In [75]:

```
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 DecisionTreeClassifier5
import pandas as pd
import numpy as np
from statistics import mean
import matplotlib.pyplot as plt
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[75], line 5
      3 from sklearn.metrics import accuracy_score, confusion_matrix
      4 from sklearn import datasets
----> 5 from sklearn.tree import DecisionTreeClassifier5
      6 import pandas as pd
      7 import numpy as np

ImportError: cannot import name 'DecisionTreeClassifier5' from 'sklearn.tree' (/Users/jennalali/anaconda3/lib/python3.10/site-pac
kages/sklearn/tree/__init__.py)
```

In [89]:

```
# 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 validation)
# 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    = "NeighborhoodFood.csv"
TRAINING_PART     = 0.60
MAX_DEPTH         = 4
MINIMUMSPLIT      = 45
OUTPUT_COLUMN     = 'ACCESS'
```

In [90]:

```
#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 = ['OBJECTID', OUTPUT_COLUMN])
target = df[OUTPUT_COLUMN]
print(features)
```

```
           GEOID10  0N_RESIDENTIAL  TOTAL_LPSS  LPSS_PER1000  \
0     4.210100e+11               0          25     30.674847   
1     4.210100e+11               0          21     28.806584   
2     4.210100e+11               0           7     17.114914   
3     4.210100e+11               0          15     19.480519   
4     4.210100e+11               0          17     25.914634   
...            ...             ...         ...           ...   
1331  4.210100e+11               0          19     24.547804   
1332  4.210100e+11               0          18      8.628955   
1333  4.210100e+11               0          25     33.377837   
1334  4.210100e+11               0          17     34.136546   
1335  4.210100e+11               0          22     48.034934   

      SUPERMARKET_ACCESS  PCT_VEHICLE_AVAILABILITY  TOTAL_RESTAURANTS  \
0                      1                 44.268775                  0   
1                      1                 67.611336                  2   
2                      0                 37.356322                  1   
3                      1                 52.824859                  0   
4                      1                 70.408163                  1   
...                  ...                       ...                ...   
1331                   1                 62.121212                  1   
1332                   1                 83.832335                 13   
1333                   0                 59.943182                  0   
1334                   0                 70.270270                  0   
1335                   0                100.000000                  0   

      PCT_POVERTY  HIGH_POVERTY   Shape__Area  
0       54.969325             1  275942.09770  
1       37.860082             1  176880.88280  
2       57.212714             1   74520.02734  
3       19.480519             0  185771.99220  
4       52.134146             1  242486.90630  
...           ...           ...           ...  
1331     9.948320             0  245228.17580  
1332     9.779482             0  407054.95310  
1333    16.688919             0  238290.52340  
1334     0.000000             0  315044.52340  
1335    14.410480             0  129360.87500  

[1336 rows x 10 columns]
```

In [91]:

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

In [92]:

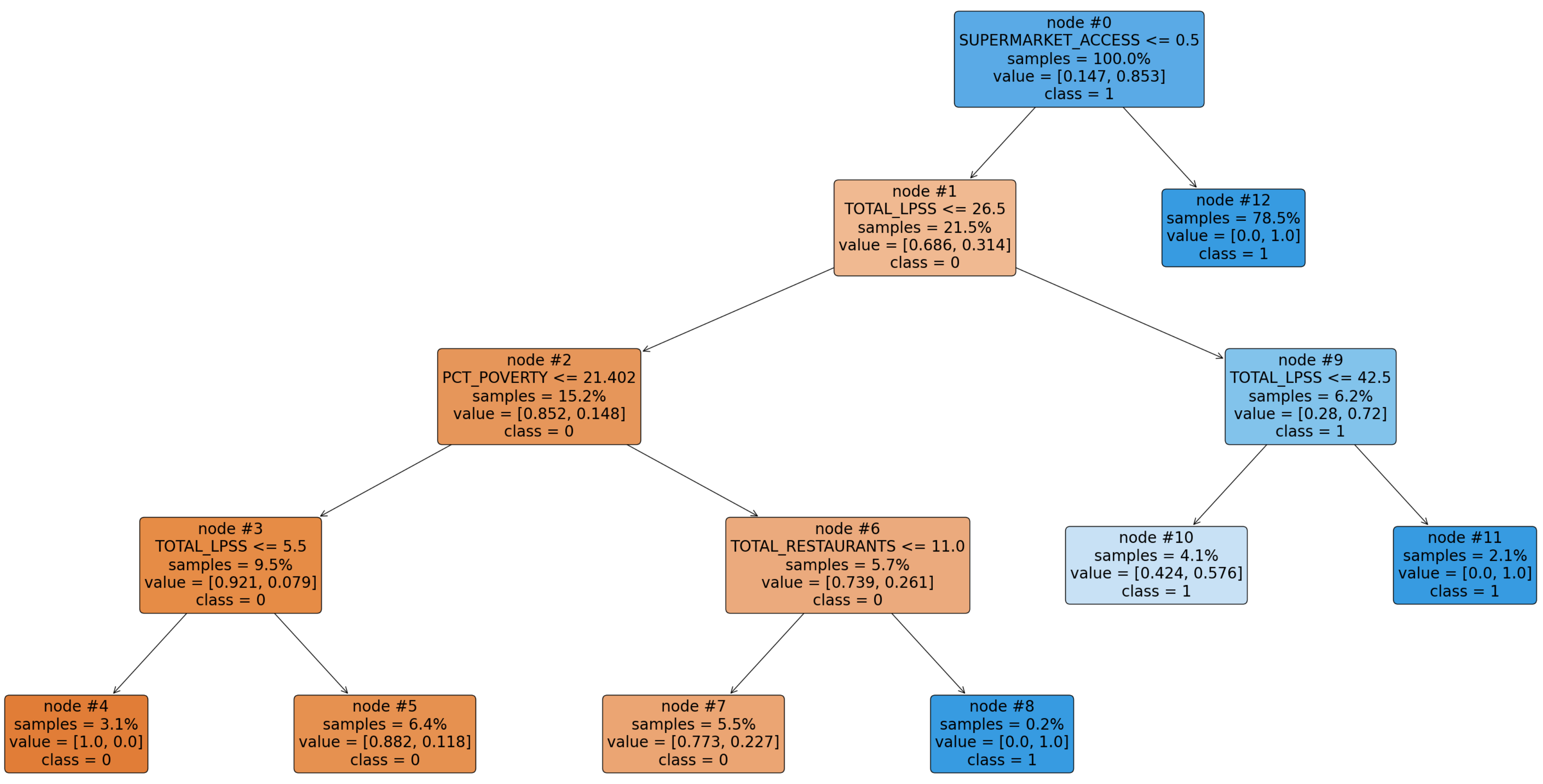
```
#splitting the dataset into a training and testing set
xTrain,xTest,yTrain,yTest = train_test_split(dummyFeatures, target, train_size = TRAINING_PART, random_state = 0, stratify = targe

#setting parameters for decision tree
dTree = DecisionTreeClassifier(max_depth = MAX_DEPTH, min_samples_split = MINIMUMSPLIT, random_state = 0)

#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, fontsize=20, feature_names = featureNames, class_names
plt.show()
```



In [93]:

```
#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)
```

```
0.9625468164794008 0.9626168224299065
```

In [94]:

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

print(confusionMatrix)
```

```
[[104  14]
 [ 16 667]]
```

In [95]:

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

print(confusionMatrixVal)
```

```
[[ 65  14]
 [  6 450]]
```

In [96]:

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

```
The correct classification rate based on the training set is 96.25%
The correct classification rate based on the validation set is 96.26%
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: