mohsensho / machineLearningwithPython Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

machineLearningwithPython / Mercedes-BenzGreenerManufacturing_Project1.ipynb

Mohsensho apply xgboost History

0 contributors

2877 lines (2877 sloc) | 115 KB

In [1]:

```python
# Importing library
import pandas as pd # data processing, CV file I/0 (e.g. pd.read csv)
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing # Import Label Encoder
```

In [5]:

```python
# Read csv
train_df = pd.read_csv('Datasets/Mercedes_train.csv')
test_df = pd.read_csv('Datasets/Mercedes_test.csv')
print (train_df.shape) # Find Number of rows and columns
print (train_df.columns)
print (test_df.shape) # Find Number of rows and columns
print (test_df.columns)
train_df.head()
# Show first 5 records
```

```
(4209, 378)
Index(['ID', 'y', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8',
       ...
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=378)
(4209, 377)
Index(['ID', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10',
       ...
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=377)
```

Out[5]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X3
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	0	0	
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	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	0	0	
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	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	0	0	

5 rows × 378 columns

In [6]:

```python
# Describe the dataset i.r.t its data Distribution
train_df.describe()
```

Out[6]:

	ID	y	X10	X11	X12	X13	X14	X15	
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	4209.000000	4209.000000	4209.000000	4209.
mean	4205.960798	100.669318	0.013305	0.0	0.075077	0.057971	0.428130	0.000475	0
std	2437.608688	12.679381	0.114590	0.0	0.263547	0.233716	0.494867	0.021796	0
min	0.000000	72.110000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.
25%	2095.000000	90.820000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.
50%	4220.000000	99.150000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.
75%	6314.000000	109.010000	0.000000	0.0	0.000000	0.000000	1.000000	0.000000	0.
max	8417.000000	265.320000	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	1.

8 rows × 370 columns

In [12]:

```python
#If for any column(s), the variance is equal to zero, then you need to remove those variable(s).
# Check the variance
train_df.var(numeric_only=True)
```

Out[12]: ID 5.941936e+06

Out[12]:
```
ID      5.941936e+06
y       1.607667e+02
X10     1.313092e-02
X11     0.000000e+00
X12     6.945713e-02
            ...
X380    8.014579e-03
X382    7.546747e-03
X383    1.660732e-03
X384    4.750593e-04
X385    1.423823e-03
Length: 370, dtype: float64
```

In [13]:
```
# Find out the variance is equal to zero for any columns
(train_df.var(numeric_only=True) == 0)
```

Out[13]:
```
ID      False
y       False
X10     False
X11      True
X12     False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 370, dtype: bool
```

In [15]:
```
(train_df.var(numeric_only=True) == 0).values
```

Out[15]:
```
array([False, False, False,  True, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False,  True, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False,  True,  True, False, False,
        True, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
```

```
          False])
```

In [25]:

```
variance_with_zero = train_df.var(numeric_only=True)[train_df.var(numeric_only=True)==0].index.va
variance_with_zero
```

Out[25]:

```
array([], dtype=object)
```

In [27]:

```
# Drop zero variance variables
train_df = train_df.drop(variance_with_zero, axis=1)
```

In [28]:

```
print(train_df.shape)
```

```
(4209, 366)
```

In [29]:

```
# As ID column is irrelevant for our prediction hence we drop this column
train_df = train_df.drop(('ID'), axis=1)
```

In [30]:

```
train_df.head()
```

Out[30]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X
0	130.81	k	v	at	a	d	u	j	o	0	...	0	0	1	0	0	0	0	0	0	
1	88.53	k	t	av	e	d	y	l	o	0	...	1	0	0	0	0	0	0	0	0	
2	76.26	az	w	n	c	d	x	j	x	0	...	0	0	0	0	0	0	1	0	0	
3	80.62	az	t	n	f	d	x	l	e	0	...	0	0	0	0	0	0	0	0	0	
4	78.02	az	v	n	f	d	h	d	n	0	...	0	0	0	0	0	0	0	0	0	

5 rows × 365 columns

In [31]:

```
#Check for null and unique values for test and train sets.
train_df.isnull().sum().values
```

Out[31]:

```
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])
```

In [32]:

```
train_df.isnull().any()
```

Out[32]:

```
y       False
X0      False
X1      False
X2      False
X3      False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
```

```
X385    False
Length: 365, dtype: bool
```

In [33]:

```
test_df.isnull().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, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0])
```

In [34]:

```
# Find unique records
train_df.nunique()
```

Out[34]:

```
y       2545
X0        47
X1        27
X2        44
X3         7
        ... 
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 365, dtype: int64
```

In [37]:

```
#Filter out the columns having object datatype
object_datatypes = train_df.select_dtypes(include=[object])
object_datatypes
```

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

4209 rows × 8 columns

In [38]:

```
object_datatype_columns = object_datatypes.columns
object_datatype_columns
```

Out[38]: Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype='object')

In [39]:
```
#Apply label encoder.
# Initialize Label Encoder object
label_encoder = preprocessing.LabelEncoder()
train_df['X0'].unique()
```

Out[39]:
```
array(['k', 'az', 't', 'al', 'o', 'w', 'j', 'h', 's', 'n', 'ay', 'f', 'x',
       'y', 'aj', 'ak', 'am', 'z', 'q', 'at', 'ap', 'v', 'af', 'a', 'e',
       'ai', 'd', 'aq', 'c', 'aa', 'ba', 'as', 'i', 'r', 'b', 'ax', 'bc',
       'u', 'ad', 'au', 'm', 'l', 'aw', 'ao', 'ac', 'g', 'ab'],
      dtype=object)
```

In [42]:
```
# Encode and transform object data to interger
train_df['X0'] = label_encoder.fit_transform(train_df['X0'])
```

In [43]:
```
train_df['X0'].unique()
```

Out[43]:
```
array([32, 20, 40,  9, 36, 43, 31, 29, 39, 35, 19, 27, 44, 45,  7,  8, 10,
       46, 37, 15, 12, 42,  5,  0, 26,  6, 25, 13, 24,  1, 22, 14, 30, 38,
       21, 18, 23, 41,  4, 16, 34, 33, 17, 11,  3, 28,  2])
```

In [46]:
```
# Apply same for all columns having object type data
train_df['X1'] = label_encoder.fit_transform(train_df['X1'])
train_df['X2'] = label_encoder.fit_transform(train_df['X2'])
train_df['X3'] = label_encoder.fit_transform(train_df['X2'])
train_df['X4'] = label_encoder.fit_transform(train_df['X3'])
train_df['X5'] = label_encoder.fit_transform(train_df['X4'])
train_df['X6'] = label_encoder.fit_transform(train_df['X5'])
train_df['X8'] = label_encoder.fit_transform(train_df['X8'])
```

In [47]:
```
train_df.head()
```

Out[47]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X
0	130.81	32	23	17	17	17	17	17	14	0	...	0	0	1	0	0	0	0	0	0	
1	88.53	32	21	19	19	19	19	19	14	0	...	1	0	0	0	0	0	0	0	0	
2	76.26	20	24	34	34	34	34	34	23	0	...	0	0	0	0	0	0	1	0	0	
3	80.62	20	21	34	34	34	34	34	4	0	...	0	0	0	0	0	0	0	0	0	
4	78.02	20	23	34	34	34	34	34	13	0	...	0	0	0	0	0	0	0	0	0	

5 rows × 365 columns

In [49]:
```
#Perform dimensionality reduction (PCA)
from sklearn.decomposition import PCA
# PCA with 958
sklearn_pca = PCA(n_components=0.95)

sklearn_pca.fit(train_df)
```

Out[49]: PCA(n_components=0.95)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [50]:
```
x_train_transformed = sklearn_pca.transform(train_df)
```

In [51]:
```
print(x_train_transformed.shape)
```

```
(4209, 5)
```

In [52]:

```
# PCA with 988
sklearn_pca_98 = PCA(n_components=0.98)
```

In [53]:

```
sklearn_pca_98.fit(train_df)
```

Out[53]:

```
PCA(n_components=0.98)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [54]:

```
x_train_transformed_98 = sklearn_pca_98.transform(train_df)
print(x_train_transformed_98.shape)
```

```
(4209, 5)
```

In [56]:

```
train_df.y
```

Out[56]:

```
0       130.81
1        88.53
2        76.26
3        80.62
4        78.02
         ...
4204    107.39
4205    108.77
4206    109.22
4207     87.48
4208    110.85
Name: y, Length: 4209, dtype: float64
```

In [57]:

```
# Train and Test split on Train dataset
X = train_df.drop('y', axis=1)
y = train_df.y
xtrain,xtest,ytrain,ytest = train_test_split(X,y, test_size=0.3, random_state=42)
```

In [58]:

```
print (xtrain)
print (xtrain.shape)
```

```
      X0  X1  X2  X3  X4  X5  X6  X8  X10  X12  ...  X375  X376  X377  X378  \
370   35  13  16  16  16  16  16  19    0    0  ...     0     0     0     0
3392  15  10  16  16  16  16  16  16    0    0  ...     0     0     1     0
2208  31   3  16  16  16  16  16  21    0    0  ...     0     0     1     0
3942  35  20   8   8   8   8   8  14    0    1  ...     1     0     0     0
1105  36  13  16  16  16  16  16   0    0    0  ...     0     0     0     0
...   ..  ..  ..  ..  ..  ..  ..  ..  ...  ...  ...   ...   ...   ...   ...
3444  31  10  16  16  16  16  16  17    0    0  ...     0     0     1     0
466   20  25  25  25  25  25  25   9    0    0  ...     0     0     0     0
3092  45  24   3   3   3   3   3   2    0    0  ...     1     0     0     0
3772  45  19   8   8   8   8   8   1    0    1  ...     0     0     0     0
860   22   1   7   7   7   7   7  17    0    0  ...     1     0     0     0

      X379  X380  X382  X383  X384  X385
370      0     0     0     0     0     0
3392     0     0     0     0     0     0
2208     0     0     0     0     0     0
3942     0     0     0     0     0     0
1105     0     0     0     0     0     0
...    ...   ...   ...   ...   ...   ...
3444     0     0     0     0     0     0
466      0     0     1     0     0     0
3092     0     0     0     0     0     0
3772     0     0     0     0     0     0
860      0     0     0     0     0     0

[2946 rows x 364 columns]
(2946, 364)
```

In [60]:

In [60]:

```
print(ytrain)
print(ytrain.shape)
```

```
370      95.13
3392    117.36
2208    109.01
3942     93.77
1105    103.41
         ...
3444    109.42
466      78.25
3092     92.18
3772     91.92
860      87.71
Name: y, Length: 2946, dtype: float64
(2946,)
```

In [61]:

```
print (xtest)
print (xtest.shape)
```

```
      X0  X1  X2  X3  X4  X5  X6  X8  X10  X12  ...  X375  X376  X377  X378  \
1073   9  16   7   7   7   7   7  11    0    0  ...     0     0     0     0
144   27  13   3   3   3   3   3  22    0    0  ...     0     0     0     0
2380  31   1  21  21  21  21  21  14    1    0  ...     1     0     0     0
184   20  25  22  22  22  22  22  11    0    0  ...     0     0     0     0
2587   8  23   8   8   8   8   8  17    0    0  ...     0     0     0     0
...   ..  ..  ..  ..  ..  ..  ..  ..  ...  ...  ...   ...   ...   ...   ...
2493  27  20  16  16  16  16  16   5    0    0  ...     0     0     1     0
3388  40  19  24  24  24  24  24  19    0    0  ...     0     0     0     0
3997  22   3   7   7   7   7   7  18    0    0  ...     0     0     1     0
383   40   1  16  16  16  16  16   0    0    0  ...     1     0     0     0
3364  27   4  33  33  33  33  33  24    0    0  ...     0     0     1     0

      X379  X380  X382  X383  X384  X385
1073     0     0     0     0     0     0
144      0     0     0     0     0     0
2380     0     0     0     0     0     0
184      0     0     1     0     0     0
2587     0     0     0     0     0     0
...    ...   ...   ...   ...   ...   ...
2493     0     0     0     0     0     0
3388     0     0     0     0     0     0
3997     0     0     0     0     0     0
383      0     0     0     0     0     0
3364     0     0     0     0     0     0

[1263 rows x 364 columns]
(1263, 364)
```

In [62]:

```
# PCA with 958 for xtrain
pca_xtrain = PCA(n_components=0.95)
pca_xtrain.fit(xtrain)
```

Out[62]:

```
PCA(n_components=0.95)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [64]:

```
pca_xtrain_transformed = pca_xtrain.transform(xtrain)
print(pca_xtrain_transformed.shape)
```

```
(2946, 4)
```

In [66]:

```
#PCA with 958 for xtest
pca_xtest = PCA(n_components=0.95)
pca_xtest.fit(xtest)
```

Out[66]:

```
PCA(n_components=0.95)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [67]:

```
pca_xtest_transformed = pca_xtest.transform(xtest)
print (pca_xtest_transformed.shape)
```

```
(1263, 4)
```

In [69]:

```
print (pca_xtest.explained_variance_)
print (pca_xtest.explained_variance_ratio_)
```

```
[626.87462873 195.51684495  62.32915791  48.49870614]
[0.65711093 0.20494729 0.06533551 0.05083796]
```

In [70]:

```
#PCA for test df dataset
test_df
```

Out[70]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8410	aj	h	as	f	d	aa	j	e	0	...	0	0	0	0	0	0	0	0	0
4205	8411	t	aa	ai	d	d	aa	j	y	0	...	0	1	0	0	0	0	0	0	0
4206	8413	y	v	as	f	d	aa	d	w	0	...	0	0	0	0	0	0	0	0	0
4207	8414	ak	v	as	a	d	aa	c	q	0	...	0	0	1	0	0	0	0	0	0
4208	8416	t	aa	ai	c	d	aa	g	r	0	...	1	0	0	0	0	0	0	0	0

4209 rows × 377 columns

In [71]:

```
test_object_datatypes = test_df.select_dtypes(include=[object])
test_object_datatypes
```

Out[71]:

	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 [72]:

```
test_df['X0'] = label_encoder.fit_transform(test_df['X0'])
test_df['X1'] = label_encoder.fit_transform(test_df['X1'])
test_df['X2'] = label_encoder.fit_transform(test_df['X2'])
test_df['X3'] = label_encoder.fit_transform(test_df['X3'])
```

```
test_df['X4'] = label_encoder.fit_transform(test_df['X4'])
test_df['X5'] = label_encoder.fit_transform(test_df['X5'])
test_df['X6'] = label_encoder.fit_transform(test_df['X6'])
test_df['X8'] = label_encoder.fit_transform(test_df['X8'])
```

In [73]:

```
print(test_df)
print(test_df.shape)
```

```
        ID  X0  X1  X2  X3  X4  X5  X6  X8  X10  ...  X375  X376  X377  X378  \
0        1  21  23  34   5   3  26   0  22    0  ...     0     0     0     1
1        2  42   3   8   0   3   9   6  24    0  ...     0     0     1     0
2        3  21  23  17   5   3   0   9   9    0  ...     0     0     0     1
3        4  21  13  34   5   3  31  11  13    0  ...     0     0     0     1
4        5  45  20  17   2   3  30   8  12    0  ...     1     0     0     0
...    ...  ..  ..  ..  ..  ..  ..  ..  ..  ...  ...   ...   ...   ...   ...
4204  8410   6   9  17   5   3   1   9   4    0  ...     0     0     0     0
4205  8411  42   1   8   3   3   1   9  24    0  ...     0     1     0     0
4206  8413  47  23  17   5   3   1   3  22    0  ...     0     0     0     0
4207  8414   7  23  17   0   3   1   2  16    0  ...     0     0     1     0
4208  8416  42   1   8   2   3   1   6  17    0  ...     1     0     0     0

      X379  X380  X382  X383  X384  X385
0        0     0     0     0     0     0
1        0     0     0     0     0     0
2        0     0     0     0     0     0
3        0     0     0     0     0     0
4        0     0     0     0     0     0
...    ...   ...   ...   ...   ...   ...
4204     0     0     0     0     0     0
4205     0     0     0     0     0     0
4206     0     0     0     0     0     0
4207     0     0     0     0     0     0
4208     0     0     0     0     0     0

[4209 rows x 377 columns]
(4209, 377)
```

In [74]:

```
test_df = test_df.drop('ID',axis=1)
```

In [75]:

```
# PCA with 958 for test df
pca_test_df = PCA(n_components=0.95)
pca_test_df.fit(test_df)
```

Out[75]:

```
PCA(n_components=0.95)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [80]:

```
pca_test_df_transformed = pca_test_df.transform(test_df)
print (pca_test_df_transformed.shape)
```

```
(4209, 6)
```

In [79]:

```
print(pca_test_df.explained_variance_)
print(pca_test_df.explained_variance_ratio_)
```

```
[247.07875325 100.33535335  77.48364816  62.33258307  48.95689653
   8.14203723]
[0.43515102 0.17670897 0.13646292 0.10977912 0.08622208 0.01433962]
```

In [81]:

```
y
```

Out[81]:

```
0       130.81
1        88.53
2        76.26
3        80.62
4        78.02
```

```
         ...
4204    107.39
4205    108.77
4206    109.22
4207     87.48
4208    110.85
Name: y, Length: 4209, dtype: float64
```

In [92]:

```
#Perform XGboost
from sklearn import svm
from sklearn import model_selection
import xgboost as xgb
```

In [98]:

```
from xgboost import XGBRegressor
xgb = XGBRegressor()
xgb.fit(xtrain, ytrain)
```

Out[98]:

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [97]:

```
pred = xgb.predict(xtest)
pred
```

Out[97]:

```
array([ 93.334366,  94.90043 , 114.082596, ...,  88.13724 ,  96.5509  ,
       111.94551 ], dtype=float32)
```

In [99]:

```
df_res = pd.DataFrame(pred, columns = ["yHat"])
df_res
```

Out[99]:

	yHat
0	93.334366
1	94.900429
2	114.082596
3	77.287384
4	111.437325
...	...
1258	93.598534
1259	95.489334
1260	88.137238
1261	96.550903
1262	111.945511

1263 rows × 1 columns

In [100…

```
df_res.to_csv('submission.csv',index=False)
```