The Rotten Tomatoes Effect

By: Brandon Zhao

Growing up, my family would have "Movie Night" every Saturday, where we would all get together and watch some movie that my dad would pick out for us together. These movie nights are some of my favorite memories from my childhood, and they also instilled in me a love for the world of cinema. Even now, I continue to try to watch and learn about important films in cinema, and one my greatest resources in doing so has been Rotten Tomatoes.

Practically everyone in the modern world who watches movies has heard of Rotten Tomatoes, but for those who have not, it is a Review-Aggregation Website for films and tv shows. It takes reviews from critics and uses them to gives films a rating out of 100, as well as a label of "Fresh" or "Rotten". It's by far one of the most popular about films, and I myself use it quite frequently to learn more about different films and decide which ones to watch.

One day while using Rotten Tomatoes, I became curious about its origins and decided to look it up on Wikipedia. After reading about how the site came to be, I came to an interesting section of the Wikipedia page that talked about the site's influence on the film industry. (https://en.wikipedia.org/wiki/Rotten_Tomatoes#Influence) It stated how many film studios have come to dislike Rotten Tomatoes because of how much impact a bad rating on the site can have on the box office success of a film, named "The Rotten Tomatoes Effect". I was instantly interested in whether or not this phenomena was real. Will a bad rating or a "Rotten" label lead to less box office earnings? Will a good rating or a "Fresh" Label lead to more?

In this data analysis project, I will be answering these questions. I will be looking into how reviews on Rotten Tomatoes impact a film's box office earnings. I will also attempt to create a model that will predict a film's earnings based on its reviews on Rotten Tomatoes.

These are the packages I will be using in this project

In [1]:

```
# importing packages
import numpy as np
import pandas as pd
pd.options.mode.chained_assignment = None
import matplotlib.pyplot as plt
import seaborn as sns
from pandasql import sqldf
import scipy
from sklearn.linear_model import LinearRegression
import sklearn.model_selection
```

Obtaining the Data

For this data analysis project, I will be using two datasets of movies from Kaggle. The first is from https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset, and it contains information taken from the IMDb pages of movies. The second is from https://www.kaggle.com/heyueyuan/rottentomatoesmoviesandcriticsdatasets/data, and it contains information taken from the Rotten Tomatoes pages of movies. The two datasets will need to be loaded into the workspace as "imdb_movies.csv" and "rotten_tomatoes_movies.csv" for this script to run properly.

In [2]:

```
# importing data
imdb_df = pd.read_csv("imdb_movies.csv")
rt_df = pd.read_csv("rotten_tomatoes_movies.csv")
```

Here are the available fields from each original dataset.

In [3]:

```
# printing summary statistics
print("Field names of IMDb Dataset:")
print(imdb_df.columns)
print("\n")
print("Field names of Rotten Tomatoes Dataset:")
print(rt_df.columns)
```

```
Field names of IMDb Dataset:
Index(['imdb_title_id', 'title', 'original_title', 'year', 'date_published',
       'genre', 'duration', 'country', 'language', 'director', 'writer',
       'production_company', 'actors', 'description', 'avg_vote', 'votes',
       'budget', 'usa_gross_income', 'worlwide_gross_income', 'metascore',
       'reviews_from_users', 'reviews_from_critics'],
```

dtype='object')

Field names of Rotten Tomatoes Dataset:
Index(['rotten_tomatoes_link', 'movie_title', 'movie_info',
'critics_consensus', 'poster_image_url', 'rating', 'genre', 'directors',
'writers', 'cast', 'in_theaters_date', 'on_streaming_date',
'runtime_in_minutes', 'studio_name', 'tomatometer_status',
'tomatometer_rating', 'tomatometer_count', 'audience_status',
'audience_rating', 'audience_count', 'audience_top_critics_count',
'audience_fresh_critics_count', 'audience_rotten_critics_count'],
dtype='object')

Data Cleaning

To get the necessary rows and fields from each dataset, I will be using the package pandasql. This allows for working with Pandas dataframes with SQL statements. I will be using this package because it allows for easier manipulation of dataframes and it allows me to practice my SQL querying skills. I will need the title, year, country, budget, and worlwide_gross_income from the IMDb Dataset and the tomatometer_status, tomatometer_rating, and audience_rating from the Rotten Tomatoes Dataset. Furthermore, I will need to filter for title matches between the two dataframes, budget and worlwide_gross_income values that aren't missing, and movies released after 1998, the year Rotten Tomatoes was founded.

In [4]:

```
# getting needed data from datasets
movies_df = sqldf("""SELECT title, year, country, budget, worlwide_gross_income AS income,
                     tomatometer_status, tomatometer_rating, audience_rating
                  FROM imdb_df
                  INNER JOIN rt_df
                  ON imdb_df.title = rt_df.movie_title
                  WHERE year > 1998 AND budget IS NOT NULL AND worlwide_gross_income IS NOT NULL""", globals())
```

This is information about the resulting dataframe to be used for the rest of this data analysis

In [5]:

```
# printing summary statistics
print(movies_df.head())
print("\n")
print(movies_df.describe(include = "all"))
print("\n")
print(movies_df.dtypes)
```

```
            title  year country      budget       income tomatometer_status  \
0   Kate & Leopold  2001     USA  $ 48000000   $ 76019048            Rotten
1  The Fantasticks  2000     USA  $ 10000000      $ 49666            Rotten
2          Glitter  2001     USA  $ 22000000    $ 5271666            Rotten
3    Baby Geniuses  1999     USA  $ 12000000   $ 36450736            Rotten
4      Three Kings  1999     USA  $ 75000000  $ 107752036    Certified Fresh

   tomatometer_rating  audience_rating
0                  50             62.0
1                  50             49.0
2                   7             48.0
3                   2             24.0
4                  94             77.0


        title         year country      budget     income tomatometer_status  \
count    4706  4706.000000    4706        4706       4706              4706
unique   4251          NaN     697         569       4321                 3
top      Home          NaN     USA  $ 20000000  $ 2046433            Rotten
freq       10          NaN    2488         182          5              2715
mean      NaN  2008.980663     NaN         NaN        NaN               NaN
std       NaN     5.622848     NaN         NaN        NaN               NaN
min       NaN  1999.000000     NaN         NaN        NaN               NaN
25%       NaN  2005.000000     NaN         NaN        NaN               NaN
50%       NaN  2009.000000     NaN         NaN        NaN               NaN
75%       NaN  2014.000000     NaN         NaN        NaN               NaN
max       NaN  2019.000000     NaN         NaN        NaN               NaN

        tomatometer_rating  audience_rating
count          4706.000000      4685.000000
unique                 NaN              NaN
top                    NaN              NaN
freq                   NaN              NaN
mean             51.355291        57.370971
std              27.702189        19.372110
```

```
min            0.000000       5.000000
25%           27.000000      42.000000
50%           51.000000      58.000000
75%           76.000000      73.000000
max          100.000000      97.000000
```

```
title                object
year                  int64
country              object
budget               object
income               object
tomatometer_status   object
tomatometer_rating    int64
audience_rating     float64
dtype: object
```

The following data cleaning steps before analysis on the data can begin:

1) Strings were imported as objects, but should be converted back to strings

2) Budget and Income have a dollar sign and space in front- this should be gotten rid of so that they can be converted from strings to numeric values. Some budgets are also of different currencies. These values need to be converted to American dollars, and then to numeric values.

3) The country has 697 unique values, which seems incorrect as there are only 195 countries in the world. This is because some values in the country column contain a list of countries that the movie was made in, in order of importance. A new column titled from_USA should be added that would contain boolean values as to whether the country or the first country in the list of countries is USA.

4) A new column titled earnings should be added that would tell net gain/loss of a movie in terms of a movie by subtracting its budget from its income.

In [6]:

```
# Converting columns that are objects into strings
movies_df["title"] = movies_df["title"].astype(str)
movies_df["country"] = movies_df["country"].astype(str)
movies_df["budget"] = movies_df["budget"].astype(str)
movies_df["income"] = movies_df["income"].astype(str)
movies_df["tomatometer_status"] = movies_df["tomatometer_status"].astype(str)
movies_df["tomatometer_rating"] = movies_df["tomatometer_rating"].astype(float)

# Converting all Budget and Income into USD and then into numeric values
movies_df["income"] = movies_df["income"].str.replace("$", "")
movies_df["budget"] = movies_df["budget"].str.replace("$", "")
for i, b in movies_df["budget"].items() :
    if b[0:3] == "GBP" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * 1.3
    elif b[0:3] == "EUR" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * 1.18
    elif b[0:3] == "DEM" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .6
    elif b[0:3] == "CAD" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .75
    elif b[0:3] == "AUD" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .72
    elif b[0:3] == "INR" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .013
    elif b[0:3] == "DKK" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .16
    elif b[0:3] == "NOK" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .11
    elif b[0:3] == "JPY" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .0095
    elif b[0:3] == "EGP" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .063
    elif b[0:3] == "THB" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .032
    elif b[0:3] == "SEK" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .11
    elif b[0:3] == "PLN" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .27
    elif b[0:3] == "ARS" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .014
    elif b[0:3] == "NZD" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .66
    elif b[0:3] == "NGN" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .0026
    elif b[0:3] == "CNY" :
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .14
    elif b[0:3] == "KRW" :
```

```
        movies_df["budget"][i] = float(movies_df["budget"][i][3:]) * .00084
movies_df["income"] = movies_df["income"].astype(float)
movies_df["budget"] = movies_df["budget"].astype(float)

# Creating from_USA column
movies_df["from_USA"] = False
for i, c in movies_df["country"].items() :
    movies_df["from_USA"][i] = (movies_df["country"][i][0:3] == "USA")

# Creating earnings column
movies_df["earnings"] = movies_df["income"] - movies_df["budget"]
```

This is the cleaned dataframe that is now ready to be used for analysis

In [7]:

```
# printing summary statistics
print(movies_df.head())
print("\n")
print(movies_df.describe(include = "all"))
print("\n")
print(movies_df.dtypes)
```

```
            title  year country      budget       income tomatometer_status  \
0   Kate & Leopold  2001     USA  48000000.0   76019048.0            Rotten
1  The Fantasticks  2000     USA  10000000.0      49666.0            Rotten
2          Glitter  2001     USA  22000000.0    5271666.0            Rotten
3    Baby Geniuses  1999     USA  12000000.0   36450736.0            Rotten
4      Three Kings  1999     USA  75000000.0  107752036.0    Certified Fresh

   tomatometer_rating  audience_rating  from_USA    earnings
0                50.0             62.0      True  28019048.0
1                50.0             49.0      True  -9950334.0
2                 7.0             48.0      True -16728334.0
3                 2.0             24.0      True  24450736.0
4                94.0             77.0      True  32752036.0


        title         year country        budget        income  \
count   4706  4706.000000    4706  4.706000e+03  4.706000e+03
unique  4251          NaN     697           NaN           NaN
top     Home          NaN     USA           NaN           NaN
freq      10          NaN    2488           NaN           NaN
mean     NaN  2008.980663     NaN  3.460721e+07  9.515962e+07
std      NaN     5.622848     NaN  4.361019e+07  1.895866e+08
min      NaN  1999.000000     NaN  1.870000e+00  2.400000e+01
25%      NaN  2005.000000     NaN  6.105000e+06  2.128990e+06
50%      NaN  2009.000000     NaN  1.900000e+07  2.639905e+07
75%      NaN  2014.000000     NaN  4.400000e+07  9.749679e+07
max      NaN  2019.000000     NaN  3.560000e+08  2.797801e+09

       tomatometer_status  tomatometer_rating  audience_rating from_USA  \
count                4706         4706.000000      4685.000000     4706
unique                  3                 NaN              NaN        2
top                Rotten                 NaN              NaN     True
freq                 2715                 NaN              NaN     3353
mean                  NaN           51.355291        57.370971      NaN
std                   NaN           27.702189        19.372110      NaN
min                   NaN            0.000000         5.000000      NaN
25%                   NaN           27.000000        42.000000      NaN
50%                   NaN           51.000000        58.000000      NaN
75%                   NaN           76.000000        73.000000      NaN
max                   NaN          100.000000        97.000000      NaN

           earnings
count   4.706000e+03
unique           NaN
top              NaN
freq             NaN
mean    6.055241e+07
std     1.583541e+08
min    -1.110072e+08
25%    -4.414990e+06
50%     4.332754e+06
75%     5.744224e+07
max     2.552968e+09


title                object
year                  int64
```

```
country              object
budget              float64
income              float64
tomatometer_status   object
tomatometer_rating  float64
audience_rating     float64
from_USA               bool
earnings            float64
dtype: object
```

There seems to be some NA values in the audience_rating column. As this makes up a relatively small number of rows, rows with NA values will be dropped.

In [8]:

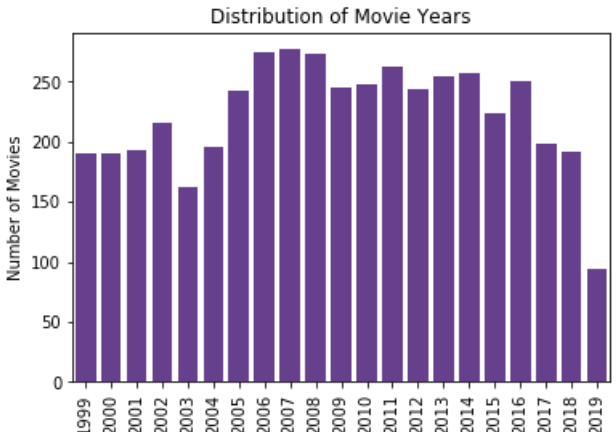
```
# getting rid of columns with NA values
movies_df = movies_df.dropna()
```

Exploring Data Analysis

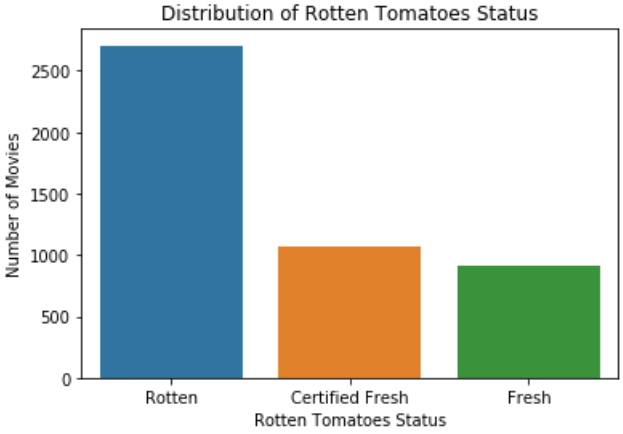
First, here are the distributions of each of the columns

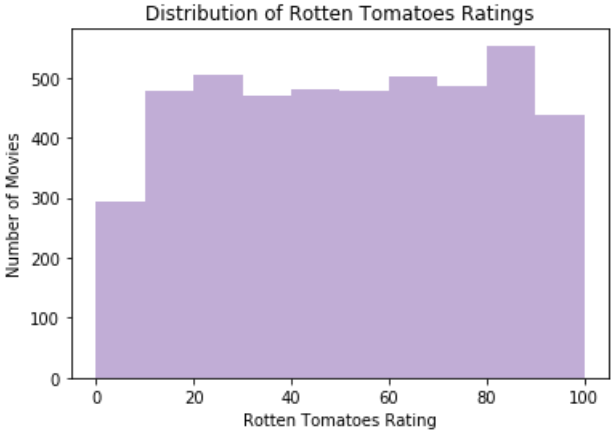
In [9]:

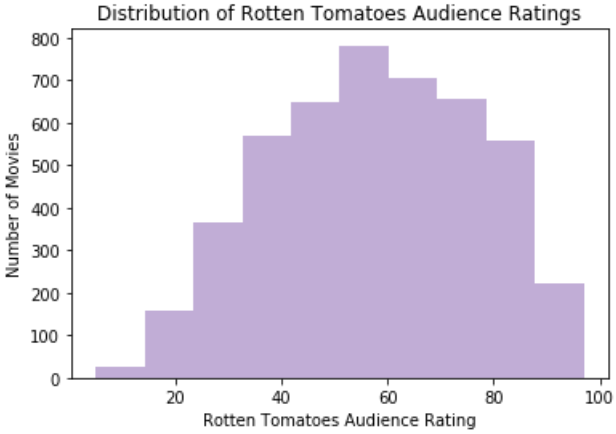
```
# creating summary plots
sns.countplot(movies_df.year, color = "rebeccapurple")
plt.xticks(rotation = 90)
plt.ylabel("Number of Movies")
plt.xlabel("Year")
plt.title("Distribution of Movie Years")
plt.show()

sns.countplot(movies_df.tomatometer_status)
plt.ylabel("Number of Movies")
plt.xlabel("Rotten Tomatoes Status")
plt.title("Distribution of Rotten Tomatoes Status")
plt.show()

sns.distplot(movies_df.tomatometer_rating, kde = False, bins = 10, color = "rebeccapurple")
plt.ylabel("Number of Movies")
plt.xlabel("Rotten Tomatoes Rating")
plt.title("Distribution of Rotten Tomatoes Ratings")
plt.show()

sns.distplot(movies_df.audience_rating, kde = False, bins = 10, color = "rebeccapurple")
plt.ylabel("Number of Movies")
plt.xlabel("Rotten Tomatoes Audience Rating")
plt.title("Distribution of Rotten Tomatoes Audience Ratings")
plt.show()

sns.countplot(movies_df.from_USA)
plt.ylabel("Number of Movies")
plt.xlabel("Origin")
plt.title("Distribution of Movies from USA")
plt.show()

sns.distplot(movies_df.earnings, kde = False, color = "rebeccapurple")
plt.ylabel("Number of Movies")
plt.xlabel("Earnings (in billions of dollars)")
plt.title("Distribution of Film Earnings")
plt.show()
```

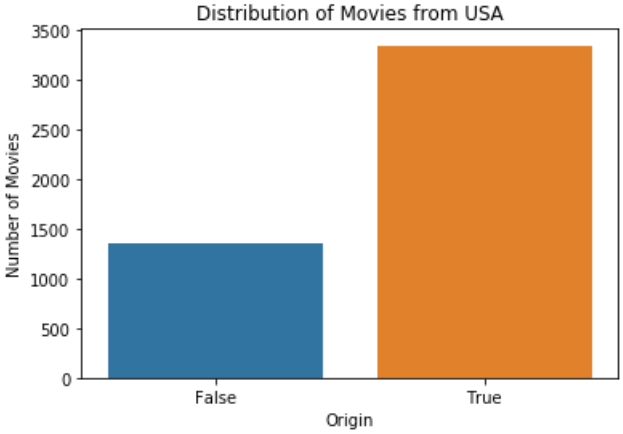


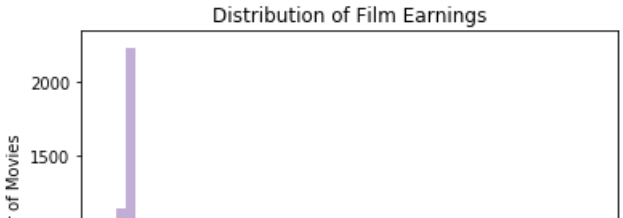
Year

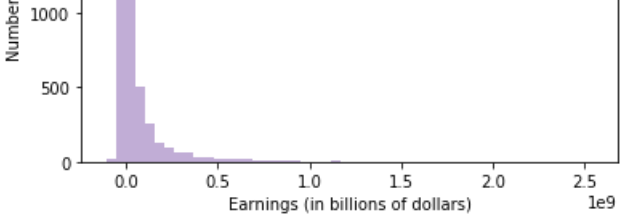

Distribution of Rotten Tomatoes Status
Number of Movies
2500
2000
1500
1000
500
0
Rotten
Certified Fresh
Fresh
Rotten Tomatoes Status


Distribution of Rotten Tomatoes Ratings
Number of Movies
500
400
300
200
100
0
0
20
40
60
80
100
Rotten Tomatoes Rating


Distribution of Rotten Tomatoes Audience Ratings
Number of Movies
800
700
600
500
400
300
200
100
0
20
40
60
80
100
Rotten Tomatoes Audience Rating


Distribution of Movies from USA
Number of Movies
3500
3000
2500
2000
1500
1000
500
0
False
True
Origin


Distribution of Film Earnings
2000
1500



The distribution of years is approximately uniform. The year 2019 seems to have less films than the other years- this is likely because the dataset was originally created in mid-2019, so not all films that were released in 2019 are in the dataset.

The distribution of Rotten Tomatotoes Status indicates that the total number of movies with the "Rotten" status is slightly more than the number of movies with the "Certified Fresh" and "Fresh" status. The number of movies with the "Certified Fresh" and "Fresh" status is approximately equal.

The distribution of Rotten Tomatoes Ratings is approximately uniform. However, there seems to be less movies with extremely low ratings (0 - 10%).

The distribution of Rotten Tomatoes Audience Ratings is closer to a normal distribution, centered at 50 - 60%. The distribution is slightly left tailed, with more movies receiving higher ratings than lower ratings.

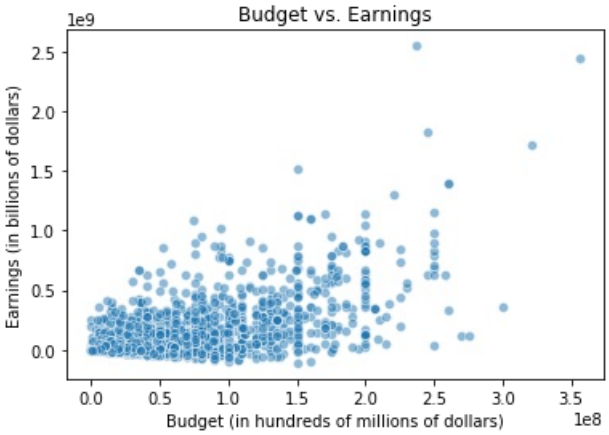
The distribution of movies from the USA shows that there is approximately three times the number of movies made in the USA as movies made outside of the USA in this dataset.

The distribution of film earnings shows that most movies earn 0 - 100 million dollars, with many movies actually losing money. The distribution is also extremely right skewed, meaning that there are many outlier movies that earn much more money than the others.

Now, it is time to take a look at the correlation between budget and earnings

In [10]:

```
# creating budget vs earnings plot
sns.scatterplot(x = "budget", y = "earnings", data = movies_df, alpha = .5)
plt.xlabel("Budget (in hundreds of millions of dollars)")
plt.ylabel("Earnings (in billions of dollars)")
plt.title("Budget vs. Earnings")
plt.show()
print("Pearson Correlation Coefficient: " + str(scipy.stats.pearsonr(movies_df.budget, movies_df.earnings)[0]))
```



Pearson Correlation Coefficient: 0.6489519253306044

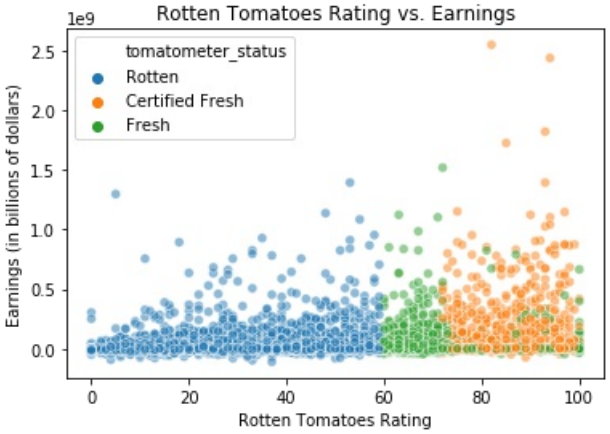
The scatterplot and pearson correlation coefficient indicate that there is a moderate, positive, and linear correlation between budget and earnings. This make sense as higher budget movies will have more money for production and marketing, leading to more people seeing the movie and therefore larger earnings

Now, it is time to take a look at the correlation between Rotten Tomatoes Rating, Rotten Tomatoes Status, Rotten Tomatoes Audience Rating and earnings/budget

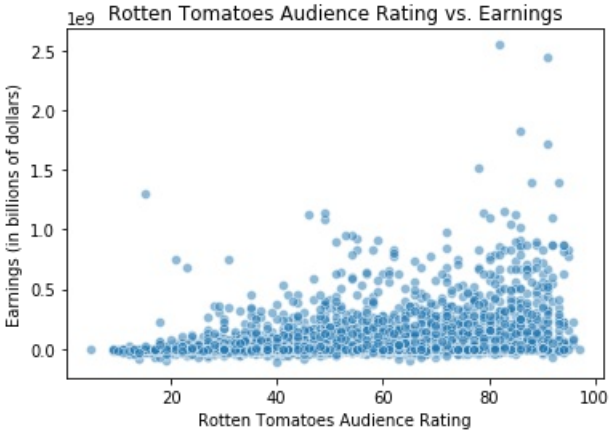
In [11]:

```
# creating ratings vs earnings plots
sns.scatterplot(x = "tomatometer_rating", y = "earnings", data = movies_df, hue = "tomatometer_status", alpha = .5)
plt.xlabel("Rotten Tomatoes Rating")
plt.ylabel("Earnings (in billions of dollars)")
plt.title("Rotten Tomatoes Rating vs. Earnings")
plt.show()
print("Pearson Correlation Coefficient: " + str(scipy.stats.pearsonr(movies_df.tomatometer_rating, movies_df.earnings)[0]))

sns.scatterplot(x = "audience_rating", y = "earnings", data = movies_df, alpha = .5)
```

```
plt.xlabel("Rotten Tomatoes Audience Rating")
plt.ylabel("Earnings (in billions of dollars)")
plt.title("Rotten Tomatoes Audience Rating vs. Earnings")
plt.show()
print("Pearson Correlation Coefficient: " + str(scipy.stats.pearsonr(movies_df.audience_rating, movies_df.earnings)[0]))
```



Pearson Correlation Coefficient: 0.1915282211747506



Pearson Correlation Coefficient: 0.24465953949415498

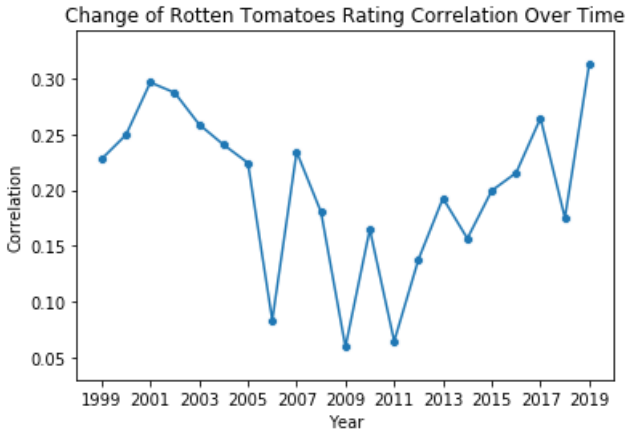
The scatterplots and pearson correlation coefficients indicate that there is a small, positive, and linear correlation between Rotten Tomatoes Rating and earnings, and a slightly larger, positive, and linear correlation between Rotten Tomatoes Audience Rating and earnings. This seems to indicate that "The Rotten Tomatoes Effect", the idea that a film's Rotten Tomatoes Rating will impact its box office performance, is likely heavily exagerrated.

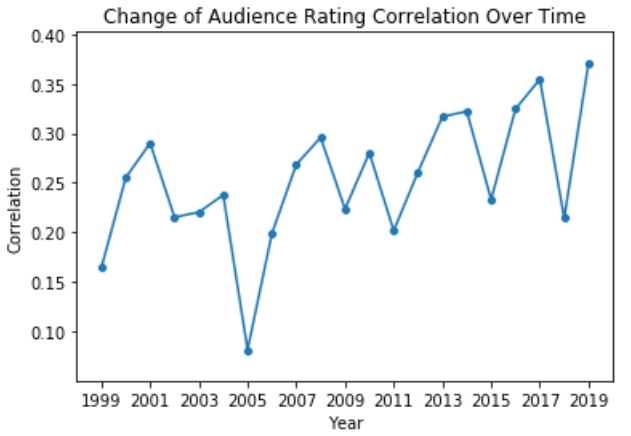
Now, it is time to take a look at how the correlation between Rotten Tomatoes Rating, Rotten Tomatoes Audience Rating, and earnings has changed over time. It would make sense that, as the Rotten Tomatoes site has grown more and more popular, the correlation between them would increase if a film's Rotten Tomatoes Rating did impact its box office performance.

In [12]:

```
# creating lists of years and correlation for those years
years = list(set(movies_df.year.values))
t_corr = []
a_corr = []
for y in years :
    temp_df = movies_df[movies_df.year == y]
    t_corr.append(scipy.stats.pearsonr(temp_df.tomatometer_rating, temp_df.earnings)[0])
    a_corr.append(scipy.stats.pearsonr(temp_df.audience_rating, temp_df.earnings)[0])
correlation_year_df = pd.DataFrame({"year" : years, "tomatometer_correlation" : t_corr, "audience_correlation" : a_corr})

# creating plots for correlation change over time
sns.scatterplot(x = "year", y = "tomatometer_correlation", data = correlation_year_df)
sns.lineplot(x = "year", y = "tomatometer_correlation", data = correlation_year_df)
plt.xticks([1999, 2001, 2003, 2005, 2007, 2009, 2011, 2013, 2015, 2017, 2019])
plt.xlabel("Year")
plt.ylabel("Correlation")
plt.title("Change of Rotten Tomatoes Rating Correlation Over Time")
plt.show()

sns.scatterplot(x = "year", y = "audience_correlation", data = correlation_year_df)
sns.lineplot(x = "year", y = "audience_correlation", data = correlation_year_df)
plt.xticks([1999, 2001, 2003, 2005, 2007, 2009, 2011, 2013, 2015, 2017, 2019])
```

```
plt.xlabel("Year")
plt.ylabel("Correlation")
plt.title("Change of Audience Rating Correlation Over Time")
plt.show()
```





Based on these scatterplots, the correlation between Rotten Tomatoes Rating and earnings started higher approximately .25 - .3 when the Rotten Tomatoes site first launched, decreased over time until around 2008 - 2011 when it reached approximately .10, and then increased over time until present day where it has reached approximately .25 - .3 again. The correlation between Rotten Tomatoes Audience Rating and earnings has generally increased over time, starting at approximately .20 when the Rotten Tomatoes site first launched and reaching approximately .35 in the present day.

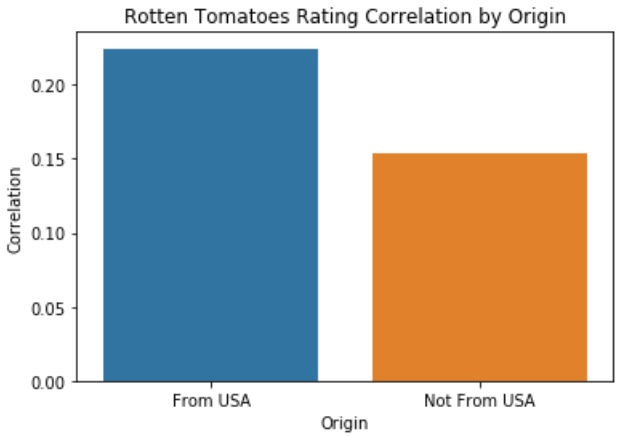
This shows again that the Rotten Tomatoes effect is likely exaggerated- if movie earnings were affected significantly by Rotten Tomatoes Ratings, it would be expected that the correlation between the two would increase over time as the site became more popular. This also supports the idea that the Rotten Tomatoes audience rating might have a slight effect on a film's earnings. The correlation between Rotten Tomatoes Audience Ratings and earnings does increase over time, and as shown in the previous scatterplots the correlation is higher than that of the Rotten Tomatoes Rating.

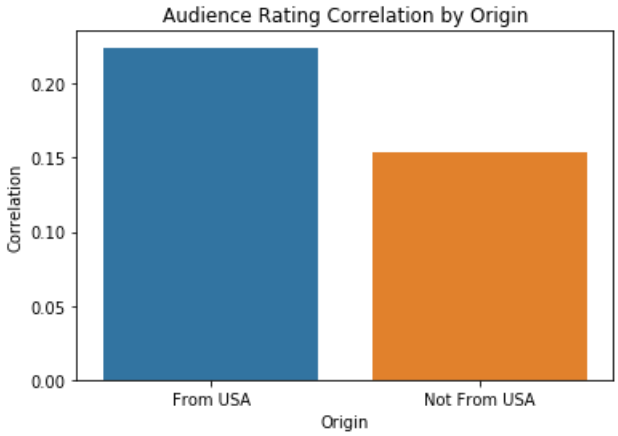
Now, it is time to take a look at how the correlation between Rotten Tomatoes Rating, Rotten Tomatoes Audience Rating, and earnings changes depending on whether or not a film was created in the US. It would make sense that the correlation between them would increase for films created in the US if a film's Rotten Tomatoes Rating did impact its box office performance

In [13]:

```
# getting correlation values based on country of origin
from_USA_t_corr = scipy.stats.pearsonr(movies_df[movies_df.from_USA].tomatometer_rating,
                                       movies_df[movies_df.from_USA].earnings)[0]
not_USA_t_corr = scipy.stats.pearsonr(movies_df[movies_df.from_USA == False].tomatometer_rating,
                                      movies_df[movies_df.from_USA == False].earnings)[0]
from_USA_a_corr = scipy.stats.pearsonr(movies_df[movies_df.from_USA].audience_rating,
                                       movies_df[movies_df.from_USA].earnings)[0]
not_USA_a_corr = scipy.stats.pearsonr(movies_df[movies_df.from_USA == False].audience_rating,
                                      movies_df[movies_df.from_USA == False].earnings)[0]

# plotting correlation based on country of origin
sns.barplot(x = ["From USA", "Not From USA"], y = [from_USA_t_corr, not_USA_t_corr])
plt.xlabel("Origin")
plt.ylabel("Correlation")
plt.title("Rotten Tomatoes Rating Correlation by Origin")
plt.show()

sns.barplot(x = ["From USA", "Not From USA"], y = [from_USA_t_corr, not_USA_t_corr])
plt.xlabel("Origin")
plt.ylabel("Correlation")
plt.title("Audience Rating Correlation by Origin")
```

```
plt.show()
```





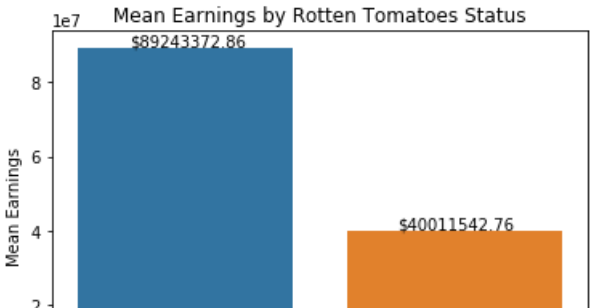
Based on these barplots, there is a higher correlation between Rotten Tomatoes Rating and earnings and between Rotten Tomatoes Audience Rating and earnings for countries from the USA than countries not from the USA.

This would support the idea of the Rotten Tomatoes Effect, as films from the USA would likely have a larger correlation if the Rotten Tomatoes Effect were true as the Rotten Tomatoes site is based in the USA. However, the correlation with earnings is still relatively low, and there could be many other reasons for this trend.

It's now time to take a look at whether or not films with heigher Rotten Tomatoes Ratings have significantly higher earnings than movies with lower Rotten Tomatoes Ratings. Conveniently, the Rotten Tomatoes Status can be used to divide movies into those with low scores with a Rotten status, and those with high ratings with a Fresh or Certified Fresh status. Here, movies with a Fresh or Certified Fresh status will be considered together as Fresh. Here are the mean earnings of the two.

In [14]:

```
# calculating and plotting mean earnings based on Fresh or Rotten
e_mean_f = np.mean(movies_df[movies_df.tomatometer_status != ("Rotten")].earnings)
e_mean_r = np.mean(movies_df[movies_df.tomatometer_status == ("Rotten")].earnings)

sns.barplot(x = ["Fresh", "Rotten"], y = [e_mean_f, e_mean_r])
plt.text(x = -.21, y = e_mean_f + 550000, s = "$" + str(round(e_mean_f, 2)))
plt.text(x = .79, y = e_mean_r + 550000, s = "$" + str(round(e_mean_r, 2)))
plt.xlabel("Rotten Tomatoes Status")
plt.ylabel("Mean Earnings")
plt.title("Mean Earnings by Rotten Tomatoes Status")
plt.show()

e_diff_means = e_mean_f - e_mean_r
print("Difference in Means: $" + str(round(e_diff_means, 2)))
```



Difference in Means: $49231830.11

From these barplots, it can be seen that films given a Fresh or Certified Fresh status do indeed on average have higher earnings than films with a Rotten status. However, it is not clear yet whether or not this difference is significant. In order to determine if this difference is significant, a hypothesis test will be performed.

First, the experimental difference in means between fresh and rotten movies has already been calculated. Then, a random permutation of the earnings data will be generated. Then, the earnings values will be assigned fresh or rotten to have the same number of each as the original data. Then, the difference in means will be calculated. These three steps will occur many times until a distribution can be created of the difference in means. Finally, a p-value can then be calculated from the distribution and the experimental difference in means.

H0: The mean earnings of Fresh movies is not higher than the mean earnings of Rotten movies

Alpha: .05

In [15]:

```
# setting a random seed so that this is repeatable. I'll be using 18 for my current age
np.random.seed(18)

# generating distribution of difference in means
num_f = np.sum(movies_df.tomatometer_status != "Rotten")
diff_means = []
for i in range(10000) :
    rand_permutation = np.random.permutation(movies_df.earnings)
    perm_sample_f = rand_permutation[0:num_f]
    perm_sample_r = rand_permutation[num_f:]
    diff_means.append(np.mean(perm_sample_f) - np.mean(perm_sample_r))

# displaying histogram of distribution of difference in means with experimental difference in means
sns.distplot(diff_means, kde = False, bins = 10, color = "rebeccapurple")
plt.ylabel("")
plt.xlabel("Difference in Means")
plt.title("Distribution of Difference in Means")
plt.plot([e_diff_means, e_diff_means], [0, 3000])
plt.show()

# calculating and printing p-value
p_val = np.sum(np.array(diff_means) >= e_diff_means) / len(diff_means)
print("P-Value: " + str(p_val))
```

P-Value: 0.0

Here is the distribution of difference in means generated by creating random permutations of the earnings data, assigning the rotten and fresh labels, and calculating the difference in means plotted with the experimental difference in means. The p-value is approximately 0, which is less than our alpha value of .05. Therefore, there we reject the null hypothesis. There is sufficient evidence to suggest that the mean earnings of Fresh movies is higher than the mean earnings of Rotten Movies.

From this analysis, it seems likely that the Rotten Tomatoes Effect has been exagerrated and that a movie's Rotten Tomatoes Rating has little effect on its earnings. Specifically, the low correlation between Rotten Tomatoes Rating, Rotten Tomatoes Audience Rating, and earnings and the lack of increase in correlation over time between Rotten Tomatoes Rating and earnings suggest that a movie's rating and status does not affect its earnings. The higher correlation and increase in correlation over time for the Rotten Tomatoes Audience Rating suggests that, if the Rotten Tomatoes Effect

were true, the Rotten Tomatoes Audience Rating is probably the aspect of Rotten Tomatoes that affects a movie's earnings.

However, the small correlation that does exist and the fact that movies with a "Fresh" status have a significantly higher mean earnings than movies with a "Rotten" status suggest that a movie's Rotten Tomatoes Rating, Rotten Tomatoes Audience Rating, and Rotten Tomatoes Status might be useful in predicting a movie's earnings even if they do not actually have a large affect on the earnings.

Multiple Linear Regression

The goal here is to create a multiple linear regression model to predict the earnings of a movie. Through the exploratory data analysis, it seems that a movie's budget, Rotten Tomatoes Rating, Rotten Tomatoes Audience Rating, Rotten Tomatoes Status, Year, and Budget can be useful in predicting a movie's earnings. First, arrays for the features and response variable will be created.

In [16]:

```
# creating response variable array
y = movies_df["earnings"].values

# creating features array/ preprocessing
movies_df_copy = movies_df
movies_df_copy["from_USA"] = movies_df_copy.from_USA.astype(int)
movies_df_copy = movies_df_copy.drop(["title", "country", "income", "earnings"], axis = 1)
movies_df_copy = pd.get_dummies(movies_df_copy)
print(list(movies_df_copy.columns))
X = movies_df_copy.values
```

['year', 'budget', 'tomatometer_rating', 'audience_rating', 'from_USA', 'tomatometer_status_Certified Fresh', 'tomatometer_status_Fresh', 'tomatometer_status_Rotten']

First, all the possible features that are available, listed above, will be used. A certain portion of the available data will be set aside to be used as training data, with the rest to be used to test the final model at the end. A 5-fold Cross Validation will be done using the training data, and the mean of the five R-Squared values will be calculated.

In [17]:

```
# creating training and testing split in data
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y, test_size = .3, random_state = 18)
model_first = LinearRegression()
cv_scores = sklearn.model_selection.cross_val_score(model_first, X_train, y_train, cv = 5)
print("Mean 5-Fold CV Score: " + str(np.mean(cv_scores)))
```

Mean 5-Fold CV Score: 0.46933541920250377

Now, new models will be created, getting rid of one of the features each time. The new Mean 5-Fold CV Score will be calculated each time, and if it is not noticeably less than that of the original model, then that feature is not necessary for the final model and can be eliminated.

In [18]:

```
# creating model with no year information
no_year = movies_df_copy.drop("year", axis = 1)
X_ny = no_year.values
X_train_ny, X_test_ny, y_train_ny, y_test_ny = sklearn.model_selection.train_test_split(X_ny,
                                                    y,
                                                    test_size = .3,
                                                    random_state = 18)
model_ny = LinearRegression()
cv_scores_ny = sklearn.model_selection.cross_val_score(model_ny, X_train_ny, y_train_ny, cv = 5)
print("Mean 5-Fold CV Score With No Year: " + str(np.mean(cv_scores_ny)))

# creating model with no budget information
no_budget = movies_df_copy.drop("budget", axis = 1)
X_nb = no_budget.values
X_train_nb, X_test_nb, y_train_nb, y_test_nb = sklearn.model_selection.train_test_split(X_nb,
                                                    y,
                                                    test_size = .3,
                                                    random_state = 18)
model_nb = LinearRegression()
cv_scores_nb = sklearn.model_selection.cross_val_score(model_nb, X_train_nb, y_train_nb, cv = 5)
print("Mean 5-Fold CV Score With No Budget: " + str(np.mean(cv_scores_nb)))

# creating model with no tomatometer rating information
no_tr = movies_df_copy.drop("tomatometer_rating", axis = 1)
X_ntr = no_tr.values
X_train_ntr, X_test_ntr, y_train_ntr, y_test_ntr = sklearn.model_selection.train_test_split(X_ntr,
                                                    y,
```

```
                                                    test_size = .3,
                                                    random_state = 18)
model_ntr = LinearRegression()
cv_scores_ntr = sklearn.model_selection.cross_val_score(model_ntr, X_train_ntr, y_train_ntr, cv = 5)
print("Mean 5-Fold CV Score With No Tomatometer Rating: " + str(np.mean(cv_scores_ntr)))

# creating model with no audience rating information
no_ar = movies_df_copy.drop("audience_rating", axis = 1)
X_nar = no_ar.values
X_train_nar, X_test_nar, y_train_nar, y_test_nar = sklearn.model_selection.train_test_split(X_nar,
                                                    y,
                                                    test_size = .3,
                                                    random_state = 18)
model_nar = LinearRegression()
cv_scores_nar = sklearn.model_selection.cross_val_score(model_nar, X_train_nar, y_train_nar, cv = 5)
print("Mean 5-Fold CV Score With No Audience Rating: " + str(np.mean(cv_scores_nar)))

# creating model with no origin country information
no_country = movies_df_copy.drop("from_USA", axis = 1)
X_nc = no_country.values
X_train_nc, X_test_nc, y_train_nc, y_test_nc = sklearn.model_selection.train_test_split(X_nc,
                                                  y,
                                                  test_size = .3,
                                                  random_state = 18)
model_nc = LinearRegression()
cv_scores_nc = sklearn.model_selection.cross_val_score(model_nc, X_train_nc, y_train_nc, cv = 5)
print("Mean 5-Fold CV Score With No Origin Country: " + str(np.mean(cv_scores_nc)))

# creating model with no tomatometer status information
no_ts = movies_df_copy.drop(["tomatometer_status_Certified Fresh",
                             "tomatometer_status_Fresh",
                             "tomatometer_status_Rotten"], axis = 1)
X_nts = no_ts.values
X_train_nts, X_test_nts, y_train_nts, y_test_nts = sklearn.model_selection.train_test_split(X_nts,
                                                    y,
                                                    test_size = .3,
                                                    random_state = 18)
model_nts = LinearRegression()
cv_scores_nts = sklearn.model_selection.cross_val_score(model_nts, X_train_nts, y_train_nts, cv = 5)
print("Mean 5-Fold CV Score With No Tomatometer Status: " + str(np.mean(cv_scores_nts)))
```

```
Mean 5-Fold CV Score With No Year: 0.4572133387882439
Mean 5-Fold CV Score With No Budget: 0.10271412393164799
Mean 5-Fold CV Score With No Tomatometer Rating: 0.46926761216676144
Mean 5-Fold CV Score With No Audience Rating: 0.4605566195661944
Mean 5-Fold CV Score With No Origin Country: 0.46832138561456915
Mean 5-Fold CV Score With No Tomatometer Status: 0.46663240729610206
```

From these Mean 5-Fold CV Scores, each without one of the original features, it can be seen that removing the Tomatometer Rating and Origin Country features does not really lower the CV Scores. Therefore, these features should be able to be reviewed without significantly impacting the performance of the model. It should also be noted that removing the Audience Rating and Tomatometer Status lowers the CV scores by very little, and that removing the budget lowers the CV scores by the most.

In [19]:

```
# creating the final model
final_data = movies_df_copy.drop(["tomatometer_rating", "from_USA"], axis = 1)
X_final = final_data.values
X_train_final, X_test_final, y_train_final, y_test_final = sklearn.model_selection.train_test_split(X_final,
                                                    y,
                                                    test_size = .3,
                                                    random_state = 18)
final_model = LinearRegression()
final_cv_scores = sklearn.model_selection.cross_val_score(final_model, X_train_final, y_train_final, cv = 5)
print("Final Mean 5-Fold CV Score: " + str(np.mean(final_cv_scores)))
final_model.fit(X_train_final, y_train_final)
print("R^2 of test set: " + str(final_model.score(X_test_final, y_test_final)))
print("Coefficients of the model: " + str(final_model.coef_))
print("Features: " + str(list(final_data.columns)))
```

```
Final Mean 5-Fold CV Score: 0.46840956413705614
R^2 of test set: 0.4551403317071834
Coefficients of the model: [ 3.12865839e+06  2.19701899e+00  1.12201560e+06  2.17245312e+07
 -1.06878782e+07 -1.10366530e+07]
Features: ['year', 'budget', 'audience_rating', 'tomatometer_status_Certified Fresh', 'tomatometer_status_Fresh', 'tomatometer_status_Rotten']
```

This is the final multiple linear regression model based on a movie's year, budget, Rotten Tomatoes Audience Rating, and Rotten Tomatoes Status

to predict a movie's earnings. The coefficients for these features of the model are listed above, and the model's accuracy on the test set, data that it had never seen before, had an R^2 value of .45514. While this level of accuracy is not perfect, it is not bad for predicting human actions. This model relies mostly on the movie's budget to predict its earnings, and relies least on the Rotten Tomatoes Audience Rating and Rotten Tomatoes Status

Concluding Thoughts

Through this data analysis and multiple linear regression modeling, a lot of information has been gained regarding the Rotten Tomatoes Effect. To reiterate, the Rotten Tomatoes Effect is the idea that, because of the increasing popularity of the Rotten Tomatoes movie review website, the ratings on the website are affecting the box office earnings of movies. While some studies have been done on this subject, they were done using surveys of the public and not using data. This data analysis looked at budget, income, year, and country data from IMDb and Rotten Tomatoes Rating, Audience Rating, and Status data from Rotten Tomatoes.

Analyzing the data, it was found that, movies with a "Fresh" or "Certified Fresh" status do have significantly higher earnings than movies with a "Rotten" status. However, there is not enough correlation between Rotten Tomatoes Ratings, Audience Ratings, and earnings and the correlation does not increase enough over time to suggest that they are affecting movie earnings by a notable degree.

Using multiple linear regression, a model was created to predict a movie's earning using the movie's year, budget, Rotten Tomatoes Audience Rating, and Rotten Tomatoes Status. The model is not perfectly accurate, performing with an R^2 value of .45514 on , but it still predicts earnings with a reasonable accuracy. The model itself also supports the findings of the data analysis- because the Rotten Tomatoes Ratings data had almost no impact on the accuracy of the data and the Rotten Tomatoes Audience Ratings and Status data had only a small impact, it is unlikely that they are affecting movie earnings by a notable degree.

The findings and model give a lot of insight into the Rotten Tomatoes Effect and predicting of earnings for the film industry in general. The Rotten Tomatoes Effect itself has likely been largely exagerrated- the evidence shows that Rotten Tomatoes Ratings, Audience Ratings, and Status are not affecting film earnings. The model itself can be used to help movie studios predict the earnings of their films with greater accuracy, and the findings can be extrapolated to help movie streaming services decide which movies will be profitable if put on their site. The combination of the findings and the model could also help these companies refine current models in place to predict film earnings and profitability- namely, they help show that reviews on sites like Rotten Tomatoes are not good predictors on their own.

Further research into this topic could include analyzing the prominence of different film review sites based on how well their ratings correlate earnings. Namely, the correlation between ratings and earnings could be analyzed for a multitude of sites, such as Rotten Tomatoes, IMDb, Flixster, etc. could be analyzed to determine which site's rating has the most affect on earnings. Also, further research can be done to see what other features may be important in predicting film earnings, such as the month of release of the film, the amount of money spent on marketing vs. other expenses, etc.