**Experiment – 05**

**Aim**: Implement Decision tree to classify the given data set.

**Objective**: Develop practical skills in implementing and evaluating Decision Tree classifiers for given dataset using Python on CoLab.

**Implement an application of Decision Tree algorithm:**

DataSet: Wine Quality from kaggle.

https://www.kaggle.com/datasets/rajyellow46/wine-quality

This dataset contains chemical properties of different wines along with their quality ratings.

# Import necessary libraries

!pip install graphviz

!pip install category\_encoders

import pandas as pd

from sklearn.metrics import confusion\_matrix

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier, export\_graphviz

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

import matplotlib.pyplot as plt

import seaborn as sns

import graphviz

from sklearn import tree

import category\_encoders as ce # import category encoders

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt # data visualization

import seaborn as sns # statistical data visualization

%matplotlib inline

# Load the Wine Quality dataset

data = '/content/winequality-red.csv'

df = pd.read\_csv(data, header=None)

X = df.drop(['quality'], axis=1)

y = df['quality']

# Split the dataset into training and testing sets

X X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.33, random\_state = 42)

X\_train.shape, X\_test.shape

## Encode variables with ordinal encoding

encoder = ce.OrdinalEncoder(cols= ['fixedacidity',  'volatileacidity',  'citricacid', 'residualsugar',  'chlorides',  'freesulfurdioxide',  'totalsulfurdioxide', 'density',  'pH', 'sulphates',  'alcohol'])

X\_train = encoder.fit\_transform(X\_train)

X\_test = encoder.transform(X\_test)

# Initialize the Decision Tree Classifier

clf\_gini = DecisionTreeClassifier(criterion='gini', max\_depth=3, random\_state=0)

# Train the classifier and Make predictions on the testing set

clf\_gini.fit(X\_train, y\_train)

y\_pred\_gini = clf\_gini.predict(X\_test)

# Calculate accuracy

print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred\_gini)))

print('Training-set accuracy score: {0:0.4f}'. format(accuracy\_score(y\_train, y\_pred\_train\_gini)))

#Visualize decision-trees

plt.figure(figsize=(12,8))

tree.plot\_tree(clf\_gini.fit(X\_train, y\_train))

#Visualize decision-trees with graphviz

dot\_data = tree.export\_graphviz(clf\_gini, out\_file=None,

                              feature\_names=X\_train.columns,

                              class\_names=y\_train,

                              filled=True, rounded=True,

                              special\_characters=True)

graph = graphviz.Source(dot\_data)

graph

**Theory: Add theory related to following points**

**Decision Trees:** Decision trees are a supervised learning algorithm used for both classification and regression tasks. They work by recursively splitting the data based on feature values to make decisions. Here's a brief overview of the theory behind decision trees:

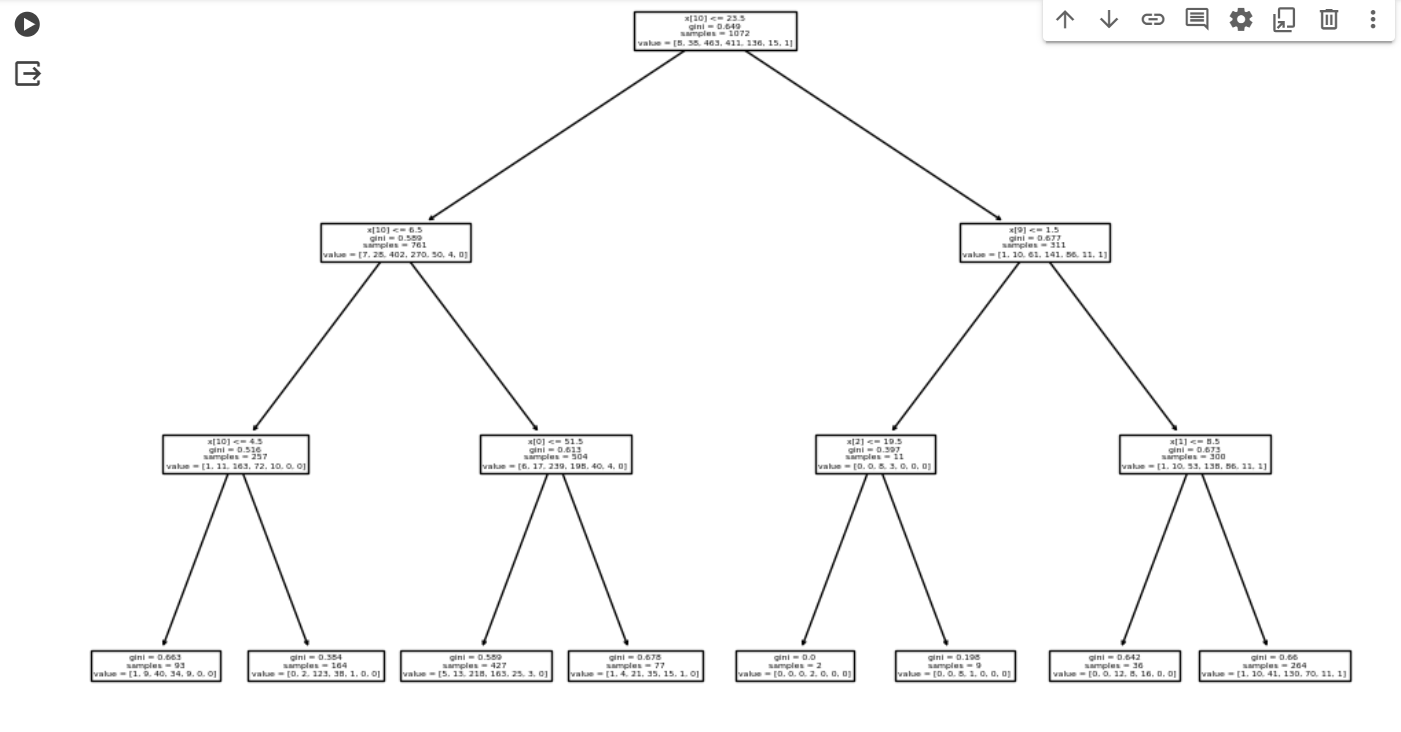
🡪Tree Structure: A decision tree is a hierarchical structure consisting of nodes and branches. Each internal node represents a decision based on the value of a particular feature, and each leaf node represents the predicted class or value.

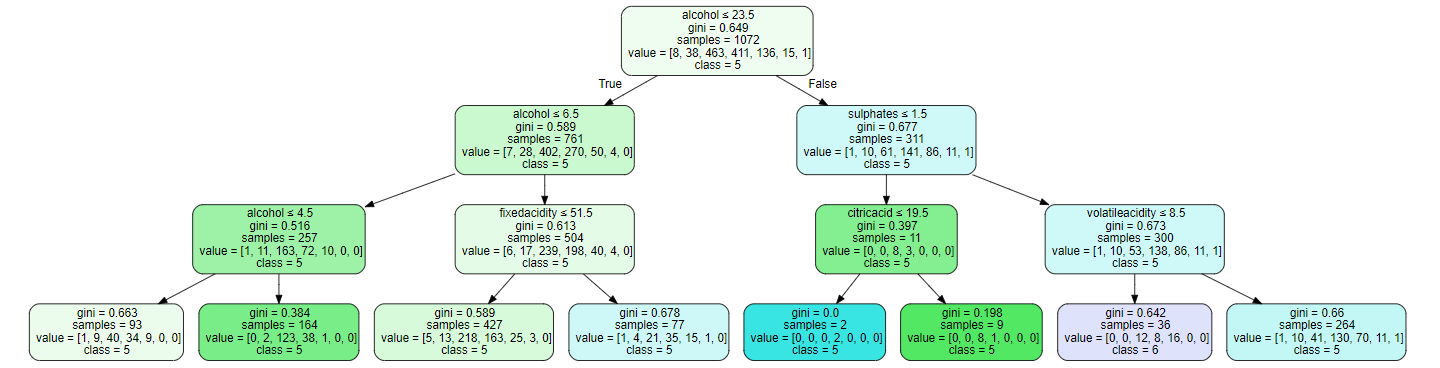
🡪Splitting Criteria: Decision trees split the data at each node based on a criterion that maximizes the homogeneity of the target variable within each subset. Common splitting criteria include Gini impurity and information gain (entropy).

🡪Building the Tree: Decision trees are typically built using a top-down, greedy approach. At each step, the algorithm selects the best feature to split the data and recursively builds the tree until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf).

🡪Pruning: To prevent overfitting, decision trees can be pruned after construction. Pruning involves removing nodes from the tree that do not significantly improve its predictive performance on a validation set.

**Result**





**Conclusion:**

1. In this experiment, we build a Decision-Tree Classifier model to predict the qualityof the wine. we build two models, one with criterion gini index and another one with criterion entropy. The model yields average performance as indicated by the model accuracy in both the cases which was found to be 0.5076.
2. In the model with criterion gini index, the training-set accuracy score is 0.5336 while the test-set accuracy to be 0.5057. These two values are quite comparable. So, there is no sign of overfitting.
3. Similarly, in the model with criterion entropy, the training-set accuracy score is 0.5252 while the test-set accuracy to be 0.5076.We get the same values as in the case with criterion gini. So, there is no sign of overfitting.
4. In both the cases, the training-set and test-set accuracy score is the same. It may happen because of small dataset.
5. The confusion matrix and classification report yields good model performance.

**GitHub Link:**