GIFT UNIVERSITY
Converting Knowledge into Practical Experience

ASSIGNEMENT-REPORT

GROUP MEMBERS:

M U H A M M A D F A R A Z – 221 9 8 0 0 0 7

H A S S A N R A S H E E D – 22190038

I J A Z U L H A S S A N - 221 980030

H A S E E B B U T T – 221900 04

COURSE

M A C H I N E L E A R N I N G

INSTRUCTOR

M U H A M M A D U S M A N

Objective:

This report outlines the implementation of k-Nearest Neighbors (k-NN) and Principal Component Analysis (PCA) for diagnosing motor faults based on Current-A data. The goal is to classify motor conditions into 14 categories, including healthy and various faulty states, to demonstrate the effectiveness of machine learning techniques in predictive maintenance.

Dataset Description:

The dataset consists of instantaneous Current-A values from three-phase current data of induction motors under varying conditions:

- Fault Types:
  - Inner and outer race bearing faults with severities from 0.7mm to 1.7mm.
  - Broken rotor bar faults.
  - Healthy motor conditions.
- Load Conditions: 100W, 200W, 300W.
- Sampling Rate: 10 kHz, with 1000 samples per block.
- Classes: 14 distinct motor health conditions.

The dataset includes over 100,000 samples per file, and 39 files corresponding to different motor conditions and loads.

Question 1: Implementation of k-NN Algorithm

Step 1: Data Preprocessing:

1. **Data Loading:**
   - Data was loaded from CSV files representing different motor conditions (e.g., 0.7_inner_bearing_unhealthy.csv).

- Each file contains over 100,000 samples of three-phase current data.

2. **Data Cleaning:**
   - Retained only the Current-A column, removing Time Stamp, Current-B, and Current-C.
   - Extracted the first 100,000 rows from each file for consistency.
3. **Data Reshaping:**
   - Divided the Current-A data into blocks of 1000 samples. o Assigned a label to each block corresponding to the motor condition (e.g., healthy_motor, 0.7_inner_bearing_unhealthy).
4. **Data Merging:**
   - Combined processed blocks from all conditions into a single DataFrame.
   - Saved the merged dataset as motor dataset.csv for subsequent analysis.

Step 2: Calculating Euclidean Distance:

- A custom function was implemented to calculate the Euclidean distance between test and training samples:

Step 3: k-NN Model Implementation:

- The k-NN algorithm was implemented without using built-in libraries. It predicts the class of a test sample by identifying the most common label among its k nearest neighbors.
- Applied k-NN with k=2 to the test data and generated predictions.

Step 4: Hold-Out and Cross-Validation:

1. **Hold-Out Validation:**
   - Split the dataset into training (80%) and testing (20%) subsets.
   - Converted DataFrames to NumPy arrays for compatibility with the k-NN function.
2. **10-Fold Cross-Validation:**
   - Used 10-fold cross-validation to assess model performance across different splits of training and validation data.

Step 5: Classification Metrics:

- Calculated the following performance measures:
  - **Accuracy:** Proportion of correct predictions.

- **Precision, Recall, F1-Score:** Evaluated for each class and averaged (micro, macro, weighted).
- **Confusion Matrix:** Visualized the performance of the classifier across all 14 classes.

Step 6: Optimizing k Value:

- Evaluated k values ranging from 1 to 14.
- Plotted test accuracy for each k value to identify the optimal setting.

Question 2: Applying PCA

Step 1: Data Preprocessing with PCA:

1. **Normalization:**
   - Applied Min-Max scaling to normalize feature values.
2. **Dimensionality Reduction:** ○ Used PCA to reduce data dimensions while

   retaining 95% of variance.

Step 2: k-NN Implementation with PCA

- Re-applied the k-NN algorithm to PCA-transformed data.
- Repeated steps from Question 1, including hold-out validation, cross-validation, and hyperparameter tuning.

Question 3: Model Comparison

Observations

1. **Without PCA:**
   - Accuracy: 27.14%.
   - Metrics:
     - Micro F1-Score: 27%.
     - Macro F1-Score: 29%.
     - Weighted F1-Score: 28%.
   - Confusion Matrix: High confusion among similar fault classes.
2. **With PCA:**
   - Accuracy: 68 %.

    - Slight improvement in accuracy due to reduced dimensionality.
    - PCA enhances separability by eliminating redundant features.

Comparison:

- The PCA-enhanced k-NN model performs slightly better due to improved feature representation.
- The choice of k impacts performance significantly, with the optimal k identified through evaluation.
- For further improvement, advanced algorithms like Random Forests or SVM can be explored.

Conclusion:

This assignment demonstrates the application of k-NN and PCA for diagnosing motor faults. The results highlight the importance of preprocessing, feature selection, and model tuning in machine learning workflows.