Project: Investigate The Movie Database (TMDb)

February 25, 2019

1 Project: Investigate The Movie Database (TMDb)

--by Lu Tang

1.1 Table of Contents

Introduction

Dataset: This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue. Data can be download fromhere. The final two columns ending with "_adj" show the budget and revenue of the associated movie in terms of 2010 dollars, accounting for inflation over time.

The project aims to explore the following questions: - Question 1: What are the trend for movie industry? Are movie industry making more money over years - Question 2: Are newer movies more popular? - Question 3: What kinds of properties are associated with movies that have high revenues? - Question 4. Is it possible to make extremely high profit movies with low budget? - Question 5: What are the top 10 rated movies? and how is their profitibility?

```
In [36]: # import library that will be used in this project
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         %matplotlib inline
```

Data Wrangling

1.1.1 General Properties

```
In [37]: # loading data
         tmdb=pd.read_csv('tmdb-movies.csv')
         # show number of rows and columns
         print(tmdb.shape)
```

```
# to avoid truncated output
pd.options.display.max_columns = 150
# show first 2 rows
tmdb.head(2)
```

```
(10866, 21)
```

Out[37]:

```
       id    imdb_id  popularity     budget     revenue      original_title  \
0  135397  tt0369610   32.985763  150000000  1513528810      Jurassic World
1   76341  tt1392190   28.419936  150000000   378436354  Mad Max: Fury Road

                                                cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...

                        homepage          director             tagline  \
0  http://www.jurassicworld.com/  Colin Trevorrow   The park is open.
1    http://www.madmaxmovie.com/    George Miller  What a Lovely Day.

                                            keywords  \
0  monster|dna|tyrannosaurus rex|velociraptor|island
1   future|chase|post-apocalyptic|dystopia|australia

                                            overview  runtime  \
0  Twenty-two years after the events of Jurassic ...      124
1  An apocalyptic story set in the furthest reach...      120

                                      genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller

                                production_companies release_date  vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15        5562
1  Village Roadshow Pictures|Kennedy Miller Produ...      5/13/15        6185

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
```

Initial observation: - Our focus will be analyzing movie properties associated with high revenue, some columns are irrelavant for our analysis, e.g `id`,`imdb_id`, `homepage`, `tagline`, `keywords`, `overview`, `production_companies`, `release_date` (since we already have `release_year`). - we can also remove `budget` and `revenue`, since we have `budget_adj` and `revenue_adj` to analyze.

In [38]:

```
# Drop extraneous columns
drop_col=['id','imdb_id','homepage','tagline','keywords','overview','production_compa
tmdb = tmdb.drop(drop_col, axis=1)
```

```
# check the result
tmdb.head(1)
```

Out[38]:

```
   popularity original_title  \
0   32.985763 Jurassic World

                                                cast          director  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...  Colin Trevorrow

   runtime                                      genres  vote_count  \
0      124  Action|Adventure|Science Fiction|Thriller        5562

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
```

In [39]:

```
# check data type and missing values
tmdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 11 columns):
popularity        10866 non-null float64
original_title    10866 non-null object
cast              10790 non-null object
director          10822 non-null object
runtime           10866 non-null int64
genres            10843 non-null object
vote_count        10866 non-null int64
vote_average      10866 non-null float64
release_year      10866 non-null int64
budget_adj        10866 non-null float64
revenue_adj       10866 non-null float64
dtypes: float64(4), int64(3), object(4)
memory usage: 933.9+ KB
```

In [40]:

```
# check statistical information
tmdb.describe(include='all')
```

Out[40]:

```
          popularity original_title        cast     director       runtime  \
count   10866.000000          10866       10790        10822  10866.000000
unique           NaN          10571       10719         5067           NaN
top              NaN         Hamlet  Louis C.K.  Woody Allen           NaN
freq             NaN              4           6           45           NaN
mean        0.646441            NaN         NaN          NaN    102.070863
std         1.000185            NaN         NaN          NaN     31.381405
min         0.000065            NaN         NaN          NaN      0.000000
25%         0.207583            NaN         NaN          NaN     90.000000
50%         0.383856            NaN         NaN          NaN     99.000000
```

```
75%       0.713817            NaN         NaN          NaN    111.000000
max      32.985763            NaN         NaN          NaN    900.000000

        genres    vote_count  vote_average  release_year    budget_adj  \
count    10843  10866.000000  10866.000000  10866.000000  1.086600e+04
unique    2039           NaN           NaN           NaN           NaN
top      Drama           NaN           NaN           NaN           NaN
freq       712           NaN           NaN           NaN           NaN
mean       NaN    217.389748      5.974922   2001.322658  1.755104e+07
std        NaN    575.619058      0.935142     12.812941  3.430616e+07
min        NaN     10.000000      1.500000   1960.000000  0.000000e+00
25%        NaN     17.000000      5.400000   1995.000000  0.000000e+00
50%        NaN     38.000000      6.000000   2006.000000  0.000000e+00
75%        NaN    145.750000      6.600000   2011.000000  2.085325e+07
max        NaN   9767.000000      9.200000   2015.000000  4.250000e+08

         revenue_adj
count   1.086600e+04
unique           NaN
top              NaN
freq             NaN
mean    5.136436e+07
std     1.446325e+08
min     0.000000e+00
25%     0.000000e+00
50%     0.000000e+00
75%     3.369710e+07
max     2.827124e+09
```

Insights: - Some columns contain NaN values, but the amount is not significant; we don't need to drop all the nulls at the beginning. - Data type are all correct. - The minimum `runtime` is 0, which is impossible, and some movies have extremely long runtime, we will investigate the outlier data - `budget_adj` and `revenue_adj` have minimum and median value as 0 too, which is odd, and the difference from 75% to maximum is huge, we need to investigate in the later anaysis process - `popularity` , `vote_count` has very uneven distribution, with some extreme high value data.

1.1.2 Data Cleaning

This dataseat is generally clean, column names are also clear and with preferred snakecase. For some string columns that contains '|', we will clean and analyze in the later part specific to the question we want to answer.

1. Remove duplicated data

In [41]:
```
# check how many rows are duplicated
sum(tmdb.duplicated())
```

Out[41]: 1

```
In [42]: # Drop duplicated rows
         tmdb.drop_duplicates(inplace=True)

         # douch check the results
         sum(tmdb.duplicated())
```

```
Out[42]: 0
```

2. Cleaning abnormal data for runtime

```
In [43]: # Find out how many rows are 0 for runtime
         sum(tmdb["runtime"]==0)
```

```
Out[43]: 31
```

```
In [44]: # Since it is impossible to have runtime as 0, we will remove these.
         tmdb=tmdb[tmdb["runtime"]>0]

         #double check the result
         sum(tmdb["runtime"]==0)
```

```
Out[44]: 0
```

3. Cleaning abnormal data for budget

```
In [45]: sum(tmdb["budget_adj"]==0)
```

```
Out[45]: 5668
```

```
In [46]: # It is impossible to make a movie without any budget, we will remove these data
         tmdb=tmdb[tmdb["budget_adj"]>0]

         # Double check the result
         sum(tmdb["budget_adj"]==0)
```

```
Out[46]: 0
```

4. Cleaning abnormal data for revenue

```
In [47]: sum(tmdb["revenue_adj"]==0)
```

```
Out[47]: 1312
```

```
In [48]: # It is impossible to make a movie without any budget, we will remove these data
         tmdb=tmdb[tmdb["revenue_adj"]>0]

         # Double check the result
         sum(tmdb["revenue_adj"]==0)
```

```
Out[48]: 0
```

```
In [49]: # Double check the cleaning result
         print(tmdb.shape)
         tmdb.head(1)
```

```
(3854, 11)
```

```
Out[49]:    popularity  original_title  \
         0   32.985763  Jurassic World

                                                          cast         director  \
         0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...  Colin Trevorrow

            runtime                                      genres  vote_count  \
         0      124  Action|Adventure|Science Fiction|Thriller        5562

            vote_average  release_year    budget_adj   revenue_adj
         0           6.5          2015  1.379999e+08  1.392446e+09
```

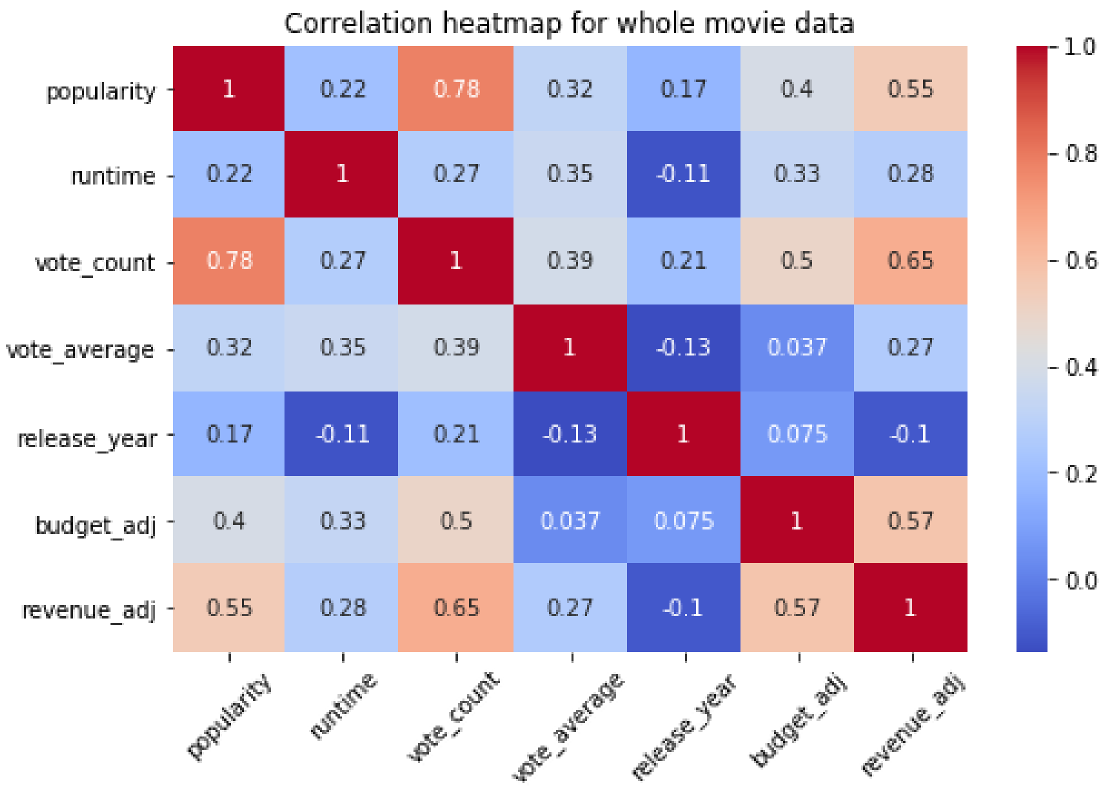
Exploratory Data Analysis

1.2 1. Find pattern and visualize relationship

1_1. Explore relations with `revenue_adj`

```
In [50]: # plot a heatmap to see correlation with `revenue_adj` for each columns
         plt.figure(figsize=(8,5))
         sns.heatmap(tmdb.corr(),annot=True,cmap='coolwarm')
         plt.xticks(rotation=45)
         plt.title('Correlation heatmap for whole movie data')
```

```
Out[50]: Text(0.5, 1.0, 'Correlation heatmap for whole movie data')
```



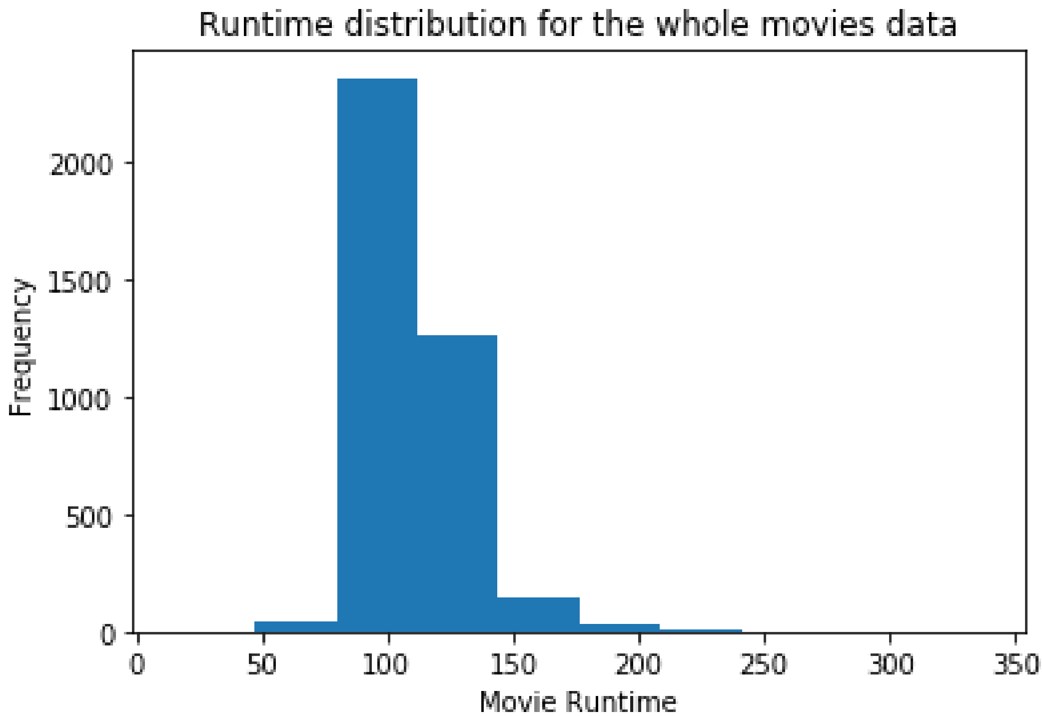
Conclusion: >- `revenue_adj` is positive-correlated with `popularity`, `vote_count` and `budget_adj`, which makes sense, the more popular, the more vote_count and and more revenues. And high budget movies are expected with high revenue too. >- `popularity` and `vote_count` are strongly correlated eith each other. >- `runtime`, `vote_average` and `release_year` do not have strong relation with any other columns. In fact `release_year` is slighly negative-correlated with `revenue_adj`.

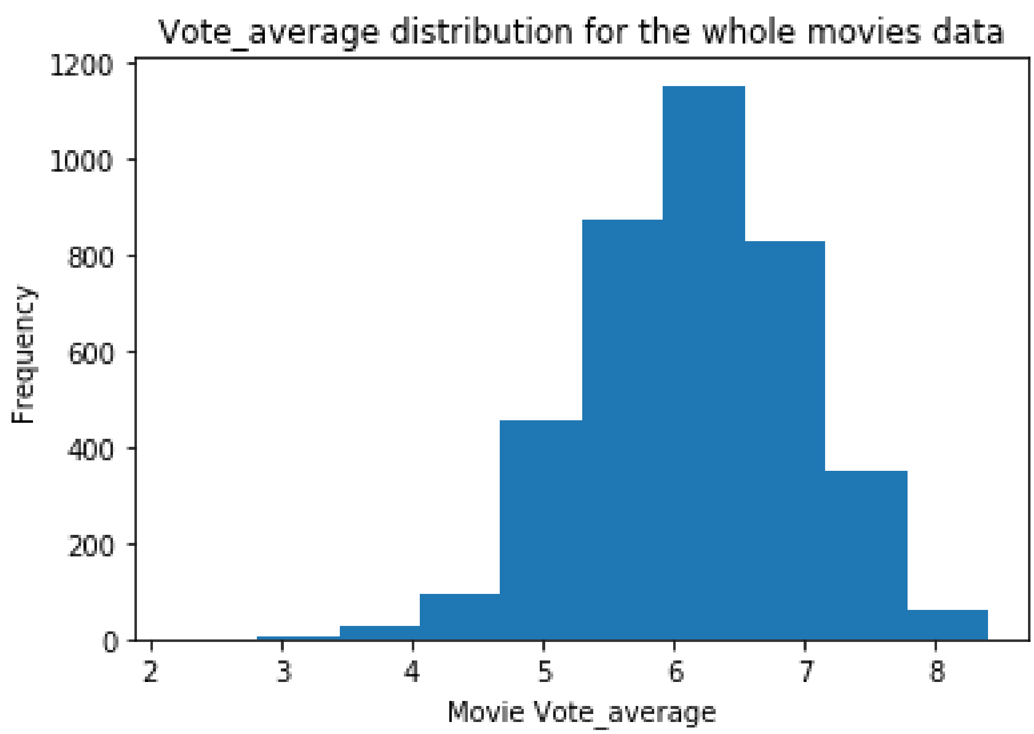
1_2. Plotting charts to find out the distribution for the variables that do not have strong correlation with Movie Revenue, i.e. `runtime`, `vote_average`, `release_year`.

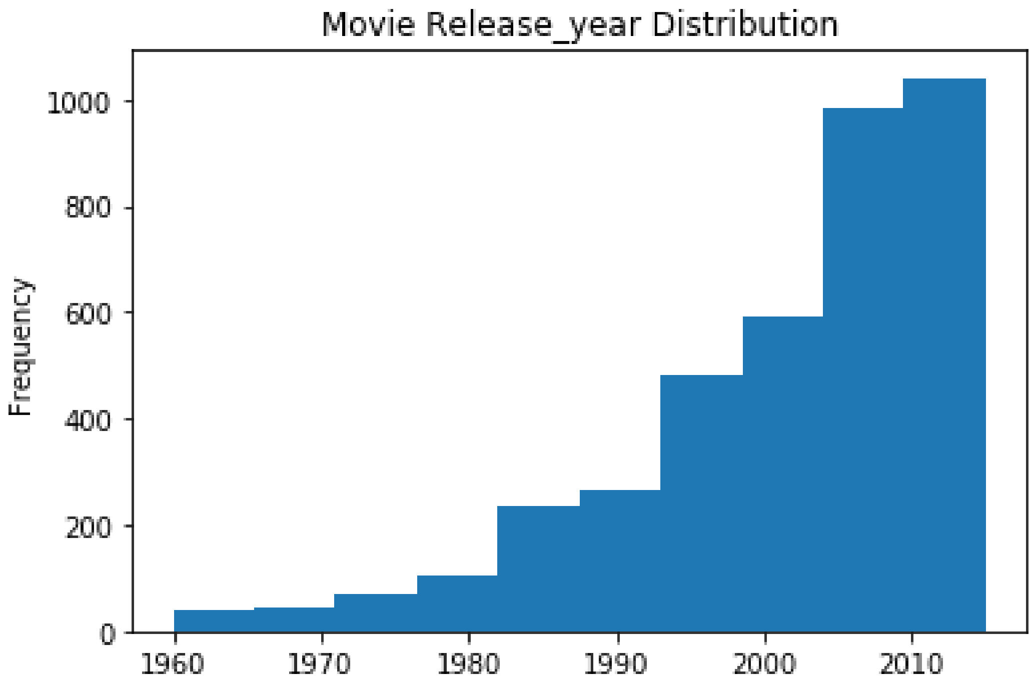
In [51]:
```
# plotting distribution for 'runtime'
tmdb['runtime'].plot.hist(title='Runtime distribution for the whole movies data')
plt.xlabel('Movie Runtime')
plt.show()

# plotting distribution for 'vote_average'
tmdb['vote_average'].plot.hist(title='Vote_average distribution for the whole movies d
plt.xlabel('Movie Vote_average')
plt.show()

# plotting distribution for 'vote_average'
tmdb['release_year'].plot.hist(title=('Movie Release_year Distribution'))
plt.show()
```


Runtime distribution for the whole movies data
Frequency
2000
1500
1000
500
0
0
50
100
150
200
250
300
350
Movie Runtime


Vote_average distribution for the whole movies data
Frequency
1200
1000
800
600
400
200
0
2
3
4
5
6
7
8
Movie Vote_average

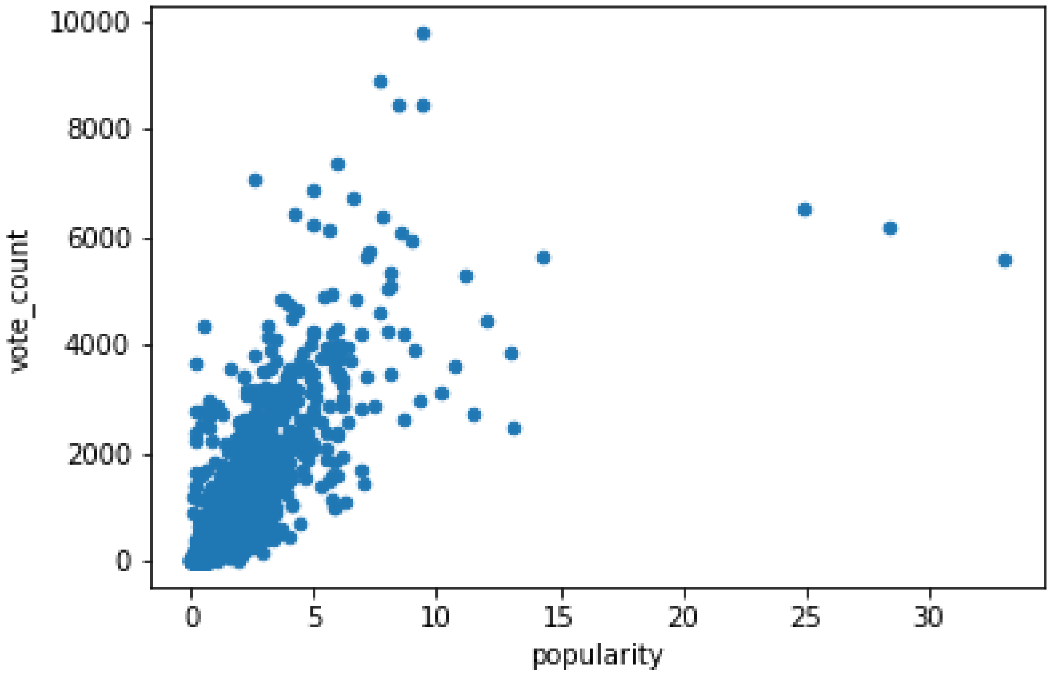


Conclusion: >- Most movies have median length from about 100 minutes to 180 minutes. >- 'vote_average' has normal distribution, with average around 6. >- There are more movies produced over time.

1_3. Plotting scatter chart to explore detailed relationship between `popularity` and `vote_count`, and find out outliers.

```
In [52]: # plotting relation for 'popularity' and 'vote_count'
         tmdb.plot.scatter(x='popularity',y='vote_count')

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1a16b7b860>
```

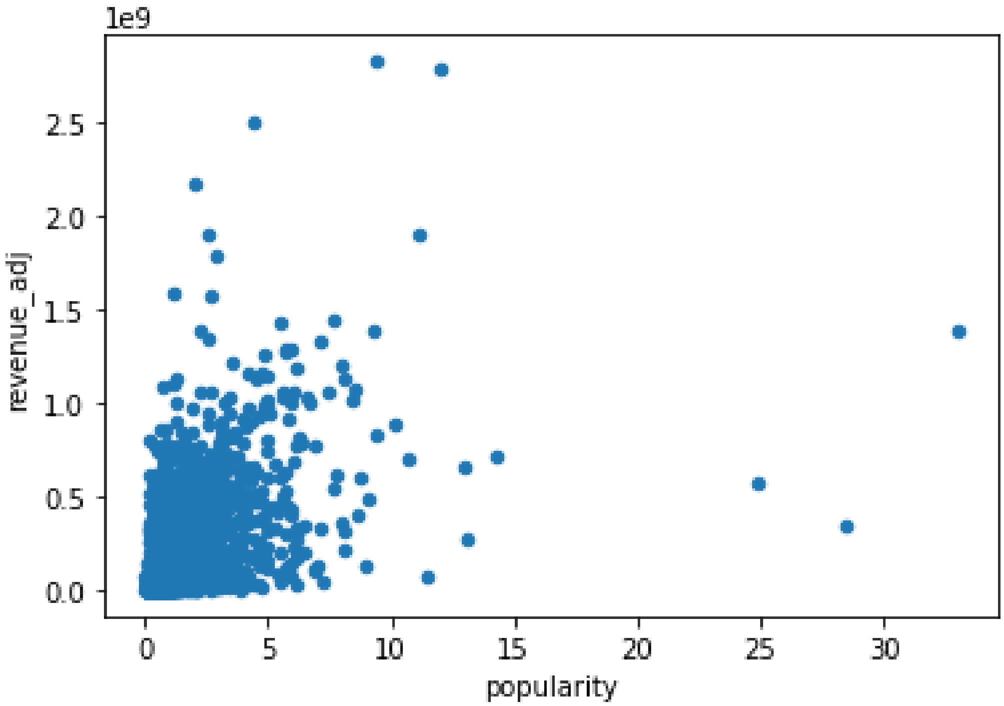


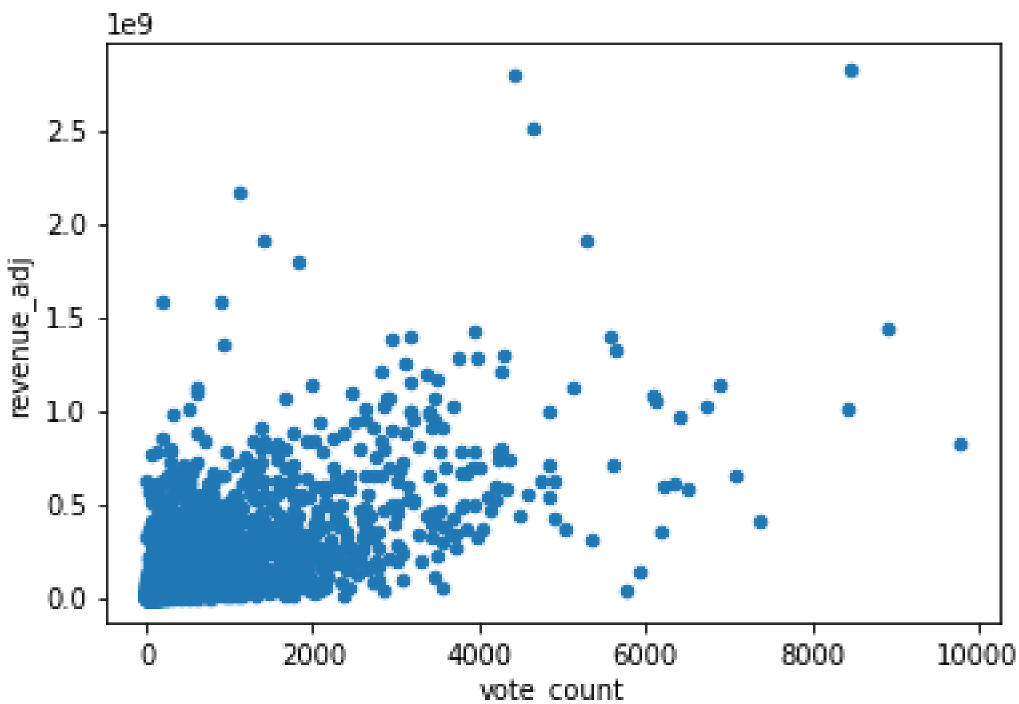
Conclusion: >- From the scatter chart, we can confirm `popularity` and `vote_count` have strong positive correlation, same result as from the heatmap; however, we can also notice there are three movies rated extremely high popularity, but vote count is not extremely high. >- If we run regression model to decide movie revenues, we have to choose of one of them as an independent variable, but this is beyond the goal of this project.

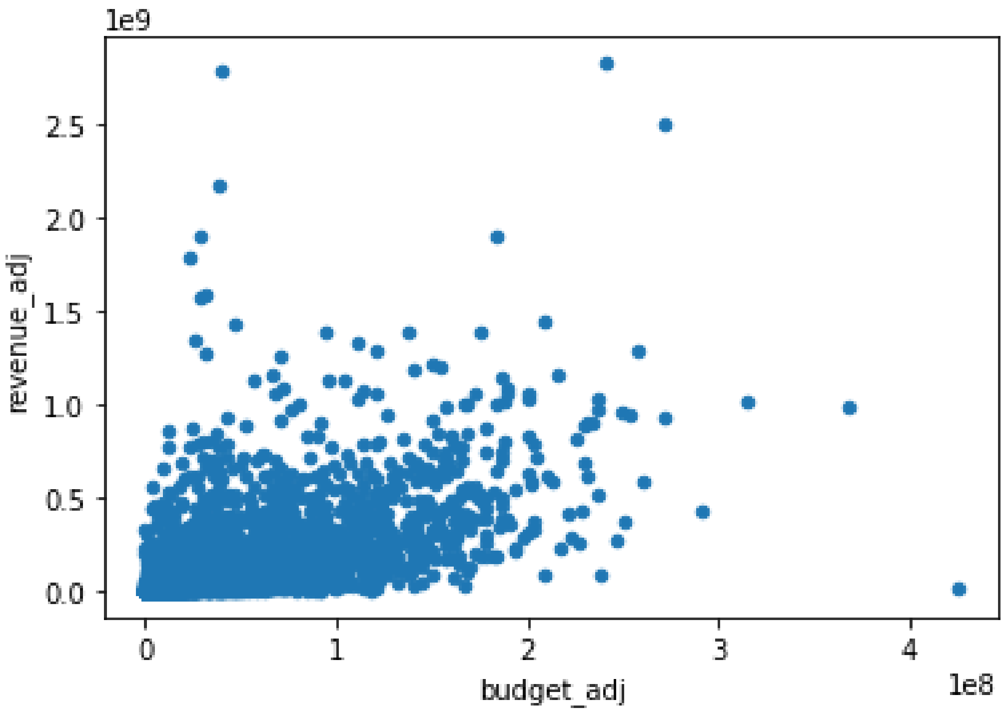
1_4. Plotting scatter charts to furthur explore the relation with `revenue_adj` for the varibles of `popularity`, `vote_count` and `budget_adj`.

```
In [53]: tmdb.plot.scatter(x='popularity', y='revenue_adj')
         tmdb.plot.scatter(x='vote_count',y='revenue_adj')
         tmdb.plot.scatter(x='budget_adj',y='revenue_adj')
```

```
Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x1a15516940>
```


1e9
2.5
2.0
1.5
1.0
0.5
0.0
revenue_adj
0
5
10
15
20
25
30
popularity


1e9
2.5
2.0
1.5
1.0
0.5
0.0
revenue_adj
0
2000
4000
6000
8000
10000
vote_count



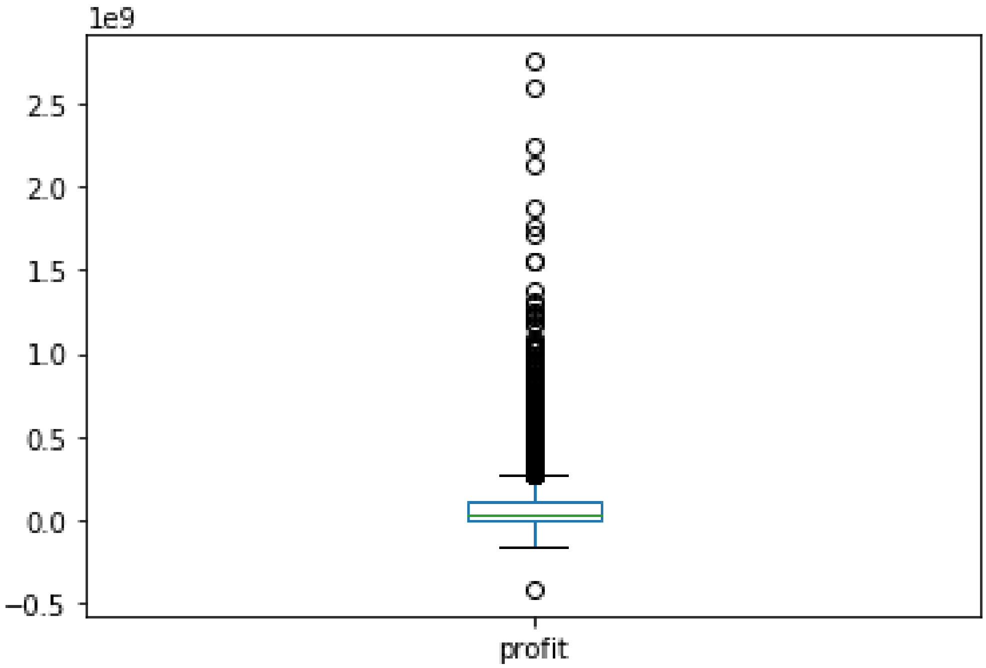
Conclusion: >- In general, the three variabels(`popularity`, `vote_count` and `budget_adj`) are all positively correlated with `revenue_adj`, but the correlation is not very strong, which is the same conclusion from the heatmap; >- There are many outlier data, some movies with extremely high popularity and high vote_count do not have extremely high revenue. These movies maybe controversial, and popularity and vote_count alone are not good indicator for movie success. >- Also, some extremely high budget movies do not have very high revenue, which means they maybe losing money.

1.3 2. Explore Answers for research questions

1.3.1 Question 1. What are the profitibility trend for movie industry?

```
In [54]: 2. # create a column for profit
         tmdb['profit']=tmdb['revenue_adj']-tmdb['budget_adj']
         tmdb['profit'].plot.box()
```

```
Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x1a16a39588>
```



- **Some movies are losing money; others, however have huge profit.**

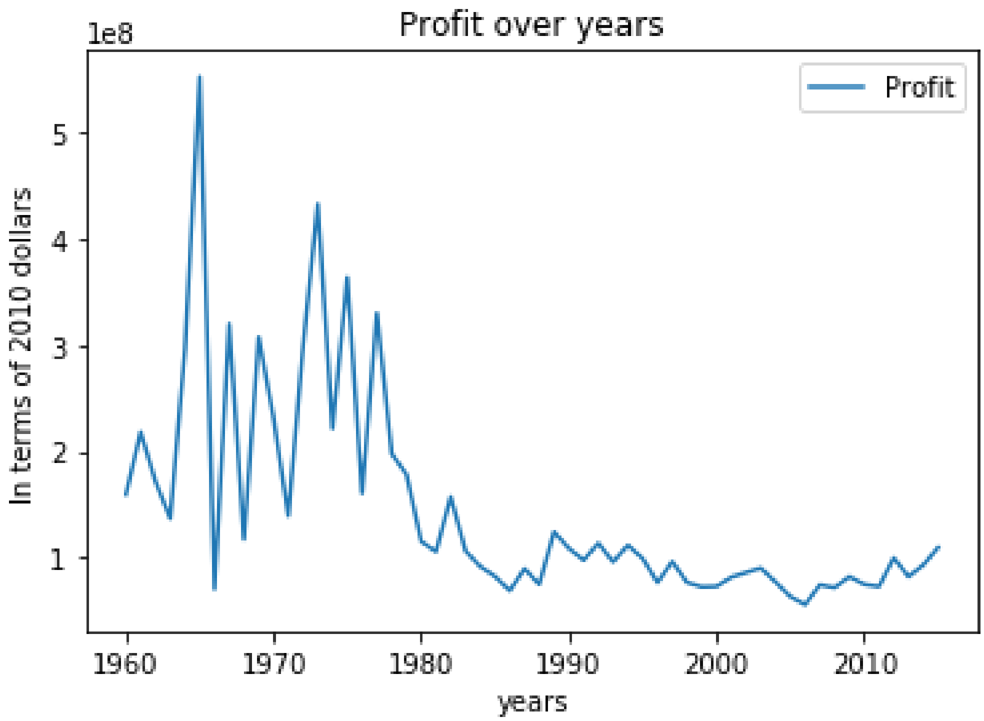
In [55]:
```
# Create a dataframe that group by year and calculate mean value
year_mean=tmdb.groupby('release_year').mean()
year_mean.head(2)
```

Out[55]:
```
              popularity  runtime  vote_count  vote_average    budget_adj  \
release_year
1960            1.324513    130.0       372.6          7.40  3.068179e+07
1961            0.787718    132.5       191.4          6.62  2.818516e+07

               revenue_adj        profit
release_year
1960          1.902299e+08  1.595481e+08
1961          2.463622e+08  2.181770e+08
```

In [56]:
```
# plotting line chart for profit
plt.plot(year_mean.index, year_mean['profit'],label='Profit')
plt.xlabel('years')
plt.ylabel('In terms of 2010 dollars')
plt.title('Profit over years')
plt.legend()
```

Out[56]:
```
<matplotlib.legend.Legend at 0x1a15605550>
```

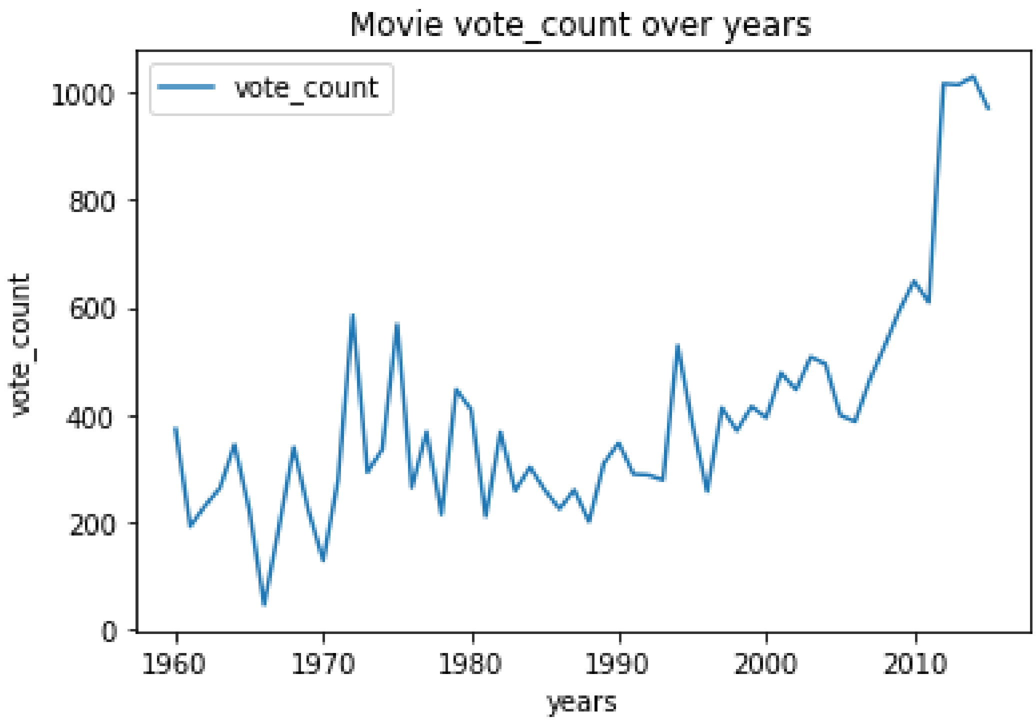


Conclusion: >- Overrall, the average profit per year is lower since 1980; There is less profit to making a movie compared with three decades ago. >- In the earlier years from 1960 to 1980, film industry have higher profit but with very high fluctuation too, and the profit trend is more stable in recent years.We can conclude in the earlier years, film industry is relatively new, high risk is associated with high profit.

1.3.2 Question 2: Are newer movies more popular?

In [57]:
```
# Since popularuty variable has some outlier data, we use vote_count to count for pop
plt.plot(year_mean.index, year_mean['vote_count'], label='vote_count')
plt.xlabel('years')
plt.ylabel('vote_count')
plt.title('Movie vote_count over years')
plt.legend()
```

Out[57]:
```
<matplotlib.legend.Legend at 0x1a16c10da0>
```



Conclusion: >- Yes. There is clear trend that the newer movies are more popular.

1.3.3 Question 3: What kinds of properties are associated with movies that have high revenues?

1. Find out the movies with very high revenues

```
In [58]: # check the distribution for 'revenue_adj'
         tmdb.revenue_adj.describe([.8,.9]).iloc[3:]
```

```
Out[58]: min     2.370705e+00
         50%     6.173068e+07
         80%     2.033811e+08
         90%     3.538761e+08
         max     2.827124e+09
         Name: revenue_adj, dtype: float64
```

```
In [59]: # Create an ordinal data column to categorize movies with different levels of revenue
         bin_edge=[0, 2.872138e+07, 1.496016e+08, 2.880722e+08, 3e+09]
         bin_names=['very_low','low','high','very_high']
         tmdb['revenue_level']=pd.cut(tmdb.revenue_adj,bin_edge,labels=bin_names)
         tmdb['revenue_level'].value_counts()
```

```
Out[59]: low          1550
         very_low     1271
```

```
very_high     517
high          516
Name: revenue_level, dtype: int64
```

- Half of the movies have very_low or even negative revenue
- There are **517 movies with very high_revenue**, let's focus on these movies.

In [60]:
```python
# Create a dataframe that only contains movies with very high revenues
very_high=tmdb[tmdb['revenue_level']=='very_high']
# View top 5 very_high revenue movies
very_high.head()
```

Out[60]:
```
   popularity                original_title  \
0   32.985763                Jurassic World
1   28.419936            Mad Max: Fury Road
3   11.173104  Star Wars: The Force Awakens
4    9.335014                     Furious 7
5    9.110700                  The Revenant

                                                cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...
3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
4  Vin Diesel|Paul Walker|Jason Statham|Michelle ...
5  Leonardo DiCaprio|Tom Hardy|Will Poulter|Domhn...

                             director  runtime  \
0                     Colin Trevorrow      124
1                       George Miller      120
3                         J.J. Abrams      136
4                           James Wan      137
5  Alejandro GonzÃ¡lez IÃ±Ã¡rritu      156

                                genres  vote_count  vote_average  \
0  Action|Adventure|Science Fiction|Thriller        5562           6.5
1  Action|Adventure|Science Fiction|Thriller        6185           7.1
3   Action|Adventure|Science Fiction|Fantasy        5292           7.5
4                    Action|Crime|Thriller        2947           7.3
5            Western|Drama|Adventure|Thriller        3929           7.2

   release_year    budget_adj   revenue_adj        profit revenue_level
0          2015  1.379999e+08  1.392446e+09  1.254446e+09     very_high
1          2015  1.379999e+08  3.481613e+08  2.101614e+08     very_high
3          2015  1.839999e+08  1.902723e+09  1.718723e+09     very_high
4          2015  1.747999e+08  1.385749e+09  1.210949e+09     very_high
5          2015  1.241999e+08  4.903142e+08  3.661143e+08     very_high
```

2. Find out among the very_high profit movies, what kinds of generes are the most common ones.

In [61]:
```
# separate the generes with '|' and make it a new table
generes=very_high['genres'].str.split('|', expand=True)
# view the new table
generes.head()
```

Out[61]:

	0	1	2	3	4
0	Action	Adventure	Science Fiction	Thriller	None
1	Action	Adventure	Science Fiction	Thriller	None
3	Action	Adventure	Science Fiction	Fantasy	None
4	Action	Crime	Thriller	None	None
5	Western	Drama	Adventure	Thriller	None

In [62]:
```
# Count the frequency for each genere for column 0, sorted as index.
col_0=generes.loc[:,0].value_counts().sort_index()
col_0
```

Out[62]:
```
Action             121
Adventure          116
Animation           44
Comedy              63
Crime               12
Drama               61
Family              12
Fantasy             26
History              3
Horror               9
Music                6
Mystery              3
Romance              6
Science Fiction     21
Thriller             9
War                  3
Western              2
Name: 0, dtype: int64
```

In [63]:
```
# Convert the pandas Series to a data frame
df_0=pd.DataFrame(data=col_0, index=col_0.index)
df_0
```

Out[63]:

	0
Action	121
Adventure	116
Animation	44
Comedy	63
Crime	12
Drama	61
Family	12
Fantasy	26
History	3

```
Horror              9
Music               6
Mystery             3
Romance             6
Science Fiction    21
Thriller            9
War                 3
Western             2
```

In [64]:
```python
# do the same for other columns:
col_1=generes.loc[:,1].value_counts().sort_index()
df_1=pd.DataFrame(data=col_1, index=col_1.index)

col_2=generes.loc[:,2].value_counts().sort_index()
df_2=pd.DataFrame(data=col_2, index=col_2.index)

col_3=generes.loc[:,3].value_counts().sort_index()
df_3=pd.DataFrame(data=col_3, index=col_3.index)

col_4=generes.loc[:,4].value_counts().sort_index()
df_4=pd.DataFrame(data=col_4, index=col_4.index)
```

In [65]:
```python
# join the other 4 dataframe together
generes_join=df_0.join(df_1).join(df_2).join(df_3).join(df_4)
generes_join.head()
```

Out[65]:

	0	1	2	3	4
Action	121	79	33	3.0	4.0
Adventure	116	74	36	14.0	1.0
Animation	44	23	9	3.0	2.0
Comedy	63	44	37	13.0	5.0
Crime	12	16	19	11.0	2.0

In [66]:
```python
generes_join.fillna(0)
```

Out[66]:

	0	1	2	3	4
Action	121	79	33	3.0	4.0
Adventure	116	74	36	14.0	1.0
Animation	44	23	9	3.0	2.0
Comedy	63	44	37	13.0	5.0
Crime	12	16	19	11.0	2.0
Drama	61	52	32	5.0	1.0
Family	12	36	40	28.0	9.0
Fantasy	26	42	23	15.0	10.0
History	3	5	4	1.0	0.0
Horror	9	5	3	1.0	1.0
Music	6	6	3	2.0	4.0
Mystery	3	13	7	10.0	2.0
Romance	6	29	20	10.0	5.0

```
Science Fiction   21  18  37  32.0   6.0
Thriller           9  38  66  28.0   7.0
War                3   1  11   4.0   1.0
Western            2   2   3   0.0   0.0
```

In [67]:
```
# calculate the sum of each genere's frequency
generes_join['sum_generes']=generes_join.sum(axis=1)
```

In [69]:
```
# sort value based on frequency sum and show top 5
generes_join.sort_values('sum_generes', ascending=False).head()
```

Out[69]:

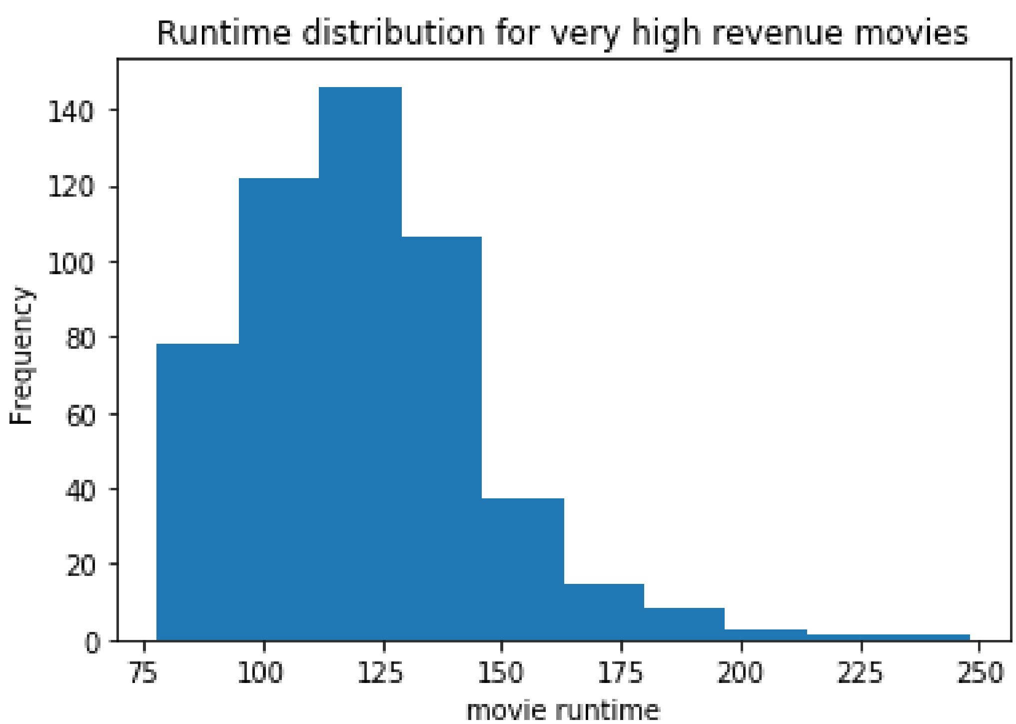
	0	1	2	3	4	sum_generes
Adventure	116	74	36	14.0	1.0	241.0
Action	121	79	33	3.0	4.0	240.0
Comedy	63	44	37	13.0	5.0	162.0
Drama	61	52	32	5.0	1.0	151.0
Thriller	9	38	66	28.0	7.0	148.0

- Now we can see among these very_high revenue movies, the generes with **'Action', 'Adventure', 'Comedy','Drama', 'Thriller'** are top five generes

3. Explore movie runtime for very_high revenue movies

In [70]:
```
# check the runtime distribution
very_high['runtime'].plot.hist()
plt.xlabel('movie runtime')
plt.title('Runtime distribution for very high revenue movies')
```

Out[70]: `Text(0.5, 1.0, 'Runtime distribution for very high revenue movies')`

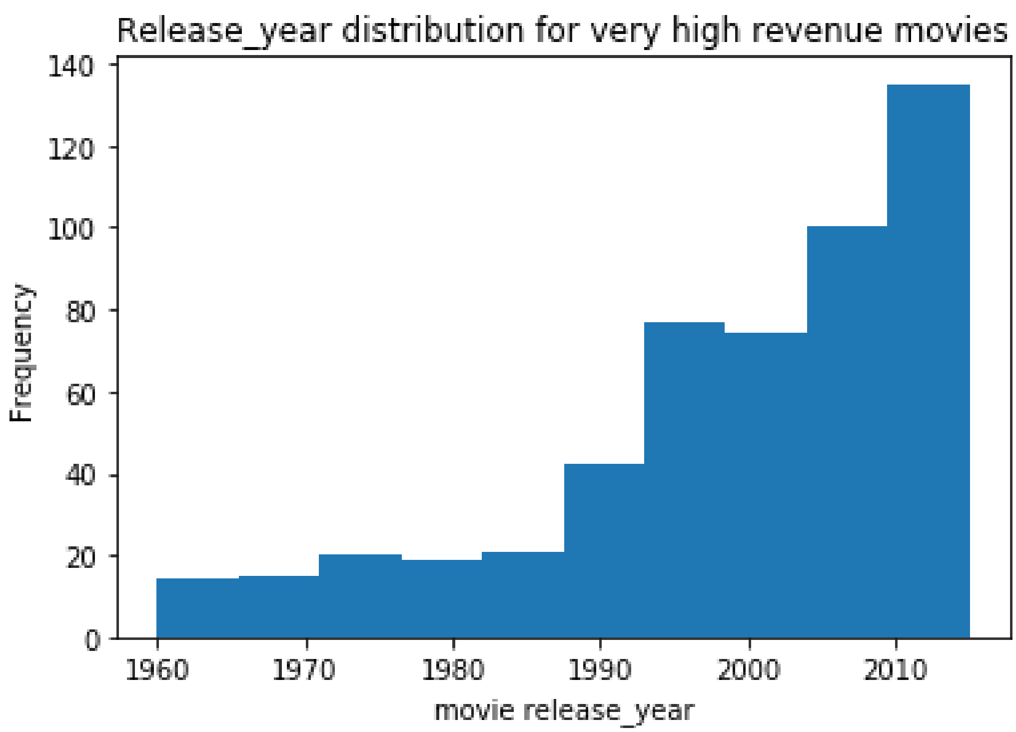


- Most movies in this dataset have runtime ranging from **75 to 150 minutes, with 100 to 125 the most popular**.
- Although some movies that have long runtime also have high revenue,a movie made under 1 hour is less likely to have very high revenue.

4. Explore release_year for very_high revenue movies

```
In [71]: # check the release_year distribution
         very_high['release_year'].plot.hist()
         plt.xlabel('movie release_year')
         plt.title('Release_year distribution for very high revenue movies')

Out[71]: Text(0.5, 1.0, 'Release_year distribution for very high revenue movies')
```

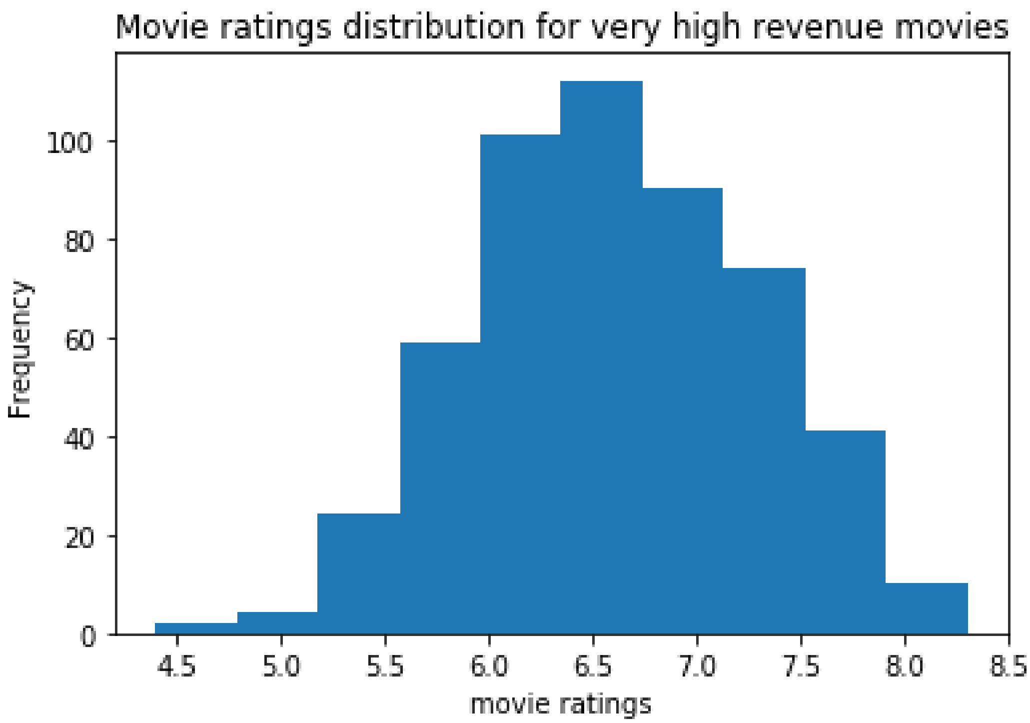


- The movies in this dataset range from 1960 to 2015, which is very similiar distribution with the plot from whole dataset.
- Since there are more movies produced over years and from the line chart previously, we can conclude newer movies does not mean higher profit, it will depend on the movie itself.

5. Explore movie ratings for very_high revenue movies

```
In [72]: # check the vote_average' distribution
         very_high['vote_average'].plot.hist()
         plt.xlabel('movie ratings')
         plt.title('Movie ratings distribution for very high revenue movies')
```

```
Out[72]: Text(0.5, 1.0, 'Movie ratings distribution for very high revenue movies')
```

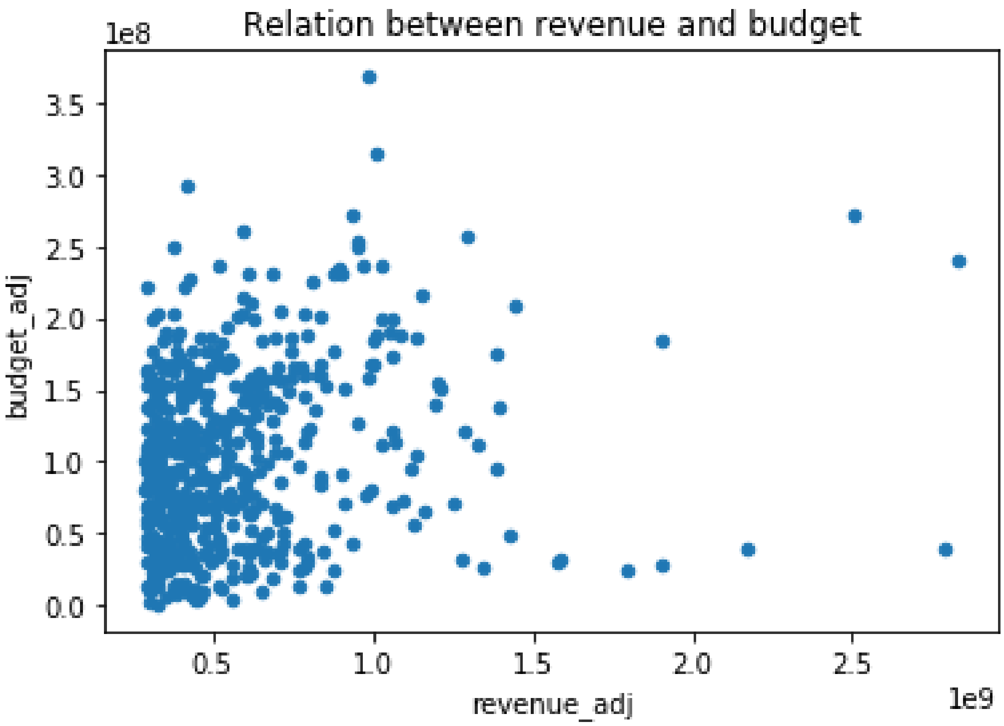


- The chart reveals similar pattern compared with the plotting for whole dataset, but with higher rating in general ,ranging from 4.5 to 8.5, and average is roughly 6.5. From the heatmap earlier, we also see there isn't strong relation between vote_average and high revenue, but a movie with low rating will not have very high revenue.

6. Explore budget for very_high revenue movies

```
In [73]: # Are these high revenue movies made with high budget?
         very_high.plot.scatter(x='revenue_adj', y='budget_adj')
         plt.title('Relation between revenue and budget')
```

```
Out[73]: Text(0.5, 1.0, 'Relation between revenue and budget')
```



- From the heatmap for movies with whole dataset, we found budget and revenue are positively related with each other, however, as we can see among these high_revenue movies, the correlation pattern is weak.
- There are some high reveue moives with low budget and vice versa

1.3.4 Question 4. Is it possible to make extremely high profit movies with low budget?

In [74]:
```
# find out the median value of budget for the whole data
tmdb.budget_adj.median()
```

Out[74]:
```
30016111.9054567
```

In [75]:
```
# create a table for low_budget movies
low_budget=tmdb[tmdb['budget_adj']<30016111.9054567]
# check how many rows are in this table
print(low_budget.shape[0])
# view the distribution of 'revenue_level' in low_budget movie table
low_budget['revenue_level'].value_counts()
```

```
1927
```

Out[75]:
```
very_low     1020
low           746
```

```
high          100
very_high      61
Name: revenue_level, dtype: int64
```

In [76]:
```python
# Percentage of very_high revenue movies with low_budget
61/1927
```

Out[76]:
```
0.0316554229372081
```

- There are only 0.03 of the movies in the low budget group but with very_high revenue, i.e most low budget movies does not yeild high revenue, but it does not mean it is imposible.

In [89]:
```python
# Create a table for these movies
low_budget_very_high_revenue=low_budget[low_budget['revenue_level']=='very_high']
# check how many rows are in this table
print(low_budget_very_high_revenue.shape[0])
low_budget_very_high_revenue.head(3)
```

```
61
```

Out[89]:
```
      popularity        original_title  \
1340    0.602862  Saturday Night Fever
1403    2.367474        The Blind Side
1922    5.293180            Black Swan

                                                 cast          director  \
1340  John Travolta|Karen Lynn Gorney|Barry Miller|J...       John Badham
1403  Sandra Bullock|Quinton Aaron|Kathy Bates|Tim M...   John Lee Hancock
1922  Natalie Portman|Mila Kunis|Vincent Cassel|Barb...  Darren Aronofsky

      runtime                 genres  vote_count  vote_average  release_year  \
1340      118            Drama|Music         192           6.3          1977
1403      129                  Drama        1078           7.1          2009
1922      108  Drama|Mystery|Thriller        2597           7.1          2010

        budget_adj   revenue_adj        profit revenue_level
1340  1.259223e+07  8.530813e+08  8.404891e+08     very_high
1403  2.947561e+07  3.142795e+08  2.848038e+08     very_high
1922  1.300000e+07  3.278037e+08  3.148037e+08     very_high
```

In [90]:
```python
# calculate the profit mean value in the category
low_budget_very_high_revenue['profit'].mean()
```

Out[90]:
```
519202694.07224876
```

In [91]:
```python
# We can also sort profit from the whole data to see the top 10 most profitable movie
top_10_profit=tmdb.sort_values('profit',ascending=False).head(10)
top_10_profit
```

Out[91]:
```
       popularity                  original_title  \
1329    12.037933                       Star Wars
1386     9.432768                          Avatar
5231     4.355219                         Titanic
10594    2.010733                    The Exorcist
9806     2.563191                            Jaws
8889     2.900556      E.T. the Extra-Terrestrial
3       11.173104    Star Wars: The Force Awakens
8094     1.136610                         The Net
10110    2.631987  One Hundred and One Dalmatians
7309     5.488441         The Empire Strikes Back

                                                   cast  \
1329   Mark Hamill|Harrison Ford|Carrie Fisher|Peter ...
1386   Sam Worthington|Zoe Saldana|Sigourney Weaver|S...
5231   Kate Winslet|Leonardo DiCaprio|Frances Fisher|...
10594  Linda Blair|Max von Sydow|Ellen Burstyn|Jason ...
9806   Roy Scheider|Robert Shaw|Richard Dreyfuss|Lorr...
8889   Henry Thomas|Drew Barrymore|Robert MacNaughton...
3      Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
8094   Sandra Bullock|Jeremy Northam|Dennis Miller|We...
10110  Rod Taylor|J. Pat O'Malley|Betty Lou Gerson|Ma...
7309   Mark Hamill|Harrison Ford|Carrie Fisher|Billy ...

                                               director  runtime  \
1329                                       George Lucas      121
1386                                      James Cameron      162
5231                                      James Cameron      194
10594                                  William Friedkin      122
9806                                   Steven Spielberg      124
8889                                   Steven Spielberg      115
3                                           J.J. Abrams      136
8094                                      Irwin Winkler      114
10110  Clyde Geronimi|Hamilton Luske|Wolfgang Reitherman       79
7309                                     Irvin Kershner      124

                                        genres  vote_count  vote_average  \
1329           Adventure|Action|Science Fiction        4428           7.9
1386   Action|Adventure|Fantasy|Science Fiction        8458           7.1
5231                     Drama|Romance|Thriller        4654           7.3
10594                     Drama|Horror|Thriller        1113           7.2
9806                  Horror|Thriller|Adventure        1415           7.3
8889   Science Fiction|Adventure|Family|Fantasy        1830           7.2
3      Action|Adventure|Science Fiction|Fantasy        5292           7.5
8094        Crime|Drama|Mystery|Thriller|Action         201           5.6
10110         Adventure|Animation|Comedy|Family         913           6.6
7309           Adventure|Action|Science Fiction        3954           8.0
```

```
       release_year   budget_adj   revenue_adj        profit revenue_level
1329           1977 3.957559e+07 2.789712e+09 2.750137e+09     very_high
1386           2009 2.408869e+08 2.827124e+09 2.586237e+09     very_high
5231           1997 2.716921e+08 2.506406e+09 2.234714e+09     very_high
10594          1973 3.928928e+07 2.167325e+09 2.128036e+09     very_high
9806           1975 2.836275e+07 1.907006e+09 1.878643e+09     very_high
8889           1982 2.372625e+07 1.791694e+09 1.767968e+09     very_high
3              2015 1.839999e+08 1.902723e+09 1.718723e+09     very_high
8094           1995 3.148127e+07 1.583050e+09 1.551568e+09     very_high
10110          1961 2.917944e+07 1.574815e+09 1.545635e+09     very_high
7309           1980 4.762866e+07 1.424626e+09 1.376998e+09     very_high
```

```
In [92]: # calculate the profit mean value in the categary
         top_10_profit['profit'].mean()
```

```
Out[92]: 1953865824.9082134
```

```
In [93]: # compare the difference between the average of top 10 movies's revenue and low_budge
         top_10_profit['profit'].mean()-low_budget_very_high_revenue['profit'].mean()
```

```
Out[93]: 1434663130.8359647
```

Conclusion: >- Even though some low_budget movies made very_high revenues, the difference between the average revenue for top_10_profit movies and low_budget_very_high_revenue movies are huge. >- This can imply that for the movies made with huge profit, they are made with huge budget too. We can not expect a movie with low budget to make extremely high profit; however it is possible to make low budget movies with moderately high profit, but the chances are not significant, there are only 61 very_high revenue movies in the low_budget table.

1.3.5 Question 5. What are the top 10 rated movies? and how is their profitibility?

Since some movies have more vote_count, we can not directly compare a movie rated 10 with only 3 counts to the movie rated 7 with 100 counts. We will use IMDB'a definition to calculated weighted average for rating score.

```
In [94]: # m is the minimum votes required to be listed in the chart;
         m= tmdb['vote_count'].quantile(0.9)
         m
```

```
Out[94]: 1371.7000000000003
```

```
In [95]: # C is the mean vote across the whole report
         C=tmdb['vote_count'].mean()
         print(C)
```

```
527.7202906071614
```

```
In [96]: # Create a table for top 10% highest rated movies
         q_movies = tmdb.copy().loc[tmdb['vote_count'] >= m]
         q_movies.shape
```

Out[96]: (386, 13)

In [97]:
```
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)

# Define a new feature 'score' and calculate its value with `weighted_rating()`
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
```

In [98]:
```
# show the top 10 rated movies
q_movies.sort_values('score',ascending=False).head(10)
```

Out[98]:
```
       popularity             original_title  \
2912     1.499784                 Cloverfield
2643     2.449323           The Mummy Returns
6980     2.175284              Ocean's Twelve
7884     2.484654                Ghostbusters
21       5.337064                    Southpaw
658      3.813740     Exodus: Gods and Kings
3416     1.499109                       Rango
6631     0.890909  The Pursuit of Happyness
6578     1.603140               Blood Diamond
10094    0.142486                  Home Alone

                                                    cast            director  \
2912   Lizzy Caplan|Jessica Lucas|Odette Annable|Mich...         Matt Reeves
2643   Brendan Fraser|Rachel Weisz|John Hannah|Arnold...     Stephen Sommers
6980   George Clooney|Brad Pitt|Catherine Zeta-Jones|...  Steven Soderbergh
7884   Bill Murray|Dan Aykroyd|Sigourney Weaver|Harol...        Ivan Reitman
21     Jake Gyllenhaal|Rachel McAdams|Forest Whitaker...      Antoine Fuqua
658    Christian Bale|Joel Edgerton|John Turturro|Aar...       Ridley Scott
3416   Johnny Depp|Isla Fisher|Ned Beatty|Bill Nighy|...     Gore Verbinski
6631   Will Smith|Jaden Smith|Thandie Newton|Brian Ho...   Gabriele Muccino
6578   Leonardo DiCaprio|Djimon Hounsou|Jennifer Conn...       Edward Zwick
10094  Macaulay Culkin|Joe Pesci|Daniel Stern|John He...     Chris Columbus

       runtime                                     genres  vote_count  \
2912        85           Action|Thriller|Science Fiction        1373
2643       130        Action|Adventure|Drama|Fantasy|Horror        1372
6980       125                            Thriller|Crime        1376
7884       107  Fantasy|Action|Comedy|Science Fiction|Family        1383
21         123                              Action|Drama        1386
658        153                    Adventure|Drama|Action        1377
3416       107     Animation|Comedy|Family|Western|Adventure        1385
6631       117                                     Drama        1392
6578       143                    Drama|Thriller|Action        1394
```

```
10094      103                                  Comedy|Family        1393

       vote_average  release_year   budget_adj   revenue_adj       profit  \
2912            6.4          2008  2.531967e+07  1.729475e+08  1.476279e+08
2643            5.8          2001  1.206858e+08  5.332507e+08  4.125649e+08
6980            6.4          2004  1.269890e+08  4.187685e+08  2.917795e+08
7884            7.2          1984  6.297126e+07  6.196634e+08  5.566921e+08
21              7.3          2015  2.759999e+07  8.437300e+07  5.677302e+07
658             5.6          2014  1.289527e+08  2.468817e+08  1.179290e+08
3416            6.5          2011  1.308687e+08  2.382049e+08  1.073362e+08
6631            7.5          2006  5.949180e+07  3.321560e+08  2.726642e+08
6578            7.2          2006  1.081669e+08  1.848334e+08  7.666646e+07
10094           7.0          1990  3.004017e+07  7.955384e+08  7.654982e+08

      revenue_level       score
2912           high  266.936686
2643      very_high  266.731612
6980      very_high  266.652226
7884      very_high  266.392537
21              low  266.160831
658            high  266.156773
3416           high  265.852803
6631      very_high  265.699578
6578           high  265.361653
10094     very_high  265.354260
```

- Above are the movies with highest rating score, as we can see, most have high or very_high revenue too, but it is different from the top_10_profit movies.

In [99]:
```
q_movies['revenue_level'].value_counts()
```

Out[99]:
```
very_high    242
high          75
low           64
very_low       5
Name: revenue_level, dtype: int64
```

In [100]:
```
# Count for percentage of the one with low or very_low revenue
(64+5)/q_movies.shape[0]
```

Out[100]:
```
0.17875647668393782
```

- About 17% of the highly rated movies have low revenue, most have very_high revenue.
- This can confirm that a good rating score can yield high revenue.

Conclusions
In this project we did comprehensive analysis on the movie database with a focus on movie revenues and other properties like generes and rating score. We did initial data exploration and answered all the questions

Summarize some of the featured findings: >- In general, higher budget can yield higher revenues; movies made with low budget can have moderately high revenue, but successful rate is not very high. >- Popular movies have more vote_count and also have higher revenues, and newer movies are more popular >- Although much more movies are produced over time, movies industry is getting more stable and annual average profit is lower compared with movies made 3 decades ago. >- ’Action’, ’Adventure’, ’Comedy’,’Drama’, ’Thriller’ are the most common generes for movies with very_high revenue >- Movies with high rating scores also have high revenues, but not nessesarily yield the highest or extremely high profit.

Limitation of the project >- The dataset we are using contains some incorrect information. As we see in data cleaning process, more than half of the data is deleted since it contains 0 value for runtime, budget or revenue. If we can have a more complete data, the analysis can be more accurate. >- Since the project is main focusing on movie revenue analysis, and we find popularity, budget and rating score can have impact on revenue, but these information is not enough to make revenue prediction as there might be other factors that can affect movie revenue but not included in this dataset.