CS362 – Artificial Intelligence Laboratory

# **Group Name – QCSLY**

## Abhay Upadhyay – 202151001

## Admulwad Gajanan Sambhaji – 202151009

## Deshmukh Kedar Dattrao – 202151047

## Kale Atharva Anna – 202151069

## Prasad Keshav Kapure – 202151115

### **Construction and evaluation of identification (decision) tree classifier**

**Description of dataset :**

**Title :-** Car evaluation database

**Source :-** *Marko Bohanec*

**Donors :-** Marko Bohanec and *Blaz Zupan*

**Past Usage :-** This dataset originates from a hierarchical decision model initially introduced by Bohanec and Rajkovic in 1988 for knowledge acquisition and explanation in multi-attribute decision making. It has been utilized in machine learning contexts, notably in the evaluation of the HINT (Hierarchy Induction Tool) and comparative studies with other algorithms such as C4.5.

**Relevant Information :-** The Car Evaluation Database is derived from a hierarchical decision model used for evaluating cars based on multiple attributes. These attributes include buying price, maintenance cost, number of doors, passenger capacity, luggage boot size, and estimated safety. Each attribute contributes to the overall assessment of car acceptability. The dataset contains examples with structural information removed, focusing solely on the relationship between cars and the aforementioned attributes.

**Dataset Statistics :-**

* **Number of instances :-** 1728
* **Number of attributes :-** 6
* **Missing attributes values :-** None

**Attribute Information :-**

* **Buying :-** This attribute represents the buying price of the car and can take on values: v-high, high, med, low.
* **Maintenance :**- This attribute indicates the maintenance cost of the car and can take on values: v-high, high, med, low.
* **Doors :-** This attribute denotes the number of doors of the car and can take on values: 2, 3, 4, 5-more.
* **Persons :**- This attribute represents the passenger capacity of the car and can take on values: 2, 4, more.
* **Luggage Boot :**- This attribute describes the size of the luggage boot of the car and can take on values: small, med, big.
* **Safety :**- This attribute estimates the safety level of the car and can take on values: low, med, high.

**Class Distribution (Number of instances per class) :-**

* **Unacceptable (unacc) :-** 1210 instances (70.023 %)
* **Acceptable (acc) :**- 384 instances (22.222 %)
* **Good :**- 69 instances (3.993 %)
* **Very good (V –** **good) :-** 65 instances (3.762 %)

**Problem statements :**

**Problem statement – 1**

Randomly select 60 percent of labeled data (from each class) for constructing the tree (training).  Test for the rest of 40 percent data.  Find out the accuracy of the classification tree with the help of confusion matrix and F-score. Use the entropy measure for selection of attributes.

**Solution :**

Installing necessary python packages.

pip install category\_encoders

Importing necessary python libraries and modules

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

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

import category\_encoders as ce

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn import tree

import graphviz

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import f1\_score

Reading the ‘CSV’ file named ‘car\_evaluation.csv’, which is our dataset.

df = pd.read\_csv('/content/car\_evaluation.csv', header=None)

Getting the shape of the dataset. The dataset contains 1728 rows and 6 columns.

df.shape

Printing first 6 rows of data from the dataset.

df.head()



Assigning meaningful names to the columns of the dataset.

col\_names = ['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety', 'class']

df.columns = col\_names

col\_names

Creating two new data-frames X and y, where ‘X’ contains all the columns except ‘class’ and ‘y’ contains the target label ‘class’. Here ‘X’ is our input features and ‘y’ is our target variable.

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

y = df['class']

Splitting the dataset into training set and testing set for both, the input features (X) and the target variable (y). 60% of data is used for training the model and 40% data will be used for testing.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.4, random\_state = 42)

The data present within various columns inside the dataset is in the categorical format and also there is some sort of priorities within those values. So, we apply ordinal encoding on such columns containing categorical data and convert it into numerical values.

encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety'])

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

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

Here, an object of DecisionTreeClassifier is instantiated and trained using training data after encoding. The object is configured to use entropy as a criterion for making decisions.

clf\_en = DecisionTreeClassifier(criterion='entropy')

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



Here, the trained object of DecisionTreeClassifier (clf\_en) is used to predict the target variables for the set of input features (X\_test).

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

Printing the accuracy score of the decision tree classifier with entropy as criterion.

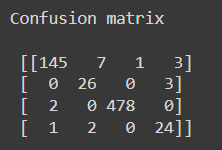
print('Model accuracy score with criterion entropy: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred\_en)))



Calculating and printing the confusion matrix.

cm = confusion\_matrix(y\_test, y\_pred\_en)

print('Confusion matrix\n\n', cm)

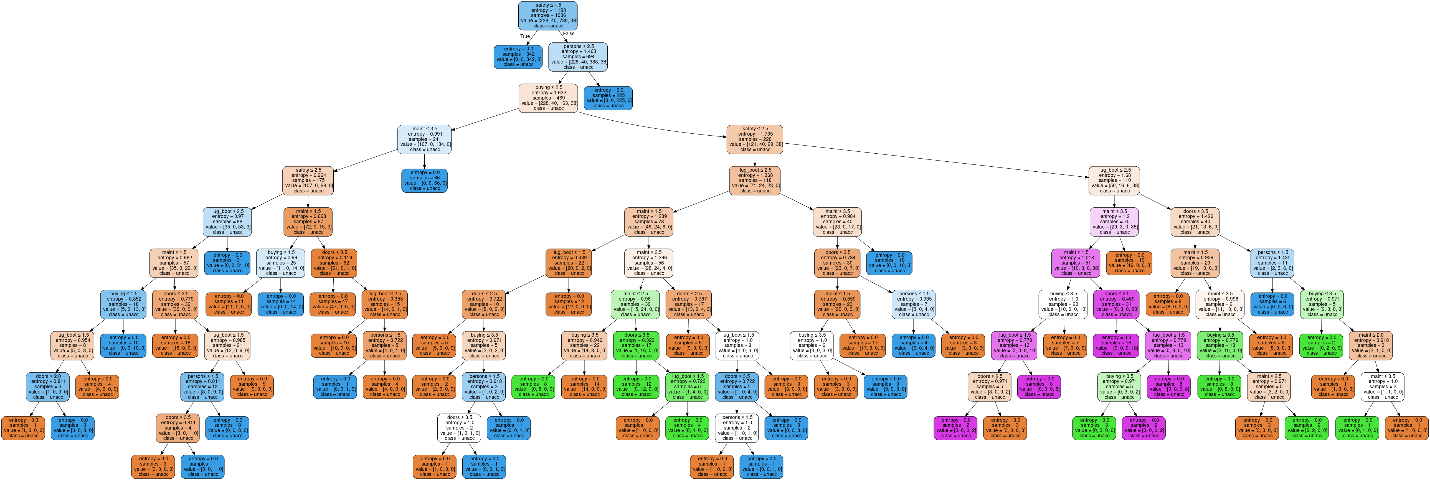


Calculating and printing the F – score.

f1 = f1\_score(y\_test, y\_pred\_en, average='weighted')

print('F1-score: {:.4f}'.format(f1))





**Problem statement – 2**

Repeat the above exercise 20 times. Calculate the average accuracy of classification.

Installing necessary python packages.

pip install category\_encoders

Now, we use the same previous code and iterate it 20 times. We have created two lists to store accuracy scores and F – scores for each iterations. We have used the iteration number as seed value of randomly splitting the data into training and testing sets. Here also our training set is 60% and testing set is 40% of the dataset. At last we have computed and printed the mean of all the accuracies and F – scores.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

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

import category\_encoders as ce

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, f1\_score

import graphviz

# Load the dataset

df = pd.read\_csv('/content/car\_evaluation.csv', header=None)

col\_names = ['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety', 'class']

df.columns = col\_names

# Define lists to store results

accuracy\_scores = []

f1\_scores = []

# Perform 20 iterations

for i in range(20):

    X = df.drop(['class'], axis=1)

    y = df['class']

    # Split data into train and test sets

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=i)

    # Encode categorical features

    encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety'])

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

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

    # Train Decision Tree classifier

    clf\_en = DecisionTreeClassifier(criterion='entropy')

    clf\_en.fit(X\_train\_enc, y\_train)

    # Predict labels

    y\_pred\_en = clf\_en.predict(X\_test\_enc)

    # Calculate accuracy and F1-score

    accuracy = accuracy\_score(y\_test, y\_pred\_en)

    f1 = f1\_score(y\_test, y\_pred\_en, average='weighted')

    # Append results to lists

    accuracy\_scores.append(accuracy)

    f1\_scores.append(f1)

# Calculate average accuracy and F1-score

avg\_accuracy = np.mean(accuracy\_scores)

avg\_f1 = np.mean(f1\_scores)

# Print average accuracy and F1-score

print('Model average accuracy score with criterion entropy:', avg\_accuracy)

print('Model average F1-score:', avg\_f1)



**Problem statement – 3**

Repeat steps 1 and 2 with Gini index as a measure for selection of attributes.

**STEP - 1**

Installing necessary python packages.

pip install category\_encoders

Importing necessary python libraries and modules

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

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

import category\_encoders as ce

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn import tree

import graphviz

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import f1\_score

Reading the ‘CSV’ file named ‘car\_evaluation.csv’, which is our dataset.

df = pd.read\_csv('/content/car\_evaluation.csv', header=None)

Getting the shape of the dataset. The dataset contains 1728 rows and 6 columns.

df.shape

Printing first 6 rows of data from the dataset.

df.head()



Assigning meaningful names to the columns of the dataset.

col\_names = ['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety', 'class']

df.columns = col\_names

col\_names

Creating two new data-frames X and y, where ‘X’ contains all the columns except ‘class’ and ‘y’ contains the target label ‘class’. Here ‘X’ is our input features and ‘y’ is our target variable.

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

y = df['class']

Splitting the dataset into training set and testing set for both, the input features (X) and the target variable (y). 60% of data is used for training the model and 40% data will be used for testing.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.4, random\_state = 42)

The data present within various columns inside the dataset is in the categorical format and also there is some sort of priorities within those values. So, we apply ordinal encoding on such columns containing categorical data and convert it into numerical values.

encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety'])

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

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

Here, an object of DecisionTreeClassifier is instantiated and trained using training data after encoding. The object is configured to use Gini index as a criterion for making decisions.

clf\_gini = DecisionTreeClassifier(criterion='gini')

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



Here, the trained object of DecisionTreeClassifier (clf\_gini) is used to predict the target variables for the set of input features (X\_test).

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

Printing the accuracy score of the decision tree classifier with Gini as criterion.

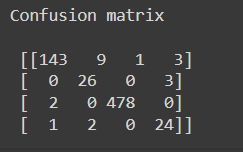
print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred\_gini)))



Calculating and printing the confusion matrix.

cm = confusion\_matrix(y\_test, y\_pred\_gini)

print('Confusion matrix\n\n', cm)

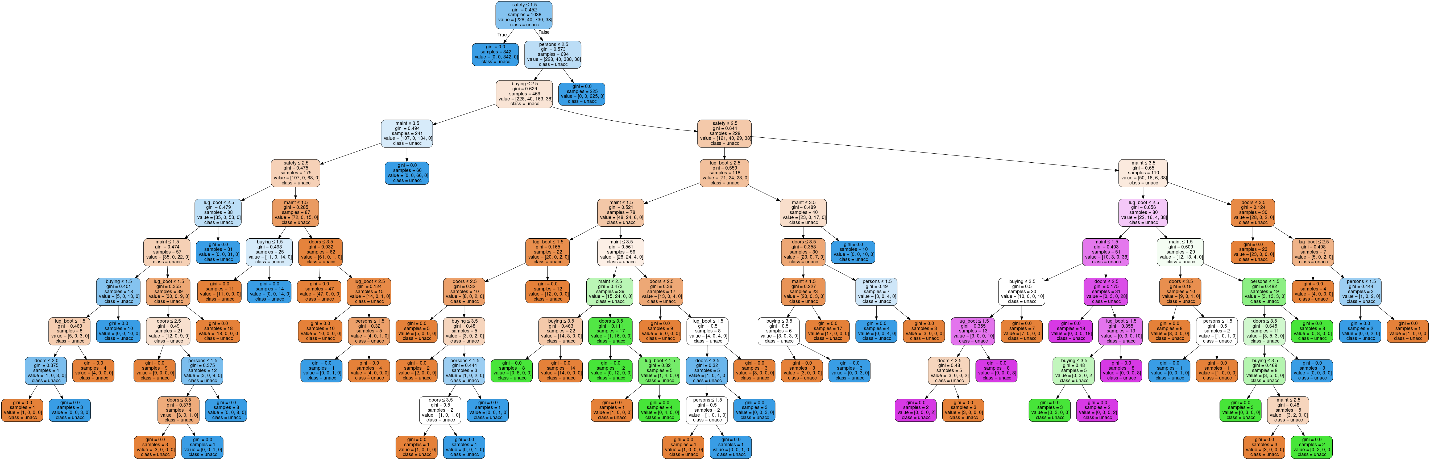


Calculating and printing the F – score.

f1 = f1\_score(y\_test, y\_pred\_gini, average='weighted')

print('F1-score: {:.4f}'.format(f1))





**STEP – 2**

Repeat the above exercise 20 times. Calculate the average accuracy of classification.

Installing necessary python packages.

pip install category\_encoders

Now, we use the same previous code and iterate it 20 times. We have created two lists to store accuracy scores and F – scores for each iterations. We have used the iteration number as seed value of randomly splitting the data into training and testing sets. Here also our training set is 60% and testing set is 40% of the dataset. At last we have computed and printed the mean of all the accuracies and F – scores.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

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

import category\_encoders as ce

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, f1\_score

import graphviz

import time

current\_time = int(time.time())

# Load the dataset

df = pd.read\_csv('/content/car\_evaluation.csv', header=None)

col\_names = ['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety', 'class']

df.columns = col\_names

# Define lists to store results

accuracy\_scores = []

f1\_scores = []

# Perform 20 iterations

for i in range(20):

    X = df.drop(['class'], axis=1)

    y = df['class']

    # Split data into train and test sets

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=i)

    # Encode categorical features

    encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety'])

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

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

    # Train Decision Tree classifier

    clf\_gini = DecisionTreeClassifier(criterion='gini')

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

    # Predict labels

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

    # Calculate accuracy and F1-score

    accuracy = accuracy\_score(y\_test, y\_pred\_gini)

    f1 = f1\_score(y\_test, y\_pred\_gini, average='weighted')

    # Append results to lists

    accuracy\_scores.append(accuracy)

    f1\_scores.append(f1)

# Calculate average accuracy and F1-score

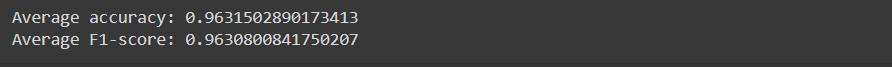
avg\_accuracy = np.mean(accuracy\_scores)

avg\_f1 = np.mean(f1\_scores)

# Print average accuracy and F1-score

print('Average accuracy:', avg\_accuracy)

print('Average F1-score:', avg\_f1)

****

**Problem statement – 4**

Repeat the steps 1, 2 and 3 considering 70 and 80 percent data (random selection) for training.

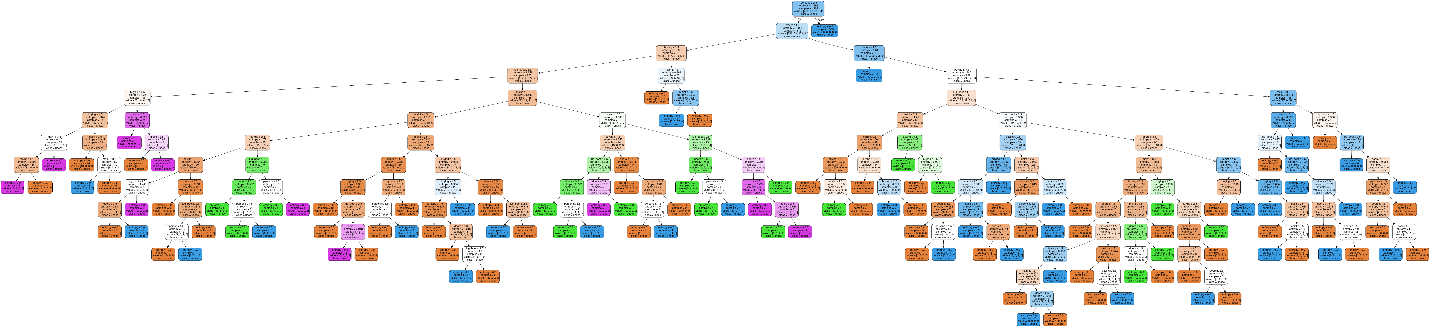
* Considering 70% data for training and 30% data for testing.

**STEP – 1**

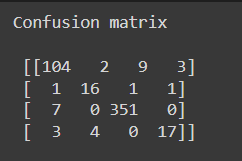
**Model accuracy with entropy as criterion**

****

**Tree**

****

**Confusion Matrix**

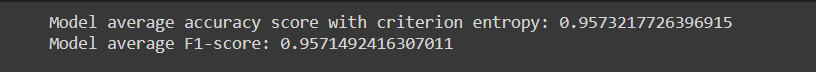
****

**F – Scores**

****

**STEP – 2**

Average accuracy score and average F – score of 20 iterations using entropy as criterion.

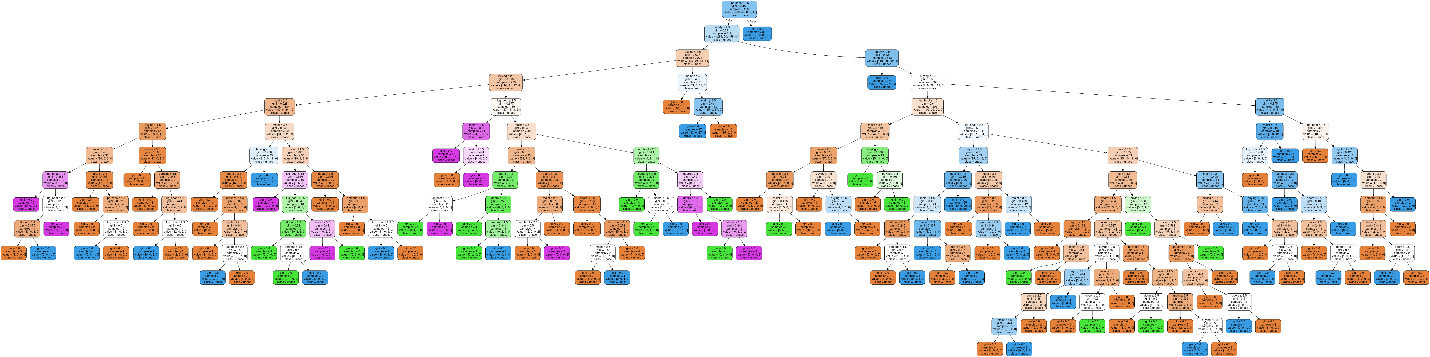


**STEP – 3**

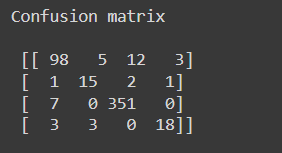
**Model accuracy using with Gini index as criterion**

****

**Tree**

****

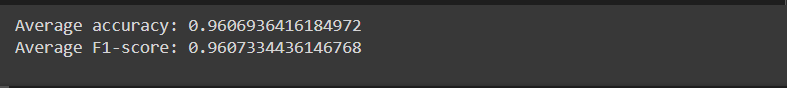
**Confusion Matrix**

****

**F – scores**

****

Average accuracy score and average F – score of 20 iterations using Gini index as criterion.



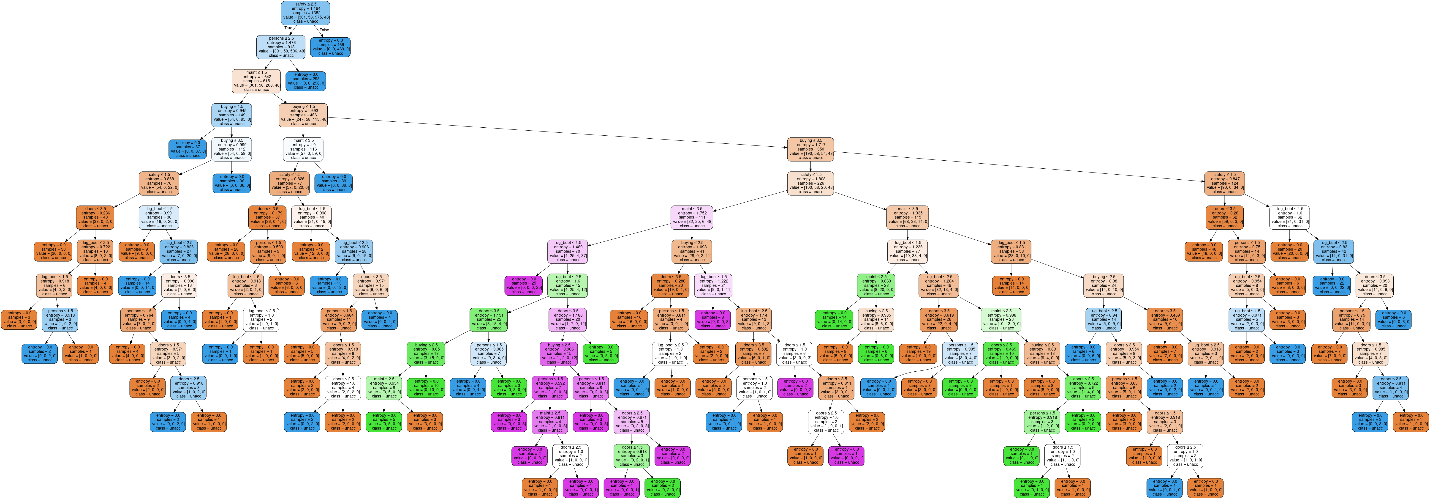
* Considering 80% data for training and 20% data for testing.

**STEP - 1**

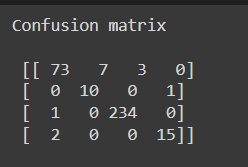
**Model accuracy with entropy as criterion**



**Tree**

****

**Confusion Matrix**

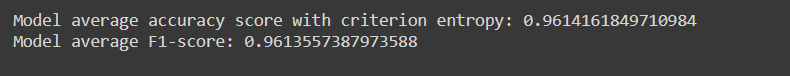
****

**F – scores**

****

**STEP – 2**

Average accuracy score and average F – score of 20 iterations using entropy as criterion.

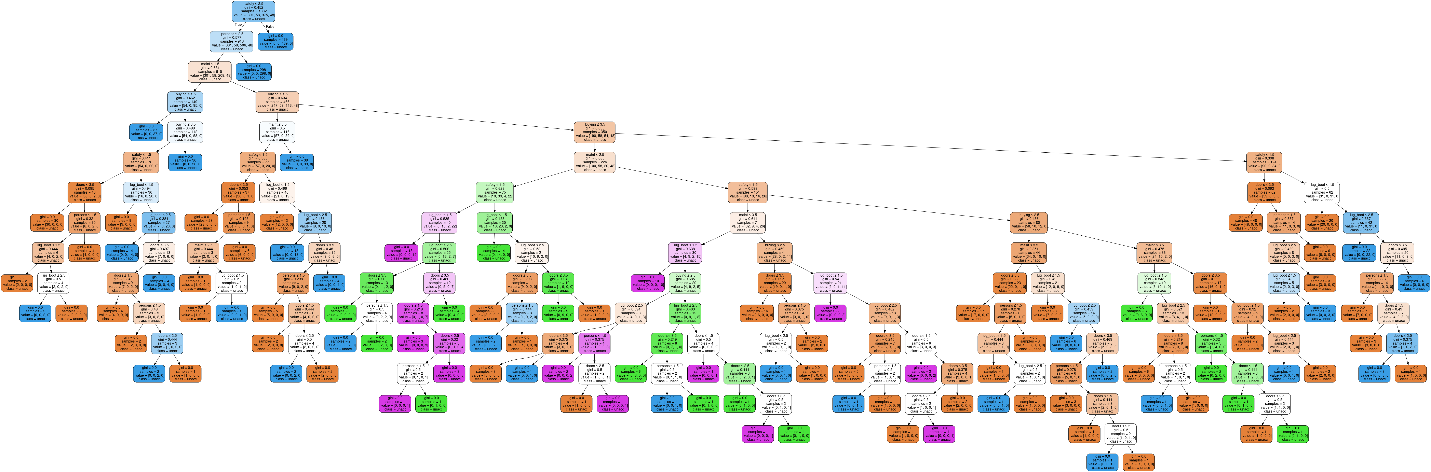
****

**STEP – 3**

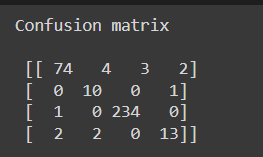
**Model accuracy using with Gini index as criterion**

****

**Tree**

****

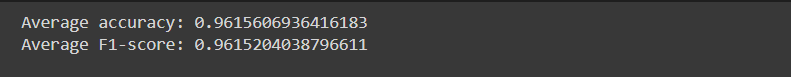
**Confusion Matrix**

****

**F – scores**

****

Average accuracy score and average F – score of 20 iterations using Gini index as criterion.

****