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\chait\Downloads\BreastCancerPrediction.csv")
df
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	sm
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	
...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	
565	926682	M	20.13	28.25	131.20	1261.0	
566	926954	M	16.60	28.08	108.30	858.1	
567	927241	M	20.60	29.33	140.10	1265.0	
568	92751	B	7.76	24.54	47.92	181.0	

569 rows × 33 columns

DATA CLEANING

In [3]:
```
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 [4]:
```
x=['area_se','symmetry_mean']
y=['M','B']
all_inputs=df[x]
all_classes=df['diagnosis']
```

In [5]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_siz
```

DECISION TREE

In [6]:
```
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
```

Out[6]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [7]:
```
clf.score(x_test,y_test)
```

Out[7]:
```
0.8421052631578947
```

RANDOM FOREST

In [8]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [9]:
```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

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

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

In [11]:
```
from sklearn.model_selection import GridSearchCV
```

In [12]:
```
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
grid_search.fit(x_train,y_train)
```

Out[12]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [13]:
```
grid_search.best_score_
```

Out[13]:
```
0.8743718592964824
```

In [14]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

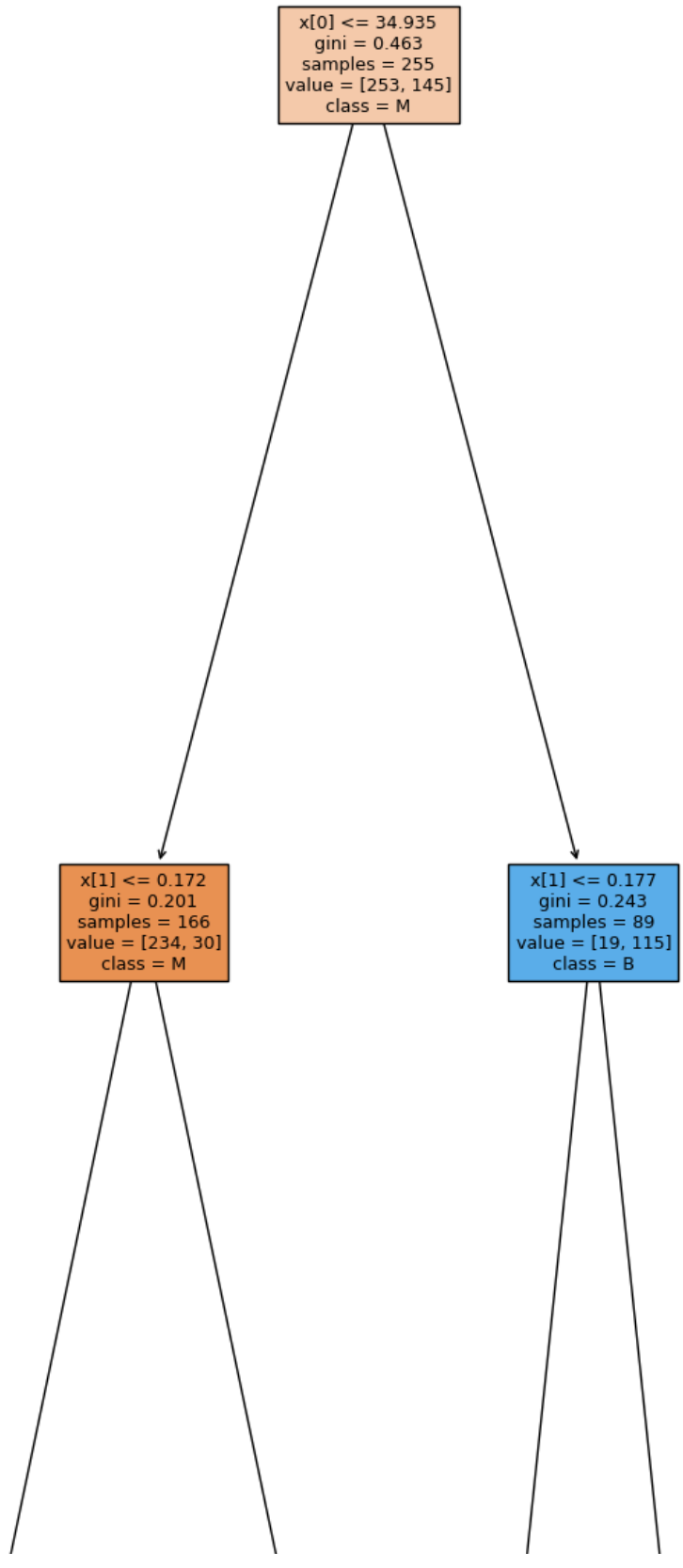
```
RandomForestClassifier(max_depth=5, min_samples_leaf=20, n_estimators=10)
```

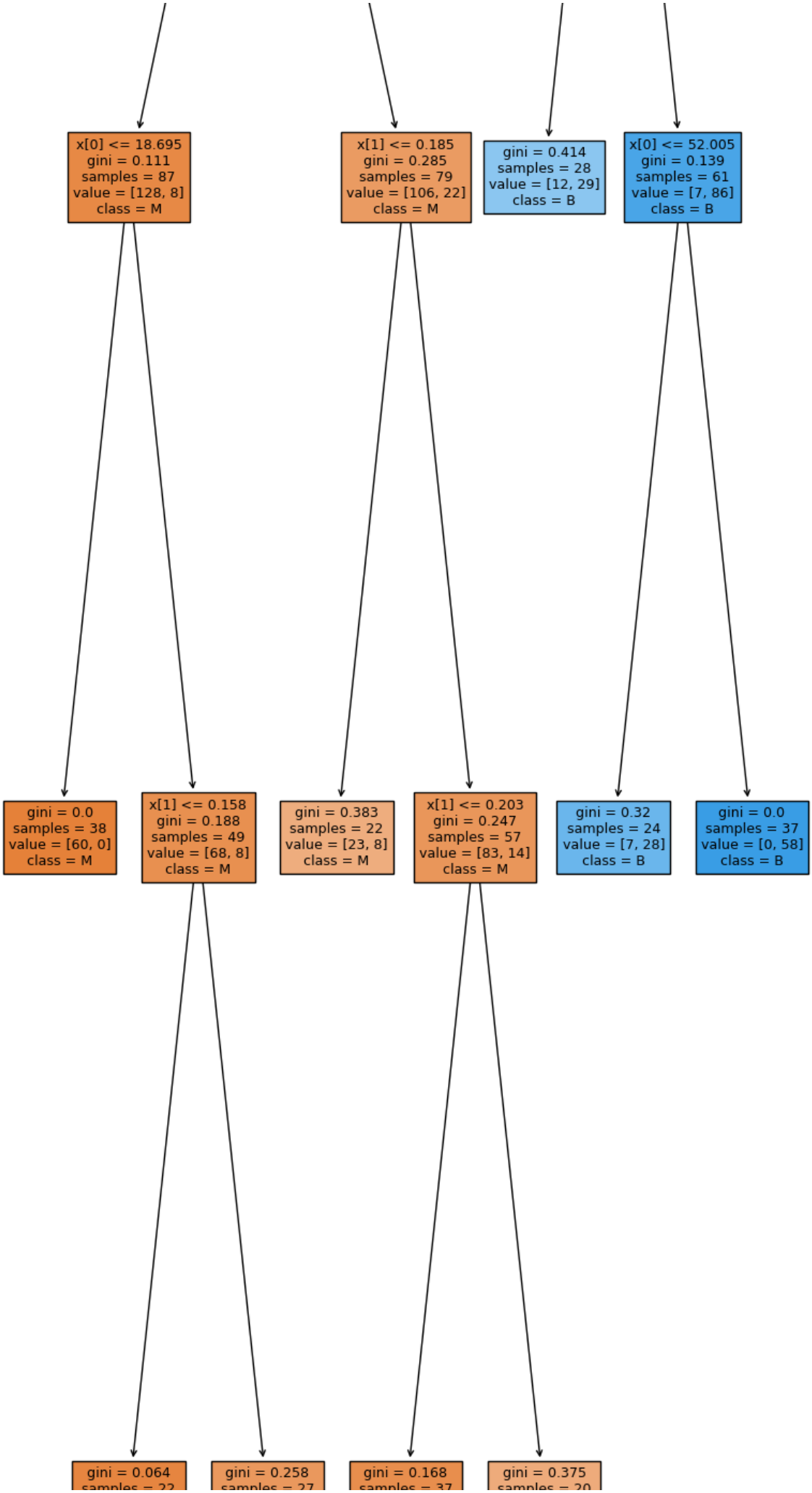
In [15]:
```
x=df.drop('diagnosis',axis=1)
y=df['diagnosis']
```

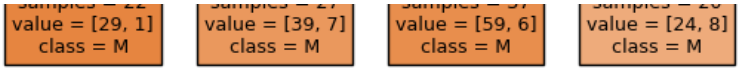
In [16]:
```
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
```

In [17]:
```
plt.figure(figsize=(10,40))
plot_tree(rf_best.estimators_[5],class_names=['M','B'],filled=True)
```

Out[17]:
```
[Text(0.5416666666666666, 0.9, 'x[0] <= 34.935\ngini = 0.463\nsamples = 255\nva
lue = [253, 145]\nclass = M'),
 Text(0.3333333333333333, 0.7, 'x[1] <= 0.172\ngini = 0.201\nsamples = 166\nval
ue = [234, 30]\nclass = M'),
 Text(0.16666666666666666, 0.5, 'x[0] <= 18.695\ngini = 0.111\nsamples = 87\nva
lue = [128, 8]\nclass = M'),
 Text(0.08333333333333333, 0.3, 'gini = 0.0\nsamples = 38\nvalue = [60, 0]\ncla
ss = M'),
 Text(0.25, 0.3, 'x[1] <= 0.158\ngini = 0.188\nsamples = 49\nvalue = [68, 8]\nc
lass = M'),
 Text(0.16666666666666666, 0.1, 'gini = 0.064\nsamples = 22\nvalue = [29, 1]\nc
lass = M'),
 Text(0.3333333333333333, 0.1, 'gini = 0.258\nsamples = 27\nvalue = [39, 7]\ncl
ass = M'),
 Text(0.5, 0.5, 'x[1] <= 0.185\ngini = 0.285\nsamples = 79\nvalue = [106, 22]\n
class = M'),
 Text(0.4166666666666667, 0.3, 'gini = 0.383\nsamples = 22\nvalue = [23, 8]\ncl
ass = M'),
 Text(0.5833333333333334, 0.3, 'x[1] <= 0.203\ngini = 0.247\nsamples = 57\nvalu
e = [83, 14]\nclass = M'),
 Text(0.5, 0.1, 'gini = 0.168\nsamples = 37\nvalue = [59, 6]\nclass = M'),
 Text(0.6666666666666666, 0.1, 'gini = 0.375\nsamples = 20\nvalue = [24, 8]\ncl
ass = M'),
 Text(0.75, 0.7, 'x[1] <= 0.177\ngini = 0.243\nsamples = 89\nvalue = [19, 115]
\nclass = B'),
 Text(0.6666666666666666, 0.5, 'gini = 0.414\nsamples = 28\nvalue = [12, 29]\nc
lass = B'),
 Text(0.8333333333333334, 0.5, 'x[0] <= 52.005\ngini = 0.139\nsamples = 61\nval
ue = [7, 86]\nclass = B'),
 Text(0.75, 0.3, 'gini = 0.32\nsamples = 24\nvalue = [7, 28]\nclass = B'),
 Text(0.9166666666666666, 0.3, 'gini = 0.0\nsamples = 37\nvalue = [0, 58]\nclas
s = B')]
```


x[0] <= 34.935
gini = 0.463
samples = 255
value = [253, 145]
class = M
x[1] <= 0.172
gini = 0.201
samples = 166
value = [234, 30]
class = M
x[1] <= 0.177
gini = 0.243
samples = 89
value = [19, 115]
class = B


x[0] <= 18.695
gini = 0.111
samples = 87
value = [128, 8]
class = M
x[1] <= 0.185
gini = 0.285
samples = 79
value = [106, 22]
class = M
gini = 0.414
samples = 28
value = [12, 29]
class = B
x[0] <= 52.005
gini = 0.139
samples = 61
value = [7, 86]
class = B
gini = 0.0
samples = 38
value = [60, 0]
class = M
x[1] <= 0.158
gini = 0.188
samples = 49
value = [68, 8]
class = M
gini = 0.383
samples = 22
value = [23, 8]
class = M
x[1] <= 0.203
gini = 0.247
samples = 57
value = [83, 14]
class = M
gini = 0.32
samples = 24
value = [7, 28]
class = B
gini = 0.0
samples = 37
value = [0, 58]
class = B
gini = 0.064
gini = 0.258
gini = 0.168
gini = 0.375



In [18]:
```
rf_best.feature_importances_
```

Out[18]:
```
array([0.82035361, 0.17964639])
```

In [19]:
```
imp_df=pd.DataFrame({'Varname':x_train.columns,'Imp':rf_best.feature_importances
imp_df.sort_values(by='Imp',ascending=False)
```

Out[19]:

	Varname	Imp
0	area_se	0.820354
1	symmetry_mean	0.179646

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.