Name: Janmejay Mohanty

CS-559-B

Homework 3 Assignment

**Solution 1: (1)**

Entropy

Entropy

Starting with Attribute 1 . Putting all instances into a table:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| T |  |  |
| F |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| T |  |  |
| F |  |  |

Entropy

Information Gain

Values of Attribute 2, lies in the following range of After doing sorting on the Attribute 2 we now split positions midway between neighboring values.

|  |  |  |
| --- | --- | --- |
| Instances | Attribute 2 | Class |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Split 1:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| E/IG | .991 |  |

Entropy

1

Information Gain:

Split 2:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| E/IG |  |  |

Entropy

Entropy

Weighted Average

Information Gain:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| E/IG |  |  |

Split 3:

Entropy

Entropy

Weighted Average

Information Gain:

Split 4:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| E/IG |  |  |

Entropy

Entropy

Weighted Average

Information Gain:

Split 5:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| E/IG |  |  |

Entropy

Entropy

Weighted Average

Information Gain:

Split 6:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| E/IG |  |  |

Entropy

Entropy

Weighted Average

Information Gain:

Split 7:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| E/IG |  |  |

Entropy

Entropy

Weighted Average

Information Gain:

Split 8:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| E/IG |  |  |

Entropy

Entropy

Weighted Average

Information Gain:

From the above calculation, we can conclude that the split position with maximum information gain at  **with Entropy and Information Gain .**

Information Gain Information Gain

**Therefore, according to the information gain, the best first splitting for decision tree is due to its higher information gain in comparison to .**

**(2)**

If we use “Instances” as new attribute then attribute “Instances” has no predictive power since new rows are assigned to new Instances values.

The Information gain for each Instances value is 0. Therefore, the overall Information gain for Instances is 0.

Therefore, the instances are not suitable attribute for a decision in the tree.

**Solution 2: (1)**

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | Class Label | |
|  |  |
| T | T |  |  |
| T | F |  |  |
| F | T |  |  |
| F | F |  |  |

The contingency tables after splitting on Attributes A and B are:

|  |  |  |
| --- | --- | --- |
| A |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| B |  |  |
|  |  |  |
|  |  |  |

The gain in Gini after splitting on A is:

The gain in Gini after splitting on B is:

**As,.**

**Therefore, Attribute A will be chosen to split the node.**

**(2)**

|  |  |  |  |
| --- | --- | --- | --- |
| Cost Matrix | Attribute Value | | |
| Actual Class |  | T | F |
|  |  |  |
|  |  |  |

Accuracy

Precision

Recall

F1-Score

|  |  |  |
| --- | --- | --- |
| A |  |  |
|  |  |  |
|  |  |  |

Accuracy

Precision

Recall

F1-Score

|  |  |  |
| --- | --- | --- |
| B |  |  |
|  |  |  |
|  |  |  |

Accuracy

Precision

Recall

F1-Score

**From the above calculation, the best first splitting will be from Attribute A.**

**Solution 3: (1)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID |  |  |  |  |  |  |  |  |  |  |
| X |  |  |  |  |  |  |  |  |  |  |
| Y |  |  |  |  |  |  |  |  |  |  |

Considering Hypotheses, .

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID |  |  |  |  |  |  |  |  |  |  |
| X |  |  |  |  |  |  |  |  |  |  |
| Y |  |  |  |  |  |  |  |  |  |  |

For those who are not classified correctly:

For those who are classified correctly:

Considering Hypotheses, .

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID |  |  |  |  |  |  |  |  |  |  |
| X |  |  |  |  |  |  |  |  |  |  |
| Y |  |  |  |  |  |  |  |  |  |  |

For those who are not classified correctly:

For those who are classified correctly:

Considering Hypotheses, .

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID |  |  |  |  |  |  |  |  |  |  |
| X |  |  |  |  |  |  |  |  |  |  |
| Y |  |  |  |  |  |  |  |  |  |  |

For those who are not classified correctly:

For those who are classified correctly:

**(2)**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

From the above table, instance 2 needs to be reweighted after first iteration. Because its error rate is larger than the other two instances error rate.

**Solution 4: Part I**

**(1)**

The test point (triangle), .

Taking random points which appears to be nearest to the test point :

Point 1,

Using Euclidean Distance:

Point 2,

Point 3,

Point 4,

Point 5,

Point 6,

Point 7,

Point 8,

As the distances keeps increases and also, we had calculated 8 Euclidean distances till now. So, there is no further need to calculate Euclidean distances for remaining points.

Above 8 Euclidean Distances, we can say that:

So, the 5 nearest neighbors according to the Euclidean Distances are,

3 Negative Class points

2 Positive Class points

**(2)**

Manhattan Distance,

Manhattan Distance Weighted,

Taking again the test point (triangle), .

Point 1,

Using Manhattan Distance:

**Manhattan Distance Weighted,**

Point 2,

Using Manhattan Distance:

**Manhattan Distance Weighted,**

Point 3,

Using Manhattan Distance:

**Manhattan Distance Weighted,**

**The Manhattan Distance Weighted 3- nearest neighbor are 1, 0.25, 0.111.**

**Part II**

#Author : Janmejay Mohanty

import numpy as np

import pandas as pd

import warnings

warnings.filterwarnings("ignore", category=FutureWarning)

testdata = pd.read\_csv('test.csv')

testdata.head()

traindata = pd.read\_csv('train.csv')

traindata.head()

X = testdata.drop(['actual-class','ID'],axis=1,)

y = testdata['actual-class']

y.head()

X.head()

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.30)

from sklearn.neighbors import KNeighborsClassifier

KNN = KNeighborsClassifier(n\_neighbors=3)

KNN.fit(X\_train,y\_train)

predict = KNN.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test,predict))

print(classification\_report(y\_test,predict))

def euclidean\_dist(A, B): #Euclidean Distance function

return np.sqrt(sum(np.square(A-B)))

def euclidean\_weight(d): #Euclidean Weighted Function

return 1/d\*\*2

X = traindata.drop(['class'],axis=1,)

y = traindata['class']

y.head()

X.head()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.30)

KNN = KNeighborsClassifier(n\_neighbors=3)

KNN.fit(X\_train,y\_train)

predict = KNN.predict(X\_test)

print(confusion\_matrix(y\_test,predict))

print(classification\_report(y\_test,predict))

#test.csv has more accuracy as compared to train.csv