Leaf Data Analysis Using Supervised Learning Techniques

Supervisor: *Mrs. W. M. L. N. Wanninayake, Department of Computing and Information Systems, Wayamba University of Sri Lanka*.

1st Kanisha Liyanage
Department of Computing and Information Systems
Faculty of Applied Sciences
Wayamba University of Sri Lanka
Colombo, Sri Lanka
Email: kanishaliyanage@ieee.org

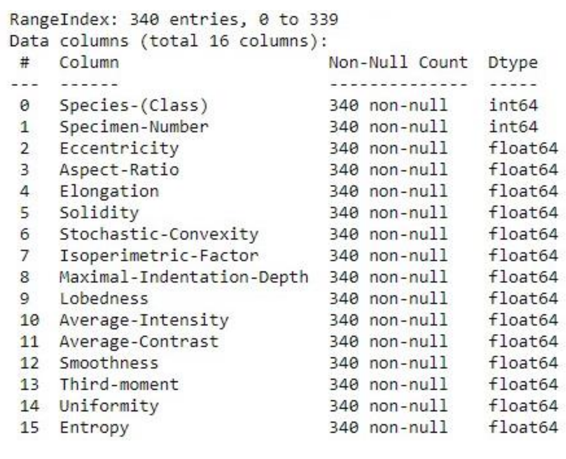
***Abstract*—This report presents an analysis of the UCI Leaf Dataset using supervised learning techniques. The goal of the analysis is to accurately classify the different species of leaves represented in the dataset. To achieve this goal, various classifiers such as decision trees, K-Nearest Neighbors (KNN), Linear Discriminant Analysis, Logistic Regression, Naïve Bayes, and Support Vector Machine (SVM) are trained and tested on the data. Also used Artificial Neural Network (ANN) approach as a supervised deep learning technique. The results of the analysis show that the Linear Discriminant Analysis classifier outperforms the other classifiers, achieving average accuracy of 85%.**

Keywords—accuracy, ANN, classification, dataset, leaf, neural networks, species, supervised learning

I. Introduction

The identification and classification of plant species is an important task in the fields of biology and agriculture. Accurate classification can help in the understanding of plant ecology, the management of ecosystems, and the breeding of crops. In recent years, machine learning techniques have been applied to the problem of plant classification, with the goal of automating the process and increasing its accuracy.

One widely studied dataset in this field is the UCI Leaf Data Set, which contains morphological data of leaves belonging to different species. This leaf dataset contains 30 different classes (species) and a total of 340 leaves data. In this research paper, I applied various supervised learning techniques to the UCI Leaf Data Set with the goal of accurately classifying the different species of leaves. I compare the performance of different classifiers and identify the one that performs the best on the dataset. Finally, I build an artificial neural network to train the dataset.



```
RangeIndex: 340 entries, 0 to 339
Data columns (total 16 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Species-(Class)          340 non-null    int64
 1   Specimen-Number          340 non-null    int64
 2   Eccentricity             340 non-null    float64
 3   Aspect-Ratio             340 non-null    float64
 4   Elongation               340 non-null    float64
 5   Solidity                 340 non-null    float64
 6   Stochastic-Convexity     340 non-null    float64
 7   Isoperimetric-Factor     340 non-null    float64
 8   Maximal-Indentation-Depth 340 non-null    float64
 9   Lobedness                340 non-null    float64
 10  Average-Intensity        340 non-null    float64
 11  Average-Contrast         340 non-null    float64
 12  Smoothness               340 non-null    float64
 13  Third-moment             340 non-null    float64
 14  Uniformity               340 non-null    float64
 15  Entropy                  340 non-null    float64
```

Fig. 1. Leaf dataset information

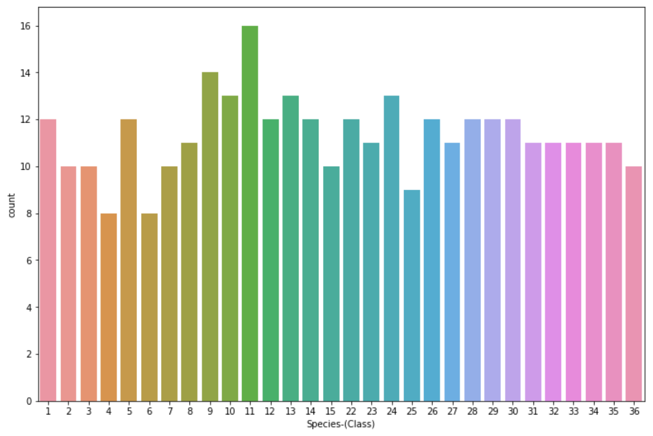


Fig. 2. Data distribution

II. Ease of Use

UCI Leaf Data Analysis Using supervised learning would likely focus on the simplicity and user-friendliness of the machine learning techniques applied to the UCI Leaf dataset. The paper might compare different supervised learning algorithms and evaluate how easy they are to implement and use for the purpose of leaves classification. The paper might also discuss the importance of making machine learning and deep learning techniques accessible and easy to use for researchers and practitioners. Overall, the paper would aim to demonstrate that supervised learning methods can be applied effectively to the UCI Leaf Data Set with a minimal amount of effort and without requiring advanced technical skills.

III. Steps of the Analysis Process

First, I identified the problem and decided to use different classification techniques and different ANN models to solve the same problem. Then I analysed the dataset contains csv file and abstracted what are the features (Independent variables also known as the predictors) and the classes (Dependent variable also known as Labels) of the dataset. Then I named the columns of the table. After done the data preparation part, I started to work on data pre-processing part. In the pre-processing part I have done several techniques to find the missing values and filling those missing values, encoding the data and split the data into train and test set. The test size and train size I have used to split to dataset are 20% and 80% respectively. After the done the data splitting, I did the feature scaling to standardize the range of features of the dataset because it can help to improve the performance and interpretability of the model.

IV. Techniques and Technlogies

A. *Classification*

Classification is a machine learning task in which an algorithm is trained to predict a class label for a given input. The algorithm is given a set of labeled training data and learns to assign a class label to new, unseen examples.

In this leaf classification I have used below classification techniques,

1) *Decision Tree* Classification
2) *K-Nearest Neighbors (KNN)*
3) *Linear Discriminant Analysis (LDA)*
4) *Logistic Regression*
5) *Naïve Bayes (NB)*
6) *Support Vector Machine (SVM)*

B. *Artificial Neural Networks (ANNs)*

Artificial neural networks (ANNs) are a type of machine learning model that are inspired by the structure and function of the human brain. They are composed of layers of interconnected "neurons," which process and transmit information. ANNs are able to learn patterns and relationships in data and can be used for a variety of tasks. In this problem, I used ANN for the classification task and I trained different models and adjusted the hyperparameters in the model to get better performance from the network.

C. *Technologies*

The main technology that I used for this analysis is supervised learning under the machine learning. Supervised learning is a type of machine learning in which an algorithm is trained on labelled training data. The training data consists of a set of input examples and corresponding desired output labels, and the goal of the supervised learning algorithm is to make predictions about the output labels for new, unseen input examples.

Other related technologies and tools that are used for this project, Python as the programming language, Scikit-Learn as the library for the machine learning, Keras, Tensorflow as the libraries for the deep learning, Pandas as the data manipulation library, Matplotlib and Seaborn as the data visualization libraries, Numpy as the numerical computing library, and other machine learning related libraries. I have used as the package management and development distribution as Anaconda Distribution. Finally, for live coding IDE, I mainly used Google Colab, and alternatively used Jupyter Notebook.

I used the latest versions of the above-mentioned technologies, tools, and software for this project.

V. Equations and Parameters

I have used python libraries like Scikit-Learn as I mentioned above techniques and technologies section. The Scikit-Learn is an API which provides a range of supervised and unsupervised learning algorithms in Python. So, the importing equations, and doing the calculations are done by the libraries. Also, the parameters can be managed easily, by importing only the functions.

A. *Equations*

here below are some of the equations used in the backend,

1) *Standardisation*

$$x_{stand} = \frac{x - mean(x)}{standard\ deviation(x)}$$

2) *Naïve Bayes*

$$P(c|x) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

3) *Precision*

$$Precision = \frac{True\ Positives}{(True\ Positives + False\ Positives)}$$

4) *F-measure*

$$F - \text{measure} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

5) *Rectified Linear Unit (ReLU)*

$$f(x) = max(0, x)$$

B. *Parameters*

The parameters I used to obtain the best classifier performance when training with different classifiers are the following,

a) Decision Tree Classification

In this classifier first, I used default parameters such as “Gini” for criterion, “best” for splitter, “none” for random_state, etc. After that, I used “entropy” as the criterion, “42” as the random state, and others as default.

b) K-Nearest Neighbors (KNN) Classifier

In KNN I used default parameters in the first experiment. Then I used “10” as the n_neighbors, “distance” as the weights, “p = 1” for manhattan distance with the “minkowski” metric, and when others remain as default values.

c) Linear Discriminant Analysis (LDA)

In the LDA classifier I trained the dataset with default parameters.

d) Logistic Regression

Logistic Regression classifier trained with the it’s default parameters.

e) Naive Bayes (NB)

For this classifier I used only default parameters.

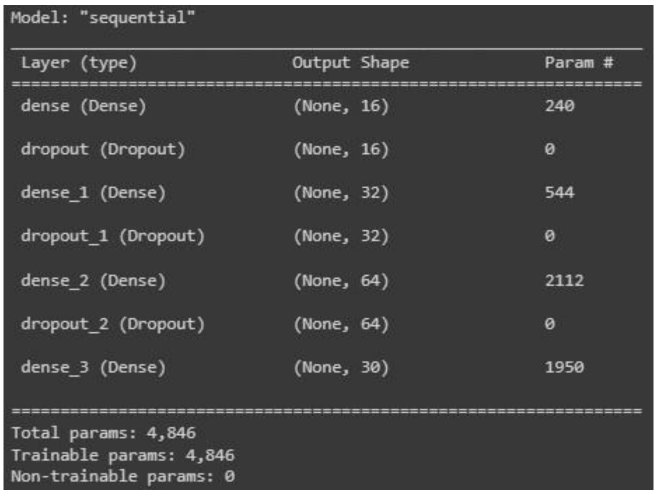
f) Support Vector Machine (SVM)

In SVM classifier, I trained the dataset using the default parameters and then trained with the custom parameters by changing the kernel to “poly”.

g) Artificial Neural Network (ANN)

In this ANN model, I tested with different hyperparameters to achieve the best accuracy from the model, and for the final model I used the following parameters,

- The number of units (neurons) in each layer: 16 units in the first hidden layer, 32 units in the second hidden layer, and 64 units in the third hidden layer.
- The type of activation function used in each layer: ReLU activation in the first three hidden layers and softmax activation in the output layer.
- The dropout rates: 0.15 in the first hidden layer, 0.1 in the second hidden layer, and 0.25 in the third hidden layer. Dropout is a regularization technique that helps to prevent overfitting by randomly dropping out (setting to zero) a certain proportion of the units in the network during training.
- The input dimension: 14, which corresponds to the number of independent variables in the dataset.
- The number of units in the output layer: 30, which corresponds to the number of classes to get predictions for.
- The batch size: 16, which determines the number of training examples to use in each iteration of the optimization algorithm.
- The number of epochs: 500, which determines the number of times the optimization algorithm will go through the entire training dataset.
- The validation data: a tuple of the testing dataset (X_test) and the true labels (y_test), which will be used to evaluate the model's performance after each epoch.


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                240

 dropout (Dropout)           (None, 16)                0

 dense_1 (Dense)             (None, 32)                544

 dropout_1 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 64)                2112

 dropout_2 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 30)                1950

=================================================================
Total params: 4,846
Trainable params: 4,846
Non-trainable params: 0
```

Fig. 3. Model summary

VI. RESULTS ANALYSIS

For this leaf dataset, I trained models using various parameters for the classifiers mentioned in the section V. Following table (TABLE I) shows the performance report of each classifier.

TABLE I. CLASSIFIER PERFORMANCE REPORT

Classifier	***Parameters***	***Avg. Precision***	***Avg. Recall***	***Avg. F-measure***	***Accuracy***
Decision Tree	Default	65.69%	60.29%	60.60%	60.29%
	criterion = "entropy" random_state = 42	65.69%	60.29%	60.60%	60.29%
KNN	Default	78.16%	70.59%	71.41%	70.59%
	n_neighbors = 10 weights = "distance" p = 1	77.86%	70.59%	70.96%	70.59%
LDA	Default	89.83%	85.29%	85.12%	85.29%
Logistic Regression	Default	86.81%	80.88%	81.18%	80.88%
NB	Default	80.76%	72.06%	72.79%	72.06%
SVM	Default	81.05%	75.00%	74.12%	75.00%
	Kernel = "poly"	51.33%	48.53%	46.56%	48.53%

Finally for this leaf dataset, I trained many ANN models with different hyperparameters as I mentioned in the section VI. Following figures are showing performance of the final best ANN model I constructed after done many experiments.

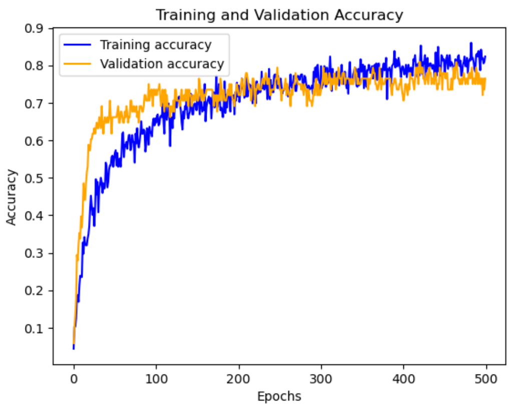


Fig. 4. Training and validating accuracy plot according to epochs

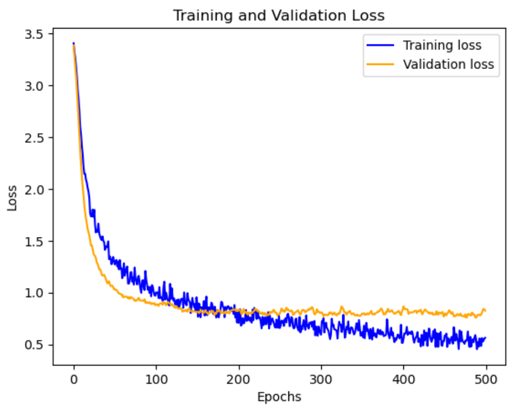


Fig. 5. Training and validating loss plot according to epochs

According to the above Fig. 3. and Fig.4. the ANN model has performed well without any significant overfitting or underfitting in total 500 of epochs. This model gave more than 82% of training accuracy and 76% validation accuracy in last 100 epochs. Finally, this model gave 70.17% of average training accuracy and 71.78% of average validation accuracy.

VII. CONCLUSION

In this paper, I conducted an experiment to predict the leaf species data on the predictive performance of different classifiers and deep learning techniques. I selected six popular classifiers and build an ANN model for conducting my data analysis.

Based on the results of each classifier with different parameters, Linear Discriminant Analysis (LDA) appears to be the best classifier, with an accuracy of 85.29%. This classifier has the highest accuracy, which means that it is able to correctly classify most instances. Additionally, it also has the highest precision and F1-score, which indicates that it has a low false positive rate and a good balance between precision and recall.

Logistic Regression also performed well with an accuracy of 80.88% and a good precision of 86.81%.

Naïve Bayes and SVM have reasonable performance, however, KNN and Decision Tree did not perform as well comparatively. It may be the case that they are not well suited to the specific dataset or task, or that the parameters used for these classifiers were not optimized.

KNN with custom parameters and default parameters have similar accuracy, precision, and recall values of 70.59%. I think the KNN is not a good fit for the leaf dataset.

Both Decision Tree approaches have the lowest accuracy and F1-score, which means that it is not able to correctly classify many instances, and it has a low balance between precision and recall.

In conclusion, it appears that Linear Discriminant Analysis and Logistic Regression are suitable for the leaf dataset. However, it is important to note that the results of these classifiers would depend on the dataset. I think 340 data points are not sufficient to make accurate predictions for a classification problem with 30 classes, especially if each class only contains 8-16 data points. This is because, with so few examples per class, it may be difficult for the model to learn the characteristics that distinguish each class from one another. In addition, the model will be more likely to overfit the limited number of training examples, resulting in poor generalization to new data. This can be solved by adding more data points or different approaches of models or pre-processing techniques.

REFERENCES

[1] UCI Machine Learning Repository, available at: UCI Machine Learning Repository: Leaf Data Set.

[2] Scikit-Learn API Reference, available at: API Reference — scikit-learn 1.2.0 documentation.