```
# Importing Necessary Librarys

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

Loading the datasets

```
#Training data
train= pd.read_csv("train.csv")

#Testing dara
test= pd.read_csv("test.csv")

train.head()
```

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

   X380  X382  X383  X384  X385
0     0     0     0     0     0
1     0     0     0     0     0
2     0     1     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0

[5 rows x 378 columns]
```

```
train_target =train['y']

train_df=train.drop(['ID','y'],axis=1)

#final train data
train_df.head()
```

```
   X0 X1  X2 X3 X4 X5 X6 X8  X10  X11  ...  X375  X376  X377  X378
X379  \
0   k  v  at  a  d  u  j  o    0    0  ...     0     0     1     0
0
1   k  t  av  e  d  y  l  o    0    0  ...     1     0     0     0
0
2  az  w   n  c  d  x  j  x    0    0  ...     0     0     0     0
```

```
0
3  az  t   n  f  d  x  l  e    0    0  ...     0     0     0     0
0
4  az  v   n  f  d  h  d  n    0    0  ...     0     0     0     0
0

   X380  X382  X383  X384  X385
0     0     0     0     0     0
1     0     0     0     0     0
2     0     1     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0

[5 rows x 376 columns]
```

```
test_df=test.drop(['ID'],axis=1)

#Final Test data
test_df.head()
```

```
   X0 X1  X2 X3 X4 X5 X6 X8  X10  X11  ...  X375  X376  X377  X378
X379  \
0  az  v   n  f  d  t  a  w    0    0  ...     0     0     0     1
0
1   t  b  ai  a  d  b  g  y    0    0  ...     0     0     1     0
0
2  az  v  as  f  d  a  j  j    0    0  ...     0     0     0     1
0
3  az  l   n  f  d  z  l  n    0    0  ...     0     0     0     1
0
4   w  s  as  c  d  y  i  m    0    0  ...     1     0     0     0
0

   X380  X382  X383  X384  X385
0     0     0     0     0     0
1     0     0     0     0     0
2     0     0     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0

[5 rows x 376 columns]
```

```
print(train_df.shape)
print(train_df.columns)
```

```
(4209, 376)
Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10', 'X11',
       ...
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383',
'X384',
```

```
       'X385'],
      dtype='object', length=376)
```

```
print(test_df.shape)
print(test_df.columns)
```

```
(4209, 376)
Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10', 'X11',
       ...
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383',
'X384',
       'X385'],
      dtype='object', length=376)
```

Check variance is equal to zero

```
from sklearn.feature_selection import VarianceThreshold
variance =VarianceThreshold(threshold=0)
```

```
train_df_removed_zeros=variance.fit_transform(train_df.iloc[:,9:])
train_df_removed_zeros
```

```
array([[0, 1, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [1, 1, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
label_data=train_df.iloc[:,0:8]
label_data.head()
```

```
   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
```

```
label_data.nunique()
```

```
X0    47
X1    27
X2    44
X3     7
X4     4
X5    29
X6    12
X8    25
dtype: int64
```

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

label_df1=label_data.apply(label.fit_transform)

label_df1.head()
```

```
   X0  X1  X2  X3  X4  X5  X6  X8
0  32  23  17   0   3  24   9  14
1  32  21  19   4   3  28  11  14
2  20  24  34   2   3  27   9  23
3  20  21  34   5   3  27  11   4
4  20  23  34   5   3  12   3  13
```

```
label_df1.var()
```

```
X0    188.741938
X1     72.777974
X2    118.808135
X3      3.027295
X4      0.005461
X5     68.076236
X6      8.508730
X8     49.531868
dtype: float64
```

```
without_zeros_train_df = pd.DataFrame(train_df_removed_zeros)
without_zeros_train_df.head()
```

```
   0    1    2    3    4    5    6    7    8    9    ...  345  346
347  348  \
0    0    1    0    0    0    0    1    0    0    1  ...    0    0
1    0
1    0    0    0    0    0    0    1    0    0    0  ...    1    0
0    0
2    0    0    0    0    0    1    0    0    0    0  ...    0    0
0    0
3    0    0    0    0    0    0    0    0    0    0  ...    0    0
0    0
4    0    0    0    0    0    0    0    0    0    0  ...    0    0
0    0

   349  350  351  352  353  354
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

[5 rows x 355 columns]
```

```
train_df_final = pd.concat([label_df1,without_zeros_train_df],axis=1)
train_df_final.head()
```

```
   X0  X1  X2  X3  X4  X5  X6  X8  0  1  ...  345  346  347  348  349
350  \
0  32  23  17   0   3  24   9  14  0  1  ...    0    0    1    0    0
0
1  32  21  19   4   3  28  11  14  0  0  ...    1    0    0    0    0
0
2  20  24  34   2   3  27   9  23  0  0  ...    0    0    0    0    0
0
3  20  21  34   5   3  27  11   4  0  0  ...    0    0    0    0    0
0
4  20  23  34   5   3  12   3  13  0  0  ...    0    0    0    0    0
0

   351  352  353  354
0    0    0    0    0
1    0    0    0    0
2    1    0    0    0
3    0    0    0    0
4    0    0    0    0

[5 rows x 363 columns]
```

```
train_df_final.isna().sum().any()
```

```
False
```

```
test_df.head()
```

```
   X0 X1  X2 X3 X4 X5 X6 X8  X10  X11  ...  X375  X376  X377  X378
X379  \
0  az  v   n  f  d  t  a  w    0    0  ...     0     0     0     1
0
1   t  b  ai  a  d  b  g  y    0    0  ...     0     0     1     0
0
2  az  v  as  f  d  a  j  j    0    0  ...     0     0     0     1
0
3  az  l   n  f  d  z  l  n    0    0  ...     0     0     0     1
0
4   w  s  as  c  d  y  i  m    0    0  ...     1     0     0     0
0

   X380  X382  X383  X384  X385
0     0     0     0     0     0
1     0     0     0     0     0
2     0     0     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0
```

```
[5 rows x 376 columns]
```

```
test_df.var().sort_values().head(10)
```

```
X295    0.000000
X369    0.000000
X296    0.000000
X257    0.000000
X258    0.000000
X278    0.000238
X233    0.000238
X280    0.000238
X290    0.000238
X293    0.000238
dtype: float64
```

```
test_df_removed_zeors =variance.transform(test_df.iloc[:,9:])
test_df_removed_zeors
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
without_zeros_test_df=pd.DataFrame(test_df_removed_zeors)
without_zeros_test_df.head()
```

```
   0    1    2    3    4    5    6    7    8    9    ...  345  346
347  348  \
0    0    0    0    0    0    0    0    0    0    0  ...    0    0
0    1
1    0    0    0    0    0    0    0    1    0    0  ...    0    0
1    0
2    0    0    1    0    0    0    0    0    0    0  ...    0    0
0    1
3    0    0    0    0    0    0    0    0    0    0  ...    0    0
0    1
4    0    0    1    0    0    0    0    0    0    0  ...    1    0
0    0

   349  350  351  352  353  354
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
```

```
[5 rows x 355 columns]
```

```
label_data2=test_df.iloc[:,0:8]
label_data.head()
```

```
   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
```

```
label_df3=label_data2.apply(label.fit_transform)
```

```
label_df3.head()
```

```
   X0  X1  X2  X3  X4  X5  X6  X8
0  21  23  34   5   3  26   0  22
1  42   3   8   0   3   9   6  24
2  21  23  17   5   3   0   9   9
3  21  13  34   5   3  31  11  13
4  45  20  17   2   3  30   8  12
```

```
test_df_final = pd.concat([label_df3,without_zeros_test_df],axis=1)
test_df_final.head()
```

```
   X0  X1  X2  X3  X4  X5  X6  X8  0  1  ...  345  346  347  348  349
350  \
0  21  23  34   5   3  26   0  22  0  0  ...    0    0    0    1    0
0
1  42   3   8   0   3   9   6  24  0  0  ...    0    0    1    0    0
0
2  21  23  17   5   3   0   9   9  0  0  ...    0    0    0    1    0
0
3  21  13  34   5   3  31  11  13  0  0  ...    0    0    0    1    0
0
4  45  20  17   2   3  30   8  12  0  0  ...    1    0    0    0    0
0
```

```
   351  352  353  354
0    0    0    0    0
1    0    0    0    0
2    0    0    0    0
3    0    0    0    0
4    0    0    0    0
```

```
[5 rows x 363 columns]
```

Perform dimensionality reduction.

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 0.2)

# Train and Test split on Train dataset
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test =
train_test_split(train_df_final ,train_target,test_size=0.3,random_sta
te=42)

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

((2946, 363), (1263, 363), (2946,), (1263,))

x_train =pca.fit_transform(x_train)
x_test = pca.transform(x_test)

/usr/local/lib/python3.7/site-packages/sklearn/utils/
validation.py:1692: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str'].
An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:169
2: FutureWarning: Feature names only support names that are all
strings. Got feature names with dtypes: ['int', 'str']. An error will
be raised in 1.2.
  FutureWarning,

test_df=pca.transform(test_df_final)

/usr/local/lib/python3.7/site-packages/sklearn/utils/
validation.py:1692: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str'].
An error will be raised in 1.2.
  FutureWarning,

pca.n_components_

1

pca.explained_variance_ratio_

array([0.38296186])
```

XGBoost

```
from sklearn import svm
from sklearn.metrics import r2_score,mean_squared_error
from xgboost import XGBRegressor

xgbreg = XGBRegressor(random_state=42)

model = xgbreg.fit(x_train,y_train)
```

```
ypred_train = model.predict(x_train)
ypred_train
```

```
array([ 97.8411  , 105.01268 ,  99.95553 , ...,  93.519264,
94.643814,
        98.87075 ], dtype=float32)
```

```
ypred_test = model.predict(x_test)
ypred_test
```

```
array([ 94.144424, 104.580864, 104.58693 , ...,  98.87075 ,
101.180534,
       102.3027  ], dtype=float32)
```

```
print(r2_score(ypred_train,y_train))
```

```
0.06373876967202363
```

```
print(mean_squared_error(ypred_train,y_train))
```

```
54.37626614461824
```

```
prediction = pd.DataFrame({'ytest ':y_test,'ypred':ypred_test})
```

```
prediction
```

```
      ytest        ypred
1073   97.94   94.144424
144    96.41  104.580864
2380  105.83  104.586929
184    79.09   80.366035
2587  108.69  111.047722
...      ...         ...
2493  115.25   98.354218
3388   88.59  101.401352
3997   92.90   98.870750
383    98.24  101.180534
3364   91.46  102.302696

[1263 rows x 2 columns]
```