Decision Tree Classifier

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*—Decision trees [1] are useful for solving classification and regression tasks in machine learning and data analysis. This research paper explores the construction and evaluation of decision trees for species classification on the Iris dataset [10]. The classifier is toggled with various splitting criterion like gini and entropy [2] to evaluate the performance under each criterion. The tree is also visualized using Graphviz [8] to visibly understand the way the tree classifies the data. This analysis is essential in establishing a decent understanding of Decision trees necessary in the world of Data science research and industrial applications.**

***Index Terms*—Iris dataset, Decision tree, Graphviz, Scikit-Learn, gini index, entropy**

I. Introduction

The objective is to train a Decision tree model over the Iris dataset. The dataset is loaded using the Pandas module [11] into a DataFrame [12] and split into train and test. The train part is used while training and the test part is used while evaluating the model. The trained model is also visualized using the graphviz module to view and understand the underlying tree. Further, the impurity metric is modified to understand the performance change for different impurities.

A brief overview of the columns in the Iris dataset is available in the Table I. The objective is to predict the Dependent variable ***Class*** using the Independent variables ***Sepal Length, Sepal Width, Petal Length, Petal Width***.

Column	Type
Sepal Length	Independent
Sepal Width	Independent
Petal Length	Independent
Petal Width	Independent
Class	Dependent

TABLE I
IRIS DATASET

In the subsequent sections, information like the prerequisites to work with decision trees, the procedure to read a CSV file in Python, method to split the data into training and testing parts, building a decision tree model using the entropy and Gini index, and finding its accuracy and plotting it using graphviz are explained in detail in the Methodology section. The results and the possible explanation for the same are discussed in the Results and Conclusion sections respectively.

II. Methodology

A. Prerequisites

Before training a Decision tree classifier and plotting the tree using graphviz, there are some prerequisites that need to setup or installed.

1) **Python** : You should set up Python on your computer; you may do so by downloading it from the official Python website.

2) **IDLE** : Any Python IDLE like pycharm [14], Google Colab [15], Jupyter Notebook [16] can be used to work with decision trees.

3) **Packages** : Numpy [18], pandas, sklearn, csv, graphviz are needed to work with decision trees and plot the tree.

4) **Data set** : The Iris dataset can be found on multiple sites like the official website [10], the Kaggle website [13] or various other links available on Google.

B. Importing Packages

To work with datasets and decision trees, the following packages are required.

1) **CSV**: CSV [17] is an integrated module that handles Comma-Separated Values (CSV) files. It includes functions and classes to help you read and write CSV files, which are commonly used for storing tables of data.

2) **Pandas**: Pandas is a freely available Python library that manages and analyzes data effectively. It offers user-friendly tools and functions for handling structured data, like tables or spreadsheets.

3) **sklearn** : Scikit-Learn [7] is an open-source machine learning package that is often used in the Python community and is commonly referred to as ”sklearn” in Python. It offers resources for a range of machine learning projects.

4) **Graphviz**: Graphviz is an open-source graph visualization package that allows you to generate visuals like graphs and networks. In Python, you can use the Graphviz library to interface with Graphviz and generate various types

of visualizations, including directed and undirected graphs, flowcharts, and decision trees.

Figure 1 shows all the imports done for the training of the Decision tree classifier.

```
#importing packages
import sklearn
import numpy as np
import csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
```

Fig. 1. Importing the required packages

C. Reading a CSV file

Python supports operations on various types of files. To read a CSV (Comma-Separated Values) file in Python, you can use the CSV module or the pandas library in Python. Here's a step-by-step process on how to do it.

Step 1: Start by importing the panda's module.

Step 2: Use the ***pd.read_csv()*** [19] function to read the CSV file. You should provide the **path** to the CSV file as an **argument**. You can also specify various optional parameters to customize the import, such as specifying the **delimiter, encoding, header row,** etc.

Step 3: After reading the CSV file, the data is stored in a Pandas DataFrame. You can perform various operations on this DataFrame, such as displaying its contents, setting the column names etc.

Figure 2 shows the code used to read a CSV using Pandas.

```
#read the csv file
data = pd.read_csv('IrisNew.csv')
df = pd.DataFrame(data, columns=["Sepal Length","Sepal Width","Petal Length","Petal Width","Class"])
print(df)
```

Fig. 2. Reading the CSV file using pandas

D. Data partitioning into Training and Testing

Dividing your data into training and testing portions is an essential procedure in machine learning for evaluating how well your model works. In Python, you can achieve this task with the help of libraries such as NumPy or Scikit-Learn. Here's a method for doing it with Scikit-Learn, which is a widely used in Machine Learning Industry.

Here, the first four columns in the Iris data set represent the independent variables and the fifth column is the dependent variable.

The Scikit-Learn function ***train_test_split()*** [20] divides the data into sets for training and testing. The **test_size** parameter specifies the fraction of data to be used for testing (e.g., 0.3 for 30 percent testing data). The **random_state** is an optional parameter used to seed the random number generator for reproducibility.

The characteristics of the training dataset are stored in the X_train variable.

The Features from the testing dataset are stored in the X_test variable.

The y_train variable contains the training dataset's corresponding labels.

The y_test variable has the testing dataset's related labels.

Figure 3 shows the code to perform splitting using Scikit-Learn.

```
#split the data into training and testing
X = df.values[:,0:4] # first 4 columns are independent variables
Y = df.values[:,4] # last column is dependent variable
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3,random_state=100)
#test size = 70% training and 30 percent testing
```

Fig. 3. Data partitioning into Training and Testing sets

Once you've split your data using these variables, you can proceed to train and evaluate the machine learning models.

In general, the **Class** column is converted to numerals using ***LabelEncoder()*** [27]. But, in case of Decision tree, it is not necessary as Decision tree can be trained using directly categorical variables. So, that approach was not implemented.

E. Building a decision tree using entropy

A decision tree can be built using using the ***DecisionTreeClassifier()*** [7] with ***entropy*** as the **criterion parameter**, available in the Scikit-Learn library of Python. Here's a step-by-step process to achieve that:

Step 1: Start by importing the required libraries, including Scikit-Learn.

Step 2: Ensure that you have your dataset prepared with features (X) and corresponding labels (y).

Step 3: Initialize the *DecisionTreeClassifier()* with the criterion set to 'entropy'. You can also specify other hyperparameters to control the tree's growth and prevent overfitting.

Step 4: Train the *DecisionTreeClassifier()* on your training data using the *fit()* method.

Step 5: After training the classifier, you can use it to make predictions on new, unseen data (e.g., your testing data).

Step 6: Use a variety of measures, such as precision, recall, accuracy, or F1-score [21], to assess the effectiveness of your decision tree model. Scikit-Learn methods like ***accuracy_score()*** [22] are also available.

Figure 4 shows the code to build a Decision tree classifier with entropy as criterion.

```
#building a decision tree model using entropy
clf_entropy = DecisionTreeClassifier(criterion = "entropy",random_state = 100)
clf_entropy.fit(X_train, y_train)
y_pred = clf_entropy.predict(X_test)
print("Predicted values:")
print(y_pred)
print ("Accuracy : ", accuracy_score(y_test,y_pred)*100)
```

Fig. 4. Decision Tree Classifier using entropy

F. Building a decision tree using Gini Index

To build a decision tree classifier using Gini index, follow the steps in the above section, but in the *DecisionTreeClassifier()* class, set the criterion parameter to **gini**.

Figure 5 shows the code to build a Decision tree classifier with gini index as criterion.

```
#building a decision tree model using gini index
clf_entropy = DecisionTreeClassifier(criterion = "gini",
            random_state = 100)
clf_entropy.fit(X_train, y_train)
y_pred = clf_entropy.predict(X_test)
print("Predicted values:")
print(y_pred)
print ("Accuracy : ", accuracy_score(y_test,y_pred)*100)
```

Fig. 5. Decision Tree Classifier using Gini index

G. Plotting the decision treen using graphviz

Graphviz is an open-source graph visualization package that allows you to generate visuals like graphs and networks. In Python, you can use the Graphviz library to interface with Graphviz and generate various types of visualizations, including directed and undirected graphs, flowcharts, and decision trees.

The ***export_graphviz()*** [23] method from the ***sklearn.tree*** module may be used to export a decision tree in the DOT [24] format, which can then be visualized using the Graphviz library in Python.Here's a step-by-step process.

Step 1: First, import the necessary libraries, including scikit-learn and Graphviz.

Step 2: If you have an *DecisionTreeClassifier()* instance that is fine, else, Create an instance of the *DecisionTreeClassifier()* and train it with your data.

Step 3: Use the *export_graphviz()* function to export the decision tree to the DOT format. Specify the **model, feature_names, class_names,** and other parameters. Here, by setting **out_file** parameter to **None**, we capture the DOT data as a string. Other parameters like *feature_names* - A list of feature names (optional), *class_names* - A list of class names (optional) and, *filled, rounded, and special_characters* are for adjusting the appearance of the tree.

Step 4: Use the Graphviz library to render the DOT data and display the decision tree. Figure 6 describes the code to plot the decision tree using graphviz.

```
#Plotting the results using graphviz
import graphviz
feature_cols = ['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width']
dot_data = tree.export_graphviz(clf_entropy, out_file=None,feature_names = feature_cols,
 class_names=['setosa','versicolor','virginica'], filled=True, rounded=True, special_characters=True
graph = graphviz.Source(dot_data)
graph
```

Fig. 6. Plotting the decision treen using graphviz

The plot generated is then compared with the one available in the official Scikit-Learn documentation [25] to compare both the images.

H. Cross Validation for model performance

Cross-validation is a method for determining a machine learning model's performance, including Decision Tree classifiers. It helps in estimating how well the model will perform on unseen data and whether it's overfitting or underfitting the training data. In Python, you can perform cross-validation with Decision Trees using the scikit-learn library. This is how you do it:

Step 1: Import the required libraries, such as Scikit-Learn.

Step 2: If you have an *DecisionTreeClassifier()* instance that is fine else, create an instance of the *DecisionTreeClassifier()* and train it with your data. The trained model is now use in the ***cross_val_score()*** [26] function to perform the cross validation scores. Figure 7 shows the code to do the same. Here clf_entropy is the decision tree classifier you created. X_train is the feature data, y_train is the target labels, cv is the number of cross-validation folds. You can adjust these parameters based on your needs.

Step 3: To assess the model's performance following cross-validation, you may compute several performance measures like mean precision, mean accurateness, mean recall, or mean F1-score.

RESULTS

After reading the csv file, you can print the sample data in the data frame. Figure 8 shows a sample of the Iris data stored in the Dataframe.

Decision tree built over the whole dataset without any train and test split is show in the Figure 14.

```
#performing cross validation for model performance
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf_entropy, X_train,y_train, cv=3)
print("Cross-validation scores:", scores)

# Calculate and print the mean and standard deviation of the scores
print("Mean accuracy:", scores.mean())
```

Fig. 7. Cross Validation

	Sepal Length	Sepal Width	Petal Length	Petal Width	Class
0	4.3	3.0	1.1	0.1	setosa
1	4.4	2.9	1.4	0.2	setosa
2	4.4	3.0	1.3	0.2	setosa
3	4.4	3.2	1.3	0.2	setosa
4	4.5	2.3	1.3	0.3	setosa

Fig. 8. Sample DataFrame of Iris dataset

The Decision tree provided in the Decision tree webpage of the Scikit-Learn website [25] is shown in Figure 10.

Comparing both the trees, one can see that, though the splitting of the tree is completely same in both the cases, but the split conditions are different in one of the sepal width conditions of the tree.

After building the decision tree using entropy, you can see the predicted values of the test data and you can also see the accuracy using the accuracy-score function. Below is the output of Figure 11.

After building the decision tree using gini, you can see the predicted values of the test data and you can also see the accuracy using the ***accuracy_score()*** [22] function.
Entropy, Gini index are two commonly used impurity measures in decision tree classifiers. They are used to determine

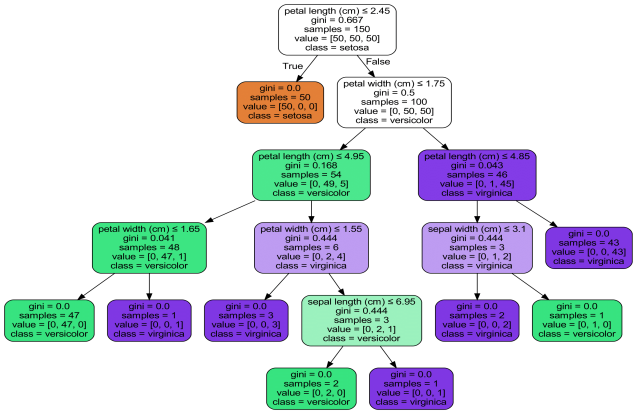

Fig. 9. Decision tree built on complete Iris dataset

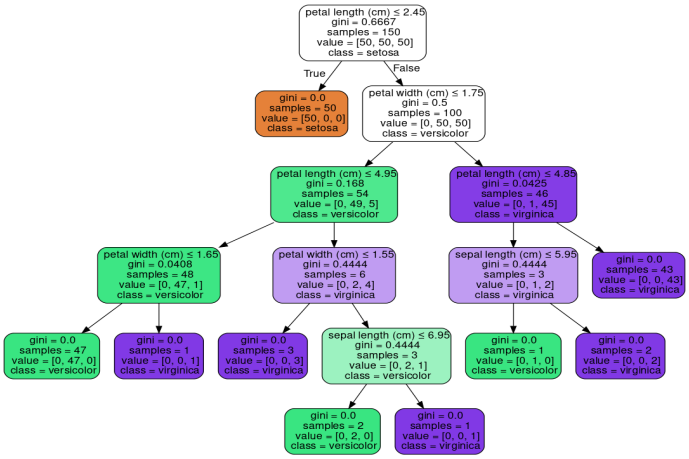

Fig. 10. Decision tree provided in the Scikit-Learn website [25]

```
Predicted values:
['virginica' 'setosa' 'virginica' 'setosa' 'virginica' 'virginica'
 'versicolor' 'setosa' 'virginica' 'setosa' 'setosa' 'versicolor' 'setosa'
 'setosa' 'virginica' 'versicolor' 'versicolor' 'virginica' 'virginica'
 'virginica' 'versicolor' 'setosa' 'virginica' 'setosa' 'versicolor'
 'virginica' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'virginica' 'setosa' 'versicolor' 'setosa' 'versicolor'
 'versicolor' 'virginica' 'virginica' 'versicolor' 'versicolor'
 'virginica' 'virginica' 'setosa']
Accuracy :  95.55555555555556
```

Fig. 11. Output of Decision tree on test data using *entropy* as impurity

how well a particular attribute or feature splits the data into different classes. Entropy measures the disorder in a dataset. In the context of decision trees, it quantifies the impurity of a node by measuring the degree of disorder in the class labels of the data samples within that node.
The Gini index serves as an indicator of how likely it is to make an incorrect classification if a randomly selected item were assigned to a class based on the node's class distribution. There is no a huge difference in the prediction values or in the accuracy if we use the criterion as entropy or gini in the DecisionTreeClassifier. Figure 12 shows the output received using gini as the impurity measure for the same test dataset.

```
Predicted values:
['virginica' 'setosa' 'virginica' 'setosa' 'virginica' 'virginica'
 'versicolor' 'setosa' 'virginica' 'setosa' 'setosa' 'versicolor' 'setosa'
 'setosa' 'virginica' 'versicolor' 'versicolor' 'virginica' 'virginica'
 'virginica' 'versicolor' 'setosa' 'virginica' 'setosa' 'versicolor'
 'virginica' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'virginica' 'setosa' 'versicolor' 'setosa' 'versicolor'
```

Fig. 12. Output of Decision tree on test data using *gini* as impurity

Figure 13 and 14 shows the Decision tree built on top of the dataset obtained after performing a train-test split with **gini index** and **entropy** respectively. Both the trees are pretty much similar even when the impurity criterion is modified, but they are completely different from the one that was trained on the whole dataset (Figure 11).

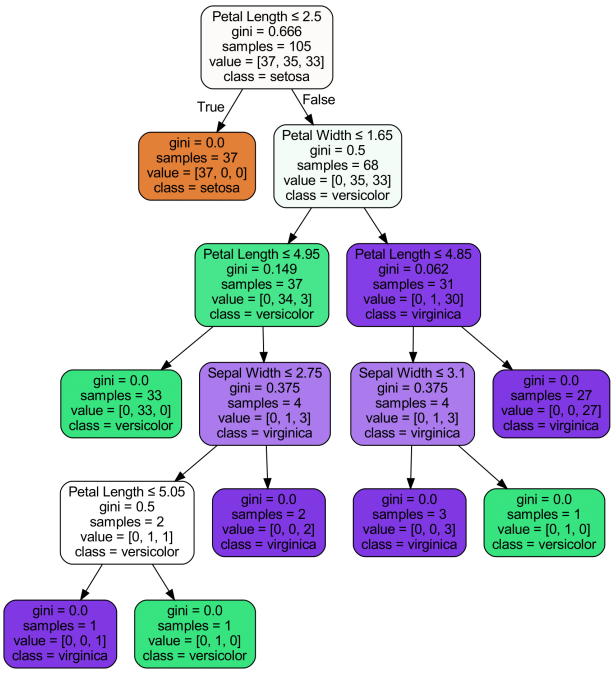

Fig. 13. Decision tree with gini impurity trained after 30% train-test split

Table II shows the output after performing cross validation. The data was divided with 30% for testing. So, the number of folds of cross validation came out to be **3**. The cross validation and the mean accuracy scores show that the model is performing very well on the unseen data.

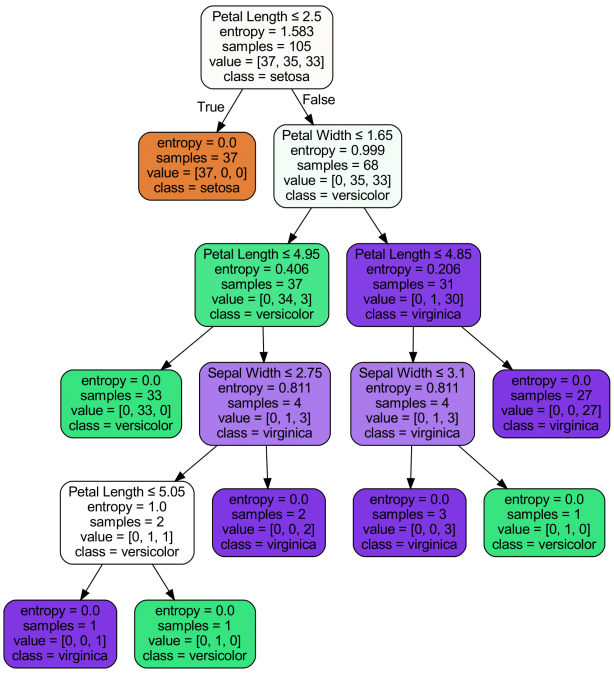

Fig. 14. Decision tree with entropy impurity trained after 30% train-test split

Cross-validation score	{ 0.91428571, 1. , 0.97142857 }
Mean accuracy	0.9619047619047619

TABLE II
CROSS-VALIDATION AND MEAN ACCURACY SCORES

CONCLUSION

This study explored decision tree construction and evaluation for species classification using the Iris dataset. Essential prerequisites, including Python and relevant packages, data reading, and data splitting are covered. Decision trees were built using various metrics like entropy and the Gini index, revealing no substantial accuracy difference. This might be due to the low amount of data used for training and testing. The importance of using Graphviz for visualizing decision trees is also highlighted. Cross-validation played a pivotal role in assessing model performance showing that the model is perfectly trained. This research provides a concise roadmap for leveraging decision trees in machine learning and data analysis, emphasizing their versatility in guiding data-driven insights. In the future, more amount of data can be used to understand in deeper sense as to where gini index and entropy provide different results.

REFERENCES

[1] https://www.geeksforgeeks.org/decision-tree/
[2] https://www.geeksforgeeks.org/gini-impurity-and-entropy-in-decision-tree-ml/
[3] https://medium.com/analytics-vidhya/exploratory-data-analysis-iris-dataset-4df6f045cda
[4] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[5] https://www.analyticsvidhya.com/blog/2021/06/analyzing-decision-tree-and-k-means-clustering-using-iris-dataset/
[6] https://pypi.org/project/graphviz/
[7] https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/
[8] https://scikitlearn.org/stable/modules/generated/sklearn.tree.export.graphviz.html
[9] https://quantdare.com/decision-trees-gini-vs-entropy/
[10] https://archive.ics.uci.edu/dataset/53/iris
[11] https://pandas.pydata.org/docs/getting_started/intro_tutorials/01_table_oriented.html
[12] https://www.databricks.com/glossary/what-are-dataframes
[13] https://www.kaggle.com/datasets/uciml/iris
[14] https://www.jetbrains.com/pycharm/
[15] https://colab.research.google.com/
[16] https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html
[17] https://docs.python.org/3/library/csv.html
[18] https://numpy.org/doc/stable/user/absolute_beginners.html
[19] https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html
[20] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[21] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html
[22] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
[23] https://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html
[24] https://graphviz.org/doc/info/lang.html
[25] https://scikit-learn.org/stable/modules/tree.html#classification
[26] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
[27] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html