WEEK - 07

Assessment Bulid decision tree-based model in python for like Breast cancer Wisconsin(diagnostic) dataset from sci-kit learn Or any classification dataset from UCI, Kaggle

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [13]:
```
data=pd.read_csv(r'C:\Users\God\Downloads\data.csv')
data
```

Out[13]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
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 [14]:
```
data.head()
```

Out[14]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	p
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	

5 rows × 33 columns

In [15]:
```
data.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  date                    569 non-null    int64
dtypes: float64(30), int64(2), object(1)
memory usage: 146.8+ KB
```

In [16]:
```
data=data.drop(["id"],axis=1)
```

In [17]:
```
data.head(3)
```

Out[17]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	M	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710
1	M	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017
2	M	19.69	21.25	130.0	1203.0	0.10960	0.15990	0.1974	0.12790

3 rows × 32 columns

In [18]:
```
M=data[data.diagnosis=="M"]
```

In [20]:
```
M.head(5)
```

Out[20]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

5 rows × 32 columns

In [21]:
```
B=data[data.diagnosis=="B"]
```

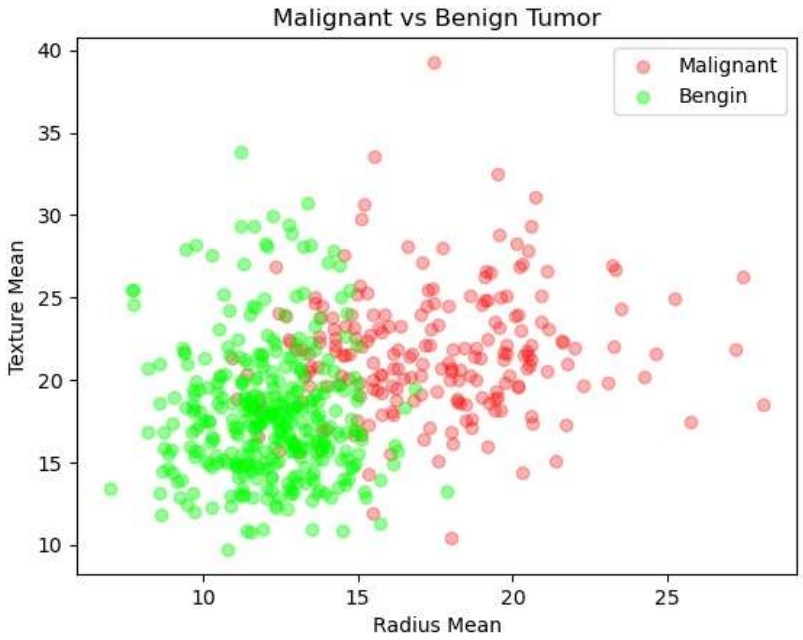
In [22]:
```
B.head(5)
```

Out[22]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concav points_mea
19	B	13.540	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781
20	B	13.080	15.71	85.63	520.0	0.10750	0.12700	0.04568	0.03110
21	B	9.504	12.44	60.34	273.9	0.10240	0.06492	0.02956	0.02076
37	B	13.030	18.42	82.61	523.8	0.08983	0.03766	0.02562	0.02923
46	B	8.196	16.84	51.71	201.9	0.08600	0.05943	0.01588	0.00591

5 rows × 32 columns

In [24]:
```
plt.title("Malignant vs Benign Tumor")
plt.xlabel("Radius Mean")
plt.ylabel("Texture Mean")
plt.scatter(M.radius_mean,M.texture_mean,color='red',label='Malignant',alpha=0.3)
plt.scatter(B.radius_mean,B.texture_mean,color='lime',label='Bengin',alpha=0.4)
plt.legend()
plt.show()
```



In [25]:
```
data.diagnosis=[1 if i == "M" else 0 for i in data.diagnosis]
```

In [59]:
```
x=data.drop(["date"],axis=1)
y=data.date.values
```

In [61]:
```
#Normalization
x=(x-np.min(x)) / (np.max(x)-np.min(x))
```

In [62]:
```
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=42)
```

DecisionTreeClassifier

It's type of supervised ML used to categorize or make prediction based on how a previous set of question were answerd

In [63]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [64]:
```
dt=DecisionTreeClassifier()
```

In [65]:
```
dt.fit(x_train, y_train)
```

Out[65]:
```
DecisionTreeClassifier()
```

In [66]:
```
dt.score(x_test,y_test)
```

Out[66]:
```
1.0
```