mercedes-benz-sandhya-1

August 24, 2024

[1]:
```
# Import the required libraries
```

[302]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
!pip install xgboost
from xgboost import XGBRegressor
```

```
Requirement already satisfied: xgboost in /opt/anaconda3/lib/python3.11/site-
packages (2.1.1)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.11/site-
packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.11/site-
packages (from xgboost) (1.11.4)
```

[231]:
```
# Read the input datasets train.csv and test.csv
# Print the shapes of the datasets
df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
print(df_train.shape)
print(df_test.shape)
```

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

[70]:
```
# check the data typoe of all elements
print (train_data.dtypes)
```

```
ID        int64
y       float64
X0       object
X1       object
X2       object
         …
X380      int64
X382      int64
X383      int64
```

```
X384      int64
X385      int64
Length: 378, dtype: object
```

[49]:
```
# Check if any null values are present for elements
df_train.isna().sum()
```

[49]:
```
ID      0
y       0
X0      0
X1      0
X2      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 378, dtype: int64
```

[50]:
```
df_train.head()
```

[50]:

	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	

	X380	X382	X383	X384	X385
0	0	0	0	0	0
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

[5 rows x 378 columns]

[51]:
```
df_train.describe()
```

[51]:

	ID	y	X10	X11	X12	\
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	
mean	4205.960798	100.669318	0.013305	0.0	0.075077	
std	2437.608688	12.679381	0.114590	0.0	0.263547	
min	0.000000	72.110000	0.000000	0.0	0.000000	
25%	2095.000000	90.820000	0.000000	0.0	0.000000	
50%	4220.000000	99.150000	0.000000	0.0	0.000000	

```
75%    6314.000000   109.010000     0.000000     0.0     0.000000
max    8417.000000   265.320000     1.000000     0.0     1.000000

                X13          X14          X15          X16          X17  …  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000  …
mean      0.057971     0.428130     0.000475     0.002613     0.007603  …
std       0.233716     0.494867     0.021796     0.051061     0.086872  …
min       0.000000     0.000000     0.000000     0.000000     0.000000  …
25%       0.000000     0.000000     0.000000     0.000000     0.000000  …
50%       0.000000     0.000000     0.000000     0.000000     0.000000  …
75%       0.000000     1.000000     0.000000     0.000000     0.000000  …
max       1.000000     1.000000     1.000000     1.000000     1.000000  …

               X375         X376         X377         X378         X379  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      0.318841     0.057258     0.314802     0.020670     0.009503
std       0.466082     0.232363     0.464492     0.142294     0.097033
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       1.000000     0.000000     1.000000     0.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

               X380         X382         X383         X384         X385
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      0.008078     0.007603     0.001663     0.000475     0.001426
std       0.089524     0.086872     0.040752     0.021796     0.037734
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       0.000000     0.000000     0.000000     0.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

[8 rows x 370 columns]
```

[118]:
```
# dropping column 'ID'
df_train = df_train.drop('ID',axis =1)
```

[119]:
```
df_train.head()
```

[119]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	…	X375	X376	X377	X378	X379	\
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	

```
   X380  X382  X383  X384  X385
0     0     0     0     0     0
1     0     0     0     0     0
2     0     1     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0

[5 rows x 377 columns]
```

[120]:
```
df_train.shape
```

[120]:
```
(4209, 377)
```

[58]:
```
# Separate numerical and categorical data from train dataset
df_cat = df_train.select_dtypes(include = object)
df_num = df_train.select_dtypes(exclude = object)
```

[115]:
```
df_cat.head()
```

[115]:
```
   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
```

[116]:
```
df_num.head()
```

[116]:
```
   X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  …  X375  X376  X377  \
0    0    0    0    1    0    0    0    0    1    0  …     0     0     1
1    0    0    0    0    0    0    0    0    1    0  …     1     0     0
2    0    0    0    0    0    0    0    1    0    0  …     0     0     0
3    0    0    0    0    0    0    0    0    0    0  …     0     0     0
4    0    0    0    0    0    0    0    0    0    0  …     0     0     0

   X378  X379  X380  X382  X383  X384  X385
0     0     0     0     0     0     0     0
1     0     0     0     0     0     0     0
2     0     0     0     1     0     0     0
3     0     0     0     0     0     0     0
4     0     0     0     0     0     0     0

[5 rows x 368 columns]
```

[61]:
```
#drop dependent variable from num data of train set
df_num =df_num.drop('y',axis=1)
```

[62]:
```
df_num.head()
```

[62]:
```
   X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  …  X375  X376  X377  \
0    0    0    0    1    0    0    0    0    1    0  …     0     0     1
1    0    0    0    0    0    0    0    0    1    0  …     1     0     0
2    0    0    0    0    0    0    0    1    0    0  …     0     0     0
3    0    0    0    0    0    0    0    0    0    0  …     0     0     0
4    0    0    0    0    0    0    0    0    0    0  …     0     0     0

   X378  X379  X380  X382  X383  X384  X385
0     0     0     0     0     0     0     0
1     0     0     0     0     0     0     0
2     0     0     0     1     0     0     0
3     0     0     0     0     0     0     0
4     0     0     0     0     0     0     0

[5 rows x 368 columns]
```

[137]:
```
df_num.shape
```

[137]:
```
(4209, 368)
```

[138]:
```
columns = df_num.columns
columns
```

[138]:
```
Index(['X10', 'X11', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=368)
```

[88]:
```
# Applying minmax scaling to train dataset
from sklearn.preprocessing import MinMaxScaler,StandardScaler
```

[89]:
```
mn=MinMaxScaler()
```

[90]:
```
df_mn = mn.fit_transform(df_num)
```

[91]:
```
df_mn
```

[91]:
```
array([[0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       …,
       [0., 0., 1., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.]])
```

[92]:
```
df_num_sc = pd.DataFrame(df_mn, index=df_num.index, columns=df_num.columns)
df_num_sc.head()
```

[92]:

	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	…	X375	X376	X377	\
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	…	0.0	0.0	1.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	…	1.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	…	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	

	X378	X379	X380	X382	X383	X384	X385
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0

[5 rows x 368 columns]

[77]:
```
# We see that after performimg scaling, the values are ranging from 0 tO 1
```

Q No:1 - If for any column(s), the variance is equal to zero, then you need to remove those variable(s).

[93]:
```
# Finding the variance of numerical data
df_variance = df_num.var()
```

[79]:
```
df_variance
```

[79]:
```
X10     0.013131
X11     0.000000
X12     0.069457
X13     0.054623
X14     0.244893
          …
X380    0.008015
X382    0.007547
X383    0.001661
X384    0.000475
X385    0.001424
Length: 368, dtype: float64
```

[99]:
```
# finding the count of zero variance
variable_var_zero = [ ]
count =0
for i in range(0,len(df_variance)):
```

```
    if df_variance[i]==0: #checking if the variance for the df_num dataframe␣
 ↪column has zero
        variable_var_zero.append(columns[i])
        count=count+1
print('No of columns with zero variance = ',count)
```

```
No of columns with zero variance =  12
```

```
/var/folders/tq/s28mvypn1cb1fp3bvs6nghxh0000gn/T/ipykernel_94602/1799531437.py:5
: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In
a future version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if df_variance[i]==0: #checking if the variance for the df_num dataframe
column has zero
```

[100]:
```
np.ravel(variable_var_zero)
```

[100]:
```
array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'], dtype='<U4')
```

[107]:
```
# We have to drop those columns which have zero variance
df_variance_drop_zero_var = df_num.drop(['X11', 'X93', 'X107', 'X233', 'X235',␣
 ↪'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'],axis=1)
```

[109]:
```
# printing dataset after dropping columns with zero variance
df_variance_drop_zero_var.head()
```

[109]:

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	…	X375	X376	X377	\
0	0	0	1	0	0	0	0	1	0	0	…	0	0	1	
1	0	0	0	0	0	0	0	1	0	0	…	1	0	0	
2	0	0	0	0	0	0	1	0	0	0	…	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	…	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	…	0	0	0	

	X378	X379	X380	X382	X383	X384	X385
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

[5 rows x 356 columns]

[110]:
```
df_variance_drop_zero_var.describe()
```

[110]:

	X10	X12	X13	X14	X15	\
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	
mean	0.013305	0.075077	0.057971	0.428130	0.000475	
std	0.114590	0.263547	0.233716	0.494867	0.021796	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	1.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

	X16	X17	X18	X19	X20	…	\
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	…	
mean	0.002613	0.007603	0.007840	0.099549	0.142789	…	
std	0.051061	0.086872	0.088208	0.299433	0.349899	…	
min	0.000000	0.000000	0.000000	0.000000	0.000000	…	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	…	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	…	
75%	0.000000	0.000000	0.000000	0.000000	0.000000	…	
max	1.000000	1.000000	1.000000	1.000000	1.000000	…	

	X375	X376	X377	X378	X379	\
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	
mean	0.318841	0.057258	0.314802	0.020670	0.009503	
std	0.466082	0.232363	0.464492	0.142294	0.097033	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	1.000000	0.000000	1.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

	X380	X382	X383	X384	X385
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000
mean	0.008078	0.007603	0.001663	0.000475	0.001426
std	0.089524	0.086872	0.040752	0.021796	0.037734
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

[8 rows x 356 columns]

[112]:
```
df_variance_drop_zero_var.shape
```

[112]: (4209, 356)

df_train.shape

[122]:
```
df_cat.shape
```

[122]:
```
(4209, 8)
```

Q No: 2 - Check for null and unique values for test and train sets.

[130]:
```
# Checking for null values in train and test datasets
df_train.isna().sum()
```

[130]:
```
y       0
X0      0
X1      0
X2      0
X3      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 377, dtype: int64
```

[125]:
```
df_test.isna().sum()
```

[125]:
```
ID      0
X0      0
X1      0
X2      0
X3      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 377, dtype: int64
```

[126]:
```
# Checking for number of unique values
df_train.nunique()
```

[126]:
```
y       2545
X0        47
X1        27
X2        44
X3         7
       …
X380       2
```

```
X382       2
X383       2
X384       2
X385       2
Length: 377, dtype: int64
```

[132]:
```
df_train.head()
```

[132]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	…	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	130.81	k	v	at	a	d	u	j	o	0	…	0	0	1	0	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	0
2	76.26	az	w	n	c	d	x	j	x	0	…	0	0	0	0	0	0	1	0	0	0
3	80.62	az	t	n	f	d	x	l	e	0	…	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	0

[5 rows x 377 columns]

[145]:
```
tra_feature_names =df_train.columns.values.ravel()
tra_feature_names
```

[145]:
```
array(['y', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10', 'X11',
       'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19', 'X20',
       'X21', 'X22', 'X23', 'X24', 'X26', 'X27', 'X28', 'X29', 'X30',
       'X31', 'X32', 'X33', 'X34', 'X35', 'X36', 'X37', 'X38', 'X39',
       'X40', 'X41', 'X42', 'X43', 'X44', 'X45', 'X46', 'X47', 'X48',
       'X49', 'X50', 'X51', 'X52', 'X53', 'X54', 'X55', 'X56', 'X57',
       'X58', 'X59', 'X60', 'X61', 'X62', 'X63', 'X64', 'X65', 'X66',
       'X67', 'X68', 'X69', 'X70', 'X71', 'X73', 'X74', 'X75', 'X76',
       'X77', 'X78', 'X79', 'X80', 'X81', 'X82', 'X83', 'X84', 'X85',
       'X86', 'X87', 'X88', 'X89', 'X90', 'X91', 'X92', 'X93', 'X94',
       'X95', 'X96', 'X97', 'X98', 'X99', 'X100', 'X101', 'X102', 'X103',
       'X104', 'X105', 'X106', 'X107', 'X108', 'X109', 'X110', 'X111',
       'X112', 'X113', 'X114', 'X115', 'X116', 'X117', 'X118', 'X119',
       'X120', 'X122', 'X123', 'X124', 'X125', 'X126', 'X127', 'X128',
       'X129', 'X130', 'X131', 'X132', 'X133', 'X134', 'X135', 'X136',
       'X137', 'X138', 'X139', 'X140', 'X141', 'X142', 'X143', 'X144',
       'X145', 'X146', 'X147', 'X148', 'X150', 'X151', 'X152', 'X153',
       'X154', 'X155', 'X156', 'X157', 'X158', 'X159', 'X160', 'X161',
       'X162', 'X163', 'X164', 'X165', 'X166', 'X167', 'X168', 'X169',
       'X170', 'X171', 'X172', 'X173', 'X174', 'X175', 'X176', 'X177',
```

```
'X178', 'X179', 'X180', 'X181', 'X182', 'X183', 'X184', 'X185',
'X186', 'X187', 'X189', 'X190', 'X191', 'X192', 'X194', 'X195',
'X196', 'X197', 'X198', 'X199', 'X200', 'X201', 'X202', 'X203',
'X204', 'X205', 'X206', 'X207', 'X208', 'X209', 'X210', 'X211',
'X212', 'X213', 'X214', 'X215', 'X216', 'X217', 'X218', 'X219',
'X220', 'X221', 'X222', 'X223', 'X224', 'X225', 'X226', 'X227',
'X228', 'X229', 'X230', 'X231', 'X232', 'X233', 'X234', 'X235',
'X236', 'X237', 'X238', 'X239', 'X240', 'X241', 'X242', 'X243',
'X244', 'X245', 'X246', 'X247', 'X248', 'X249', 'X250', 'X251',
'X252', 'X253', 'X254', 'X255', 'X256', 'X257', 'X258', 'X259',
'X260', 'X261', 'X262', 'X263', 'X264', 'X265', 'X266', 'X267',
'X268', 'X269', 'X270', 'X271', 'X272', 'X273', 'X274', 'X275',
'X276', 'X277', 'X278', 'X279', 'X280', 'X281', 'X282', 'X283',
'X284', 'X285', 'X286', 'X287', 'X288', 'X289', 'X290', 'X291',
'X292', 'X293', 'X294', 'X295', 'X296', 'X297', 'X298', 'X299',
'X300', 'X301', 'X302', 'X304', 'X305', 'X306', 'X307', 'X308',
'X309', 'X310', 'X311', 'X312', 'X313', 'X314', 'X315', 'X316',
'X317', 'X318', 'X319', 'X320', 'X321', 'X322', 'X323', 'X324',
'X325', 'X326', 'X327', 'X328', 'X329', 'X330', 'X331', 'X332',
'X333', 'X334', 'X335', 'X336', 'X337', 'X338', 'X339', 'X340',
'X341', 'X342', 'X343', 'X344', 'X345', 'X346', 'X347', 'X348',
'X349', 'X350', 'X351', 'X352', 'X353', 'X354', 'X355', 'X356',
'X357', 'X358', 'X359', 'X360', 'X361', 'X362', 'X363', 'X364',
'X365', 'X366', 'X367', 'X368', 'X369', 'X370', 'X371', 'X372',
'X373', 'X374', 'X375', 'X376', 'X377', 'X378', 'X379', 'X380',
'X382', 'X383', 'X384', 'X385'], dtype=object)
```

[148]:
```
train_feature_names= df_train[tra_feature_names].values.ravel()
train_feature_names
```

[148]:
```
array([130.81, 'k', 'v', …, 0, 0, 0], dtype=object)
```

[147]:
```
# Finding unique values
train_unique_values =  pd.unique(train_feature_names)
train_unique_values
```

[147]:
```
array([130.81, 'k', 'v', …, 85.71, 108.77, 87.48], dtype=object)
```

Q no:3 - Apply label encoder.

[149]:
```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(handle_unknown = "ignore")
```

[152]:
```
df_cat_dum = ohe.fit_transform(df_cat).toarray()
col_names = ohe.get_feature_names_out()
col_names = np.array(col_names).ravel()
df_cat_oh  =pd.DataFrame(df_cat_dum, columns=col_names)
```

```
df_cat_oh.head()
```

[152]:
```
   X0_a  X0_aa  X0_ab  X0_ac  X0_ad  X0_af  X0_ai  X0_aj  X0_ak  X0_al  … \
0   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …
1   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …
2   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …
3   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …
4   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …

   X8_p  X8_q  X8_r  X8_s  X8_t  X8_u  X8_v  X8_w  X8_x  X8_y
0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0
3   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[5 rows x 195 columns]
```

[153]:
```
df_cat_oh.shape
```

[153]:
```
(4209, 195)
```

[154]:
```
#concatenating categorical and numerical data into single dataframe of training␣
 ↪data
df_train_final = pd.concat([df_variance_drop_zero_var, df_cat_oh], axis = 1)
```

[156]:
```
df_train_final.head()
```

[156]:
```
   X10  X12  X13  X14  X15  X16  X17  X18  X19  X20  …  X8_p  X8_q  X8_r  \
0    0    0    1    0    0    0    0    1    0    0  …   0.0   0.0   0.0
1    0    0    0    0    0    0    0    1    0    0  …   0.0   0.0   0.0
2    0    0    0    0    0    0    1    0    0    0  …   0.0   0.0   0.0
3    0    0    0    0    0    0    0    0    0    0  …   0.0   0.0   0.0
4    0    0    0    0    0    0    0    0    0    0  …   0.0   0.0   0.0

   X8_s  X8_t  X8_u  X8_v  X8_w  X8_x  X8_y
0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2   0.0   0.0   0.0   0.0   0.0   1.0   0.0
3   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[5 rows x 551 columns]
```

[157]:
```
df_train_final.shape
```

[157]:
```
(4209, 551)
```

Q No: 4 - Perform dimensionality reduction.

[158]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=24)
```

[159]:
```
pca
```

[159]:
```
PCA(n_components=24)
```

[160]:
```
df_train.dtypes
```

[160]:
```
y       float64
X0       object
X1       object
X2       object
X3       object
         …
X380      int64
X382      int64
X383      int64
X384      int64
X385      int64
Length: 377, dtype: object
```

[164]:
```
x_pca = pca.fit_transform(df_train_final)
```

[165]:
```
x_pca
```

[165]:
```
array([[ 0.85024767, -1.25251472,  2.02164021, …,  1.12174192,
        -0.21244172,  1.30038277],
       [-0.10930195, -1.29966202, -0.04580101, …,  0.62511141,
        -0.33634916,  1.36375534],
       [-0.67365271, -2.36769682,  1.78779241, …, -0.55332572,
         0.55589807,  0.6730131 ],
       …,
       [-0.99490673, -0.37444814,  1.86208763, …,  0.21099725,
        -0.05303558, -0.47433803],
       [ 0.37556312, -1.29070716, -3.05746746, …,  0.32820392,
        -0.59452102, -0.25658008],
       [ 0.9521682 , -0.94658739, -0.8812571 , …,  0.35460966,
        -0.72878003, -0.78151516]])
```

[166]:
```
df_train_final.shape
```

[166]:
```
(4209, 551)
```

[167]:
```
df_pca = pd.DataFrame(x_pca)
```

[169]:
```
df_pca.head()
```

[169]:
```
          0         1         2         3         4         5         6   \
0  0.850248 -1.252515  2.021640  0.865224  1.592171 -0.056847  0.563840
1 -0.109302 -1.299662 -0.045801 -0.796931  0.277976  0.140880  1.108070
2 -0.673653 -2.367697  1.787792  2.345645  0.356806  3.753878 -1.188809
3 -0.480940 -2.695789  0.524340  2.881771 -0.485304  3.765186 -0.307380
4 -0.516369 -2.692792  0.334140  3.103397 -0.723453  3.866238 -0.451954

          7         8         9   …        14        15        16        17  \
0 -1.030709  0.205178 -0.264531  …  0.036302  0.296275 -0.518219 -0.474323
1 -0.726633 -0.032188  0.612273  … -0.981884 -0.648002 -0.005564  0.096023
2  0.679650 -0.924715 -0.215843  …  0.295184  0.844538 -0.353463 -0.827058
3 -0.014646 -1.239942  0.254637  …  0.240149  0.359278  0.273777 -0.779260
4  0.151802 -1.801274 -0.298129  … -0.112547 -0.216358 -0.090530 -0.204632

          18        19        20        21        22        23
0 -0.523538  0.411156 -0.336177  1.121742 -0.212442  1.300383
1  0.855982 -0.191009 -0.883710  0.625111 -0.336349  1.363755
2  0.561779  0.590886  0.883524 -0.553326  0.555898  0.673013
3  0.819120  0.624917 -0.352133 -0.308350  0.242219 -0.220344
4  0.415983  0.163549 -0.025904  0.420751  0.340215  0.269334

[5 rows x 24 columns]
```

[170]:
```
pca.explained_variance_ratio_
```

[170]:
```
array([0.11327864, 0.07799109, 0.07358181, 0.05848106, 0.04943089,
       0.04191889, 0.03310021, 0.0282729 , 0.02515469, 0.02153505,
       0.02077602, 0.01725079, 0.01505285, 0.01435205, 0.01385206,
       0.01296764, 0.01205455, 0.01092876, 0.0098421 , 0.00913204,
       0.00883392, 0.00843739, 0.00823173, 0.0077231 ])
```

[232]:
```
# Repeat the above steps for test data set
df_test.head()
```

[232]:
```
   ID  X0 X1  X2 X3 X4 X5 X6 X8  X10  …  X375  X376  X377  X378  X379  X380  \
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

   X382  X383  X384  X385
0     0     0     0     0
1     0     0     0     0
2     0     0     0     0
```

```
3     0     0     0     0
4     0     0     0     0

[5 rows x 377 columns]
```

[233]:
```
# Check for null data in test data set
df_test.isna().sum()
```

[233]:
```
ID      0
X0      0
X1      0
X2      0
X3      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 377, dtype: int64
```

[234]:
```
df_test.nunique()
```

[234]:
```
ID      4209
X0        49
X1        27
X2        45
X3         7
        …
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 377, dtype: int64
```

[235]:
```
df_test = df_test.drop('ID', axis=1)
df_test.head()
```

[235]:
```
   X0 X1  X2 X3 X4 X5 X6 X8  X10  X11  …  X375  X376  X377  X378  X379  \
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

   X380  X382  X383  X384  X385
```

```
0     0     0     0     0     0
1     0     0     0     0     0
2     0     0     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0

[5 rows x 376 columns]
```

[236]:
```
# Unique values for test sets
tes_feature_names =df_test.columns.values.ravel()
tes_feature_names
```

[236]:
```
array(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10', 'X11',
       'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19', 'X20',
       'X21', 'X22', 'X23', 'X24', 'X26', 'X27', 'X28', 'X29', 'X30',
       'X31', 'X32', 'X33', 'X34', 'X35', 'X36', 'X37', 'X38', 'X39',
       'X40', 'X41', 'X42', 'X43', 'X44', 'X45', 'X46', 'X47', 'X48',
       'X49', 'X50', 'X51', 'X52', 'X53', 'X54', 'X55', 'X56', 'X57',
       'X58', 'X59', 'X60', 'X61', 'X62', 'X63', 'X64', 'X65', 'X66',
       'X67', 'X68', 'X69', 'X70', 'X71', 'X73', 'X74', 'X75', 'X76',
       'X77', 'X78', 'X79', 'X80', 'X81', 'X82', 'X83', 'X84', 'X85',
       'X86', 'X87', 'X88', 'X89', 'X90', 'X91', 'X92', 'X93', 'X94',
       'X95', 'X96', 'X97', 'X98', 'X99', 'X100', 'X101', 'X102', 'X103',
       'X104', 'X105', 'X106', 'X107', 'X108', 'X109', 'X110', 'X111',
       'X112', 'X113', 'X114', 'X115', 'X116', 'X117', 'X118', 'X119',
       'X120', 'X122', 'X123', 'X124', 'X125', 'X126', 'X127', 'X128',
       'X129', 'X130', 'X131', 'X132', 'X133', 'X134', 'X135', 'X136',
       'X137', 'X138', 'X139', 'X140', 'X141', 'X142', 'X143', 'X144',
       'X145', 'X146', 'X147', 'X148', 'X150', 'X151', 'X152', 'X153',
       'X154', 'X155', 'X156', 'X157', 'X158', 'X159', 'X160', 'X161',
       'X162', 'X163', 'X164', 'X165', 'X166', 'X167', 'X168', 'X169',
       'X170', 'X171', 'X172', 'X173', 'X174', 'X175', 'X176', 'X177',
       'X178', 'X179', 'X180', 'X181', 'X182', 'X183', 'X184', 'X185',
       'X186', 'X187', 'X189', 'X190', 'X191', 'X192', 'X194', 'X195',
       'X196', 'X197', 'X198', 'X199', 'X200', 'X201', 'X202', 'X203',
       'X204', 'X205', 'X206', 'X207', 'X208', 'X209', 'X210', 'X211',
       'X212', 'X213', 'X214', 'X215', 'X216', 'X217', 'X218', 'X219',
       'X220', 'X221', 'X222', 'X223', 'X224', 'X225', 'X226', 'X227',
       'X228', 'X229', 'X230', 'X231', 'X232', 'X233', 'X234', 'X235',
       'X236', 'X237', 'X238', 'X239', 'X240', 'X241', 'X242', 'X243',
       'X244', 'X245', 'X246', 'X247', 'X248', 'X249', 'X250', 'X251',
       'X252', 'X253', 'X254', 'X255', 'X256', 'X257', 'X258', 'X259',
       'X260', 'X261', 'X262', 'X263', 'X264', 'X265', 'X266', 'X267',
       'X268', 'X269', 'X270', 'X271', 'X272', 'X273', 'X274', 'X275',
       'X276', 'X277', 'X278', 'X279', 'X280', 'X281', 'X282', 'X283',
       'X284', 'X285', 'X286', 'X287', 'X288', 'X289', 'X290', 'X291',
       'X292', 'X293', 'X294', 'X295', 'X296', 'X297', 'X298', 'X299',
```

```
       'X300', 'X301', 'X302', 'X304', 'X305', 'X306', 'X307', 'X308',
       'X309', 'X310', 'X311', 'X312', 'X313', 'X314', 'X315', 'X316',
       'X317', 'X318', 'X319', 'X320', 'X321', 'X322', 'X323', 'X324',
       'X325', 'X326', 'X327', 'X328', 'X329', 'X330', 'X331', 'X332',
       'X333', 'X334', 'X335', 'X336', 'X337', 'X338', 'X339', 'X340',
       'X341', 'X342', 'X343', 'X344', 'X345', 'X346', 'X347', 'X348',
       'X349', 'X350', 'X351', 'X352', 'X353', 'X354', 'X355', 'X356',
       'X357', 'X358', 'X359', 'X360', 'X361', 'X362', 'X363', 'X364',
       'X365', 'X366', 'X367', 'X368', 'X369', 'X370', 'X371', 'X372',
       'X373', 'X374', 'X375', 'X376', 'X377', 'X378', 'X379', 'X380',
       'X382', 'X383', 'X384', 'X385'], dtype=object)
```

[237]:
```
test_feature_names= df_train[tes_feature_names].values.ravel()
test_feature_names
```

[237]:
```
array(['k', 'v', 'at', …, 0, 0, 0], dtype=object)
```

[238]:
```
df_test.shape
```

[238]:
```
(4209, 376)
```

[239]:
```
df_test.dtypes
```

[239]:
```
X0      object
X1      object
X2      object
X3      object
X4      object
         …
X380     int64
X382     int64
X383     int64
X384     int64
X385     int64
Length: 376, dtype: object
```

[240]:
```
# Seperate the numerical and categorical columns for test data
test_df_cat = df_test.select_dtypes(include = object)
test_df_num = df_test.select_dtypes(exclude = object)
```

[241]:
```
test_df_cat.head()
```

[241]:

	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
```

[242]:
```
test_df_num.head()
```

[242]:
```
   X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  …  X375  X376  X377  \
0    0    0    0    0    0    0    0    0    0    0  …     0     0     0
1    0    0    0    0    0    0    0    0    0    1  …     0     0     1
2    0    0    0    0    1    0    0    0    0    0  …     0     0     0
3    0    0    0    0    0    0    0    0    0    0  …     0     0     0
4    0    0    0    0    1    0    0    0    0    0  …     1     0     0

   X378  X379  X380  X382  X383  X384  X385
0     1     0     0     0     0     0     0
1     0     0     0     0     0     0     0
2     1     0     0     0     0     0     0
3     1     0     0     0     0     0     0
4     0     0     0     0     0     0     0

[5 rows x 368 columns]
```

[243]:
```
test_df_num.shape
```

[243]:
```
(4209, 368)
```

[244]:
```
test_columns = test_df_num.columns
test_columns
```

[244]:
```
Index(['X10', 'X11', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=368)
```

[245]:
```
#Apply scaling for test data set
test_df_num_sc = mn.transform(test_df_num)
```

[246]:
```
test_df_num_df = pd.DataFrame(test_df_num_sc, index = test_df_num.index,␣
 ↪columns=test_df_num.columns)
```

[247]:
```
test_df_num_df.head()
```

[247]:
```
   X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  …  X375  X376  X377  \
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …   0.0   0.0   0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  …   0.0   0.0   1.0
2  0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  …   0.0   0.0   0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …   0.0   0.0   0.0
4  0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  …   1.0   0.0   0.0
```

```
   X378  X379  X380  X382  X383  X384  X385
0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
1   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2   1.0   0.0   0.0   0.0   0.0   0.0   0.0
3   1.0   0.0   0.0   0.0   0.0   0.0   0.0
4   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[5 rows x 368 columns]
```

We can see that after scaling, the values have become in the range of 0 to 1

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

[248]:
```
test_variance_df_num = test_df_num.var()
```

[249]:
```
test_variable_var_zero = [ ]
cntr =0
```

[250]:
```
for i in range(0,len(test_variance_df_num)):
        if test_variance_df_num[i]==0: #checking if the variance for the df_num␣
 ↪dataframe column has zero
                    test_variable_var_zero.append(test_columns[i])
                    cntr=cntr+1
```

```
/var/folders/tq/s28mvypn1cb1fp3bvs6nghxh0000gn/T/ipykernel_94602/3262422608.py:2
: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In
a future version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if test_variance_df_num[i]==0: #checking if the variance for the df_num
dataframe column has zero
```

[251]:
```
print('No of columns with zero variance = ',count)
```

```
No of columns with zero variance =  12
```

[252]:
```
np.ravel(test_variable_var_zero)
```

[252]:
```
array(['X257', 'X258', 'X295', 'X296', 'X369'], dtype='<U4')
```

[253]:
```
test_df_num_variance_with_zero_drop = test_df_num.drop(['X257', 'X258', 'X295',␣
 ↪'X296', 'X369'], axis = 1)
```

[254]:
```
test_df_num_variance_with_zero_drop.head()
```

[254]:
```
   X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  …  X375  X376  X377  \
0    0    0    0    0    0    0    0    0    0    0  …     0     0     0
1    0    0    0    0    0    0    0    0    0    1  …     0     0     1
```

```
2    0    0    0    0    1    0    0    0    0    0  …     0     0     0
3    0    0    0    0    0    0    0    0    0    0  …     0     0     0
4    0    0    0    0    1    0    0    0    0    0  …     1     0     0

   X378  X379  X380  X382  X383  X384  X385
0     1     0     0     0     0     0     0
1     0     0     0     0     0     0     0
2     1     0     0     0     0     0     0
3     1     0     0     0     0     0     0
4     0     0     0     0     0     0     0

[5 rows x 363 columns]
```

[255]:
```
test_df_num_variance_with_zero_drop.shape
```

[255]:
```
(4209, 363)
```

Apply one hot encoder on test data

[256]:
```
test_df_cat_dum = ohe.transform(test_df_cat).toarray()
test_col_names = ohe.get_feature_names_out()
```

[257]:
```
test_col_names = np.array(test_col_names).ravel()
```

[258]:
```
test_df_cat_oh  =pd.DataFrame(test_df_cat_dum, columns=test_col_names)
```

[259]:
```
test_df_cat_oh.head()
```

[259]:
```
   X0_a  X0_aa  X0_ab  X0_ac  X0_ad  X0_af  X0_ai  X0_aj  X0_ak  X0_al  … \
0   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …
1   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …
2   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …
3   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …
4   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0  …

   X8_p  X8_q  X8_r  X8_s  X8_t  X8_u  X8_v  X8_w  X8_x  X8_y
0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0
1   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0
2   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
3   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[5 rows x 195 columns]
```

[268]:
```
# concatenating both categorical and numerical values of test set
df_test_final = pd.concat([test_df_num_variance_with_zero_drop,␣
 ↪test_df_cat_oh], axis = 1)
```

[269]:
```
df_test_final.head()
```

[269]:

	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	…	X8_p	X8_q	X8_r	\
0	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
1	0	0	0	0	0	0	0	0	0	1	…	0.0	0.0	0.0	
2	0	0	0	0	1	0	0	0	0	0	…	0.0	0.0	0.0	
3	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
4	0	0	0	0	1	0	0	0	0	0	…	0.0	0.0	0.0	

	X8_s	X8_t	X8_u	X8_v	X8_w	X8_x	X8_y
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0

[5 rows x 558 columns]

[270]:
```
print(df_train_final.shape)
print(df_test_final.shape)
```

```
(4209, 551)
(4209, 558)
```

Apply PCA for test dataset

[271]:
```
# while dropping columns with 0 variance for train and test data sets feature␣
 ↪results are different,
# hence to balance the feature in train and test sets, added dropped dummy␣
 ↪columns with NAN values to apply PCA
# reset the test data features to align with train features
test_df_newdata = df_test_final.reindex(labels=df_train_final.columns,axis=1)
test_df_newdata.head()
```

[271]:

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	…	X8_p	X8_q	X8_r	\
0	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
1	0	0	0	0	0	0	0	0	1	0	…	0.0	0.0	0.0	
2	0	0	0	1	0	0	0	0	0	0	…	0.0	0.0	0.0	
3	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
4	0	0	0	1	0	0	0	0	0	0	…	0.0	0.0	0.0	

	X8_s	X8_t	X8_u	X8_v	X8_w	X8_x	X8_y
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
[5 rows x 551 columns]
```

[285]:
```
# fill NAN values with 0 to fit to PCA
test_df_newdata["X257"] = test_df_newdata["X257"].fillna(0)
test_df_newdata["X258"] = test_df_newdata["X257"].fillna(0)
test_df_newdata["X295"] = test_df_newdata["X257"].fillna(0)
test_df_newdata["X296"] = test_df_newdata["X257"].fillna(0)
test_df_newdata["X369"] = test_df_newdata["X257"].fillna(0)
test_df_newdata.head()
```

[285]:

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	…	X8_p	X8_q	X8_r	\
0	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
1	0	0	0	0	0	0	0	0	1	0	…	0.0	0.0	0.0	
2	0	0	0	1	0	0	0	0	0	0	…	0.0	0.0	0.0	
3	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
4	0	0	0	1	0	0	0	0	0	0	…	0.0	0.0	0.0	

	X8_s	X8_t	X8_u	X8_v	X8_w	X8_x	X8_y
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
[5 rows x 551 columns]
```

[286]:
```
test_x_pca = pca.transform(test_df_newdata)
```

[287]:
```
test_x_pca
```

[287]:
```
array([[-0.40548801, -2.85209672,  0.3627893 , …,  0.24018876,
         0.14670626,  0.06964329],
       [ 3.86471172,  0.46316351,  1.47860722, …,  0.70285221,
        -0.62543936, -0.64940942],
       [-1.32175332, -0.70974177,  0.50255699, …, -0.75870236,
        -0.80091288,  1.11442233],
       …,
       [-1.4499784 ,  0.44860692, -2.93132904, …,  0.54406364,
        -0.40472711,  0.42916979],
       [-2.29187603,  1.37884593,  1.58995814, …,  0.58223029,
        -0.374961  ,  0.08420352],
       [ 2.00966243, -0.97120894, -0.53004386, …,  0.75945756,
         0.46731808,  0.3105639 ]])
```

[288]:
```
# X_train and y Values of train data set
X_train = df_train_final
y_train = df_train['y']
```

[289]:
```
X_train.head()
```

[289]:
```
   X10  X12  X13  X14  X15  X16  X17  X18  X19  X20  …  X8_p  X8_q  X8_r  \
0    0    0    1    0    0    0    0    1    0    0  …   0.0   0.0   0.0
1    0    0    0    0    0    0    0    1    0    0  …   0.0   0.0   0.0
2    0    0    0    0    0    0    1    0    0    0  …   0.0   0.0   0.0
3    0    0    0    0    0    0    0    0    0    0  …   0.0   0.0   0.0
4    0    0    0    0    0    0    0    0    0    0  …   0.0   0.0   0.0

   X8_s  X8_t  X8_u  X8_v  X8_w  X8_x  X8_y
0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2   0.0   0.0   0.0   0.0   0.0   1.0   0.0
3   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[5 rows x 551 columns]
```

[290]:
```
y_train.head()
```

[290]:
```
0    130.81
1     88.53
2     76.26
3     80.62
4     78.02
Name: y, dtype: float64
```

[291]:
```
# X_test values of test data set
X_test = test_df_newdata
```

[292]:
```
X_test.head()
```

[292]:
```
   X10  X12  X13  X14  X15  X16  X17  X18  X19  X20  …  X8_p  X8_q  X8_r  \
0    0    0    0    0    0    0    0    0    0    0  …   0.0   0.0   0.0
1    0    0    0    0    0    0    0    0    1    0  …   0.0   0.0   0.0
2    0    0    0    1    0    0    0    0    0    0  …   0.0   0.0   0.0
3    0    0    0    0    0    0    0    0    0    0  …   0.0   0.0   0.0
4    0    0    0    1    0    0    0    0    0    0  …   0.0   0.0   0.0

   X8_s  X8_t  X8_u  X8_v  X8_w  X8_x  X8_y
0   0.0   0.0   0.0   0.0   1.0   0.0   0.0
1   0.0   0.0   0.0   0.0   0.0   0.0   1.0
2   0.0   0.0   0.0   0.0   0.0   0.0   0.0
3   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[5 rows x 551 columns]
```

Q no:5 - Predict your test_df values using XGBoost.

[305]:
```
xgb = XGBRegressor()
```

[306]:
```
xgb.fit(X_train, y_train)
```

[306]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=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,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, …)
```

[307]:
```
pred = xgb.predict(X_test)
```

[308]:
```
pred
```

[308]:
```
array([ 89.49934 , 116.30191 ,  88.787895, …,  97.76178 , 107.47624 ,
        90.692856], dtype=float32)
```

[309]:
```
df_res = pd.DataFrame(pred, columns = ["yHat"])
df_res
```

[309]:

	yHat
0	89.499336
1	116.301910
2	88.787895
3	76.412582
4	112.421379
…	…
4204	104.103584
4205	93.762901
4206	97.761780
4207	107.476242
4208	90.692856

[4209 rows x 1 columns]

[313]:
```
df_res.to_csv('final result.csv',index=False)
```

[314]:
```
df_sub= pd.read_csv('final result.csv')
```

[315]:
```
df_sub.head()
```

[315]:

	yHat
0	89.499340
1	116.301910
2	88.787895
3	76.412580
4	112.421380

[ ]:
```
The final result after prediction has been moved to 'final result.csv'
```