Decision Tree from Scratch

By Sunil Ghimire- "**Stop talking! Show me your progress!!**"

In the Data Science and Machine Learning from Scratch series, it's our second algorithm where we have covered all mathematical concepts and a project from scratch with a detailed explanation.

CONTENT COVERAGE:

1. INTRODUCTION

A decision tree is essentially a series of if-then statements, that, when applied to a record in a data set, results in the classification of that record. Therefore, once you've created your decision tree, you will be able to run a data set through the program and get a classification for each record within the data set. What this means to you, as a manufacturer of quality widgets, is that the program you create from this article will be able to predict the likelihood of each user, within a data set, purchasing your finely crafted product.

A decision tree is a type of supervised learning algorithm (**having a pre-defined target variable**) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on the most significant splitter/differentiator in input variables.

2. HOW DECISION TREE WORKS ??

The understanding level of the Decision Trees algorithm is so easy compared with other classification algorithms.

The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

3. DECISION TREE ALGORITHM PSEUDOCODE

- Place the best attribute of the dataset at the root of the tree.
- Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
- Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

4. DECISION TREE CLASSIFIER

In the decision trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. Based on the comparison, we follow the branch corresponding to that value and jump to the next node.

We continue comparing our record's attribute values with other internal nodes of the tree until we reach a leaf node with the predicted class value. As we know how the modeled decision tree can be used to predict the target class or the value. Now let's understand how we can create the decision tree model

5. ASSUMPTIONS WHILE CREATING DECISION TREE

The below are some of the assumptions we make while using the Decision tree:

- In the beginning, the whole training set is considered as the root.
- Feature values are preferred to be categorical. If the values are continuous then they are discredited prior to building the model.
- Records are distributed recursively based on attribute values.
- Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

6. HOW TO SPLIT NODES?

There are a few algorithms to find an optimum split. Let's look at the following to understand the mathematics behind it.

a. **Entropy**:

An alternative splitting criterion for decision tree learning algorithms is information gain. It measures how well a particular attribute distinguishes among different target classifications. Information gain is measured in terms of the expected reduction in the entropy or impurity of the data. The entropy of a set of probabilities is:

$$\text{Entropy} = \sum_{i=1}^{c} -p_i * log_2(p_i)$$

If we have a set of binary responses from some variable, all of which are positive/true/1, then knowing the values of the variable does not hold any predictive value for us, since all the outcomes are positive. Hence, the entropy is zero

The entropy calculation tells us how much additional information we would obtain with knowledge of the variable.

So, if we have a set of candidate covariates from which to choose as a node in a decision tree, we should choose the one that gives us the most information about the response variable (i.e. the one with the highest entropy).

b. Gini Index:

The Gini index is simply the expected error rate:

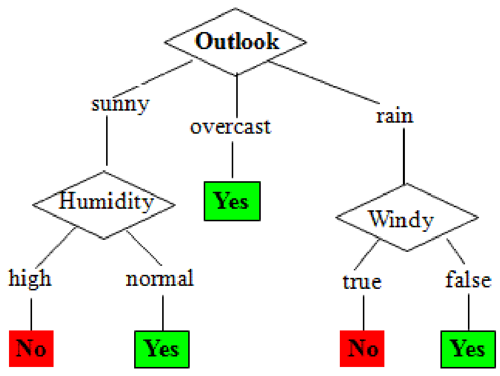
$$\text{Gini} = 1 - \sum_{i=1}^{c} (p_i)^2$$

c. ID3

A given cost function can be used to construct a decision tree via one of several algorithms. The Iterative Dichotomies 3 (ID3) is on such an algorithm, which uses entropy, and a related concept, information gain, to choose features and partitions at each classification step in the tree.

Information gain is the difference between the current entropy of a system and the entropy measured after a feature is chosen.

Gain (T, X) = Entropy(T) - Entropy(T, X)



See Detail about ID3 (Here)

7. DECISION TREE USING SKLEARN

To proceed, you need to import all the required libraries that we require in our further coding. Here we have imported Graphviz to visualize the decision tree diagram. This is can be install in conda environment using conda install python-graphviz.

CODE:

```
import numpy as np
import pandas as pd
from sklearn.tree import export_graphviz
import IPython, graphviz, re
RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)
```

We're going to use the iris dataset. The task for us is now to find the best "way" to split the dataset such that the best nodes can be achieved.

CODE:

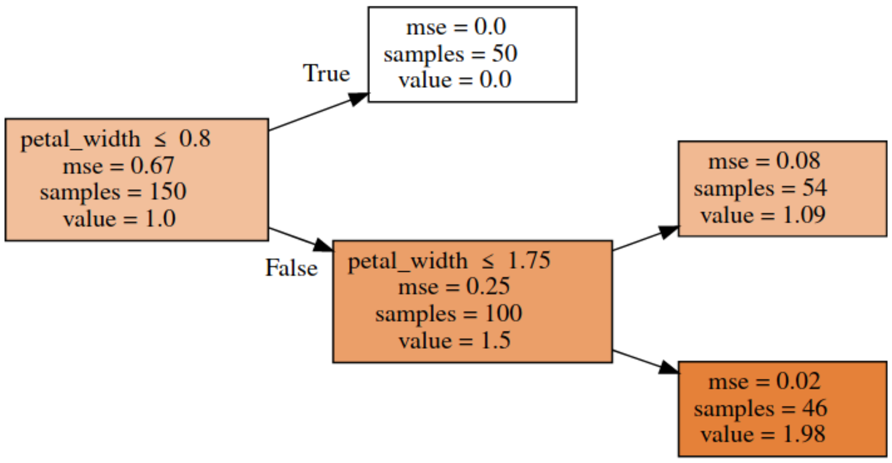
```
df = pd.read_csv("iris.csv")

df['species_label'],_ = pd.factorize(df['species'])
y = df['species_label']
X = df[['petal_length', 'petal_width']]
```

Now let's define a class that draws a representation of a random forest in IPython.

CODE:

```
def draw_tree(t, df, size=10, ratio=0.6, precision=0):
    s=export_graphviz(t, out_file=None, feature_names=df.columns, filled=True,
                      special_characters=True, rotate=True, precision=precision)
    IPython.display.display(graphviz.Source(re.sub('Tree {',
       f'Tree {{ size={size}; ratio={ratio}', s)))
```



Decision tree models can be used for both classification and regression. The algorithms for building trees break down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision (used for regression). The topmost decision node in a tree that corresponds to the best predictor (most important feature) is called a root node. Decision trees can handle both categorical and numerical data.

CODE:

```
X_train,X_test,y_train,y_test = model_selection.train_test_split(X, y,
                                test_size=0.3, random_state=1)
dtree = tree.DecisionTreeClassifier(criterion='entropy' , max_depth=3,
                                   random_state = 0)
dtree.fit(X_train, y_train)
```

Finally, let’s look at how we use all this to make predictions.

CODE:

```
y_pred = dtree.predict(X_test)

count_misclassified = (y_test != y_pred).sum()
print('Misclassified samples: {}'.format(count_misclassified))

accuracy = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: {:.2f}'.format(accuracy))

Output:
    Misclassified samples: 2
    Accuracy: 0.96
```

8. CROSS-VALIDATION

Cross-Validation is a technique that involves reserving a particular sample of a data set on which you do not train the model. Later, you test the model on this sample before finalizing the model.

Here are the steps involved in cross-validation:

- You reserve a sample data set.
- Train the model using the remaining part of the data set.
- Use the reserve sample of the data set test (validation) set. This will help you to know the effectiveness of model performance. If your model delivers a positive result on validation data, go ahead with the current model. It rocks!

9. OVERFITTING

Overfitting is a practical problem while building a decision tree model. The model is having an issue of overfitting is considered when the algorithm continues to go deeper and deeper in to reduce the training set error but results with an increased test set error i.e, Accuracy of prediction for our model goes down. It generally happens when it builds many branches due to outliers and irregularities in data.

Two approaches that we can use to avoid overfitting are:

- Pre-Pruning
- Post-Pruning

Pre-Pruning: In pre-pruning, it stops the tree construction bit early. It is preferred not to split a node if its goodness measure is below a threshold value. But it’s difficult to choose an appropriate stopping point.

Post-Pruning: In post-pruning first, it goes deeper and deeper in the tree to build a complete tree. If the tree shows the overfitting problem then pruning is done as a post-pruning step. We use cross-validation data to check the effect of our pruning. Using cross-validation data, it tests whether expanding a node will make an improvement or not.

If it shows an improvement, then we can continue by expanding that node. But if it shows a reduction in accuracy then it should not be expanded i.e, the node should be converted to a leaf node.

10. ADVANTAGES AND DISADVANTAGES

a. Advantages:

- Decision Trees are easy to explain. It results in a set of rules.
- It follows the same approach as humans generally follow while making decisions.
- The interpretation of a complex Decision Tree model can be simplified by its visualizations. Even a naive person can understand the logic.
- The Number of hyper-parameters to be tuned is almost null.

b. Disadvantages:

- There is a high probability of overfitting in the Decision Tree.
- Generally, it gives low prediction accuracy for a dataset as compared to other machine learning algorithms.
- Information gain in a decision tree with categorical variables gives a biased response for attributes with a

greater no. of categories.

- Calculations can become complex when there are many class labels.

Jupyter Notebook File of Decision tree from scratch is provided here. (LINK)