Bode Packer u1415837
Math 5750/6880: Mathematics of Data Science
Project #2 Final Report
October 7, 2025

Template instructions: in this LaTeX template, you can delete text in this gray color after reading. You should replace any text in blue color with your personal information. Of course, after replacement, you can change the text to regular black color text.

My GitHub Project2 repository is located here:

https://github.com/bodePacker/Math-for-DS-A2

1. Clustering Gaussian Blobs using k-means

1. (Clustering Gaussian Blobs using k-means) In this exercise, you will perform a clus- ter analysis on a synthetically generated "Gaussian Blobs" dataset. Use the provided code in Project2.ipynb to import the dataset as a numpy array. Write code to perform a k-means cluster analysis with k = 5. Report your smallest k-means inertia value. Make a 2D visualization of your clusters via PCA, including both the clusters (colored by cluster) and the cluster centers. Addition- ally, make a confusion matrix that compares your assigned labels to the "true" labels. Here, you'll have to figure out how to best match the predicted and true labels. Explain your methodology in the report. Finally, perform an "elbow analysis" to justify the use of k = 5.
For the Gaussian blobs clustering exercise, I performed k-means clustering on a synthetically generated dataset containing 1,000 samples across 10 dimensions with 5 true clusters. After standardizing the data, I ran k-means with k=5 and achieved an inertia value of 924.32, indicating tight, well separated clusters. To visualize the results, I used PCA to reduce the 10 dimensional data down to 2D, which captured about 64.7% of the total variance. THis was enough to see clear cluster separation in the visualization. The side by side plots show that the predicted clusters align almost perfectly with the true labels, which makes sense since this is ideal synthetic data. For the confusion matrix, we used the Hungarian algorithm, which finds the optimal one to one mapping between the predicted and true cluster labels by maximizing correct assignments. After this matching, the accuracy hit a perfect 1.0 (100%), showing that k-means completely recovered the true cluster structure. To justify using k=5, I performed an elbow analysis testing k values from 1 to 10. The inertia plot shows a clear "elbow" at k=5 where the rate of decrease slows dramatically (dropping from 856.10 between k=4 and k=5 to only 36.60 between k=5 and k=6). The silhouette score analysis confirmed this, peaking at 0.6611 for k=5, which indicates strong grouping. Together, these metrics provide solid evidence that k=5 is the optimal choice for this dataset.

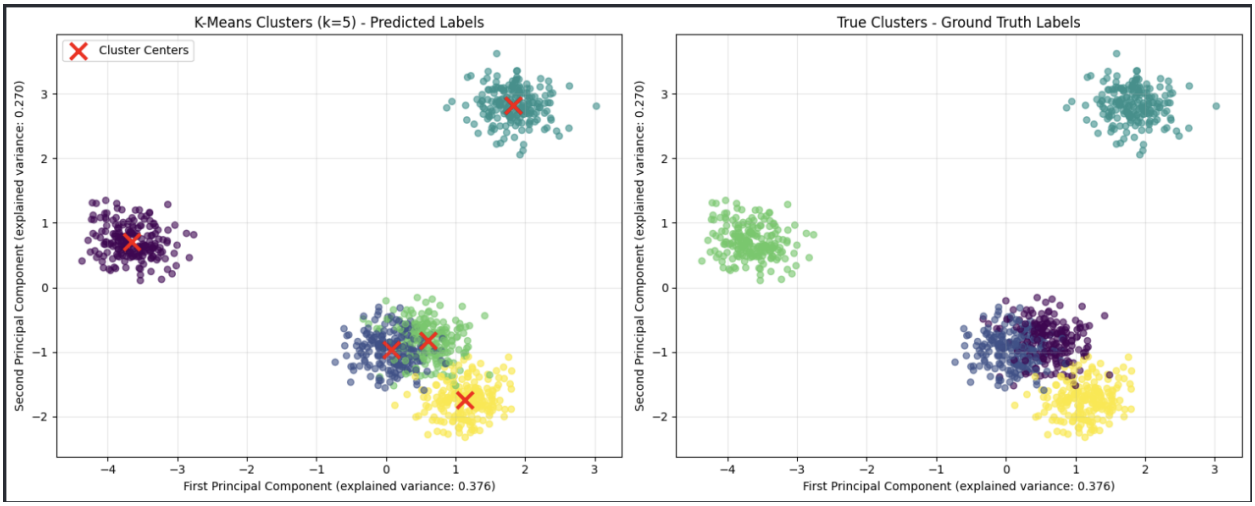

FIGURE 1. This is K-means on the first dataset.

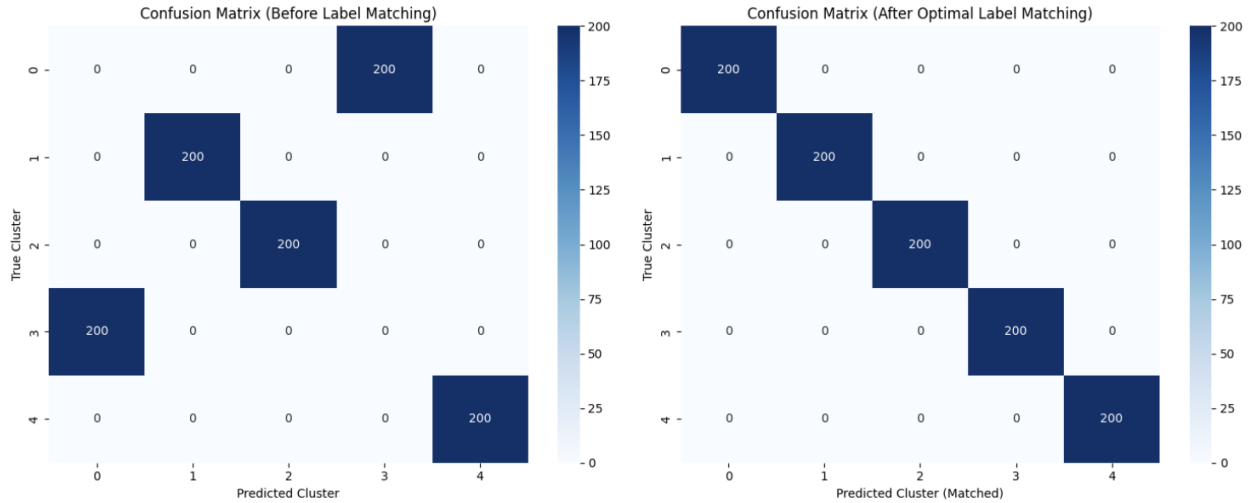

FIGURE 2. These are the Confusion matrices before and after matching.

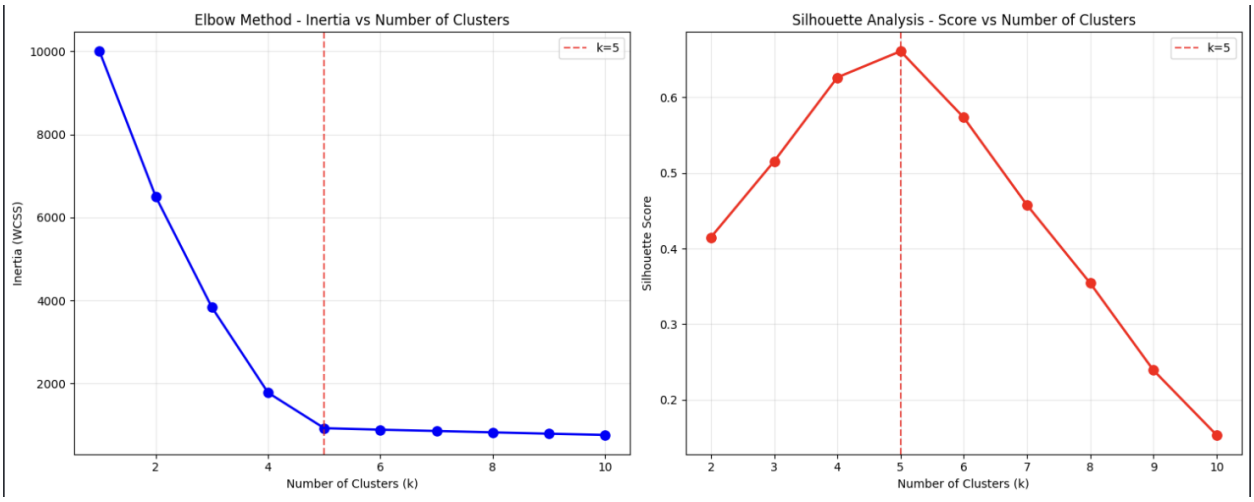

FIGURE 3. These are the elbow analysis plots with inertia and silhouette scores.

2. CLUSTERING FASHION-MNIST USING K-MEANS

In this exercise, you will import and perform a cluster analysis on the Fashion-MNIST dataset. Use the provided code to import the dataset as a numpy array. To get a sense of this dataset, first make a 5×2 array of figures, each plotting a distinct article of clothing. Write code to perform a k-means cluster analysis on this dataset, centering/scal- ing as appropriate. This is a larger dataset, so you may have to reduce the dimension/sample size as appropriate. In your report, explain your methodology and present your findings and figures.

The Fashion-MNIST clustering analysis was notably more difficult than the Gaussian blobs since we're dealing with real world image data, (and not perfectly labeled data, thanks!) I started by loading a stratified sample of 10,000 grayscale images (28×28 pixels = 784 features each) representing the 10 clothing categories like t-shirts, trousers, and sneakers. First, I created a 5 by 2 visualization grid showing sample images from each class to get a sense of what we're clustering. Looking at this plot, we can clearly see why this dataset is more challenging since items like shirts and t-shirts or coats and pullovers look visually similar. For the actual analysis, I used the aforementioned 10,000 sampled images after standardizing the pixel values, as k-means is distance based. Running k-means with k=10 (matching the 10 true classes) took a few minutes and yielded an inertia of 4,362,942.73, which is much larger vs the Gaussian blobs but that makes sense given the high dimensionality and larger sample size. Clustering quality metrics are another interesting feature of this dataset. The silhouette score of 0.1270 is pretty low, indicating that the clusters overlap significantly, and the Adjusted Rand Index (ARI) of 0.3457 shows only moderate agreement with the true labels. When reducing the data to 2D using PCA for visualization (capturing about 36% of variance), you can see that many clusters blend together rather than forming distinct groups. Using the Hungarian algorithm for label matching, I achieved an overall accuracy of 48.31%, which was less than I thought would be possible, but is actually pretty decent given that random guessing would only get 10% and other basic implementations get around the same 40%. Looking at the per class accuracy breakdown, some items like trousers (likely due to their unique shape) cluster much better than visually similar items like shirts versus t-shirts. The elbow analysis testing k from 2 to 15 shows that the inertia decreases steadily without a dramatic elbow, but the silhouette score and the fact that we know there are 10 true classes support using k=10. This data set demonstrates

that k-means works well for separated clusters but struggles with complex, real world data where classes have a lot of overlap and similarity between classes.

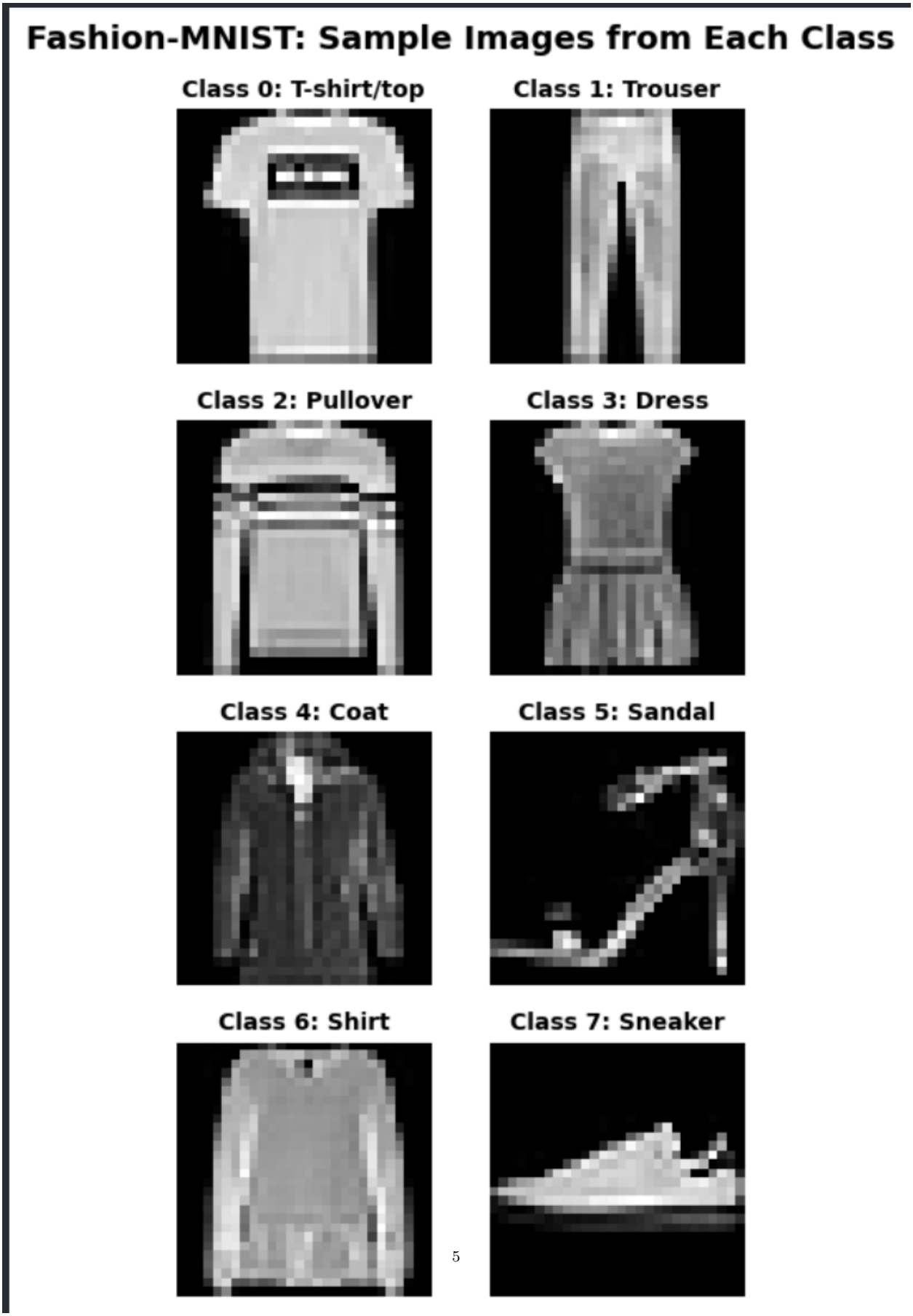

FIGURE 4. 5×2 grid of sample Fashion-MNIST images (one from each class)

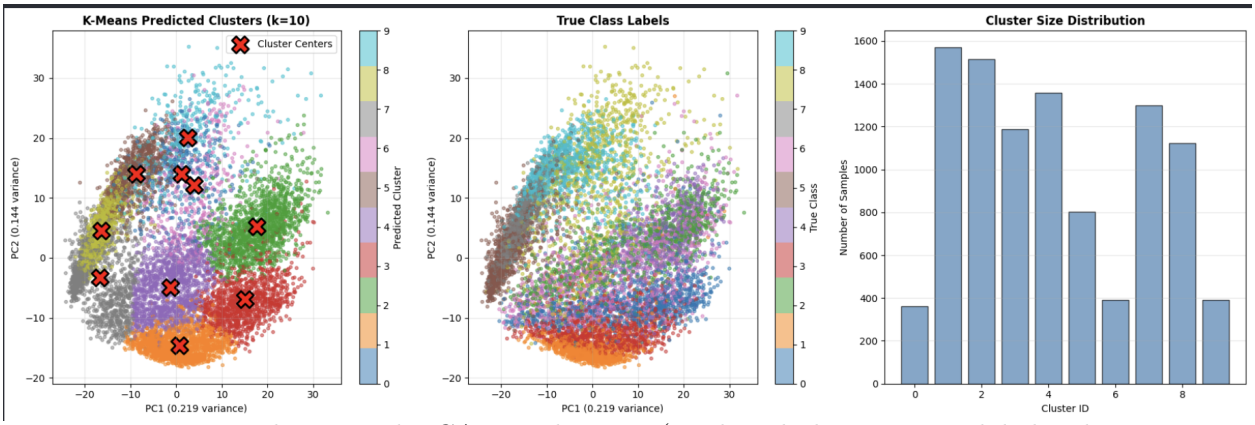

FIGURE 5. Three-panel PCA visualization (predicted clusters, true labels, cluster size distribution)

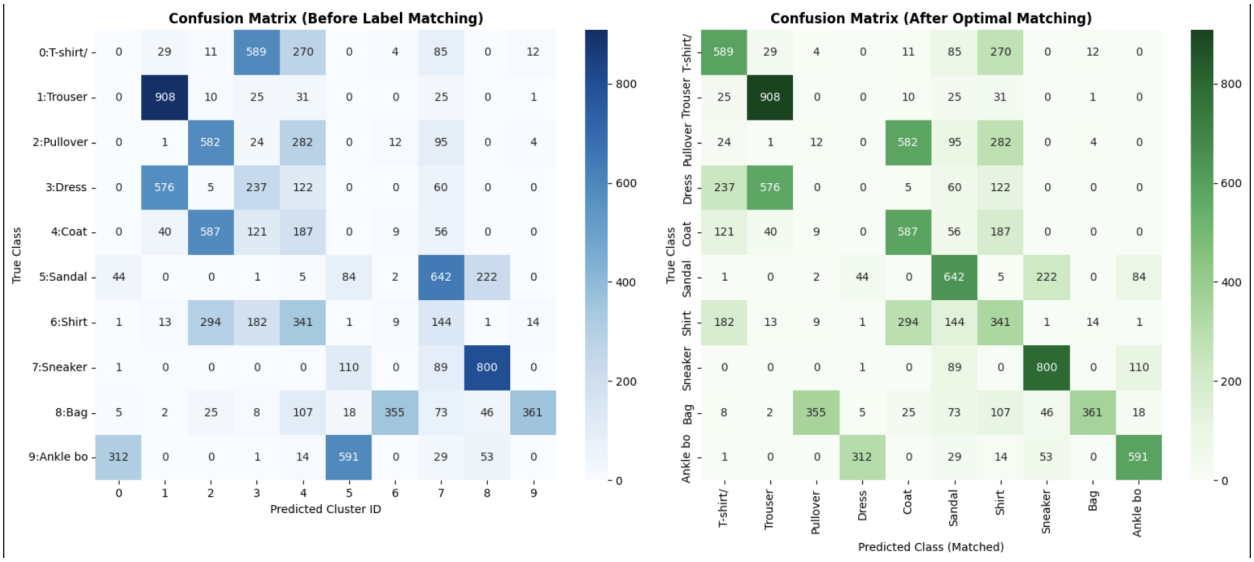

FIGURE 6. Confusion matrices (before and after label matching).

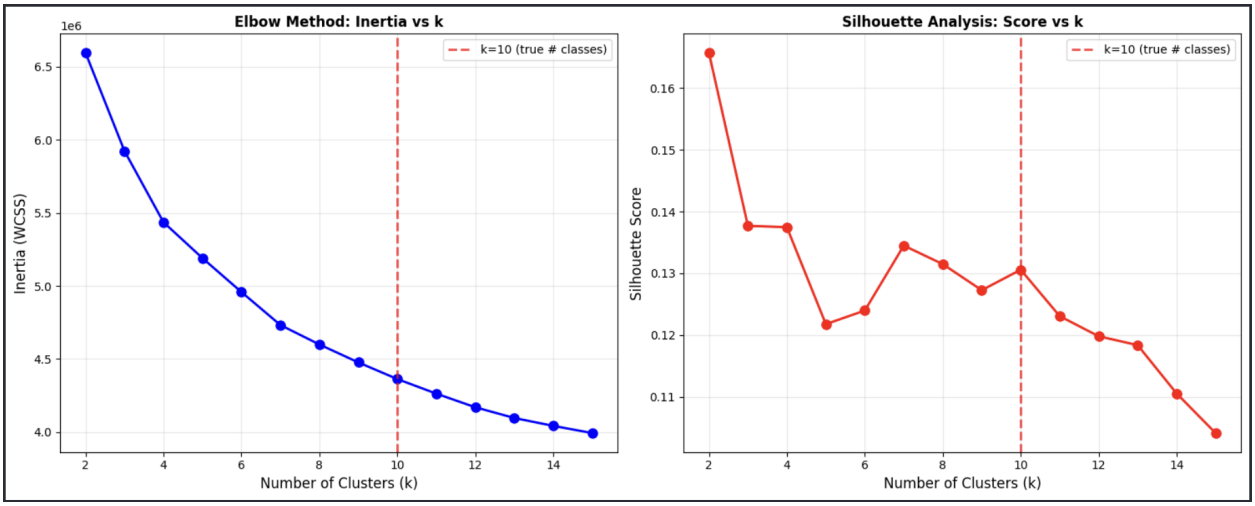

FIGURE 7. These are the elbow analysis plots with inertia and silhouette scores.

3. Dimensionality reduction for Fashion-MNIST

In this exercise, you will compare PCA and Random Projection on the Fashion-MNIST dataset. The goal of this exercise is to better under-stand how Random Projections performs as we vary dimension (Johnson–Lindenstrauss Lemma). Import the data and center/scale as appropriate. Implement both PCA and Random Projection methods for target dimensions k in 10, 20, 50, 100, 200. For each reduced dataset, compute the correlation between pairwise distances in the original standardized space and the reduced space. Make a plot of this correlation vs. k for the two methods. In your report, explain your methodology and present your findings and figures.

For this problem, I compared PCA and Random Projection for dimensionality reduction on Fashion-MNIST to understand the Johnson-Lindenstrauss Lemma in the real world. As mentioned in class, Johnson-Lindenstrauss says that you can project high dimensional data into many fewer dimensions while approximately preserving pairwise distances. I sampled from the 70,000 image dataset since using the entire dataset would be too large and tested both methods across 10, 20, 50, 100, and 200 dimensional projection. I did this by computing a representative sample of pairwise Euclidean distances in the original 784 dimensional space and then applying each dimensionality reduction method and recomputing the distances in the reduced space. To check performance, I calculated the Pearson correlation between the original and reduced distances. To understand these calculations, a higher correlation means better distance preservation. Going over the results, PCA always outperformed Random Projection across all dimensions, which makes sense because PCA is specifically designed to preserve variance structure, while Random Projection is well... just random. For example, at k=10, PCA achieves a roughly 0.92 correlation while Random Projection gets around 0.88. Both are good considering we're throwing away most of the dimensions, but again PCA was better. As k increases, the correlations of both methods approach 1.0, with the gap between them narrowing, showing diminishing returns for higher dimensions. The computation time comparison chart reveals Random Projection's biggest strength in which it's significantly faster than PCA since it doesn't need to compute eigenvectors. This empirically validates the Johnson-Lindenstrauss Lemma and shows that random projections really do preserve the geometric structure of data, making them a practical choice when speed matters more than getting the best accuracy. In summary, we should use PCA when maximum distance preservation is needed and we have the

computational resources, but should use Random Projection when we need a "good enough" result, especially for exploratory analysis or as preprocessing for other algorithms.

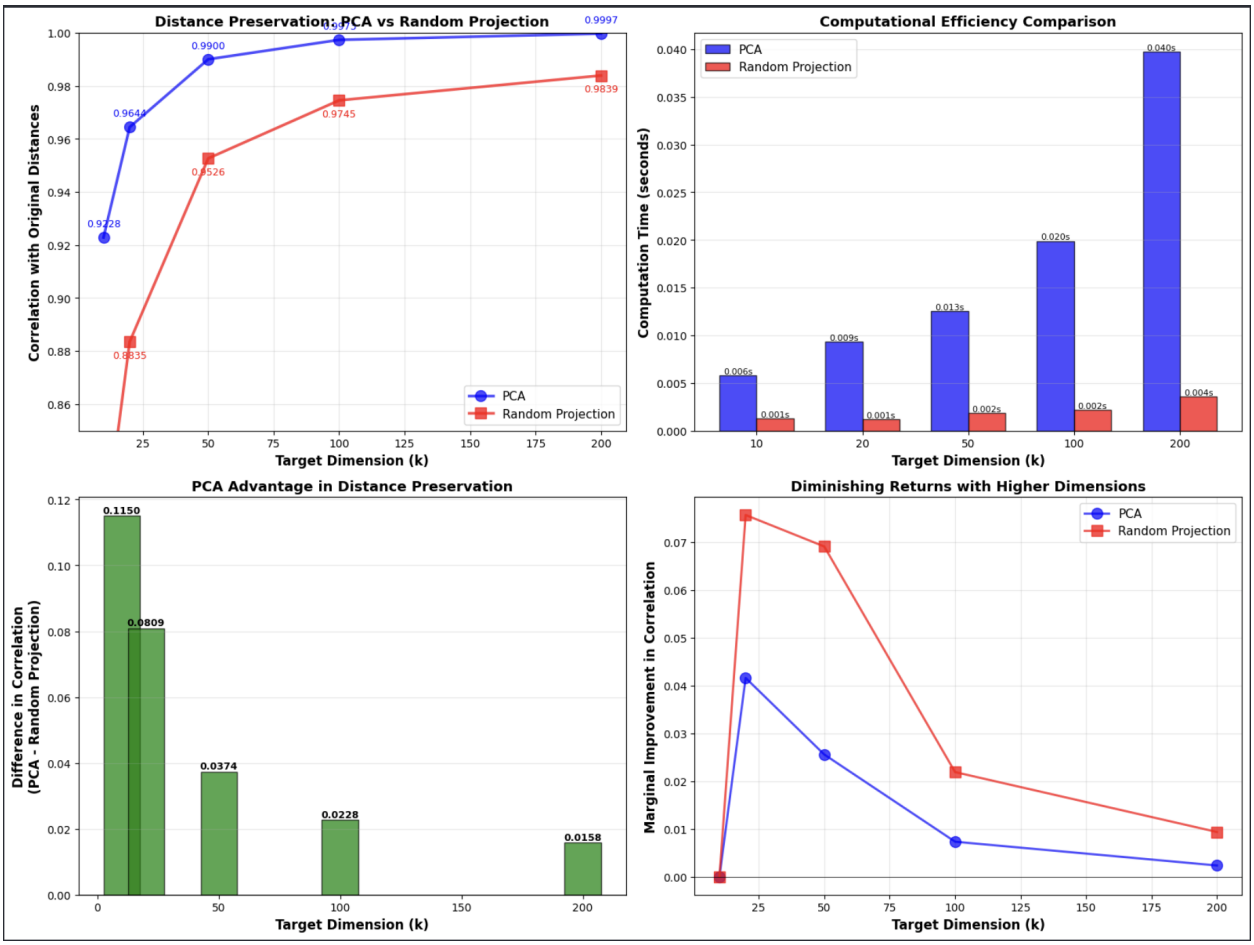

FIGURE 8. Main correlation vs. k plot (PCA and Random Projection curves)

4. CLUSTERING FASHION-MNIST USING SPECTRAL CLUSTERING

In this exercise, you will again perform a cluster analysis on the Fashion-MNIST dataset. Write code to use spectral clustering to cluster the data. Again, this is a larger dataset, so you may have to reduce the dimension/sample size as appropriate. In your report, explain your methodology and present your findings and figures. Compare your findings with Exercise 2.

For this final question, I implemented spectral clustering on Fashion-MNIST to compare it with the k-means results from Exercise 2, running both methods on a sampled 1000 image dataset under identical conditions for a fair comparison. Spectral clustering is fundamentally different from k-means in which instead of assuming spherical clusters, it builds a similarity matrix between all data points, treats it as a graph, and finds clusters by analyzing the graph's structure. This theoretically allows it to capture non-traditional, complex cluster shapes. I first reduced the data to 50 dimensions using PCA to make the affinity matrix computation feasible, then tested different gamma values for the Gaussian kernel on a 5,000 sample subset to find optimal parameters. Using the best gamma value, I ran spectral clustering on the 1000 point samples with k=10 clusters. The results

were much different then I thought they might be after doing k-means cluttering. The spectral clustering achieved a silhouette score of 0.1819 and ARI of 0.2375, while k-means on the same data got 0.1999 and 0.3434 respectively, meaning k-means outperformed spectral clustering across most metrics. After optimal label matching with the Hungarian algorithm, spectral clustering achieved 43.51% accuracy compared to k-means' 48.31%. Looking at the 2D PCA visualizations side by side, both methods produce similar looking cluster assignments, but k-means seems to capture the true class structure slightly better. The computation time is also notably different. The spectral clustering took much longer than k-means due to the affinity matrix operations, even with dimensionality reduction. The per class analysis reveals that both methods struggle with the classes that are visually similar such as shirts vs. t-shirts or coats vs. pullovers, but handle distinct items like trousers and bags well. Using the data to make some claims, it looks like spectral clustering's theoretical advantage of capturing non convex structures wasn't useful on this dataset. This is likely because Fashion-MNIST clusters aren't particularly non convex and the most significant problem is high inter class similarities. The practical conclusion is that while spectral clustering is powerful for data with complex manifold structures, for Fashion-MNIST, the simpler and faster k-means algorithm is the better choice for the Fashion-MNIST dataset. This demonstrates that in real world applications, more sophisticated algorithms aren't always better and knowing how your data is aligned is often more important.

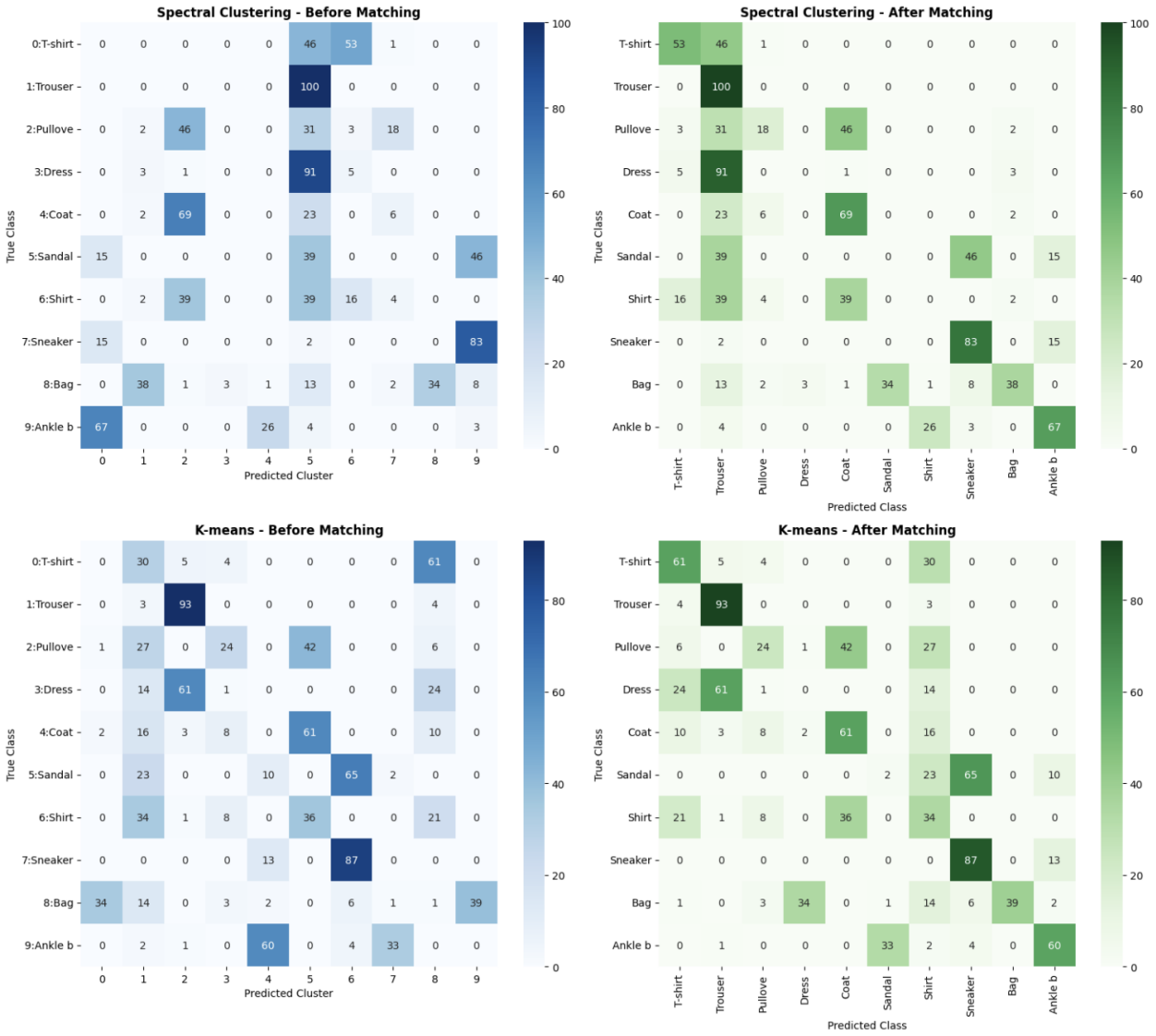

FIGURE 9. Four confusion matrices (2×2 grid: Spectral before/after matching, K-means before/after matching)

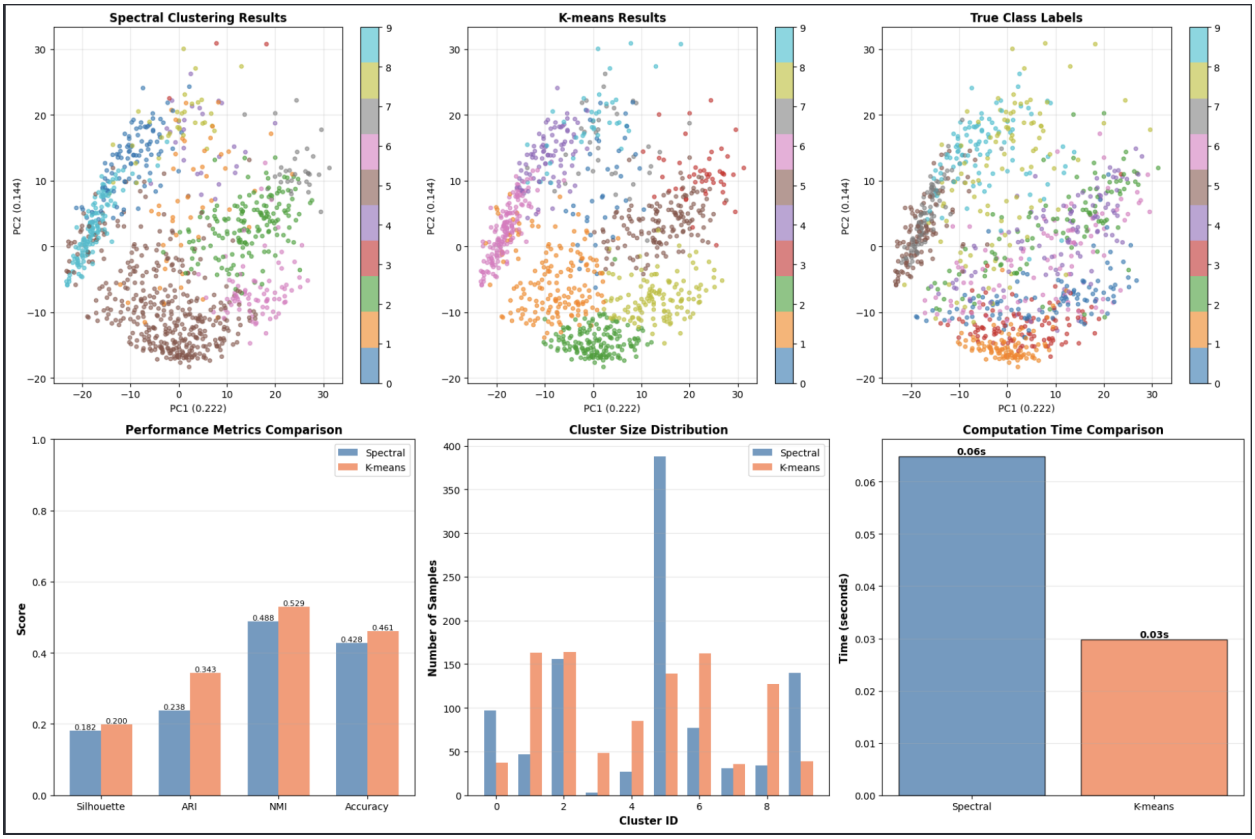

FIGURE 10. Lots of things but mainly cluster size, distribution comparison, and performance.