# Breast Cancer Classification

# Load Data

In [1]:
```
from sklearn.datasets import load_breast_cancer
cancer=load_breast_cancer()
```

In [2]:
```
cancer
```

Out[2]:
```
{'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, 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, 0, 1, 1, 0, 1, 0, 1, 1, 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, 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, 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, predic
tive attributes 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 lengths)\n        - compactness (perimeter^2 / area - 1.0)\n        - concavity (seve
rity of concave portions of the contour)\n        - concave points (number of concave portions of the contour)\n        - s
ymmetry\n        - fractal dimension ("coastline approximation" - 1)\n\n        The mean, standard error, and "worst" or la
rgest (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 SE, field 20 is Worst Radius.\n\n
- class:\n                - WDBC-Malignant\n                - WDBC-Benign\n\n    :Summary Statistics:\n\n    =============
====================== ====== ======\n                                           Min    Max\n    ========================
============ ====== ======\n    radius (mean):                        6.981  28.11\n    texture (mean):
9.71   39.28\n    perimeter (mean):                     43.79  188.5\n    area (mean):                          143.5  250
1.0\n    smoothness (mean):                    0.053  0.163\n    compactness (mean):                   0.019  0.345\n    co
ncavity (mean):                      0.0    0.427\n    concave points (mean):                0.0    0.201\n    symmetry (mea
n):                      0.106  0.304\n    fractal dimension (mean):             0.05   0.097\n    radius (standard error):
0.112  2.873\n    texture (standard error):             0.36   4.885\n    perimeter (standard error):           0.757  21.9
8\n    area (standard error):                6.802  542.2\n    smoothness (standard error):          0.002  0.031\n    comp
actness (standard error):        0.002  0.135\n    concavity (standard error):           0.0    0.396\n    concave points
(standard error):      0.0    0.053\n    symmetry (standard error):            0.008  0.079\n    fractal dimension (standar
d error):   0.001  0.03\n    radius (worst):                       7.93   36.04\n    texture (worst):
12.02  49.54\n    perimeter (worst):                    50.41  251.2\n    area (worst):                         185.2  425
4.0\n    smoothness (worst):                   0.071  0.223\n    compactness (worst):                  0.027  1.058\n    co
ncavity (worst):                     0.0    1.252\n    concave points (worst):               0.0    0.291\n    symmetry (wor
st):                     0.156  0.664\n    fractal dimension (worst):            0.055  0.208\n    =======================
============= ====== ======\n\n    :Missing Attribute Values: None\n\n    :Class Distribution: 212 - Malignant, 357 - Benig
n\n\n    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n    :Date: No
vember, 1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures
are computed from a digitized image of a fine needle\naspirate (FNA) of a breast mass.  They describe\ncharacteristics of t
he 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 Intelli
gence and Cognitive Science Society,\npp. 97-101, 1992], a classification method which uses linear\nprogramming to construc
t a decision tree.  Relevant features\nwere selected using an exhaustive search in the space of 1-4\nfeatures and 1-3 separ
ating planes.\n\nThe actual linear program used to obtain the separating plane\nin the 3-dimensional space is that describe
d in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Linearly Inseparable Set
s",\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.. topic:: References\n\n   - W.N. Street, W.H. Wolb
erg and O.L. Mangasarian. Nuclear feature extraction \n     for breast tumor diagnosis. IS&T/SPIE 1993 International Sympos
ium on \n     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L.
Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and \n     prognosis via linear programming. Operations
Research, 43(4), pages 570-577, \n     July-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine lea
rning techniques\n     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) \n     163-171.',
 '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 [3]:
```
cancer.keys()
```

Out[3]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

In [4]:
```
print(cancer['feature_names'])
```

```
['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']
```

In [5]:
```
print(cancer['data'][0])
```

```
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
```

In [6]:
```
cancer['data'].shape
```

Out[6]:
```
(569, 30)
```

In [7]:
```
import pandas as pd
import numpy as np
df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']], columns = np.append(cancer['feature_names'], ['target']))
df_cancer
```

Out[7]:

| | 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 | worst area | worst smoothness |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | 2019.0 | 0.16220 |
| **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 | 1956.0 | 0.12380 |
| **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 | 1709.0 | 0.14440 |
| **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 | 567.7 | 0.20980 |
| **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 | 1575.0 | 0.13740 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **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 | 2027.0 | 0.14100 |
| **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 | 1731.0 | 0.11660 |
| **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 | 1124.0 | 0.11390 |
| **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 | 1821.0 | 0.16500 |
| **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 | 268.6 | 0.08996 |

569 rows × 31 columns

In [8]:
```
df_cancer.shape
```

Out[8]:
```
(569, 31)
```

In [9]:
```
df_cancer.describe()
```

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 | per |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | ... | 569.000000 | 569.0 |
| **mean** | 14.127292 | 19.289649 | 91.969033 | 654.889104 | 0.096360 | 0.104341 | 0.088799 | 0.048919 | 0.181162 | 0.062798 | ... | 25.677223 | 107.2 |
| **std** | 3.524049 | 4.301036 | 24.298981 | 351.914129 | 0.014064 | 0.052813 | 0.079720 | 0.038803 | 0.027414 | 0.007060 | ... | 6.146258 | 33.6 |
| **min** | 6.981000 | 9.710000 | 43.790000 | 143.500000 | 0.052630 | 0.019380 | 0.000000 | 0.000000 | 0.106000 | 0.049960 | ... | 12.020000 | 50.4 |
| **25%** | 11.700000 | 16.170000 | 75.170000 | 420.300000 | 0.086370 | 0.064920 | 0.029560 | 0.020310 | 0.161900 | 0.057700 | ... | 21.080000 | 84.1 |
| **50%** | 13.370000 | 18.840000 | 86.240000 | 551.100000 | 0.095870 | 0.092630 | 0.061540 | 0.033500 | 0.179200 | 0.061540 | ... | 25.410000 | 97.6 |
| **75%** | 15.780000 | 21.800000 | 104.100000 | 782.700000 | 0.105300 | 0.130400 | 0.130700 | 0.074000 | 0.195700 | 0.066120 | ... | 29.720000 | 125.4 |
| **max** | 28.110000 | 39.280000 | 188.500000 | 2501.000000 | 0.163400 | 0.345400 | 0.426800 | 0.201200 | 0.304000 | 0.097440 | ... | 49.540000 | 251.2 |

8 rows × 31 columns

## Data Preprocessing

In [10]:
```
#check for duplicates
df_cancer.duplicated().sum()
```

Out[10]: 0

In [11]:
```
df_cancer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    float64
dtypes: float64(31)
memory usage: 137.9 KB
```

In [12]:
```
df_cancer.isna().sum()
```

Out[12]:
```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

In [13]:
```
# creating features and label
x=df_cancer.drop('target',axis=1)
y=df_cancer['target']
x
```

Out[13]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension | ... | worst radius | worst texture | worst perimeter | worst area | smo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.30010 | 0.14710 | 0.2419 | 0.07871 | ... | 25.380 | 17.33 | 184.60 | 2019.0 | |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.08690 | 0.07017 | 0.1812 | 0.05667 | ... | 24.990 | 23.41 | 158.80 | 1956.0 | |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.19740 | 0.12790 | 0.2069 | 0.05999 | ... | 23.570 | 25.53 | 152.50 | 1709.0 | |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.24140 | 0.10520 | 0.2597 | 0.09744 | ... | 14.910 | 26.50 | 98.87 | 567.7 | |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.19800 | 0.10430 | 0.1809 | 0.05883 | ... | 22.540 | 16.67 | 152.20 | 1575.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **564** | 21.56 | 22.39 | 142.00 | 1479.0 | 0.11100 | 0.11590 | 0.24390 | 0.13890 | 0.1726 | 0.05623 | ... | 25.450 | 26.40 | 166.10 | 2027.0 | |
| **565** | 20.13 | 28.25 | 131.20 | 1261.0 | 0.09780 | 0.10340 | 0.14400 | 0.09791 | 0.1752 | 0.05533 | ... | 23.690 | 38.25 | 155.00 | 1731.0 | |
| **566** | 16.60 | 28.08 | 108.30 | 858.1 | 0.08455 | 0.10230 | 0.09251 | 0.05302 | 0.1590 | 0.05648 | ... | 18.980 | 34.12 | 126.70 | 1124.0 | |
| **567** | 20.60 | 29.33 | 140.10 | 1265.0 | 0.11780 | 0.27700 | 0.35140 | 0.15200 | 0.2397 | 0.07016 | ... | 25.740 | 39.42 | 184.60 | 1821.0 | |
| **568** | 7.76 | 24.54 | 47.92 | 181.0 | 0.05263 | 0.04362 | 0.00000 | 0.00000 | 0.1587 | 0.05884 | ... | 9.456 | 30.37 | 59.16 | 268.6 | |

569 rows × 30 columns

In [14]:
```
y
```

Out[14]:
```
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0
      ...
564    0.0
565    0.0
566    0.0
567    0.0
568    1.0
Name: target, Length: 569, dtype: float64
```

In [15]:
```
# splitting data into training and test set
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test= train_test_split(x,y)
```

In [16]:
```
x_train.shape
```

Out[16]:
```
(426, 30)
```

In [17]:
```
y_train.shape
```

Out[17]:
```
(426,)
```

In [18]:
```
x_test.shape
```

Out[18]:
```
(143, 30)
```

In [19]:
```
y_test.shape
```

Out[19]:
```
(143,)
```

In [20]:
```
# scaling data
from sklearn.preprocessing import StandardScaler
scaler= StandardScaler()

x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

In [21]:
```
x_train
```

Out[21]:
```
array([[ 0.26807115, -1.43440407,  0.19973726, ..., -0.50071715,
        -0.68343436, -0.89751222],
       [-1.00248878,  0.1736082 , -0.89650681, ...,  0.45008549,
        -0.50631851,  1.8749575 ],
       [-0.26820414,  0.15296325, -0.22999041, ...,  0.2914225 ,
        -0.10700277,  2.13451237],
       ...,
       [ 0.11681402, -0.96874573,  0.12479403, ...,  0.35018657,
         0.38570133,  0.00669217],
       [-0.81272983,  0.08414675, -0.85066387, ..., -1.28609892,
        -0.57072427, -0.4488531 ],
       [-0.80172931, -0.29205013, -0.74582308, ...,  0.08721736,
         0.63849396,  2.38877019]])
```

In [22]:
```
x_test
```

Out[22]:
```
array([[ 0.58782491, -1.1919673 ,  0.55764373, ...,  0.45916577,
        -0.23985747, -0.74535711],
       [ 0.22774611,  0.67866282,  0.34484476, ...,  2.75513378,
         2.29385191,  3.00850199],
       [-0.57844802,  0.97156411, -0.52025955, ..., -0.02874906,
        -1.11457089,  0.76041698],
       ...,
       [-1.2712014 , -0.36987459, -1.25416101, ..., -0.95128873,
         0.19749924, -0.54110593],
       [-1.55034214, -1.07628359, -1.558292  , ..., -1.67779737,
         0.20078764, -0.26083893],
       [-1.68704462, -1.3962598 , -1.64313342, ..., -0.44537987,
         0.18763406,  1.51462603]])
```

## Logistic Regression

In [23]:
```
# fitting data to model
from sklearn.linear_model import LogisticRegression
log_reg=LogisticRegression()
log_reg.fit(x_train,y_train)
```

Out[23]:
```
▾ LogisticRegression
LogisticRegression()
```

In [24]:
```
# model predictions
y_pred=log_reg.predict(x_test)
```

In [25]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

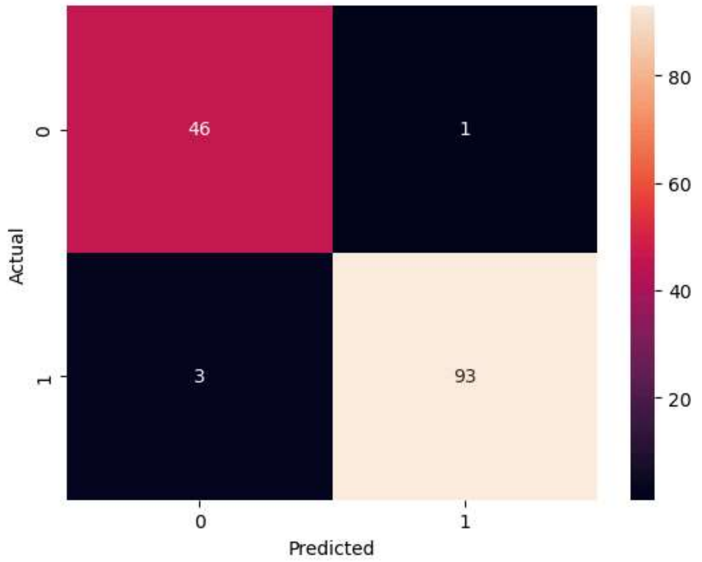
In [26]:
```
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[46  1]
 [ 3 93]]
```

In [27]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [28]:
```

cm=confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




In [29]:
```
# classification report
cr=classification_report(y_test,y_pred)
print("Classification Report:")
print(cr)
```

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.94      0.98      0.96        47
         1.0       0.99      0.97      0.98        96

    accuracy                           0.97       143
   macro avg       0.96      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143
```

In [30]:
```
# accuracy score
log_reg_acc = accuracy_score(y_test, y_pred)
print(log_reg_acc)
```

```
0.972027972027972
```

# Decision Tree Classifier

In [31]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
```

Out[31]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

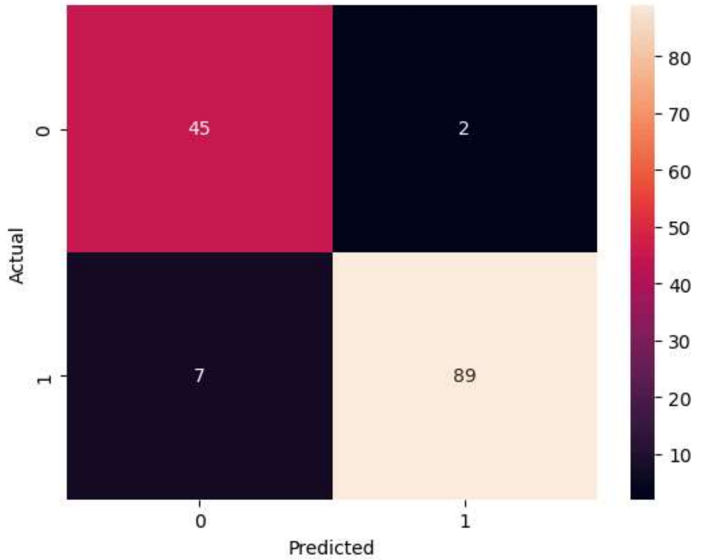
In [32]:
```
y_pred=dtc.predict(x_test)
y_pred
```

Out[32]:
```
array([1., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1., 0., 1., 0., 0., 1., 0.,
       1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0., 1., 0., 0., 1.,
       1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0.,
       0., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 1., 1., 0.,
       1., 1., 0., 0., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0.,
       1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1., 0., 1., 1.,
       1., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1.,
       0., 0., 1., 1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 1., 0., 1., 0.,
       0., 1., 1., 0., 1., 1., 1.])
```

In [33]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[45  2]
 [ 7 89]]
```

In [34]:
```
cm=confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




In [35]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.96      0.91        47
         1.0       0.98      0.93      0.95        96

    accuracy                           0.94       143
   macro avg       0.92      0.94      0.93       143
weighted avg       0.94      0.94      0.94       143
```

In [36]:
```
# accuracy score
dtc_acc = accuracy_score(y_test, y_pred)
print(dtc_acc)
```

```
0.9370629370629371
```

## Random Forest Classifier

In [37]:
```
from sklearn.ensemble import RandomForestClassifier
rand_clf=RandomForestClassifier()
rand_clf.fit(x_train,y_train)
```

Out[37]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

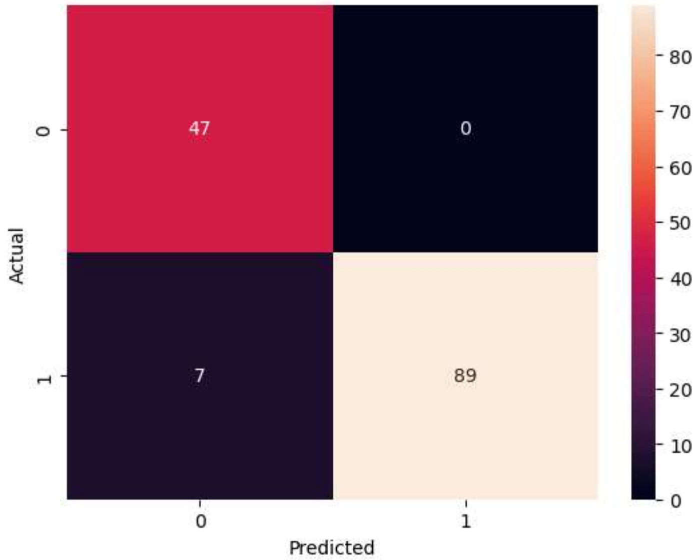
In [38]:
```
y_pred=rand_clf.predict(x_test)
y_pred
```

Out[38]:
```
array([0., 0., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 0., 0., 1., 0.,
       1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0., 1., 0., 0., 1.,
       1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 0., 0.,
       0., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 1., 1., 0.,
       1., 1., 0., 0., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0.,
       1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1., 0., 1., 1.,
       1., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1.,
       0., 0., 1., 1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 1., 0., 1., 0.,
       0., 1., 1., 1., 1., 1., 1.])
```

In [39]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[47  0]
 [ 7 89]]
```

In [40]:
```
cm=confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




In [41]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.87      1.00      0.93        47
         1.0       1.00      0.93      0.96        96

    accuracy                           0.95       143
   macro avg       0.94      0.96      0.95       143
weighted avg       0.96      0.95      0.95       143
```

In [42]:
```
# accuracy score
ran_clf_acc = accuracy_score(y_test, y_pred)
print(ran_clf_acc)
```

```
0.951048951048951
```

## Support Vector Machine

In [43]:
```
from sklearn.svm import SVC
svc_clf=SVC()
svc_clf.fit(x_train,y_train)
```

Out[43]:
```
▾ SVC
SVC()
```

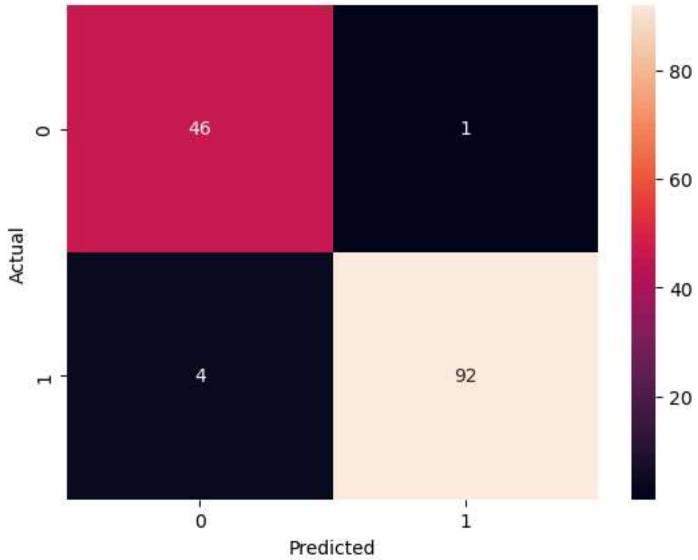
In [44]:
```
y_pred=svc_clf.predict(x_test)
y_pred
```

Out[44]:
```
array([1., 0., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 0., 1., 1., 0.,
       1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0., 1., 0., 0., 1.,
       1., 1., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 0.,
       0., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 0., 1., 1., 0.,
       1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0.,
       1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1., 0., 1., 0.,
       1., 1., 1., 0., 0., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
       0., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0.,
       0., 1., 1., 1., 1., 1., 1.])
```

In [45]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[46  1]
 [ 4 92]]
```

In [46]:
```
cm=confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




In [47]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.92      0.98      0.95        47
         1.0       0.99      0.96      0.97        96

    accuracy                           0.97       143
   macro avg       0.95      0.97      0.96       143
weighted avg       0.97      0.97      0.97       143
```

In [48]:
```
# accuracy score
svc_clf_acc = accuracy_score(y_test, y_pred)
print(svc_clf_acc)
```

```
0.965034965034965
```

## K-Nearest Neighbor (KNN)

In [49]:
```
from sklearn.neighbors import KNeighborsClassifier
knn_clf= KNeighborsClassifier()
knn_clf.fit(x_train,y_train)
```

Out[49]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

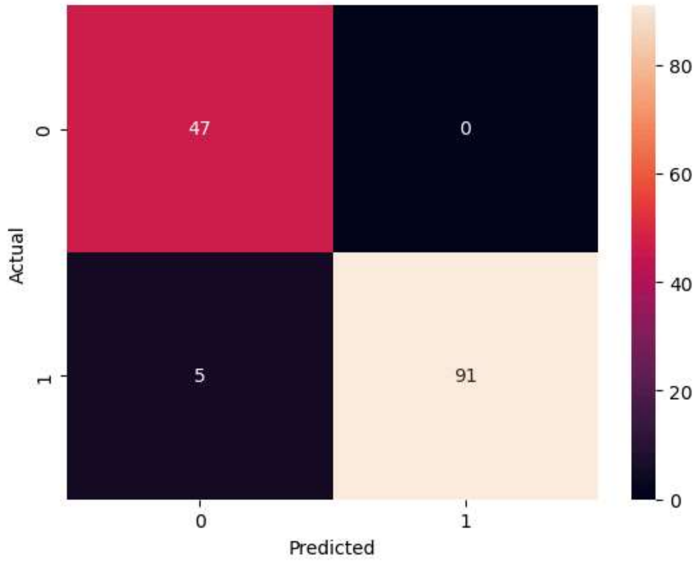
In [50]:
```
y_pred=knn_clf.predict(x_test)
y_pred
```

Out[50]:
```
array([1., 0., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 0., 1., 1., 0.,
       1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0., 1., 0., 0., 1.,
       1., 1., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 0.,
       0., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 0., 1., 1., 0.,
       1., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 1., 1., 1., 1., 0.,
       1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1., 0., 1., 1.,
       1., 1., 1., 0., 0., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
       0., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0.,
       0., 1., 1., 0., 1., 1., 1.])
```

In [51]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[47  0]
 [ 5 91]]
```

In [52]:
```
cm=confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




In [53]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.90      1.00      0.95        47
         1.0       1.00      0.95      0.97        96

    accuracy                           0.97       143
   macro avg       0.95      0.97      0.96       143
weighted avg       0.97      0.97      0.97       143
```

In [54]:
```
# accuracy score
knn_clf_acc = accuracy_score(y_test, y_pred)
print(knn_clf_acc)
```

```
0.965034965034965
```

## Model Comparison

In [55]:
```
':['Lgdtic Reggression','Decision Tree Classfier','Random Forest Classifier','Support Vector Machine','k-Nearest Neighbors'],
```

In [56]:
```
df_model=pd.DataFrame(dict)
df_model.sort_values(by='Score',ascending=False)
```

Out[56]:

| | model | Score |
|---|---|---|
| 0 | Lgdtic Reggression | 0.972028 |
| 3 | Support Vector Machine | 0.965035 |
| 4 | k-Nearest Neighbors | 0.965035 |
| 2 | Random Forest Classifier | 0.951049 |
| 1 | Decision Tree Classfier | 0.937063 |

In [ ]: