Assignment 4:



A diagram of weather forecast

Description automatically generated

1. Implemented code:

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

    '''

    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

    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

1. Car data set:

A diagram of a safety system

Description automatically generated