Exploring the Performance of K-Nearest Neighbors (KNN) on the Fisher Iris Dataset

1st Loc Pham
Department of Computer Science
Denison University
Granville, Ohio
pham_l2@denison.edu

2nd Phuc Nguyen
Department of Computer Science
Denison University
Granville, Ohio
nguyen_p3@denison.edu

***Abstract*—In this paper, we evaluate the effectiveness of the k-Nearest Neighbors (KNN) algorithm for species classification using the Fisher Iris dataset, with a focus on how the choice of k impacts accuracy. Our findings reveal two key insights. First, KNN performs well on this dataset, achieving consistently high accuracy. Second, the relationship between k and classification accuracy follows an inverted U-shape: accuracy is low with small k values due to overfitting, increases to a peak as k grows, and then declines with larger k values as underfitting sets in.**

I. Introduction

The k-Nearest Neighbors (KNN) algorithm [1] is a fundamental technique in machine learning known for its simplicity and effectiveness in classification tasks. KNN works by assigning a label to an unknown data point based on the majority label of its nearest neighbors. Despite its ease of use, KNN can be sensitive to several factors, including the number of neighbors k, the distribution of the dataset, and the absence of feature scaling. This paper focuses on evaluating the impact of the parameter k on KNN's performance on the Fisher Iris dataset [2], a well-known dataset used in classification problems.

The remainder of this paper is organized as follows: Section 2 details the methodology employed in our study. Section 3 presents and discusses the results of our experiments. Finally, Section 4 summarizes our findings and discusses their implications.

II. Methodology

We use the Fisher Iris dataset for our experiment. It contains 150 samples of Iris flowers, divided equally among 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 goal is to classify each sample into one of the three species based on these features. The dataset is suitable for evaluating the effectiveness of KNN and the impact of parameter k due to its relatively small size and balanced distribution of classes.

Given the differences in scales between the features, preprocessing steps such as normalization can be used before applying the KNN algorithm to ensure that all features contribute equally to the distance calculations. Without this step, features with larger scales, such as petal length, could dominate the distance metric, potentially skewing the results. To achieve this, we use the preprocessing module in the scikit-learn library.

In addition, for a supervised learning task, the data should be divided into a training set and a testing set. In this experiment, we use 2/3 of the data for training and 1/3 for testing. The decision to employ an 2/3 - 1/3 split for the dataset is motivated by the need to balance the training effectiveness and robust model evaluation. But still, the results of a single training-testing split can be misleading in this case, as the model's performance depends heavily on the specific samples chosen for training and testing. To address this, we employ stratified k-fold sampling built in the scikit-learn library with $k = 50$. This technique ensures that each fold maintains the same class distribution as the full dataset, preventing imbalances in the training and testing sets. By averaging the results over 50 different training-testing splits, we obtain a more accurate estimate of the model's generalization performance.

For the implementaion of the KNN algorithm, we make use of the scikit-learn library again. In essence, it is a lazy, instance-based learning algorithm where it operates by calculating the distance between a query point and all other points in the dataset and assigns the label of the majority class among the k nearest neighbors. KNN's simplicity lies in its lack of an explicit training phase and its reliance on memorizing the training data for classification at the testing stage. Pseudocode is provided in Algorithm 1 below.

Algorithm 1 k-Nearest Neighbors (KNN) Algorithm

```
Input: A dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ with labeled instances, query instance $q$, and parameter $k$ (number of neighbors)
Output: Predicted label for the query instance $q$
For each query instance $q$:
    1) Calculate the distance between $q$ and all instances in $D$
    2) Select the $k$ nearest neighbors based on the calculated distances
    3) Assign the label to $q$ based on the majority label of the $k$ nearest neighbors
```

III. Result discussion

In this section, we conduct a detailed analysis of the graphical representations derived from our experimental data. The aim is to interpret the patterns in the graphs, thereby providing a comprehensive understanding of the KNN's performance on the Fisher Iris dataset.

A. *Impact of k on Classification Accuracy*

The value of k is critical as it determines how many neighbors are considered when making a classification decision. Smaller values of k lead to a more "local" decision rule, where the nearest neighbor has a significant influence on the classification of a sample. As k increases, the decision rule becomes more global, considering a wider neighborhood around the query point.

The following graph in Figure 1 illustrates the accuracy of KNN across different values of k, averaged over 50 stratified folds. As a whole, it yields a relatively high accuracy (above 0.93) regardless of k we choose, which shows the effectiveness of KNN on this dataset. In addition, the graph has an inverted-U shape, and several key observations can be made:

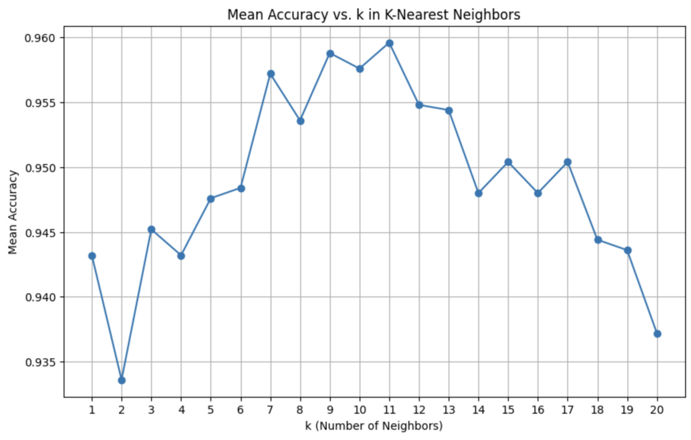

Fig. 1: Graph of Accuracy vs k.

- For $k = 1 \ldots 2$, the algorithm's accuracy starts low on the testing set, indicating a tendency toward overfitting. With only one or two neighbors considered, the model is highly sensitive to noise and small variations in the data, which often results in misclassifying outliers or anomalous points in the test data.
- As k increases to moderate values (e.g., $k = 9 \ldots 11$) and the algorithm incorporates more neighbors in its decision, the accuracy on the testing set increases and peaks. This suggests that the model has struck a balance between sensitivity to local structure and generalization, reducing the risk of overfitting while maintaining high accuracy.
- For larger values of k (e.g., $k \geq 11$), the algorithm exhibits signs of underfitting, where the decision boundaries become too smooth and fail to capture the underlying patterns of the data. In these cases, KNN begins to misclassify points because the set of nearest neighbors includes too many instances from other classes, diluting the local structure of the data.

These trends align with the intuition that smaller k values are better at capturing local patterns, while larger k values lead to more generalized but less precise predictions.

B. *Visualization of Misclassifications*

To better understand our model's behavior, we use a stacked bar chart in Figure 2 to depict the percentage of misclassifications for each k value, broken down by species.

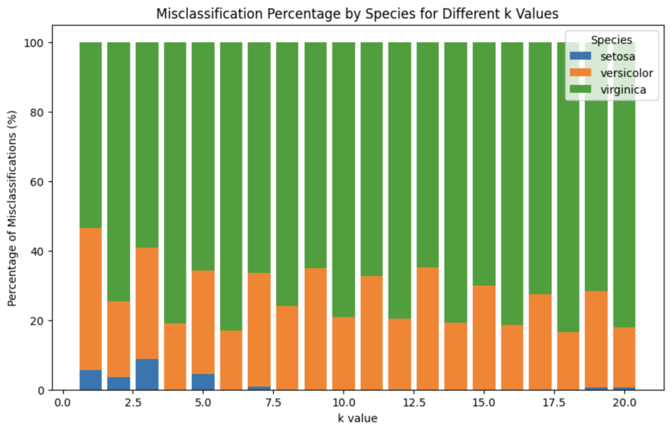

Fig. 2: Graph of Misclassification Percentage by Species for Different k Values

The graph reveals that our KNN model struggles the most with distinguishing between Iris Virginica and Iris Versicolor. Regardless of the k value, misclassifications involving these two species consistently dominate. This is likely due to the inherent similarity in their feature distributions. In contrast, Iris Setosa is almost always classified correctly, especially when k exceeds 4. This suggests that the Setosa class is more distinct in feature space, while the overlap between Virginica and Versicolor leads to classification challenges.

C. *Visualization of the Dataset on 2D Space*

To gain deeper insights into why the KNN model performs as it does, we can conduct a 2D projection of the Fisher Iris dataset using Principal Component Analysis (PCA) [3]. This allows us to reduce the dimensionality from four features to two, while still preserving the majority of the variance in the data. Figure 3 provides this 2D visualization, where the three Iris species are represented by different colors.

The plot clearly illustrates the separability of the three species. Iris Setosa is linearly separable from the other two species, reinforcing its nearly flawless classification performance. However, Iris Versicolor and Iris Virginica exhibit significant overlap in the 2D space. This overlap explains the difficulty our KNN model has in distinguishing between these two species, as seen in both the misclassification chart and overall accuracy metrics.

This 2D visualization also provides further evidence regarding the impact of k. When k is too small, KNN becomes highly

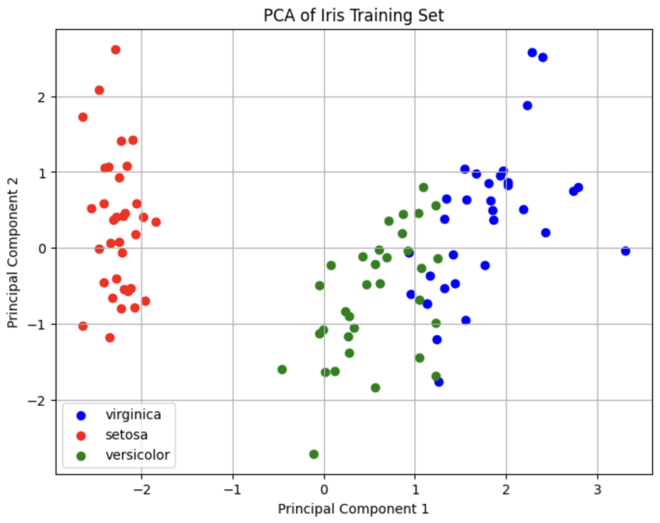

Fig. 3: 2D Visualization of Fisher Iris Dataset

sensitive to noise, often misclassifying boundary points. On the other hand, when k is too large, the model smooths over the boundaries between overlapping classes, leading to increased misclassification in regions where Versicolor and Virginica are close in feature space. This observation aligns with the trends seen in the accuracy and misclassification graphs, providing a visual confirmation of our earlier conclusions.

IV. Conclusion

Our experiment has provided valuable insights into the application of KNN in species classification in the Fisher Iris dataset and the importance of choosing an appropriate k. Our first finding highlights the algorithm's ability to classify species with high accuracy. The second conclusion reveals nuances in choosing the optimal number of nearest neighbors k. Our results indicate that while KNN is prone to overfitting with small values of k, increasing k too much can lead to underfitting. Multiple runs with different values of k can be done to determine an optimal k at which the algorithm can balance the need for local structure while avoiding noise.

Moving forward, future work could explore the effectiveness of KNN in different settings such as with various feature scaling techniques, as well as on datasets with varying sizes, dimensionalities, or class imbalances.

References

[1] "K-Nearest Neighbors Algorithm," Wikipedia, Wikimedia Foundation, 4 Oct. 2024, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.
[2] "Iris Flower Data Set." Wikipedia, Wikimedia Foundation, 29 Sep. 2024, https://en.wikipedia.org/wiki/Iris_flower_data_set.
[3] "Principal Component Analysis." Wikipedia, Wikimedia Foundation, 21 Oct. 2024, https://en.wikipedia.org/wiki/Principal_component_analysis.