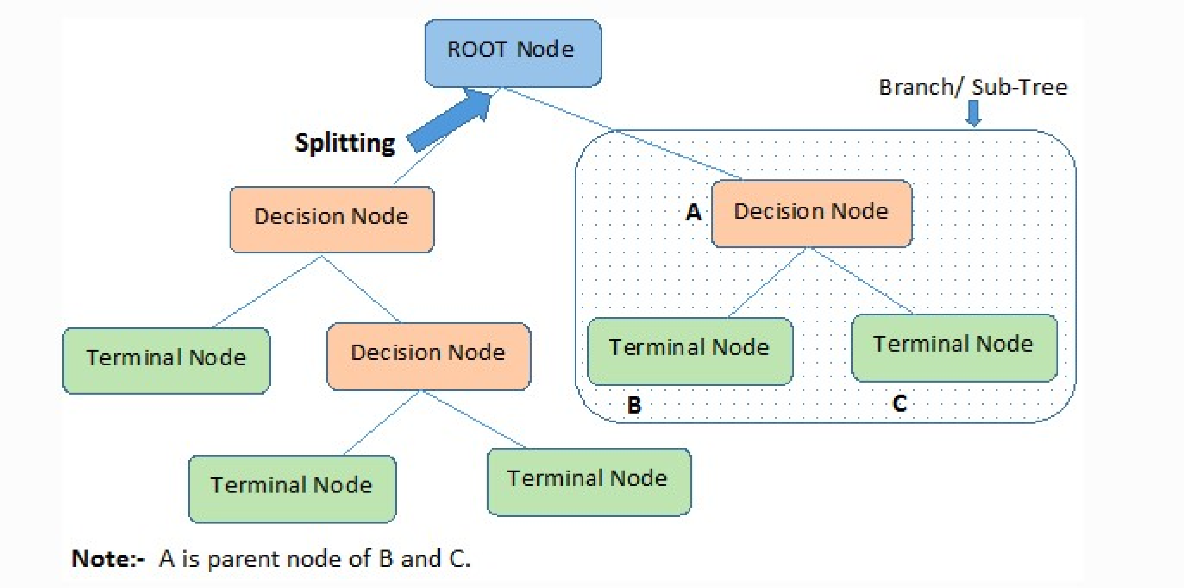
Decision Tree Classifier

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

In 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. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

Terminology Related to Decision Tree



How a Decision Tree Is Grown

The tree is built by splitting the records at each node according to a function of a single input field. The first task, therefore, is to decide which of the input fields makes the best split. The best split is defined as one that does the best job of separating the records into groups where a single class predominates in each group.

The measure used to evaluate a potential split is purity.

Measures of Calculating the Purity

A number of different measures are available to evaluate potential splits. Algorithms developed in the machine learning community focus on the increase in purity resulting from a split, while those developed in the statistics community focus on the statistical significance of the difference between the distributions of the child nodes. Alternate splitting criteria often lead to trees that look quite different from one another, but have similar performance. That is because there are usually many candidate splits with very similar performance. Different purity measures lead to different candidates being selected, but since all of the measures are trying to capture the same idea, the resulting models tend to behave similarly Purity measures for evaluating splits for categorical target variables include:

• Gini (also called population diversity)

• Entropy (also called information gain)

• Information gain ratio

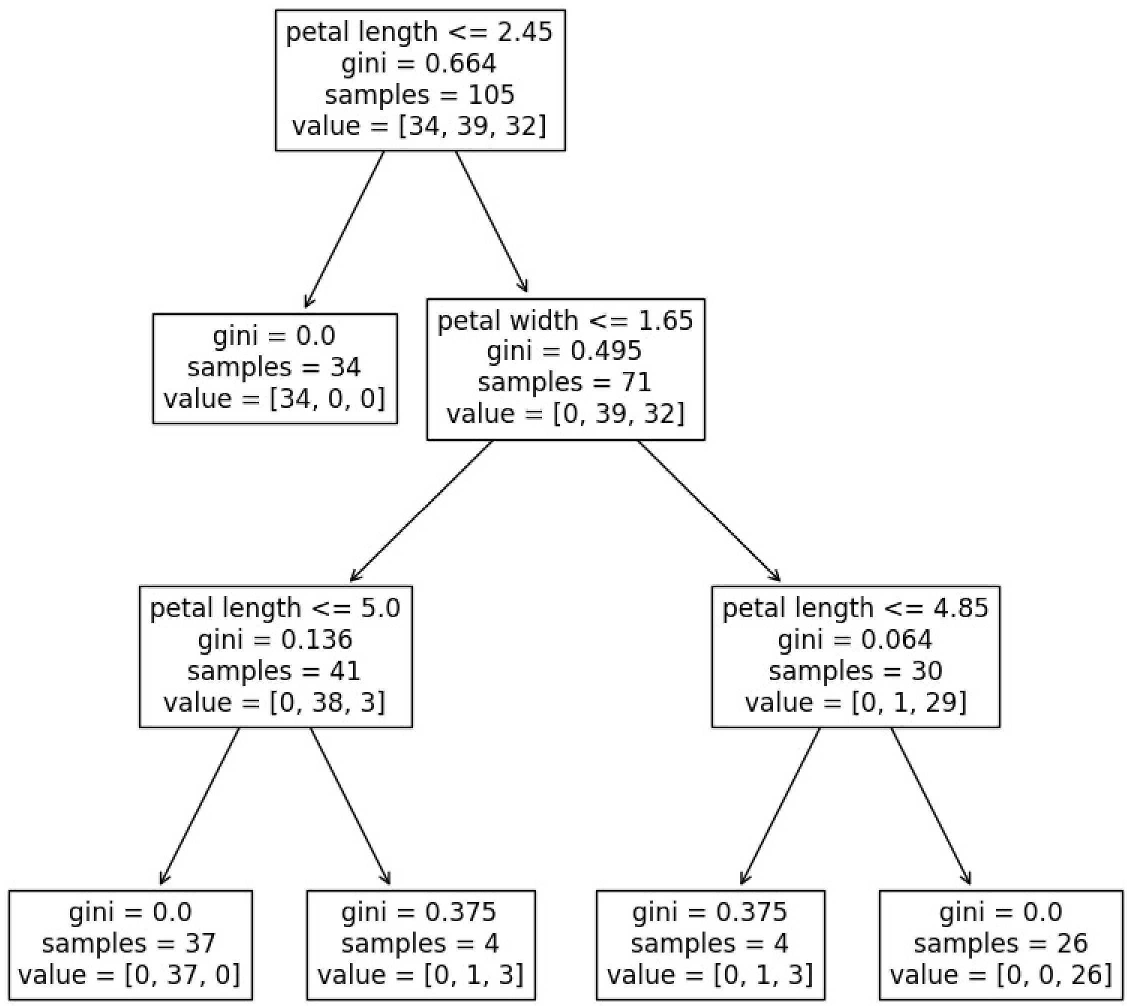
• Chi square test

In [24]:
```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn import tree
from matplotlib import pyplot as plt

names = ['Sepal Length', 'Sepal Width','Petal length','Petal Width','Species']
data = pd.read_csv('Iris.csv', names=names)
df = pd.DataFrame(data)
X = df.drop(['Species'], axis=1)
Y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, ran
```

In [25]:
```

#build decision tree
clf_gini = tree.DecisionTreeClassifier(criterion='gini', max_depth=4,min_sampl
#max_depth represents max level allowed in each tree, min_samples_leaf minumum
clf_entropy = tree.DecisionTreeClassifier(criterion='entropy', max_depth=4,min
#fit the tree to iris dataset
clf_gini.fit(X_train,y_train)
clf_entropy.fit(X_train,y_train)
```

Out[25]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_leaf=4)
```

In [26]:
```
#plot decision tree
fig, ax = plt.subplots(figsize=(10, 10)) #figsize value changes the size of pl
tree.plot_tree(clf_gini,ax=ax,feature_names=['sepal length','sepal width','pet
plt.show()
```



In [27]:
```python
# Function to make predictions
def prediction(X_test, clf_object):

    # Predicton on test with giniIndex
    y_pred = clf_object.predict(X_test)
    print("Predicted values on:")
    print(y_pred)
    return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):

    print("Confusion Matrix: ",
        confusion_matrix(y_test, y_pred))

    print ("Accuracy : ",
    accuracy_score(y_test,y_pred)*100)

    print("Report : ",
    classification_report(y_test, y_pred))
```

In [28]:
```python
y_pred_gini = prediction(X_test,clf_gini)
#y_pred_entropy = prediction(X_test,clf_entropy)
df_pred=pd.DataFrame()
df_pred['y_test']=y_test
df_pred['y_pred']=y_pred_gini
df_pred.head(n=10)
```

```
Predicted values on:
['Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa']
```

Out[28]:

	y_test	y_pred
129	Iris-virginica	Iris-virginica
12	Iris-setosa	Iris-setosa

In [29]:
```
cal_accuracy(y_test,y_pred_gini)
```

```
Confusion Matrix:  [[16  0  0]
 [ 0 10  1]
 [ 0  1 17]]
Accuracy :  95.55555555555556
Report :                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       0.91      0.91      0.91        11
 Iris-virginica       0.94      0.94      0.94        18

       accuracy                           0.96        45
      macro avg       0.95      0.95      0.95        45
   weighted avg       0.96      0.96      0.96        45
```

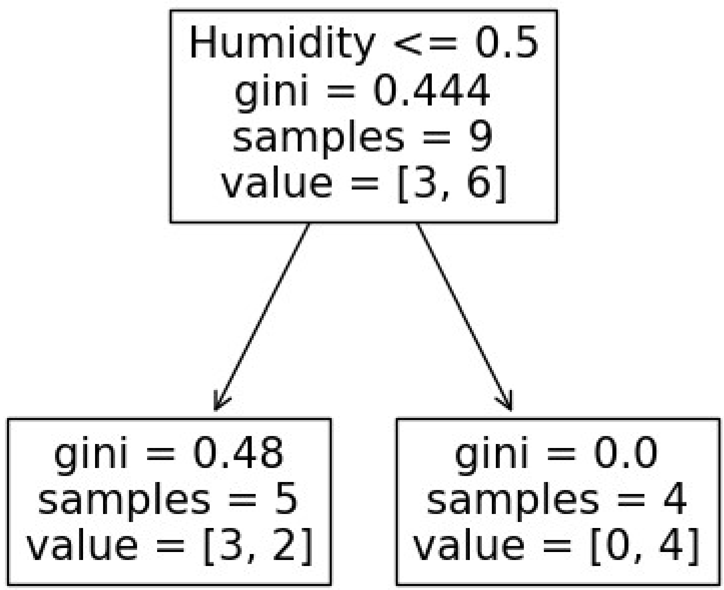
Example 2

In [30]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data = pd.read_csv('PlayTennis.csv')
df = pd.DataFrame(data)
print(df.head())
df['Outlook']= le.fit_transform(df['Outlook'])
df['Temperature']= le.fit_transform(df['Temperature'])
df['Humidity']= le.fit_transform(df['Humidity'])
df['Wind']= le.fit_transform(df['Wind'])
df['Play Tennis']= le.fit_transform(df['Play Tennis'])
X = df.drop(['Play Tennis'],axis=1)
y = df['Play Tennis']

print(df.head())
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_s

df.columns
```

```
    Outlook Temperature Humidity    Wind Play Tennis
0     Sunny         Hot     High    Weak          No
1     Sunny         Hot     High  Strong          No
2  Overcast         Hot     High    Weak         Yes
3      Rain        Mild     High    Weak         Yes
4      Rain        Cool   Normal    Weak         Yes
   Outlook  Temperature  Humidity  Wind  Play Tennis
0        2            1         0     1            0
1        2            1         0     0            0
2        0            1         0     1            1
3        1            2         0     1            1
4        1            0         1     1            1
```

Out[30]:
```
Index(['Outlook', 'Temperature', 'Humidity', 'Wind', 'Play Tennis'], dtype='o
bject')
```

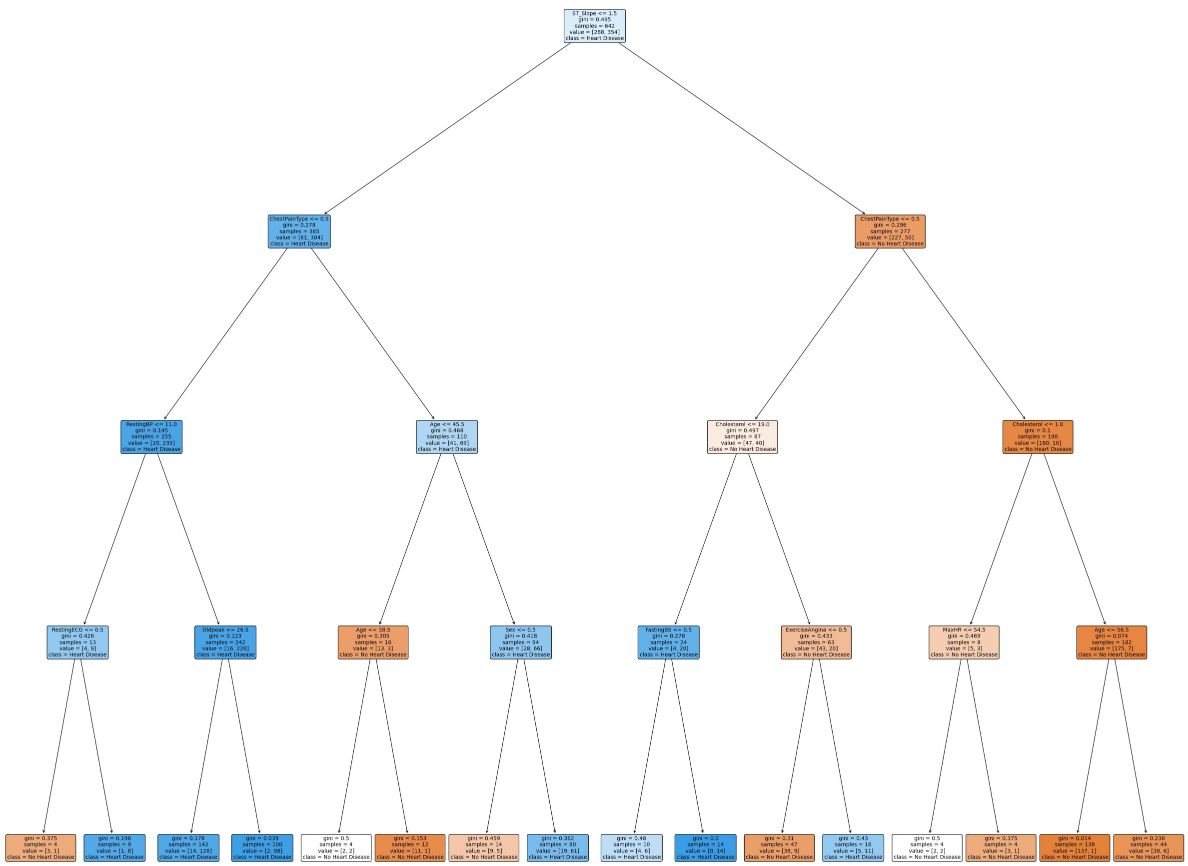
In [31]:
```
#build decision tree
clf_gini = tree.DecisionTreeClassifier(criterion='gini', max_depth=4,min_sampl
#max_depth represents max level allowed in each tree, min_samples_leaf minumum
clf_entropy = tree.DecisionTreeClassifier(criterion='entropy', max_depth=4,min
#fit the tree to iris dataset
clf_gini.fit(X_train,y_train)
clf_entropy.fit(X_train,y_train)

#plot decision tree
fig, ax = plt.subplots(figsize=(5, 5)) #figsize value changes the size of plot
tree.plot_tree(clf_gini,ax=ax,feature_names=['Outlook', 'Temperature', 'Humidi
plt.show()
```



Task 2

In [32]:
```
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn import tree
from matplotlib import pyplot as plt

data = pd.read_csv('heart.csv')
df = pd.DataFrame(data)
print(df.head())

le = LabelEncoder()
df['Sex'] = le.fit_transform(df['Sex'])
df['ChestPainType'] = le.fit_transform(df['ChestPainType'])
df['RestingBP'] = le.fit_transform(df['RestingBP'])
df['Cholesterol'] = le.fit_transform(df['Cholesterol'])
df['FastingBS'] = le.fit_transform(df['FastingBS'])
df['RestingECG'] = le.fit_transform(df['RestingECG'])
df['MaxHR'] = le.fit_transform(df['MaxHR'])
df['ExerciseAngina'] = le.fit_transform(df['ExerciseAngina'])
df['Oldpeak'] = le.fit_transform(df['Oldpeak'])
df['ST_Slope'] = le.fit_transform(df['ST_Slope'])
df['HeartDisease'] = le.fit_transform(df['HeartDisease'])

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando

clf_gini = tree.DecisionTreeClassifier(criterion='gini', max_depth=4, min_samp

clf_entropy = tree.DecisionTreeClassifier(criterion='entropy', max_depth=4, mi

clf_gini.fit(X_train, y_train)
clf_entropy.fit(X_train, y_train)

fig, ax = plt.subplots(figsize=(40, 35))
tree.plot_tree(clf_gini, ax=ax, feature_names=X.columns.tolist(), class_names=
plt.show()
```

```
   Age Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG  MaxHR  \
0   40   M           ATA        140          289          0     Normal    172
1   49   F           NAP        160          180          0     Normal    156
2   37   M           ATA        130          283          0         ST     98
3   48   F           ASY        138          214          0     Normal    108
4   54   M           NAP        150          195          0     Normal    122

  ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0              N      0.0       Up             0
1              N      1.0     Flat             1
2              N      0.0       Up             0
3              Y      1.5     Flat             1
4              N      0.0       Up             0
```

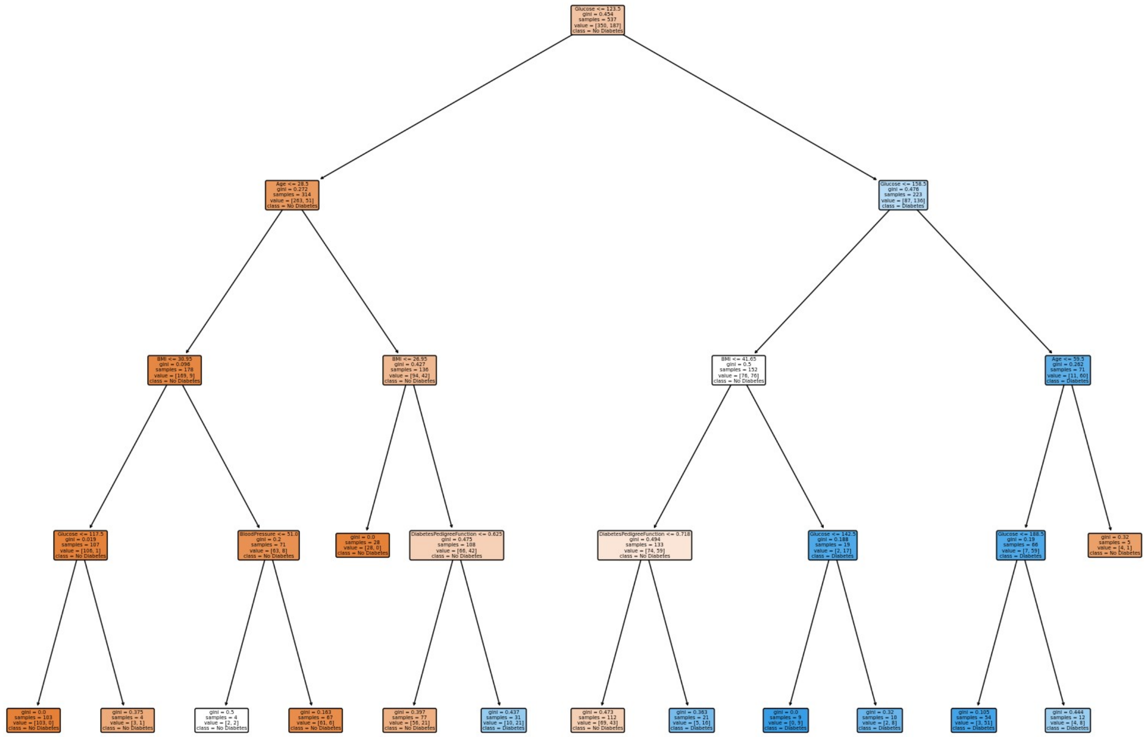


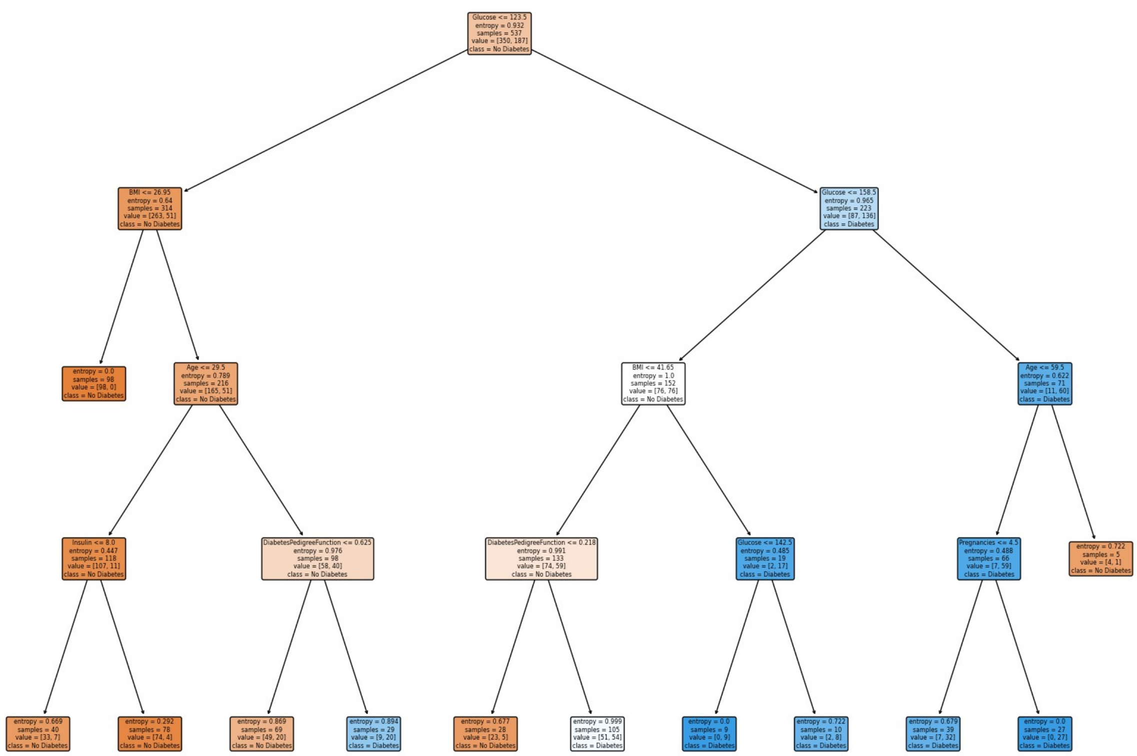
Task 3

In [33]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
import matplotlib.pyplot as plt

data = pd.read_csv('diabetes.csv')
df = pd.DataFrame(data)
print(df.head())

le = LabelEncoder()
df['Outcome'] = le.fit_transform(df['Outcome'])

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando

clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=4, min_samples_l

clf_entropy = DecisionTreeClassifier(criterion='entropy', max_depth=4, min_sam

clf_gini.fit(X_train, y_train)
clf_entropy.fit(X_train, y_train)

fig, ax = plt.subplots(figsize=(20, 15))
plot_tree(clf_gini, filled=True, feature_names=X.columns.tolist(), class_names
plt.show()

fig, ax = plt.subplots(figsize=(20, 15))
plot_tree(clf_entropy, filled=True, feature_names=X.columns.tolist(), class_na
plt.show()

accuracy_gini = clf_gini.score(X_test, y_test)
accuracy_entropy = clf_entropy.score(X_test, y_test)

print("Accuracy (Gini):", accuracy_gini)
print("Accuracy (Entropy):", accuracy_entropy)
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```






```
Accuracy (Gini): 0.70995670995671
Accuracy (Entropy): 0.6536796536796536
```