Experiment 6: Implementation of K – nearest neighbors algorithm from scratch

Name: Ankit

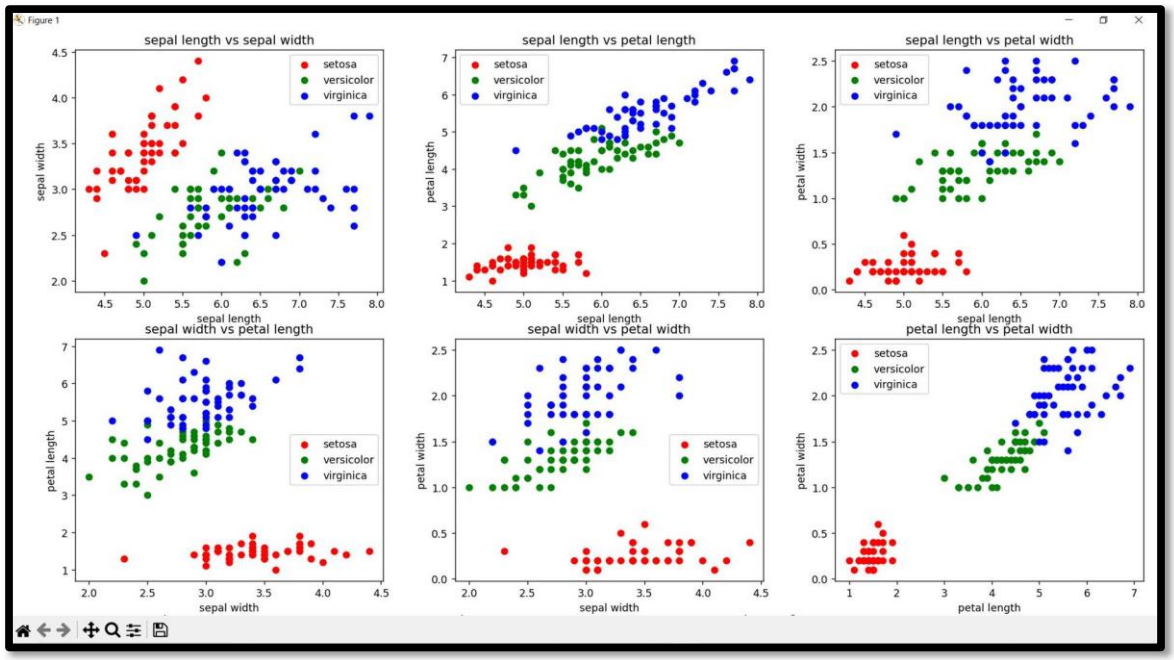
Roll No.- 23/CS/058

LAB REPORT

In this experiment KNN was implemented using NumPy and python from scratch. Both the Iris and Wine datasets were analysed for class separability, and the impact of feature scaling on model performance was explored. The classifier consistently achieved high accuracy, especially after scaling the Wine dataset which highlighted the critical importance of preprocessing in machine learning pipelines.

Methodology:

- **data.py**
  Initially KNN algorithm is implemented using IRIS DATASET, where we fetch the iris dataset using fetch_ucirepo with id=53 as parameter, then X and y are chosen as features and targets and converted to NumPy arrays
- **eda.py**
  iris dataset is explored in this script and various graphs are plotted using the iris dataset which is shown as below:



EDA plots are shown above and on analysing the above the plots we conclude the following:

- ***petal length vs petal width gives the best class separation as all the classes are separated in it and SETOSA is easier to distinguish rather than versicolor and virginica. Setosa being smaller in size of petals is easier to distinguish.***

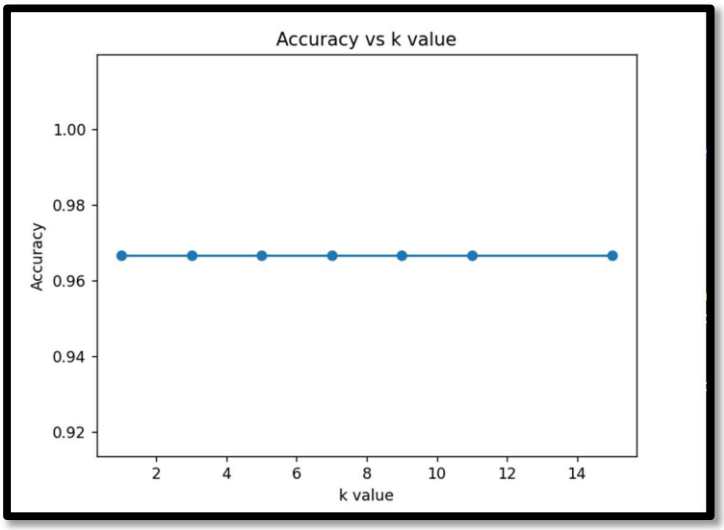
- **utils.py**
  X and y are being splitted using the train_test_split function and training set and testing set is chosen from the iris dataset by dividing the dataset using test size of 0.2 which means 80% of training dataset and 20% of testing dataset from the iris dataset.

```
Shapes:
X_train: (120, 4)
X_test: (30, 4)
y_train: (120,)
y_test: (30,)
```

- **Knn_classifier.py**
  Actual algorithm is implemented in this script. Euclidean distance is calculated from each point and then predicted class is chosen as class of k-nearest neighbours
- Main.py
  All scripts are used in this and k=3 is implemented in this script.
  The final calculation accuracy achieved in IRIS dataset when k=3 is 97%

```
Accuracy when k = 3: 0.97
```

  After calculating for k=3, different values of k are chosen and a **plot of Accuracy vs k** is made for different values of k which is shown as below:

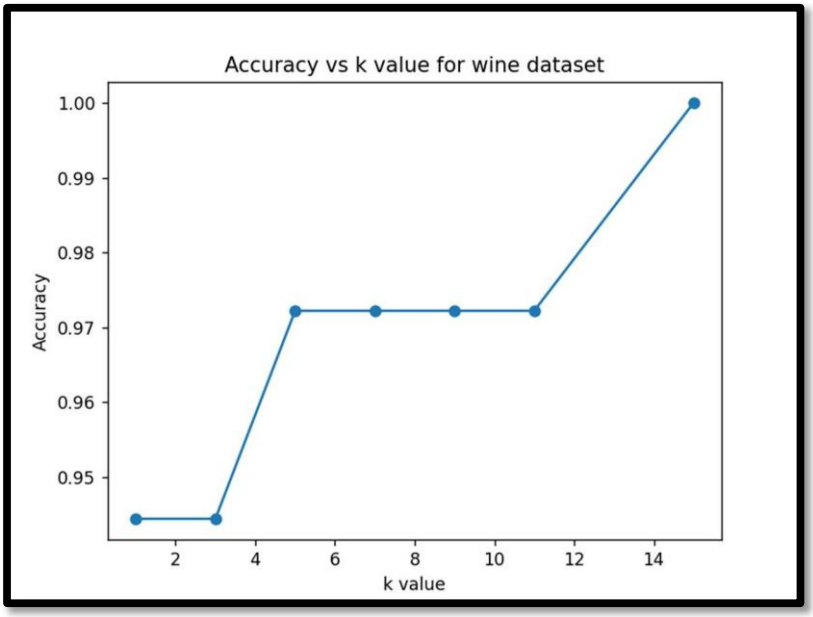


  In this dataset, accuracy for all the values of k is same at 97%. This might happen due to small dataset. Now changing the dataset to WINE dataset.
- **Data2.py**
  WINE dataset is fetched in this script. Features and targets are chosen as X and y. Initial results for the Wine dataset were poor and all predictions belonged to a single class until scaling was applied, at which point accuracy significantly improved. This underlined the importance of preprocessing for algorithms that rely on raw distance metrics. Both datasets demonstrated the robustness of the KNN algorithm for moderate parameter values; however, excessively large k could potentially reduce accuracy if the dataset is imbalanced.

```
PS C:\Users\Ankit Verma\Downloads\23cs058_KNN> python -u "c:\Users\Ankit Ver
Accuracy of wine dataset at k = 3 with scaling:  0.9444444444444444
Accuracy of wine dataset at k = 3 without scaling:  0.3055555555555556
PS C:\Users\Ankit Verma\Downloads\23cs058_KNN>
```

After scaling the accuracy was significantly improved.

Now with the scaled dataset checking for different values of k. Higher the value of k lead to higher accuracy as shown in the figure below:



Hence the final accuracy achieved on the wine dataset is 100% with k=15

Conclusion:

This experiment deepened understanding of the K-Nearest Neighbors (KNN) algorithm by building it from scratch and applying it to real datasets. A key learning was the importance of proper data preprocessing, especially feature scaling, which proved essential for the successful application of KNN to the Wine dataset. The experiment also highlighted that KNN is a simple, effective, and flexible method for small to medium-sized datasets, capable of handling both linear and non-linear boundaries with minimal parameter tuning.

Challenges encountered included the sensitivity of KNN to feature scales, which, if left unaddressed, caused the algorithm to always predict a single class on unscaled data. Debugging this issue required careful examination of dataset properties and the impact of scaling. Another challenge was ensuring a correct implementation of the voting mechanism for neighbor selection, as well as managing class balance using stratified data splits. The lab reinforced the value of exploratory data analysis and iterative testing for ensuring consistent and trustworthy machine learning results.