```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
train_data=pd.read_csv('train.csv')
```

```
train_data
```

```
        ID       y  X0 X1  X2 X3 X4  X5 X6 X8  ...  X375  X376  X377
X378  \
0        0  130.81   k  v  at  a  d   u  j  o  ...     0     0     1
0
1        6   88.53   k  t  av  e  d   y  l  o  ...     1     0     0
0
2        7   76.26  az  w   n  c  d   x  j  x  ...     0     0     0
0
3        9   80.62  az  t   n  f  d   x  l  e  ...     0     0     0
0
4       13   78.02  az  v   n  f  d   h  d  n  ...     0     0     0
0
...    ...     ...  .. ..  .. .. ..  .. .. ..  ...   ...   ...   ...
...
4204  8405  107.39  ak  s  as  c  d  aa  d  q  ...     1     0     0
0
4205  8406  108.77   j  o   t  d  d  aa  h  h  ...     0     1     0
0
4206  8412  109.22  ak  v   r  a  d  aa  g  e  ...     0     0     1
0
4207  8415   87.48  al  r   e  f  d  aa  l  u  ...     0     0     0
0
4208  8417  110.85   z  r  ae  c  d  aa  g  w  ...     1     0     0
0

      X379  X380  X382  X383  X384  X385
0        0     0     0     0     0     0
1        0     0     0     0     0     0
2        0     0     1     0     0     0
3        0     0     0     0     0     0
4        0     0     0     0     0     0
...    ...   ...   ...   ...   ...   ...
4204     0     0     0     0     0     0
4205     0     0     0     0     0     0
4206     0     0     0     0     0     0
4207     0     0     0     0     0     0
4208     0     0     0     0     0     0

[4209 rows x 378 columns]
```

```
train_data.isna().sum(0)
```

```
ID      0
y       0
X0      0
X1      0
X2      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 378, dtype: int64
```

```
train_data.describe()
```

```
                ID            y          X10     X11          X12  \
count  4209.000000  4209.000000  4209.000000  4209.0  4209.000000   
mean   4205.960798   100.669318     0.013305     0.0     0.075077   
std    2437.608688    12.679381     0.114590     0.0     0.263547   
min       0.000000    72.110000     0.000000     0.0     0.000000   
25%    2095.000000    90.820000     0.000000     0.0     0.000000   
50%    4220.000000    99.150000     0.000000     0.0     0.000000   
75%    6314.000000   109.010000     0.000000     0.0     0.000000   
max    8417.000000   265.320000     1.000000     0.0     1.000000   

               X13          X14          X15          X16          X17
...  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
...   
mean      0.057971     0.428130     0.000475     0.002613     0.007603
...   
std       0.233716     0.494867     0.021796     0.051061     0.086872
...   
min       0.000000     0.000000     0.000000     0.000000     0.000000
...   
25%       0.000000     0.000000     0.000000     0.000000     0.000000
...   
50%       0.000000     0.000000     0.000000     0.000000     0.000000
...   
75%       0.000000     1.000000     0.000000     0.000000     0.000000
...   
max       1.000000     1.000000     1.000000     1.000000     1.000000
...   

              X375         X376         X377         X378         X379
\
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
```

```
mean      0.318841     0.057258     0.314802     0.020670     0.009503
std       0.466082     0.232363     0.464492     0.142294     0.097033
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       1.000000     0.000000     1.000000     0.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

              X380         X382         X383         X384         X385
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      0.008078     0.007603     0.001663     0.000475     0.001426
std       0.089524     0.086872     0.040752     0.021796     0.037734
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       0.000000     0.000000     0.000000     0.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

[8 rows x 370 columns]
```

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

```
variance=pow(train_data.drop(columns={'ID','y'}).std(),2).to_dict()
```

```
for key,value in variance.items():
    if(value==0):
        print("Name=",key)
```

```
Name= X11
Name= X93
Name= X107
Name= X233
```

```
Name= X235
Name= X268
Name= X289
Name= X290
Name= X293
Name= X297
Name= X330
Name= X347
```

```
# Now we Will drop this columns
train_data=train_data.drop(columns={'X11','X93','X107','X233','X235','
X268','X289','X290','X293','X297','X330','X347'})
```

```
train_data.shape
```

```
(4209, 366)
```

```
train_data.isnull().sum().any()
```

```
False
```

```
# creating dependent and independent variables
```

```
train_data_feature=train_data.drop(columns={'ID','y'})
train_data_target=train_data.y
```

```
train_data_feature.shape
```

```
(4209, 364)
```

```
train_data_target.shape
```

```
(4209,)
```

```
train_data_feature.describe(include='object')
```

```
          X0    X1    X2    X3    X4    X5    X6    X8
count   4209  4209  4209  4209  4209  4209  4209  4209
unique    47    27    44     7     4    29    12    25
top        z    aa    as     c     d     w     g     j
freq     360   833  1659  1942  4205   231  1042   277
```

```
# So we Got the columns which are obj Hence we Apply Label Encoding in
this
```

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

```
for i in train_data_feature.columns:
    data_type=train_data_feature[i].dtype
    if data_type=='object':
        train_data_feature[i]=lr.fit_transform(train_data_feature[i])
```

Perform dimensionality reduction.

```
from sklearn.decomposition import PCA
pca=PCA(n_components=0.95)
```

```
train_data_feature_trans=pca.fit_transform(train_data_feature)
```

```
train_data_feature_trans.shape
```

```
(4209, 6)
```

```
# Predict your test_df values using XGBoost.
```

```
# Split the dataset into train set & test set
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(train_data_feature_trans,train_data_target,test_size=.3,random_state=42)
```

```
# Checkthe Shape Of data
print(X_train.shape)
print(X_train.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2946, 6)
(2946, 6)
(2946,)
(1263,)
```

```
# importing the XG Boost( Extreme gradient boosting)
```

```
!pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: xgboost in
/usr/local/lib/python3.7/site-packages (1.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/site-
packages (from xgboost) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/site-
packages (from xgboost) (1.21.5)
WARNING: You are using pip version 22.0.3; however, version 22.3.1 is
available.
You should consider upgrading via the '/usr/local/bin/python3 -m pip
install --upgrade pip' command.
```

```
import xgboost as xgb
```

```
# Train the model
xgb_reg=xgb.XGBRegressor()
model=xgb_reg.fit(X_train,y_train)
# Prediction
y_pred=model.predict(X_test)
```

```
# Evaluation
from sklearn.metrics import mean_squared_error

print("RMSE IS : ",np.sqrt(mean_squared_error(y_pred,y_test)))
```

```
RMSE IS :  11.813608308644344
```

```
#Saving the model

import joblib
joblib.dump(model,'xgbmodel.pkl')
```

```
['xgbmodel.pkl']
```

```
# oad the Model by using loaded model
loaded_model=joblib.load('xgbmodel.pkl')
print('Model loaded successfully')
```

```
Model loaded successfully
```

```
# Now On test data
test_data=pd.read_csv('test.csv')

test_data=test_data.drop(columns={'X11','X93','X107','X233','X235','X2
68','X289','X290','X293','X297','X330','X347'})

# Check For The null Values

test_data.isnull().sum().any()
```

```
False
```

```
test_data_feature=test_data.drop(columns={'ID'})

test_data_feature.shape
```

```
(4209, 364)
```

```
# Apply label encoder.

for i in test_data_feature.columns:
    data_type=test_data_feature[i].dtype
    if data_type=='object':
        test_data_feature[i]=lr.fit_transform(test_data_feature[i])

test_data_feature
```

```
      X0  X1  X2  X3  X4  X5  X6  X8  X10  X12  ...  X375  X376  X377  
X378  \
0     21  23  34   5   3  26   0  22    0    0  ...     0     0     0  
1  
1     42   3   8   0   3   9   6  24    0    0  ...     0     0     1  
0  
2     21  23  17   5   3   0   9   9    0    0  ...     0     0     0  
```

```
1
3     21  13  34   5   3  31  11  13    0    0  ...     0     0     0
1
4     45  20  17   2   3  30   8  12    0    0  ...     1     0     0
0
...   ..  ..  ..  ..  ..  ..  ..  ..  ...  ...  ...   ...   ...   ...
...
4204   6   9  17   5   3   1   9   4    0    0  ...     0     0     0
0
4205  42   1   8   3   3   1   9  24    0    0  ...     0     1     0
0
4206  47  23  17   5   3   1   3  22    0    0  ...     0     0     0
0
4207   7  23  17   0   3   1   2  16    0    0  ...     0     0     1
0
4208  42   1   8   2   3   1   6  17    0    0  ...     1     0     0
0

      X379  X380  X382  X383  X384  X385
0        0     0     0     0     0     0
1        0     0     0     0     0     0
2        0     0     0     0     0     0
3        0     0     0     0     0     0
4        0     0     0     0     0     0
...    ...   ...   ...   ...   ...   ...
4204     0     0     0     0     0     0
4205     0     0     0     0     0     0
4206     0     0     0     0     0     0
4207     0     0     0     0     0     0
4208     0     0     0     0     0     0

[4209 rows x 364 columns]
```

```
# Perform dimensionality reduction.

from sklearn.decomposition import PCA
pca=PCA(n_components=.95)

test_data_feature_trans=pca.fit_transform(train_data_feature)

test_data_feature_trans.shape
```

```
(4209, 6)
```

```
# Predict your test_df values using XGBoost.

test_pred=loaded_model.predict(test_data_feature_trans)

test_pred
```

```
array([ 92.21506 ,  90.38116 ,  73.274506, ..., 109.36014 ,
88.40276 ,
        99.617065], dtype=float32)
```