Assignment 2

Assignment Details:

- **Duration:** 1 week
- **Team:** same team members as the previous one

Tasks for Each Team:

1. Dataset Selection:

- Select the **same dataset** you worked with in Assignment 1.

---

2. Data Splitting:

- **Split the data into three parts:**
  - **Training set** (60%)
  - **Validation set** (20%)
  - **Test set** (20%)
- Use the **training set** for fitting models.
- Use the **validation set** for tuning hyperparameters (like the best **K** in KNN).
- Use the **test set** for final evaluation after model selection.

---

3. K-Nearest Neighbors (KNN) Algorithm:

- Implement KNN using a Python library (e.g., scikit-learn).
- Test the model performance by trying different values of **K**.
- Use the **validation set** to find the optimal **K** value.
- Record the final **accuracy** on the **test set**.

---

4. Cross-Validation:

- Apply **k-Fold Cross-Validation** (suggested: **5-fold or 10-fold**) on the training set.
- Report:
  - **Average accuracy** across folds.
- **Compare**:
  - Performance using cross-validation vs. validation set vs. test set results.
  - Discuss if cross-validation gives a more **robust performance estimate**.

---

5. Confusion Matrix:

- Once you select the final model, evaluate on the **test set** using:
  - **Confusion matrix**.
  - Calculate:
    - **Accuracy**
    - **Precision**
    - **Recall**
    - **F1-score**
- Analyze:
  - Any patterns observed in the confusion matrix.

---

6. Overfitting and Model Improvement:

- Discuss possible **overfitting** issues:
  - Does the model perform much better on training than validation/test sets?
- Techniques to **reduce overfitting** (choose one or more to try):
  - **Tune K value**: Higher K usually reduces overfitting.
  - **Feature Selection/Reduction**: Use fewer or more relevant features.
  - **Data Augmentation**: If applicable, create more data samples.
  - **Cross-validation-based model selection**: Choose model based on cross-validation scores, not training scores.

---

7. Visualization:

- Plot:
  - **Confusion matrix** as a heatmap.
- Optional:
  - Visualize the dataset (2D/3D plots) if meaningful to show class separation.

Deliverables:

A. Code Submission:

- Submit clean, modular, and well-commented Python Notebook.
- Cover:
  - Data Preprocessing
  - Train/Validation/Test Splitting
  - KNN Implementation
  - Cross-Validation
  - Confusion Matrix Computation
  - Visualizations

B. Final Report:

Include the following sections:

1. **Introduction**: Dataset description and problem.
2. **Data Preprocessing**: Steps taken (splitting, cleaning, etc.).
3. **KNN Model**: Model setup, training, tuning, and evaluation.
4. **Cross-Validation**: Setup and performance comparison.
5. **Confusion Matrix Analysis**: Metrics and insights.
6. **Overfitting Discussion**: Observations and solutions applied.
7. **Visualizations**: Plots and explanation.
8. **Conclusion**: Summary and improvement suggestions.

Additional Notes:

- Use Jupyter Notebooks or Python scripts.
- Collaboration and clear task division are important.
- Cite external resources if used.
- Explain the **concepts** behind every technique you apply.

Deadline:

Due: 3rd May, 2025