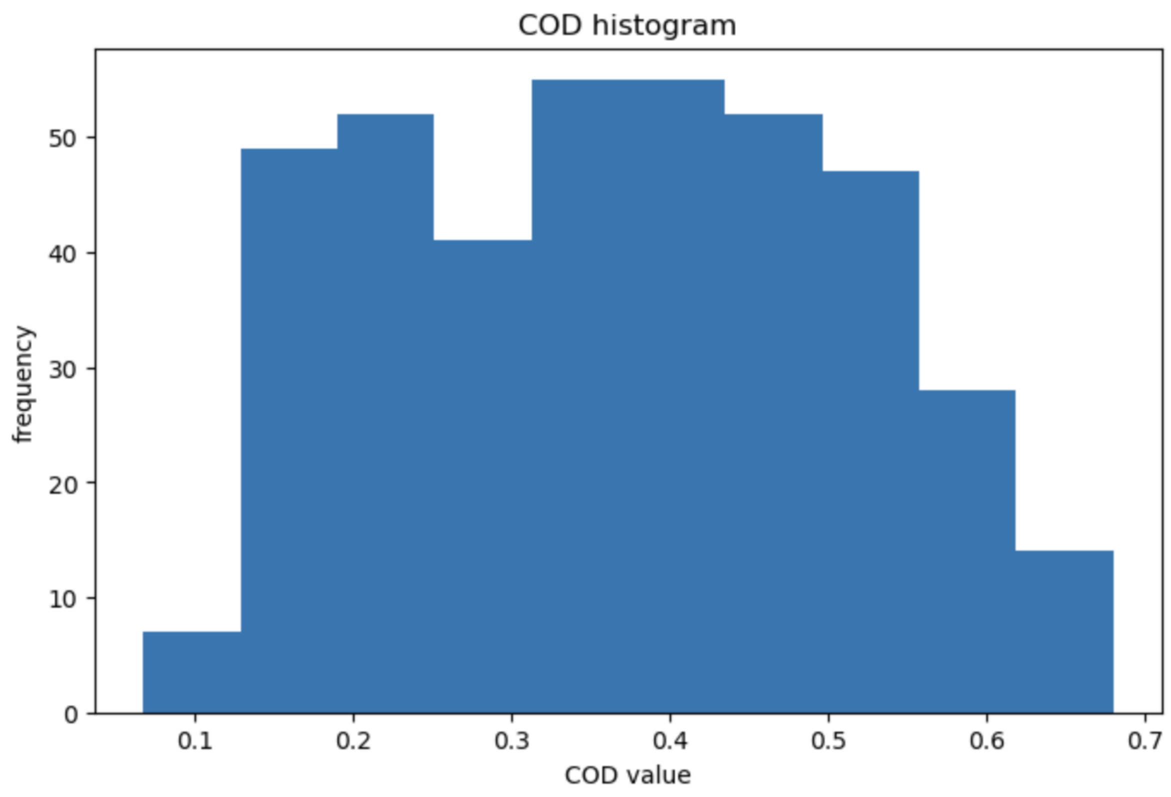
Data Analysis Project 2

Question 1:

D: Firstly, to handle the missing ratings, we have replaced each NaN rating in the dataset with a 50/50 blend of the arithmetic mean of its associated column and the arithmetic mean of its associated row. For example, if the rating of user 350 for movie 200 is missing and that the average rating of this user for other movies is 4 and the average rating (by other users) for this movie is 3, the to-be-imputed rating would be 3.5.
To solve this problem, we iterated over the 400 movies, and for each movie, we used the ratings from the other 399 movies to predict its movie ratings via building a simple linear regression model. Hence, we are building 399 simple linear regression models for each movie, and in each of these models, the predictor(x) is the ratings of one of the 399 other movies, and the outcome(y) is the ratings of this movie. While building each linear regression model, we calculate their CODs, and only store the movie that is associated with the highest COD when predicting the ratings for each movie. As a result, we have found the best predictors for each movie. Then we calculated the mean for those 400 best CODs and plotted a histogram for those 400 CODs. Finally, we sort the movies twice: first time we sort based on how easily their ratings can be predicted, and second time we sort based on how hard their ratings can be predicted, given that higher COD means easier to predict. We made two tables with three columns, representing the 10 easiest movies and 10 hardest movies to have their ratings being predicted, and their associated CODs, and the associated movie names for their best predictors.
Y: We have imputed the missing ratings for movies in the dataset, because to build either simple linear regression models or multiple regression models, we need the sizes of predictor columns and outcome columns to be the same and they should not contain any NaN values. Hence, we need to either drop the rows containing NAs or impute the missing data. After observing the raw dataset, we noticed that there are a large portion of NA ratings for each movie, so dropping rows containing NA values will be a huge loss of information, as so many rows would be dropped in this case. As a result, we imputed the missing ratings with a 50/50 blend of the arithmetic mean of its associated column and the arithmetic mean of its associated row, which makes the imputed value a fair representation of the information both from its row and its column. We have plotted a histogram for the 400 CODs associated with the 400 simple linear regression models in order to observe the distribution of those COD values. When plotting this histogram using *Matplotlib*, we have set its Density=False because we do not need to normalize them for fair comparisons, as we are only analyzing a single dataset and thus having quantity on the y-axis is already good enough. We have made two tables, one for the 10 movies that are easiest to predict, and the other one for the 10 movies that are hardest to predict, because we want to observe these two kinds of movies separately instead of putting them into a single table, and in this way we can have a clear observation for the order of the 10 movies that are easiest to predict as well as the order of the 10 movies that are hardest to predict. We use the COD value to judge whether a movie is easy or hard to predict, because higher COD means higher proportion of the variation in the dependent variable that is predictable from the independent variable, and as a result, higher COD means easier to predict and lower COD means harder to predict.
F&A: *The average COD for those 400 simple linear regression is 0.3690687979237157.*
The histogram for those 400 CODs is shown below:



The table for the 10 movies that are easiest to predict is shown below:

	10 movies that are easiest to predict	associated COD value	Which movie’s ratings are the best predictor
0	Erik the Viking (1989)	0.680359	I.Q. (1994)
1	I.Q. (1994)	0.680359	Erik the Viking (1989)
2	The Lookout (2007)	0.658955	Patton (1970)
3	Patton (1970)	0.658955	The Lookout (2007)
4	Congo (1995)	0.656521	The Straight Story (1999)
5	The Straight Story (1999)	0.656521	Congo (1995)
6	The Bandit (1996)	0.652789	Best Laid Plans (1999)
7	Best Laid Plans (1999)	0.652789	The Bandit (1996)
8	Ran (1985)	0.647819	Heavy Traffic (1973)
9	Heavy Traffic (1973)	0.647819	Ran (1985)

The table for the 10 movies that are hardest to predict is shown below:

	10 movies that are hardest to predict	associated COD value	Which movie’s ratings are the best predictor
0	Avatar (2009)	0.067887	Pirates of the Caribbean: Dead Man's Chest (2006)
1	Interstellar (2014)	0.081612	Torque (2004)
2	Black Swan (2010)	0.094861	Once Upon a Time in America (1984)
3	La La Land (2016)	0.110207	The Lookout (2007)
4	Clueless (1995)	0.110905	Love Story (1970)
5	The Cabin in the Woods (2012)	0.115476	The Evil Dead (1981)
6	Grown Ups 2 (2013)	0.123269	Knight and Day (2010)
7	Back to the Future (1985)	0.129831	3000 Miles to Graceland (2001)
8	The Wolf of Wall Street (2013)	0.132793	Memento (2000)
9	Planet of the Apes (2001)	0.134899	Equilibrium (2002)

Question 2:

D: We have first dropped the rows that contain NAs in the three columns representing gender identity, sibship status, and social viewing preferences. Next, we iterate over the 10 movies that are most easily/hardest to predict, and thus we have iterated over 20 movies in total. During each iteration, we built a multiple regression model using the best predicting movie's ratings, gender identity, sibship status, and social viewing preferences as predictors(x), and using that selected movie's ratings as the outcome(y). Then calculate the predicted outcome as well as R^2, and record the value of R^2. After this, we deduct the new R^2 values that are obtained from building multiple regression models from the old R^2 values that were obtained from building simple linear regression models, and record the difference to observe how the R^2 has changed for the 10 movies that are most easily to predict and the 10 movies that are hardest to predict respectively. Finally, we made a scatterplot where the old COD(R^2) values are on the x-axis and the new COD(R^2) values are on the y-axis.

Y: We have dropped the rows that contain NAs in the columns of the three additional predictors because to build multiple regression models, we need the size of predictor columns and outcome columns to be the same and they should not contain any NA value. Hence, we need to either drop the rows containing NAs or impute the missing data. After computation we have observed that there are 24 NAs in the gender identity column, 0 NAs in both sibship status and social viewing preferences columns, so the number of NA values are very few when compared to the whole column length of 1097. As a result, there would hardly be any effect on our multiple regression models after dropping those rows that contain NAs, and we then decided to drop those rows. We decided to record the R^2 values because they are equivalent to the CODs. We decided to deduct the new R^2 values from the old R^2 values because we want to tell how the R^2 values have changed from the difference. The deduction would return a list of numbers, and the negative numbers means R^2 has increased while the positive numbers means R^2 has decreased with respect to the R^2 in question 1. We decided to observe how the R^2 has changed for the 10 movies that are easiest to predict and the 10 movies that are hardest to predict respectively because R^2 has changed in different directions for these two groups of movies.

F:The list of differences in R^2 values obtained from old R^2 values minus new R^2 values is shown below:

```
[ 0.00980779  0.01163775  0.0152538   0.016132    0.01796494  0.01626328
  0.00973873  0.01105917  0.01537402  0.01471662 -0.00585621  0.00597381
  0.00921905  0.00318956 -0.00040659 -0.00103162 -0.00104656  0.00249461
 -0.00537834  0.00246517]
```

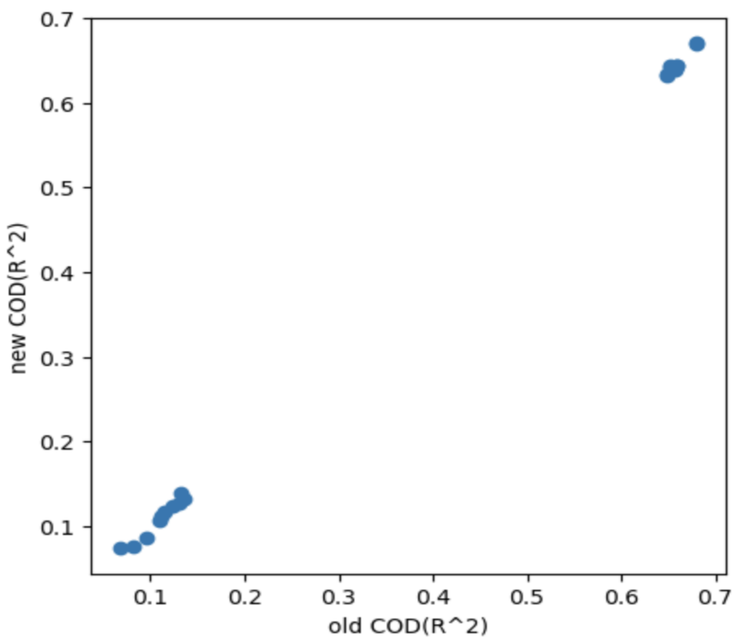
The first 10 numbers are the differences from 10 movies that are easiest to predict, and all of them are positive, meaning that for the 10 movies that are easiest to predict, the R^2 has decreased after applying multiple regression models.

The last 10 numbers are the differences from 10 movies that are hardest to predict, with 5 of them being negative and 5 of them being positive, which means that for the 10 movies that are hardest to predict, half of them have increased R^2 and half of them have decreased R^2 after applying multiple regression models.

In general, since only 5 numbers out of 20 numbers are negative, if we observe all these differences as a whole, we can conclude that for these 20 movies, most of their R^2 has slightly decreased after applying multiple regression models.

A:Given the analysis above, we can conclude that in general, most of the R^2 values have decreased slightly, but when looking at the movies that are easiest to predict and the movies that

are hardest to predict separately, we observed that for the 10 movies that are easiest to predict, their R^2 values have all decreased, and for the 10 movies that are hardest to predict, half of their R^2 values have decreased while the other half have increased.
We have also made a scatterplot where the old COD(R^2) values are on the x-axis and the new COD(R^2) values are on the y-axis, which is shown below:



Question 3:
D: I conducted Ridge Regression analysis for 30 specified movies in the middle of the COD range, using ratings from 10 randomly selected movies as predictors. The dataset was split into an 80/20 train/test partition, and hyperparameter tuning was performed using GridSearchCV to identify the optimal alpha value(among various alpha values spanning from a broad range of different orders of magnitude) for each model. Finally, we calculated the betas and RMSEs for each of these 30 movies. Also, we set random seeds for reproducibility.
Y: Ridge Regression was chosen to manage potential multicollinearity among predictors and prevent overfitting. The 80/20 split provides a robust test set while retaining sufficient training data. Hyperparameter tuning ensures the model's regularization strength is optimized for each target movie, balancing bias and variance tradeoff, and during hyperparameter tuning we chose to compare mean squared error while using 5-fold cross-validation for a balance of computational efficiency and model evaluation reliability.
F: The RMSE values for the 30 movies[Figure3.1] varied, indicating differing model accuracies. For example, '28 Days Later (2002)' had an RMSE of 0.304, while 'Star Wars: Episode II - Attack of the Clones (2002)' had a higher RMSE of 0.773. The best alpha values[Figure3.2] varied significantly among movies, reflecting the different regularization needs of each model. Generally, they are in the range of 25-80, and even with high regularization strength(compared to as we will see later in Lasso), our selected alpha values don't exclude features entirely but shrinks them towards zero. Beta coefficients were also obtained for each predictor, providing insights into their influence on the ratings of the target movies. For our 10 random movies as predictors(['The Holiday (2006)', 'Batman: The Dark Knight (2008)', 'The Silence of the Lambs (1991)', 'Ace Ventura: When Nature Calls (1995)', 'Home Alone (1990)', 'L.A. Confidential

(1997)', 'The Prestige (2006)', 'Gangs of New York (2002)', 'Traffic (2000)', 'Die Hard (1988)']), we also found the betas for each of the 30 models. For example, while predicting for the movie 'Blues Brothers 2000 (1998)', the coefficients are as follows:[0.10421785, -0.00109654, 0.05763488, 0.08847489, 0.02984069, -0.0068507 , 0.10517169, 0.09989415, 0.11897351, 0.06148169]
A: To interpret the coefficients of our model, if we look at the same example for the movie 'Blues Brothers 2000 (1998)', its beta coefficients indicate that 'Traffic (2000)' (β = 0.119), 'The Prestige (2006)' (β = 0.105), and 'Gangs of New York (2002)' (β = 0.100) had the strongest positive influence on its ratings, while 'L.A. Confidential (1997)' had a small negative impact (β = -0.007), and 'Batman: The Dark Knight (2008)' had a negligible effect (β = -0.001). Regarding the RMSE, all 30 RMSE values were below 1(around 0.5), indicating a generally good predictive performance of the models relative to the scale of the ratings. Therefore, we think the analysis reveals that Ridge Regression with 10 other random movies as predictors can decently predict movie ratings. Some limitations in our method include the assumptions of linear relationships, potential biases in the imputation method, and the randomness in selecting predictor movies.

Question 4:
D: I performed LASSO Regression for the same 30 movies as in Question 3, using the same 10 randomly selected movies as predictors. The dataset was similarly split into an 80/20 train/test partition. Hyperparameter tuning for the alpha parameter was conducted using GridSearchCV, and the best alpha values were identified for each model. RMSE values and beta coefficients were calculated for each movie. Also, we set random seed for better reproducibility.
Y: LASSO Regression was chosen for its ability to perform variable selection and regularization simultaneously, potentially enhancing model interpretiveness by reducing the number of predictors. The same predictors and 80/20 split was used for consistency with Question 3 and to maintain a balance between training and validation. Hyperparameter tuning for alpha using 5-fold GridSearchCV was crucial to optimize the balance between model complexity and performance.
F:The RMSE values for the 30 movies[Figure4.1] showed variation in model accuracy and was very similar to the Ridge Regression results. For example, '28 Days Later (2002)' had an RMSE of 0.300, and 'Star Wars: Episode II - Attack of the Clones (2002)' had an RMSE of 0.775. The best alpha values[Figure4.2] also varied across movies, indicating different levels of regularization needed. This time, we notice that the alpha values are in a much smaller range(around 10^(-3) range), and this is because L1 regularization can actually perform feature selection and shrink coefficients to exactly 0, versus in L2 where it only shrinks towards 0 while keeping the feature. This meant that for our small model with only ten features, having high alphas will indeed shrink coefficients to zero and drop them entirely, and this probably resulted in worse results in our GridSearchCV procedure and therefore smaller regularization strength was favored. Again, using the example of 'Blues Brothers 2000 (1998)', the best alpha was 0.00483, and its beta coefficients were as follows: [0.108, -0.000, 0.052, 0.089, 0.019, -0.000, 0.117, 0.108, 0.161, 0.047]. This meant that our best Lasso models for predicting Blue Brothers 2000 (1998) chose to drop the features 'Batman: The Dark Knight (2008)' and 'L.A. Confidential (1997)', and we can also see that 'Traffic (2000)' had the most significant positive influence (β = 0.161), and that, on average, an increase of one point in the rating for 'Traffic (2000)' corresponds to an increase of approximately 0.161 points in the predicted rating for 'Blues Brothers 2000 (1998)', holding all other predictors constant.

A: Ultimately, our results indicate that the models requires some degree of regularization to avoid overfitting, but not to the extent that would drastically reduce the number of features (which would happen with a higher alpha in LASSO). Our LASSO Regression models demonstrated a range of decent predictive accuracies, as indicated by the RMSE values. The best alpha values and beta coefficients vary among movies, reflecting the differing impacts of the predictors. The analysis reveals that LASSO can provide decent and valuable insights into predictor importance while it shares similar limitations as in our Ridge regression, including potential biases from imputation and assumptions of linear relationships.

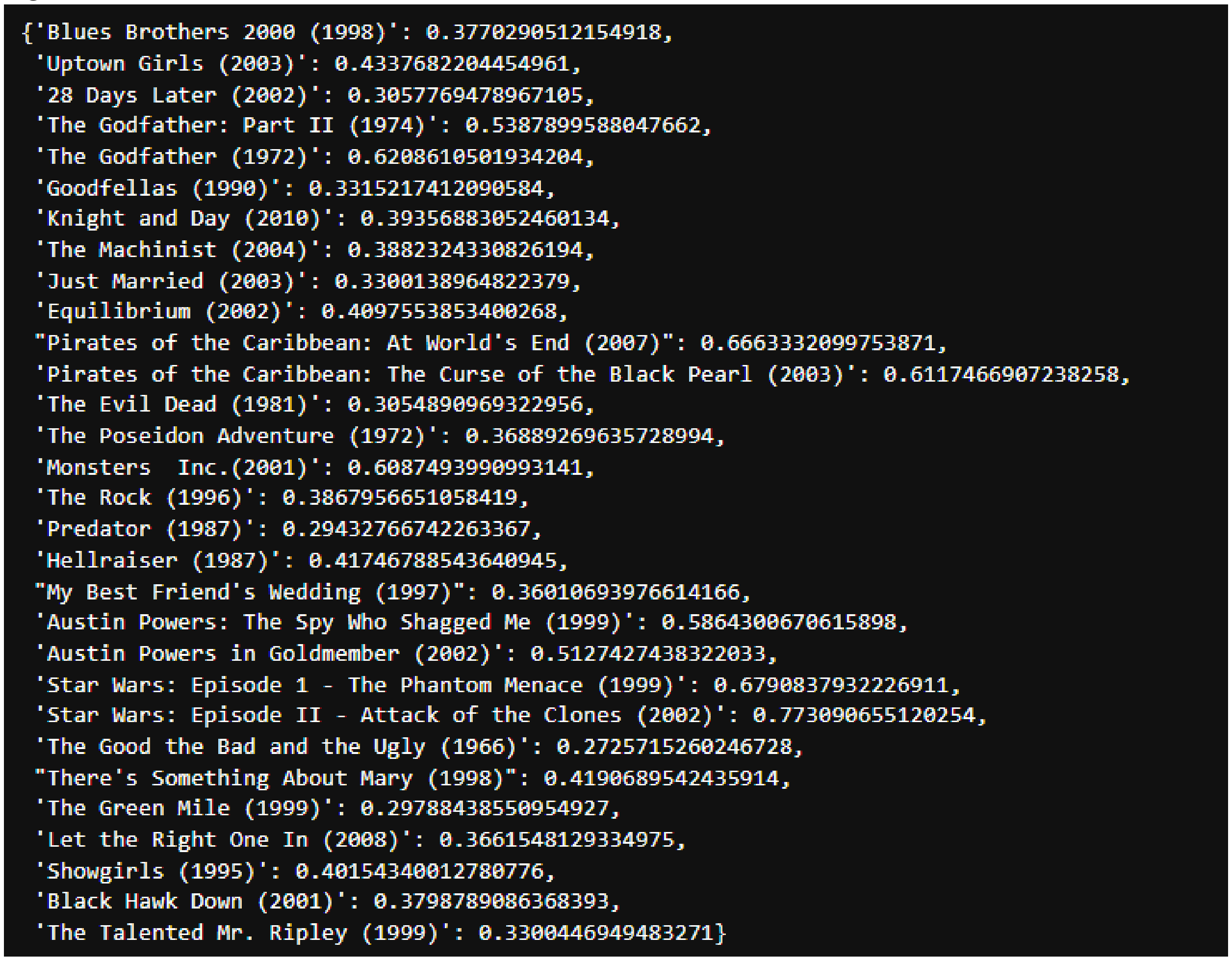
Question 5:
D: I performed logistic regression models for four middle-range movies ('Aliens (1986)', 'Meet the Parents (2000)', 'Independence Day (1996)', 'Who Framed Roger Rabbit (1988)'), using average movie enjoyment as the predictor (X). The models predicted enjoyment, coded as 1 (enjoyed) or 0 (not enjoyed) based on a median split of imputed movie ratings. Cross-validation and AUC calculations were included for model assessment.
Y: Logistic regression was chosen for its suitability in binary classification problems and its ability to provide probabilistic outputs. Using average movie enjoyment as a predictor leverages user-specific tendencies, while median splitting for target variable coding simplifies the enjoyment measure into a binary classification task. Cross-validation was employed to ensure robustness and minimize overfitting.
F: The models exhibited good predictive performance, as indicated by AUC values[Figure5.1]: 'Aliens (1986)' - 0.890, 'Meet the Parents (2000)' - 0.865, 'Independence Day (1996)' - 0.861, 'Who Framed Roger Rabbit (1988)' - 0.868. The mean beta_1 coefficients were 4.564, 3.943, 3.682, and 4.009, respectively. The beta coefficients and intercepts for the logistic regression models of the four selected movies indicate a strong relationship between the average movie ratings of users and their enjoyment of these specific movies. Specifically, for 'Aliens (1986)', the beta coefficient (beta_1) is 4.56 with an intercept of -13.65, indicating a significant positive impact of higher average ratings on the likelihood of enjoying this movie. This pattern is consistent across the other movies and the positive beta_1 values across all models suggest that users with higher average ratings are more likely to enjoy these movies. The negative intercepts indicate that, at a baseline (when average ratings are zero), the likelihood of enjoying these movies is low. The high AUC values imply good predictive performance.
A: The logistic regression models effectively predict user enjoyment for specific movies based on their average movie enjoyment, as demonstrated by high AUC values. The beta coefficients signify a positive correlation between average movie enjoyment and the probability of enjoying these particular movies. However, the approach is subject to limitations inherent in our methods, such as oversimplification and the inability to capture complex user preferences. Our reliance on average ratings as a sole predictor doesn't account for other factors that might influence enjoyment, such as movie genre preferences, mood, or contextual aspects of the viewing experience. Moreover, the use of imputed data for target variable preparation may introduce biases, as the imputation process could skew the original distribution of ratings. Therefore, the models offer great insights but should only be interpreted with consideration of these limitations.

Extra Credit:
Question: *Stress and worries are two important factors that can negatively impact our lives a lot, and they can even influence how people rate movies. We want to discuss that when being used as the single predictor, which one of those two negative features——the stress of life, or the extent of worries ——can predict movie ratings better. In this case, the stress of life is represented by the column "My life is very stressful", and the extent of worries is represented by the column "Worries a lot". For each movie, build a simple linear regression model using each of these two features as the predictor; that is, build two simple linear regression models for each movie. Then, calculate the mean of the CODs for all the linear regression models for each of the two negative features, and compare these two means. Then, list the 10 movies that are easiest to be predicted by the stress of life, and the 10 movies that are easiest to be predicted by the extent of worries, and compare their associated CODs. Finally, conclude which feature in general predicts movie ratings better.*
D: We first check the number of NAs in each of the two columns of the stress of life and the extent of worries. Then, we dropped the rows containing NAs from either of the two columns. Next, for each movie, build one simple linear regression model using the stress of life as predictor and build another simple linear regression model using the extent of worries as predictor; record their CODs and calculate the mean of each 400 CODs from the two predictors, and compare which predictor is associated with greater average COD. Then plot histograms to compare the distribution of the CODs associated with each of the two predictors. Finally, sort the COD for each predictor, and list the 10 movies that are easiest to be predicted by the stress of life, and the 10 movies that are easiest to be predicted by the extent of worries; deduct the 10 COD values of the worries predictor from the 10 COD values of the stress predictor to compare which predictor is better.
Y: We have checked the number of NAs in each of the two predictor's columns to see whether it is reasonable to drop the rows containing missing values for those two columns. We have then observed that there are only 13 NAs from the stress predictor column and 3 NAs from the worries predictor, which are very few compared with the total column lengths of 1097, so we decided to drop the rows containing NAs from either of the two columns. We decided to plot two histograms to compare the distributions of the CODs associated with each of the two predictors because comparing both the mean value and the distribution can make the comparison fairer. When plotting histograms, we have set density=True because we want to normalize the data for fair comparisons between the CODs of the stress predictor and the CODs of the worries predictor. When comparing the 10 movies that are easiest to be predicted from the two predictors, we decided to deduct the 10 COD values of the worries predictor from the 10 COD values of the stress predictor in order to easily observe which predictors have better COD values in general for the 10 most easily predicted movies: if the resulted array have much more than half of negative values, then the CODs of the worries predictor is greater, and vice versa.
F&A:
The average COD value for the stress of life feature over all movies' models is 0.0007103024647777417.
The average COD value for the extent of worries feature over all movies' models is 0.002038182919227809.
As ~0.0020 > ~0.0007, the average COD of the worries predictor is greater than that of the stress predictor. We also noticed that these two means are extremely small when compared to any of the average COD values from question 1, where the lowest COD is approximately 0.07.

Then we look at the two plotted histograms for those COD values from the stress of life predictor and the extent of worries predictor, which is shown in Figure 6.1 in the appendix.
From the two plots, we noticed that there are more higher CODs associated with the worries predictor, so the worries predictor is better than the stress predictor in general. However, most of the CODs, either from the stress predictor or the worries predictor, are very close to 0, so both predictors are not good predictors.
Finally, we calculate the difference between the CODs from the 10 movies that are easiest to be predicted by the stress of life and the 10 movies that are easiest to be predicted by the extent of worries by deducting the 10 greatest COD values of the worries predictor from the 10 greatest COD values of the stress predictor. The result is: [-0.0022276 , -0.00232893, -0.00424778, -0.00344317, -0.00361032, -0.00347668, -0.00345536, -0.00349303, -0.00337982, -0.00346848]
Since all of the numbers are negative, we can conclude the 10 greatest CODs from the worries predictor are all greater than the 10 greatest CODs from the stress predictor, which is in line with our previous conclusion that the extent of worries is a better predictor than the stress of life.
Hence, in conclusion, the extent of worries ("Worries a lot") predicts movie ratings better than the stress of life ("My life is very stressful"). Nevertheless, none of them are good predictors when being used as the single predictor in a simple linear regression model, as their greatest associated COD is very low compared to any COD from question 1.

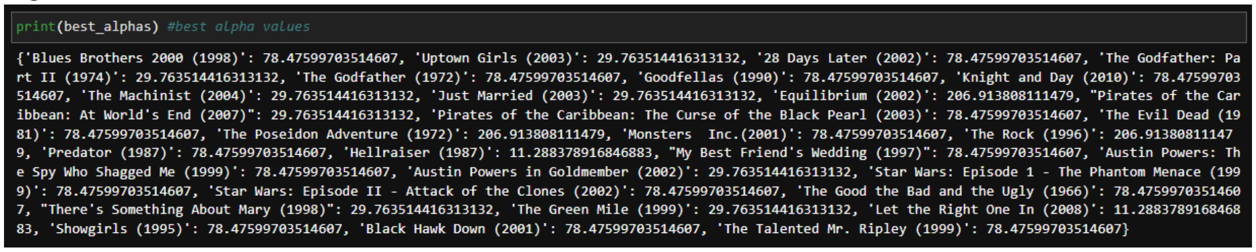
Appendix: Figures and Results

Figure3.1:



```
{'Blues Brothers 2000 (1998)': 0.3770290512154918,
 'Uptown Girls (2003)': 0.4337682204454961,
 '28 Days Later (2002)': 0.3057769478967105,
 'The Godfather: Part II (1974)': 0.5387899588047662,
 'The Godfather (1972)': 0.6208610501934204,
 'Goodfellas (1990)': 0.3315217412090584,
 'Knight and Day (2010)': 0.39356883052460134,
 'The Machinist (2004)': 0.3882324330826194,
 'Just Married (2003)': 0.3300138964822379,
 'Equilibrium (2002)': 0.4097553853400268,
 "Pirates of the Caribbean: At World's End (2007)": 0.6663332099753871,
 'Pirates of the Caribbean: The Curse of the Black Pearl (2003)': 0.6117466907238258,
 'The Evil Dead (1981)': 0.3054890969322956,
 'The Poseidon Adventure (1972)': 0.36889269635728994,
 'Monsters  Inc.(2001)': 0.6087493990993141,
 'The Rock (1996)': 0.3867956651058419,
 'Predator (1987)': 0.29432766742263367,
 'Hellraiser (1987)': 0.41746788543640945,
 "My Best Friend's Wedding (1997)": 0.36010693976614166,
 'Austin Powers: The Spy Who Shagged Me (1999)': 0.5864300670615898,
 'Austin Powers in Goldmember (2002)': 0.5127427438322033,
 'Star Wars: Episode 1 - The Phantom Menace (1999)': 0.6790837932226911,
 'Star Wars: Episode II - Attack of the Clones (2002)': 0.773090655120254,
 'The Good the Bad and the Ugly (1966)': 0.2725715260246728,
 "There's Something About Mary (1998)": 0.4190689542435914,
 'The Green Mile (1999)': 0.29788438550954927,
 'Let the Right One In (2008)': 0.3661548129334975,
 'Showgirls (1995)': 0.40154340012780776,
 'Black Hawk Down (2001)': 0.3798789086368393,
 'The Talented Mr. Ripley (1999)': 0.3300446949483271}
```

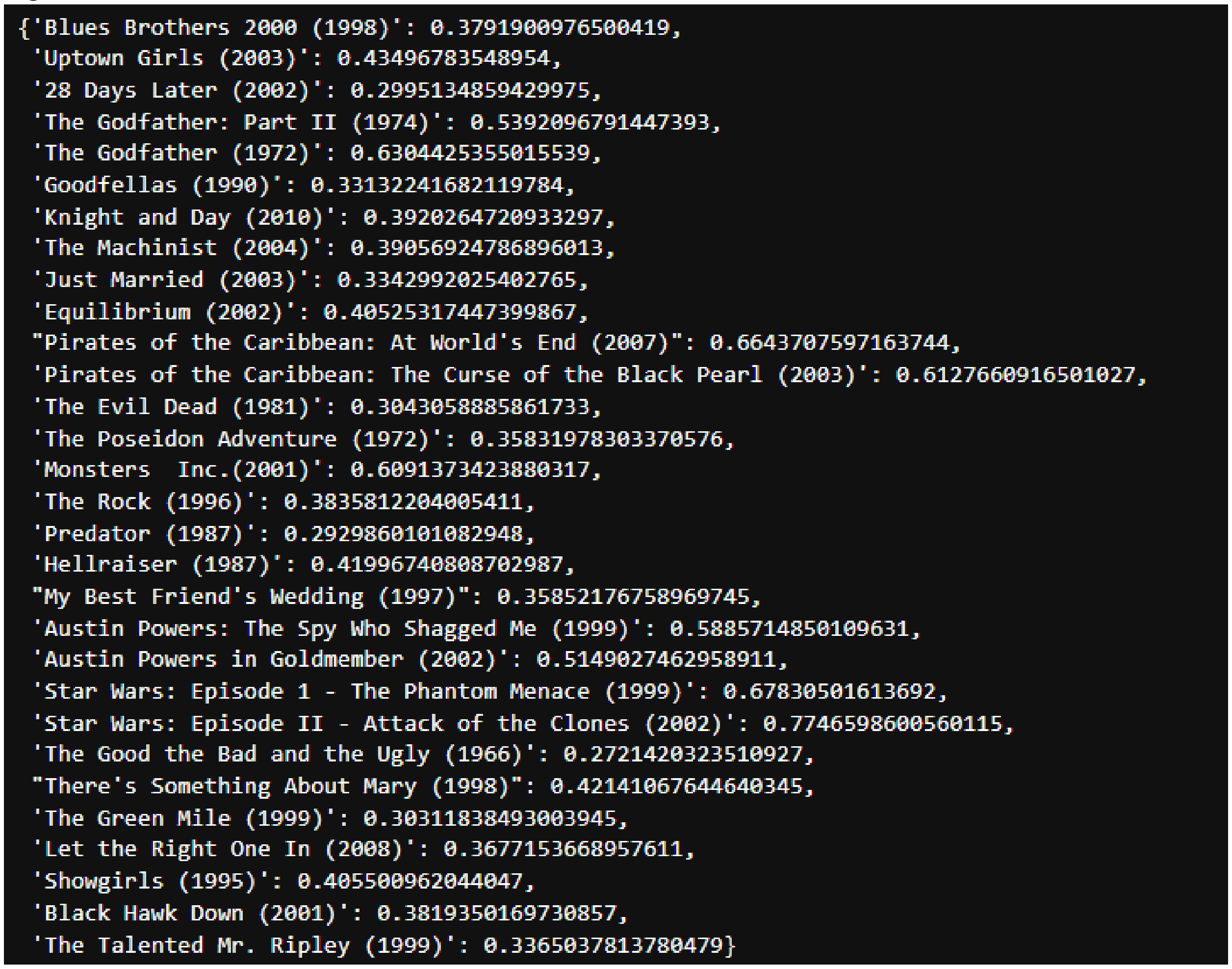
Figure3.2:



```
print(best_alphas) #best alpha values
```

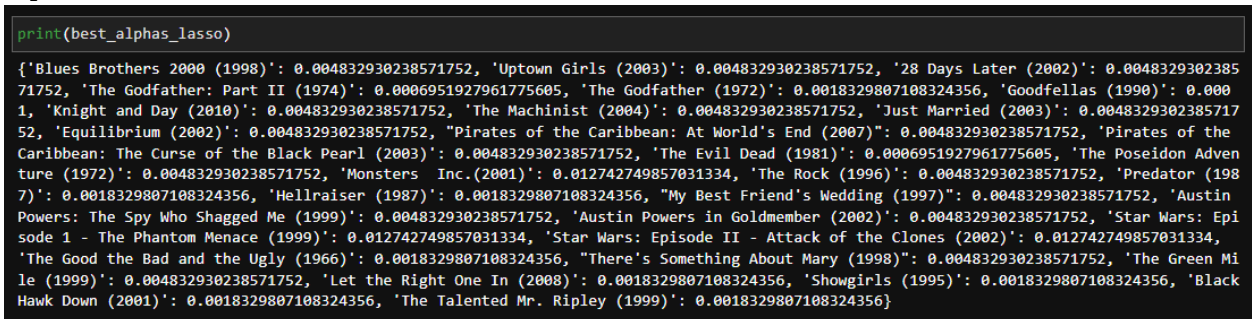
```
{'Blues Brothers 2000 (1998)': 78.47599703514607, 'Uptown Girls (2003)': 29.763514416313132, '28 Days Later (2002)': 78.47599703514607, 'The Godfather: Pa
rt II (1974)': 29.763514416313132, 'The Godfather (1972)': 78.47599703514607, 'Goodfellas (1990)': 78.47599703514607, 'Knight and Day (2010)': 78.47599703
514607, 'The Machinist (2004)': 29.763514416313132, 'Just Married (2003)': 29.763514416313132, 'Equilibrium (2002)': 206.913808111479, "Pirates of the Car
ibbean: At World's End (2007)": 29.763514416313132, 'Pirates of the Caribbean: The Curse of the Black Pearl (2003)': 78.47599703514607, 'The Evil Dead (19
81)': 78.47599703514607, 'The Poseidon Adventure (1972)': 206.913808111479, 'Monsters  Inc.(2001)': 78.47599703514607, 'The Rock (1996)': 206.9138081147
9, 'Predator (1987)': 78.47599703514607, 'Hellraiser (1987)': 11.288378916846883, "My Best Friend's Wedding (1997)": 78.47599703514607, 'Austin Powers: Th
e Spy Who Shagged Me (1999)': 78.47599703514607, 'Austin Powers in Goldmember (2002)': 29.763514416313132, 'Star Wars: Episode 1 - The Phantom Menace (199
9)': 78.47599703514607, 'Star Wars: Episode II - Attack of the Clones (2002)': 78.47599703514607, 'The Good the Bad and the Ugly (1966)': 78.4759970351460
7, "There's Something About Mary (1998)": 29.763514416313132, 'The Green Mile (1999)': 29.763514416313132, 'Let the Right One In (2008)': 11.2883789168468
83, 'Showgirls (1995)': 78.47599703514607, 'Black Hawk Down (2001)': 78.47599703514607, 'The Talented Mr. Ripley (1999)': 78.47599703514607}
```

Figure4.1:



```
{'Blues Brothers 2000 (1998)': 0.3791900976500419,
 'Uptown Girls (2003)': 0.43496783548954,
 '28 Days Later (2002)': 0.2995134859429975,
 'The Godfather: Part II (1974)': 0.5392096791447393,
 'The Godfather (1972)': 0.6304425355015539,
 'Goodfellas (1990)': 0.33132241682119784,
 'Knight and Day (2010)': 0.3920264720933297,
 'The Machinist (2004)': 0.39056924786896013,
 'Just Married (2003)': 0.3342992025402765,
 'Equilibrium (2002)': 0.40525317447399867,
 "Pirates of the Caribbean: At World's End (2007)": 0.6643707597163744,
 'Pirates of the Caribbean: The Curse of the Black Pearl (2003)': 0.6127660916501027,
 'The Evil Dead (1981)': 0.3043058885861733,
 'The Poseidon Adventure (1972)': 0.35831978303370576,
 'Monsters  Inc.(2001)': 0.6091373423880317,
 'The Rock (1996)': 0.3835812204005411,
 'Predator (1987)': 0.2929860101082948,
 'Hellraiser (1987)': 0.41996740808702987,
 "My Best Friend's Wedding (1997)": 0.35852176758969745,
 'Austin Powers: The Spy Who Shagged Me (1999)': 0.5885714850109631,
 'Austin Powers in Goldmember (2002)': 0.5149027462958911,
 'Star Wars: Episode 1 - The Phantom Menace (1999)': 0.67830501613692,
 'Star Wars: Episode II - Attack of the Clones (2002)': 0.7746598600560115,
 'The Good the Bad and the Ugly (1966)': 0.2721420323510927,
 "There's Something About Mary (1998)": 0.42141067644640345,
 'The Green Mile (1999)': 0.30311838493003945,
 'Let the Right One In (2008)': 0.3677153668957611,
 'Showgirls (1995)': 0.405500962044047,
 'Black Hawk Down (2001)': 0.3819350169730857,
 'The Talented Mr. Ripley (1999)': 0.3365037813780479}
```

Figure4.2:



```
print(best_alphas_lasso)
```

```
{'Blues Brothers 2000 (1998)': 0.004832930238571752, 'Uptown Girls (2003)': 0.004832930238571752, '28 Days Later (2002)': 0.0048329302385
71752, 'The Godfather: Part II (1974)': 0.0006951927961775605, 'The Godfather (1972)': 0.0018329807108324356, 'Goodfellas (1990)': 0.000
1, 'Knight and Day (2010)': 0.004832930238571752, 'The Machinist (2004)': 0.004832930238571752, 'Just Married (2003)': 0.0048329302385717
52, 'Equilibrium (2002)': 0.004832930238571752, "Pirates of the Caribbean: At World's End (2007)": 0.004832930238571752, 'Pirates of the
Caribbean: The Curse of the Black Pearl (2003)': 0.004832930238571752, 'The Evil Dead (1981)': 0.0006951927961775605, 'The Poseidon Adven
ture (1972)': 0.004832930238571752, 'Monsters  Inc.(2001)': 0.012742749857031334, 'The Rock (1996)': 0.004832930238571752, 'Predator (198
7)': 0.0018329807108324356, 'Hellraiser (1987)': 0.0018329807108324356, "My Best Friend's Wedding (1997)": 0.004832930238571752, 'Austin
Powers: The Spy Who Shagged Me (1999)': 0.004832930238571752, 'Austin Powers in Goldmember (2002)': 0.004832930238571752, 'Star Wars: Epi
sode 1 - The Phantom Menace (1999)': 0.012742749857031334, 'Star Wars: Episode II - Attack of the Clones (2002)': 0.012742749857031334,
'The Good the Bad and the Ugly (1966)': 0.0018329807108324356, "There's Something About Mary (1998)": 0.004832930238571752, 'The Green Mi
le (1999)': 0.004832930238571752, 'Let the Right One In (2008)': 0.0018329807108324356, 'Showgirls (1995)': 0.0018329807108324356, 'Black
Hawk Down (2001)': 0.0018329807108324356, 'The Talented Mr. Ripley (1999)': 0.0018329807108324356}
```

Figure5.1:

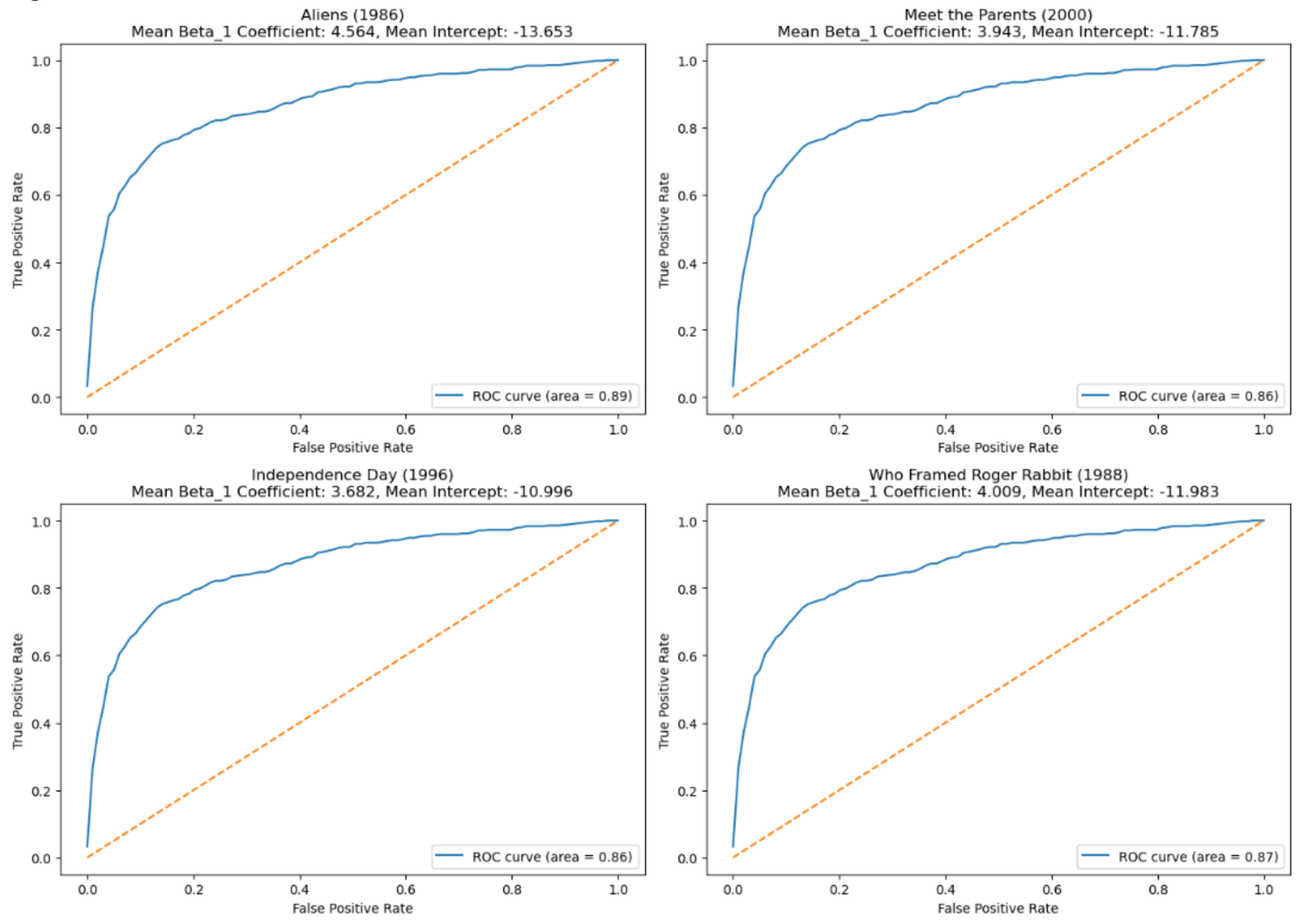
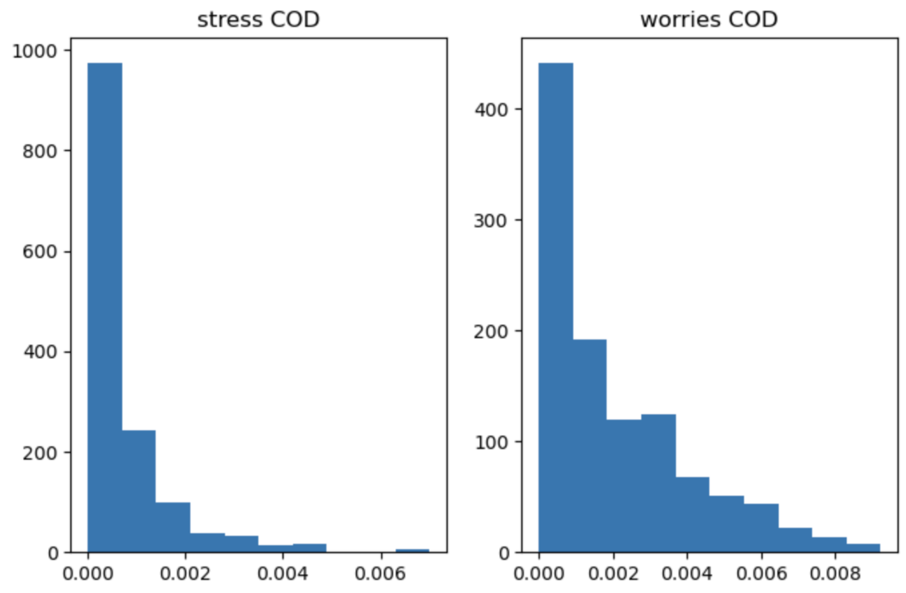


Figure6.1:



All of our codes are attached below:

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
data = pd.read_csv("movieReplicationSet.csv")
num_users_rows = 1097
num_movie_cols = 400
```

Imputation

In [ ]:
```
col_mean_dict={}
for col in range(0,num_movie_cols):
    col_mean = np.mean(data.iloc[:,col])
    col_mean_dict[col] = col_mean
row_mean_dict={}
for row in range(0,num_users_rows):
    row_mean = np.mean(data.iloc[row,:])
    row_mean_dict[row] = row_mean
```

In [ ]:
```
for row in range(0,num_users_rows):
    for col in range(0,num_movie_cols):
        if pd.isna(data.iloc[row,col]):
            imputation = (col_mean_dict[col]+ row_mean_dict[row])/2
            data.iloc[row,col] = imputation
data
```

Out[ ]:

	The Life of David Gale (2003)	Wing Commander (1999)	Django Unchained (2012)	Alien (1979)	Indiana Jones and the Last Crusade (1989)	Snatch (2000)	Rambo: First Blood Part II (1985)	Farg (199
0	2.531137	2.466043	4.000000	2.809286	3.000000	2.754308	2.638172	2.9052
1	2.535254	2.470159	1.500000	2.813402	2.848905	2.758424	2.642288	2.9093
2	2.635754	2.570660	3.136807	2.913902	2.949405	2.858924	2.742788	3.0098
3	2.468740	2.403645	2.000000	2.746888	3.000000	2.691910	2.575774	2.8428
4	2.379504	2.314410	3.500000	2.657652	0.500000	2.602674	0.500000	1.0000
...	...	...	...	...	...	...	...	...
1092	2.677848	2.612753	3.178901	2.955996	3.500000	2.901018	2.784882	3.0519
1093	3.000000	4.000000	3.450559	3.227654	4.000000	4.000000	2.500000	3.3236
1094	2.672533	2.607438	3.173586	2.950681	2.986184	2.895703	2.779567	3.5000
1095	2.743089	2.677995	3.244142	3.021237	3.056740	2.966259	2.850124	3.1172
1096	2.610232	2.545138	4.000000	2.888381	2.500000	2.833403	2.717267	3.0000

1097 rows × 477 columns

question 1

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
from tqdm import tqdm
```

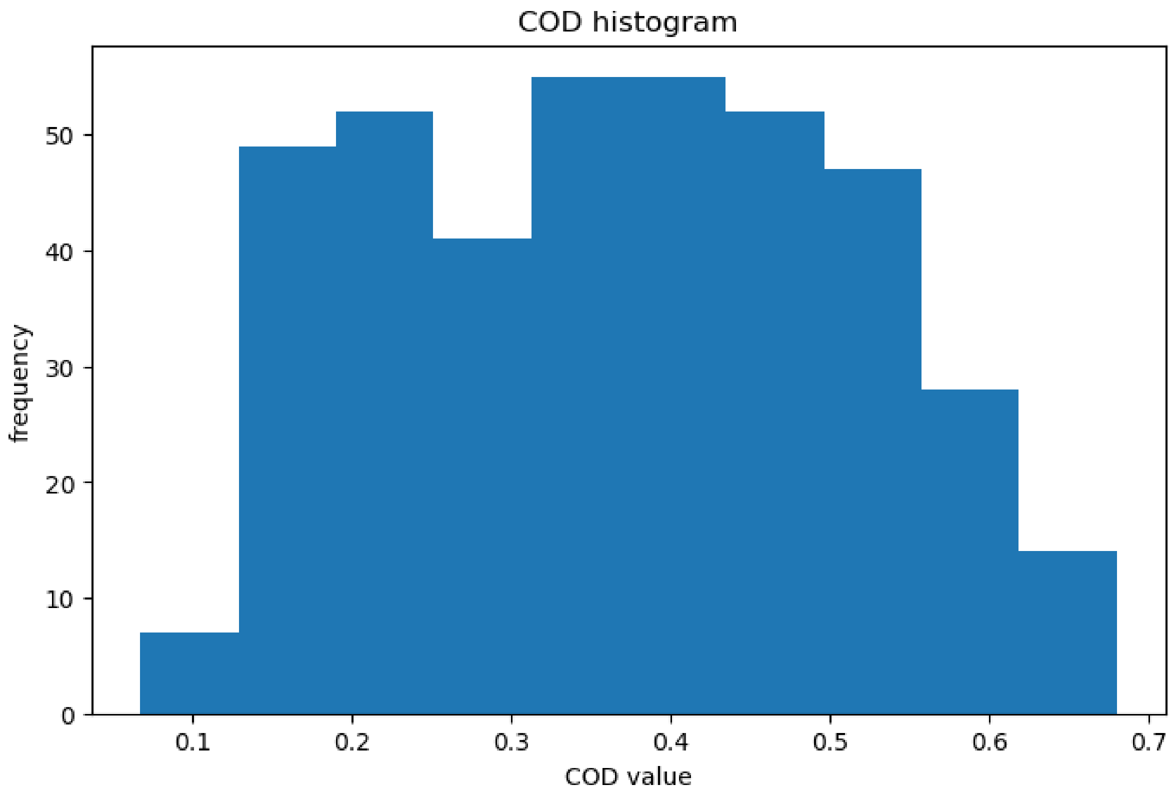
In [ ]:
```
movie_COD_other = {}
for y_index in tqdm(range(0,num_movie_cols)):
    y_vals = data.iloc[:,y_index].values #this is the y (ratings for t
his movie)
    movie_name = data.columns[y_index]#this is the movie name
    movie_COD_other[movie_name]=[float("-inf"),""]#initialize in dict
    for x_index in range(0,num_movie_cols):
        if x_index!=y_index:#use all other movies to predict
            x_vals = data.iloc[:,x_index].values#the x values (ratings
for other movies)
            x_vals = x_vals.reshape(-1,1)
            reg = LinearRegression()
            reg.fit(x_vals,y_vals)
            y_hat = reg.predict(x_vals)
            r2 = r2_score(y_vals,y_hat)#COD = r^2
            if r2>movie_COD_other[movie_name][0]:#consistently finding
greater COD
                other_movie_name = data.columns[x_index]
                movie_COD_other[movie_name]=[r2, other_movie_name] #up
date COD and its associated movie
```

```
100%|██████████| 400/400 [03:34<00:00,  1.86it/s]
```

In [ ]:
```
#report the average of those 400 models:
CODs = [movie_COD_other[i][0] for i in movie_COD_other]
np.mean(CODs)
```

Out[ ]:
```
0.3690687979237157
```

```
In [ ]: #plot histogram
        plt.figure(figsize=(8,5))
        plt.hist(CODs)
        plt.title('COD histogram')
        plt.xlabel("COD value")
        plt.ylabel("frequency")
        plt.show()
```



```
In [ ]: top_best = sorted(movie_COD_other.items(), key=lambda i: i[1][0],rever
        se=True)
        temp_best = top_best[0:10]
        temp_best
```

```
Out[ ]: [('Erik the Viking (1989)', [0.6803586481291373, 'I.Q. (1994)']),
         ('I.Q. (1994)', [0.6803586481291373, 'Erik the Viking (1989)']),
         ('The Lookout (2007)', [0.6589551291567151, 'Patton (1970)']),
         ('Patton (1970)', [0.6589551291567151, 'The Lookout (2007)']),
         ('Congo (1995)', [0.656520890202376, 'The Straight Story (1999)']),
         ('The Straight Story (1999)', [0.656520890202376, 'Congo (1995)']),
         ('The Bandit (1996)', [0.6527893662798822, 'Best Laid Plans (199
        9)']),
         ('Best Laid Plans (1999)', [0.652789366279882, 'The Bandit (1996)']),
         ('Ran (1985)', [0.6478194879446582, 'Heavy Traffic (1973)']),
         ('Heavy Traffic (1973)', [0.6478194879446582, 'Ran (1985)'])]
```

In [ ]:
```
top_best_list = [[i[0],i[1][0],i[1][1]] for i in temp_best ]
df_best = pd.DataFrame(top_best_list, columns = ['10 movies that are e
asiest to predict','associated COD value','Which movie’s ratings are t
he best predictor'])
df_best
```

Out[ ]:

	10 movies that are easiest to predict	associated COD value	Which movie’s ratings are the best predictor
0	Erik the Viking (1989)	0.680359	I.Q. (1994)
1	I.Q. (1994)	0.680359	Erik the Viking (1989)
2	The Lookout (2007)	0.658955	Patton (1970)
3	Patton (1970)	0.658955	The Lookout (2007)
4	Congo (1995)	0.656521	The Straight Story (1999)
5	The Straight Story (1999)	0.656521	Congo (1995)
6	The Bandit (1996)	0.652789	Best Laid Plans (1999)
7	Best Laid Plans (1999)	0.652789	The Bandit (1996)
8	Ran (1985)	0.647819	Heavy Traffic (1973)
9	Heavy Traffic (1973)	0.647819	Ran (1985)

In [ ]:
```
top_worst = sorted(movie_COD_other.items(),key = lambda i: i[1][0])
temp_worst = top_worst[0:10]
temp_worst
```

Out[ ]:
```
[('Avatar (2009)',
  [0.06788740393778625, "Pirates of the Caribbean: Dead Man's Chest (2
006)"]),
 ('Interstellar (2014)', [0.08161205184687814, 'Torque (2004)']),
 ('Black Swan (2010)',
  [0.09486062776297288, 'Once Upon a Time in America (1984)']),
 ('La La Land (2016)', [0.11020663183772739, 'The Lookout (2007)']),
 ('Clueless (1995)', [0.11090474997229227, 'Love Story (1970)']),
 ('The Cabin in the Woods (2012)',
  [0.11547617566324209, 'The Evil Dead (1981)']),
 ('Grown Ups 2 (2013)', [0.12326924270646411, 'Knight and Day (201
0)']),
 ('Back to the Future (1985)',
  [0.1298307779130956, '3000 Miles to Graceland (2001)']),
 ('The Wolf of Wall Street (2013)', [0.13279266266282352, 'Memento (20
00)']),
 ('Planet of the Apes (2001)', [0.1348988779460075, 'Equilibrium (200
2)'])]
```

In [ ]:
```
top_worst_list = [[i[0],i[1][0],i[1][1]] for i in temp_worst]
df_worst = pd.DataFrame(top_worst_list, columns = ['10 movies that are hardest to predict','associated COD value','Which movie’s ratings are the best predictor'])
df_worst
```

Out[ ]:

	10 movies that are hardest to predict	associated COD value	Which movie’s ratings are the best predictor
0	Avatar (2009)	0.067887	Pirates of the Caribbean: Dead Man's Chest (2006)
1	Interstellar (2014)	0.081612	Torque (2004)
2	Black Swan (2010)	0.094861	Once Upon a Time in America (1984)
3	La La Land (2016)	0.110207	The Lookout (2007)
4	Clueless (1995)	0.110905	Love Story (1970)
5	The Cabin in the Woods (2012)	0.115476	The Evil Dead (1981)
6	Grown Ups 2 (2013)	0.123269	Knight and Day (2010)
7	Back to the Future (1985)	0.129831	3000 Miles to Graceland (2001)
8	The Wolf of Wall Street (2013)	0.132793	Memento (2000)
9	Planet of the Apes (2001)	0.134899	Equilibrium (2002)

Question 2

In [ ]:
```
#calculate number of NAs
one = data.columns[474]
nulls = [i for i in data[one].isna() if i==True]
len(nulls)
```

Out[ ]:
```
24
```

In [ ]:
```
#drop the rows containing NAs in the three given columns
subset = data.columns[474:477]
nadropped_df = data.dropna(subset = subset, inplace=False)
```

In [ ]:
```
gender_identity_vals = nadropped_df.iloc[:,474].values.reshape(-1,1)
sibship_vals = nadropped_df.iloc[:,475].values.reshape(-1,1)
social_view_vals = nadropped_df.iloc[:,476].values.reshape(-1,1)
```

In [ ]:
```
#build multiple lr model for movies with best CODs
COD_list_best = []
for bests in temp_best:
    best_predictor = nadropped_df[bests[1][1]].values.reshape(-1,1)
    x = np.concatenate((gender_identity_vals, sibship_vals, social_vie
w_vals,best_predictor), axis=1)
    y = nadropped_df[bests[0]].values.reshape(-1,1)
    reg = LinearRegression().fit(x,y)
    y_hat = reg.predict(x)
    r2 = r2_score(y,y_hat)
    COD_list_best.append(r2)
COD_list_best
```

Out[ ]:
```
[0.6705508584276447,
 0.6687209018918612,
 0.6437013269706906,
 0.6428231266182935,
 0.6385559520236557,
 0.6402576126993955,
 0.6430506320418278,
 0.6417301965939766,
 0.6324454638112478,
 0.633102863165075]
```

In [ ]:
```
#build multiple lr model for movies with lowest CODs
COD_list_worst = []
for worsts in temp_worst:
    best_predictor = nadropped_df[worsts[1][1]].values.reshape(-1,1)
    x = np.concatenate((gender_identity_vals, sibship_vals, social_vie
w_vals,best_predictor), axis=1)
    y = nadropped_df[worsts[0]].values.reshape(-1,1)
    reg = LinearRegression().fit(x,y)
    y_hat = reg.predict(x)
    r2 = r2_score(y,y_hat)
    COD_list_worst.append(r2)
COD_list_worst
```

Out[ ]:
```
[0.07374361632107052,
 0.07563823698608574,
 0.08564157960716079,
 0.10701706758525897,
 0.111311339751342,
 0.11650779443424453,
 0.12431579982363905,
 0.12733616747013843,
 0.13817100249059577,
 0.13243370404334054]
```

```
In [ ]: new_COD_twenty = []
        new_COD_twenty.extend(COD_list_best)
        new_COD_twenty.extend(COD_list_worst)
        new_COD_twenty
```

```
Out[ ]: [0.6705508584276447,
         0.6687209018918612,
         0.6437013269706906,
         0.642823126618293５,
         0.6385559520236557,
         0.6402576126993955,
         0.6430506320418278,
         0.6417301965939766,
         0.6324454638112478,
         0.633102863165075,
         0.07374361632107052,
         0.07563823698608574,
         0.08564157960716079,
         0.10701706758525897,
         0.111311339751342,
         0.11650779443424453,
         0.12431579982363905,
         0.12733616747013843,
         0.13817100249059577,
         0.13243370404334054]
```

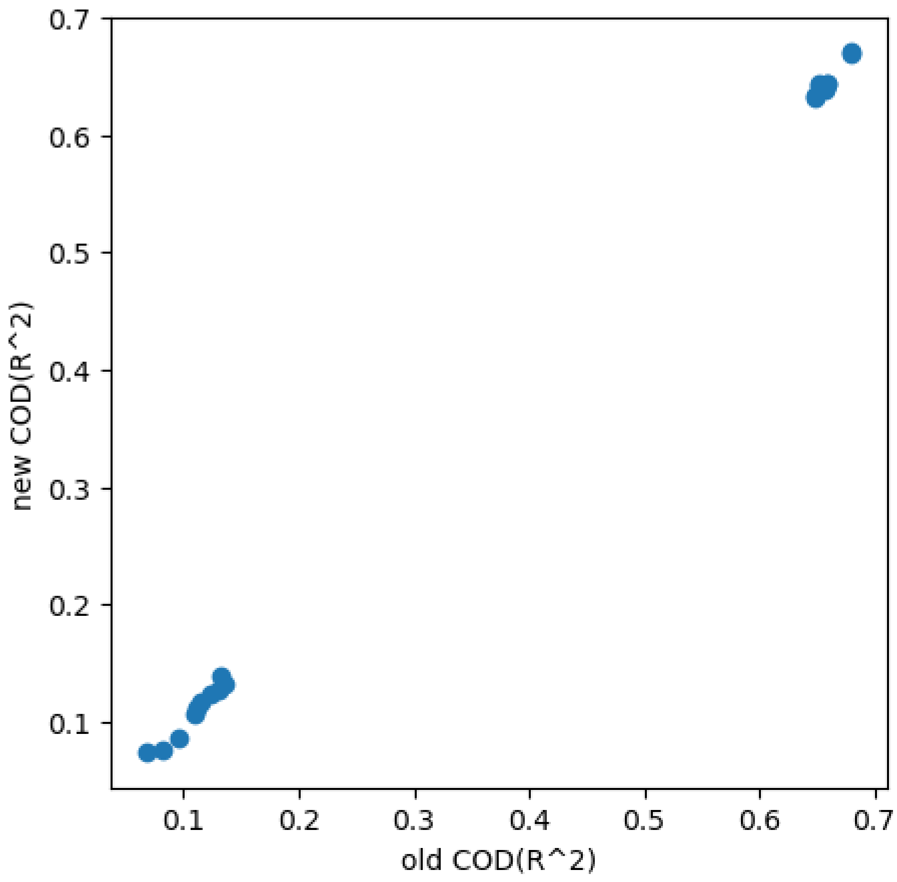
```
In [ ]: old_COD_twenty = []
        old_COD_twenty.extend([i[1][0] for i in temp_best])
        old_COD_twenty.extend([i[1][0] for i in temp_worst])
        old_COD_twenty
```

```
Out[ ]: [0.6803586481291373,
         0.6803586481291373,
         0.6589551291567151,
         0.6589551291567151,
         0.656520890202376,
         0.656520890202376,
         0.6527893662798822,
         0.652789366279882,
         0.6478194879446582,
         0.6478194879446582,
         0.06788740393778625,
         0.08161205184687814,
         0.09486062776297288,
         0.11020663183772739,
         0.11090474997229227,
         0.11547617566324209,
         0.12326924270646411,
         0.1298307779130956,
         0.13279266266282352,
         0.13489887779460075]
```

In [ ]:
```
diff = np.array(old_COD_twenty)-np.array(new_COD_twenty)
print(diff)
```

```
[ 0.00980779  0.01163775  0.0152538   0.016132    0.01796494  0.016263
28
  0.00973873  0.01105917  0.01537402  0.01471662 -0.00585621  0.005973
81
  0.00921905  0.00318956 -0.00040659 -0.00103162 -0.00104656  0.002494
61
 -0.00537834  0.00246517]
```

In [ ]:
```
plt.figure(figsize=(5,5))
plt.scatter(old_COD_twenty,new_COD_twenty)
plt.xlabel('old COD(R^2)')
plt.ylabel('new COD(R^2)')
plt.show()
```



Question 3

In [ ]:
```
#finding the middle most 30 movies
sorted_movies = sorted(movie_COD_other.items(), key=lambda x: x[1][0])
middle_index = len(sorted_movies) // 2
middle_movies = sorted_movies[max(0, middle_index - 15): middle_index
+ 15]
middle_movie_names = [movie[0] for movie in middle_movies]
print(middle_movie_names)
```

```
['Blues Brothers 2000 (1998)', 'Uptown Girls (2003)', '28 Days Later
(2002)', 'The Godfather: Part II (1974)', 'The Godfather (1972)', 'Goo
dfellas (1990)', 'Knight and Day (2010)', 'The Machinist (2004)', 'Jus
t Married (2003)', 'Equilibrium (2002)', "Pirates of the Caribbean: At
World's End (2007)", 'Pirates of the Caribbean: The Curse of the Black
Pearl (2003)', 'The Evil Dead (1981)', 'The Poseidon Adventure (197
2)', 'Monsters  Inc.(2001)', 'The Rock (1996)', 'Predator (1987)', 'He
llraiser (1987)', "My Best Friend's Wedding (1997)", 'Austin Powers: T
he Spy Who Shagged Me (1999)', 'Austin Powers in Goldmember (2002)',
'Star Wars: Episode 1 - The Phantom Menace (1999)', 'Star Wars: Episod
e II - Attack of the Clones (2002)', 'The Good the Bad and the Ugly (1
966)', "There's Something About Mary (1998)", 'The Green Mile (1999)',
'Let the Right One In (2008)', 'Showgirls (1995)', 'Black Hawk Down (2
001)', 'The Talented Mr. Ripley (1999)']
```

In [ ]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
import random

random.seed(0)
np.random.seed(0)
# selecting 10 random movies
all_movie_names = data.columns[:400].tolist()
input_movie_names = random.sample([movie for movie in all_movie_names
if movie not in middle_movie_names], 10)
X = data[input_movie_names]
Y = data[middle_movie_names]
# 80/20 train/test split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.
2, random_state=0)
# Hyperparameter Tuning
parameters = {'alpha': np.logspace(-4, 4, 20)}
ridge_reg = Ridge()
grid_search = GridSearchCV(ridge_reg, parameters, scoring='neg_mean_sq
uared_error', cv=5)
rmse_results = {}
best_alphas = {}
beta_coefficients = {}
for movie in middle_movie_names:
    grid_search.fit(X_train, Y_train[movie])
    best_model = grid_search.best_estimator_
    best_alphas[movie] = best_model.alpha
    beta_coefficients[movie] = best_model.coef_
    y_pred = best_model.predict(X_test)
    rmse = mean_squared_error(Y_test[movie], y_pred, squared=False)
    rmse_results[movie] = rmse
print(input_movie_names)
rmse_results
```

```
['The Holiday (2006)', 'Batman: The Dark Knight (2008)', 'The Silence
of the Lambs (1991)', 'Ace Ventura: When Nature Calls (1995)', 'Home A
lone (1990)', 'L.A. Confidential (1997)', 'The Prestige (2006)', 'Gang
s of New York (2002)', 'Traffic (2000)', 'Die Hard (1988)']
```

```
Out[ ]: {'Blues Brothers 2000 (1998)': 0.3770290512154918,
         'Uptown Girls (2003)': 0.4337682204454961,
         '28 Days Later (2002)': 0.3057769478967105,
         'The Godfather: Part II (1974)': 0.5387899588047661,
         'The Godfather (1972)': 0.6208610501934204,
         'Goodfellas (1990)': 0.33152174120905836,
         'Knight and Day (2010)': 0.39356883052460134,
         'The Machinist (2004)': 0.3882324330826194,
         'Just Married (2003)': 0.3300138964822379,
         'Equilibrium (2002)': 0.4097553853400268,
         "Pirates of the Caribbean: At World's End (2007)": 0.666333209975387
        1,
         'Pirates of the Caribbean: The Curse of the Black Pearl (2003)': 0.61
        17466907238258,
         'The Evil Dead (1981)': 0.30548909693229553,
         'The Poseidon Adventure (1972)': 0.36889269635728994,
         'Monsters  Inc.(2001)': 0.6087493990993141,
         'The Rock (1996)': 0.38679566510584196,
         'Predator (1987)': 0.29432766742263367,
         'Hellraiser (1987)': 0.4174678854364095,
         "My Best Friend's Wedding (1997)": 0.36010693976614166,
         'Austin Powers: The Spy Who Shagged Me (1999)': 0.5864300670615898,
         'Austin Powers in Goldmember (2002)': 0.5127427438322033,
         'Star Wars: Episode 1 - The Phantom Menace (1999)': 0.679083793222691
        1,
         'Star Wars: Episode II - Attack of the Clones (2002)': 0.773090655120
        2539,
         'The Good the Bad and the Ugly (1966)': 0.2725715260246728,
         "There's Something About Mary (1998)": 0.4190689542435914,
         'The Green Mile (1999)': 0.2978843855095492,
         'Let the Right One In (2008)': 0.36615481293349744,
         'Showgirls (1995)': 0.40154340012780776,
         'Black Hawk Down (2001)': 0.3798789086368393,
         'The Talented Mr. Ripley (1999)': 0.3300446949483271}
```

In [ ]:
```
print(best_alphas)
beta_coefficients
```

```
{'Blues Brothers 2000 (1998)': 78.47599703514607, 'Uptown Girls (200
3)': 29.763514416313132, '28 Days Later (2002)': 78.47599703514607, 'T
he Godfather: Part II (1974)': 29.763514416313132, 'The Godfather (197
2)': 78.47599703514607, 'Goodfellas (1990)': 78.47599703514607, 'Knigh
t and Day (2010)': 78.47599703514607, 'The Machinist (2004)': 29.76351
4416313132, 'Just Married (2003)': 29.763514416313132, 'Equilibrium (2
002)': 206.913808111479, "Pirates of the Caribbean: At World's End (20
07)": 29.763514416313132, 'Pirates of the Caribbean: The Curse of the
Black Pearl (2003)': 78.47599703514607, 'The Evil Dead (1981)': 78.475
99703514607, 'The Poseidon Adventure (1972)': 206.913808111479, 'Monst
ers  Inc.(2001)': 78.47599703514607, 'The Rock (1996)': 206.9138081114
79, 'Predator (1987)': 78.47599703514607, 'Hellraiser (1987)': 11.2883
78916846883, "My Best Friend's Wedding (1997)": 78.47599703514607, 'Au
stin Powers: The Spy Who Shagged Me (1999)': 78.47599703514607, 'Austi
n Powers in Goldmember (2002)': 29.763514416313132, 'Star Wars: Episod
e 1 - The Phantom Menace (1999)': 78.47599703514607, 'Star Wars: Episo
de II - Attack of the Clones (2002)': 78.47599703514607, 'The Good the
Bad and the Ugly (1966)': 78.47599703514607, "There's Something About
Mary (1998)": 29.763514416313132, 'The Green Mile (1999)': 29.76351441
6313132, 'Let the Right One In (2008)': 11.288378916846883, 'Showgirls
(1995)': 78.47599703514607, 'Black Hawk Down (2001)': 78.4759970351460
7, 'The Talented Mr. Ripley (1999)': 78.47599703514607}
```

```
Out[ ]: {'Blues Brothers 2000 (1998)': array([ 0.10421785, -0.00109654,  0.057
        63488,  0.08847489,  0.02984069,
               -0.0068507 ,  0.10517169,  0.09989415,  0.11897351,  0.0614816
        9]),
         'Uptown Girls (2003)': array([0.11303273, 0.01009508, 0.01470929, 0.0
        9858094, 0.07913261,
               0.12820935, 0.08879813, 0.12577867, 0.19540011, 0.03651784]),
         '28 Days Later (2002)': array([0.08610734, 0.04499372, 0.04619875, 0.
        06569097, 0.02973814,
               0.13853559, 0.00089156, 0.09445855, 0.08777451, 0.09048686]),
         'The Godfather: Part II (1974)': array([-0.04351937,  0.05944793,  0.
        15845424,  0.02732277,  0.06106063,
                0.09909288,  0.11719082,  0.0295088 ,  0.1587471 ,  0.0954063
        ]),
         'The Godfather (1972)': array([-0.02468894,  0.05740419,  0.11707037,
        0.02261946,  0.08282087,
                0.16147627,  0.14168522,  0.07441065,  0.12420171,  0.0530563
        4]),
         'Goodfellas (1990)': array([0.08538897, 0.07665281, 0.07162591, 0.038
        98504, 0.03020231,
               0.12478127, 0.07429231, 0.09284991, 0.13964946, 0.10329866]),
         'Knight and Day (2010)': array([0.11660174, 0.03633087, 0.04680581,
        0.0779302 , 0.03236207,
               0.12004341, 0.01433673, 0.06937991, 0.13967294, 0.04476494]),
         'The Machinist (2004)': array([ 0.04702851,  0.00846952,  0.07002494,
        0.01938729, -0.00209036,
                0.20174103,  0.13021884,  0.10683732,  0.06736649,  0.1261770
        7]),
         'Just Married (2003)': array([ 0.14518875, -0.01131334, -0.02425364,
        0.13564108,  0.02546483,
                0.08664989,  0.09900319,  0.07962338,  0.22663627,  0.1031504
        4]),
         'Equilibrium (2002)': array([0.03683852, 0.04016067, 0.04876588, 0.05
        261222, 0.04439   ,
               0.05934672, 0.03148587, 0.02946297, 0.03793404, 0.03825246]),
         "Pirates of the Caribbean: At World's End (2007)": array([ 0.1625048
        1,  0.14295827,  0.00749683,  0.14314977,  0.17208803,
               -0.05884938,  0.03614583,  0.06211753,  0.2028286 ,  0.0570138
        9]),
         'Pirates of the Caribbean: The Curse of the Black Pearl (2003)': arra
        y([0.156544  , 0.14450383, 0.01474639, 0.05149022, 0.12707103,
               0.07507885, 0.04161142, 0.07303207, 0.10243162, 0.09169824]),
         'The Evil Dead (1981)': array([0.05919547, 0.04774772, 0.07125641, 0.
        06530839, 0.00851411,
               0.08539003, 0.07750912, 0.15228052, 0.09965365, 0.08978918]),
         'The Poseidon Adventure (1972)': array([0.06039747, 0.00612352, 0.034
        64006, 0.07038818, 0.02048557,
               0.05533202, 0.04297327, 0.02252837, 0.05986089, 0.05270561]),
         'Monsters  Inc.(2001)': array([0.10825098, 0.10439449, 0.06454912, 0.
        02768477, 0.17835542,
               0.04560116, 0.11046297, 0.0479184 , 0.11022299, 0.11864688]),
         'The Rock (1996)': array([0.03380806, 0.01994487, 0.05755392, 0.06328
        989, 0.03057539,
               0.05655795, 0.0499505 , 0.08172027, 0.07937254, 0.07209775]),
         'Predator (1987)': array([0.0520309 , 0.01238676, 0.06070122, 0.06706
        218, 0.01547755,
               0.15992772, 0.12417913, 0.06793996, 0.09349456, 0.12291642]),
 'Hellraiser (1987)': array([ 0.04955692,  0.05251974,  0.00808925,
0.01491566, -0.00040776,
         0.28725143,  0.04198557,  0.04163894,  0.14932608,  0.0830991
9]),
 "My Best Friend's Wedding (1997)": array([0.13226588, 0.02265893, 0.0
4225103, 0.11175475, 0.02704748,
        0.09467783, 0.0963202 , 0.04889879, 0.07405239, 0.06937926]),
 'Austin Powers: The Spy Who Shagged Me (1999)': array([0.09808085, 0.
08175132, 0.02153042, 0.17144351, 0.09202925,
        0.11008717, 0.06303284, 0.0695189 , 0.1437994 , 0.06229336]),
 'Austin Powers in Goldmember (2002)': array([0.02956455, 0.06213025,
0.06377986, 0.18414144, 0.05958894,
        0.07407391, 0.04681634, 0.07182155, 0.23489105, 0.08925013]),
 'Star Wars: Episode 1 - The Phantom Menace (1999)': array([ 0.0562767
1,  0.05367114,  0.00162433,  0.09424419,  0.10126897,
         0.11788438,  0.09707192,  0.09184598,  0.1659522 , -0.0300358
]),
 'Star Wars: Episode II - Attack of the Clones (2002)': array([ 0.0045
3572, -0.00552246,  0.04564772,  0.0959494 ,  0.07473331,
         0.16277778,  0.03141245,  0.0510813 ,  0.12326227, -0.0054771
4]),
 'The Good the Bad and the Ugly (1966)': array([0.07323144, 0.0391016
8, 0.05606972, 0.066691  , 0.01735256,
        0.08566119, 0.06974708, 0.18346186, 0.13879325, 0.10578017]),
 "There's Something About Mary (1998)": array([ 0.12893435, -0.0169544
,  0.03334583,  0.10641801,  0.04100056,
         0.18704253,  0.08585868,  0.03747874,  0.21530435,  0.0913448
2]),
 'The Green Mile (1999)': array([0.04804208, 0.01923786, 0.02871344,
0.03901426, 0.01723793,
        0.27672258, 0.10320759, 0.08995717, 0.12230395, 0.0604801 ]),
 'Let the Right One In (2008)': array([ 0.07049707,  0.01178787,  0.06
079484,  0.02873911,  0.01925573,
         0.15493692, -0.0055545 ,  0.16159093,  0.17595095,  0.0581923
7]),
 'Showgirls (1995)': array([0.08080137, 0.00519615, 0.03744794, 0.0758
7129, 0.02145775,
        0.0680024 , 0.06462214, 0.06388871, 0.11103735, 0.05091305]),
 'Black Hawk Down (2001)': array([0.0540913 , 0.05475109, 0.06151908,
0.07767804, 0.04115051,
        0.10994027, 0.10059684, 0.04858241, 0.10854518, 0.06054923]),
 'The Talented Mr. Ripley (1999)': array([0.0443515 , 0.04226704, 0.08
342898, 0.11139676, 0.02335282,
        0.07087056, 0.03516071, 0.10841018, 0.14621125, 0.06023353])}
```

Question 4

```
In [ ]: from sklearn.linear_model import Lasso
        random.seed(0)
        np.random.seed(0)
        # Hyperparameter Tuning
        lasso_reg = Lasso()
        grid_search_lasso = GridSearchCV(lasso_reg, parameters, scoring='neg_m
        ean_squared_error', cv=5)

        # LASSO
        rmse_results_lasso = {}
        best_alphas_lasso = {}
        beta_coefficients_lasso = {}
        for movie in middle_movie_names:
            grid_search_lasso.fit(X_train, Y_train[movie])
            best_model_lasso = grid_search_lasso.best_estimator_
            best_alphas_lasso[movie] = best_model_lasso.alpha
            beta_coefficients_lasso[movie] = best_model_lasso.coef_
            y_pred_lasso = best_model_lasso.predict(X_test)
            rmse_lasso = mean_squared_error(Y_test[movie], y_pred_lasso, squar
        ed=False)
            rmse_results_lasso[movie] = rmse_lasso

        rmse_results_lasso
```

```
Out[ ]: {'Blues Brothers 2000 (1998)': 0.3791900976500419,
         'Uptown Girls (2003)': 0.43496783548954004,
         '28 Days Later (2002)': 0.2995134859429975,
         'The Godfather: Part II (1974)': 0.5392096791447394,
         'The Godfather (1972)': 0.6304425355015539,
         'Goodfellas (1990)': 0.33132241682119784,
         'Knight and Day (2010)': 0.3920264720933297,
         'The Machinist (2004)': 0.3905692478689602,
         'Just Married (2003)': 0.3342992025402765,
         'Equilibrium (2002)': 0.4052531744739986,
         "Pirates of the Caribbean: At World's End (2007)": 0.664370759716374
        4,
         'Pirates of the Caribbean: The Curse of the Black Pearl (2003)': 0.61
        27660916501027,
         'The Evil Dead (1981)': 0.30430588858617325,
         'The Poseidon Adventure (1972)': 0.3583197830337057,
         'Monsters  Inc.(2001)': 0.6091373423880316,
         'The Rock (1996)': 0.3835812204005411,
         'Predator (1987)': 0.2929860101082948,
         'Hellraiser (1987)': 0.41996740808702987,
         "My Best Friend's Wedding (1997)": 0.3585217675896975,
         'Austin Powers: The Spy Who Shagged Me (1999)': 0.5885714850109632,
         'Austin Powers in Goldmember (2002)': 0.5149027462958911,
         'Star Wars: Episode 1 - The Phantom Menace (1999)': 0.67830501613692,
         'Star Wars: Episode II - Attack of the Clones (2002)': 0.774659860056
        0115,
         'The Good the Bad and the Ugly (1966)': 0.27214203235109263,
         "There's Something About Mary (1998)": 0.42141067644640345,
         'The Green Mile (1999)': 0.30311838493003945,
         'Let the Right One In (2008)': 0.3677153668957611,
         'Showgirls (1995)': 0.405500962044047,
         'Black Hawk Down (2001)': 0.3819350169730858,
         'The Talented Mr. Ripley (1999)': 0.3365037813780479}
```

In [ ]:
```
print(best_alphas_lasso)
beta_coefficients_lasso
```

```
{'Blues Brothers 2000 (1998)': 0.004832930238571752, 'Uptown Girls (20
03)': 0.004832930238571752, '28 Days Later (2002)': 0.0048329302385717
52, 'The Godfather: Part II (1974)': 0.0006951927961775605, 'The Godfa
ther (1972)': 0.0018329807108324356, 'Goodfellas (1990)': 0.0001, 'Kni
ght and Day (2010)': 0.004832930238571752, 'The Machinist (2004)': 0.0
04832930238571752, 'Just Married (2003)': 0.004832930238571752, 'Equil
ibrium (2002)': 0.004832930238571752, "Pirates of the Caribbean: At Wo
rld's End (2007)": 0.004832930238571752, 'Pirates of the Caribbean: Th
e Curse of the Black Pearl (2003)': 0.004832930238571752, 'The Evil De
ad (1981)': 0.0006951927961775605, 'The Poseidon Adventure (1972)': 0.
004832930238571752, 'Monsters  Inc.(2001)': 0.012742749857031334, 'The
Rock (1996)': 0.004832930238571752, 'Predator (1987)': 0.0018329807108
324356, 'Hellraiser (1987)': 0.0018329807108324356, "My Best Friend's
Wedding (1997)": 0.004832930238571752, 'Austin Powers: The Spy Who Sha
gged Me (1999)': 0.004832930238571752, 'Austin Powers in Goldmember (2
002)': 0.004832930238571752, 'Star Wars: Episode 1 - The Phantom Menac
e (1999)': 0.012742749857031334, 'Star Wars: Episode II - Attack of th
e Clones (2002)': 0.012742749857031334, 'The Good the Bad and the Ugly
(1966)': 0.0018329807108324356, "There's Something About Mary (1998)":
0.004832930238571752, 'The Green Mile (1999)': 0.004832930238571752,
'Let the Right One In (2008)': 0.0018329807108324356, 'Showgirls (199
5)': 0.0018329807108324356, 'Black Hawk Down (2001)': 0.00183298071083
24356, 'The Talented Mr. Ripley (1999)': 0.0018329807108324356}
```

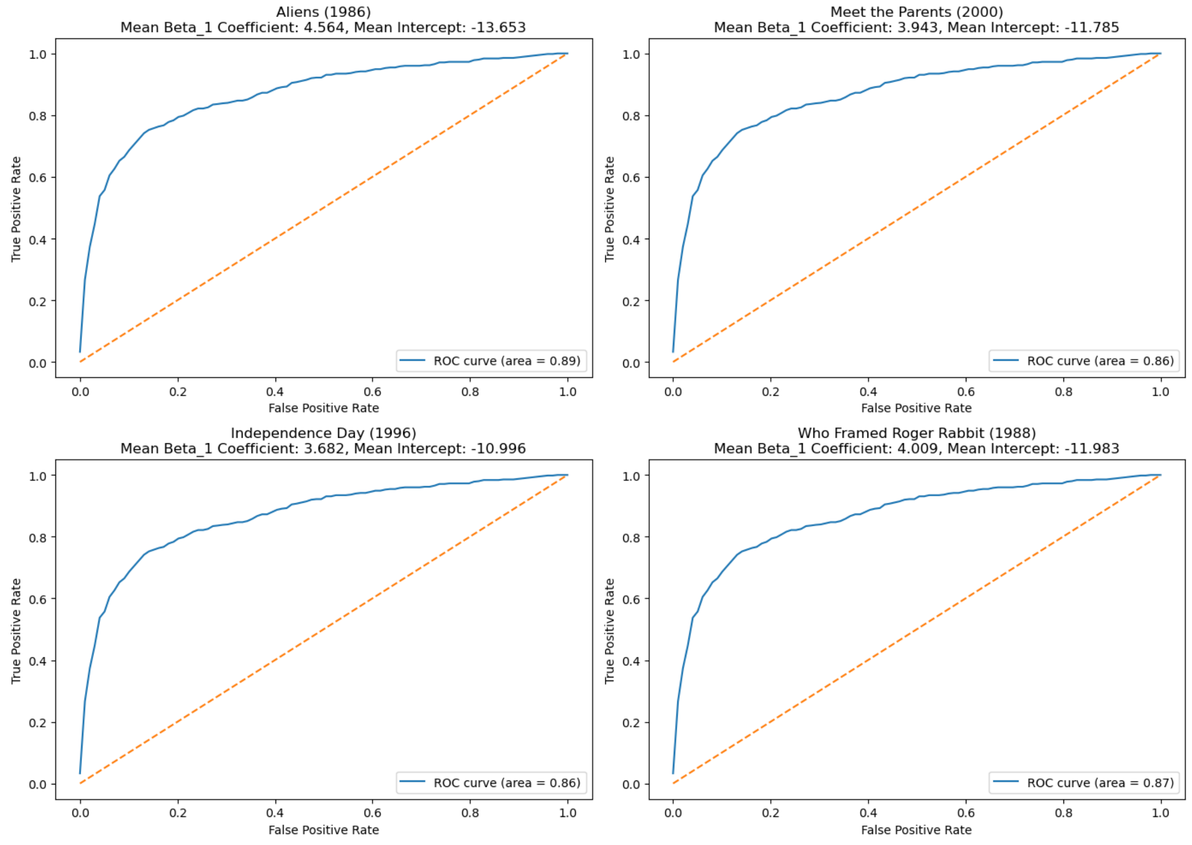
```
Out[ ]: {'Blues Brothers 2000 (1998)': array([ 0.10822944, -0.        ,  0.052
        3368 ,  0.08932749,  0.01862272,
                -0.        ,  0.11654603,  0.10780171,  0.16128885,  0.0472315
        3]),
         'Uptown Girls (2003)': array([0.10989936, 0.00138649, 0.00792715, 0.0
        9756276, 0.07824743,
                0.13054638, 0.08146676, 0.12491898, 0.23425292, 0.02027379]),
         '28 Days Later (2002)': array([ 0.08387676,  0.03598596,  0.03690003,
        0.05573743,  0.02093047,
                 0.24747633, -0.        ,  0.10065447,  0.06531187,  0.0816597
        3]),
         'The Godfather: Part II (1974)': array([-0.05822413,  0.05566289,  0.
        16662859,  0.02213592,  0.05994914,
                 0.10137501,  0.13238064,  0.00040981,  0.21202483,  0.0979532
        1]),
         'The Godfather (1972)': array([-0.06256489,  0.04337277,  0.12619412,
        0.        ,  0.08551706,
                 0.3057297 ,  0.17386659,  0.05198992,  0.15902645,  0.0166888
        2]),
         'Goodfellas (1990)': array([0.08542962, 0.07502783, 0.07192867, 0.022
        98631, 0.02011882,
                0.19579769, 0.06100039, 0.09009982, 0.21165512, 0.10317807]),
         'Knight and Day (2010)': array([ 0.12580922,  0.02396458,  0.0433710
        2,  0.07648791,  0.02365906,
                 0.18023225, -0.        ,  0.04329435,  0.20962061,  0.0195316
        1]),
         'The Machinist (2004)': array([0.0359521 , 0.        , 0.06482931, 0.
        00426235, 0.        ,
                0.27946937, 0.13039139, 0.10650353, 0.02375173, 0.12161962]),
         'Just Married (2003)': array([ 0.14624524, -0.        , -0.01197232,
        0.14135597,  0.01612519,
                 0.03561999,  0.09027942,  0.04970959,  0.29377772,  0.1045522
        ]),
         'Equilibrium (2002)': array([0.03040779, 0.04023685, 0.05659492, 0.06
        579741, 0.04544592,
                0.18715288, 0.        , 0.00356169, 0.02195785, 0.02457795]),
         "Pirates of the Caribbean: At World's End (2007)": array([ 0.1620790
        4,  0.1446664 ,  0.        ,  0.14649405,  0.1745641 ,
                -0.        ,  0.        ,  0.03077887,  0.23146634,  0.0401625
        6]),
         'Pirates of the Caribbean: The Curse of the Black Pearl (2003)': arra
        y([0.19113984, 0.16032491, 0.00153617, 0.0386574 , 0.13293872,
                0.0646207 , 0.00844097, 0.06793911, 0.13432787, 0.10288878]),
         'The Evil Dead (1981)': array([ 0.05405467,  0.04268597,  0.07121711,
        0.05970226, -0.00025667,
                 0.09860812,  0.07755943,  0.22827125,  0.12144948,  0.0831641
        9]),
         'The Poseidon Adventure (1972)': array([ 0.0730342 , -0.        ,  0.
        03103099,  0.09649113,  0.00384193,
                 0.09287712,  0.0311954 ,  0.        ,  0.1157074 ,  0.0585442
        ]),
         'Monsters  Inc.(2001)': array([0.11258721, 0.10312216, 0.05534668, 0.
        , 0.19377532,
                0.        , 0.12968595, 0.        , 0.13357699, 0.1391183 ]),
         'The Rock (1996)': array([0.01048358, 0.0023752 , 0.06204719, 0.07494
        823, 0.01816029,
                0.04275194, 0.0466951 , 0.13590252, 0.17619057, 0.08050755]),
         'Predator (1987)': array([ 0.03335514, -0.        ,  0.05407165,  0.0
        590349 ,  0.00289832,
                 0.30600879,  0.14498782,  0.03900553,  0.08641278,  0.1291085
        8]),
         'Hellraiser (1987)': array([0.04335583, 0.05102221, 0.00438212, 0.009
        85372, 0.        ,
                0.34125436, 0.02749538, 0.02894611, 0.14526108, 0.07673141]),
         "My Best Friend's Wedding (1997)": array([0.15546957, 0.00936712, 0.0
        374505 , 0.12364225, 0.01704738,
                0.13036596, 0.11100044, 0.01956988, 0.0721933 , 0.06808639]),
         'Austin Powers: The Spy Who Shagged Me (1999)': array([0.09123185, 0.
        08162459, 0.00891424, 0.20633738, 0.08810564,
                0.14189218, 0.03365257, 0.03611225, 0.22416907, 0.04649139]),
         'Austin Powers in Goldmember (2002)': array([0.01202373, 0.06064731,
        0.06085951, 0.2019321 , 0.05482771,
                0.03024388, 0.03354095, 0.04343097, 0.31773675, 0.09046057]),
         'Star Wars: Episode 1 - The Phantom Menace (1999)': array([ 0.0278540
        3,  0.04226566,  0.        ,  0.09184386,  0.09840989,
                 0.08989813,  0.0908427 ,  0.04138136,  0.24558595, -0.
        ]),
         'Star Wars: Episode II - Attack of the Clones (2002)': array([0.
        , 0.        , 0.02674878, 0.09120582, 0.06680612,
                0.25833855, 0.        , 0.        , 0.11363967, 0.        ]),
         'The Good the Bad and the Ugly (1966)': array([0.0696925 , 0.0315870
        6, 0.0506389 , 0.05895041, 0.00550535,
                0.06521414, 0.05872744, 0.27394585, 0.20083241, 0.10220633]),
         "There's Something About Mary (1998)": array([ 0.12254816, -0.0036885
        5,  0.02396987,  0.10720768,  0.03317107,
                 0.22255291,  0.06374674,  0.        ,  0.25605996,  0.0864561
        2]),
         'The Green Mile (1999)': array([0.03175111, 0.01146945, 0.02022261,
        0.02781071, 0.01350809,
                0.40840766, 0.08433375, 0.07707975, 0.09444839, 0.03819029]),
         'Let the Right One In (2008)': array([ 0.06663744,  0.00782675,  0.05
        893037,  0.02367291,  0.01775045,
                 0.16069238, -0.        ,  0.16580692,  0.18311548,  0.0521614
        1]),
         'Showgirls (1995)': array([ 0.08590035, -0.        ,  0.03512049,  0.
        08263488,  0.0125437 ,
                 0.06146287,  0.06751392,  0.05485488,  0.18655601,  0.0446019
        ]),
         'Black Hawk Down (2001)': array([0.0402588 , 0.04803367, 0.06119431,
        0.08085457, 0.03468756,
                0.17455348, 0.11859682, 0.01340259, 0.16056375, 0.04938532]),
         'The Talented Mr. Ripley (1999)': array([0.02878558, 0.03786385, 0.09
        007805, 0.13214191, 0.0128357 ,
                0.05029535, 0.0054658 , 0.12960785, 0.25876371, 0.04620263])}
```

Question 5

In [ ]:
```
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, roc_curve, auc
import matplotlib.pyplot as plt
data_original = pd.read_csv("movieReplicationSet.csv")
# average movie enjoyment for each user
real_data_df = data_original.iloc[:, :400]
average_enjoyment = real_data_df.mean(axis=1)  # Average rating for ea
ch user
# middle 4 movies
median_ratings = real_data_df.median().sort_values()
middle_movies = median_ratings.iloc[len(median_ratings)//2 - 2 : len(m
edian_ratings)//2 + 2].index.tolist()
valid_indices = ~average_enjoyment.isna() #dropping the user with all
nan values
X = average_enjoyment[valid_indices].values.reshape(-1, 1)
# Logistic Regression Models
models = {}
for movie in middle_movies:
    median_rating = data[movie].median()
    Y = (data.loc[valid_indices,movie] >= median_rating).astype(int)
    log_reg = LogisticRegression()
    # cv
    cv_results = cross_validate(log_reg, X, Y, cv=5, scoring='roc_au
c', return_estimator=True)
    # AUC
    auc_values = cv_results['test_score']
    mean_auc = np.mean(auc_values)
    models[movie] = {
        'mean_auc': mean_auc,
        'estimators': cv_results['estimator']
    }
```

In [ ]:
```
fig, axes = plt.subplots(2, 2, figsize=(14, 10))
axes = axes.flatten()
roc_curves = {}
mean_betas = {}
for i, movie in enumerate(middle_movies):
    # Extracting beta coefficients and intercepts
    beta_coefficients = [est.coef_[0][0] for est in models[movie]['est
imators']]
    intercepts = [est.intercept_[0] for est in models[movie]['estimato
rs']]
    mean_beta = np.mean(beta_coefficients)
    mean_intercept = np.mean(intercepts)
    # ROC curve
    tprs = []
    mean_fpr = np.linspace(0, 1, 100)
    for est in models[movie]['estimators']:
        fpr, tpr, thresholds = roc_curve(Y, est.predict_proba(X)[:,
1])
        tprs.append(np.interp(mean_fpr, fpr, tpr))
    mean_tpr = np.mean(tprs, axis=0)
    roc_curves[movie] = {'fpr': mean_fpr, 'tpr': mean_tpr}

    axes[i].plot(mean_fpr, mean_tpr, label=f'ROC curve (area = {models
[movie]["mean_auc"]:.2f})')
    axes[i].plot([0, 1], [0, 1], linestyle='--')
    axes[i].set_title(f'{movie}\nMean Beta_1 Coefficient: {mean_beta:.
3f}, Mean Intercept: {mean_intercept:.3f}')
    axes[i].set_xlabel('False Positive Rate')
    axes[i].set_ylabel('True Positive Rate')
    axes[i].legend(loc='lower right')
    mean_betas[movie] = {'beta_1': mean_beta, 'intercept': mean_interc
ept}
plt.tight_layout()
plt.show()
mean_betas
```



```
Out[ ]: {'Aliens (1986)': {'beta_1': 4.563638917670304,
          'intercept': -13.65318071017865},
         'Meet the Parents (2000)': {'beta_1': 3.943401401032678,
          'intercept': -11.784674813130398},
         'Independence Day (1996)': {'beta_1': 3.681566022209158,
          'intercept': -10.996001492179811},
         'Who Framed Roger Rabbit (1988)': {'beta_1': 4.009186469406261,
          'intercept': -11.982865443516681}}
```

Extra Credit

```
In [ ]: #first, check number of the null values in each column
        nulls = [i for i in data['My life is very stressful'].isna() if i==Tru
        e]
        len(nulls)#observed both columns contain very less NAs

Out[ ]: 3
```

```
In [ ]: #drop rows containing NaNs for these two features
        extra_c_data = data.dropna(subset=['My life is very stressful', 'Worri
        es a lot'], inplace=False)
```

```
In [ ]: x1_vals = extra_c_data['My life is very stressful'].values.reshape(-1,
        1)
        x2_vals = extra_c_data['Worries a lot'].values.reshape(-1,1)
```

In [ ]:
```
COD_stress = {}
for y_index in tqdm(range(0,num_movie_cols)):#iterate over the 400 mov
ies
    y_vals = extra_c_data.iloc[:,y_index].values #this is the y (ratin
gs for this movie)
    movie_name = extra_c_data.columns[y_index]#this is the movie name
    reg = LinearRegression()
    reg.fit(x1_vals,y_vals)
    y_hat = reg.predict(x1_vals)
    r2 = r2_score(y_vals,y_hat)#COD = r^2
    COD_stress[movie_name] = r2
len(COD_stress)
```

```
  0%|          | 0/400 [00:00<?, ?it/s]100%|██████████| 400/400 [00:00
<00:00, 698.22it/s]
```

Out[ ]:
```
400
```

In [ ]:
```
COD_worries = {}
for y_index in tqdm(range(0,num_movie_cols)):#iterate over the 400 mov
ies
    y_vals = extra_c_data.iloc[:,y_index].values #this is the y (ratin
gs for this movie)
    movie_name = extra_c_data.columns[y_index]#this is the movie name
    reg = LinearRegression()
    reg.fit(x2_vals,y_vals)
    y_hat = reg.predict(x2_vals)
    r2 = r2_score(y_vals,y_hat)#COD = r^2
    COD_worries[movie_name] = r2
len(COD_worries)
```

```
100%|██████████| 400/400 [00:00<00:00, 758.58it/s]
```

Out[ ]:
```
400
```

In [ ]:
```
COD_stress_list = [COD_stress[i] for i in COD_stress]
COD_worries_list = [COD_worries[i] for i in COD_worries]
print("average COD for stress is", np.mean(COD_stress_list))
print("average COD for worries is",np.mean(COD_worries_list))
```

```
average COD for stress is 0.0007103024647777417
average COD for worries is 0.002038182919227809
```

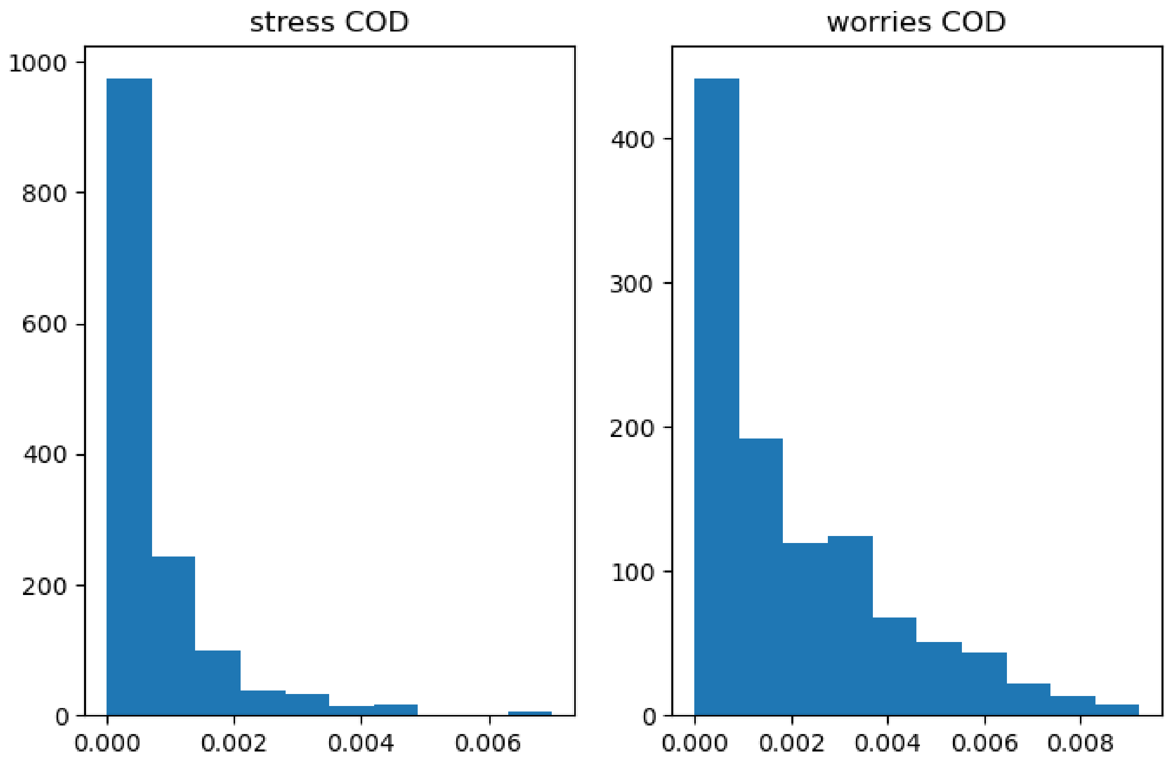
```
In [ ]: best_stress = sorted(COD_stress.items(), key=lambda i: i[1],reverse=Tr
        ue)
        temp_best_stress = best_stress[0:10]
        temp_best_stress
```

```
Out[ ]: [('Jaws (1975)', 0.0070030172741089025),
         ('Hellraiser (1987)', 0.006779591100630311),
         ('American Pie (1999)', 0.004771931787327044),
         ('10 Things I Hate About You (1999)', 0.004754714511158387),
         ('Stand By Me (1986)', 0.004587511285915391),
         ('Avatar (2009)', 0.0044346717416856585),
         ('Se7en (1995)', 0.004232640993365844),
         ('The Lord of the Rings: The Fellowship of the Ring (2001)',
          0.003892004983122921),
         ('La La Land (2016)', 0.0037673205461459247),
         ('Ice Age (2002)', 0.0036513186027213873)]
```

```
In [ ]: best_worries = sorted(COD_worries.items(), key=lambda i: i[1],reverse=
        True)
        temp_best_worries = best_worries[0:10]
        temp_best_worries
```

```
Out[ ]: [('Downfall (2004)', 0.009230615503234008),
         ('Barb Wire (1996)', 0.009108522239194339),
         ('Avatar (2009)', 0.009019711796414254),
         ('The Babadook (2014)', 0.008197888426725197),
         ('25th Hour (2002)', 0.008197830718148613),
         ('Erik the Viking (1989)', 0.007911349121206945),
         ('Stir Crazy (1980)', 0.007688001814572187),
         ('The Truman Show (1998)', 0.007385037115796744),
         ('A.I. Artificial Intelligence (2001)', 0.007147144215032042),
         ('The Proposal (2009)', 0.0071198015957023575)]
```

```
In [ ]: #plot histogram
        plt.figure(figsize=(8,5))
        plt.subplot(1,2,1)
        plt.hist(COD_stress_list, density=True)
        plt.title('stress COD')
        plt.subplot(1,2,2)
        plt.hist(COD_worries_list,density=True)
        plt.title('worries COD')
        plt.show()
```



```
In [ ]: strs = np.array([i[1] for i in temp_best_stress])
        wrs = np.array([i[1] for i in temp_best_worries])
        strs - wrs

Out[ ]: array([-0.0022276 , -0.00232893, -0.00424778, -0.00344317, -0.0036103
        2,
               -0.00347668, -0.00345536, -0.00349303, -0.00337982, -0.0034684
        8])
```