In [1]:

```
# import modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
#load datasets
path='D://SimpliLearn Projects//Machine Learning//Machine-Learning--Projects-master//Projec
train = pd.read_csv(path+'train.csv')
test = pd.read_csv(path+'test.csv')
```

In [3]:

```
# check size of datasets
print(train.shape)
print(test.shape)
```

```
(4209, 378)
(4209, 377)
```

In [4]:

```
train.describe()
```

Out[4]:

	ID	y	X10	X11	X12	X13	X14
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	4209.000000	4209.000000
mean	4205.960798	100.669318	0.013305	0.0	0.075077	0.057971	0.428130
std	2437.608688	12.679381	0.114590	0.0	0.263547	0.233716	0.494867
min	0.000000	72.110000	0.000000	0.0	0.000000	0.000000	0.000000
25%	2095.000000	90.820000	0.000000	0.0	0.000000	0.000000	0.000000
50%	4220.000000	99.150000	0.000000	0.0	0.000000	0.000000	0.000000
75%	6314.000000	109.010000	0.000000	0.0	0.000000	0.000000	1.000000
max	8417.000000	265.320000	1.000000	0.0	1.000000	1.000000	1.000000

8 rows × 370 columns

In [5]:

```
test.describe()
```

Out[5]:

	ID	X10	X11	X12	X13	X14	
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000
mean	4211.039202	0.019007	0.000238	0.074364	0.061060	0.427893	0.000
std	2423.078926	0.136565	0.015414	0.262394	0.239468	0.494832	0.026
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	2115.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	4202.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
75%	6310.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000
max	8416.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000

8 rows × 369 columns

In [6]:

```
# check null values in train dataset
#train.isnull().sum()
print ("Top Traing Columns having missing values:")
missing_df = train.isnull().sum().to_frame()
missing_df = missing_df.sort_values(0, ascending = False)
missing_df.head()
```

```
Top Traing Columns having missing values:
```

Out[6]:

	0
ID	0
X254	0
X263	0
X262	0
X261	0

In [7]:

```
# check null values in test dataset
#test.isnull().sum()
print ("Top Testing Columns having missing values:")
missing_df = test.isnull().sum().to_frame()
missing_df = missing_df.sort_values(0, ascending = False)
missing_df.head()
```

```
Top Testing Columns having missing values:
```

Out[7]:

	0
ID	0
X255	0
X264	0
X263	0
X262	0

In [8]:

```
# unique value
train['y'].unique()
```

Out[8]:

```
array([130.81,  88.53,  76.26, ...,  85.71, 108.77,  87.48])
```

In [9]:

```
# Apply label encoder
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data = train.apply(le.fit_transform)
print(data.shape)
```

```
(4209, 378)
```

In [10]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data_test = test.apply(le.fit_transform)
print(data_test.shape)
```

```
(4209, 377)
```

In [11]:

```
data.head()
```

Out[11]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X38
0	0	2466	32	23	17	0	3	24	9	14	...	0	0	1	0	0	0	(
1	1	366	32	21	19	4	3	28	11	14	...	1	0	0	0	0	0	(
2	2	69	20	24	34	2	3	27	9	23	...	0	0	0	0	0	0	
3	3	133	20	21	34	5	3	27	11	4	...	0	0	0	0	0	0	(
4	4	106	20	23	34	5	3	12	3	13	...	0	0	0	0	0	0	(

5 rows × 378 columns

In [12]:

```
data_test.head()
```

Out[12]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382
0	0	21	23	34	5	3	26	0	22	0	...	0	0	0	1	0	0	0
1	1	42	3	8	0	3	9	6	24	0	...	0	0	1	0	0	0	0
2	2	21	23	17	5	3	0	9	9	0	...	0	0	0	1	0	0	0
3	3	21	13	34	5	3	31	11	13	0	...	0	0	0	1	0	0	0
4	4	45	20	17	2	3	30	8	12	0	...	1	0	0	0	0	0	0

5 rows × 377 columns

In [13]:

```
X = data.drop('y', axis=1)
y = data['y']
from sklearn.model_selection import train_test_split
X_train, X_test , y_train , y_test = train_test_split (X, y, test_size =0.2, random_state =
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3367, 377)
(842, 377)
(3367,)
(842,)
```

In [ ]:

```
# Perform dimensionality reduction.
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as lda
sklearn_lda = lda()
sklearn_lda.fit(X_train,y_train)
X_train = sklearn_lda.transform(X_train)
#X_test = sklearn_lda.transform(X_test)
print("Number of components in transformed shape {}".format(X_train.shape[1]))
```

In [ ]:

```
# Perform dimensionality reduction.
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as lda
sklearn_lda = lda()
sklearn_lda.fit(X_test,y_test)
X_test = sklearn_lda.transform(X_test)
print("Number of components in transformed shape {}".format(X_test.shape[1]))
```

In [ ]:

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

In [ ]:

```
# generate feature sets (X)
X_train = data_test.values[0:3367]
X_test = data_test.values[3367:]
```

In [ ]:

```
print(X_train.shape)
print(X_test.shape)
```

In [14]:

```
# transform data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train_scale = scaler.fit_transform(X_train)
X_test_scale = scaler.fit_transform(X_test)
```

```
C:\Users\Digesh\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:33
4: DataConversionWarning: Data with input dtype int32, int64 were all conver
ted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
C:\Users\Digesh\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:33
4: DataConversionWarning: Data with input dtype int32, int64 were all conver
ted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

In [15]:

```
print(X_train_scale.shape)
print(X_test_scale.shape)
```

```
(3367, 377)
(842, 377)
```

In [16]:

```
# split training feature and target sets into training and validation subsets
from sklearn.model_selection import train_test_split

X_train_sub, X_validation_sub, y_train_sub, y_validation_sub = train_test_split(X_train_sca
```

In [17]:

```
print(X_train_sub.shape)
print(X_validation_sub.shape)
print(y_train_sub.shape)
print(y_validation_sub.shape)
```

```
(2525, 377)
(842, 377)
(2525,)
(842,)
```

In [18]:

```
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier(n_estimators=10, learning_rate = 0.05, max_features=2, max_

gb.fit(X_train_sub, y_train_sub)
print("Accuracy score (training): {0:.3f}".format(gb.score(X_train_sub, y_train_sub)))
print("Accuracy score (validation): {0:.3f}".format(gb.score(X_validation_sub, y_validatior
```

```
Accuracy score (training): 0.064
Accuracy score (validation): 0.001
```

In [ ]:

```

```