In [1]:
```
from pykalman import KalmanFilter
import numpy as np
import pandas as pd
import sys
import matplotlib
import matplotlib.pyplot as plt
from skimage.color import lab2rgb
from sklearn import model_selection
from sklearn.naive_bayes import GaussianNB
import skimage
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import FunctionTransformer, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from functools import reduce
import statsmodels.api as sm
lowess = sm.nonparametric.lowess
from scipy import stats
```

In [2]:
```
def to_timestamp(dateTime):
    return dateTime.timestamp()

def map_genre(row):
   result = []
   for genre_code in row:
       matches = genres[genres['wikidata_id'] == genre_code]['genre_label'].va
lues
       for match in matches:
           result.append(match)
   return result
```

In [3]:
```
wikidata = pd.read_json('movies/data/wikidata-movies.json.gz', orient='record'
, lines=True, encoding="utf8", convert_dates=['publication_date'])
genres = pd.read_json('movies/data/genres.json.gz', orient='record', lines=Tru
e, encoding="utf8")
```

In [4]:
```
wikidata = wikidata[wikidata['made_profit'].notnull()].reset_index(drop=True)
wikidata['publication_timestamp'] = wikidata['publication_date'].apply(to_time
stamp)
wikidata['genre_names'] = wikidata['genre'].apply(map_genre)
```

In [5]:
```
rotten_tomatoes = pd.read_json('movies/data/rotten-tomatoes.json.gz', orient=
'record', lines=True)
omdb = pd.read_json('movies/data/omdb-data.json.gz', orient='record', lines=Tr
ue)
combined = wikidata.join(rotten_tomatoes.set_index('rotten_tomatoes_id'), on=
'rotten_tomatoes_id', rsuffix='_rt')
combined = combined.join(omdb.set_index('imdb_id'), on='imdb_id')
```

In [6]:
```
combined
```

Out[6]:

	based_on	cast_member	country_of_origin	director	enwiki_title	filming_loca
0	NaN	[Q5126010, Q3390414, Q5676024, Q237021]	Q29	[Q51892574]	Orbiter 9	NaN
1	NaN	NaN	Q30	[Q3384479, Q351884]	Despicable Me	NaN
2	NaN	[Q386349, Q1605965, Q3805579, Q271162, Q463226...	Q30	[Q2071]	Eraserhead	[Q99]
3	Q17017426	[Q117500, Q1376880, Q11930, Q311169, Q951634, ...	Q30	[Q11930]	Dances with Wolves	[Q1558]
4	NaN	[Q38111, Q211553, Q177311, Q8927, Q173399, Q20...	Q145	[Q25191]	Inception	[Q99, Q3870 Q17, Q90, Q1951, Q7275217, Q126...
5	NaN	[Q229313, Q445772, Q727988, Q3163137, Q1372392...	Q16	[Q6385039]	Mama (2013 film)	[Q172, Q133 Q13939]
6	Q243556	[Q34012, Q41163, Q95043, Q464714, Q171736, Q32...	Q30	[Q56094]	The Godfather	[Q18438, Q6 Q1408, Q14
7	NaN	[Q483118, Q23547, Q108283, Q215072, Q270664, Q...	Q30	[Q483118]	Argo (2012 film)	[Q406, Q65, Q43]

	based_on	cast_member	country_of_origin	director	enwiki_title	filming_loca
8	Q7857661	[Q317343, Q57147, Q244674, Q343616, Q208649, Q...	Q145	[Q706475]	12 Years a Slave (film)	[Q34404]
9	NaN	[Q295803, Q200534, Q228865, Q200405, Q314133, ...	Q145	[Q191755]	Only Lovers Left Alive	[Q183, Q365
10	NaN	[Q2376200, Q2299195]	Q159	[Q28664905]	The PyraMMMid	[Q2280]
11	Q40354	[Q189490, Q217004, Q32045, Q201279, Q219373, Q...	Q30	[Q561387]	The Hunger Games: Mockingjay – Part 1	[Q23556, Q6 Q90, Q1428
12	NaN	[Q4271506, Q4333656]	Q159	[Q4215049]	Inadequate People	NaN
13	NaN	[Q4495971, Q1074254, Q4079472, Q18008969, Q410...	Q159	[Q4491501]	The Best Movie 2	NaN
14	NaN	[Q3479732, Q1074254, Q4273944, Q777625]	Q159	NaN	Our Russia. The Balls of Fate	NaN
15	Q2545790	[Q503706, Q44158, Q190162, Q525065, Q355168, Q...	Q30	[Q717015]	Guardians of the Galaxy (film)	[Q84, Q2278 Q6673670, Q44057]
16	NaN	NaN	Q30	[Q357627, Q7366035]	Inside Out (2015 film)	NaN

	based_on	cast_member	country_of_origin	director	enwiki_title	filming_loca
17	Q8065468	NaN	Q30	[Q2549739, Q926614, Q280187, Q3525855, Q138537...	Pinocchio (1940 film)	NaN
18	NaN	[Q264840, Q351290, Q528126, Q233502, Q94913, Q...	Q30	[Q374286]	The Conjuring	[Q1454]
19	NaN	[Q193517, Q313039, Q29809869, Q38410137]	Q30	[Q313039]	A Quiet Place (film)	NaN
20	Q83279	[Q299282, Q591238, Q4337, Q4488, Q454102, Q415...	Q30	[Q3078869]	The SpongeBob Movie: Sponge Out of Water	[Q1428]
21	NaN	[Q442830, Q3479732, Q1966992, Q282818]	Q159	[Q4222061, Q4077720, Q2833792, Q4130936]	Yolki 2	NaN
22	NaN	[Q299317, Q1336685, Q3308078, Q360477, Q313545...	Q30	[Q346508]	Butch and Sundance: The Early Days	NaN
23	Q2944381	[Q873, Q40523, Q165518, Q201418, Q294583, Q273...	Q30	[Q2465518]	August: Osage County (film)	NaN
24	Q140527	[Q544641, Q183535, Q3553607, Q2119044, Q415100...	Q159	[Q4534523]	The Return of the Musketeers, or The Treasures...	NaN

	based_on	cast_member	country_of_origin	director	enwiki_title	filming_loca
25	Q17014869	[Q162492, Q37459, Q542571, Q208667, Q705522, Q...	Q408	[Q16730387]	The Railway Man (film)	[Q408, Q869 Q23436]
26	NaN	[Q298777, Q4315866, Q286690]	Q159	[Q13630494]	Moscow Heat	NaN
27	NaN	[Q4077949, Q4494681, Q4254527, Q4157470]	Q159	[Q4171916]	What Men Talk About	NaN
28	NaN	[Q175535, Q23844, Q80966, Q29250, Q215072, Q18...	Q183	[Q23844]	The Monuments Men	[Q64, Q183]
29	NaN	[Q456047, Q816434, Q207179, Q242504, Q458188, ...	Q145	[Q355300]	About Time (2013 film)	NaN
...	...	...	...	...	...	...
761	NaN	[Q16239385, Q312712, Q18379490, Q162492, Q8040...	Q145	[Q2593]	Kingsman: The Golden Circle	NaN
762	NaN	[Q54314, Q41422, Q178348, Q41396, Q295803, Q29...	Q30	[Q18018415, Q20675767]	Avengers: Infinity War	NaN
763	NaN	NaN	Q30	[Q913976]	The Emoji Movie	NaN

	based_on	cast_member	country_of_origin	director	enwiki_title	filming_loca
764	Q1649084	[Q129591, Q16296, Q27560621, Q611096, Q23814, ...	Q30	[Q433893]	Logan (2017 film)	[Q1522, Q15 Q1494]
765	NaN	NaN	Q30	[Q1077862, Q18921842]	Big Hero 6 (film)	NaN
766	NaN	[Q213864, Q10738, Q311232, Q169963, Q298551, Q...	Q30	[Q374286]	Furious 7	[Q65, Q2355 Q1261, Q15
767	NaN	[Q321131, Q4678990, Q235519, Q26231, Q462354, ...	Q30	[Q323076]	This Is Where I Leave You	[Q60]
768	Q214016	[Q444146, Q508404, Q212064, Q173637, Q313388, ...	Q30	[Q13638984, Q3378803]	22 Jump Street	[Q34404]
769	Q632908	[Q155775, Q342617, Q233563, Q19960315, Q156394...	Q145	[Q7151786]	Paddington (film)	[Q84]
770	Q815739	[Q201198, Q317761, Q311804, Q15725509, Q345362...	Q30	[Q552731]	Warcraft (film)	[Q24639]
771	NaN	[Q294586, Q164782, Q428819, Q44380, Q439438, Q...	Q30	[Q2576503]	Annie (2014 film)	[Q60]

	based_on	cast_member	country_of_origin	director	enwiki_title	filming_loca
772	Q4720469	[Q208117, Q310295, Q1239933, Q2469915, Q230278...	Q30	[Q3312946]	Alexander and the Terrible, Horrible, No Good,...	[Q65]
773	NaN	[Q512353, Q129591, Q245075, Q102124, Q350014, ...	Q30	[Q715838]	Chappie (film)	[Q1812844, Q34647]
774	NaN	[Q2832626, Q2329850, Q4360641, Q397682, Q48715...	Q159	[Q4102539]	Love in Vegas	NaN
775	NaN	NaN	Q30	[Q2630467, Q7519046]	Penguins of Madagascar	NaN
776	NaN	[Q192682, Q173158, Q317337, Q259760, Q267613, ...	Q30	[Q383768]	Self/less	[Q34404, Q6
777	Q1517252	[Q402764, Q41396, Q459384, Q272972, Q229535, Q...	Q30	[Q167522]	Everest (2015 film)	[Q3037, Q1043277, Q837, Q38, Q145, Q661
778	NaN	[Q43416, Q262278, Q552026, Q232343, Q240187, Q...	Q30	[Q5236475]	John Wick (film)	[Q60]
779	NaN	[Q526620, Q258220, Q472482, Q673007]	Q30	[Q361290]	Annabelle (film)	[Q65]
780	Q857000	NaN	Q30	[Q1181049]	How to Train Your Dragon 2	NaN

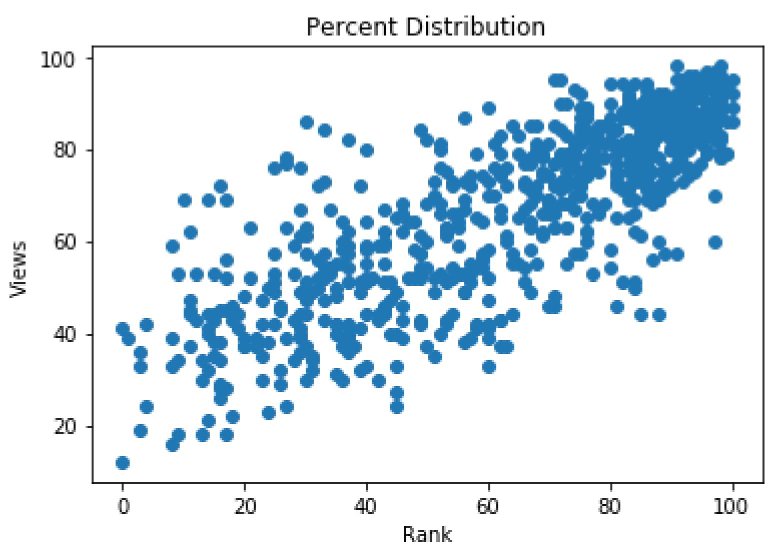
	based_on	cast_member	country_of_origin	director	enwiki_title	filming_loca
781	NaN	[Q45772, Q481832, Q205707, Q189490, Q36949, Q2...	Q30	[Q314342]	American Hustle	[Q60]
782	NaN	[Q3015088, Q563895, Q4196443, Q726105, Q239012...	Q159	[Q3341663]	Legend № 17	[Q863096, Q207294]
783	Q1066948	[Q34436, Q117906, Q352540, Q26372, Q106275, Q4...	Q30	[Q175062]	Ghost in the Shell (2017 film)	[Q23661]
784	NaN	[Q204299, Q201279, Q316446, Q459384, Q310937, ...	Q30	[Q372394]	Three Billboards Outside Ebbing, Missouri	[Q2043861]
785	NaN	[Q188955, Q36301, Q229313, Q123351, Q603317, Q...	Q30	[Q25191]	Interstellar (film)	[Q189, Q65, Q1951]
786	NaN	[Q193048, Q798656, Q449822, Q15069963, Q207406...	Q30	[Q2578679]	The Signal (2014 film)	[Q1522]
787	NaN	[Q192682, Q133313, Q4947838, Q705522, Q4791200...	Q30	[Q977624]	Life (2017 film)	NaN
788	NaN	[Q233563, Q41449, Q229254, Q461309, Q313043, Q...	Q30	[Q219124]	The Shape of Water	[Q133116]

	based_on	cast_member	country_of_origin	director	enwiki_title	filming_loca
789	NaN	[Q2090275, Q4147975, Q20510404, Q4159892, Q135...	Q159	[Q4065391]	Earthquake (2016 film)	NaN
790	NaN	NaN	Q30	[Q1357018]	I Am Not Your Negro	NaN

791 rows × 28 columns

Is there a difference between the positivity of critics and the audience?

In [7]:
```
plt.title('Percent Distribution')
plt.xlabel('Rank')
plt.ylabel('Views')
plt.scatter(combined['critic_percent'], combined['audience_percent'])
plt.show()
```



In [8]:
```
test3 = combined[combined['audience_percent'].notnull() & combined['critic_percent'].notnull()]
print(stats.normaltest(test3['audience_percent']).pvalue) #<0.05, therefore not normal
print(stats.mannwhitneyu(test3['critic_percent'], test3['audience_percent']).pvalue) #>0.05, therefore equal
print(test3['audience_percent'].mean())
print(test3['critic_percent'].mean())
```

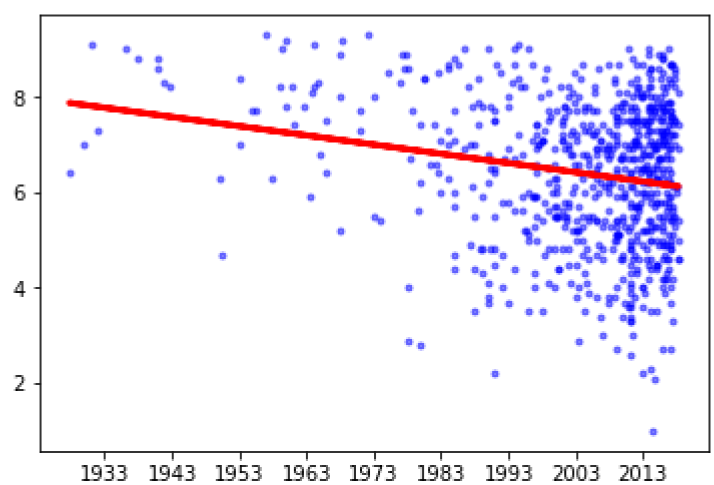
```
3.608996083927233e-17
0.4397286644629225
68.09782608695652
65.06657608695652
```

Have average ratings changed over time?

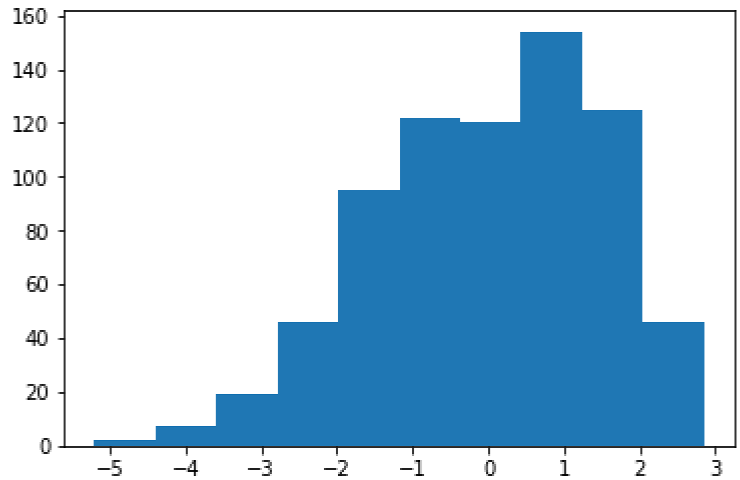
In [9]:
```
critic_average_test = combined[['publication_date','publication_timestamp','critic_average']].dropna()
fit = stats.linregress(critic_average_test['publication_timestamp'], critic_average_test['critic_average'])
critic_average_test['prediction'] = critic_average_test['publication_timestamp']*fit.slope + fit.intercept
print(fit.pvalue) #p < 0.05, therefore we can conclude that critic ratings are decreasing.
print(fit.rvalue) #correlation coefficient is low, so it is not very correlated
```

```
6.156831173958292e-08
-0.19792833987738834
```

In [10]:
```
plt.plot(critic_average_test['publication_date'], critic_average_test['critic_average'], 'b.', alpha=0.5)
plt.plot(critic_average_test['publication_date'], critic_average_test['prediction'], 'r-', linewidth=3)
plt.show()
```



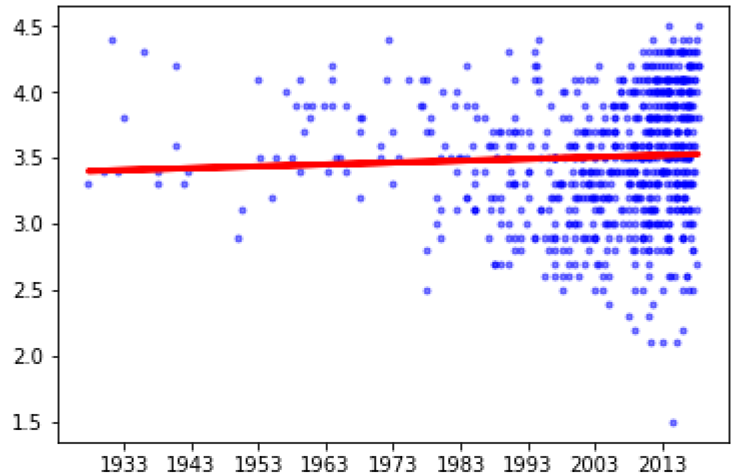
In [11]:
```
plt.hist(np.subtract(critic_average_test['critic_average'],critic_average_test
['prediction']))
plt.show()
#By the central limit theorem, this is close enough to being normal.
#We expect a greater decline on the high end because the average critic rating
 is higher than the middle rating, 5.
```



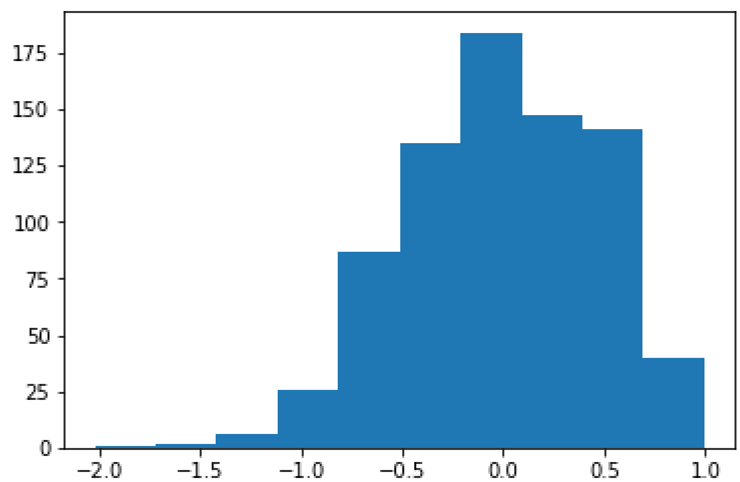
In [12]:
```
audience_average_test = combined[['publication_date','publication_timestamp',
'audience_average']].dropna()
fit = stats.linregress(audience_average_test['publication_timestamp'], audienc
e_average_test['audience_average'])
audience_average_test['prediction'] = audience_average_test['publication_times
tamp']*fit.slope + fit.intercept
print(fit.pvalue) #p > 0.05, therefore we cannot conclude that the audience ra
tings are changing.
print(fit.rvalue) #correlation coefficient is low, so it is not very correlate
d
```

```
0.20019655801512026
0.046244255512890665
```

In [13]:
```
plt.plot(audience_average_test['publication_date'], audience_average_test['aud
ience_average'], 'b.', alpha=0.5)
plt.plot(audience_average_test['publication_date'], audience_average_test['pre
diction'], 'r-', linewidth=3)
plt.show()
```



In [14]:
```
plt.hist(np.subtract(audience_average_test['audience_average'],audience_averag
e_test['prediction']))
plt.show()
#By the central limit theorem, this is close enough to being normal.
#We expect a greater decline on the high end because the average audience rati
ng is higher than the middle rating, 2.5.
```

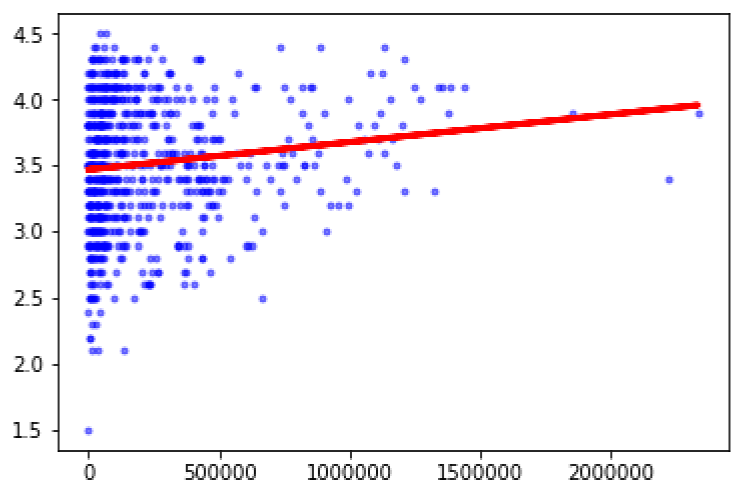


Do average audience ratings change based on its popularity?

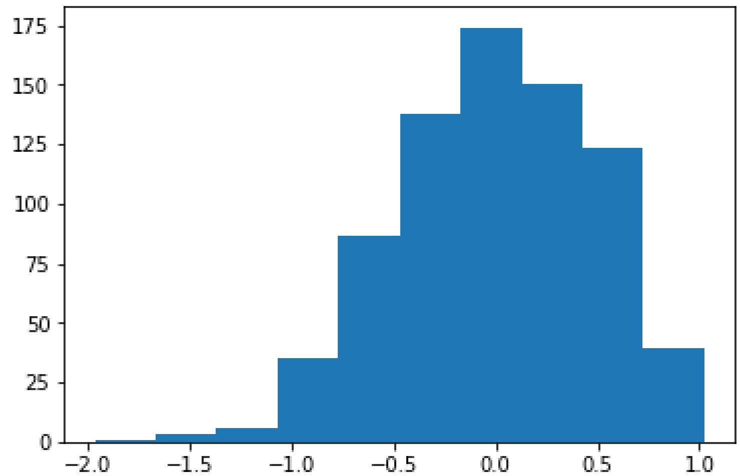
In [15]:
```
audience_ratings_test = combined[['publication_date','publication_timestamp',
'audience_average','audience_ratings']].dropna()
#Removing movies with n >= 10000000 ratings as they seem like outliers
audience_ratings_test = audience_ratings_test[audience_ratings_test['audience_
ratings'] < 10000000]
fit = stats.linregress(audience_ratings_test['audience_ratings'], audience_rat
ings_test['audience_average'])
audience_ratings_test['prediction'] = audience_ratings_test['audience_ratings'
]*fit.slope + fit.intercept
print(fit.pvalue) #p < 0.05, therefore we can conclude that higher averages co
rrelate with more popular movies.
print(fit.rvalue) #correlation coefficient is low, so it is not very correlate
d
```

```
0.0003085653249741354
0.1309596810010819
```

In [16]:
```
plt.plot(audience_ratings_test['audience_ratings'], audience_ratings_test['aud
ience_average'], 'b.', alpha=0.5)
plt.plot(audience_ratings_test['audience_ratings'], audience_ratings_test['pre
diction'], 'r-', linewidth=3)
plt.show()
```



In [17]:
```
plt.hist(np.subtract(audience_ratings_test['audience_average'],audience_rating
s_test['prediction']))
plt.show()
#By the central limit theorem, this is close enough to being normal.
#We expect a greater decline on the high end because the average audience rati
ng is higher than the middle rating, 2.5.
```



Does genre have an effect on profitability?

In [18]:
```
def genre_profit_agg(combined_row):
    for genre_id in combined_row['genre']:
        genre_test.loc[genre_test['wikidata_id'] == genre_id,'total']+=1
        if (combined_row['made_profit'] == 1.0):
            genre_test.loc[genre_test['wikidata_id'] == genre_id,'profit']+=1
```

In [19]:
```
genre_test = genres
genre_test['profit'] = 0
genre_test['total'] = 0
combined.apply(genre_profit_agg, axis=1)
genre_test = genre_test[genre_test['total'] > 0]
```

In [20]:
```
genre_test['loss'] = genre_test['total'] - genre_test['profit']
genre_test = genre_test[genre_test['profit'] >= 5]
genre_test = genre_test[genre_test['loss'] >= 5]
contingency = genre_test[['profit','loss']]
#contingency = contingency[contingency['profit'] >= 5]
#contingency = contingency[contingency['loss'] >= 5]
chi2, p, dof, expected = stats.chi2_contingency(contingency)
print(p) # p < 0.05, therefore genre effects profitability
```

```
0.01956332775267009
```

```
/opt/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWi
thCopyWarning:
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: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

In [21]:
```
genre_test['avg'] = genre_test['profit']/genre_test['total']
genre_test.sort_values(by='avg',ascending=False).reset_index(drop=True)
#Science fiction films are the most profitable.
```

Out[21]:

	genre_label	wikidata_id	profit	total	loss	avg
0	science fiction film	Q471839	130	142	12	0.915493
1	romantic comedy	Q860626	53	59	6	0.898305
2	romance film	Q1054574	44	49	5	0.897959
3	horror film	Q200092	70	78	8	0.897436
4	war film	Q369747	40	45	5	0.888889
5	thriller film	Q2484376	108	122	14	0.885246
6	film based on literature	Q52162262	119	135	16	0.881481
7	adventure film	Q319221	104	119	15	0.873950
8	fantasy film	Q157394	109	125	16	0.872000
9	comedy film	Q157443	162	189	27	0.857143
10	musical film	Q842256	54	63	9	0.857143
11	crime film	Q959790	60	71	11	0.845070
12	drama film	Q130232	226	272	46	0.830882
13	action film	Q188473	198	239	41	0.828452
14	dystopian film	Q20443008	28	34	6	0.823529
15	teen film	Q1146335	27	33	6	0.818182
16	biographical film	Q645928	41	52	11	0.788462
17	children's film	Q2143665	29	37	8	0.783784
18	family film	Q1361932	20	26	6	0.769231
19	heist film	Q496523	11	16	5	0.687500
20	drama	Q21010853	19	28	9	0.678571
21	Western	Q172980	6	11	5	0.545455

Does country of origin have an effect on profitability?

In [22]:
```
countries = pd.DataFrame(columns=['country_id', 'made_profit'])
#countries.loc[len(countries)] = ['Q123',1.0]

def add_country_profit(combined_row):
    countries.loc[len(countries)] = [combined_row['country_of_origin'], combin
ed_row['made_profit']]
    return

combined_with_countries = combined[combined['country_of_origin'].notnull()]
combined_with_countries.apply(add_country_profit, axis=1)
countries_groupby = countries.groupby(['country_id'])
countries_avg = countries_groupby.mean()
countries_count = countries_groupby.count()
countries_sum = countries_groupby.sum()
countries_stats = countries_avg
countries_stats['total'] = countries_count
countries_stats['sum'] = countries_sum
countries_stats = countries_stats.reset_index()
countries_stats.columns = ['country_id','percent','total','profit']
countries_stats['loss'] = countries_stats['total'] - countries_stats['profit']
#countries_stats = countries_stats[countries_stats['profit'] > 5]
#countries_stats = countries_stats[countries_stats['loss'] > 5]
countries_stats = countries_stats[countries_stats['total'] > 5]
```

In [23]:
```
contingency = countries_stats[['profit','loss']]
chi2, p, dof, expected = stats.chi2_contingency(contingency)
print(p) # p < 0.05, therefore country effects profitability
```

```
0.008871929501706175
```

In [24]:
```
countries_stats.sort_values(by='percent',ascending=False).reset_index(drop=Tru
e) #Country Q30, presumably the US, is the best country to make a movie in for
 profit
```

Out[24]:

	country_id	percent	total	profit	loss
0	Q16	0.875000	8	7.0	1.0
1	Q30	0.860465	602	518.0	84.0
2	Q142	0.838710	31	26.0	5.0
3	Q145	0.838710	62	52.0	10.0
4	Q159	0.677419	31	21.0	10.0
5	Q408	0.636364	11	7.0	4.0
6	Q183	0.625000	16	10.0	6.0

Does cast member have an effect on profitability?

In [25]:
```
cast = pd.DataFrame(columns=['cast_id', 'made_profit'])

def add_cast_profit(combined_row):
    for cast_member in combined_row['cast_member']:
        cast.loc[len(cast)] = [cast_member, combined_row['made_profit']]
    return

combined_with_cast = combined[combined['cast_member'].notnull()]
combined_with_cast.apply(add_cast_profit, axis=1)
cast_groupby = cast.groupby(['cast_id'])
```

In [26]:
```
cast_avg = cast_groupby.mean()
cast_count = cast_groupby.count()
cast_sum = cast_groupby.sum()
cast_stats = cast_avg
cast_stats['total'] = cast_count
cast_stats['sum'] = cast_sum
cast_stats = cast_stats.reset_index()
cast_stats.columns = ['cast_id','percent','total','profit']
cast_stats['loss'] = cast_stats['total'] - cast_stats['profit']
cast_stats = cast_stats[cast_stats['total'] > 5]
cast_stats
```

Out[26]:

	cast_id	percent	total	profit	loss
7	Q102124	0.833333	6	5.0	1.0
20	Q103157	0.636364	11	7.0	4.0
41	Q104061	1.000000	7	7.0	0.0
55	Q104791	0.666667	6	4.0	2.0
84	Q1060758	0.833333	6	5.0	1.0
98	Q106706	1.000000	6	6.0	0.0
107	Q10738	0.500000	6	3.0	3.0
159	Q110374	1.000000	8	8.0	0.0
184	Q112536	1.000000	6	6.0	0.0
193	Q113206	1.000000	7	7.0	0.0
194	Q1132632	1.000000	6	6.0	0.0
379	Q123351	0.818182	11	9.0	2.0
401	Q125017	1.000000	6	6.0	0.0
402	Q125106	0.833333	6	5.0	1.0
407	Q125354	0.714286	7	5.0	2.0
411	Q125904	1.000000	6	6.0	0.0
470	Q129591	1.000000	10	10.0	0.0
472	Q129817	0.666667	6	4.0	2.0
526	Q132430	1.000000	8	8.0	0.0
528	Q132616	0.875000	8	7.0	1.0
539	Q133313	1.000000	6	6.0	0.0
690	Q1388769	0.714286	7	5.0	2.0
757	Q14537	0.714286	7	5.0	2.0
786	Q150482	1.000000	6	6.0	0.0
805	Q151168	1.000000	6	6.0	0.0
952	Q160432	0.666667	6	4.0	2.0
990	Q161916	1.000000	9	9.0	0.0
1061	Q162492	0.714286	7	5.0	2.0
1088	Q164119	0.777778	9	7.0	2.0
1111	Q165219	1.000000	8	8.0	0.0
...	...	...	...	...	...
6273	Q481832	0.875000	8	7.0	1.0

	cast_id	percent	total	profit	loss
6275	Q483118	0.916667	12	11.0	1.0
6277	Q48337	0.866667	15	13.0	2.0
6280	Q483771	0.888889	9	8.0	1.0
6281	Q483907	1.000000	6	6.0	0.0
6434	Q503706	1.000000	6	6.0	0.0
6510	Q511554	0.714286	7	5.0	2.0
6574	Q520651	0.833333	6	5.0	1.0
6679	Q532169	1.000000	6	6.0	0.0
6723	Q53680	0.833333	6	5.0	1.0
6777	Q54314	1.000000	11	11.0	0.0
6949	Q57147	0.777778	9	7.0	2.0
6967	Q57614	0.714286	7	5.0	2.0
6985	Q58444	0.888889	9	8.0	1.0
7135	Q621490	1.000000	6	6.0	0.0
7260	Q65932	0.818182	11	9.0	2.0
7417	Q705602	1.000000	6	6.0	0.0
7545	Q722001	0.857143	7	6.0	1.0
7608	Q73007	0.833333	6	5.0	1.0
7799	Q777625	1.000000	6	6.0	0.0
7874	Q80405	1.000000	6	6.0	0.0
7888	Q80966	0.916667	12	11.0	1.0
7892	Q81328	1.000000	8	8.0	0.0
7895	Q81520	1.000000	6	6.0	0.0
7916	Q83338	0.875000	8	7.0	1.0
7922	Q83492	1.000000	8	8.0	0.0
7959	Q873	1.000000	6	6.0	0.0
7977	Q8927	0.900000	10	9.0	1.0
8024	Q920607	1.000000	6	6.0	0.0
8071	Q935167	0.666667	6	4.0	2.0

202 rows × 5 columns

In [27]:
```
contingency = cast_stats[['profit','loss']]
chi2, p, dof, expected = stats.chi2_contingency(contingency)
print(p) # p > 0.05, therefore cast effects doesn't effect profitability.
```

```
0.21051014915423152
```

How well can we predict profitability based on ratings?

In [28]:
```
predict_profit = combined
predict_profit = predict_profit[predict_profit['critic_average'].notnull()]
predict_profit = predict_profit[predict_profit['audience_average'].notnull()]
predict_profit = predict_profit[predict_profit['critic_percent'].notnull()]
predict_profit = predict_profit[predict_profit['audience_percent'].notnull()]
predict_profit = predict_profit.reset_index(drop=True)
X = predict_profit[['critic_average','audience_average','critic_percent','audi
ence_percent']]
y = predict_profit['made_profit']

X_train, X_test, y_train, y_test = model_selection.train_test_split(X,y)
model = make_pipeline(
    StandardScaler(),
    SVC(kernel='rbf', C=20000)
)
model.fit(X_train, y_train)
print(model.score(X_test, y_test)) #0.8+ score, so pretty well
#model.fit(X, y)
```

```
0.7554347826086957
```

Can we predict things based on genre? (nope)

I didn't realise that X needs to be floats... gg what a waste of time T_T

In [29]:
```
#def genre_average_rating_agg(combined_row):
#    for genre_id in combined_row['genre']:
#        genre_test.loc[genre_test['wikidata_id'] == genre_id,'total']+=1
#        genre_test.loc[genre_test['wikidata_id'] == genre_id,'total_aud_avg']
+=combined_row['audience_average']
#        genre_test.loc[genre_test['wikidata_id'] == genre_id,'total_cri_avg']
+=combined_row['critic_average']
#        if (combined_row['made_profit'] == 1.0):
#            genre_test.loc[genre_test['wikidata_id'] == genre_id,'profit']+=1
```

In [30]:
```
#combined_no_nan_ratings = combined[combined['critic_average'].notnull()]
#combined_no_nan_ratings = combined_no_nan_ratings[combined_no_nan_ratings['au
dience_average'].notnull()].reset_index(drop=True)
#genre_test = genres
#genre_test['profit'] = 0
#genre_test['total_aud_avg'] = 0
#genre_test['total_cri_avg'] = 0
#genre_test['total'] = 0
#combined.apply(genre_average_rating_agg, axis=1)
#genre_test = genre_test[genre_test['total'] > 0]
```

In [31]:
```
#genre_test.loc[:,'aud_avg'] = genre_test['total_aud_avg']/genre_test['total']
#genre_test.loc[:,'cri_avg'] = genre_test['total_cri_avg']/genre_test['total']
#Dont know about the SettingWithCopyWarning, can probably just ignore since it
 is just a warning
```

In [32]:
```
#genre_test = genre_test.reset_index(drop=True)
#X = genre_test.drop(columns=['aud_avg','cri_avg','profit','total','total_aud_
avg','total_cri_avg','genre_label'])
#y = genre_test['aud_avg']

#X_train, X_test, y_train, y_test = model_selection.train_test_split(X,y)
#model = make_pipeline(
#    StandardScaler(),
#    SVC(kernel='rbf', C=20000)
#)
#model.fit(X_train, y_train)
#print(model.score(X_test, y_test))
#model.fit(X, y)
```

NATURAL LANGUAGE PROCESSING WORK HERE

In [33]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [34]:
```
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(test3["omdb_plot"])
```

In [35]:
```
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
```

In [36]:
```
from sklearn.naive_bayes import MultinomialNB
y=test3["audience_average"]
y=y.astype('int')
clf = MultinomialNB().fit(X_train_tfidf, y)
```

In [37]:
```
docs_new = ['The']
X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)
predicted = clf.predict(X_new_tfidf)
```

In [38]:
```
for doc, category in zip(docs_new, predicted):
     print('%r => %s' % (doc, test3["omdb_plot"][category])) #not working?
```

```
'The' => Lt. John Dunbar is dubbed a hero after he accidentally leads Union t
roops to a victory during the Civil War. He requests a position on the wester
n frontier, but finds it deserted. He soon finds out he is not alone, but mee
ts a wolf he dubs "Two-socks" and a curious Indian tribe. Dunbar quickly make
s friends with the tribe, and discovers a white woman who was raised by the I
ndians. He gradually earns the respect of these native people, and sheds his
white-man's ways.
```

In [39]:
```
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(test3["omdb_plot"])
feature = vectorizer.get_feature_names()
vocab = np.array(feature)
```

In [40]:
```
feature #list of key words extracted
```

Out[40]:
```
['000',
 '007',
 '04',
 '10',
 '100',
 '1000',
 '100th',
 '101',
 '10s',
 '10th',
 '11',
 '1100',
 '1101',
 '117',
 '12',
 '120',
 '1200',
 '127',
 '12th',
 '13',
 '1357',
 '13th',
 '14',
 '140',
 '15',
 '155',
 '15th',
 '16',
 '1621',
 '163',
 '1630',
 '16th',
 '17',
 '170',
 '1776',
 '1790s',
 '18',
 '1823',
 '1839',
 '1848',
 '1860',
 '1863',
 '1865',
 '1868',
 '1890',
 '1890s',
 '1899',
 '1912',
 '1914',
 '1918',
 '1920s',
 '1926',
 '1930',
 '1930s',
 '1939',
 '1940',
 '1941',
```

```
'1942',
'1944',
'1945',
'1946',
'1950',
'1950s',
'1955',
'1956',
'1957',
'1958',
'1959',
'1960',
'1960s',
'1961',
'1962',
'1963',
'1964',
'1967',
'1968',
'1969',
'1970',
'1970s',
'1971',
'1972',
'1976',
'1977',
'1979',
'1980',
'1980s',
'1982',
'1984',
'1987',
'1990s',
'1991',
'1993',
'1994',
'1996',
'19th',
'1st',
'20',
'200',
'2000',
'2001',
'2002',
'2003',
'2004',
'2005',
'2009',
'2010',
'2012',
'2014',
'2018',
'2019',
'2029',
'2054',
'2058',
'2093',
```

```
'20s',
'20th',
'21',
'2154',
'22',
'220',
'23',
'24',
'24601',
'25',
'250',
'258',
'26',
'27',
'28',
'28th',
'29',
'2e',
'2nd',
'30',
'300',
'300m',
'30m',
'32',
'3234',
'33',
'35',
'36',
'3d',
'3po',
'40',
'400',
'426',
'43',
'44',
'48',
'480',
'50',
'500',
'50s',
'52',
'56',
'57',
'5th',
'60',
'626',
'64',
'65',
'657',
'66',
'70',
'700',
'73',
'78',
'80',
'800',
'84',
```

```
'90',
'911',
'92s',
'999',
'9th',
'aaa',
'aaron',
'abalam',
'abandoned',
'abandons',
'abbey',
'abbie',
'abducted',
'abduction',
'abducts',
'abe',
'abel',
'abernathy',
'abigail',
'abilene',
'abilities',
'ability',
'abin',
'abject',
'able',
'aboard',
'abolitionist',
'abomination',
'abort',
'abortion',
'abortions',
'abound',
'about',
'abouts',
'above',
'abraham',
'abramovich',
'abrasive',
'abroad',
'abrupt',
'abruptly',
'absconded',
'absence',
'absent',
'absolute',
'absolutely',
'abundant',
'abuse',
'abused',
'abusive',
'academy',
'accelerated',
'accept',
'accepted',
'accepting',
'accepts',
'access',
```

```
'accessible',
'accession',
'accident',
'accidentally',
'acclaimed',
'accommodate',
'accommodating',
'accompanied',
'accompanies',
'accompany',
'accomplice',
'accomplish',
'accomplished',
'accomplishing',
'according',
'accordingly',
'account',
'accountant',
'accounting',
'accusations',
'accused',
'accuser',
'accuses',
'accusing',
'acerbic',
'achieve',
'achieved',
'achievements',
'achieves',
'achieving',
'achillas',
'achilles',
'acknowledge',
'acknowledging',
'acme',
'acquaintance',
'acquire',
'acquires',
'acquisitions',
'acrobat',
'across',
'act',
'acting',
'action',
'actions',
'actium',
'activate',
'active',
'actively',
'activist',
'activities',
'activity',
'actor',
'actors',
'actress',
'acts',
'actually',
```

```
'ad',
'ada',
'adaline',
'adam',
'adams',
'adaptation',
'adapted',
'add',
'added',
'addict',
'addiction',
'addictive',
'adding',
'addition',
'address',
'adelie',
'adelies',
'adenoid',
'adept',
'adequate',
'adhd',
'adjust',
'administer',
'administration',
'administrative',
'administrator',
'administrators',
'admiral',
'admiration',
'admire',
'admiring',
'admission',
'admit',
'admits',
'admitted',
'admitting',
'adolescence',
'adolescent',
'adolescents',
'adopted',
'adopts',
'adrenaline',
'adrift',
'adulation',
'adult',
'adulthood',
'adults',
'advance',
'advanced',
'advancements',
'advancing',
'advantage',
'advantages',
'adventure',
'adventurer',
'adventurers',
'adventures',
```

```
'adventurous',
'adversarial',
'adversary',
'adverse',
'advertised',
'advertising',
'advice',
'advise',
'advised',
'adviser',
'advisers',
'advises',
'advisor',
'aether',
'affair',
'affairs',
'affect',
'affected',
'affecting',
'affections',
'affects',
'affinity',
'afflicted',
'affluent',
'afghanistan',
'afghans',
'afoul',
'afraid',
'africa',
'african',
'africans',
'after',
'aftermath',
'afternoon',
'afterwards',
'again',
'against',
'agatha',
'age',
'aged',
'agee',
'agencies',
'agency',
'agenda',
'agendas',
'agent',
'agents',
'ages',
'aggressive',
'aggressiveness',
'aging',
'ago',
'agreement',
'agrees',
'aguilar',
'ahead',
'ahkmenrah',
```

```
'aibileen',
'aid',
'aide',
'aided',
'ailing',
'aim',
'aiming',
'aimlessly',
'ain',
'air',
'airborne',
'aircraft',
'airline',
'airlines',
'airplane',
'airport',
'aka',
'akan',
'akin',
'al',
'ala',
'alabama',
'alain',
'alan',
'alarmed',
'alaska',
'albanese',
'albeit',
'albert',
'alberto',
'alcohol',
'alcoholic',
'alcoholism',
'aldo',
'aleksey',
'alert',
'alex',
'alexander',
'alexey',
'alexi',
'alfie',
'alfonso',
'alfred',
'alias',
'alibi',
'alice',
'alicia',
'alien',
'alienate',
'alienation',
'aliens',
'alike',
'alimony',
'alive',
'all',
'allegations',
'allegedly',
```

```
'allegiance',
'allegory',
'allen',
'allergic',
'alleviate',
'alleviating',
'alley',
'alliance',
'alliances',
'allied',
'allies',
'alligators',
'allison',
'allow',
'allowed',
'allowing',
'allows',
'alluding',
'allure',
'alluring',
'ally',
'almost',
'alone',
'along',
'alongside',
'aloysius',
'alpha',
'alps',
'already',
'also',
'altars',
'alter',
'altered',
'altering',
'alternate',
'alters',
'although',
'altman',
'alvy',
'always',
'alyssa',
'alzheimer',
'am',
'amadeus',
'amanda',
'amara',
'amasses',
'amateur',
'amato',
'amazement',
'amazing',
'amazon',
'amazons',
'ambassador',
'ambition',
'ambitions',
'ambitious',
```

```
'ambushed',
'amelia',
'amendment',
'america',
'american',
'americans',
'amerika',
'amid',
'amidala',
'amidst',
'amigos',
'amin',
'amistad',
'ammunition',
'amnesia',
'amnesiac',
'among',
'amongst',
'amoral',
'amorous',
'amount',
'amphibious',
'amputation',
'amsterdam',
'amy',
'an',
'anakin',
'analyst',
'analytical',
'anarchist',
'anarchists',
'anastasia',
'anatoly',
'ancestor',
'ancestry',
'ancient',
'and',
'anders',
'anderson',
'anderton',
'andrea',
'andrew',
'android',
'androids',
'andrée',
'anduin',
'andy',
'angel',
'angela',
'angeles',
'angels',
'anger',
'angered',
'angie',
'angier',
'anglo',
'angrily',
```

```
'angry',
'animal',
'animals',
'animated',
'animation',
'animators',
'anita',
'ann',
'anna',
'annabel',
'annabelle',
'annabeth',
'annals',
'anne',
'annie',
'annihilation',
'annika',
'anniversary',
'announces',
'annoy',
'annoyance',
'annoying',
'annual',
'anomaly',
'anonymous',
'anonymously',
'another',
'anshel',
'answer',
'answers',
'ant',
'antarctic',
'antarctica',
'anthony',
'anti',
'antibes',
'anticipate',
'anticipation',
'antidote',
'antiwar',
'antoine',
'anton',
'antonio',
'antony',
'anxieties',
'anxiety',
'anxious',
'anxiously',
'any',
'anya',
'anybody',
'anymore',
'anyone',
'anything',
'anyway',
'anywhere',
'apart',
```

```
'apartment',
'apatosaurus',
'ape',
'apes',
'apocalypse',
'apocalyptic',
'apollo',
'apollodorus',
'apologize',
'apostle',
'app',
'apparent',
'apparently',
'appear',
'appearance',
'appeared',
'appearing',
'appears',
'apple',
'applies',
'appointed',
'appointing',
'appreciate',
'apprehend',
'apprentice',
'approach',
'approached',
'approaches',
'approaching',
'appropriate',
'apps',
'april',
'aquarium',
'aquila',
'arabic',
'aragorn',
'aranha',
'arbiter',
'arcade',
'arch',
'archaeologists',
'archdeacon',
'archenemies',
'archeology',
'archer',
'architect',
'architecture',
'archive',
'archived',
'arctic',
'are',
'area',
'areas',
'aren',
'arendelle',
'arglist',
'argues',
```

```
'arguing',
'argument',
'ari',
'ariel',
'arises',
'aristocrat',
'aristocratic',
'aristocrats',
'arius',
'arizona',
'arliss',
'arlo',
'arm',
'armada',
'armed',
'armies',
'armor',
'armored',
'arms',
'army',
'arnold',
'aron',
'aronnax',
'around',
'arranged',
'arranges',
'array',
'arrest',
'arrested',
'arrests',
'arrival',
'arrive',
'arrived',
'arrives',
'arriving',
'arrogance',
'arrogant',
'arrows',
'art',
'artemus',
'arterton',
'artery',
'arthur',
'articles',
'artie',
'artifact',
'artifacts',
'artificial',
'artificially',
'artist',
'artistic',
'artistically',
'artists',
'arts',
'aryan',
'as',
'ascension',
```

```
'ascent',
'ascribed',
'asgard',
'asgardian',
'asher',
'ashkenazic',
'ashmita',
'ashore',
'ashton',
'asia',
'aside',
'ask',
'asked',
'asking',
'asks',
'asleep',
'aspect',
'aspects',
'aspirations',
'aspires',
'aspiring',
'ass',
'assange',
'assassin',
'assassinate',
'assassinated',
'assassinating',
'assassination',
'assassinations',
'assassins',
'assaults',
'assembles',
'assembly',
'assertion',
'assertive',
'assets',
'assigned',
'assignment',
'assigns',
'assimilate',
'assist',
'assistance',
'assistant',
'assistants',
'assisted',
'associate',
'associated',
'associates',
'association',
'assortment',
'assumed',
'assumes',
'assuming',
'assumption',
'assumptions',
'assured',
'asteroid',
```

```
'astonishing',
'astounded',
'astounding',
'astray',
'astrid',
'astronaut',
'astronauts',
'astronomer',
'astute',
'asylum',
'at',
'atheist',
'athens',
'athlete',
'athletes',
'athletic',
'atlanta',
'atlantic',
'atlantis',
'atmosphere',
'atrocities',
'attached',
'attack',
'attacked',
'attackers',
'attacking',
'attacks',
'attar',
'attempt',
'attempting',
'attempts',
'attend',
'attendance',
'attendant',
'attended',
'attending',
'attention',
'attentions',
'attic',
'attila',
'attire',
'attitude',
'attorney',
'attorneys',
'attracted',
'attraction',
'attractive',
'attracts',
'attuned',
'auctioned',
'audience',
'audiences',
'audition',
'auditions',
'augur',
'august',
'aunt',
```

```
'auror',
'aurélia',
'auschwitz',
'austin',
'australia',
'australian',
'austrian',
'author',
'authoritative',
'authorities',
'authority',
'authors',
'auto',
'automatic',
'automatically',
'autonomy',
'available',
'avalon',
'avatar',
'avatars',
'ave',
'avenge',
'avengers',
'avenue',
'average',
'averill',
'aviation',
'aviator',
'avidor',
'avigdor',
'avoid',
'avoiding',
'avowed',
'await',
'awaited',
'awaiting',
'awaits',
'awakened',
'awakening',
'awakens',
'awakes',
'awaking',
'award',
'aware',
'away',
'awe',
'awesome',
'awkward',
'awry',
'axel',
'ayatollah',
'aykroyd',
'ayubi',
'azeroth',
'azir',
'babies',
'baby',
```

```
'bachelor',
'back',
'backdrop',
'backed',
'background',
'backgrounds',
'backlash',
'backseat',
'backup',
'backward',
'backwards',
'bacteria',
'bad',
'badass',
'baddest',
'badly',
'baffles',
'baggins',
'baghdad',
'bagheera',
'bail',
'bailey',
'baird',
'baker',
'balance',
'bald',
'baldacci',
'baldwin',
'bale',
'balian',
'ballet',
'balloons',
'balls',
'baloo',
'bambi',
'ban',
'band',
'banderas',
'bandit',
'bandits',
'bane',
'banega',
'bang',
'banged',
'banished',
'banishing',
'banishment',
'bank',
'banker',
'banking',
'banks',
'banky',
'banner',
'bans',
'baptiste',
'bar',
'barbara',
```

```
 'barbaric',
 'barber',
 'barbra',
 'barcelona',
 'bardem',
 'bare',
 'barebone',
 'barely',
 'barents',
 'bargain',
 'bargained',
 'barge',
 'barinholtz',
 'barman',
 'barnes',
 'barney',
 'baron',
 'barrel',
 'barrels',
 'barrens',
 'barrie',
 'barriers',
 'barry',
 'bars',
 'bartender',
 'bartholomew',
 'base',
 'baseball',
 'based',
 'basement',
 'bases',
 ...]
```

In [41]:

```
doc = 0
feature_index = tfidf_matrix[doc,:].nonzero()[1]
tfidf_scores = zip(feature_index, [tfidf_matrix[doc, x] for x in feature_index
])
```

In [42]:
```
for w, s in [(feature[i], s) for (i, s) in tfidf_scores]:
  print (w, s) #showing tfidf score for each word in the summary
```

```
helena 0.44994830287228665
is 0.0313254157653011
young 0.034445749219816774
girl 0.04124583321382185
who 0.04117339272962841
spent 0.06110947647223666
all 0.08785899163498494
her 0.1305810708682046
life 0.028510313451956933
in 0.07483826251551558
space 0.27166123481520066
pod 0.1433487708116238
just 0.03935140526206201
after 0.0556949649512731
birth 0.06110947647223666
traveling 0.056434429793391354
from 0.09141913017567936
earth 0.08753467605892976
to 0.11319755553481937
distant 0.06368234032073075
planet 0.10741769474075274
where 0.03410825925872209
she 0.09254048288700302
will 0.057020626903913865
reunite 0.0669993387269666
with 0.07183623067344654
others 0.05370884737037637
colonials 0.07966643049089307
the 0.2263242798960254
voice 0.06522196846906535
of 0.07819068208685259
on 0.019475290305502265
board 0.06522196846906535
computer 0.056434429793391354
as 0.04303063239856396
only 0.03226771030131625
one 0.027529829222765145
company 0.051015248556804275
arriving 0.07499138381204777
station 0.056434429793391354
for 0.039710874781110767
maintenance 0.07966643049089307
works 0.04965720028419483
meets 0.04484067840939954
álex 0.39833215245446535
repairman 0.07966643049089307
falling 0.06368234032073075
love 0.035031158386641933
him 0.026453704134735613
quickly 0.05150596447800864
but 0.0450506583297649
still 0.04602428940923985
traumatized 0.07966643049089307
by 0.06124914551434492
ghosts 0.07499138381204777
his 0.033483020337606625
own 0.03757403500416075
```

```
past 0.04699924318002639
decides 0.04402645194090731
some 0.041665155199113665
days 0.048442384708310186
later 0.043265951044277336
meet 0.04699924318002639
break 0.05370884737037637
rules 0.06001054211106964
and 0.074952965826134
reveal 0.0669993387269666
truth 0.05311743138715548
that 0.07765253347572305
part 0.052018497025988486
secret 0.08604640694344622
experiment 0.06001054211106964
test 0.054332246963040134
effects 0.06368234032073075
human 0.045417302750039114
body 0.052554876705138864
an 0.0207595644276498
elongated 0.07966643049089307
travel 0.048442384708310186
was 0.0336731101140325
offered 0.06910152155731782
fathers 0.07499138381204777
well 0.04402645194090731
another 0.046665014450408916
babies 0.07499138381204777
hope 0.052554876705138864
runaway 0.06910152155731782
progressively 0.07966643049089307
more 0.0381891186326625
polluted 0.07966643049089307
radioactive 0.07966643049089307
escaping 0.0669993387269666
fake 0.0716743854058119
exits 0.07499138381204777
world 0.029885430689653762
discovering 0.06522196846906535
it 0.025574796223136246
at 0.05407822660312154
side 0.052554876705138864
being 0.03515189335171995
both 0.04188067705934655
prosecuted 0.07966643049089307
hugo 0.1433487708116238
director 0.05722992338398419
project 0.17425318354189676
searches 0.07499138381204777
way 0.03684863332431837
keep 0.050544567538661406
during 0.04145358809241322
connivance 0.07966643049089307
realizes 0.052554876705138864
revelation 0.07499138381204777
change 0.05150596447800864
same 0.046665014450408916
```

```
time 0.03346154300733206
receives 0.06522196846906535
superior 0.07499138381204777
order 0.04699924318002639
eliminate 0.06910152155731782
before 0.04045033962322902
they 0.023543350325146834
can 0.03217358664547307
spoil 0.07966643049089307
press 0.0716743854058119
public 0.054332246963040134
eye 0.05900729364188544
```