Experiment 5

Aim

To implement and analyze the **K-Nearest Neighbors (KNN)** classification model on a real-world Heart Disease dataset using Python, and evaluate its performance using appropriate classification metrics.

---

1. Dataset Source

Dataset – KNN Classification
Iris Dataset

Target Variable: `Species`

Source:
https://www.kaggle.com/datasets/uciml/iris

---

2. Dataset Description

Features:

- **SepalLengthCm** — sepal length in cm
- **SepalWidthCm** — sepal width in cm
- **PetalLengthCm** — petal length in cm
- **PetalWidthCm** — petal width in cm

Target:

- **Species** — class of iris flower
    - *Iris-setosa*
    - *Iris-versicolor*
    - *Iris-virginica*

3. Mathematical Formulation of the Algorithm

K-Nearest Neighbors (KNN)

KNN is a non-parametric supervised classifier that assigns a class based on the majority vote among the *k* closest training samples.

1. Distance Calculation (Euclidean Distance)

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Where:

- xxx = a test point
- yyy = a training point
- nnn = number of features

2. Classification Rule

For a given test point, find *k* nearest neighbors → the predicted class is the **most frequent class** among them.

$$\hat{y} = \mathrm{mode}(y_1, y_2, \ldots, y_k)$$

4. Algorithm Limitations

- Choice of K matters: low *k* → noisy decision boundaries; high *k* → smoothing may reduce detail.
- Computationally intense for large training sets.
- Performance decreases in high dimensions.

- Sensitive to feature scaling; distance metric assumes similar scales for all features.

---

5. Methodology / Workflow

Common Workflow

1. Load dataset using pandas
2. Select independent (X) and dependent (y) variables
3. Scale features (standardize)
4. Split into train and test sets
5. Train KNN model
6. Predict on test set
7. Evaluate using classification metrics

---

6. Performance Analysis

Evaluation Metrics

- Accuracy Score
- Confusion Matrix
- Precision
- Recall
- F1-Score

Interpretation

- **Higher accuracy** indicates better overall classification
- **Confusion matrix** reveals true vs false classifications
- **Precision** indicates proportion of correct positive predictions
- **Recall** measures ability to identify true positives
- **F1-Score** balances precision and recall

---

7. Hyperparameter Tuning

Parameter	Meaning
n_neighbors (K)	Number of neighbors to consider
weights	‘uniform’ or ‘distance’
metric	Distance measure (e.g., Euclidean)
algorithm	How neighbors are computed

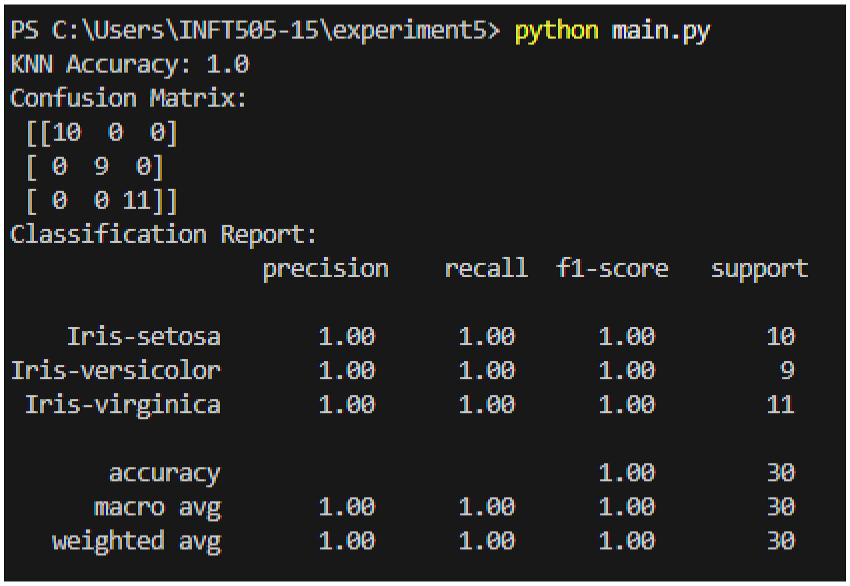
Exercise 1: Decision Tree Classification

Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix

# Load dataset
df = pd.read_csv('Iris.csv')  # from Kaggle
```

```
# Feature and target selection
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm']]
y = df['Species']

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)

# Model training
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)

# Predictions
y_pred_knn = knn_model.predict(X_test)

# Evaluation
print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
print("Confusion Matrix:\n", confusion_matrix(y_test,
y_pred_knn))
print("Classification Report:\n", classification_report(y_test,
y_pred_knn))
```

Output



```
PS C:\Users\INFT505-15\experiment5> python main.py
KNN Accuracy: 1.0
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Classification Report:
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

Conclusion

This experiment successfully implemented the **K-Nearest Neighbors (KNN)** classification model on the *Iris* dataset. The KNN classifier predicts the species of an iris plant based on the characteristics of similar (nearest) flowers in the training dataset. Feature scaling was essential for distance-based modeling. Evaluation metrics such as accuracy, confusion matrix, precision, recall, and F1-score provided insights into model performance for multi-class classification.