IS Lab

Decision Tree based ID3 algorithm

Name- Adhyyan Tripathi

Sec/roll no. – C/8

Reg. no. – 201700403

Decision tree using ID3 Algorithm

The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes. ID3 stands for Iterative Dichotomiser 3. It is one of the many algorithms used to make decision trees. The Algorithm selection is based upon the type of target variables e.g.

#C4.5(successor of ID3)

#CART(Classification And

Regression Tree)

#CHAID(Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)

#MARS(multivariate adaptive regression splines)

ID3 Algorithm iteratively divides the features into two or more groups at each step

Steps in ID3 algorithm:

#It begins with the original set S as the root node.

#On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates Entropy(H) and Information gain(IG) of this attribute.

#It then selects the attribute which has the smallest Entropy or Largest

Information gain. #The set S is then split by the selected attribute to

produce a subset of the data.

#The algorithm continues to recur on each subset, considering only attributes never selected before.

Using a sample dataset of iris flower identification.

In [89]:

```
importnumpyasnp
importpandasaspd
import                    asplt
```

In [90]:

```
input_data=pd.read_csv("Iris.csv")
```

In [91]:

```
input_data.head()
```

Out[91]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [92]:

```
=np.finfo(float).
[illegible]
```

Calculating Entropy of each feature

In [93]:
```
defent(´nput_data,attr´bute):
                        =´nput_data.Spec´es.un´que()
    variables=´nput  data[attr´bute].un´que()
                         =0

    forvariableinvariables:                           =0
        for                    in

            num=len(input_data[attribute][input_data[attribute]==variable][inpde
            n=len(input_data[attribute][input_data[attribute]==variable])
            fraction=num/(den+eps)

                                     += -
        fraction*              +eps)fraction2=den/len(input_data)
                            += -fraction2*
```

In [94]:
```
a_entropy={k:ent(input_data,k)forkininput_data.keys()[:-1]}a_entropy
```

Out[94]:
```
{'Id': 0.0,
 'SepalLengthCm': 0.7080248798300978,
 'SepalWidthCm': 1.0740925365975489,
 'PetalLengthCm': 0.1386459770753558,
 'PetalWidthCm': 0.14906466204571406}
```

Find Information Gain

In [95]:
```
defig
```

In [96]:
```
IG={k:´g(entropy  node,a  entropy[k])forkina_entropy}
```

In [97]:
```
IG
```

Out[97]:
```
{'Id': 1.584962500721156,
 'SepalLengthCm': 0.8769376208910583,
 'SepalWidthCm': 0.5108699641236072,
 'PetalLengthCm': 1.4463165236458002,
 'PetalWidthCm': 1.4358978386754423}
```

Find entropy of original dataset S

In [98]:
```
def
    Species=input_data.keys()[-
    1]entropy=0
    values=input_data[Spec es].un que()
    forvalueinvalues:

        fraction=input_data[Species].value_counts()[value]/len(input_data[Specen
        tropy+=-fraction*np.log2(fraction)
```

In [99]:
```
def
                    =[]

    IG=[]

    forkeyininputdata.keys()[:-1]:
        Entropy_att.append(ent(inp utdata,key))
```

In [100]:

```
defget_subtable(input_data,node,value):
        returninput_data[input_data[node]==value].reset_index(drop= )
```

In [101]:

```
defbuildTree(input_data,tree=None):S
        pecies=input_data.keys()[-1]



        node=f nd_w nner( nput_data)

        attValue=np.unique(input_data[node])


        iftreeisNone:tree={}
                tree[node]={}


        forvalueinattValue: subtable=                    (input_data,node,value)

                clValue,counts=np.unique(subtable['Species'],return_counts= )



                iflen(counts)==1:
                        tree[node][value]=clValue[0]
```

In [102]:
```
t=buildTree(input_data)
```

In [103]:
```
importpprint
```

In [104]:
```
                 (t)
```

```
                 -setosa',
             2:'Iris-setosa',
             3:'Iris-setosa',
             4:'Iris-setosa',
             5:'Iris-setosa',
             6:'Iris-setosa',
             7:'Iris-setosa',
             8:'Iris-setosa',
           9:'Iris-setosa',
```