Evaluating the Performance of K-Nearest Neighbors Classification

Lillian Mueller lmuelle1@umd.edu
Regina Hong rhong@umd.edu

***Abstract*— Given a dataset of Iris characteristics, this report explored the development of a similarity model to classify Iris species. A K-Nearest Neighbor (K-NN) model was used in this classification process and modified by varying the number of nearest neighbors and the distance metric used during implementation. After performing several iterations of k-fold cross validation, this model was then compared to a Decision Tree model as well as a Logistic Regression model. It was observed that the Euclidean and Manhattan metrics of determining nearest neighbors resulted in a higher accuracy model. Additionally, the K-NN model (with 10 nearest neighbors) generally performed better than the Decision Tree model but not as well compared to the Logistic Regression model. K-NN classification has many parameters that can optimize the model to a dataset; understanding those parameters and how each affects the model is important to developing an adequate model for a dataset.**

I. Introduction

K-nearest neighbors (K-NN) is a type of classification model using similarity to determine the class of a new data point [1]. The data point(s) most similar, or the nearest neighbor(s) of the new data point, is based on the measures, or the features, of a dataset. K-NN usually uses Euclidean distance to calculate the nearest neighbors, but other methods may be used. This kind of model works best on smaller datasets that do not have a lot of noise.

The focal dataset is the Iris dataset, supplied by the sci-kitlearn library. This dataset contains four feature measurements for each Iris flower: sepal length, sepal width, petal length, and petal width. Alongside these features, the classification of each Iris flower is given; the flower is classified as setosa, versicolor, or virginica. The K-nearest neighbors model was developed using the `neighbors.KneighborsClassifier` class under the aforementioned library. The model was created with the intent of seeing how accurate it would be in classifying an Iris flower based on its feature measurements. In addition to the K-NN model, a logistic regression model and a decision tree model were also generated to compare the generative performance of the three models with respect to the same dataset.

In this study, we use both K-nearest neighbors and k-fold cross validation. To reduce confusion between the variables, nearest neighbor values will be denoted by K and number of folds in cross validation will be denoted by k.

The Methodology section outlines the methodology used in creating the Python-based K-NN model and its performance as compared to a decision tree and logistic regression model. The Results section explores the results from the methods used, and the importance of this investigation is reviewed in the Discussion section.

II. Methodology

This investigation entails using scikitlearn's `KNeighbors` class from the `neighbors` module to develop the k-nearest neighbors model. Several Python packages were utilized including `sklearn`, `matplotlib`, `pandas`, and `numpy`. Specifically from `sklearn`, we use the following modules: `linear_model`, `preprocessing`, `model_selection`, `metrics`, `pipeline`, `inspection`, and `tree`.

The Iris dataset was loaded as an `sklearn.utils.Bunch` object called `iris_data` via the `load_iris()` function. To make the data easier to read, it was transformed into the `df_iris` dataframe, where the data parameter was set as the `data` attribute of the dataset and the columns parameter as the `feature_names` attribute. A new column called "class" was added to this dataframe which contains the `target` variable of the Iris dataset; this is the class of the Iris plant - setosa, versicolor, or virginica. Since the `target` attribute contains an array with values from 0-2, the `replace` function was used to map these numerical values to their corresponding classifications: setosa for 0, versicolor for 1, and virginica for 2.

After the Iris dataset was loaded in and processed, it was split into a train group and test group using the `test_train_split` function, with 2/3 of the data as the train group and the remainder as the test group.

We created two functions; the first function uses `KNeighborsClassifier` to perform K-NN on the testing and training data. The parameters include: testing and training data, nearest neighbor value, whether the data is scaled, and which method is used for the nearest neighbor distance calculation. The second function is used to plot the results of the K-NN model iterations. A `Pipeline` was used so that when the Iris dataset is scaled data, it can be graphed [2]. For plotting purposes, only two features were selected for all the K-NN graphs: sepal length and sepal width.

To determine which combination resulted in the best performing K-NN model, two `KNeighborsClassifier` parameters were altered: `metric` and `n_neighbors`. Keeping K=5, the `metric` parameter was set to `euclidean`, `manhattan`, and `cosine`. These distance calculations were

performed on both unscaled and scaled data. Then, setting `metric = euclidean`, the K-NN model was tested with nearest neighbor values (K) of 3, 5, and 10.

To compare the results of the K-NN model with other models, we created a decision tree model and a linear regression model using the same methods from our previous studies [3], [4]. The parameters for each model are as follows: for the linear regression model, `penalty=None`, for the decision tree, `criterion = entropy`, and for K-NN, `n_neighbors =10`, and `metric=euclidean`. K-fold cross validation [5] was used with K=5 folds to compare the mean and standard deviation of accuracy values among the three models. Additionally, the k value was changed to K=6, 8, 10, 12, and 14 to observe how the three models performed during multiple iterations of cross validation.

III. Results

A. Impact of Parameter Manipulation of K-NN Model

The results of changing the `metric` parameter are shown below. Both scaled and unscaled data were used, but only the results of scaled data are graphed.

The training data accuracy, testing data accuracy, and r2 score for Euclidean, Manhattan, and cosine distance calculations are shown in Table I, with scaled data and a consistent nearest neighbor value of 5.

Metric	Train Data Accuracy	Test Data Accuracy	r2 Score
Uniform	0.98	0.96	0.935442
Euclidean	0.98	0.96	0.935442
Manhattan	0.98	0.96	0.935442
Cosine	0.94	0.82	0.709490

TABLE I
Accuracy and r2 Scores for Euclidean, Manhattan, and Cosine Distance with Scaled Data

Figures 1, 2, 3, and 4 show the K-NN classification plots stemming from each of the different distance calculations as well as using no weights at all. Using Euclidean and Manhattan distance as well as uniform weights resulted in similar if not identical accuracy scores. The boundaries between the three Iris species are similar. Compared to the model using a uniform approach, the models using distance as a metric to weigh each nearest neighbor seems to overfit the data. Meanwhile, using cosine distance resulted in a poorly performing model by comparison. The boundaries between Iris versicolor and Iris virginica are rather erratic, resulting in streaks instead of smooth boundaries.

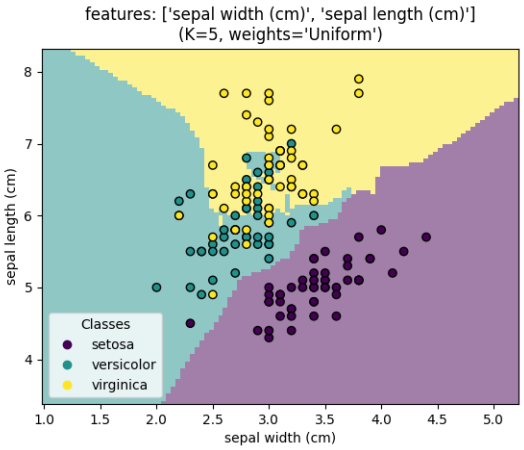

Fig. 1. Uniform Weight with K=5

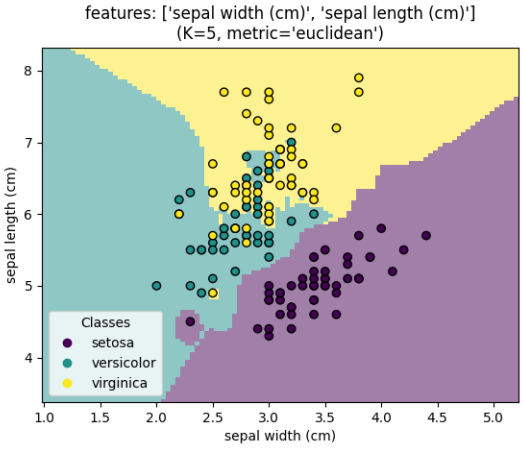

Fig. 2. Euclidean Distance Metric with K=5

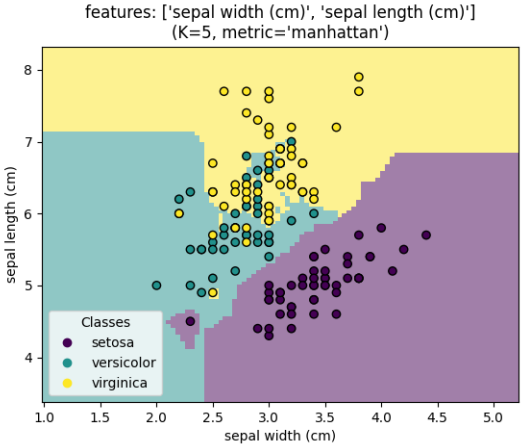

Fig. 3. Manhattan Distance Metric with K=5

A look at using the same three distance measures, this time with unscaled data, is seen in Table II. Again, Euclidean distance and Manhattan distance both resulted in similar or identical accuracy and r2 scores to using uniform weights. However, it looks like there is an improvement in cosine distance.

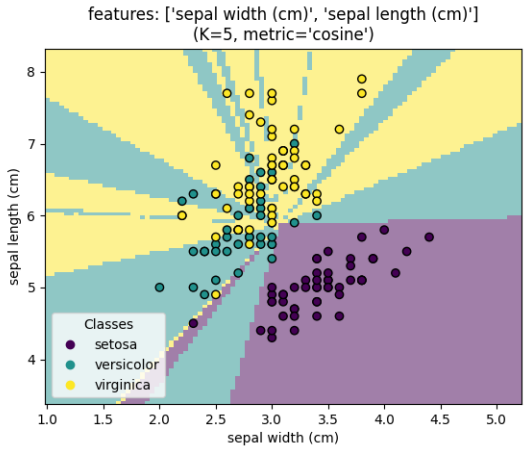

Fig. 4. Cosine Distance Metric with K=5

Metric	Train Data Accuracy	Test Data Accuracy	r2 Score
Uniform	0.97	0.98	0.967721
Euclidean	0.97	0.98	0.967721
Manhattan	0.97	0.98	0.967721
Cosine	0.98	0.96	0.935442

TABLE II
ACCURACY AND R2 SCORES FOR EUCLIDEAN, MANHATTAN, AND COSINE DISTANCE WITH UNSCALED DATA

With the `metric` parameter set to Euclidean distance, Table III shows the train and test data accuracies as well as the r2 score for different nearest neighbors values: K=3, 5, and 10. Figures 5 and 6 show the K-NN plots for K=3 and K=10 (refer to Figure 2 for K=5). A higher K-value resulted in smoother boundaries between the three Iris species. A lower K-value resulted in more oddly shaped regions even though the accuracy scores are similar to the model using K=5.

Metric	Train Data Accuracy	Test Data Accuracy	r2 Score
K=3	0.97	0.96	0.935442
K=5	0.98	0.96	0.935442
K=10	0.96	0.96	0.935442

TABLE III
ACCURACY AND R2 SCORES FOR K=3, 5, 10 WITH SCALED DATA

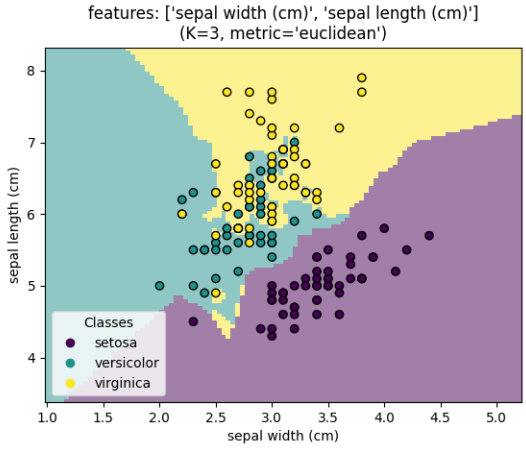

Fig. 5. Euclidean Distance Metric with K=3

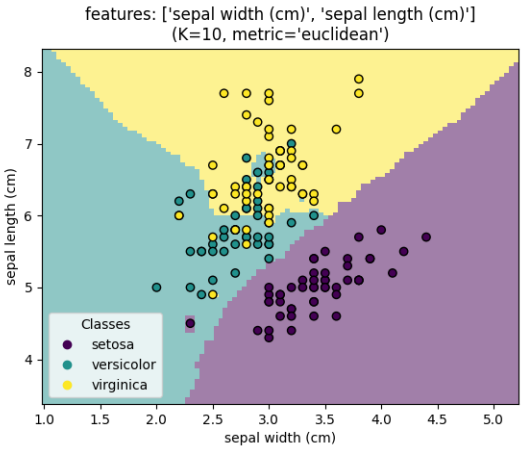

Fig. 6. Euclidean Distance Metric with K=10

From these results, the model using K=10 is the most effective as the training data accuracy goes down while the retaining its accuracy for the testing data. Additionally, visually, using K=10 and Euclidean metrics as a weight parameter results in smoother boundaries and a more generalized model. While the other models had identical accuracy when applied to the testing data set, the graphs and training set accuracy indicate those models are overfitting the data and would not be effective for future predictions.

B. Classification Model Comparison Using Cross Validation

A generalized performance comparison of the K-NN model (using Euclidean distance and 10 nearest neighbors) with the decision tree and logistic regression models is shown below in Table IV. The number of folds for cross validation was set to k=5. It should be noted that the data was not scaled. The K-NN model resulted in highest accuracy.

Statistic	Logistic Regression	Decision Tree	K-NN
Mean	0.973333	0.96	0.98
Standard Deviation	0.036515	0.036515	0.029814

TABLE IV
ACCURACY STATISTICS FOR LOGISTIC REGRESSION, DECISION TREE, AND K-NN, K=5 FOLDS

Altering the number of folds in k-fold cross validation, the results for k=6, 8, 10, 12, and 14 are shown in Table V.

k folds	Statistic	Log. Reg.	Decision Tree	K-NN
k=6	Mean	0.973	0.96	0.966
	Standard Dev.	0.03266	0.035	0.046
k=8	Mean	0.980	0.953	0.966
	Standard Dev.	0.039	0.043	0.039
k=10	Mean	0.98	0.953	0.966
	Standard Dev.	0.044	0.044	0.044
k=12	Mean	0.966	0.946	0.965
	Standard Dev.	0.081	0.062	0.055
k=14	Mean	0.980	0.940	0.966
	Standard Dev.	0.03	0.058	0.046

TABLE V
CROSS VALIDATION STATISTICS FOR LOGISTIC REGRESSION, DECISION TREE, AND K-NEAREST NEIGHBORS MODELS (NUMBER OF FOLDS=6, 8, 10, 12, AND 14)

After modifying the k-folds in the cross validation, it can be seen that the logistic regression model consistently has greater accuracy than the decision tree and K-NN models. However, the K-NN average accuracy scores have less variance which may indicate the K-NN model is more stable.

IV. Discussion

Through this study, we were able to confirm that a higher K or nearest neighbor value would result in smoother boundaries between the three Iris classes. We expected these results because the K value controls the sensitivity of the model and is known as the complexity parameter. The lower the K value, the more sensitive the model is. Among the decision tree, logistic regression, and K-nearest neighbor models, K-NN had the highest accuracy and smallest standard deviation values when only 5 folds were used. Seeing that nearly all the Iris setosa data points ($\frac{1}{3}$ of the whole dataset) are apparently closely grouped together, it makes sense that a model that relies on distance as one of its parameters would result in high accuracy. It is interesting to note that beyond k=5 folds, the logistic regression model outperformed both the decision tree and K-NN models; the mean accuracy of the logistic regression model was higher than the other two models at k=6, 8, 10, 12, and 14 folds. This could indicate that the K-NN model is not the most effective predictive model for this dataset as it may be overfitting the data and cannot accurately make predictions or generalizations for new data.

When the data was scaled, using Euclidean and Manhattan distance calculations resulted in higher accuracies than using the cosine calculation. The increase in accuracy using the cosine distance calculation when the data was unscaled could be attributed to the fact that it does not consider the magnitudes of feature values. Understanding the effects of the data itself on the model is just as important as understanding the parameters of the model to effectively optimize the K-NN model to the dataset at hand.

In the future, it would be beneficial to use other combinations of features (e.g., sepal length and petal length, sepal width and petal width, sepal length and petal width, etc.) to see which combination of features results in highest accuracy. Scaling the data may become much more important because the petal width values are much lower in magnitude than the other features, especially sepal length. Additionally, a future analyst could assign greater weight to specific attributes of the dataset. This would allow some features to have more significance over other features that could determine a point's classification.

References

[1] A. Kumar, "KNN Algorithm: What?When?Why?How?," Medium. Accessed: Oct. 16, 2023. [Online]. Available: https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f

[2] "Nearest Neighbors Classification," scikit-learn. Accessed: Oct. 16, 2023. [Online]. Available: https://scikit-learn/stable/auto_examples/neighbors/plot_classification.html

[3] L. Mueller and R. Hong, "Investigating Decision Trees," Sep. 18, 2023.

[4] L. Mueller and R. Hong, "Iris Classification Using Logistic Regression," Sep. 25, 2023.

[5] L. Mueller and R. Hong, "Using K-Fold Cross Validation on Decision Tree and Logistic Regression Models to Classify Iris Species," Oct. 5, 2023.