### Assignment 2 Walkthrough

#### 1. Filter Outliers

**Overview:** This method identifies and removes outliers from a dataset. An outlier is defined as any data point that lies more than two standard deviations from the dataset’s mean. This process helps in refining the dataset for further analysis or model training.

**Detailed Pseudo Code:**

FUNCTION filter\_outliers(data)  
 CALCULATE 'mean' of 'data'  
 CALCULATE 'standard deviation' of 'data'  
 FOR each 'value' in 'data'  
 IF 'value' is less than 'mean' - 2 \* 'standard deviation' OR greater than 'mean' + 2 \* 'standard deviation'  
 MARK 'value' as outlier  
 REMOVE all marked outliers from 'data'  
 RETURN 'data' without outliers  
END FUNCTION

**Implementation Guide:** - Use numpy functions to calculate the mean and standard deviation of the dataset. - Iterate over each data point in the dataset and compare it against the defined outlier criteria. - Remove the data points identified as outliers. - Return the filtered dataset.

#### 2. Normalize Features

**Overview:** This function normalizes the features in a dataset to ensure they have zero mean and unit variance. This standardization of the dataset is crucial for many machine learning algorithms to perform optimally.

**Detailed Pseudo Code:**

FUNCTION normalize\_features(X)  
 FOR each 'feature' in 'X'  
 CALCULATE mean of 'feature'  
 CALCULATE standard deviation of 'feature'  
 FOR each 'value' in 'feature'  
 NORMALIZE 'value' to ('value' - mean) / standard deviation  
 RETURN normalized 'X'  
END FUNCTION

**Implementation Guide:** - Iterate over each feature (column) in the dataset. - For each feature, calculate its mean and standard deviation. - Normalize each value by subtracting the mean and dividing by the standard deviation. - Return the dataset with normalized features.

#### 3. Implement k-Nearest Neighbors (kNN)

**Overview:** Implement the k-Nearest Neighbors algorithm from scratch. This method calculates the Euclidean distance between each test sample and all training samples to find the k nearest neighbors. It then predicts the label of the test samples based on the majority vote of these neighbors.

**Detailed Pseudo Code:**

FUNCTION implement\_knn(X\_train, y\_train, X\_test, k)  
 INITIALIZE 'predictions' as an empty list  
 FOR each 'test\_sample' in 'X\_test'  
 CALCULATE Euclidean distance between 'test\_sample' and all samples in 'X\_train'  
 IDENTIFY the 'k' smallest distances and their corresponding labels from 'y\_train'  
 DETERMINE the most common label among the identified labels  
 APPEND the most common label to 'predictions'  
 RETURN 'predictions'  
END FUNCTION

**Implementation Guide:** - For each sample in the test dataset, calculate the Euclidean distance to each sample in the training dataset. - Identify the k closest training samples and take note of their labels. - Predict the label for the test sample based on the most frequent label among the k closest training samples. - Return the predictions for the entire test dataset.