Brody Anderson

Math 5750/6880: Mathematics of Data Science

Project #2 Final Report

October 7, 2025

My GitHub Project 2 repository is located here:

https://github.com/brodyanderson/math-6880-Project2

1. Clustering Gaussian Blobs using k-means

According to the project instructions, I performed a k-means cluster analysis with k=5 on the "Gaussian Blobs" dataset. The smallest inertia value, of the 5 blobs, came back as 179.318. To get a better idea of what the k-means analysis was clustering, the 2D visualization of the clusters via PCA is provided below.

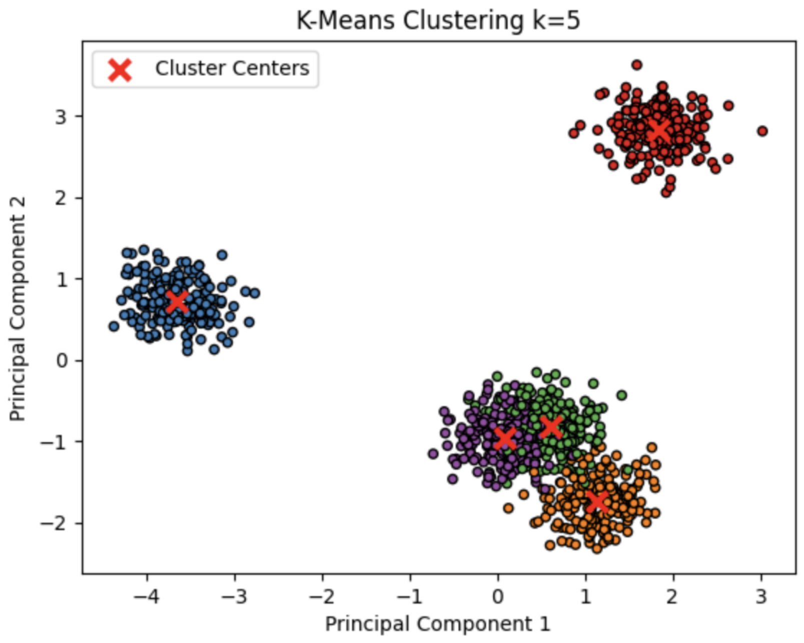


FIGURE 1. K-Means Clustering — PCA

From Figure 1, it looks like three of the clusters are very similar to each other, at least when looking at the PCA reduced 2-dimensional graph. However, when looking at the confusion matrix, we see that they are actually quite distinct.

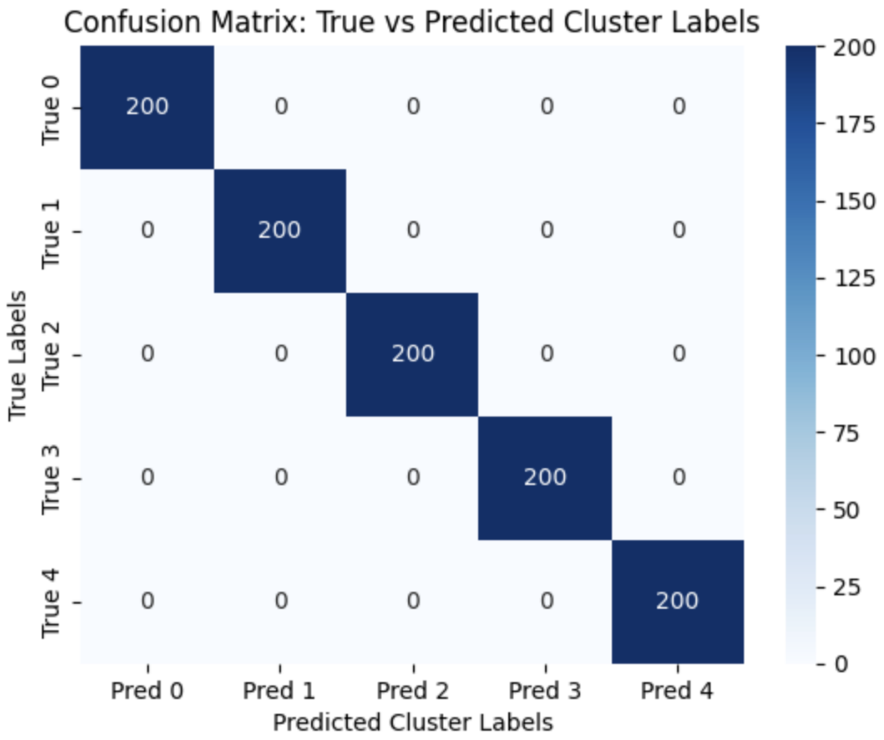


FIGURE 2. K-Means Confusion Matrix

According to Figure 2, each of the k-means clusters perfectly groups together the blobs that were created by the Guassian blob generator. It is important to note that the model did not assign prediction group 0 to the true group 0 (or any other group for that matter), but simply clustered similar data points together. In order to match the predicted labels to their true label counterparts, I employed linear_sum_assignment(). This function implemented the Hungarian Algorithm to find the optimal one-to-one mapping between predicted and true labels for the confusion matrix. The result yields the output above, and shows that the k-means model perfectly clustered all of the data together into their correct blob.

Next, I performed an elbow analysis to determine if 5 clusters were the optimal number of clusters in this setting.

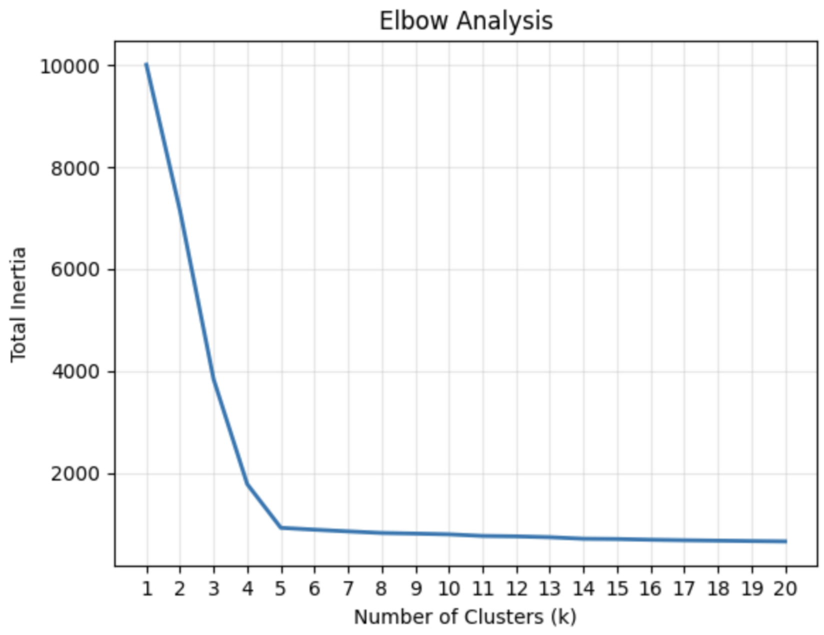


FIGURE 3. Elbow Analysis

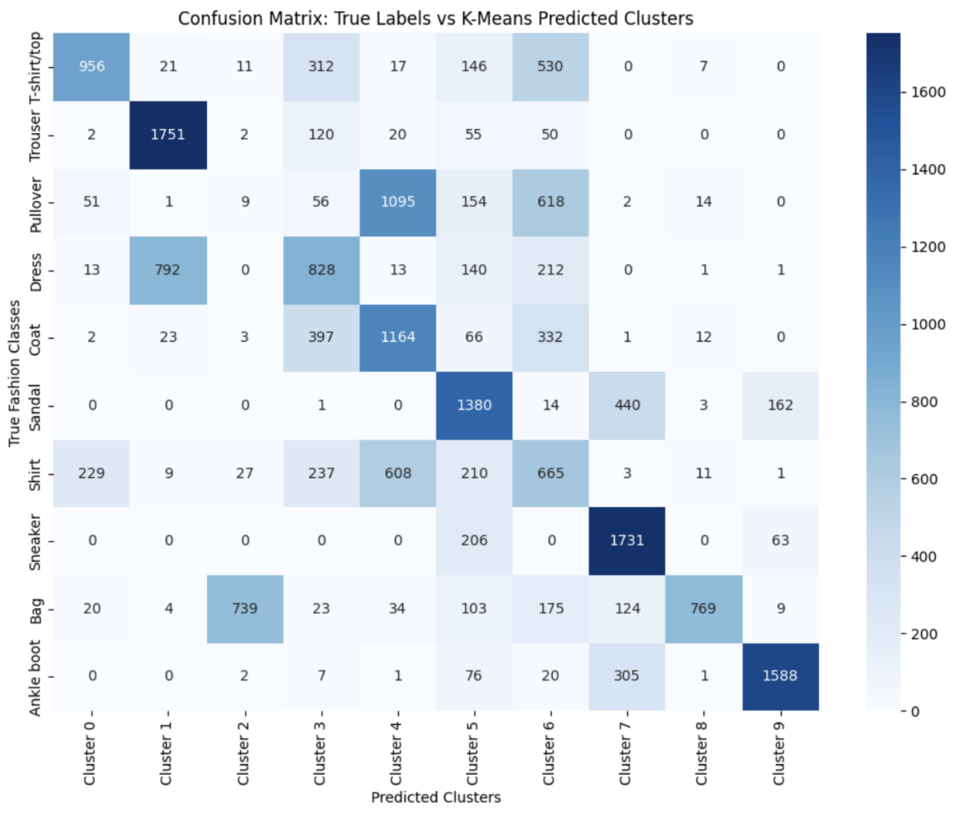
The figure above clearly shows that the total inertia when k={1,2,3,4} is quite high, and drops drastically as k increases, until k=5. The total inertia hardly decreases for each additional k after 5, signaling that k=5 is the optimal choice for the k-means model on this data, as there are very diminished returns for each additional cluster.

2. **Clustering Fashion-MNIST using k-means**

The Fashion-MNIST dataset contains 70,000 different pictures of clothing. Each image (or row) has a corresponding label, that groups the image into a class of similar clothing items. The class labels include: t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot, each of which has a corresponding class value 0 through 9. Every picture contains 28 pixels in height and 28 pixels in width, forming a 784 pixel image.

My purpose was to use k-means clustering to group each image, based off of the information contained within the 784 dimensions of the dataset. However, because the dataset is so large and my computer cannot handle such massive computation, I sampled the data. In order to preserve as much information as possible about each image, I left the 784 dimensions alone and only sampled the 70,000 rows of pictures. The full dataset contains 7,000 images per class, so I randomly sampled 2,000 images per class. This resulted in a sample dataset with 20,000 rows and 784 dimensions per row. The corresponding labels were included as well.

Given that I knew the number of groups in the dataset was equal to 10, I first employed a k-means model with k=10. As always, I scaled the data using StandardScaler() prior to making the model. Using the same method described in part 1, I created a confusion matrix to see how well the model clustered the data into each class.



	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9
T-shirt/top	956	21	11	312	17	146	530	0	7	0
Trouser	2	1751	2	120	20	55	50	0	0	0
Pullover	51	1	9	56	1095	154	618	2	14	0
Dress	13	792	0	828	13	140	212	0	1	1
Coat	2	23	3	397	1164	66	332	1	12	0
Sandal	0	0	0	1	0	1380	14	440	3	162
Shirt	229	9	27	237	608	210	665	3	11	1
Sneaker	0	0	0	0	0	206	0	1731	0	63
Bag	20	4	739	23	34	103	175	124	769	9
Ankle boot	0	0	2	7	1	76	20	305	1	1588

FIGURE 4. Fashion-MNIST K-Means Confusion Matrix

When looking at the confusion matrix, we hope to see a dark blue along the diagonal. This would illustrate that each cluster grouped a large number of clothing items into their correct classes. However, there are several clusters have an even mix of items grouped together. For example, cluster 6 groups many items from the t-shirt/top, pullover and shirt classes together. When looking at individual images of clothing items in each of these classes, the pictures look similar to one another. Logically, this makes sense as t-shirts, pullovers and tops are all clothing items worn on the upper body and resemble each other in shape and appearance. Thus, the classes likely have similar dimensional values in the dataset, which is reflected in the confusion matrix above.

Another way to view this model is to look at how well each class is grouped together. Essentially, look across the matrix instead of up and down the columns as we explored in the preceding paragraph. Sneakers, trousers and ankle boots were largely grouped together, even if the corresponding cluster contained other items.

Even though we know there are 10 classes in the set-up, it is interesting to see how a different k value would look in this case. For example, maybe with fewer clusters, the model groups t-shirt/top and shirt together really well. An elbow analysis was performed on the sample data to explore this possibility.

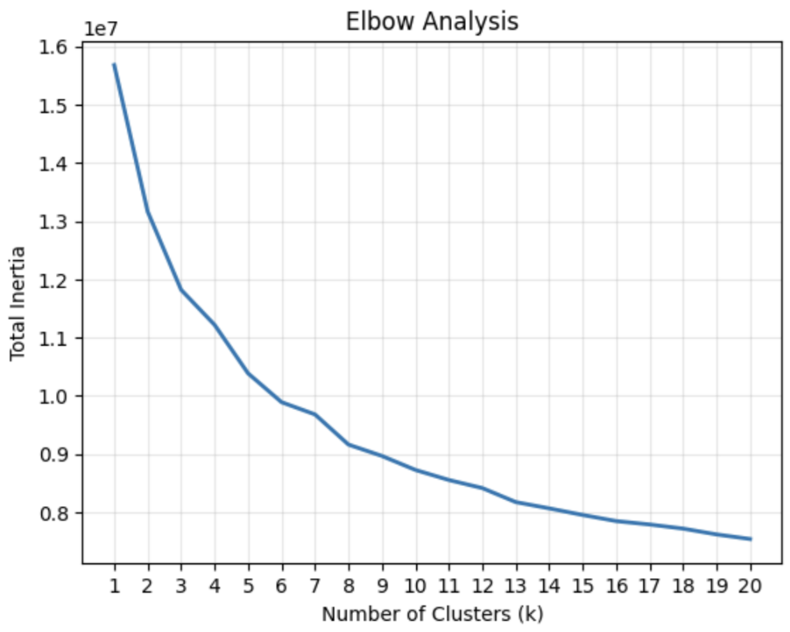


Figure 5. Fashion-MNIST Elbow Analysis

Figure 5, however, shows that 10 is actually a good number of clusters for the k-means model on this dataset. There is not a stark contrast in total inertia for any increase in k—like the elbow analysis from the Gaussian blob data. However, we do see diminishing returns in a sense as k increases. Given the graph, k=10 seems to be a valid choice for the model, and k-means does an OK job at clustering the data.

3. **Dimensionality reduction for Fashion-MNIST**

As explained previously, the Fashion-MNIST has 70,000 rows of data with 784 columns per row. My laptop simply cannot compute the optimal projection for such a large space, especially for 5 different k values (10, 20, 50, 100, 200). Consequently, I used the same sampling method described in section 2, but I sampled 250 rows in each class, resulting in a sample dataset with 2500 rows, while keeping the 784 columns for each row untouched. In order to compute the correlation between pairwise distances from the original standardized space and the reduced space, I compared the euclidean distances for each k by way of a for loop. I looped both PCA and random projections to determine the correlation coefficient for each approach at the given k value. The results are presented below.

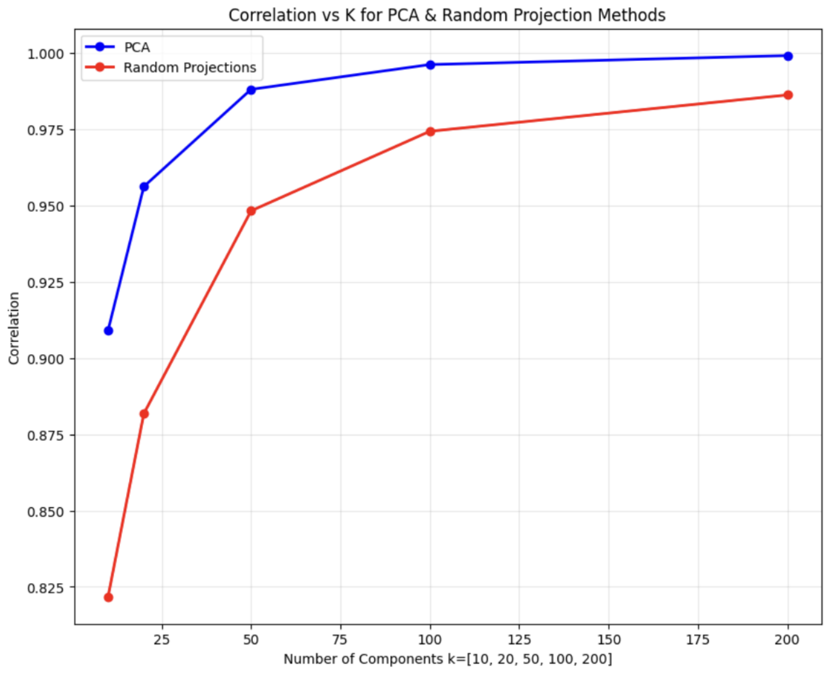


Figure 6. Correlation vs K — PCA and Random Projection Methods

Unsurprisingly, as the number of components increases for both PCA and random projections, their correlation with the original standardized space increases as well. However, the key takeaway from this graph is understanding how well random projections performs in comparison to PCA. At k=10, the random projection method returns a correlation value of 0.8218, which is much worse than its PCA counterpart at 0.9091. However, at k=100, PCA is 0.9962 and random projections is 0.9743. As the number of components increases, random projections performs better and better, shrinking the gap between the two methods. Additionally, because PCA finds the optimal projection for the space, it takes much more computing power than using random projections. With similar performance at high k values, random projections is a great alternative for dimensionality reduction.

4. **Clustering Fashion-MNIST using spectral clustering**

In order to accurately compare the two approaches (k-means vs spectral clustering), the same sample was taken from the Fashion-MNIST dataset. I used the same random seed and obtained the exact number of samples from each class as performed previously. Once the sample was obtained, a spectral clustering model was fit using affinity='nearest_neighbors' and setting n_neighbors=10. I tried to use the default affinity='rbf', but my laptop could not handle that parameterization. Notwithstanding, a confusion matrix for the spectral clustering model is presented below.

When comparing the above confusion matrix to Figure 4, we see that a greater number of images are correctly clustered together. However, because comparing the two confusion matrices is pretty difficult, purity scores were also computed to better understand the models. The purity scores essentially look up the column for each cluster and determine how concentrated, or homogeneous, each cluster is. For example, in Figure 7 we see that Cluster 1 contains 1736 trousers, 6 dresses and 1 coat. This returns a purity of 99.6% as the cluster is largely comprised of trousers. The purity scores for both k-means and spectral clustering are also provided below.

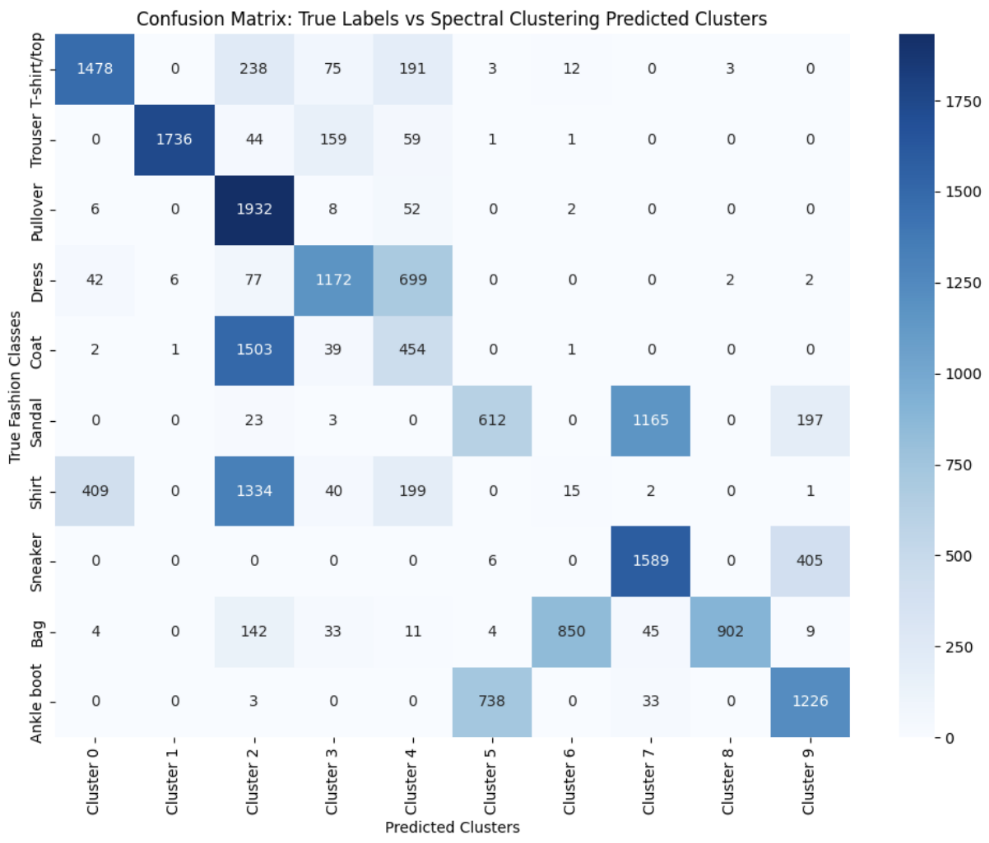


FIGURE 7. Fashion-MNIST Spectral Clustering Confusion Matrix

	K-Means Clustering	**Spectral Clustering**
Cluster 0	75.1%	76.1%
Cluster 1	67.3%	99.6%
Cluster 2	93.2%	36.5%
Cluster 3	41.8%	76.7%
Cluster 4	39.4%	42.0%
Cluster 5	54.4%	54.1%
Cluster 6	25.4%	96.5%
Cluster 7	66.4%	56.1%
Cluster 8	94.0%	99.4%
Cluster 9	87.1%	66.6%
Mean	64.4%	70.4%

While comparing each cluster individually may not be the best method to compare the two models, looking at the average purity for all clusters reveals that spectral clustering produced more homogeneous clusters on average. Given that spectral clustering groups clothing classes together in higher concentration, it is safe to conclude that spectral clustering performed better than K-means on this data.