In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df_train=pd.read_csv("P:\\Data Science with ML\\Datasets\\Machine-Learning--Projects
```

In [3]:
```
df_test=pd.read_csv("P:\\Data Science with ML\\Datasets\\Machine-Learning--Projects\
```

In [4]:
```
df_train.head()
```

Out[4]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382
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 rows × 378 columns

In [5]:
```
df_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 [6]:
```
df_train.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
ID	4209.0	4205.960798	2437.608688	0.00	2095.00	4220.00	6314.00	8417.00
y	4209.0	100.669318	12.679381	72.11	90.82	99.15	109.01	265.32
X10	4209.0	0.013305	0.114590	0.00	0.00	0.00	0.00	1.00
X11	4209.0	0.000000	0.000000	0.00	0.00	0.00	0.00	0.00
X12	4209.0	0.075077	0.263547	0.00	0.00	0.00	0.00	1.00
...	...	...	...	...	...	...	...	...
X380	4209.0	0.008078	0.089524	0.00	0.00	0.00	0.00	1.00
X382	4209.0	0.007603	0.086872	0.00	0.00	0.00	0.00	1.00
X383	4209.0	0.001663	0.040752	0.00	0.00	0.00	0.00	1.00

	count	mean	std	min	25%	50%	75%	max
X384	4209.0	0.000475	0.021796	0.00	0.00	0.00	0.00	1.00
X385	4209.0	0.001426	0.037734	0.00	0.00	0.00	0.00	1.00

370 rows × 8 columns

In [7]:
```
df_train.isnull().sum()
```

Out[7]:
```
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 [9]:
```
#check zero Variance in train data
#def zero_var_colums(df):
    #var_df=pd.DataFrame(df.var(),columns=['Variance'])
    #return(list(var_df[var_df.Variance==0].index))
df_train.var().sort_values().head(15)
```

Out[9]:
```
X289    0.000000
X330    0.000000
X268    0.000000
X347    0.000000
X107    0.000000
X235    0.000000
X233    0.000000
X290    0.000000
X11     0.000000
X297    0.000000
X293    0.000000
X93     0.000000
X257    0.000238
X207    0.000238
X280    0.000238
dtype: float64
```

In [10]:
```
df_train=df_train.drop(['X289','X330','X268','X347','X107','X235','X233','X290','X11
```

In [11]:
```
# dropping those columns which zero variance
df_train=df_train.drop(['X11','X93','X107','X233','X235','X268','X289','X290','X293'
```

In [11]:
```
df_train.shape
```

Out[11]:
```
(4209, 363)
```

In [12]:
```
df_train.head()
```

Out[12]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382
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 rows × 363 columns

In [13]:
```
df_train['X0'].value_counts()
```

Out[13]:
```
z     360
ak    349
y     324
ay    313
t     306
x     300
o     269
f     227
n     195
w     182
j     181
az    175
aj    151
s     106
ap    103
h      75
d      73
al     67
v      36
af     35
ai     34
m      34
e      32
ba     27
at     25
a      21
ax     19
aq     18
am     18
i      18
u      17
aw     16
l      16
ad     14
au     11
k      11
b      11
as     10
r      10
bc      6
ao      4
c       3
q       2
aa      2
g       1
ac      1
ab      1
Name: X0, dtype: int64
```

In [14]:
```
def transform_X0(df):
    x0_grp=df.groupby('X0').aggregate(func='count')['ID'].reset_index()
    df['X0'].replace(to_replace=x0_grp[x0_grp['ID']<100]['X0'].values
                              ,value='OT'
                              ,inplace=True)
```

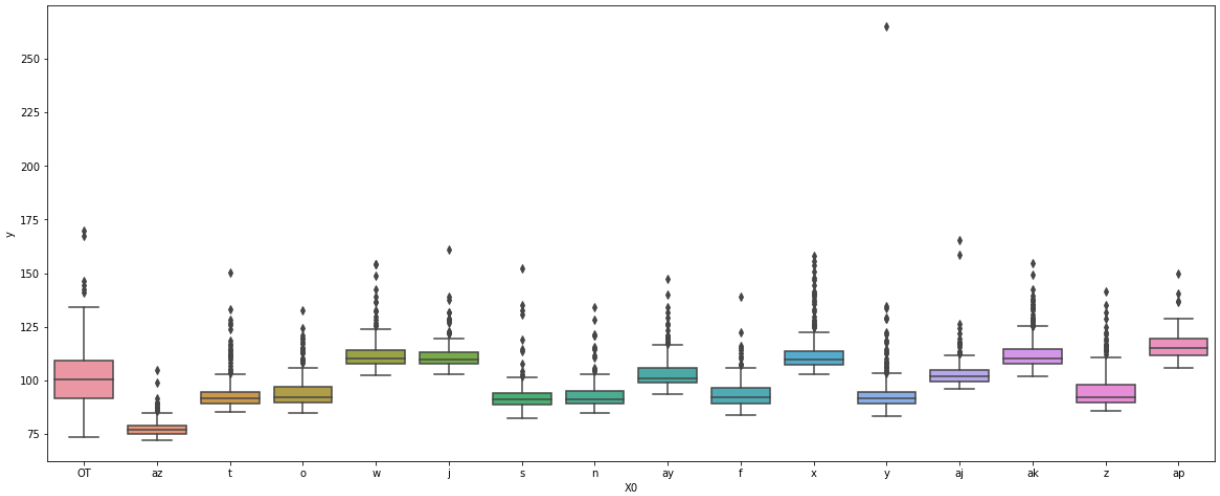
In [15]:
```
transform_X0(df_train)
```

In [16]:
```
df_train['X0'].value_counts()
```

Out[16]:
```
OT    668
z     360
ak    349
y     324
ay    313
t     306
x     300
o     269
f     227
n     195
w     182
j     181
az    175
aj    151
s     106
ap    103
Name: X0, dtype: int64
```

In [17]:
```
plt.figure(figsize=(20,8))
sns.boxplot(x=df_train['X0'],y=df_train['y'])
```

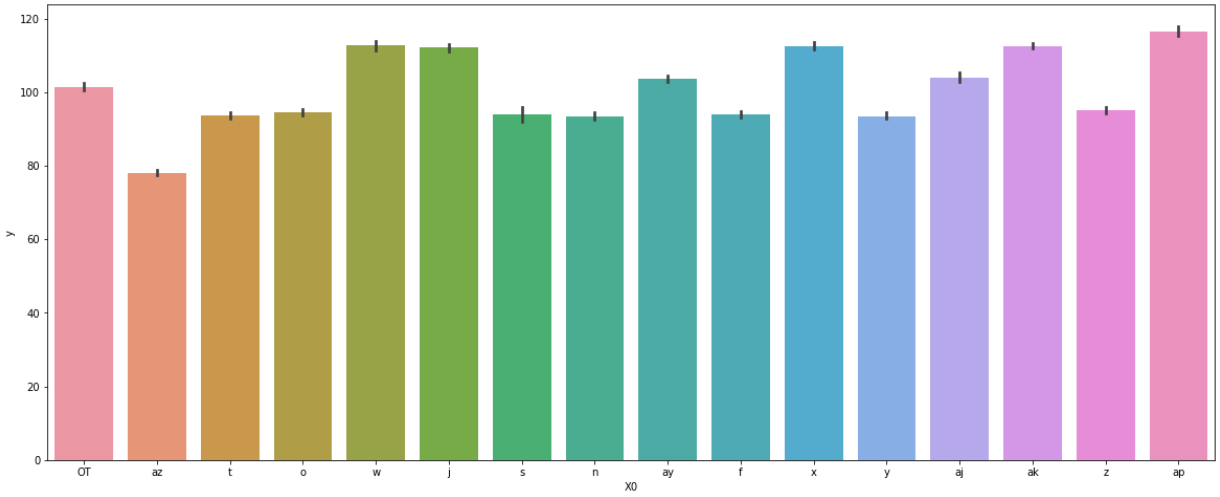
Out[17]:
```
<AxesSubplot:xlabel='X0', ylabel='y'>
```



In [18]:
```
df_train.drop(labels=((df_train[df_train['X0']=='y']['y'].sort_values(ascending=Fals
```

In [19]:
```
plt.figure(figsize=(20,8))
sns.barplot(x=df_train['X0'],y=df_train['y'])
```

Out[19]:
```
<AxesSubplot:xlabel='X0', ylabel='y'>
```



In [20]:
```
df_train['X1'].value_counts()
```

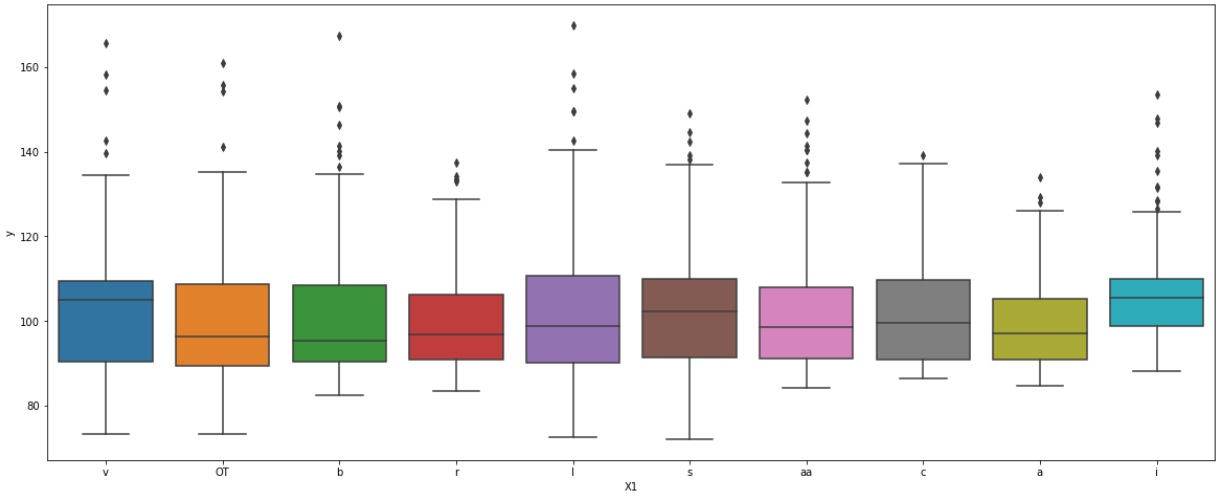
Out[20]:
```
aa    833
s     598
b     592
l     590
v     408
r     250
i     203
a     143
c     121
o      82
w      52
z      46
u      37
e      33
m      32
t      31
h      29
f      23
y      23
j      22
n      19
k      17
p       9
g       6
q       3
d       3
ab      3
Name: X1, dtype: int64
```

In [21]:
```
def transform_X1(df):
    x1_grp=df.groupby('X1').aggregate(func='count')['ID'].reset_index()
    df['X1'].replace(to_replace=x1_grp[x1_grp['ID']<100]['X1'].values
                                ,value='OT'
                                ,inplace=True)
```

In [22]:
```
transform_X1(df_train)
```

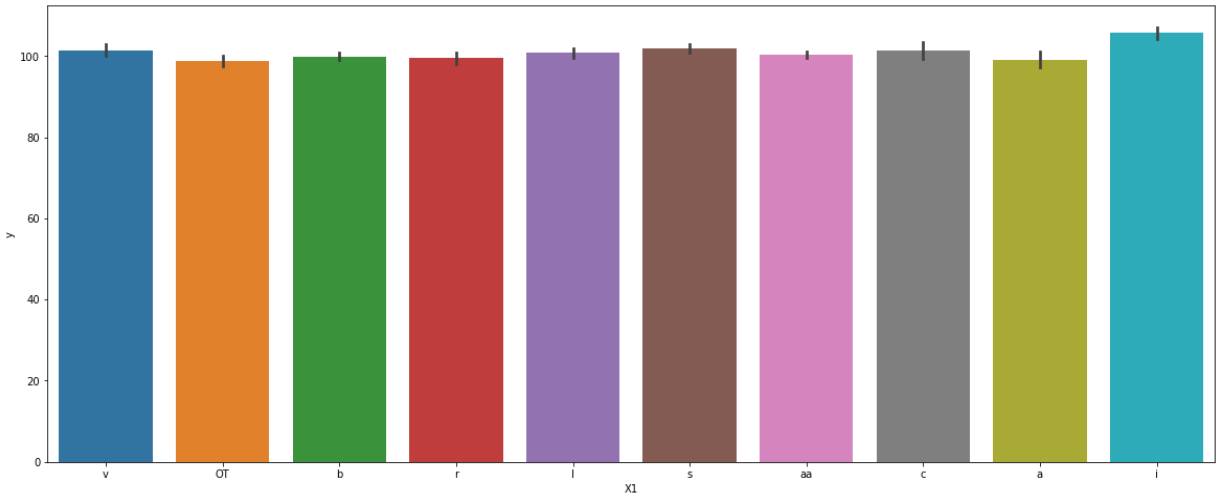
In [23]:
```
plt.figure(figsize=(20,8))
sns.boxplot(x=df_train['X1'],y=df_train['y'])
```

Out[23]:
```
<AxesSubplot:xlabel='X1', ylabel='y'>
```



In [24]:
```
plt.figure(figsize=(20,8))
sns.barplot(x=df_train['X1'],y=df_train['y'])
```

Out[24]: <AxesSubplot:xlabel='X1', ylabel='y'>

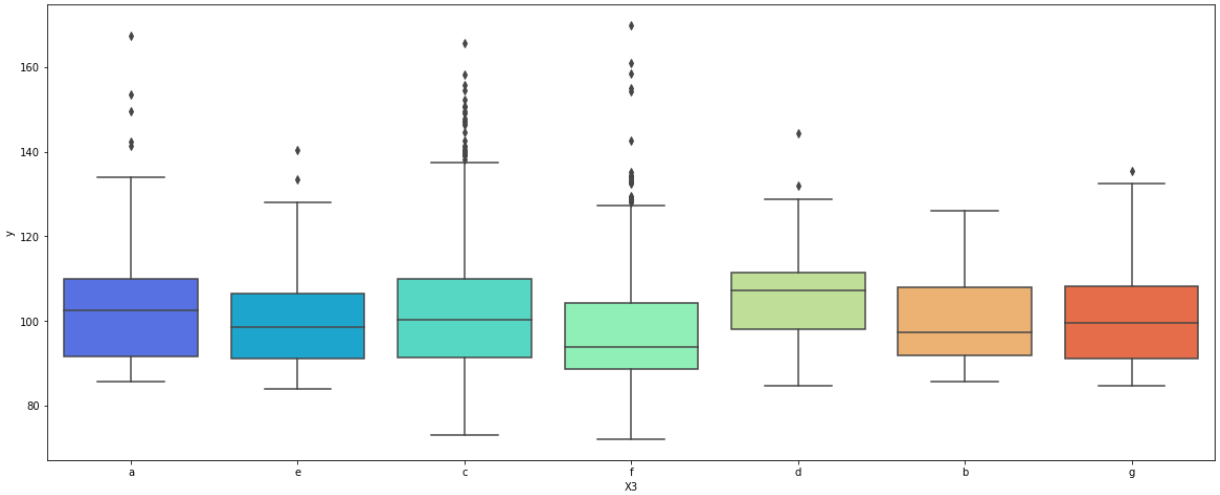


In [25]:
```
df_train['X3'].value_counts()
```

Out[25]:
```
c    1942
f    1075
a     440
d     290
g     241
e     163
b      57
Name: X3, dtype: int64
```

In [26]:
```
plt.figure(figsize=(20,8))
sns.boxplot(x=df_train['X3'],y=df_train['y'],palette='rainbow')
```

Out[26]: <AxesSubplot:xlabel='X3', ylabel='y'>



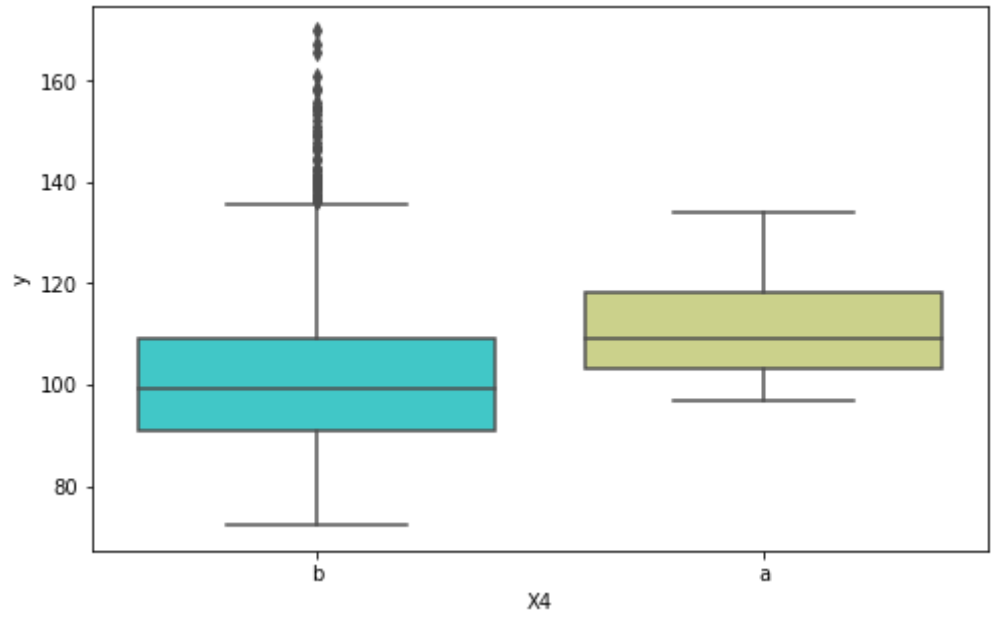
In [27]:
```
df_train['X4'].value_counts()
```

Out[27]:
```
d    4204
a       2
b       1
c       1
Name: X4, dtype: int64
```

In [28]:
```
df_train['X4']=df_train['X4'].replace({'b':'a','c':'a','d':'b'})
```

In [29]:
```
plt.figure(figsize=(8,5))
sns.boxplot(x=df_train['X4'],y=df_train['y'],palette='rainbow')
```

Out[29]:
```
<AxesSubplot:xlabel='X4', ylabel='y'>
```



In [30]:
```
df_train['X5'].value_counts()
```

Out[30]:
```
v     231
w     231
q     220
r     215
s     214
d     214
n     212
```

```
p     208
m     208
i     207
ae    205
ag    203
ac    200
ab    197
l     195
af    188
ad    185
k     177
c     131
j     125
aa    112
ah     97
o      20
f       7
x       2
g       1
y       1
u       1
h       1
Name: X5, dtype: int64
```

In [31]:
```
def transform_X5(df):
    x5_grp=df.groupby('X5').aggregate(func='count')['ID'].reset_index()
    df['X5'].replace(to_replace=x5_grp[x5_grp['ID']<100]['X5'].values
                             ,value='OT'
                             ,inplace=True)
```

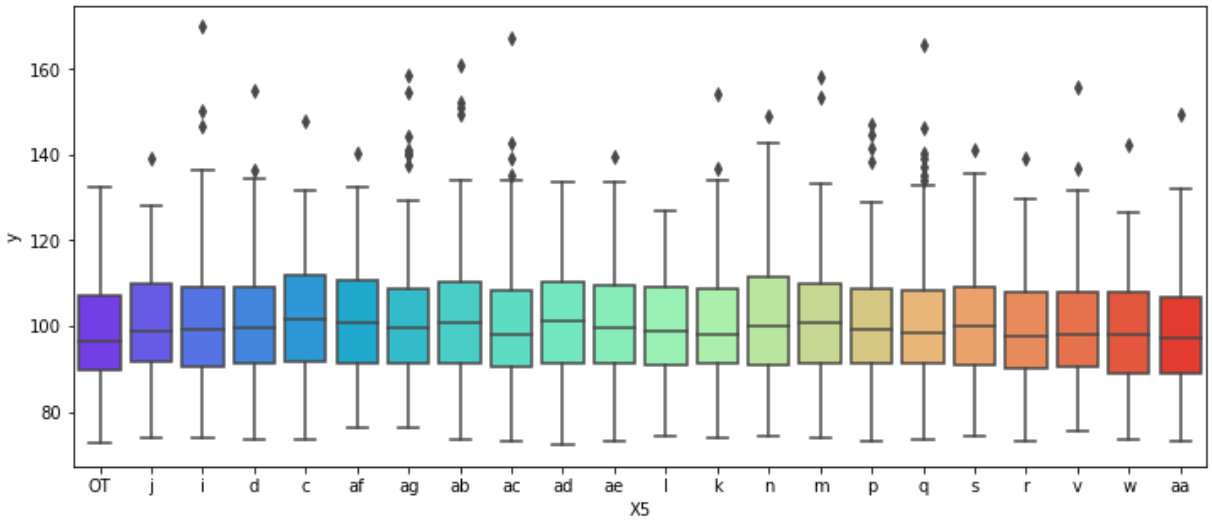
In [32]:
```
transform_X5(df_train)
```

In [33]:
```
df_train['X5'].value_counts()
```

Out[33]:
```
w     231
v     231
q     220
r     215
s     214
d     214
n     212
p     208
m     208
i     207
ae    205
ag    203
ac    200
ab    197
l     195
af    188
ad    185
k     177
c     131
OT    130
j     125
aa    112
Name: X5, dtype: int64
```

In [34]:
```
plt.figure(figsize=(12,5))
sns.boxplot(x=df_train['X5'],y=df_train['y'],palette='rainbow')
```

Out[34]:
```
<AxesSubplot:xlabel='X5', ylabel='y'>
```



In [35]:
```
df_train['X6'].value_counts()
```

Out[35]:
```
g    1042
j    1039
d     625
i     488
l     477
a     206
h     190
k      43
c      38
b      28
f      20
e      12
Name: X6, dtype: int64
```

In [36]:
```
df_train['X8'].value_counts()
```

Out[36]:
```
j    277
s    255
f    243
n    242
i    237
e    225
r    219
a    210
w    196
v    194
b    190
k    176
o    163
m    155
g    130
u    119
t    118
q    117
h    117
y    116
x    105
d    103
l    101
c    100
p    100
Name: X8, dtype: int64
```

In [37]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder
```

```
columns=df_train.select_dtypes(include="object").columns
```

In [38]:
```
X=df_train.drop(['ID','y'],axis=1)
y=df_train[['y']]
```

In [39]:
```
def transform_labels(df,x):
    columns=df.select_dtypes(include="object").columns
    le=LabelEncoder()
    for i in columns:
        le.fit(x[i])
        x[i]=le.transform(x[i])
```

In [40]:
```
transform_labels(df_train,X)
```

In [41]:
```
X.head()
```

Out[41]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X380	X382
0	0	9	17	0	1	0	9	14	0	0	...	0	0	1	0	0	0	0
1	0	0	19	4	1	0	11	14	0	0	...	1	0	0	0	0	0	0
2	5	0	34	2	1	0	9	23	0	0	...	0	0	0	0	0	0	1
3	5	0	34	5	1	0	11	4	0	0	...	0	0	0	0	0	0	0
4	5	9	34	5	1	0	3	13	0	0	...	0	0	0	0	0	0	0

5 rows × 361 columns

In [42]:
```
df_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
```

In [43]:
```
df_test.head()
```

Out[43]:

	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

5 rows × 377 columns

In [45]:
```
#def zero_var_colums(df):
    #var_df=pd.DataFrame(df.var(),columns=['Variance'])
    #return(list(var_df[var_df.Variance==0].index))
#zero_var_colums(df_test)



df_test.var().sort_values().head(15)
```

Out[45]:
```
X295    0.000000
X369    0.000000
X296    0.000000
X257    0.000000
X258    0.000000
X278    0.000238
X233    0.000238
X280    0.000238
X290    0.000238
X293    0.000238
X330    0.000238
X235    0.000238
X288    0.000238
X210    0.000238
X297    0.000238
dtype: float64
```

In [46]:
```
df_test.drop(['X295','X369','X296','X257','X258','X278','X233','X280','X290','X293',
```

In [47]:
```
transform_X0(df_test)
```

In [48]:
```
transform_X1(df_test)
```

In [49]:
```
transform_X5(df_test)
```

In [50]:
```
X_test1=df_test.drop(['ID'],axis=1)
```

In [51]:
```
transform_labels(df_test,X_test1)
```

In [52]:
```
X_test1.head()
```

Out[52]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382
0	5	9	34	5	3	0	0	22	0	0	...	0	0	0	1	0	0	0
1	11	3	8	0	3	0	6	24	0	0	...	0	0	1	0	0	0	0
2	5	9	17	5	3	0	9	9	0	0	...	0	0	0	1	0	0	0
3	5	6	34	5	3	0	11	13	0	0	...	0	0	0	1	0	0	0
4	12	8	17	2	3	0	8	12	0	0	...	1	0	0	0	0	0	0

5 rows × 361 columns

In [53]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_sta
```

In [54]:
```
from sklearn.decomposition import PCA
pc=PCA(n_components=0.95)
pc.fit(X_train)
X_train=pc.transform(X_train)
X_test=pc.transform(X_test)
X_test1=pc.transform(X_test1)
```

In [55]:
```
pc.components_
```

Out[55]:
```
array([[-4.25450753e-02,  3.71504395e-02,  9.93627724e-01, ...,
        -9.31298942e-05, -7.20599995e-06, -1.99369349e-05],
       [ 4.38929120e-03, -5.36670128e-03,  6.55311134e-02, ...,
         1.92453067e-04,  3.64150704e-05,  2.56209915e-04],
       [ 2.05976434e-02, -2.61846223e-02, -1.54345719e-02, ...,
         1.08146422e-05, -2.37078471e-05, -1.77575136e-04],
       ...,
       [ 3.33117532e-02, -1.83753087e-01,  2.18746289e-03, ...,
         8.23887860e-05,  1.80662190e-04,  5.94784782e-04],
       [ 3.91566198e-02,  1.30193146e-02, -1.73516736e-02, ...,
         5.38944699e-04,  1.57544229e-04, -5.71908098e-04],
       [ 7.10504755e-02, -9.65532265e-02, -1.22691205e-02, ...,
         1.59891938e-03,  5.28979201e-05, -3.40748095e-04]])
```

In [56]:
```
pc.explained_variance_
```

Out[56]:
```
array([119.78524835,  49.68485394,  40.04402493,  28.17031354,
         9.36077629,   7.85660411,   4.07826049,   2.23087707,
         1.63060743,   1.29010309,   1.26791744])
```

In [57]:
```
pc.explained_variance_ratio_
```

Out[57]:
```
array([0.43030348, 0.17848246, 0.14384979, 0.10119597, 0.03362663,
       0.02822321, 0.0146503 , 0.00801396, 0.00585762, 0.00463443,
       0.00455473])
```

In [58]:
```
from sklearn import svm
from sklearn.metrics import r2_score, mean_squared_error
from xgboost import XGBRegressor
xgbr= XGBRegressor(random_state=42)
```

In [59]:
```
model = xgbr.fit(X_train,y_train)
```

In [60]:
```
ypred_test = model.predict(X_test)
ypred_test
```

Out[60]:
```
array([100.99419 , 101.231316, 107.688   , 105.37767 ,  91.34555 ,
       102.887566, 100.04225 , 110.02699 , 115.707535,  95.04412 ,
        93.3165  ,  90.10474 ,  95.111336,  94.79391 , 101.465164,
       114.22748 ,  76.06961 ,  93.79366 , 100.53496 , 100.28454 ,
       112.3369  , 105.844666, 118.7009  , 115.39442 ,  93.31118 ,
       107.466995,  93.783745,  91.09463 , 104.02201 ,  95.40309 ,
        93.65064 ,  82.50751 ,  93.79172 ,  96.44883 , 107.52754 ,
        96.002396,  95.48173 , 111.72134 , 106.514244,  97.405525,
       108.331566, 110.329666, 112.33025 , 110.26092 ,  96.33606 ,
```

```
 98.0585  ,  92.81745 , 106.935135, 100.679115,  95.85865 ,
114.16628 , 100.15581 , 126.85878 , 110.38824 ,  93.235794,
113.409164, 104.42806 , 120.353455,  91.32784 , 101.23965 ,
108.01204 , 117.96115 ,  94.48694 ,  94.59156 ,  95.45444 ,
101.94987 ,  94.23243 ,  98.04184 , 113.02262 , 103.70452 ,
113.83915 ,  95.39426 , 105.908936, 101.77586 ,  89.98409 ,
117.80235 , 101.908066, 105.17805 ,  93.067085, 108.82498 ,
110.289795, 111.764755, 108.60635 , 108.42717 , 103.52861 ,
102.71065 ,  91.33906 ,  97.14404 , 113.25149 , 100.80239 ,
108.921425, 104.58007 , 106.83358 ,  95.27252 , 115.272224,
 91.079315, 102.76957 , 104.38412 ,  94.947105, 104.25186 ,
101.570465,  95.45956 , 109.252556, 112.789246,  89.63333 ,
109.68666 ,  93.34646 ,  94.113594,  99.97618 ,  77.380455,
110.139885, 114.6661  , 100.29217 , 100.75897 , 101.39516 ,
 98.5269  ,  92.62535 , 100.420204, 102.144806,  90.904274,
107.141   ,  94.960045,  99.51538 ,  99.310394,  99.88198 ,
111.10294 ,  97.44551 ,  92.58301 , 100.360176,  78.860825,
105.54151 , 115.26738 , 107.05684 , 113.820694, 102.14999 ,
118.64847 ,  94.94961 , 111.21904 ,  95.835846, 115.0293  ,
108.21262 , 111.96856 ,  90.75115 ,  96.52394 ,  96.677124,
 96.08522 ,  96.48014 , 104.89717 , 110.2671  , 108.161835,
 94.38399 , 120.363396,  93.7052  , 104.08911 ,  93.83495 ,
 98.34692 , 101.614365, 104.63597 , 112.89813 , 116.47383 ,
104.80095 ,  91.329704, 111.821014,  96.66159 , 108.995995,
110.20771 ,  89.79245 , 110.97056 ,  98.88193 , 107.77642 ,
111.79505 ,  91.97517 ,  92.18739 ,  92.93292 ,  89.39213 ,
 75.81248 , 101.72808 ,  93.405136,  75.97707 ,  90.79959 ,
100.07425 , 100.243645, 113.863914, 112.2068  ,  77.35782 ,
100.73284 ,  96.808914,  94.814766, 111.83206 , 103.09739 ,
 72.97969 ,  92.44379 , 101.30404 , 106.49136 , 102.861145,
 89.44401 ,  92.21432 , 115.41259 , 104.33681 , 108.21631 ,
114.518555,  92.51649 ,  97.87998 ,  94.62765 , 103.73836 ,
100.29021 , 103.63215 , 110.66335 ,  95.79309 , 112.4076  ,
 92.8793  ,  96.542244, 110.23176 ,  97.619774,  95.728004,
109.91812 , 109.91812 ,  94.38656 , 106.4023  , 107.51505 ,
 99.75061 ,  90.337135, 108.5698  , 109.9854  , 112.54698 ,
107.77386 ,  93.15649 ,  88.18222 ,  94.37823 ,  98.19169 ,
 99.65956 ,  93.19019 ,  97.79879 , 110.19123 , 102.14664 ,
110.74217 ,  92.40173 ,  93.438385, 108.272415,  95.01614 ,
101.17783 ,  99.02806 , 108.320625, 108.893326, 120.82099 ,
 99.08437 , 101.38872 ,  93.33867 , 107.82924 , 111.94778 ,
 77.41061 , 108.80406 ,  97.37536 , 114.82116 ,  76.34367 ,
101.97034 ,  98.26353 , 105.74384 , 100.197525, 100.46902 ,
102.642426, 100.52357 , 116.938896,  97.103294,  89.61608 ,
108.96554 , 104.396965,  95.8209  , 101.20454 , 106.85793 ,
 86.75256 , 102.22313 , 110.85101 , 103.94198 ,  92.47687 ,
 94.72206 , 108.39758 ,  93.03967 ,  85.91476 ,  94.042725,
109.37262 , 105.761925, 118.30954 , 108.86345 , 101.08439 ,
 90.12677 ,  94.61979 ,  89.90845 , 108.08127 ,  97.21637 ,
 92.00087 , 117.07001 , 106.2829  , 105.13431 , 104.742355,
101.44666 , 105.49995 , 110.95368 , 103.193954,  93.03481 ,
 94.870964, 112.89508 ,  89.89829 ,  99.444664,  90.83267 ,
109.33448 , 107.32167 , 101.914505,  96.60568 , 104.69536 ,
113.25506 , 108.69687 , 105.676   , 113.97348 , 125.01331 ,
 99.99385 , 102.0813  ,  91.13341 ,  94.83191 , 100.48446 ,
112.98732 , 108.72809 ,  97.38698 ,  91.48521 , 102.59076 ,
102.45255 , 108.83893 , 100.65271 ,  92.283356,  97.66338 ,
 91.94479 , 102.805786, 115.00394 , 101.533   , 106.55304 ,
 98.67086 , 113.47268 , 113.60822 , 100.35072 ,  76.04711 ,
111.07905 , 106.14099 , 111.47639 , 119.84052 , 105.4155  ,
106.01275 , 105.417564,  85.42231 ,  75.36625 ,  90.24478 ,
 77.55019 , 105.793816,  92.88377 ,  90.507454, 101.536026,
 98.399666, 110.04213 ,  94.58017 ,  99.42125 ,  96.65041 ,
 97.800835,  89.50255 , 110.588066, 115.626945,  88.94857 ,
108.691154,  99.18912 , 111.27462 ,  91.95475 ,  93.43948 ,
 78.9865  ,  91.62982 ,  95.440346, 110.03672 , 104.728195,
 92.81399 , 109.315704,  90.442986, 103.30026 , 103.17703 ,
102.1883  ,  91.93773 , 104.42634 ,  90.705635,  99.54507 ,
109.5369  , 110.00512 , 108.225296,  98.88042 , 102.80311 ,
```

```
112.42455 ,  93.423195, 103.4981  , 109.83349 , 114.12915 ,
114.02011 ,  97.11037 , 100.08464 ,  92.81261 ,  96.162155,
101.38695 , 116.20181 ,  91.496025, 113.309944,  93.946754,
 92.05242 ,  90.99809 , 111.47639 , 109.98602 , 106.11342 ,
 95.14581 , 100.82259 ,  91.88018 , 111.472664, 101.812355,
102.97715 ,  93.13238 , 101.15999 , 118.89356 ,  93.144356,
104.77072 ,  79.4206  ,  90.04172 ,  94.9877  , 101.46908 ,
 93.66144 ,  96.837135, 110.85457 ,  96.85262 ,  92.36994 ,
101.313896, 101.48662 ,  93.15859 , 111.51091 , 112.959946,
116.726845, 100.35435 ,  90.141945,  96.416664, 107.76494 ,
 94.16796 , 101.1183  ,  95.26695 , 110.89929 ,  86.38852 ,
100.80286 , 102.79395 ,  95.513016,  87.53876 , 113.5468  ,
 95.41167 , 109.93954 ,  94.00001 ,  94.72475 , 101.271965,
110.6563  , 101.5705  ,  93.42862 ,  91.49629 ,  90.286385,
109.668304,  99.9191  , 110.687935, 110.12615 ,  98.81443 ,
116.92799 ,  80.83013 ,  99.26932 ,  92.33703 ,  91.28296 ,
102.839874, 110.04082 , 111.28231 , 101.5705  , 103.312485,
111.51932 , 119.07185 ,  90.32578 , 102.157   ,  91.971176,
 90.8392  ,  93.25125 , 109.739845, 100.42816 , 106.813736,
106.79586 , 114.22748 ,  91.50661 , 111.48011 ,  96.80911 ,
 99.323456,  94.250656, 102.11946 , 101.57575 , 110.841675,
107.5502  , 105.95819 , 111.6037  ,  88.82932 ,  99.4537  ,
103.89149 ,  97.73223 , 111.38739 ,  91.090416, 108.94256 ,
109.05126 , 114.49491 , 112.02964 ,  96.0722  , 100.42816 ,
 96.57037 ,  90.802315,  95.22799 ,  98.36165 ,  92.676796,
100.80286 , 107.781975,  94.05046 , 111.61334 , 101.16854 ,
102.4088  , 115.64471 , 114.189125,  99.90782 ,  89.258   ,
 94.423195,  93.98523 , 108.72692 , 103.902664,  92.81294 ,
108.86386 , 100.008484, 103.973915, 100.75499 ,  92.580414,
111.032104,  94.95616 , 116.044846, 101.14357 ,  99.94628 ,
119.96523 ,  93.54803 ,  77.71027 , 105.71809 , 102.14941 ,
 95.50874 , 100.517395, 122.98757 , 107.815186, 106.56449 ,
 99.129234, 104.82263 , 110.41034 ,  90.518326, 100.677666,
108.101524, 115.33774 , 103.472466, 110.44541 , 111.47639 ,
114.62898 ,  97.39118 , 103.22525 , 102.84344 ,  95.49423 ,
102.523445,  93.949356,  97.270035, 104.72082 ,  92.91417 ,
108.71816 , 110.28731 , 105.73259 ,  91.23382 ,  96.57564 ,
103.22691 ,  98.6038  , 116.27588 ,  96.54176 ,  93.09193 ,
 94.641396, 105.61907 ,  94.277336, 101.3795  , 112.18808 ,
113.1859  ,  93.10387 , 104.658966, 111.19827 , 110.14878 ,
110.33341 , 111.79003 ,  90.61679 , 101.43172 , 104.12199 ,
102.87369 , 110.19008 ,  91.970146,  92.751236, 105.34453 ,
 96.369064,  94.65359 , 113.1408  ,  94.29656 , 117.07001 ,
103.57513 , 114.94196 ,  93.74387 ,  89.83177 ,  92.93969 ,
100.46634 , 103.18666 ,  91.95402 ,  90.39902 , 110.59774 ,
109.22875 , 118.48195 , 110.14109 ,  99.70264 ,  97.17046 ,
109.73049 ,  93.16762 ,  97.46559 ,  91.292915, 117.27996 ,
103.670044,  93.92334 ,  95.54128 ,  90.54099 ,  90.45874 ,
101.5248  ,  94.91677 , 100.19115 , 105.87965 ,  99.95767 ,
106.97232 , 103.30889 ,  94.44135 , 103.549965, 107.95853 ,
 94.57173 , 119.58549 ,  98.453316,  95.19098 , 112.64688 ,
 93.483665, 117.101746,  94.96509 ,  93.423195, 111.19783 ,
112.269554, 112.84059 ,  91.06594 ,  90.28442 , 115.48058 ,
105.19637 ,  94.57473 , 102.646675, 100.473305, 108.15422 ,
112.29169 ,  97.470634, 103.11832 , 103.11594 , 113.7936  ,
 96.43029 , 108.995995,  93.60247 ,  93.6326  , 118.05413 ,
102.098694,  95.93561 ,  95.32312 ,  88.02217 , 106.89523 ,
106.50627 , 104.70546 , 109.232376,  93.387   ,  76.28055 ,
104.95347 ,  91.96914 ,  90.92138 , 106.92748 ,  97.22925 ,
109.358025, 101.25301 ,  93.81464 , 113.057304, 112.79463 ,
110.90112 , 110.20226 ,  93.75135 , 114.38684 , 102.642426,
 73.57013 ,  79.18503 ,  92.99038 , 112.7579  ,  92.49695 ,
 89.9627  ,  96.336555, 105.21487 ,  94.95219 ,  94.52028 ,
102.86424 ,  93.87087 , 111.08644 ,  94.324356, 102.77143 ,
112.28664 ,  95.20547 ,  94.40266 , 108.38588 ,  93.56046 ,
110.41054 , 102.101585, 113.24008 , 110.428795, 102.63565 ,
 94.41151 ,  98.15246 ,  96.30569 ,  95.6727  , 103.94405 ,
104.54692 , 103.27606 ,  94.33418 ,  91.146904,  95.720314,
113.89502 ,  99.58721 , 110.13101 , 101.96042 ,  98.87385 ,
```

```
 90.455574,  99.067696, 114.38462 ,  99.62067 ,  91.005554,
103.42034 ,  94.01984 ,  93.22683 , 106.7599  , 106.93581 ,
 96.736275, 116.000244, 109.26226 ,  93.97169 ,  98.12063 ,
 79.60432 , 103.68867 , 108.60635 , 112.049   , 108.79084 ,
103.453606,  93.88054 , 113.31768 ,  89.152664,  99.78163 ,
 90.9184  , 108.28418 , 113.31768 ,  91.94798 , 105.83466 ,
114.53513 , 114.60073 ,  93.323555, 116.23117 ,  75.15386 ,
118.3473  , 115.39442 , 111.21904 ,  93.10986 ,  92.13752 ,
 81.10999 , 117.694084, 105.779236,  99.64544 , 101.06354 ,
107.38225 ,  98.265564, 106.13959 ,  88.37153 , 106.38045 ,
 92.90871 , 112.371925, 112.10475 ,  91.549065,  83.480995,
 90.33705 , 108.001   , 107.18128 ,  94.42443 , 108.261566,
112.39655 ,  80.487335,  93.636314,  95.95124 ,  96.033745,
 90.73597 ,  93.42256 , 115.55604 ,  97.71796 ,  91.512245,
103.57497 ,  91.86244 ,  91.101036, 102.61935 , 111.67142 ,
118.81601 ,  90.80399 , 105.741356,  92.484856,  97.31036 ,
 95.700645, 120.31693 ,  89.27364 , 102.12338 , 113.18463 ,
111.566536,  96.09706 ,  91.123375,  94.98837 ,  92.86031 ,
 91.83852 , 107.45041 , 108.44219 ,  97.34553 ,  89.56844 ,
117.32538 ,  95.513016, 119.55242 , 103.49347 ,  89.871414,
107.31294 ,  98.77034 , 106.26067 ,  94.93544 , 101.08556 ,
 88.001976, 106.59395 ], dtype=float32)
```

In [61]:
```
ypred_train = model.predict(X_train)
ypred_train
```

Out[61]:
```
array([121.61863 , 105.48161 ,  91.68057 , ..., 108.94938 ,  99.92531 ,
       105.008804], dtype=float32)
```

In [62]:
```
print(r2_score(ypred_train, y_train))
```

```
0.9016925733694292
```

In [63]:
```
print(mean_squared_error(ypred_train, y_train))
```

```
11.697073995477288
```

In [64]:
```
df_test_prediction=model.predict(X_test1)
df_test_prediction
```

Out[64]:
```
array([111.800644,  99.86678 ,  97.033104, ..., 104.33736 , 100.37338 ,
       106.952415], dtype=float32)
```

In [ ]:

In [ ]:

In [ ]: