In [44]:

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

In [45]:

```
# Step1: Read the train.csv and test.csv data
df_test = pd.read_csv('test.csv')
df_train=pd.read_csv("train.csv")
```

In [46]:

```
df_train.shape
```

Out[46]:

```
(4209, 378)
```

In [47]:

```
df_test.shape
```

Out[47]:

```
(4209, 377)
```

In [48]:

```
df_train
```

Out[48]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8405	107.39	ak	s	as	c	d	aa	d	q	...	1	0	0	0	0
4205	8406	108.77	j	o	t	d	d	aa	h	h	...	0	1	0	0	0
4206	8412	109.22	ak	v	r	a	d	aa	g	e	...	0	0	1	0	0
4207	8415	87.48	al	r	e	f	d	aa	l	u	...	0	0	0	0	0
4208	8417	110.85	z	r	ae	c	d	aa	g	w	...	1	0	0	0	0

4209 rows × 378 columns

In [49]:

```
df_test
```

Out[49]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X38
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4204	8410	aj	h	as	f	d	aa	j	e	0	...	0	0	0	0	0	
4205	8411	t	aa	ai	d	d	aa	j	y	0	...	0	1	0	0	0	
4206	8413	y	v	as	f	d	aa	d	w	0	...	0	0	0	0	0	
4207	8414	ak	v	as	a	d	aa	c	q	0	...	0	0	1	0	0	
4208	8416	t	aa	ai	c	d	aa	g	r	0	...	1	0	0	0	0	

4209 rows × 377 columns

In [50]:

```
df_train.describe()
```

Out[50]:

	ID	y	X10	X11	X12	X13	X1
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	4209.000000	4209.00000
mean	4205.960798	100.669318	0.013305	0.0	0.075077	0.057971	0.42813
std	2437.608688	12.679381	0.114590	0.0	0.263547	0.233716	0.49486
min	0.000000	72.110000	0.000000	0.0	0.000000	0.000000	0.00000
25%	2095.000000	90.820000	0.000000	0.0	0.000000	0.000000	0.00000
50%	4220.000000	99.150000	0.000000	0.0	0.000000	0.000000	0.00000
75%	6314.000000	109.010000	0.000000	0.0	0.000000	0.000000	1.00000
max	8417.000000	265.320000	1.000000	0.0	1.000000	1.000000	1.00000

8 rows × 370 columns

In [51]:

```
#extract useful columns for training by removing ID and Y columns
x_train=df_train.iloc[:,2:]
x_test=df_test.iloc[:,1:]
# extract the target column Y
y_train=df_train['y'].values
```

In [52]:

```
x_test
```

Out[52]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X38(
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	(
4208	t	aa	ai	c	d	aa	g	r	0	0	...	1	0	0	0	0	(

4209 rows × 376 columns

In [53]:

```
#Check for null in the test and train sets.
x_train.isna().any()
```

Out[53]:

```
X0      False
X1      False
X2      False
X3      False
X4      False
        ...  
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 376, dtype: bool
```

In [54]:

```
x_test.isna().any()
```

Out[54]:

```
X0      False
X1      False
X2      False
X3      False
X4      False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 376, dtype: bool
```

In [55]:

```
#Check for unique values for test and train sets and drop them.
for col in x_train.columns:
    car=len(np.unique(x_train[col]))
    if car==1:
        print(col)
        x_train.drop(col, axis=1,inplace=True)
        x_test.drop(col, axis=1, inplace=True)
```

```
X11
X93
X107
X233
X235
X268
X289
X290
X293
X297
X330
X347
```

In [56]:

```
x_train
```

Out[56]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X38(
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	(
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	(
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	(
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	(
4	az	v	n	f	d	h	d	n	0	0	...	0	0	0	0	0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
4204	ak	s	as	c	d	aa	d	q	0	0	...	1	0	0	0	0	(
4205	j	o	t	d	d	aa	h	h	0	0	...	0	1	0	0	0	(
4206	ak	v	r	a	d	aa	g	e	0	1	...	0	0	1	0	0	(
4207	al	r	e	f	d	aa	l	u	0	0	...	0	0	0	0	0	(
4208	z	r	ae	c	d	aa	g	w	0	0	...	1	0	0	0	0	(

4209 rows × 364 columns

In [57]:

```
# Import label encoder
from sklearn import preprocessing
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

for col in x_train.columns:
    typ = df_train[col].dtype
    if typ == 'object':
        # Encode labels in column 'species'.
        x_train[col]= label_encoder.fit_transform(x_train[col])
        x_test[col]= label_encoder.fit_transform(x_test[col])
```

In [58]:

```
x_train
```

Out[58]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X38(
0	32	23	17	0	3	24	9	14	0	0	...	0	0	1	0	0	(
1	32	21	19	4	3	28	11	14	0	0	...	1	0	0	0	0	(
2	20	24	34	2	3	27	9	23	0	0	...	0	0	0	0	0	(
3	20	21	34	5	3	27	11	4	0	0	...	0	0	0	0	0	(
4	20	23	34	5	3	12	3	13	0	0	...	0	0	0	0	0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
4204	8	20	16	2	3	0	3	16	0	0	...	1	0	0	0	0	(
4205	31	16	40	3	3	0	7	7	0	0	...	0	1	0	0	0	(
4206	8	23	38	0	3	0	6	4	0	1	...	0	0	1	0	0	(
4207	9	19	25	5	3	0	11	20	0	0	...	0	0	0	0	0	(
4208	46	19	3	2	3	0	6	22	0	0	...	1	0	0	0	0	(

4209 rows × 364 columns

In [59]:

```
x_train.describe()
```

Out[59]:

	X0	X1	X2	X3	X4	X5	
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.(
mean	29.760751	11.113566	17.306486	2.919696	2.997862	13.340223	6.8
std	13.738338	8.531001	10.899914	1.739912	0.073900	8.250832	2.9
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.(
25%	19.000000	3.000000	8.000000	2.000000	3.000000	5.000000	6.(
50%	35.000000	13.000000	16.000000	2.000000	3.000000	15.000000	7.(
75%	43.000000	20.000000	25.000000	5.000000	3.000000	21.000000	9.(
max	46.000000	26.000000	43.000000	6.000000	3.000000	28.000000	11.(

8 rows × 364 columns

In [60]:

```
x_test.shape
```

Out[60]:

```
(4209, 364)
```

In [61]:

```
x_test
```

Out[61]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X38(
0	21	23	34	5	3	26	0	22	0	0	...	0	0	0	1	0	(
1	42	3	8	0	3	9	6	24	0	0	...	0	0	1	0	0	(
2	21	23	17	5	3	0	9	9	0	0	...	0	0	0	1	0	(
3	21	13	34	5	3	31	11	13	0	0	...	0	0	0	1	0	(
4	45	20	17	2	3	30	8	12	0	0	...	1	0	0	0	0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
4204	6	9	17	5	3	1	9	4	0	0	...	0	0	0	0	0	(
4205	42	1	8	3	3	1	9	24	0	0	...	0	1	0	0	0	(
4206	47	23	17	5	3	1	3	22	0	0	...	0	0	0	0	0	(
4207	7	23	17	0	3	1	2	16	0	0	...	0	0	1	0	0	(
4208	42	1	8	2	3	1	6	17	0	0	...	1	0	0	0	0	(

4209 rows × 364 columns

In [62]:

```
#Perform dimensionality reduction.
from sklearn.decomposition import PCA

sklearn_pca = PCA(n_components=0.95, random_state=420)
sklearn_pca.fit(x_train)
```

Out[62]:

```
PCA(copy=True, iterated_power='auto', n_components=0.95, random_state=420,
    svd_solver='auto', tol=0.0, whiten=False)
```

In [63]:

```
x_train_transformed=sklearn_pca.transform(x_train)
```

In [64]:

```
print(x_train_transformed.shape)
```

```
(4209, 6)
```

In [65]:

```
print(x_test.shape)
```

```
(4209, 364)
```

In [66]:

```
x_test_transformed=sklearn_pca.transform(x_test)
```

In [67]:

```
print(x_test_transformed.shape)
```

```
(4209, 6)
```

In [68]:

```
print(x_test_transformed)
```

```
[[ 15.12259658  12.42634376  16.57568771   0.3815911   10.74927236
    6.77829903]
 [-16.4185227   -6.08780452  -5.81810847  -0.64384353  11.84673987
    0.97206332]
 [ 11.31088967  -2.24098735  -5.68320971  15.24959691  -2.7772942
   -2.55753002]
 ...
 [-13.46766391   3.52415451  -0.38763446  20.58670915   9.05195372
    3.60970753]
 [ 24.08692488  -6.5122012   -6.38950949  12.2127527    4.55765907
    4.37652389]
 [-16.55794181  -5.49565202 -13.7038912    2.25695886   5.14286793
    1.17658806]]
```

In [69]:

```
from xgboost import XGBRegressor
```

In [70]:

```
# train model using XGBoost
model = XGBRegressor(objective='reg:squarederror', n_estimators=1000)
model.fit(x_train_transformed, y_train)
```

Out[70]:

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=Fa
lse,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=na
n,
             monotone_constraints='()', n_estimators=1000, n_jobs=8,
             num_parallel_tree=1, objective='reg:squarederror',
             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 [71]:

```
y_pred=model.predict(x_train_transformed)
```

In [72]:

```
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt
print(sqrt(mean_squared_error(y_train, y_pred)))
```

```
2.144388503310576
```

In [73]:

```
#Predict your test_df values using XGBoost
y_test_pred=model.predict(x_test_transformed)
```

In [74]:

```
y_test_pred
```

Out[74]:

```
array([ 81.000244,  97.20449 ,  84.81835 , ..., 103.20449 , 108.65345 ,
       102.384254], dtype=float32)
```