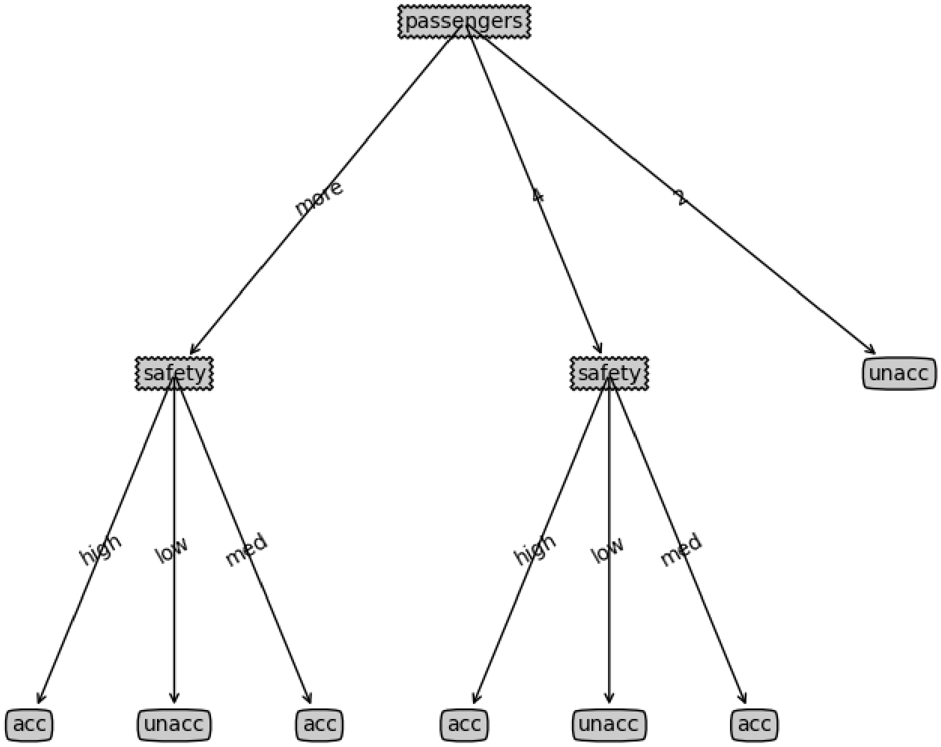
Assignment 4 - Report

1.) The tree drawn based on the output obtained from the Car Dataset using the algorithm.

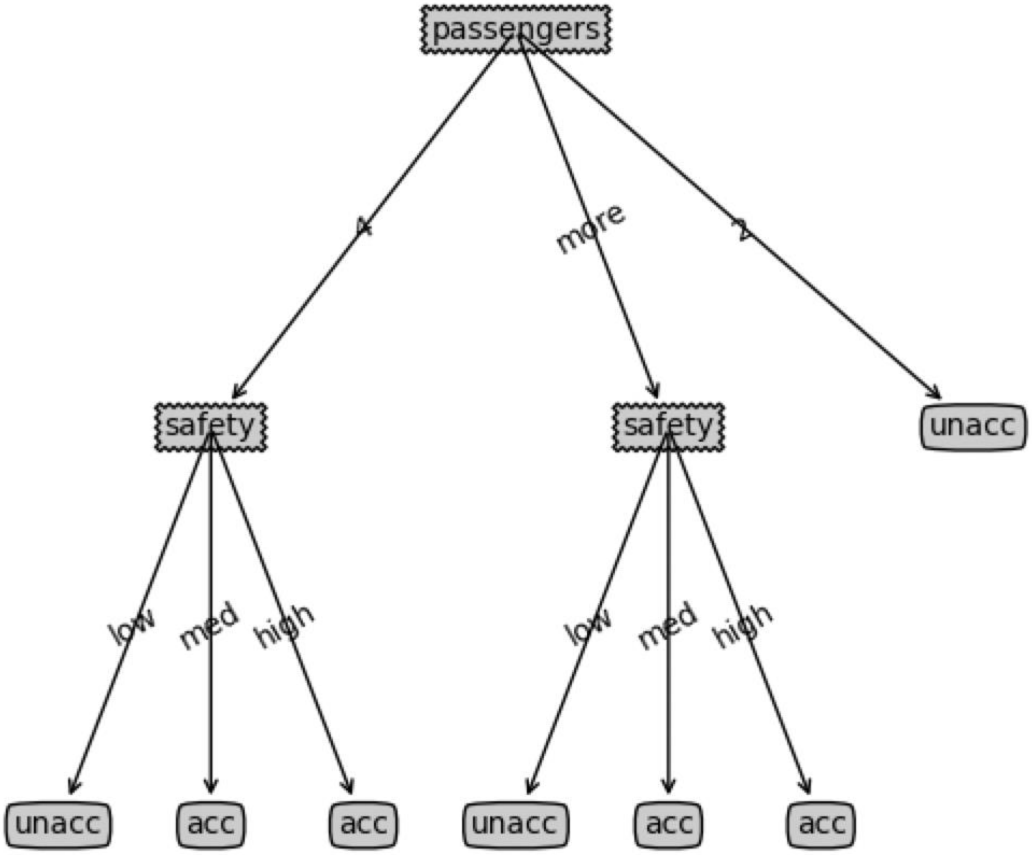


2.) The code of the functions that were implemented:

```
import treeplot
import numpy as np

# Function to implement Gini Index Calculations:

def gini_index(Splitting_Feature):
    """
    Given the observations of a Feature, calculating the GINI index (measure for impurity)
    """

    observations = list()

    for unq_values in np.unique(Splitting_Feature):
        y_count = 0
        for values in Splitting_Feature:
            if values == unq_values:     # example: Unique Attribute Values for Labels: Yes, No
                y_count = y_count + 1
        observations.append(y_count)

    if(len(np.unique(Splitting_Feature)) != 1):
        n = sum(observations)
        p1 = observations[0]/n
        p2 = observations[1]/n
        gini_ind = (1.0 - ((p1**2)+(p2**2)))

    else:
        n = sum(observations)
        p1 = observations[0]/n
        gini_ind = (1.0 - p1**2)

    return gini_ind


def chooseBestFeature(dataSet):
    '''
    choose best feature to split based on Gini index

    Parameters
    -----------------
    dataSet: 2-D list
        [n_sampels, m_features + 1]
        the last column is class label

    Returns
    ------------------
    bestFeatId: int
        index of the best feature
    '''

    #TODO

    classlabels = list()
    classlabels = [row[len(dataSet[0])-1] for row in dataSet]

    gini_classlabels = gini_index(classlabels)

    # Initialization
    InfoGain = list()
    bestFeatId = 999
    bestInfoGain = -1

    for index in range(len(dataSet[0])-1):
        feature = list()      # contains attribute values
        gini_ind = list()     # to store the required gini index values accordingly

        for row in dataSet:
            feature.append(row[index])

        n = len(feature)      # no. of values

        for fval in np.unique(feature):      # Consider only unique attribute values
            value = list()     # contains unique attribute values
            value.append(fval)
            subset1 = list()      # to find the subset based on the given axis and feature values
            value = set(value)

            for row in dataSet:
                if value.issubset(row):
                    subset1.append(row[len(dataSet[0])-1])

            n1 = len(subset1)     # no. of values in subset

            gini = gini_index(subset1)
            gini_ind.append((n1/n)* gini)
            gini_feature = sum(gini_ind)
            InfoGain.append(gini_classlabels - gini_feature)

        bestFeatId = InfoGain.index(max(InfoGain))
        bestInfoGain = max(InfoGain)

        # Find best gain and corresponding feature ID
        return bestFeatId


    def stopCriteria(dataSet):
        '''
        Criteria to stop splitting:
        1) if all the classe labels are the same, then return the class label;
        2) if there are no more features to split, then return the majority label of the subset.

        Parameters
        -----------------
        dataSet: 2-D list
            [n_sampels, m_features + 1]
            the last column is class label

        Returns
        ------------------
        assignedLabel: string
            if satisfying stop criteria, assignedLabel is the assigned class label;
            else, assignedLabel is None
        '''
        assignedLabel = None
        # TODO

        no_of_columns = len(dataSet[0])
        classlabels = []
        classlabels = [row[no_of_columns - 1] for row in dataSet]


        # A set cannot have duplicates.
        # So if all the elements in the original list are identical,
        # the set will have just one element.
        if len(set(classlabels)) == 1:
            assignedLabel = classlabels[0]

        # Finding Feature Space:
        bestFeatId = chooseBestFeature(dataSet)
        features = [index for index in range(len(dataSet[0])-1) if index != bestFeatId]

        if len(features) == 0:  # Implies: no more features to split
            assignedLabel = max(set(classlabels), key = classlabels.count)   # Finding Mode of the classlabels list - Python: Naive Approach

        return assignedLabel
```

Tree:

Output:


passengers
4
more
2
safety
safety
unacc
low
med
high
low
med
high
unacc
acc
acc
unacc
acc
acc