```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
import statsmodels.api as sm
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.cluster import KMeans
from sklearn.linear_model import SGDRegressor
import threading


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['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()

def month_diff(a, b):
    if a >= b:
        earlier = b
        later = a
    else:
        earlier = a
        later = b
    year_diff = (later.year - earlier.year)*12
    month_diff = (later.month - earlier.month)*(later.month >= earlier.month) + (la
    return year_diff+month_diff

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_gen
                                                        sixth_genre=new[5], seventh_
```

```
movies_df = movies_df.drop(labels=["genres"], axis=1)
y = pd.get_dummies(movies_df[["first_genre", "second_genre", "third_genre", "fo

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", "four
y.reset_index(drop=True, inplace=True)
concat = pd.concat([movies_df, y], axis=1)
return concat
```

```
movies_df = onehotencode(movies_df)


def movie_ols(movieID, ratings = ratings_df, print = False):
    movie_rating_df = ratings_df[ratings_df["movieId"] == movieID]
    movie_ratings_grouped = movie_rating_df.groupby([movie_rating_df['timestamp'].d
    earliest = movie_rating_df['timestamp'].min()
    movie_rating_df["months_delta"] = movie_rating_df['timestamp'].apply(lambda x:
    movie_rating_df

    freq = movie_rating_df.groupby("months_delta").count()["userId"]
    ratings = movie_rating_df.groupby("months_delta").mean()["rating"]
    data = pd.concat([freq,ratings], axis = 1)
    data
    data["count"] = data["userId"]
    data["ones"] = np.ones(data.shape[0])
    data["months_delta"] = data.index
    y = data["rating"]
    x = data[["count", "months_delta", "ones"]]

    model = sm.OLS(y, x, missing='drop')
    res = model.fit()
    if print:
        print(res.summary())
    return res.params, res.pvalues



movies_after95 = movies_df[movies_df["release"] >= 1995].reset_index()
movies_after95["movieId"][0]


results = []
for i in range(1000):
    results.append(movie_ols(movies_after95["movieId"][i]))
    print(i/100)


count_pvalues = [results[i][1][0] for i in range(len(results))]
months_pvalues = [results[i][1][1] for i in range(len(results))]

count_coeff = [results[i][0][0] for i in range(len(results))]
months_coeff = [results[i][0][1] for i in range(len(results))]


plt.hist(months_pvalues)
```

```
# new variables
movie_rating_df = ratings_df[ratings_df["movieId"] == movieID]
movie_ratings_grouped = movie_rating_df.groupby([movie_rating_df['timestamp'].dt.ye
movie_rating_df["months_delta"] = movie_rating_df['timestamp'].apply(lambda x: mont
movie_rating_df


ratings_df.groupby("userId").count()


user_counts = ratings_df.groupby("userId").count()
user_rating_count = user_counts["rating"]
user_meanrating = ratings_df.groupby("userId").mean()["rating"]
groupby_month = movie_rating_df.groupby("months_delta")
reviewcount_monthly = []
meanrating_monthly = []
for name, group in groupby_month:
    count = len(group["userId"])
    group_users = group["userId"]
    group_reviews = user_rating_count[group["userId"]]
    group_mean = user_meanrating[group["userId"]]
    mean_review_count = group_reviews.mean()
    mean_ratings = group_mean.mean()
    reviewcount_monthly.append(mean_review_count)
    meanrating_monthly.append(mean_ratings)
```

```
def movie_timeseries(movieID, ratings = ratings_df):
    movie_rating_df = ratings_df[ratings_df["movieId"] == movieID]
    earliest = movie_rating_df['timestamp'].min()
    movie_rating_df["months_delta"] = movie_rating_df['timestamp'].apply(lambda x:
#     print(1)
    freq = movie_rating_df.groupby("months_delta").count()["userId"]
    ratings = movie_rating_df.groupby("months_delta").mean()["rating"]
    data = pd.concat([freq,ratings], axis = 1)
    data
#     print(2)
    groupby_month = movie_rating_df.groupby("months_delta")
    groupby = ratings_df.groupby("userId")["rating"]
    user_rating_count = groupby.count()
    user_meanrating = groupby.mean()
    reviewcount_monthly = []
    meanrating_monthly = []
    months_delta = []
#     print(3)
    for name, group in groupby_month:
        count = len(group["userId"])
        group_users = group["userId"]
        group_reviews = user_rating_count[group["userId"]]
        group_mean = user_meanrating[group["userId"]]
        mean_review_count = group_reviews.mean()
        mean_ratings = group_mean.mean()
        reviewcount_monthly.append(mean_review_count)
        meanrating_monthly.append(mean_ratings)
        months_delta.append(group["months_delta"].mean())

    # print(4)
    additional_data = pd.concat([pd.Series(reviewcount_monthly),pd.Series(meanratin
    additional_data.columns = ["userreviewcount", "usermeanrating", "months_delta"]
    data["months_delta"] = data.index
    data.index.names = ["index"]
    full_data = data.merge(additional_data, on = ["months_delta"])
    full_data
    # print(5)
    full_data["months_delta"] = full_data["months_delta"] + 1
    full_data["count"] = full_data["userId"]
    full_data = full_data.drop(columns = ["userId"])
    return full_data


movie_timeseries(1)
```

```
def apply_ols(results, printout = False):
    results["ones"] = np.ones(results.shape[0])
    y = results["rating"]
    x = results[["months_delta", "usermeanrating", "ones"]]
    model = sm.OLS(y, x, missing='drop')
    res = model.fit()
    if printout:
        print(res.summary())
    return res.params, res.pvalues


ols = apply_ols(movie_timeseries(1))


ols


results = []
for i in range(100):
    results.append(apply_ols(movie_timeseries(movies_after95["movieId"][i])))
    print(i/100)


# three variables analysis
months_pvalues = [results[i][1][0] for i in range(len(results))]
userreviewcount_pvalues = [results[i][1][1] for i in range(len(results))]
usermeanrating_pvalues = [results[i][1][2] for i in range(len(results))]

months_coeffs = [results[i][0][0] for i in range(len(results))]
userreviewcount_coeffs = [results[i][0][1] for i in range(len(results))]
usermeanrating_coeffs = [results[i][0][2] for i in range(len(results))]


# two variables analysis
months_pvalues = [results[i][1][0] for i in range(len(results))]
usermeanrating_pvalues = [results[i][1][1] for i in range(len(results))]
# usermeanrating_pvalues = [results[i][1][2] for i in range(len(results))]

months_coeffs = [results[i][0][0] for i in range(len(results))]
usermeanrating_coeffs = [results[i][0][1] for i in range(len(results))]
# usermeanrating_coeffs = [results[i][0][2] for i in range(len(results))]


plt.hist(usermeanrating_coeffs, bins = 50)


months_pvalues.index(max(months_pvalues))
months_pvalues
```

```
movies_after95_ids = pd.read_csv("movies_after95.csv")
more_than_10_months = movies_after95_ids[movies_after95_ids["months_w_data"] >= 24]
more_than_100 = more_than_10_months[(more_than_10_months["rating"] < 1000) & (more_
regression_movies = more_than_100.reset_index().drop(columns = ["index", "Unnamed:
# regression_movies.index
# regression_movies["movieId"][7577]
regression_movies


apply_ols(movie_timeseries(regression_movies["movieId"][2411]))


results = {}
for i in regression_movies.index:
# for i in range(10):
#     print(regression_movies["movieId"][i])
    movieId = regression_movies["movieId"][i]
    ols_result = apply_ols(movie_timeseries(movieId))

    results[movieId] = ols_result
    print(i/len(regression_movies))


# len(results)
months_pvalues = [results_geq_1000[key][1][0] for key in results_geq_1000.keys()] +
usermeanrating_pvalues = [results_geq_1000[key][1][1] for key in results_geq_1000.k




months_coeff = [results_geq_1000[key][0][0] for key in results_geq_1000.keys()] + [
usermeanrating_coeff = [results_geq_1000[key][0][1] for key in results_geq_1000.key


results[1]


# plt.hist(months_pvalues, bins = 10)
plt.hist(months_coeff, bins = 30, range = [-0.03, 0.03])


apply_ols(movie_timeseries(1), True)
```