Load the tmdb movies dataset and explore the data first

In [2]:
```
import json
import pandas as pd
```

In [3]:
```
def load_movies(path):
    data_file = pd.read_csv(path)
    data_file['release_date'] = pd.to_datetime(data_file['release_date'
]).apply(lambda x: x.date())
    columns = ['genres', 'keywords', 'production_countries', 'production
_companies', 'spoken_languages']
    for column in columns:
        data_file[column] = data_file[column].apply(json.loads)
    return data_file

def load_credits(path):
    data_file = pd.read_csv(path)
    columns = ['cast', 'crew']
    for column in columns:
       data_file[column] = data_file[column].apply(json.loads)
    return data_file
```

In [4]:
```
def safe_access(container, index_values):
    result = container
    try:
        for idx in index_values:
            result = result[idx]

        return result

    except IndexError or KeyError:
        return pd.np.nan



def get_director(crew_data):
    directors = [x['name'] for x in crew_data if x['job'] == 'Director']
    return safe_access(directors, [0])

def convert_to_original_format(movies, credits):
    tmdb_movies = movies.copy()
    tmdb_movies['year'] = pd.to_datetime(tmdb_movies['release_date']).ap
ply(lambda x: x.year)
    tmdb_movies['country'] = tmdb_movies['production_countries'].apply(l
ambda x: safe_access(x, [0, 'name']))
    tmdb_movies['language'] = tmdb_movies['spoken_languages'].apply(lamb
da x: safe_access(x, [0, 'name']))
    tmdb_movies['genres'] = tmdb_movies['genres'].apply(lambda x: '|'.jo
in([i['name'] for i in x]))
    tmdb_movies['keywords'] = tmdb_movies['keywords'].apply(lambda x:
'|'.join([i['name'] for i in x]))
    tmdb_movies['director'] = credits['crew'].apply(get_director)
    tmdb_movies['actor_1'] = credits['cast'].apply(lambda x: safe_access
(x, [1, 'name']))
    tmdb_movies['actor_2'] = credits['cast'].apply(lambda x: safe_access
(x, [2, 'name']))
    tmdb_movies['actor_3'] = credits['cast'].apply(lambda x: safe_access
(x, [3, 'name']))
    return tmdb_movies
```

In [5]:
```
import sys

!{sys.executable} -m pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /anaconda3/lib/python3.
7/site-packages (0.21.2)
Requirement already satisfied: joblib>=0.11 in /anaconda3/lib/python3.
7/site-packages (from scikit-learn) (0.13.2)
Requirement already satisfied: numpy>=1.11.0 in /anaconda3/lib/python3.
7/site-packages (from scikit-learn) (1.16.4)
Requirement already satisfied: scipy>=0.17.0 in /anaconda3/lib/python3.
7/site-packages (from scikit-learn) (1.3.0)
```

In [ ]:

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import math, nltk
from nltk.corpus import wordnet
from sklearn.neighbors import NearestNeighbors

PS = nltk.stem.PorterStemmer()
credits = load_credits("./tmdb_5000_credits.csv")
movies = load_movies("./tmdb_5000_movies.csv")
data_initial = convert_to_original_format(movies, credits)
print('Shape:',data_initial.shape)
tab_info=pd.DataFrame(data_initial.dtypes).T.rename(index={0:'column typ
e'})
tab_info=tab_info.append(pd.DataFrame(data_initial.isnull().sum()).T.ren
ame(index={0:'null values'}))
tab_info=tab_info.append(pd.DataFrame(data_initial.isnull().sum()/data_i
nitial.shape[0]*100).T.
                        rename(index={0:'null values (%)'}))
tab_info
```

```
Shape: (4803, 27)
```

Out[7]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview
column type	int64	object	object	int64	object	object	object	object
null values	0	0	3091	0	0	0	0	3
null values (%)	0	0	64.3556	0	0	0	0	0.062461

3 rows × 27 columns

In [8]:
```
tab_info.describe()
```

Out[8]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	pop
count	3	3	3	3	3	3	3	3	
unique	2	2	3	2	2	2	2	3	
top	0	0	object	0	0	0	0	object	
freq	2	2	1	2	2	2	2	1	

4 rows × 27 columns

In [9]:
```
data_initial.describe()
```

Out[9]:

	budget	id	popularity	revenue	runtime	vote_average	vc
count	4.803000e+03	4803.000000	4803.000000	4.803000e+03	4801.000000	4803.000000	480
mean	2.904504e+07	57165.484281	21.492301	8.226064e+07	106.875859	6.092172	69
std	4.072239e+07	88694.614033	31.816650	1.628571e+08	22.611935	1.194612	123
min	0.000000e+00	5.000000	0.000000	0.000000e+00	0.000000	0.000000	
25%	7.900000e+05	9014.500000	4.668070	0.000000e+00	94.000000	5.600000	5
50%	1.500000e+07	14629.000000	12.921594	1.917000e+07	103.000000	6.200000	23
75%	4.000000e+07	58610.500000	28.313505	9.291719e+07	118.000000	6.800000	73
max	3.800000e+08	459488.000000	875.581305	2.787965e+09	338.000000	10.000000	1375

In [10]:
```
data_initial[['title', 'genres', 'year', 'vote_average', 'director', 'actor_1', 'actor_2', 'actor_3']].head(10)
```

Out[10]:

	title	genres	year	vote_average	director	actor_1	act
0	Avatar	Action|Adventure|Fantasy|Science Fiction	2009.0	7.2	James Cameron	Zoe Saldana	Sigou We
1	Pirates of the Caribbean: At World's End	Adventure|Fantasy|Action	2007.0	6.9	Gore Verbinski	Orlando Bloom	I Knig
2	Spectre	Action|Adventure|Crime	2015.0	6.3	Sam Mendes	Christoph Waltz	Sey
3	The Dark Knight Rises	Action|Crime|Drama|Thriller	2012.0	7.6	Christopher Nolan	Michael Caine	Olc
4	John Carter	Action|Adventure|Science Fiction	2012.0	6.1	Andrew Stanton	Lynn Collins	Sama Mc
5	Spider-Man 3	Fantasy|Action|Adventure	2007.0	5.9	Sam Raimi	Kirsten Dunst	Ja Fr
6	Tangled	Animation|Family	2010.0	7.4	Byron Howard	Mandy Moore	D Mu
7	Avengers: Age of Ultron	Action|Adventure|Science Fiction	2015.0	7.3	Joss Whedon	Chris Hemsworth	I Ru
8	Harry Potter and the Half-Blood Prince	Adventure|Fantasy|Family	2009.0	7.4	David Yates	Rupert Grint	E Wa
9	Batman v Superman: Dawn of Justice	Action|Adventure|Fantasy	2016.0	5.7	Zack Snyder	Henry Cavill	Gal G

keywords search

As a movie recommendation system, it allows users to input some keywords and the system provides some recommendations. Films described by similar keywords should have similar contents. Therefore, keywords play as an important factor in our system. Firstly, list the keywords which are in the dataset:

In [11]:
```
set_keywords = set()
for keyword in data_initial['keywords'].str.split('|').values:
    if isinstance(keyword, float): continue
    set_keywords = set_keywords.union(keyword)
```

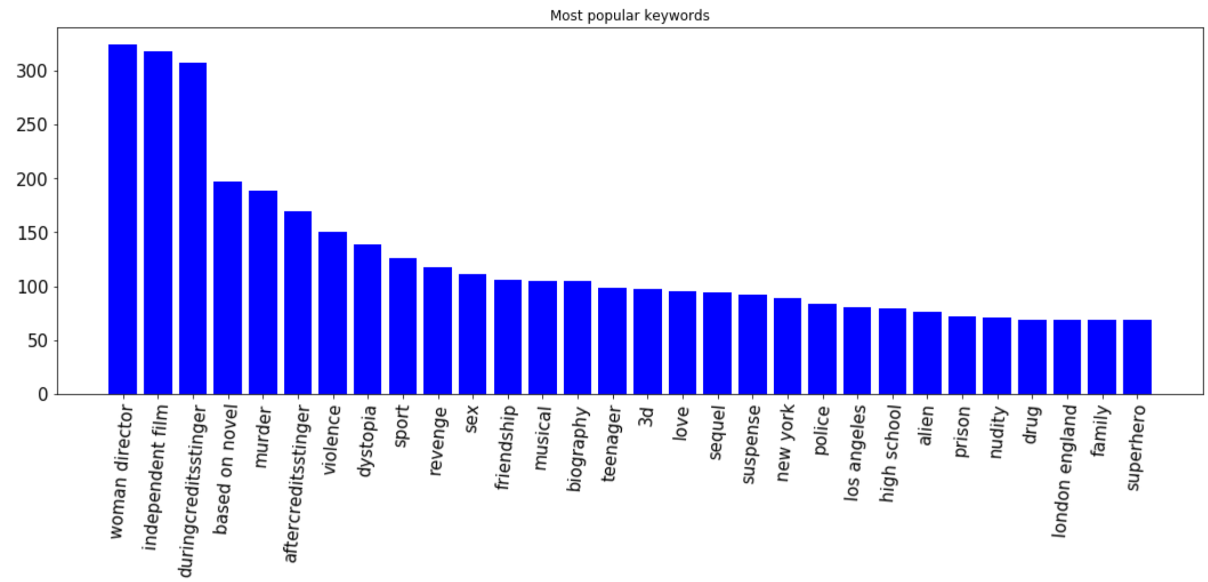
In [12]:
```
def count_word(df, col, lis):
    count = dict()
    for s in lis: count[s] = 0
    for keywords in df[col].str.split('|'):
        if type(keywords) == float and pd.isnull(keywords): continue
        for s in [s for s in keywords if s in lis]:
            if pd.notnull(s): count[s] += 1
    occurences = []
    for k,v in count.items():
        occurences.append([k,v])
    occurences.sort(key = lambda x:x[1], reverse = True)
    return occurences, count
```

In [13]:
```
occurences, keyword_count = count_word(data_initial, 'keywords', set_key
words)
occurences[:5]
```

Out[13]:
```
[['', 412],
 ['woman director', 324],
 ['independent film', 318],
 ['duringcreditsstinger', 307],
 ['based on novel', 197]]
```

In [14]:
```
occurences = [x for x in occurences if x[0]]
```

In [15]:
```
fig = plt.figure(1, figsize=(18,13))
plot = fig.add_subplot(2,1,2)
tmp = occurences[0:30]
y_axis = [i[1] for i in tmp]
x_axis = [k for k,i in enumerate(tmp)]
x_label = [i[0] for i in tmp]
plt.xticks(rotation=85, fontsize = 15)
plt.yticks(fontsize = 15)
plt.xticks(x_axis, x_label)
plt.title('Most popular keywords')
plot.bar(x_axis, y_axis, color='blue')
plt.show()
```



As in every analysis, we will have to deal with the missing values

In [16]:
```
missing_data = data_initial.isnull().sum(axis=0).reset_index()
missing_data.columns = ['column_name', 'missing_count']
missing_data['filling_factor'] = (data_initial.shape[0]
                                  - missing_data['missing_count']) / data_
initial.shape[0] * 100
missing_data.sort_values('filling_factor').reset_index(drop = True)
```

Out[16]:

	column_name	missing_count	filling_factor
0	homepage	3091	35.644389
1	tagline	844	82.427649
2	country	174	96.377264
3	actor_3	93	98.063710
4	language	86	98.209452
5	actor_2	63	98.688320
6	actor_1	53	98.896523
7	director	30	99.375390
8	overview	3	99.937539
9	runtime	2	99.958359
10	release_date	1	99.979180
11	year	1	99.979180
12	production_countries	0	100.000000
13	genres	0	100.000000
14	id	0	100.000000
15	keywords	0	100.000000
16	original_language	0	100.000000
17	vote_count	0	100.000000
18	vote_average	0	100.000000
19	title	0	100.000000
20	original_title	0	100.000000
21	status	0	100.000000
22	spoken_languages	0	100.000000
23	popularity	0	100.000000
24	revenue	0	100.000000
25	production_companies	0	100.000000
26	budget	0	100.000000

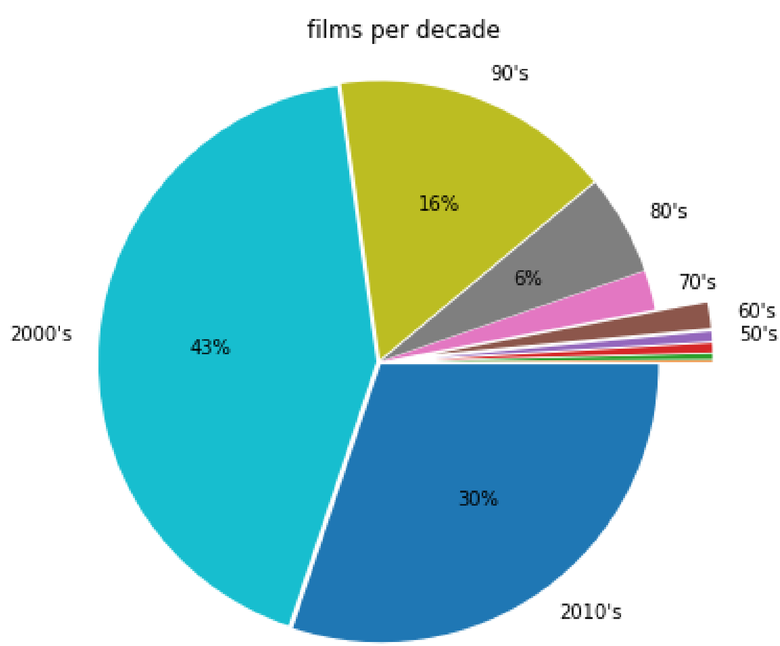
Based on the result above, I found that only 2 of them have a filling factor below 90%.

Number of films per decade

Group the films by decades and show in a pie chart.

In [17]:
```
data_initial['decade'] = data_initial['year'].apply(lambda x:((x-1900)//10)*10)
def get_stats(gr):
    return {'min':gr.min(),'max':gr.max(),'count': gr.count(),'mean':gr.mean()}
test = data_initial['year'].groupby(data_initial['decade']).apply(get_stats).unstack()
```

In [18]:
```
def label(s):
    val = (1900 + s, s)[s < 100]
    chaine = '' if s < 50 else "{}'s".format(int(val))
    return chaine
f, ax = plt.subplots(figsize=(11, 6))
labels = [label(s) for s in  test.index]
sizes  = test['count'].values
explode = [0.2 if sizes[i] < 100 else 0.01 for i in range(11)]
ax.pie(sizes, explode = explode, labels=labels,
       autopct = lambda x:'{:1.0f}%'.format(x) if x > 3 else '')
ax.axis('equal')
ax.set_title('films per decade');
data_initial.drop('decade', axis=1, inplace = True)
```



Genres

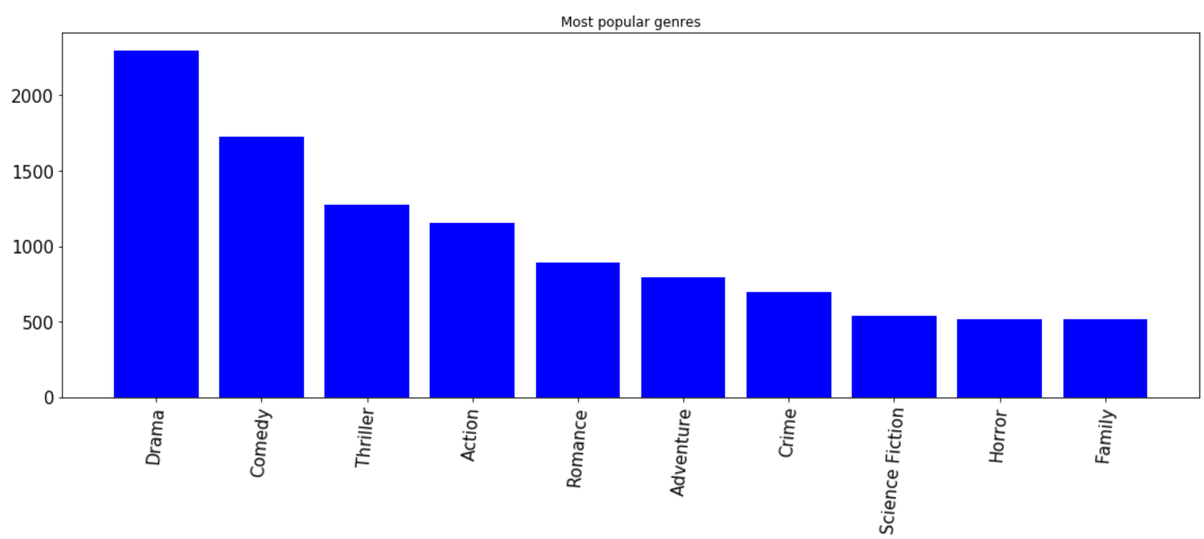
I want to explore genres. Firstly, list the genres which are in the dataset

In [19]:
```
genre_labels = set()
for s in data_initial['genres'].str.split(',').values:
    genre_labels = genre_labels.union(set(s))
```

In [20]:
```
occurences, dum = count_word(data_initial, 'genres', genre_labels)
occurences = [x for x in occurences if x[0]]
occurences[:5]
```

Out[20]:
```
[['Drama', 2297],
 ['Comedy', 1722],
 ['Thriller', 1274],
 ['Action', 1154],
 ['Romance', 894]]
```

In [21]:
```
fig = plt.figure(1, figsize=(18,13))
plot = fig.add_subplot(2,1,2)
tmp = occurences[0:10]
y_axis = [i[1] for i in tmp]
x_axis = [k for k,i in enumerate(tmp)]
x_label = [i[0] for i in tmp]
plt.xticks(rotation=85, fontsize = 15)
plt.yticks(fontsize = 15)
plt.xticks(x_axis, x_label)
plot.bar(x_axis, y_axis, color='blue')
plt.title('Most popular genres')
plt.show()
```



Dealing with duplicates

In [22]:
```
dup_entries = data_initial[data_initial.id.duplicated()]
dup_entries.shape
```

Out[22]: (0, 27)

In [23]:
```
data_temp = data_initial
list_dups = data_temp['title'].map(data_temp['title'].value_counts() > 1
)
```

In [24]:
```
#Show the movies with same title and the director of these movies.
data_temp[list_dups][['title', 'director', 'year']].sort_values('title')
```

Out[24]:

	title	director	year
1359	Batman	Tim Burton	1989.0
4267	Batman	Leslie H. Martinson	1966.0
3647	Out of the Blue	Dennis Hopper	1980.0
3693	Out of the Blue	Robert Sarkies	2006.0
972	The Host	Andrew Niccol	2013.0
2877	The Host	Bong Joon-ho	2006.0

In [49]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import datasets
from sklearn import svm
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
```

In [50]:
```
data_initial
```

Out[50]:

	budget	genres	homepag
0	237000000	Action|Adventure|Fantasy|Science Fiction	http://www.avatarmovie.com
1	300000000	Adventure|Fantasy|Action	http://disney.go.com/disneypictures/pirates
2	245000000	Action|Adventure|Crime	http://www.sonypictures.com/movies/spectre
3	250000000	Action|Crime|Drama|Thriller	http://www.thedarkknightrises.com
4	260000000	Action|Adventure|Science Fiction	http://movies.disney.com/john-carte
5	258000000	Fantasy|Action|Adventure	http://www.sonypictures.com/movies/spider-man3
6	260000000	Animation|Family	http://disney.go.com/disneypictures/tangled
7	280000000	Action|Adventure|Science Fiction	http://marvel.com/movies/movie/193/avengers_ag..
8	250000000	Adventure|Fantasy|Family	http://harrypotter.warnerbros.com/harrypottera..
9	250000000	Action|Adventure|Fantasy	http://www.batmanvsupermandawnofjustice.com
10	270000000	Adventure|Fantasy|Action|Science Fiction	http://www.superman.con
11	200000000	Adventure|Action|Thriller|Crime	http://www.mgm.com/view/movie/234/Quantum-of-S..
12	200000000	Adventure|Fantasy|Action	http://disney.go.com/disneypictures/pirates

	budget	genres	homepage
13	255000000	Action|Adventure|Western	http://disney.go.com/the-lone-ranger
14	225000000	Action|Adventure|Fantasy|Science Fiction	http://www.manofsteel.com
15	225000000	Adventure|Family|Fantasy	NaN
16	220000000	Science Fiction|Action|Adventure	http://marvel.com/avengers_movie
17	380000000	Adventure|Action|Fantasy	http://disney.go.com/pirates/index-on-stranger...
18	225000000	Action|Comedy|Science Fiction	http://www.sonypictures.com/movies/meninblack3
19	250000000	Action|Adventure|Fantasy	http://www.thehobbit.com
20	215000000	Action|Adventure|Fantasy	http://www.theamazingspiderman.com
21	200000000	Action|Adventure	http://www.robinhoodthemovie.com
22	250000000	Adventure|Fantasy	http://www.thehobbit.com
23	180000000	Adventure|Fantasy	http://www.goldencompassmovie.com/index_german...
24	207000000	Adventure|Drama|Action	NaN
25	200000000	Drama|Romance|Thriller	http://www.titanicmovie.com

	budget	genres	homepage
26	250000000	Adventure|Action|Science Fiction	http://marvel.com/captainamericapremiere
27	209000000	Thriller|Action|Adventure|Science Fiction	NaN
28	150000000	Action|Adventure|Science Fiction|Thriller	http://www.jurassicworld.com
29	200000000	Action|Adventure|Thriller	http://www.skyfall-movie.com
...	...	...	..
4773	27000	Comedy	http://www.miramax.com/movie/clerks
4774	27000	Drama|Romance	NaN
4775	0	Drama|Comedy	NaN
4776	0	Comedy|Drama	NaN
4777	0	Drama	NaN
4778	0	Action|Drama|Crime|Thriller	NaN
4779	0	Comedy	NaN
4780	0	Thriller|Crime|Drama	NaN
4781	22000	Comedy|Romance	https://www.facebook.com/DrySpellMovie
4782	0	Drama|Family	NaN

	budget	genres	homepage
4783	0	Thriller|Horror	NaN
4784	0	Drama|Comedy|Romance	http://www.thepuffychairmovie.com
4785	0	Drama	NaN
4786	0	Comedy|Romance	NaN
4787	0	Science Fiction|Thriller	NaN
4788	12000	Horror|Comedy|Crime	NaN
4789	0	Drama	NaN
4790	0	Drama|Foreign	NaN
4791	13	Horror	http://tincanmanthemovie.com
4792	20000	Crime|Horror|Mystery|Thriller	NaN
4793	0	Drama	NaN
4794	0	Thriller|Horror|Comedy	NaN
4795	0	Drama	NaN
4796	7000	Science Fiction|Drama|Thriller	http://www.primermovie.com

	budget	genres	homepage
4797	0	Foreign|Thriller	NaN
4798	220000	Action|Crime|Thriller	NaN
4799	9000	Comedy|Romance	NaN
4800	0	Comedy|Drama|Romance|TV Movie	http://www.hallmarkchannel.com/signedsealeddel...
4801	0		http://shanghaicalling.com
4802	0	Documentary	NaN

4803 rows × 27 columns

Data selection: 1. As in our data analysis, we can see the movie data collected before and after 2010 have relatively equal amount, so we decided to use past(<2010) to predict for future(>2010)

1. As in our genres analysis, we can see the drama movie has the most amount of data which shows its popularity, we will study futher by developing a regression model only of drama movies and compare with the all movie regression results

In [101]:
```
train_set = data_initial[data_initial.year < 2010]
test_set = data_initial[data_initial.year >= 2010]
for col in train_set.columns:
    print(col)
```

```
budget
genres
homepage
id
keywords
original_language
original_title
overview
popularity
production_companies
production_countries
release_date
revenue
runtime
spoken_languages
status
tagline
title
vote_average
vote_count
year
country
language
director
actor_1
actor_2
actor_3
```

Liner regression of Revenue on Budget,release_date,runtime,vote_count and year

In [100]:
```
from math import sqrt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as pl
from sklearn.metrics import r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn import svm
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets

feature_cols = ['budget',
'release_date',
 'runtime',
'vote_count',
'vote_average'
]
x_train = train_set.loc[:, [column_name for column_name in feature_cols
]].values
y_train = train_set.revenue
x_test = test_set.loc[:, [column_name for column_name in feature_cols]].
values
y_test = test_set.revenue

# label encoder to transform our data
le = LabelEncoder()
for i in range(len(feature_cols)):
    x_train[:,i] = le.fit_transform(x_train[:,i])
for i in range(len(feature_cols)):
    x_test[:,i] = le.fit_transform(x_test[:,i])

lm = LinearRegression(normalize=True)
lm.fit(x_train,y_train)


y_pred = lm.predict(x_test)
rmse = sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)


# The coefficients
print ("Coefficient", lm.coef_)
# The intercepts
print ("Intercept: ", lm.intercept_)
# mean square error
print ("Mean Square Error: ", np.mean((lm.predict(x_test) - y_test) ** 2
))
print ("Variance Score: ", lm.score(x_test, y_test))
# we have a 0.35 r2 score
print ("R2 Score is:", r2)
```

```
Coefficient [  140682.14470851     2289.74459353   775034.80071044   26
8278.99289734
 -1562024.79072582]
Intercept:  -15773714.691070095
Mean Square Error:  2.5535886670646824e+16
Variance Score:  0.3523001126835744
R2 Score is: 0.3523001126835744
```

regression of revenue prediction of drama movies

In [98]:
```
train_set2 = train_set[train_set.genres.str.contains("Drama")]
test_set2 = test_set[test_set.genres.str.contains("Drama")]
```

Out[98]:

	budget	genres	h
24	207000000	Adventure|Drama|Action	
25	200000000	Drama|Romance|Thriller	http://www.titanicm
60	200000000	Animation|Drama	http://disney.go.com/disneypictures/ach
65	185000000	Drama|Action|Crime|Thriller	http://thedarkknight.warnerbros.com
100	150000000	Fantasy|Drama|Thriller|Mystery|Romance	http://www.benjaminbu
104	160000000	Adventure|Action|Drama|Thriller	http://www2.warnerbros.com/
112	155000000	War|History|Action|Adventure|Drama|Romance	
116	150000000	Drama|Horror|Action|Thriller|Science Fiction	http://iamlegend.warnerb
119	150000000	Action|Crime|Drama	http://www2.warnerbros.com/batmanbegins
145	175000000	Adventure|Drama|War	
156	140000000	Drama|Action|War|History	
165	137000000	Drama|Action|Science Fiction	
167	130000000	Action|Adventure|Comedy|Drama|Mystery	

	budget	genres	ho
180	70000000	Action|Drama|Mystery|Thriller	http://www.universalstudiosentertainmen
192	130000000	Drama	
214	120000000	Drama	
236	115000000	Drama|Action|History|War	
246	110000000	Drama|History|War|Action	
248	110000000	Action|Comedy|Drama|Thriller	
250	116000000	Drama	
264	107000000	Drama	
267	130000000	Drama|Action|Adventure|History|War	http://www.kingdomofheaven
274	103000000	Action|Drama|Adventure	
280	80000000	History|Crime|Drama	http://www.publicene
281	100000000	Drama|Crime	http://www.americangan

	budget	genres	h
283	100000000	Thriller|Drama|Crime	http://www.sonypictures.com/movies/thet
286	100000000	Action|Drama|Mystery|Thriller	http://www.warnerbros.co
288	100000000	Drama|Animation|Family	
297	100000000	Drama|Thriller|Action	
314	10000000	Crime|Drama|Mystery|Thriller	
...	...	...	
493	60000000	Drama|Romance	http://www.abeautifulm
494	45000000	Family|Animation|Drama	http://movies.disney.com/the
510	76000000	Drama|Action|Thriller|Science Fiction	http://www.universalstudiosentertainmen
521	60000000	Comedy|Drama	http://www.theterminal-them
524	75000000	Comedy|Drama|History	http://www.charliewilsor
526	70000000	Drama	http://www.dreamgirlsm
528	70000000	Drama|Action|History|Thriller	

	budget	genres	h
529	70000000	Action|Drama|War	
535	55000000	Action|Adventure|Drama|Romance	
536	75000000	Drama|History|Romance	
538	52000000	Mystery|Drama|Thriller|Crime	
542	0	Action|Drama|Horror|Science Fiction|Thriller	
544	45000000	Action|Adventure|Drama|Thriller	
545	75000000	Science Fiction|Thriller|Drama	
552	75000000	Comedy|Drama	http://www.funnypeoplem
553	70000000	Thriller|Action|Drama	http://www.thekingdomm
556	72000000	Action|Drama|History|War	
557	72000000	Drama|War	
559	72000000	Drama|Romance	
561	74500000	Adventure|Drama|Family	

	budget	genres	h
562	60000000	Drama|Mystery|Thriller	
564	72000000	Drama|Thriller|Science Fiction|Mystery	
571	70000000	Drama|Action|Thriller|War	http://www.inglouriousbasterds-m
575	68000000	Drama|Mystery|Romance|Science Fiction|Thriller	
576	75000000	Drama|Thriller|Fantasy|Mystery	
583	70000000	Adventure|Fantasy|Drama	
590	70000000	Drama|Action|Thriller|Crime	
592	70000000	Adventure|Drama|History	
599	70000000	Drama|War	
611	68000000	Thriller|Action|Drama	http://www.paramount.com/movies/sum

100 rows × 27 columns

In [99]:
```
feature_cols = ['budget',
'release_date',
 'runtime',
'vote_count',
'vote_average'
]
x_train = train_set2.loc[:, [column_name for column_name in feature_cols
]].values
y_train = train_set2.revenue
x_test = test_set2.loc[:, [column_name for column_name in feature_cols]]
.values
y_test = test_set2.revenue

# label encoder to transforme the data
le = LabelEncoder()
for i in range(len(feature_cols)):
    x_train[:,i] = le.fit_transform(x_train[:,i])
for i in range(len(feature_cols)):
    x_test[:,i] = le.fit_transform(x_test[:,i])

lm = LinearRegression(normalize=True)
lm.fit(x_train,y_train)

y_pred = lm.predict(x_test)
rmse = sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)


# The coefficients
print ("Coefficient", lm.coef_)
# The intecept
print ("Intercept: ", lm.intercept_)
# lower mean squared error than first model
print ("Mean Square Error: ", np.mean((lm.predict(x_test) - y_test) ** 2
))
print ("Variance Score: ", lm.score(x_test, y_test))
# we have a 0.3 r2 score
print ("R2 Score is:", r2)
```

```
Coefficient [  166008.34217393   -10344.49772209   470666.89483907   28
8108.14564433
 -1346243.29103771]
Intercept:  2839323.7121284977
Mean Square Error:  1.0998356300853142e+16
Variance Score:  0.30324713615510035
R2 Score is: 0.30324713615510035
```

In [ ]: