**Experiment No.7**

**Title:** Attribute subset selection

|  |
| --- |
|  |

**Batch: A4 Roll No.: 1914078 Experiment No.: 7**

**Aim:** Attribute subset selection

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

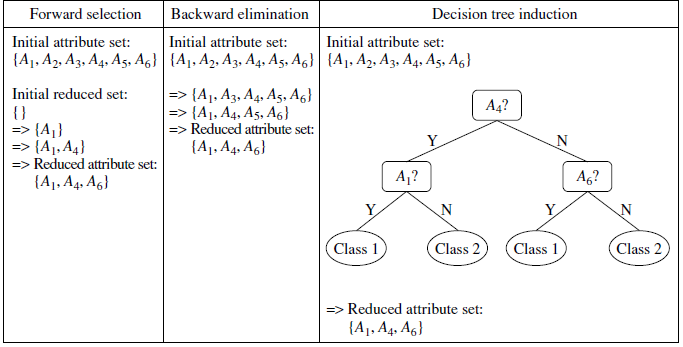
**Resources needed:** Any programming language, any data source (RDBMS/Excel/CSV)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Theory:**

While applying different data mining techniques such as clustering and classification, it is essential to check for the relevant attributes which will contribute in the analysis process. Attribute subset section is a process of extracting such attributes from the data set.

There are several methods that can be used for attribute subset selection mention in below given table [1]:



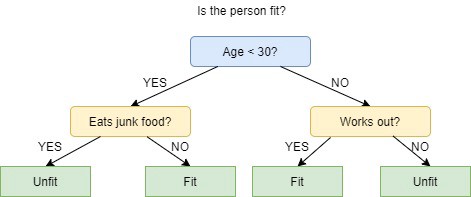
**Figure1. Greedy (heuristic) methods for attribute subset selection [1]**

In Stepwise forward selection, first, it considers an empty set of attributes as the reduced set. The best of the original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.

In Stepwise backward elimination, it starts with the full set of attributes and at each step, it removes the worst attribute remaining in the set. These techniques, forward selection and backward elimination can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.

Decision tree is a structure that contains nodes (rectangular boxes) and edges(arrows) and is built from a dataset (table of columns representing features/attributes and rows corresponds to records). Each node is either used to make a decision (known as decision node) or represent an outcome (known as leaf node).

**Example of Decision Tree:** The picture above depicts a decision tree that is used to classify whether a person is **Fit**or **Unfit.** The decision nodes here are questions like ‘’*‘Is the person less than 30 years of age?’*, *‘Does the person eat junk?’*, etc.andthe leaves are one of the two possible outcomes viz. **Fit**and **Unfit**. Looking at the Decision Tree we can say make the following decisions: if a person is less than 30 years of age and doesn’t eat junk food then he is Fit, if a person is less than 30 years of age and eats junk food then he is Unfit and so on.



In Decision tree induction approach, Decision tree algorithms (e.g., ID3, C4.5, and CART)

which are intended for classification can be used for select the important attributes, it constructs a flowchart like structure where each internal (nonleaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. The other attributes which are not part of the tree can be discarded.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Procedure / Approach /Algorithm / Activity Diagram:**

Implement the Decision tree induction approach (ID3) to identify the subset of attributes for the dataset.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Results: (Program printout with output / Document printout as per the format)**

import numpy as np

import pandas as pd

eps = np.finfo(float).eps

from numpy import log2 as log

outlook = 'overcast,overcast,overcast,overcast,rainy,rainy,rainy,rainy,rainy,sunny,sunny,sunny,sunny,sunny'.split(',')

temp = 'hot,cool,mild,hot,mild,cool,cool,mild,mild,hot,hot,mild,cool,mild'.split(',')

humidity = 'high,normal,high,normal,high,normal,normal,normal,high,high,high,high,normal,normal'.split(',')

windy = 'FALSE,TRUE,TRUE,FALSE,FALSE,FALSE,TRUE,FALSE,TRUE,FALSE,TRUE,FALSE,FALSE,TRUE'.split(',')

play = 'yes,yes,yes,yes,yes,yes,no,yes,no,no,no,no,yes,yes'.split(',')

dataset ={'outlook':outlook,'temp':temp,'humidity':humidity,'windy':windy,'play':play}

df = pd.DataFrame(dataset,columns=['outlook','temp','humidity','windy','play'])

df

****

entropy\_node = 0

values = df.play.unique()

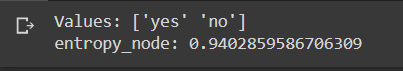
for value in values:

    fraction = df.play.value\_counts()[value]/len(df.play)

    entropy\_node += -fraction\*np.log2(fraction)

print(f'Values: {values}')

print(f'entropy\_node: {entropy\_node}')

****

def ent(df,attribute):

    target\_variables = df.play.unique()

    variables = df[attribute].unique()

    entropy\_attribute = 0

    for variable in variables:

        entropy\_each\_feature = 0

        for target\_variable in target\_variables:

            num = len(df[attribute][df[attribute]==variable][df.play ==target\_variable])

            den = len(df[attribute][df[attribute]==variable])

            fraction = num/(den+eps)

            entropy\_each\_feature += -fraction\*log(fraction+eps)

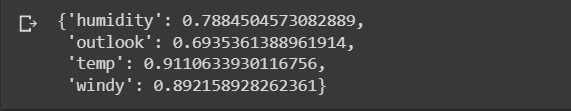
        fraction2 = den/len(df)

        entropy\_attribute += -fraction2\*entropy\_each\_feature

    return(abs(entropy\_attribute))

a\_entropy = {k:ent(df,k) for k in df.keys()[:-1]}

a\_entropy

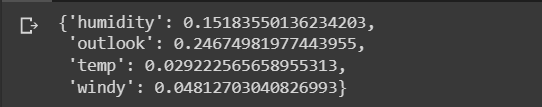
****

def ig(e\_dataset,e\_attr):

    return(e\_dataset-e\_attr)

IG = {k:ig(entropy\_node,a\_entropy[k]) for k in a\_entropy}

IG

****

def find\_entropy(df):

    Class = df.keys()[-1]

    entropy = 0

    values = df[Class].unique()

    for value in values:

        fraction = df[Class].value\_counts()[value]/len(df[Class])

        entropy += -fraction\*np.log2(fraction)

    return entropy

def find\_entropy\_attribute(df,attribute):

  Class = df.keys()[-1]

  target\_variables = df[Class].unique()

  variables = df[attribute].unique()

  entropy2 = 0

  for variable in variables:

      entropy = 0

      for target\_variable in target\_variables:

          num = len(df[attribute][df[attribute]==variable][df[Class] ==target\_variable])

          den = len(df[attribute][df[attribute]==variable])

          fraction = num/(den+eps)

          entropy += -fraction\*log(fraction+eps)

      fraction2 = den/len(df)

      entropy2 += -fraction2\*entropy

  return abs(entropy2)

def find\_winner(df):

    Entropy\_att = []

    IG = []

    for key in df.keys()[:-1]:

        IG.append(find\_entropy(df)-find\_entropy\_attribute(df,key))

    return df.keys()[:-1][np.argmax(IG)]

def get\_subtable(df, node,value):

  return df[df[node] == value].reset\_index(drop=True)

def buildTree(df,tree=None):

    Class = df.keys()[-1]

    node = find\_winner(df)

    attValue = np.unique(df[node])

    if tree is None:

        tree={}

        tree[node] = {}

    for value in attValue:

        subtable = get\_subtable(df,node,value)

        clValue,counts = np.unique(subtable[Class],return\_counts=True)

        if len(counts)==1:

            tree[node][value] = clValue[0]

        else:

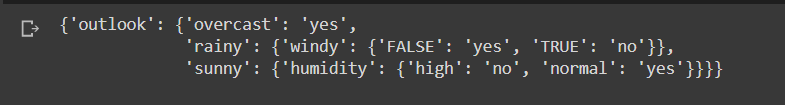
            tree[node][value] = buildTree(subtable)

    return tree

t  = buildTree(df)

import pprint

pprint.pprint(t)

****

**Questions:**

1. What are the benefits of applying the attribute subset section method while analyzing the data?

**Answer:**

The goal of attribute subset selection is to find a minimum set of attributes such that dropping of those irrelevant attributes does not much affect the utility of data and the cost of data analysis could be reduced. Mining on a reduced data set also makes the discovered pattern easier to understand.

1. Briefly describe how chi Merge Works. Implement chi Merge Algorithm for data discretization (for IRIS dataset) for each of the four numeric attributes. Take stopping criteria be max interval = 6.

**Answer:**

import pandas as pd

from collections import Counter

import numpy as np

import matplotlib.pyplot as plt

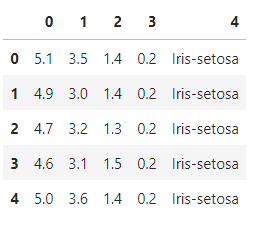
from tabulate import tabulate

import warnings

warnings.filterwarnings('ignore')

iris = pd.read\_csv('iris.data', header=None)

iris.head()

****

class Discretization:

    ''' A process that transforms quantitative data into qualitative data '''

    def \_\_init\_\_(cls):

        print('Data discretization process started')

    def get\_new\_intervals(cls, intervals, chi, min\_chi):

        ''' To merge the interval based on minimum chi square value '''

        min\_chi\_index = np.where(chi == min\_chi)[0][0]

        new\_intervals = []

        skip = False

        done = False

        for i in range(len(intervals)):

            if skip:

                skip = False

                continue

            if i == min\_chi\_index and not done:

                t = intervals[i] + intervals[i+1]

                new\_intervals.append([min(t), max(t)])

                skip = True

                done = True

            else:

                new\_intervals.append(intervals[i])

        return new\_intervals

    def get\_chimerge\_intervals(cls, data, colName, label, max\_intervals):

        '''

            1. Compute the χ 2 value for each pair of adjacent intervals

            2. Merge the pair of adjacent intervals with the lowest χ 2 value

            3. Repeat œ and  until χ 2 values of all adjacent pairs exceeds a threshold

        '''

        distinct\_vals = np.unique(data[colName])

        labels = np.unique(data[label])

        empty\_count = {l: 0 for l in labels}

        intervals = [[distinct\_vals[i], distinct\_vals[i]] for i in range(len(distinct\_vals))]

        while len(intervals) > max\_intervals:

            chi = []

            for i in range(len(intervals)-1):

                # Find chi square for Interval 1

                row1 = data[data[colName].between(intervals[i][0], intervals[i][1])]

                # Find chi square for Interval 2

                row2 = data[data[colName].between(intervals[i+1][0], intervals[i+1][1])]

                total = len(row1) + len(row2)

                count\_0 = np.array([v for i, v in {\*\*empty\_count, \*\*Counter(row1[label])}.items()])

                count\_1 = np.array([v for i, v in {\*\*empty\_count, \*\*Counter(row2[label])}.items()])

                count\_total = count\_0 + count\_1

                expected\_0 = count\_total\*sum(count\_0)/total

                expected\_1 = count\_total\*sum(count\_1)/total

                chi\_ = (count\_0 - expected\_0)\*\*2/expected\_0 + (count\_1 - expected\_1)\*\*2/expected\_1

                chi\_ = np.nan\_to\_num(chi\_)

                chi.append(sum(chi\_))

            min\_chi = min(chi)

            intervals = cls.get\_new\_intervals(intervals, chi, min\_chi)

        print(' Min chi square value is ' + str(min\_chi))

        return intervals

if \_\_name\_\_ == '\_\_main\_\_':

    max\_intervals = 6

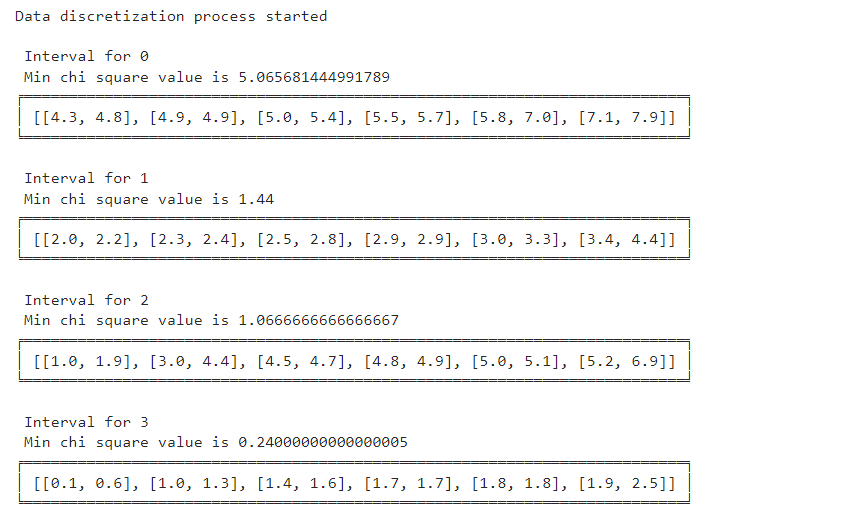
    obj = Discretization()

    for colName in iris.columns[0:-1]:

        print('\n Interval for', colName)

        intervals = obj.get\_chimerge\_intervals(iris, colName, iris.columns[-1], max\_intervals)

        print(tabulate([[intervals]], tablefmt='fancy\_grid'))

****

1. Draw following flowchart to summarize the procedures for attributes subset selection:

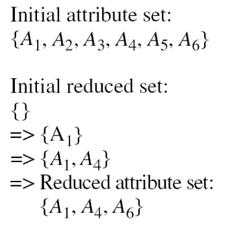
a) stepwise forward selection

b) stepwise backward selection

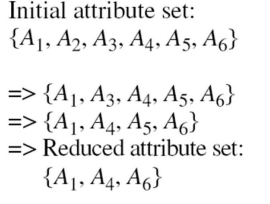
c) a combination of and backward selection and backward elimination

**Answer:**

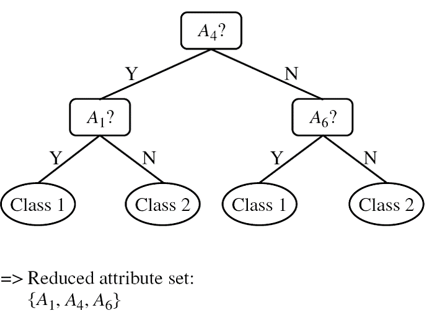
1. stepwise forward selection



1. stepwise backward selection



1. a combination of and backward selection and backward elimination



**Outcomes:**

Apply the transformations required on data to make it suitable for mining.

**Conclusion: (Conclusion to be based on the objectives and outcomes achieved)**

In this experiment, we successfully understood Attribute Subset Selection and implemented the same using the ID3 Decision Tree Algorithm.

**Grade: AA / AB / BB / BC / CC / CD /DD**

Signature of faculty in-charge with date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**References:**

Books/ Journals/ Websites:

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3nd Edition