machine-learning-assignment-1

December 6, 2023

Q1. Explore scikit learn Library.

Ans= Step 1: Load a dataset:- A dataset is nothing but a collection of data. A dataset generally has two main components:

i)Features: (also known as predictors, inputs, or attributes) they are simply the variables of our data. They can be more than one and hence represented by a feature matrix.

ii)Response: (alsoknown as the target, label, or output) This is the output variable depending on the feature variables. We generally have a single response column and it is represented by a response vector.

The sklearn.datasets package embeds some small toy datasets as introduced in the Getting Started section. This package also features helpers to fetch larger datasets commonly used by the machine learning community to benchmark algorithms on data that comes from the 'real world'. For Example - Iris plants dataset, Diabetes dataset, Wine recognition dataset, Breast cancer wisconsin (diagnostic) dataset.

Step 2: Splitting the dataset:-

i)Split the dataset into two pieces: a training set and a testing set.

ii)Train the model on the training set.

iii)Test the model on the testing set, and evaluate how well our model did.

klearn.model_selection.train_test_split(*arrays, test_size=None train_size=None, random_state=None, shuffle=True, stratify=None).

Split arrays or matrices into random train and test subsets.

Step 3: Training the model:- Scikit-learn provides a wide range of machine learning algorithms that have a unified/consistent interface for fitting, predicting accuracy, etc.

sklearn.tree.DecisionTreeClassifier is a class in the scikit-learn library that implements a decision tree classifier. Decision trees are a type of supervised learning algorithm that can be used for both classification and regression tasks.

sklearn.linear_model is a module in the scikit-learn library that provides a wide range of linear models for regression, classification, and other tasks. The LinearRegression class in 'sklearn.

Now we use fit(), predict(), accuracy_score(),metrics.accuracy_score() functions.

Q2. Explore Datasets Online (can refer Kaggle, UCI ML, etc.)

a) Load dataset in google colab.

[2]:
```python
from sklearn.datasets import load_wine
import pandas as pd

wine = load_wine()
wine_df = pd.DataFrame(data=wine.data, columns=wine.feature_names)
wine_df['target'] = wine.target
```

b) Print first five values and last five values in dataset.

[3]:
```python
print("First five values in the Wine dataset:",wine_df.head())
print("*"*125)
print("Last five values in the Wine dataset:",wine_df.tail())
```

```
First five values in the Wine dataset:    alcohol  malic_acid   ash
alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80
1    13.20        1.78  2.14               11.2      100.0           2.65
2    13.16        2.36  2.67               18.6      101.0           2.80
3    14.37        1.95  2.50               16.8      113.0           3.85
4    13.24        2.59  2.87               21.0      118.0           2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0        3.06                  0.28             2.29             5.64  1.04
1        2.76                  0.26             1.28             4.38  1.05
2        3.24                  0.30             2.81             5.68  1.03
3        3.49                  0.24             2.18             7.80  0.86
4        2.69                  0.39             1.82             4.32  1.04

   od280/od315_of_diluted_wines  proline  target
0                          3.92   1065.0       0
1                          3.40   1050.0       0
2                          3.17   1185.0       0
3                          3.45   1480.0       0
4                          2.93    735.0       0
********************************************************************************
*********************************************
Last five values in the Wine dataset:      alcohol  malic_acid   ash
alcalinity_of_ash  magnesium  total_phenols  \
173    13.71        5.65  2.45               20.5       95.0           1.68
174    13.40        3.91  2.48               23.0      102.0           1.80
175    13.27        4.28  2.26               20.0      120.0           1.59
176    13.17        2.59  2.37               20.0      120.0           1.65
177    14.13        4.10  2.74               24.5       96.0           2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
173        0.61                  0.52             1.06              7.7  0.64
174        0.75                  0.43             1.41              7.3  0.70
175        0.69                  0.43             1.35             10.2  0.59
```

```
176        0.68                    0.53             1.46                9.3  0.60
177        0.76                    0.56             1.35                9.2  0.61

     od280/od315_of_diluted_wines  proline  target
173                          1.74    740.0       2
174                          1.56    750.0       2
175                          1.56    835.0       2
176                          1.62    840.0       2
177                          1.60    560.0       2
```

c) check correlation between fields present in dataset.

[5]:
```
import seaborn as sns

correlation_matrix = wine_df.corr()
print("Correlation between fields in the Wine dataset: ",correlation_matrix)

sns.heatmap(correlation_matrix, annot=True, cmap='ocean')
```

```
Correlation between fields in the Wine dataset:
alcohol  malic_acid       ash  \
alcohol                       1.000000    0.094397  0.211545
malic_acid                    0.094397    1.000000  0.164045
ash                           0.211545    0.164045  1.000000
alcalinity_of_ash            -0.310235    0.288500  0.443367
magnesium                     0.270798   -0.054575  0.286587
total_phenols                 0.289101   -0.335167  0.128980
flavanoids                    0.236815   -0.411007  0.115077
nonflavanoid_phenols         -0.155929    0.292977  0.186230
proanthocyanins               0.136698   -0.220746  0.009652
color_intensity               0.546364    0.248985  0.258887
hue                          -0.071747   -0.561296 -0.074667
od280/od315_of_diluted_wines  0.072343   -0.368710  0.003911
proline                       0.643720   -0.192011  0.223626
target                       -0.328222    0.437776 -0.049643

                              alcalinity_of_ash  magnesium  total_phenols  \
alcohol                               -0.310235   0.270798       0.289101
malic_acid                             0.288500  -0.054575      -0.335167
ash                                    0.443367   0.286587       0.128980
alcalinity_of_ash                      1.000000  -0.083333      -0.321113
magnesium                             -0.083333   1.000000       0.214401
total_phenols                         -0.321113   0.214401       1.000000
flavanoids                            -0.351370   0.195784       0.864564
nonflavanoid_phenols                   0.361922  -0.256294      -0.449935
proanthocyanins                       -0.197327   0.236441       0.612413
color_intensity                        0.018732   0.199950      -0.055136
hue                                   -0.273955   0.055398       0.433681
```

od280/od315_of_diluted_wines	-0.276769	0.066004	0.699949
proline	-0.440597	0.393351	0.498115
target	0.517859	-0.209179	-0.719163

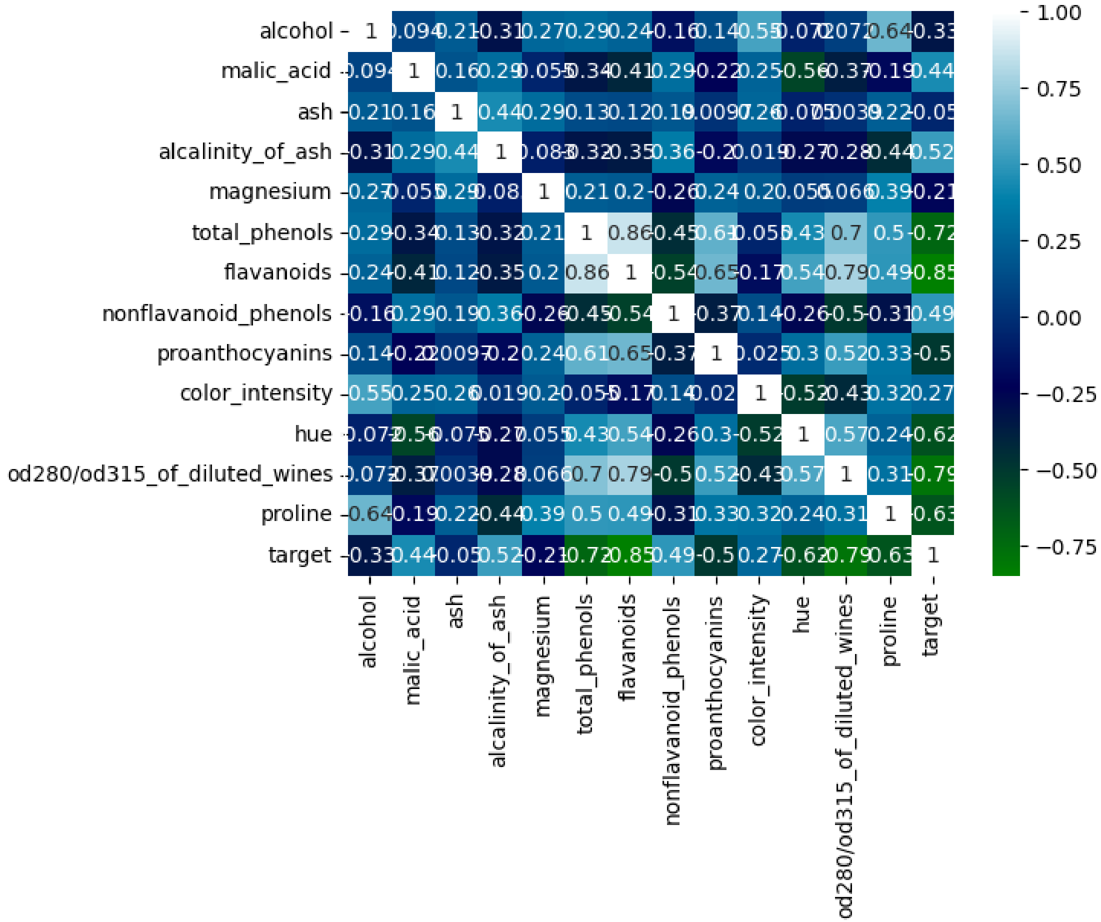
	flavanoids	nonflavanoid_phenols	\
alcohol	0.236815	-0.155929	
malic_acid	-0.411007	0.292977	
ash	0.115077	0.186230	
alcalinity_of_ash	-0.351370	0.361922	
magnesium	0.195784	-0.256294	
total_phenols	0.864564	-0.449935	
flavanoids	1.000000	-0.537900	
nonflavanoid_phenols	-0.537900	1.000000	
proanthocyanins	0.652692	-0.365845	
color_intensity	-0.172379	0.139057	
hue	0.543479	-0.262640	
od280/od315_of_diluted_wines	0.787194	-0.503270	
proline	0.494193	-0.311385	
target	-0.847498	0.489109	

	proanthocyanins	color_intensity	hue	\
alcohol	0.136698	0.546364	-0.071747	
malic_acid	-0.220746	0.248985	-0.561296	
ash	0.009652	0.258887	-0.074667	
alcalinity_of_ash	-0.197327	0.018732	-0.273955	
magnesium	0.236441	0.199950	0.055398	
total_phenols	0.612413	-0.055136	0.433681	
flavanoids	0.652692	-0.172379	0.543479	
nonflavanoid_phenols	-0.365845	0.139057	-0.262640	
proanthocyanins	1.000000	-0.025250	0.295544	
color_intensity	-0.025250	1.000000	-0.521813	
hue	0.295544	-0.521813	1.000000	
od280/od315_of_diluted_wines	0.519067	-0.428815	0.565468	
proline	0.330417	0.316100	0.236183	
target	-0.499130	0.265668	-0.617369	

	od280/od315_of_diluted_wines	proline	target
alcohol	0.072343	0.643720	-0.328222
malic_acid	-0.368710	-0.192011	0.437776
ash	0.003911	0.223626	-0.049643
alcalinity_of_ash	-0.276769	-0.440597	0.517859
magnesium	0.066004	0.393351	-0.209179
total_phenols	0.699949	0.498115	-0.719163
flavanoids	0.787194	0.494193	-0.847498
nonflavanoid_phenols	-0.503270	-0.311385	0.489109
proanthocyanins	0.519067	0.330417	-0.499130
color_intensity	-0.428815	0.316100	0.265668
hue	0.565468	0.236183	-0.617369

```
od280/od315_of_diluted_wines                      1.000000  0.312761 -0.788230
proline                                           0.312761  1.000000 -0.633717
target                                           -0.788230 -0.633717  1.000000
```

[5]: <Axes: >



Q3.Implement Decision tree classification on iris data, weather data and purchase data.

a) *Use different criterion like gini index and entropy observe it is affecting the performance or not.*

b) *Test the model with your new data and find the results.*

Ans= *i) using Iris Dataset.*

[1]:
```python
from sklearn.datasets import load_iris
iris= load_iris()
```

[2]:
```python
#features
print(iris.data)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.2]
 [5.  3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.6 1.4 0.1]
 [4.4 3.  1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.  3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
```

```
[5.3 3.7 1.5 0.2]
[5.  3.3 1.4 0.2]
[7.  3.2 4.7 1.4]
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1. ]
[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.  2.  3.5 1. ]
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
```

```
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.  2. ]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2. ]
[7.7 2.8 6.7 2. ]
[6.3 2.7 4.9 1.8]
[6.7 3.3 5.7 2.1]
[7.2 3.2 6.  1.8]
[6.2 2.8 4.8 1.8]
[6.1 3.  4.9 1.8]
[6.4 2.8 5.6 2.1]
[7.2 3.  5.8 1.6]
[7.4 2.8 6.1 1.9]
[7.9 3.8 6.4 2. ]
[6.4 2.8 5.6 2.2]
[6.3 2.8 5.1 1.5]
[6.1 2.6 5.6 1.4]
[7.7 3.  6.1 2.3]
[6.3 3.4 5.6 2.4]
[6.4 3.1 5.5 1.8]
[6.  3.  4.8 1.8]
[6.9 3.1 5.4 2.1]
[6.7 3.1 5.6 2.4]
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
[6.8 3.2 5.9 2.3]
```

```
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
```

[3]:
```
#labels
print(iris.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

[4]:
```
X = iris.data
y = iris.target
```

[5]:
```
from sklearn.model_selection import train_test_split as tts
x_train, x_test,y_train, y_test = tts(X,y,test_size=0.3, random_state=10)
print(y_train)
```

```
[0 1 1 2 2 1 2 1 1 1 0 0 1 0 2 0 0 2 1 2 0 2 0 1 1 0 2 2 2 2 2 0 1 2 1 0 2
 1 1 0 0 0 1 2 2 1 0 0 0 2 2 1 1 2 2 2 2 1 0 0 1 0 0 2 1 0 0 0 1 0 1 0 1 2
 0 1 1 2 0 2 0 1 1 2 2 0 1 2 2 1 1 2 0 2 0 0 1 0 2 2 2 1 0 2 0]
```

[6]:
```
#model
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion= 'entropy')
clf.fit(x_train,y_train)
```

[6]:
```
DecisionTreeClassifier(criterion='entropy')
```

[7]:
```
#evaluating using test data
y_pred = clf.predict(x_test)
print(y_pred)
```

```
[1 2 0 1 0 1 1 1 0 1 1 2 1 0 0 2 1 0 0 0 2 2 2 0 1 0 1 1 1 2 1 1 1 2 2 0 2
 2 2 2 0 0 1 0 1]
```

[15]:
```
from sklearn import metrics
acc = metrics.accuracy_score(y_test, y_pred)
print(acc*100)
```

```
75.0
```

[9]:
```
print(y_test)
```

```
[1 2 0 1 0 1 1 1 0 1 1 2 1 0 0 2 1 0 0 0 2 2 2 0 1 0 1 1 1 2 1 1 2 2 2 0 2
 2 2 2 0 0 1 0 1]
```

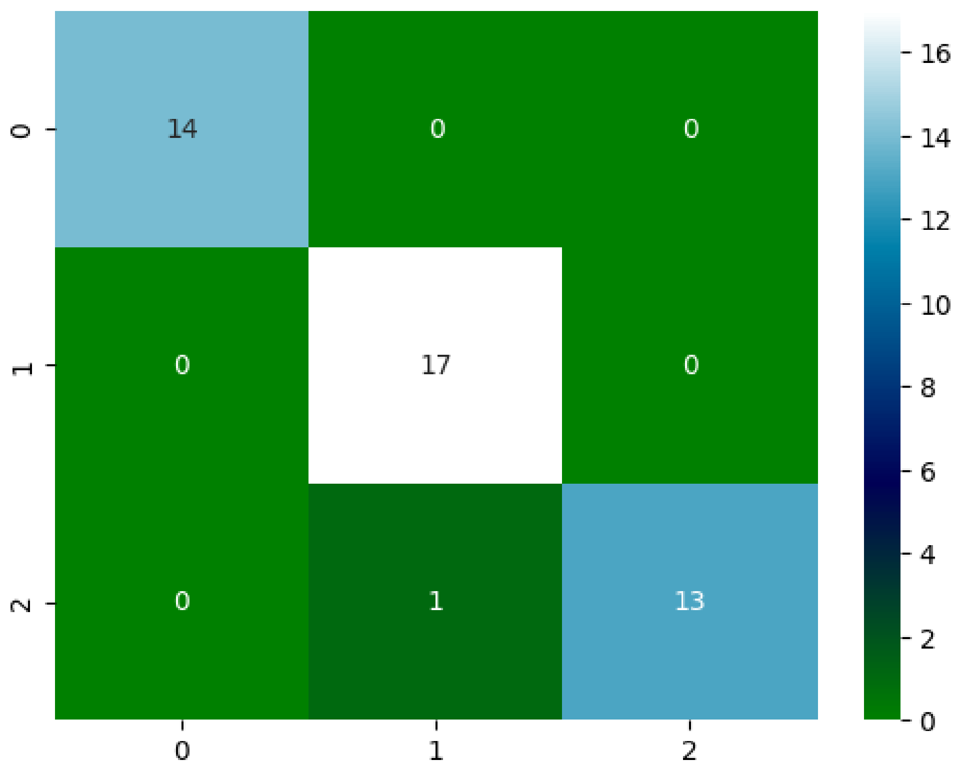
[10]:
```
print(y_pred)
```

```
[1 2 0 1 0 1 1 1 0 1 1 2 1 0 0 2 1 0 0 0 2 2 2 0 1 0 1 1 1 2 1 1 1 2 2 0 2
 2 2 2 0 0 1 0 1]
```

[11]:
```
cn= metrics.confusion_matrix(y_test, y_pred)
```

[12]:
```
import seaborn as sns
sns.heatmap(cn, annot=True, cmap='ocean')
```

[12]:
```
<Axes: >
```



[13]:
```
acc0 = metrics.accuracy_score(y_test, y_pred)
print("Accuracy is ",acc0*100)
```

```
Accuracy is  97.77777777777777
```

ii) using Weather Dataset.

[2]:
```
import pandas as pd
import seaborn as sns

df= pd.read_csv("weather.csv")
print(df)
```

```
     outlook temperature humidity  windy play
0   overcast         hot     high  False  yes
1   overcast        cool   normal   True  yes
2   overcast        mild     high   True  yes
3   overcast         hot   normal  False  yes
4      rainy        mild     high  False  yes
5      rainy        cool   normal  False  yes
6      rainy        cool   normal   True   no
7      rainy        mild   normal  False  yes
8      rainy        mild     high   True   no
9      sunny         hot     high  False   no
10     sunny         hot     high   True   no
11     sunny        mild     high  False   no
12     sunny        cool   normal  False  yes
13     sunny        mild   normal   True  yes
```

[3]:
```
df['outlook']= df.outlook.map({'overcast':0,'rainy':1,'sunny':2 })
df.head()
```

[3]:
```
   outlook temperature humidity  windy play
0        0         hot     high  False  yes
1        0        cool   normal   True  yes
2        0        mild     high   True  yes
3        0         hot   normal  False  yes
4        1        mild     high  False  yes
```

[4]:
```
from sklearn.preprocessing import LabelEncoder

le= LabelEncoder()
lw=le.fit_transform(df['windy'])
print(lw)
```

```
[0 1 1 0 0 0 1 0 1 0 1 0 0 1]
```

[5]:
```
df.drop("windy", axis=1, inplace=True)
df["windy"]=lw
df.head()
```

[5]:
```
   outlook temperature humidity play  windy
0        0         hot     high  yes      0
1        0        cool   normal  yes      1
```

```
2         0         mild      high  yes       1
3         0          hot    normal  yes       0
4         1         mild      high  yes       0
```

[6]:
```
df['temperature']= df.temperature.map({'hot':0,'cool':1,'mild':2})
df['humidity']= df.humidity.map({'high':0,'normal':1})
df['play']= df.play.map({'no':0,'yes':1})
```

[7]:
```
X1=df.drop('play',axis=1)
y1=df[['play']]
```

[8]:
```
from sklearn.model_selection import train_test_split as tts
x_train, x_test,y_train, y_test = tts(X1,y1,test_size=0.25, random_state=10)
```

[9]:
```
#create model
from sklearn.tree import DecisionTreeClassifier as dtc

clf1= dtc(criterion='entropy')
```

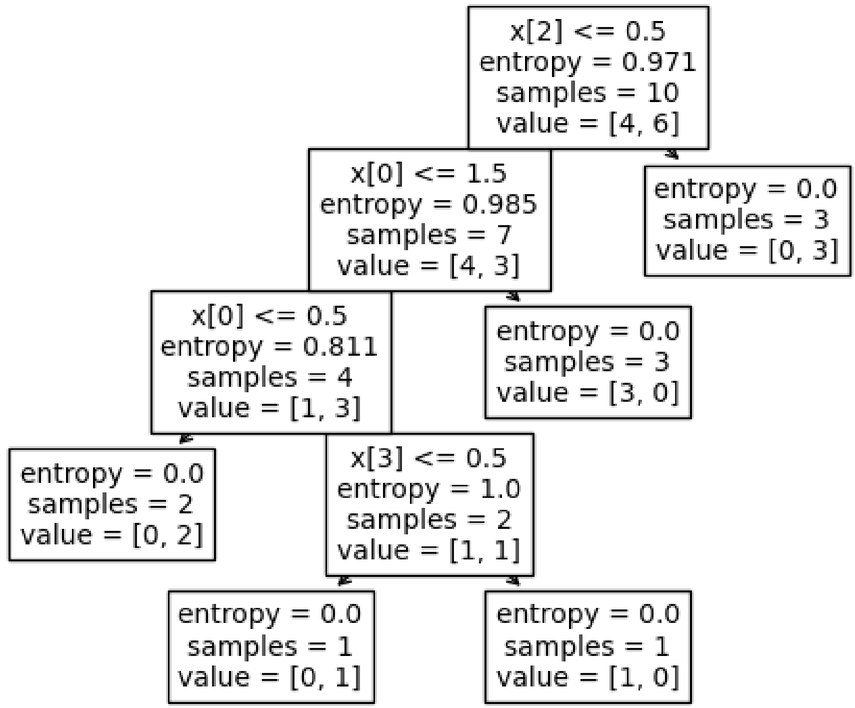
[10]:
```
#train the dataset
clf1.fit(x_train, y_train)
```

[10]:
```
DecisionTreeClassifier(criterion='entropy')
```

DecisionTreeClassifier(criterion='entropy')

[12]:
```
from sklearn import tree
tree.plot_tree(clf1)
```

[12]:
```
[Text(0.6666666666666666, 0.9, 'x[2] <= 0.5\nentropy = 0.971\nsamples =
10\nvalue = [4, 6]'),
 Text(0.5, 0.7, 'x[0] <= 1.5\nentropy = 0.985\nsamples = 7\nvalue = [4, 3]'),
 Text(0.3333333333333333, 0.5, 'x[0] <= 0.5\nentropy = 0.811\nsamples = 4\nvalue
= [1, 3]'),
 Text(0.16666666666666666, 0.3, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.5, 0.3, 'x[3] <= 0.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(0.3333333333333333, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.6666666666666666, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.6666666666666666, 0.5, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.8333333333333334, 0.7, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]')]
```



[13]:

```
y_pred = clf1.predict(x_test)
print(y_pred)
print(y_test)
```

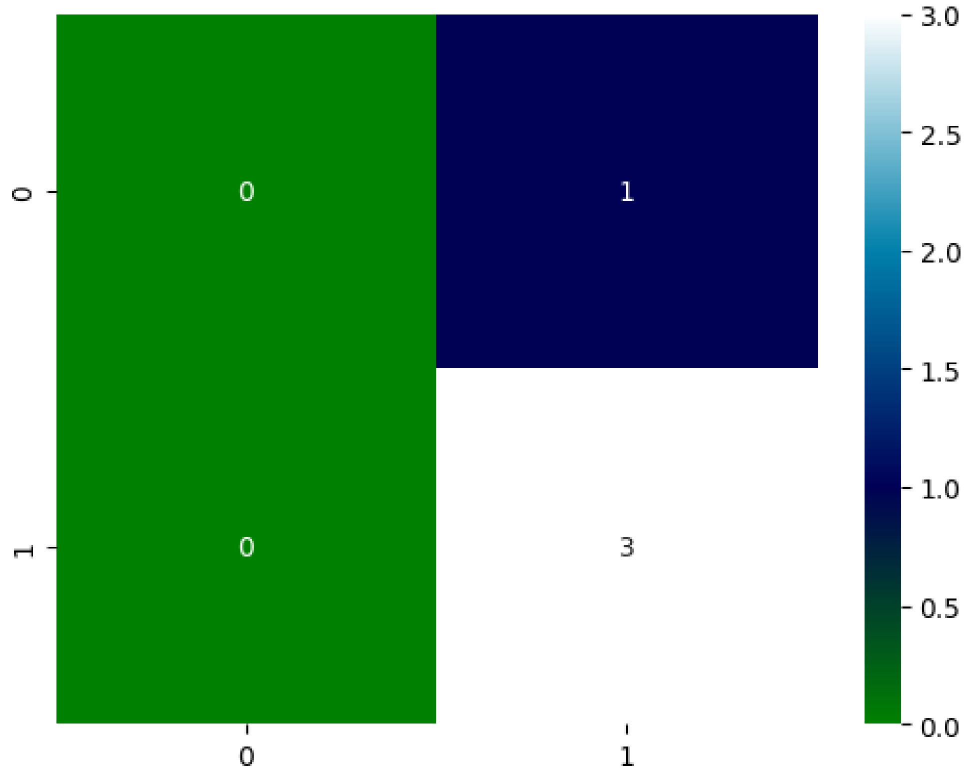
```
[1 1 1 1]
    play
3      1
7      1
12     1
6      0
```

[16]:

```
cn1= metrics.confusion_matrix(y_test, y_pred)
sns.heatmap(cn1, annot=True, cmap='ocean')
```

[16]:

```
<Axes: >
```



[17]:
```
import sklearn.metrics as mat

acc1 = metrics.accuracy_score(y_test, y_pred)
print("Accuracy is ",acc1*100)
```

```
Accuracy is  75.0
```

[18]:
```
print(mat.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.75      1.00      0.86         3

    accuracy                           0.75         4
   macro avg       0.38      0.50      0.43         4
weighted avg       0.56      0.75      0.64         4


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
```

```
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

iii) using Purchase dataset.

[19]:
```python
df= pd.read_csv("/content/Purchase_new.csv")
print(df)
```

```
   Holiday Discount Free Delivery Purchase
0       No      Yes           Yes      Yes
1       No      Yes           Yes      Yes
2       No       No            No       No
3      yes      Yes           Yes      Yes
4      yes      Yes           Yes      Yes
5      yes       No            No       No
6      yes      Yes            No      Yes
7       No      Yes           Yes      Yes
8      yes      Yes           Yes      Yes
9      yes      Yes           Yes      Yes
10     yes       No           Yes      Yes
11     yes       No            No       No
12     yes      Yes           Yes      Yes
13     yes      Yes           Yes      Yes
14     yes      Yes           Yes      Yes
15      No      Yes           Yes      Yes
16     yes       No           Yes      Yes
17      No      Yes            No      Yes
18     yes       No            No      Yes
19     yes       No           Yes      Yes
20      No      Yes           Yes      Yes
21     yes      Yes           Yes       No
22     yes       No           Yes      Yes
23      No      Yes           Yes      Yes
24     yes       No            No       No
25      No       No           Yes       No
26      No      Yes           Yes      Yes
27      No      Yes           Yes      Yes
28     yes      Yes           Yes      Yes
29     yes      Yes           Yes      Yes
```

[20]:
```
le= LabelEncoder()
lw=le.fit_transform(df['Holiday'])
df.drop("Holiday", axis=1, inplace=True)
df["Holiday"]=lw
df.rename(columns = {'Free Delivery':'Free_Delivery'}, inplace = True)
df['Discount']= df.Discount.map({'No':0,'Yes':1})
df['Free_Delivery']= df.Free_Delivery.map({'No':0,'Yes':1})
df['Purchase']= df.Purchase.map({'No':0,'Yes':1 })
df.head()
```

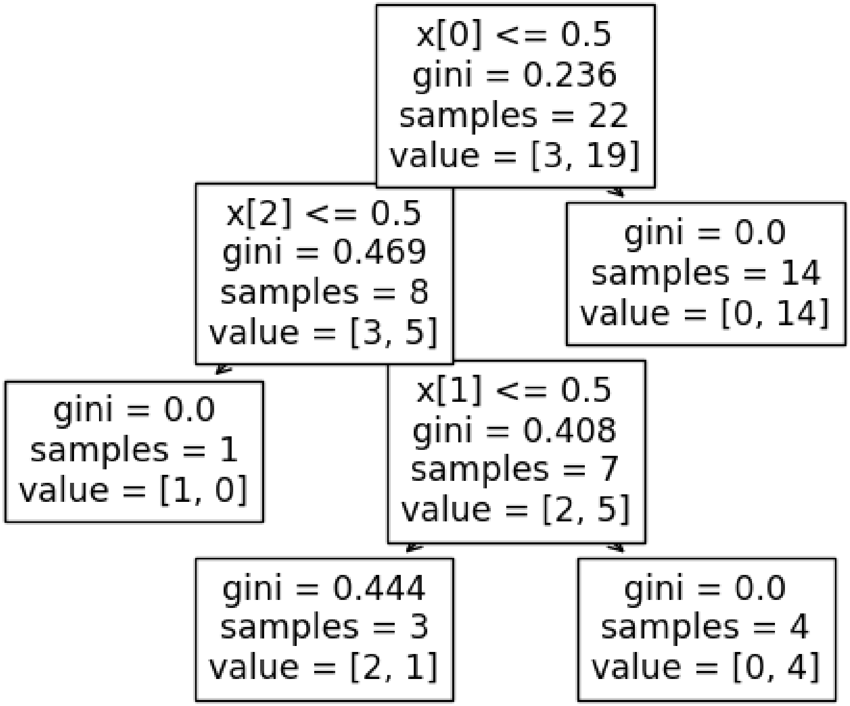
[20]:

	Discount	Free_Delivery	Purchase	Holiday
0	1	1	1	0
1	1	1	1	0
2	0	0	0	0
3	1	1	1	1
4	1	1	1	1

[21]:
```
X2=df.drop("Purchase",axis=1)
y2=df[['Purchase']]
```

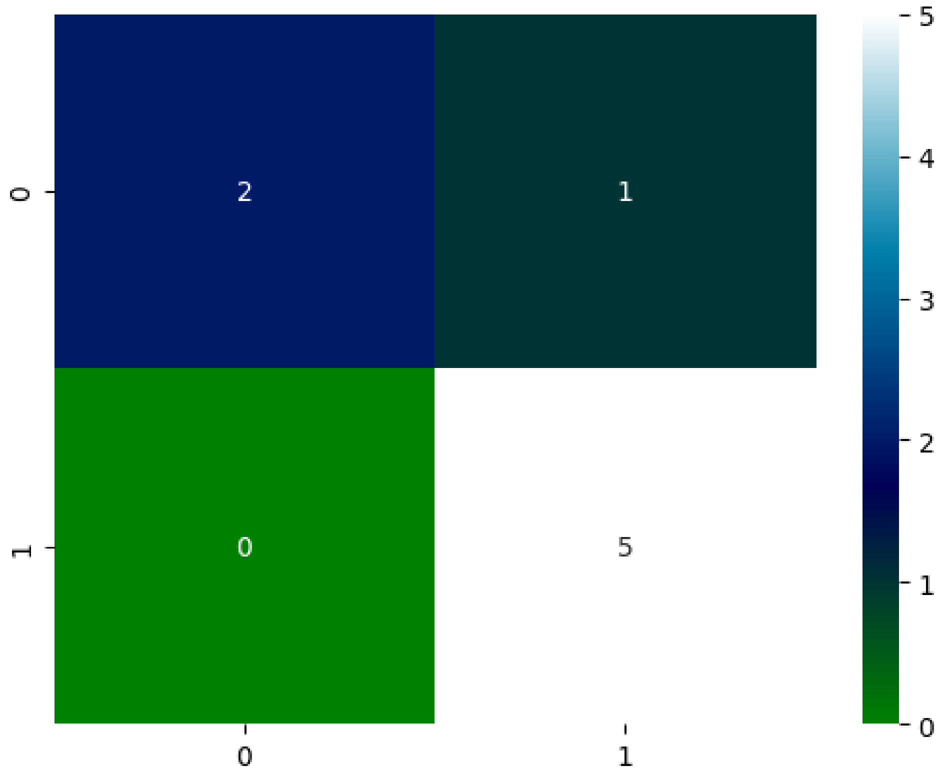
[22]:
```
x_train, x_test,y_train, y_test = tts(X2,y2,test_size=0.25, random_state=10)
#create model
from sklearn.tree import DecisionTreeClassifier as dtc
clf1= dtc(criterion='gini')
#train the dataset
clf1.fit(x_train, y_train)
tree.plot_tree(clf1)
y_pred = clf1.predict(x_test)
print(y_pred)
print(y_test)
```

```
[1 1 0 0 1 1 1 1]
    Purchase
20         1
7          1
5          0
2          0
3          1
21         0
13         1
27         1
```



[23]:
```
cn2= metrics.confusion_matrix(y_test, y_pred)
sns.heatmap(cn2, annot=True, cmap='ocean')
```

[23]: <Axes: >



[24]:
```
acc2 = metrics.accuracy_score(y_test, y_pred)
print("Accuracy is",acc2*100)
print(mat.classification_report(y_test,y_pred))
```

```
Accuracy is 87.5
              precision    recall  f1-score   support

           0       1.00      0.67      0.80         3
           1       0.83      1.00      0.91         5

    accuracy                           0.88         8
   macro avg       0.92      0.83      0.85         8
weighted avg       0.90      0.88      0.87         8
```