```
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 matplotlib.dates as mdates
import random
from datetime import date, timedelta
```

```
ind_movies_df = pd.read_csv("data/movies_w_genre_profits.csv")
ind_movies_full_df = pd.read_csv("data/movie_industry.csv", engine = "python")
ratings_df = pd.read_csv("data/movie_lense/ratings.csv")
```

```
ratings_df = pd.read_csv("data/movie_lense/ratings.csv")
ratings_df['timestamp'] = pd.to_datetime(ratings_df['timestamp'], unit = 's')
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['title'] = movies_df['title'].str.rstrip()
```

```
ind_movies_df["ml_rating"] = ind_movies_df["movieId"].apply(get_ml_rating)
```

```
ind_movies_df["ml_count"] = ind_movies_df["movieId"].apply(get_ml_count)
```

```
def get_ml_rating(movieId):
    return ratings_df[ratings_df["movieId"] == movieId]["rating"].mean()

def get_ml_count(movieId):
    return ratings_df[ratings_df["movieId"] == movieId]["rating"].count()
```

```
ind_movies_w_genre_df = ind_movies_df.merge(ind_movies_full_df, on = "name")
ind_movies_w_genre_df
```

```
ind_movies_df_nums = ind_movies_df.drop(columns = ["writer", "director", "country", "r
```

```
ind_movies_df_nums = ind_movies_df.drop(columns = ["writer", "director", "country", "c
ind_movies_df_nums


ind_movies_corr = ind_movies_df_nums[["budget", "gross", "runtime", "score", "votes",
ind_movies_corr



mask = np.triu(np.ones_like(ind_movies_corr.corr(), dtype=bool))

sns.heatmap(ind_movies_corr.corr(), cmap = "YlGnBu", mask = mask, xticklabels = ind_mo




ind_movies_corr.corr()


ratings_df['timestamp'] = pd.to_datetime(ratings_df['timestamp'], unit = 's')


ratings_df["month"] = pd.DatetimeIndex(ratings_df['timestamp']).month
ratings_df["year"] = pd.DatetimeIndex(ratings_df['timestamp']).year
ratings_df.sort_values("timestamp")


ratings_monthly = ratings_df.groupby(["month", "year"]).mean()["rating"]


ratings_monthly = ratings_monthly.to_frame()



ratings_monthly.index = ratings_monthly.index.set_names(['month', 'year'])


heatmap1_data = pd.pivot_table(ratings_monthly, values='rating',
                     index=['month'],
                     columns='year')


ratings_monthly = ratings_monthly.drop(index = [0])


heatmap1_data_no2018 = heatmap1_data.drop(columns = [2018])


sns.heatmap(heatmap1_data_no2018)


movies_df_after96 = [movies_df["release"] >= 1996]


ind_movies_df.columns
```

```
ind_movies_release = ind_movies_df[["movieId", "released"]]


# ratings_release_df = ratings_df.merge(movies_df, on = "movieId")
# ratings_release96_df = ratings_release_df[ratings_release_df["release"] >= 1996]
# ratings_release96_df.reset_index


ratings_release_df = ratings_df.merge(ind_movies_release, on = "movieId")


ratings_release_df["released"] = pd.to_datetime(ratings_release_df["released"])
ratings_release_df


ratings_release_df['nb_months'] = ((ratings_release_df.timestamp - ratings_release_df.
ratings_release_df


ratings_release_df["movieId"].nunique()


ratings_release_df["nb_months"] = ratings_release_df["nb_months"].clip(lower = 0)
# ratings_release_df["nb_months"]
plt.hist(ratings_release_df["nb_months"], bins = 50)


ind_movies_release_genre = ind_movies_w_genre_df[["movieId", "genre", "released_x"]]


ratings_release_genre_df = ratings_df.merge(ind_movies_release_genre, on = "movieId")
ratings_release_genre_df["released_x"] = pd.to_datetime(ratings_release_genre_df["rele
ratings_release_genre_df['timestamp'] = pd.to_datetime(ratings_release_genre_df['times
ratings_release_genre_df['nb_months'] = ((ratings_release_genre_df.timestamp - ratings


for_plotting = ratings_release_genre_df[["genre", "nb_months"]]
for_plotting["nb_months"] = for_plotting["nb_months"].clip(lower = 0)
genres = for_plotting.genre.unique().tolist()
random.shuffle(genres)
for i in range(int(len(genres)/2)):
# for i in range(1):
    plt.hist(for_plotting[for_plotting["genre"] == genres[i]]["nb_months"], alpha = 0.

plt.legend(fontsize = 8)
plt.xlabel("Months after release")
plt.ylabel("Review Count (%)")
plt.title("Post-Release Review Distribution by Genre")


for i in range(int(len(genres)/2), len(genres)):
# for i in range(1):
    plt.hist(for_plotting[for_plotting["genre"] == genres[i]]["nb_months"], alpha = 0.
```

```
plt.legend(fontsize = 8)
plt.xlabel("Months after release")
plt.ylabel("Review Count (%)")
plt.title("Post-Release Review Distribution by Genre")


# ind_movies_w_genre_df
ind_movies_genre_count = ind_movies_w_genre_df[["released_y", "genre"]]
ind_movies_genre_count["released_y"] = pd.to_datetime(ind_movies_genre_count["releasec
ind_movies_genre_count["month"] = pd.DatetimeIndex(ind_movies_genre_count['released_y'
ind_movies_genre_count["year"] = pd.DatetimeIndex(ind_movies_genre_count['released_y']
groupby_genres = ind_movies_genre_count.groupby(["year", "month", "genre"]).count()


groupby_genres.reset_index(inplace = True)


groupby_genres_1996 = groupby_genres[groupby_genres["year"] >= 1996]


groupby_genres_1996


groupby_genres_1996
total_count = []
for genre in genres:
# for genre in ['Action']:
    temp = []
    by_genre = groupby_genres_1996[groupby_genres_1996["genre"] == genre]
    for year in groupby_genres_1996.year.unique()[:len(groupby_genres_1996.year.unique
        try:
            by_year = by_genre[by_genre["year"] == year]
        except:
            temp = temp + [0] * 12
        for month in groupby_genres_1996.month.unique():
            current = by_year[by_year["month"] == month]["released_y"]
            if len(current) == 0:
                temp.append(0)
            else:
                temp.append(by_year[by_year["month"] == month]["released_y"].mean())
    total_count.append(temp)

total_count


tc_df = pd.DataFrame(total_count)
tc_df


tc_df = pd.DataFrame(total_count)
for i in range(len(tc_df.columns)):
    tc_df[i] = tc_df[i]/tc_df[i].sum()
tc_df = tc_df.fillna(0)
```

```
tc_df = tc_df.fillna(0)

tc_df


tc_df = tc_df.fillna(0)


yrs = [1996, 2017]

totalMonths= 12*(np.max(yrs) - np.min(yrs)+1)
dates = mdates.date2num([date(np.min(yrs)+(i//12),i%12+1,1) for i in range(totalMonths


plt.figure(figsize=(30,6))
plt.stackplot(dates,tc_df)


groupby_genres_year = ind_movies_genre_count.groupby(["year","genre"]).count()
groupby_genres_year.reset_index(inplace = True)
groupby_genres_year_1996 = groupby_genres_year[groupby_genres_year["year"] >= 1996]
groupby_genres_year_1996

total_count = []
for genre in genres:
# for genre in ['Action']:
    temp = []
    by_genre = groupby_genres_1996[groupby_genres_1996["genre"] == genre]
    for year in groupby_genres_1996.year.unique()[:len(groupby_genres_1996.year.unique
        by_year = by_genre[by_genre["year"] == year]
        if len(by_year) == 0:
            temp.append(0)
        else:
            temp.append(by_year["released_y"].sum())
    total_count.append(temp)

total_count


len(total_count[0])


years = [1996]
for i in range(1,2017-1996+1):
    years.append(i+years[0])
len(years)


tc_df = pd.DataFrame(total_count)
for i in range(len(tc_df.columns)):
    tc_df[i] = tc_df[i]/tc_df[i].sum()
tc_df = tc_df.fillna(0)
plt.figure(figsize=(10,6))

NUM_COLORS = 16
```

```
NUM_COLORS = 16
sns.reset_orig()  # get default matplotlib styles back
clrs = sns.color_palette('husl', n_colors=NUM_COLORS)  # a list of RGB tuples

plt.stackplot(years,tc_df, labels = genres, colors = clrs)
plt.legend(genres)
```