Basic anomaly detection from scratch

Objective:

The primary goal of this project is to develop an anomaly detection system to identify patients with unusual characteristics within breast cancer diagnosis data. The dataset, patients.csv, contains records of 310 individuals, each described by d=30 different features related to breast cancer diagnosis. While most of these individuals are healthy, a few exhibit significant differences. The main objective is to build a tool capable of identifying these unique cases—individuals who display distinct characteristics that might indicate breast cancer-related health issues. It's important to note that this project emphasizes anomaly detection, not patient classification, as there are no clear labels or categories provided.

Getting Started:

This project description offers suggested steps and guidance to assist you in completing the project. However, these steps are flexible, and you're encouraged to adapt and structure the project in your own way, using these suggestions as a helpful framework. Before you begin, take a moment to read the entire project description to gain a comprehensive understanding.

Note: Your code/solution should be provided in the form of a Jupyter notebook.

Project Description:

(A) Data Exploration

Begin by loading the provided patients.csv dataset into a Pandas DataFrame. Take this opportunity to explore the data freely. Here are some suggested tasks to help you become familiar with the dataset:

- Examine the names of the 30 features used to characterize the patients.

- Display the initial rows of the dataset to understand its structure.

- Calculate and print basic statistics, such as the mean and standard deviation, for each of the 30 features to identify variations in scales or ranges.

- Select a feature and investigate the distribution of its values by plotting a histogram of these values using `plt.hist(the_values_of_feature_i)`. For guidance on plotting histograms and their meaning, you can refer to: https://www.w3schools.com/python/matplotlib_histograms.asp

(B) Data Preprocessing and Visualization

Normalization: If the features have different scales, use the `StandardScaler` class from scikit-learn to normalize their values.

Visualization: Visualize the dataset using a scatter plot and try to see (approximately) how many anomalous data-points there might be. Keep in mind that the dataset is high-dimensional (30 features), so consider transforming it into low-dimensional data, such as 2

features, using Principal Component Analysis (PCA); then, create the scatter plot based on this transformed data.

(C) Calculating Euclidean Distance

Define a Python function, `euclidean_distance(point1, point2)`, that calculates the Euclidean distance between two data-points, each represented as an array of 30 values. You can use NumPy arrays for efficient vector operations.

Reminder: The Euclidean distance measures how far apart two points are in d-dimensional space. Smaller distances mean the points are close, and larger distances mean they are far apart. For example, the distance between two points `p and q` (corresponding to two patients) is:

$$\sqrt{\sum_{i=0}^{d-1}(p_i - q_i)^2}$$

(D) Anomaly Score Calculation

The objective here is to compute an anomaly score for a given data-point, indicating its abnormality. The anomaly score associated with a data-point can be defined as the average distance from that point to its *k* nearest neighbors.

Implement a function `calculate_anomaly_score(data, query_point, k)` that calculates an anomaly score for a given query point within the dataset. This function should:

- Calculate the Euclidean distances between the query point and all other data points in the dataset (using the Euclidean distance function you implemented earlier).
- Select the k smallest distances as potential neighbors.
- Compute a single anomaly score for the `query_point` based on these *k* distances. The score can be calculated as the average of these distances.

(E) Identifying Anomalies

Now, we aim to identify patients with potential health issues related to breast cancer. Anomalous data points in the dataset may correspond to patients who exhibit unusual characteristics warranting further examination. Implement a loop to calculate an anomaly score for each of the 310 patients in the dataset using the `calculate_anomaly_score` function.

Your earlier data exploration and visualization should have given you a rough idea about the number (*n*) of anomalies to expect. Find the top *n* anomalous patients based on their anomaly scores.

(F) Visualization of Anomalies

Create visualizations to highlight the anomalies in the dataset. For example, you can:

- Create a histogram of anomaly scores to show their distribution.
- Use a scatter plot to mark the anomalous data-points in a distinct color or shape (you can plot them on top of the original data to make them visually stand out).
- Compare the distribution of feature values among normal vs abnormal patients.

- etc.

(G) Additional Recommendations

- Object-Oriented Programming: You can optionally consider organizing the entire project into a class with relevant methods and attributes for better organization and reusability.
- Optimization: Explore ways to optimize your code for efficiency, especially when dealing with larger datasets. Reduce unnecessary computations where possible.
- Experiment with different values of *k* and see their impact on the number of identified anomalies.