In [1]:

```
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]:

```
# INPUT_FILENAME     The name of the file that contains the data (CSV format)
# TRAINING_PART      The amount of data used to train the model
#                         (0.5=50% of observations for training; 50% for valida
# 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    = "titanic.csv"
TRAINING_PART     = 0.6
MAX_DEPTH         = 4
MINIMUMSPLIT      = 63
OUTPUT_COLUMN     = 'Survived'
```

In [3]:

```
#turning csv file to pandas dataframe & separating features and the label
df = pd.read_csv(INPUT_FILENAME)
df = df.dropna(axis=0, how='any')

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

```
     Male    Age   Fare
0       1  80.00  30.00
1       1  74.00   7.78
2       1  71.00  34.65
3       1  71.00  49.50
4       1  70.50   7.75
..    ...    ...    ...
709     1   0.83  18.75
710     0   0.75  19.26
711     0   0.75  19.26
712     1   0.67  14.50
713     1   0.42   8.52

[714 rows x 3 columns]
```

In [4]:

```
#getting the dummy values of the dataframe
dummyFeatures = pd.get_dummies(features)
```

In [5]:

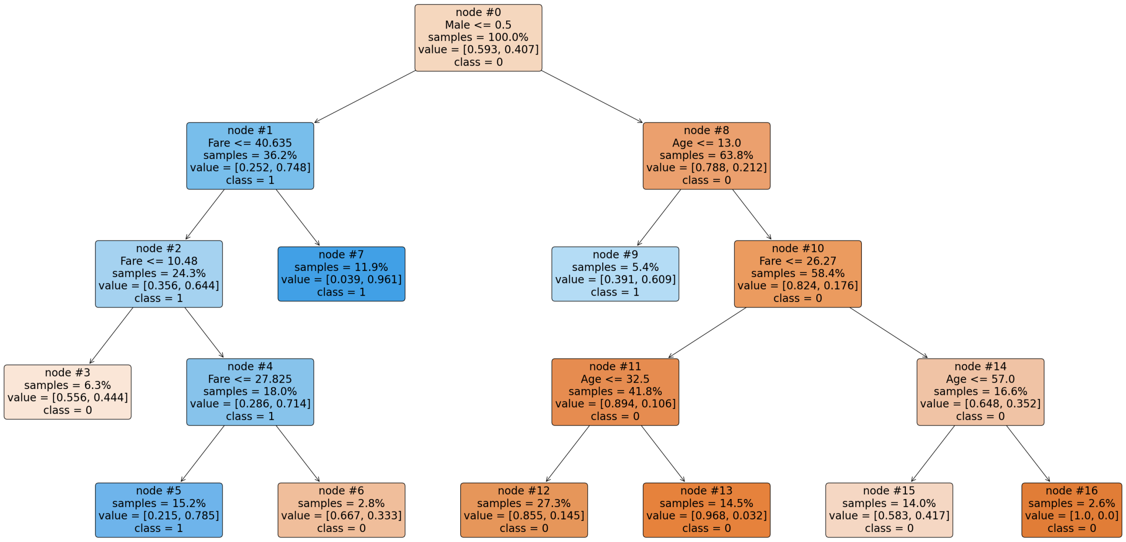
```
#splitting the dataset into a training and testing set
xTrain,xTest,yTrain,yTest = train_test_split(dummyFeatures, target, train_size

#setting parameters for decision tree
dTree = DecisionTreeClassifier(max_depth = MAX_DEPTH, min_samples_split = MININ

#fitting the tree to the training model
dTree.fit(xTrain, yTrain)

featureNames = list(dummyFeatures.columns)

fig, ax = plt.subplots(figsize = (40,20))
tree.plot_tree(dTree, node_ids = True, proportion = True, impurity = False, for
plt.show()
```



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)
```

```
0.8014018691588785 0.7657342657342657
```

In [7]:

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

print(confusionMatrix)
```

```
[[229  25]
 [ 60 114]]
```

In [8]:

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

print(confusionMatrixVal)
```

```
[[150  20]
 [ 47  69]]
```

In [9]:

```
# Correct Classification Rate:
# Check whether there is a match between each predicted value (in pred) and the
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 " + trainin
print("The correct classification rate based on the validation set is " + valid
```

```
The correct classification rate based on the training set is 80.14%
The correct classification rate based on the validation set is 76.57%
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: