In [1]:
```
#Import the required libraries
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
```

In [2]:
```
#Read the data from train.csv
df_train = pd.read_csv('train.csv')

#Let us understand the data
print(df_train.head())
print(df_train.shape)
```

```
   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]
(4209, 378)
```

In [3]:
```
#Let us collect the y values into an array
#Seperate the y value from the data as we will use this value to learn as prediction output
y_train = df_train['y'].values
y_train
```

Out[3]:
```
array([130.81,  88.53,  76.26, ..., 109.22,  87.48, 110.85])
```

In [4]:
```
#Let us understand the data types we have

#Iterate through all the columns which has X in the name of the columns
cols = [c for c in df_train.columns if 'X' in c]
print('Number of features: {}'.format(len(cols)))

print('Feature types')
df_train[cols].dtypes.value_counts()
```

```
Number of features: 376
Feature types
```

Out[4]:
```
int64     368
object      8
dtype: int64
```

In [5]:
```
#Let us count the data in each column

counts = [[],[],[]]

for c in cols:
    typ = df_train[c].dtype
    uniq = len(np.unique(df_train[c]))
    if uniq == 1:
        counts[0].append(c)
    elif uniq == 2:
        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 [6]:
```
#Read the Test data
df_test = pd.read_csv('test.csv')

#Remove columns 'ID' and 'y' as they are not used for learning
usable_columns = list(set(df_train.columns) - set(['ID', 'y']))
y_train = df_train['y'].values
id_test = df_test['ID'].values

x_train = df_train[usable_columns]
x_test = df_test[usable_columns]
```

In [7]:
```
#Check for Null and unique values for train and test data set
def check_missing_values(df):
    if df.isnull().any().any():
        print('There are missing values in the DataFrame')
    else:
        print('There are no missing values in the DataFrame')
check_missing_values(df_train)
check_missing_values(df_test)
```

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

In [8]:
```
#If for any of the columns variance is zero then we need to remove those variables
#Apply LabelEncoder
for column in usable_columns:
    cardinality = len(np.unique(x_train[column]))
    if cardinality == 1:
        #Column with only one value is useless so we drop it
        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()
```

Out[8]:

	X210	X116	X94	X76	X368	X141	X56	X346	X98	X44	...	X102	X35	X17	X82	X385	X364	X80	X286	X47	X106
0	0	1	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	...	0	1	0	0	0	0	1	0	0	0
2	0	0	0	1	0	0	0	0	1	0	...	0	1	1	0	0	0	1	1	0	0
3	0	0	0	1	0	0	0	0	1	0	...	0	1	0	0	0	0	1	1	0	0
4	0	0	0	1	0	0	0	0	1	0	...	0	1	0	0	0	0	1	1	0	0

5 rows × 376 columns

In [9]:
```
#All Data is now changed to numericals
print('Feature types', x_train[cols].dtypes.value_counts())
```

```
Feature types int64    376
dtype: int64
```

In [10]:
```
#Performing dimensionality reduction using pricipal component analysis
n_comp = 12
pca = PCA(n_components=n_comp, random_state=420)
pca2_results_train = pca.fit_transform(x_train)
pca2_results_test = pca.transform(x_test)
```

In [11]:
```
#Training using xgboost

import sys
!{sys.executable} -m pip install xgboost
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(pca2_results_train, y_train, test_size=0.2, random_state=4242)
```

```
Requirement already satisfied: xgboost in /Users/glikithvinayaka/opt/anaconda3/lib/python3.9/site-packages (1.6.1)
Requirement already satisfied: numpy in /Users/glikithvinayaka/opt/anaconda3/lib/python3.9/site-packages (from xgbo
ost) (1.21.5)
Requirement already satisfied: scipy in /Users/glikithvinayaka/opt/anaconda3/lib/python3.9/site-packages (from xgbo
ost) (1.7.3)
```

In [12]:
```
import xgboost as xgb

d_train = xgb.DMatrix(x_train, label = y_train)
d_valid = xgb.DMatrix(x_valid, label = y_valid)
#d_test = xgb.DMatrix(x_test)
d_test = xgb.DMatrix(pca2_results_test)
```

In [13]:
```
params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.02
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 [14]:
```
#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[14]:

	ID	y
0	1	82.688263
1	2	97.398170
2	3	83.233788
3	4	77.283653
4	5	112.651413