Breast cancer dataset

Problem Statement: Analysing Diagnosis based on remaining parameters.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\BreastCancerPrediction.csv")
df
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poi
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	
...	...	...	...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	

569 rows × 33 columns

In [3]:
```
pd.set_option('display.max_rows',100000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [4]:
```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

```
This DataFrame has 569 Rows and 33 columns
```

In [5]:
```
df.head()
```

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	point
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

In [6]:
```
1 df.tail()
```

Out[6]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	(point
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	

In [7]:
```
1 df.describe()
```

Out[7]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	co points_
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.0
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.0
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.0
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.0
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.0
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.0
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.0
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.2

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [9]:
```
convert={'diagnosis':{'M':1,'B':2}}
df=df.replace(convert)
df
```

109	864018	2	11.340	21.26	72.48	396.5	0.08759	0.06575	0.05133
110	864033	2	9.777	16.99	62.50	290.2	0.10370	0.08404	0.04334
111	86408	2	12.630	20.76	82.15	480.4	0.09933	0.12090	0.10650
112	86409	2	14.260	19.65	97.83	629.9	0.07837	0.22330	0.30030
113	864292	2	10.510	20.19	68.64	334.2	0.11220	0.13030	0.06476
114	864496	2	8.726	15.83	55.84	230.9	0.11500	0.08201	0.04132
115	864685	2	11.930	21.53	76.53	438.6	0.09768	0.07849	0.03328
116	864726	2	8.950	15.76	58.74	245.2	0.09462	0.12430	0.09263
117	864729	1	14.870	16.67	98.64	682.5	0.11620	0.16490	0.16900
118	864877	1	15.780	22.91	105.70	782.6	0.11550	0.17520	0.21330
119	865128	1	17.950	20.01	114.20	982.0	0.08402	0.06722	0.07293

In [10]:
```
features=df.columns[2:31]
```

In [11]:
```
target=df.columns[1]
```

In [28]:
```
x=np.array(df[features])
y=np.array(df[target])
```

In [13]:
```
from sklearn.linear_model import LogisticRegression
lor = LogisticRegression(max_iter=10000)
```

In [14]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=2)
```

In [15]:
```
lor.fit(x_train,y_train)
```

Out[15]:
```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [16]:
```
score=lor.score(x_test,y_test)
print(score)
```

```
0.9473684210526315
```

Decision Tree

In [17]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [18]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [19]:
```
clf.fit(x_train,y_train)
```

Out[19]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [20]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.9181286549707602
```

Random Forest

In [21]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[21]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [22]:
```
rf=RandomForestClassifier()
```

In [30]:
```
x=df.drop('diagnosis',axis=1)
y=df['diagnosis']
```

In [31]:
```
params={'max_depth':[2,3,5,10,20],
       'min_samples_leaf':[5,10,20,50,100,200],
       'n_estimators':[10,25,30,50,100,200]}
```

In [32]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[32]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [33]:
```
grid_search.best_score_
```

Out[33]:
```
0.9623115577889447
```

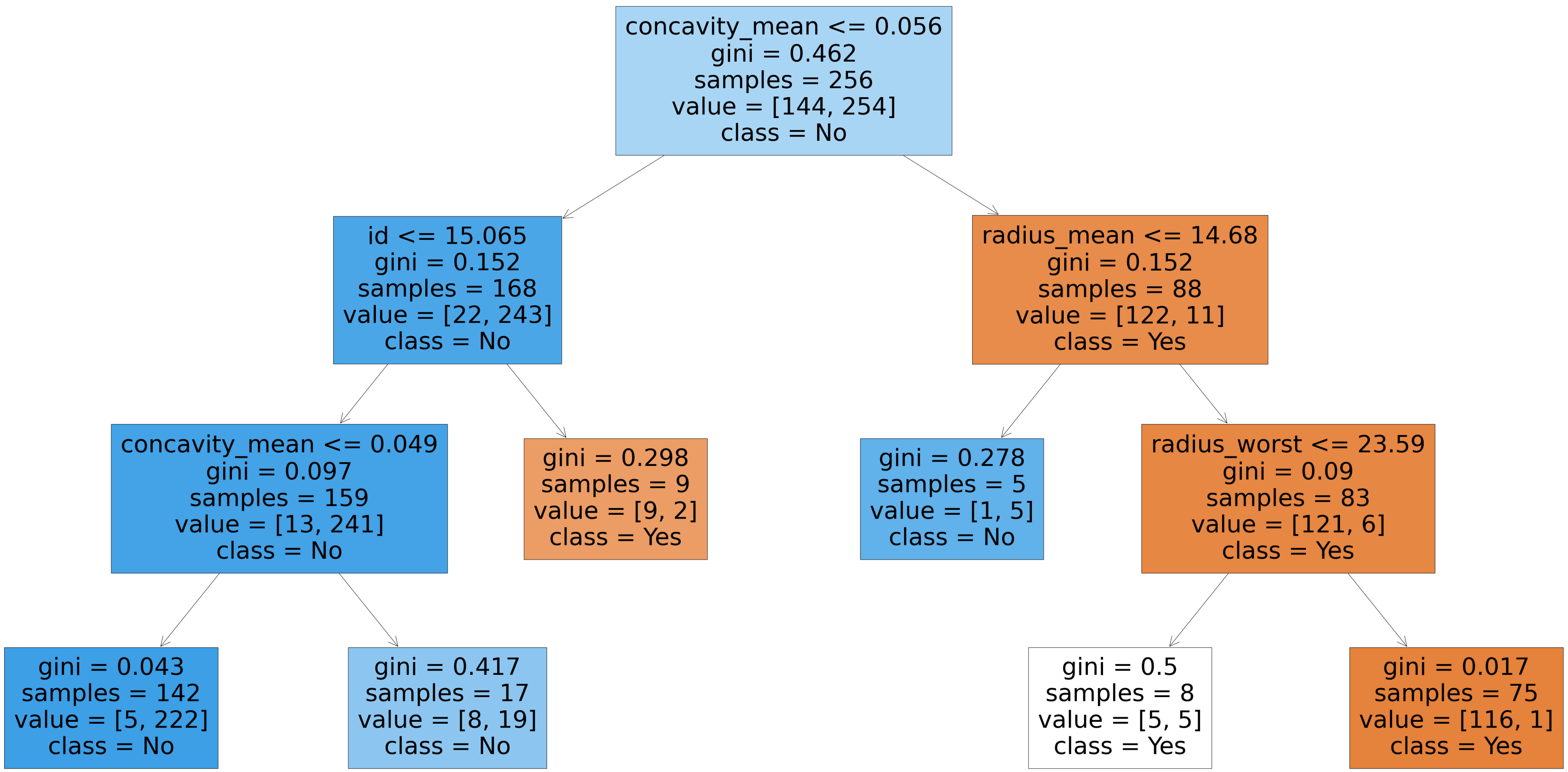
In [34]:
```
rf_best=grid_search.best_estimator_
```

In [35]:

```
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True)
```

Out[35]:

```
[Text(0.5, 0.875, 'concavity_mean <= 0.056\ngini = 0.462\nsamples = 256\nvalue = [144, 254]\nclass = No'),
 Text(0.3, 0.625, 'id <= 15.065\ngini = 0.152\nsamples = 168\nvalue = [22, 243]\nclass = No'),
 Text(0.2, 0.375, 'concavity_mean <= 0.049\ngini = 0.097\nsamples = 159\nvalue = [13, 241]\nclass = No'),
 Text(0.1, 0.125, 'gini = 0.043\nsamples = 142\nvalue = [5, 222]\nclass = No'),
 Text(0.3, 0.125, 'gini = 0.417\nsamples = 17\nvalue = [8, 19]\nclass = No'),
 Text(0.4, 0.375, 'gini = 0.298\nsamples = 9\nvalue = [9, 2]\nclass = Yes'),
 Text(0.7, 0.625, 'radius_mean <= 14.68\ngini = 0.152\nsamples = 88\nvalue = [122, 11]\nclass = Yes'),
 Text(0.6, 0.375, 'gini = 0.278\nsamples = 5\nvalue = [1, 5]\nclass = No'),
 Text(0.8, 0.375, 'radius_worst <= 23.59\ngini = 0.09\nsamples = 83\nvalue = [121, 6]\nclass = Yes'),
 Text(0.7, 0.125, 'gini = 0.5\nsamples = 8\nvalue = [5, 5]\nclass = Yes'),
 Text(0.9, 0.125, 'gini = 0.017\nsamples = 75\nvalue = [116, 1]\nclass = Yes')]
```



Conclusion

The Accuracy for LogisticRegression is 0.9473684210526315

The Accuracy for DecisionTree is 0.9181286549707602

The Accuracy for RandomForest is 0.9623115577889447

In [ ]:

```
 
```