In [ ]:
```
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.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import SGDRegressor
import threading
```

Data Preprocessing

In [ ]:
```
ratings_df = pd.read_csv("data/movie_lense/ratings.csv")
ratings_df['timestamp'] = pd.to_datetime(ratings_df['timestamp'], unit
= 's')
ratings_df
```

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
```

In [ ]:
```
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) + (later.month < earlier.month)*(- earlier.month + later.month
)
    return year_diff+month_diff
```

I organized the RDD code into the functions below

In [ ]:
```
# Run RDD on a single given movie
def RDD_Movie(movie_rating_monthly):
  X = movie_rating_monthly[['month', 'postCutoff', 'ones']]
  y = movie_rating_monthly[['rating']]
  model = sm.OLS(y, X, missing='drop')
  res = model.fit()
  #print(res.summary())
  return res.params
```

In [ ]:
```
def processMovie(movieId, numMonthsConsidered, cutoffNumMonths):
  movie_rating_df = ratings_df[ratings_df["movieId"] == movieId]
  movie_ratings_grouped = movie_rating_df.groupby([movie_rating_df['ti
mestamp'].dt.year, movie_rating_df['timestamp'].dt.month])
  movie_rating_df["months_delta"] = movie_rating_df['timestamp'].apply
(lambda x: month_diff(x, movie_rating_df['timestamp'].min()))
  movie_rating_monthly = movie_rating_df.groupby("months_delta").mean
()
  movie_rating_monthly = movie_rating_monthly.drop(movie_rating_monthl
y.index[0])
  movie_rating_monthly = movie_rating_monthly[movie_rating_monthly.ind
ex < movie_rating_monthly.index[0] + numMonthsConsidered]
  movie_rating_monthly["after_numMonths"] = movie_rating_monthly.index
> movie_rating_monthly.index[0] + cutoffNumMonths
  movie_rating_monthly["postCutoff"] = movie_rating_monthly.index * mo
vie_rating_monthly["after_numMonths"]
  movie_rating_monthly['ones'] = np.ones(movie_rating_monthly.shape[0
])
  movie_rating_monthly['month'] = movie_rating_monthly.index
  return movie_rating_monthly
```

In [ ]:
```
# Plot the RDD analysis for a movie
def plotRDD(movie_df, params):
  plt.plot(movie_df["month"], params[0]*movie_df["month"] + params[1]*
movie_df["postCutoff"] + params[2]*movie_df["ones"])
```

In [ ]:
```
# plot ratings vs months for a movie
def plotRatings(movie_df):
  movie_df.plot.scatter("month", "rating")
```

In [ ]:
```
def doMovie(movie_Id, totalMonths=1000, cutoffMonths=3):
  # Add necessary columns
  movie_df = processMovie(movie_Id, totalMonths, cutoffMonths)
  # Plot ratings vs months
  plotRatings(movie_df)

  # Get X, y and scale them, save the scale parameters into scalarX an
d scalarY
  X, y = getXy(movie_df, degree=6)
  scaled_X, scaled_y, scalarX, scalarY = scaleXy(X, y)

  # Run regression and rescale the coefficients
  scaled_coefficients = runRegression(scaled_X, scaled_y)
  coefficients = unscale_coefficients(scaled_coefficients, scalarX, sc
alarY)

  plotRegression(X, coefficients)

  numMonths = len(movie_df.index)
  return coefficients, numMonths
```

In [ ]:
```
def unscale_coefficients(scaled_coefficients, scalarX, scalarY):
  unscaled = scaled_coefficients / scalarX.scale_ * scalarY.scale_
  unscaled[0] = scaled_coefficients[0] * scalarY.scale_ + scalarY.mean
_ - sum(scalarY.scale_ / scalarX.scale_ * scaled_coefficients * scalar
X.mean_)
  print("Rescaled coefficients:", unscaled)
  return unscaled
```

In [ ]:
```
def scaleXy(X, y):
  scalarX, scalarY = StandardScaler(), StandardScaler()
  X = scalarX.fit_transform(X)
  y = scalarY.fit_transform(y)
  return X, y, scalarX, scalarY
```

In [ ]:
```
def getXy(movie_df, degree):
  X = np.array(movie_df["month"]).reshape(-1,1)
  y = movie_df[["rating"]]
  X = PolynomialFeatures(degree=degree).fit_transform(X)
  return X, y
```

In [ ]:
```
def plotRegression(X, coefficients):
  plt.plot(X[:, 1], X @ coefficients.T)
```

In [ ]:
```
def runRegression(X, y):
  #reg = LinearRegression().fit(X, y)
  reg = Ridge(alpha=0.00002).fit(X, y)
  #reg = SGDRegressor(penalty='l1', alpha=0, eta0=0.1, max_iter=1000
0).fit(X, y)

  coefficients = np.concatenate((reg.intercept_, reg.coef_.flatten()[1
:]), axis=0)
  return coefficients
```

In [ ]:
```
# Toy Story
doMovie(1)
```

In [ ]:
```
# Jumanji
doMovie(2)
```

In [ ]:
```
# Mulan
doMovie(1907)
```

In [ ]:
```
# Iron Man
doMovie(59315)
```

In [ ]:
```
# Skyfall
doMovie(96079)
```

In [ ]:
```
# Casino Royale
doMovie(5796)
```

In [ ]:
```
# Ice Age
doMovie(5218)
```

In [ ]:
```
def threadedGenerateEffectSizes(i, movieIds, coefficients, numMonths):
    try:
        coef, numMonth = doMovie(i)
        with lock:
            print(i)
            coefficients.append(coef)
            numMonths.append(numMonth)
            movieIds.append(i)
    except:
            x=1
```

In [ ]:
```
threads = []
movieIds = []
coefficients = []
numMonths = []
lock = threading.Lock()

for i in movies_df["movieId"].unique():
    thread = threading.Thread(target=threadedGenerateEffectSizes, args
=(i, movieIds, coefficients, numMonths,))
    threads.append(thread)
    thread.start()

for t in threads:
    t.join()
```

In [ ]:
```
movie_coefficients_df = pd.DataFrame(np.hstack([np.array(movieIds).res
hape(-1, 1), np.array(numMonths).reshape(-1 ,1), np.array(coefficients
)]))
```

In [ ]:
```
#code works, but not going to re-run because it will take hours to pro
cess
movie_coefficients_df = movie_coefficients_df.sort_values(by=[0], asce
nding = True)
```

In [ ]:
```
movie_coefficients_df.to_csv("movie_coefficents.csv", index=False)
```

In [ ]:
```
movie_coefficients_df.tail(5)
```

In [ ]:
```
movies_df["movieId"].unique().size
```

In [ ]:
```
len(movieIds)
```

In [ ]:
```

```