Importing libraries and datasets along with data manipulation, some data visualization and data wrangling

In [1]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
os.getcwd()
```

Out[2]:

```
'C:\\Users\\Preksha\\Simplilearn\\ML\\Project'
```

In [3]:

```
data_train = pd.read_csv('train.csv')
```

In [4]:

```
data_test = pd.read_csv('test.csv')
```

In [5]:

```
data_train.head()
```

Out[5]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	

5 rows × 378 columns

In [6]:
```
data_test.head()
```

Out[6]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X38
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	

5 rows × 377 columns

In [7]:
```
data_train.size
```

Out[7]:
```
1591002
```

In [8]:
```
data_test.shape
```

Out[8]:
```
(4209, 377)
```

In [9]:
```
data_train.shape
```

Out[9]:
```
(4209, 378)
```

In [10]:
```
data_test.size
```

Out[10]:
```
1586793
```

In [11]:
```
data_train.dtypes
```

Out[11]:
```
ID          int64
y         float64
X0         object
X1         object
X2         object
           ...
X380        int64
X382        int64
X383        int64
X384        int64
X385        int64
Length: 378, dtype: object
```

In [12]:
```
data_train.describe()
```

Out[12]:

	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 [13]:
```
data_test.describe()
```

Out[13]:

	ID	X10	X11	X12	X13	X14	
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.0
mean	4211.039202	0.019007	0.000238	0.074364	0.061060	0.427893	0.0
std	2423.078926	0.136565	0.015414	0.262394	0.239468	0.494832	0.02
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	2115.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
50%	4202.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
75%	6310.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.0
max	8416.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0

8 rows × 369 columns

In [14]:

```
data_train.corr()
```

Out[14]:

	ID	y	X10	X11	X12	X13	X14	X15	
ID	1.000000	-0.055108	0.001602	NaN	0.058988	-0.031917	-0.025438	0.002237	-0
y	-0.055108	1.000000	-0.026985	NaN	0.089792	0.048276	0.193643	0.023116	0
X10	0.001602	-0.026985	1.000000	NaN	-0.033084	-0.028806	-0.100474	-0.002532	-0
X11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
X12	0.058988	0.089792	-0.033084	NaN	1.000000	0.214825	-0.246513	-0.006212	-0
...	...	...	...	...	...	...	...	...	
X380	-0.013577	0.040932	-0.010479	NaN	-0.005566	0.023045	0.007743	-0.001968	-0
X382	-0.038171	-0.159815	-0.010164	NaN	-0.024937	-0.021713	0.012713	-0.001908	-0
X383	-0.009332	0.040291	-0.004740	NaN	-0.011628	-0.010125	0.023604	-0.000890	-0
X384	-0.015355	-0.004591	-0.002532	NaN	-0.006212	0.041242	0.025199	-0.000475	-0

In [15]:

```
plt.figure(figsize=(10,8))
sns.distplot(data_train['y'])
```

```
C:\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function
for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]:

```
<AxesSubplot:xlabel='y', ylabel='Density'>
```

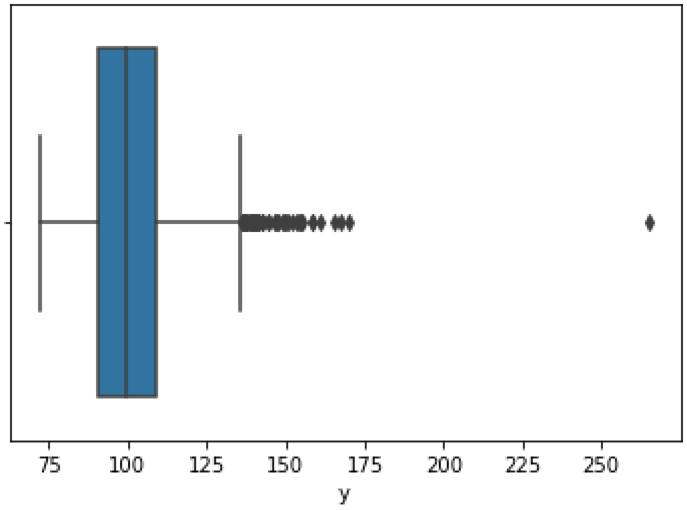
In [16]:

```
sns.boxplot(data_train['y'])
```

```
C:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only v
alid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[16]:

```
<AxesSubplot:xlabel='y'>
```



In [17]:

```
filter = data_train['y'].values < 175
```

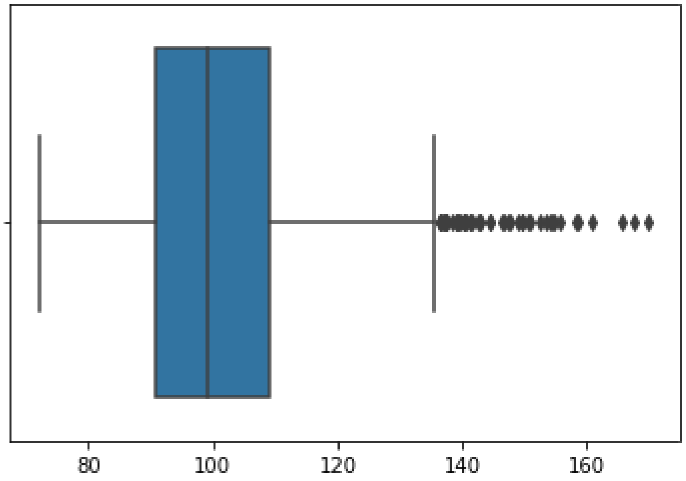
In [18]:

```
data_train = data_train[filter]
```

```
In [19]: sns.boxplot(data_train['y'])
```

```
ts without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```

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



Since now there are many outliers, None of them are removed since they can be significant.

1. If for any column(s), the variance is equal to zero, then you need to remove those variable(s).

```
In [20]: print(data_train.var(axis = 0))
```

```
ID      5.941938e+06
y       1.543594e+02
X10     1.313400e-02
X11     0.000000e+00
X12     6.947230e-02
            ...
X380    8.016469e-03
X382    7.548527e-03
X383    1.661126e-03
X384    4.751722e-04
X385    1.424161e-03
Length: 370, dtype: float64
```

```
In [21]: variance_zero_train = data_train.var()[data_train.var() == 0].index.values
```

```
In [22]: variance_zero_train
```

```
Out[22]: array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
                'X293', 'X297', 'X330', 'X347'], dtype=object)
```

In [23]:
```
variance_zero_train = data_train.var()[data_train.var() == 0].index.values
```

In [24]:
```
variance_zero_train
```

Out[24]:
```
array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'], dtype=object)
```

In [25]:
```
data_train = data_train.drop(variance_zero_train, axis = 1)
```

In [26]:
```
data_train = data_train.drop(['ID'], axis = 1)
```

In [27]:
```
data_train
```

Out[27]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X38
0	130.81	k	v	at	a	d	u	j	o	0	...	0	0	1	0	0	
1	88.53	k	t	av	e	d	y	l	o	0	...	1	0	0	0	0	
2	76.26	az	w	n	c	d	x	j	x	0	...	0	0	0	0	0	
3	80.62	az	t	n	f	d	x	l	e	0	...	0	0	0	0	0	
4	78.02	az	v	n	f	d	h	d	n	0	...	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4204	107.39	ak	s	as	c	d	aa	d	q	0	...	1	0	0	0	0	
4205	108.77	j	o	t	d	d	aa	h	h	0	...	0	1	0	0	0	
4206	109.22	ak	v	r	a	d	aa	g	e	0	...	0	0	1	0	0	
4207	87.48	al	r	e	f	d	aa	l	u	0	...	0	0	0	0	0	
4208	110.85	z	r	ae	c	d	aa	g	w	0	...	1	0	0	0	0	

4208 rows × 365 columns

In [28]:
```
variance_zero_test = data_test.var()[data_test.var() == 0].index.values
```

In [29]:
```
variance_zero_test
```

Out[29]:
```
array(['X257', 'X258', 'X295', 'X296', 'X369'], dtype=object)
```

In [30]:
```
data_test = data_test.drop(variance_zero_test, axis = 1)
```

In [31]:
```
data_test = data_test.drop(['ID'], axis = 1)
```

In [32]:
```
data_test
```

Out[32]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X
0	az	v	n	f	d	t	a	w	0	0	...	0	0	0	1	0	
1	t	b	ai	a	d	b	g	y	0	0	...	0	0	1	0	0	
2	az	v	as	f	d	a	j	j	0	0	...	0	0	0	1	0	
3	az	l	n	f	d	z	l	n	0	0	...	0	0	0	1	0	
4	w	s	as	c	d	y	i	m	0	0	...	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4204	aj	h	as	f	d	aa	j	e	0	0	...	0	0	0	0	0	
4205	t	aa	ai	d	d	aa	j	y	0	0	...	0	1	0	0	0	
4206	y	v	as	f	d	aa	d	w	0	0	...	0	0	0	0	0	
4207	ak	v	as	a	d	aa	c	q	0	0	...	0	0	1	0	0	

2. Check for null and unique values for test and train sets.

In [33]:
```
data_train.isna().sum().values
```

Out[33]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [34]:
```
np.sum(data_train.isnull().sum())
```

Out[34]:
```
0
```

In [35]:
```
data_train.nunique().values
```

```
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
       2,     2], dtype=int64)
```

3. Apply label encoder.

In [36]:
```
object_columns_train = data_train.select_dtypes(include = [object])
```

In [37]:
```
object_columns_train
```

Out[37]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	k	v	at	a	d	u	j	o
1	k	t	av	e	d	y	l	o
2	az	w	n	c	d	x	j	x
3	az	t	n	f	d	x	l	e
4	az	v	n	f	d	h	d	n
...	...	...	...	...	...	...	...	...
4204	ak	s	as	c	d	aa	d	q
4205	j	o	t	d	d	aa	h	h
4206	ak	v	r	a	d	aa	g	e
4207	al	r	e	f	d	aa	l	u
4208	z	r	ae	c	d	aa	g	w

4208 rows × 8 columns

In [38]:
```
object_columns_test = data_test.select_dtypes(include = [object])
```

In [39]:

```
object_columns_test
```

Out[39]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	az	v	n	f	d	t	a	w
1	t	b	ai	a	d	b	g	y
2	az	v	as	f	d	a	j	j
3	az	l	n	f	d	z	l	n
4	w	s	as	c	d	y	i	m
...	...	...	...	...	...	...	...	...
4204	aj	h	as	f	d	aa	j	e
4205	t	aa	ai	d	d	aa	j	y
4206	y	v	as	f	d	aa	d	w
4207	ak	v	as	a	d	aa	c	q
4208	t	aa	ai	c	d	aa	g	r

4209 rows × 8 columns

In [40]:

```
from sklearn.preprocessing import LabelEncoder
```

In [41]:

```
label = LabelEncoder()
```

In [42]:

```
data_train['X0'] = label.fit_transform(data_train['X0'])
data_train['X1'] = label.fit_transform(data_train['X1'])
data_train['X2'] = label.fit_transform(data_train['X2'])
data_train['X3'] = label.fit_transform(data_train['X3'])
data_train['X4'] = label.fit_transform(data_train['X4'])
data_train['X5'] = label.fit_transform(data_train['X5'])
data_train['X6'] = label.fit_transform(data_train['X6'])
data_train['X8'] = label.fit_transform(data_train['X8'])
```

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

Out[43]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380
0	130.81	32	23	17	0	3	24	9	14	0	...	0	0	1	0	0	0
1	88.53	32	21	19	4	3	28	11	14	0	...	1	0	0	0	0	0
2	76.26	20	24	34	2	3	27	9	23	0	...	0	0	0	0	0	0
3	80.62	20	21	34	5	3	27	11	4	0	...	0	0	0	0	0	0
4	78.02	20	23	34	5	3	12	3	13	0	...	0	0	0	0	0	0

5 rows × 365 columns

In [44]:
```
data_test['X0'] = label.fit_transform(data_test['X0'])
data_test['X1'] = label.fit_transform(data_test['X1'])
data_test['X2'] = label.fit_transform(data_test['X2'])
data_test['X3'] = label.fit_transform(data_test['X3'])
data_test['X4'] = label.fit_transform(data_test['X4'])
data_test['X5'] = label.fit_transform(data_test['X5'])
data_test['X6'] = label.fit_transform(data_test['X6'])
data_test['X8'] = label.fit_transform(data_test['X8'])
```

In [45]:
```
data_test.head()
```

Out[45]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X3
0	21	23	34	5	3	26	0	22	0	0	...	0	0	0	1	0	0	
1	42	3	8	0	3	9	6	24	0	0	...	0	0	1	0	0	0	
2	21	23	17	5	3	0	9	9	0	0	...	0	0	0	1	0	0	
3	21	13	34	5	3	31	11	13	0	0	...	0	0	0	1	0	0	
4	45	20	17	2	3	30	8	12	0	0	...	1	0	0	0	0	0	

5 rows × 371 columns

In [46]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [47]:
```
scaler = MinMaxScaler()
```

In [48]:

```
data_train['X0'] = scaler.fit_transform(data_train[['X0']])
data_train['X1'] = scaler.fit_transform(data_train[['X1']])
data_train['X2'] = scaler.fit_transform(data_train[['X2']])
data_train['X3'] = scaler.fit_transform(data_train[['X3']])
data_train['X4'] = scaler.fit_transform(data_train[['X4']])
data_train['X5'] = scaler.fit_transform(data_train[['X5']])
data_train['X6'] = scaler.fit_transform(data_train[['X6']])
data_train['X8'] = scaler.fit_transform(data_train[['X8']])
```

In [49]:

```
data_test['X0'] = scaler.fit_transform(data_test[['X0']])
data_test['X1'] = scaler.fit_transform(data_test[['X1']])
data_test['X2'] = scaler.fit_transform(data_test[['X2']])
data_test['X3'] = scaler.fit_transform(data_test[['X3']])
data_test['X4'] = scaler.fit_transform(data_test[['X4']])
data_test['X5'] = scaler.fit_transform(data_test[['X5']])
data_test['X6'] = scaler.fit_transform(data_test[['X6']])
data_test['X8'] = scaler.fit_transform(data_test[['X8']])
```

In [50]:

```
data_train
```

Out[50]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10
0	130.81	0.695652	0.884615	0.395349	0.000000	1.0	0.857143	0.818182	0.583333	0
1	88.53	0.695652	0.807692	0.441860	0.666667	1.0	1.000000	1.000000	0.583333	0
2	76.26	0.434783	0.923077	0.790698	0.333333	1.0	0.964286	0.818182	0.958333	0
3	80.62	0.434783	0.807692	0.790698	0.833333	1.0	0.964286	1.000000	0.166667	0
4	78.02	0.434783	0.884615	0.790698	0.833333	1.0	0.428571	0.272727	0.541667	0
...	...	...	...	...	...	...	...	...	...	...
4204	107.39	0.173913	0.769231	0.372093	0.333333	1.0	0.000000	0.272727	0.666667	0
4205	108.77	0.673913	0.615385	0.930233	0.500000	1.0	0.000000	0.636364	0.291667	0
4206	109.22	0.173913	0.884615	0.883721	0.000000	1.0	0.000000	0.545455	0.166667	0
4207	87.48	0.195652	0.730769	0.581395	0.833333	1.0	0.000000	1.000000	0.833333	0
4208	110.85	1.000000	0.730769	0.069767	0.333333	1.0	0.000000	0.545455	0.916667	0

4208 rows × 365 columns

In [51]:
```
data_test
```

Out[51]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X1
0	0.437500	0.884615	0.772727	0.833333	1.0	0.838710	0.000000	0.916667	0	(
1	0.875000	0.115385	0.181818	0.000000	1.0	0.290323	0.545455	1.000000	0	(
2	0.437500	0.884615	0.386364	0.833333	1.0	0.000000	0.818182	0.375000	0	(
3	0.437500	0.500000	0.772727	0.833333	1.0	1.000000	1.000000	0.541667	0	(
4	0.937500	0.769231	0.386364	0.333333	1.0	0.967742	0.727273	0.500000	0	(
...	...	...	...	...	...	...	...	...	...	..
4204	0.125000	0.346154	0.386364	0.833333	1.0	0.032258	0.818182	0.166667	0	(
4205	0.875000	0.038462	0.181818	0.500000	1.0	0.032258	0.818182	1.000000	0	(
4206	0.979167	0.884615	0.386364	0.833333	1.0	0.032258	0.272727	0.916667	0	(
4207	0.145833	0.884615	0.386364	0.000000	1.0	0.032258	0.181818	0.666667	0	(

In [52]:
```
X = data_train.drop('y', axis=1)
y = data_train.y
```

In [53]:
```
X.shape, y.shape
```

Out[53]:
```
((4208, 364), (4208,))
```

In [54]:
```
from sklearn.model_selection import train_test_split
```

In [55]:
```
X_train, X_val, y_train, y_val = train_test_split(X, y, train_size = 0.7, rar
```

In [56]:
```
X_train.shape, X_val.shape, y_train.shape, y_val.shape
```

Out[56]:
```
((2945, 364), (1263, 364), (2945,), (1263,))
```

4. Perform dimensionality reduction.

In [57]:
```
from sklearn.decomposition import PCA
```

In [58]:
```
pca = PCA(0.98, svd_solver = 'full')
```

In [59]:
```
pca.fit(X)
```

Out[59]:
```
PCA(n_components=0.98, svd_solver='full')
```

In [60]:
```
pca.n_components_
```

Out[60]:
```
110
```

In [61]:
```
pca.explained_variance_ratio_
```

Out[61]:
```
array([0.12972041, 0.0875348 , 0.08510149, 0.06749473, 0.05638419,
       0.04696922, 0.03762569, 0.03240524, 0.02779476, 0.0247543 ,
       0.02309815, 0.01950321, 0.01695995, 0.0162877 , 0.01521659,
       0.01464838, 0.01375511, 0.01211821, 0.01030446, 0.01012229,
       0.00959678, 0.00888843, 0.00876491, 0.00834633, 0.0079112 ,
       0.00755361, 0.00721222, 0.0064738 , 0.00638357, 0.00575064,
       0.00527769, 0.00506438, 0.00475303, 0.00460527, 0.00437103,
       0.00423123, 0.00420064, 0.00403213, 0.00401001, 0.00384583,
       0.00359433, 0.0034374 , 0.003424  , 0.00335713, 0.00330594,
       0.00314226, 0.0030995 , 0.00297767, 0.00289155, 0.00282024,
       0.00270373, 0.00260173, 0.00257974, 0.0024423 , 0.0023691 ,
       0.00234018, 0.00226474, 0.0021309 , 0.00209198, 0.00200379,
       0.00197952, 0.00194329, 0.00183793, 0.00178381, 0.00172609,
       0.00170382, 0.00165383, 0.00160349, 0.00159155, 0.00152277,
       0.00147535, 0.00142073, 0.00138873, 0.00137832, 0.00134117,
       0.00130356, 0.00127309, 0.00123871, 0.00122182, 0.00119618,
       0.00115391, 0.00111155, 0.00110866, 0.00107272, 0.00105643,
       0.00104779, 0.00101136, 0.00098695, 0.00096187, 0.00096009,
       0.00089859, 0.00088045, 0.00086495, 0.00084355, 0.00083346,
       0.00080651, 0.00077997, 0.00076909, 0.00074416, 0.00073261,
       0.00072064, 0.00070978, 0.00068515, 0.00065235, 0.00062175,
       0.00061682, 0.00060398, 0.00058774, 0.00057035, 0.00056411])
```

In [62]:
```
X_train_reduced = pd.DataFrame(pca.transform(X_train))
```

In [63]:
```
X_val_reduced = pd.DataFrame(pca.transform(X_val))
```

In [64]:
```
X_train_reduced.shape, X_val_reduced.shape
```

Out[64]:
```
((2945, 110), (1263, 110))
```

5. Predict your test_df values using XGBoost.

In [65]:
```
import xgboost
```

In [66]:
```
xgb_reg = xgboost.XGBRegressor(objective='reg:linear',learning_rate=0.1)
```

In [67]:

```
xgb_reg.fit(X_train_reduced, y_train)
```

```
[21:51:01] WARNING: c:\ci\xgboost-split_1638290375667\work\src\objective\re
gression_obj.cu:188: reg:linear is now deprecated in favor of reg:squareder
ror.
```

Out[67]:

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=Fal
se,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.1, max_delta_step=
0,
             max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, objective='reg:linear', predictor='auto',
             random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
             subsample=1, tree_method='exact', validate_parameters=1,
             verbosity=None)
```

In [68]:

```
y_pred = xgb_reg.predict(X_val_reduced)
```

In [69]:

```
y_pred
```

Out[69]:

```
array([ 99.68166 ,  97.86926 , 102.90199 , ...,  92.90996 , 107.921036,
        92.72516 ], dtype=float32)
```

In [70]:

```
y_val
```

Out[70]:

```
764      98.97
3951    110.93
2250    112.82
879     109.39
3737    118.44
         ...  
485      87.28
1196    114.26
3555     90.85
3024    108.90
3165    135.13
Name: y, Length: 1263, dtype: float64
```

In [71]:

```
from sklearn import metrics
```

In [72]:

```
np.sqrt(metrics.mean_squared_error(y_val, y_pred))
```

Out[72]:

```
8.693084426606354
```