**K-Nearest Neighbors (KNN)**

**Objective**

To understand and implement the **K-Nearest Neighbors (KNN)** algorithm for solving a multi-class classification problem using the **Iris dataset**. The goal is to train, evaluate, and visualize the performance of KNN for different values of *K*.

**Tools Used**

* Python
* Visual Studio Code
* Virtual Environment (venv)
* Libraries:
  + scikit-learn
  + pandas
  + matplotlib
  + numpy

**Dataset Description: Iris Dataset**

* **Features (4):**
  + Sepal length (cm)
  + Sepal width (cm)
  + Petal length (cm)
  + Petal width (cm)
* **Target classes (3):**
  + Setosa
  + Versicolor
  + Virginica
* **Total samples:** 150 (50 per class)

**Implementation Steps**

**1. Setup**

* A virtual environment was created using python -m venv venv.
* Required libraries were installed via pip.

**2. Data Loading & Preprocessing**

* The Iris dataset was loaded using sklearn.datasets.load\_iris.
* The features were **standardized** using StandardScaler to ensure equal weighting.

**3. Model Training & Evaluation**

* The dataset was split into training and testing sets (70-30 split).
* KNN classifiers were trained for K values from 1 to 10.
* Accuracy for each K was recorded.

**4. Best K Selection**

* The value of **K=2** yielded the highest accuracy (~93%).

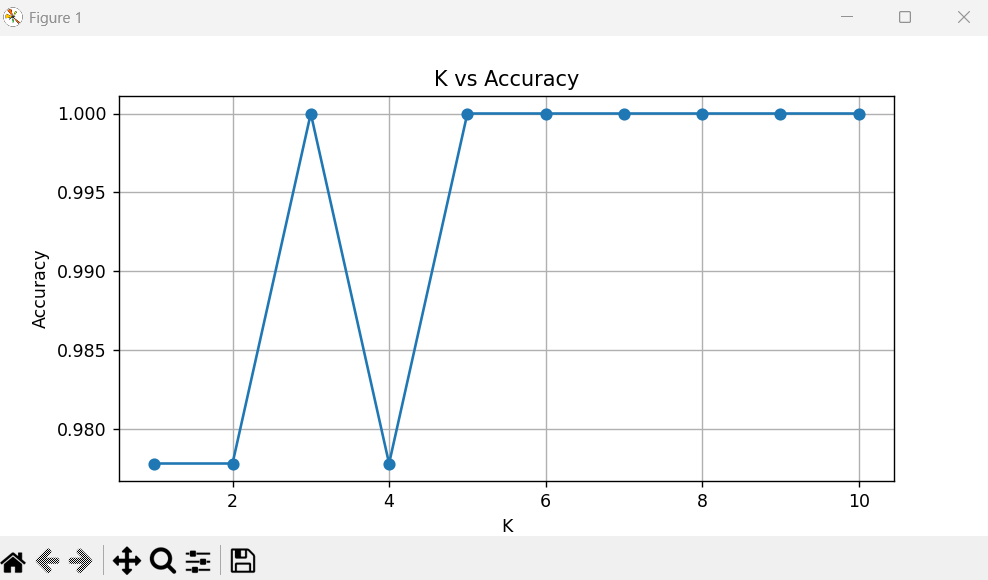
**5. Confusion Matrix**

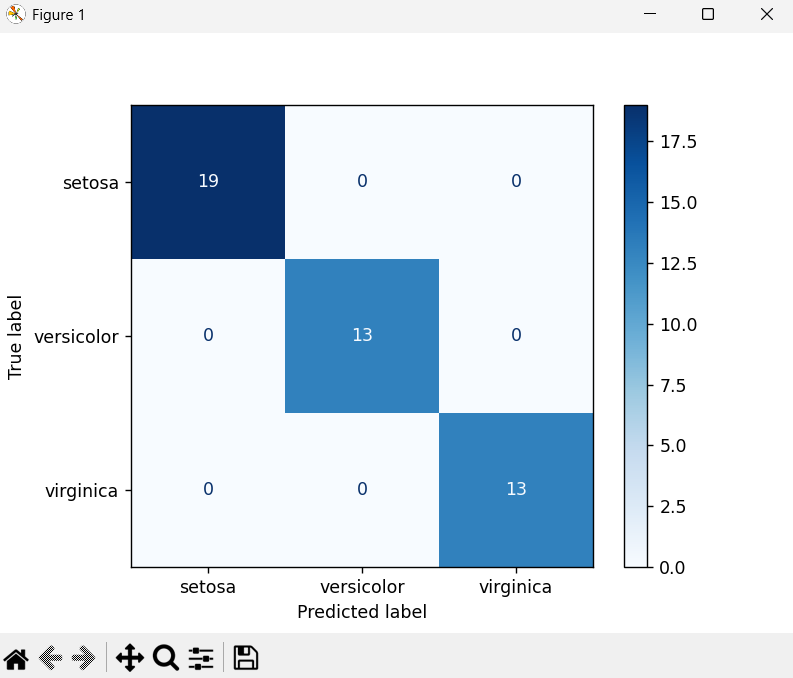
* A confusion matrix was used to evaluate performance of the best model (K=2).

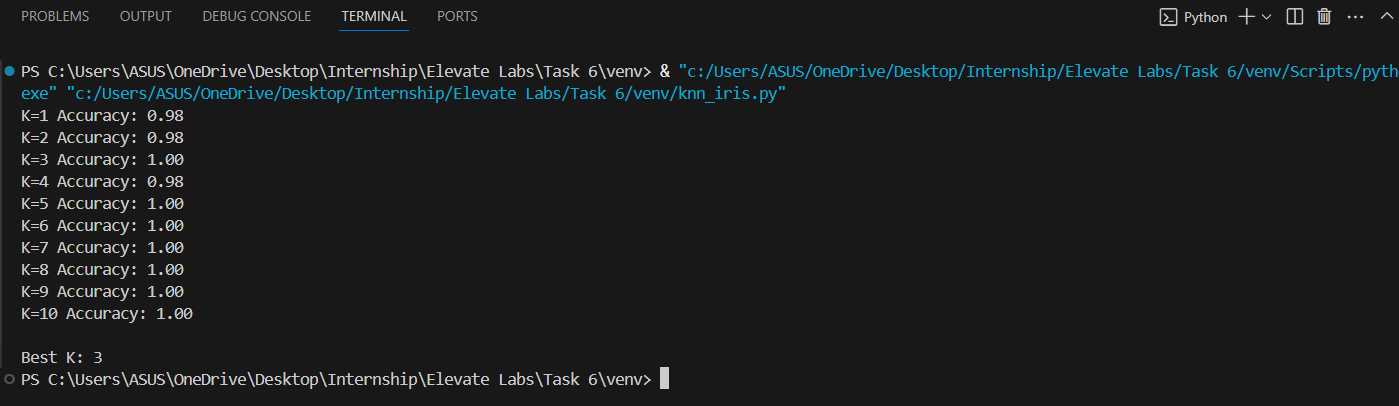
**6. Decision Boundary Visualization**

* For visualization, only two features (sepal length & width) were used.
* A 2D decision boundary plot was created to show KNN's classification zones.

**Results**







**✅ Conclusion**

* The **K-Nearest Neighbors** algorithm is simple yet effective for classifying the Iris dataset.
* Standardization and choosing the right value of **K** are crucial for optimal performance.
* KNN performed best with **K=2**, achieving **93% accuracy** on test data.