In [1]:
```python
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
import numpy as np
from statistics import mean
import matplotlib.pyplot as plt
```

In [2]:
```python
# INPUT_FILENAME     The name of the file that contains the data (CSV fo
# TRAINING_PART      The amount of data used to train the model
#                         (0.5=50% of observations for training; 50% for
# MINIMUMSPLIT       Controls the number of observations in each node
# MAX_DEPTH          Controls the number of nodes in the tree
# OUTPUT_COLUMN      The name of the column we'd like to predict
INPUT_FILENAME    = "banana_quality.csv"
TRAINING_PART     = 0.60
MAX_DEPTH         = 4
MINIMUMSPLIT      = 2000
OUTPUT_COLUMN     = 'Quality'
```

In [3]:
```python
#turning csv file to pandas dataframe & separating features and the labe
df = pd.read_csv(INPUT_FILENAME)
df = df.dropna(axis=0, how='any')

features = df.drop(columns = ['Size', OUTPUT_COLUMN])
target = df[OUTPUT_COLUMN]
print(features)
```

```
        Weight  Sweetness  Softness  HarvestTime  Ripeness   Acidity
0     0.468078   3.077832 -1.472177     0.294799  2.435570  0.271290
1     0.486870   0.346921 -2.495099    -0.892213  2.067549  0.307325
2     1.483176   1.568452 -2.645145    -0.647267  3.090643  1.427322
3     1.566201   1.889605 -1.273761    -1.006278  1.873001  0.477862
4     1.319199  -0.022459 -1.209709    -1.430692  1.078345  2.812442
...        ...        ...       ...          ...       ...       ...
7995  0.723565   1.134953  2.952763     0.297928 -0.156946  2.398091
7996 -2.217875  -2.812175  0.489249    -1.323410 -2.316883  2.113136
7997 -1.907665  -2.532364  0.964976    -0.562375 -1.834765  0.697361
7998 -2.742600  -1.008029  2.126946    -0.802632 -3.580266  0.423569
7999 -2.044666   0.159026  1.499706    -1.581856 -1.605859  1.435644

[8000 rows x 6 columns]
```

In [4]:
```python
#getting the dummy values of the dataframe
dummyFeatures = pd.get_dummies(features)
```

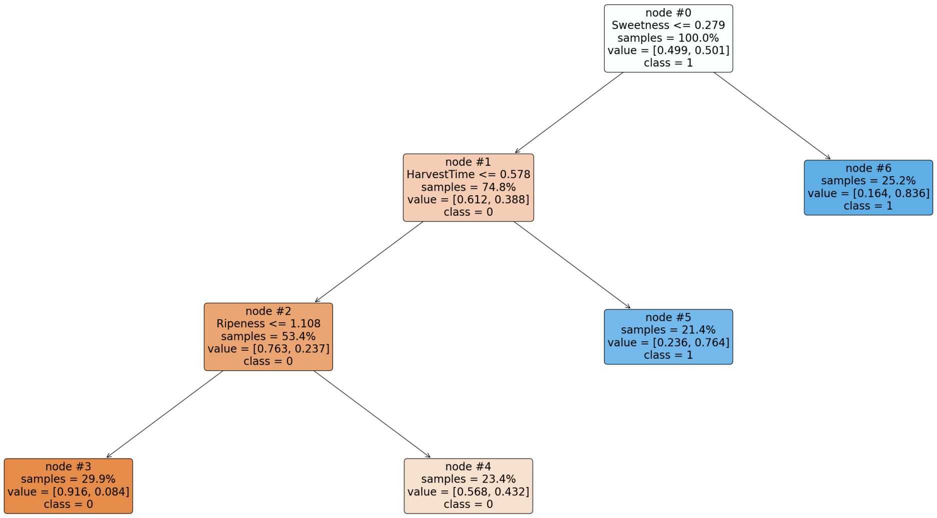
In [5]:
```
the dataset into a training and testing set
t,yTrain,yTest = train_test_split(dummyFeatures, target, train_size = TRA

rameters for decision tree
isionTreeClassifier(max_depth = MAX_DEPTH, min_samples_split = MINIMUMSPL

e tree to the training model
Train, yTrain)

s = list(dummyFeatures.columns)

lt.subplots(figsize = (40,20))
ree(dTree, node_ids = True, proportion = True, impurity = False, fontsize
```



In [6]:
```
#Getting predictions based on training and test sets
yTrainPred = dTree.predict(xTrain)
yTestPred = dTree.predict(xTest)

#evaluating the accuracy of each
trainAccuracy = accuracy_score(yTrainPred, yTrain)
testAccuracy = accuracy_score(yTestPred, yTest)
print(trainAccuracy, testAccuracy)
```

```
0.7814583333333334 0.7796875
```

In [7]:
```
# Generating Confusion Matrices for the training set:
predicted = yTrainPred
observed = yTrain
confusionMatrix = confusion_matrix(observed, predicted)

print(confusionMatrix)
```

```
[[1954  442]
 [ 607 1797]]
```

In [8]:
```
# Generating Confusion Matrices for the validation set:
predictedVal = yTestPred
observedVal = yTest
confusionMatrixVal = confusion_matrix(observedVal, predictedVal)

print(confusionMatrixVal)
```

```
[[1292  306]
 [ 399 1203]]
```

In [9]:
```
# Correct Classification Rate:
# Check whether there is a match between each predicted value (in pred) a
predRateTraining = mean(yTrainPred == yTrain)
predRateValidation = mean(yTestPred == yTest)
trainingPercentage = "{:.2%}".format(predRateTraining)
validationPercentage = "{:.2%}".format(predRateValidation)

print("The correct classification rate based on the training set is " +
print("The correct classification rate based on the validation set is " +
```

```
The correct classification rate based on the training set is 78.15%
The correct classification rate based on the validation set is 77.97%
```

In [ ]: