```
import pandas as pd
import numpy as np
df=pd.DataFrame(np.random.randn(5,3),index=['a','c','e','f','h'],columns=['one','two','three'])
df=df.reindex(['a','b','c','d','e','f','g','h'])
print(df['one'].isnull())
```

```
a    False
b     True
c    False
d     True
e    False
f    False
g     True
h    False
Name: one, dtype: bool
```

Missing Values

```
df=pd.DataFrame(np.random.randn(5,3),index=['a','c','e','f','h'],columns=['one','two','three'])
print(df)
df=df.reindex(['a','b','c','d','e','f','g','h'])
print(df)
```

```
        one       two     three
a -0.039497  1.045655 -0.138289
c -0.369325 -1.385964 -0.580685
e  0.158656  1.565347  0.087451
f  1.092764  1.218509 -0.550862
h  0.008518  1.282266  0.442202
        one       two     three
a -0.039497  1.045655 -0.138289
b       NaN       NaN       NaN
c -0.369325 -1.385964 -0.580685
d       NaN       NaN       NaN
e  0.158656  1.565347  0.087451
f  1.092764  1.218509 -0.550862
g       NaN       NaN       NaN
h  0.008518  1.282266  0.442202
```

```
df = pd.DataFrame(np.random.randn(3,3),index=['a','c','e'],columns=['one','two','three'])
df = df.reindex(['a','b','c'])
print(df)
print("NaN replaced with '0':")
print(df.fillna(0))
```

```
        one       two     three
a -0.662758  0.638865  1.460741
b       NaN       NaN       NaN
c -0.946790  0.584346  1.968561
NaN replaced with '0':
        one       two     three
a -0.662758  0.638865  1.460741
b  0.000000  0.000000  0.000000
c -0.946790  0.584346  1.968561
```

```
df = pd.DataFrame(np.random.randn(5,3),index=['a','c','e','f','h'],columns=['one','two','three'])
df = df.reindex(['a','b','c','d','e','f','g','h'])
print(df)
print('-------------------------------')
print(df.fillna(method='pad'))
```

```
        one       two     three
a  0.950124 -1.565339 -1.909730
b       NaN       NaN       NaN
c  0.051079 -0.767547  1.370560
d       NaN       NaN       NaN
e -0.687186 -0.579887  1.457962
f  0.463522 -1.184759 -0.139350
g       NaN       NaN       NaN
h -0.318872  0.053867 -2.192390
-------------------------------
        one       two     three
a  0.950124 -1.565339 -1.909730
b  0.950124 -1.565339 -1.909730
c  0.051079 -0.767547  1.370560
d  0.051079 -0.767547  1.370560
e -0.687186 -0.579887  1.457962
f  0.463522 -1.184759 -0.139350
g  0.463522 -1.184759 -0.139350
h -0.318872  0.053867 -2.192390
```

```
df = pd.DataFrame(np.random.randn(5,3),index=['a','c','e','f','h'],columns=['one','two','three'])
df = df.reindex(['a','b','c','d','e','f','g','h'])
print(df.fillna(method='bfill'))
```

```
        one       two     three
a -1.120243  1.486683 -0.093951
b -0.125297  0.466293  0.824516
c -0.125297  0.466293  0.824516
d  1.121361  0.520289 -1.270749
e  1.121361  0.520289 -1.270749
f  0.027017 -0.031351 -0.883342
g -0.720911  0.815735  1.013695
h -0.720911  0.815735  1.013695
```

```
df = pd.DataFrame(np.random.randn(5,3),index=['a','c','e','f','h'],columns=['one','two','three'])
df = df.reindex(['a','b','c','d','e','f','g','h'])
print(df)
print(df.dropna())
```

```
        one       two     three
a  0.641054  0.489022 -0.361579
b       NaN       NaN       NaN
c -0.643466  0.929190 -0.408637
d       NaN       NaN       NaN
e -1.877724  0.369620  2.092137
f  0.269918  2.400600 -1.373824
g       NaN       NaN       NaN
h  0.962634  1.555593  1.215637
        one       two     three
a  0.641054  0.489022 -0.361579
c -0.643466  0.929190 -0.408637
e -1.877724  0.369620  2.092137
f  0.269918  2.400600 -1.373824
h  0.962634  1.555593  1.215637
```

```
df1 = pd.DataFrame({'one':[1000,23,24,25,26,27], 'two' : [2022,32,25,26,20,22,]})
print(df1)
print(df1.replace({1000:22,2022:22}))
```

```
    one   two
0  1000  2022
1    23    32
2    24    25
3    25    26
4    26    20
5    27    22
   one  two
0   22   22
1   23   32
2   24   25
3   25   26
4   26   20
5   27   22
```

```
import pandas as pd
df=pd.read_csv('/content/titanic.csv')
df.info()
```

```
df.describe()
```

```
cols=['Name','Ticket','Cabin']
df=df.drop(cols,axis=1)
df.info()
```

```
df=df.dropna()
df.info()
```

```
dummies=[]
cols=['Pclass','Sex','Embarked']
for col in cols:
  dummies.append(pd.get_dummies(df[cols]))
```

```
titanic_dummies=pd.concat(dummies,axis=1)
```

```
df=pd.concat((df,titanic_dummies),axis=1)
print(df)
```

```
df = pd.concat((df,titanic_dummies),axis=1)
print(df)
```

Mix Max Scaler and Standardization

```
from sklearn.preprocessing import MinMaxScaler
data = [[-1,2],[-0.5,6],[0,10],[1,18]]
scaler = MinMaxScaler()
print(scaler.fit(data))
print('-------------')
MinMaxScaler()
print(scaler.data_max_)
print('--------------------')
print(scaler.transform(data))
```

```
from numpy import asarray
from sklearn.preprocessing import StandardScaler
#define data
data = asarray([[100,0.001],
                [8,0.05],
                [50,0.005],
                [88,0.07],
                [4,0.1]])
print(data)
```

```
scaler = StandardScaler()
scaler.fit(data)
data = scaler.transform(data)
```

```
data
```

```
import numpy as np
data = [1,2,2,2,3,1,1,15,2,2,2,3,1,1,2]
mean = np.mean(data)
std = np.std(data)
print("Mean of the dataset is : ",mean)
print("Std deviation is: ",std)
threshold = 3
outlier = []
for i in data:
  z = (i-mean)/std
  if z > threshold:
    outlier.append(i)
print("outlier in dataset is:",outlier)
```

Interquartile range to detect outliers in dataset

- Q1 = 25%
- Q2 = 50%
- Q3 = 75%

---

if a dataset has 2n/2n+1 data points then,

- Q1 = median of the dataset
- Q2 = meadian of n smallest data points
- Q3 = median of n highest data points

IQR is the range between the first and the third quantiles namely Q1 and Q3

IQR = Q3 - Q1

```
#Step1 : import the necessary libraries
import numpy as np
import seaborn as sns
```

```
#Step2 : Take the data and sort it in ascending order
data = [6,2,3,4,5,1,50]
sort_data = np.sort(data)
print(sort_data)
```

```
#step3 : Calculate Q1,Q2,Q3 and IQR
Q1 = np.percentile(data,25,interpolation = 'midpoint')
Q2 = np.percentile(data,50,interpolation = 'midpoint')
Q3 = np.percentile(data,75,interpolation = 'midpoint')
print(" Q1 25 percentile of the given data is : ",Q1)
print(" Q2 50 percentile of the given data is : ",Q2)
print(" Q3 75 percentile of the given data is : ",Q3)

IQR = Q3 - Q1
print("Interquartile range is : ",IQR)



#Step 4:
low_lim=Q1-1.5*IQR
up_lim=Q3+1.5*IQR
print('Low limit is ',low_lim)
print('Up limit is ',up_lim)


#Step 5: DAta points greater than the upper limit or less thean the lower limit are

outlier = []
for x in data:
  if ((x  > up_lim) or (x < low_lim)):
    outlier.append(x)
    print("Outlier in the dataset is ",outlier)



#Step 6: Plot the box plot to highlight outliers
sns.boxplot(data)


import pandas as pd
def load_data():
  df_all = pd.read_csv("/content/2,1 dataset titanic (1).csv")
  #Take a subset
  return df_all.loc[:300,['Survived','Pclass','Sex','Cabin','Embarked']]
df = load_data()
df
```

Finding duplicate rows

```
df.Cabin.duplicated()
```

Breast Cancer Dataset

Principal component analysis

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
df = pd.read_csv("/content/2.2 dataset breast cancer.csv")
```

```
df.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  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

```
breast = load_breast_cancer()
breast_data = breast.data
print(breast_data)
print(breast_data.shape)
```

```
[[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]]
(569, 30)
```

```
breast_labels = breast.target
print(breast_labels)
print(breast_labels.shape)
```

```
[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 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 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]
(569,)
```

```
labels = np.reshape(breast_labels,(569,1))
final_breast_data = np.concatenate([breast_data,labels],axis=1)
print(final_breast_data.shape)
```

```
(569, 31)
```

```
breast_dataset = pd.DataFrame(final_breast_data)
print(breast_dataset.head())
```

```
       0      1       2       3        4        5       6        7       8  \
0  17.99  10.38  122.80  1001.0  0.11840  0.27760  0.3001  0.14710  0.2419
1  20.57  17.77  132.90  1326.0  0.08474  0.07864  0.0869  0.07017  0.1812
2  19.69  21.25  130.00  1203.0  0.10960  0.15990  0.1974  0.12790  0.2069
3  11.42  20.38   77.58   386.1  0.14250  0.28390  0.2414  0.10520  0.2597
4  20.29  14.34  135.10  1297.0  0.10030  0.13280  0.1980  0.10430  0.1809
```

```
         9   ...     21      22      23      24      25      26      27  \
0  0.07871  ...  17.33  184.60  2019.0  0.1622  0.6656  0.7119  0.2654
1  0.05667  ...  23.41  158.80  1956.0  0.1238  0.1866  0.2416  0.1860
2  0.05999  ...  25.53  152.50  1709.0  0.1444  0.4245  0.4504  0.2430
3  0.09744  ...  26.50   98.87   567.7  0.2098  0.8663  0.6869  0.2575
4  0.05883  ...  16.67  152.20  1575.0  0.1374  0.2050  0.4000  0.1625

       28       29   30
0  0.4601  0.11890  0.0
1  0.2750  0.08902  0.0
2  0.3613  0.08758  0.0
3  0.6638  0.17300  0.0
4  0.2364  0.07678  0.0

[5 rows x 31 columns]
```

```
features = breast.feature_names
print(features)
```

```
['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']
```

```
features_labels = np.append(features,'label')
```

```
breast_dataset.columns = features_labels
breast_dataset.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sy
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

5 rows × 31 columns

```
breast_dataset['label'].replace(0, 'Benign' , inplace=True)
breast_dataset['label'].replace(1, 'Maligant',inplace=True)
breast_dataset.tail()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000

5 rows × 31 columns

```
from sklearn.preprocessing import StandardScaler
x = breast_dataset.loc[:,features].values
x = StandardScaler().fit_transform(x) #normalising the features
print(x.shape)
```

```
(569, 30)
```

```
np.mean(x),np.std(x)
```

```
(-6.118909323768877e-16, 1.0)
```

```
feat_cols = ['feature'+str(i) for i in range(x.shape[1])]
```

```
normalised_breast = pd.DataFrame(x,columns=feat_cols)
print(normalised_breast)
```

```
     feature0  feature1  feature2  feature3  feature4  feature5  feature6  \
0    1.097064 -2.073335  1.269934  0.984375  1.568466  3.283515  2.652874
1    1.829821 -0.353632  1.685955  1.908708 -0.826962 -0.487072 -0.023846
2    1.579888  0.456187  1.566503  1.558884  0.942210  1.052926  1.363478
3   -0.768909  0.253732 -0.592687 -0.764464  3.283553  3.402909  1.915897
4    1.750297 -1.151816  1.776573  1.826229  0.280372  0.539340  1.371011
..        ...       ...       ...       ...       ...       ...       ...
564  2.110995  0.721473  2.060786  2.343856  1.041842  0.219060  1.947285
565  1.704854  2.085134  1.615931  1.723842  0.102458 -0.017833  0.693043
566  0.702284  2.045574  0.672676  0.577953 -0.840484 -0.038680  0.046588
567  1.838341  2.336457  1.982524  1.735218  1.525767  3.272144  3.296944
568 -1.808401  1.221792 -1.814389 -1.347789 -3.112085 -1.150752 -1.114873

     feature7  feature8  feature9  ...  feature20  feature21  feature22  \
0    2.532475  2.217515  2.255747  ...   1.886690  -1.359293   2.303601
1    0.548144  0.001392 -0.868652  ...   1.805927  -0.369203   1.535126
2    2.037231  0.939685 -0.398008  ...   1.511870  -0.023974   1.347475
3    1.451707  2.867383  4.910919  ...  -0.281464   0.133984  -0.249939
4    1.428493 -0.009560 -0.562450  ...   1.298575  -1.466770   1.338539
..        ...       ...       ...  ...        ...        ...        ...
564  2.320965 -0.312589 -0.931027  ...   1.901185   0.117700   1.752563
565  1.263669 -0.217664 -1.058611  ...   1.536720   2.047399   1.421940
566  0.105777 -0.809117 -0.895587  ...   0.561361   1.374854   0.579001
567  2.658866  2.137194  1.043695  ...   1.961239   2.237926   2.303601
568 -1.261820 -0.820070 -0.561032  ...  -1.410893   0.764190  -1.432735

     feature23  feature24  feature25  feature26  feature27  feature28  \
0     2.001237   1.307686   2.616665   2.109526   2.296076   2.750622
1     1.890489  -0.375612  -0.430444  -0.146749   1.087084  -0.243890
2     1.456285   0.527407   1.082932   0.854974   1.955000   1.152255
3    -0.550021   3.394275   3.893397   1.989588   2.175786   6.046041
4     1.220724   0.220556  -0.313395   0.613179   0.729259  -0.868353
..         ...        ...        ...        ...        ...        ...
564   2.015301   0.378365  -0.273318   0.664512   1.629151  -1.360158
565   1.494959  -0.691230  -0.394820   0.236573   0.733827  -0.531855
566   0.427906  -0.809587   0.350735   0.326767   0.414069  -1.104549
567   1.653171   1.430427   3.904848   3.197605   2.289985   1.919083
568  -1.075813  -1.859019  -1.207552  -1.305831  -1.745063  -0.048138

     feature29
0     1.937015
1     0.281190
2     0.201391
3     4.935010
4    -0.397100
..         ...
564  -0.709091
565  -0.973978
566  -0.318409
567   2.219635
568  -0.751207

[569 rows x 30 columns]
```

```
normalised_breast.tail()
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6	feature7
564	2.110995	0.721473	2.060786	2.343856	1.041842	0.219060	1.947285	2.320965
565	1.704854	2.085134	1.615931	1.723842	0.102458	-0.017833	0.693043	1.263669
566	0.702284	2.045574	0.672676	0.577953	-0.840484	-0.038680	0.046588	0.105777
567	1.838341	2.336457	1.982524	1.735218	1.525767	3.272144	3.296944	2.658866
568	-1.808401	1.221792	-1.814389	-1.347789	-3.112085	-1.150752	-1.114873	-1.261820

5 rows × 30 columns

```
from sklearn.decomposition import PCA
pca_breast = PCA(n_components=2)
principalComponents_breast = pca_breast.fit_transform(x)
principal_breast_Df = pd.DataFrame(data = principalComponents_breast
                                   , columns = ['principal component 1','principal component 2'])
principal_breast_Df.tail()
```

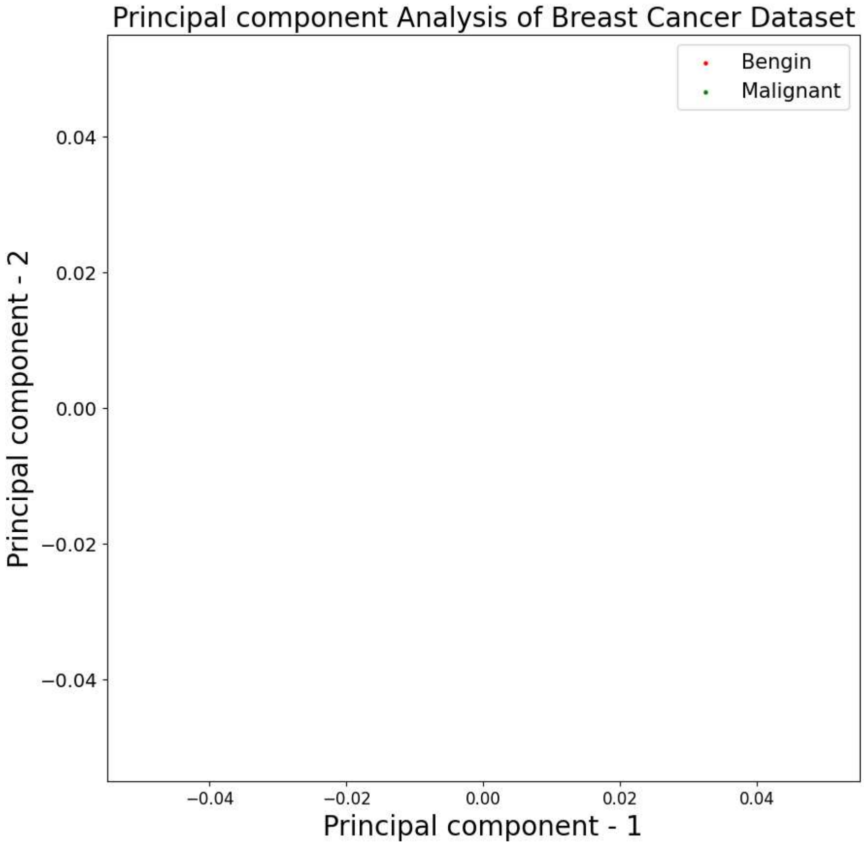
	principal component 1	principal component 2
564	6.439315	-3.576817
565	3.793382	-3.584048
566	1.256179	-1.902297
567	10.374794	1.672010
568	-5.475243	-0.670637

```
print(f"Explained variation per principal component : {pca_breast.explained_variance_ratio_}")
```

```
Explained variation per principal component : [0.44272026 0.18971182]
```

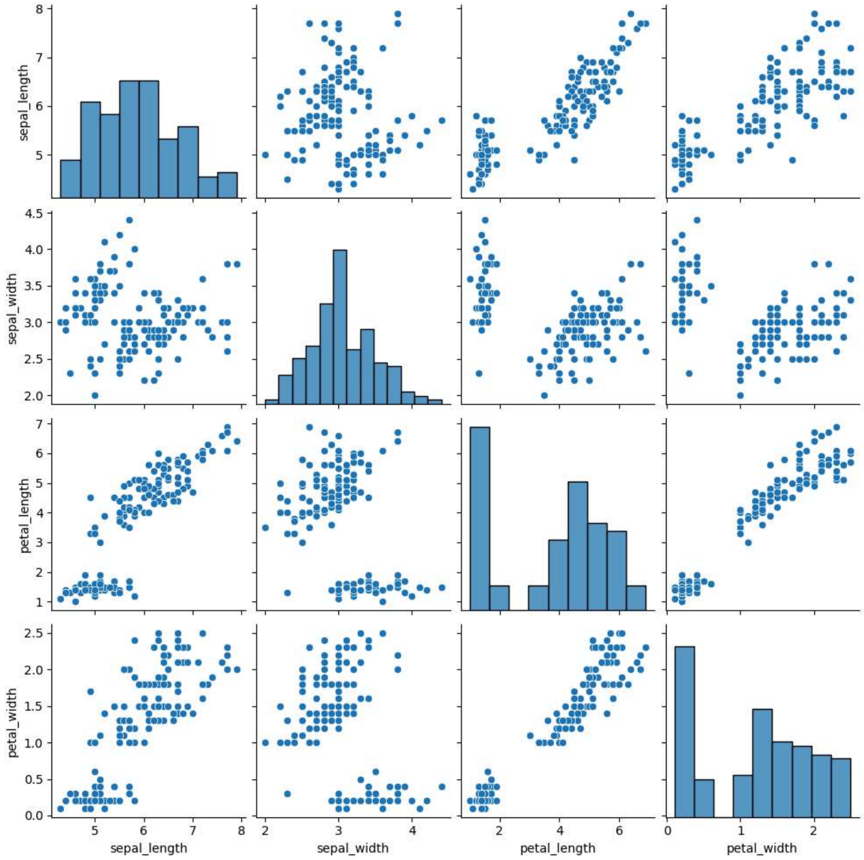
```
import matplotlib.pyplot as plt
plt.figure()
plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel("Principal component - 1",fontsize=20)
plt.ylabel("Principal component - 2",fontsize=20)
plt.title("Principal component Analysis of Breast Cancer Dataset",fontsize=20)
targets = ['Bengin','Malignant']
colors =['r','g']
for target, color in zip(targets,colors):
  indicesToKeep = breast_dataset['label'] == target
  plt.scatter(principal_breast_Df.loc[indicesToKeep, 'principal component 1']
              ,principal_breast_Df.loc[indicesToKeep , 'principal component 2'], c = color , s = 5)
plt.legend(targets,prop={'size': 15})
```

```
<matplotlib.legend.Legend at 0x79406ddbe410>
<Figure size 640x480 with 0 Axes>
```



CORRELATION REGRESSION

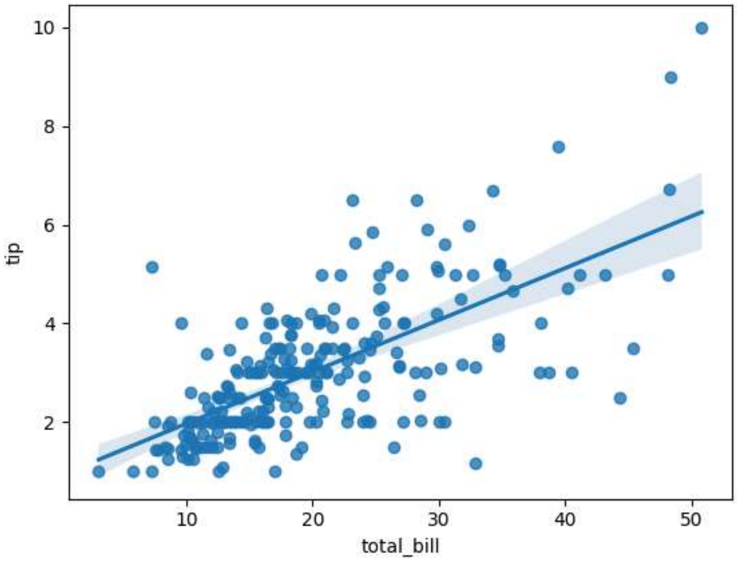
```
import matplotlib.pyplot as plt
import seaborn as sns
df = sns.load_dataset('iris')
#Without regresssion
sns.pairplot(df,kind="scatter")
plt.show()
```




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
from matplotlib import pyplot as pyplot
df = sns.load_dataset('tips')
sns.regplot(x = "total_bill" , y = "tip" , data =df)
plt.show()
```




```
from scipy import stats
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,81,88,111,86,103,87,94,78,77,85,86]
```

```
slope, intercept, r, p, std_err = stats.linregress(x,y)
```

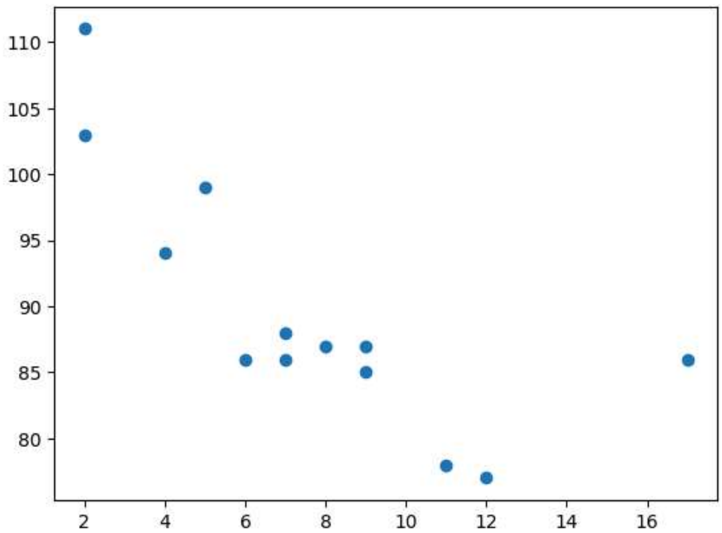
Start coding or generate with AI.

```
mymodel=list(map(myfunc,x))
```

draw the original scatter plot

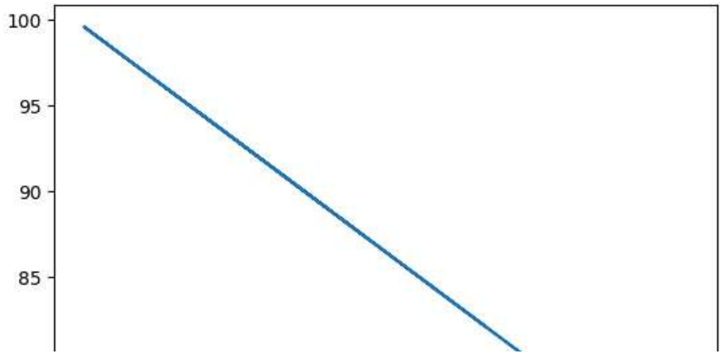
```
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x7b817bdadd20>
```




```
plt.plot(x,mymodel)
```

```
[<matplotlib.lines.Line2D at 0x7b817c0fdba0>]
```




```
def estimator_coeff(p,q):
  #here we will estimate the total number of points or observation
  n1=np.size(p)
  #now we will calculate the mean of a and b vector
  m_p=np.mean(p)
  m_q=np.mean(q)
  #here we will calculate the cross deviation and deviation
  SS_pq=np.sum(q*p)-n1*m_q*m_p
  SS_pp=np.sum(p*p)-n1*m_q*m_p
    #here we will calculate the regression coefficient
  b_1=SS_pq/SS_pp
  b_0=m_q-b_1*m_p
  return(b_0,b_1)


def plot_regression_line(p,q,b):
  #now we will plot actual points or observation as scatter plot
  plt.scatter(p,q,color="m",marker="o",s=30)
  #Here we will calculate the predicted response vector
  q_pred=b[0]+b[1]*p
  #here we will plot regression line
  plt.plot(p,q_pred,color="g")
  plt.xlabel('p')
  plt.ylable=('q')
  plt.show()


def main():
  #entering the observation points and data
  p=np.array([10,11,12,13,14,15,16,17,18,19])
  p=np.array([11,13,12,15,17,18,18,19,20,22])
  #now we will plot the regression line
  plot_regression_line(p,q,b)
if __name__=="__main__":
  main()
```

Start coding or generate with AI.