18BCE101

DM prac 4

AIM:

Construct a decision tree, find using entropy or information gain which attribute can be used as 'root' of the decision tree.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets, preprocessing
from sklearn.tree import export_graphviz
from sklearn.model_selection import train_test_split
```

Given data

In [2]:
```
data = pd.read_csv('car_sales.csv')
data.head()
```

Out[2]:

	ID	Age	Income	Student	Credit Rating	Buy Car
0	1	Young	High	No	Fair	No
1	2	Young	High	No	Good	No
2	3	Middle	High	No	Fair	Yes
3	4	Old	Medium	No	Fair	Yes
4	5	Old	Low	Yes	Fair	Yes

In [3]:
```
data.shape
```

Out[3]:
```
(14, 6)
```

In [4]:
```
data = data.drop('ID', axis=1)
data.head(14)
```

Out[4]:

	Age	Income	Student	Credit Rating	Buy Car
0	Young	High	No	Fair	No
1	Young	High	No	Good	No
2	Middle	High	No	Fair	Yes
3	Old	Medium	No	Fair	Yes
4	Old	Low	Yes	Fair	Yes
5	Old	Low	Yes	Good	No
6	Middle	Low	Yes	Good	Yes
7	Young	Medium	No	Fair	No
8	Young	Low	Yes	Fair	Yes
9	Old	Medium	Yes	Fair	Yes
10	Young	Medium	Yes	Good	Yes
11	Middle	Medium	No	Good	Yes
12	Middle	High	Yes	Fair	Yes
13	Old	Medium	No	Good	No

In [5]:
```
for i in data:
    l = data[i].unique()
    temp = 0
    for j in l:
        data.loc[(data[i] == j), i] = temp
        temp += 1
data.head(14)
```

Out[5]:

	Age	Income	Student	Credit Rating	Buy Car
0	0	0	0	0	0
1	0	0	0	1	0
2	1	0	0	0	1
3	2	1	0	0	1
4	2	2	1	0	1
5	2	2	1	1	0
6	1	2	1	1	1
7	0	1	0	0	0
8	0	2	1	0	1
9	2	1	1	0	1
10	0	1	1	1	1
11	1	1	0	1	1
12	1	0	1	0	1
13	2	1	0	1	0

Calculating Information gain

```
Calculate the entropy of a dataset.
The only parameter of this function is the target_col parameter which specifies the ta
rget column
```

In [6]:
```
def entropy(target_col):
    elements,counts = np.unique(target_col,return_counts = True)
    entropy = np.sum([(-counts[i]/np.sum(counts))*np.log2(counts[i]/np.sum(counts)) for i in
range(len(elements))])
    return entropy
```

Calculate the information gain of a dataset. This function takes three parameters:

```
1. data = The dataset for whose feature the IG should be calculated
2. split_attribute_name = the name of the feature for which the information gain shoul
d be calculated
3. target_name = the name of the target feature.
```

In [7]:
```
def InfoGain(data,split_attribute_name,target_name):

    #Calculate the values and the corresponding counts for the split attribute
    vals,counts= np.unique(data[split_attribute_name],return_counts=True)

    #Calculate the weighted entropy
    Weighted_Entropy = np.sum([(counts[i]/np.sum(counts))*entropy(data.where(data[split_attr
ibute_name]==vals[i]).dropna()[target_name]) for i in range(len(vals))])

    #Calculate the information gain
    return Weighted_Entropy
```

In [8]:
```
gain = entropy(data['Buy Car'])
print(gain)
```

```
0.9402859586706311
```

In [9]:
```
information_gains = {}
columns = ['Age','Income', 'Student', 'Credit Rating']

for col in columns:
    information_gain = InfoGain(data, col, 'Buy Car')
    information_gains[col] = gain - information_gain
    print("information gain of "+str(col)+" = "+str(information_gain))
```

```
information gain of Age = 0.6935361388961918
information gain of Income = 0.9110633930116763
information gain of Student = 0.7884504573082896
information gain of Credit Rating = 0.8921589282623617
```

In [10]:
```
root = max(information_gains, key=information_gains.get)
print(root)
```

```
Age
```

visualizing

In [17]:
```
X = data[['Age','Income','Student', 'Credit Rating']]
y = data[['Buy Car']]
label_encoder=preprocessing.LabelEncoder ()
for col in columns:
    data[i] = label_encoder.fit_transform(data[i])
```

In [18]:
```
tree = DecisionTreeClassifier(criterion = 'entropy').fit(X,y)
```

In [19]:
```
export_graphviz(tree, out_file='cde.dot')
```


Real world

In [13]:
```
X,y = datasets.load_iris(return_X_y=True)
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [15]:
```
tree = DecisionTreeClassifier(criterion = 'entropy').fit(X_train,y_train)
```

In [16]:
```
export_graphviz(tree, out_file='abc.dot')
```


Conclusion

A decision tree algorithm, is a machine learning technique, for making predictions. It behaves like a tree structure. The decision tree is built by, repeatedly splitting, training data, into smaller and smaller samples. Decision Tree works on, the principle of conditions. The algorithm checks conditions, at a node, and split the data, as per the result, of the conditional statement. Decision Tree algorithm belongs to, the family of, supervised machine learning algorithms. It can be used to, build classification, as well as regression models.

ID#, Gini index and C4.5 are different attribute selection methods In this program we found the root using information gain and worked on it from scratch. As the values were discrete we opted for Information gain

In [ ]:
```
```