In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
movies_metadata = pd.read_csv('movies_metadata.csv')
movies_metadata.head()
```

```
/Users/Jayashri/anaconda/lib/python3.6/site-packages/IPython/core/inte
ractiveshell.py:2717: DtypeWarning: Columns (10) have mixed types. Spe
cify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[2]:

	adult	belongs_to_collection	budget	genres	homepage	id	imd
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt011
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt011
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt011
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt011
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11862	tt011

5 rows × 24 columns

In [3]:
```
movies_metadata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
adult                     45466 non-null object
belongs_to_collection     4494 non-null object
budget                    45466 non-null object
genres                    45466 non-null object
homepage                  7782 non-null object
id                        45466 non-null object
imdb_id                   45449 non-null object
original_language         45455 non-null object
original_title            45466 non-null object
overview                  44512 non-null object
popularity                45461 non-null object
poster_path               45080 non-null object
production_companies      45463 non-null object
production_countries      45463 non-null object
release_date              45379 non-null object
revenue                   45460 non-null float64
runtime                   45203 non-null float64
spoken_languages          45460 non-null object
status                    45379 non-null object
tagline                   20412 non-null object
title                     45460 non-null object
video                     45460 non-null object
vote_average              45460 non-null float64
vote_count                45460 non-null float64
dtypes: float64(4), object(20)
memory usage: 8.3+ MB
```

In [4]:
```
# Cleanup by dropping all the rows with null values

movies_cols_df = movies_metadata[['title', 'revenue', 'budget', 'genres',
                                  'runtime', 'vote_average', 'vote_count']
movies_cols_df.head()
```

Out[4]:

	title	revenue	budget	genres	release_date	runtime	vote_average	vote_coun
0	Toy Story	373554033.0	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	1995-10-30	81.0	7.7	5415.
1	Jumanji	262797249.0	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	1995-12-15	104.0	6.9	2413.
2	Grumpier Old Men	0.0	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	1995-12-22	101.0	6.5	92.
3	Waiting to Exhale	81452156.0	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	1995-12-22	127.0	6.1	34.
4	Father of the Bride Part II	76578911.0	0	[{'id': 35, 'name': 'Comedy'}]	1995-02-10	106.0	5.7	173.

In [5]:
```
movies_metadata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
adult                     45466 non-null object
belongs_to_collection     4494 non-null object
budget                    45466 non-null object
genres                    45466 non-null object
homepage                  7782 non-null object
id                        45466 non-null object
imdb_id                   45449 non-null object
original_language         45455 non-null object
original_title            45466 non-null object
overview                  44512 non-null object
popularity                45461 non-null object
poster_path               45080 non-null object
production_companies      45463 non-null object
production_countries      45463 non-null object
release_date              45379 non-null object
revenue                   45460 non-null float64
runtime                   45203 non-null float64
spoken_languages          45460 non-null object
status                    45379 non-null object
tagline                   20412 non-null object
title                     45460 non-null object
video                     45460 non-null object
vote_average              45460 non-null float64
vote_count                45460 non-null float64
dtypes: float64(4), object(20)
memory usage: 8.3+ MB
```

In [6]:
```
movies_cols_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 8 columns):
title           45460 non-null object
revenue         45460 non-null float64
budget          45466 non-null object
genres          45466 non-null object
release_date    45379 non-null object
runtime         45203 non-null float64
vote_average    45460 non-null float64
vote_count      45460 non-null float64
dtypes: float64(4), object(4)
memory usage: 2.8+ MB
```

In [7]:
```
# Delete all the rows where budget = 0 or revenue = 0
movies_cols_df = movies_cols_df[(movies_cols_df['budget'] != 0)
                                & (movies_cols_df['revenue'] != 0)]
```

In [8]:
```
movies_cols_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7414 entries, 0 to 45422
Data columns (total 8 columns):
title           7408 non-null object
revenue         7408 non-null float64
budget          7414 non-null object
genres          7414 non-null object
release_date    7410 non-null object
runtime         7402 non-null float64
vote_average    7408 non-null float64
vote_count      7408 non-null float64
dtypes: float64(4), object(4)
memory usage: 521.3+ KB
```

In [9]:
```
# Drop all rows with any null value
movies_cols_df = movies_cols_df.dropna()
movies_cols_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7401 entries, 0 to 45422
Data columns (total 8 columns):
title           7401 non-null object
revenue         7401 non-null float64
budget          7401 non-null object
genres          7401 non-null object
release_date    7401 non-null object
runtime         7401 non-null float64
vote_average    7401 non-null float64
vote_count      7401 non-null float64
dtypes: float64(4), object(4)
memory usage: 520.4+ KB
```

In [10]:
```
movies_cols_df['genres'].iloc[0]
```

Out[10]:
```
"[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id'
: 10751, 'name': 'Family'}]"
```

In [11]:
```
import ast
# """Return a list with tokens"""
def get_token(x):
    alist = []  # empty list
    for token in x:
       alist.append(token['name'])

    return alist

print(get_token(ast.literal_eval("[{'id': 12, 'name': 'Adventure'}, {'id'
```

```
['Adventure', 'Action', 'Thriller']
```

In [12]:
```
# create a new column 'genre_list'
movies_cols_df['genre_list'] = movies_cols_df['genres'].apply(lambda x :
                                                              get_token(a
```

In [13]:
```
movies_cols_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7401 entries, 0 to 45422
Data columns (total 9 columns):
title           7401 non-null object
revenue         7401 non-null float64
budget          7401 non-null object
genres          7401 non-null object
release_date    7401 non-null object
runtime         7401 non-null float64
vote_average    7401 non-null float64
vote_count      7401 non-null float64
genre_list      7401 non-null object
dtypes: float64(4), object(5)
memory usage: 578.2+ KB
```

In [14]:
```
movies_cols_df.head()
```

Out[14]:

	title	revenue	budget	genres	release_date	runtime	vote_average	vote_count
0	Toy Story	373554033.0	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	1995-10-30	81.0	7.7	5415.0
1	Jumanji	262797249.0	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	1995-12-15	104.0	6.9	2413.0
3	Waiting to Exhale	81452156.0	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	1995-12-22	127.0	6.1	34.0
4	Father of the Bride Part II	76578911.0	0	[{'id': 35, 'name': 'Comedy'}]	1995-02-10	106.0	5.7	173.0
5	Heat	187436818.0	60000000	[{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...	1995-12-15	170.0	7.7	1886.0

In [15]:
```
#get number of titles
N = movies_cols_df['title'].count()
N
```

Out[15]: 7401

In [16]:
```
# Ref: https://stackoverflow.com/questions/12680754/
# split-explode-pandas-dataframe-string-entry-to-separate-rows/40449726
# lst_cols contains the list of columns to explode
def explode(df, lst_cols, fill_value=''):
    # make sure `lst_cols` is a list
    if lst_cols and not isinstance(lst_cols, list):
        lst_cols = [lst_cols]
    # all columns except `lst_cols`
    idx_cols = df.columns.difference(lst_cols)

    # calculate lengths of lists
    lens = df[lst_cols[0]].str.len()

    if (lens > 0).all():
        # ALL lists in cells aren't empty
        return pd.DataFrame({
            col:np.repeat(df[col].values, lens)
            for col in idx_cols
        }).assign(**{col:np.concatenate(df[col].values) for col in lst_co
          .loc[:, df.columns]
    else:
        # at least one list in cells is empty
        return pd.DataFrame({
            col:np.repeat(df[col].values, lens)
            for col in idx_cols
        }).assign(**{col:np.concatenate(df[col].values) for col in lst_co
          .append(df.loc[lens==0, idx_cols]).fillna(fill_value) \
          .loc[:, df.columns]
```

In [17]:
```
mymovie_df = explode(movies_cols_df, ['genre_list'], fill_value='')
```

In [18]:
```
mymovie_df.head()
```

Out[18]:

	title	revenue	budget	genres	release_date	runtime	vote_average	vote_count
0	Toy Story	373554033.0	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	1995-10-30	81.0	7.7	5415.0
1	Toy Story	373554033.0	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	1995-10-30	81.0	7.7	5415.0
2	Toy Story	373554033.0	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	1995-10-30	81.0	7.7	5415.0
3	Jumanji	262797249.0	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	1995-12-15	104.0	6.9	2413.0
4	Jumanji	262797249.0	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	1995-12-15	104.0	6.9	2413.0

In [19]:
```
mymovie_df.count()
```

Out[19]:
```
title           18511
revenue         18511
budget          18511
genres          18511
release_date    18511
runtime         18511
vote_average    18511
vote_count      18511
genre_list      18511
dtype: int64
```

In [20]:
```
mymovie_df.rename(columns= {"genre_list" : "movie_genre"}, inplace = True
```

In [21]:
```
mymovie_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18511 entries, 0 to 44152
Data columns (total 9 columns):
title           18511 non-null object
revenue         18511 non-null float64
budget          18511 non-null object
genres          18511 non-null object
release_date    18511 non-null object
runtime         18511 non-null float64
vote_average    18511 non-null float64
vote_count      18511 non-null float64
movie_genre     18511 non-null object
dtypes: float64(4), object(5)
memory usage: 1.4+ MB
```

In [22]:
```
mymovie_df['movie_genre'].value_counts()
```

Out[22]:
```
Drama              3681
Comedy             2603
Thriller           1872
Action             1736
Romance            1437
Adventure          1120
Crime              1086
Science Fiction     747
Horror              735
Family              678
Fantasy             630
Mystery             550
Animation           385
History             295
Music               267
War                 244
Documentary         221
Western             117
Foreign              84
                     22
TV Movie              1
Name: movie_genre, dtype: int64
```

In [23]:
```
type(mymovie_df['budget'].iloc[0])
```

Out[23]:
```
str
```

In [24]:
```
type(mymovie_df['revenue'].iloc[0])
```

Out[24]:
```
numpy.float64
```

In [25]:
```
#convert budget column to a float
mymovie_df['budget'] = pd.to_numeric(mymovie_df['budget'])
```

In [26]:
```
type(mymovie_df['budget'].iloc[0])
```

Out[26]:
```
numpy.int64
```

In [27]:
```
my_genre_list = ['Drama', 'Comedy', 'Thriller', 'Action', 'Romance', 'Adv
                 'Crime', 'Science Fiction', 'Horror', 'Family', 'Fantasy'
                 'Animation', 'History', 'Music', 'War', 'Western', 'Docum

for item in my_genre_list:
    print(item)
    print('Correlation Matrix for genre = ', item)
    df = mymovie_df[mymovie_df['movie_genre'] == item]
    print(df[['budget', 'revenue', 'runtime', 'vote_average']].corr())
    print('----------------------------------------')
```

```
----------------------------------------
Horror
Correlation Matrix for genre =  Horror
                budget   revenue   runtime  vote_average
budget        1.000000  0.621161  0.299906      0.017294
revenue       0.621161  1.000000  0.259144      0.218766
runtime       0.299906  0.259144  1.000000      0.283756
vote_average  0.017294  0.218766  0.283756      1.000000
----------------------------------------
Family
Correlation Matrix for genre =  Family
                budget   revenue   runtime  vote_average
budget        1.000000  0.714648  0.195635      0.187743
revenue       0.714648  1.000000  0.207469      0.332557
runtime       0.195635  0.207469  1.000000      0.198776
vote_average  0.187743  0.332557  0.198776      1.000000
----------------------------------------
Fantasy
Correlation Matrix for genre =  Fantasy
                budget   revenue   runtime  vote average
```

In [28]:
```
#Data Cleaning -- drop all rows with any 0 value for revenue and budget

mymovie_df = mymovie_df[(mymovie_df['budget'] != 0) &
                        (mymovie_df['revenue'] != 0)]
mymovie_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14019 entries, 0 to 44152
Data columns (total 9 columns):
title           14019 non-null object
revenue         14019 non-null float64
budget          14019 non-null int64
genres          14019 non-null object
release_date    14019 non-null object
runtime         14019 non-null float64
vote_average    14019 non-null float64
vote_count      14019 non-null float64
movie_genre     14019 non-null object
dtypes: float64(4), int64(1), object(4)
memory usage: 1.1+ MB
```

In [ ]:
```
```

In [29]:
```
# Create a column is_Comedy from column movie_genre
def get_genre_val(x, genre):
    if x == genre:
        return 1
    else:
        return 0

mymovie_df['is_Comedy'] = mymovie_df['movie_genre'].apply(lambda x:
                                                  get_genre_val(x, 'Comedy'))
mymovie_df['is_Comedy'].head()
```

Out[29]:
```
0    0
1    1
2    0
3    0
4    0
Name: is_Comedy, dtype: int64
```

In [30]:
```
#Define column for other categories
mymovie_df['is_Drama'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Drama'))
mymovie_df['is_Action'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Action'))
mymovie_df['is_Thriller'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Thriller'))
mymovie_df['is_Romance'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Romance'))
mymovie_df['is_Adventure'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Adventure'))
mymovie_df['is_Crime'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Crime'))
mymovie_df['is_ScienceFict'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Science Ficti
mymovie_df['is_Horror'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Horror'))
mymovie_df['is_Family'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Family'))
mymovie_df['is_Fantasy'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Fantasy'))
mymovie_df['is_Mystery'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Mystery'))
mymovie_df['is_Animation'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Animation'))
mymovie_df['is_History'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'History'))
mymovie_df['is_Music'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Music'))
mymovie_df['is_War'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'War'))
mymovie_df['is_Documentary'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Documentary')
mymovie_df['is_Western'] = mymovie_df['movie_genre'].apply(lambda x:
                                                get_genre_val(x, 'Western'))
```

In [31]:
```
mymovie_df.columns
```

Out[31]:
```
Index(['title', 'revenue', 'budget', 'genres', 'release_date', 'runtim
e',
       'vote_average', 'vote_count', 'movie_genre', 'is_Comedy', 'is_D
rama',
       'is_Action', 'is_Thriller', 'is_Romance', 'is_Adventure', 'is_C
rime',
       'is_ScienceFict', 'is_Horror', 'is_Family', 'is_Fantasy', 'is_M
ystery',
       'is_Animation', 'is_History', 'is_Music', 'is_War', 'is_Documen
tary',
       'is_Western'],
      dtype='object')
```

In [32]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

print('Linear Regression Model')
X = mymovie_df[['budget', 'vote_average', 'is_Comedy', 'is_Drama',
       'is_Action', 'is_Thriller', 'is_Romance', 'is_Adventure',
       'is_Crime','is_ScienceFict', 'is_Horror', 'is_Family',
       'is_Fantasy', 'is_Mystery','is_Animation', 'is_History',
       'is_Music', 'is_War', 'is_Documentary', 'is_Western']]

y = mymovie_df['revenue']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state = 101)
lm = LinearRegression()
lm.fit(X_train, y_train)
print('lm.intercept = ' , lm.intercept_)
coeff_df = pd.DataFrame(lm.coef_, X.columns, columns = ['Coefficients'])
print(coeff_df)

#Predictions for the model
predictions = lm.predict(X_test)
#Scatter plot of y_test and predictions
print('----------------------')
#print('Scatter Plot of y_test and predictions')
#plt.scatter(y_test, predictions)
#Residual histogram
#print('Residual Histogram Plot')
#sns.distplot(y_test - predictions, bins = 50)
print('Evaluation Metrics')
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print('R**2:', metrics.explained_variance_score(y_test, predictions))
```

```
print('R**2:', metrics.explained_variance_score(y_test, predictions))
```

```
Linear Regression Model
lm.intercept =  -220723722.562
                Coefficients
budget           3.004219e+00
vote_average     4.116416e+07
is_Comedy       -3.146942e+07
is_Drama        -5.948486e+07
is_Action       -4.499432e+07
is_Thriller     -4.576934e+07
is_Romance      -3.228734e+07
is_Adventure    -1.913664e+07
is_Crime        -5.386947e+07
is_ScienceFict -3.580142e+07
is_Horror       -1.797424e+07
is_Family       -3.873156e+06
is_Fantasy      -2.077658e+07
is_Mystery      -5.545730e+07
is_Animation    -9.687570e+06
is_History      -8.800441e+07
is_Music        -3.391751e+07
is_War          -7.380587e+07
is_Documentary -4.786747e+07
is_Western      -8.000805e+07
----------------------
Evaluation Metrics
MAE: 67901596.2776
MSE: 1.30811857359e+16
RMSE: 114373011.396
R**2: 0.588861249891
```

In [33]:
```
#Log Transformation
mymovie_df['log_budget'] = np.log(mymovie_df['budget'])
mymovie_df['log_revenue'] = np.log(mymovie_df['revenue'])
mymovie_df[['budget', 'log_budget', 'revenue', 'log_revenue']].head()
```

Out[33]:

	budget	log_budget	revenue	log_revenue
0	30000000	17.216708	373554033.0	19.738573
1	30000000	17.216708	373554033.0	19.738573
2	30000000	17.216708	373554033.0	19.738573
3	65000000	17.989898	262797249.0	19.386893
4	65000000	17.989898	262797249.0	19.386893

In [34]:
```
#Linear Model with Log Transformation

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

print('Linear Regression Model')
X = mymovie_df[['log_budget', 'vote_average', 'is_Comedy', 'is_Drama',
       'is_Action', 'is_Thriller', 'is_Romance', 'is_Adventure',
       'is_Crime','is_ScienceFict', 'is_Horror', 'is_Family',
       'is_Fantasy', 'is_Mystery','is_Animation', 'is_History',
       'is_Music', 'is_War', 'is_Documentary', 'is_Western']]

y = mymovie_df['log_revenue']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state = 101)
lm = LinearRegression()
lm.fit(X_train, y_train)
print('lm.intercept = ' , lm.intercept_)
coeff_df = pd.DataFrame(lm.coef_, X.columns, columns = ['Coefficients'])
print(coeff_df)

#Predictions for the model
predictions = lm.predict(X_test)
#Scatter plot of y_test and predictions
print('----------------------')
#print('Scatter Plot of y_test and predictions')
#plt.scatter(y_test, predictions)
#Residual histogram
#print('Residual Histogram Plot')
#sns.distplot(y_test - predictions, bins = 50)
print('Evaluation Metrics')
```

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print('R**2:', metrics.explained_variance_score(y_test, predictions))
```

```
Linear Regression Model
lm.intercept =  -2.56267319755
                Coefficients
log_budget          0.870112
vote_average        0.586438
is_Comedy           1.596545
is_Drama            1.098620
is_Action           1.649174
is_Thriller         1.506675
is_Romance          1.346564
is_Adventure        1.795432
is_Crime            1.372911
is_ScienceFict      1.642236
is_Horror           2.033949
is_Family           1.970325
is_Fantasy          1.762462
is_Mystery          1.379897
is_Animation        1.835207
is_History          1.169721
is_Music            1.381629
is_War              1.027423
is_Documentary      0.999010
is_Western          0.941187
----------------------
Evaluation Metrics
MAE: 1.14482324263
MSE: 2.9556070487
RMSE: 1.71918790384
R**2: 0.572116892449
```

In [35]:
```
#Linear regression without transformation and adding 'runtime' to X
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

print('Linear Regression Model')
X = mymovie_df[['budget','vote_average','runtime','is_Comedy','is_Drama',
       'is_Action', 'is_Thriller', 'is_Romance', 'is_Adventure',
       'is_Crime','is_ScienceFict', 'is_Horror', 'is_Family',
       'is_Fantasy', 'is_Mystery','is_Animation', 'is_History',
       'is_Music', 'is_War', 'is_Documentary', 'is_Western']]

y = mymovie_df['revenue']
```

```
y = myMovie_df['revenue']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state = 101)
lm = LinearRegression()
lm.fit(X_train, y_train)
print('lm.intercept = ' , lm.intercept_)
coeff_df = pd.DataFrame(lm.coef_, X.columns, columns = ['Coefficients'])
print(coeff_df)

#Predictions for the model
predictions = lm.predict(X_test)
#Scatter plot of y_test and predictions
print('-----------------------')
#print('Scatter Plot of y_test and predictions')
#plt.scatter(y_test, predictions)
#Residual histogram
#print('Residual Histogram Plot')
#sns.distplot(y_test - predictions, bins = 50)
print('Evaluation Metrics')
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print('R**2:', metrics.explained_variance_score(y_test, predictions))
```

```
Linear Regression Model
lm.intercept =  -239176598.817
                Coefficients
budget          2.972448e+00
vote_average    3.939413e+07
runtime         2.577016e+05
is_Comedy      -2.777745e+07
is_Drama       -5.847826e+07
is_Action      -4.346146e+07
is_Thriller    -4.357280e+07
is_Romance     -3.118601e+07
is_Adventure   -1.660793e+07
is_Crime       -5.165695e+07
is_ScienceFict -3.273207e+07
is_Horror      -1.419526e+07
is_Family       2.016564e+06
is_Fantasy     -1.738855e+07
is_Mystery     -5.322581e+07
is_Animation   -1.146022e+06
is_History     -9.198162e+07
is_Music       -3.215451e+07
is_War         -7.563237e+07
is_Documentary -4.278506e+07
is_Western     -8.051641e+07
```

```
----------------------
Evaluation Metrics
MAE: 68248838.5538
MSE: 1.31092009084e+16
RMSE: 114495418.723
R**2: 0.587987444448
```

In [36]:
```
#Runtime does not increase R^2.
```

In [37]:
```
#Linear Regression with Square root transformation on revenue and budget
#Square root  Transformation
mymovie_df['sqrt_budget'] = np.sqrt(mymovie_df['budget'])
mymovie_df['sqrt_revenue'] = np.sqrt(mymovie_df['revenue'])
mymovie_df[['budget', 'sqrt_budget', 'revenue', 'sqrt_revenue']].head()
```

Out[37]:

	budget	sqrt_budget	revenue	sqrt_revenue
0	30000000	5477.225575	373554033.0	19327.545964
1	30000000	5477.225575	373554033.0	19327.545964
2	30000000	5477.225575	373554033.0	19327.545964
3	65000000	8062.257748	262797249.0	16211.022454
4	65000000	8062.257748	262797249.0	16211.022454

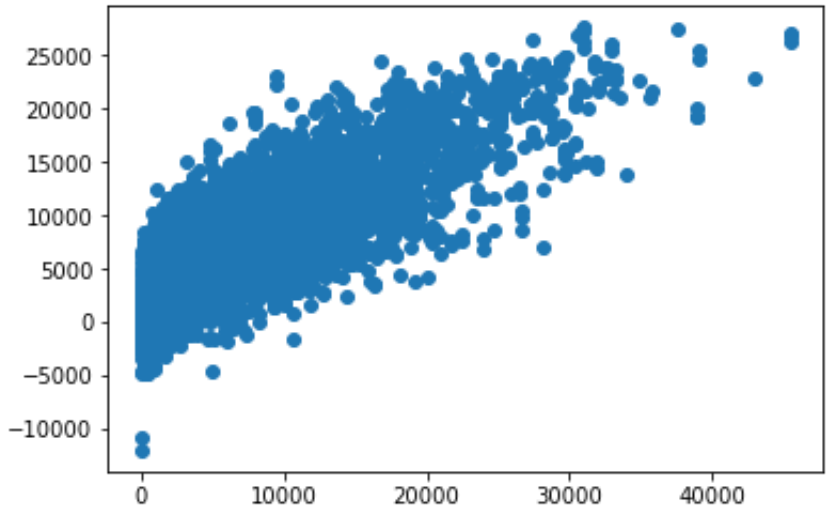
In [38]:
```
#Linear Regression Model with square root transformation

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

print('Linear Regression Model')
X = mymovie_df[['sqrt_budget', 'vote_average', 'is_Comedy', 'is_Drama',
       'is_Action', 'is_Thriller', 'is_Romance', 'is_Adventure',
       'is_Crime','is_ScienceFict', 'is_Horror', 'is_Family',
       'is_Fantasy', 'is_Mystery','is_Animation', 'is_History',
       'is_Music', 'is_War', 'is_Documentary', 'is_Western']]

y = mymovie_df['sqrt_revenue']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state = 101)
lm = LinearRegression()
lm.fit(X_train, y_train)
print('lm.intercept = ' , lm.intercept_)
coeff_df = pd.DataFrame(lm.coef_, X.columns, columns = ['Coefficients'])
print(coeff_df)
```

```
#Predictions for the model
predictions = lm.predict(X_test)
#Scatter plot of y_test and predictions
print('----------------------')
#print('Scatter Plot of y_test and predictions')
#plt.scatter(y_test, predictions)
#Residual histogram
#print('Residual Histogram Plot')
#sns.distplot(y_test - predictions, bins = 50)
print('Evaluation Metrics')
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print('R**2:', metrics.explained_variance_score(y_test, predictions))
```

```
is_Romance       -775.292193
is_Adventure      -16.829304
is_Crime        -1443.435579
is_ScienceFict   -747.581100
is_Horror         424.349766
is_Family         542.730854
is_Fantasy       -194.437639
is_Mystery      -1485.450086
is_Animation      303.142749
is_History      -2790.110527
is_Music         -629.890140
is_War          -2281.148866
is_Documentary  -1145.393443
is_Western      -2229.856326
----------------------
Evaluation Metrics
MAE: 2954.61102162
MSE: 15534536.3132
RMSE: 3941.38761265
R**2: 0.639791254962
```

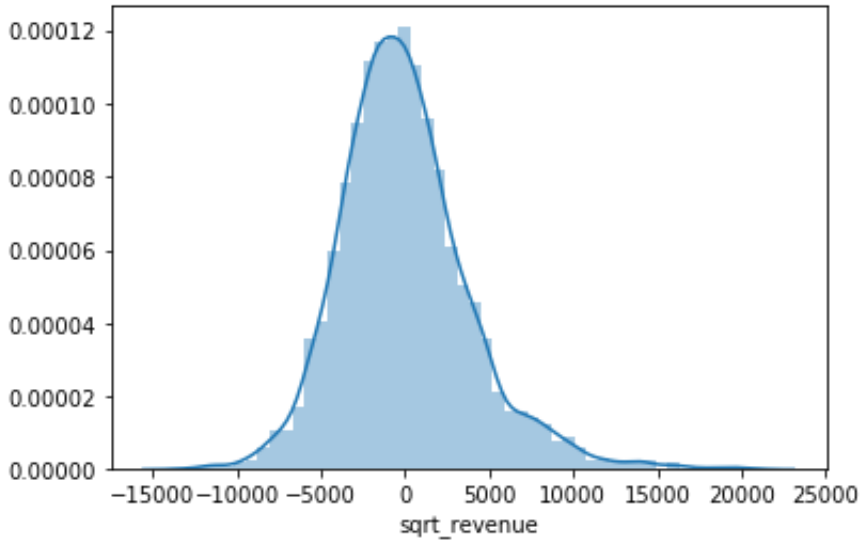
In [39]:
```
#Plot the y_test and predictions
plt.scatter(y_test, predictions)
```

Out[39]: <matplotlib.collections.PathCollection at 0x1a22d30f60>



In [40]:
```
#Histogram of the residuals
sns.distplot((y_test - predictions))
```

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1a22cfc198>



In [41]:
```
#Plot of the residuals looks fairly normally distributed with a mean of 0
```

In [42]:
```
#Linear Regression model with sqrt of revenue on sqrt budget & runtime
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

```
print('Linear Regression Model')
X = mymovie_df[['sqrt_budget','vote_average', 'runtime','is_Comedy',
       'is_Drama','is_Action', 'is_Thriller', 'is_Romance',
       'is_Adventure','is_Crime','is_ScienceFict','is_Horror','is_Family'
       'is_Fantasy', 'is_Mystery','is_Animation', 'is_History',
       'is_Music', 'is_War', 'is_Documentary', 'is_Western']]

y = mymovie_df['sqrt_revenue']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state = 101)
lm = LinearRegression()
lm.fit(X_train, y_train)
print('lm.intercept = ' , lm.intercept_)
coeff_df = pd.DataFrame(lm.coef_, X.columns, columns = ['Coefficients'])
print(coeff_df)

#Predictions for the model
predictions = lm.predict(X_test)
#Scatter plot of y_test and predictions
print('----------------------')
#print('Scatter Plot of y_test and predictions')
#plt.scatter(y_test, predictions)
#Residual histogram
#print('Residual Histogram Plot')
#sns.distplot(y_test - predictions, bins = 50)
print('Evaluation Metrics')
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print('R**2:', metrics.explained_variance_score(y_test, predictions))
```

```
Linear Regression Model
lm.intercept =  -10713.3361773
                Coefficients
sqrt_budget         1.511604
vote_average     1710.809797
runtime             8.343676
is_Comedy         -340.237041
is_Drama         -1803.980620
is_Action         -779.824597
is_Thriller       -952.564219
is_Romance        -717.265352
is_Adventure        86.277584
is_Crime         -1347.606051
is_ScienceFict    -625.968373
is_Horror          563.487479
is_Family          757.724080
is_Fantasy         -62.713630
```

```
is_Mystery      -1390.148571
is_Animation      604.698668
is_History      -2893.138197
is_Music         -552.716169
is_War          -2315.744018
is_Documentary   -976.683594
is_Western      -2228.139027
----------------------
Evaluation Metrics
MAE: 2957.30352232
MSE: 15560358.9591
RMSE: 3944.66208427
R**2: 0.639195631373
```

In [43]:
```
#R^2 has gone down with runtime.
#Best model so far is sqrt transformation.
```

In [44]:
```
#Square root transformation revisited
#Linear Regression Model with square root transformation

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

print('Linear Regression Model')
X = mymovie_df[['sqrt_budget', 'vote_average', 'is_Comedy', 'is_Drama',
       'is_Action', 'is_Thriller', 'is_Romance', 'is_Adventure',
       'is_Crime','is_ScienceFict', 'is_Horror', 'is_Family',
       'is_Fantasy', 'is_Mystery','is_Animation', 'is_History',
       'is_Music', 'is_War', 'is_Documentary', 'is_Western']]

y = mymovie_df['sqrt_revenue']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state = 101)
lm = LinearRegression()
lm.fit(X_train, y_train)
print('lm.intercept = ' , lm.intercept_)
coeff_df = pd.DataFrame(lm.coef_, X.columns, columns = ['Coefficients'])
print(coeff_df)

#Predictions for the model
predictions = lm.predict(X_test)
#Scatter plot of y_test and predictions
print('----------------------')
#print('Scatter Plot of y_test and predictions')
#plt.scatter(y_test, predictions)
#Residual histogram
#print('Residual Histogram Plot')
#sns.distplot(y_test - predictions, bins = 50)
```

```
#sns.distplot(y_test - predictions, bins = 50)
print('Evaluation Metrics')
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print('R**2:', metrics.explained_variance_score(y_test, predictions))
```

```
Linear Regression Model
lm.intercept =  -10143.7525593
                Coefficients
sqrt_budget         1.526340
vote_average     1770.106350
is_Comedy        -481.511392
is_Drama        -1857.362402
is_Action        -852.414567
is_Thriller     -1046.908597
is_Romance       -773.292153
is_Adventure      -16.829304
is_Crime        -1443.435579
is_ScienceFict   -747.581100
is_Horror         424.349766
is_Family         542.730854
is_Fantasy       -194.437639
is_Mystery      -1485.450086
is_Animation      303.142749
is_History      -2790.110527
is_Music         -629.890140
is_War          -2281.148866
is_Documentary  -1145.393443
is_Western      -2229.856326
----------------------
Evaluation Metrics
MAE: 2954.61102162
MSE: 15534536.3132
RMSE: 3941.38761265
R**2: 0.639791254962
```

In [47]:
```
from sklearn.linear_model import LinearRegression
lr2 = LinearRegression()
lr2.fit(X_train, y_train)
predictions2 = lr2.predict(X_train)

import statsmodels.formula.api as sm
model2 = sm.ols(formula= 'sqrt_revenue ~ sqrt_budget + vote_average + \
        is_Comedy + is_Drama + is_Action + is_Thriller +is_Romance +\
        is_Adventure + is_Crime + is_ScienceFict + is_Horror +\
        is_Family + is_Fantasy + is_Mystery + is_Animation + \
        is_History + is_Music + is_War + is_Documentary +is_Western',
        data = mymovie_df)
fitted2 = model2.fit()
fitted2.summary()
```

```
fitted2.summary()
```

Out[47]:

OLS Regression Results

Dep. Variable:	sqrt_revenue	**R-squared:**	0.633
Model:	OLS	**Adj. R-squared:**	0.633
Method:	Least Squares	**F-statistic:**	1208.
Date:	Mon, 06 Aug 2018	**Prob (F-statistic):**	0.00
Time:	10:53:43	**Log-Likelihood:**	-1.3616e+05
No. Observations:	14019	**AIC:**	2.724e+05
Df Residuals:	13998	**BIC:**	2.725e+05
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-9720.2668	624.068	-15.576	0.000	-1.09e+04	-8497.010
sqrt_budget	1.5344	0.011	136.324	0.000	1.512	1.556
vote_average	1750.0792	38.665	45.263	0.000	1674.291	1825.868
is_Comedy	-800.6830	598.971	-1.337	0.181	-1974.747	373.381
is_Drama	-2082.5211	597.757	-3.484	0.000	-3254.205	-910.838
is_Action	-1124.6527	602.463	-1.867	0.062	-2305.561	56.255
is_Thriller	-1432.7550	601.068	-2.384	0.017	-2610.928	-254.582
is_Romance	-1189.6132	605.066	-1.966	0.049	-2375.624	-3.603
is_Adventure	-501.5149	608.176	-0.825	0.410	-1693.621	690.592
is_Crime	-1880.5947	607.969	-3.093	0.002	-3072.294	-688.895
is_ScienceFict	-1054.8824	613.947	-1.718	0.086	-2258.301	148.536
is_Horror	45.2812	613.312	0.074	0.941	-1156.893	1247.455
is_Family	-9.5362	618.735	-0.015	0.988	-1222.339	1203.267
is_Fantasy	-484.7798	619.788	-0.782	0.434	-1699.648	730.088
is_Mystery	-1807.4874	622.200	-2.905	0.004	-3027.082	-587.893
is_Animation	-125.5242	639.332	-0.196	0.844	-1378.700	1127.652
is_History	-3081.3516	648.839	-4.749	0.000	-4353.163	-1809.541
is_Music	-1193.3298	658.982	-1.811	0.070	-2485.022	98.362
is_War	-2482.5049	656.505	-3.781	0.000	-3769.342	-1195.668

is_Documentary	-1518.2456	788.841	-1.925	0.054	-3064.480	27.989
is_Western	-3044.3906	729.222	-4.175	0.000	-4473.763	-1615.018

Omnibus:	2352.594	**Durbin-Watson:**	0.728
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6315.487
Skew:	0.914	**Prob(JB):**	0.00
Kurtosis:	5.734	**Cond. No.**	4.55e+05

In [49]:
```
# dropping is_Horror, is_Family and is_Animation
from sklearn.linear_model import LinearRegression
lr3 = LinearRegression()
lr3.fit(X_train, y_train)
predictions2 = lr3.predict(X_train)

import statsmodels.formula.api as sm
model3 = sm.ols(formula= 'sqrt_revenue ~ sqrt_budget + vote_average + \
        is_Comedy + is_Drama + is_Action + is_Thriller +is_Romance +\
        is_Adventure + is_Crime + is_ScienceFict +\
        is_Fantasy + is_Mystery +  \
        is_History + is_Music + is_War + is_Documentary +is_Western',
        data = mymovie_df)
fitted3 = model3.fit()
fitted3.summary()
```

Out[49]:

OLS Regression Results

Dep. Variable:	sqrt_revenue	**R-squared:**	0.633
Model:	OLS	**Adj. R-squared:**	0.633
Method:	Least Squares	**F-statistic:**	1422.
Date:	Mon, 06 Aug 2018	**Prob (F-statistic):**	0.00
Time:	11:10:08	**Log-Likelihood:**	-1.3616e+05
No. Observations:	14019	**AIC:**	2.724e+05
Df Residuals:	14001	**BIC:**	2.725e+05
Df Model:	17		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-9713.4419	262.275	-37.035	0.000	-1.02e+04	-9199.347
sqrt_budget	1.5334	0.011	138.901	0.000	1.512	1.555
vote_average	1748.0488	38.436	45.479	0.000	1672.708	1823.390

is_Comedy	-790.8622	140.342	-5.635	0.000	-1065.950	-515.774
is_Drama	-2072.3420	132.961	-15.586	0.000	-2332.964	-1811.720
is_Action	-1113.3346	149.553	-7.444	0.000	-1406.479	-820.191
is_Thriller	-1422.4262	147.169	-9.665	0.000	-1710.897	-1133.955
is_Romance	-1179.9312	164.523	-7.172	0.000	-1502.419	-857.444
is_Adventure	-489.0537	167.451	-2.921	0.003	-817.281	-160.827
is_Crime	-1870.1261	172.507	-10.841	0.000	-2208.263	-1531.989
is_ScienceFict	-1043.5018	190.448	-5.479	0.000	-1416.805	-670.199
is_Fantasy	-472.5855	206.418	-2.289	0.022	-877.193	-67.978
is_Mystery	-1796.9923	217.386	-8.266	0.000	-2223.097	-1370.887
is_History	-3069.8661	282.646	-10.861	0.000	-3623.890	-2515.842
is_Music	-1183.3372	307.920	-3.843	0.000	-1786.901	-579.773
is_War	-2471.1454	300.102	-8.234	0.000	-3059.385	-1882.905
is_Documentary	-1509.9611	533.287	-2.831	0.005	-2555.274	-464.648
is_Western	-3033.5869	437.491	-6.934	0.000	-3891.128	-2176.046

Omnibus:	2350.136	**Durbin-Watson:**	0.728
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6309.084
Skew:	0.913	**Prob(JB):**	0.00
Kurtosis:	5.733	**Cond. No.**	9.71e+04

In [ ]: