Home Assignment – 1

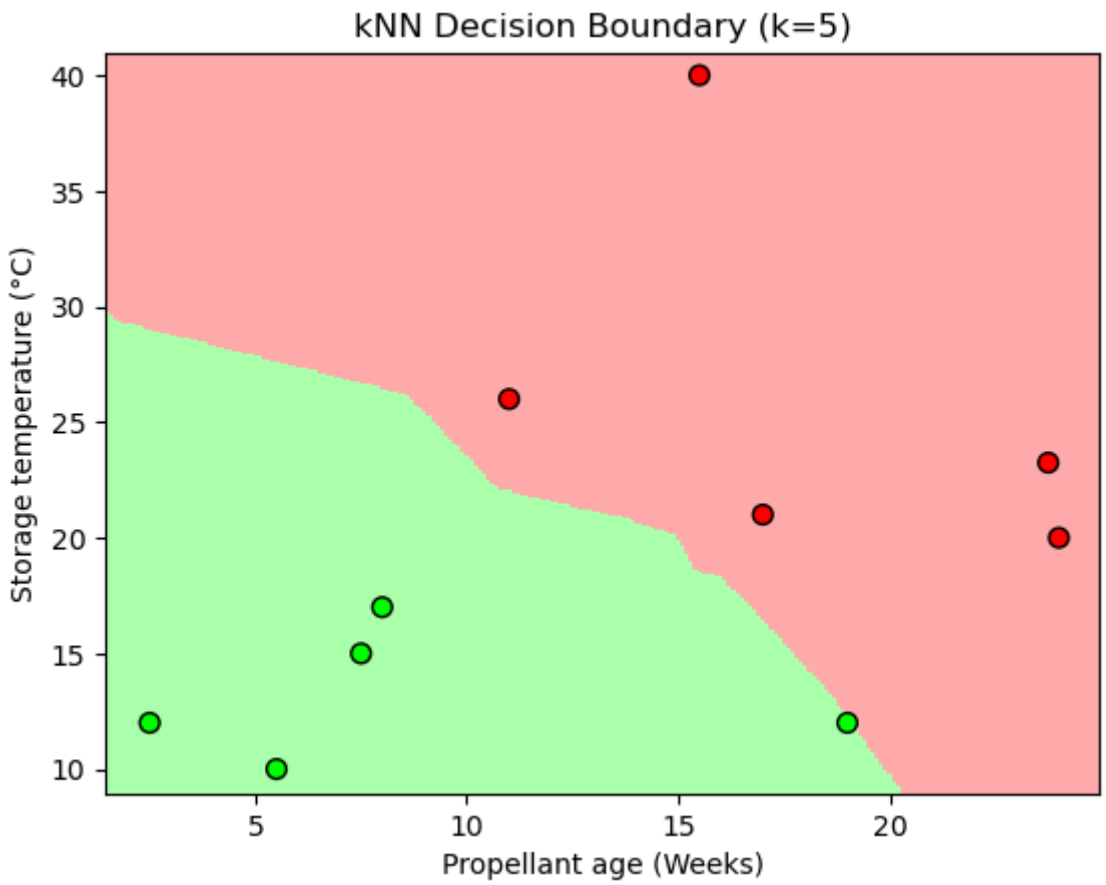
Aarav Aryaman, 220012

Problem:

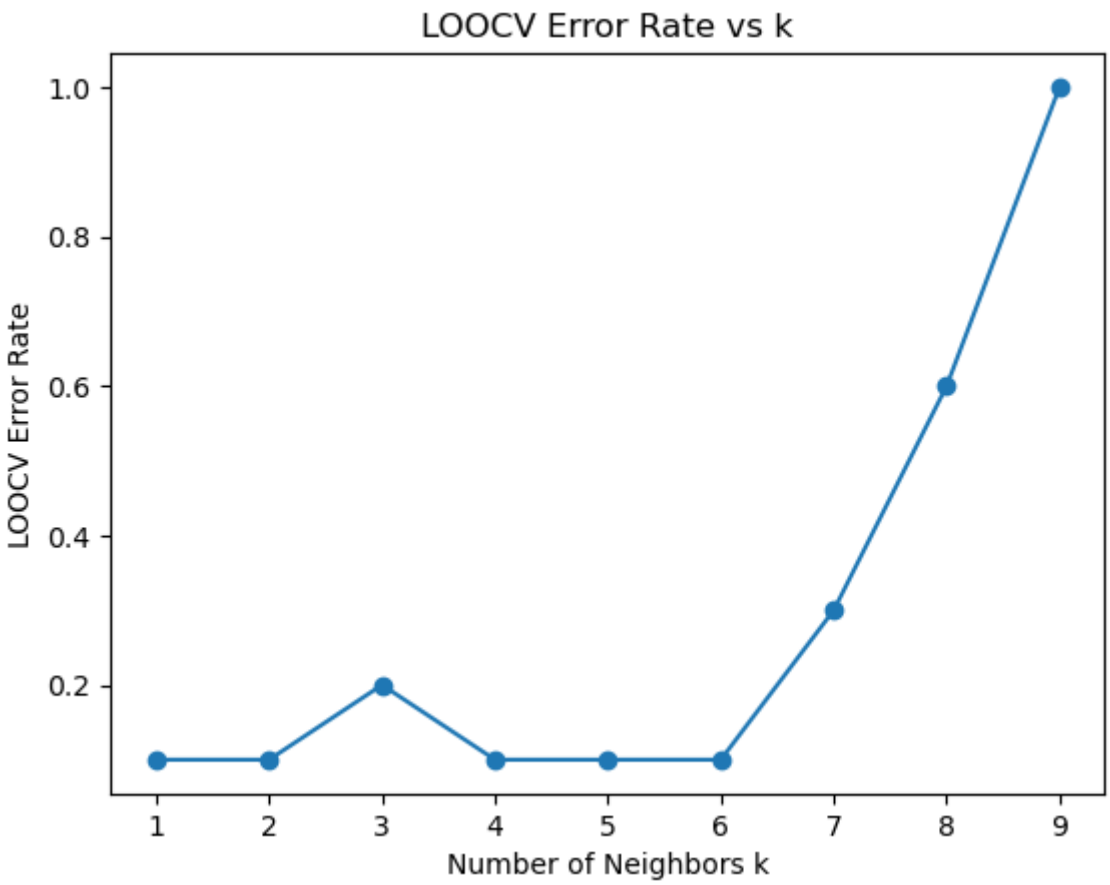
Build a model using k-Nearest Neighbours (kNN) to classify samples as Pass/Fail applications based on the Propellant Age and Storage Temperature. Compute and plot a Pass/Fail Decision Boundary using kNN = 5. Discuss possible validation procedures. Conduct a Leave One Out Cross-Validation (LOOCV) and plot the result. Find optimum value of k.

Solution Algorithm:

1. **Data Preparation**:
    - Collect the propellant test data with features (propellant age and storage temperature) and labels (pass/fail).
    - X contains the features (propellant age and storage temperature) and y contains the labels (0 for fail, 1 for pass).
2. **kNN Implementation**:
    - Use the k-nearest neighbours (kNN) algorithm (KNeighborsClassifier from sklearn) to classify each point. The decision boundary will be determined based on the majority vote of the 5 nearest neighbours.
    - Store the training data. There is no explicit model training as in other algorithms.
    - For a new data point, calculate its distance to all training points (using metrics like Euclidean distance).
    - Identify the k nearest neighbours and assign the class that is most common among these neighbours to the new data point (pass/fail).
    - Parameter k, the number of neighbours to consider, is a key parameter that influences the classifier's performance.
3. **Plotting Decision Boundary**:
    - Generate a mesh grid over the feature space (age vs. temperature).
    - For each point in the grid, classify using kNN.
    - Plot the decision boundary by differentiating between the pass and fail regions.



4. **Validation Procedure (LOOCV)**:
    - For each data point, leave it out from the dataset and train the kNN classifier on the remaining data.
    - Test the classifier on the left-out data point.
    - Repeat the process for each data point and calculate the error.
    - The error rate is calculated and averaged to assess model performance.
    - **The LOOCV error rate calculated for k = 5 is 0.10**.
5. **Optimal k Selection**:
    - Iterate over different values of k, perform LOOCV for each k, and record the error rate.
    - Select the value of k that provides the lowest error rate.
    - **From the plot, optimal value of k is k = 1**. Other values of k, k = 2, 4, 5, 6 also have the same error as k = 1, i.e, 0.10. However, the least value of k is more efficient at validation than the higher values. Hence, optimal value is k = 1.



Possible Validation Procedures:

- **Leave-One-Out Cross-Validation (LOOCV)** is a special case of K-Fold Cross-Validation with K=1/n, for n training data. This method is a good technique for small datasets because it maximizes the use of the available data. However, it can be computationally expensive for large datasets since it involves training the model as many times as there are data points.
- **K-Fold Cross-Validation** splits the dataset into k equal parts (folds). The model is trained on k−1 folds and tested on the remaining fold, repeating this process k times. It provides a more stable estimate of model performance compared to a single train-test split. However, it can be computationally expensive, especially when k is large.
- **Train-Test Split (Holdout Validation)** divides the dataset into two parts: one for training and one for testing. It is simple and fast, making it suitable for large datasets. However, the model's performance can vary depending on how the data is split, which may lead to biased estimates, especially in small datasets.
- **Stratified K-Fold Cross-Validation** is similar to K-Fold Cross-Validation but ensures that each fold has a similar distribution of class labels. This method is particularly useful for imbalanced datasets, providing more reliable performance estimates. It adds complexity compared to standard K-Fold but helps maintain the class balance in each fold.

- **Repeated K-Fold Cross-Validation** involves performing K-Fold Cross-Validation multiple times with different random splits. It helps to reduce variance in the performance estimate by averaging results across multiple runs. However, it is more computationally intensive than a single K-Fold Cross-Validation.
- **Bootstrapping** generates multiple new datasets by sampling with replacement from the original dataset. The model is trained on these bootstrap samples and tested on the out-of-bag samples (those not included in the training set). Bootstrapping provides robust estimates of model performance and variance, but it can be computationally expensive and may include duplicate samples in training sets.