Summary Part

1. **In your own words, explain how KNN works**

   K-Nearest Neighbors (KNN) is a simple and intuitive machine learning algorithm used for both classification and regression tasks. The basic idea is to find the "K" data points in the training set that are closest to a new, unseen data point (based on some distance metric like Euclidean distance). Once the nearest neighbors are identified, the algorithm makes a prediction based on the majority class (for classification) or the average value (for regression) of these neighbors.

2. **What is the difference between KNN for regression and KNN for classification?**

   KNN for Classification: The algorithm assigns the new data point to the class that is most common among the K nearest neighbors. Essentially, it's a voting system where each of the K neighbors casts a vote for its class, and the class with the most votes is chosen as the prediction.
   KNN for Regression: Instead of voting, the algorithm calculates the average value of the K nearest neighbors and assigns this average as the predicted value for the new data point. Here, KNN is used to predict a continuous output (e.g., temperature, price) rather than a categorical one.

3. **What is "feature extraction"?**

   Feature extraction is the process of transforming raw data into a set of features (or variables) that can be effectively used in a machine learning model. These features are intended to capture the most relevant information from the data while reducing its dimensionality. For example, in image processing, feature extraction might involve identifying edges, textures, or shapes that are important for recognizing objects in an image. The goal is to simplify the data without losing critical information, making it easier for the model to learn and make accurate predictions.

4. **Could you improve the code for the KNN function in Exercise 1 by using the DRY principle?**

   Refactored Code:

```
distances = []
for feature in features:
    distance = round(euclidean_distance(feature, chosen_feature), 2)
    weight = feature[0]
    distances.append((distance, weight))
distances.sort()
k_nearest_labels = distances[:k]
```

to

The Function Code:

```
def get_k_nearest_neighbors(features, chosen_feature, k, value_index):
 distances = []
 for feature in features:
     distance = round(euclidean_distance(feature, chosen_feature), 2)
     value = feature[value_index]
     distances.append((distance, value))
 distances.sort()
 return distances[:k]
```

Self-reflection part

What did you learn?

This week, I learned about the K-Nearest Neighbors (KNN) algorithm, including how it works for both classification and regression tasks.

What went smoothly?

Understanding the core concepts of KNN. The practical coding exercises helped reinforce these concepts, making them easier to grasp and apply in real-world scenarios.

What was difficult about the content this week?

The most challenging part was initially identifying the redundant code in the KNN functions and figuring out the best way to refactor it without losing clarity.

How will you approach things differently next time?

I will try to identify potential areas for code refactoring earlier in the process.

Do you have any feedback about the content for this week?

Good!