Patricia Marie Guirao
Introduction to Data Science

Final Project

Data Summary:

As the movies dataset is quite large, I often had to go through various levels of transformations and adjustments of the data to arrive to my conclusions. For each question, I created new datasets derived from the original dataset in order to preserve the original dataset to be used in every question. In each explanation, I will elaborate on how exactly I cleaned/transfored the data to arrive at my answers to each question. Additionally, I will also explain what methods and calculations I used to arrive to each answer. It's worth noting that the most common way I would clean the data is dealing with missing values in which I would often decide to just remove the row of data as a whole.

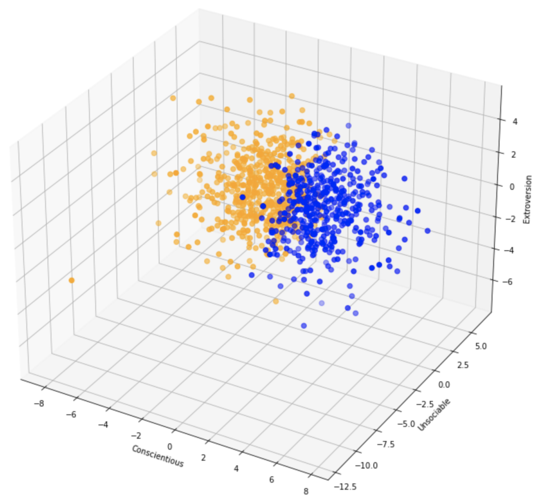
1) What is the relationship between sensation seeking and movie experience?

I first created a new dataset with just the sensation seeking and movie experience columns from the original dataset, dropping the rows with any NaN values in any of the columns. I then ran a PCA on the sensation seeking columns and through that chose to focus on the top 6 factors from the PCA as selected by the Kaiser criterion. After looking at the loadings in comparison to the sensation seeking questions, I labeled the top 6 factors as "Impulsive", "Risky", "Carefree", "Reckless", "Thrill Seeking", and "Adrenaline Junkie". I then ran a PCA on the movie experience columns and through that chose to focus on the top 2 factors from the PCA as selected by the Kaiser criterion. After looking at the loadings in comparison to the movie experience questions, I labeled the top 2 factors as "Immersive" and "Emotional". From here, to find the relationship between sensation seeking and movie experience, I correlated the data coordinates of the top 6 factors from the sensation seeking PCA with the data coordinates of the top 2 factors from the movie experience PCA to get the correlation matrix shown below. From this matrix I've observed a relatively low correlation between sensation seeking and movie experience.

	Immersive	Emotional
Impulsive	0.012482	-0.044313
Risky	-0.131829	0.028770
Carefree	0.080482	0.042814
Reckless	-0.116775	0.008714
Thrill Seeking	0.139366	-0.058626
Adrenaline Junkie	-0.020143	-0.032056

2) Is there evidence of personality types based on the data of these research participants? If so, characterize these types both quantitatively and narratively.

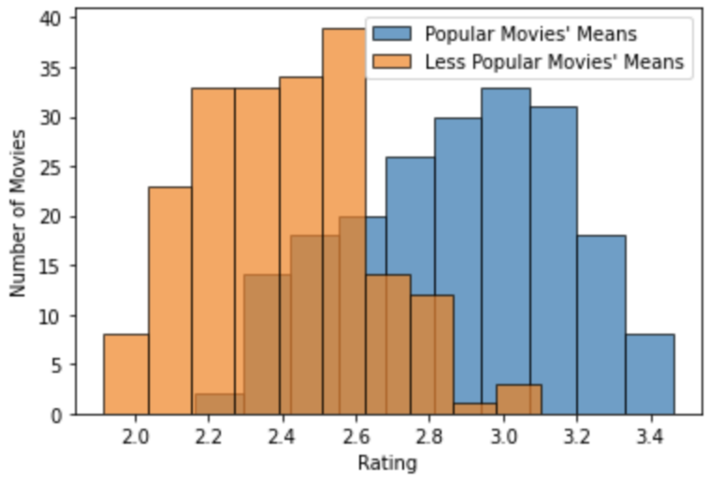
I first ran a PCA on the personality columns in the data to reduce the dimensions used. From the PCA, I decided to use the top 3 factors derived from the personality data. The proportion of the variance explained by these top 3 PCs is 0.325. I first began by interpreting the top 3 factors from the personality questions through looking at the loadings for each of them. Through this I determined that factor 1 would be being conscientious, factor 2 would be being unsociable, and factor 3 would be being extroverted. I then determined the appropriate number of clusters to be 2 through looking at the silhouette scores and picking the cluster with the highest score. I then computed the kMeans using 2 clusters and plotted the color-coded data with the factors as the 3 dimensions as shown below. Through looking at this graph, I've determined that there is evidence of personality types based on the data of these research participants. From just looking at the top 3 factors, we can see 2 personality types through the 2 different clusters: conscientious personalities that have varying degrees of unsocability and extroversion and non-conscientious personalities that also have varying degrees of unsocability and extroversion.



3) Are movies that are more popular rated higher than movies that are less popular?

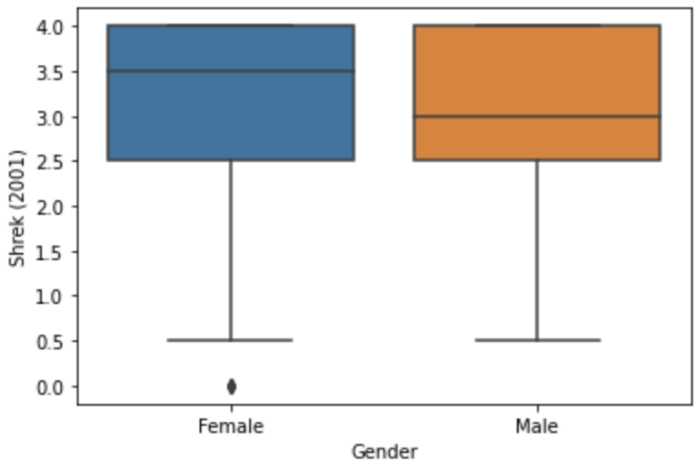
I first began by splitting the dataset in half based on movie popularity which I determined would be defined by the number of ratings. I did this by finding the median number of ratings for a movie (**197.5 ratings**) and creating two new datasets, one containing the 200 movies that had more ratings than the median and the other containing the 200 movies that had fewer ratings than the median. I then ran an independent t-test on these two datasets and got a **t-value of 17.756049269873714** and a **p-value of 2.269653027656258e-52**. As the very large magnitude of this t-value indicates that these two groups are very different, the positive sign of this t-value indicates that the first group inputted (which I put as the "popular movie" group) has a higher mean than the second, and this very low p-value (much lower than 5% or even 1%) indicates that it's a very low probability that this t-value occurred by chance, we **can reject the null hypothesis** that **movies that are more popular are not rated higher than movies that are**

less popular. In other words, it can be concluded that **movies that are more popular are rated higher than movies that are less popular**. The histogram below shows the distribution of means of the “popular movie dataset” and the “less popular movie dataset”.



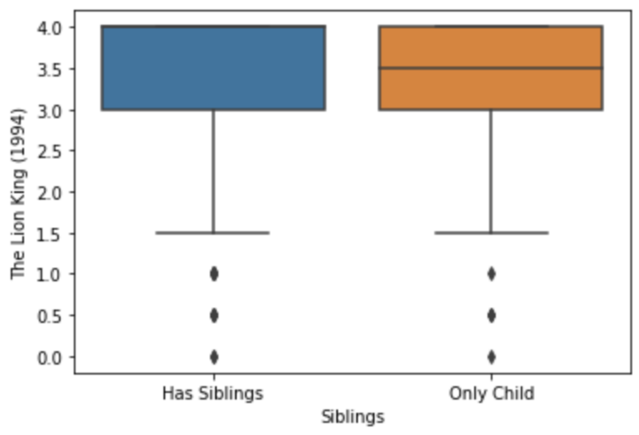
4) Is enjoyment of ‘Shrek (2001)’ gendered, i.e. do male and female viewers rate it differently?

I first began by creating a new dataset that contained just the column that had the “Shrek (2001)” ratings and the column that contains the participants’ gender identities. For simplicity, I renamed the “Gender identity (1 = female; 2 = male; 3 = self-described)” column to simply just “Gender”. I then removed any rows that contained NaN values in either column as well as any rows that contained a “self-described” gender in the gender column. I then went on to split this dataset into two further datasets: one that contained just the Shrek ratings of females and another that contained just the Shrek ratings of males. I then ran an independent t-test on these two datasets and got an **absolute t-value of 1.1016699726285888** and a **p-value of 0.27087511813734183**. As this small t-value indicates that the two groups are generally similar/only slightly different and this p-value that is greater than 5% indicates that it’s probable that this t-value occurred by chance, we **cannot reject the null hypothesis** that **the enjoyment of “Shrek (2001) is not gendered**. To put it in other words, it can be concluded that male and female viewers **do not rate “Shrek (2001)” differently**. The boxplot below visualizes the distribution of the “Shrek (2001)” ratings of female and male participants.



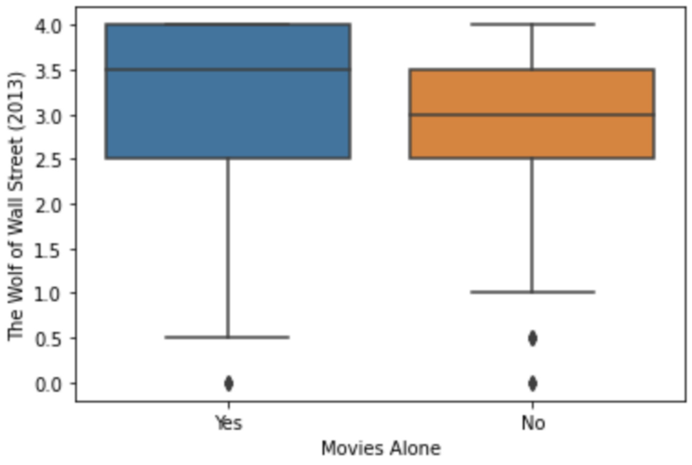
5) Do people who are only children enjoy 'The Lion King (1994)' more than people with siblings?

I first began by creating a new dataset that contained just the column that had the "Lion King (1994)" ratings and the column that contains information on whether or not the participants are only children are not. For simplicity, I renamed the "Are you an only child? (1: Yes; 0: No; -1: Did not respond)" column to simply just "Siblings". I then removed any rows that contained NaN values in either column as well as any rows that contained a "No response" response in the siblings column. I then went on to split this dataset into two further datasets: one that contained just the Lion King ratings of only children and another that contained just the Lion King ratings of those with siblings. I then ran an independent t-test on these two datasets and got a **t-value of -2.0538889960589857** and a **p-value of 0.04026705526268264**. As the magnitude of this t-value indicates that these two groups are slightly different, the negative sign of this t-value indicates that the first group inputted (which I put as the "only children" group) has a lower mean than the second, and this p-value just below 5% indicates that it's unlikely that this t-value occurred by chance, we **can reject the null hypothesis** that **people who are only children do not enjoy "The Lion King (1994)" more than people with siblings**. In other words, it can be concluded that **people who are only children do not enjoy "The Lion King (1994)" more than people with siblings** and furthermore, that **people with siblings enjoy "The Lion King 1994)" more than people with siblings**. The boxplot below visualizes the distribution of the "The Lion King (1994)" ratings of only children and participants with siblings.



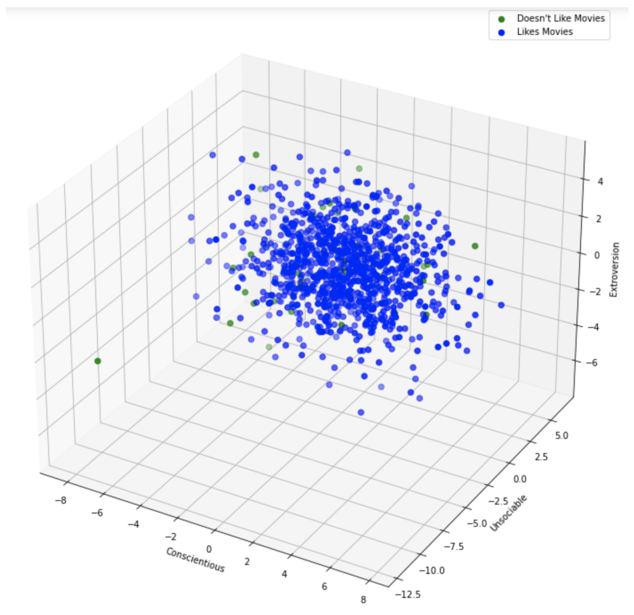
6) Do people who like to watch movies socially enjoy 'The Wolf of Wall Street (2013)' more than those who prefer to watch them alone?

I first began by creating a new dataset that contained just the column that had the "The Wolf of Wall Street (2013)" ratings and the column that contains information on whether or not the participants prefer to watch movies alone or not. For simplicity, I renamed the "Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)" column to simply just "Movies Alone". I then removed any rows that contained NaN values in either column as well as any rows that contained a "No response" response in the "Movies Alone" column. I then went on to split this dataset into two further datasets: one that contained just the The Wolf on Wall Street ratings of those who prefer to watch movies alone and another that contained just the The Wolf on Wall Street ratings of those who don't prefer to watch movies alone. I then ran an independent t-test on these two datasets and got an **absolute t-value of 1.567873874504994** and a **p-value of 0.11738913665664574**. As this small t-value indicates that the two groups are generally similar/only slightly different and this p-value that is greater than 5% indicates that it's probable that this t-value occurred by chance, we **cannot reject the null hypothesis** that **people who like to watch movies socially don't enjoy "The Wolf of Wall Street (2013)" more than those who prefer to watch them alone**. In other words, **there is no perceived difference in the ratings of "The Wolf of Wall Street (2013)" between those who watch movies socially and those who prefer to watch them alone.** The boxplot below visualizes the distribution of the "The Wolf of Wall Street (2013)" ratings of only children and participants with siblings.



8) Build a prediction model of your choice (regression or supervised learning) to predict movie ratings (for all 400 movies) from personality factors only. Make sure to use cross-validation methods to avoid overfitting and characterize the accuracy of your model.

I first began by creating a new dataset with just the movie ratings of all 400 movies. From there, I created a new column that gave the average movie rating each participant. I then further created a new column from the “Average Rating” column that categorized participants as either liking movies overall if their ratings average was above 2, and as not liking movies overall if their ratings average was below or equal to 2, labeling the colun as “Likes Movies Overall” and having the entry “1” meaning they like movies and the entry “0” meaning they don’t. I then created a new dataset by concating this new dataset I just created with a dataset I created for an earlier question containing just the personality columns of the original dataset, inner joining to remove any rows where there were NaN values in the personalnity columns. Using the top 3 factors I derived earlier from the personality data after running a PCA (conscientious, unsociable, extroversion), I plotted the data in these 3 dimensions, from there classifying the datapoints based on the “Likes Movies Overall” column. In other words, the graph visualizes the prediction of whether each participant overall likes movies or not based on these top 3 factors derived from the personality data. To cross-validate my model and characterize its accuracy, I found the **SVM model accuracy which is 0.9628140703517588**. As this model’s accuracy is quite high, this is a relatively good prediction model.



9) Build a prediction model of your choice (regression or supervised learning) to predict movie ratings (for all 400 movies) from gender identity, sibship status and social viewing preferences (columns 475-477) only. Make sure to use cross-validation methods to avoid overfitting and characterize the accuracy of your model.

To begin with, I used the dataset I created in the previous question with just movies and the “Average Rating” column, dropping the “Likes Movies Overall” column for simplicity. I then went on to create a new data set with just the columns with information on participants’ gender, sibship, and social movie viewing preferences, dropping rows with NaN values in any of those columns. I then created a new dataset by concating the dataset with just movie ratings and the “Average Rating” column with the new dataset I just created with just gender, sibship, and social viewing preferences, inner joining to remove any rows where there were NaN values in the the latter three columns. From here, I created a training set with the first 75% of the data, training a linear regression model with gender, sibship, and social viewing prefrences as predictors for participants’ average movie ratings. From this model, I got the betas in the table below and calculated an **R^2 value of 0.007153632607742821**. I cross-validated my model with a test set that I created from the last 25% of the data, calculating the **RMSE to be 0.46977563715539306**. As this model has a very low R^2 value, these three predictors (Gender, Sibship, and Social Viewing Preferences) are not good as this model explains a very low proportion of the variance.

Predictors	Betas
Gender	-0.011149
Only Child	-0.100664
Movies Alone	-0.018623

10) Build a prediction model of your choice (regression or supervised learning) to predict movie ratings (for all 400 movies) from all available factors that are not movie ratings (columns 401-477). Make sure to use cross-validation methods to avoid overfitting and characterize the accuracy of your model.

To begin with, I used the dataset with just movies I created in previous questions. I then went on to create a new data set with just the columns that were not movie ratings, dropping rows with NaN values in any of those columns. I then created a new dataset by concating the dataset with just movie ratings and the "Average Rating" column with the new dataset I just created with all columns that were not movie ratings, inner joining to remove any rows where there were NaN values in the columns that were not movie ratings. From here, I created a training set with the first 75% of the data. I then ran a PCA on the sensation seeking columns, the personality columns, and the movie experience columns, from there choosing to focus on the top 7, 8, and 7 factors (as selected by the Kaiser criterion) respectively for the regression model. I then went on to train a linear regression model with these factors as well as gender, sibship, and social viewing preferences as predictors for participants' average movie ratings. From this model, I got the betas in the table below and calculated an **R^2 value of 0.07688275320397875**. I cross-validated my model with a test set that I created from the last 25% of the data, calculating the **RMSE to be 0.46775454394044624**. As this model has a low R^2 value, these 25 predictors are not good as this model explains a low proportion of the variance.

Predictors	Betas
x1	0.001742
x2	-0.023412
x3	0.002336
x4	-0.014107
x5	0.032257
x6	-0.034589
x7	-0.017223
x8	0.011087
x9	0.009637
x10	0.003260
x11	-0.012666
x12	0.001502
x13	0.023776

x14	-0.006960
x15	-0.002290
x16	0.011087
x17	0.009637
x18	0.003260
x19	-0.012666
x20	0.001502
x21	0.023776
x22	-0.006960
x23	-0.009433
x24	-0.110066
x25	-0.008849

Code Used

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from scipy.stats import ttest_rel
from sklearn.decomposition import PCA
from scipy import stats
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples
from sklearn.cluster import DBSCAN
import statistics
import seaborn as sns
from sklearn import svm
from sklearn.metrics import accuracy_score

df = pd.read_csv ('movieReplicationSet.csv')
```

Question 1

```
sensation = df.iloc[:, 400:420]
movie_experience = df.iloc[:, 464:474]
sensation_movie_experience = pd.concat([sensation, movie_experience], axis = 1).dropna()

sensation = sensation_movie_experience.iloc[:, 0:20]
sensation = stats.zscore(sensation)
pca_sensation = PCA().fit(sensation)
sensation_rotated_data = pca_sensation.fit_transform(sensation)
sensation_loadings = pca_sensation.components_*-1
sensation_eigVals = pca_sensation.explained_variance_

x = np.linspace(1,20,20)
plt.bar(x, sensation_eigVals, color='gray')
plt.plot([0,20],[1,1],color='orange') # Orange Kaiser criterion line for the fox
plt.xlabel('Principal component')
plt.ylabel('Eigenvalue')
plt.show()

threshold = 1
print('Number of factors selected by Kaiser criterion:', np.count_nonzero(sensation_eigVals >
threshold))

whichPrincipalComponent = 1
plt.bar(x,sensation_loadings[whichPrincipalComponent,:])
plt.xlabel('Sensation Questions')
plt.ylabel('Loading')
plt.plot([0,20],[0.2,0.2],color='orange')
plt.show()

whichPrincipalComponent = 2
plt.bar(x,sensation_loadings[whichPrincipalComponent,:])
plt.xlabel('Sensation Questions')
plt.ylabel('Loading')
plt.plot([0,20],[0.2,0.2],color='orange')
plt.show()

whichPrincipalComponent = 3
plt.bar(x,sensation_loadings[whichPrincipalComponent,:])
plt.xlabel('Sensation Questions')
plt.ylabel('Loading')
plt.plot([0,20],[0.2,0.2],color='orange')
plt.show()

whichPrincipalComponent = 4
```

```
plt.bar(x,sensation_loadings[whichPrincipalComponent,:])
plt.xlabel('Sensation Questions')
plt.ylabel('Loading')
plt.plot([0,20],[0.2,0.2],color='orange')
plt.show()

whichPrincipalComponent = 5
plt.bar(x,sensation_loadings[whichPrincipalComponent,:])
plt.xlabel('Sensation Questions')
plt.ylabel('Loading')
plt.plot([0,20],[0.2,0.2],color='orange')
plt.show()

whichPrincipalComponent = 6
plt.bar(x,sensation_loadings[whichPrincipalComponent,:])
plt.xlabel('Sensation Questions')
plt.ylabel('Loading')
plt.plot([0,20],[0.2,0.2],color='orange')
plt.show()

movie_experience = sensation_movie_experience.iloc[:, 20:30]
movie_experience = stats.zscore(movie_experience)
pca_movie_experience = PCA().fit(movie_experience)
movie_expereince_rotated_data = pca_movie_experience.fit_transform(movie_experience)
movie_experience_loadings = pca_movie_experience.components_*-1
movie_experience_eigVals = pca_movie_experience.explained_variance_

x = np.linspace(1,10,10)
plt.bar(x, movie_experience_eigVals, color='gray')
plt.plot([0,10],[1,1],color='orange') # Orange Kaiser criterion line for the fox
plt.xlabel('Principal component')
plt.ylabel('Eigenvalue')
plt.show()

threshold = 1
print('Number of factors selected by Kaiser criterion:',
np.count_nonzero(movie_experience_eigVals > threshold))

whichPrincipalComponent = 1
plt.bar(x,movie_experience_loadings[whichPrincipalComponent,:])
plt.xlabel('Movie Experience Questions')
plt.ylabel('Loading')
plt.plot([0,10],[0.2,0.2],color='orange')
plt.show()
```

```
whichPrincipalComponent = 2
plt.bar(x,movie_experience_loadings[whichPrincipalComponent,:])
plt.xlabel('Movie Experience Questions')
plt.ylabel('Loading')
plt.plot([0,10],[0.2,0.2],color='orange')
plt.show()

sensation_data_array = np.array([sensation_rotated_data[:,0], sensation_rotated_data[:,1],
sensation_rotated_data[:,2], sensation_rotated_data[:,3], sensation_rotated_data[:,4],
sensation_rotated_data[:,5]])
sensation_data = pd.DataFrame(sensation_data_array.T)
sensation_data

movie_experience_data_array = np.array([movie_expereince_rotated_data[:,0],
movie_expereince_rotated_data[:,1]])
movie_experience_data = pd.DataFrame(movie_experience_data_array.T)
movie_experience_data

corr_data = pd.concat([sensation_data, movie_experience_data], axis=1).corr().iloc[:6, 6:]
corr_data
```

Question 2

```
personality = df.iloc[:, 421:465].dropna()
personality_zscore = stats.zscore(personality)
pca_personality = PCA().fit(personality_zscore)
personality_loadings = pca_personality.components_*-1
personality_eigVals = pca_personality.explained_variance_
personality_data_coordinates = pca_personality.fit_transform(personality_zscore)*-1

x = np.linspace(1,44,44)
plt.bar(x, personality_eigVals, color='gray')
plt.plot([0,44],[1,1],color='orange') # Orange Kaiser criterion line for the fox
plt.xlabel('Principal component')
plt.ylabel('Eigenvalue')
plt.show()

print('Proportion variance explained by the first 3
PCs:',np.sum(personality_eigVals[:3]/np.sum(personality_eigVals)).round(3))

whichPrincipalComponent = 1
plt.bar(x,personality_loadings[whichPrincipalComponent,:])
plt.xlabel('Personality Questions')
plt.ylabel('Loading')
plt.plot([0,44],[0.2,0.2],color='orange')
plt.show()

whichPrincipalComponent = 2
plt.bar(x,personality_loadings[whichPrincipalComponent,:])
plt.xlabel('Personality Questions')
plt.ylabel('Loading')
plt.plot([0,44],[0.2,0.2],color='orange')
plt.show()

whichPrincipalComponent = 3
plt.bar(x,personality_loadings[whichPrincipalComponent,:])
plt.xlabel('Personality Questions')
plt.ylabel('Loading')
plt.plot([0,44],[0.2,0.2],color='orange')
plt.show()

numClusters = 9
Q = np.empty([numClusters,1])*np.NaN

for ii in range(2, 11):
    kMeans = KMeans(n_clusters = int(ii)).fit(X)
    cId = kMeans.labels_
    cCoords = kMeans.cluster_centers_
    s = silhouette_samples(X,cId)
    Q[ii-2] = sum(s)
    # Plot data:
    plt.subplot(3,3,ii-1)
    plt.hist(s,bins=20)
    plt.xlim(-0.2,1)
    plt.ylim(0,250)
    plt.xlabel('Silhouette score')
    plt.ylabel('Count')
    plt.title('Sum: {}'.format(int(Q[ii-2])))
    plt.tight_layout()

plt.plot(np.linspace(2,10,9),Q)
plt.xlabel('Number of clusters')
plt.ylabel('Sum of silhouette scores')
plt.show()

numClusters = 2
kMeans = KMeans(n_clusters = numClusters).fit(X)
```

```
cId = kMeans.labels_
cCoords = kMeans.cluster_centers_

model = KMeans(n_clusters = 2, init = "k-means++", max_iter = 300, n_init = 10, random_state
= 0)
clusters = model.fit_predict(X)
fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[clusters == 0,0],X[clusters == 0,1],X[clusters == 0,2], s = 40 , color = 'blue', label =
"cluster 1")
ax.scatter(X[clusters == 1,0],X[clusters == 1,1],X[clusters == 1,2], s = 40 , color = 'orange', label
= "cluster 2")
ax.set_xlabel('Conscientious')
ax.set_ylabel('Unsociable')
ax.set_zlabel('Extroversion')
plt.tight_layout()
plt.show()
```

Question 3

```
num_ratings = []
for movie in range(400):
    num_ratings.append(1097 - df.iloc[:, movie].isnull().sum())

median_num_ratings = statistics.median(num_ratings)
median_num_ratings

popular = pd.DataFrame()
less_popular = pd.DataFrame()

for movie in range(400):
    if 1097 - df.iloc[:, movie].isnull().sum() > median_num_ratings:
        movie_col = pd.DataFrame(df.iloc[:, movie])
        popular = pd.concat([popular, movie_col], axis=1)
    else:
        movie_col = pd.DataFrame(df.iloc[:, movie])
        less_popular = pd.concat([less_popular, movie_col], axis=1)

sm.stats.ttest_ind(popular_means, less_popular_means)

plt.hist(popular_means, alpha = 0.7, ec = 'black', label = "Popular Movies' Means")
plt.hist(less_popular_means, alpha = 0.7, ec = 'black', label = "Less Popular Movies' Means")
plt.xlabel('Rating')
plt.ylabel('Number of Movies')
plt.legend()
plt.show()
```

Question 4

```
shrek = df[["Shrek (2001)"]]
gender = df[["Gender identity (1 = female; 2 = male; 3 = self-described)"]]
gender = gender.rename(columns={"Gender identity (1 = female; 2 = male; 3 = self-described)":
"Gender"})

gender_shrek = pd.concat([shrek, gender], axis = 1).dropna()
gender_shrek = gender_shrek[gender_shrek["Gender"] != 3]
gender_shrek

female = gender_shrek.loc[gender_shrek["Gender"] == 1]
female

male = gender_shrek.loc[gender_shrek["Gender"] == 2]
male

sm.stats.ttest_ind(female["Shrek (2001)"], male["Shrek (2001)"])

gender_shrek["Gender"] = np.where(gender_shrek["Gender"] == 1, "Female", "Male")
sns.boxplot(x = "Gender", y = "Shrek (2001)", data = gender_shrek)
```

Question 5

```
lion_king = df[["The Lion King (1994)"]]
siblings = df[["Are you an only child? (1: Yes; 0: No; -1: Did not respond)"]]
siblings = siblings.rename(columns={"Are you an only child? (1: Yes; 0: No; -1: Did not
respond)": "Siblings"})

siblings_lion_king = pd.concat([lion_king, siblings], axis = 1).dropna()
siblings_lion_king = siblings_lion_king[siblings_lion_king["Siblings"] != -1]
siblings_lion_king

sm.stats.ttest_ind(only_child["The Lion King (1994)"], has_siblings["The Lion King (1994)"])

siblings_lion_king["Siblings"] = np.where(siblings_lion_king["Siblings"] == 1, "Only Child", "Has
Siblings")
sns.boxplot(x = "Siblings", y = "The Lion King (1994)", data = siblings_lion_king)
```

Question 6

```
wolf_wallstreet = df[["The Wolf of Wall Street (2013)"]]
lonely = df[["Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)"]]
lonely = lonely.rename(columns={"Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not
respond)": "Movies Alone"})

lonely_wolf_wallstreet = pd.concat([wolf_wallstreet, lonely], axis = 1).dropna()
lonely_wolf_wallstreet = lonely_wolf_wallstreet[lonely_wolf_wallstreet["Movies Alone"] != -1]
lonely_wolf_wallstreet

alone = lonely_wolf_wallstreet.loc[lonely_wolf_wallstreet["Movies Alone"] == 1]
alone

not_alone = lonely_wolf_wallstreet.loc[lonely_wolf_wallstreet["Movies Alone"] == 0]
not_alone

sm.stats.ttest_ind(alone["The Wolf of Wall Street (2013)"], not_alone["The Wolf of Wall Street
(2013)"])

lonely_wolf_wallstreet["Movies Alone"] = np.where(lonely_wolf_wallstreet["Movies Alone"] == 1,
"Yes", "No")
sns.boxplot(x = "Movies Alone", y = "The Wolf of Wall Street (2013)", data =
lonely_wolf_wallstreet)
```

Question 8

```
movies = df.iloc[:, 0:400]

movies["Average Rating"] = movies.mean(axis=1)

movies['Likes Movies Overall'] = 0
movies.loc[movies['Average Rating'] > 2, 'Likes Movies Overall'] = 1

movies_personality = pd.concat([movies, personality], join = 'inner', axis = 1)

likes_movies = movies_personality["Likes Movies Overall"].to_numpy()

fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[np.argwhere(likes_movies==0),0],X[np.argwhere(likes_movies==0),1],X[np.argwhe
re(likes_movies==0),2],s=40,color='green', label = "Doesn't Like Movies")
ax.scatter(X[np.argwhere(likes_movies==1),0],X[np.argwhere(likes_movies==1),1],X[np.argwhe
re(likes_movies==1),2],s=40,color='blue', label = "Likes Movies")
ax.set_xlabel('Conscientious')
ax.set_ylabel('Unsociable')
ax.set_zlabel('Extroversion')
ax.legend()
plt.tight_layout()
plt.show()

clf = svm.SVC().fit(X,likes_movies)
predictions = clf.predict(X)
modelAccuracy = accuracy_score(likes_movies,predictions)
print('SVM model accuracy:',modelAccuracy)
```

Question 9

```
gender_sibship_social = df.iloc[:, 474:477].dropna()
gender_sibship_social = gender_sibship_social.rename(columns={"Gender identity (1 = female;
2 = male; 3 = self-described)": "Gender", "Are you an only child? (1: Yes; 0: No; -1: Did not
respond)": "Only Child", "Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)":
"Movies Alone"})

movies = movies.drop(['Likes Movies Overall'], axis=1)
movies_gender_sibship_social = pd.concat([movies, gender_sibship_social], join = 'inner', axis
= 1)

train = movies_gender_sibship_social.iloc[0:805]

x_train = train[["Gender", "Only Child", "Movies Alone"]]
y_train = train["Average Rating"]
lin_reg_train = lin_reg(x_train, y_train)

betas = lin_reg_train[0]
betas

test = movies_gender_sibship_social.iloc[805:]

x_test = test[["Gender", "Only Child", "Movies Alone"]]
y_test = test["Average Rating"]

yHat1 = betas[0] + betas[1]*x_train["Gender"] + betas[2]*x_train["Only Child"] +
betas[3]*x_train["Movies Alone"]
yHat2 = betas[0] + betas[1]*x_test["Gender"] + betas[2]*x_test["Only Child"] +
betas[3]*x_test["Movies Alone"]
rmse = np.sqrt(np.mean((yHat2 - y_test)**2))
rmse

r_squared = lin_reg_train[2]
```

```
r_squared
```

Question 10

```
not_movies = df.iloc[:, 400:].dropna()
not_movies

movies_other_factors = pd.concat([movies, not_movies], join = 'inner', axis = 1)
movies_other_factors

train = movies_other_factors.iloc[0:709]

sensation = train.iloc[:, 401:421]
sensation_zscores = stats.zscore(sensation)
pca_sensation = PCA().fit(sensation_zscores)
sensation_rotated_data = pca_sensation.fit_transform(sensation_zscores)
sensation_loadings = pca_sensation.components_*-1
sensation_eigVals = pca_sensation.explained_variance_
sensation_data_coordinates = pca_sensation.fit_transform(sensation_zscores)*-1

threshold = 1
print('Number of factors selected by Kaiser criterion:', np.count_nonzero(sensation_eigVals >
threshold))

personality = train.iloc[:, 421:465]
personality_zscores = stats.zscore(personality)
pca_personality = PCA().fit(personality_zscores)
personality_loadings = pca_personality.components_*-1
personality_eigVals = pca_personality.explained_variance_
personality_data_coordinates = pca_personality.fit_transform(personality_zscores)*-1

threshold = 1
print('Number of factors selected by Kaiser criterion:', np.count_nonzero(personality_eigVals >
threshold))

movie_experience = train.iloc[:, 465:475]
movie_experience_zscores = stats.zscore(movie_experience)
pca_movie_experience = PCA().fit(movie_experience_zscores)
movie_expereince_rotated_data = pca_movie_experience.fit_transform(movie_experience)
movie_experience_loadings = pca_movie_experience.components_*-1
movie_experience_eigVals = pca_movie_experience.explained_variance_
movie_experience_data_coordinates =
pca_movie_experience.fit_transform(personality_zscores)*-1
```

```
threshold = 1
print('Number of factors selected by Kaiser criterion:',
np.count_nonzero(movie_experience_eigVals > threshold))

gender = train[["Gender identity (1 = female; 2 = male; 3 = self-described)"]].to_numpy().flatten()
only_child = train[["Are you an only child? (1: Yes; 0: No; -1: Did not
respond)"]].to_numpy().flatten()
alone = train[["Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not
respond)"]].to_numpy().flatten()

x_train = np.array([sensation_data_coordinates[:,0], sensation_data_coordinates[:,1],
sensation_data_coordinates[:,2], sensation_data_coordinates[:,3],
sensation_data_coordinates[:,4], sensation_data_coordinates[:,5],
sensation_data_coordinates[:,6], personality_data_coordinates[:,0],
personality_data_coordinates[:,1], personality_data_coordinates[:,2],
personality_data_coordinates[:,3], personality_data_coordinates[:,4],
personality_data_coordinates[:,5], personality_data_coordinates[:,6],
personality_data_coordinates[:,7], movie_experience_data_coordinates[:,0],
movie_experience_data_coordinates[:,1], movie_experience_data_coordinates[:,2],
movie_experience_data_coordinates[:,3], movie_experience_data_coordinates[:,4],
movie_experience_data_coordinates[:,5], movie_experience_data_coordinates[:,6], gender,
only_child, alone]).T
y_train = train['Average Rating']
lin_reg_train = lin_reg(x_train, y_train)

betas = lin_reg_train[0]
betas

test = movies_other_factors.iloc[709:]

sensation_test = test.iloc[:, 401:421]
sensation_zscores_test = stats.zscore(sensation_test)
pca_sensation_test = PCA().fit(sensation_zscores_test)
sensation_rotated_data_test = pca_sensation.fit_transform(sensation_zscores_test)
sensation_loadings_test = pca_sensation_test.components_*-1
sensation_eigVals_test = pca_sensation_test.explained_variance_
sensation_data_coordinates_test =
pca_sensation_test.fit_transform(sensation_zscores_test)*-1

personality_test = test.iloc[:, 421:465]
personality_zscores_test = stats.zscore(personality_test)
pca_personality_test = PCA().fit(personality_zscores_test)
personality_loadings_test = pca_personality_test.components_*-1
personality_eigVals_test = pca_personality_test.explained_variance_
```

```
personality_data_coordinates_test =
pca_personality_test.fit_transform(personality_zscores_test)*-1

movie_experience_test = test.iloc[:, 465:475]
movie_experience_zscores_test = stats.zscore(movie_experience_test)
pca_movie_experience_test = PCA().fit(movie_experience_zscores_test)
movie_expereince_rotated_data_test =
pca_movie_experience_test.fit_transform(movie_experience_test)
movie_experience_loadings_test = pca_movie_experience_test.components_*-1
movie_experience_eigVals_test = pca_movie_experience_test.explained_variance_
movie_experience_data_coordinates_test =
pca_movie_experience_test.fit_transform(movie_experience_zscores_test)*-1

gender = test[["Gender identity (1 = female; 2 = male; 3 = self-described)"]].to_numpy().flatten()
only_child = test[["Are you an only child? (1: Yes; 0: No; -1: Did not
respond)"]].to_numpy().flatten()
alone = test[["Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not
respond)"]].to_numpy().flatten()

x_test = np.array([sensation_data_coordinates_test[:,0], sensation_data_coordinates_test[:,1],
sensation_data_coordinates_test[:,2], sensation_data_coordinates_test[:,3],
sensation_data_coordinates_test[:,4], sensation_data_coordinates_test[:,5],
sensation_data_coordinates_test[:,6], personality_data_coordinates_test[:,0],
personality_data_coordinates_test[:,1], personality_data_coordinates_test[:,2],
personality_data_coordinates_test[:,3], personality_data_coordinates_test[:,4],
personality_data_coordinates_test[:,5], personality_data_coordinates_test[:,6],
personality_data_coordinates_test[:,7], movie_experience_data_coordinates_test[:,0],
movie_experience_data_coordinates_test[:,1], movie_experience_data_coordinates_test[:,2],
movie_experience_data_coordinates_test[:,3], movie_experience_data_coordinates_test[:,4],
movie_experience_data_coordinates_test[:,5], movie_experience_data_coordinates_test[:,6],
gender, only_child, alone]).T
y_test = test['Average Rating']

yHat1 = betas[0] + betas[1]*x_train[:,0] + betas[2]*x_train[:,1] + betas[3]*x_train[:,2] +
betas[4]*x_train[:,3] + betas[5]*x_train[:,4] + betas[6]*x_train[:,5] + betas[7]*x_train[:,6] +
betas[8]*x_train[:,7] + betas[9]*x_train[:,8] + betas[10]*x_train[:,9] + betas[11]*x_train[:,10] +
betas[12]*x_train[:,11] + betas[13]*x_train[:,12] + betas[14]*x_train[:,13] + betas[15]*x_train[:,14]
+ betas[16]*x_train[:,15] + betas[17]*x_train[:,16] + betas[18]*x_train[:,17] +
betas[19]*x_train[:,18] + betas[20]*x_train[:,19] + betas[21]*x_train[:,20] + betas[22]*x_train[:,21]
+ betas[23]*x_train[:,22] + betas[24]*x_train[:,23] + betas[25]*x_train[:,24]
yHat1 = betas[0] + betas[1]*x_test[:,0] + betas[2]*x_test[:,1] + betas[3]*x_test[:,2] +
betas[4]*x_test[:,3] + betas[5]*x_test[:,4] + betas[6]*x_test[:,5] + betas[7]*x_test[:,6] +
betas[8]*x_test[:,7] + betas[9]*x_test[:,8] + betas[10]*x_test[:,9] + betas[11]*x_test[:,10] +
betas[12]*x_test[:,11] + betas[13]*x_test[:,12] + betas[14]*x_test[:,13] + betas[15]*x_test[:,14] +
betas[16]*x_test[:,15] + betas[17]*x_test[:,16] + betas[18]*x_test[:,17] + betas[19]*x_test[:,18] +
betas[20]*x_test[:,19] + betas[21]*x_test[:,20] + betas[22]*x_test[:,21] + betas[23]*x_test[:,22] +
betas[24]*x_test[:,23] + betas[25]*x_test[:,24]
rmse = np.sqrt(np.mean((yHat2 - y_test)**2))
rmse

r_squared = lin_reg_train[2]
r_squared
```