2024-01-26

Project Report

Fashion-MNIST Dataset Analysis

Marcin Knapczyk
WFIIS AGH

Table of Contents

The project's repository can be found at: https://github.com/Nautirius/FoDS-Final-Project

Dataset overview

Fashion-MNIST is a dataset of Zalando's product images. It consists of 60000 training and 10000 test examples split between 10 classes representing different articles of clothing. Each example is a 28x28 grayscale image.

In each row there are 785 columns, first of which contain the associated class label. The other 784 fields represent pixel lightness values. The pixel-value is an integer between 0 and 255, where 255 indicates a black pixel.

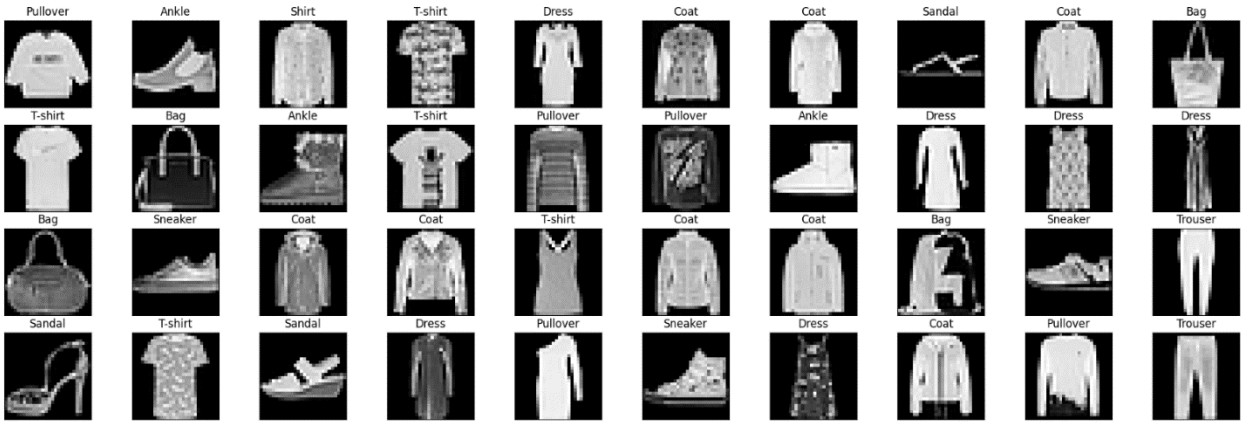

Figure 1: Visual interpretation of the dataset

Each example is assigned to one of the following class labels: 0 T-shirt, 1 Trouser, 2 Pullover, 3 Dress, 4 Coat, 5 Sandal, 6 Shirt, 7 Sneaker, 8 Bag, 9 Ankle boot.

Each class is equally represented in the dataset. What it means is that there is an equal chance to randomly select an item of each class, and there is no need to address any class imbalances.

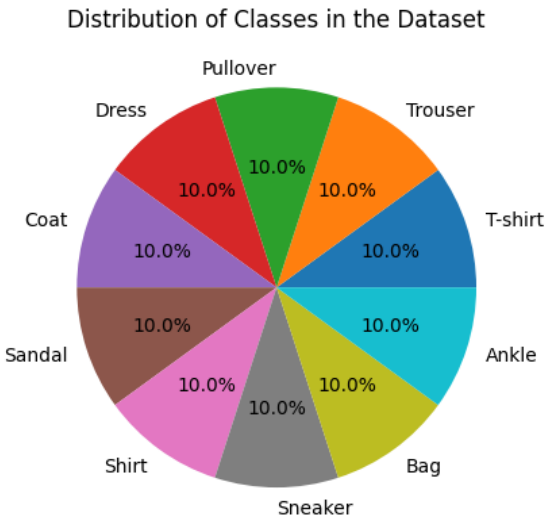

Figure 2: Pie chart illustrating class representation

Before further analysis I decided to preform data scaling and normalization, as it is the standard practice and should improve performance and give better results.

Dimensionality reduction

1. PCA (Principal Component Analysis)

When visualizing the outcome of PCA (Principal Component Analysis) in both two and three dimensions, one can note that elements from the same classes tend to form clusters, indicating a high degree of similarity within each class. However, within each class, there are noticeable outliers.

It is noticeable that many of the class clusters occupy the same space (as seen in instances like the Trouser and T-shirt classes in the 2D visualization). The differences between some of the classes may be greater in a higher-dimensional space. It may be worth to take a look at the Scree Plot for further insights.

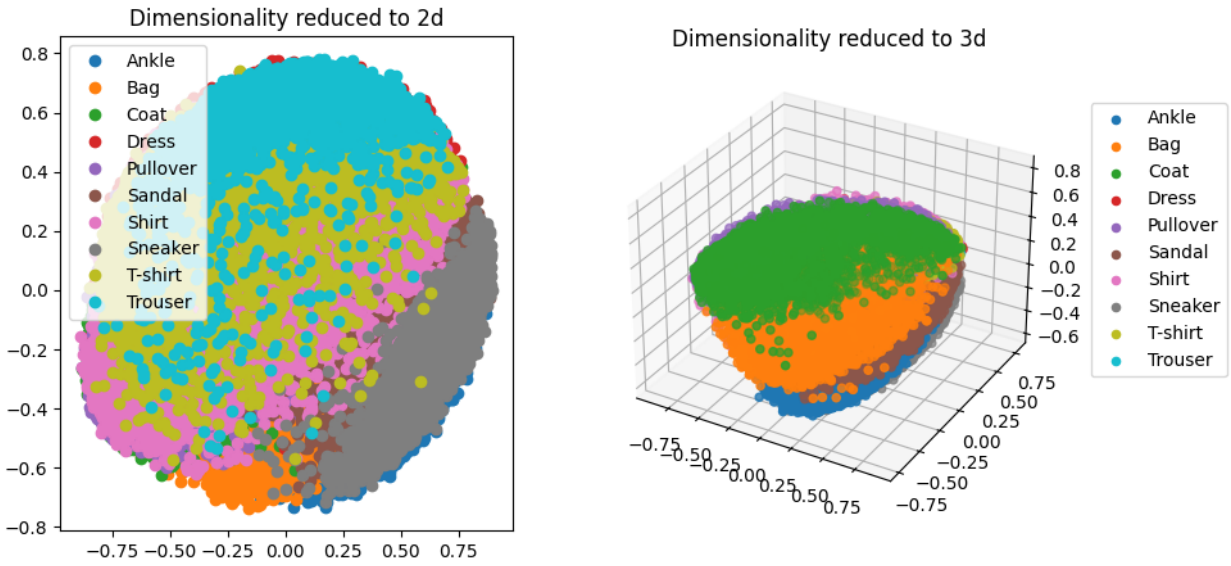

Figure 3: Visualization of PCA dimensionality reduction in two and three dimensions

Looking at the Scree Plot there is a sharp decline of the variance explained by the first few components. This is followed by a gradual flattening as the curve rapidly approaches zero on the Y axis, to later become almost horizontal. That long flat tail means that we can drop a lot of dimensions with little to none loss of variance.

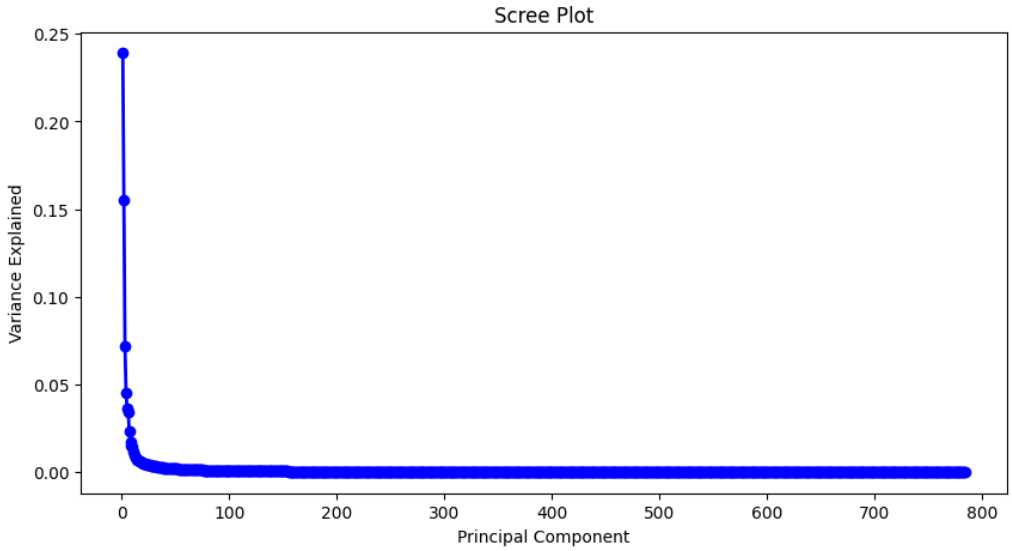

Figure 4: The Scree Plot of the PCA components

To determine the optimal number of dimensions to drop, it is helpful to make a cumulative variance plot. Examining the curve, we can once again observe that the number of explained variance quickly rises for the first few components. As the number of components increases past a couple of dozen, the curve flattens.

To decide where to make the cut I calculated the number of elements that collectively account for 95% of the variance. It ended up being 257 and this number is marked by the red vertical line.

I decided to use the first 257 PCA components for some of the more complex algorithms, as it will improve the performance drastically (number of data will drop by nearly 2/3) and should not worsen the results.

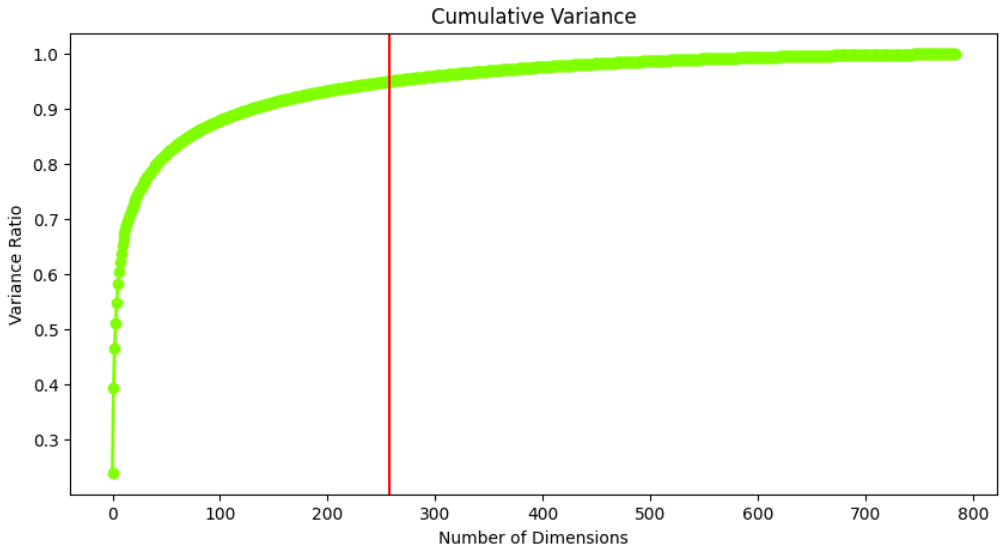

Figure 5: Cumulative variance. The red line indicates the number of components that cumulatively explain 95% of variance

2. t-SNE (t-Distributed Stochastic Neighbor)

Similarly to the previous method, when plotting the t-SNE (t-Distributed Stochastic Neighbor) embedding in both two and three dimensions, we can observe the formation of clusters representing components from the same classes.

This time however, the clusters appear to be more distinct. This distinction might be attributed to the fact that t-SNE aims at revealing clusters and non-linear relationships while PCA aims to capture the maximum variance, preserving global structure and distances between all data points.

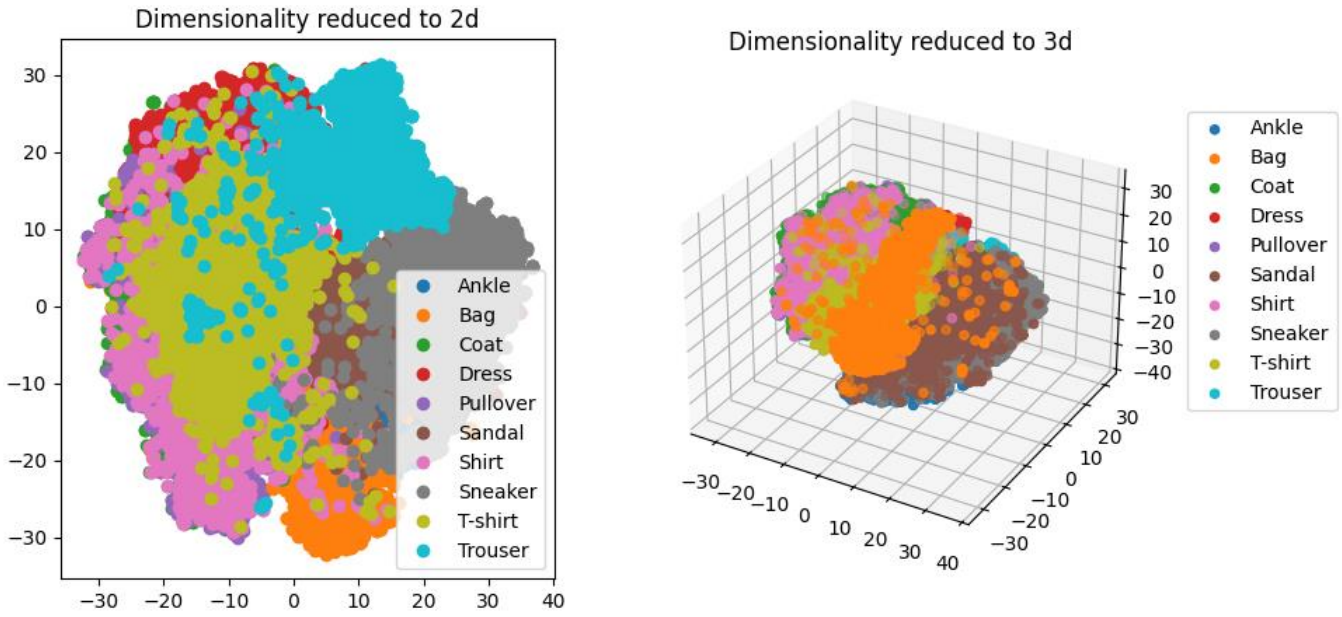

Figure 6: Visualization of t-SNE dimensionality reduction in two and three dimensions

Clustering

While attempting to cluster the entire dataset, I ran into performance issues (32 GB of RAM was not enough). To address this, I decided to down sample the dataset to 10000 examples, believing that this reduced number of data points would still be sufficiently large for the analysis while being more manageable.

1. Hierarchical clustering

Initially, I decided to create a dendrogram to take a look at the hierarchical structure of clusters formed during the process of hierarchical clustering.

The dendrogram reveals that data points are partitioned into two major clusters of approximately equal size. It seems that the dataset could also be split between six smaller clusters.

After that I performed agglomerative clustering and compared the outcomes of different metrics and linkages.

The best result was obtained with the combination of the Euclidean metric and ward linkage, resulting in a rand score of 0.43 (with 1.0 indicating a perfect match) that indicates rather poor accuracy.

The combination of cosine distance and average linkage performed even worse at 0.28, and the least favourable outcome was obtained with Manhattan distance and average linkage, resulting in a negative value that indicates a result worse than the expected score when assigning random labels.

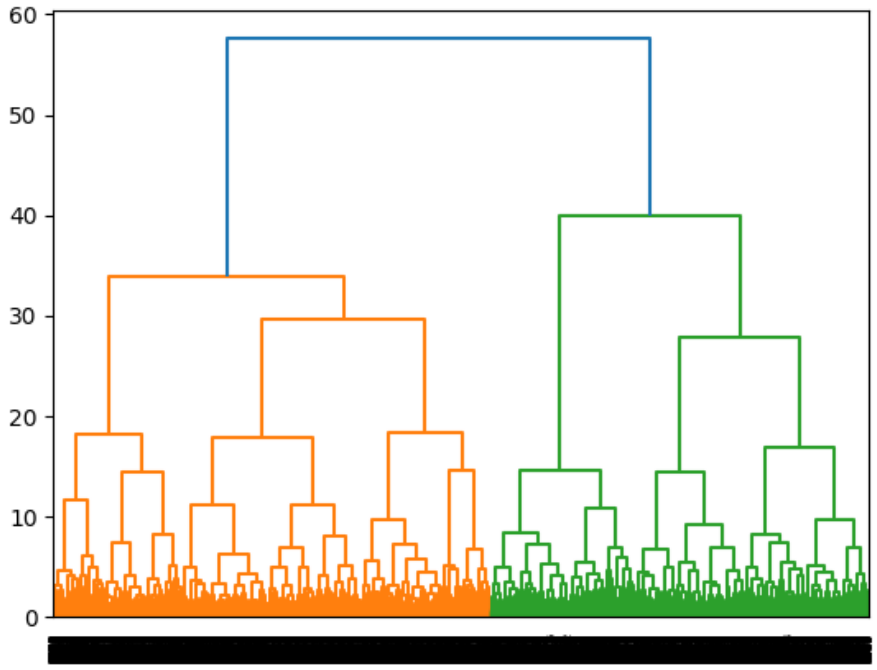

Figure 7: Dendrogram

2. Partition clustering

Conducting partition clustering using the K-Means algorithm resulted in a rand score of 0.36 for the original values and 0.38 for the 257 most important PCA components.

Subsequently, I decided to plot the clusters to see how the clusters are determined. The result differs significantly from the plot depicting the true classes.

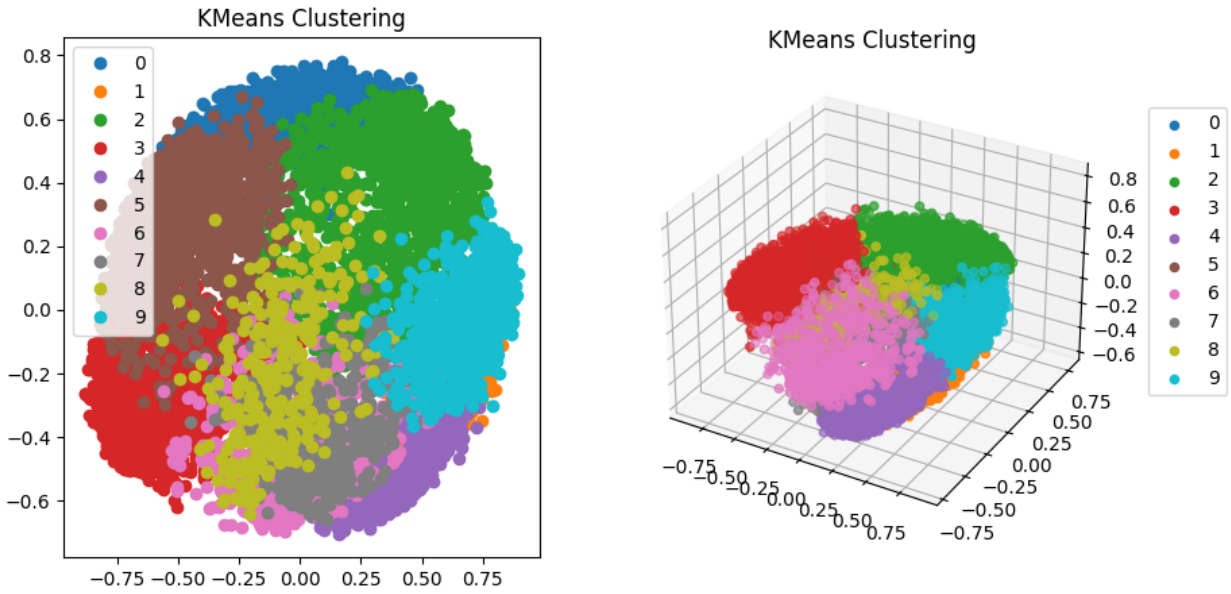

Figure 8: Visualization of K-Means clustering

Examining the distribution of classes within clusters we can observe some interesting dependencies. A distinct cluster is formed by nearly all Trousers paired with half of the Dresses. The algorithm has grouped Pullovers, Coats and Shirts together. Sandals and Sneakers each form their own distinct clusters. Additionally, Bags rarely end up being grouped with examples of other classes suggesting that articles of the Bag class are distant from other datapoints.

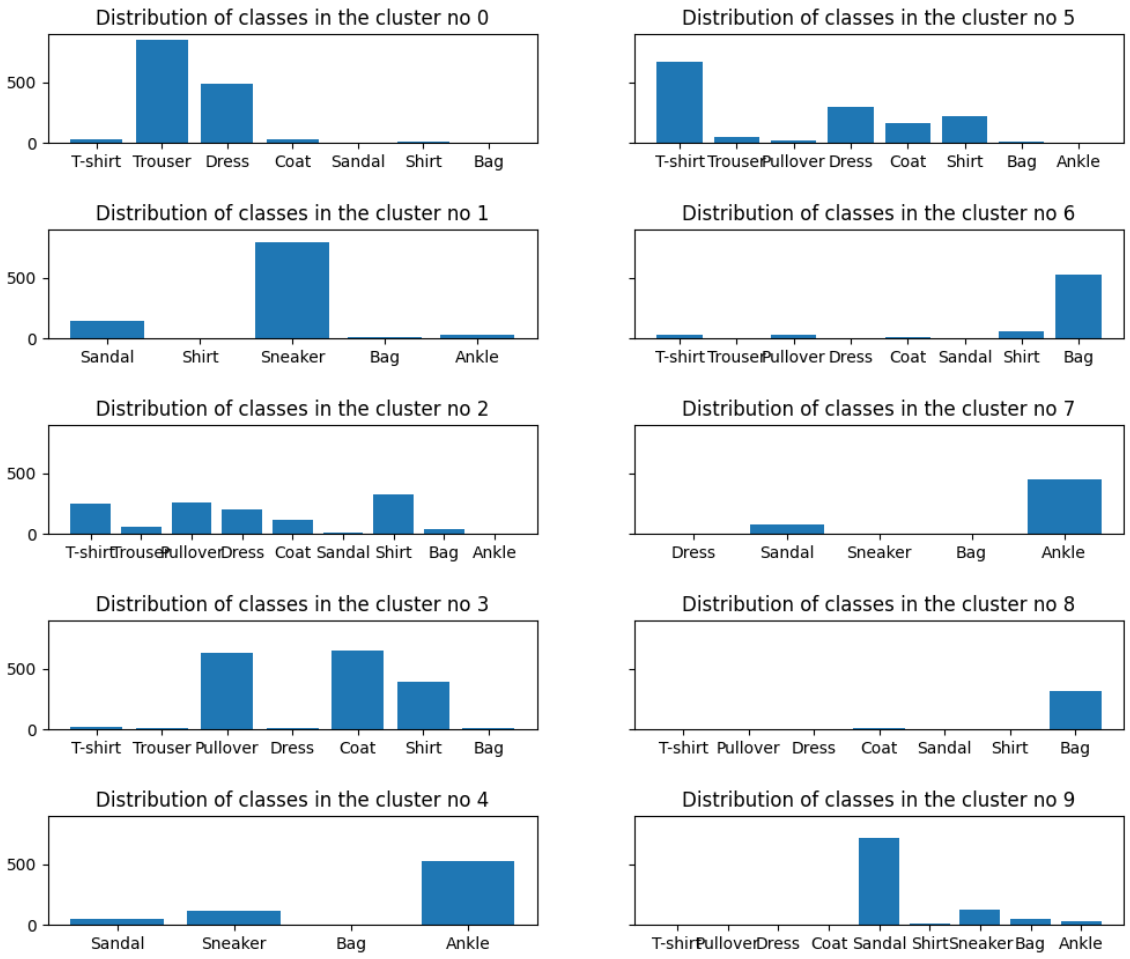

Figure 9: Distribution of classes in each cluster

Classification

1. k-Nearest Neighbors Classifier

When performing the classification task, I used the k-Fold method to split the dataset into training and testing subsets. Using 5-fold cross-validation we can observe how the result score changes with each iteration.

After testing several different parameters, I determined that the best mean rand index score is 0.86, indicating a relatively high accuracy, with 86% of cases being correctly classified.

2. Decision Tree

Classification using the Decision Tree method resulted in a decent score of 0.79. While this score is lower than that of the previous method, it is anticipated because the Decision Tree method prioritizes the visual representation of decision-making over accuracy. However, in this particular instance, the resulting tree is extensive, making manual examination difficult.

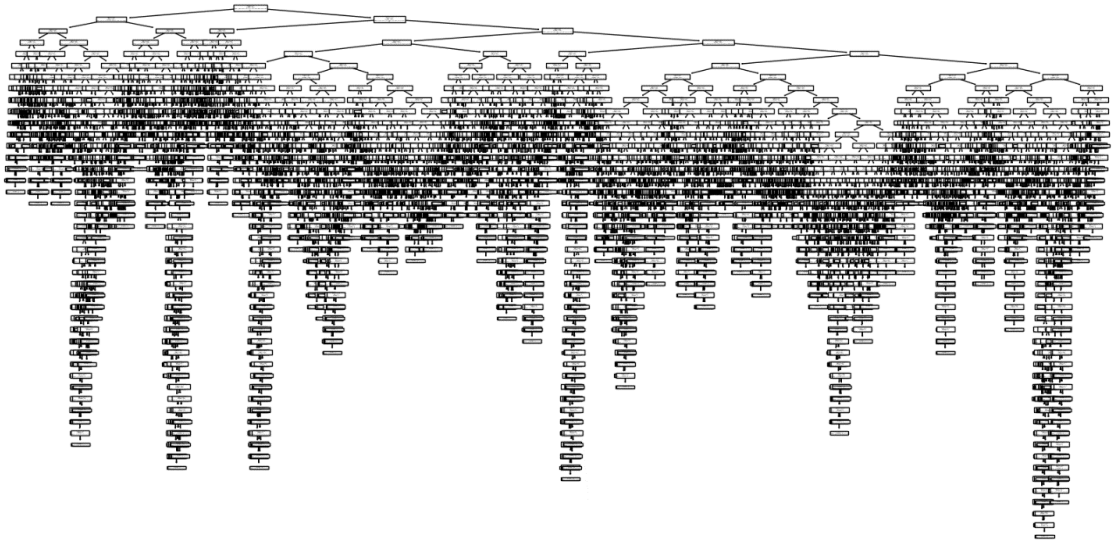

Figure 10: The decision tree

ChatGPT

1. Data summary

ChatGPT explained that the process of creating a data summary involves condensing and presenting key characteristics and insights about a dataset and that it typically includes descriptive statistics, visualizations, and other relevant information. It then lists the most important steps, selection of which is listed below:

- Familiarize yourself with the dataset's structure, variables, and overall content
- Handle Missing Values
- Descriptive Statistics

- Data Distribution
- Outliers Detection
- Visualizations
- Summary Insights

That does not differ from what we have discussed during the lectures.

2. Dimensionality reduction

When asked about dimensionality reduction, it listed the PCA and t-SNE algorithms. We have learned about these approaches during the labs and I have already included them in the report.

It also mentioned the LDA algorithm, which is a supervised technique that seeks to maximize the separation between classes while minimizing the variance within each class.

When plotting the resulting datapoints into 2D and 3D space we can observe that the clusters are more separated than in the PCA method.

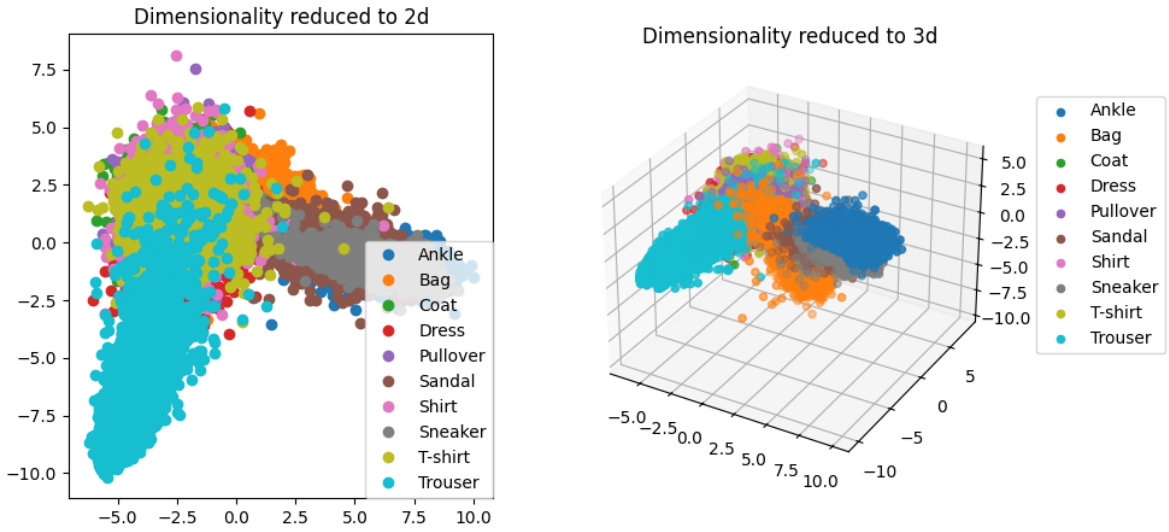

Figure 11: Visualization of LDA dimensionality reduction in two and three dimensions

3. Clustering

Asked about clustering, ChatGPT answered with the following list of methods:

- K-Means: Partitioning method that assigns data points to K clusters.
- Hierarchical Clustering: Builds a tree-like hierarchy of clusters.
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Identifies clusters based on dense regions of points.
- Agglomerative Clustering: Bottom-up approach, merging the nearest clusters iteratively.
- Gaussian Mixture Models (GMM): Assumes that the data is generated from a mixture of Gaussian distributions.

These approaches were discussed during the lectures. I decided to try implementing the DBSCAN method, but it does not seem to work well with this dataset, as it marks half of the datapoints as noise. Methods more suitable for the Fashion-MNIST dataset are already included in the report.

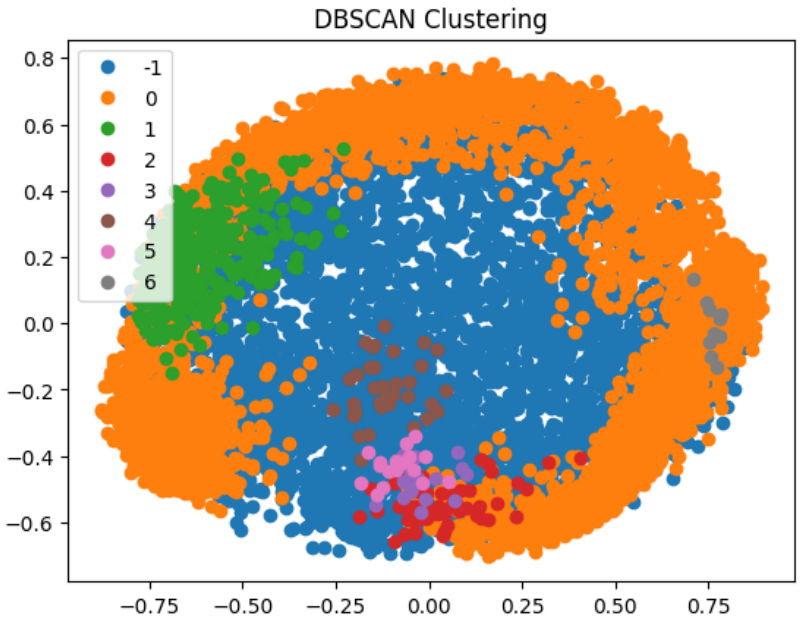

Figure 12: Visualization of DBSCAN clustering

4. Classification

ChatGPT presents a list of some of the most popular methods:

- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- k-Nearest Neighbors (k-NN)
- Linear Discriminant Analysis (LDA)
- Neural Networks

The Decision Tree and k-NN have been already implemented and discussed.

The Random Forest is a learning method that builds a collection of decision trees and merges their predictions to improve overall accuracy.

The accuracy improved compared to a single Decision Tree and reached a weighted average of 0.84.

Linear Discriminant Analysis (LDA) is a dimensionality reduction and classification technique. It seeks to transform the data into a lower-dimensional space while preserving class separability.

Performing the classification task using LDA resulted in an average score of 0.83, which is similar to accuracy of the k-NN approach.

5. My experience working with ChatGPT

This is my first experience with ChatGPT and it proved to be surprisingly helpful. It helped with tweaking parameters of algorithms by explaining how they work. Furthermore, it helped with writing boilerplate code (like plotting the data and generating repetitive steps in implementation of various algorithms). However, I had to take each of the responses with a grain of salt, knowing that the Chat may be hallucinating.