DecisionTreeClass

March 28, 2024

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

[ ]:
```
data = pd.read_csv('cancer.csv')
data.head()
```

[ ]:

	index	id	radius_mean	texture_mean	perimeter_mean	area_mean	\
0	1	842302	17.99	10.38	122.80	1001.0	
1	2	842517	20.57	17.77	132.90	1326.0	
2	3	84300903	19.69	21.25	130.00	1203.0	
3	4	84348301	11.42	20.38	77.58	386.1	
4	5	84358402	20.29	14.34	135.10	1297.0	

	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean	\
0	0.11840	0.27760	0.3001	0.14710	
1	0.08474	0.07864	0.0869	0.07017	
2	0.10960	0.15990	0.1974	0.12790	
3	0.14250	0.28390	0.2414	0.10520	
4	0.10030	0.13280	0.1980	0.10430	

	…	smoothness_worst	compactness_worst	concavity_worst	\
0	…	0.1622	0.6656	0.7119	
1	…	0.1238	0.1866	0.2416	
2	…	0.1444	0.4245	0.4504	
3	…	0.2098	0.8663	0.6869	
4	…	0.1374	0.2050	0.4000	

	concave points_worst	symmetry_worst	fractal_dimension_worst	N Stage	\
0	0.2654	0.4601	0.11890	N1	
1	0.1860	0.2750	0.08902	N2	
2	0.2430	0.3613	0.08758	N3	
3	0.2575	0.6638	0.17300	N1	
4	0.1625	0.2364	0.07678	N1	

	6th Stage	differentiate	diagnosis

```
0        IIA      Poorly differentiated          M
1       IIIA  Moderately differentiated          M
2       IIIC  Moderately differentiated          M
3        IIA      Poorly differentiated          M
4        IIB      Poorly differentiated          M

[5 rows x 36 columns]
```

[ ]:
```
# Prepare the model
y = data["diagnosis"] # our target variable
X = data.drop(["diagnosis","index","id"], axis=1) # our predictors
X.shape
```

[ ]:
```
(569, 33)
```

[ ]:
```
# Taking care missing data
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X.iloc[:, 0:29])
X.iloc[:, 0:29] = imputer.transform(X.iloc[:, 0:29])
```

[ ]:
```
# One hot encoding
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [30,31,32])],␣
 ↪remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

[ ]:
```
# Spliting the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
 ↪random_state = 0)
```

[ ]:
```
#Feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train
```

[ ]:
```
array([[-1.49240501,  2.21735578, -0.40488817, …, -0.38664354,
         0.32349851, -0.7578486 ],
       [ 0.67005939, -0.45098762, -0.40488817, …, -1.48895322,
         0.62563098, -1.03071387],
       [-1.49240501,  2.21735578, -0.40488817, …,  0.71907312,
        -0.51329768, -0.96601386],
       …,
```

```
        [ 0.67005939, -0.45098762, -0.40488817, …, -1.01972052,
         -0.69995543, -0.12266325],
        [ 0.67005939, -0.45098762, -0.40488817, …, -1.80208475,
         -1.56206114, -1.00989735],
        [-1.49240501,  2.21735578, -0.40488817, …, -0.30719919,
         -1.24094654,  0.2126516 ]])
```

[ ]:
```
from sklearn.tree import DecisionTreeClassifier
```

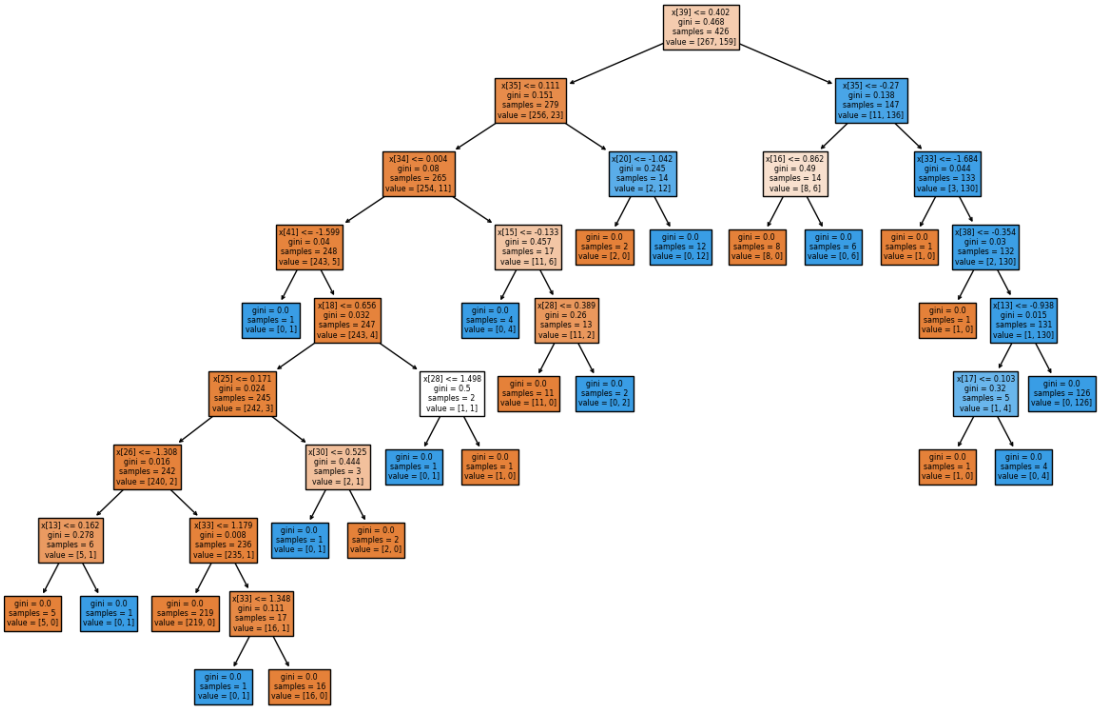
[ ]:
```
classifier=DecisionTreeClassifier()
classifier.fit(X_train,y_train)
```

[ ]:
```
DecisionTreeClassifier()
```

[ ]:
```
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(classifier,filled=True)
```

[ ]:
```
[Text(0.6390086206896551, 0.95, 'x[39] <= 0.402\ngini = 0.468\nsamples =
426\nvalue = [267, 159]'),
 Text(0.4849137931034483, 0.85, 'x[35] <= 0.111\ngini = 0.151\nsamples =
279\nvalue = [256, 23]'),
 Text(0.38362068965517243, 0.75, 'x[34] <= 0.004\ngini = 0.08\nsamples =
265\nvalue = [254, 11]'),
 Text(0.28448275862068967, 0.65, 'x[41] <= -1.599\ngini = 0.04\nsamples =
248\nvalue = [243, 5]'),
 Text(0.25, 0.55, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.31896551724137934, 0.55, 'x[18] <= 0.656\ngini = 0.032\nsamples =
247\nvalue = [243, 4]'),
 Text(0.22413793103448276, 0.45, 'x[25] <= 0.171\ngini = 0.024\nsamples =
245\nvalue = [242, 3]'),
 Text(0.13793103448275862, 0.35, 'x[26] <= -1.308\ngini = 0.016\nsamples =
242\nvalue = [240, 2]'),
 Text(0.06896551724137931, 0.25, 'x[13] <= 0.162\ngini = 0.278\nsamples =
6\nvalue = [5, 1]'),
 Text(0.034482758620689655, 0.15, 'gini = 0.0\nsamples = 5\nvalue = [5, 0]'),
 Text(0.10344827586206896, 0.15, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.20689655172413793, 0.25, 'x[33] <= 1.179\ngini = 0.008\nsamples =
236\nvalue = [235, 1]'),
 Text(0.1724137931034483, 0.15, 'gini = 0.0\nsamples = 219\nvalue = [219, 0]'),
 Text(0.2413793103448276, 0.15, 'x[33] <= 1.348\ngini = 0.111\nsamples =
17\nvalue = [16, 1]'),
 Text(0.20689655172413793, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.27586206896551724, 0.05, 'gini = 0.0\nsamples = 16\nvalue = [16, 0]'),
 Text(0.3103448275862069, 0.35, 'x[30] <= 0.525\ngini = 0.444\nsamples =
3\nvalue = [2, 1]'),
 Text(0.27586206896551724, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
```

```
 Text(0.3448275862068966, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.41379310344827586, 0.45, 'x[28] <= 1.498\ngini = 0.5\nsamples = 2\nvalue
= [1, 1]'),
 Text(0.3793103448275862, 0.35, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.4482758620689655, 0.35, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.4827586206896552, 0.65, 'x[15] <= -0.133\ngini = 0.457\nsamples =
17\nvalue = [11, 6]'),
 Text(0.4482758620689655, 0.55, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.5172413793103449, 0.55, 'x[28] <= 0.389\ngini = 0.26\nsamples =
13\nvalue = [11, 2]'),
 Text(0.4827586206896552, 0.45, 'gini = 0.0\nsamples = 11\nvalue = [11, 0]'),
 Text(0.5517241379310345, 0.45, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.5862068965517241, 0.75, 'x[20] <= -1.042\ngini = 0.245\nsamples =
14\nvalue = [2, 12]'),
 Text(0.5517241379310345, 0.65, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.6206896551724138, 0.65, 'gini = 0.0\nsamples = 12\nvalue = [0, 12]'),
 Text(0.7931034482758621, 0.85, 'x[35] <= -0.27\ngini = 0.138\nsamples =
147\nvalue = [11, 136]'),
 Text(0.7241379310344828, 0.75, 'x[16] <= 0.862\ngini = 0.49\nsamples =
14\nvalue = [8, 6]'),
 Text(0.6896551724137931, 0.65, 'gini = 0.0\nsamples = 8\nvalue = [8, 0]'),
 Text(0.7586206896551724, 0.65, 'gini = 0.0\nsamples = 6\nvalue = [0, 6]'),
 Text(0.8620689655172413, 0.75, 'x[33] <= -1.684\ngini = 0.044\nsamples =
133\nvalue = [3, 130]'),
 Text(0.8275862068965517, 0.65, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.896551724137931, 0.65, 'x[38] <= -0.354\ngini = 0.03\nsamples =
132\nvalue = [2, 130]'),
 Text(0.8620689655172413, 0.55, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.9310344827586207, 0.55, 'x[13] <= -0.938\ngini = 0.015\nsamples =
131\nvalue = [1, 130]'),
 Text(0.896551724137931, 0.45, 'x[17] <= 0.103\ngini = 0.32\nsamples = 5\nvalue
= [1, 4]'),
 Text(0.8620689655172413, 0.35, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.9310344827586207, 0.35, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.9655172413793104, 0.45, 'gini = 0.0\nsamples = 126\nvalue = [0, 126]')]
```



[ ]:
```
y_pred=classifier.predict(X_test)
y_pred
```

[ ]:
```
array(['M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B',
       'M', 'B', 'M', 'B', 'M', 'M', 'M', 'M', 'M', 'B', 'B', 'M', 'B',
       'B', 'M', 'B', 'M', 'B', 'M', 'B', 'M', 'B', 'M', 'B', 'M', 'B',
       'M', 'B', 'B', 'M', 'B', 'M', 'M', 'B', 'B', 'B', 'M', 'M', 'M',
       'M', 'B', 'B', 'B', 'M', 'B', 'M', 'M', 'M', 'M', 'B', 'B', 'M',
       'M', 'M', 'M', 'M', 'B', 'B', 'M', 'B', 'B', 'M', 'B', 'B', 'B',
       'B', 'B', 'M', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'M', 'M', 'B',
       'M', 'B', 'M', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'M',
       'M', 'B', 'M', 'B', 'M', 'B', 'B', 'M', 'M', 'B', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'B',
       'M', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'M', 'B', 'B', 'B', 'M'],
      dtype=object)
```

[ ]:
```
from sklearn.metrics import accuracy_score,classification_report
```

[ ]:
```
score=accuracy_score(y_pred,y_test)
print(score)
```

```
0.9300699300699301
```

[ ]:
```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           B       0.92      0.97      0.94        86
           M       0.94      0.88      0.91        57

    accuracy                           0.93       143
   macro avg       0.93      0.92      0.93       143
weighted avg       0.93      0.93      0.93       143
```