In [1]:

```python
import pandas as pd
import numpy as np
```

In [2]:

```python
df_train=pd.read_csv("train1.csv")
df_test=pd.read_csv('tes1t.csv')
```

In [3]:

```python
df_train.head()
```

Out[3]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	0	0	0
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	0	0
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	0	0	0
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	0	0	0

5 rows × 378 columns

In [4]:

```python
df_test.head()
```

Out[4]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	0	0

5 rows × 377 columns

In [5]:

```python
df_train.shape
```

Out[5]:

```
(4209, 378)
```

In [6]:

```python
df_test.shape
```

Out[6]:

```
(4209, 377)
```

In [7]:

```python
y_train=df_train['y'].values
```

In [8]:

```python
y_train
```

Out[8]:

```
array([130.81,  88.53,  76.26, ..., 109.22,  87.48, 110.85])
```

TASK1-If for any column(s), the variance is equal to zero, then you need to remove those variable(s).

In [9]:

```python
df_train.var()
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_2640\3907004805.py:1: FutureWarning: Dropping o
f nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in
a future version this will raise TypeError.  Select only valid columns before calling th
e reduction.
  df_train.var()
```

Out[9]:

```
ID      5.941936e+06
y       1.607667e+02
X10     1.313092e-02
X11     0.000000e+00
X12     6.945713e-02
            ...
X380    8.014579e-03
X382    7.546747e-03
X383    1.660732e-03
X384    4.750593e-04
X385    1.423823e-03
Length: 370, dtype: float64
```

In [10]:

```
df_train.var () ==0
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_2640\2195562811.py:1: FutureWarning: Dropping o
f nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in
a future version this will raise TypeError.  Select only valid columns before calling th
e reduction.
  df_train.var () ==0
```

Out[10]:

```
ID      False
y       False
X10     False
X11      True
X12     False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 370, dtype: bool
```

In [11]:

```
(df_train.var () ==0).values
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_2640\4044553682.py:1: FutureWarning: Dropping o
f nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in
a future version this will raise TypeError.  Select only valid columns before calling th
e reduction.
  (df_train.var () ==0).values
```

Loading [MathJax]/extensions/Safe.js

```
Out[11]: array([False, False, False,  True, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False,  True, False, False, False, False, False, False,
                False, False, False, False, False, False, False,  True, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False,  True, False,  True, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False,  True, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False,  True,  True, False, False,
                 True, False, False, False,  True, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                 True, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False,  True,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False])
```

```
In [12]: variance_with_zero=df_train.var()[df_train.var()==0].index.values
         variance_with_zero
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_2640\3026110461.py:1: FutureWarning: Dropping o
f nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in
a future version this will raise TypeError.  Select only valid columns before calling th
e reduction.
  variance_with_zero=df_train.var()[df_train.var()==0].index.values
```

```
Out[12]: array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
                'X293', 'X297', 'X330', 'X347'], dtype=object)
```

```
In [13]: df_train =df_train.drop(variance_with_zero, axis=1)
```

```
In [14]: df_train.shape
```

```
Out[14]: (4209, 366)
```

```
In [15]: df_train=df_train.drop('ID',axis=1)
```

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

Loading [MathJax]/extensions/Safe.js

```
Out[16]: (4209, 365)
```

```
In [17]: df_train.head()
```

Out[17]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X38
0	130.81	k	v	at	a	d	u	j	o	0	...	0	0	1	0	0	0	0	0	
1	88.53	k	t	av	e	d	y	l	o	0	...	1	0	0	0	0	0	0	0	
2	76.26	az	w	n	c	d	x	j	x	0	...	0	0	0	0	0	0	1	0	
3	80.62	az	t	n	f	d	x	l	e	0	...	0	0	0	0	0	0	0	0	
4	78.02	az	v	n	f	d	h	d	n	0	...	0	0	0	0	0	0	0	0	

5 rows × 365 columns

TASK2-Check for null and unique values for test and train sets.

```
In [18]: df_train.isnull().sum().values
```

```
Out[18]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

```
In [19]: df_test.isnull().sum().values
```

```
Out[19]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0], dtype=int64)
```

TASK3-Apply label encoder

In [20]:

```
train_object_datasets=df_train.select_dtypes(include='object')
train_object_datasets
```

Out[20]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	k	v	at	a	d	u	j	o
1	k	t	av	e	d	y	l	o
2	az	w	n	c	d	x	j	x
3	az	t	n	f	d	x	l	e
4	az	v	n	f	d	h	d	n
...	...	...	...	...	...	...	...	...
4204	ak	s	as	c	d	aa	d	q
4205	j	o	t	d	d	aa	h	h
4206	ak	v	r	a	d	aa	g	e
4207	al	r	e	f	d	aa	l	u
4208	z	r	ae	c	d	aa	g	w

4209 rows × 8 columns

In [21]:

```
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [22]:

```
df_train['X0'].unique()
```

Out[22]:

```
array(['k', 'az', 't', 'al', 'o', 'w', 'j', 'h', 's', 'n', 'ay', 'f', 'x',
       'y', 'aj', 'ak', 'am', 'z', 'q', 'at', 'ap', 'v', 'af', 'a', 'e',
       'ai', 'd', 'aq', 'c', 'aa', 'ba', 'as', 'i', 'r', 'b', 'ax', 'bc',
       'u', 'ad', 'au', 'm', 'l', 'aw', 'ao', 'ac', 'g', 'ab'],
      dtype=object)
```

In [23]:

```
df_train['X0'] = le.fit_transform(df_train['X0'])
```

In [24]:

```
df_train['X0'].unique()
```

Out[24]:

```
array([32, 20, 40,  9, 36, 43, 31, 29, 39, 35, 19, 27, 44, 45,  7,  8, 10,
       46, 37, 15, 12, 42,  5,  0, 26,  6, 25, 13, 24,  1, 22, 14, 30, 38,
       21, 18, 23, 41,  4, 16, 34, 33, 17, 11,  3, 28,  2])
```

In [25]:

```
df_train['X1'] = le.fit_transform(df_train['X1'])
df_train['X2'] = le.fit_transform(df_train['X2'])
df_train['X3'] = le.fit_transform(df_train['X3'])
df_train['X4'] = le.fit_transform(df_train['X4'])
df_train['X5'] = le.fit_transform(df_train['X5'])
df_train['X6'] = le.fit_transform(df_train['X6'])
df_train['X8'] = le.fit_transform(df_train['X8'])
```

In [26]:

```
df_train.head()
```

Out[26]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X38
0	130.81	32	23	17	0	3	24	9	14	0	...	0	0	1	0	0	0	0	0	
1	88.53	32	21	19	4	3	28	11	14	0	...	1	0	0	0	0	0	0	0	
2	76.26	20	24	34	2	3	27	9	23	0	...	0	0	0	0	0	0	1	0	
3	80.62	20	21	34	5	3	27	11	4	0	...	0	0	0	0	0	0	0	0	
4	78.02	20	23	34	5	3	12	3	13	0	...	0	0	0	0	0	0	0	0	

5 rows × 365 columns

TASK4-Perform dimensionality reduction.

In [27]:

```
from sklearn import model_selection
from sklearn.model_selection import train_test_split
```

In [28]:

```
X = df_train.drop('y', axis=1)
y = df_train.y
x_train, x_test, y_train, y_test = train_test_split(X,y,random_state=42,test_size=0.3)
```

In [29]:

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[29]:

```
((2946, 364), (1263, 364), (2946,), (1263,))
```

In [30]:

```
from sklearn.decomposition import PCA
```

In [31]:

```
sklearn_pca=PCA(n_components=0.95)
```

In [32]:

```
sklearn_pca.fit(x_train)
```

Out[32]:

```
PCA(n_components=0.95)
```

In [33]:

```
x_train=sklearn_pca.transform(x_train)
```

In [34]:

```
x_train.shape
```

Out[34]:

```
(2946, 6)
```

In [35]:

```
sklearn_pca=PCA(n_components=0.95)
sklearn_pca.fit(x_test)
```

Out[35]:

```
PCA(n_components=0.95)
```

In [36]:

```
x_test=sklearn_pca.transform(x_test)
x_test.shape
```

Out[36]:

```
(1263, 6)
```

pca for test_df

In [37]:

```
df_test.head()
```

Out[37]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	0	0

5 rows × 377 columns

In [38]:

```
test_object_datasets=df_test.select_dtypes(include='object')
test_object_datasets
```

Out[38]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	az	v	n	f	d	t	a	w
1	t	b	ai	a	d	b	g	y
2	az	v	as	f	d	a	j	j
3	az	l	n	f	d	z	l	n
4	w	s	as	c	d	y	i	m
...	...	...	...	...	...	...	...	...
4204	aj	h	as	f	d	aa	j	e
4205	t	aa	ai	d	d	aa	j	y
4206	y	v	as	f	d	aa	d	w
4207	ak	v	as	a	d	aa	c	q
4208	t	aa	ai	c	d	aa	g	r

4209 rows × 8 columns

In [39]:

```
df_test['X0'].unique()
```

Out[39]:

```
array(['az', 't', 'w', 'y', 'x', 'f', 'ap', 'o', 'ay', 'al', 'h', 'z',
       'aj', 'd', 'v', 'ak', 'ba', 'n', 'j', 's', 'af', 'ax', 'at', 'aq',
       'av', 'm', 'k', 'a', 'e', 'ai', 'i', 'ag', 'b', 'am', 'aw', 'as',
       'r', 'ao', 'u', 'l', 'c', 'ad', 'au', 'bc', 'g', 'an', 'ae', 'p',
       'bb'], dtype=object)
```

In [40]:

```
df_test['X0'] = le.fit_transform(df_test['X0'])
df_test['X1'] = le.fit_transform(df_test['X1'])
df_test['X2'] = le.fit_transform(df_test['X2'])
df_test['X3'] = le.fit_transform(df_test['X3'])
df_test['X4'] = le.fit_transform(df_test['X4'])
df_test['X5'] = le.fit_transform(df_test['X5'])
df_test['X6'] = le.fit_transform(df_test['X6'])
df_test['X8'] = le.fit_transform(df_test['X8'])
```

In [41]:

```
df_test.head()
```

Out[41]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384
0	1	21	23	34	5	3	26	0	22	0	...	0	0	0	1	0	0	0	0	0
1	2	42	3	8	0	3	9	6	24	0	...	0	0	1	0	0	0	0	0	0
2	3	21	23	17	5	3	0	9	9	0	...	0	0	0	1	0	0	0	0	0
3	4	21	13	34	5	3	31	11	13	0	...	0	0	0	1	0	0	0	0	0
4	5	45	20	17	2	3	30	8	12	0	...	1	0	0	0	0	0	0	0	0

5 rows × 377 columns

In [42]:

```
df_test = df_test.drop('ID',axis=1)
```

In [43]:

```
df_test.shape
```

Out[43]:

```
(4209, 376)
```

In [44]:

```
from sklearn.decomposition import PCA
```

In [45]:

```
sklearn_pca = PCA(n_components=0.95)
```

In [46]:

```
sklearn_pca.fit(df_test)
```

Out[46]:

```
PCA(n_components=0.95)
```

In [47]:

```
df_test=sklearn_pca.transform(df_test)
```

In [48]:

```
df_test.shape
```

Out[48]:

```
(4209, 6)
```

TASK5-Predict your test_df values using XGBoost.

In [49]:

```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\hp\anaconda3\lib\site-packages (1.7.
1)
Requirement already satisfied: numpy in c:\users\hp\anaconda3\lib\site-packages (from xg
boost) (1.21.5)
Requirement already satisfied: scipy in c:\users\hp\anaconda3\lib\site-packages (from xg
boost) (1.7.3)
Note: you may need to restart the kernel to use updated packages.
```

In [50]:

```
from sklearn import svm
from sklearn import model_selection
import xgboost as xgb
```

In [51]:

```
model = xgb.XGBRegressor(objective="reg:linear",learning_rate=0.1)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
y_pred
model.predict(df_test)
```

```
[20:56:07] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling-group-i-
03de431ba26204c4d-1/xgboost/xgboost-ci-windows/src/objective/regression_obj.cu:213: reg:
linear is now deprecated in favor of reg:squarederror.
```

```
Out[51]: array([ 79.527405,  97.88469 ,  98.678825, ..., 108.041794, 111.803894,
                 93.16432 ], dtype=float32)
```

```
In [ ]:
```

```
In [ ]:
```

Loading [MathJax]/extensions/Safe.js