```
import pandas as pd
import numpy as np


from sklearn.datasets import load_breast_cancer


breastCancer = load_breast_cancer()


breastCancer.target.size
```

```
569
```

```
breastCancer.data.size
```

```
17070
```

```
breastCancer.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')
```

```
breastCancer.target_names
```

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

```
print(breastCancer.DESCR)
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
```

```
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
    symmetry (standard error):            0.008  0.079
    fractal dimension (standard error):   0.001  0.03
```

```
breastCancer.data.shape
```

```
(569, 30)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(breastCancer.data, breastCancer.target, r
```

```
X_train.shape
```

```
(426, 30)
```

```
X_test.shape
```

```
(143, 30)
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
```

```
knn.fit(X=X_train, y=y_train)
```

```
KNeighborsClassifier()
```

```
predicted = knn.predict(X=X_test)
expected = y_test

wrong = [(p, e) for (p, e) in zip(predicted, expected) if p != e]
print(wrong)
print(len(wrong))
print(len(X_test))
```

```
[(1, 0), (0, 1), (1, 0), (1, 0), (1, 0), (1, 0), (1, 0), (1, 0)]
8
143
```

```
#Estimator Method Score
print(f'{knn.score(X_test, y_test):.2%}')
```

```
94.41%
```

```
from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(y_true=expected, y_pred=predicted)
print(confusion)
```

```
[[43  7]
 [ 1 92]]
array([[43,  7],
       [ 1, 92]])
```

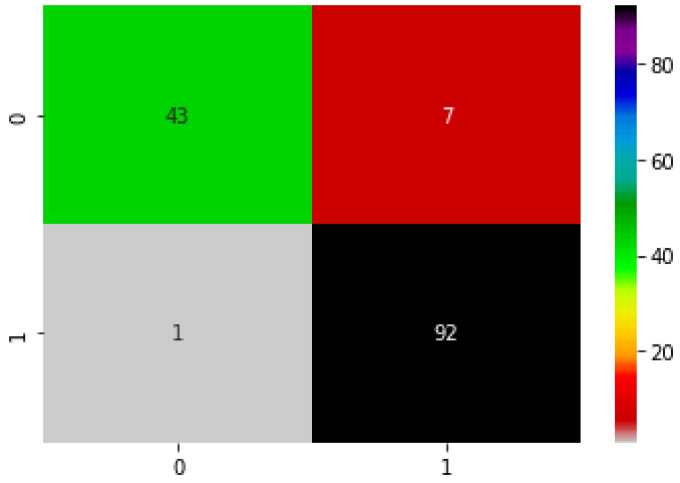
```
from sklearn.metrics import classification_report
names = [str(breastCancer) for breastCancer in breastCancer.target_names]
print(classification_report(expected, predicted, target_names=names))
```

```
                  precision    recall  f1-score   support

       malignant       0.98      0.86      0.91        50
          benign       0.93      0.99      0.96        93
```

```
          accuracy                           0.94       143
         macro avg       0.95      0.92      0.94       143
      weighted avg       0.95      0.94      0.94       143
```

```
import seaborn as sns
confusion_df = pd.DataFrame(confusion)
axes = sns.heatmap(confusion_df, annot=True, cmap='nipy_spectral_r')
```




```
from sklearn.model_selection import KFold
kfold = KFold(n_splits=10, random_state=11, shuffle=True)

from sklearn.model_selection import cross_val_score
scores = cross_val_score(estimator=knn, X=breastCancer.data,
    y=breastCancer.target, cv=kfold)


print(scores)
print(f'Mean accuracy: {scores.mean():.2%}')
print(f'Accuracy standard deviation: {scores.std():.2%}')
```

```
[0.92982456 0.94736842 0.89473684 0.92982456 0.92982456 0.94736842
 0.89473684 0.94736842 0.94736842 0.91071429]
Mean accuracy: 92.79%
Accuracy standard deviation: 2.01%
```

```
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
estimators = {'GaussianNB': GaussianNB(),
    'KNeighborsClassifier': KNeighborsClassifier(),
    'LogisticRegression': LogisticRegression(solver='lbfgs', multi_class='ovr', max_iter=1000
    'SVC': SVC(gamma='scale')}


for estimator_name, estimator_object in estimators.items():
```

```
kfold = KFold(n_splits=10, random_state=11, shuffle=True)
scores = cross_val_score(estimator=estimator_object, X=breastCancer.data, y=breastCancer.ta
print(f'{estimator_name:>20}: ' + f'mean accuracy={scores.mean():.2%}; ' + f'standard devia
```

```
           GaussianNB: mean accuracy=93.85%; standard deviation=2.75%
KNeighborsClassifier: mean accuracy=92.79%; standard deviation=2.01%
  LogisticRegression: mean accuracy=95.08%; standard deviation=3.02%
                 SVC: mean accuracy=91.92%; standard deviation=3.52%
```

Colab paid products - Cancel contracts here

✓ 10s completed at 7:07 PM