1. **The name of student**:

   Zhuoyou Shen

1. **How to run your code (what command-line switches they are, what happens when you invoke the code, etc.)**

   You can get the clean data used in this notebook analysis simply from data subfolder where the data sets have existed already,

   or you can run the followin files to get the data sets from Internet.

   To do so, using command-line: cd src python main.py

   open the data and perform analysis on stored data, or python main.py scraped the complete data, store in a database and then perform analysis on the database ,then json file will be stored in data subfolder python3 extract.py ,run the command-line can transformer json file to csv This project requires the following packages:

   - pandas, numpy, seaborn, requests and beautifulsoup,sklearn

1. **Any major “gotchas” to the code (i.e. things that don’t work, go slowly, could be improved, etc.)**

   it takes more than 10 minutes to scrape datasets from websites.

1. **Anything else you feel is relevant to the grading of your project your project.**

   My data is about movie ratings, movie budgets, movie box office and number of ratings, movie genres, etc. I hope to use the random forest model to predict movie ratings by analyzing the correlation between them and selecting features. At the same time, I also conducted some statistical analysis on it, and found some characteristics of high-rated movies.

1. **What did you set out to study? (i.e. what was the point of your project? This should be close to your Milestone 1 assignment, but if you switched gears or changed things, note it here.)**

   My project wants to predict movie ratings by analyzing data from the past three years.

1. **What did you Discover/what were your conclusions (i.e. what were your findings? Were your original assumptions confirmed, etc.?)**

   The results show that the movie genres, duration, and budget can indeed greatly affect the score. Using the random forest model to predict, it is found that the correct rate on the test set is more than 95%:

   Therefore, the conclusion is that these factors are indeed related to movie ratings, and we can use these factors to avoid bad movies.

   In addition, I did some statistical analysis and concluded:

   A. More than 2/3 of the hotter movies are all action movies

   B. Action movies have far more followers than other types

   C. Animation duration is usually shorter than other types of duration
   D. Animation is easier to get high scores
   E. Generally speaking, movies with a large budget are more likely to get high scores

1. **What difficulties did you have in completing the project?**

   The most difficult part must be scrape the imdb information data from IMdb website, because the structure of the web page is complicated. I failed many times and I needed to wait for awhile every time. Encoding issues in the process of crawling data also took a lot of time.

1. **What skills did you wish you had while you were doing the project?**

   The skill I want to acquire most for this project is to use more methods and dig out more information from the data.

1. **What would you do “next” to expand or augment the project?**

   I hope that more detailed information can be collected to expand the characteristics of the model.

In [1]:
```
import pandas as pd
from bs4 import BeautifulSoup
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
# Read in the three data sets

movie_info_ = pd.read_csv('movie_imb_data.csv')
movie_info = movie_info_.where(movie_info_['title_year']>=2018).dropna(subset=['movie_title'])
```

In [3]:
```
#Take a look of the columns
```

In [4]:
```
movie_info.head()
```

Out[4]:

	Unnamed: 0	movie_title	num_user_for_reviews	plot_keywords	genres	time	title_year	imdb_score	aspect_ratio	num_critic_for_reviews	
0	0.0	Avengers: Endgame	9300.0	time travel	Action	181.0	2019.0	8.4	NaN	583.0	http
1	1.0	Avengers: Infinity War	4400.0	death of recurring character	Action	149.0	2018.0	8.4	NaN	624.0	http
2	2.0	Star Wars: Episode IX - The Rise of Skywalker	7900.0	evil	Action	141.0	2019.0	6.5	2.39	518.0	http
4	4.0	No Time to Die	2700.0	shooting	Action	163.0	2021.0	7.5	2.39	387.0	http
5	5.0	Tenet	5200.0	NaN	Action	150.0	2020.0	7.4	2.20	498.0	http

In [5]:
```
#The first data set contains information of the movie metadata
#like rate, and so on

#Get the average score by year
mean_score_per_year = movie_info.groupby(movie_info['title_year']).mean()
mean_score_per_year.reset_index(inplace=True)
```

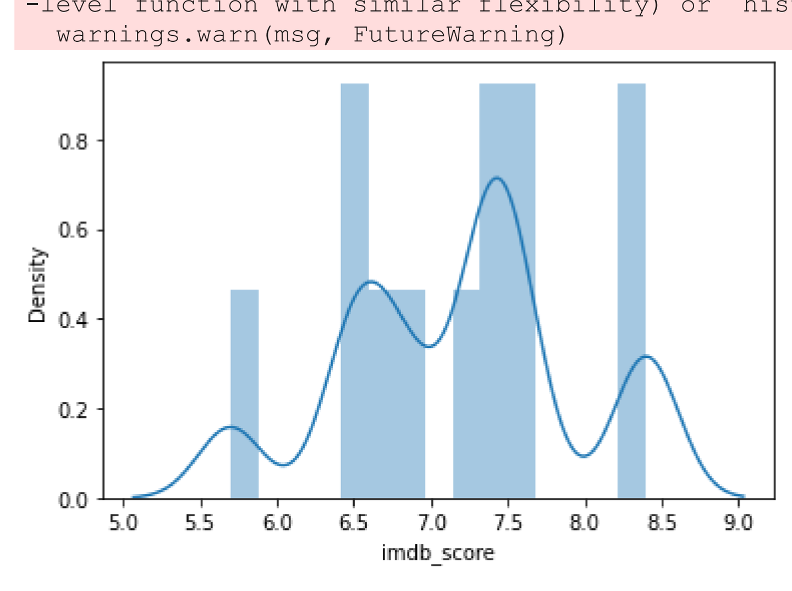
In [6]:
```
mean_score_per_year.head()
```

Out[6]:

	title_year	Unnamed: 0	num_user_for_reviews	time	imdb_score	aspect_ratio	num_critic_for_reviews
0	2018.0	317.000000	2333.333333	133.666667	7.500000	2.390000	459.333333
1	2019.0	313.000000	8600.000000	161.000000	7.450000	2.390000	550.500000
2	2020.0	320.666667	2953.666667	122.333333	6.833333	2.326667	359.666667
3	2021.0	320.000000	2525.000000	146.250000	7.100000	2.267500	338.500000

In [7]:
```
#Trying to figure out whether the price correlates the scores
#Get the regression line
fig, axes = plt.subplots(1, 1)
m1 = sns.distplot(movie_info.imdb_score, bins=15)
```

```
C:\Users\49181\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



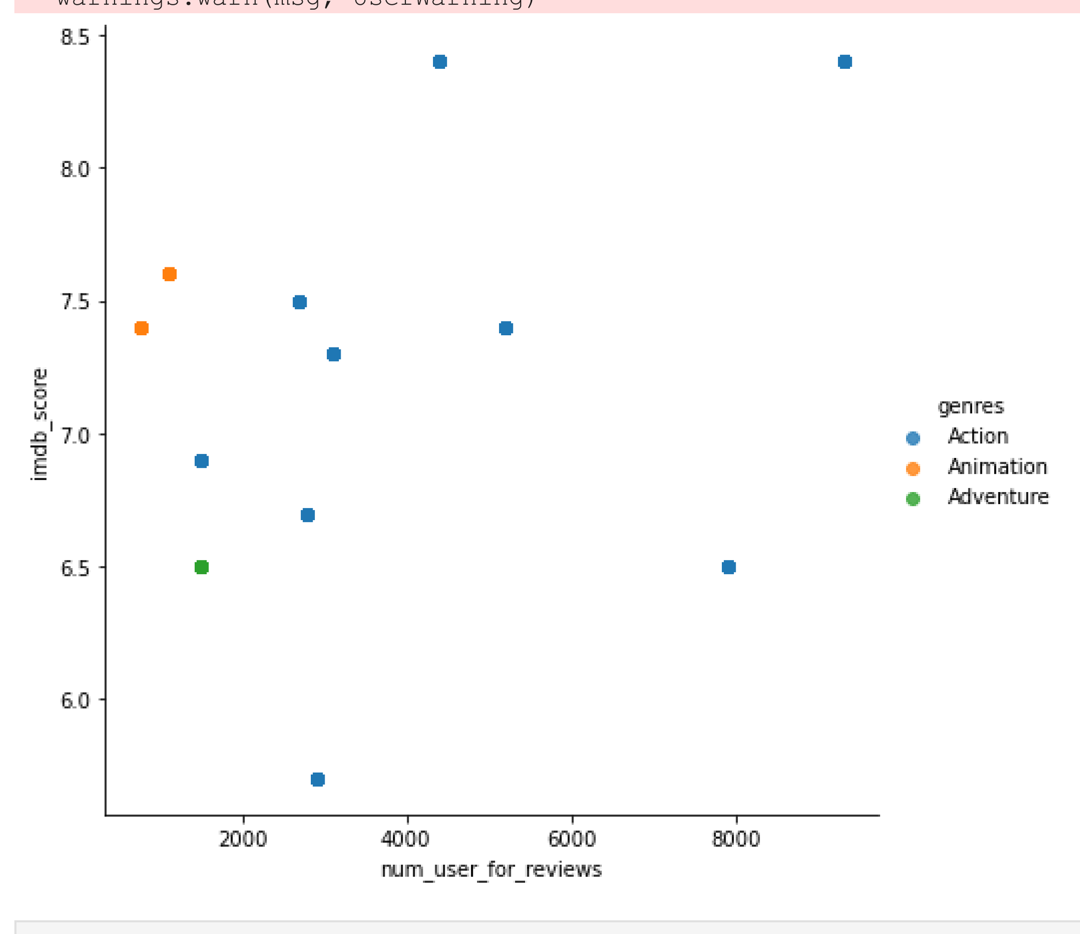
In [8]:
```
# genre
movie_info.genres.value_counts()
```

Out[8]:
```
Action       441
Animation     98
Adventure     49
Name: genres, dtype: int64
```

In [9]:
```
# More than 2/3 of the hotter movies are all action movies
```

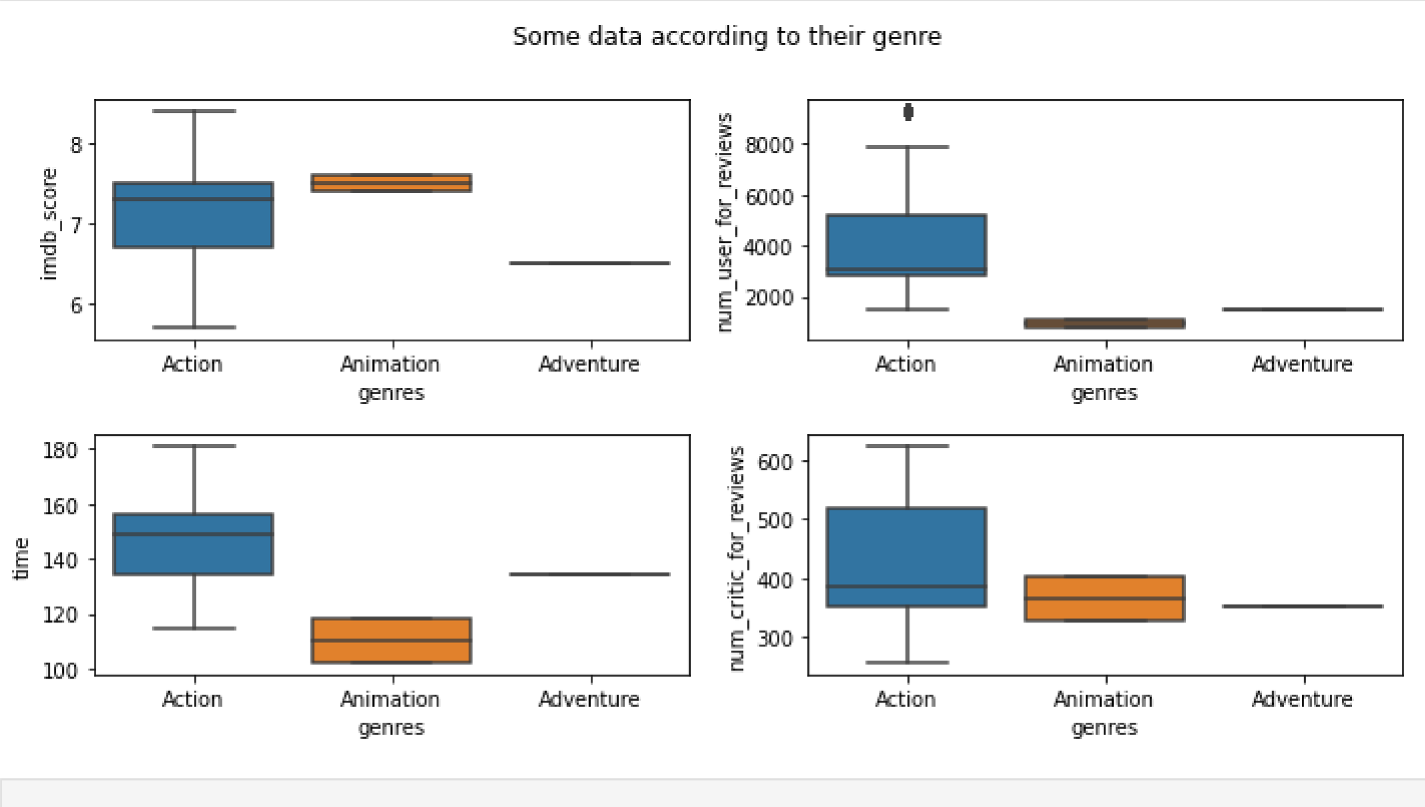
In [10]:
```
vis1 = sns.lmplot(data=movie_info, x='num_user_for_reviews', y='imdb_score', \
                  fit_reg=False, hue='genres', size=6, aspect=1)
```

```
C:\Users\49181\anaconda3\lib\site-packages\seaborn\regression.py:580: UserWarning: The `size` parameter has bee
n renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



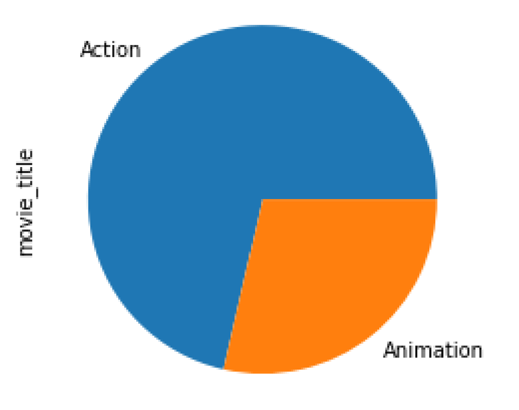
In [11]:
```
#Action movies have far more followers than other types
```

In [12]:
```
fig, axes = plt.subplots(2, 2)
fig.set_size_inches(11.7, 5.27)
plt.subplots_adjust(wspace=0.2, hspace=0.4)

sns.boxplot(data=movie_info, x='genres', y='imdb_score', ax=axes[0, 0])
sns.boxplot(data=movie_info, x='genres', y='num_user_for_reviews', ax=axes[0, 1])
sns.boxplot(data=movie_info, x='genres', y='time', ax=axes[1, 0])
sns.boxplot(data=movie_info, x='genres', y='num_critic_for_reviews', ax=axes[1, 1])

plt.suptitle('Some data according to their genre')

plt.show()
```



In [13]:
```
# high score genres movies
df1 = movie_info.where(movie_info['imdb_score']>=7)
pl = df1.groupby('genres').movie_title.count()
pl.plot.pie()
```

Out[13]: `<AxesSubplot:ylabel='movie_title'>`



In [14]:
```
# transform data
from sklearn.preprocessing import LabelEncoder
columnsToEncode = list(movie_info[['title_year','genres']])
print(columnsToEncode)
le = LabelEncoder()
df = movie_info[['title_year','genres','num_user_for_reviews','plot_keywords',
                 'time','imdb_score','num_critic_for_reviews']]
#df.head()
for feature in columnsToEncode:
    try:
        df[feature] = le.fit_transform(df[feature])
    except:
        print('Error encoding ' + feature)
df.head()
```

```
['title_year', 'genres']
<ipython-input-14-12582017a042>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  df[feature] = le.fit_transform(df[feature])
```

Out[14]:

	title_year	genres	num_user_for_reviews	plot_keywords	time	imdb_score	num_critic_for_reviews
0	1	0	9300.0	time travel	181.0	8.4	583.0
1	0	0	4400.0	death of recurring character	149.0	8.4	624.0
2	1	0	7900.0	evil	141.0	6.5	518.0
4	3	0	2700.0	shooting	163.0	7.5	387.0
5	2	0	5200.0	NaN	150.0	7.4	498.0

In [15]:
```
# get x y
X = df.dropna()
y = X['imdb_score']
X = X.drop(['imdb_score'], axis=1)
```

In [117…
```
#split data

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X = scaler.fit_transform(X)
y = np.array(y*10).astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=33)
#print(y_test)
```

```
[67 67 74 75 84 84 84 67 74 84 84 65 74 67 84 57 75 75 84 84 84 73 75 84
 65 65 67 76 84 84 74 69 74 65 84 76 69 84 84 73 57 76 57 76 73 67 84 74
 69 84 74 76 57 73 75 69 76 69 84 84 67 84 84 69 74 75 74 84 67 74 57 74
 84 84 65 75 76 84 84 57 57 74 65 76 65 65 84 65 73 75 69 73 57 75 65 84
 57 67 76 84 74 74 67 65 65 65 57 74 65 73 57 65 73 74 74 67 74 65 65 75
 74 74 75 67 65 74 74 69 74 74 67 65 84 76 65 76 69 69 65 75 69 65 65 65
 65 67 84 65 74 74 65 73 84 84 65 69 74 73 76 74 84 65 69 74 65 73 74 84
 74 57 74 84 76 76 69 76 76 74 74 75 84 67 57 73 69 74 69 65 84 84 73 84
 74 73 84 65 73 84 57 76 84 65 57 76 69 67 57 84 84 75 69 75 57 73 76 84
 67 73 84 65 75 65 75 76 74 65 57 84 74 65 76 57 57 84 65 75 67 65 65 84
 65 84 74 65 75 74 74 67 74 73 74 69 74 84 73 67 57 65 75 73 65 65 67 75
 84 67 75 84 74 65 65 67 84 74 76 74 69 74 67 65 75 76 76 57 65 73 74 84
 73 65 65 75 65 73 84 74 67 74 74 75 73 84 76 73 76 76 75 76 74 67 84 67
 74 74 74 84 73 76 57 73 65 84 73 67 73 74 67 74 57 69 69 76 73 73 69 76
 76 74 74 57 57 75 65 76 75 76 74 65 57 57 69 73 65 76 57 65 84 84 84 65
 74 57 84 65 75 67 69 57 73 74 84 65 74 65 65 84 69 84 75 84 69 84 74 67
 74 57 65 57 74 67 69 69 65 74 73 75 74 57 84 76 69 57 74 69 74 84 65 76
 73 65 76 65 57 69 84 74 84 65 57 73 67 76 65 73 57 57 65 84 65 69 67 73
 84 65 74 67 73 69 69 74 69 75 74 65 73 84 69 69 84 75 84 84 67 74 84 65
 65 57 74 65 74 65 69 84 74 76 67 74 73 74 84 76 65 65 69 69 84 65 75 75
 65 75 74 57 74 84 73 67 76 76 84 67 74 74 65 57 57 74 75 75 67 65 74 69
 67 75 69 65 75 84 67 67 74 57 65 73 84 76 65 75 84 84 84 74 84 69 57 74
 75 57 76 75 73 76 74 65 65 73 84 74 65 73 69 74 69 67 65 75 75 84 84 57
 73 65 67 76 57 76 74 69 57 67 73 65 74 67 75 65 75 65 65 69 84 84 67 73
 74]
```

In [123…
```
# predict randomForest
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier(n_estimators=20)
print(X_train)
clf = clf.fit(X_train, y_train)
clf_y_predict = clf.predict(X_test)
#print(accuracy_score(y_test, clf_y_predict))
```

```
[[ 1.13137085 -0.5488213  -0.34840069 -1.06904497  1.11058519 -0.19336964]
 [-1.41421356  2.08552094 -0.97002445  0.93541435 -1.02363546 -0.04617783]
 [-0.56568542 -0.5488213   1.67187652 -1.46993683  0.06718843  0.94087319]
 [-1.41421356 -0.5488213   0.31207455 -1.87082869  0.44660543  1.85865746]
 [ 1.13137085 -0.5488213  -0.3095492  -0.26726124 -0.26480145 -0.42714488]
 [ 1.13137085 -0.5488213  -0.34840069 -1.06904497  1.11058519 -0.19336964]
 [ 0.28284271  2.08552094 -1.10173098  0.93541435 -1.78246947 -0.72152852]
 [ 0.28284271  2.08552094 -1.10173098  0.93541435 -1.78246947 -0.72152852]
 [-0.56568542 -0.5488213   1.67187652 -1.46993683  0.06718843  0.94087319]
 [-1.41421356  0.76834982 -0.81461851  0.53452248 -0.26480145 -0.5137283 ]
 [ 1.13137085 -0.5488213  -0.81461851 -0.6681531   0.77859531 -1.32761247]]
```