Naive Bayes and Linear Discriminant Analysis Classifiers

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

***Abstract*—This paper focuses on building the Naive Bayes Classifier and Linear Discriminant Analysis Classifier, and evaluating the performance of five machine learning models, the Decision Tree, Logistic Regression, k-Nearest Neighbor, Naive Bayes, Linear Discriminant analysis 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 the logistic regression and Linear Discriminant Analysis classifiers produce the higher accuracy and perform well on the the Iris dataset.**

***Index Terms*—Iris dataset, Logistic Regression, Naive Bayes, Linear Discriminant Analysis, 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 two different classifiers, they are, Naive Bayes classifier [2], and the Linear Discriminant Analysis classifier [3]. 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 Naive Bayes, Linear Discriminant Analysis 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 Naive Bayes, Linear Discriminant Analysis 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 **Google Colab** [6]. The exact versions of them are mentioned in table II

Software	Version
Python	3.8.17
Google Colab	Colab Pro

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 Naive Bayes, Linear Discriminant Analysis Classifiers. 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 [21]provides various classes and functions to build Machine learning models.

To build the Naive Bayes model, the GaussianNB is imported form the sklearn.naive_bayes module, the GaussianNB [23] refers to the Gaussian Naive Bayes classifier, it a variant of the Naive Bayes algorithm that assumes the likelihood of the features is Gaussian (normal distribution). After importing the naive bayes classifier, an instance of the classifier is built, after that the model is fit to the training data. After training the model, predicitions are made on the test data.

Similarly, the LinearDiscriminantAnalysis (LDA) [24] is a classifier under the sklearn.discriminant_analysis module. LDA is a dimensionality reduction technique commonly used in the field of pattern recognition and machine learning. It's often used as a classification algorithm, especially in scenarios where the classes are well-separated and follow a normal distribution.

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.

```
# Creating and training Naïve Bayes classifier
from sklearn.naive_bayes import GaussianNB
NB_model = GaussianNB()
NB_model.fit(X_train, y_train)
model_statistics(NB_model, "Naive_Bayes")
```

Fig. 1. Code to build the Naive Bayes Classifier.

```
# Creating and training Linear Discriminant Analysis (LDA) classifier

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
LDA_model = LinearDiscriminantAnalysis()
LDA_model.fit(X_train, y_train)
model_statistics(LDA_model, "Linear_Discriminant_Analysis")
```

Fig. 2. Code to build the Linear Discriminant Classifier.

E. Evaluating the Performance

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()*** **[17],** ***accuracy_score()*** **[19],** ***classification_report()*** **[18],*recall_score()*** **[20],** ***F1 Score()*** **[22],** ***ConfusionMatrixDisplay()***, these functions should be imported to calculate the accuracy, recall, and to view the confusion matrix and classification report.

All the description and how to calculate these metrics is given in the previous paper **Classification Metrics** [25].

RESULTS

A. Accuracy

Out of the all five models, K-Nearest Neighbors and Linear Discriminant Analysis models has the same and highest accuracy of 1.0, and the Naives Bayes, Logistic Regression has the accuracy of 0.9777777777777777, the decision tree with criterion as entropy and gini has the accuracy of 0.9555555555555556. The Iris dataset is small has distinct classes, so the accuracy is high for LDA and KNN.

Model	Accuracy
Decision Tree with entropy	0.9555555555555556
Decision Tree with gini	0.9555555555555556
Logistic Regression	0.9777777777777777
K-Nearest Neighbors	1.0
Naive Bayes	0.9777777777777777
Linear Discriminant Analysis	1.0

TABLE IV
ACCURACY OF ALL THE MODEL'S

B. Recall

Out of the all five models, K-Nearest Neighbors and Linear Discriminant Analysis models has the same and highest Recall of 1.0, and the Naives Bayes, Logistic Regression has the recall of 0.9777777777777779, the decision tree with criterion as entropy and gini has the recall of 0.9581699346405229. The

Iris dataset is small has distinct classes, so the recall is high for LDA and KNN.

Model	Recall
Decision Tree with entropy	0.9581699346405229
Decision Tree with gini	0.9581699346405229
Logistic Regression	0.9777777777777779
K-Nearest Neighbors	1.0
Naive Bayes	0.9777777777777779
Linear Discriminant Analysis	1.0

TABLE V
RECALL OF ALL THE MODEL'S

C. F1 Score

Out of the all five models, K-Nearest Neighbors and Linear Discriminant Analysis models has the same and highest F1 Score of 1.0, and the Naives Bayes, Logistic Regression has the F1 Score of 0.9789819376026273, the decision tree with criterion as entropy and gini has the recall of 0.9581699346405229. The Iris dataset is small has distinct classes, so the F1 Score is high for LDA and KNN.

Model	F1 Score
Decision Tree with entropy	0.9581699346405229
Decision Tree with gini	0.9581699346405229
Logistic Regression	0.9789819376026273
K-Nearest Neighbors	1.0
Naive Bayes	0.9789819376026273
Linear Discriminant Analysis	1.0

TABLE VI
F1 SCORE 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 Naive Bayes is given below.

Fig. 3. Confusion matrix for Naive Bayes.

The confusion matrix for the linear Discriminant Analysis Classifier is shown below.

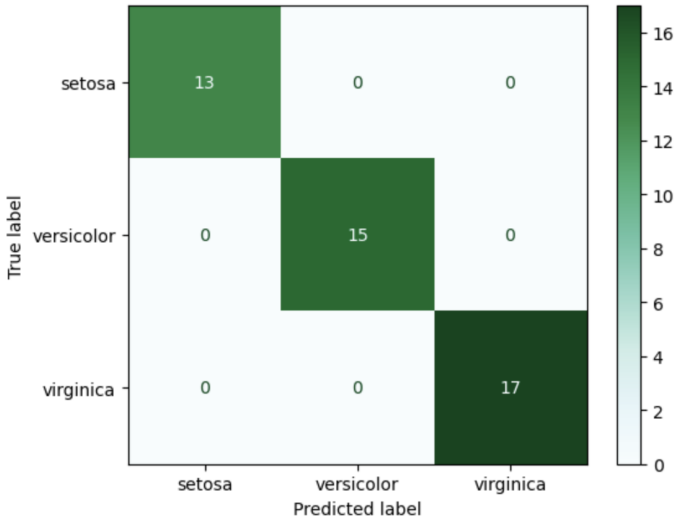

Fig. 4. Confusion matrix for Logistic regression.

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 Naive Bayes is below.

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. 5. Classification report for Naive Bayes.

The classification report for the linear Discriminant analysis is shown below.

Classification Report:

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

Fig. 6. Classification report for linear Discriminant analysis.

CONCLUSION

In this study, we applied various machine learning models, including Naïve Bayes, Linear Discriminant Analysis (LDA), Logistic Regression, Decision Tree, and k-Nearest

Neighbors (KNN), on the well-known Iris dataset. Notably, LDA, Logistic Regression, and KNN demonstrated exceptional performance, achieving perfect accuracy, recall, and F1 scores of **1.0**. Naïve Bayes also exhibited high accuracy and F1 score at **0.978**, while Decision Tree slightly lagged behind with an accuracy of **0.956**. These results suggest that LDA, Logistic Regression, and KNN are particularly effective for the Iris dataset, showcasing their robust classification capabilities. and Linear Discriminant Analysis (LDA) performed well due to its ability to find a linear combination of features that maximizes class separability, which is advantageous in the Iris dataset. k-Nearest Neighbors (KNN) excelled because of the dataset's distinct class clusters, allowing KNN to effectively capture local patterns and classify instances based on their proximity to neighbors in the feature space.

REFERENCES

[1] https://archive.ics.uci.edu/dataset/53/iris
[2] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[3] https://www.knowledgehut.com/blog/data-science/linear-discriminant-analysis-for-machine-learning
[4] https://www.ibm.com/topics/knn
[5] https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide
[6] https://colab.research.google.com/?utm_source=scs-index
[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://scikit-learn.org/stable/
[22] https://www.v7labs.com/blog/f1-score-guide
[23] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
[24] https://machinelearningmastery.com/linear-discriminant-analysis-with-python/
[25] Classification Metrics