Practical 3 Decision Tree

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
print("Libraries imported ")

Libraries imported

df = pd.read\_csv('data.csv')  
print("dataframe of dataset created")

dataframe of dataset created

df.head()

id diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean \  
0 842302 M 17.99 10.38 122.80 1001.0   
1 842517 M 20.57 17.77 132.90 1326.0   
2 84300903 M 19.69 21.25 130.00 1203.0   
3 84348301 M 11.42 20.38 77.58 386.1   
4 84358402 M 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   
  
 ... texture\_worst perimeter\_worst area\_worst smoothness\_worst \  
0 ... 17.33 184.60 2019.0 0.1622   
1 ... 23.41 158.80 1956.0 0.1238   
2 ... 25.53 152.50 1709.0 0.1444   
3 ... 26.50 98.87 567.7 0.2098   
4 ... 16.67 152.20 1575.0 0.1374   
  
 compactness\_worst concavity\_worst concave points\_worst symmetry\_worst \  
0 0.6656 0.7119 0.2654 0.4601   
1 0.1866 0.2416 0.1860 0.2750   
2 0.4245 0.4504 0.2430 0.3613   
3 0.8663 0.6869 0.2575 0.6638   
4 0.2050 0.4000 0.1625 0.2364   
  
 fractal\_dimension\_worst Unnamed: 32   
0 0.11890 NaN   
1 0.08902 NaN   
2 0.08758 NaN   
3 0.17300 NaN   
4 0.07678 NaN   
  
[5 rows x 33 columns]

df = df[['diagnosis','radius\_mean','texture\_mean','perimeter\_mean','area\_mean','smoothness\_mean','compactness\_mean','concavity\_mean','concave points\_mean','radius\_worst','texture\_worst','perimeter\_worst','area\_worst','smoothness\_worst','compactness\_worst','concavity\_worst','concave points\_worst','symmetry\_worst','fractal\_dimension\_worst']]

df.tail()

diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean \  
564 M 21.56 22.39 142.00 1479.0   
565 M 20.13 28.25 131.20 1261.0   
566 M 16.60 28.08 108.30 858.1   
567 M 20.60 29.33 140.10 1265.0   
568 B 7.76 24.54 47.92 181.0   
  
 smoothness\_mean compactness\_mean concavity\_mean concave points\_mean \  
564 0.11100 0.11590 0.24390 0.13890   
565 0.09780 0.10340 0.14400 0.09791   
566 0.08455 0.10230 0.09251 0.05302   
567 0.11780 0.27700 0.35140 0.15200   
568 0.05263 0.04362 0.00000 0.00000   
  
 radius\_worst texture\_worst perimeter\_worst area\_worst \  
564 25.450 26.40 166.10 2027.0   
565 23.690 38.25 155.00 1731.0   
566 18.980 34.12 126.70 1124.0   
567 25.740 39.42 184.60 1821.0   
568 9.456 30.37 59.16 268.6   
  
 smoothness\_worst compactness\_worst concavity\_worst \  
564 0.14100 0.21130 0.4107   
565 0.11660 0.19220 0.3215   
566 0.11390 0.30940 0.3403   
567 0.16500 0.86810 0.9387   
568 0.08996 0.06444 0.0000   
  
 concave points\_worst symmetry\_worst fractal\_dimension\_worst   
564 0.2216 0.2060 0.07115   
565 0.1628 0.2572 0.06637   
566 0.1418 0.2218 0.07820   
567 0.2650 0.4087 0.12400   
568 0.0000 0.2871 0.07039

df.isnull().sum()

diagnosis 0  
radius\_mean 0  
texture\_mean 0  
perimeter\_mean 0  
area\_mean 0  
smoothness\_mean 0  
compactness\_mean 0  
concavity\_mean 0  
concave points\_mean 0  
radius\_worst 0  
texture\_worst 0  
perimeter\_worst 0  
area\_worst 0  
smoothness\_worst 0  
compactness\_worst 0  
concavity\_worst 0  
concave points\_worst 0  
symmetry\_worst 0  
fractal\_dimension\_worst 0  
dtype: int64

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['diagnosis'] = le.fit\_transform(df.diagnosis)

df.head()

diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean \  
0 1 17.99 10.38 122.80 1001.0   
1 1 20.57 17.77 132.90 1326.0   
2 1 19.69 21.25 130.00 1203.0   
3 1 11.42 20.38 77.58 386.1   
4 1 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   
  
 radius\_worst texture\_worst perimeter\_worst area\_worst smoothness\_worst \  
0 25.38 17.33 184.60 2019.0 0.1622   
1 24.99 23.41 158.80 1956.0 0.1238   
2 23.57 25.53 152.50 1709.0 0.1444   
3 14.91 26.50 98.87 567.7 0.2098   
4 22.54 16.67 152.20 1575.0 0.1374   
  
 compactness\_worst concavity\_worst concave points\_worst symmetry\_worst \  
0 0.6656 0.7119 0.2654 0.4601   
1 0.1866 0.2416 0.1860 0.2750   
2 0.4245 0.4504 0.2430 0.3613   
3 0.8663 0.6869 0.2575 0.6638   
4 0.2050 0.4000 0.1625 0.2364   
  
 fractal\_dimension\_worst   
0 0.11890   
1 0.08902   
2 0.08758   
3 0.17300   
4 0.07678

df.tail()

diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean \  
564 1 21.56 22.39 142.00 1479.0   
565 1 20.13 28.25 131.20 1261.0   
566 1 16.60 28.08 108.30 858.1   
567 1 20.60 29.33 140.10 1265.0   
568 0 7.76 24.54 47.92 181.0   
  
 smoothness\_mean compactness\_mean concavity\_mean concave points\_mean \  
564 0.11100 0.11590 0.24390 0.13890   
565 0.09780 0.10340 0.14400 0.09791   
566 0.08455 0.10230 0.09251 0.05302   
567 0.11780 0.27700 0.35140 0.15200   
568 0.05263 0.04362 0.00000 0.00000   
  
 radius\_worst texture\_worst perimeter\_worst area\_worst \  
564 25.450 26.40 166.10 2027.0   
565 23.690 38.25 155.00 1731.0   
566 18.980 34.12 126.70 1124.0   
567 25.740 39.42 184.60 1821.0   
568 9.456 30.37 59.16 268.6   
  
 smoothness\_worst compactness\_worst concavity\_worst \  
564 0.14100 0.21130 0.4107   
565 0.11660 0.19220 0.3215   
566 0.11390 0.30940 0.3403   
567 0.16500 0.86810 0.9387   
568 0.08996 0.06444 0.0000   
  
 concave points\_worst symmetry\_worst fractal\_dimension\_worst   
564 0.2216 0.2060 0.07115   
565 0.1628 0.2572 0.06637   
566 0.1418 0.2218 0.07820   
567 0.2650 0.4087 0.12400   
568 0.0000 0.2871 0.07039

df['radius\_mean'] = le.fit\_transform(df.radius\_mean)

df.tail()

diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean \  
564 1 438 22.39 142.00 1479.0   
565 1 413 28.25 131.20 1261.0   
566 1 340 28.08 108.30 858.1   
567 1 429 29.33 140.10 1265.0   
568 0 3 24.54 47.92 181.0   
  
 smoothness\_mean compactness\_mean concavity\_mean concave points\_mean \  
564 0.11100 0.11590 0.24390 0.13890   
565 0.09780 0.10340 0.14400 0.09791   
566 0.08455 0.10230 0.09251 0.05302   
567 0.11780 0.27700 0.35140 0.15200   
568 0.05263 0.04362 0.00000 0.00000   
  
 radius\_worst texture\_worst perimeter\_worst area\_worst \  
564 25.450 26.40 166.10 2027.0   
565 23.690 38.25 155.00 1731.0   
566 18.980 34.12 126.70 1124.0   
567 25.740 39.42 184.60 1821.0   
568 9.456 30.37 59.16 268.6   
  
 smoothness\_worst compactness\_worst concavity\_worst \  
564 0.14100 0.21130 0.4107   
565 0.11660 0.19220 0.3215   
566 0.11390 0.30940 0.3403   
567 0.16500 0.86810 0.9387   
568 0.08996 0.06444 0.0000   
  
 concave points\_worst symmetry\_worst fractal\_dimension\_worst   
564 0.2216 0.2060 0.07115   
565 0.1628 0.2572 0.06637   
566 0.1418 0.2218 0.07820   
567 0.2650 0.4087 0.12400   
568 0.0000 0.2871 0.07039

# X- Features y- Label  
X = df[['radius\_mean','texture\_mean','perimeter\_mean','area\_mean','smoothness\_mean','compactness\_mean','concavity\_mean','concave points\_mean','radius\_worst','texture\_worst','perimeter\_worst','area\_worst','smoothness\_worst','compactness\_worst','concavity\_worst','concave points\_worst','symmetry\_worst','fractal\_dimension\_worst']]  
y= df['diagnosis']

from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=101)

from sklearn.tree import DecisionTreeClassifier  
clf = DecisionTreeClassifier(random\_state=0,criterion='gini')  
clf.fit(X\_train,y\_train)

DecisionTreeClassifier(random\_state=0)

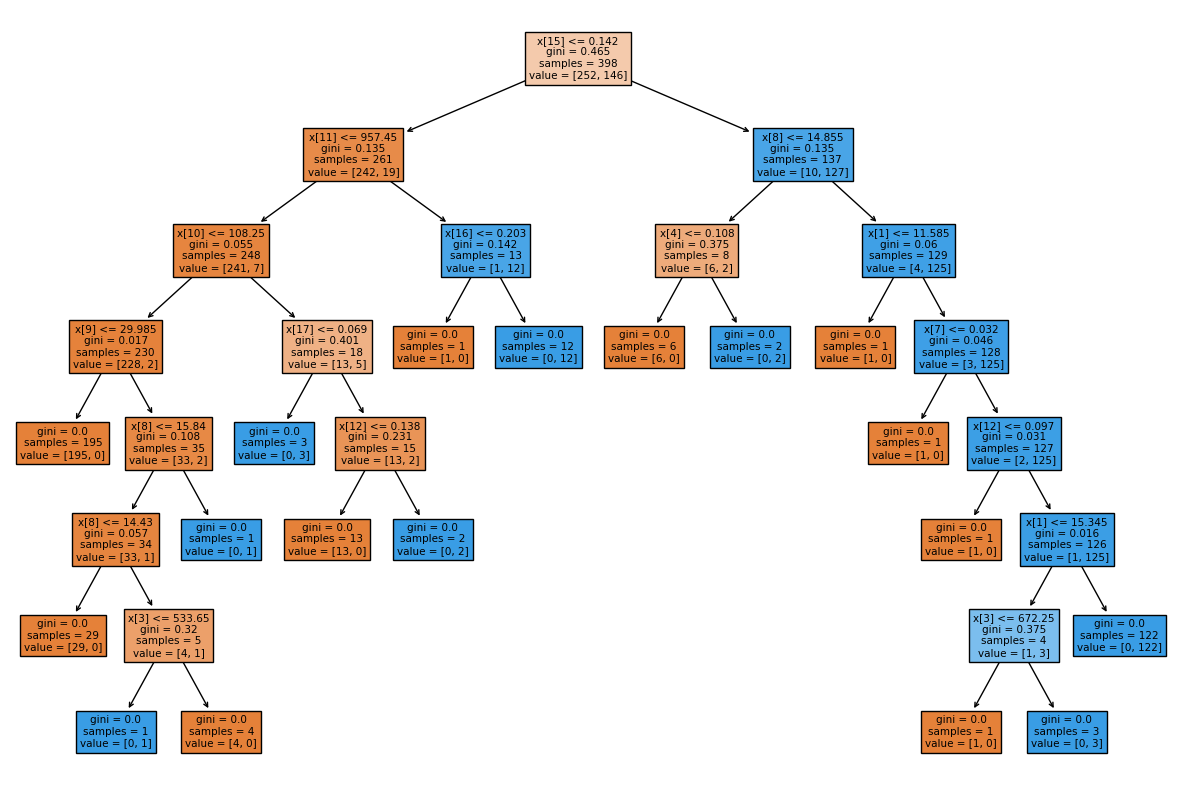
from sklearn.metrics import accuracy\_score  
import math  
predictions\_test=clf.predict(X\_test)  
print("Accuracy : ",accuracy\_score(y\_test, predictions\_test)\*100)

Accuracy : 91.22807017543859

predictions\_train = clf.predict(X\_train)  
accuracy\_score(y\_train,predictions\_train)

1.0

from sklearn import tree  
plt.figure(figsize=(15,10))  
tree.plot\_tree(clf,filled=True)  
plt.show()



from sklearn.metrics import classification\_report,confusion\_matrix  
print(classification\_report(y\_test,predictions\_test))  
print(confusion\_matrix(y\_test,predictions\_test))

precision recall f1-score support  
  
 0 0.92 0.93 0.93 105  
 1 0.89 0.88 0.89 66  
  
 accuracy 0.91 171  
 macro avg 0.91 0.91 0.91 171  
weighted avg 0.91 0.91 0.91 171  
  
[[98 7]  
 [ 8 58]]

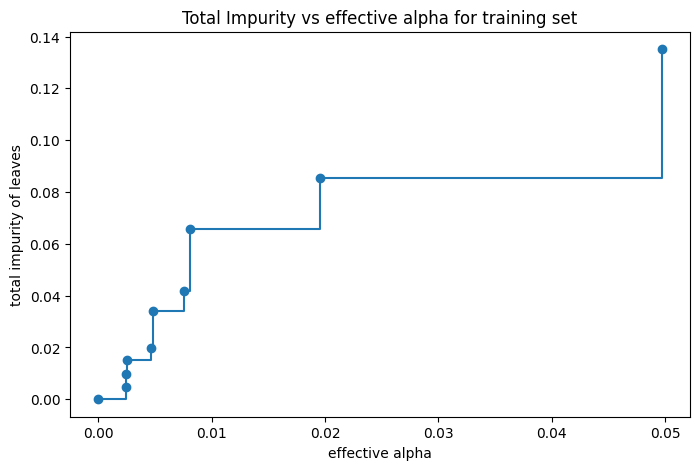
from sklearn.metrics import roc\_curve,auc  
from sklearn.metrics import roc\_auc\_score  
dt\_probs = clf.predict\_proba(X\_test)[:,1]  
fpr\_dt, tpr\_dt, thresholds\_dt = roc\_curve(y\_test,dt\_probs)  
print("FPR :",fpr\_dt)  
print("TPR :",tpr\_dt)  
print("Threshold :",thresholds\_dt)

FPR : [0. 0.06666667 1. ]  
TPR : [0. 0.87878788 1. ]  
Threshold : [inf 1. 0.]

path = clf.cost\_complexity\_pruning\_path(X\_train, y\_train)  
ccp\_alphas, impurities = path.ccp\_alphas, path.impurities

fig, ax = plt.subplots(figsize=(8,5))  
ax.plot(ccp\_alphas[:-1], impurities[:-1], marker='o', drawstyle="steps-post")  
ax.set\_xlabel("effective alpha")  
ax.set\_ylabel("total impurity of leaves")  
ax.set\_title("Total Impurity vs effective alpha for training set")

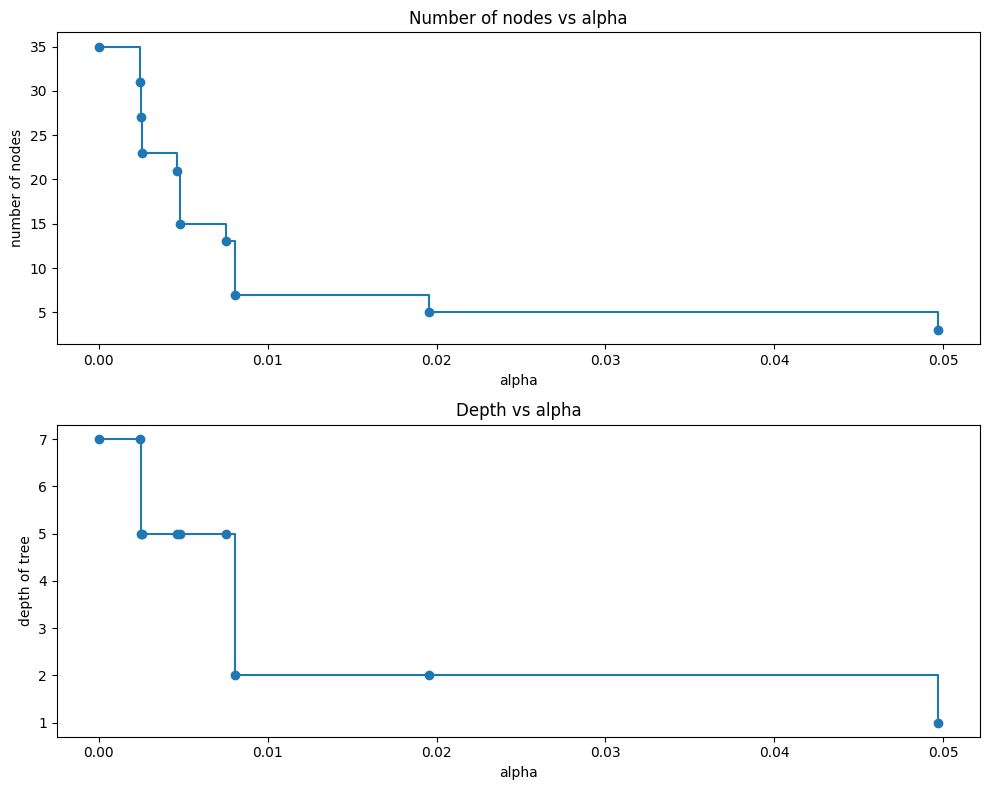
Text(0.5, 1.0, 'Total Impurity vs effective alpha for training set')



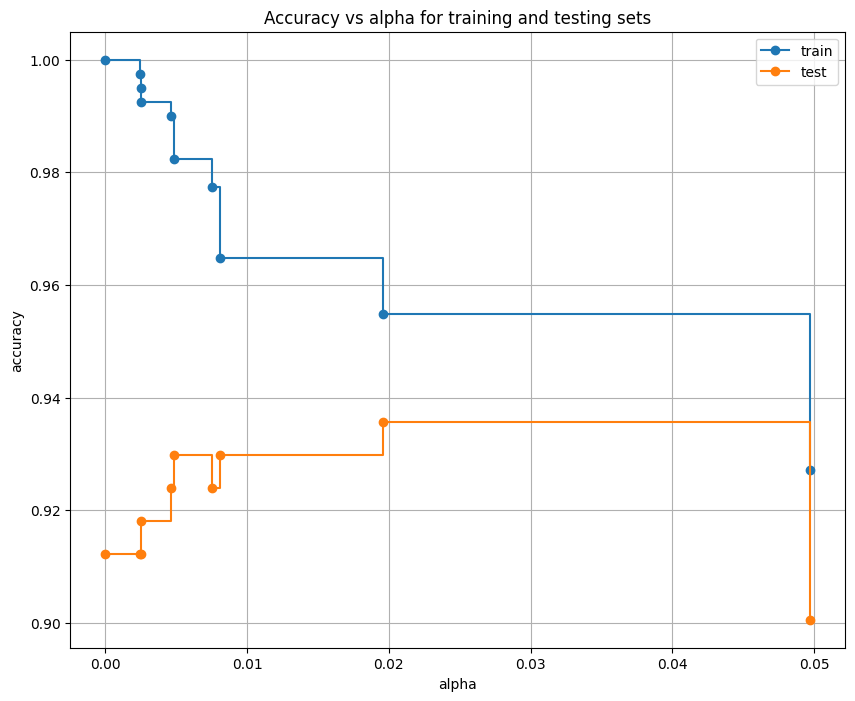
clfs = []  
for ccp\_alpha in ccp\_alphas:  
 clf = DecisionTreeClassifier(random\_state=0, ccp\_alpha=ccp\_alpha)  
 clf.fit(X\_train, y\_train)  
 clfs.append(clf)  
print("Number of nodes in the last tree is: {} with ccp\_alpha: {}".format(  
 clfs[-1].tree\_.node\_count, ccp\_alphas[-1]))

Number of nodes in the last tree is: 1 with ccp\_alpha: 0.3294235167086822

clfs = clfs[:-1]  
ccp\_alphas = ccp\_alphas[:-1]  
  
node\_counts = [clf.tree\_.node\_count for clf in clfs]  
depth = [clf.tree\_.max\_depth for clf in clfs]  
fig, ax = plt.subplots(2, 1,figsize=(10,8))  
ax[0].plot(ccp\_alphas, node\_counts, marker='o', drawstyle="steps-post")  
ax[0].set\_xlabel("alpha")  
ax[0].set\_ylabel("number of nodes")  
ax[0].set\_title("Number of nodes vs alpha")  
ax[1].plot(ccp\_alphas, depth, marker='o', drawstyle="steps-post")  
ax[1].set\_xlabel("alpha")  
ax[1].set\_ylabel("depth of tree")  
ax[1].set\_title("Depth vs alpha")  
fig.tight\_layout()



train\_scores = [clf.score(X\_train, y\_train) for clf in clfs]  
test\_scores = [clf.score(X\_test, y\_test) for clf in clfs]  
  
fig, ax = plt.subplots(figsize=(10,8))  
ax.set\_xlabel("alpha")  
ax.set\_ylabel("accuracy")  
ax.set\_title("Accuracy vs alpha for training and testing sets")  
ax.plot(ccp\_alphas, train\_scores, marker='o', label="train",  
 drawstyle="steps-post")  
ax.plot(ccp\_alphas, test\_scores, marker='o', label="test",  
 drawstyle="steps-post")  
ax.legend()  
plt.grid()  
plt.show()



clf = DecisionTreeClassifier(random\_state=0, ccp\_alpha=0.016)  
clf.fit(X\_train,y\_train)

DecisionTreeClassifier(ccp\_alpha=0.016, random\_state=0)

from sklearn.metrics import accuracy\_score  
pred=clf.predict(X\_test)  
accuracy\_score(y\_test, pred)

0.9298245614035088

pred\_1 = clf.predict(X\_train)  
accuracy\_score(y\_train,pred\_1)

0.964824120603015

from sklearn import tree  
plt.figure(figsize=(15,10))  
tree.plot\_tree(clf,filled=True)

[Text(0.5, 0.8333333333333334, 'x[15] <= 0.142\ngini = 0.465\nsamples = 398\nvalue = [252, 146]'),  
 Text(0.25, 0.5, 'x[11] <= 957.45\ngini = 0.135\nsamples = 261\nvalue = [242, 19]'),  
 Text(0.125, 0.16666666666666666, 'gini = 0.055\nsamples = 248\nvalue = [241, 7]'),  
 Text(0.375, 0.16666666666666666, 'gini = 0.142\nsamples = 13\nvalue = [1, 12]'),  
 Text(0.75, 0.5, 'x[8] <= 14.855\ngini = 0.135\nsamples = 137\nvalue = [10, 127]'),  
 Text(0.625, 0.16666666666666666, 'gini = 0.375\nsamples = 8\nvalue = [6, 2]'),  
 Text(0.875, 0.16666666666666666, 'gini = 0.06\nsamples = 129\nvalue = [4, 125]')]

