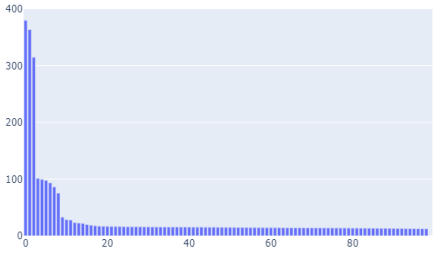
MATH 122
Final Project - Report

**Looking at the Data:**

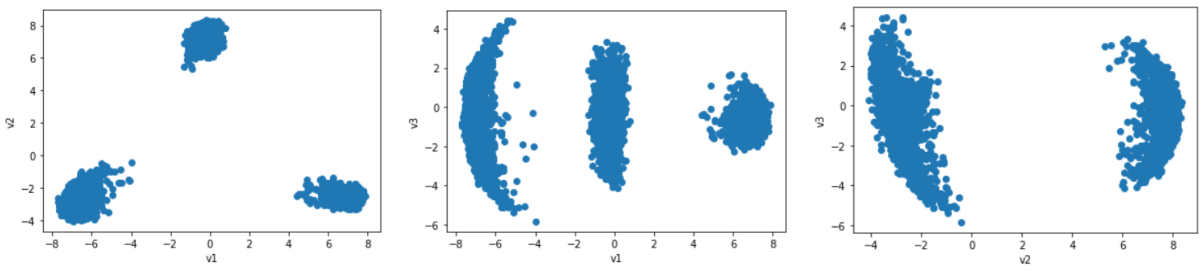
We have been given 2 datasets, one called user_history and the other called user_ratings. User_history contains the data for 4500 users and the amount of time they spent on a website per day. Each user is given a unique ID and each row corresponds with said ID. Each column is a different website and each cell under a given column is a numeric measurement of engagement with a given website by each given user. Our other dataset, the user_rating dataset has 33725 rows and 3 columns with one for userID corresponding to those under user_history, product name, and product rating. Since the IDs under userID are and the names of the products under products are not unique. After reorganizing the data so that each unique ID corresponds with a unique product, we see there are 3000 users and 75 products with ratings from 0 to 10. Some products were not given a rating by a user and 1500 users are missing in this data set. We will try to predict the missing values in user_ratings given the information we know about the users in user_history.

**Approach:**

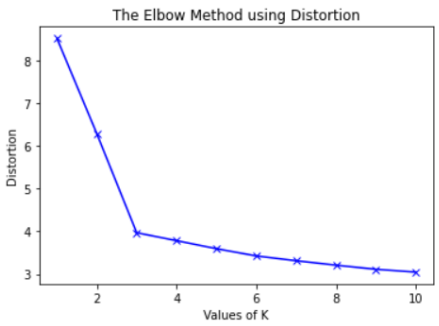
First we standardized his_M so that all variables have the same standard deviation so that they have the same weight and PCA returns the true principal components. We performed a SVD and plotted each singular value against the number of components and observed the top three were the most relevant.

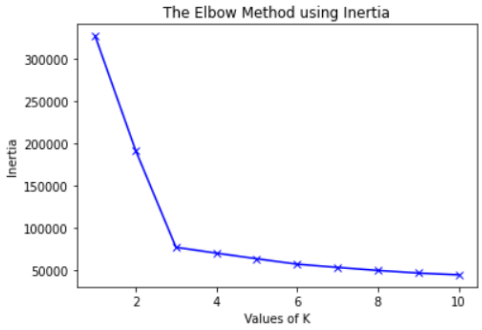



We tried to visualize the data by looking at combinations of the top three principal components.

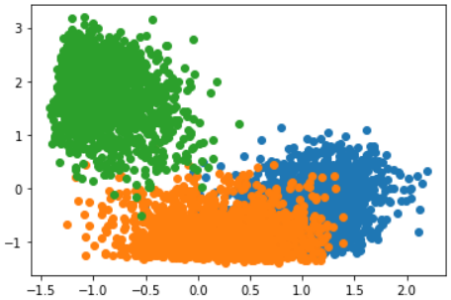


Our 2-D visualization shows that it looks like there are 3 groups of users in our user_history dataset. We also performed an elbow method to find the optimal k-means, which confirmed that it was 3 clusters.



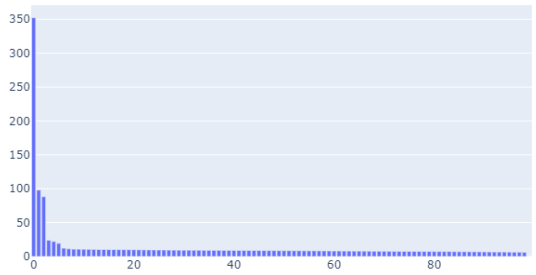




We used k-means as it is a common way of dividing the users into groups with common features.

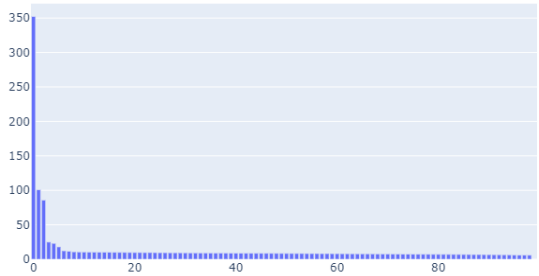


To keep track of which group each user belonged to, we added specific labels for each cluster. Next, we performed a PCA on each cluster to reduce the dimension where we found the top three components of each cluster had the most variation.

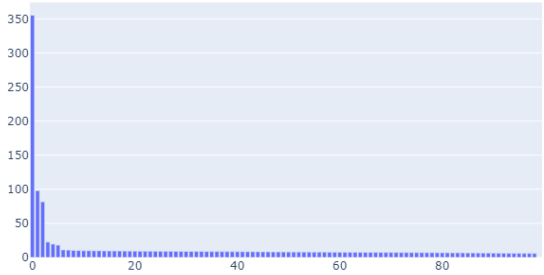
To keep track of which group each user belonged to, we added specific labels for each cluster. Next, we performed a PCA on each cluster to reduce the dimension where we found the top three components of each cluster had the most variation and calculated the covariance matrix of each cluster. We also notice that the next 3 principal components also look to have a larger effect than the rest of the 100 components so we also decide to look at 6 components and compare the differences in results.




cluster 1




cluster 2




cluster 3

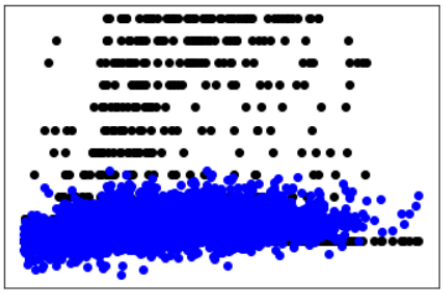
We put the processed cluster back in place and made sure that each of the data points still have the usersID label.

Part 2. Matrix Completion

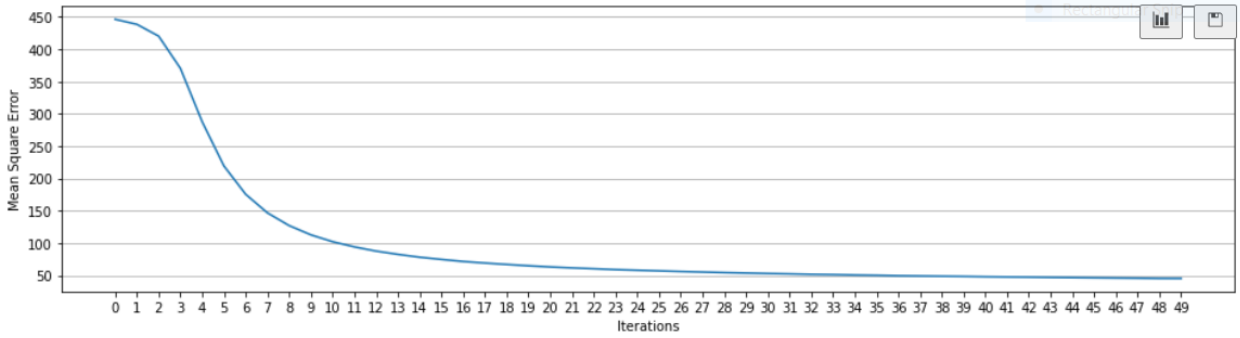
Next, we moved into the analysis of user_rating data. We obtained the data frame of user_rating data given in the template, with products on the horizontal axis, and user ID on the vertical axis. The entries are filled with ratings given by each user to every product. Then we convert the data frame to a matrix and replace all the “NaN”s with “0”s. We use the reorganized matrix called user_rating_table. Since we want to use the users from the user_history dataset to predict the ratings of the users in user_rating, we first only look at the userIDs that are in both datasets. There are 1500 users in the user_history dataset who did not provide a rating to the products in the user_ratings dataset. Since their data could not be used towards the training part, we got rid of them and left with the 3000 users who did rate. After that, we apply the standard regression model to the history data to obtain the prediction based on the full features of the user_history data. The coefficient matrix is:

```
[-0.15627289  0.10045419  0.18088314  0.09840717 -0.09606579  0.05738816
-0.22745738 -0.27879114 -0.19402719 -0.01781725 -0.07629769 -0.09640856
-0.01711316  0.14183769  0.43987017 -0.42548302 -0.05370199 -0.14138967
 0.05094748 -0.02990476  0.28095832 -0.18766124  0.11833953 -0.09715069
-0.2935032  -0.18047345  0.07887989  0.00388338  0.01789712 -0.32201804
 0.32117282 -0.18887334 -0.08628224 -0.24821569  0.22285391 -0.40863919
-0.03985729  0.00867417  0.02053839  0.14957277 -0.21544793 -0.01105673
-0.1755516  -0.17677167  0.30869235 -0.11967231  0.16973678  0.11320999
 0.24170644  0.06175494  0.1468441  -0.12045702 -0.15932461  0.08806965
-0.35092353 -0.26511454  0.13509404 -0.00666074  0.04637064  0.02449703
 0.19987408 -0.29935342  0.20086279 -0.08567959 -0.12313557 -0.00153406
 0.11343604  0.03314426  0.07390504  0.08724655  0.24524159 -0.011683
-0.15538853 -0.10308439  0.2464226  -0.03533681  0.10375116  0.15805232
 0.23195298 -0.04556665 -0.21520984  0.14486738  0.30893841  0.11451947
-0.06227093 -0.30669826  0.13316826  0.16626935  0.11463723  0.0129384
 0.25811521 -0.10697088 -0.07850355 -0.05993794  0.13414708 -0.25522624
 0.34168806 -0.0524449   0.29959986 -0.10566391]
```

We got a mean squared error (MSE) of 4.329353826660119.

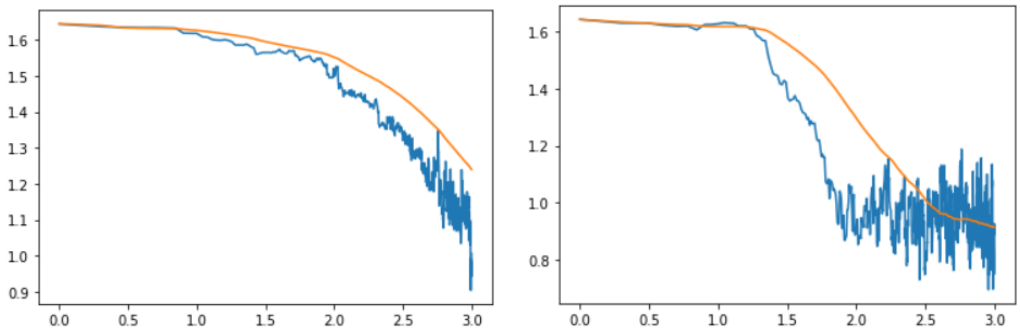


Next, we have created a matrix completion class that implements stochastic gradient descent. We use this to find the approximated user_ratings matrix. To see if the approximation is good, we look at the MSE.




As the code continues to iterate, we see that the MSE does converge, meaning that the predicted points are close to the actual data. By comparing the matrix completion and the original incomplete rating matrix, we found that the ratings of those entries in the prediction matrix is very close to the ones in the original matrix.

We then look to solve $\hat{\beta}$when there are six features and when there are three features. Then we went through gradient descent to optimize the solution. We implemented a convergence test, which showed that the error is converging.




6 features 3 features

Also, the mean square error between the prediction and the result we directly obtained from least square regression is small for both cases (six features and three features).

MSE for 6 features: 0.36310863348286493
MSE for 3 features: 0.12906597918191196

Then we looked at the MSE between the prediction and the original data for given ratings in the initial user_ratings dataset.

MSE for 6 features: 5.229419547647396
MSE for 3 features: 5.039449124719974

The MSEs show that predicting with 3 features is more accurate. Finally we used the prediction with 3 features as an output and exported it to the prediction_of_ratings.csv file.

**Conclusion**:
We used techniques like PCA, gradient descent, and matrix completion to create a prediction for what a user would rate a specific product using other information we know about these users. We first group users with similarities using k-means clustering. Then we reduced the dimensions using PCA to keep the most relevant features. We decided to look at the difference between keeping three features versus six features and how it would affect the predictions we get. We performed a linear regression on the users that were in both datasets. Then we calculated a $\hat{\beta}$for the 6 features case and 3 features case, optimizing it through gradient descent. Looking at the MSE, we saw that using 3 features provided more accuracy so we used that to calculate our final 4500 by 75 ratings prediction matrix. Although using 3 features led to the best accuracy in our

process, the MSE is still quite large when comparing the predicted ratings to the original data for given ratings in the initial user_ratings dataset. This shows that even though our code has given a decent prediction, there are things we could do to improve on it. Maybe we could instead reduce the dimensions before we cluster the data or even try a different prediction method.