Data Preprocessing and Clustering with DBSCAN

Applied Machine Learning in Engineering - Exercise 04

TU Berlin, Summer Term 2025

Prof. Dr.-Ing. Merten Stender – merten.stender@tu-berlin.de

Learning Objectives

By the end of this assignment, you will be able to:

- Implement a reusable class for z-scoring and its inverse transformation.
- Apply data normalization to multi-dimensional datasets.
- Use DBSCAN from `scikit-learn` to identify clusters in real-world datasets.
- Evaluate clustering results using silhouette scores.
- Interpret and visualize clustering results.

Task 1: Z-Scoring Implementation (30 minutes, at home)

Implement an object-oriented approach for z-scoring a dataset, including its inverse transformation.

$$\tilde{\mathbf{x}} = \underbrace{\frac{1}{\sigma\left(\mathbf{x}\right)}}_{\text{unit standard deviation}} \cdot \underbrace{\left(\mathbf{x} - \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\right)}_{\text{zero mean}}, \quad \mathbf{x} \in \mathbb{R}^{N,n} \tag{1}$$

Use the dataset `data_clustering.csv` and load it as follows:
`data = np.loadtxt('data_clustering.csv', delimiter=',')`

Implement the following methods in a class called `Zscorer`:

(a) `Zscorer.fit(x: np.ndarray)` : Estimate mean and standard deviation for the dataset `x` and set class attributes.

(b) `Zscorer.transform(x: np.ndarray)` : Apply z-scoring to dataset `x` . Return z-scored data set to user.

(c) `Zscorer.inverse_transform(x)` : Revert z-scored data to the original scale and return to user.

(d) Ensure that your implementation handles multi-dimensional input data (`X.shape = [N, n]`). Use the `axis` argument in NumPy functions such as `np.mean(..., axis=)` and `np.std(..., axis=)` to compute statistics along the correct dimensions.

(e) Validate your implementation by comparing it with `StandardScaler` from `scikit-learn`.

Task 2: Clustering with DBSCAN (60 minutes)

This task involves analyzing real-world data using the DBSCAN clustering algorithm. The data is provided in the file `secondary_hand_car_sales.npy` and can be loaded using

```
np.load('secondary_hand_car_sales.npy', allow_pickle=True)
```

The dataset includes the following features:

1. Manufacturer
2. Model
3. Engine size
4. Fuel type
5. Year of manufacture
6. Mileage (in miles)
7. Price (in pounds)

Instructions:

(a) Load the dataset and extract only the columns for: Year of manufacture, Mileage, Price

(b) Apply z-scoring to the selected subset.

(c) Run DBSCAN from Scikit-learn using default parameters to cluster the data.

(d) Determine the number of clusters found. DBSCAN assigns the label `-1` to outliers. To count the number of non-outlier clusters:

```
num_clusters = len(set(cluster_labels)) - (1 if -1 in cluster_labels else 0
```

(e) Compute the silhouette score to evaluate the quality of the clustering:

```
from sklearn.metrics import silhouette_score
```

```
score = silhouette_score(x, cluster_labels)
```

(f) Perform a sensitivity analysis of the DBSCAN hyperparameter ϵ:

- Use a logarithmic grid from 0.001 to 1.0:
  `eps_values = np.logspace(-3, 0, num=40)`
- For each value of ϵ, run DBSCAN and track:
  - Number of clusters
  - Silhouette coefficient

(g) Identify the ϵ value that gives the highest silhouette score.

(h) Visualize the clustering results using your preferred plotting tool (e.g., scatter plots with cluster coloring).

(i) Analyze the clusters by mapping them back to the original dataset:

- Which manufacturers and models appear in the same cluster?
- Are the groupings reasonable based on your intuition?