k-Nearest Neighbors (k-NN)

Roll No: G25AIT1072

Name : Jayant Parmar

Abstract

This report details the implementation of the k-Nearest Neighbors (k-NN) classification algorithm from scratch using Python. The primary objective was to build, evaluate, and understand the behavior of the k-NN model on the well-known Iris dataset. The project involved comprehensive exploratory data analysis (EDA), building a custom k-NN class, performing hyperparameter tuning for the value of 'k', and visualizing the model's decision boundaries. The analysis focuses on the impact of 'k' on model performance and its relationship with the bias-variance trade-off. Additionally, a bonus investigation into alternative distance metrics and weighted voting was conducted.

1. Methodology

1.1. Dataset and Preprocessing

The project utilized the Iris dataset from the UCI Machine Learning Repository, which contains 150 samples of iris flowers across three species: Iris-setosa, Iris-versicolor, and Iris-virginica. Each sample is described by four features: sepal length, sepal width, petal length, and petal width.

The data preparation followed these key steps:
- Data Splitting: The dataset was partitioned into an 80% training set (120 samples) and a 20% testing set (30 samples). Stratification was used to maintain the same proportion of classes in both sets.
- Feature Scaling: As a distance-based algorithm, k-NN is sensitive to the scale of features. Therefore, all four features were standardized to have a mean of 0 and a standard deviation of 1 using StandardScaler from scikit-learn.

1.2. k-NN Algorithm Implementation

A custom Python class, KnnClassifierAndPipeline, was developed to encapsulate the entire workflow without relying on built-in k-NN classifiers like sklearn.neighbors.KNeighborsClassifier. The implementation adheres to the fundamental principles of the k-NN algorithm:

- fit(X_train, y_train): Stores the training data and corresponding labels.
- predict(X_test): For each point in the test set:
  - Computes the Euclidean distance between the test point and all training points.
  - Identifies the 'k' nearest neighbors.
  - Performs majority voting to predict the class.

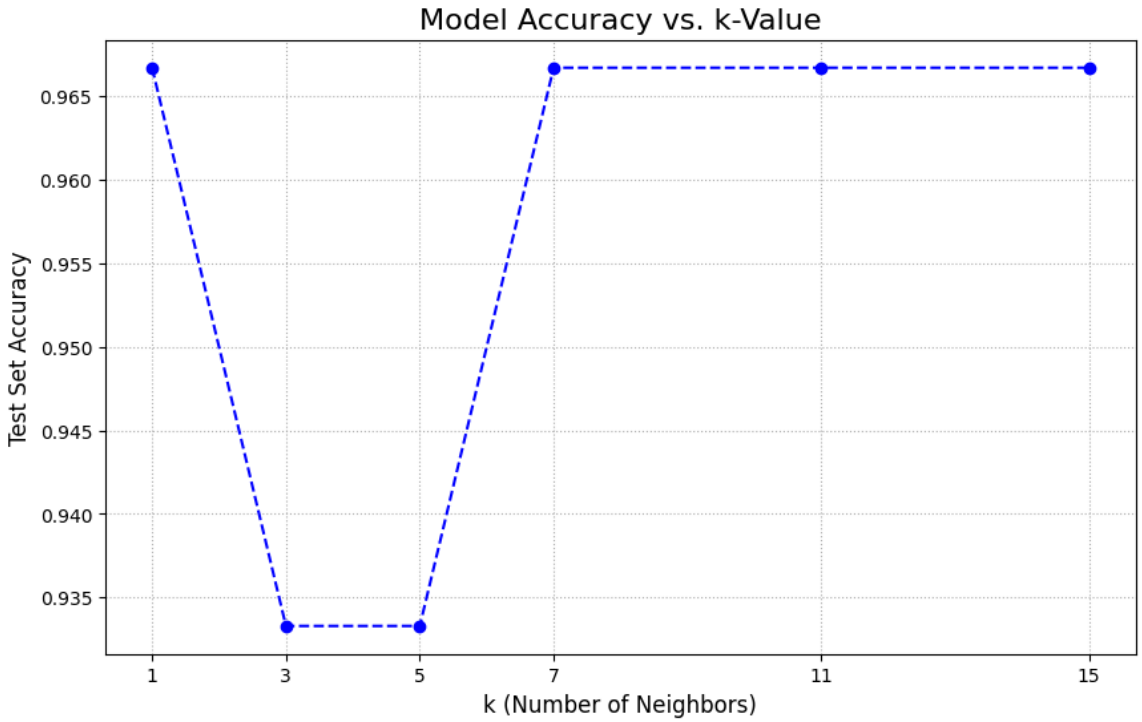
2. Exploratory Data Analysis (EDA)

A preliminary EDA revealed several key characteristics of the dataset:
- The dataset is clean, with no missing values, and balanced with 50 instances for each of the three species.
- The pairplot and correlation matrix showed strong positive correlations between petal_length, petal_width, and sepal_length.
- Visually, the Iris-setosa species is linearly separable, while Iris-versicolor and Iris-virginica overlap.
- Petal length and petal width were identified as the most informative features.

3. Results and Evaluation

3.1. Hyperparameter Tuning for 'k'

The model's performance was evaluated on the test set for various values of 'k': 1, 3, 5, 7, 11, and 15.



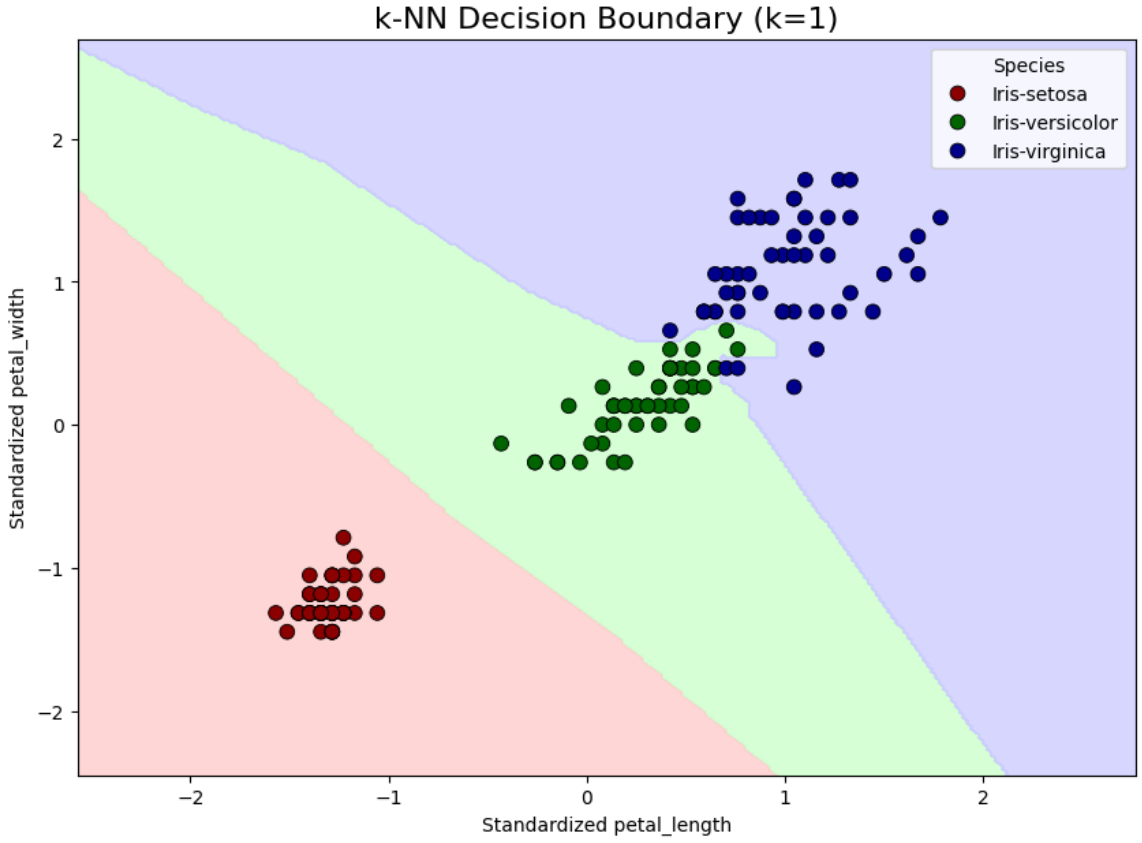
k-Value	Test Accuracy
1	96.67%
3	93.33%
5	93.33%
7	96.67%

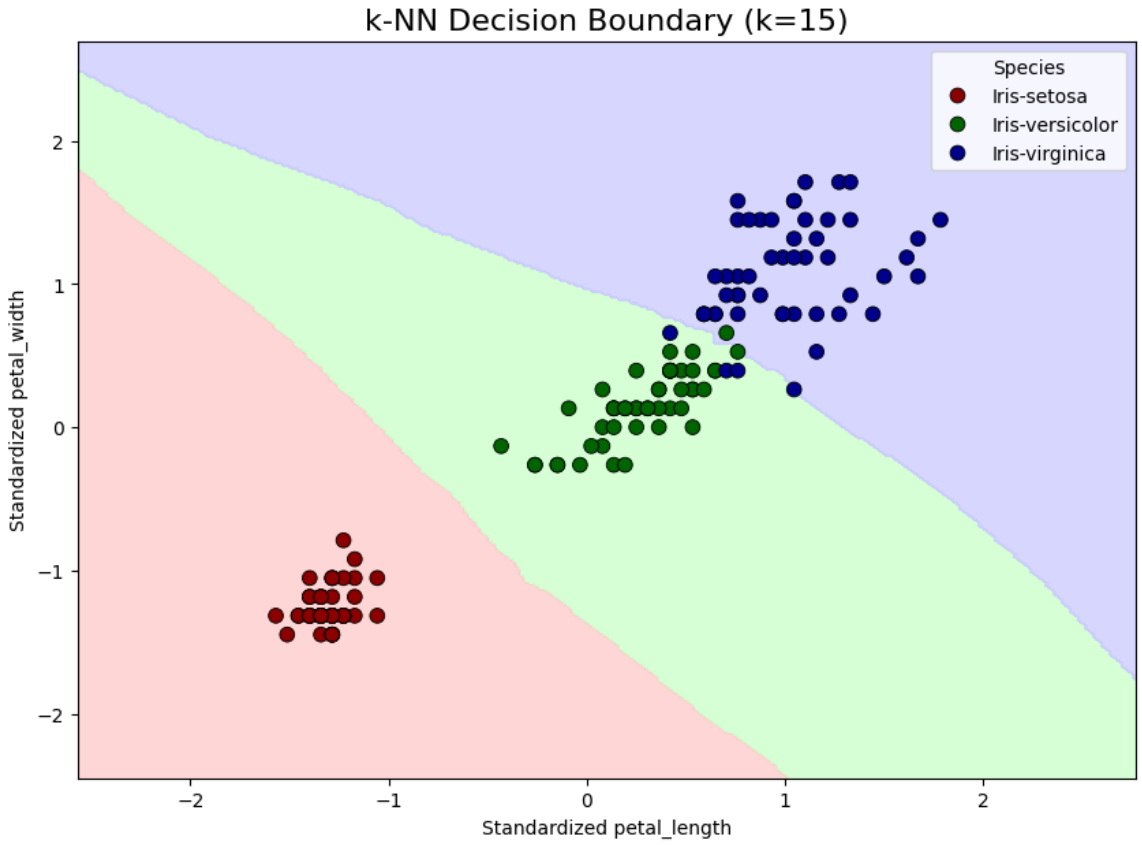
11	96.67%
15	96.67%

The relationship between 'k' and model accuracy is visualized in the plot below (not included).

3.2. Decision Boundary Visualization

The model was retrained using petal length and petal width, and decision boundary plots were generated for k=1 and k=15.
Figure 1: Decision boundary for k=1 — complex boundary, high variance.
Figure 2: Decision boundary for k=15 — smoother boundary, high bias.





4. Analysis and Discussion

4.1. Analysis of 'k' Value

The 'Accuracy vs. k-Value' plot shows peak accuracy at k=1, slight dips at k=3 and k=5, then stabilization at 96.67% for k≥7. k=7 offers a balanced choice between variance and bias.

4.2. The Bias-Variance Trade-off

Low k (k=1): Low bias, high variance. The decision boundary closely fits training data — risk of overfitting.
High k (k=15): High bias, low variance. The boundary is smoother — risk of underfitting.

5. Bonus Challenge Results

Model Configuration	Test Accuracy
Standard (Euclidean, k=7)	96.67%
Manhattan Distance (k=7)	96.67%
Weighted Voting (Euclidean, k=7)	100.00%

Weighted k-NN achieved 100% accuracy by giving higher weights to closer neighbors, outperforming standard majority voting.

6. Conclusion

The k-NN algorithm was successfully implemented from scratch and evaluated on the Iris dataset. An optimal 'k' value around 7 achieved 96.67% accuracy. Decision boundary visualization illustrated the bias-variance trade-off, and a weighted voting scheme further improved accuracy to 100%.

7. Resources

Dataset: Dua, D. & Graff, C. (2019). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science.
Iris Dataset Link: https://archive.ics.uci.edu/dataset/53/iris