Classification Metrics

Sai Ganesh Pendela
Maryland Applied Graduate Engineering
University of Maryland
Collge Park, MD, United States
spendela@umd.edu

***Abstract*—This paper presents a analysis on the performance of three machine learning models, the Decision Tree, Logistic Regression, and k-Nearest Neighbor, applied to the Iris dataset. We construct these models to predict iris species and evaluate their performance using confusion matrix, classification report, accuracy, precision and recall. The results highlight key differences among the models, and the results show that logistic regression and Nearest Neighbours classifiers produce the same accuracy for the Iris dataset.**

***Index Terms*—Iris dataset, Logistic Regression, Decision Tree, K-Nearest Neighbors, Classification report, confusion matrix, accuracy and recall.**

I. Introduction

In this paper, the Iris dataset [1] is used to train and test three different classifiers, they are, Decision Tree classifier [2], Logistic Regression classifier [3], and the Nearest Neighbor classifier [4]. Subsequently, the results are compared.

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 Decision tree, Logistic Regression and Nearest Neighbor classifier, and it also describes how to calculate various classification metrics [5]. The Results section provides comparison of the results of the models. In the Conclusion section, possible reasons for the Results obtained are explained.

II. Methodology

In this section, the steps to build Decision tree, Logistic regression and K-Nearest Neighbors classifiers on the Iris dataset is explained. Also, the functions used to calculate the metrics is also 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 is readily available for download in CSV format from several public sources like Kaggle. Once downloaded, you can import the dataset into a Jupyter notebook as a DataFrame [11] using the Pandas *read_csv()* [12] function. This function accepts the *file path* as an input and returns a DataFrame object. Utilizing a DataFrame to load the data is advantageous because it enables us to access the dataset's columns individually, unlike the conventional row-wise approach commonly used by most software. The loading of the iris dataset is same as mentioned in the previuos paper **Decision Tree Classifier** [14].

C. Data partitioning

Data partitioning is done to train and test models, prevent overfitting, assess generalization, and facilitate hyperparameter tuning and model selection. To perform data partitioning, the *train_test_split()* [13] 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*.

Data partitioning results in four variables:

X train (for training data), X test (for testing data), y train (ground truths for training), and y test (ground truths for evaluation).

A *30%* test split is chosen, offering a suitable number of samples for both training and testing, such as *100* training samples and *50* testing samples in the case of the Iris dataset.

Following the data split using these variables, you can conduct data preprocessing, machine learning model training, and model evaluation. The above process is same as mentioned in the previous paper **Decision Tree Classifier** [14].

D. Building models

Scikit-Learn provides various classes and functions to build Machine learning models. Decision tree, Logistic regression models and K- Nearest Neighbors are built using the methodology described in the papers **Decision Tree Classifier** [14], **Understanding Logistic Regression** [15], **Nearest Neighbor Algorithm** [16] respectively.

E. Calculating the metrics

The *sklearn.metrics* is a module in the scikit-learn (sklearn) library, which is a popular machine learning library in Python. This module contains various functions and classes for evaluating the performance of machine learning models. The ***confusion_matrix()***, ***accuracy_score()***, ***classification_report()***,***recall_score()***, ***precision_score()***, ***ConfusionMatrixDisplay()***, these functions should be imported to calculate the accuracy, recall, and to view the confusion matrix and classification report.

The ***confusion_matrix()*** [17] has two required arguments, they are, the true target values (ground truth) from your dataset, and the predicted target values generated by a machine learning model.

The ***classification_report()*** [18], typically generated provides a comprehensive summary of various performance metrics for evaluating the model's predictions takes several arguments, but the minimum required arguments are the true target values and the predicted target values.

The ***accuracy_score()*** [19] is calculated by dividing the number of correctly predicted instances by the total number of instances in the dataset, it takes two arguments as input the true target values and the predicted target values.

$$Accuracy = \frac{TotalNumberofPredictions}{NumberofCorrectPredictions} \quad (1)$$

The ***recall_score()*** [20] measures the ability of a model to correctly identify positive instances out of all actual positive instances. The function has the true target values or labels and the predicted values as arguments. and the *average="macro"* option computes the recall for each class and then takes the unweighted average (mean) of the recall scores across all classes.

$$Recall = \frac{TruePositivies}{TruePositivies + FalseNegativies} \quad (2)$$

Similarly for the ***precision_score()*** [21] function, the *average="macro"* option calculates the precision for each class individually and then takes the unweighted average of all class precisions.

$$Precision = \frac{TruePositivies}{TruePositivies + FalsePositivies} \quad (3)$$

Using the all the functions above the accuracy, recall, are calculated and the confusion matrix, confusion report is obtained. The code snippet for calculating the metrics of decision tree is shown below, in the same way all the metrics are calculated for remaining models.

```
# Confusion Matrix and Metrics for Decision Tree
dt_confusion_matrix = confusion_matrix(y_test, dt_predictions)
dt_classification_rep = classification_report(y_test, dt_predictions)
dt_accuracy = accuracy_score(y_test, dt_predictions)
dt_recall = recall_score(y_test, dt_predictions, average='macro')
dt_precision = precision_score(y_test, dt_predictions, average='macro')
```

Fig. 1. Code snippet for calculating all the metrics of decision tree classifier.

RESULTS

A. Accuracy

Out of the all three models, K-Nearest Neighbors and Logistic Regression has the same and highest accuracy of 0.9777777777777777, and the decision tree with criterion as entropy and gini gave the same accuracy of 0.9555555555555556. The Iris dataset is small has distinct classes, so the accuracy is high for logistic regression and KNN.

Model	Accuracy
Decision Tree with entropy	0.9555555555555556
Decision Tree with gini	0.9555555555555556
Logistic Regression	0.9777777777777777
K-Nearest Neighbors	0.9777777777777777

TABLE IV
ACCURACY OF ALL THE MODEL'S

B. Recall

Out of the all three models, K-Nearest Neighbors and Logistic Regression has the same and highest recall of 0.9777777777777779. and the decision tree with criterion as entropy and gini gave the same recall of 0.9581699346405229.

Model	Recall
Decision Tree with entropy	0.9581699346405229
Decision Tree with gini	0.9581699346405229
Logistic Regression	0.9777777777777779
K-Nearest Neighbors	0.9777777777777779

TABLE V
RECALL OF ALL THE MODEL'S

C. Precision

Out of the all three models, K-Nearest Neighbors and Logistic Regression has the same and highest precision of 0.9814814814814815. and the decision tree with criterion as precision and gini gave the same recall of 0.9581699346405229.

Model	Precision
Decision Tree with entropy	0.9581699346405229
Decision Tree with gini	0.9581699346405229
Logistic Regression	0.9814814814814815
K-Nearest Neighbours	0.9814814814814815

TABLE VI
PRECISION OF ALL THE MODEL'S

D. Confusion Matrix

Here, for the iris dataset, confusion matrix is a 3x3 matrix with three rows and three columns, with each row and column representing a specific class. The confusion matrix helps assess the model's accuracy, precision, recall, and F1-score. The confusion matrix for the decision tree with criterion as entropy and gini index is shown below.

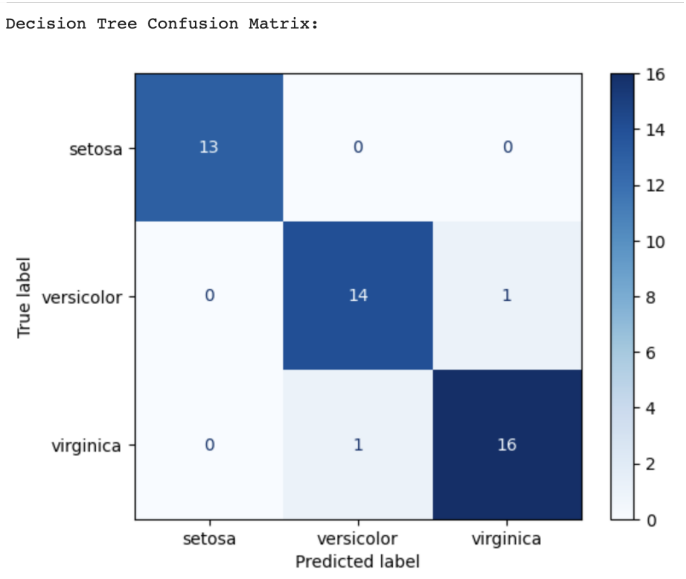

Fig. 2. Confusion matrix with criterion as gini and entropy.

The confusion matrix for the logistic regression is shown below.

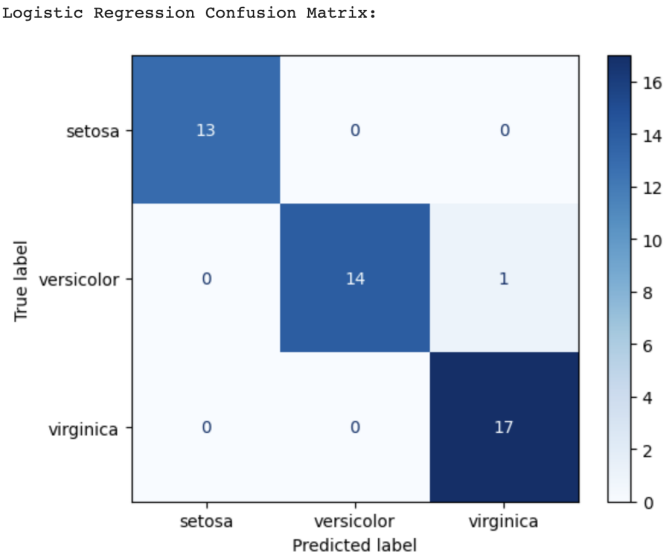

Fig. 3. Confusion matrix for Logistic regression.

The confusion matrix for the K Nearest Neighbors is shown below. The KNN classifies all the classes very well.

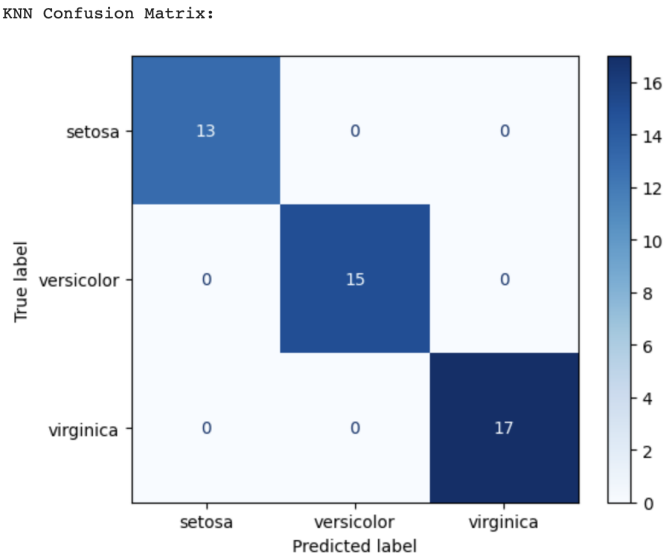

Fig. 4. Confusion matrix for KNN.

E. Classification Report

The classification report provides the precision, recall, f1-score, support, accuracy, average, weighted average for the model. The classification report provides all the above metrics for each class. The classification report for the decision tree is below.

Decision Tree Classification Report:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	13
versicolor	0.93	0.93	0.93	15
virginica	0.94	0.94	0.94	17
accuracy			0.96	45
macro avg	0.96	0.96	0.96	45
weighted avg	0.96	0.96	0.96	45

Fig. 5. Classification report for decision tree.

The classification report for the logistic regression is shown below.

Logistic Regression Classification Report:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	13
versicolor	1.00	0.93	0.97	15
virginica	0.94	1.00	0.97	17
accuracy			0.98	45
macro avg	0.98	0.98	0.98	45
weighted avg	0.98	0.98	0.98	45

Fig. 6. Classification report for logistic regression .

The classification report for the K Nearest Neighbors is shown below.

KNN Classification Report:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	13
versicolor	1.00	0.93	0.97	15
virginica	0.94	1.00	0.97	17
accuracy			0.98	45
macro avg	0.98	0.98	0.98	45
weighted avg	0.98	0.98	0.98	45

Fig. 7. Classification report for KNN.

CONCLUSION

The classification metrics are used to identify which model is performing well on the particular dataset. The K-Nearest Neighbors and the Logistic Regression works good for the iris dataset, producing high accuracy score, high recall and high precision. The decision tree also performs well, but it produces accuracy less than the Logistic regression and the K Nearest Neighbor model, beacause, the Iris dataset is relatively small, well-balanced, and its classes are distinct, making it a suitable scenario for KNN and logistic regression which is the reason why they have higher accuracy than decision tree.

REFERENCES

[1] https://archive.ics.uci.edu/dataset/53/iris
[2] https://en.wikipedia.org/wiki/Decision_Trees.
[3] https://www.geeksforgeeks.org/understanding-logistic-regression/
[4] https://www.ibm.com/topics/knn
[5] https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide
[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.geeksforgeeks.org/python-pandas-dataframe/
[12] https://www.geeksforgeeks.org/python-read-csv-using-pandas-read_csv/.
[13] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[14] Decision Tree Classifier Paper
[15] Understanding Logistic regression
[16] Nearest Neighbor Algorithm
[17] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion _matrix.html
[18] https://scikitlearn.org/stable/modules/generated/sklearn.metrics.classification_report.html
[19] https://scikitlearn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
[20] https://scikitlearn.org/stable/modules/generated/sklearn.metrics.recall_score.html
[21] https://scikitlearn.org/stable/modules/generated/sklearn.metrics.precision_score.html