Data Cleaning

```
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'])
print(df)
df=df.reindex(['a','b','c','d','e','f','g','h'])
print(df)
```

```
        one       two     three
a -0.478899  0.147160  0.839220
c  2.368158  2.636489  0.781151
e  0.225512 -1.010787  0.378416
f  0.349830 -0.991095  1.221360
h  0.786228 -0.032687  1.121054
        one       two     three
a -0.478899  0.147160  0.839220
b       NaN       NaN       NaN
c  2.368158  2.636489  0.781151
d       NaN       NaN       NaN
e  0.225512 -1.010787  0.378416
f  0.349830 -0.991095  1.221360
g       NaN       NaN       NaN
h  0.786228 -0.032687  1.121054
```

Missing values

```
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
```

Replacing Missing Values

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

```
        one       two     three
a  0.505707  1.633075 -1.302445
c -0.504708  0.166678 -2.440838
e  0.362044  0.498025 -0.027647
        one       two     three
a  0.505707  1.633075 -1.302445
b       NaN       NaN       NaN
c -0.504708  0.166678 -2.440838
NaN replaced with '0' :
        one       two     three
a  0.505707  1.633075 -1.302445
b  0.000000  0.000000  0.000000
c -0.504708  0.166678 -2.440838
```

Fill NA Forward NA Backword

```
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 -0.074901 -1.548806 -0.060162
b       NaN       NaN       NaN
c  1.427199  0.802253 -0.388188
d       NaN       NaN       NaN
e -0.912891 -0.697399 -0.283572
f  0.221375 -1.142681 -0.710734
g       NaN       NaN       NaN
h  0.103521 -0.364679 -0.526936
--------------------------------
        one       two     three
a -0.074901 -1.548806 -0.060162
b -0.074901 -1.548806 -0.060162
c  1.427199  0.802253 -0.388188
d  1.427199  0.802253 -0.388188
e -0.912891 -0.697399 -0.283572
f  0.221375 -1.142681 -0.710734
g  0.221375 -1.142681 -0.710734
h  0.103521 -0.364679 -0.526936
```

```
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='bfill'))
```

```
        one       two     three
a -0.429247  0.692197 -1.940063
b       NaN       NaN       NaN
c  0.528100 -0.368937 -0.951022
d       NaN       NaN       NaN
e  0.995892 -0.448006 -0.645197
f  0.776726 -0.993705 -2.070430
g       NaN       NaN       NaN
h -0.419377 -0.284991 -0.474422
--------------------------------
        one       two     three
a -0.429247  0.692197 -1.940063
b  0.528100 -0.368937 -0.951022
c  0.528100 -0.368937 -0.951022
d  0.995892 -0.448006 -0.645197
e  0.995892 -0.448006 -0.645197
f  0.776726 -0.993705 -2.070430
g -0.419377 -0.284991 -0.474422
h -0.419377 -0.284991 -0.474422
```

Drop the missing values

```
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.dropna())
```

```
        one       two     three
a -0.825442 -0.120668  0.182013
b       NaN       NaN       NaN
c -1.078704  0.131047  1.884462
d       NaN       NaN       NaN
e -0.699292  1.237147  1.868485
f -1.200506 -1.325262 -0.532936
g       NaN       NaN       NaN
h -0.127553  0.490895 -0.732149
--------------------------------
        one       two     three
a -0.825442 -0.120668  0.182013
c -1.078704  0.131047  1.884462
e -0.699292  1.237147  1.868485
f -1.200506 -1.325262 -0.532936
h -0.127553  0.490895 -0.732149
```

```
df=pd.read_csv('titanic.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
Start coding or generate with AI.
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Sex          891 non-null    object
 4   Age          714 non-null    float64
 5   SibSp        891 non-null    int64
 6   Parch        891 non-null    int64
 7   Fare         891 non-null    float64
 8   Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(2)
memory usage: 62.8+ KB
```

drop the row having no value

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  712 non-null    int64
 1   Survived     712 non-null    int64
 2   Pclass       712 non-null    int64
 3   Sex          712 non-null    object
 4   Age          712 non-null    float64
 5   SibSp        712 non-null    int64
 6   Parch        712 non-null    int64
 7   Fare         712 non-null    float64
 8   Embarked     712 non-null    object
dtypes: float64(2), int64(5), object(2)
memory usage: 55.6+ KB
```

creating dummy variable

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  712 non-null    int64  
 1   Survived     712 non-null    int64  
 2   Pclass       712 non-null    int64  
 3   Sex          712 non-null    object 
 4   Age          712 non-null    float64
 5   SibSp        712 non-null    int64  
 6   Parch        712 non-null    int64  
 7   Fare         712 non-null    float64
 8   Embarked     712 non-null    object 
dtypes: float64(2), int64(5), object(2)
memory usage: 55.6+ KB
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  712 non-null    int64  
 1   Survived     712 non-null    int64  
 2   Pclass       712 non-null    int64  
 3   Sex          712 non-null    object 
 4   Age          712 non-null    float64
 5   SibSp        712 non-null    int64  
 6   Parch        712 non-null    int64  
 7   Fare         712 non-null    float64
 8   Embarked     712 non-null    object 
dtypes: float64(2), int64(5), object(2)
memory usage: 55.6+ KB
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 27 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  712 non-null    int64  
 1   Survived     712 non-null    int64  
 2   Pclass       712 non-null    int64  
 3   Sex          712 non-null    object 
 4   Age          712 non-null    float64
 5   SibSp        712 non-null    int64  
 6   Parch        712 non-null    int64  
 7   Fare         712 non-null    float64
 8   Embarked     712 non-null    object 
 9   Pclass       712 non-null    int64  
 10  Sex_female   712 non-null    uint8  
 11  Sex_male     712 non-null    uint8  
 12  Embarked_C   712 non-null    uint8  
 13  Embarked_Q   712 non-null    uint8  
 14  Embarked_S   712 non-null    uint8  
 15  Pclass       712 non-null    int64  
 16  Sex_female   712 non-null    uint8  
 17  Sex_male     712 non-null    uint8  
 18  Embarked_C   712 non-null    uint8  
 19  Embarked_Q   712 non-null    uint8  
 20  Embarked_S   712 non-null    uint8  
 21  Pclass       712 non-null    int64  
 22  Sex_female   712 non-null    uint8  
 23  Sex_male     712 non-null    uint8  
 24  Embarked_C   712 non-null    uint8  
 25  Embarked_Q   712 non-null    uint8  
 26  Embarked_S   712 non-null    uint8  
dtypes: float64(2), int64(8), object(2), uint8(15)
memory usage: 82.7+ KB
```

```
df=df.drop(['Pclass','Sex','Embarked'],axis=1)
print(df)
```

```
     PassengerId  Survived   Age  SibSp  Parch     Fare  Sex_female  Sex_male  \
0              1         0  22.0      1      0   7.2500           0         1   
1              2         1  38.0      1      0  71.2833           1         0   
```

```
2             3         1  26.0      0      0   7.9250           1         0
3             4         1  35.0      1      0  53.1000           1         0
4             5         0  35.0      0      0   8.0500           0         1
..          ...       ...   ...    ...    ...      ...         ...       ...
885         886         0  39.0      0      5  29.1250           1         0
886         887         0  27.0      0      0  13.0000           0         1
887         888         1  19.0      0      0  30.0000           1         0
889         890         1  26.0      0      0  30.0000           0         1
890         891         0  32.0      0      0   7.7500           0         1

     Embarked_C  Embarked_Q  ...  Sex_female  Sex_male  Embarked_C  \
0             0           0  ...           0         1           0
1             1           0  ...           1         0           1
2             0           0  ...           1         0           0
3             0           0  ...           1         0           0
4             0           0  ...           0         1           0
..          ...         ...  ...         ...       ...         ...
885           0           1  ...           1         0           0
886           0           0  ...           0         1           0
887           0           0  ...           1         0           0
889           1           0  ...           0         1           1
890           0           1  ...           0         1           0

     Embarked_Q  Embarked_S  Sex_female  Sex_male  Embarked_C  Embarked_Q  \
0             0           1           0         1           0           0
1             0           0           1         0           1           0
2             0           1           1         0           0           0
3             0           1           1         0           0           0
4             0           1           0         1           0           0
..          ...         ...         ...       ...         ...         ...
885           1           0           1         0           0           1
886           0           1           0         1           0           0
887           0           1           1         0           0           0
889           0           0           0         1           1           0
890           1           0           0         1           0           1

     Embarked_S
0             1
1             0
2             1
3             1
4             1
..          ...
885           0
886           1
887           1
889           0
890           0

[712 rows x 21 columns]
```

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

```
-----------------------------------
MinMaxScaler()
[ 1. 18.]
-----------------------------------
[[0.   0.  ]
 [0.25 0.25]
 [0.5  0.5 ]
 [1.   1.  ]]
```

```
from numpy import asarray
from sklearn.preprocessing import StandardScaler
#define data
data=asarray([[100,0.001],
              [8,0.05],
              [88,0.07],
              [88,0.07],
              [4,0.1]])
print(data)
#define Standard Scaler
scaler=StandardScaler()
#Transform the data
scaled=scaler.fit_transform(data)
print(scaled)
```

```
     [[1.0e+02 1.0e-03]
      [8.0e+00 5.0e-02]
      [8.8e+01 7.0e-02]
      [8.8e+01 7.0e-02]
      [4.0e+00 1.0e-01]]
     [[ 1.00053443 -1.74624133]
      [-1.1704365  -0.2503353 ]
      [ 0.71736431  0.3602386 ]
      [ 0.71736431  0.3602386 ]
      [-1.26482654  1.27609943]]
```

```
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('Meanof 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)
```

```
     Meanof the dataset is 2.6666666666666665
     std. deviation is 3.3598941782277745
     Outlier in dataset is [15]
```

InterQuartile Range

- Q1 25
- Q2 50
- Q3 75

```
#Step 1: Import necessary libraries
import numpy as np
import seaborn as sns
#Step 2: Take the data and sort it in ascending order
data=[6,2,3,4,5,1,50]
sort_data=np.sort(data)
sort_data
```

```
     array([ 1,  2,  3,  4,  5,  6, 50])
```

```
#Step 3: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)
```

```
     Q1 25 percentile of the given data is,  2.5
     Q2 50 percentile of the given data is,  4.0
```

```
Q3 75 percentile of the given data is,  5.5
Interquartile range is  3.0
```

```
#Step 4: Find the lower and upper limits as Q1-1.5 IQR and Q3+1.5 IQR, respectively
low_lim=Q1-1.5*IQR
up_lim=Q3+1.5*IQR
print('Low limit is ',low_lim)
print('Up limit is ',up_lim)
```

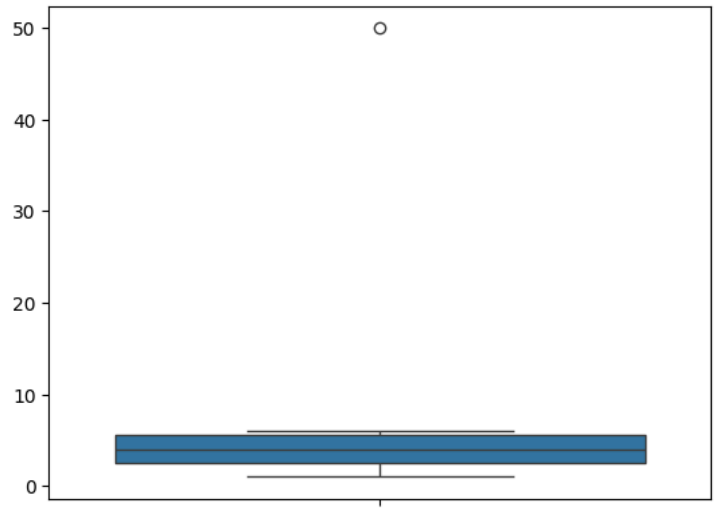
```
Low limit is  -2.0
Up limit is  10.0
```

```
#Step 5:Data points greater than the upper limit or less than lower limit
outlier=[]
for x in data:
  if((x>up_lim)or(x<low_lim)):
    outlier.append(x)
print('Outlier in the datset is ',outlier)
```

```
Outlier in the datset is  [50]
```

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

```
<Axes: >
```




```
def load_data():
  df_all=pd.read_csv('titanic.csv')
  #Take a Subset
  return df_all.loc[:300,['Survived','Pclass','Sex','Cabin','Embarked']]

  #Load subset
df=load_data()
```

```
#For single column
duplicates = df.loc[df.Cabin.duplicated()]
print(duplicates)
```

```
     Survived  Pclass     Sex    Cabin Embarked
2           1       3  female      NaN        S
4           0       3    male      NaN        S
5           0       3    male      NaN        Q
7           0       3    male      NaN        S
8           1       3  female      NaN        S
..        ...     ...     ...      ...      ...
294         0       3    male      NaN        S
295         0       1    male      NaN        C
296         0       3    male      NaN        C
299         1       1  female  B58 B60        C
300         1       3  female      NaN        Q

[245 rows x 5 columns]
```

```
df.duplicated()
```

```
0      False
1      False
2      False
3      False
4       True
       ...
296     True
297    False
298    False
299    False
300     True
Length: 301, dtype: bool
```

```
#To consider certain columns for idenifying duplicates
df.Cabin.duplicated().sum()
```

```
245
```

```
df.duplicated().sum()
```

```
218
```

```
df.loc[df.duplicated(keep='first'),:]
```

	Survived	Pclass	Sex	Cabin	Embarked
4	0	3	male	NaN	S
7	0	3	male	NaN	S
8	1	3	female	NaN	S
12	0	3	male	NaN	S
13	0	3	male	NaN	S
...	...	...	...	...	...
293	0	3	female	NaN	S
294	0	3	male	NaN	S
295	0	1	male	NaN	C
296	0	3	male	NaN	C
300	1	3	female	NaN	Q

218 rows × 5 columns

Principle Component **Analysis**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
```

```
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 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 0 1 0 1 1 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0 1 0 0 1 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 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 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 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 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 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 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 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 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 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 1 0 0 1 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 1 1 0 0 0 0 0 0 1]
(569,)
```

Reshape the dataset by adding label to it Concatenate the dataset with label

```
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]
```

print the features that are there in breastcancer

```
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']
```

Here the label field is missing so add it

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

Embedding the column names to the dataframe

```
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	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0

5 rows × 31 columns

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

	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
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
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
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
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
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

5 rows × 31 columns

PCA

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

```
(569, 30)
```

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

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

Lets convert the normalized features into a tabular format

```
feat_cols=['feature'+str(1) for i in range(x.shape[1])]
normalized_breast=pd.DataFrame(x,columns=feat_cols)
```

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

```
     feature1  feature1  feature1  feature1  feature1  feature1  feature1  \
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
```

```
     feature1  feature1  feature1  ...  feature1  feature1  feature1  \
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

     feature1  feature1  feature1  feature1  feature1  feature1  feature1
0    2.001237  1.307686  2.616665  2.109526  2.296076  2.750622  1.937015
1    1.890489 -0.375612 -0.430444 -0.146749  1.087084 -0.243890  0.281190
2    1.456285  0.527407  1.082932  0.854974  1.955000  1.152255  0.201391
3   -0.550021  3.394275  3.893397  1.989588  2.175786  6.046041  4.935010
4    1.220724  0.220556 -0.313395  0.613179  0.729259 -0.868353 -0.397100
..        ...       ...       ...       ...       ...       ...       ...
564  2.015301  0.378365 -0.273318  0.664512  1.629151 -1.360158 -0.709091
565  1.494959 -0.691230 -0.394820  0.236573  0.733827 -0.531855 -0.973978
566  0.427906 -0.809587  0.350735  0.326767  0.414069 -1.104549 -0.318409
567  1.653171  1.430427  3.904848  3.197605  2.289985  1.919083  2.219635
568 -1.075813 -1.859019 -1.207552 -1.305831 -1.745063 -0.048138 -0.751207

[569 rows x 30 columns]
```

```
normalized_breast.tail()
```

	feature1	feature1	feature1	feature1	feature1	feature1	feature1	feature1	fe
564	2.110995	0.721473	2.060786	2.343856	1.041842	0.219060	1.947285	2.320965	-0.
565	1.704854	2.085134	1.615931	1.723842	0.102458	-0.017833	0.693043	1.263669	-0.
566	0.702284	2.045574	0.672676	0.577953	-0.840484	-0.038680	0.046588	0.105777	-0
567	1.838341	2.336457	1.982524	1.735218	1.525767	3.272144	3.296944	2.658866	2.
568	-1.808401	1.221792	-1.814389	-1.347789	-3.112085	-1.150752	-1.114873	-1.261820	-0.

5 rows × 30 columns

Projecting the thirty-dimensional Breast Cancer data to two dimensional

```
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

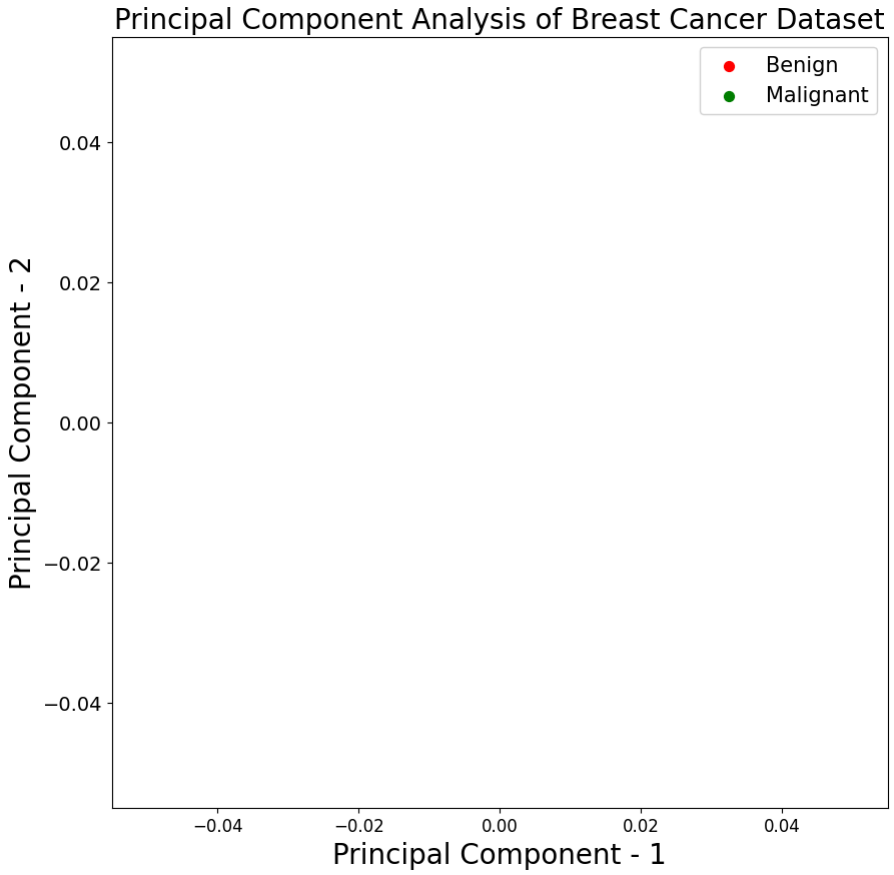
```
print('Explained variation per principal component:{}'.format(pca_breast.explained_variance_ratio_))
```

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

Plot PCA

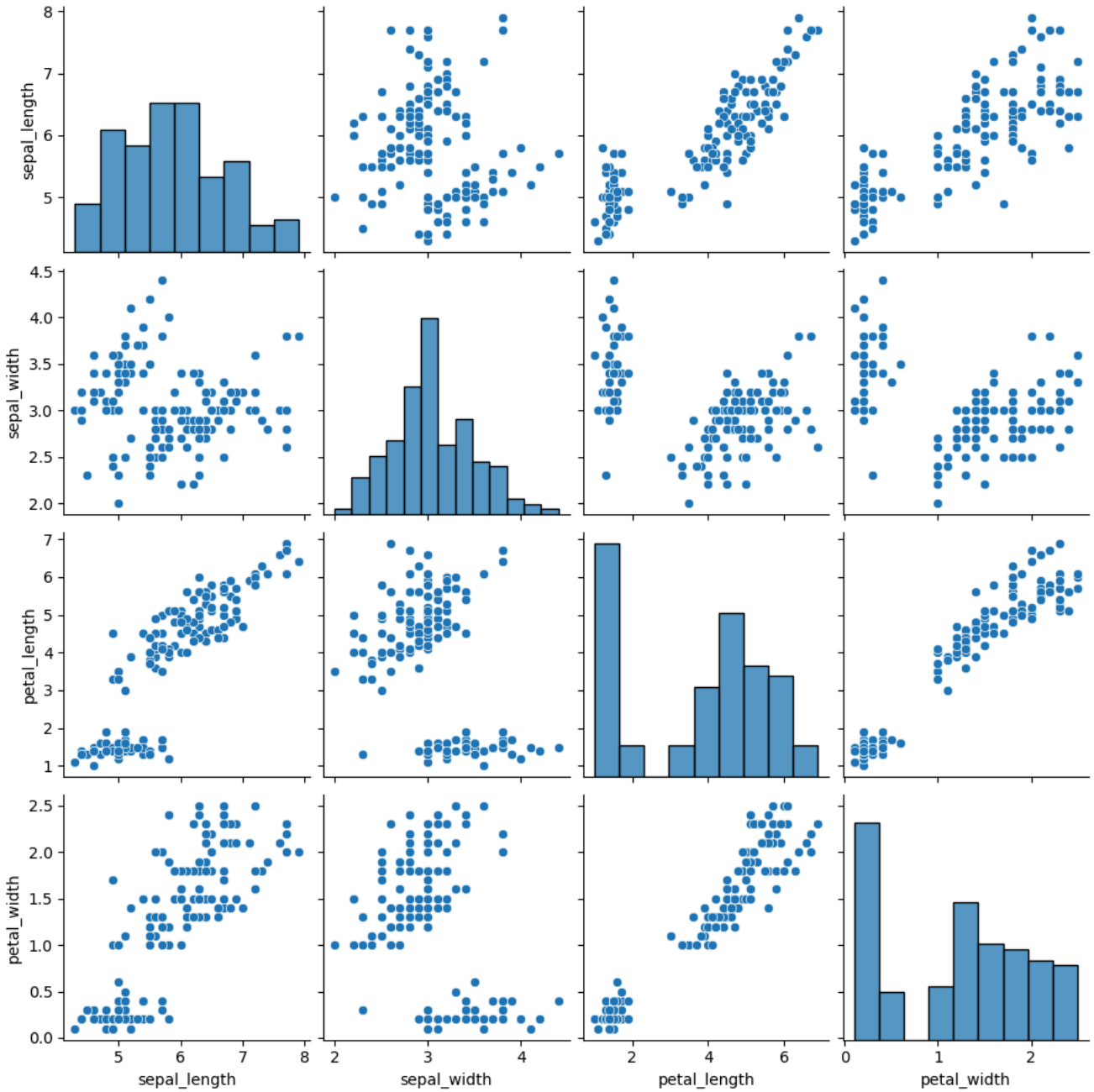
```
import matplotlib.pyplot as plt
plt.figure()
plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel('Principal Component - 1',fontsize=20)
plt.ylabel('Principal Component - 2',fontsize=20)
plt.title('Principal Component Analysis of Breast Cancer Dataset',fontsize=20)
targets=['Benign','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 1'],
plt.legend(targets,prop={'size':15})
```

```
<matplotlib.legend.Legend at 0x7dd6ebbd9a20>
<Figure size 640x480 with 0 Axes>
```

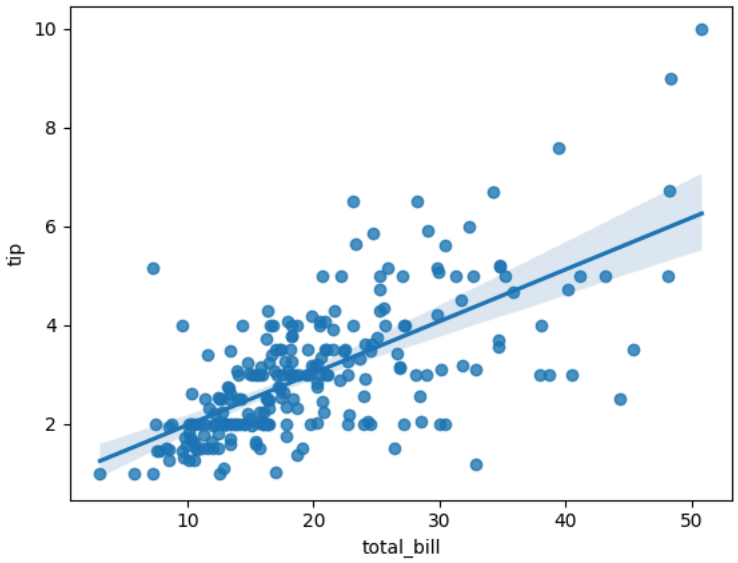


2.3 Correlation Regression

```
import matplotlib.pyplot as plt
import seaborn as sns
df=sns.load_dataset('iris')
sns.pairplot(df,kind="scatter")
plt.show()
```




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




```
import matplotlib.pyplot as plt
from scipy import stats
```

Create an array for x and y axis

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

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

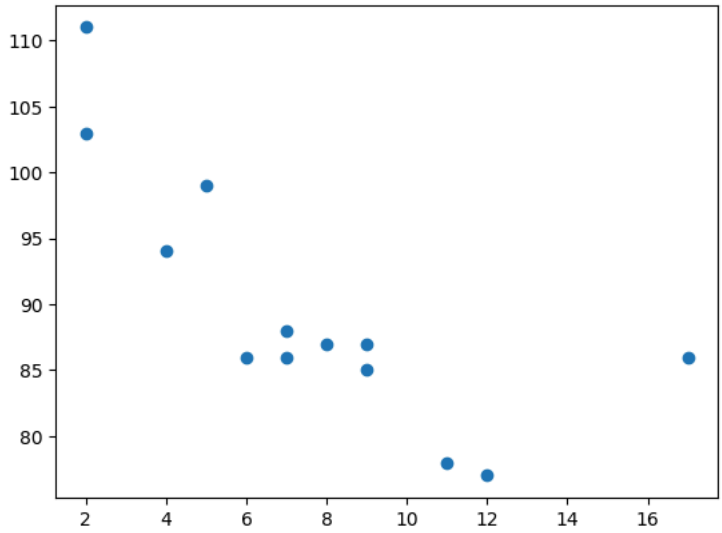
```
def myfunc(x):
  return slope* x +intercept
```

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

draw the original scatter plot

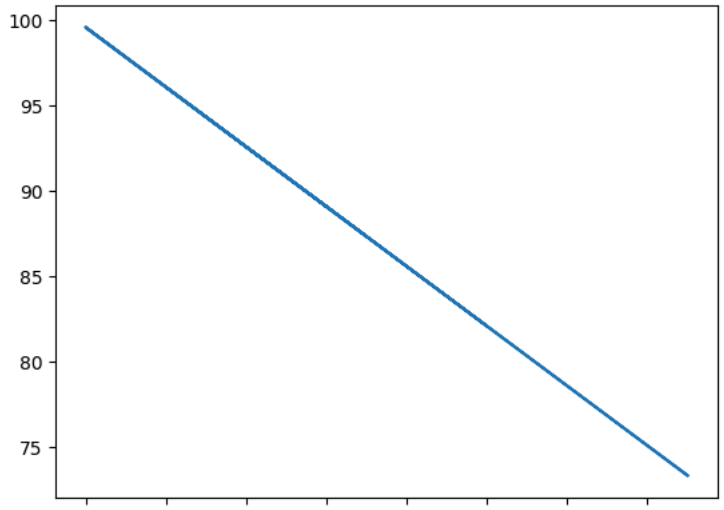
```
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x7dd6eb0f0640>
```




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

```
[<matplotlib.lines.Line2D at 0x7dd6e9705810>]
```




```
def estimate_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_p*m_p
  #here we will calculate the regression coefficient
  b_1=SS_pq/SS_pp
  b_0=m_q-b_1*m_p
```