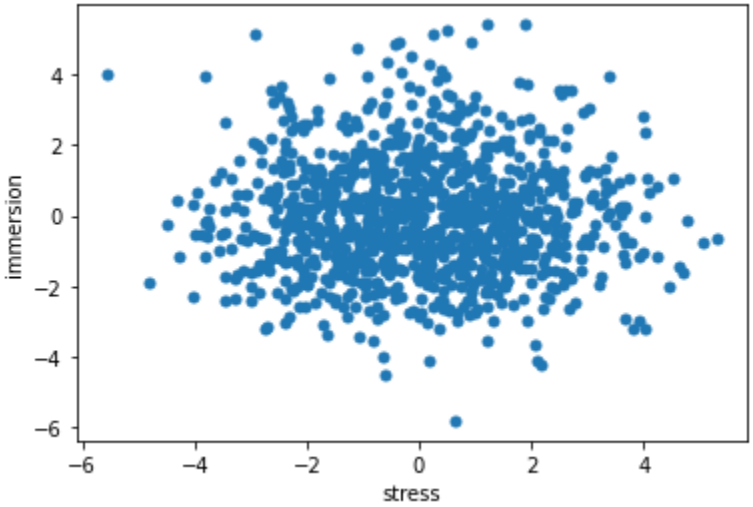
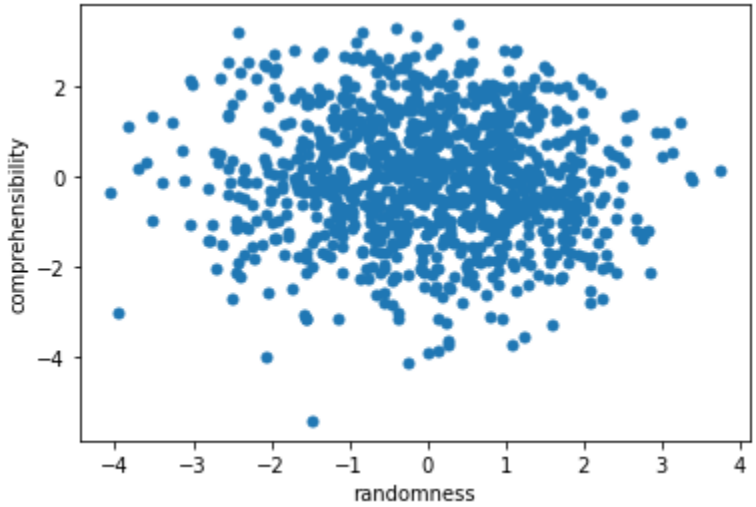
Tung-Yu (Tommy) Wu
Prof. Wallisch
2022/05/10

Capstone Project Report

Data cleaning: For initializing, I imported the data using NumPy and separated the data by directly slicing data points that were necessary to answer each question. For all Sensation, personality, and movie experience data, a PCA was conducted to determine useful factors to deconstruct the data. Each PCA was then met with the Kaiser criterion to determine useful factors. For compiling scatter plots, there may be dimension issues after PCA as to how the nans are distributed. I had to first compile the big data which fits all three data sets and eliminate nans both row-wise and column-wise to make sure they consist of the same dimensions for PCA analysis. For questions 8 to 10, in order to address the size issue, as I continuously encountered the issues of mismatched sizes and dimensions, I had to utilize pandas and systematically eliminate the nans from the data set. Rather than just eliminating the nans row-wise, I had to eliminate nans while pairing up a specific movie and PCA components.

1) For the first question, to determine the relationship between Sensation and Movie experience data, I utilized the Kaiser criterion and determined to utilize two major factors. For sensation data, there were 6 major points based on the Kaiser criterion, but only 2 for movie experience data, hence I had to keep the formalities of the same dimension and used only 2 for both. The two factor yields two scatter plots like the following: where the one to the right is a scatterplot using PC1s of Sensation data and movie data, whereas the one to the right is a scatter plot using PC2s of sensation data and sensation data, with movie experience data on the x-axis and movie experience data on the y-axis.

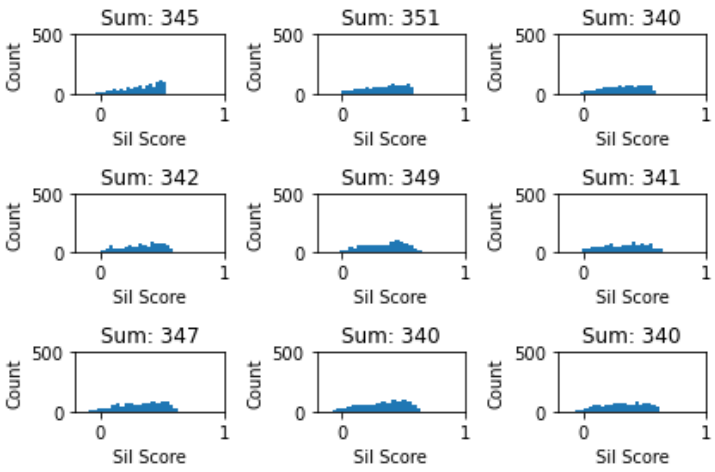


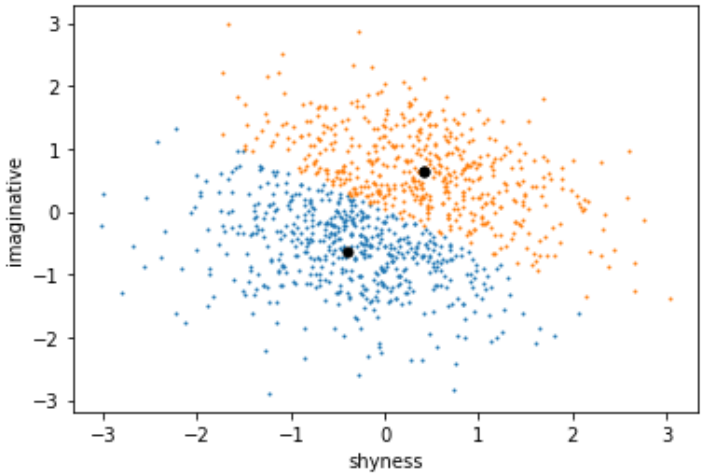


As we can see from the scatter plots, the relationship between the two seems clustered at the center, but there is a slightly slanted best fit line that goes upwards in both scatter plots. Thus it can be said that the two data demonstrate a small but positive correlation

between the two. Yet, this to the eye is extremely minimal and thus we cannot further conclude that there is a positive correlation between the two as both PCA correlations showed no obvious trends, being dead in the center. The two graphs looked even spread out from the center and thus lacked a solid foundation for correlation.

2) There is some evidence of personality types based on the data. Through the ideas of clustering, we can first isolate the talkative factor (for easier computation) as an outcome and navigate through the rest (as predictors) using PCA to determine usable factors. Under the Kaiser criterion, there are 8 total important factors and upon examination, by using PC3 and PC4 we can best summarize the data. These are the two that upset each other the most and split the total data into two categories. Later on, we compile K means by finding the best cluster using the silhouette method and yield that the data is best fitted to two clusters, hence two personality types. It is possible that the outcome of two personality types comes from a 2-dimensional chart, as an n-dimensional chart will exponentially reveal more information on the data.



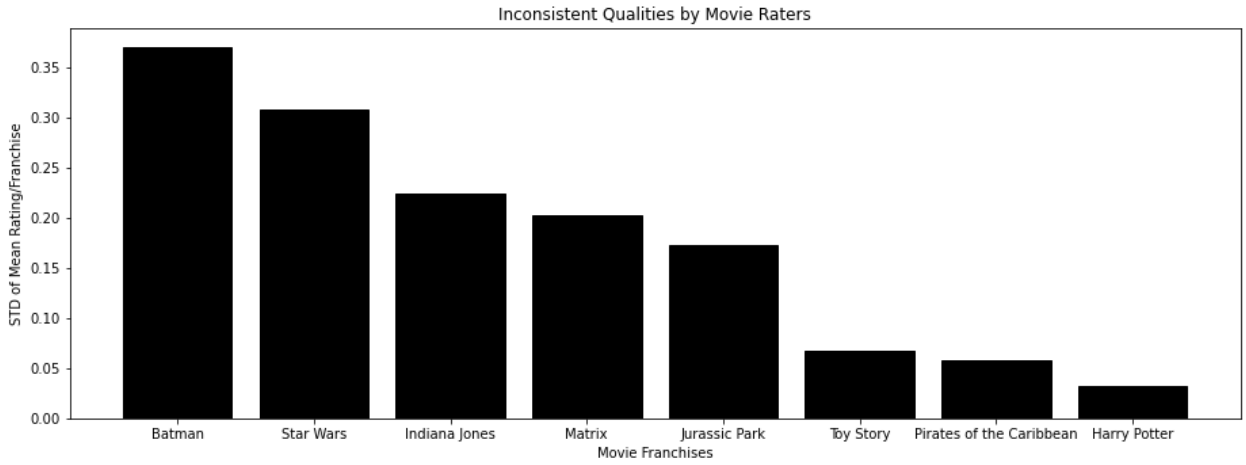


3) To answer this question we utilize the data entry itself, by having more inputs (fewer nans) we can determine the movie is more popular and vice versa, whereas 199.5 is the median cutoff between popular and unpopular movies. After such, we find out the means for each individual column (movie and their respective ratings) and then compile a student T-test, where the hypothesis is that there are no differences between the ratings and the alternate hypothesis is that there is a difference in favor of popular movies. With the independent T-test yielding a p-value of 7.95e-52, significantly smaller than alpha at 0.05, we can conclude that there is a significant difference where popular movies are rated higher. T statistics is at -17.62.

4) To answer this question, I first combined the data on Shrek and gender identity. Then I utilized the code of ~np.isnan to eliminate row-wise nans between the two columns. Then I sorted the data based on gender identity, making it in ascending order and taking T-tests between data for Shrek from respectively gender claims of 1 or 2 ignoring the 3(by

simply slicing them out). The p-value came back to 0.26377, and T-stats at 1.118, with alpha at 0.05, the p-value is greater than alpha thus we have no evidence to reject the null hypothesis which is that Shrek is not viewed differently between genders. The alternative hypothesis here is set to two-sided.

5) Similar to question 4, the methodology conducted here is the same, I first isolated the data, movie rating for Lion King, and also data for an only child, then I again got rid of the nans row-wise and sort data. After such, I was able to conduct a clean slicing to get rid of the -1s and do a T-test on the data. The p-value returned from the test was 0.0178 and the T-stats was at 2.103. With alpha at 0.05, the p-value is smaller than alpha hence we have evidence to reject the null which is that Lion King is viewed differently by those who have and do not have siblings. The alternative hypothesis set here was that people without siblings' enjoyment is greater than people with siblings'
6) Similar to questions 4 and 5, the same methodology was conducted here. The output of the t-test returned a p-value of 0.937 and t-stats of -1.53. With alpha at 0.05, we fail to reject the null hypothesis that there is a significant difference between people's enjoyment of "The Wolf of Wall Street" based on their movie watching preference of being alone. The alternative hypothesis here is set to people who like to watch movies and socially enjoy them more.
7) For the discoveries of inconsistent qualities within the movie franchises, I took the standard deviations of means between each rating, omitting the ones that only rated partially (i.e, only rated 3 out of the 4 movies in the franchise). The means of each movie are then stored into an array to compile into individual movie standard deviations and then compiled into a total list of all franchises to compile the graph. The movie franchise that has the biggest inconsistency according to the raters is Batman, with a standard deviation behind the three franchise movies being 0.370. Second place came in with Star Wars at 0.307 between the six movies. The movie with the most consistent quality according to the data was Harry Potter, which yielded a 0.0318 standard deviation between the four movies. The graph depicts a descending order for better readability.



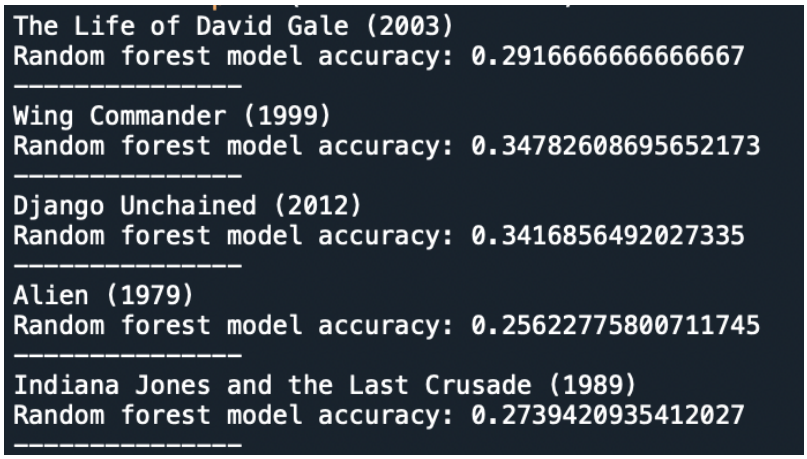
8) For the initiation of this question, for personality data alone, there are 43 columns of data thus warrants some dimensional reduction. A PCA was conducted here and it points to doing 8 factors as I used the Kaiser criterion. For the actual prediction portion, I utilized the random forest method. By using $clf = RandomForestClassifier(n\_estimators = 100).fit(x, y\_transformed)$ I was able to get predictions per personality data and determine the best fit for the candidate. For prior works, I had to first compile pcaperso to be a dataframe that stores all 8 factors of the PCA, and use it as the predictor, then I had to compile all data from 400 movies into a dataframe and stack with pcaperso. By doing this this allows me in my for loop to conduct the argument listed above. Then by utilizing the data, I split the data into 80% training and 20% testing, to cross-validate the model. To further determine the accuracy of the model, I utilized accuracy_score to calculate each time a prediction was made and store them into a list which then is taken out and averaged for demonstration of the overall accuracy, 0.99975. This accuracy data was taken when the test set was in place and predictions were made accordingly. Sample of the output is as following:

```
The Life of David Gale (2003)
Random forest model accuracy: 1.0
---------------
Wing Commander (1999)
Random forest model accuracy: 1.0
---------------
Django Unchained (2012)
Random forest model accuracy: 1.0
---------------
Alien (1979)
Random forest model accuracy: 1.0
---------------
Indiana Jones and the Last Crusade (1989)
Random forest model accuracy: 0.9976133651551312
---------------
Snatch (2000)
Random forest model accuracy: 1.0
---------------
```

The cross validation method used here is the leave-one-out method as shown with a brief screenshot of the for loop of model.

```
modelacurrate=[]

for i in range (400):

    x_train = np.delete(x,i,0)
    x_test = x[i].reshape(1,np.size(x_train,axis=1))
    y_train = np.delete(y_transformed,i,0)
    y_test=y_transformed[i]

    clfmodel = RandomForestClassifier(n_estimators=100).fit(x_train,y_train)

    predictions = clf.predict(x_test)
    print("prediction for this candidate is "+ predictions)

    modelAccuracy = accuracy_score(y_test,predictions)
    print('Random forest model accuracy:',modelAccuracy)
    modelacurrate.append(modelAccuracy)
    print('---------------')
```

9) Similar procedures were conducted here on question nine, but with us isolating out the only three criteria, PCA is not needed for this particular question. For this question, I also utilized the random forest method with cross validation of training and test set being on an 80/20 split. I also used the leave-one-out method to finalize the model accuracy. Compared to question 8, the model accuracy for prediction is surprisingly poor, yielding only a 0.3007 of accuracy, not even to the halfway mark. The following is the screenshot of example output



The accuracy of the model here is similar to how it was done in question 8, for each iteration, the accuracy of the model was stored in a list, then we simply take the sum and divide it by len(data) to get an overall accuracy. Model accuracy comes from the function accuracy_score.

10) For this question, the approach was basically identical to the ones done in question 8, the only difference this time is that we get to use all the factors. For initialization, I started by isolating the factors in the data and compiling them into a dataframe where a row-wise nan elimination is then performed. After doing so, I was surprised to see that there were still 946 rows of available data, meaning that 946 people out of 1098 answered every factor question. I then conducted a PCA on this and again used the Kaiser criterion, where it denoted that there are 18 important factors to be accounted for. The predictor here is thus changed to 18 factors by utilizing the rotatedData I got from doing the PCA, and making them into a dataframe for later analysis. The y here is again the movie ratings. Similar to 8 and 9, the prediction for loop also starts by eliminating row-wise nans through pairing up specific movies with the 18 factors. For cross validation, again, I also initiated a 80/20 split between training and testing sets and conducted the leave-one-out method to compile data accuracy. The output of accuracy_score overall was not as surprising as all factors are able to be used; the returned value of 1 is expected. The following is the example output and brief snapshot of code used for this question.

```
Analyze That (2002)
Random forest model accuracy: 1.0
---------------
Braveheart (1995)
Random forest model accuracy: 1.0
---------------
Inception (2010)
Random forest model accuracy: 1.0
---------------
Groundhog Day (1993)
Random forest model accuracy: 1.0
---------------
The Lookout (2007)
Random forest model accuracy: 1.0
---------------
21 Grams (2003)
Random forest model accuracy: 1.0
---------------
```

```
from sklearn import preprocessing
for i in range (400):
    dataset=pd.DataFrame.merge(pcaall,Ftitle[title[i]], left_index =True , right_index = True,how = 'inner')
    dataset.dropna(inplace= True)
    lab = preprocessing.LabelEncoder()
    x = dataset.iloc[:,:18]
    y = dataset.iloc[:,18:].to_numpy()
    y_transformed = lab.fit_transform(y)

    x_train, x_test, y_train, y_test = train_test_split(x, y_transformed, test_size = 0.2, random_state = 0)

    clf = RandomForestClassifier(n_estimators=100).fit(x_train,y_train)

    predictions = clf.predict(x_test)

modelacurrateQ10=[]

for i in range (400):

    x_train = np.delete(x,i,0)
    x_test = x[i].reshape(1,np.size(x_train,axis=1))
    y_train = np.delete(y_transformed,i,0)
    y_test=y_transformed[i]

    clfmodel = RandomForestClassifier(n_estimators=100).fit(x_train,y_train)

    predictions = clf.predict(x_test)
    print(title[i])

    modelAccuracy = accuracy_score(y_test,predictions)
    print('Random forest model accuracy:',modelAccuracy)
    modelacurrateQ10.append(modelAccuracy)
    print('---------------')
```

11) For the extra credit, I wanted to look at preferences of people, specifically, the preference of two polar extremes and try to see if there are any similarities between the two. For the data, I chose to compare people that are quiet and people who curse and yell at scenes in the movie. By doing a Spearman correlation between the two (as they are ordinal values), I found that they have a correlation of -0.09, which was expected. But with the addition of another variable, something unexpected showed up. With the addition of whether or not a person enjoys movies alone, the correlation between people who curse and being alone came back at 0.025, and the correlation between people who tend to be quiet and being alone came back at 0.044. The spearman correlation ranges from -1 to 1 with each end being exact monotonic, 1 meaning as x increases y also increases and vice versa.This somewhat points to that possibility that deep down there are similarities between people who are drastically different. Of course there are some confounders that should explain such data, but taking it at face value generates an interesting undertone in the data and in humans, which was interesting to me.