In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
from sklearn.datasets import load_breast_cancer
data= load_breast_cancer()
```

In [3]:
```
data
```

Out[3]:
```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
```

```
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset\n--------------------------------------
------\n\n**Data Set Characteristics:**\n\n:Number of Instances: 569\n\n:Number of Attributes: 30 numeric, predictive attribu
tes and the class\n\n:Attribute Information:\n    - radius (mean of distances from center to points on the perimeter)\n    -
texture (standard deviation of gray-scale values)\n    - perimeter\n    - area\n    - smoothness (local variation in radius l
engths)\n    - compactness (perimeter^2 / area - 1.0)\n    - concavity (severity of concave portions of the contour)\n    - c
oncave points (number of concave portions of the contour)\n    - symmetry\n    - fractal dimension ("coastline approximation"
- 1)\n\n    The mean, standard error, and "worst" or largest (mean of the three\n    worst/largest values) of these features
were computed for each image,\n    resulting in 30 features.  For instance, field 0 is Mean Radius, field\n    10 is Radius S
E, field 20 is Worst Radius.\n\n    - class:\n            - WDBC-Malignant\n            - WDBC-Benign\n\n:Summary Statistic
s:\n\n===================================== ====== ======\n                                        Min    Max\n==============
====================== ====== ======\nradius (mean):                      6.981  28.11\ntexture (mean):
9.71   39.28\nperimeter (mean):                   43.79  188.5\narea (mean):                       143.5  2501.0\nsmooth
ness (mean):                   0.053  0.163\ncompactness (mean):                 0.019  0.345\nconcavity (mean):
0.0    0.427\nconcave points (mean):              0.0    0.201\nsymmetry (mean):                   0.106  0.304\nfractal
dimension (mean):            0.05   0.097\nradius (standard error):            0.112  2.873\ntexture (standard error):
0.36   4.885\nperimeter (standard error):         0.757  21.98\narea (standard error):             6.802  542.2\nsmoothn
ess (standard error):          0.002  0.031\ncompactness (standard error):       0.002  0.135\nconcavity (standard error):
0.0    0.396\nconcave points (standard error):     0.0    0.053\nsymmetry (standard error):         0.008  0.079\nfractal
dimension (standard error):   0.001  0.03\nradius (worst):                     7.93   36.04\ntexture (worst):
12.02  49.54\nperimeter (worst):                  50.41  251.2\narea (worst):                      185.2  4254.0\nsmooth
ness (worst):                  0.071  0.223\ncompactness (worst):                0.027  1.058\nconcavity (worst):
0.0    1.252\nconcave points (worst):            0.0    0.291\nsymmetry (worst):                  0.156  0.664\nfractal
dimension (worst):           0.055  0.208\n===================================== ====== ======\n\n:Missing Attribute Values:
None\n\n:Class Distribution: 212 - Malignant, 357 - Benign\n\n:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mang
asarian\n\n:Donor: Nick Street\n\n:Date: November, 1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) data
sets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a fine needle\naspirate (FNA) of a breast mas
s.  They describe\ncharacteristics of the cell nuclei present in the image.\n\nSeparating plane described above was obtained
using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Programming." Proceedings of
the 4th\nMidwest Artificial Intelligence and Cognitive Science Society,\npp. 97-101, 1992], a classification method which use
s linear\nprogramming to construct a decision tree.  Relevant features\nwere selected using an exhaustive search in the space
of 1-4\nfeatures and 1-3 separating planes.\n\nThe actual linear program used to obtain the separating plane\nin the 3-dimens
ional space is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Li
nearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis database is also available through the
UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n|details-start|\n**References**\n|d
etails-split|\n\n- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction\n  for breast tumor diagnosis.
IS&T/SPIE 1993 International Symposium on\n  Electronic Imaging: Science and Technology, volume 1905, pages 861-870,\n  San J
ose, CA, 1993.\n- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and\n  prognosis via linear program
ming. Operations Research, 43(4), pages 570-577,\n  July-August 1995.\n- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Mac
hine learning techniques\n  to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)\n  163-171.\n\n|de
tails-end|\n',
```

```
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'breast_cancer.csv',
 'data_module': 'sklearn.datasets.data'}
```

In [4]:
```
type(data)
```

Out[4]:
```
sklearn.utils._bunch.Bunch
```

In [5]:
```
data.feature_names
```

Out[5]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [6]:
```
data.target
```

Out[6]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

In [7]:
```
data.target_names
```

Out[7]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [8]:
```
df=pd.DataFrame(np.c_[data.data,data.target],columns=[list(data.feature_names)+['target']])
```

In [9]:
```
df
```

Out[9]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	w a
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60	20
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	19
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	17
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	5
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	20
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	17
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	11
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	18
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16	2

569 rows × 31 columns

In [10]:
```
x=df.drop('target',axis=1)
y=df['target']
```

```
C:\Users\91912\AppData\Local\Temp\ipykernel_3732\1217858060.py:1: PerformanceWarning: dropping on a non-lexsorted multi-index w
ithout a level parameter may impact performance.
  x=df.drop('target',axis=1)
```

In [11]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=23)
```

In [12]:
```
print('shape of xtrain',xtrain.shape)
print('shape of xtest',xtest.shape)
```

```
print('shape of ytrain',ytrain.shape)
print('shape of ytest',ytest.shape)
```

```
shape of xtrain (398, 30)
shape of xtest (171, 30)
shape of ytrain (398, 1)
shape of ytest (171, 1)
```

In [13]:
```
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators=100,criterion='gini')
classifier.fit(xtrain,ytrain)
```

```
C:\Users\91912\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:1474: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

Out[13]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [14]:
```
classifier.score(xtest,ytest)
```

Out[14]:
```
0.9766081871345029
```

In [17]:
```
y_pred=classifier.predict(xtest)
y_pred[:5]
```

Out[17]:
```
array([1., 1., 0., 1., 0.])
```

In [18]:
```
ytest[:5]
```

Out[18]:

	target
155	1.0
541	1.0
202	0.0
409	1.0
64	0.0

In [19]:
```
patient=[19.69,
21.25,
130,
1203,
0.1096,
0.1599,
0.1974,
0.1279,
0.2069,
0.05999,
0.7456,
0.7869,
4.585,
94.03,
0.00615,
0.04006,
0.03832,
0.02058,
0.0225,
0.004571,
23.57,
25.53,
152.5,
1709,
0.1444,
0.4245,
0.4504,
0.243,
```

```
0.3613,
0.08758]
```

In [23]:
```
patient=np.array([patient])
```

classifier.predict(patient)

In [24]:
```
print('target names:',data.target_names)
```

```
target names: ['malignant' 'benign']
```

In [26]:
```
pred=classifier.predict(patient)
if pred[0]==0:
    print('patient has cancer')
else:
    print('no cancer')
```

```
patient has cancer
```

In [ ]: