Final Notebook - AI Academy Capstone

August 2, 2022

1 Final Notebook

1.1 Table of Contents

1.2 Introduction

This cell will be used to give a little intro.

Package Importing

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as st
import warnings
warnings.filterwarnings('ignore')
import statsmodels.api as sm
import sqlite3
```

2 Data Importing

[2]:
```
conn = sqlite3.connect('zippedData/im.db')
imdb_movie_basics = pd.read_sql_query("SELECT * FROM movie_basics", conn)
imdb_movie_ratings = pd.read_sql_query("SELECT * FROM movie_ratings", conn)
bom_movie_gross = pd.read_csv('zippedData/bom.movie_gross.csv.gz')
tmdb_movies = pd.read_csv('zippedData/tmdb.movies.csv.gz')
tmdb_movies = tmdb_movies.drop('Unnamed: 0', axis = 1)                    ␣
 ↪#remove unneccesary column
tn_movies_budgets = pd.read_csv('zippedData/tn.movie_budgets.csv.gz')
tn_movies_budget = pd.read_csv('zippedData/tn.movie_budgets.csv.gz')
rt_movies_info = pd.read_csv('zippedData/rt.movie_info.tsv.gz', sep = '\t')
rt_movies_reviews = pd.read_csv('zippedData/rt.reviews.tsv.gz', sep = '\t',␣
 ↪encoding='latin')
```

3 Data Cleaning

3.0.1 Movie Gross Data Cleaning

In this section we are cleaning the data to better analyze patterns between specific movies over the past decade. After checking for the null values in the dataset, we decided that because foreign gross had a substantial amount of missing data, we would only use domestic gross.

[3]:
```
movie_gross = bom_movie_gross

#scale domestic and foreing gross by a million
movie_gross[['domestic_gross']] = movie_gross[['domestic_gross']].mul(1/1000000)

#2010
movie_gross2010 = movie_gross[(movie_gross.year == 2010)]
movie_gross2010 = movie_gross2010.sort_values(by='domestic_gross',␣
 ↪ascending=False).head(10)
#2011
movie_gross2011 = movie_gross[(movie_gross.year == 2011)]
```

```
movie_gross2011 = movie_gross2011.sort_values(by='domestic_gross',␣
 ↪ascending=False).head(10)
#2012
movie_gross2012 = movie_gross[(movie_gross.year == 2012)]
movie_gross2012 = movie_gross2012.sort_values(by='domestic_gross',␣
 ↪ascending=False).head(10)
#2013
movie_gross2013 = movie_gross[(movie_gross.year == 2013)]
movie_gross2013 = movie_gross2013.sort_values(by='domestic_gross',␣
 ↪ascending=False).head(10)
#2014
movie_gross2014 = movie_gross[(movie_gross.year == 2014)]
movie_gross2014 = movie_gross2014.sort_values(by='domestic_gross',␣
 ↪ascending=False).head(10)
#2015
movie_gross2015 = movie_gross[(movie_gross.year == 2015)]
movie_gross2015 = movie_gross2015.sort_values(by='domestic_gross',␣
 ↪ascending=False).head(10)
#2016
movie_gross2016 = movie_gross[(movie_gross.year == 2016)]
movie_gross2016 = movie_gross2016.sort_values(by='domestic_gross',␣
 ↪ascending=False).head(10)
#2017
movie_gross2017 = movie_gross[(movie_gross.year == 2017)]
movie_gross2017 = movie_gross2017.sort_values(by='domestic_gross',␣
 ↪ascending=False).head(10)
#2018
movie_gross2018 = movie_gross[(movie_gross.year == 2018)]
movie_gross2018 = movie_gross2018.sort_values(by='domestic_gross',␣
 ↪ascending=False).head(10)
```

3.0.2 Popularity Data Cleaning

Creating a top 10 dataframe using the movies column dataframe.

[4]:
```
#Top 10
movies = tmdb_movies
popular = movies.sort_values(by='popularity', ascending=False).head(10)
```

3.0.3 Genre Data Cleaning

In this section we are cleaning the imdb_movie_basics dataframe in order to find the most popular genre

[5]:
```
#find all of the different unique genres
unique_genres = imdb_movie_basics.genres.str.split(pat = ",", expand = True)
unique_genres = unique_genres.melt()
```

```
unique_genres = unique_genres.value.unique()


# create a dataframe of all of the genres with repeating movies to represent␣
 ↪movies as multiple genres

#define function that isolates all the movies that are a genre, then create a␣
 ↪new column with the genre name
def genre_dataframe(genre):
    genre_dataframe = imdb_movie_basics[imdb_movie_basics['genres'].str.
 ↪contains(genre) == True].copy()
    genre_dataframe['genre'] = genre
    return genre_dataframe

#empty dataframe
genres = pd.DataFrame()


#for loop that creates dataframes of all the movies that are each genre and␣
 ↪concatenates them together, with repeating movies
for genre in unique_genres:
    genre1 = genre_dataframe(str(genre))
    genres = pd.concat([genres, genre1], axis = 0)

genres.reset_index(inplace = True)

# select specific columns and the join the movie_gross and genres dataframes

genres = genres[['primary_title', 'genre']]

domestic_gross = movie_gross[['title','domestic_gross', 'year']]

genres_gross = genres.merge(domestic_gross, how = 'left', left_on =␣
 ↪'primary_title', right_on = 'title')

#drop null values and isolate only the needed columns
genres_gross = genres_gross.dropna()
genres_gross = genres_gross[['genre', 'domestic_gross', 'year']]

#find the genres that have the most domestic gross revenue over the entire time␣
 ↪period
top_genres = genres_gross[['genre', 'domestic_gross']].groupby(['genre']).mean().
 ↪reset_index()
top_genres =top_genres.sort_values(by = 'domestic_gross', ascending = False)
```

```
#group by genre and year, used mean to control for number of movies released,␣
 ↪could also use sum
new_genres_gross = genres_gross.groupby(['genre', 'year'], as_index =␣
 ↪False)['domestic_gross'].mean()

#create function to plot the genres and their domestic gross revenue
def plot_genres(name):
    all_genres = new_genres_gross[new_genres_gross['genre'] == name]
    if name == 'Adventure':
        color = 'forestgreen'
    elif name == 'Action':
        color = 'springgreen'
    elif name == 'Comedy':
        color = 'slategrey'
    elif name == 'Sci-Fi':
        color = 'dodgerblue'
    elif name == 'Animation':
        color = 'navy'
    elif name == 'Fantasy':
        color = 'lightskyblue'

    plt.plot(all_genres.year, all_genres.domestic_gross, label = name, color =␣
 ↪color, linewidth = 2.5)

#create list of top grossing genres to plot
top_genres_list = ['Adventure', 'Action', 'Comedy', 'Sci-Fi', 'Animation',␣
 ↪'Fantasy']
```

3.0.4 Monthly Analysis Data Cleaning

[6]:
```
#we are using movie budgets from TheNumbers

#convert the dollar values for the budgets into floats
tn_movies_budgets[['production_budget', 'domestic_gross', 'worldwide_gross']] =␣
 ↪tn_movies_budgets[['production_budget', 'domestic_gross', 'worldwide_gross']].
 ↪replace("[$,]", "", regex=True).astype(float)

#scale the budget and gross by dividing by a million
tn_movies_budgets[['production_budget', 'domestic_gross', 'worldwide_gross']] =␣
 ↪tn_movies_budgets[['production_budget', 'domestic_gross', 'worldwide_gross']].
 ↪mul(1/1000000)

#make release date into datetime, then create variable called release_month
tn_movies_budgets['release_date'] = pd.
 ↪to_datetime(tn_movies_budgets['release_date'])
tn_movies_budgets['release_month'] = tn_movies_budgets['release_date'].dt.month
```

The movie budgets table from The Numbers source required a lot of data cleaning. The first step was to convert the dollar values in the dataframe into floats from strings. This allowed the plot to recognize them as quantitative variables. The next step was to convert release date into the desired datetime format. This allowed the code to decipher months based off the numeric in the month spot of the release date. Thus creating a new variable that would be used for monthly financial classification purposes.

[7]:
```
#create domestic and worldwide profits as the gross minus the budget
tn_movies_budgets['domestic_profit'] = tn_movies_budgets['domestic_gross'] -␣
 ↪tn_movies_budgets['production_budget']
tn_movies_budgets['worldwide_profit'] = tn_movies_budgets['worldwide_gross'] -␣
 ↪tn_movies_budgets['production_budget']
```

This part of the code was done in an effort to create new return on investment variables based off existing data revenue and production costs. The production_budget column was subtracted from both domestic_gross and worldwide_gross. The new variables displayed the return on investment for each movie both domestically and worldwide.

3.0.5 Budget vs. Revenue Data Cleaning

In the following cell, we add new columns named "production_budgetINT", "domestic_grossINT", and "worldwide_grossINT" to the *The Numbers* dataset. These are the columns "production_budget", "domestic_gross", and "worldwide_gross" converted from strings into integers. We then divide all of those columns by a million and remove all movies with a worldwide gross of 0.

[8]:
```
movies_budget = tn_movies_budget

# creates a production column int column
production_budget = []

for budget in movies_budget['production_budget']:
    budget = int(budget[1:].replace(",", ""))
    production_budget.append(budget)

movies_budget['production_budgetINT'] = production_budget

# creates a domestic gross int column
domestic_gross = []

for gross in movies_budget['domestic_gross']:
    gross = int(gross[1:].replace(",", ""))
    domestic_gross.append(gross)

movies_budget['domestic_grossINT'] = domestic_gross

# creates a worldwide gross int column
worldwide_gross = []
```

```
for gross in movies_budget['worldwide_gross']:
    gross = int(gross[1:].replace(",", ""))
    worldwide_gross.append(gross)

movies_budget['worldwide_grossINT'] = worldwide_gross

#removes worldwide budgets of 0
movies_budget = movies_budget[movies_budget.worldwide_grossINT > 0]



#scale by one million
movies_budget[['production_budgetINT', 'worldwide_grossINT',␣
 ↪'domestic_grossINT']] = movies_budget[['production_budgetINT',␣
 ↪'worldwide_grossINT', 'domestic_grossINT']].mul(1/1000000)
```

3.0.6 Top Studios' Budgets Data Cleaning

In the cell below, we inner join the *The Numbers* budget dataframe and the *Box Office Mojo* movie gross dataframe on the movie titles. We name this new dataframe "movie_studio_budget." We then drop the one null value and combine the duplicate Warner Brothers studio observations.

[9]:
```
movie_gross = bom_movie_gross

#inner join two datasets
movie_studio_budget = movies_budget.merge(movie_gross, how = 'inner', left_on =␣
 ↪'movie', right_on = 'title')
movie_studio_budget = movie_studio_budget[['release_date', 'movie',␣
 ↪'studio','production_budgetINT','worldwide_grossINT']]
#make release date datetime
movie_studio_budget['release_date'] = pd.
 ↪to_datetime(movie_studio_budget['release_date'])
#drop one null studio row
movie_studio_budget = movie_studio_budget.dropna()
#make new variable called studio to edit studio names
studios = movie_studio_budget['studio']

#rename studios that are the same with different names in the dataset
studios[studios.str.contains('WB')] = 'WB'


# merge the studio variable back into dataframe
movie_studio_budget['studio'] = studios
```

The following cell create dataframes for the top studios by their average worldwide gross per movie and their total worldwide gross across all of their movies. These dataframes all for us to compare a studios budget and their worldwide gross.

[10]:
```
#isolate top 10 studios by their average movie budget and worldwide gross, sort␣
 ↪by worldwide gross
top_studios = movie_studio_budget[['studio', 'production_budgetINT',␣
 ↪'worldwide_grossINT']].groupby(['studio']).mean()
top_studios = top_studios.sort_values(by = 'worldwide_grossINT', ascending =␣
 ↪False).reset_index().head(10)

#isolate top 10 studios by their total budget and revenues
top_studios_total = movie_studio_budget[['studio', 'production_budgetINT',␣
 ↪'worldwide_grossINT']].groupby(['studio']).sum()
top_studios_total = top_studios_total.sort_values(by = 'worldwide_grossINT',␣
 ↪ascending = False).reset_index().head(10)
```

4 Type of Movie

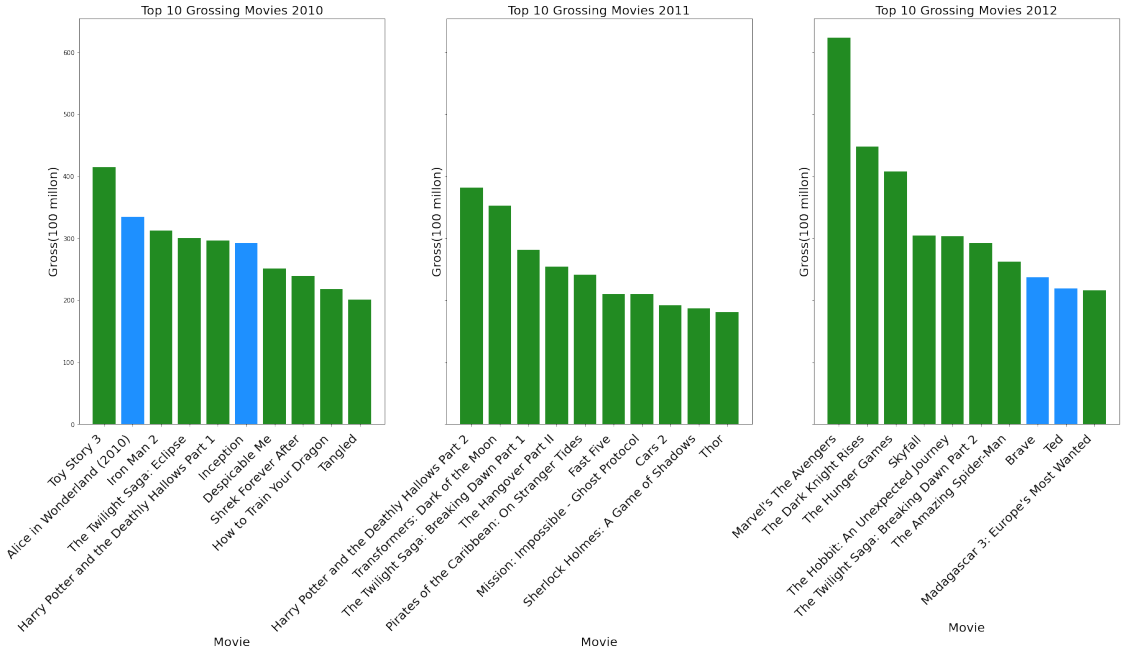
In this section, we will use data visualizations to lead our analysis to our recommendation.

4.1 Top 10 Grossing movies in each year

In this cell we take a graph from each of the groups above and closely analyze them. The green bars signify the movies that belong to a franchise and the blue bars represent the movies that aren't.

[11]:
```
fig, axs = plt.subplots(1, 3, figsize=(30, 12), sharey=True)

## 2010
axs[0].bar(movie_gross2010['title'],
         movie_gross2010['domestic_gross'],
         color=['forestgreen', 'dodgerblue', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen',
                'dodgerblue', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen'],
         align='center')
axs[0].set_title('Top 10 Grossing Movies 2010',fontsize = 20)
axs[0].set_xlabel('Movie',fontsize = 20)
axs[0].set_ylabel('Gross(100 millon)',fontsize = 20)
axs[0].set_xticklabels(movie_gross2010['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[0].tick_params(axis="x", direction="out", pad=10)

## 2011
axs[1].bar(movie_gross2011['title'],
         movie_gross2011['domestic_gross'],
         color=['forestgreen', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen',
```

```
                'forestgreen', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen'],
        align='center')
axs[1].set_title('Top 10 Grossing Movies 2011',fontsize = 20)
axs[1].set_xlabel('Movie',fontsize = 20)
axs[1].set_ylabel('Gross(100 millon)',fontsize = 20)
axs[1].set_xticklabels(movie_gross2011['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[1].tick_params(axis="x", direction="out", pad=10)

## 2012
axs[2].bar(movie_gross2012['title'],
        movie_gross2012['domestic_gross'],
        color=['forestgreen', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen',
                'forestgreen', 'forestgreen', 'dodgerblue', 'dodgerblue',␣
 ↪'forestgreen'],
        align='center')
axs[2].set_title('Top 10 Grossing Movies 2012',fontsize = 20)
axs[2].set_xlabel('Movie',fontsize = 20)
axs[2].set_ylabel('Gross(100 millon)',fontsize = 20)
axs[2].set_xticklabels(movie_gross2012['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[2].tick_params(axis="x", direction="out", pad=10)
```

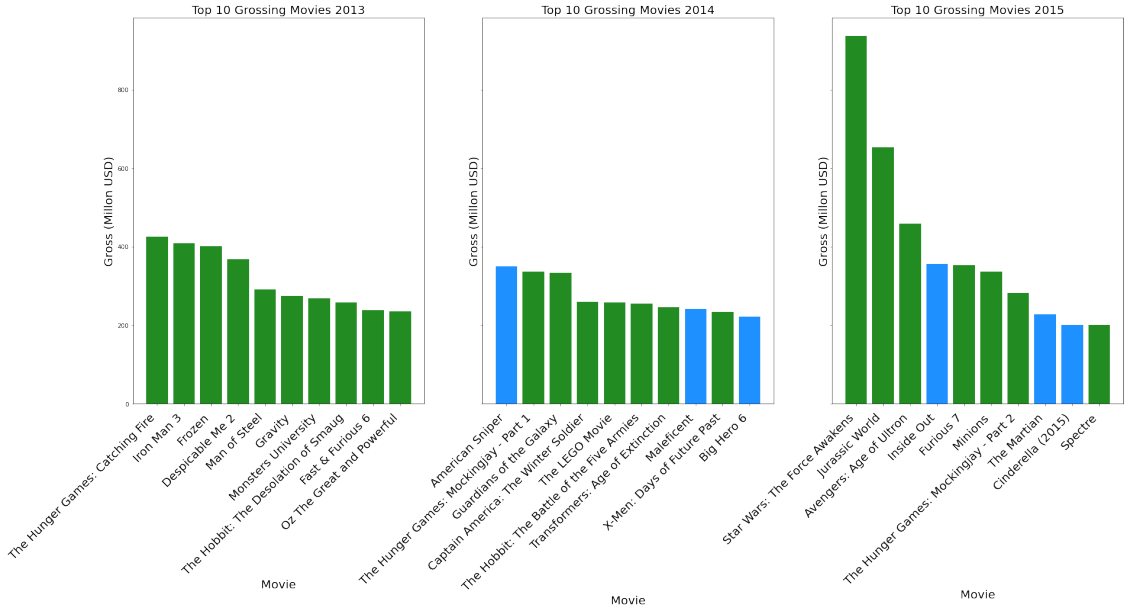


These 3 years show the beginning of a trend. Each year has at least one top grossing superhero

movie and animation. 2010 specifically had 5 movies in the top 10 that grossed over 200 million domestically. The main genre of each movie seems to be action. The superhero movies all and animated movies all seem to belong to a franchise.

[12]:
```
fig, axs = plt.subplots(1, 3, figsize=(30, 12), sharey=True)

## 2013
axs[0].bar(movie_gross2013['title'],
        movie_gross2013['domestic_gross'],
        color=['forestgreen', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen',
               'forestgreen', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen'],
        align='center')
axs[0].set_title('Top 10 Grossing Movies 2013',fontsize = 20)
axs[0].set_xlabel('Movie',fontsize = 20)
axs[0].set_ylabel('Gross (Millon USD)',fontsize = 20)
axs[0].set_xticklabels(movie_gross2013['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[0].tick_params(axis="x", direction="out", pad=10)

## 2014
axs[1].bar(movie_gross2014['title'],
        movie_gross2014['domestic_gross'],
        color=['dodgerblue', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen',
               'forestgreen', 'forestgreen', 'dodgerblue', 'forestgreen',␣
 ↪'dodgerblue'],
        align='center')
axs[1].set_title('Top 10 Grossing Movies 2014',fontsize = 20)
axs[1].set_xlabel('Movie',fontsize = 20)
axs[1].set_ylabel('Gross (Millon USD)',fontsize = 20)
axs[1].set_xticklabels(movie_gross2014['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[1].tick_params(axis="x", direction="out", pad=10)

## 2015
axs[2].bar(movie_gross2015['title'],
        movie_gross2015['domestic_gross'],
        color=['forestgreen', 'forestgreen', 'forestgreen', 'dodgerblue',␣
 ↪'forestgreen',
               'forestgreen', 'forestgreen', 'dodgerblue', 'dodgerblue',␣
 ↪'forestgreen'],
        align='center')
axs[2].set_title('Top 10 Grossing Movies 2015',fontsize = 20)
axs[2].set_xlabel('Movie',fontsize = 20)
axs[2].set_ylabel('Gross (Millon USD)',fontsize = 20)
```

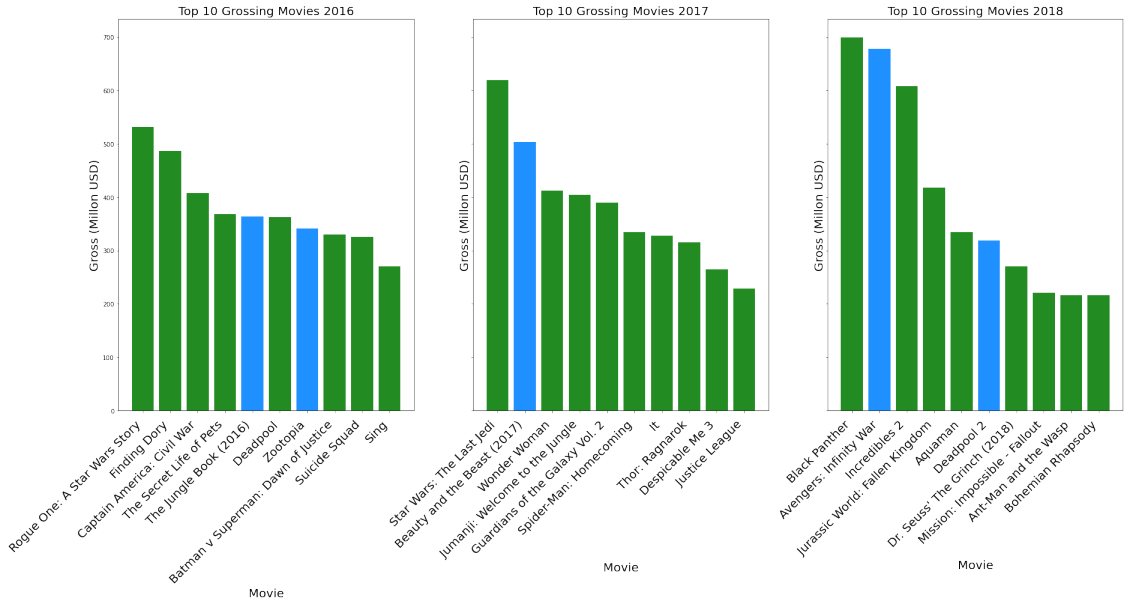
```
axs[2].set_xticklabels(movie_gross2015['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[2].tick_params(axis="x", direction="out", pad=10)
```



These 3 years seem to continue the pattern. Once again each year has a top grossing superhero or animated movie focused on action or comedy. Franchise are being continued and built on.

[13]:
```
fig, axs = plt.subplots(1, 3, figsize=(30, 12), sharey=True)

## 2016
axs[0].bar(movie_gross2016['title'],
        movie_gross2016['domestic_gross'],
        color=['forestgreen', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'dodgerblue',
               'forestgreen', 'dodgerblue', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen'],
        align='center')
axs[0].set_title('Top 10 Grossing Movies 2016',fontsize = 20)
axs[0].set_xlabel('Movie',fontsize = 20)
axs[0].set_ylabel('Gross (Millon USD)',fontsize = 20)
axs[0].set_xticklabels(movie_gross2016['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[0].tick_params(axis="x", direction="out", pad=10)

## 2017
axs[1].bar(movie_gross2017['title'],
        movie_gross2017['domestic_gross'],
```

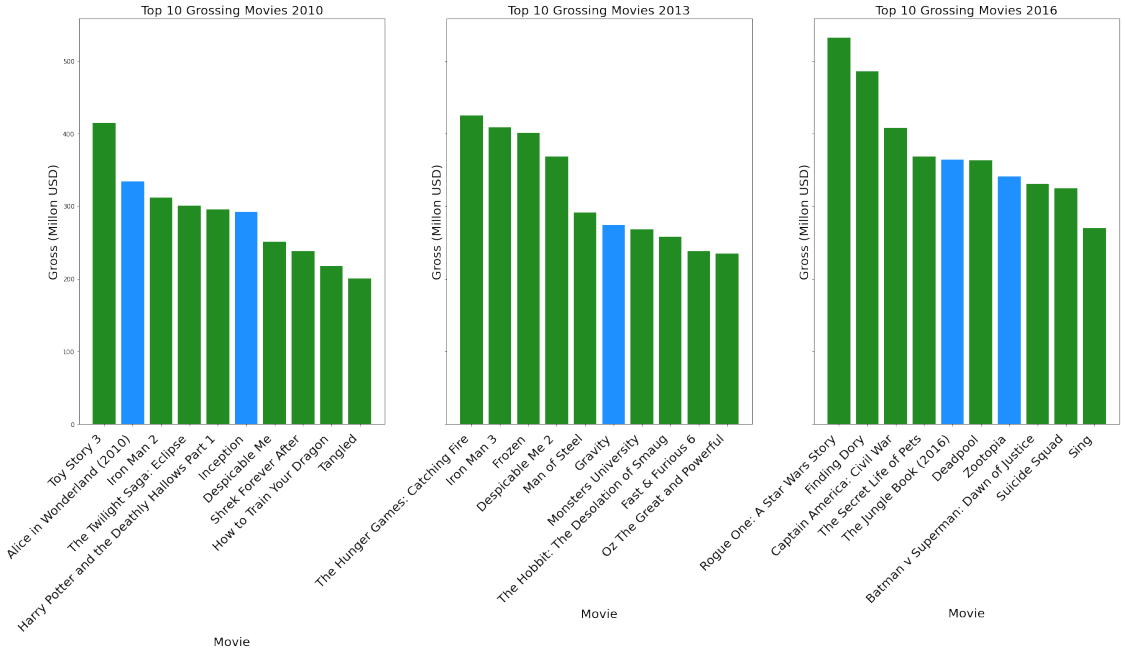
```
        color=['forestgreen', 'dodgerblue', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen',
               'forestgreen', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen'],
        align='center')
axs[1].set_title('Top 10 Grossing Movies 2017',fontsize = 20)
axs[1].set_xlabel('Movie',fontsize = 20)
axs[1].set_ylabel('Gross (Millon USD)',fontsize = 20)
axs[1].set_xticklabels(movie_gross2017['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[1].tick_params(axis="x", direction="out", pad=10)

## 2018
axs[2].bar(movie_gross2018['title'],
        movie_gross2018['domestic_gross'],
        color=['forestgreen', 'dodgerblue', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen',
               'dodgerblue', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen'],
        align='center')
axs[2].set_title('Top 10 Grossing Movies 2018',fontsize = 20)
axs[2].set_xlabel('Movie',fontsize = 20)
axs[2].set_ylabel('Gross (Millon USD)',fontsize = 20)
axs[2].set_xticklabels(movie_gross2018['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[2].tick_params(axis="x", direction="out", pad=10)
```



These final graphs follow the same pattern specifically the years 2017 and 2018 which had 5 of their top 10 movies be superhero focused. Franchises are continued and some reach their climax(Avengers Infinity War)

[14]:
```
fig, axs = plt.subplots(1, 3, figsize=(30, 12), sharey=True)

## 2010
axs[0].bar(movie_gross2010['title'],
        movie_gross2010['domestic_gross'],
        color=['forestgreen', 'dodgerblue', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen',
               'dodgerblue', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen'],
        align='center')
axs[0].set_title('Top 10 Grossing Movies 2010',fontsize = 20)
axs[0].set_xlabel('Movie',fontsize = 20)
axs[0].set_ylabel('Gross (Millon USD)',fontsize = 20)
axs[0].set_xticklabels(movie_gross2010['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[0].tick_params(axis="x", direction="out", pad=10)

## 2013
axs[1].bar(movie_gross2013['title'],
        movie_gross2013['domestic_gross'],
        color=['forestgreen', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen',
               'dodgerblue', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen'],
        align='center')
axs[1].set_title('Top 10 Grossing Movies 2013',fontsize = 20)
axs[1].set_xlabel('Movie',fontsize = 20)
axs[1].set_ylabel('Gross (Millon USD)', fontsize = 20)
axs[1].set_xticklabels(movie_gross2013['title'], rotation=45,  va='top',␣
 ↪ha='right',fontsize = 20)
axs[1].tick_params(axis="x", direction="out", pad=10)

## 2016
axs[2].bar(movie_gross2016['title'],
        movie_gross2016['domestic_gross'],
        color=['forestgreen', 'forestgreen', 'forestgreen', 'forestgreen',␣
 ↪'dodgerblue',
               'forestgreen', 'dodgerblue', 'forestgreen', 'forestgreen',␣
 ↪'forestgreen'],
        align='center')
axs[2].set_title('Top 10 Grossing Movies 2016',fontsize = 20)
axs[2].set_xlabel('Movie',fontsize = 20)
axs[2].set_ylabel('Gross (Millon USD)',fontsize = 20)
```

```
axs[2].set_xticklabels(movie_gross2016['title'], rotation=45,  va='top',␣
 ↪ha='right', fontsize = 20)
axs[2].tick_params(axis="x", direction="out", pad=10)
```



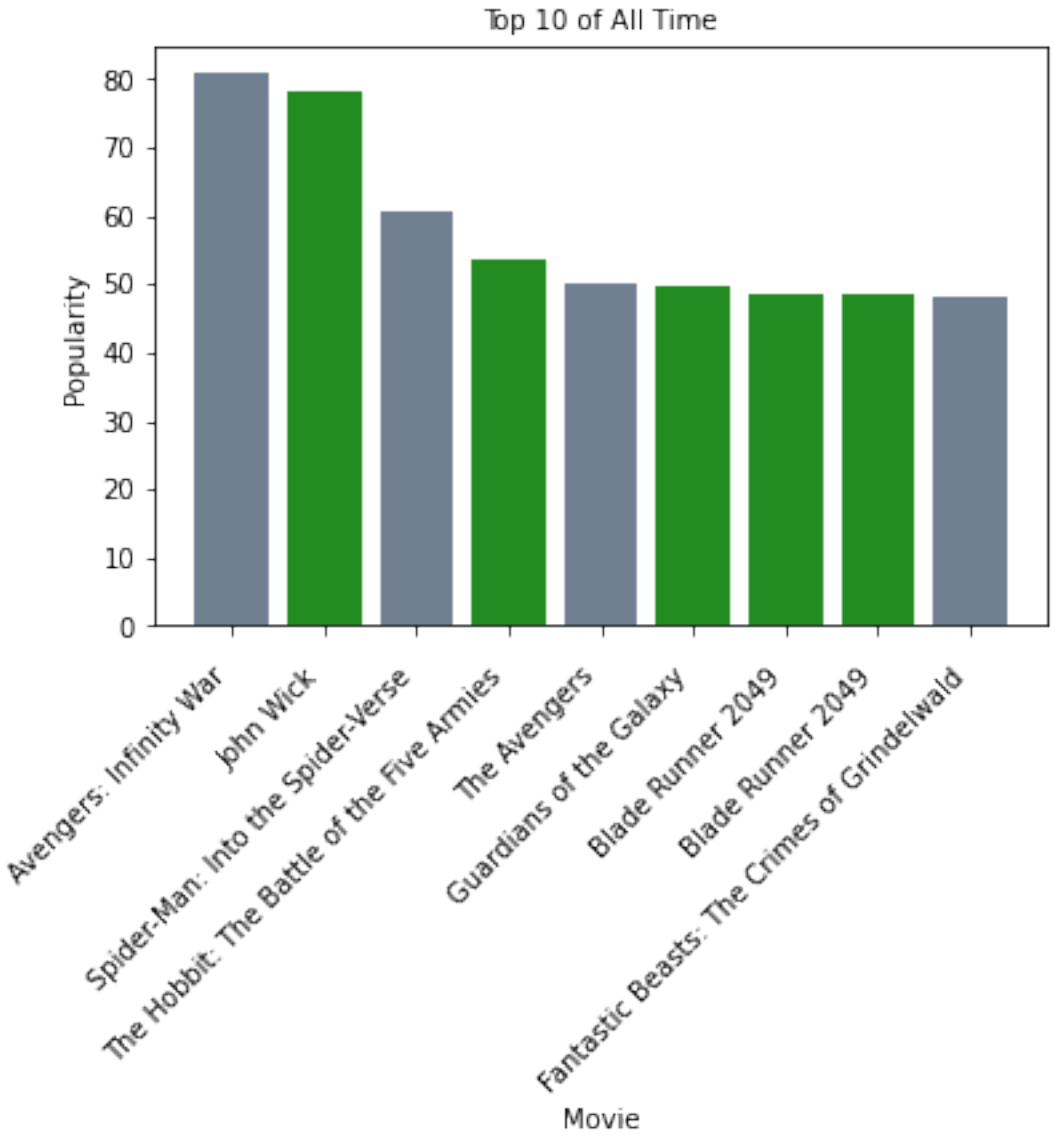
These 3 years show the beginning of a trend. Each year has at least one top grossing superhero movie and animation. 2010 specifically had 5 movies in the top 10 that grossed over 200 million domestically. The main genre of each movie seems to be action. The superhero movies all and animated movies all seem to belong to a franchise.

4.2 Top 10 Popular

In the this cell we see the top 10 movies of all time according to the movies datafarame. All of these movies are action focused and 4 of the 10 are superhero movies.

[15]:
```
fig, ax = plt.subplots()

## TOP 10 of all time
ax.bar(popular['original_title'],
       popular['popularity'],
       color=['slategrey', 'forestgreen', 'slategrey', 'forestgreen',␣
 ↪'slategrey',
              'forestgreen', 'slategrey', 'forestgreen', 'forestgreen',␣
 ↪'slategrey'],
       align='center')
ax.set_title('Top 10 of All Time', fontsize = 10)
ax.set_xlabel('Movie',  fontsize = 10)
```

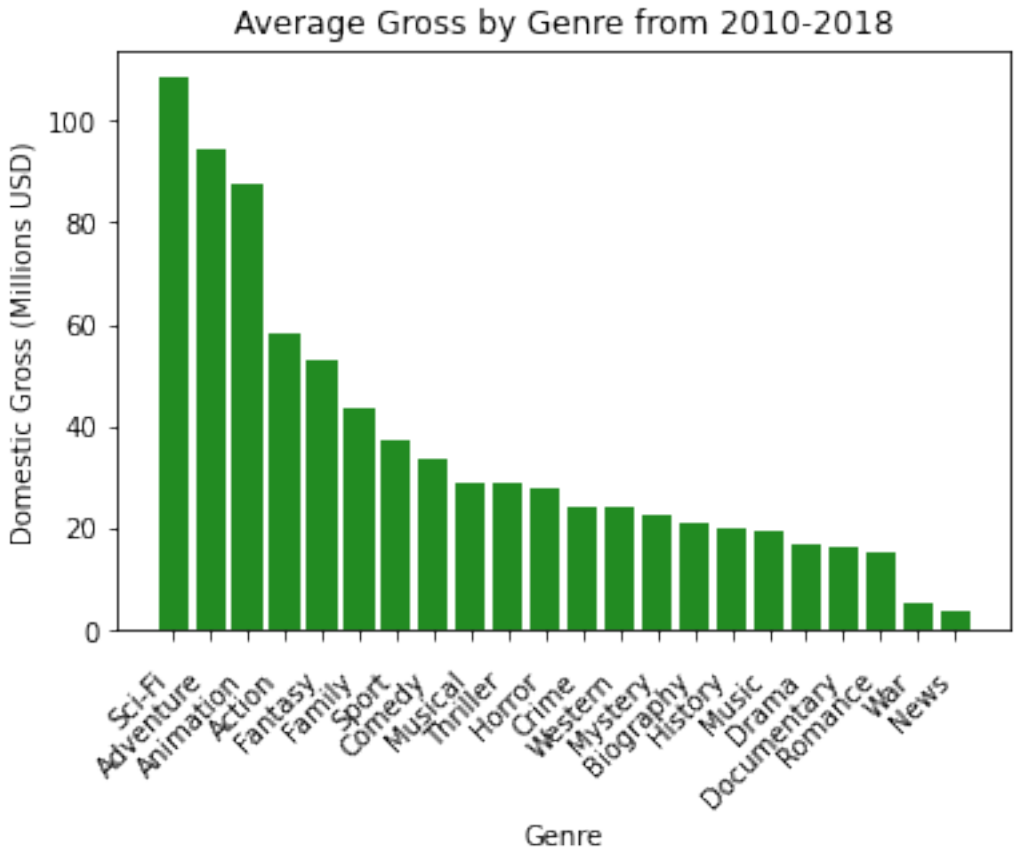
```
ax.set_ylabel('Popularity',  fontsize = 10)
ax.set_xticklabels(popular['original_title'], rotation=45,  va='top',␣
 ↪ha='right', fontsize = 10)
ax.tick_params(axis="x", direction="out", pad=10)
```



4.3 Genres and Domestic Gross

[16]:
```
#bar plot of top movies
fig, axs = plt.subplots()

axs.bar(top_genres['genre'],
```

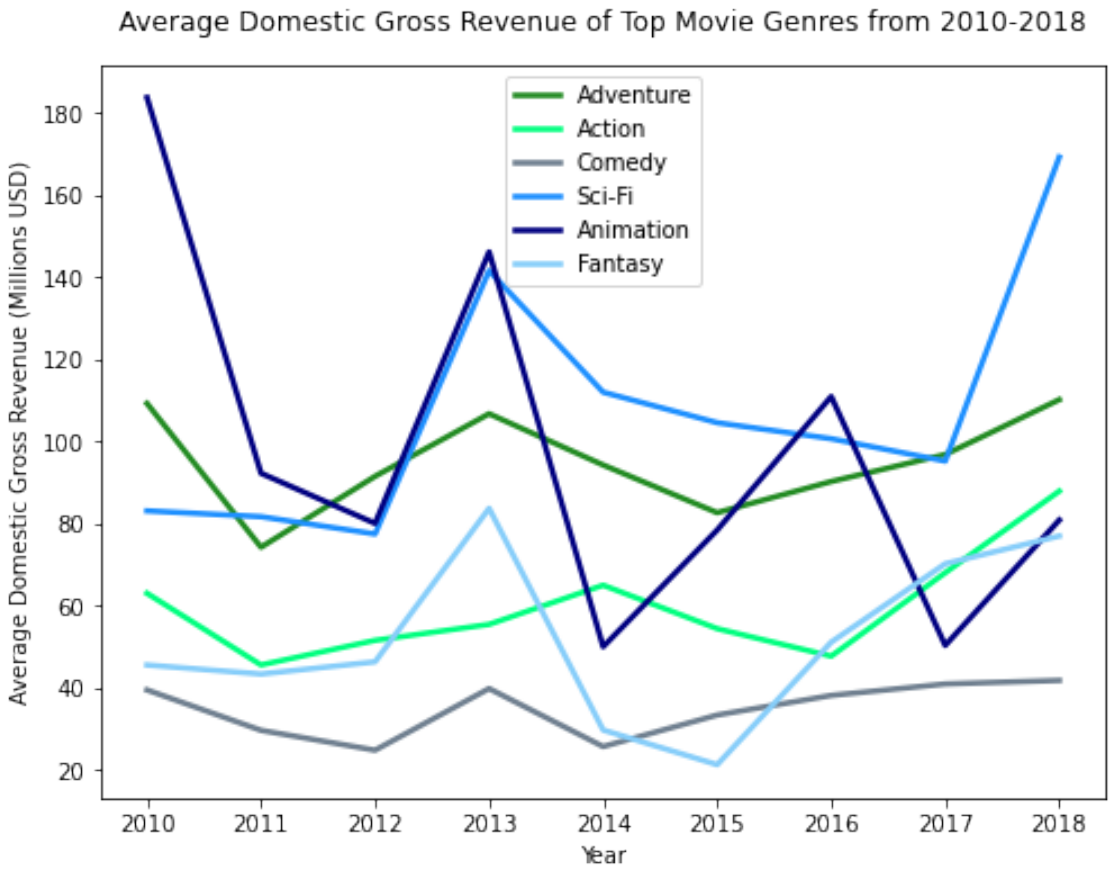
```
        top_genres['domestic_gross'], color ='forestgreen')
axs.set_title('Average Gross by Genre from 2010-2018')
axs.set_xlabel('Genre')
axs.set_ylabel('Domestic Gross (Millions USD)')
axs.set_xticklabels(top_genres['genre'], rotation=45, va='top', ha='right')
axs.tick_params(axis="x", direction="out", pad=10)
```



The graph shows the top grossing genre in order. The top 4 are Sci-FI, Adventure, Animation and Action these results support our previous graphs which showed the top movies of each year.

[17]:
```
#create plot and call function to graph each of the top genres
plt.figure(figsize = (8,6))

for g in top_genres_list:
    plot_genres(g)

plt.title("Average Domestic Gross Revenue of Top Movie Genres from 2010-2018",␣
 ↪pad = 15)
plt.ylabel("Average Domestic Gross Revenue (Millions USD)")
plt.xlabel('Year')
```

```
plt.legend()

plt.show()
```



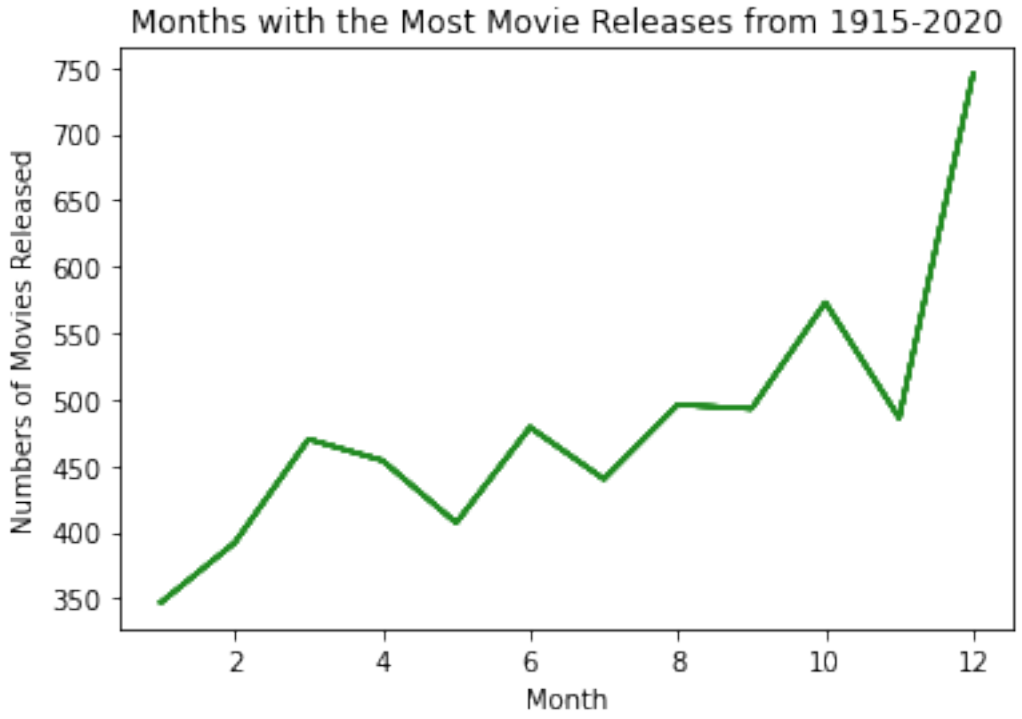
4.4 Recommendation #1

According to this Data a new studio should invest in establishing an action based franchise which would connect all its movies. If possible this franchise should also be superhero based as superhero movies can do a great job at encapsulating all the top genres such as Sci-Fi, Action and Animation.

5 When to Release Movie

5.1 Months With Most Movie Releases from 1915-2020

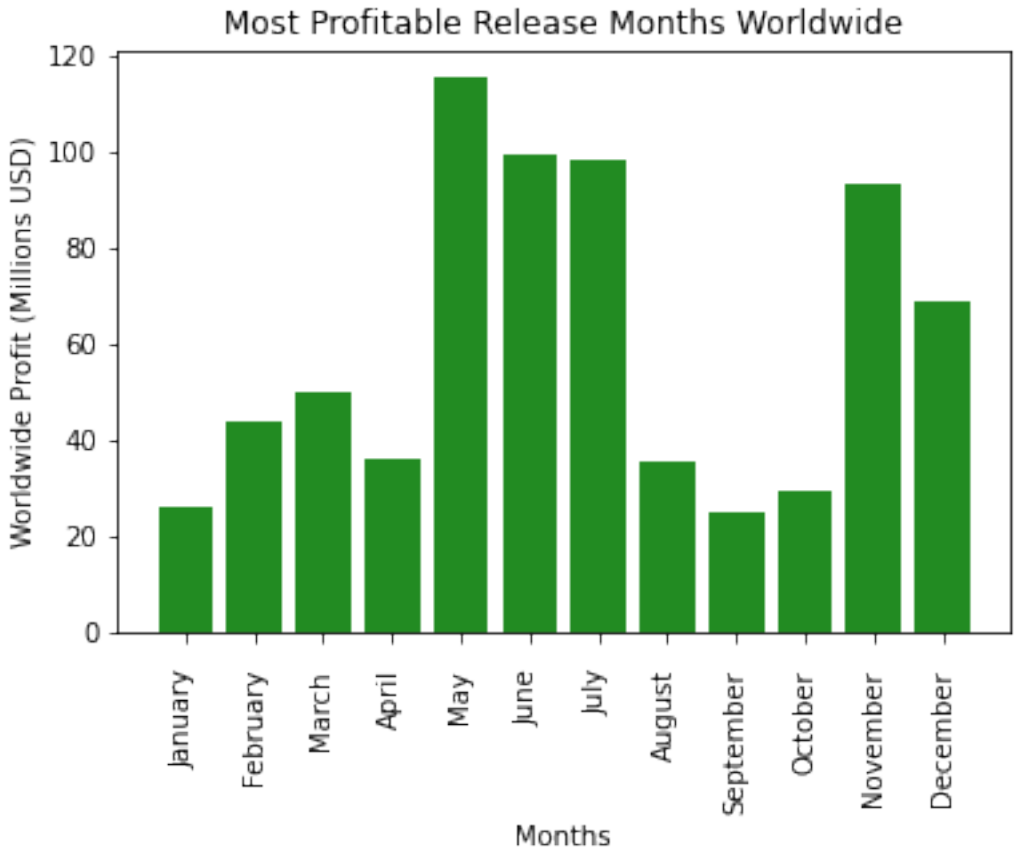
[18]:
```
# months that have the most movie releases, from youtube
months = [month for month, df in tn_movies_budgets.groupby('release_month')]

plt.plot(months, tn_movies_budgets.groupby(['release_month']).count(), color =␣
 ↪'forestgreen')

plt.title("Months with the Most Movie Releases from 1915-2020")
plt.ylabel("Numbers of Movies Released")
plt.xlabel('Month')

plt.show()
```



The plot above shows the correlation between month in a year and the number of movies released. The releases range from 350 to 750 per month from the years 1915 to 2020. The dataset is large enough to indicate industry trends as release months towards the end of the year is more favoured. This is partly due to the consideration of award (Oscars, Emmys, etc.) deadlines. The voting committes prefer to remember and favor movies closer to the deadline which conincides with when voting takes place and consequently when movies are more likely to be released.

5.2 Months with the Most Profit Worldwide

[19]:
```
month_names = ['January', 'February', 'March', 'April', 'May', 'June', 'July',␣
 ↪'August', 'September', 'October', 'November', 'December']

#bar plot of top movies
fig, axs = plt.subplots()

axs.bar(months,
        tn_movies_budgets.groupby(['release_month'])['worldwide_profit'].
 ↪mean(),color = 'forestgreen')
axs.set_title('Most Profitable Release Months Worldwide')
axs.set_xlabel('Months')
axs.set_ylabel('Worldwide Profit (Millions USD)')
axs.set_xticklabels(month_names, rotation=90)
axs.set_xticks(months)
axs.tick_params(axis="x", direction="out", pad=10)
axs.style = "plain"
```

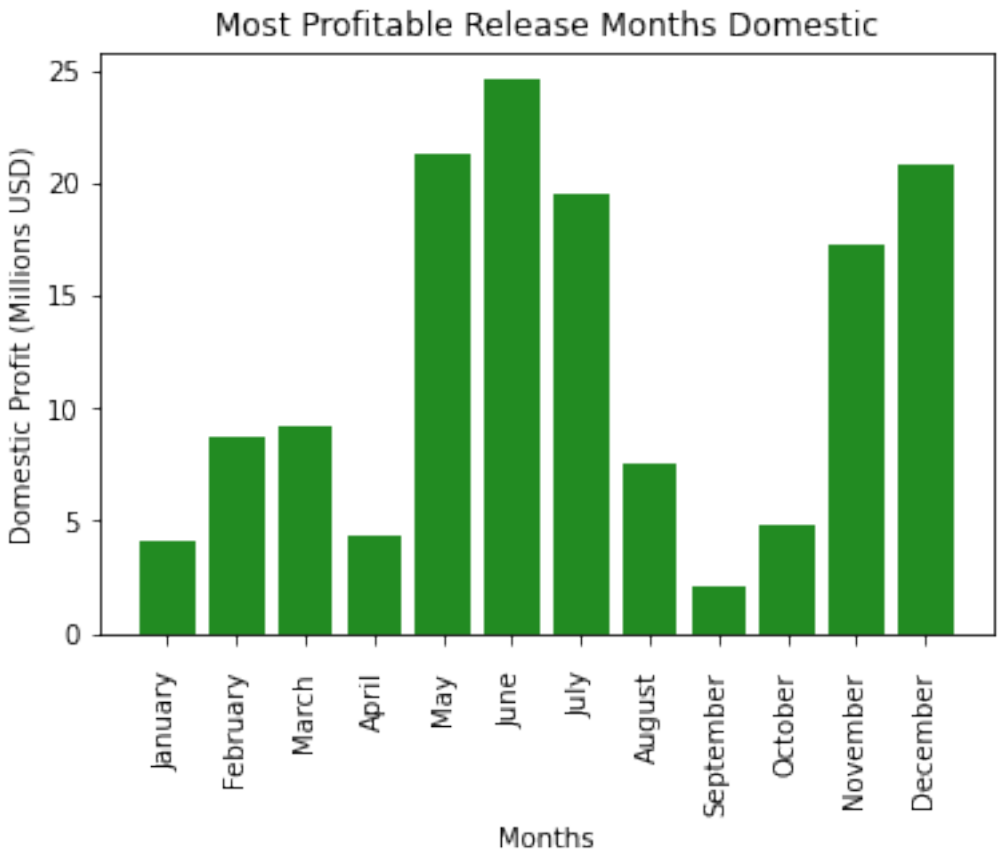


The plot above shows the distibution for worlwide profit (USD in hundreds of millions) for all of

the months from 1915-2020. The x-axis represents the 12 months in a year. The y-axis represents the average profit worldwide for each month based on the films that were released in each of those correlated months. The data bar plot displays a large rise in profit during the months May-July and November-December.

5.3 Months with the Most Profit Domestic

[20]:
```
#some movies were not released domestically, take them out of dataset
tn_movies_budgets_domestic =␣
 ↪tn_movies_budgets[tn_movies_budgets['domestic_gross'] != 0]

#bar plot of top months
fig, axs = plt.subplots()

axs.bar(months,
        tn_movies_budgets_domestic.
 ↪groupby(['release_month'])['domestic_profit'].mean(),color = 'forestgreen')
axs.set_title('Most Profitable Release Months Domestic')
axs.set_xlabel('Months')
axs.set_ylabel('Domestic Profit (Millions USD)')
axs.set_xticklabels(month_names, rotation=90)
axs.set_xticks(months)
axs.tick_params(axis="x", direction="out", pad=10)
```



Similar to the plot above this plot also shows the distibution for domestic profit (USD in hundreds of millions) for all of the months from 1915-2020. The x-axis represents the 12 months in a year. The y-axis represents the average profit domestically for each month based on the films that were released in each of those correlated months. The data bar plot displays a similar pattern to the worldwide profit as their is a large rise in profit during the months from May-July and November-December.

5.4 Recommendation #2

The large amounts of profits in the summer (May - July) and winter (November - December) indicates the schedules of audience free time dedicated to recreational activities such as watching a movie with family or friends. The movie studio would be wise to line up large project or blockbuster types of movies with these high activity months as they are more prone to being exposed to the public audience.

6 Budget

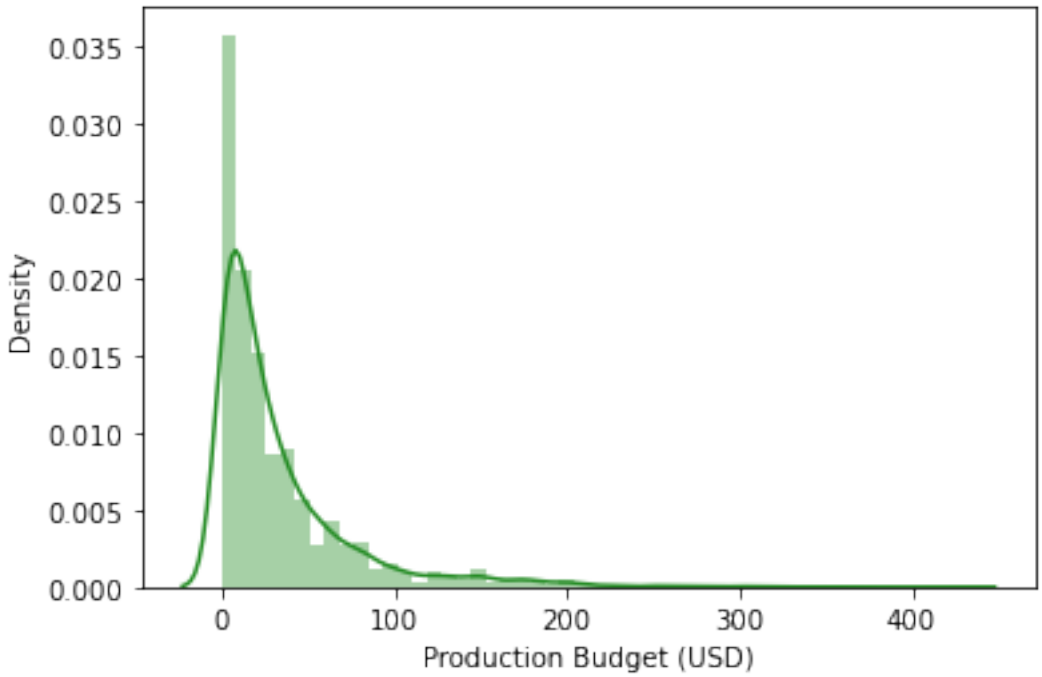
6.1 Budget and Revenue Statistical Analysis

In this section, we will perform a statistical analysis of the relationship between production budget and worldwide gross. For our analysis, we run a simple linear regression with worldwide gross as our dependent variable and production budget as our independent variable.

In the two plots below, we have the distributions for production budget and worldwide gross, respectively. Both of the distributions are skewed to the right, which is confirmed by our rejection of the null hypothesis in our normal tests. Given the fact that we will be performing a regression using OLS, the classical linear modeling assumptions allow us to generally relax the normality assumption when we have many observations, meaning we do not transform the data.

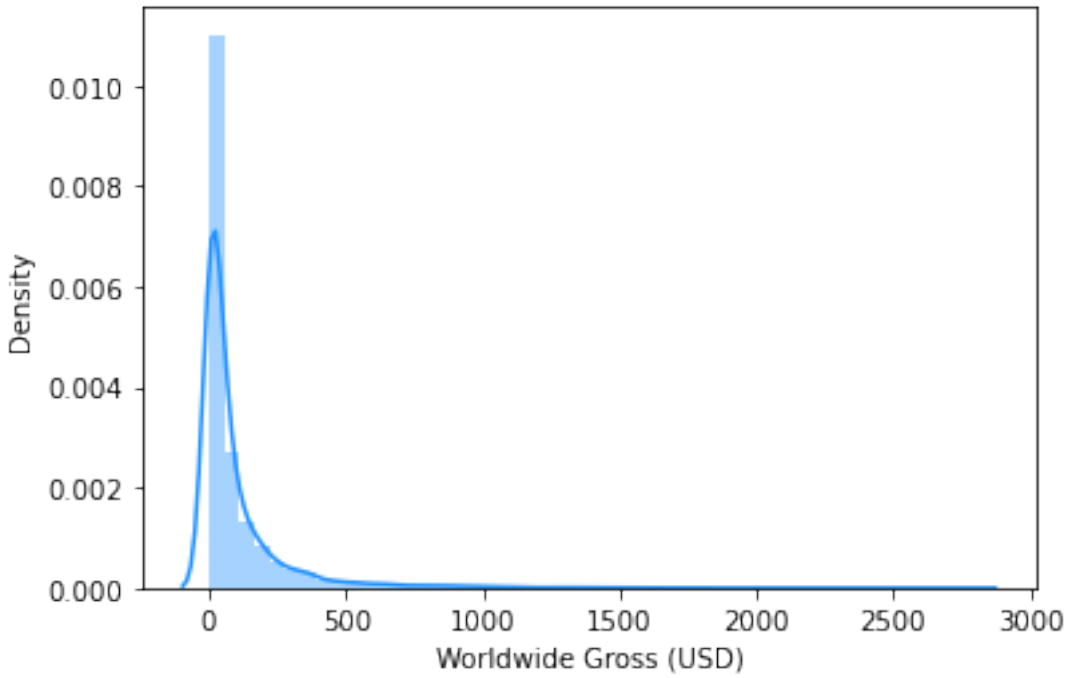
[21]:
```
#plot distribution
x = pd.Series(movies_budget['production_budgetINT'], name = "Production Budget␣
 ↪(USD)")
sns.distplot(x, color = "forestgreen")
```

[21]:
```
<AxesSubplot:xlabel='Production Budget (USD)', ylabel='Density'>
```



[22]:
```
#plot distribution
x = pd.Series(movies_budget['worldwide_grossINT'], name = "Worldwide Gross␣
 ↪(USD)")
sns.distplot(x, color = 'dodgerblue')
```

[22]: <AxesSubplot:xlabel='Worldwide Gross (USD)', ylabel='Density'>



Test for if the production budget is normally distributed. A p-value less than 0.05 rejects the null hypothesis that the data is normally distributed:

[23]:
```
#test for normality
st.normaltest(movies_budget['production_budgetINT'])
```

[23]: NormaltestResult(statistic=3117.059393788194, pvalue=0.0)

Test for if worldwide gross is normally distributed. A p-value less than 0.05 rejects the null hypothesis that the data is normally distributed:

[24]:
```
#test for normality
st.normaltest(movies_budget['worldwide_grossINT'])
```

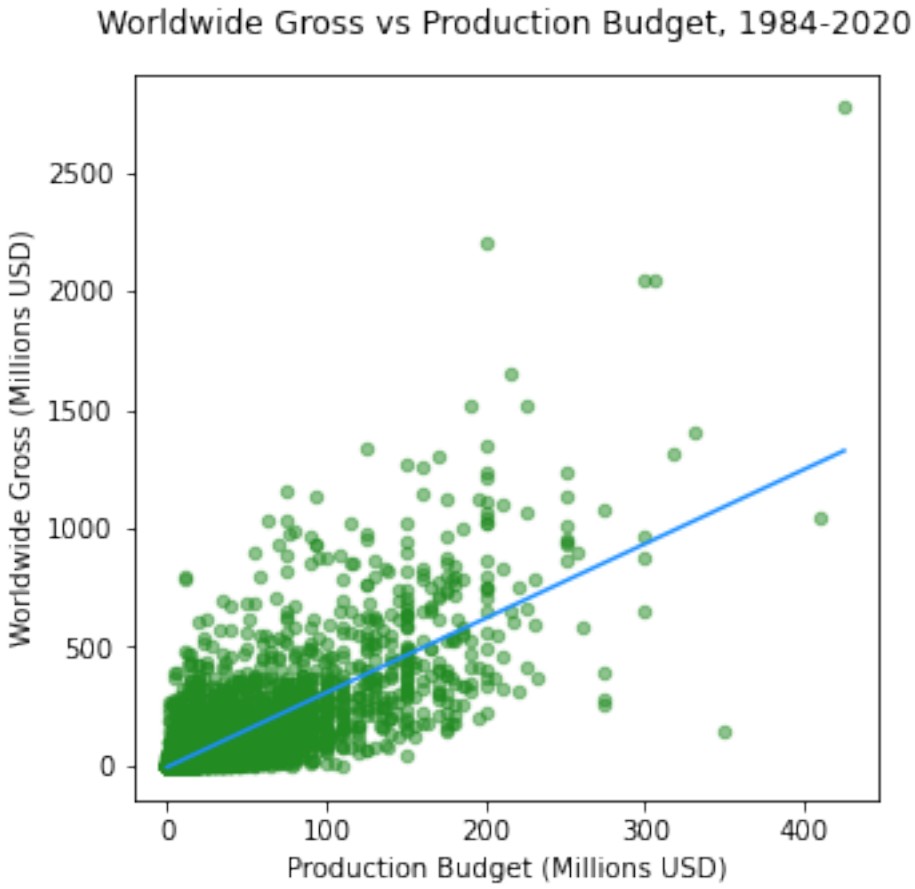
[24]: NormaltestResult(statistic=4983.028689614257, pvalue=0.0)

Below we plot a scatter of Production Budget and Worldwide Gross for all movies in the movies_budget dataset and apply a line of best fit. There is a positive correlation between budget and gross, suggesting that an increase in production budget is associated with an increase in worldwide gross. However, we also see below the line of best fit that there are a few points that seem to be following a trend that would suggest diminishing returns of production budget.

[25]:
```
#plot scatterplot of budget and revenue to show relationship between the two
fig = plt.subplots(figsize=(5, 5))
```

```
x = movies_budget['production_budgetINT']

y = movies_budget['worldwide_grossINT']

z = np.polyfit(x, y, 1)
p = np.poly1d(z)
plt.title('Worldwide Gross vs Production Budget, 1984-2020', pad = 15)

plt.xlabel('Production Budget (Millions USD)')
plt.ylabel('Worldwide Gross (Millions USD)')

plt.scatter(x, y, c ="forestgreen", alpha = .5, s = 20)
plt.plot(x, p(x), color = "dodgerblue")
```

[25]: [<matplotlib.lines.Line2D at 0x25ca9577f40>]



We perform a Simple Linear Regression, regressing production budget on worldwide gross to learn more about the relationship. Some important characteristics provided in the regression output table are the R^2 value and the coefficient with its corresponding p-value.

[26]:
```
lm = sm.OLS.from_formula('worldwide_grossINT ~ production_budgetINT',data =␣
 ↪movies_budget)
result = lm.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     worldwide_grossINT   R-squared:                       0.557
Model:                            OLS   Adj. R-squared:                  0.556
Method:                 Least Squares   F-statistic:                     6793.
Date:                Tue, 02 Aug 2022   Prob (F-statistic):               0.00
Time:                        10:03:26   Log-Likelihood:                -33567.
No. Observations:                5415   AIC:                         6.714e+04
Df Residuals:                    5413   BIC:                         6.715e+04
Df Model:                           1
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------------------
--------
Intercept               -6.9150      2.057     -3.362      0.001     -10.947
-2.883
production_budgetINT     3.1405      0.038     82.421      0.000       3.066
3.215
==============================================================================
Omnibus:                     3873.225   Durbin-Watson:                   1.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           143251.883
Skew:                           2.978   Prob(JB):                         0.00
Kurtosis:                      27.484   Cond. No.                         68.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

The coefficient from our simple linear regression suggests that an increase in 1 dollar in production budget is associated with an increase in 3.14 dollars in worldwide gross, statistically signficant at all reasonable levels.

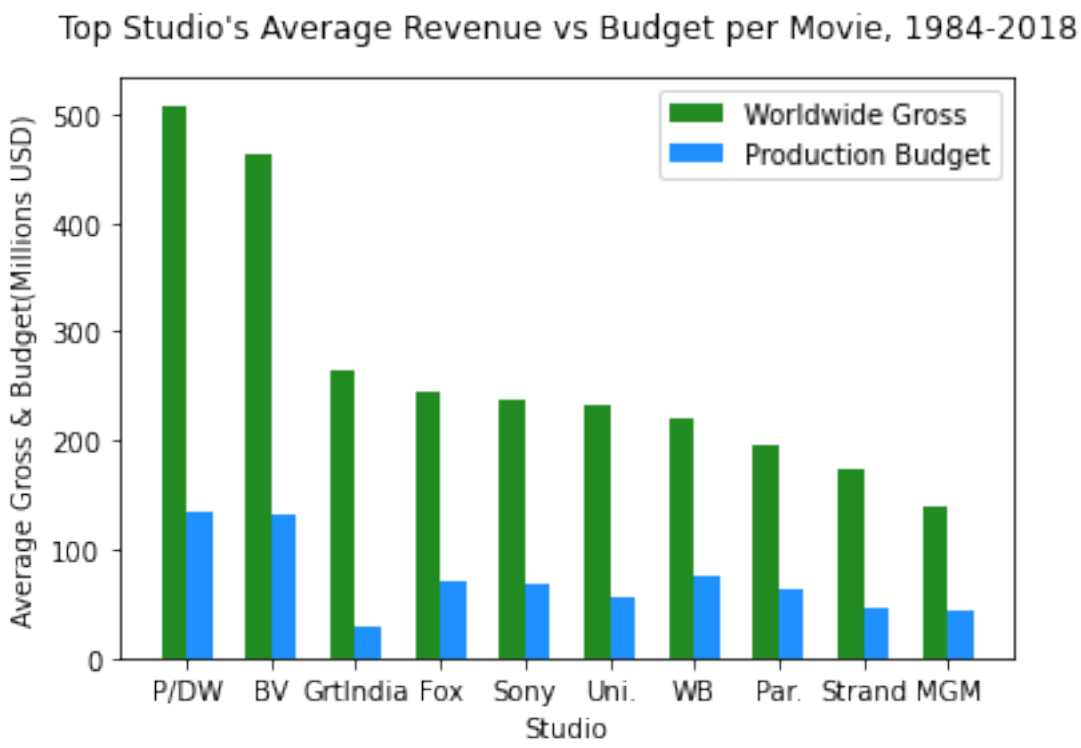
The R-squared, which is a measure of "goodness of fit" is 0.557. This means that production budget explains 55.7% of the sample variation of worldwide gross.

Given our time constraints and the scope of this project, we were unable to make a more accurate and complex model. Our simple linear regression has the problem of omitted variable bias and we did not test for other issues such as heteroskedasticity. However, this regression is sufficient when looking for a general picture of the relationship between budget and revenue.

6.2 Top Studios and their Budgets

[27]:
```
w=0.3
x = list(top_studios['studio'])
worldwide_gross = list(top_studios['worldwide_grossINT'])
production_budget = list(top_studios['production_budgetINT'])

bar1 = np.arange(len(x))
bar2 = [i+w for i in bar1]

plt.bar(bar1,worldwide_gross,w,label='Worldwide Gross', color = 'forestgreen')
plt.bar(bar2,production_budget,w,label='Production Budget', color = 'dodgerblue')
plt.title("Top Studio's Average Revenue vs Budget per Movie, 1984-2018", pad =␣
↪15)
plt.xticks(bar1+w/2,x)
plt.xlabel('Studio')
plt.ylabel('Average Gross & Budget(Millions USD)')
plt.legend()
```

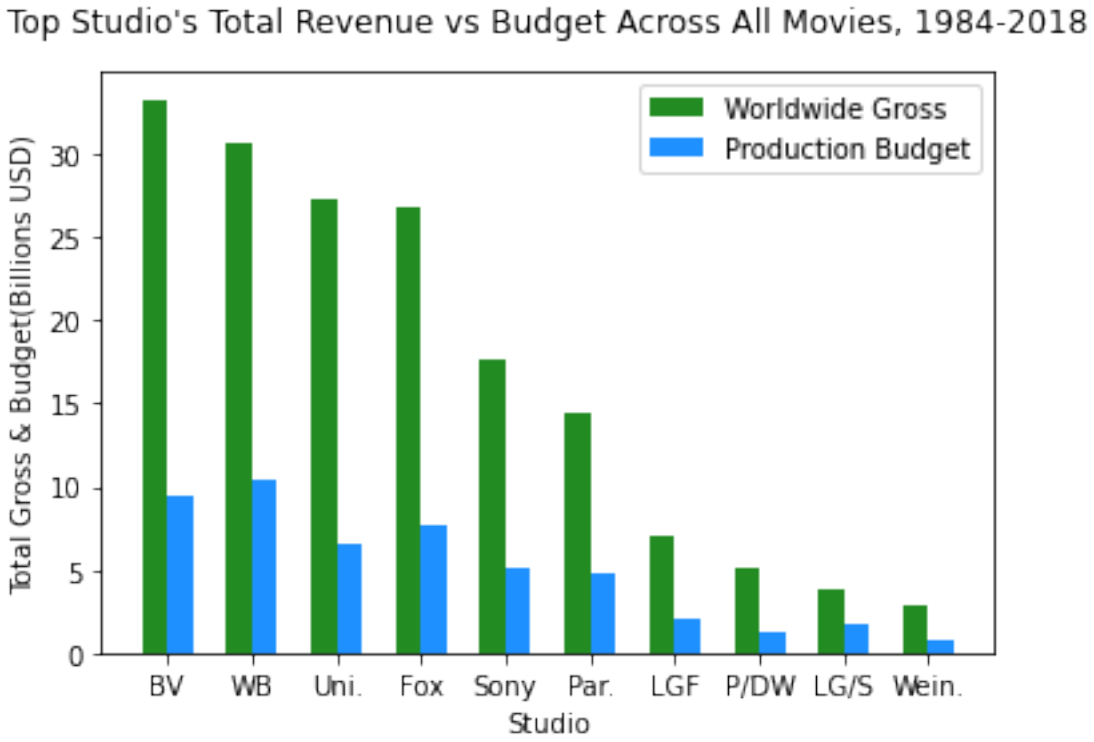
[27]:
```
<matplotlib.legend.Legend at 0x25ca8e2ff10>
```



[28]:
```
w=0.3
x = list(top_studios_total['studio'])
```

```
worldwide_gross = list(top_studios_total['worldwide_grossINT'].mul(1/1000))
production_budget = list(top_studios_total['production_budgetINT'].mul(1/1000))

bar1 = np.arange(len(x))
bar2 = [i+w for i in bar1]

plt.bar(bar1,(worldwide_gross),w,label='Worldwide Gross', color = 'forestgreen')
plt.bar(bar2,(production_budget),w,label='Production Budget', color =␣
 ↪'dodgerblue')
plt.title("Top Studio's Total Revenue vs Budget Across All Movies, 1984-2018",␣
 ↪pad = 15)
plt.xticks(bar1+w/2,x)
plt.xlabel('Studio')
plt.ylabel('Total Gross & Budget(Billions USD)')
plt.legend()
```

[28]: <matplotlib.legend.Legend at 0x25ca8c0dd90>



We found there is a significant correlation between budget and revenue, suggesting that an increase in production budget is associated with an increase in gross revenue. However, our linear model doesn't take into account what looks to be diminishing returns to increasing production budget. It looks like the diminishing returns begin between 200,000,000 USD and 250,000,000 USD. Looking at other studios, our top competitors specifically, we can see that they spend be-

tween 43,000,000 USD and 130,000,000 USD in their production budget per movie.

6.3 Recommendation #3

Taking into account the typical budgets of competitors and the positive relationship between production budget and worldwide gross, that also looks to have the possibility of diminishing returns, we recommend that Computing Vision keeps their movie budgets greater than 43,000,000 USD and less than 200,000,000 USD per movie.

6.4 Conclusion

In this notebook, we explored three topics of focus to help Computing Vision break into the movie industry. These topics related to which type of movie to release, when to release it, and how much should be spent on each movie.

Our first recommendation is to produce movies that superhero movies; of the Sci-Fi, Action, or Adventure genres; and are part of a larger franchise that fans can follow.

Our second recommendation is to release the movie in the winter or summer time to optimize profits from "summer blockbusters" and award ceremonies.

Lastly, our third recommendation is to spend more than 43,000,000 USD in production on a movie, but no more than 200,000,000 USD as trends suggest possible diminishing returns.

6.5 Further Research

Given more time, we would likely want to control for inflation in our research as a change in prices may effect the revenue and profit trends over time.

We would also be interested in looking more specifically at ROI (return on investment) to have a more targeted approach to maximizing earnings for Computing Vision.

We also would like to include more of the academic research as a part of our analysis.

Relevant papers and articles:

*Study: How Critical Are Critical Reviews? The Box Office Effects of Film Critics, Star Power, and Budgets

*Study: A Machine Learning Approach to Predict Movie Box-Office Success

*Study: Predicting Movie Success with Machine Learning Techniques: Ways to Improve Accuracy

*Study: A Predictor for Movie Success

*New York Times Article: A Big Star May Not a Profitable Movie Make - The New York Times

An interesting avenue of further research would be looking at the effect of popular actors and directors on movie performance. Intuitively, one may think that popular actors cause movies to be successful, however a study in the early 2000s by the New York Times suggested that was not true.