mercedes benz

December 3, 2022

0.1 Name - Samiksha Borade

0.2 Title - Mercedes-Benz Greener Manufacturing

[ ]:
```
## Importing Libraries
```

[1]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.feature_selection import VarianceThreshold
variance = VarianceThreshold(threshold=0)
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder
```

[2]:
```
import os
os.getcwd()
```

[2]:
```
'C:\\Users\\HP'
```

[3]:
```
# import dataset
```

[4]:
```
train = pd.read_csv(r"C:\Users\HP\Downloads\archive (1)\train.csv")
train.head()       #getting first 5 rows of dataset
```

[4]:

	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]
```

[5]:
```
test = pd.read_csv(r"C:\Users\HP\Downloads\archive (1)\test.csv")
test.head()          # getting first 5 rows of dataset
```

[5]:
```
   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]
```

[6]:
```
train.describe()  # getting summary
```

[6]:
```
                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]
```

[7]:
```
test.describe()    #getting summary
```

[7]:
```
                ID          X10          X11          X12          X13  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean   4211.039202     0.019007     0.000238     0.074364     0.061060
std    2423.078926     0.136565     0.015414     0.262394     0.239468
min       1.000000     0.000000     0.000000     0.000000     0.000000
25%    2115.000000     0.000000     0.000000     0.000000     0.000000
50%    4202.000000     0.000000     0.000000     0.000000     0.000000
75%    6310.000000     0.000000     0.000000     0.000000     0.000000
max    8416.000000     1.000000     1.000000     1.000000     1.000000

               X14          X15          X16          X17          X18  …  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000  …
mean      0.427893     0.000713     0.002613     0.008791     0.010216  …
std       0.494832     0.026691     0.051061     0.093357     0.100570  …
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     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      0.325968     0.049656     0.311951     0.019244     0.011879
std       0.468791     0.217258     0.463345     0.137399     0.108356
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     0.008791     0.000475     0.000713     0.001663
std       0.089524     0.093357     0.021796     0.026691     0.040752
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 369 columns]
```

[10]:
```
train.isnull().sum()       # finding null values
```

[10]:
```
ID      0
y       0
X0      0
X1      0
X2      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 378, dtype: int64
```

[11]:
```
train_target = train["y"]
train_data = train.drop(["y","ID"], axis = 1)    # drop column
```

[12]:
```
train_data.head(5)       # getting first 5 rows
```

[12]:
```
   X0 X1  X2 X3 X4 X5 X6 X8  X10  X11  …  X375  X376  X377  X378  X379  \
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

   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 376 columns]
```

[13]:
```
train_data.var().sort_values().head(15)  # check variance of each data
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11252\2491115096.py:1: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  train_data.var().sort_values().head(15)
```

[13]:
```
X330    0.000000
X297    0.000000
X268    0.000000
X290    0.000000
X235    0.000000
X347    0.000000
X107    0.000000
X233    0.000000
X289    0.000000
X93     0.000000
X11     0.000000
X293    0.000000
X257    0.000238
X207    0.000238
X280    0.000238
dtype: float64
```

[14]:
```
train_data_without_zero_var = variance.fit_transform(train_data.iloc[:,9:])
train_data_without_zero_var           #the variance is equal to zero, then you
 ↪need to remove those variable(s).
```

[14]:
```
array([[0, 1, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0],
       …,
       [1, 1, 0, …, 0, 0, 0],
       [0, 0, 1, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0]], dtype=int64)
```

[15]:
```
labeled_data = train_data.iloc[:,0:8]
labeled_data.head()       # getting first 5 rows
```

[15]:
```
   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
```

[16]:
```
labeled_data.nunique()           # ununique labeled data
```

[16]:
```
X0    47
X1    27
X2    44
X3     7
X4     4
X5    29
X6    12
X8    25
dtype: int64
```

[17]:
```
labeled_data1 = labeled_data.apply(label().fit_transform)
labeled_data1.head()
```

[17]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	32	23	17	0	3	24	9	14
1	32	21	19	4	3	28	11	14
2	20	24	34	2	3	27	9	23
3	20	21	34	5	3	27	11	4
4	20	23	34	5	3	12	3	13

[18]:
```
labeled_data1.var()                # check variance in labeled data
```

[18]:
```
X0    188.741938
X1     72.777974
X2    118.808135
X3      3.027295
X4      0.005461
X5     68.076236
X6      8.508730
X8     49.531868
dtype: float64
```

[19]:
```
train_data_zero_var_final = pd.DataFrame(train_data_without_zero_var)
train_data_zero_var_final.head()          # getting 5 rows
```

[19]:

	0	1	2	3	4	5	6	7	8	9	…	345	346	347	348	\
0	0	1	0	0	0	0	1	0	0	1	…	0	0	1	0	
1	0	0	0	0	0	0	1	0	0	0	…	1	0	0	0	
2	0	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	0	
4	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	

```
   349  350  351  352  353  354
0    0    0    0    0    0    0
1    0    0    0    0    0    0
2    0    0    1    0    0    0
3    0    0    0    0    0    0
4    0    0    0    0    0    0

[5 rows x 355 columns]
```

[20]:
```
final_train_data = pd.concat([labeled_data1,train_data_zero_var_final],axis =
↪1)     # concat the data
final_train_data.head()       # getting 5 rows
```

[20]:
```
   X0  X1  X2  X3  X4  X5  X6  X8  0  1  …  345  346  347  348  349  350  \
0  32  23  17   0   3  24   9  14  0  1  …    0    0    1    0    0    0
1  32  21  19   4   3  28  11  14  0  0  …    1    0    0    0    0    0
2  20  24  34   2   3  27   9  23  0  0  …    0    0    0    0    0    0
3  20  21  34   5   3  27  11   4  0  0  …    0    0    0    0    0    0
4  20  23  34   5   3  12   3  13  0  0  …    0    0    0    0    0    0

   351  352  353  354
0    0    0    0    0
1    0    0    0    0
2    1    0    0    0
3    0    0    0    0
4    0    0    0    0

[5 rows x 363 columns]
```

[21]:
```
final_train_data.isnull().any()    # finding null values
```

[21]:
```
X0     False
X1     False
X2     False
X3     False
X4     False
       …
350    False
351    False
352    False
353    False
354    False
Length: 363, dtype: bool
```

[22]:
```
test = test.drop(["ID"],axis =1)
test.head()
```

[22]:
```
  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]
```

[23]:
```
test.head()          # getting 5 rows
```

[23]:
```
  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]
```

[24]:
```
test.nunique()
```

[24]:
```
X0      49
X1      27
X2      45
X3       7
X4       4
       ..
X380     2
X382     2
X383     2
X384     2
X385     2
```

```
Length: 376, dtype: int64
```

[25]:
```
test.isnull().any()     # finding full values
```

[25]:
```
X0      False
X1      False
X2      False
X3      False
X4      False
        …
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 376, dtype: bool
```

[26]:
```
test.var().sort_values().head(15)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11252\1038450595.py:1: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  test.var().sort_values().head(15)
```

[26]:
```
X295    0.000000
X369    0.000000
X296    0.000000
X257    0.000000
X258    0.000000
X278    0.000238
X233    0.000238
X280    0.000238
X290    0.000238
X293    0.000238
X330    0.000238
X235    0.000238
X288    0.000238
X210    0.000238
X297    0.000238
dtype: float64
```

[27]:
```
test_data_without_zero_var = variance.fit_transform(test.iloc[:,9:])
test_data_without_zero_var
```

[27]:
```
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, 1, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0]], dtype=int64)
```

[28]:
```
labeled_data = test.iloc[:,0:8]
labeled_data.head()        # getting 5 rows
```

[28]:
```
   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
```

[29]:
```
test_label = labeled_data.apply(label().fit_transform)     # Apply label␣
 ↪encoding
test_label.head()
```

[29]:
```
   X0  X1  X2  X3  X4  X5  X6  X8
0  21  23  34   5   3  26   0  22
1  42   3   8   0   3   9   6  24
2  21  23  17   5   3   0   9   9
3  21  13  34   5   3  31  11  13
4  45  20  17   2   3  30   8  12
```

[30]:
```
test_data_final = pd.concat([test_label,train_data_zero_var_final],axis = 1)  #␣
 ↪concate the data
test_data_final.head()      # getting 5 rows
```

[30]:
```
   X0  X1  X2  X3  X4  X5  X6  X8  0  1  …  345  346  347  348  349  350  \
0  21  23  34   5   3  26   0  22  0  1  …    0    0    1    0    0    0
1  42   3   8   0   3   9   6  24  0  0  …    1    0    0    0    0    0
2  21  23  17   5   3   0   9   9  0  0  …    0    0    0    0    0    0
3  21  13  34   5   3  31  11  13  0  0  …    0    0    0    0    0    0
4  45  20  17   2   3  30   8  12  0  0  …    0    0    0    0    0    0

   351  352  353  354
0    0    0    0    0
1    0    0    0    0
2    1    0    0    0
3    0    0    0    0
4    0    0    0    0

[5 rows x 363 columns]
```

[31]:
```
test_data_final = pd.concat([test_label,train_data_zero_var_final],axis = 1)
test_data_final.head()
```

[31]:

	X0	X1	X2	X3	X4	X5	X6	X8	0	1	…	345	346	347	348	349	350	\
0	21	23	34	5	3	26	0	22	0	1	…	0	0	1	0	0	0	
1	42	3	8	0	3	9	6	24	0	0	…	1	0	0	0	0	0	
2	21	23	17	5	3	0	9	9	0	0	…	0	0	0	0	0	0	
3	21	13	34	5	3	31	11	13	0	0	…	0	0	0	0	0	0	
4	45	20	17	2	3	30	8	12	0	0	…	0	0	0	0	0	0	

	351	352	353	354
0	0	0	0	0
1	0	0	0	0
2	1	0	0	0
3	0	0	0	0
4	0	0	0	0

[5 rows x 363 columns]

[170]:
```
# find correlation
train.corr()
```

[170]:

	ID	y	X10	X11	X12	X13	X14	\
ID	1.000000	-0.055108	0.001602	NaN	0.058988	-0.031917	-0.025438	
y	-0.055108	1.000000	-0.026985	NaN	0.089792	0.048276	0.193643	
X10	0.001602	-0.026985	1.000000	NaN	-0.033084	-0.028806	-0.100474	
X11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
X12	0.058988	0.089792	-0.033084	NaN	1.000000	0.214825	-0.246513	
…	…	…	…	…	…	…	…	
X380	-0.013577	0.040932	-0.010479	NaN	-0.005566	0.023045	0.007743	
X382	-0.038171	-0.159815	-0.010164	NaN	-0.024937	-0.021713	0.012713	
X383	-0.009332	0.040291	-0.004740	NaN	-0.011628	-0.010125	0.023604	
X384	-0.015355	-0.004591	-0.002532	NaN	-0.006212	0.041242	0.025199	
X385	0.029059	-0.022280	-0.004387	NaN	-0.010765	-0.009373	0.043667	

	X15	X16	X17	…	X375	X376	X377	\
ID	0.002237	-0.036480	-0.038171	…	0.045229	-0.080259	-0.022965	
y	0.023116	0.048946	-0.159815	…	0.029100	0.114005	0.061403	
X10	-0.002532	-0.005944	-0.010164	…	0.165277	-0.028618	-0.074244	
X11	NaN	NaN	NaN	…	NaN	NaN	NaN	
X12	-0.006212	-0.014584	-0.024937	…	-0.107864	-0.070214	0.030134	
…	…	…	…	…	…	…	…	
X380	-0.001968	-0.004619	-0.007899	…	-0.061741	-0.022240	-0.061168	
X382	-0.001908	-0.004480	1.000000	…	-0.059883	-0.021571	-0.059327	
X383	-0.000890	-0.002089	-0.003572	…	-0.015413	-0.010059	0.035107	
X384	-0.000475	-0.001116	-0.001908	…	-0.014917	-0.005373	0.008694	
X385	-0.000824	-0.001934	-0.003307	…	0.055225	-0.009311	-0.025610	

```
          X378      X379      X380      X382      X383      X384      X385
ID    0.030371  0.023382 -0.013577 -0.038171 -0.009332 -0.015355  0.029059
y    -0.258679  0.067919  0.040932 -0.159815  0.040291 -0.004591 -0.022280
X10  -0.016870 -0.011374 -0.010479 -0.010164 -0.004740 -0.002532 -0.004387
X11        NaN       NaN       NaN       NaN       NaN       NaN       NaN
X12  -0.016043 -0.027907 -0.005566 -0.024937 -0.011628 -0.006212 -0.010765
…         …         …         …         …         …         …
X380 -0.013110 -0.008839  1.000000 -0.007899 -0.003683 -0.001968 -0.003410
X382 -0.012716 -0.008573 -0.007899  1.000000 -0.003572 -0.001908 -0.003307
X383 -0.005930 -0.003998 -0.003683 -0.003572  1.000000 -0.000890 -0.001542
X384 -0.003168 -0.002136 -0.001968 -0.001908 -0.000890  1.000000 -0.000824
X385 -0.005489 -0.003701 -0.003410 -0.003307 -0.001542 -0.000824  1.000000

[370 rows x 370 columns]
```

[171]:
```
test.corr()
```

[171]:
```
           X10       X11       X12       X13       X14       X15       X16  \
X10   1.000000 -0.002146 -0.039453 -0.035496 -0.120379 -0.003717 -0.007125
X11  -0.002146  1.000000 -0.004369 -0.003931  0.017825 -0.000412 -0.000789
X12  -0.039453 -0.004369  1.000000  0.283228 -0.245127 -0.007570 -0.014509
X13  -0.035496 -0.003931  0.283228  1.000000 -0.076145 -0.006811 -0.013054
X14  -0.120379  0.017825 -0.245127 -0.076145  1.000000 -0.023097 -0.044269
…         …         …         …         …         …         …
X380 -0.012561 -0.001391 -0.025578  0.054582  0.007787 -0.002410 -0.004619
X382 -0.013108 -0.001452 -0.016991 -0.024015  0.000864 -0.002515 -0.004821
X383 -0.003035 -0.000336 -0.006180 -0.005560  0.025212 -0.000582 -0.001116
X384 -0.003717 -0.000412 -0.007570 -0.006811  0.030881 -0.000713 -0.001367
X385 -0.005681 -0.000629 -0.011569 -0.010408  0.047195 -0.001090 -0.002089

           X17       X18       X19  …      X375      X376      X377  \
X10  -0.013108 -0.014142 -0.049351  …  0.189023 -0.031817 -0.086214
X11  -0.001452 -0.001566 -0.005466  … -0.010720 -0.003524 -0.010380
X12  -0.016991 -0.028796 -0.100493  … -0.148812 -0.064790  0.080843
X13  -0.024015 -0.025908 -0.090413  … -0.177340 -0.058291  0.359450
X14   0.000864 -0.087862 -0.306620  …  0.107496  0.043260 -0.139742
…         …         …         …  …         …         …
X380 -0.008498  0.043621 -0.023568  … -0.062756 -0.020628 -0.060764
X382  1.000000  0.066366 -0.033389  … -0.065490 -0.021526 -0.063411
X383 -0.002053 -0.002215 -0.007730  … -0.015163 -0.004984  0.008850
X384 -0.002515 -0.002713 -0.009469  …  0.000420 -0.006105  0.020448
X385 -0.003844 -0.004147 -0.014471  …  0.058691 -0.009330 -0.027482

          X378      X379      X380      X382      X383      X384      X385
X10  -0.019498 -0.015262 -0.012561 -0.013108 -0.003035 -0.003717 -0.005681
X11  -0.002159 -0.001690 -0.001391 -0.001452 -0.000336 -0.000412 -0.000629
```

```
X12  -0.006747 -0.022720 -0.025578 -0.016991 -0.006180 -0.007570 -0.011569
X13  -0.035722 -0.027961  0.054582 -0.024015 -0.005560 -0.006811 -0.010408
X14  -0.051238  0.113487  0.007787  0.000864  0.025212  0.030881  0.047195
…         …        …         …         …         …         …         …
X380 -0.012641 -0.009895  1.000000 -0.008498 -0.001968 -0.002410 -0.003683
X382 -0.013192 -0.010326 -0.008498  1.000000 -0.002053 -0.002515 -0.003844
X383 -0.003054 -0.002391 -0.001968 -0.002053  1.000000 -0.000582 -0.000890
X384 -0.003741 -0.002928 -0.002410 -0.002515 -0.000582  1.000000 -0.001090
X385 -0.005717 -0.004475 -0.003683 -0.003844 -0.000890 -0.001090  1.000000

[368 rows x 368 columns]
```

[35]:
```
# scaling the data
from sklearn import preprocessing
```

[81]:
```
from sklearn.preprocessing import MinMaxScaler
```

[82]:
```
min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))
```

[83]:
```
x_after_min_max_scaler = min_max_scaler.fit_transform(test_data_final)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got
feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got
feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

[84]:
```
print ("\nAfter min max Scaling : \n", x_after_min_max_scaler)
```

```
After min max Scaling :
 [[0.4375     0.88461538 0.77272727 … 0.         0.         0.        ]
 [0.875      0.11538462 0.18181818 … 0.         0.         0.        ]
 [0.4375     0.88461538 0.38636364 … 0.         0.         0.        ]
 …
 [0.97916667 0.88461538 0.38636364 … 0.         0.         0.        ]
 [0.14583333 0.88461538 0.38636364 … 0.         0.         0.        ]
 [0.875      0.03846154 0.18181818 … 0.         0.         0.        ]]
```

[85]:
```
Standardisation = preprocessing.StandardScaler()
```

[86]:
```
x_after_Standardisation = Standardisation.fit_transform(test_data_final)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got
```

```
feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got
feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

[87]:
```
print ("\nAfter Standardisation : \n", x_after_Standardisation)
```

```
After Standardisation :
 [[-0.62521149  1.39576032  1.58606761 … -0.04081511 -0.02180363
  -0.03778296]
 [ 0.75460919 -0.94519929 -0.95644521 … -0.04081511 -0.02180363
  -0.03778296]
 [-0.62521149  1.39576032 -0.07634462 … -0.04081511 -0.02180363
  -0.03778296]
 …
 [ 1.08313793  1.39576032 -0.07634462 … -0.04081511 -0.02180363
  -0.03778296]
 [-1.54509194  1.39576032 -0.07634462 … -0.04081511 -0.02180363
  -0.03778296]
 [ 0.75460919 -1.17929525 -0.95644521 … -0.04081511 -0.02180363
  -0.03778296]]
```

0.3 Perform dimensionality reduction

[159]:
```
from sklearn.model_selection import train_test_split
```

[160]:
```
X_train, X_test, y_train, y_test= train_test_split(final_train_data,␣
 ↪train_target, test_size=0.3)
```

[161]:
```
# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got
feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got
feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got
```

```
feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

[162]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[162]:
```
((2946, 363), (1263, 363), (2946,), (1263,))
```

[163]:
```
from sklearn.decomposition import PCA
```

[164]:
```
pca = PCA(n_components = 2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

[165]:
```
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
test_data_final=pca.transform(test_data_final)
```

0.4 XGBoost

[98]:
```
!pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: xgboost in
c:\users\hp\appdata\roaming\python\python39\site-packages (1.7.1)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.7.3)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.21.5)
```

[99]:
```
import xgboost as xgb
```

[100]:
```
!pip install XGRegressor
```

```
Defaulting to user installation because normal site-packages is not writeable

ERROR: Could not find a version that satisfies the requirement XGRegressor (from
versions: none)
ERROR: No matching distribution found for XGRegressor
```

[101]:
```
model = XGBRegressor()
```

[102]:
```
from sklearn import svm
from sklearn.metrics import r2_score, mean_squared_error
```

[103]:
```
#importing libraries
import xgboost as xgb
from xgboost.sklearn import XGBRegressor
from sklearn.model_selection import GridSearchCV
```

[104]:
```
# Various hyper-parameters to tune
xgb1 = XGBRegressor()
parameters = {'nthread':[4], #when use hyperthread, xgboost may become slower
              'objective':['reg:linear'],
              'learning_rate': [.03, 0.05, .07], #so called `eta` value
              'max_depth': [5, 6, 7],
              'min_child_weight': [4,5,6],
              'silent': [1],
              'subsample': [0.7],
              'colsample_bytree': [0.7],
              'n_estimators': [500]}

xgb_grid = GridSearchCV(xgb1,
                        parameters,
                        cv =3,
                        n_jobs = 5,
                        verbose=True)

xgb_grid.fit(X_train,
         y_train)

print(xgb_grid.best_score_)
print(xgb_grid.best_params_)
```

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
[10:14:55] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling-
group-i-03de431ba26204c4d-1/xgboost/xgboost-ci-
windows/src/objective/regression_obj.cu:213: reg:linear is now deprecated in
favor of reg:squarederror.
[10:14:55] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling-
group-i-03de431ba26204c4d-1/xgboost/xgboost-ci-windows/src/learner.cc:767:
Parameters: { "silent" } are not used.

0.2139983080421588
{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 5,
'min_child_weight': 6, 'n_estimators': 500, 'nthread': 4, 'objective':
'reg:linear', 'silent': 1, 'subsample': 0.7}
```

[105]:
```
#calling XGBoost
model=XGBRegressor(colsample_bytree= 0.7, learning_rate= 0.03, max_depth= 5,␣
 ↪min_child_weight= 5, n_estimators= 500, nthread= 4, objective='reg:linear',␣
 ↪silent=1, subsample=0.7)
model.fit(X_train,
         y_train)
```

```
[10:14:55] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling-
group-i-03de431ba26204c4d-1/xgboost/xgboost-ci-
windows/src/objective/regression_obj.cu:213: reg:linear is now deprecated in
```

```
favor of reg:squarederror.
[10:14:55] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling-
group-i-03de431ba26204c4d-1/xgboost/xgboost-ci-windows/src/learner.cc:767:
Parameters: { "silent" } are not used.
```

[105]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.7,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
             grow_policy='depthwise', importance_type=None,
             interaction_constraints='', learning_rate=0.03, max_bin=256,
             max_cat_threshold=64, max_cat_to_onehot=4, max_delta_step=0,
             max_depth=5, max_leaves=0, min_child_weight=5, missing=nan,
             monotone_constraints='()', n_estimators=500, n_jobs=4, nthread=4,
             num_parallel_tree=1, objective='reg:linear', …)
```

[166]:
```
import sklearn.metrics as met
from sklearn.metrics import mean_squared_error,r2_score
```

[167]:
```
print(X_train.shape)
print(X_test)
```

```
(2946, 2)
[[-2.36647918  1.79998262]
 [-0.79852456  0.04126494]
 [-1.02552642 -0.45391488]
 …
 [-1.05513597 -2.51977599]
 [-1.99499991  1.47614799]
 [-2.07163122  1.61698747]]
```

[168]:
```
y_pred=model.predict(X_test)
```

[169]:
```
# Finding the Evaluation Metrics
print ("training score: ",model.score(X_train,y_train))
MSE  = mean_squared_error(y_test,model.predict(X_test))
print("MSE :" , MSE)

RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)


r2 = r2_score(y_test,model.predict(X_test))
print("R2 :" ,r2)
print("Adjusted R2 : ",1-(1-r2_score(y_test,model.predict(X_test) ))*((X_test.
  ↪shape[0]-1)/(X_test.shape[0]-X_test.shape[1]-1)))
```

```
training score:  0.1711001354230154
```

```
MSE : 130.45165948176506
RMSE : 11.421543655818379
R2 : 0.16534231324112358
Adjusted R2 :  0.16401745977007776
```