In [48]:
```
# import required libraries
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
```

In [49]:
```
#import data set
mer_train= pd.read_csv('train.csv')
```

In [50]:
```
# view the dataset first 10 rows
mer_train.head(10)
```

Out[50]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X
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	
5	18	92.93	t	b	e	c	d	g	h	s	...	0	0	1	0	0	0	0	
6	24	128.76	al	r	e	f	d	f	h	s	...	0	0	0	0	0	0	0	
7	25	91.91	o	l	as	f	d	f	j	a	...	0	0	0	0	0	0	0	
8	27	108.67	w	s	as	e	d	f	i	h	...	1	0	0	0	0	0	0	
9	30	126.99	j	b	aq	c	d	f	a	e	...	0	0	1	0	0	0	0	

10 rows × 378 columns

In [51]:
```
# view the info of data
mer_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 [52]:

```
# check for null values
mer_train.isnull().sum()
```

Out[52]:

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

In [53]:

```
# convert the 'y' values into array format
# seperate the 'y' column from the dataset
y_train = mer_train['y'].values
```

In [54]:

```
y_train
```

Out[54]:

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

In [55]:

```
# count of features with having X in the name
cols= [c for c in mer_train.columns if'X'in c]
print ('Number of features: {}'. format(len(cols)))
```

```
Number of features: 376
```

In [56]:

```
# counts of each columns
print ('feature Types:')
mer_train[cols].dtypes.value_counts()
```

```
feature Types:
```

Out[56]:

```
int64     368
object      8
dtype: int64
```

In [57]:

```
# values count of constant features, binary features and Categorical features
counts = [[],[],[]]
```

In [58]:

```
for c in cols:
    typ = mer_train[c].dtype
    unique = len(np.unique(mer_train[c]))
    if unique ==1:
        counts[0].append(c)
    elif unique == 2 and typ==np.int64:
        counts[1].append(c)
    else:
        counts[2].append(c)


print('Constant features: {} Binary features: {} Categorical features: {}\n'
     .format (*[len(c) for c in counts]))
print('Constant features:', counts[0])
print('Categorical features:',counts[2])
```

```
Constant features: 12 Binary features: 356 Categorical features: 8

Constant features: ['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
'X293', 'X297', 'X330', 'X347']
Categorical features: ['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

In [59]:

```
# load the test dataset
mer_test= pd.read_csv('test.csv')
```

In [60]:

```
# select the usable columns
usable_columns = list(set(mer_train.columns)- set(['ID','y']))
y_train = mer_train['y'].values
id_test = mer_test['ID'].values

x_train = mer_train[usable_columns]
x_test = mer_test[usable_columns]
```

In [61]:

```
# check for null values in the dataset
def check_null(df):
    if df.isnull().any().any():
        print('There are missing values in the dataset')
    else:
        print('There are no missing values in the dataset')


check_null(x_train)
check_null(x_test)
```

```
There are no missing values in the dataset
There are no missing values in the dataset
```

In [62]:

```
# if for any columns , the veriance is equal to zero, then you need to remove thos
e variables.
# apply lable encoder
for column in usable_columns:
    cardinality = len(np.unique(x_train[column]))
    if cardinality == 1:
        x_train.drop(column, axis= 1)
        x_test.drop(column,axis=1)
    if cardinality > 2:
        mapper = lambda x: sum ([ord(digit) for digit in x])
        x_train[column] = x_train[column].apply(mapper)
        x_test[column] = x_test[column].apply(mapper)
x_train.head()
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:9: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
```

Out[62]:

	X152	X196	X347	X70	X370	X8	X156	X263	X270	X248	...	X65	X68	X336	X204	X211
0	0	0	0	1	0	111	1	1	0	0	...	0	1	0	1	0
1	0	0	0	1	0	111	1	1	0	0	...	0	0	1	0	0
2	0	0	0	1	0	120	0	0	0	0	...	0	1	0	0	0
3	0	0	0	1	0	101	0	0	0	0	...	0	0	0	0	0
4	0	0	0	1	0	110	0	0	0	0	...	0	0	0	0	0

5 rows × 376 columns

In [63]:

```
# check whether data is converted into numerical values or not.
print('Feature types:')
x_train[cols].dtypes.value_counts()
```

```
Feature types:
```

Out[63]:

```
int64    376
dtype: int64
```

In [64]:

```
# perform Dimensionality reduction
# linear dimensionality reduction using Singular Value Decomposition of Data to pr
oject it to lower Dimensional Space.
n_comp = 12
pca = PCA(n_components = n_comp , random_state=420)
pca2_results_train = pca.fit_transform(x_train)
pca2_results_test = pca.fit_transform(x_test)
```

In [65]:

```
# import the xgboost
# train the data using xgboost
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

In [66]:

```
x_train, x_valid ,y_train, y_valid = train_test_split(
                            pca2_results_train,y_train ,test_size= 0.2,
                            random_state= 4242)
```

In [67]:

```
d_train = xgb.DMatrix(x_train,label= y_train)
d_valid = xgb.DMatrix(x_valid,label= y_valid)

d_test = xgb.DMatrix(pca2_results_test)
```

In [68]:

```
params = {}
params ['objective'] = 'reg:linear'
params ['eta'] = 0.02
params ['max_depth'] = 4
```

In [76]:

```
def xgb_r2_score(preds , dtrain):
    labels= dtrain.get_label()
    return 'r2', r2_score(labels, preds)
```

In [77]:

```
watchlist = [(d_train ,'train'), (d_valid,'valid')]
```

In [78]:

```
clf = xgb.train(params, d_train,
                1000, watchlist, early_stopping_rounds=50,
                feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

```
[12:27:54] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear is
now deprecated in favor of reg:squarederror.
[0]     train-rmse:99.14835     valid-rmse:98.26297     train-r2:-58.35295      v
alid-r2:-67.63754
Multiple eval metrics have been passed: 'valid-r2' will be used for early stoppin
g.

Will train until valid-r2 hasn't improved in 50 rounds.
[10]    train-rmse:81.27653     valid-rmse:80.36433     train-r2:-38.88428      v
alid-r2:-44.91014
[20]    train-rmse:66.71610     valid-rmse:65.77334     train-r2:-25.87403      v
alid-r2:-29.75260
[30]    train-rmse:54.86915     valid-rmse:53.89120     train-r2:-17.17724      v
alid-r2:-19.64513
[40]    train-rmse:45.24563     valid-rmse:44.22232     train-r2:-11.36018      v
alid-r2:-12.90160
[50]    train-rmse:37.44742     valid-rmse:36.37758     train-r2:-7.46672       v
alid-r2:-8.40697
[60]    train-rmse:31.15105     valid-rmse:30.01771     train-r2:-4.85891       v
alid-r2:-5.40526
[70]    train-rmse:26.08769     valid-rmse:24.90855     train-r2:-3.10906       v
alid-r2:-3.41041
[80]    train-rmse:22.04899     valid-rmse:20.82566     train-r2:-1.93528       v
alid-r2:-2.08304
[90]    train-rmse:18.84732     valid-rmse:17.59580     train-r2:-1.14472       v
alid-r2:-1.20090
[100]   train-rmse:16.33600     valid-rmse:15.07903     train-r2:-0.61125       v
alid-r2:-0.61633
[110]   train-rmse:14.40326     valid-rmse:13.14908     train-r2:-0.25254       v
alid-r2:-0.22906
[120]   train-rmse:12.93262     valid-rmse:11.69372     train-r2:-0.00982       v
alid-r2:0.02795
[130]   train-rmse:11.81574     valid-rmse:10.61241     train-r2:0.15707        v
alid-r2:0.19941
[140]   train-rmse:10.98584     valid-rmse:9.84577      train-r2:0.27132        v
alid-r2:0.31090
[150]   train-rmse:10.37818     valid-rmse:9.31608      train-r2:0.34970        v
alid-r2:0.38305
[160]   train-rmse:9.92761      valid-rmse:8.95044      train-r2:0.40494        v
alid-r2:0.43053
[170]   train-rmse:9.59297      valid-rmse:8.71236      train-r2:0.44438        v
alid-r2:0.46042
[180]   train-rmse:9.34889      valid-rmse:8.54634      train-r2:0.47229        v
alid-r2:0.48079
[190]   train-rmse:9.16216      valid-rmse:8.44332      train-r2:0.49316        v
alid-r2:0.49323
[200]   train-rmse:9.02020      valid-rmse:8.37881      train-r2:0.50875        v
alid-r2:0.50095
[210]   train-rmse:8.91339      valid-rmse:8.34352      train-r2:0.52031        v
alid-r2:0.50514
[220]   train-rmse:8.82193      valid-rmse:8.31729      train-r2:0.53011        v
alid-r2:0.50825
[230]   train-rmse:8.76377      valid-rmse:8.30382      train-r2:0.53628        v
alid-r2:0.50984
[240]   train-rmse:8.70541      valid-rmse:8.29665      train-r2:0.54244        v
alid-r2:0.51069
[250]   train-rmse:8.66769      valid-rmse:8.29059      train-r2:0.54639        v
alid-r2:0.51140
[260]   train-rmse:8.62392      valid-rmse:8.28817      train-r2:0.55097        v
```

```
alid-r2:0.51169
[270]   train-rmse:8.59049      valid-rmse:8.28617      train-r2:0.55444        v
alid-r2:0.51192
[280]   train-rmse:8.55208      valid-rmse:8.28102      train-r2:0.55841        v
alid-r2:0.51253
[290]   train-rmse:8.52721      valid-rmse:8.28241      train-r2:0.56098        v
alid-r2:0.51236
[300]   train-rmse:8.50348      valid-rmse:8.28322      train-r2:0.56342        v
alid-r2:0.51227
[310]   train-rmse:8.47840      valid-rmse:8.27920      train-r2:0.56599        v
alid-r2:0.51274
[320]   train-rmse:8.45433      valid-rmse:8.28000      train-r2:0.56845        v
alid-r2:0.51265
[330]   train-rmse:8.43036      valid-rmse:8.27771      train-r2:0.57090        v
alid-r2:0.51292
[340]   train-rmse:8.40901      valid-rmse:8.27708      train-r2:0.57307        v
alid-r2:0.51299
[350]   train-rmse:8.38952      valid-rmse:8.27791      train-r2:0.57504        v
alid-r2:0.51289
[360]   train-rmse:8.36299      valid-rmse:8.27466      train-r2:0.57773        v
alid-r2:0.51328
[370]   train-rmse:8.33332      valid-rmse:8.27140      train-r2:0.58072        v
alid-r2:0.51366
[380]   train-rmse:8.30903      valid-rmse:8.27299      train-r2:0.58316        v
alid-r2:0.51347
[390]   train-rmse:8.28632      valid-rmse:8.27065      train-r2:0.58543        v
alid-r2:0.51375
[400]   train-rmse:8.25956      valid-rmse:8.26821      train-r2:0.58811        v
alid-r2:0.51404
[410]   train-rmse:8.23725      valid-rmse:8.26749      train-r2:0.59033        v
alid-r2:0.51412
[420]   train-rmse:8.20771      valid-rmse:8.26607      train-r2:0.59326        v
alid-r2:0.51429
[430]   train-rmse:8.18372      valid-rmse:8.26409      train-r2:0.59564        v
alid-r2:0.51452
[440]   train-rmse:8.16154      valid-rmse:8.26225      train-r2:0.59783        v
alid-r2:0.51474
[450]   train-rmse:8.13499      valid-rmse:8.26102      train-r2:0.60044        v
alid-r2:0.51488
[460]   train-rmse:8.11711      valid-rmse:8.25870      train-r2:0.60219        v
alid-r2:0.51515
[470]   train-rmse:8.08796      valid-rmse:8.25732      train-r2:0.60504        v
alid-r2:0.51531
[480]   train-rmse:8.05856      valid-rmse:8.26027      train-r2:0.60791        v
alid-r2:0.51497
[490]   train-rmse:8.03767      valid-rmse:8.25967      train-r2:0.60994        v
alid-r2:0.51504
[500]   train-rmse:8.01613      valid-rmse:8.25948      train-r2:0.61203        v
alid-r2:0.51506
[510]   train-rmse:7.99323      valid-rmse:8.25895      train-r2:0.61424        v
alid-r2:0.51512
[520]   train-rmse:7.97607      valid-rmse:8.25734      train-r2:0.61589        v
alid-r2:0.51531
Stopping. Best iteration:
[472]   train-rmse:8.08025      valid-rmse:8.25677      train-r2:0.60580        v
alid-r2:0.51538
```

In [79]:

```python
# predict the test_df values using xgboost
p_test= clf.predict(d_test)

sub = pd.DataFrame()
sub['ID'] = id_test
sub['y'] = p_test
sub.to_csv('xgb.csv', index = False)

sub.head()
```

Out[79]:

	ID	y
0	1	83.886436
1	2	104.960732
2	3	83.411240
3	4	77.086838
4	5	97.411743