In [2]:

```
import numpy as np
import pandas as pd
```

In [3]:

```
train_df = pd.read_csv("train.csv")
```

In [4]:

```
train_df.head()
```

Out[4]:

	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 [5]:

```
y = train_df['y'].values
```

In [6]:

```
y
```

Out[6]:

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

In [7]:

```
cols = [c for c in train_df.columns if 'X' in c]
print('Number of features: {}'.format(len(cols)))
```

```
Number of features: 376
```

In [8]:

```
train_df[cols].dtypes.value_counts()
```

Out[8]:

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

In [9]:

```
test_df = pd.read_csv("test.csv")
```

In [10]:

```
test_df.info
```

Out[10]:

```
<bound method DataFrame.info of          ID  X0 X1  X2 X3 X4 X5 X6 X8  X10  ...  X375  X37
6  X377  X378  X379  \
0         1  az  v   n  f  d  t  a  w    0  ...     0  0.0   0.0   0.0   1.0   0.0
```

```
0         1   az   v   n   f   d   t   a   w    0  ...   0.0   0.0   0.0   1.0   0.0
1         2    t   b  ai   a   d   b   g   y    0  ...   0.0   0.0   1.0   0.0   0.0
2         3   az   v  as   f   d   a   j   j    0  ...   0.0   0.0   0.0   1.0   0.0
3         4   az   l   n   f   d   z   l   n    0  ...   0.0   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   0.0   0.0
...     ...  ..  ..  ..  ..  ..  ..  ..  ..  ...  ...   ...   ...   ...   ...   ...
2758   5518   ay   r  as   f   d   p   g   f    0  ...   0.0   0.0   0.0   0.0   0.0
2759   5521    o   l  ae   f   d   p   d   j    0  ...   0.0   0.0   0.0   0.0   0.0
2760   5522   ay   i  as   a   d   p   j   j    0  ...   0.0   0.0   1.0   0.0   0.0
2761   5527    z   b   a   a   d   p   g   a    0  ...   0.0   1.0   0.0   0.0   0.0
2762   5528    t   r   d   f   d   p   i   t    0  ...   NaN   NaN   NaN   NaN   NaN

      X380  X382  X383  X384  X385
0      0.0   0.0   0.0   0.0   0.0
1      0.0   0.0   0.0   0.0   0.0
2      0.0   0.0   0.0   0.0   0.0
3      0.0   0.0   0.0   0.0   0.0
4      0.0   0.0   0.0   0.0   0.0
...    ...   ...   ...   ...   ...
2758   0.0   0.0   0.0   0.0   0.0
2759   0.0   0.0   0.0   0.0   0.0
2760   0.0   0.0   0.0   0.0   0.0
2761   0.0   0.0   0.0   0.0   0.0
2762   NaN   NaN   NaN   NaN   NaN

[2763 rows x 377 columns]>
```

In [11]:

```
test_df.head()
```

Out[11]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	1	az	v	n	f	d	t	a	w	0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1	2	t	b	ai	a	d	b	g	y	0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	3	az	v	as	f	d	a	j	j	0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	4	az	l	n	f	d	z	l	n	0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
4	5	w	s	as	c	d	y	i	m	0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 377 columns

In [12]:

```
usable_columns = list(set(train_df.columns) - set(['ID', 'y']))
y_train = train_df['y'].values
id_test = test_df['ID'].values
x_train = train_df[usable_columns]
x_test = test_df[usable_columns]
```

In [13]:

```
# check for any missing values
def missing_values(df):
    if df.isnull().any().any():
        print("There are missing values")
    else:
        print("There are no missing values")
missing_values(x_train)
missing_values(x_test)
```

```
There are no missing values
There are missing values
```

In [14]:

```
X_test=x_test.dropna(axis=0, inplace=False)
```

In [15]:

```
X_test.info
```

Out[15]:

```
<bound method DataFrame.info of       X268  X44  X287  X280  X90  X277  X151  X214  X326
X71  ...  X132  \
0      0.0    0   0.0   0.0  0.0   0.0   0.0   0.0   0.0  0.0  ...   1.0
1      0.0    0   0.0   0.0  0.0   0.0   0.0   0.0   0.0  0.0  ...   0.0
2      0.0    0   0.0   0.0  0.0   0.0   0.0   0.0   0.0  0.0  ...   1.0
3      0.0    0   0.0   0.0  0.0   0.0   0.0   0.0   0.0  0.0  ...   1.0
4      0.0    0   0.0   0.0  0.0   0.0   1.0   0.0   0.0  0.0  ...   1.0
...    ...  ...   ...   ...  ...   ...   ...   ...   ...  ...  ...   ...
2757   0.0    0   0.0   0.0  0.0   0.0   0.0   0.0   0.0  0.0  ...   0.0
2758   0.0    0   0.0   0.0  0.0   0.0   0.0   0.0   0.0  0.0  ...   1.0
2759   0.0    0   0.0   0.0  0.0   0.0   0.0   0.0   0.0  0.0  ...   1.0
2760   0.0    0   1.0   0.0  0.0   0.0   0.0   0.0   0.0  0.0  ...   0.0
2761   0.0    0   0.0   0.0  0.0   0.0   0.0   0.0   0.0  0.0  ...   0.0

      X102  X244  X292  X137  X295  X301  X187  X300  X163
0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1      0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   1.0
2      0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0
3      0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
4      0.0   0.0   0.0   1.0   0.0   0.0   1.0   1.0   1.0
...    ...   ...   ...   ...   ...   ...   ...   ...   ...
2757   0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0
2758   0.0   0.0   0.0   1.0   0.0   0.0   0.0   1.0   0.0
2759   0.0   0.0   0.0   1.0   0.0   0.0   1.0   0.0   0.0
2760   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2761   0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0

[2762 rows x 376 columns]>
```

In [16]:

```
#
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) # drop column with 1 value
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
```

Out[16]:

```
<bound method NDFrame.head of       X268  X44  X287  X280  X90  X277  X151  X214  X326  X
71  ...  X132  \
0         0    0     0     0    0     0     0     0     0    0  ...     0
1         0    0     0     0    0     0     0     0     0    0  ...     1
2         0    0     0     0    0     0     0     0     0    1  ...     1
3         0    0     0     0    0     0     0     0     0    1  ...     1
4         0    0     0     0    0     0     0     0     0    0  ...     1
...     ...  ...   ...   ...  ...   ...   ...   ...   ...  ...  ...   ...
4204      0    0     0     0    0     0     0     0     0    0  ...     1
4205      0    0     0     0    0     0     0     1     0    0  ...     1
4206      0    0     0     0    0     0     0     0     0    1  ...     1
4207      0    0     0     0    0     0     0     0     0    0  ...     1
4208      0    0     0     0    0     0     0     0     0    0  ...     1

      X102  X244  X292  X137  X295  X301  X187  X300  X163
0        0     0     0     1     0     0     1     0     0
1        0     0     0     0     0     0     1     0     0
2        0     1     0     1     0     0     0     0     1
3        0     1     0     0     0     0     0     0     0
4        0     0     0     0     0     0     0     0     0
...    ...   ...   ...   ...   ...   ...   ...   ...   ...
4204     0     0     0     1     0     0     1     0     1
```

```
4205     1     0     0     1     0     0     0     0     0
4206     0     1     0     0     0     0     1     0     0
4207     0     0     0     0     0     1     0     0     0
4208     0     0     0     1     0     0     0     0     0

[4209 rows x 376 columns]>
```

In [61]:

```
# apply label encoder
def encoder(x_train):
    for col in x_train.columns:
        if x_train.dtypes[col] == "object":
            le = preprocessing.LabelEncoder()
        le.fit(x_train[col])
        x_train[col] = le.transform(x_train[col])
    return x_train
```

In [60]:

```
x_train[cols].dtypes.value_counts()
```

Out[60]:

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

In [62]:

```
# Method 1- Dimensionality reduction using Missing value ratio
# checking the percentage of missing values in each variable
x_train.isnull().sum()/len(x_train)*100
```

Out[62]:

```
X268    0.0
X44     0.0
X287    0.0
X280    0.0
X90     0.0
       ...
X295    0.0
X301    0.0
X187    0.0
X300    0.0
X163    0.0
Length: 376, dtype: float64
```

In [63]:

```
# saving missing values in a variable
a = x_train.isnull().sum()/len(x_train)*100
# saving column names in a variable
variables = x_train.columns
variable = [ ]
for i in range(0,12):
    if a[i]<=5:   #setting the threshold as 5%
        variable.append(variables[i])
```

In [ ]:

```
# Method 2 - PCA for dimensinality reduction
from sklearn.decomposition import PCA
pca= PCA(n_components=2)
pca.fit(x_train)
pca_score = pca.transform(x_train)
df_pca= pd.DataFrame(pca_score)
df_score.index= x_train.index
df_pca.columns = ['PC1','PC2']
pca.explained_variance_ratio
```

In [ ]:

```
# Train using XGBoost
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split

x_train, x_valid, y_train, y_valid = train_test_split(
        x_train,
        y_train, test_size=0.3,
        random_state=42)

df_train = xgb.DMatrix(x_train, label=y_train)
df_valid = xgb.DMatrix(x_valid, label=y_valid)
#df_test = xgb.DMatrix(x_test)
df_test = xgb.DMatrix(X_test)

params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.03
params['max_depth'] = 4

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

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

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

In [ ]:

```
# Prediction done on test_df 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()
```