Exercise 4: Implement Decision Tree

Classification Datasets: You can use one of the two datasets (or optionally, both datasets).

(a) Car Evaluation dataset D1: Target attribute safety :{low, med, high}. https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

(b) Iris dataset D2: Target attribute class :{Iris Setosa, Iris Versicolour, Iris Virginica}. https://archive.ics.uci.edu/ml/datasets/Iris Implement Decision Tree. In this task you will implement a decision tree. As a starting point you can implement your complete decision tree model for classification tasks. You have to split data into two parts train and test (70% and 30% respectively). Using the train data you will build a decision tree. Use Cross Entropy as a Quality-criterion. You have to provide information on the learning process that includes.

1. Define an appropriate stopping criteria i.e. max depth gain is too small or reduction in cost is

small 2. At each decision step (or split) present the probability of each class using histogram (properly labeled figure) 3. At each decision step, plot the Cross Entropy of each attribute. 4. Note down the Information Gain at each new node created, you can store it in node structure or class. Display it at the end. 5. Print your tree using a breath first tree traversal. (you can also print node hierarchical level, information gain and decision rule, etc). 6. On a test set measure the cross entropy loss (i.e. logloss, note that this time problem is not binary classification).

In [ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import plot_tree

# Input: Dataset
iris = sns.load_dataset('iris')

target = iris['species']
iris1 = iris.copy()
iris1 = iris1.drop('species', axis =1)

# Defining the attributes
X = iris1
le = LabelEncoder()                # Label Encoding
target = le.fit_transform(target)
Y = target

# Dividing into test and training sets
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X , Y, test_size = 0.

# Create Decision Tree Classifier object
dtree = DecisionTreeClassifier(criterion = "entropy", max_depth = 5)

# Train the model using the training sets
dtree.fit(X_train,Y_train)

# Make predictions using the testing set
Y_pred = dtree.predict(X_test)
```

Output for the Decision Tree.

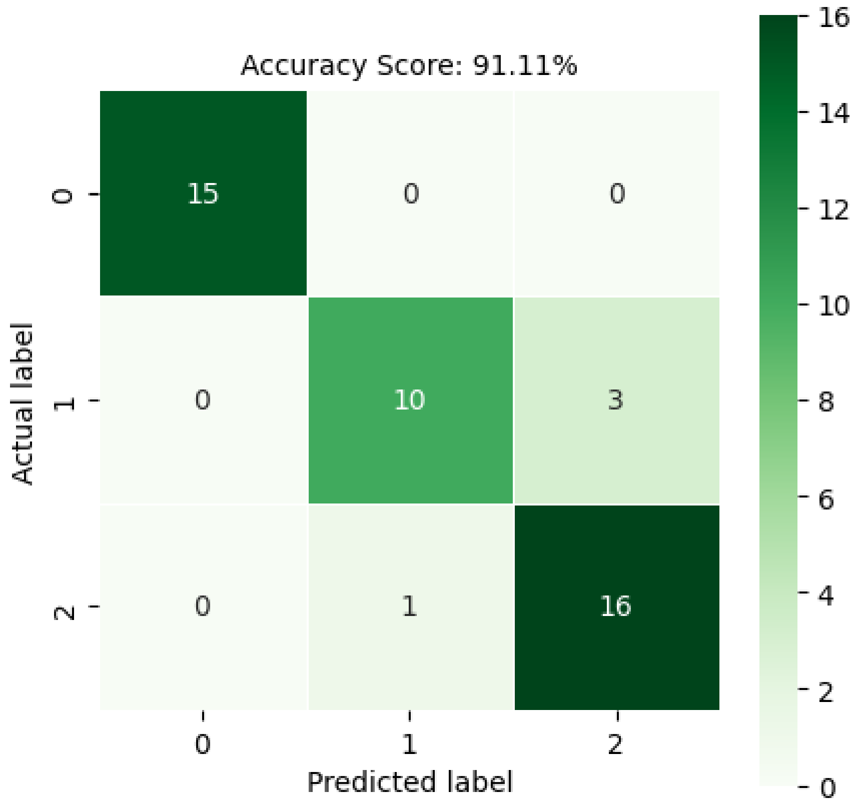
In [ ]:

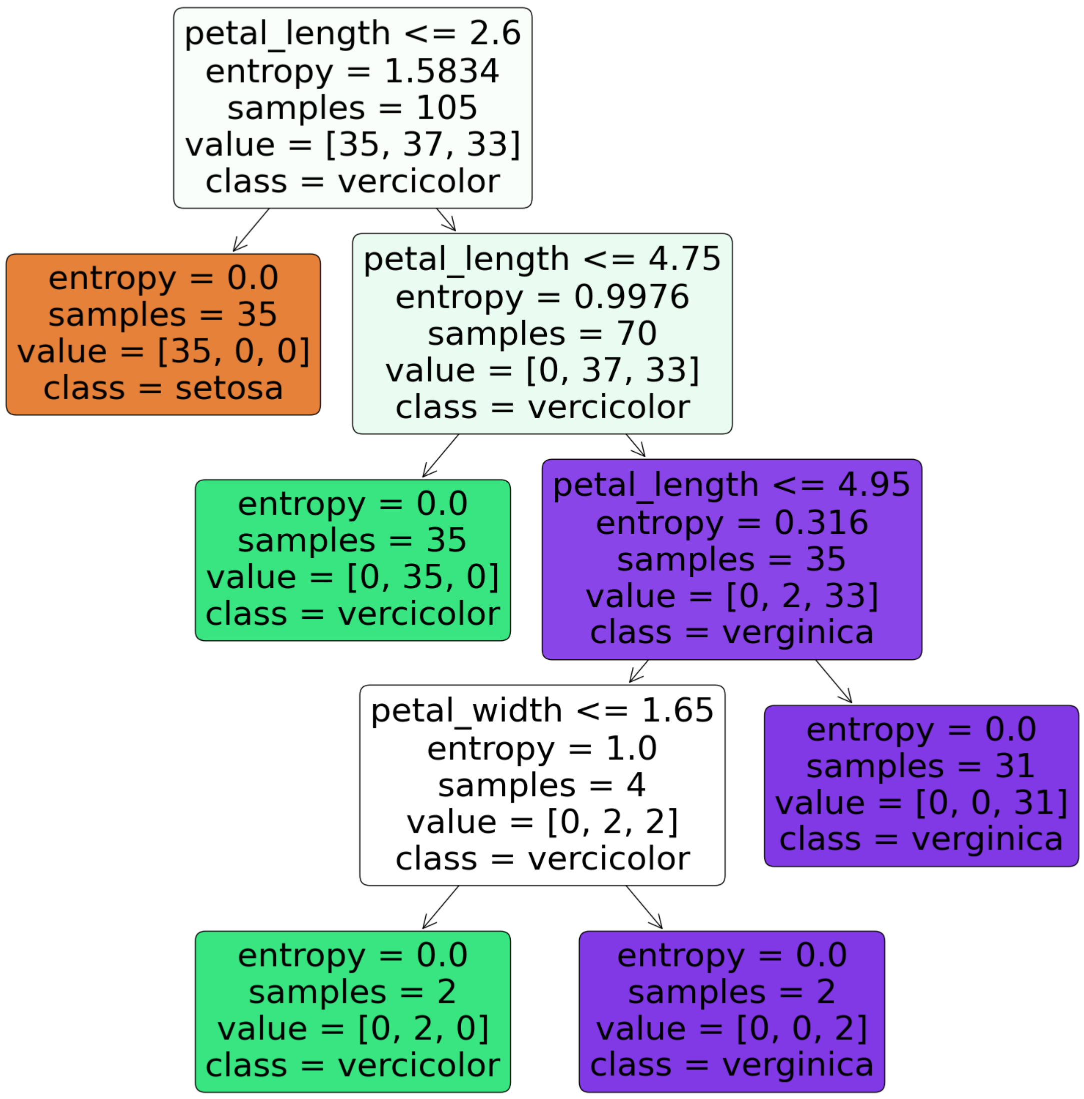
```
# Output: The Classification Report, Confusion Matrix and Decision Tree
print("Classification Report - \n", classification_report(Y_test,Y_pred))
cm = confusion_matrix(Y_test, Y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Gre
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0:.2f}%'.format(dtree.score(X_test,
plt.title(all_sample_title, size = 10)
plt.figure(figsize = (20,20))
dec_tree = plot_tree(decision_tree=dtree, feature_names = iris.columns,
                     class_names =["setosa", "vercicolor", "verginica"] ,
                     filled = True , precision = 4, rounded = True)
```

```
Classification Report -
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.91      0.77      0.83        13
           2       0.84      0.94      0.89        17

    accuracy                           0.91        45
   macro avg       0.92      0.90      0.91        45
weighted avg       0.91      0.91      0.91        45
```


Accuracy Score: 91.11%
15
0
0
0
10
3
0
1
16
Actual label
0
1
2
Predicted label
0
1
2
16
14
12
10
8
6
4
2
0


petal_length <= 2.6
entropy = 1.5834
samples = 105
value = [35, 37, 33]
class = vercicolor
entropy = 0.0
samples = 35
value = [35, 0, 0]
class = setosa
petal_length <= 4.75
entropy = 0.9976
samples = 70
value = [0, 37, 33]
class = vercicolor
entropy = 0.0
samples = 35
value = [0, 35, 0]
class = vercicolor
petal_length <= 4.95
entropy = 0.316
samples = 35
value = [0, 2, 33]
class = verginica
petal_width <= 1.65
entropy = 1.0
samples = 4
value = [0, 2, 2]
class = vercicolor
entropy = 0.0
samples = 31
value = [0, 0, 31]
class = verginica
entropy = 0.0
samples = 2
value = [0, 2, 0]
class = vercicolor
entropy = 0.0
samples = 2
value = [0, 0, 2]
class = verginica

On Implementation of Decision Tree Classifier over Iris Dataset with Maximum Depth of 5, we got an accuracy of approximately 98%. The Decision Tree shows the Entropy, Frequency of Classes and the Predicted Class for each node. Also the Information Gain can be computed for a node by subtracting its Entropy from the sum of Entropy of its Child Nodes.