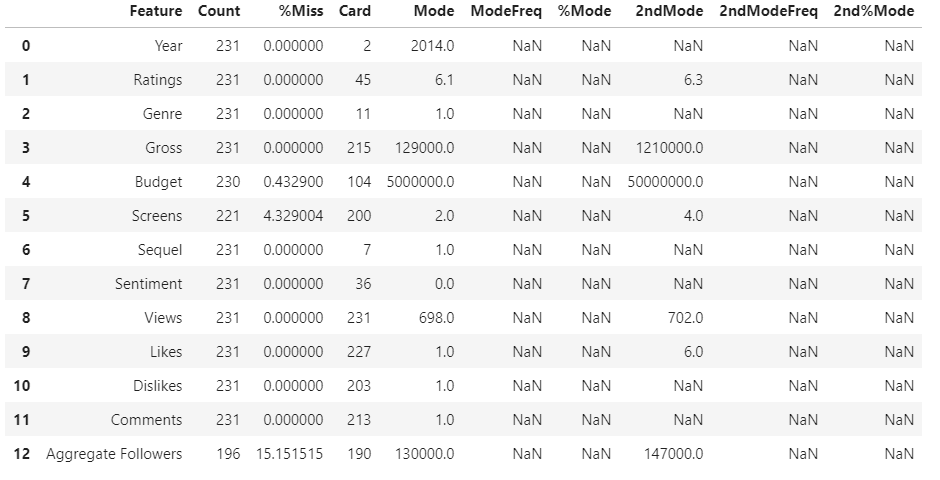
A data quality report includes tabular reports that describe the characteristics of each feature in an ABT using standard statistical measures of central tendency and variation.

The tabular reports are accompanied by data visualizations:

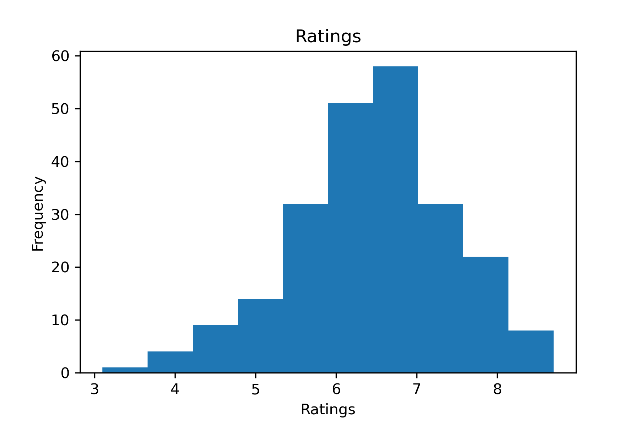
* A histogram for each continuous feature in an ABT.
* A bar plot for each categorical feature in an ABT.

Continuous Features:



Categorical Features: 

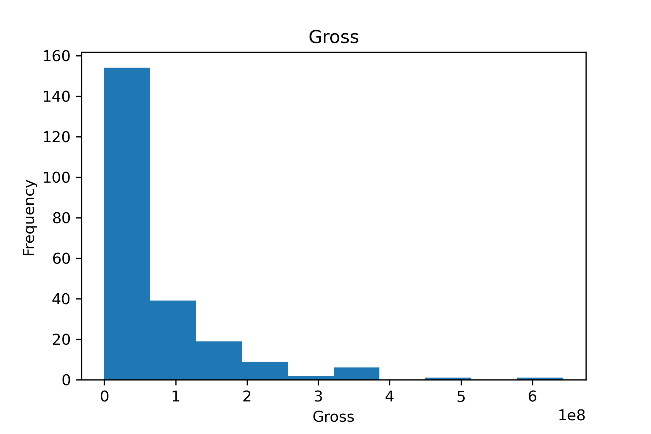
1. Ratings



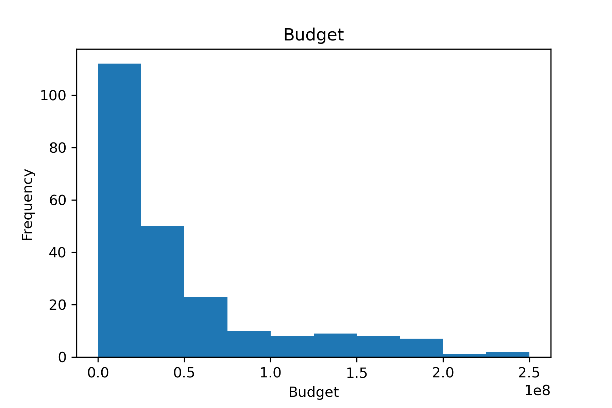
1. Genre



1. Gross



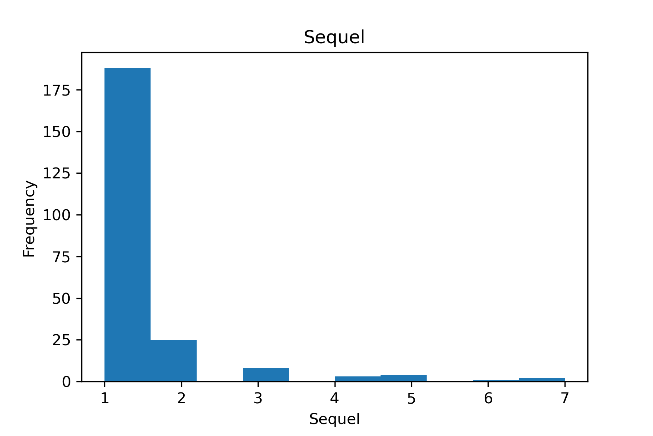
1. Budget



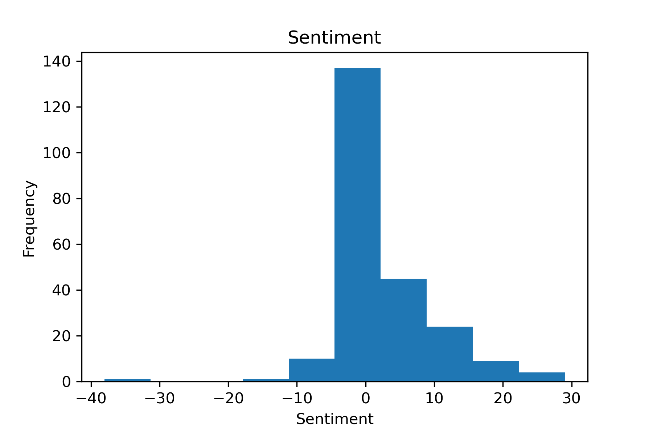
1. Screens



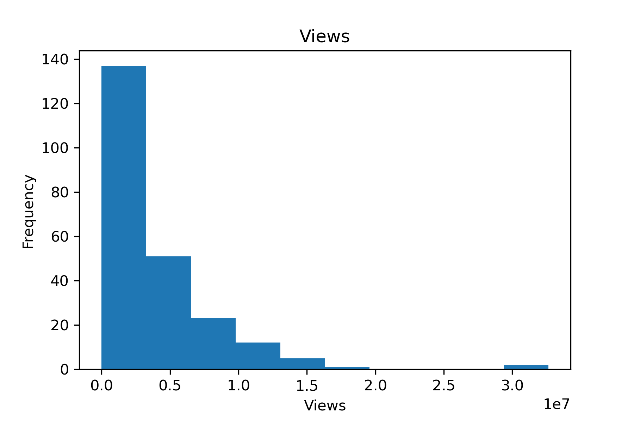
1. Sequel



1. Sentiment



1. Views



1. Likes



1. Dislikes



1. Comments



1. Aggregate



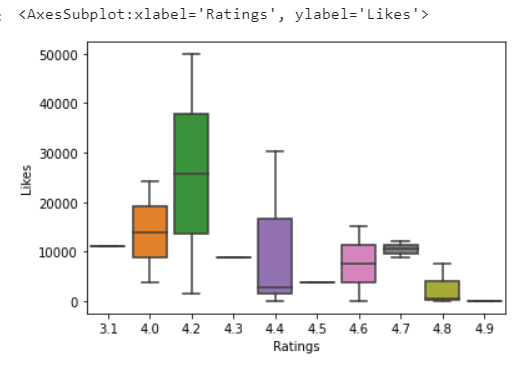
1. **Data Preprocess**

Table I shows that how we map the Rating to category variable. For methods exclude regression methods, I will use this categorical variable as dependent variable in models and predictions.

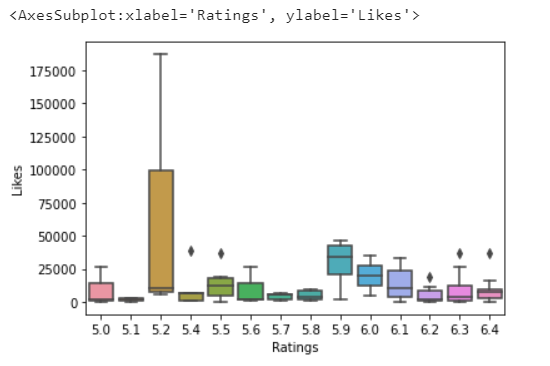
|  |  |
| --- | --- |
| Ratings | Assigned Label |
| 0-4.9 | Poor |
| 5-6.4 | Average |
| 6.5-8 | Good |
| 8-10 | Excellent |

Table I

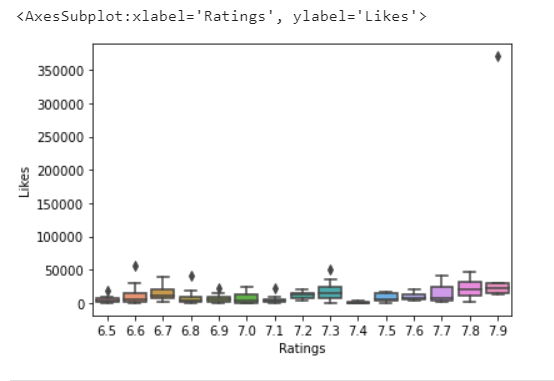
1. 0 – 4.9 Poor



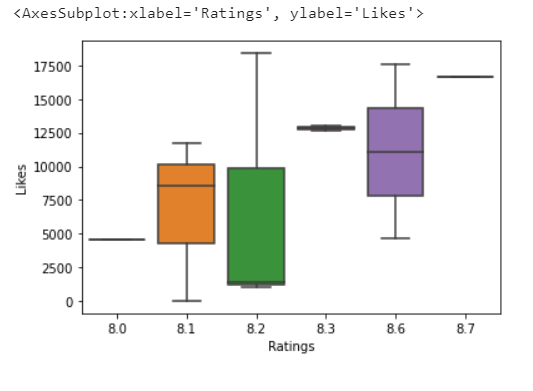
1. 5 - 6.4 Average



1. 6.5 – 8 Good



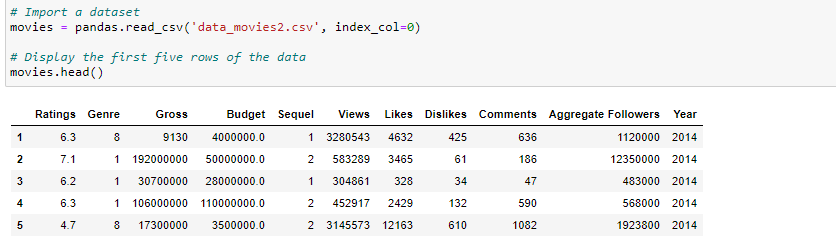
1. 8 – 10 Excellent



From the graphs taken from these numbers, we can learn that the higher the 'Likes' numbers, the higher the likelihood that a movie will be in a higher level, as well as some features can help differentiate rating levels; For other properties, we need to investigate how these properties can be used to estimate the tag of a movie's ratings.

1. **Machine Learning Models**
2. **Regression Model**

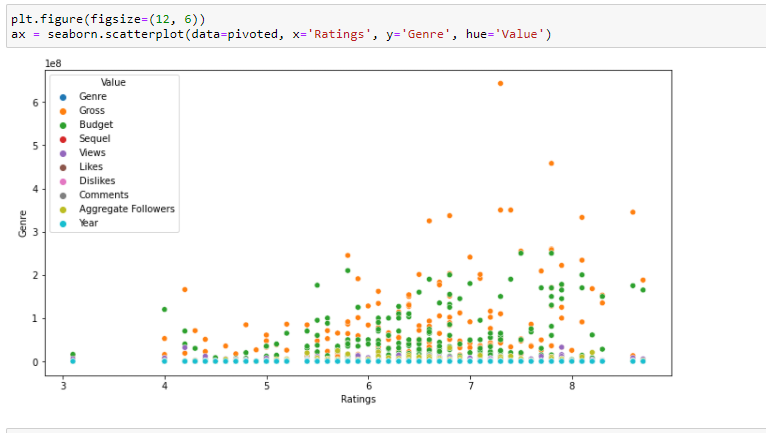
All codes are in Regression\_2014\_2015 notebook.

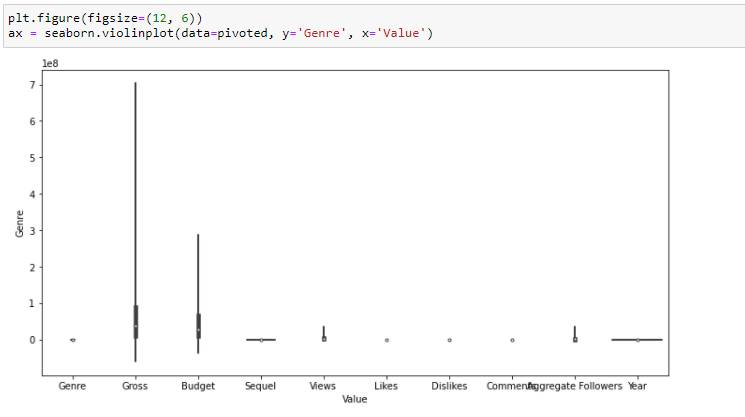


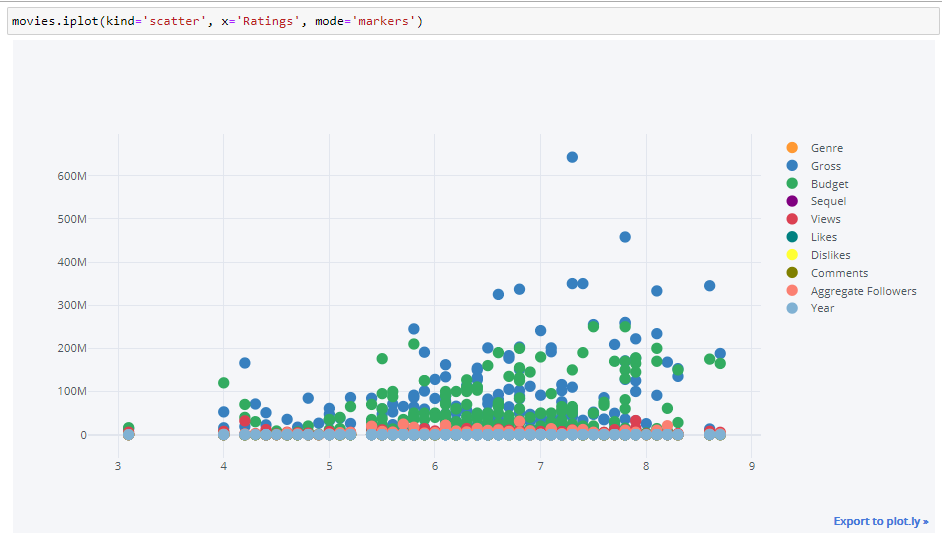
## Taking a look at the data

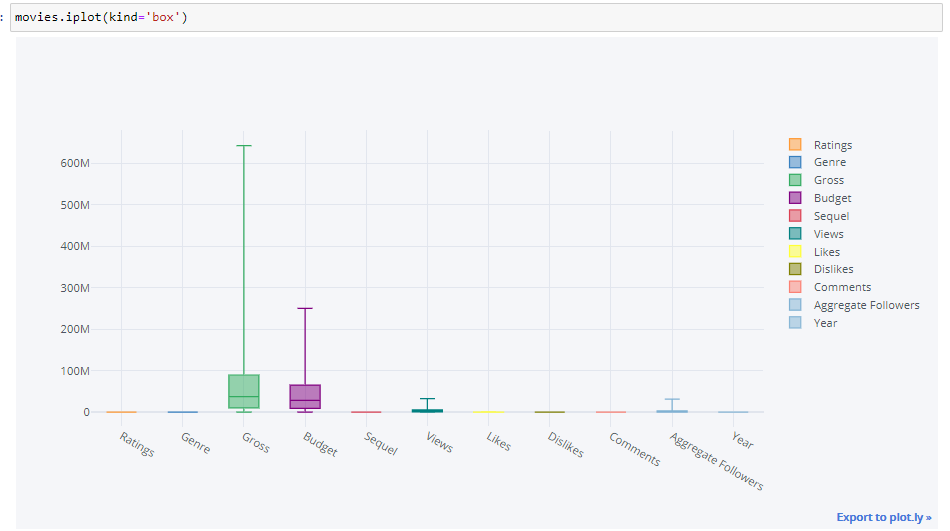
I can quickly visualise the relationships in the data.

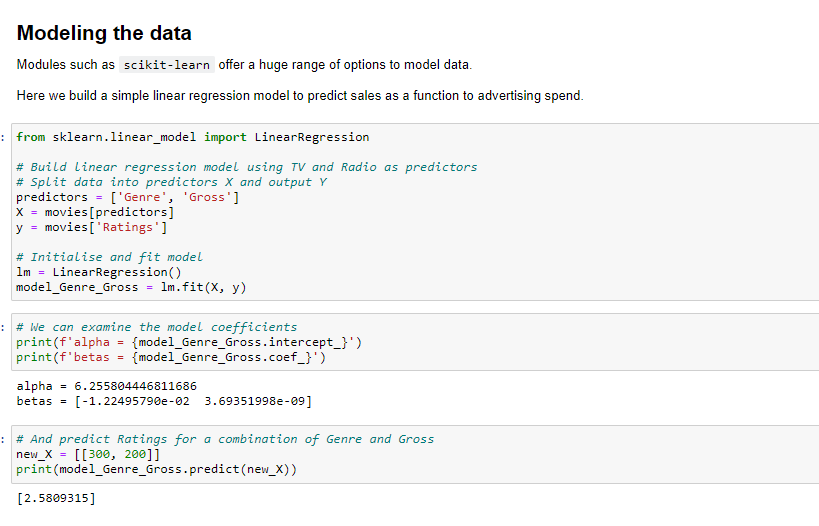
The seaborn module provides many common plots.

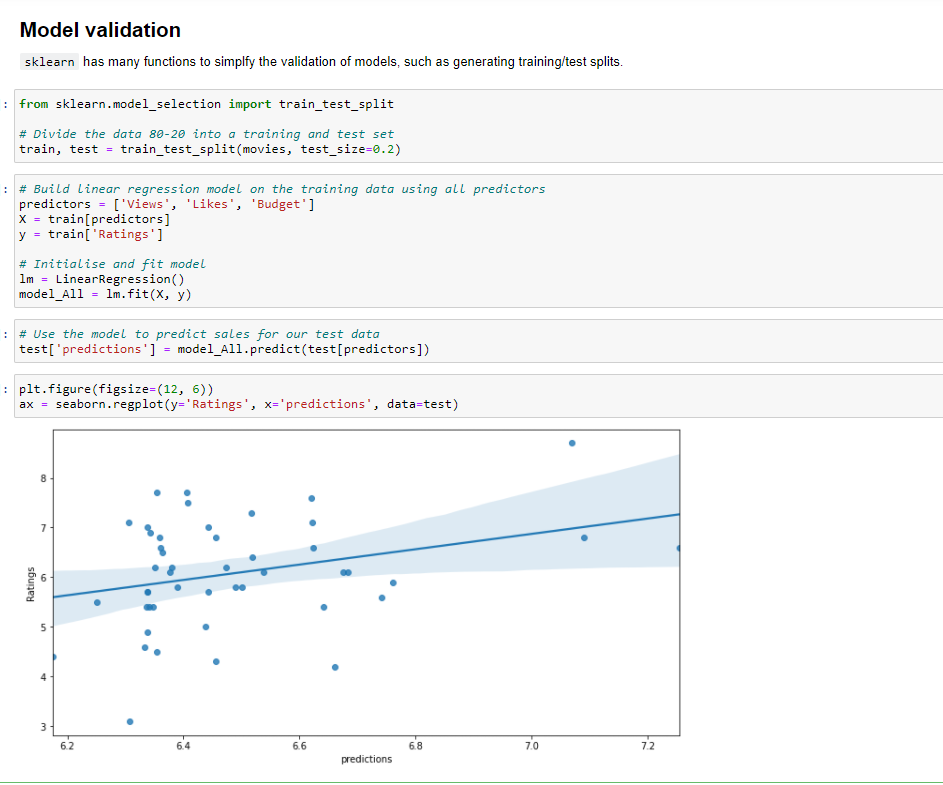


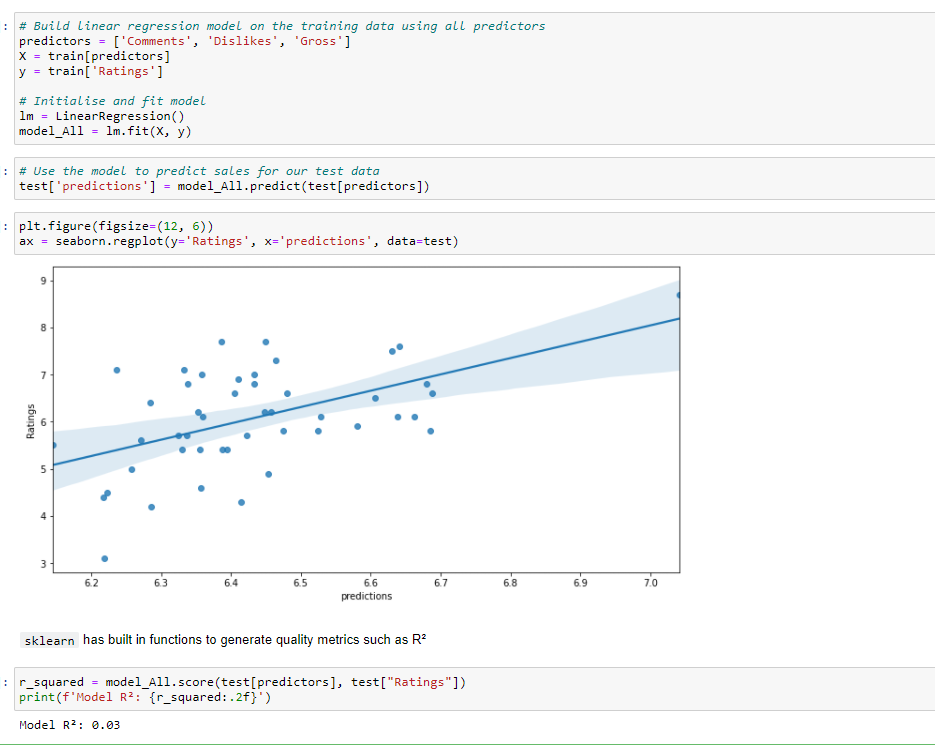






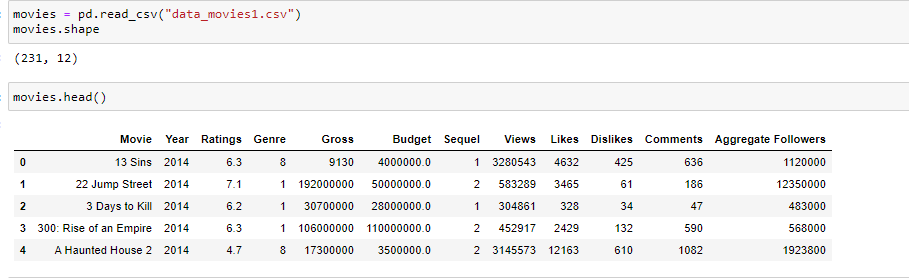






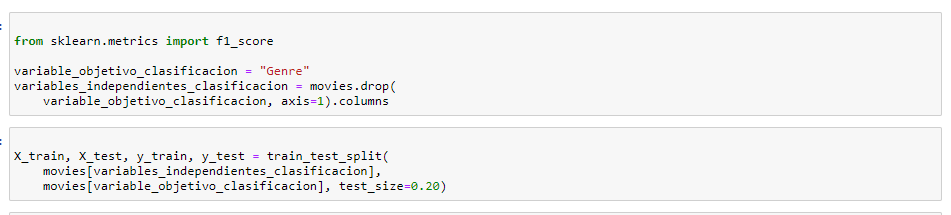
1. **KNN Model**

All codes are in KNN\_2014\_2015 notebook.



**KNN for classification problems**

I tested KNN for classification, specifically let's assume that we want to predict the genre of a movie based on its popularity.

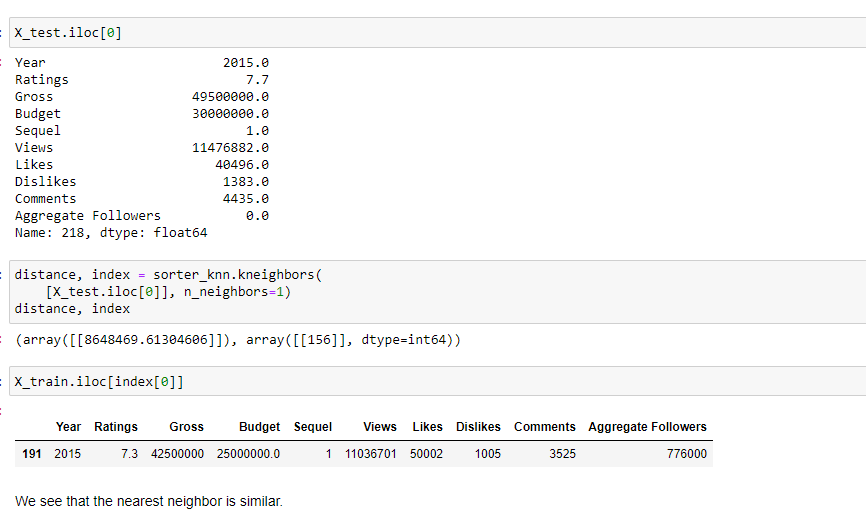


The most important parameters when using KNeighborsClasifier are:

* n\_neighbors: The value of K, that is, the number of neighbors to consider when assigning a class.
* weights: When it comes to voting, how important is it to give to the neighbors. If we choose car, it assigns the same importance to all the neighbors. If we choose distance, it assigns importance to the neighbors based on the distance from the - neighbors to the point to be classified
* metric: The metric when measuring the distance between the points. If Minkowsky distance is used, p can be chosen with the parameter p, which by default is 2 (which computes the Euclidean distance).

In this particular case we know what value to choose from K, since we can assume that the number of categories in the dataset is the total number of movie categories in the training dataset.

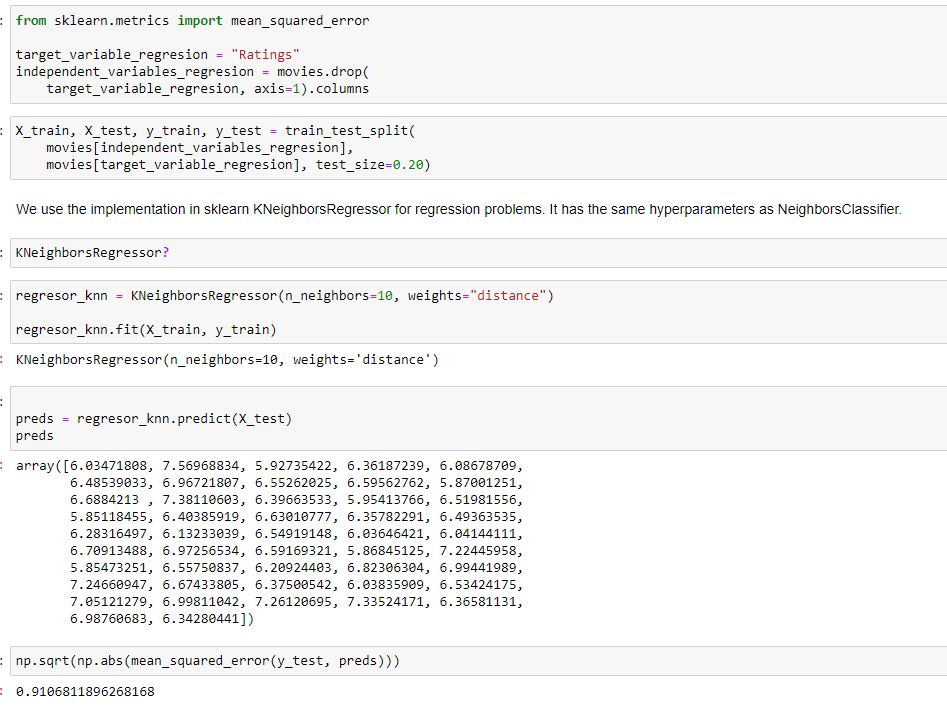


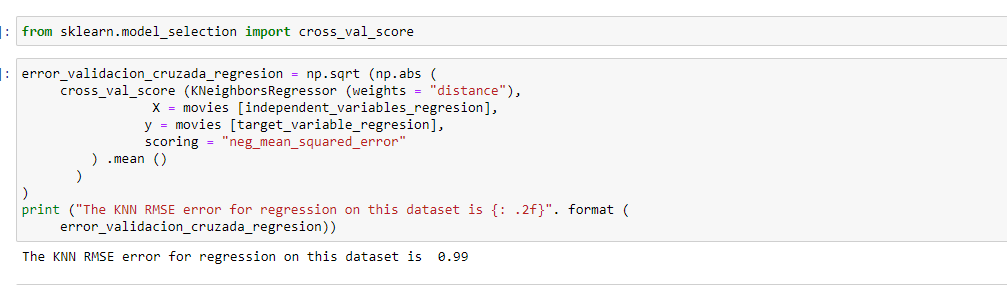


# KNN for regression problems

We are now going to use the KNN algorithm for a regression problem, KNN works the same to do regressions, simply that instead of a vote where the most common class among the closest neighbors is the chosen one, an interpolation of the values of the Objective numerical variable of the neighbors.

Specifically, we will estimate a movie's rating.





APPENDIX A: Continuous and Categorical Features Code

#2014510056- ZEYNEP KOSE

# Libraries  
import numpy as np  
import numpy.matlib  
import matplotlib.pyplot as plt  
import pandas as pd  
import math  
import os #look for directory  
%matplotlib inline

# Import database  
  
data = pd.read\_csv('data.csv')  
  
data.head(1)

Year Ratings Genre Gross Budget Screens Sequel Sentiment \  
0 2014 6.3 8 9130 4000000.0 45.0 1 0   
  
 Views Likes Dislikes Comments Aggregate Followers   
0 3280543 4632 425 636 1120000.0

The structures of the tables included in a data quality report to describe

1) Continuous features

2) Categorical features

#It is split into two data frames with different property types:  
#Continuous  
dataCont=data  
#Categorical  
dataCat=data

# Create "Continuous features" for Quality Report

QRcontinue = pd.DataFrame(columns = ['Feature','Count','%Miss','Card','Min','1Qrt','Mean','Median','3Qrt','Max','StdDev'])  
QRcontinue

Empty DataFrame  
Columns: [Feature, Count, %Miss, Card, Min, 1Qrt, Mean, Median, 3Qrt, Max, StdDev]  
Index: []

#Feature names  
QRcontinue['Feature']=list(dataCont.columns)  
#Count Values (data - NaN)  
QRcontinue['Count']=list(dataCont.count(axis=0))  
# missing values  
QRcontinue['%Miss']=list(dataCont.isnull().sum()/len(dataCont)\*100)  
#Cardinality: number of different values  
QRcontinue['Card']=list(dataCont.apply(pd.Series.nunique))  
#Minimum value  
QRcontinue['Min']=list(dataCont.min(axis=0))  
# 1st quartile  
QRcontinue['1Qrt']=list(dataCont.quantile(q=0.25,axis=0)) #0<=q<=1 25%percentil=0.25q  
#Average  
QRcontinue['Mean']= list(dataCont.mean(axis=0))  
#Median  
QRcontinue['Median']= list(dataCont.median(axis=0))  
# 3rd quartile  
QRcontinue['3Qrt']=list(dataCont.quantile(q=0.75,axis=0)) #0<=q<=1 25%percentil=0.25q  
#Maximum value  
QRcontinue['Max']=list(dataCont.max(axis=0))  
#Standard deviation  
QRcontinue['StdDev']=list(dataCont.std(axis=0))

QRcontinue

Feature Count %Miss Card Min 1Qrt \  
0 Year 231 0.000000 2 2014.0 2014.0   
1 Ratings 231 0.000000 45 3.1 5.8   
2 Genre 231 0.000000 11 1.0 1.0   
3 Gross 231 0.000000 215 2470.0 10300000.0   
4 Budget 230 0.432900 104 70000.0 9000000.0   
5 Screens 221 4.329004 200 2.0 449.0   
6 Sequel 231 0.000000 7 1.0 1.0   
7 Sentiment 231 0.000000 36 -38.0 0.0   
8 Views 231 0.000000 231 698.0 623302.0   
9 Likes 231 0.000000 227 1.0 1776.5   
10 Dislikes 231 0.000000 203 0.0 105.5   
11 Comments 231 0.000000 213 0.0 248.5   
12 Aggregate Followers 196 15.151515 190 1066.0 183025.0   
  
 Mean Median 3Qrt Max StdDev   
0 2.014294e+03 2014.0 2015.0 2015.0 4.567498e-01   
1 6.441558e+00 6.5 7.1 8.7 9.887652e-01   
2 5.359307e+00 3.0 8.0 15.0 4.141611e+00   
3 6.806603e+07 37400000.0 89350000.0 643000000.0 8.890289e+07   
4 4.792173e+07 28000000.0 65000000.0 250000000.0 5.428825e+07   
5 2.209244e+03 2777.0 3372.0 4324.0 1.463768e+03   
6 1.359307e+00 1.0 1.0 7.0 9.672406e-01   
7 2.809524e+00 0.0 5.5 29.0 6.996775e+00   
8 3.712851e+06 2409338.0 5217379.5 32626778.0 4.511104e+06   
9 1.273254e+04 6096.0 15247.5 370552.0 2.882548e+04   
10 6.790519e+02 341.0 697.5 13960.0 1.243929e+03   
11 1.825701e+03 837.0 2137.0 38363.0 3.571040e+03   
12 3.038193e+06 1052600.0 3694500.0 31030000.0 4.886278e+06

# Create "Categorical features" for Quality Report

#"Quality report"  
QRcategorical = pd.DataFrame(columns = ['Feature','Count','%Miss','Card','Mode','ModeFreq','%Mode','2ndMode','2ndModeFreq','2nd%Mode'])  
QRcategorical

Empty DataFrame  
Columns: [Feature, Count, %Miss, Card, Mode, ModeFreq, %Mode, 2ndMode, 2ndModeFreq, 2nd%Mode]  
Index: []

#Feature names  
QRcategorical['Feature']=list(dataCat.columns)  
#Count Values (data - NaN)  
QRcategorical['Count']=list(dataCat.count(axis=0))  
# missing values  
QRcategorical['%Miss']=list(dataCat.isnull().sum()/len(dataCat)\*100)  
#Cardinality: number of different values  
QRcategorical['Card']=list(dataCat.apply(pd.Series.nunique))  
#Moda  
QRcategorical['Mode']=list(dataCat.mode(axis=0).iloc[0])  
#Frequency of mode  
  
#Percentage of mode  
  
# 2nd mode  
QRcategorical['2ndMode']=list(dataCat.mode(axis=0).iloc[1])  
# 2nd mode frequency  
  
#Percentage of 2nd mode

QRcategorical

Feature Count %Miss Card Mode ModeFreq %Mode \  
0 Year 231 0.000000 2 2014.0 NaN NaN   
1 Ratings 231 0.000000 45 6.1 NaN NaN   
2 Genre 231 0.000000 11 1.0 NaN NaN   
3 Gross 231 0.000000 215 129000.0 NaN NaN   
4 Budget 230 0.432900 104 5000000.0 NaN NaN   
5 Screens 221 4.329004 200 2.0 NaN NaN   
6 Sequel 231 0.000000 7 1.0 NaN NaN   
7 Sentiment 231 0.000000 36 0.0 NaN NaN   
8 Views 231 0.000000 231 698.0 NaN NaN   
9 Likes 231 0.000000 227 1.0 NaN NaN   
10 Dislikes 231 0.000000 203 1.0 NaN NaN   
11 Comments 231 0.000000 213 1.0 NaN NaN   
12 Aggregate Followers 196 15.151515 190 130000.0 NaN NaN   
  
 2ndMode 2ndModeFreq 2nd%Mode   
0 NaN NaN NaN   
1 6.3 NaN NaN   
2 NaN NaN NaN   
3 1210000.0 NaN NaN   
4 50000000.0 NaN NaN   
5 4.0 NaN NaN   
6 NaN NaN NaN   
7 NaN NaN NaN   
8 702.0 NaN NaN   
9 6.0 NaN NaN   
10 NaN NaN NaN   
11 NaN NaN NaN   
12 147000.0 NaN NaN

# Easiest way with enough info (for numerical data)

data.describe()

Year Ratings Genre Gross Budget \  
count 231.000000 231.000000 231.000000 2.310000e+02 2.300000e+02   
mean 2014.294372 6.441558 5.359307 6.806603e+07 4.792173e+07   
std 0.456750 0.988765 4.141611 8.890289e+07 5.428825e+07   
min 2014.000000 3.100000 1.000000 2.470000e+03 7.000000e+04   
25% 2014.000000 5.800000 1.000000 1.030000e+07 9.000000e+06   
50% 2014.000000 6.500000 3.000000 3.740000e+07 2.800000e+07   
75% 2015.000000 7.100000 8.000000 8.935000e+07 6.500000e+07   
max 2015.000000 8.700000 15.000000 6.430000e+08 2.500000e+08   
  
 Screens Sequel Sentiment Views Likes \  
count 221.000000 231.000000 231.000000 2.310000e+02 231.000000   
mean 2209.244344 1.359307 2.809524 3.712851e+06 12732.536797   
std 1463.767755 0.967241 6.996775 4.511104e+06 28825.484481   
min 2.000000 1.000000 -38.000000 6.980000e+02 1.000000   
25% 449.000000 1.000000 0.000000 6.233020e+05 1776.500000   
50% 2777.000000 1.000000 0.000000 2.409338e+06 6096.000000   
75% 3372.000000 1.000000 5.500000 5.217380e+06 15247.500000   
max 4324.000000 7.000000 29.000000 3.262678e+07 370552.000000   
  
 Dislikes Comments Aggregate Followers   
count 231.000000 231.000000 1.960000e+02   
mean 679.051948 1825.701299 3.038193e+06   
std 1243.929481 3571.040447 4.886278e+06   
min 0.000000 0.000000 1.066000e+03   
25% 105.500000 248.500000 1.830250e+05   
50% 341.000000 837.000000 1.052600e+06   
75% 697.500000 2137.000000 3.694500e+06   
max 13960.000000 38363.000000 3.103000e+07

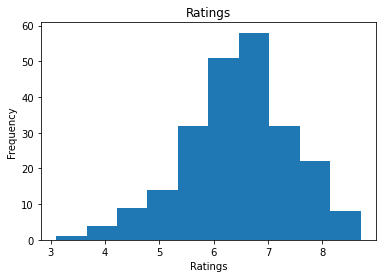
APPENDIX B: Histograms and Box-Plot Code

import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
import seaborn as sns

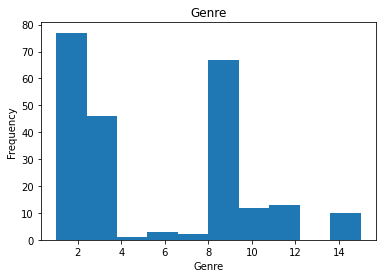
csv\_dataset = pd.read\_csv('2014\_and\_2015\_CSM\_dataset.csv')  
csv\_dataset.head(5)

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

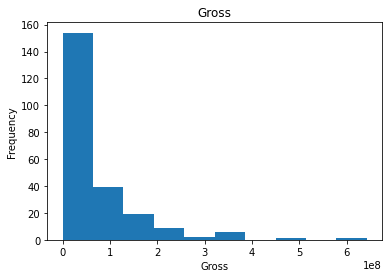
# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Ratings'])  
# set title and labels  
ax.set\_title('Ratings')  
ax.set\_xlabel('Ratings')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_ratings.png', dpi=300)



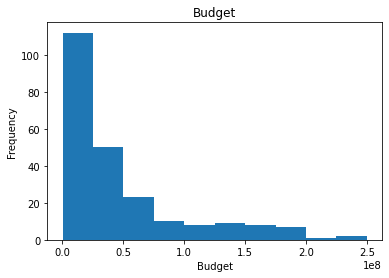
# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Genre'])  
# set title and labels  
ax.set\_title('Genre')  
ax.set\_xlabel('Genre')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_genre.png', dpi=300)



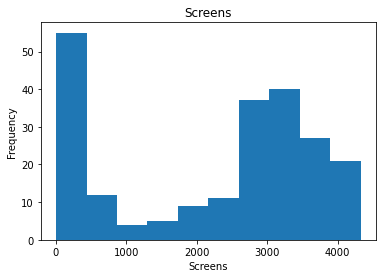
# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Gross'])  
# set title and labels  
ax.set\_title('Gross')  
ax.set\_xlabel('Gross')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_gross.png', dpi=300)



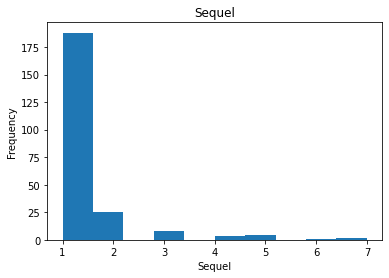
# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Budget'])  
# set title and labels  
ax.set\_title('Budget')  
ax.set\_xlabel('Budget')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_budget.png', dpi=300)



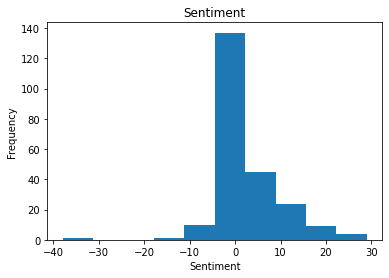
# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Screens'])  
# set title and labels  
ax.set\_title('Screens')  
ax.set\_xlabel('Screens')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_screens.png', dpi=300)



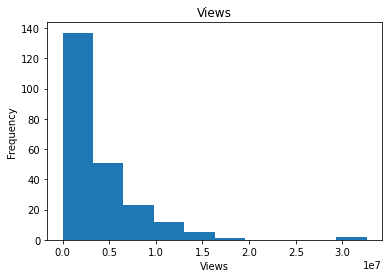
# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Sequel'])  
# set title and labels  
ax.set\_title('Sequel')  
ax.set\_xlabel('Sequel')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_sequel.png', dpi=300)



# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Sentiment'])  
# set title and labels  
ax.set\_title('Sentiment')  
ax.set\_xlabel('Sentiment')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_sentiment.png', dpi=300)



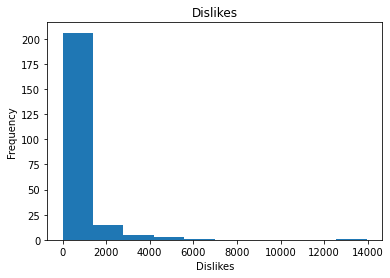
# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Views'])  
# set title and labels  
ax.set\_title('Views')  
ax.set\_xlabel('Views')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_views.png', dpi=300)



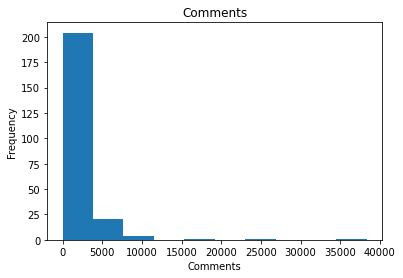
# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Likes'])  
# set title and labels  
ax.set\_title('Likes')  
ax.set\_xlabel('Likes')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_likes.png', dpi=300)



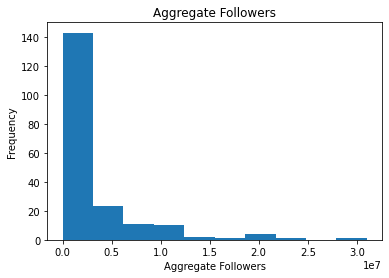
# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Dislikes'])  
# set title and labels  
ax.set\_title('Dislikes')  
ax.set\_xlabel('Dislikes')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_dislikes.png', dpi=300)



# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Comments'])  
# set title and labels  
ax.set\_title('Comments')  
ax.set\_xlabel('Comments')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_comments.png', dpi=300)



# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset['Aggregate Followers'])  
# set title and labels  
ax.set\_title('Aggregate Followers')  
ax.set\_xlabel('Aggregate Followers')  
ax.set\_ylabel('Frequency')  
plt.savefig('histogram\_aggregate.png', dpi=300)

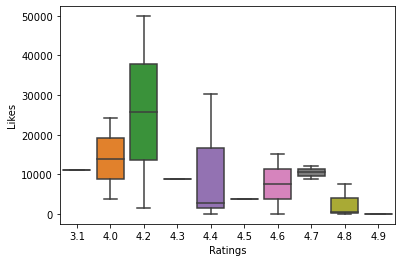


# create figure and axis  
fig, ax = plt.subplots()  
# plot histogram  
ax.hist(csv\_dataset[''])  
# set title and labels  
ax.set\_title('')  
ax.set\_xlabel('')  
ax.set\_ylabel('Frequency')

df = csv\_dataset[(csv\_dataset['Ratings']>=0) & (csv\_dataset['Ratings']<5)]  
sns.boxplot('Ratings', 'Likes', data=df)

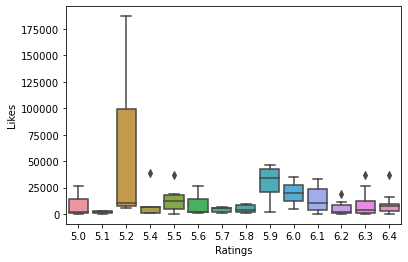
C:\Users\zeyne\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
 warnings.warn(

<AxesSubplot:xlabel='Ratings', ylabel='Likes'>



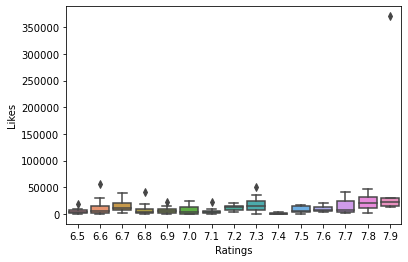
df = csv\_dataset[(csv\_dataset['Ratings']>=5) & (csv\_dataset['Ratings']<6.5)]  
sns.boxplot('Ratings', 'Likes', data=df)

<AxesSubplot:xlabel='Ratings', ylabel='Likes'>



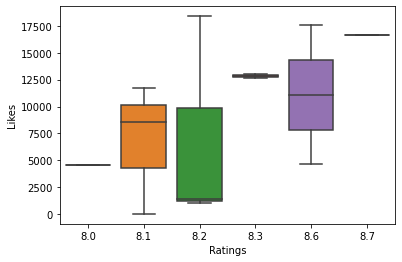
df = csv\_dataset[(csv\_dataset['Ratings']>=6.5) & (csv\_dataset['Ratings']<8)]  
sns.boxplot('Ratings', 'Likes', data=df)

<AxesSubplot:xlabel='Ratings', ylabel='Likes'>



df = csv\_dataset[(csv\_dataset['Ratings']>=8) & (csv\_dataset['Ratings']<10)]  
sns.boxplot('Ratings', 'Likes', data=df)

<AxesSubplot:xlabel='Ratings', ylabel='Likes'>



df = csv\_dataset[(csv\_dataset['Ratings']>=8) & (csv\_dataset['Ratings']<10)]  
sns.boxplot('Ratings', 'Gross', data=df)

<AxesSubplot:xlabel='Ratings', ylabel='Gross'>

