K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) Classification

Problem Statement

You are provided with the famous Iris flower dataset, which contains measurements of different iris flowers and their species.

Your task is to:

1. Preparing the dataset.
2. Implementing KNN classification from scratch **(no sklearn, no pandas)**.
3. Tuning the parameter k and comparing validation accuracy.
4. Testing the best model on the test set.
5. Reporting all results.

About the Dataset: Iris Dataset

Property	Details
Number of Instances	150 samples (50 from each class)
Number of Attributes	4 numeric features + 1 class label
Features	Sepal length (cm), Sepal width (cm), Petal length (cm), Petal width (cm)
Target Classes	Iris-setosa, Iris-versicolor, Iris-virginica

Step 1: Load the dataset into a 2D Python list.
Step 2: Randomly split the dataset into:

- Training set: 70%
- Validation set: 15%
- Test set: 15%

Use random numbers to assign samples to these groups.

```
Train_set = [], Val_set = [], Test_set = []

For each sample S in D:
   Generate random number R in [0, 1]
   If R >= 0 and R <= 0.7:
      Add S to Train_set
   Else if R > 0.7 and R <= 0.85:
      Add S to Val_set
   Else:
      Add S to Test_set
```

KNN Classification Tasks

```
For each sample V in Validation set:
   For each sample T in Training set:
      Compute Euclidean distance between V's features and T's features
      Add (T, distance) to list L
   Sort list L in ascending order by distance
   Take the first k samples from L (nearest neighbours)
   Count the majority class among these k samples → predicted_class
   Now, check if this class is correct or not

Validation Accuracy = (correct_predictions / total validation samples) * 100
```

- Code Template
- Use k = 5 initially.
- For each validation sample:
    1. Compute Euclidean distance to all training samples.
    2. Sort training samples by distance.
    3. Select the top k neighbours.
    4. Assign the majority class among neighbours as the predicted class.
    5. Check if the prediction matches the true label.
- Calculate validation accuracy as:

$$\frac{correct\ validation\ samples}{Total\ \ validation\ samples} * 100$$

- Repeat the process for: k = 1, 3, 5, 10, 15
  Make a results table:

K Value	Validation Accuracy (%)
1	
3	
5	
10	
15	

- Choose the best k (highest validation accuracy).

- Apply this best k on the test set to report final test accuracy.
- **Do not use Libraries like sklearn, scikit-learn, or pandas for the algorithm; use them only for loading the raw data if needed.**
- Ensure your code is well-commented and your report is clear.
- **Any copying or plagiarism will result in -100% penalty.**

Marks Distribution

Task	Marks
Dataset loading	4
Train/Validation/Test split	5
KNN algorithm implementation	10
K tuning + validation accuracy table	3
Best K-test accuracy calculation	3
Total	25

K-Nearest Neighbours (KNN) Regression

Problem Statement

You are provided with the Diabetes dataset, which contains medical data and a quantitative measure of disease progression one year after baseline.

Your task is to:

1. Prepare the dataset.
2. Implement KNN regression from scratch **(no sklearn or pandas)**.
3. Tune the parameter k and compare the validation error.
4. Test the best model on the test set.
5. Report all results.

About the Dataset: Iris Dataset

Property	Details
Number of Instances	768
Number of Attributes	8 numeric predictive features + 1 quantitative target
Features	Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome
Target	Disease progression measure after one year.

Dataset Preparation:

Step 1: Load the dataset into a 2D Python list.
Step 2: Randomly split the dataset into:

- Training set: 70%
- Validation set: 15%
- Test set: 15%

Use random numbers to assign samples to these groups.

```
Train_set = [], Val_set = [], Test_set = []

For each sample S in D:
   Generate a random number R in [0, 1]
   If R >= 0 and R <= 0.7:
      Add S to Train_set
```

```
  Else if R > 0.7 and R <= 0.85:
    Add S to Val_set
  Else:
    Add S to Test_set
```

KNN Regression Tasks

```
Total_Error = 0

For each sample V in the Validation set:
      L = []
   For each sample T in the Training set:
      Compute Euclidean distance between V's features and T's features
      Add (T, distance) to list L
   Sort list L in ascending order by distance
   Take first k samples from L (nearest neighbours)
   Compute the average output (target) from these k samples → Predicted_Value
   Compute squared error:
      Error = (V_true_output - Predicted_Value)^2
   Add Error to Total_Error

Mean_Squared_Error = Total_Error / (number of validation samples)
```

- Code Template
- Use k = 5 initially.
- For each validation sample:
    1. Compute Euclidean distance to all training samples.
    2. Sort training samples by distance.
    3. Select the top k neighbours.
    4. Average their target values → predicted output.
    5. Computer squared error:
- Calculate validation accuracy as:

$$\frac{correct\ validation\ samples}{Total\ \ validation\ samples} * 100$$

- Repeat the process for: k = 1, 3, 5, 10, 15
  Make a results table:

K Value	Validation Accuracy (%)
1	
3	
5	
10	
15	

- Choose the best k (highest validation accuracy).
- Apply this best k on the test set to report final test accuracy.
- **Do not use Libraries like sklearn, scikit-learn, or pandas for the algorithm; use them only for loading the raw data if needed.**
- Ensure your code is well-commented and your report is clear.
- **Any copying or plagiarism will result in -100% penalty.**

Marks Distribution

Task	**Marks**
Dataset loading	4
Train/Validation/Test split	5
KNN regression algorithm implementation	10
K tuning + validation accuracy table	3
Best K-test accuracy calculation	3
Total	25