BREAST CANCER

PROBLEM STATEMENT:To predict and study using the breast cancer diagnostic data set

In [1]:
```
import numpy as ny
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

Data Collection

In [2]:
```
df=pd.read_csv(r"C:\Users\91756\Documents\python\BreastCancerPrediction.csv")
df
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	
...	...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	

569 rows × 33 columns

Data CLeaning

In [4]:
```
df.head()
```

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 33 columns

In [5]:
```
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 [6]:
```
df.describe()
```

Out[6]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compa
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	

8 rows × 32 columns

In [7]:
```
x=['area_se','symmetry_mean']
y=['M','B']
all_inputs=df[x]
all_classes=df['diagnosis']
```

In [8]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.7
```

Decision Tree

In [9]:
```
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
```

Out[9]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [10]:
```
clf.score(x_test,y_test)
```

Out[10]: 0.8621553884711779

Random Forest

In [11]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [12]:
```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

Out[12]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [22]:
```
rf=RandomForestClassifier()
params={'max_depth':[2,3,4,5,6],"min_samples_leaf":[5,10,20,50,100,200],"n_estimators
```

In [23]:
```
from sklearn.model_selection import GridSearchCV
```

In [24]:
```
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
grid_search.fit(x_train,y_train)
```

Out[24]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [25]:
```
grid_search.best_score_
```

Out[25]:
```
0.8823529411764706
```

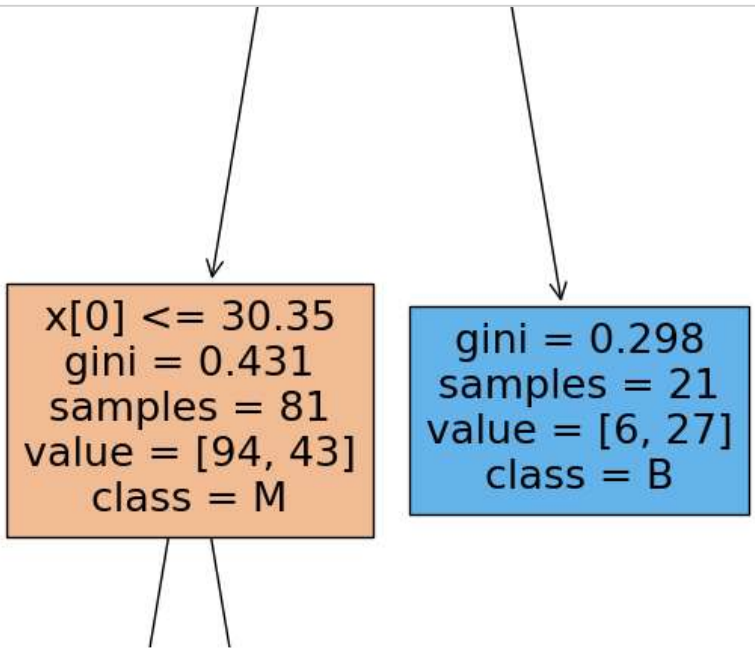
In [26]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=3, min_samples_leaf=20, n_estimators=61)
```

In [27]:
```
x=df.drop('diagnosis',axis=1)
y=df['diagnosis']
```

In [28]:
```
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
```

In [29]:
```
plt.figure(figsize=(10,40))
plot_tree(rf_best.estimators_[5],class_names=['M','B'],filled=True)
```



In [30]:
```
rf_best.feature_importances_
```

Out[30]:
```
array([0.68194581, 0.31805419])
```

In [33]:
```
imp_df=pd.DataFrame({'Varname':x_train.columns,'Imp':rf_best.feature_importances_})
imp_df.sort_values(by='Imp',ascending=False)
```

Out[33]:

	Varname	Imp
0	area_se	0.681946
1	symmetry_mean	0.318054

Conclusion

From the above the score for Decision Tree is 82.5% and for Random Forest is 88%. Compared to both Random Forest is highest in the accuracy.

In [ ]: