Date : 02/05/2024

EXPERIMENT NO. 11

AIM: To implement decision tree classifier(C5.0 and CART) and decision tree regressor using python programming.

SOFTWARE USED: Jupyter Notebook.

THEORY:

Decision trees are a popular and intuitive form of machine learning model used for both classification and regression tasks. Two well-known decision tree algorithms for classification are C5.0 and CART. Decision tree regressors also use CART (Classification and Regression Trees) as their underlying algorithm. In this guide, we'll explain the theory behind these models.

C5.0 (Decision Tree Classifier):

Overview:

- C5.0 is a decision tree algorithm developed by Ross Quinlan as an extension of the C4.5 algorithm.
- It is known for its speed and efficiency, especially with larger datasets, and improved handling of noisy data.
- C5.0 uses a divide-and-conquer approach to recursively partition the dataset.

Steps:

- Splitting: The algorithm chooses the best attribute to split the data based on a measure like information gain or gain ratio.
- Stopping Criteria: The recursion stops when all instances in a node belong to the same class or when a specified depth limit is reached.
- Pruning: C5.0 uses pruning to remove branches that do not contribute significantly to classification accuracy.
- Leaf Node Prediction: Each leaf node is assigned a class label based on the majority class in that node.
- Boosting: C5.0 can use boosting, a technique where multiple trees are combined to improve accuracy.

CART (Classification and Regression Trees):

Overview:

- CART is an algorithm for constructing binary decision trees for classification or regression tasks.
- In a classification task, CART constructs a tree based on features that split the data into two branches (binary split) such that each branch is more homogeneous in terms of class labels.

Steps:

- Splitting: At each node, CART evaluates all possible binary splits for each feature and selects the one that results in the maximum information gain (for classification) or minimum variance (for regression).

- Stopping Criteria: The recursion stops when a stopping criterion is met, such as a minimum number of samples in a node, maximum tree depth, or no further improvements in impurity.
- Pruning: CART uses a pruning technique called cost-complexity pruning to remove sections of the tree that don't improve performance on a validation set.
- Leaf Node Prediction: In classification, the majority class of the node is assigned as the class label. In regression, the average value of the target in that node is used for prediction.

Decision Tree Regressor:

Overview:

- Decision tree regression uses CART (the same algorithm as above) but instead of class labels, it predicts continuous values.
- Instead of maximizing information gain, decision tree regressors minimize the variance within the target variable.

Steps:

- Splitting: The tree is split by selecting the feature and threshold that minimize the mean squared error (MSE) or another variance-based metric.
- Stopping Criteria: As in classification, the recursion stops when a minimum number of samples in a node or another stopping criterion is met.
- Pruning: Pruning techniques are applied to prevent overfitting and improve generalization.
- Leaf Node Prediction: The predicted value in each leaf node is the mean of the target values in that node.

Common Concepts Across All Trees:

- Impurity Metrics: Metrics such as Gini impurity and entropy (for classification) or mean squared error (for regression) guide the selection of the best split at each node.
- Pruning: A method used to avoid overfitting by reducing the size of the decision tree.
- Leaf Nodes: Terminal nodes in the decision tree, representing final predictions.

These algorithms use different strategies and metrics for constructing decision trees, but the overall approach is similar: recursively split the data based on feature values to achieve increasingly pure or homogeneous nodes, and use leaf nodes to make predictions.

OUTPUT CODE:

In [5]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn  import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn import preprocessing
```

In [6]:
```
iris = pd.read_csv("Iris.csv",index_col=0)
```

In [7]:
```
iris.head()
```

Out[7]:

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

In [9]:
```
label_encoder=preprocessing.LabelEncoder()
iris['Species']=label_encoder.fit_transform(iris['Species'])
```

In [10]:
```
x=iris.iloc[:,0:4]
y=iris['Species']
```

In [11]:
```
iris
```

Out[11]:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	0
2	4.9	3.0	1.4	0.2	0
3	4.7	3.2	1.3	0.2	0
4	4.6	3.1	1.5	0.2	0
5	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
146	6.7	3.0	5.2	2.3	2
147	6.3	2.5	5.0	1.9	2
148	6.5	3.0	5.2	2.0	2
149	6.2	3.4	5.4	2.3	2
150	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [12]:
```
x
```

Out[12]:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
...	...	...	...	...
146	6.7	3.0	5.2	2.3
147	6.3	2.5	5.0	1.9
148	6.5	3.0	5.2	2.0
149	6.2	3.4	5.4	2.3
150	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [13]:
```
y
```

Out[13]:
```
Id
1      0
2      0
3      0
4      0
5      0
      ..
146    2
147    2
148    2
149    2
150    2
Name: Species, Length: 150, dtype: int32
```

In [14]:
```
iris['Species'].unique()
```

Out[14]:
```
array([0, 1, 2])
```

In [15]:
```
iris.Species.value_counts()
```

Out[15]:
```
0    50
1    50
2    50
Name: Species, dtype: int64
```

In [16]:
```
colnames=list(iris.columns)
colnames
```

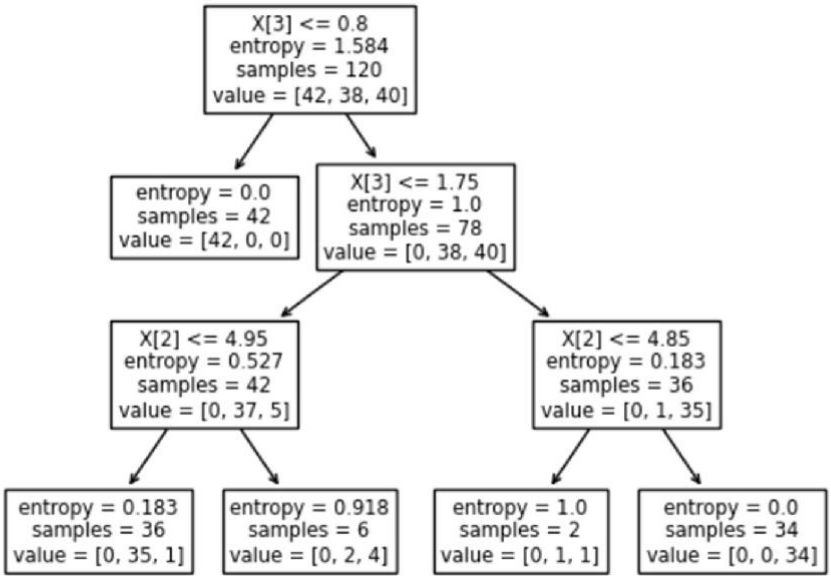
Out[16]:
```
['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Species']
```

In [18]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=40)
```

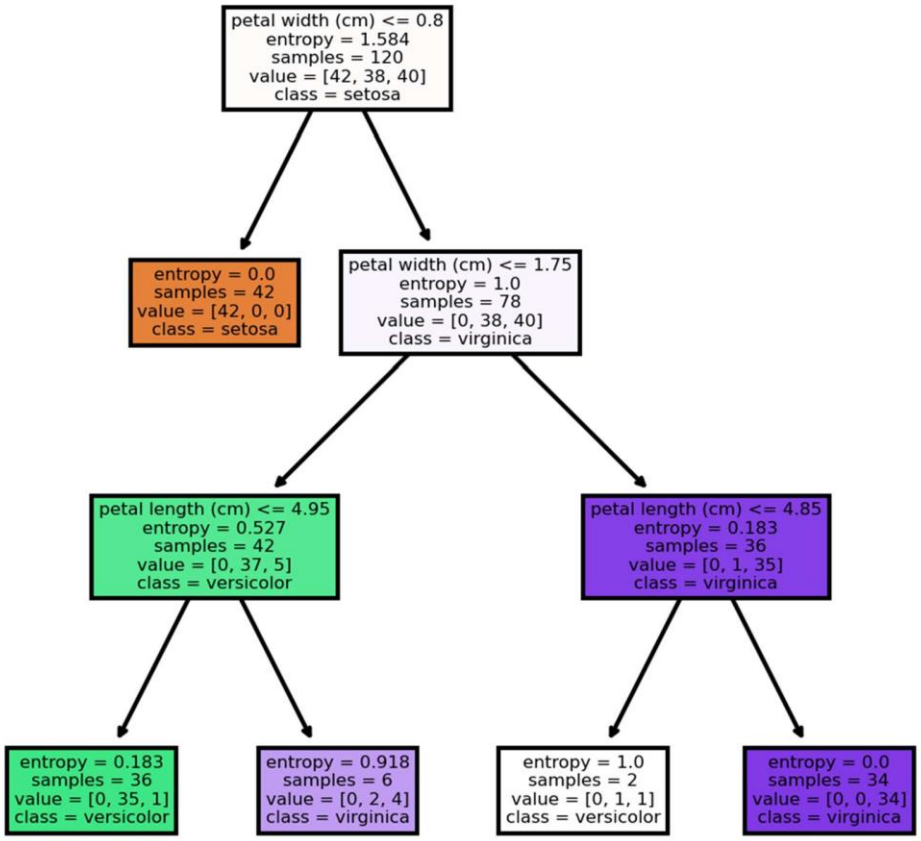
In [19]:
```
#building decision tree using entropy as we are using c5.0
model=DecisionTreeClassifier(criterion='entropy',max_depth=3)
model.fit(x_train,y_train)
```

Out[19]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [20]:
```
tree.plot_tree(model);
```



In [21]:
```
fn=['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']
cn=['setosa','versicolor','virginica']
fig,axes = plt.subplots(nrows=1, ncols=1, figsize=(4,4), dpi=300)
tree.plot_tree(model,
               feature_names=fn,
               class_names=cn,
               filled= True);
```



In [22]:
```
preds= model.predict(x_test)
pd.Series(preds).value_counts()
```

Out[22]:
```
1    13
2     9
0     8
dtype: int64
```

In [23]:
```
preds
```

Out[23]:
```
array([0, 1, 2, 2, 1, 2, 1, 1, 1, 0, 1, 0, 0, 1, 1, 2, 2, 2, 1, 1, 2, 2,
       1, 0, 1, 0, 0, 2, 0, 1])
```

```
In [24]: pd.crosstab(y_test,preds)
```

Out[24]:

col_0	0	1	2
Species			
0	8	0	0
1	0	12	0
2	0	1	9

```
In [25]: np.mean(preds==y_test)
Out[25]: 0.9666666666666667

In [26]: #using cart (giini criteria)
         model_gini=DecisionTreeClassifier(criterion='gini',max_depth=3)

In [27]: model_gini.fit(x_train,y_train)
Out[27]: DecisionTreeClassifier(max_depth=3)
```

```
In [30]: pred=model_gini.predict(x_test)
         np.mean(pred==y_test)
Out[30]: 0.9666666666666667

In [31]: #decision tree regression example
         from sklearn.tree import DecisionTreeRegressor

In [32]: array=iris.values
         x= array[:,0:3]
         y=array[:,3]

In [33]: x_train, x_test, y_train, y_test= train_test_split(x, y, test_size=0.33, random_state=1)

In [34]: model=DecisionTreeRegressor()
         model.fit(x_train,y_train)
Out[34]: DecisionTreeRegressor()

In [35]: model.score(x_test, y_test)
Out[35]: 0.8806879577380238
```

CONCLUSION:

In conclusion, decision trees are powerful machine learning models for both classification and regression tasks. C5.0 excels in classification by efficiently handling large datasets and noisy data. CART is the foundation for both classification and regression trees, providing straightforward binary splits. Pruning helps prevent overfitting in all decision trees. Overall, decision trees offer an intuitive and effective approach for various machine learning problems.