Nearest Neighbor Algorithm

Akash Perni, Sai Ganesh Pendela, Samba Sivesh Kurra, K A V Puneeth Sarma
Maryland Applied Graduate Engineering
University of Maryland
Collge Park, MD, United States
aperni@umd.edu, spendela@umd.edu, shiv1818@umd.edu, akondamu@umd.edu

***Abstract*—Nearest Neighbor Algorithm is a non-parametric model that classifies data based on the proximity to its neighbors for a classification problem. In this paper, the Nearest neighbor model's performance is compared with that of the Logistic Regression and Decision tree. Cross-validation is used to find the best k value for the Nearest Neighbor classifier and compared. The results show that Nearest Neighbors performs well compared to Decision tree in terms of accuracy but less than the Logistic regression model. In terms of generalizability over the cross-validation splits, the Nearest Neighbor model performs better. This analysis helps in learning about Nearest Neighbor Classifier.**

***Index Terms*—Iris dataset, Logistic Regression, Decision Tree, Cross-Validation.**

I. Introduction

In this paper, the Nearest Neighbor classifier [1] is trained on the Iris dataset [2] with different k values, to find the best model. Nearest Neighbor Classifier is a Machine learning model built on the assumption that neighbors that are closer together are similar. The model is then compared with others like Logistic Regression [3] and Decision tree [4]. Cross-validation [5] is also used to find the best k for the Nearest Neighbor classifier and generate other metrics like mean and standard deviation of accuracies to gain deeper insights into the performance.

As mentioned previously, Iris dataset is used for training and evaluation. A brief overview about Iris dataset is available in Table I. The objective is to predict the Dependent variable ***Class*** using the Independent variables ***Sepal Length, Sepal Width, Petal Length, Petal Width***.

Column	Variable Type	Data Type
Sepal Length	Independent	Float
Sepal Width	Independent	Float
Petal Length	Independent	Float
Petal Width	Independent	Float
Class	Dependent	String

TABLE I
IRIS DATASET

The rest of the paper is organized as follows, Section II describes the Methodology to build a Nearest Neighbor classifier. The Results section provides the Nearest Neighbor accuracy details and a comparison of the results with other models. In the Conclusion section, possible reasons for the Results obtained are explained.

II. Methodology

In this section, the steps to build a Nearest Neighbor classifier on the Iris dataset is explained. Also, the steps to also compare results with other models such as Logistic regression and Decision tree is discussed.

A. Setting up and importing necessary packages

The Machine Learning model is built using **Python 3.8.*** and **Jupyter notebook** [6]. The exact versions of them are mentioned in table II

Software	Version
Python	3.8.17
Jupyter notebook	7.0.3

TABLE II
SOFTWARE AND THEIR VERSIONS

Generally any version of Python 3.8.* should work, but the exact version should be preferred to avoid any issues.

Other than the softwares mentioned, various other packages of Python are needed to ease the development process of the Nearest Neighbor classifier. All the packages that are needed are mentioned in the table III.

Module	Version
Numpy [7]	1.24.4
Scikit-Learn [8]	1.3.0
Pandas [9]	2.0.3
Matplotlib [10]	3.7.2

TABLE III
MODULES USED AND THEIR VERSIONS

All the above mentioned packages can be imported using the Python syntax defined.

B. Loading the Iris dataset

The Iris dataset can be downloaded publicly in the form of CSV from various sources such as the official website [2], Kaggle [11] etc. After downloading, the dataset can be loaded into Jupyter notebook as a DataFrame [12] with the *read_csv()* [13] function of Pandas. The function takes the *file path* as input and returns a DataFrame object. Loading the data as a DataFrame is helpful as it lets us access the data column wise as opposed to the traditional row wise approach which most of the softwares do. This lets us separate the columns easily which will be useful during training with the train data and class label to be predicted.

C. Data partitioning

Data partitioning is useful in understanding the generalizability of the model. To perform data partitioning, the *train_test_split()* [14] function from Scikit-Learn is used. The function takes *data*, *ground truth* and *test_size* as input to return the *X_train*, *X_test*, *y_train* and *y_test*.

Some characteristics of the split dataset after data partitioning are explained below:

X_train variable stores the data that will be used for training.

X_test variable contains the data that will be used for testing.

y_train variable has the ground truths required to train the model.

y_test variable has the ground truths needed to evaluate the trained model.

The test split is set to *30%* as that split provides enough samples for both training and testing. In case of the Iris dataset, this split gives *50* samples for testing and *100* samples for training.

Once the data is split using these variables, pre-processing can be performed on the data followed by training and evaluation of the machine learning models.

D. Feature Scaling

Feature scaling is important in K-Nearest Neighbors (KNN) because KNN relies on the calculation of distances between data points to make predictions. If features have different scales, those with larger scales can dominate the distance metric, leading to incorrect or biased results. So here, *scaler.fit_transform()* [15], which standardizes the features, making them have a similar scale and *scaler.transform()* [15] which scales the testing data using the same mean and standard deviation values that were calculated from the training data are used.

E. Building models

Scikit-Learn provides various classes and functions to build Machine learning models of which Nearest Neighbors is one. The Nearest Neighbors classifier can be built using the *KNeighborsClassifier()* [16] class which is available in the *sklearn.neighbors* submodule. The class takes the *n_neighbors* which represents the number of neighbors that the model needs to compare with during predictions and *weights* which when set to the value *distance* enables weighted classification which might improve the performance in some cases.

$$w_i = 1/d_i^2 \tag{1}$$

Equation 1 describes a way of calculating the weights where each scores is calculated as the inverse of the distance with the neighbors squared. This emphasizes the usage of distance in finding out the final class which controls the effect of outliers.

Decision tree and Logistic regression models are built using the methodology described in the papers **Decision Tree Classifier** [17] and **Understanding Logistic Regression** [18] respectively.

F. Choosing the best Nearest Neighbor Classifier

The best k value for the Nearest Neighbor Classifier is calculated through the technique of Cross-validation. Different k values are chosen and the cross-validation scores are calculated for every value and compared to find the best one. *cross_val_score()* [19] function from Scikit-Learn is used to calculate the scores. This function takes *model* for which the scores need to be calculated, *train data*, *ground truths*, *cv* the number of splits for the dataset as input to return the accuracies. The number of cross-validation folds is set to *10* for consistency among all values and the mean and Standard deviation of the accuracies is extracted to find the best one.

G. Comparing the models

Cross-validation is used to compare the models. The models are tested by dividing the dataset into *10* splits and using cross_val_score() [19] to get the accuracy on each split. The mean and standard deviation of the accuracies is extracted. The mean helps in understanding the average performance of the model and the standard deviation helps in understanding the individual performance on each split of the dataset. In general, a higher mean and lower standard deviation is expected signifying the generalizability of the model on the dataset.

RESULTS

A. Best Nearest Neighbor Classifier

The best Nearest Neighbor Classifier is found by comparing the cross-validation results for every *k* value. Table IV shows the Mean and the standard deviation of the accuracies for 10 split cross validation (The accuracies are calculated in the range of (0-1).

k value	Mean Accuracy	Standard Deviation
3	0.934	0.0624
5	0.943	0.0648
7	0.952	0.0658
9	0.952	0.0658
11	0.923	0.0742

TABLE IV
MEAN ACCURACY AND STANDARD DEVIATION OF ACCURACY SCORES FOR DIFFERENT K VALUES FOR REGULAR KNN

From table IV it is evident that, the best value of k is **7, 9**.

To also understand the test performance with **k=7**, the model is tested on the *70-30* train-test split of Iris dataset. Figure 1 shows the Confusion matrix potraying the number of correctly predicted classes vs the true classes. As seen from the image, the model is performing very well with every class being correctly predicted with an accuracy of 100%.

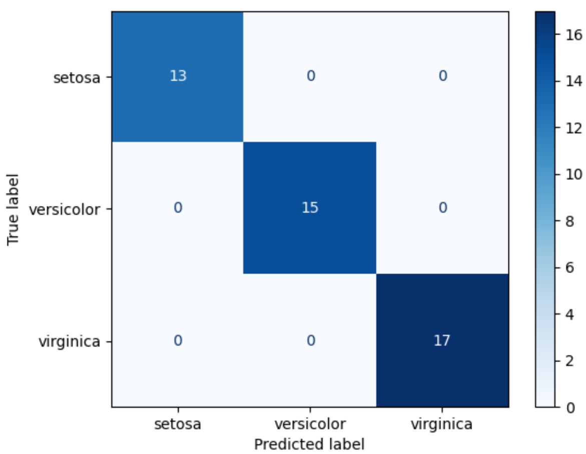

Fig. 1. Confusion matrix results of regular KNN on test data of Iris.

Other than the regular Nearest Neighbor Classifier, the weighted Nearest Neighbor classifier is also used. The scores for different k values for the same are available in table V

k value	Mean Accuracy	Standard Deviation
3	0.944	0.046
5	0.944	0.0614
7	0.943	0.0764
9	0.952	0.0658
11	0.953	0.0742

TABLE V
MEAN ACCURACY AND STANDARD DEVIATION OF ACCURACY SCORES FOR DIFFERENT K VALUES FOR WEIGHTED KNN

From table V, it can be seen that using weighted scores has significantly modified the performance for the weighted Nearest Neighbor Classifier. For regular Nearest Neighbor model, average accuracy increased until k values of 7,9 and later decreased for k=11. But, here the using weighted k value increased the average performance for 11 as well highlighting the improved performance.

The performance on the test data after 70-30 split is available in figure 2. It can be seen from the image that the model with **k=11** has performed well on the test split of the Iris dataset.

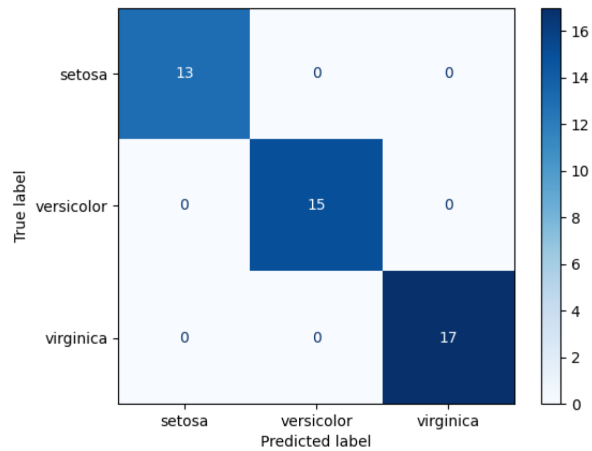

Fig. 2. Confusion matrix results for Weighted KNN on test data of Iris.

B. Comparison results with other models

The Nearest Neighbor Classifier is compared with other models like *Decision tree with entropy*, *Decision tree with gini*, *Logistic regression*.

The comparison is done in 2 ways. First, using *70-30* train-test split and cross validation.

Table VI shows the accuracy scores for all models on the 30% test data of the Iris dataset.

Model	Accuracy
Logistic Regression	0.9556
Decision Tree(Entropy)	0.9556
Decision Tree(Gini)	0.9556
Nearest Neighbours (k=7)	1.0
Weighted Nearest Neighbours (k=11)	1.0

TABLE VI
ACCURACY SCORE OF MODELS ON IRIS DATASET

The table VI shows that the Nearest Neighbor and the weighted Nearest Neighbor models are performing very well compared to other models on the test dataset.

Table VII shows the Mean and Standard deviation of the cross validation scores of the 4 models on the Iris dataset. The number of splits is set to *10* for all models.

Model	Mean Accuracy	Standard Deviation
Logistic Regression	0.96	0.0442
Decision Tree(Entropy)	0.94	0.0628
Decision Tree(Gini)	0.94	0.0628
Nearest Neighbors (k=7)	0.94	0.0467
Weighted Nearest Neighbors (k=11)	0.953	0.0427

TABLE VII
MEAN ACCURACY AND STANDARD DEVIATION OF ACCURACY SCORES OF MODELS ON IRIS DATASET

The table results show that Weighted Nearest Neighbor performs well in terms of accuracy on the Iris dataset, but less than that of the Logistic regression model with it being the highest. In terms of the standard deviation, Weighted Nearest Neighbors has the least deviation showing good performance on all the splits when compared to other models.

As the dataset can be split very nicely, the Nearest Neighbor algorithm gives very good performance further making sense of the model's results.

CONCLUSION

In this paper, K Nearest Neighbours is applied on the Iris dataset and compared with other models like Logistic regression and Decision tree. Cross validation is also used to understand the generalizability of the model. The results show that Nearest Neighbor model performs better than Decision tree in terms of the cross-validation accuracy, but less than Logistic regression. The model performs well in terms of the accuracy for each split showing good generalizability. The model also performs the best on the *70-30* split test data compared to all other models. This might be due to the good split between each class of the Iris dataset. This analysis helpful in understanding the Nearest Neighbors model's advantage in exploiting data that has a good split for every class. In future, another dataset containing outliers can be used to understand how Nearest Neighbor performs on them.

REFERENCES

[1] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[2] Fisher,R. A.. (1988). Iris. UCI Machine Learning Repository. https://doi.org/10.24432/C56C76.
[3] https://www.geeksforgeeks.org/understanding-logistic-regression/
[4] https://www.ibm.com/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a ,internal%20nodes%20and%20leaf%20nodes.
[5] https://towardsdatascience.com/what-is-cross-validation-60c01f9d9e75
[6] https://jupyter.org/try-jupyter/retro/notebooks/?path=notebooks/Intro.ipynb
[7] https://numpy.org/doc/stable/
[8] https://scikit-learn.org/stable/tutorial/index.html
[9] https://pandas.pydata.org/docs/getting_started/index.html
[10] https://matplotlib.org/stable/tutorials/pyplot.html
[11] https://www.kaggle.com/datasets/uciml/iris
[12] https://www.databricks.com/glossary/what-are-dataframes#:~:text=A%20DataFrame%20is %20a%20data,storing%20and%20working%20with%20data.
[13] https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html
[14] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[15] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
[16] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[17] Decision tree Classifier
[18] Understanding Logistic regression
[19] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html