In [168…
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [169…
```python
train=pd.read_csv('ML/train.csv')
```

In [170…
```python
test=pd.read_csv('ML/test.csv')
```

In [171…
```python
train.head()
```

Out[171]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	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	0
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	0	0	0
3	9	80.62	az	t	n	f	d	x	l	e	...	0	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	0

5 rows × 378 columns

In [172…
```python
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 378 entries, ID to X385
dtypes: float64(1), int64(369), object(8)
memory usage: 12.1+ MB
```

In [173…
```python
train.shape
```

Out[173]: (4209, 378)

In [174…
```python
test.shape
```

Out[174]: (4209, 377)

In [175…
```python
train_data=train.drop(['ID','y'],axis=1)
train_target=train.y
```

In [177…
```python
test_data=test.drop(['ID'],axis=1)
```

REMOVING COLUMN WHERE VARINACE IS ZERO

In [128…
```python
train_data.var().sort_values(ascending=True)
```

```
C:\Users\91940\AppData\Local\Temp/ipykernel_8608/720521511.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version t
his will raise TypeError.  Select only valid columns before calling the reduction.
  train_data.var().sort_values(ascending=True)
```

Out[128]:
```
X330    0.000000
X297    0.000000
X268    0.000000
X290    0.000000
X235    0.000000
          ...
X194    0.248764
X191    0.249212
X362    0.249647
X337    0.249787
X127    0.250036
Length: 368, dtype: float64
```

In [129…
```python
keep_columns = train_data.columns[train_data.nunique()>1]
labelled_data= train_data.loc[:,keep_columns].copy()
```

In [130…
```python
keep_columns1 = test_data.columns[test_data.nunique()>1]
labelled_data1= test_data.loc[:,keep_columns1].copy()
```

In [131…
```python
labelled_data.var().sort_values(ascending=True)
```

```
C:\Users\91940\AppData\Local\Temp/ipykernel_8608/508475679.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version t
his will raise TypeError.  Select only valid columns before calling the reduction.
  labelled_data.var().sort_values(ascending=True)
```

Out[131]:
```
X257    0.000238
X207    0.000238
X280    0.000238
X288    0.000238
X33     0.000238
          ...
X186    0.248764
X191    0.249212
X362    0.249647
X337    0.249787
X127    0.250036
Length: 356, dtype: float64
```

In [132…
```python
labelled_data1.var().sort_values(ascending=True)
```

```
C:\Users\91940\AppData\Local\Temp/ipykernel_8608/3052006690.py:1: FutureWarning: Dropping of 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 the reduction.
  labelled_data1.var().sort_values(ascending=True)
```

Out[132]:
```
X278    0.000238
X233    0.000238
X280    0.000238
X293    0.000238
X290    0.000238
          ...
X194    0.249053
X337    0.249332
X191    0.249489
X362    0.249738
X127    0.249866
Length: 363, dtype: float64
```

In [133…
```python
labelled_data1.shape
```

Out[133]: (4209, 371)

In [134…
```python
labelled_data.shape
```

Out[134]: (4209, 364)

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

In [135…
```python
labelled_data.isna().sum().sort_values(ascending=False).head(20)
```

Out[135]:
```
X0      0
X263    0
X261    0
X260    0
X259    0
X258    0
X257    0
X256    0
X255    0
X254    0
X253    0
X252    0
X251    0
X250    0
X249    0
X248    0
X247    0
X246    0
X245    0
X244    0
dtype: int64
```

In [136…
```python
labelled_data1.isna().sum().sort_values(ascending=False).head(20)
```

Out[136]:
```
X0      0
X252    0
X263    0
X262    0
X261    0
X260    0
X259    0
X256    0
X255    0
X254    0
X253    0
X251    0
X288    0
X250    0
X249    0
X248    0
X247    0
X246    0
X245    0
X244    0
dtype: int64
```

In [137…
```python
labelled_data.nunique().sort_values(ascending=False)
```

Out[137]:
```
X0      47
X2      44
X5      29
X1      27
X8      25
        ..
X131     2
X130     2
X129     2
X128     2
X385     2
Length: 364, dtype: int64
```

In [138…
```python
labelled_data1.nunique().sort_values(ascending=False)
```

Out[138]:
```
X0      49
X2      45
X5      32
X1      27
X8      25
        ..
X130     2
X129     2
X128     2
X127     2
X385     2
Length: 371, dtype: int64
```

Applying label encoder

In [139…
```python
labelled_data.select_dtypes(include='object')
```

Out[139]:

	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 [140…
```python
from sklearn import preprocessing
```

In [141…
```python
for col in labelled_data.columns:
    if labelled_data.dtypes[col] == "object":
        le = preprocessing.LabelEncoder()
        le.fit(labelled_data[col])
        labelled_data[col] = le.transform(labelled_data[col])
```

In [142…
```python
labelled_data[['X0','X1','X2','X3','X4','X5','X6','X8']]
```

Out[142]:

	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
...	...	...	...	...	...	...	...	...
4204	8	20	16	2	3	0	3	16
4205	31	16	40	3	3	0	7	7
4206	8	23	38	0	3	0	6	4
4207	9	19	25	5	3	0	11	20
4208	46	19	3	2	3	0	6	22

4209 rows × 8 columns

In [143…
```python
final_train_data=labelled_data
```

In [144…
```python
for col in labelled_data1.columns:
    if labelled_data1.dtypes[col] == "object":
        le = preprocessing.LabelEncoder()
        le.fit(labelled_data1[col])
        labelled_data1[col] = le.transform(labelled_data1[col])
```

In [145…
```python
final_test_data=labelled_data1
```

In [146…
```python
final_train_data.nunique().sort_values(ascending=True)
```

Out[146]:
```
X191     2
X260     2
X259     2
X258     2
X257     2
        ..
X8      25
X1      27
X5      29
X2      44
X0      47
Length: 364, dtype: int64
```

In [147…
```python
final_test_data.nunique().sort_values(ascending=True)
```

Out[147]:
```
X192     2
X262     2
X261     2
X260     2
X259     2
        ..
X8      25
X1      27
X5      32
X2      45
X0      49
Length: 371, dtype: int64
```

In [148…
```python
common_cols = list(set(final_test_data.columns).intersection(final_train_data.columns))
common_cols
```

Out[148]:
```
['X162',
 'X21',
 'X368',
 'X325',
 'X329',
 'X328',
 'X83',
 'X240',
 'X340',
 'X373',
 'X94',
 'X331',
 'X174',
 'X125',
 'X379',
 'X110',
 'X141',
 'X371',
 'X294',
 'X281',
 'X249',
 'X108',
 'X253',
 'X271',
 'X135',
 'X32',
 'X222',
 'X178',
 'X140',
 'X226',
 'X60',
 'X212',
 'X376',
 'X167',
 'X116',
 'X291',
 'X4',
 'X236',
 'X160',
 'X155',
 'X356',
 'X73',
 'X127',
 'X280',
 'X208',
 'X238',
 'X355',
 'X195',
 'X175',
 'X154',
 'X50',
 'X177',
 'X68',
 'X52',
 'X322',
 'X287',
 'X265',
 'X126',
 'X316',
 'X62',
 'X383',
 'X326',
 'X128',
 'X359',
 'X57',
 'X375',
 'X309',
 'X179',
 'X182',
 'X91',
 'X349',
 'X338',
 'X205',
 'X153',
 'X348',
 'X69',
 'X310',
 'X166',
 'X124',
 'X65',
 'X115',
 'X221',
 'X211',
 'X105',
 'X315',
 'X92',
 'X185',
 'X20',
 'X224',
 'X19',
 'X365',
 'X189',
 'X380',
 'X90',
 'X230',
 'X245',
 'X337',
 'X46',
 'X117',
 'X215',
 'X18',
 'X56',
 'X342',
 'X191',
 'X67',
 'X138',
 'X106',
 'X277',
 'X319',
 'X307',
 'X377',
 'X298',
 'X55',
 'X30',
 'X120',
 'X339',
 'X323',
 'X206',
 'X276',
 'X77',
 'X270',
 'X275',
 'X279',
 'X2',
 'X53',
 'X6',
 'X130',
 'X201',
 'X29',
 'X64',
 'X95',
 'X202',
 'X219',
 'X104',
 'X157',
 'X148',
 'X144',
 'X264',
 'X51',
 'X362',
 'X363',
 'X168',
 'X299',
 'X256',
 'X88',
 'X59',
 'X324',
 'X97',
 'X42',
 'X164',
 'X35',
 'X302',
 'X266',
 'X300',
 'X43',
 'X308',
 'X320',
 'X24',
 'X332',
 'X255',
 'X360',
 'X262',
 'X317',
 'X48',
 'X261',
 'X285',
 'X156',
 'X345',
 'X214',
 'X225',
 'X12',
 'X301',
 'X321',
 'X216',
 'X14',
 'X3',
 'X263',
 'X81',
 'X234',
 'X223',
 'X132',
 'X66',
 'X96',
 'X161',
 'X85',
 'X170',
 'X147',
 'X119',
 'X39',
 'X112',
 'X288',
 'X102',
 'X200',
 'X181',
 'X194',
 'X122',
 'X282',
 'X152',
 'X314',
 'X136',
 'X78',
 'X28',
 'X382',
 'X41',
 'X80',
 'X118',
 'X159',
 'X31',
 'X220',
 'X63',
 'X254',
 'X250',
 'X186',
 'X197',
 'X172',
 'X351',
 'X385',
 'X36',
 'X213',
 'X103',
 'X15',
 'X5',
 'X350',
 'X203',
 'X384',
 'X243',
 'X284',
 'X17',
 'X246',
 'X58',
 'X190',
 'X292',
 'X199',
 'X84',
 'X34',
 'X241',
 'X33',
 'X1',
 'X304',
 'X198',
 'X229',
 'X113',
 'X343',
 'X231',
 'X87',
 'X259',
 'X372',
 'X335',
 'X361',
 'X247',
 'X89',
 'X40',
 'X16',
 'X207',
 'X142',
 'X101',
 'X114',
 'X305',
 'X76',
 'X252',
 'X283',
 'X232',
 'X61',
 'X151',
 'X150',
 'X244',
 'X209',
 'X158',
 'X334',
 'X109',
 'X357',
 'X23',
 'X70',
 'X341',
 'X82',
 'X44',
 'X74',
 'X306',
 'X242',
 'X184',
 'X248',
 'X133',
 'X374',
 'X278',
 'X336',
 'X312',
 'X27',
 'X274',
 'X239',
 'X49',
 'X318',
 'X344',
 'X123',
 'X370',
 'X367',
 'X134',
 'X129',
 'X228',
 'X176',
 'X173',
 'X366',
 'X272',
 'X218',
 'X37',
 'X0',
 'X26',
 'X354',
 'X22',
 'X47',
 'X54',
 'X165',
 'X358',
 'X352',
 'X146',
 'X210',
 'X139',
 'X79',
 'X204',
 'X137',
 'X353',
 'X111',
 'X251',
 'X13',
 'X346',
 'X227',
 'X187',
 'X313',
 'X86',
 'X237',
 'X378',
 'X260',
 'X196',
 'X163',
 'X286',
 'X38',
 'X143',
 'X98',
 'X10',
 'X217',
 'X145',
 'X71',
 'X169',
 'X192',
 'X327',
 'X364',
 'X75',
 'X269',
 'X99',
 'X311',
 'X100',
 'X131',
 'X45',
 'X267',
 'X273',
 'X180',
 'X8',
 'X183',
 'X333',
 'X171']
```

In [149…
```python
final_test_data=final_test_data[common_cols]
final_train_data=final_train_data[common_cols]
```

In [150…
```python
final_train_data.head()
```

Out[150]:

	X162	X21	X368	X325	X329	X328	X83	X240	X340	X373	...	X100	X131	X45	X267	X273	X180	X8	X183	X333	X171
0	0	1	0	0	1	0	0	0	0	0	...	0	1	0	0	1	0	14	0	0	0
1	0	0	0	0	1	0	0	0	0	0	...	1	0	0	0	1	0	14	0	0	0
2	1	0	0	0	0	1	0	0	0	0	...	0	0	0	0	1	0	23	0	0	0
3	1	0	0	0	0	1	0	0	0	0	...	0	0	0	0	1	0	4	0	0	0
4	1	0	0	0	0	1	0	0	0	0	...	0	0	0	0	1	0	13	0	0	0

5 rows × 359 columns

In [151…
```python
final_test_data.head()
```

Out[151]:

	X162	X21	X368	X325	X329	X328	X83	X240	X340	X373	...	X100	X131	X45	X267	X273	X180	X8	X183	X333	X171
0	1	0	0	0	0	1	0	0	0	0	...	0	0	0	0	1	0	22	0	0	0
1	0	0	1	0	0	0	0	0	0	0	...	0	0	1	0	1	0	24	0	0	1
2	1	0	0	0	1	1	0	0	0	0	...	1	0	0	0	0	0	9	0	0	0
3	1	0	0	0	0	1	0	0	0	0	...	0	0	0	0	1	0	13	0	0	0
4	0	0	0	0	1	0	0	0	0	0	...	1	0	0	0	1	0	12	0	0	1

5 rows × 359 columns

Performing dimensionality reduction.

In [152…
```python
final_train_data.describe()
```

Out[152]:

	X162	X21	X368	X325	X329	X328	X83	X240	X340	X373	...	X100	X131	X45	X267	X273	X180	X8
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	...	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000
mean	0.040865	0.002613	0.062723	0.005702	0.435258	0.040152	0.001188	0.002851	0.022333	0.019244	...	0.690188	0.026610	0.253267	0.009028	0.720124	0.157995	11.611309
std	0.198000	0.051061	0.242492	0.075305	0.495850	0.196339	0.034450	0.053325	0.147782	0.137399	...	0.462471	0.160959	0.434934	0.094599	0.448992	0.364779	7.037888
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	11.000000
75%	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	18.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	24.000000

8 rows × 359 columns

In [153…
```python
from sklearn.preprocessing import StandardScaler  # to standardize the features
from sklearn.decomposition import PCA  # to apply PCA
import seaborn as sns  # to plot the heat maps
```

In [154…
```python
scalar = StandardScaler()
final_train_data = pd.DataFrame(scalar.fit_transform(final_train_data)) #scaling the data
final_test_data = pd.DataFrame(scalar.fit_transform(final_test_data))
```

In [155…
```python
final_test_data.shape
```

Out[155]: (4209, 359)

In [156…
```python
final_train_data.shape
```

Out[156]: (4209, 359)

In [157…
```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(final_train_data,train_target,test_size=0.25, random_state=0)
```

In [158…
```python
pca = PCA(n_components = 3)
x_train=pca.fit_transform(x_train)
x_test=pca.fit_transform(x_test)
final_test_data = pca.transform(final_test_data)
final_train_data = pca.transform(final_train_data)
```

Predicting test_df values using XGBoost.

In [159…
```python
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\91940\anaconda\lib\site-packages (1.7.2)
Requirement already satisfied: scipy in c:\users\91940\anaconda\lib\site-packages (from xgboost) (1.7.3)
Requirement already satisfied: numpy in c:\users\91940\anaconda\lib\site-packages (from xgboost) (1.21.5)
Note: you may need to restart the kernel to use updated packages.
```

In [160…
```python
from sklearn import svm
from sklearn.metrics import r2_score,mean_squared_error
from xgboost import XGBRegressor
xgbr=XGBRegressor(random_state=0)
```

In [161…
```python
model=xgbr.fit(x_train,y_train)
```

In [162…
```python
y_pred_train=model.predict(x_train)
y_pred_train
```

Out[162]:
```
array([104.73686,  96.26712,  92.62421, ...,  99.69573, 110.29155,
       114.31125], dtype=float32)
```

In [163…
```python
y_train
```

Out[163]:
```
2807    108.62
1070     93.41
951      91.21
1837     94.50
97      108.37
         ...
1033    140.41
3264    104.73
1653    102.77
2607    113.94
2732    110.10
Name: y, Length: 3156, dtype: float64
```

In [164…
```python
r2=r2_score(y_train,y_pred_train)
print(r2)
```

```
0.8362199177496813
```

In [165…
```python
y_pred_test=model.predict(x_test)
y_pred_test
```

Out[165]:
```
array([107.29135 ,  95.89006 , 108.21682 , ...,  90.55357 ,  90.342094,
       105.28956 ], dtype=float32)
```

In [166…
```python
y_test
```

Out[166]:
```
3431     96.49
2131     96.93
2680    114.22
195      88.10
3032     92.63
         ...
3338    111.21
71      113.40
1920     87.17
2858    110.80
169     110.91
Name: y, Length: 1053, dtype: float64
```

In [181…
```python
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

In [183…
```python
print('MAE', mean_absolute_error(y_train,y_pred_train))
```

```
MAE 3.54929422298765
```

```
>>> print("Hi", 'there. Good eve', 20)
Hi there. Good eve 20
```