# Implementation of ID3 Decision Tree Algorithm

**Project link:** https://colab.research.google.com/drive/19IuU8I6xaqtgD-y2ccyigkKV0lRo5qoP?usp=sharing

# Introduction

This report aims to implement a decision tree algorithm to predict the class of given samples, where the value of the given data is numeric, and the classification could have multiple classes. The reason that the Decision Tree algorithm is chosen to use in this implementation is that Iterative Dichotomiser 3 (ID3) help us to improve the ability of decision making in machine learning. It is one of the fundamental algorithms of data science and one of the best techniques for supervised classification.

As mentioned above, Decision Trees are widely used for supervised learning and classification. The algorithm that has been implemented examine each point of each attribute by dividing the column (attribute) into two parts (left and right) and compute the information that could be gained if that point is the threshold. When the best information gain rate is found overall, that point would become the main threshold and divide the training sample into two nodes. This process would repeat itself until all the data in one node has the same target. The process would be explained in the following parts.

The machine learning algorithm takes training (as numeric) and target data to construct the model. When the model is deployed, it takes the data as numeric to predict the decision. The algorithm is programmed in Google Colab environment. The used tools are shown in the following screenshot.

```
#Here I import necessary tools
import pandas as pd
import numpy as np
#I use wine and iris datasets as sample data.
from sklearn.datasets import load_wine
from sklearn.datasets import load_iris
#The following tool is for splitting the data set to train and validation
#dataset.
from sklearn.model_selection import train_test_split
#The following tool is for showing the dataset.
import seaborn as sns
```

# ID3 Algorithm

The algorithm consists of six main functions to build the model: Node Probability, Gini, Impurity Calculation, Calculate Splitting Point, Build Function, and Fit Function. The model also uses three stopping the process: Maximum Depth, Minimum Samples in Each Leaf, and Minimum Samples Split. You can see the initialization function of the class in the following screenshot.

```
def __init__(self, max_depth = 3, min_samples_leaf = 1, min_samples_split = 2):

    self.max_depth = max_depth # THe depth of the tree can't be more than this
    self.min_samples_leaf = min_samples_leaf # Minimum leaf at each end
    self.min_samples_split = min_samples_split # Minimum splitting at each point
    self.classes = None
    self.Tree = None
```

Also, another class named Node is created to create the branches of the tree.

```
#The following class is used for creating the branches of the tree for decision
#tree algorithm.
class Node:
    def __init__(self):

        # links to the left and right child nodes
        self.right = None
        self.left = None

        # column and threshold would be taken from splitting criteria
        self.column = None
        self.threshold = None

        # probability for object inside the Node to belong for each of the given classes
        self.probas = None

        # depth of the given node
        self.depth = None

        # if it is the root Node or not
        self.is_terminal = False
```

## Node Probability

This function calculates the probability of each class in the given node. The probability of each class is the number of samples of each class divided by the total number of samples in that node. The following screenshot contains the related code.

```
def nodeProbas(self, y):
    #Calculates probability of class in a given node
    probas = []

    for one_class in self.classes:
        proba = y[y == one_class].shape[0] / y.shape[0]
        probas.append(proba)
    return np.asarray(probas)
```

### Gini (Entropy)

Gini criterion is the measurement of the data distribution. The Gini is used to calculate the impurity of the sample in each node. Gini is same as Entropy. The Gini can be calculated as follows:

$$Gini = 1 - \Sigma P(i) \wedge 2$$

```
def gini(self, probas):
    #Calculates gini criterion
    return 1 - np.sum(probas**2)
```

### Split Point

As mentioned above, this model split the sample into two nodes to build the tree. To choose the best splitting point, the impurity of left and right of the point is used to calculate the gained information. The model compares the information gain of each point and chooses the best one based on that. The calcBestSplit() function in ID3 class is related to this part.

# Model Evaluation

The model is evaluated on two datasets. Wine and Iris datasets in the Sklearn library.

### Data Preparation

Wine dataset has 13 attributes and 3 classes as a target. This data frame contains more than 1700 samples. The famous Iris dataset contains 4 attributes and 3 classes for classification. To make datasets suitable for training, they transformed into Pandas data frame by using pd.DatFrame(XXX) function.

```
data = load_wine()
X, y, columns = data['data'], data['target'], data['feature_names']
X = pd.DataFrame(X, columns= columns)
```

### Experiment Design and Evaluation

For splitting the dataset into training and testing samples, train_test_split() function from Sklearn library has been used. The model is constructed on the training sample, which is 0.7 of the original dataset.

```
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.30, random_state=42)
```

The hyper parameters to configure the model are Maximum Depth, Minimum Samples in Each Leaf, and Minimum Samples Split. If these parameters are not mentioned, there are pre-selected ones.

```
[ ] %%time
    model = ID3(max_depth = 13, min_samples_leaf=2, min_samples_split=2)
    model.fit(X_train, y_train)
```

### Evaluation Results

When the model is trained and ready to be tested, the test samples, which are 30% of the original dataset, are given to it, and the validation is calculated based on the known targets. For the Wine dataset, the validation score was around 0.94, and for Iris dataset, it was 1.

```
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
print(f'Accuracy for Wine dataset is {accuracy_score(y_test, y_pred)}')
```

```
Accuracy for Wine dataset is 0.9444444444444444
```

```
yd_pred = m.predict(Xd_test)
print(f'Accuracy for Iris dataset is {accuracy_score(yd_test, yd_pred)}')
```

```
Accuracy for Iris dataset is 1.0
```

# Conclusion

In this report, the process of implementing the decision tree or ID3 algorithm has been explained and the model was examined on two different datasets. This was the first time that I implemented a machine learning algorithm from the scratch, and it helped me to achieve a horizon vision about the topic.