```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(6,3), index=['a','b','c','e','f','h'], columns=['one','two','three'])
print(df)
df=df.reindex(['a','b','c','d','e','f','g','h'])
print(df['one'].isnull())
```

```
        one       two     three
a -1.551936 -0.453616  1.437867
b -0.172755 -0.712136 -1.862716
c -0.646782  2.126257 -0.624434
e -0.123754  0.410572  0.961708
f  1.047388  0.095612  0.362872
h -0.121519  0.902526 -0.676436
a    False
b    False
c    False
d     True
e    False
f    False
g     True
h    False
Name: one, dtype: bool
```

```
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
```

```
#Replace the missing values
import pandas as pd
import numpy as np
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  0.225983 -0.264988  1.386396
b       NaN       NaN       NaN
c  0.917192 -0.569902  0.630565
NaN replaced with'0':
        one       two     three
a  0.225983 -0.264988  1.386396
b  0.000000  0.000000  0.000000
c  0.917192 -0.569902  0.630565
```

```
#Fill na Forward
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5,3), index=['

print(df)
print('-------')
print(df.fillna(method='pad'))
```

```
        one       two     three
a  0.859673 -0.510472 -0.160394
b       NaN       NaN       NaN
c -1.955193 -0.224864  0.987584
d       NaN       NaN       NaN
e -1.170494  1.144965 -0.582637
f -0.989585  1.577130 -0.062854
```

```
g       NaN       NaN       NaN
h -0.925883 -0.945369 -0.407574
-------
        one       two     three
a  0.859673 -0.510472 -0.160394
b  0.859673 -0.510472 -0.160394
c -1.955193 -0.224864  0.987584
d -1.955193 -0.224864  0.987584
e -1.170494  1.144965 -0.582637
f -0.989585  1.577130 -0.062854
g -0.989585  1.577130 -0.062854
h -0.925883 -0.945369 -0.407574
```

```
#Fill na Backward
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)
print(df.fillna(method='bfill'))
```

```
        one       two     three
a  0.431689 -0.065133  0.211753
b       NaN       NaN       NaN
c  0.516594 -0.612245 -1.536742
d       NaN       NaN       NaN
e -0.619300  0.577024 -0.151975
f -1.755663 -0.972693 -0.466179
g       NaN       NaN       NaN
h  1.494090 -1.264824  0.163906
        one       two     three
a  0.431689 -0.065133  0.211753
b  0.516594 -0.612245 -1.536742
c  0.516594 -0.612245 -1.536742
d -0.619300  0.577024 -0.151975
e -0.619300  0.577024 -0.151975
f -1.755663 -0.972693 -0.466179
g  1.494090 -1.264824  0.163906
h  1.494090 -1.264824  0.163906
```

```
#Drop the missng 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)
print(df.dropna())
```

```
        one       two     three
a  0.972784 -0.766895 -0.333653
b       NaN       NaN       NaN
c  1.160967  0.251932  1.863470
d       NaN       NaN       NaN
e  1.451891  0.902215 -1.279017
f  0.778869 -1.447800 -1.204904
g       NaN       NaN       NaN
h -0.652015  0.369984  1.027785
        one       two     three
a  0.972784 -0.766895 -0.333653
c  1.160967  0.251932  1.863470
e  1.451891  0.902215 -1.279017
f  0.778869 -1.447800 -1.204904
h -0.652015  0.369984  1.027785
```

```
# Data Preprocessing
import pandas as pd
import numpy as np
df=pd.read_csv("/content/2,1 dataset 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
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 183 entries, 1 to 889
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  183 non-null    int64
 1   Survived     183 non-null    int64
 2   Pclass       183 non-null    int64
 3   Name         183 non-null    object
 4   Sex          183 non-null    object
 5   Age          183 non-null    float64
 6   SibSp        183 non-null    int64
 7   Parch        183 non-null    int64
 8   Ticket       183 non-null    object
 9   Fare         183 non-null    float64
 10  Cabin        183 non-null    object
 11  Embarked     183 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 18.6+ KB
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 183 entries, 1 to 889
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  183 non-null    int64
 1   Survived     183 non-null    int64
 2   Pclass       183 non-null    int64
 3   Sex          183 non-null    object
 4   Age          183 non-null    float64
 5   SibSp        183 non-null    int64
 6   Parch        183 non-null    int64
 7   Fare         183 non-null    float64
 8   Embarked     183 non-null    object
dtypes: float64(2), int64(5), object(2)
memory usage: 14.3+ KB
```

```
#Drop the rows having no values
df=df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 183 entries, 1 to 889
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  183 non-null    int64
 1   Survived     183 non-null    int64
 2   Pclass       183 non-null    int64
 3   Sex          183 non-null    object
 4   Age          183 non-null    float64
 5   SibSp        183 non-null    int64
 6   Parch        183 non-null    int64
 7   Fare         183 non-null    float64
 8   Embarked     183 non-null    object
dtypes: float64(2), int64(5), object(2)
memory usage: 14.3+ KB
```

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

```
     PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch     Fare  \
1              2         1       1  female  38.0      1      0  71.2833
3              4         1       1  female  35.0      1      0  53.1000
6              7         0       1    male  54.0      0      0  51.8625
10            11         1       3  female   4.0      1      1  16.7000
11            12         1       1  female  58.0      0      0  26.5500
..           ...       ...     ...     ...   ...    ...    ...      ...
871          872         1       1  female  47.0      1      1  52.5542
872          873         0       1    male  33.0      0      0   5.0000
879          880         1       1  female  56.0      0      1  83.1583
887          888         1       1  female  19.0      0      0  30.0000
889          890         1       1    male  26.0      0      0  30.0000

    Embarked
1          C
3          S
6          S
10         S
11         S
..       ...
871        S
872        S
879        C
887        S
889        C

[183 rows x 9 columns]
```

```
#Transfer the eighth columns
titanic_dummies=pd.concat(dummies, axis=1)
print(df)
```

```
     PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch     Fare  \
1              2         1       1  female  38.0      1      0  71.2833
3              4         1       1  female  35.0      1      0  53.1000
6              7         0       1    male  54.0      0      0  51.8625
10            11         1       3  female   4.0      1      1  16.7000
11            12         1       1  female  58.0      0      0  26.5500
..           ...       ...     ...     ...   ...    ...    ...      ...
871          872         1       1  female  47.0      1      1  52.5542
872          873         0       1    male  33.0      0      0   5.0000
879          880         1       1  female  56.0      0      1  83.1583
887          888         1       1  female  19.0      0      0  30.0000
889          890         1       1    male  26.0      0      0  30.0000

    Embarked
1          C
3          S
6          S
10         S
11         S
..       ...
871        S
872        S
879        C
887        S
889        C

[183 rows x 9 columns]
```

```
#Concatenate the values with data frame
df=pd.concat((df,titanic_dummies),axis=1)
print(df)
```

```
     PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch     Fare  \
1              2         1       1  female  38.0      1      0  71.2833
3              4         1       1  female  35.0      1      0  53.1000
6              7         0       1    male  54.0      0      0  51.8625
10            11         1       3  female   4.0      1      1  16.7000
11            12         1       1  female  58.0      0      0  26.5500
..           ...       ...     ...     ...   ...    ...    ...      ...
871          872         1       1  female  47.0      1      1  52.5542
872          873         0       1    male  33.0      0      0   5.0000
```

```
879          880         1       1  female  56.0      0      1  83.1583
887          888         1       1  female  19.0      0      0  30.0000
889          890         1       1    male  26.0      0      0  30.0000

    Embarked  1  2  3  female  male  C  Q  S
1          C  1  0  0       1     0  1  0  0
3          S  1  0  0       1     0  0  0  1
6          S  1  0  0       0     1  0  0  1
10         S  0  0  1       1     0  0  0  1
11         S  1  0  0       1     0  0  0  1
..       ... .. .. ..     ...   ... .. .. ..
871        S  1  0  0       1     0  0  0  1
872        S  1  0  0       0     1  0  0  1
879        C  1  0  0       1     0  1  0  0
887        S  1  0  0       1     0  0  0  1
889        C  1  0  0       0     1  1  0  0

[183 rows x 17 columns]
```

```
#Removed the unwanted cols
df=df.drop(['Pclass', 'Sex', 'Embarked'],axis=1)
print(df)
```

```
     PassengerId  Survived   Age  SibSp  Parch     Fare  1  2  3  female  \
1              2         1  38.0      1      0  71.2833  1  0  0       1
3              4         1  35.0      1      0  53.1000  1  0  0       1
6              7         0  54.0      0      0  51.8625  1  0  0       0
10            11         1   4.0      1      1  16.7000  0  0  1       1
11            12         1  58.0      0      0  26.5500  1  0  0       1
..           ...       ...   ...    ...    ...      ... .. .. ..     ...
871          872         1  47.0      1      1  52.5542  1  0  0       1
872          873         0  33.0      0      0   5.0000  1  0  0       0
879          880         1  56.0      0      1  83.1583  1  0  0       1
887          888         1  19.0      0      0  30.0000  1  0  0       1
889          890         1  26.0      0      0  30.0000  1  0  0       0

     male  C  Q  S
1       0  1  0  0
3       0  0  0  1
6       1  0  0  1
10      0  0  0  1
11      0  0  0  1
..    ... .. .. ..
871     0  0  0  1
872     1  0  0  1
879     0  1  0  0
887     0  0  0  1
889     1  1  0  0

[183 rows x 14 columns]
```

```
#Min 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))
```

```
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],
[50,0.005],
[88,0.07],
[4,0.1]])
print(data)
#define standard scaler
scaler=StandardScaler()
#transform data
scaled=scaler.fit_transform(data)
print(scaled)
```

```
[[1.0e+02 1.0e-03]
 [8.0e+00 5.0e-02]
 [5.0e+01 5.0e-03]
 [8.8e+01 7.0e-02]
 [4.0e+00 1.0e-01]]
[[ 1.26398112 -1.16389967]
 [-1.06174414  0.12639634]
 [ 0.         -1.05856939]
 [ 0.96062565  0.65304778]
 [-1.16286263  1.44302493]]
```