Investigating Decision Trees

Lillian Mueller lmuelle1@umd.edu
Regina Hong rhong@umd.edu

***Abstract*— Predictive analytics and modeling are very prominent in today's business models and strategies. These techniques utilize historical data to develop the predictive model that can then be applied for new datasets to predict future target variables. A decision tree is an example of a supervised learning algorithm that may be used for classification or regression [1]. When implemented correctly, decision trees can predict attributes of any datapoint given known influential attributes defined by the model. These attributes are selected through the process of supervised segmentation. The target attribute is chosen and the data is broken down, or segmented, into groups based on information gain and entropy of the dataset or group. While working with the SKLearn Iris dataset, we investigated how manipulating the purity measure criterion affects the overall performance of the decision tree. For this dataset, we found that deriving information gain via entropy versus gini index has largely the same outcome.**

I. Introduction

In general, a decision tree consists of a root node, various branching nodes, and leaf nodes [1]. Each node in the tree represents a split in the dataset. These splits are determined through supervised segmentation. The goal of this segmentation is to make each group, or segment, as pure as possible. In this process, attributes are selected based on information gain and the segments'purity which can be measured via the entropy of the group or using the Gini index. In the case of classification, a sement's purity measures how closely the data fits into a single category (entropy) or the likelihood that a sample belongs to a single category (Gini index) [2]. This calculation helps evaluate the effectiveness of each node and how well it will perform [1]. For a segment to be pure, the entropy value should be as close to 0 as possible. The more entropy a group has, the more variability there is. Information gain is then calculated using the entropy before the split, compared to the entropy following the split. The attribute that causes the greatest information gain, or entropy decrease, is then selected as the next node in the tree. This process is repeated until the leaf nodes in the tree reach the desired purity.

There are many advantages and disadvantages to utilizing decision trees in predictive analytics and modeling. Decision trees are easy to understand and interpret. These models can be visualized and drawn out, depicting each node and what attributes they represent. Additionally, compared to other algorithms, decision trees require less data preparation and are faster. However, the downside to this method is overfitting and developing a tree for larger datasets can be difficult [3]. Overfitting in machine learning algorithms happens when the model accurately fits the training datasets but is unable to be applied accurately to testing datasets and new data [4]. This will greatly decrease the performance of the decision tree and lead to incorrect predictions. Developing methods to combat this overfitting will lead to more effective models.

This investigation will demonstrate how manipulating supervised segmentation and tuning the purity measurements can increase accuracy and performance of the decision tree model. We will be working with the iris dataset. This data consists of various characteristics of Iris flowers including sepal length and width, petal length and width, and also its class: Iris Setosa, Iris Versicolour, and Iris Virginica. Using this dataset, we will be developing a classification tree model to classify the dataset into the 3 species. To do this, we will be splitting the dataset into training sets and testing sets. Along the way, we will be experimenting with the information gain and purity measurements to optimize the performance of the decision tree model.

II. Methodology

Several Python packages were utilized in order to view the Iris dataset, build a Decision Tree model, and test the model. The majority of the modeling and analysis comes from various modules stemming from the `scikit-learn` (sklearn) package.

```python
from sklearn.datasets import load_iris
from sklearn.tree import
    ↪ DecisionTreeClassifier
from sklearn import tree, preprocessing
from sklearn.model_selection import
    ↪ train_test_split
from sklearn.metrics import
    ↪ accuracy_score, r2_score
import pandas as pd
import numpy as np
```

Listing 1. Libraries used for this assignment.

First, the Iris dataset was loaded as an `sklearn.utils.` `↪ Bunch` object called `iris_data` via the `load_iris` `↪ ()` function. This object is similar to a Python dictionary. To make the data easier to read, the pandas package was imported and used to transform said dictionary into the `df_iris` dataframe, where the data parameter was set as the data attribute of the dataset and the columns parameter as the `feature_names` attribute. A new column called "class" was added to this dataframe which contains the `target` variable of the Iris dataset; this is the class of the Iris plant —setosa, versicolor, or virginica. Since the `target` attribute contains an array with values from 0-2, the `.replace` function was used to map these numerical values to their corresponding classifications: setosa for `0`, versicolor for `1`, and virginica for `2`.

With `df_iris` ready for processing, the next step was to build an initial Decision Tree. To do this, the `DecisionTreeClassifier()` function from `sklearn` was used with default parameters.

```
dt_model = tree.DecisionTreeClassifier()
predictors, target = iris_data.data,
    ↪ iris_data.target
dt_model.fit(predictors, target)
```

Listing 2. Building the Decision Tree

Then, the actual tree diagram was created using `graphviz`, with `feature_names=iris_data.feature_names` and `class_names=iris_data.target_names` so that the resulting file contained the feature and class names for ease of reading.

The nodes of the Decision Tree diagram were compared to the process that was run in a referenced model in Excel. Using petal length as the parent node, it should be noted that the Decision Tree reflected the same splitting process as the Excel model, choosing the same values.

This same Decision Tree—making process was run again, this time specifying the `criterion` parameter of `DecisionTreeClassifier()` function as `criterion ↪ ='entropy'` to see how using Shannon entropy instead of Gini impurity changes the way the Decision Tree is created. Differences between the two Decision Trees are reviewed in the results section.

Since the Iris dataset is relatively small, two parameters in the `DecisionTreeClassifier()` function were independently changed to determine how the Decision Tree changes. The first is `max_depth` which controls the size of the tree by limiting how many levels it will be. Since each internal node requires at minimum 2 child nodes, setting this value to a smaller number stops additional splits that may lead to overfitting. The second parameter is `min_samples_split`; setting a value greater than the default of 2 ensures that splits will not happen when a node only has 2 samples, which may also reduce the possibility of overfitting.

Next, the Iris dataset was split into a test group and train group, where the train group underwent the Decision Tree Classification modeling process and the test group was then used in the resulting model. To split the data into those two groups, the `train_test_split` function was imported from the `sklearn.model_selection` module with the `test_size` parameter set to 0.33 so that around $\frac{1}{3}$ of the data is set aside for testing.

After the Decision Tree model was fitted to the training data, an accuracy score was used to see how well the results from the test data matched with actual data. To do this, the `accuracy_score` function was imported from `sklearn ↪ .metrics` module, setting the `y_true` parameter to actual data and `y_pred` to data coming from the model/-classifier.

The test and train process was repeated again with 20% of the dataset reserved as test data and the other 80% as training data. Once again, the accuracy scores of the train and test data were printed and observed. Additional iterations of this process used the same test/train data split ($\frac{1}{3}$ to $\frac{2}{3}$) but this time also varied the `max_depth` and `criterion` parameters to see their effects on the accuracy of the train and test data.

III. Results

In this investigation, SKLearn's DecisionTreeClassifier was used to create decision tree models for the Iris dataset. By manipulating the parameters for this class, we were able to develop different trees to classify our data by species. The main parameter we focused on was the "criterion" parameter which allows the developer to change the function by which to measure quality of the splits in the dataset. The first function we investigated was the Gini Impurity function. After each split, the method evaluates the likelihood a random sample in the group could be misclassified [2]. This means that the lower the Gini index, the better the split is as there is a low likelihood a sample will be misclassified. The following is a depiction of the tree derived from this method.

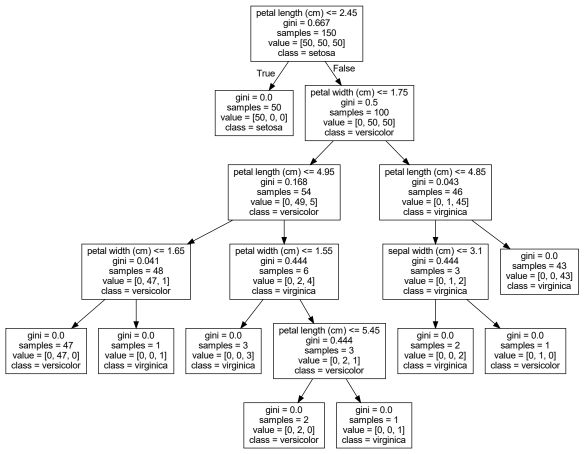
petal length (cm) <= 2.45 gini = 0.667 samples = 150 value = [50, 50, 50] class = setosa
True
False
gini = 0.0 samples = 50 value = [50, 0, 0] class = setosa
petal width (cm) <= 1.75 gini = 0.5 samples = 100 value = [0, 50, 50] class = versicolor
petal length (cm) <= 4.95 gini = 0.168 samples = 54 value = [0, 49, 5] class = versicolor
petal length (cm) <= 4.85 gini = 0.043 samples = 46 value = [0, 1, 45] class = virginica
petal width (cm) <= 1.65 gini = 0.041 samples = 48 value = [0, 47, 1] class = versicolor
petal width (cm) <= 1.55 gini = 0.444 samples = 6 value = [0, 2, 4] class = virginica
sepal width (cm) <= 3.1 gini = 0.444 samples = 3 value = [0, 1, 2] class = virginica
gini = 0.0 samples = 43 value = [0, 0, 43] class = virginica
gini = 0.0 samples = 47 value = [0, 47, 0] class = versicolor
gini = 0.0 samples = 1 value = [0, 0, 1] class = virginica
gini = 0.0 samples = 3 value = [0, 0, 3] class = virginica
petal length (cm) <= 5.45 gini = 0.444 samples = 3 value = [0, 2, 1] class = versicolor
gini = 0.0 samples = 2 value = [0, 0, 2] class = virginica
gini = 0.0 samples = 1 value = [0, 1, 0] class = versicolor
gini = 0.0 samples = 2 value = [0, 2, 0] class = versicolor
gini = 0.0 samples = 1 value = [0, 0, 1] class = virginica

Fig. 1. Decision Tree Utilizing Gini Purity Criterion

In this tree, each node shows the defining attribute, the Gini index value, the number of samples, the sample breakdown, and the favored class. Just looking at the tree, one can infer that the virginica and versicolour class of iris are difficult to tell apart as their attributes greatly overlap. While setosa can be classified within the first level of the tree, the others take up to 5 levels. To determine the accuracy of this model and evaluate its performance, we developed 1000 models using random training sets and tested the models'accuracy using the respective testing sets. The accuracy of the model using the Gini impurity methods converged to 0.9451 and had an average r2 score of 0.916. For this dataset, these statistics prove that the decision tree model is effective in classifying iris species.

The second function we investigated was entropy. This method evaluates the purity of the split by qualifying the disorder within the group. The more variability within the samples, the greater the entropy value for that node. The following is a depiction of the tree derived from this method along with its performance logistics.

Description	Train Data Accuracy	Test Data Accuracy	r2 Score
count	1000.0	1000	1000
mean	1.0	0.945100	0.915793
std	0.0	0.028636	0.044914
min	1.0	0.840000	0.750000
25%	1.0	0.920000	0.887892
50%	1.0	0.940000	0.916574
75%	1.0	0.960000	0.942824
max	1.0	1.000000	1.000000

TABLE I
STATISTICS FOR DECISION TREE UTILIZING GINI IMPURITY CRITERION

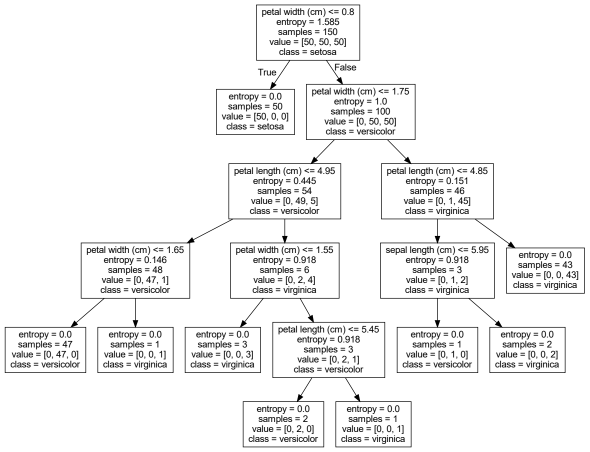
petal width (cm) <= 0.8 entropy = 1.585 samples = 150 value = [50, 50, 50] class = setosa
True
False
entropy = 0.0 samples = 50 value = [50, 0, 0] class = setosa
petal width (cm) <= 1.75 entropy = 1.0 samples = 100 value = [0, 50, 50] class = versicolor
petal length (cm) <= 4.95 entropy = 0.445 samples = 54 value = [0, 49, 5] class = versicolor
petal length (cm) <= 4.85 entropy = 0.151 samples = 46 value = [0, 1, 45] class = virginica
petal width (cm) <= 1.65 entropy = 0.146 samples = 48 value = [0, 47, 1] class = versicolor
petal width (cm) <= 1.55 entropy = 0.918 samples = 6 value = [0, 2, 4] class = virginica
sepal length (cm) <= 5.95 entropy = 0.918 samples = 3 value = [0, 1, 2] class = virginica
entropy = 0.0 samples = 43 value = [0, 0, 43] class = virginica
entropy = 0.0 samples = 47 value = [0, 47, 0] class = versicolor
entropy = 0.0 samples = 1 value = [0, 0, 1] class = virginica
entropy = 0.0 samples = 3 value = [0, 0, 3] class = virginica
petal length (cm) <= 5.45 entropy = 0.918 samples = 3 value = [0, 2, 1] class = versicolor
entropy = 0.0 samples = 1 value = [0, 1, 0] class = versicolor
entropy = 0.0 samples = 2 value = [0, 0, 2] class = virginica
entropy = 0.0 samples = 2 value = [0, 2, 0] class = versicolor
entropy = 0.0 samples = 1 value = [0, 0, 1] class = virginica

Fig. 2. Decision Tree Utilizing Entropy Criterion

Description	Train Data Accuracy	Test Data Accuracy	r2 Score
count	1000.0	1000	1000
mean	1.0	0.943680	0.913660
std	0.0	0.027291	0.043502
min	1.0	0.780000	0.633333
25%	1.0	0.920000	0.886492
50%	1.0	0.940000	0.914237
75%	1.0	0.960000	0.942562
max	1.0	1.000000	1.000000

TABLE II
STATISTICS FOR DECISION TREE UTILIZING ENTROPY CRITERION

Each node in this tree also shows similar values to the previous tree. However, rather than displaying the gini index, the nodes show the entropy measurement at each split. Where the gini measurements range from 0 to 0.5, the entropy measurements range from 0 to 1. Additionally, there are some differences within the attributes chosen in each tree. For example, the root node in the first tree refers to petal length, whereas the second tree splits the root node by petal width. After running this model 10 times over, the average accuracy converged to 0.9437 where the r2 score was 0.914. From this data, one can conclude that using entropy and the gini purity index as a measure of purity is very similar, however the entropy metric is marginally better for this dataset.

While the accuracy of each model was very high, the accuracy of the model when applied to the training set was near 1.0. This could lead the analyst to conclude the model is overfitting the data. To combat this, we reran the models but constrained the trees to 4 levels. The following is the depiction of the trees and a breakdown of its performance for the gini purity and entropy methods respectively.

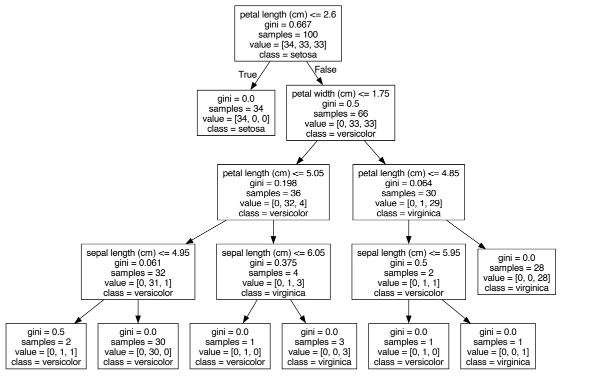
petal length (cm) <= 2.6 gini = 0.667 samples = 100 value = [34, 33, 33] class = setosa
True
False
gini = 0.0 samples = 34 value = [34, 0, 0] class = setosa
petal width (cm) <= 1.75 gini = 0.5 samples = 66 value = [0, 33, 33] class = versicolor
petal length (cm) <= 5.05 gini = 0.198 samples = 36 value = [0, 32, 4] class = versicolor
petal length (cm) <= 4.85 gini = 0.064 samples = 30 value = [0, 1, 29] class = virginica
sepal length (cm) <= 4.95 gini = 0.061 samples = 32 value = [0, 31, 1] class = versicolor
sepal length (cm) <= 6.05 gini = 0.375 samples = 4 value = [0, 1, 3] class = virginica
sepal length (cm) <= 5.95 gini = 0.5 samples = 2 value = [0, 1, 1] class = versicolor
gini = 0.0 samples = 28 value = [0, 0, 28] class = virginica
gini = 0.5 samples = 2 value = [0, 1, 1] class = versicolor
gini = 0.0 samples = 30 value = [0, 30, 0] class = versicolor
gini = 0.0 samples = 1 value = [0, 1, 0] class = versicolor
gini = 0.0 samples = 3 value = [0, 0, 3] class = virginica
gini = 0.0 samples = 1 value = [0, 1, 0] class = versicolor
gini = 0.0 samples = 1 value = [0, 0, 1] class = virginica

Fig. 3. Decision Tree Utilizing Gini Impurity Criterion with 4 Levels

Description	Train Data Accuracy	Test Data Accuracy	r2 Score
count	1000.000000	1000	1000
mean	0.992880	0.945540	0.916688
std	0.007494	0.027685	0.043817
min	0.970000	0.840000	0.678457
25%	0.990000	0.920000	0.891068
50%	0.990000	0.940000	0.916713
75%	1.000000	0.960000	0.942562
max	1.000000	1.000000	1.000000

TABLE III
STATISTICS FOR DECISION TREE UTILIZING GINI IMPURITY CRITERION

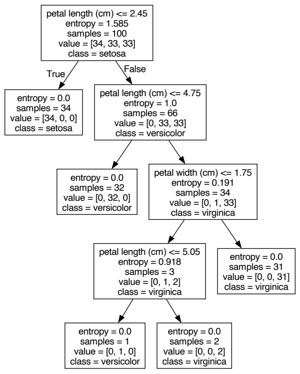
petal length (cm) <= 2.45 entropy = 1.585 samples = 100 value = [34, 33, 33] class = setosa
True
False
entropy = 0.0 samples = 34 value = [34, 0, 0] class = setosa
petal length (cm) <= 4.75 entropy = 1.0 samples = 66 value = [0, 33, 33] class = versicolor
entropy = 0.0 samples = 32 value = [0, 32, 0] class = versicolor
petal width (cm) <= 1.75 entropy = 0.191 samples = 34 value = [0, 1, 33] class = virginica
petal length (cm) <= 5.05 entropy = 0.918 samples = 3 value = [0, 1, 2] class = virginica
entropy = 0.0 samples = 31 value = [0, 0, 31] class = virginica
entropy = 0.0 samples = 1 value = [0, 1, 0] class = versicolor
entropy = 0.0 samples = 2 value = [0, 0, 2] class = virginica

Fig. 4. Decision Tree Utilizing Entropy Criterion with 4 Levels

Description	Train Data Accuracy	Test Data Accuracy	r2 Score
count	1000.000000	1000	1000
mean	0.992370	0.943140	0.912387
std	0.008953	0.027439	0.043462
min	0.960000	0.840000	0.740428
25%	0.990000	0.920000	0.885649
50%	0.990000	0.940000	0.913043
75%	1.000000	0.960000	0.941038
max	1.000000	1.000000	1.000000

TABLE IV
STATISTICS FOR DECISION TREE UTILIZING ENTROPY CRITERION

Regarding the trees developed using Gini impurity, the accuracy of the overall model increased; where the accuracy increased from 0.9451 to 0.9455 and the r2 score increased from 0.9158 to 0.9167. While the training data accuracy decreased from 1.0 to 0.9929, this demonstrates why analysts should be weary of overfitting their model when utilizing this technique. For the dataset, restricting the model resulted in greater accuracy even though the accuracy of the model when applied to the training dataset decreased.

On the other hand, the trees developed using entropy marginally decreased in all aspects. For this dataset, constraining the tree did not improve the efficacy of the decision tree model.

IV. Discussion

In this investigation, it was expected that the methods by which these models were derived would have had a greater impact on the efficacy of the model. However, we see that both methods were largely the same and resulted in comparable results. While the margins of difference for modeling this dataset are small, using decision trees for larger datasets may require more investigation. Even with the small margins, we see that there is a trade off between the accuracy when the model is applied to the training set versus the accuracy when applied to the testing dataset as seen in the case where the gini purity measurement was evaluated. These variables must be taken into account when determining the efficacy of a model.

Within the `DecisionTreeClassifier` function, the `min_samples_leaf` parameter could be utilized in future iterations, as further reading showed that setting this parameter would establish the number of samples in any leaf. This differs from `min_samples_split` and is important because setting a value to `min_samples_split` can still result in leaves with a sample size of one; setting `min_samples_leaf = 5` (as a common example) would prevent any leaves from a sample size of only 1 from occurring and thus better alleviate the possibility of overfitting.

In the future, a larger dataset would create a more concrete evaluation of each variation of this model. Since the iris dataset does not have a complex number of attributes or a large number of samples, it was expected that the accuracy of the models would be very high. With a larger dataset, there would be more data to contribute to more variability and randomness. Additionally, there would be more data to not only train, but also test. Additionally, validation is more effective with larger datasets. For example, cross-validation [5] would be valuable in testing this model. However, with the amount of samples, folding the samples might not be significantly more effective than other methods. Overall, decision trees are simple and easy to implement into any model. They can be an effective analytical tool that can be quickly developed and easily visualized.

References

[1] "What is a Decision Tree — IBM." https://www.ibm.com/topics/decision-trees (accessed Sep. 16, 2023).

[2] S. Dash, "Decision Trees Explained — Entropy, Information Gain, Gini Index, CCP Pruning..," Medium, Nov. 02, 2022. https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c (accessed Sep. 17, 2023).

[3] "Decision Tree Advantages and Disadvantages — Decision Tree Regressor," EDUCBA, Nov. 23, 2021. https://www.educba.com/decision-tree-advantages-and-disadvantages/ (accessed Sep. 16, 2023).

[4] "What is Overfitting? - Overfitting in Machine Learning Explained - AWS," Amazon Web Services, Inc. https://aws.amazon.com/what-is/overfitting/ (accessed Sep. 16, 2023).

[5] "Cross-Validation and Decision Trees — Baeldung on Computer Science." https://www.baeldung.com/cs/cross-validation-decision-trees (accessed Sep. 17, 2023).