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 valid
# 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    = "fatal_crashes.csv"
TRAINING_PART     = 0.60
MAX_DEPTH         = 4
MINIMUMSPLIT      = 90
OUTPUT_COLUMN     = 'Arrest'
```

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 = [OUTPUT_COLUMN])
target = df[OUTPUT_COLUMN]
print(features)
```

```
     year  district  dc_number                  date_   age sex hit_____ru
\
0    2023        19      14086  2023-03-14 04:00:00+00  58.0   F        Yes
1    2023        35      17583  2023-03-20 04:00:00+00  73.0   M         No
2    2023        25      16759  2023-03-21 04:00:00+00  20.0   M         No
3    2023        12      18750  2023-03-26 04:00:00+00  41.0   M         No
4    2023        12      19204  2023-03-28 04:00:00+00  27.0   M        Yes
..    ...       ...        ...                       ...   ...  ..        ...
653  2023        19      78018  2023-12-20 05:00:00+00  24.0   M         No
654  2023         8      43257  2023-12-23 05:00:00+00  19.0   M         No
655  2023        19      79022  2023-12-27 05:00:00+00  47.0   M        Yes
656  2023        35      81667  2023-12-27 05:00:00+00  84.0   F        Yes
657  2023         8      40276  2023-11-26 05:00:00+00  70.0   F        Yes

     veh1          veh2        dc_key    point_x    point_y        lat  \
0    Auto    Pedestrian  2.023190e+11 -75.219933  39.973742  39.973742
1    Auto    Pedestrian  2.023350e+11 -75.142990  40.038468  40.038468
2    Auto          Auto  2.023250e+11 -75.116964  40.026016  40.026016
3    Auto          Auto  2.023120e+11 -75.246887  39.918719  39.918719
4    Auto    Pedestrian  2.023120e+11 -75.246310  39.917185  39.917185
..    ...           ...           ...        ...        ...        ...
653  Auto    Pedestrian  2.023190e+11 -75.221131  40.003731  40.003731
654  Auto  Fixed object  2.023080e+11 -74.997236  40.080032  40.080032
655  Auto    Pedestrian  2.023190e+11 -75.225372  39.979449  39.979449
656  Auto          Auto  2.023350e+11 -75.119456  40.044490  40.044490
657  Auto    Pedestrian  2.023080e+11 -74.977081  40.081065  40.081065

          lng
0   -75.219933
1   -75.142990
2   -75.116964
3   -75.246887
4   -75.246310
..         ...
653 -75.221131
654 -74.997236
655 -75.225372
656 -75.119456
657 -74.977081

[634 rows x 14 columns]
```

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

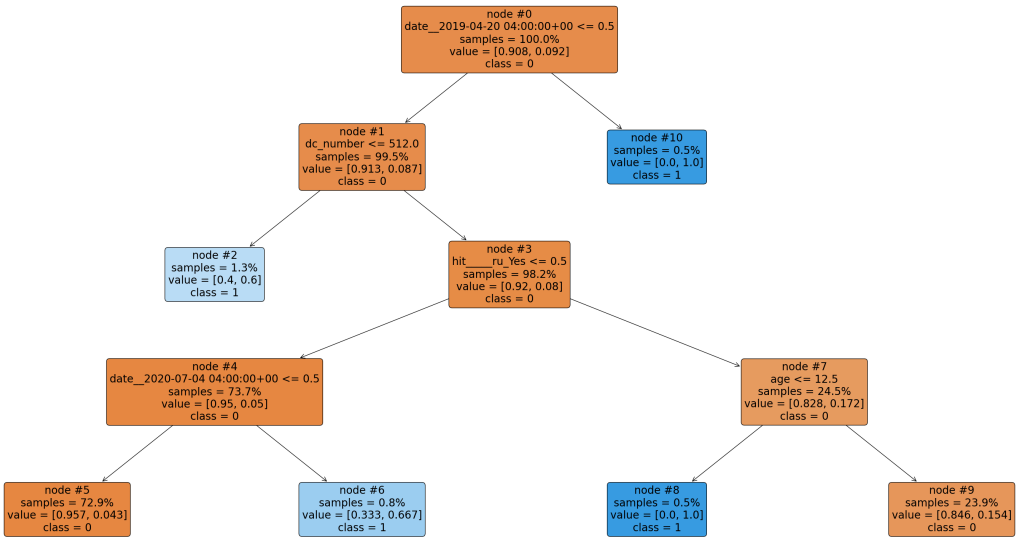
In [5]:
```
#splitting the dataset into a training and testing set
xTrain,xTest,yTrain,yTest = train_test_split(dummyFeatures, target, train_size

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

#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, fo
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)
```

```
0.9236842105263158 0.9133858267716536
```

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

print(confusionMatrix)
```

```
[[342   3]
 [ 26   9]]
```

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

print(confusionMatrixVal)
```

```
[[227   3]
 [ 19   5]]
```

In [9]:
```
# Correct Classification Rate:
# Check whether there is a match between each predicted value (in pred) and th
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 " + traini
print("The correct classification rate based on the validation set is " + vali
```

```
The correct classification rate based on the training set is 92.37%
The correct classification rate based on the validation set is 91.34%
```

In [ ]:

In [ ]:

In [ ]: