In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [32]:

```
df=pd.read_csv("Breast_Cancer.csv")
df.sample(10)
```

Out[32]:

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Norr Nucle
326	809912	10	3	3	1	2	10	7	
189	1212422	4	1	1	1	2	1	3	
32	1074610	2	1	1	2	2	1	3	
493	1297327	5	1	1	1	2	1	1	
564	824249	1	1	1	1	2	1	3	
637	1324681	4	1	1	1	2	1	2	
384	1196475	3	2	1	1	2	1	2	
93	1164066	1	1	1	1	2	1	3	
94	1165297	2	1	1	2	2	1	1	
41	1102573	5	6	5	6	10	1	3	

In [3]:

```
df.shape
```

Out[3]:

```
(683, 11)
```

In [4]:

```
df.columns
```

Out[4]:

```
Index(['Sample code number', 'Clump Thickness', 'Uniformity of Cell Size',
       'Uniformity of Cell Shape', 'Marginal Adhesion',
       'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
       'Normal Nucleoli', 'Mitoses', 'Class'],
      dtype='object')
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
Sample code number             0
Clump Thickness                0
Uniformity of Cell Size        0
Uniformity of Cell Shape       0
Marginal Adhesion              0
Single Epithelial Cell Size    0
Bare Nuclei                    0
Bland Chromatin                0
Normal Nucleoli                0
Mitoses                        0
Class                          0
dtype: int64
```

In [6]:

```
# Info of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683 entries, 0 to 682
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Sample code number           683 non-null    int64
 1   Clump Thickness              683 non-null    int64
 2   Uniformity of Cell Size      683 non-null    int64
 3   Uniformity of Cell Shape     683 non-null    int64
 4   Marginal Adhesion            683 non-null    int64
 5   Single Epithelial Cell Size  683 non-null    int64
 6   Bare Nuclei                  683 non-null    int64
 7   Bland Chromatin              683 non-null    int64
 8   Normal Nucleoli              683 non-null    int64
 9   Mitoses                      683 non-null    int64
 10  Class                        683 non-null    int64
dtypes: int64(11)
memory usage: 58.8 KB
```

In [7]:

```
df.Class.value_counts()
```

Out[7]:

```
2    444
4    239
Name: Class, dtype: int64
```

In [8]:

```
df.corr()["Class"]
```

Out[8]:

```
Sample code number             -0.084701
Clump Thickness                 0.714790
Uniformity of Cell Size         0.820801
Uniformity of Cell Shape        0.821891
Marginal Adhesion               0.706294
Single Epithelial Cell Size     0.690958
Bare Nuclei                     0.822696
Bland Chromatin                 0.758228
Normal Nucleoli                 0.718677
Mitoses                         0.423448
Class                           1.000000
Name: Class, dtype: float64
```

In [9]:

```
df.drop("Sample code number",axis=1,inplace=True)
```

In [10]:

```
df.head()
```

Out[10]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses
0	5	1	1	1	2	1	3	1	1
1	5	4	4	5	7	10	3	2	1
2	3	1	1	1	2	2	3	1	1
3	6	8	8	1	3	4	3	7	1
4	4	1	1	3	2	1	3	1	1

In [11]:

```
def mymodel(obj):
 obj.fit(xtrain,ytrain)
 ypred=obj.predict(xtest)
 print(classification_report(ytest,ypred))
```

In [12]:

```
x=df.drop("Class",axis=1)
x.head()
```

Out[12]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses
0	5	1	1	1	2	1	3	1	1
1	5	4	4	5	7	10	3	2	1
2	3	1	1	1	2	2	3	1	1
3	6	8	8	1	3	4	3	7	1
4	4	1	1	3	2	1	3	1	1

In [13]:

```
y=df["Class"]
y.head()
```

Out[13]:

```
0    2
1    2
2    2
3    2
4    2
Name: Class, dtype: int64
```

To build the best model, we have to train and test the dataset with multiple Machine Learning algorithms then we can find the best ML model. So let’s try.

First, we need to import the required packages.

In [14]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,train_size=0.70,random_state=1)
```

In [15]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

In [16]:

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

In [17]:

```
lr=LogisticRegression()
knn=KNeighborsClassifier()
svc=SVC()
dt=DecisionTreeClassifier()
```

In [18]:

```
mymodel(lr)
```

```
              precision    recall  f1-score   support

           2       0.97      0.97      0.97       133
           4       0.94      0.94      0.94        72

    accuracy                           0.96       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.96      0.96      0.96       205
```

In [19]:

```
mymodel(knn)
```

```
              precision    recall  f1-score   support

           2       0.98      0.96      0.97       133
           4       0.93      0.97      0.95        72

    accuracy                           0.97       205
   macro avg       0.96      0.97      0.96       205
weighted avg       0.97      0.97      0.97       205
```

In [20]:

```
mymodel(svc)
```

```
              precision    recall  f1-score   support

           2       1.00      0.97      0.98       133
           4       0.95      1.00      0.97        72

    accuracy                           0.98       205
   macro avg       0.97      0.98      0.98       205
weighted avg       0.98      0.98      0.98       205
```

In [21]:

```
mymodel(dt)
```

```
              precision    recall  f1-score   support

           2       0.97      0.98      0.97       133
           4       0.96      0.94      0.95        72

    accuracy                           0.97       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.97      0.97      0.97       205
```

In [22]:

```
dt1=DecisionTreeClassifier(max_depth=5)
mymodel(dt1)
```

```
              precision    recall  f1-score   support

           2       0.98      0.97      0.97       133
           4       0.95      0.96      0.95        72

    accuracy                           0.97       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.97      0.97      0.97       205
```

In [23]:

```
for i in range(1,50):
 dt1.fit(xtrain,ytrain)
 ypred=dt1.predict(xtest)
 dt1=DecisionTreeClassifier(max_depth=i)
 print(f"{i}-------{accuracy_score(ytest,ypred)}")
```

```
1-------0.9658536585365853
2-------0.9219512195121952
3-------0.9512195121951219
4-------0.9463414634146341
5-------0.9609756097560975
6-------0.9560975609756097
7-------0.9609756097560975
8-------0.9609756097560975
9-------0.9707317073170731
10-------0.9658536585365853
11-------0.9560975609756097
12-------0.9658536585365853
13-------0.9707317073170731
14-------0.9560975609756097
15-------0.9609756097560975
16-------0.9560975609756097
17-------0.9512195121951219
18-------0.9609756097560975
19-------0.9560975609756097
20-------0.9512195121951219
21-------0.9707317073170731
22-------0.9560975609756097
23-------0.9658536585365853
24-------0.9658536585365853
25-------0.9560975609756097
26-------0.9560975609756097
27-------0.9560975609756097
28-------0.9609756097560975
29-------0.9560975609756097
30-------0.9658536585365853
31-------0.9560975609756097
32-------0.9560975609756097
33-------0.9609756097560975
34-------0.9609756097560975
35-------0.9512195121951219
36-------0.9609756097560975
37-------0.9609756097560975
38-------0.9609756097560975
39-------0.9609756097560975
40-------0.9658536585365853
41-------0.9707317073170731
42-------0.9609756097560975
43-------0.9560975609756097
44-------0.9560975609756097
45-------0.9609756097560975
46-------0.9609756097560975
47-------0.9609756097560975
48-------0.9658536585365853
49-------0.9609756097560975
```

In [24]:

```
dt3=DecisionTreeClassifier(max_depth=12)
mymodel(dt3)
```

```
              precision    recall  f1-score   support

           2       0.97      0.97      0.97       133
           4       0.94      0.94      0.94        72

    accuracy                           0.96       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.96      0.96      0.96       205
```

In [25]:

```
dt4=DecisionTreeClassifier(min_samples_leaf=12)
mymodel(dt4)
```

```
              precision    recall  f1-score   support

           2       0.98      0.95      0.97       133
           4       0.92      0.96      0.94        72

    accuracy                           0.96       205
   macro avg       0.95      0.96      0.95       205
weighted avg       0.96      0.96      0.96       205
```

In [26]:

```
for i in range(1,50):
 dt4.fit(xtrain,ytrain)
 ypred=dt4.predict(xtest)
 dt4=DecisionTreeClassifier(min_samples_leaf=i)
 print(f"{i}-------{accuracy_score(ytest,ypred)}")
```

```
1-------0.9560975609756097
2-------0.9609756097560975
3-------0.9463414634146341
4-------0.9609756097560975
5-------0.9512195121951219
6-------0.9512195121951219
7-------0.9512195121951219
8-------0.9463414634146341
9-------0.9560975609756097
10-------0.9560975609756097
11-------0.9560975609756097
12-------0.9560975609756097
13-------0.9560975609756097
14-------0.9560975609756097
15-------0.9365853658536586
16-------0.9512195121951219
17-------0.9512195121951219
18-------0.9512195121951219
19-------0.9512195121951219
20-------0.9512195121951219
21-------0.9512195121951219
22-------0.9512195121951219
23-------0.9512195121951219
24-------0.9512195121951219
25-------0.9512195121951219
26-------0.9512195121951219
27-------0.9512195121951219
28-------0.9512195121951219
29-------0.9512195121951219
30-------0.9512195121951219
31-------0.9512195121951219
32-------0.9512195121951219
33-------0.9414634146341463
34-------0.9414634146341463
35-------0.9414634146341463
36-------0.9414634146341463
37-------0.9414634146341463
38-------0.9414634146341463
39-------0.9414634146341463
40-------0.9414634146341463
41-------0.9414634146341463
42-------0.9414634146341463
43-------0.9414634146341463
44-------0.9219512195121952
45-------0.9219512195121952
46-------0.9219512195121952
47-------0.9219512195121952
48-------0.9219512195121952
49-------0.9219512195121952
```

In [27]:

```
dt5=DecisionTreeClassifier(min_samples_leaf=4)
mymodel(dt5)
```

```
              precision    recall  f1-score   support

           2       0.98      0.94      0.96       133
           4       0.90      0.97      0.93        72

    accuracy                           0.95       205
   macro avg       0.94      0.96      0.95       205
weighted avg       0.95      0.95      0.95       205
```

In [28]:

```
dt6=DecisionTreeClassifier(min_samples_split=24)
mymodel(dt6)
```

```
              precision    recall  f1-score   support

           2       0.98      0.94      0.96       133
           4       0.90      0.96      0.93        72

    accuracy                           0.95       205
   macro avg       0.94      0.95      0.94       205
weighted avg       0.95      0.95      0.95       205
```

In [29]:

```
for i in range(3,50):
 dt6.fit(xtrain,ytrain)
 ypred=dt6.predict(xtest)
 dt6=DecisionTreeClassifier(min_samples_split=i)
 print(f"{i}-----{accuracy_score(ytest,ypred)}")
```

```
3-----0.9463414634146341
4-----0.9560975609756097
5-----0.9658536585365853
6-----0.9609756097560975
7-----0.9609756097560975
8-----0.9560975609756097
9-----0.9463414634146341
10-----0.9560975609756097
11-----0.9512195121951219
12-----0.9512195121951219
13-----0.9512195121951219
14-----0.9512195121951219
15-----0.9463414634146341
16-----0.9463414634146341
17-----0.9463414634146341
18-----0.9463414634146341
19-----0.9463414634146341
20-----0.9463414634146341
21-----0.9463414634146341
22-----0.9463414634146341
23-----0.9463414634146341
24-----0.9463414634146341
25-----0.9463414634146341
26-----0.9463414634146341
27-----0.9463414634146341
28-----0.9463414634146341
29-----0.9463414634146341
30-----0.9463414634146341
31-----0.9463414634146341
32-----0.9463414634146341
33-----0.9414634146341463
34-----0.9414634146341463
35-----0.9414634146341463
36-----0.9414634146341463
37-----0.9414634146341463
38-----0.9414634146341463
39-----0.9414634146341463
40-----0.9414634146341463
41-----0.9414634146341463
42-----0.9414634146341463
43-----0.9414634146341463
44-----0.9414634146341463
45-----0.9414634146341463
46-----0.9414634146341463
47-----0.9414634146341463
48-----0.9414634146341463
49-----0.9414634146341463
```

In [30]:

```
dt7=DecisionTreeClassifier(max_depth=12,min_samples_leaf=4,min_samples_split=4)
mymodel(dt7)
```

```
              precision    recall  f1-score   support

           2       0.98      0.94      0.96       133
           4       0.90      0.97      0.93        72

    accuracy                           0.95       205
   macro avg       0.94      0.96      0.95       205
weighted avg       0.95      0.95      0.95       205
```

In [31]:

```
dt8=DecisionTreeClassifier(max_depth=12)
mymodel(dt8)
```

```
              precision    recall  f1-score   support

           2       0.97      0.97      0.97       133
           4       0.94      0.94      0.94        72

    accuracy                           0.96       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.96      0.96      0.96       205
```

I have completed the Machine learning Project successfully with 97% accuracy which is great for ‘Breast Cancer Detection using Machine learning’ project.

To get more accuracy, we trained all supervised classification algorithms but you can try out a few of them which are always popular. After training all algorithms, we found that Logistic Regression and Random Forest are given high accuracy .

In [ ]:

```
```