Data Visualization and Analysis of Worldwide Box Office Revenue

by Varun Kashyap

We apply various graphical techniques using Seaborn to analyze Worldwide Box Office Revenue using a dataset of Box Office returns derived from IMDb. Following are the main goals of the project -

- Analyzing Movie Release Dates
- Preprocessing Features
- Creating Features Based on Release Date
- Using Plotly to Visualize the Number of Films Per Year
- Determining Number of Films and Revenue Per Year
- Determining if Release Days Impact Revenue
- Relationship between Runtime and Revenue

Importing relevant libraries

In [6]:

```
import numpy as np
import pandas as pd
pd.set_option('max_columns', None)
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.style.use('ggplot')
import datetime
import lightgbm as lgb
from scipy import stats
from scipy.sparse import hstack, csr_matrix
from sklearn.model_selection import train_test_split, KFold
from wordcloud import WordCloud
from collections import Counter
from nltk.corpus import stopwords
from nltk.util import ngrams
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.preprocessing import StandardScaler
import nltk
nltk.download('stopwords')
stop = set(stopwords.words('english'))
import os
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import xgboost as xgb
import lightgbm as lgb
from sklearn import model_selection
from sklearn.metrics import accuracy_score
import json
import ast
from urllib.request import urlopen
from PIL import Image
from sklearn.preprocessing import LabelEncoder
import time
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn import linear_model
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\varun\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Loading the Data and Exploration

In [7]:

```
train = pd.read_csv('data/train.csv')
test = pd.read_csv('data/test.csv')
```

In [8]:

```
train.head()
```

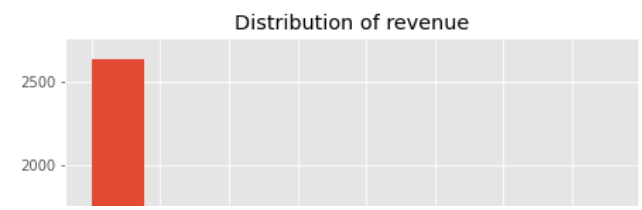
Out[8]:

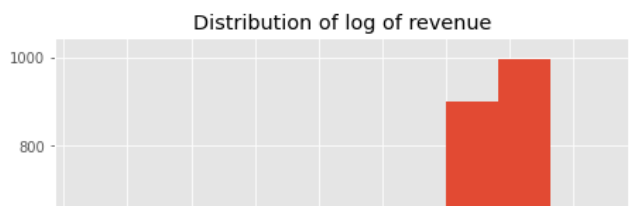
	id	budget	homepage	imdb_id	original_language	original_title	overview	popularity	
0	1	14000000	NaN	tt2637294	en	Hot Tub Time Machine 2	When Lou, who has become the "father of the In...	6.575393	/tQtWuwvMf0hCc2Q
1	2	40000000	NaN	tt0368933	en	The Princess Diaries 2: Royal Engagement	Mia Thermopolis is now a college graduate and ...	8.248895	/w9Z7A0GHEhIp7etpj0
2	3	3300000	http://sonyclassics.com/whiplash/	tt2582802	en	Whiplash	Under the direction of a ruthless instructor, ...	64.299990	/lIv1QinFqz4dlp5U
3	4	1200000	http://kahaanithefilm.com/	tt1821480	hi	Kahaani	Vidya Bagchi (Vidya Balan) arrives in Kolkata ...	3.174936	/aTXRaPrWSinhcmC
4	5	0	NaN	tt1380152	ko	마린보이	Marine Boy is the story of a former national s...	1.148070	/m22s7zvkVFDU9ir5

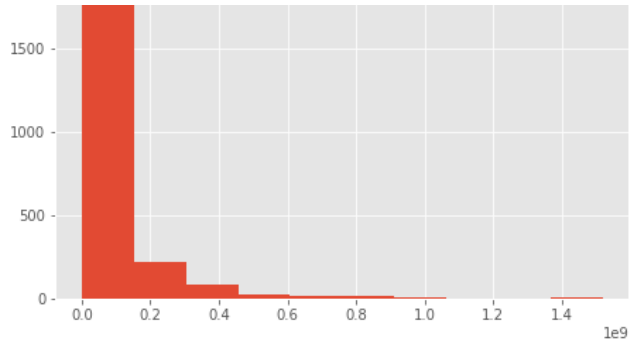
Visualizing the Target Distribution

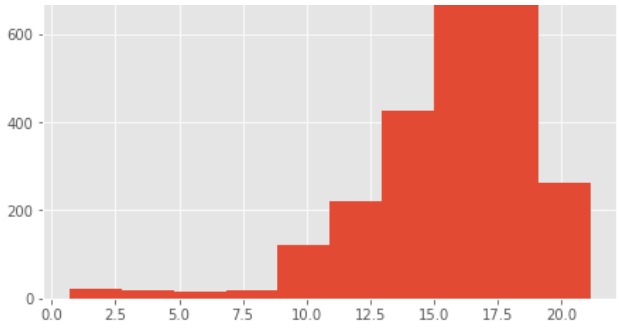
In [9]:

```
fig, ax = plt.subplots(figsize = (16, 6))
plt.subplot(1, 2, 1)
plt.hist(train['revenue']);
plt.title('Distribution of revenue');
plt.subplot(1, 2, 2)
plt.hist(np.log1p(train['revenue']));
plt.title('Distribution of log of revenue');
```









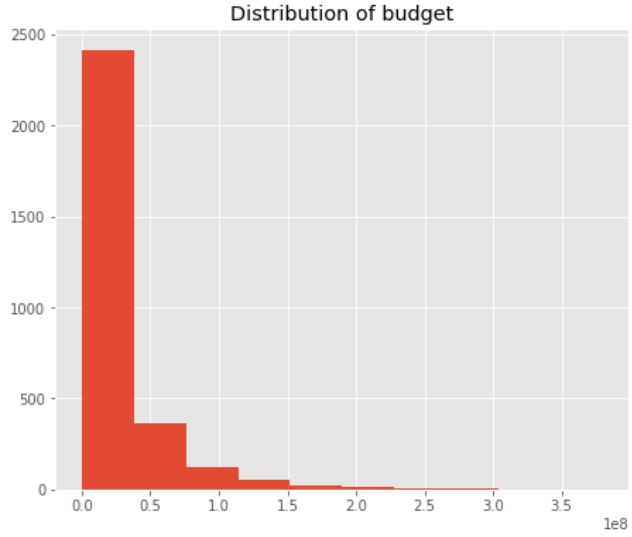
In [10]:

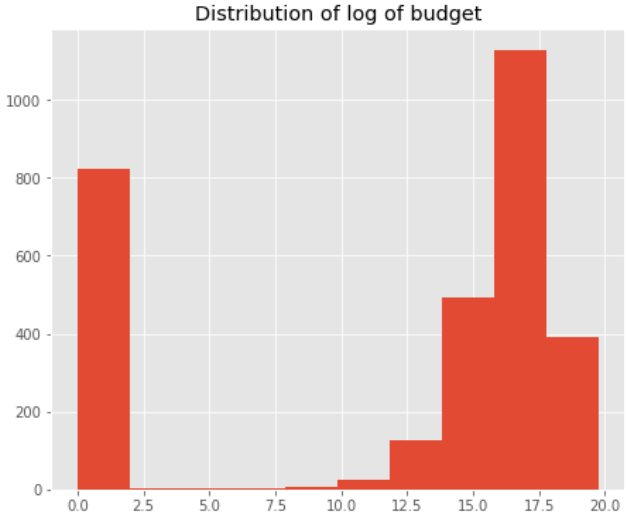
```
train['log_revenue'] = np.log1p(train['revenue'])
```

Relationship between Film Revenue and Budget

In [27]:

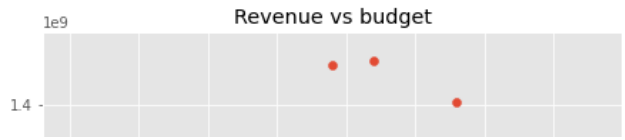
```
fig, ax = plt.subplots(figsize = (16, 6))
plt.subplot(1, 2, 1)
plt.hist(train['budget']);
plt.title('Distribution of budget');
plt.subplot(1, 2, 2)
plt.hist(np.log1p(train['budget']));
plt.title('Distribution of log of budget');
#was skewed hence plotted log distribution
```

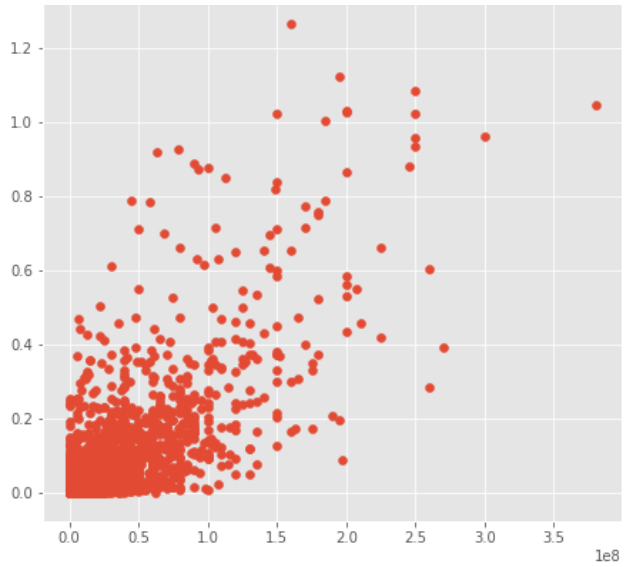


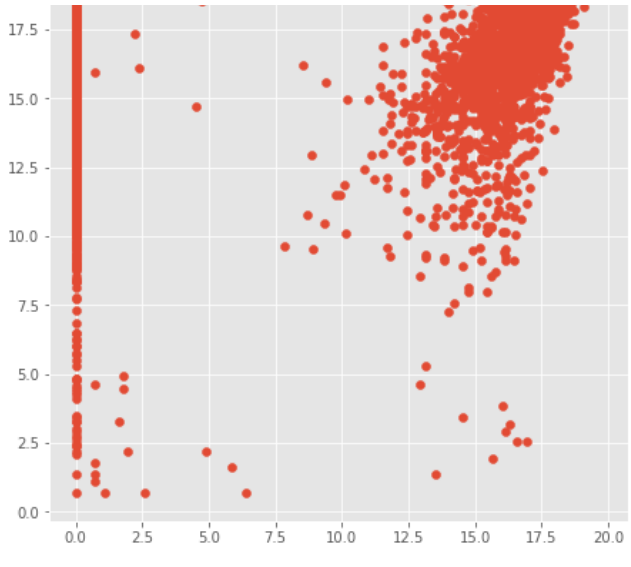


In [28]:

```
plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
plt.scatter(train['budget'], train['revenue'])
plt.title('Revenue vs budget');
plt.subplot(1, 2, 2)
plt.scatter(np.log1p(train['budget']), train['log_revenue'])
plt.title('Log Revenue vs log budget');
#log value to reduced skewed distribution
```







In [13]:

```
train['log_budget'] = np.log1p(train['budget'])
test['log_budget'] = np.log1p(test['budget'])
```

Does having an Official Homepage Affect Revenue?

In [14]:

```
train['homepage'].value_counts().head(10)
```

Out[14]:

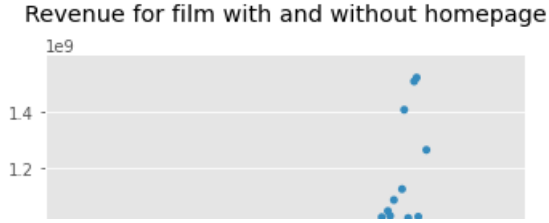
```
http://www.transformersmovie.com/                                        4
http://www.lordoftherings.net/                                           2
http://www.thehobbit.com/                                                2
http://DontThinkTwiceMovie.com                                           1
http://www.sonypictures.com/homevideo/catalog/catalogDetail_DVD043396087576.html    1
http://www.miralmovie.com/                                               1
http://alphaandomega3d.com/index.html                                    1
http://movies.disney.com/million-dollar-arm                              1
http://wwws.warnerbros.de/theprestige/                                   1
http://www.foxmovies.com/movies/joy                                      1
Name: homepage, dtype: int64
```

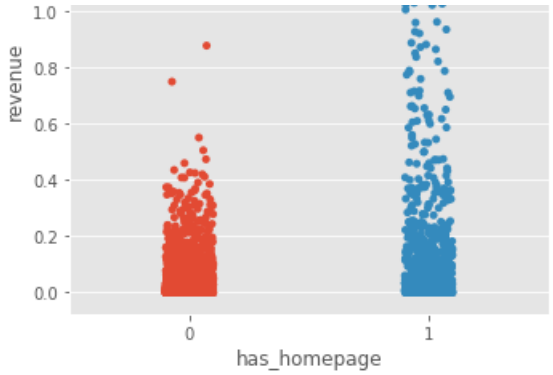
In [15]:

```
train['has_homepage'] = 0
train.loc[train['homepage'].isnull() == False, 'has_homepage'] = 1
test['has_homepage'] = 0
test.loc[test['homepage'].isnull() == False, 'has_homepage'] = 1
```

In [17]:

```
sns.catplot(x='has_homepage', y='revenue', data=train);
plt.title('Revenue for film with and without homepage\n');
```

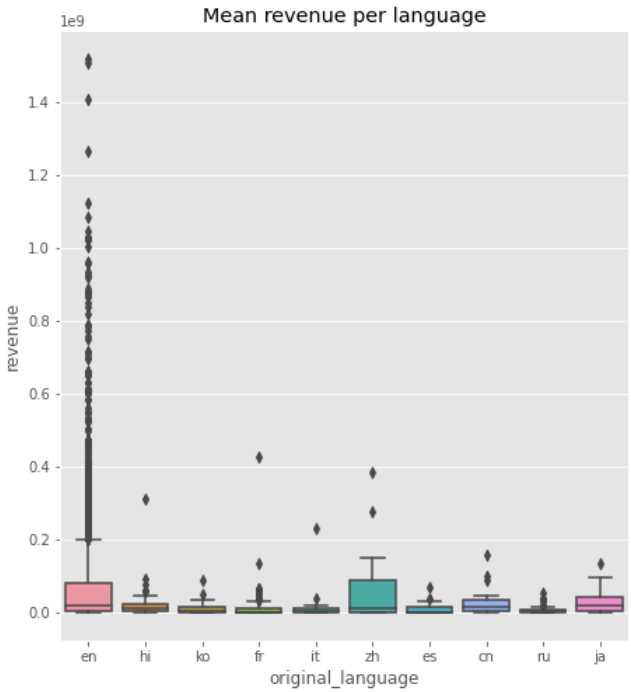


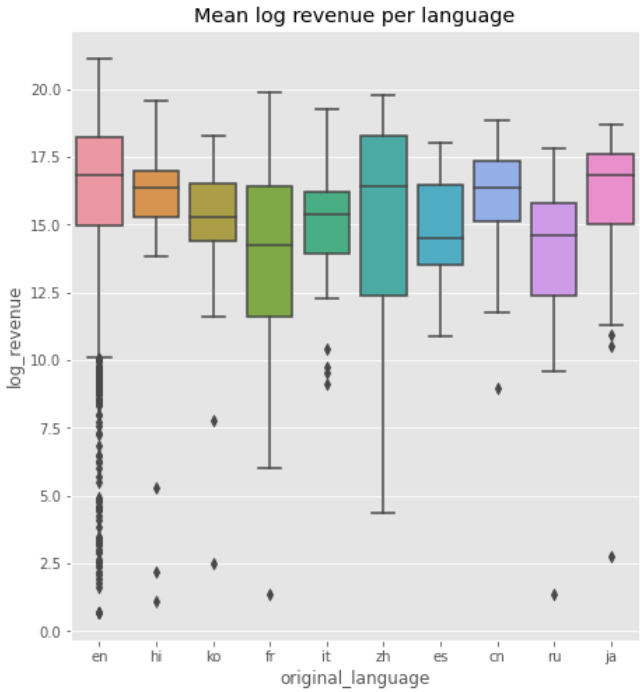


Distribution of Languages in Film

In [18]:

```
plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
sns.boxplot(x='original_language', y='revenue', data=train.loc[train['original_language'].isin(trai
n['original_language'].value_counts().head(10).index)]);
plt.title('Mean revenue per language');
plt.subplot(1, 2, 2)
sns.boxplot(x='original_language', y='log_revenue', data=train.loc[train['original_language'].isin(
train['original_language'].value_counts().head(10).index)]);
plt.title('Mean log revenue per language');
```



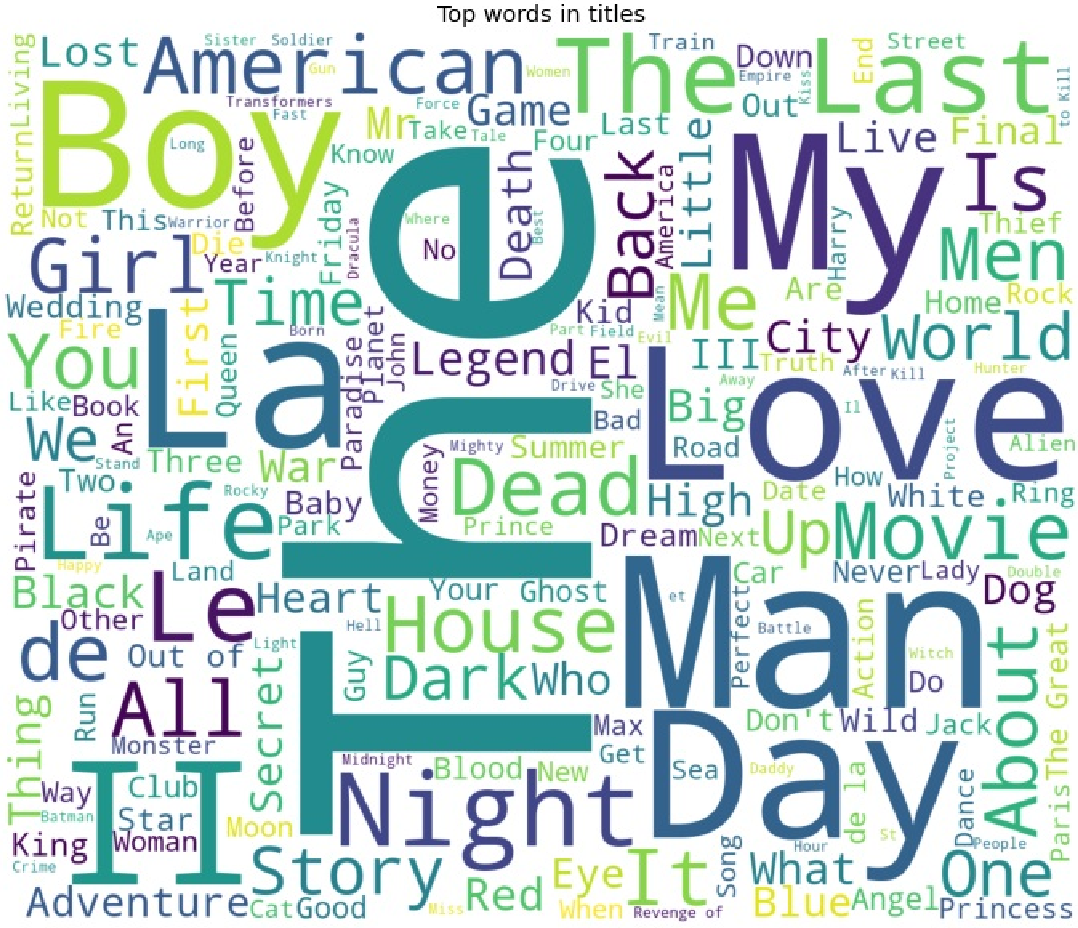


Frequent Words in Film Titles and Descriptions

In [19]:

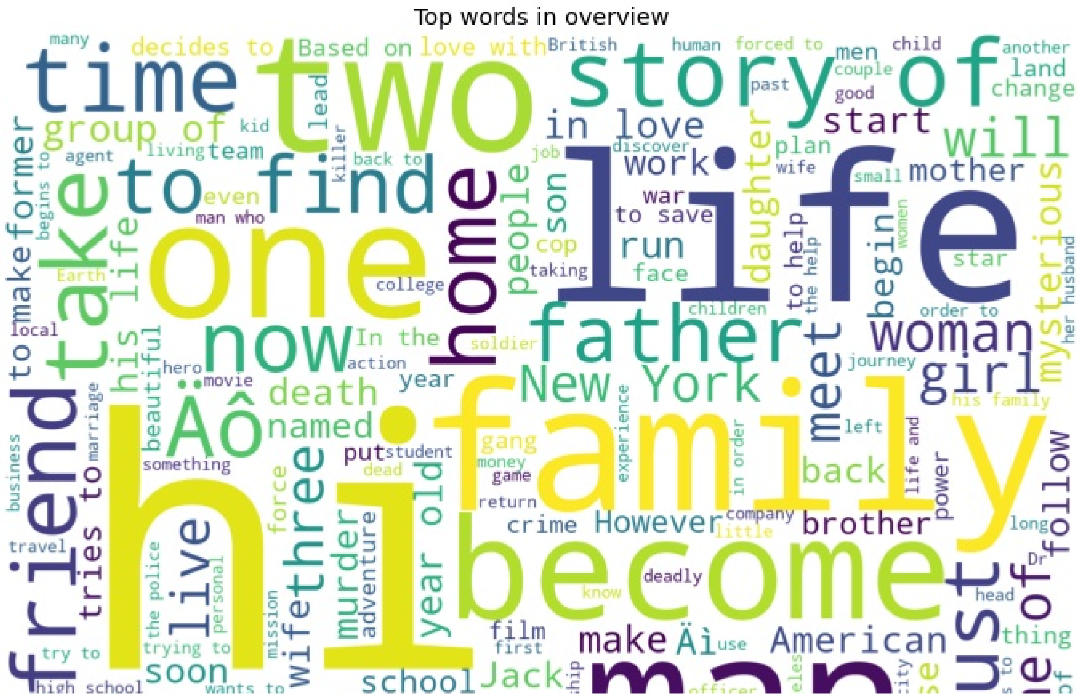
```
plt.figure(figsize = (12, 12))
text = ' '.join(train['original_title'].values)
wordcloud = WordCloud(max_font_size=None, background_color='white', width=1200, height=1000).genera
te(text)
plt.imshow(wordcloud)
plt.title('Top words in titles')
plt.axis("off")
```

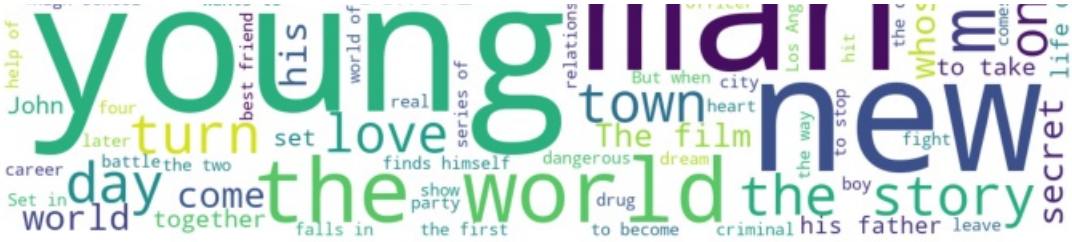
```
plt.axis("off")
plt.show()
```



In [21]:

```
plt.figure(figsize = (12, 12))
text = ' '.join(train['overview'].fillna('').values)
wordcloud = WordCloud(max_font_size=None, background_color='white', width=1200, height=1000).genera
te(text)
plt.imshow(wordcloud)
plt.title('Top words in overview')
plt.axis("off")
plt.show()
```





Do Film Descriptions Impact Revenue?

In [29]:

```
import eli5

vectorizer = TfidfVectorizer(
            sublinear_tf=True,
            analyzer='word',
            token_pattern=r'\w{1,}',
            ngram_range=(1, 2),
            min_df=5)

overview_text = vectorizer.fit_transform(train['overview'].fillna(''))
linreg = LinearRegression()
linreg.fit(overview_text, train['log_revenue'])
eli5.show_weights(linreg, vec=vectorizer, top=150, feature_filter=lambda x: x != '<BIAS>')
```

Out[29]:

y top features

Weight?	Feature
+13.074	to
+10.131	bombing
+9.981	the
+9.777	complications
+9.243	äì
+9.191	and get
+8.850	at it
+8.765	california
+8.749	that this
+8.669	others
+8.616	miami
+8.525	town s
+8.473	he attempts
+8.471	inspires
+8.458	shooting
+8.342	a
+8.314	and are
+8.308	and on
+8.286	never
+8.266	danny
+8.209	roll
+8.093	the weekend
+8.076	äì the
+8.042	agent
+7.959	diego
+7.923	still
+7.890	the chaos
+7.863	prisoner
+7.785	content
+7.763	seeks
+7.698	mia
+7.690	300
+7.682	inadvertently
+7.633	the movie
+7.552	paid
+7.540	siblings
+7.540	date
+7.499	marine
+7.496	wallace
+7.494	men who
+7.490	fbi
+7.325	previous
+7.317	continue
+7.313	trials
+7.296	war and
+7.264	can
+7.243	owners
+7.228	different

Weight	Feature
+7.147	the british
+7.214	moved
+7.152	000
+7.083	what it
+7.062	colorado
+7.061	stage
+7.057	a story
+7.032	centers
+7.027	might
+7.016	he finds
+6.987	spend the
+6.972	his teenage
+6.939	help a
… 3801 more positive …	
… 3242 more negative …	
-6.911	knew
-6.927	unexpected
-6.937	so she
-6.954	the local
-6.956	the victim
-6.963	bosses
-6.967	carefree
-6.968	present
-6.977	help her
-6.986	her friend
-6.993	the battle
-7.084	swamp
-7.140	she falls
-7.151	while in
-7.153	got
-7.157	love of
-7.162	in and
-7.167	for some
-7.202	on a
-7.221	the end
-7.239	causes
-7.243	driving
-7.247	they will
-7.252	lives in
-7.270	their father
-7.270	cinema
-7.272	small time
-7.276	model
-7.289	where she
-7.308	terry
-7.461	retired
-7.489	after death
-7.525	selling
-7.544	waits
-7.545	your
-7.622	odds with
-7.637	explores
-7.702	war in
-7.744	the pair
-7.755	and it
-7.805	you
-7.841	is no
-7.864	orphans
-7.883	than they
-7.885	sword
-7.969	so that
-7.970	overwhelmed
-8.010	provincial
-8.052	his long
-8.088	tradition
-8.104	installment
-8.156	factory
-8.232	cowboy
-8.292	through an
-8.304	end of
-8.314	attracted
-8.319	life is
-8.326	towards
-8.334	is also
-8.442	at school
-8.464	off with
-8.519	invited
-8.544	woman who
-8.589	critic
-8.700	the mother
-8.805	family with
-8.807	him on
-8.865	changed
-8.899	explore
-8.916	she will
-8.936	it was
-9.048	indian
-9.135	status
-9.281	politicians
-9.391	18
-9.481	violence
-9.628	escape and
-9.716	life they
-10.021	ones

Weight	Feature
-10.291	attracted to
-10.321	who also
-10.421	casino
-10.614	receiving
-10.759	kept
-12.139	and be
-12.939	campaign
-13.858	mike
-15.273	woman from

In [25]:

```
print('Target value:', train['log_revenue'][1000])
eli5.show_prediction(linreg, doc=train['overview'].values[1000], vec=vectorizer)
```

```
Target value: 16.44583954907521
```

Out[25]:

y (score **16.446**) top features

Contribution?	Feature
+12.762	<BIAS>
+3.684	Highlighted in text (sum)

when elizabeth returns to her mother's home after her marriage breaks up, she recreates her imaginary childhood friend, fred, to escape from the trauma of losing her husband and her job. in between the chaos and mayhem that fred creates, elizabeth attempts to win back her husband and return to normality.

Analyzing Movie Release Dates

In [26]:

```
test.loc[test['release_date'].isnull() == False, 'release_date'].head()
```

Out[26]:

```
0    7/14/07
1    5/19/58
2    5/23/97
3     9/4/10
4    2/11/05
Name: release_date, dtype: object
```

Preprocessing Features

In [30]:

```
def fix_date(x):
    year = x.split('/')[2]
    if int(year) <= 19:
        return x[:-2] + '20' + year
    else:
        return x[:-2] + '19' + year
```

In [31]:

```
test.loc[test['release_date'].isnull() == True].head()
```

Out[31]:

	id	budget	homepage	imdb_id	original_language	original_title	overview	popularity	poster_path	release_date	runtime	sta
828	3829	0	NaN	tt0210130	en	Jails, Hospitals & Hip-Hop	Jails, Hospitals & Hip-Hop is a cinematic ...	0.009057	NaN	NaN	90.0	N

In [32]:

```
test.loc[test['release_date'].isnull() == True, 'release_date'] = '05/01/00'
```

In [33]:

```
train['release_date'] = train['release_date'].apply(lambda x: fix_date(x))
test['release_date'] = test['release_date'].apply(lambda x: fix_date(x))
```

Creating Features Based on Release Date

In [34]:

```
train['release_date'] = pd.to_datetime(train['release_date'])
test['release_date'] = pd.to_datetime(test['release_date'])
```

In [35]:

```
def process_date(df):
    date_parts = ["year", "weekday", "month", 'weekofyear', 'day', 'quarter']
    for part in date_parts:
        part_col = 'release_date' + "_" + part
        df[part_col] = getattr(df['release_date'].dt, part).astype(int)

    return df

train = process_date(train)
test = process_date(test)
```

Using Plotly to Visualize the Number of Films Per Year

In [36]:

```
# Count no.of films released per year and sort the years in ascending order
# Do this for both Train and Test Sets
d1 = train['release_date_year'].value_counts().sort_index()
d2 = test['release_date_year'].value_counts().sort_index()

import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go

# x values are years, and y values are movie counts, name=legend
data = [go.Scatter(x=d1.index, y=d1.values, name='train'),
        go.Scatter(x=d2.index, y=d2.values, name='test')]

layout = go.Layout(dict(title = "Number of films per year",
                  xaxis = dict(title = 'Year'),
                  yaxis = dict(title = 'Count'),
                  ),legend=dict(
```

```
                orientation="v"))
py.iplot(dict(data=data, layout=layout))
```

Determining Number of Films and Revenue Per Year

In [37]:

```
d1 = train['release_date_year'].value_counts().sort_index()
d2 = train.groupby(['release_date_year'])['revenue'].sum()

data = [go.Scatter(x=d1.index, y=d1.values, name='film count'),
        go.Scatter(x=d2.index, y=d2.values, name='total revenue', yaxis='y2')]

layout = go.Layout(dict(title = "Number of films and total revenue per year",
                        xaxis = dict(title = 'Year'),
                        yaxis = dict(title = 'Count'),
                        yaxis2=dict(title='Total revenue', overlaying='y', side='right')),
                   legend=dict(orientation="v"))

py.iplot(dict(data=data, layout=layout))
```

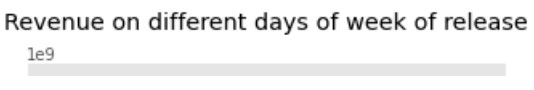
In [38]:

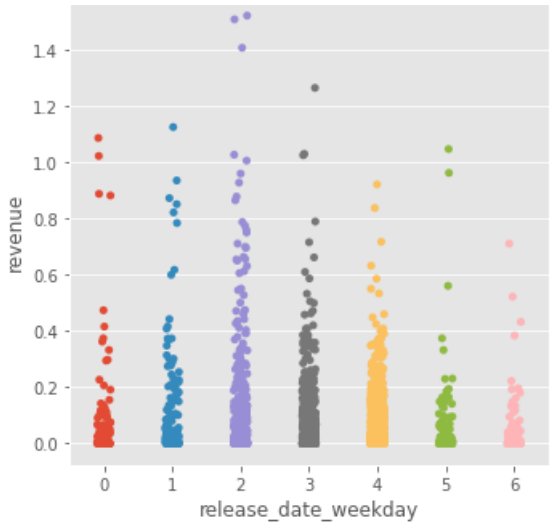
```
d1 = train['release_date_year'].value_counts().sort_index()
d2 = train.groupby(['release_date_year'])['revenue'].mean()

data = [go.Scatter(x=d1.index, y=d1.values, name='film count'),
        go.Scatter(x=d2.index, y=d2.values, name='mean revenue', yaxis='y2')]

layout = go.Layout(dict(title = "Number of films and average revenue per year",
                  xaxis = dict(title = 'Year'),
                  yaxis = dict(title = 'Count'),
                  yaxis2=dict(title='Average revenue', overlaying='y', side='right')
                  ),legend=dict(
                orientation="v"))
py.iplot(dict(data=data, layout=layout))
```

Do Release Days Impact Revenue?

In [43]:

```
sns.catplot(x='release_date_weekday', y='revenue', data=train);
plt.title('Revenue on different days of week of release\n');
```

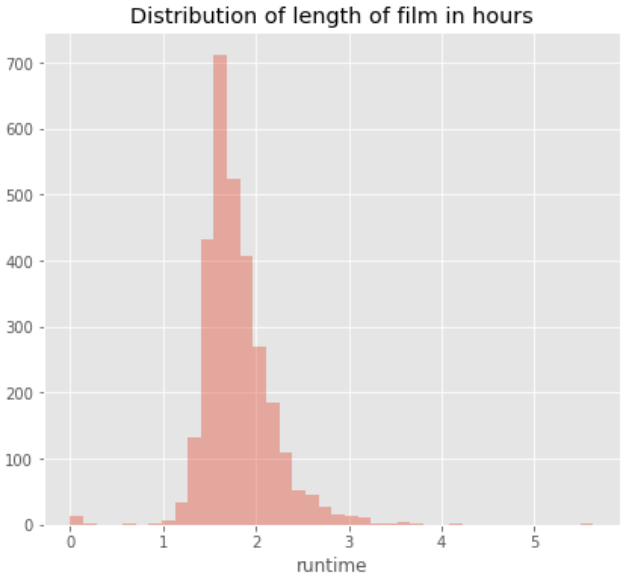


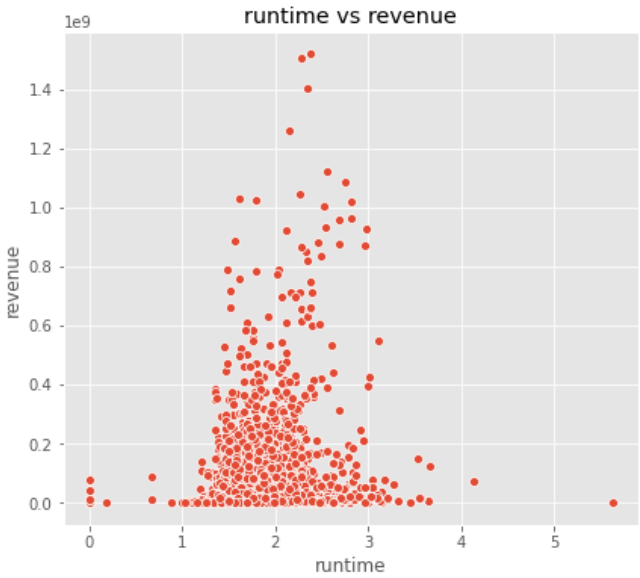


Relationship between Runtime and Revenue

In [40]:

```
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
sns.distplot(train['runtime'].fillna(0) / 60, bins=40, kde=False);
plt.title('Distribution of length of film in hours');
plt.subplot(1, 2, 2)
sns.scatterplot(train['runtime'].fillna(0)/60, train['revenue'])
plt.title('runtime vs revenue');
```

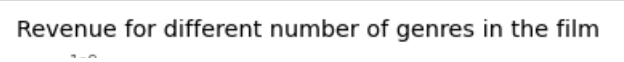


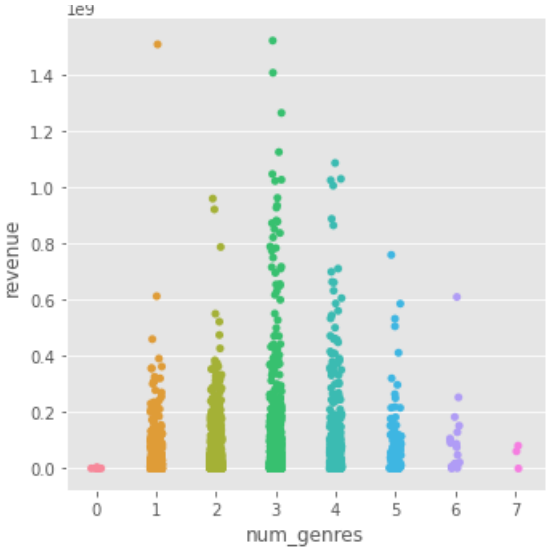


Determining Highest Grossing Genres

In [42]:

```
sns.catplot(x='num_genres', y='revenue', data=train);
plt.title('Revenue for different number of genres in the film\n');
```




1e9
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
revenue
0
1
2
3
4
5
6
7
num_genres