Project 1

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
data_train = pd.read_csv('train.csv')
data_test=pd.read_csv('test.csv')
```

In [3]:

```
data_train
```

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X38
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	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
3	9	80.62	az	t	n	f	d	x	l	e	...	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8405	107.39	ak	s	as	c	d	aa	d	q	...	1	0	0	0	0	0	0
4205	8406	108.77	j	o	t	d	d	aa	h	h	...	0	1	0	0	0	0	0
4206	8412	109.22	ak	v	r	a	d	aa	g	e	...	0	0	1	0	0	0	0
4207	8415	87.48	al	r	e	f	d	aa	l	u	...	0	0	0	0	0	0	0
4208	8417	110.85	z	r	ae	c	d	aa	g	w	...	1	0	0	0	0	0	0

4209 rows × 378 columns

In [4]:

```
data_test
```

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8410	aj	h	as	f	d	aa	j	e	0	...	0	0	0	0	0	0	0
4205	8411	t	aa	ai	d	d	aa	j	y	0	...	0	1	0	0	0	0	0
4206	8413	y	v	as	f	d	aa	d	w	0	...	0	0	0	0	0	0	0
4207	8414	ak	v	as	a	d	aa	c	q	0	...	0	0	1	0	0	0	0
4208	8416	t	aa	ai	c	d	aa	g	r	0	...	1	0	0	0	0	0	0

4209 rows × 377 columns

In [5]:

```
data_train.shape
```

```
(4209, 378)
```

In [6]:

```
data_test.shape
```

```
(4209, 377)
```

In [7]:

```
data_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 [8]:

```
data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 377 entries, ID to X385
dtypes: int64(369), object(8)
memory usage: 12.1+ MB
```

Problem statement 1

- variance = 0
- removing those values

For train data

In [9]:

```
print(len(data_train['X93'].unique()))
```

```
1
```

In [10]:

```
for each in data_train:
    if len(data_train[each].unique())==1:
        print(each)
```

```
X11
X93
X107
X233
X235
X268
X289
X290
X293
X297
X330
X347
```

In [11]:

```
data_train.drop(['X11','X93','X107','X233','X235','X268','X289','X290','X293','X297',
```

For test data

In [12]:

```
for each in data_test:
    if len(data_test[each].unique())==1:
        print(each)
```

```
X257
X258
X295
X296
X369
```

In [13]:

```
data_test.drop(['X257','X258','X295','X296','X369'],axis=1,inplace= True)
```

Problem statement 2

- finding null and unique values

In [14]:

```
for each in data_train:
    if data_train[each].isnull().sum()==True:
        print(each)
```

In [15]:

```
for each in data_test:
    if data_test[each].isnull().sum()==True:
        print(each)
```

- no null values

Problem statement 3

- Finding Categorical columns
- applying label encoder

In [16]:

```
for each in data_train:
    print(np.dtype(data_train[each]))
```

```
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
```

In [17]:

```
categorical_col_train=[c for c in data_train if data_train[c].dtype==np.dtype('O')]
```

In [18]:

```
categorical_col_train
```

```
['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

In [19]:

```
categorical_col_test=[c for c in data_test if data_test[c].dtype==np.dtype('O')]
```

In [20]:

```
categorical_col_test
```

```
['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

- applyting label encoder

- train data

In [21]:

```
catdf_train= data_train[categorical_col_train]
```

In [22]:

```
catdf_test= data_test[categorical_col_test]
```

In [23]:

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

In [24]:

```
catdf_train
```

	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 [25]:

```
catdf_test
```

	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 [26]:

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

In [27]:

```
data_train
```

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X38
0	0	130.81	32	23	17	0	3	24	9	14	...	0	0	1	0	0	0	0
1	6	88.53	32	21	19	4	3	28	11	14	...	1	0	0	0	0	0	0
2	7	76.26	20	24	34	2	3	27	9	23	...	0	0	0	0	0	0	1
3	9	80.62	20	21	34	5	3	27	11	4	...	0	0	0	0	0	0	0
4	13	78.02	20	23	34	5	3	12	3	13	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8405	107.39	8	20	16	2	3	0	3	16	...	1	0	0	0	0	0	0
4205	8406	108.77	31	16	40	3	3	0	7	7	...	0	1	0	0	0	0	0
4206	8412	109.22	8	23	38	0	3	0	6	4	...	0	0	1	0	0	0	0
4207	8415	87.48	9	19	25	5	3	0	11	20	...	0	0	0	0	0	0	0
4208	8417	110.85	46	19	3	2	3	0	6	22	...	1	0	0	0	0	0	0

4209 rows × 366 columns

- test data

In [28]:

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

In [29]:

```
data_test
```

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382
0	1	21	23	34	5	3	26	0	22	0	...	0	0	0	1	0	0	0
1	2	42	3	8	0	3	9	6	24	0	...	0	0	1	0	0	0	0
2	3	21	23	17	5	3	0	9	9	0	...	0	0	0	1	0	0	0
3	4	21	13	34	5	3	31	11	13	0	...	0	0	0	1	0	0	0
4	5	45	20	17	2	3	30	8	12	0	...	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8410	6	9	17	5	3	1	9	4	0	...	0	0	0	0	0	0	0
4205	8411	42	1	8	3	3	1	9	24	0	...	0	1	0	0	0	0	0
4206	8413	47	23	17	5	3	1	3	22	0	...	0	0	0	0	0	0	0
4207	8414	7	23	17	0	3	1	2	16	0	...	0	0	1	0	0	0	0
4208	8416	42	1	8	2	3	1	6	17	0	...	1	0	0	0	0	0	0

4209 rows × 372 columns

feature selection

In [30]:

```
X = data_train.drop(['y','ID'],axis=1)
```

In [31]:

```
X
```

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X380	X382
0	32	23	17	0	3	24	9	14	0	0	...	0	0	1	0	0	0	0
1	32	21	19	4	3	28	11	14	0	0	...	1	0	0	0	0	0	0
2	20	24	34	2	3	27	9	23	0	0	...	0	0	0	0	0	0	1
3	20	21	34	5	3	27	11	4	0	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	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8	20	16	2	3	0	3	16	0	0	...	1	0	0	0	0	0	0
4205	31	16	40	3	3	0	7	7	0	0	...	0	1	0	0	0	0	0
4206	8	23	38	0	3	0	6	4	0	1	...	0	0	1	0	0	0	0
4207	9	19	25	5	3	0	11	20	0	0	...	0	0	0	0	0	0	0
4208	46	19	3	2	3	0	6	22	0	0	...	1	0	0	0	0	0	0

4209 rows × 364 columns

In [32]:

```
y= data_train['y']
```

In [33]:

```
y
```

```
0       130.81
1        88.53
2        76.26
3        80.62
4        78.02
         ...
4204    107.39
4205    108.77
4206    109.22
4207     87.48
4208    110.85
Name: y, Length: 4209, dtype: float64
```

Model Building

In [34]:

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

In [35]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_stat
```

In [36]:

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

In [37]:

```
obj=PCA(n_components= 0.7)
```

In [38]:

```
obj.fit(X_train)
```

```
PCA(n_components=0.7)
```

In [39]:

```
X_train_PCA=obj.transform(X_train)
X_test_PCA=obj.transform(X_test)
```

In [40]:

```
X_train.shape
```

```
(2820, 364)
```

In [41]:

```
y_train.shape
```

```
(2820,)
```

In [42]:

```
X_train_PCA.shape
```

```
(2820, 3)
```

Model Building

In [43]:

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in d:\users\coold\anaconda3\lib\site-packages (1.6.2)
Requirement already satisfied: scipy in d:\users\coold\anaconda3\lib\site-packages (from xgboost) (1.7.3)
Requirement already satisfied: numpy in d:\users\coold\anaconda3\lib\site-packages (from xgboost) (1.21.5)
```

In [44]:

```
from xgboost import XGBRegressor
```

```
d:\Users\coold\anaconda3\lib\site-packages\xgboost\compat.py:36: FutureWarning: pandas.Int64Index is deprecat
andas in a future version. Use pandas.Index with the appropriate dtype instead.
  from pandas import MultiIndex, Int64Index
```

In [45]:

```
model=XGBRegressor()
```

In [46]:

```
model.fit(X_train_PCA, y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

In [47]:

```
y_pred = model.predict(X_test_PCA)
```

In [48]:

```
y_test.shape
```

```
(1389,)
```

Applying PCA on Test data

In [49]:

```
ForPred_data_test= data_test.drop(['ID'],axis=1)
```

In [50]:

```
data_test_PCA=obj.fit_transform(ForPred_data_test)
```

In [51]:

```
y_pred_data_test=model.predict(data_test_PCA)
```

In [52]:

```
y_pred_data_test
```

```
array([ 79.00339 ,  93.31714 ,  90.503395, ..., 107.86341 , 110.77077 ,
        90.692505], dtype=float32)
```

In [53]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [55]:

```
mean_squared_error(y_test,y_pred_data_test[:1389])
```

```
248.82796637257636
```