In [ ]:
```
import statsmodels.api as sm
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, LogisticRegression
import datetime as dt
from sklearn.model_selection import cross_val_score
from sklearn import tree
from sklearn import preprocessing
from sklearn import utils
import threading
```

Make sure movies with multiple names and organized as "Avengers, The" Get Properly Organized

In [ ]:
```
movies_df = pd.read_csv("data/movie_lense/movies.csv")
pattern = r'(\([0-9][0-9][0-9][0-9]\))'
a = movies_df['title'].str.contains(pattern)
movies_df['release'] = movies_df['title'].str.extract(pattern, expand=
True)
movies_df['release'] = movies_df['release'].str.replace('(', "")
movies_df['release'] = movies_df['release'].str.replace(')', "")
movies_df['title'] = movies_df['title'].str.replace(pattern, "")
movies_df['release'] = pd.to_numeric(movies_df['release'])
# movies_df['release'] = pd.to_datetime(movies_df['release'], format =
'%Y')
# movies_df['release'] = movies_df['release'].dt.year
movies_df['title'] = movies_df['title'].str.rstrip()
movies_df['title_lower'] = movies_df['title'].str.lower()
movies_df['new_title'] = [[]] * len(movies_df)
#movies_df['new_title'] = movies_df['new_title'].str.join(movies_df['t
itle_lower'].str.split(', '))

print(len(movies_df))

# not vectorized but idk the pandas way of doing it
for i in range(len(movies_df)):
    # separate foreign movie titles in parenthesis
    modified = False

    lst = movies_df.iloc[i]['title_lower'].split("(")
    if len(lst) > 1:
        modified = True
        #print("splitting")
        movie_info = list(movies_df.iloc[i])

        movie1_name = lst[0]
        movie2_name = lst[1]

        movies_df.iat[i, 5] = movie1_name[:-1]

        movie2_info = movie_info
        movie2_info[5] = movie2_name[:-1]

        movie_info_df = pd.DataFrame([movie2_info], columns=['movieId'
, 'title', 'genres', 'release', 'title_lower', 'new_title'])
        movies_df = movies_df.append(movie_info_df, ignore_index = Tru
e)

    # a.k.a.
    lst = movies_df.iloc[i]['title_lower'].split("(a.k.a.")
    if len(lst) > 1:
        modified = True
        movie_info = list(movies_df.iloc[i])

        movie1_name = lst[0]
        movie2_name = lst[1]

        print("splitting",movie2_name)

        movies_df.iat[i, 5] = movie1_name[:-1]
```

```
        movie2_info = movie_info
        movie2_info[5] = movie2_name[1:-1]

        movie_info_df = pd.DataFrame([movie2_info], columns=['movieId'
, 'title', 'genres', 'release', 'title_lower', 'new_title'])
        movies_df = movies_df.append(movie_info_df, ignore_index = Tru
e)
        #print(movies_df[movies_df['new_title'] == "twelve monkeys"])

    # Avengers, The --> The Avengers
    lst = movies_df.iloc[i]['title_lower'].split(", ")
    #print(lst)
    if len(lst) > 1:
        modified = True
        lst.insert(0, lst.pop())
        return_val = " "
        #print(lst)
        movies_df.iat[i, 5] = return_val.join(lst)
        movie_info = list(movies_df.iloc[i])
        movie_info[5] = lst[0] + ', ' + lst[1]

        movie_info_df = pd.DataFrame([movie_info], columns=['movieId',
'title', 'genres', 'release', 'title_lower', 'new_title'])
        movies_df = movies_df.append(movie_info_df, ignore_index = Tru
e)

    if not modified:
        movies_df.iat[i, 5] = movies_df.iloc[i, 4]
movies_df
```

In [ ]:
```
movies_df#[movies_df['new_title'] == "twelve monkeys"]
#movies_df = movies_df[movies_df['release'] >= 1996]
```

Intersect the movies dataframe with the industry moves dataframe

In [ ]:
```
industry_df = pd.read_csv("data/movie_industry.csv", engine = 'python'
)
industry_df['new_title'] = industry_df['name'].str.lower()
industry_df['released'] = pd.to_datetime(industry_df['released'])
industry_df = industry_df[industry_df['released'] >= pd.to_datetime('1
995-11-01')]
```

In [ ]:
```
def onehotencode(movies_df):

    def splitColumn(dataframe, column_name, delimiter):
        new = dataframe[column_name].str.split(delimiter, expand=True)
        return new

    new = splitColumn(movies_df, "genres", "|")
    movies_df = movies_df.assign(first_genre=new[0], second_genre=new[
1], third_genre=new[2], fourth_genre=new[3], fifth_genre=new[4],
                                                          sixth_genre=new
[5], seventh_genre=new[6])
    movies_df = movies_df.drop(labels=["genres"], axis=1)
    y = pd.get_dummies(movies_df[["first_genre", "second_genre", "thir
d_genre", "fourth_genre", "fifth_genre", "sixth_genre", "seventh_genr
e"]])

    mapping = {}
    def makeMapping(y):
      for i in range(80):
        if "Action" in y.columns[i]:
          mapping.update({y.columns[i]: "Action"})
        if "Adventure" in y.columns[i]:
          mapping.update({y.columns[i]: "Adventure"})
        if "Animation" in y.columns[i]:
          mapping.update({y.columns[i]: "Animation"})
        if "Children" in y.columns[i]:
          mapping.update({y.columns[i]: "Children"})
        if "Comedy" in y.columns[i]:
          mapping.update({y.columns[i]: "Comedy"})
        if "Crime" in y.columns[i]:
          mapping.update({y.columns[i]: "Crime"})
        if "Documentary" in y.columns[i]:
          mapping.update({y.columns[i]: "Documentary"})
        if "Drama" in y.columns[i]:
          mapping.update({y.columns[i]: "Drama"})
        if "Fantasy" in y.columns[i]:
          mapping.update({y.columns[i]: "Fantasy"})
        if "Horror" in y.columns[i]:
          mapping.update({y.columns[i]: "Horror"})
        if "Musical" in y.columns[i]:
          mapping.update({y.columns[i]: "Musical"})
        if "Mystery" in y.columns[i]:
          mapping.update({y.columns[i]: "Mystery"})
        if "Romance" in y.columns[i]:
          mapping.update({y.columns[i]: "Romance"})
        if "Sci-Fi" in y.columns[i]:
          mapping.update({y.columns[i]: "Sci-Fi"})
        if "Thriller" in y.columns[i]:
          mapping.update({y.columns[i]: "Thriller"})
        if "Western" in y.columns[i]:
          mapping.update({y.columns[i]: "Western"})

    makeMapping(y)
    y = y.set_index("first_genre_(no genres listed)").groupby(mapping,
axis=1).sum()
    movies_df = movies_df.drop(["first_genre", "second_genre", "third_
```

```
genre", "fourth_genre", "fifth_genre", "sixth_genre", "seventh_genre"
], axis=1)
    y.reset_index(drop=True, inplace=True)
    concat = pd.concat([movies_df, y], axis=1)
    return concat
```

In [ ]:
```
movies_df = onehotencode(movies_df)
#movies_df.to_csv("data/movies_w_genre_after96.csv", index=False)
movies_df
```

In [ ]:
```
# TODO: MERGE IT BETTER. some movies are omitted during the merge, suc
h as "The Avengers" and "Avengers, The". Problem also occurs with fore
ign films when there are translations in brackets

industry_df['release'] = pd.to_numeric(industry_df['year'])
movies_ind_subset = pd.merge(industry_df, movies_df, on =["release",
"new_title"])
merged = movies_ind_subset
original = industry_df
dup = pd.concat([merged, original])
merged#.sort_values(by=['movieId'])
# dup.drop_duplicates()
# merged
# dup.drop_duplicates(subset = ["title_lower", "release"], keep = Fals
e)
```

In [ ]:
```
merged = merged.drop(columns = ["genre", "new_title", "year"])
merged
```

In [ ]:
```
merged_w_profits = merged
#merged_w_profits = merged[merged["budget"] > 0]
#merged_w_profits = merged_w_profits[merged["gross"] > 0]
merged_w_profits["profit"] = 0
merged_w_profits["profit"] = merged_w_profits["gross"] - merged_w_prof
its["budget"]
merged_w_profits["profit_margin"] = merged_w_profits["profit"]/merged_
w_profits["gross"]

# assumption that every movie without a budget has the averaged profit
margin with the movies with a budget
for i in merged_w_profits.index:
    print("in for loop")
    if merged_w_profits.loc[i, 'budget'] == 0:
        merged_w_profits.at[i, 'budget'] = (1-avg_profit_margin) * mer
ged_w_profits.loc[i, 'gross']

#merged_w_profits[merged_w_profits['budget'] == 0] = (1-avg_profit_mar
gin) * merged_w_profits[merged_w_profits['budget'] == 0].gross

avg_profit_margin = (merged_w_profits[merged_w_profits['budget'] != 0]
.gross.sum() - merged_w_profits[merged_w_profits['budget'] != 0].budge
t.sum())/merged_w_profits[merged_w_profits['budget'] != 0].gross.sum()
```

In [ ]:
```
merged_w_profits.to_csv("data/movies_w_genre_after96.csv", index=False
)
```

In [ ]:
```
merged_w_profits.columns.tolist()#.to_csv("data/movies_w_genre_after96.csv", index=False)
```

In [ ]:
```
movies_df
```

In [ ]:
```

```