Mercedes-Benz Greener Manufacturing

August 19, 2023

[1]:
```
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
%matplotlib inline
```

[2]:
```
df_test =pd.read_csv('test.csv')
df_test.head()
```

[2]:
```
   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]
```

[3]:
```
df_train =pd.read_csv("train.csv")
df_train.head()
```

[3]:
```
   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]
```

[4]:
```
## lets print the shape,column nd info for the train and test data of␣
 ↪mercedes_benz
print("the shape of the train set",df_train.shape)
print("the shape of test set",df_test.shape)
```

```
the shape of the train set (4209, 378)
the shape of test set (4209, 377)
```

[5]:
```
df_train.columns
```

[5]:
```
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)
```

[6]:
```
df_test.columns
```

[6]:
```
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)
```

[7]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 378 entries, ID to X385
dtypes: float64(1), int64(369), object(8)
memory usage: 12.1+ MB
```

[8]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 377 entries, ID to X385
```

```
dtypes: int64(369), object(8)
memory usage: 12.1+ MB
```

[9]:

```
df_train.describe()
```

[9]:

```
                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]
```

[10]:
```
df_test.describe()
```

[10]:
```
                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]
```

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

[11]:
```
df_train.var()
```

[11]:
```
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
```

[12]:
```
## check if the train set variance is equal t zero
df_train.var()==0
```

[12]:
```
ID      False
y       False
X10     False
X11      True
X12     False
        …
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 370, dtype: bool
```

[13]:
```
(df_train.var()==0).values
```

[13]:
```
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,  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])
```

[14]:
```
train_variance_zero =df_train.var()[df_train.var()==0].index.values
train_variance_zero
```

[14]:
```
array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'], dtype=object)
```

[15]:
```
## removing the data whose variance is equal to zero
print("before dropping the vriance with zero the train set",df_train.shape)
df_train= df_train.drop(train_variance_zero,axis=1)
print("Eliminating the zero variance columns",df_train.shape)
```

```
before dropping the vriance with zero the train set (4209, 378)
Eliminating the zero variance columns (4209, 366)
```

In the bove we can see that the 12 columns has been dropped from the original columns whose variance is zero.

[16]:
```
## Similarly check for test data
df_test.var()
```

[16]:
```
ID      5.871311e+06
X10     1.865006e-02
X11     2.375861e-04
X12     6.885074e-02
X13     5.734498e-02
            …
X380    8.014579e-03
X382    8.715481e-03
X383    4.750593e-04
X384    7.124196e-04
X385    1.660732e-03
Length: 369, dtype: float64
```

[17]:
```
(df_test.var()==0).values
```

[17]:
```
array([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, 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,
        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,  True,  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,  True, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False])
```

[18]:
```
test_variance_zero = df_test.var()[df_test.var()==0].index.values
test_variance_zero
```

[18]:
```
array(['X257', 'X258', 'X295', 'X296', 'X369'], dtype=object)
```

[19]:
```
print("The test data set with zero variance",df_test.shape)
df_test= df_test.drop(test_variance_zero,axis=1)
print("The test data without zero variance",df_test.shape)
```

```
The test data set with zero variance (4209, 377)
The test data without zero variance (4209, 372)
```

we can see that 5 columns has been elimanted whose variance is zero.

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

[20]:
```
# lets check the relevent columns for our prediction
df_train.columns
```

[20]:
```
Index(['ID', 'y', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=366)
```

[21]:
```
df_test.columns
```

[21]:
```
Index(['ID', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=372)
```

we can see that ID columns has no relevancy in prediction of the data for sales,safety ,optimizing the speed.

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

[22]:
```
        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 365 columns]
```

[23]:
```
df_test =df_test.drop(["ID"],axis=1)
df_test.head()
```

[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 371 columns]
```

[24]:
```
# checking for the null values in train set and test set for mercedes_Benz
df_train.isnull().sum()
```

[24]:
```
y       0
X0      0
X1      0
X2      0
```

```
X3       0
        ..
X380     0
X382     0
X383     0
X384     0
X385     0
Length: 365, dtype: int64
```

[25]:
```python
df_test.isnull().sum()
```

[25]:
```
X0       0
X1       0
X2       0
X3       0
X4       0
        ..
X380     0
X382     0
X383     0
X384     0
X385     0
Length: 371, dtype: int64
```

[26]:
```python
df_train.isna().sum().values
```

[26]:
```
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])
```

[27]:
```python
df_test.isna().sum().values
```

[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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

[28]:
```python
# find the uniques values in test and train set
df_train.nunique()
```

[28]:
```
y       2545
X0        47
X1        27
X2        44
X3         7
        …
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 365, dtype: int64
```

[29]:
```python
df_train.nunique().values
```

[29]:
```
array([2545,   47,   27,   44,    7,    4,   29,   12,   25,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
```

```
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
        2,    2])
```

[30]:
```python
df_test.nunique()
```

[30]:
```
X0      49
X1      27
X2      45
X3       7
X4       4
        ..
X380     2
X382     2
X383     2
X384     2
X385     2
Length: 371, dtype: int64
```

[31]:
```python
df_test.nunique().values
```

[31]:
```
array([49, 27, 45,  7,  4, 32, 12, 25,  2,  2,  2,  2,  2,  2,  2,  2,  2,
        2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,
        2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,
        2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,
        2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,
        2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,
```

```
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

0.0.3 Apply label encoder.

Before applying label encoder lets bifurcate the categorical variable (object dtypes).

[32]:
```python
cat_train = df_train.select_dtypes(include=[object])
cat_train.head()
```

[32]:
```
   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
```

[33]:
```python
cat_test =df_test.select_dtypes(include=[object])
cat_test.head()
```

[33]:
```
   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
```

[34]:
```python
## lets check the columns, describe and shape of the new train categorical data
cat_train.columns
```

[34]:
```
Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype='object')
```

[35]:
```
cat_train.shape
```

[35]:
```
(4209, 8)
```

[36]:
```
cat_train.describe()
```

[36]:
```
          X0    X1    X2    X3    X4    X5    X6    X8
count   4209  4209  4209  4209  4209  4209  4209  4209
unique    47    27    44     7     4    29    12    25
top        z    aa    as     c     d     v     g     j
freq     360   833  1659  1942  4205   231  1042   277
```

[37]:
```
cat_test.columns
```

[37]:
```
Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype='object')
```

[38]:
```
cat_test.shape
```

[38]:
```
(4209, 8)
```

[39]:
```
cat_test.describe()
```

[39]:
```
          X0    X1    X2    X3    X4    X5    X6    X8
count   4209  4209  4209  4209  4209  4209  4209  4209
unique    49    27    45     7     4    32    12    25
top       ak    aa    as     c     d     v     g     e
freq     432   826  1658  1900  4203   246  1073   274
```

[40]:
```
le =LabelEncoder()
```

[41]:
```
df_train['X0'] =le.fit_transform(df_train['X0'])
```

[42]:
```
df_train.X0
```

[42]:
```
0       32
1       32
2       20
3       20
4       20
        ..
4204     8
4205    31
4206     8
4207     9
4208    46
Name: X0, Length: 4209, dtype: int64
```

[43]:
```
# similarly we encode the data for columns X1,X2,X3,X4,X5,X6,X8
df_train['X1']= le.fit_transform(df_train['X1'])
df_train['X2']=le.fit_transform(df_train['X2'])
df_train['X3']=le.fit_transform(df_train['X3'])
df_train['X4']=le.fit_transform(df_train['X4'])
df_train['X5']=le.fit_transform(df_train['X5'])
df_train['X6']=le.fit_transform(df_train['X6'])
df_train['X8']=le.fit_transform(df_train['X8'])
```

[44]:
```
df_test.X0
```

[44]:
```
0       az
1        t
2       az
3       az
4        w
        ..
4204    aj
4205     t
4206     y
4207    ak
4208     t
Name: X0, Length: 4209, dtype: object
```

[45]:
```
# Similarly we can use label Encoders for test datasets
df_test['X0']= le.fit_transform(df_test['X0'])
df_test['X1']=le.fit_transform(df_test['X1'])
df_test['X2']=le.fit_transform(df_train['X2'])
df_test['X3']=le.fit_transform(df_test['X3'])
df_test['X4']=le.fit_transform(df_test['X4'])
df_test['X5']=le.fit_transform(df_train['X5'])
df_test['X6']=le.fit_transform(df_test['X6'])
df_test['X8']=le.fit_transform(df_test['X8'])
```

[46]:
```
df_train.head()
```

[46]:

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

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

[5 rows x 365 columns]
```

[47]:
```
df_test.head()
```

[47]:
```
   X0  X1  X2  X3  X4  X5  X6  X8  X10  X11  …  X375  X376  X377  X378  \
0  21  23  17   5   3  24   0  22    0    0  …     0     0     0     1
1  42   3  19   0   3  28   6  24    0    0  …     0     0     1     0
2  21  23  34   5   3  27   9   9    0    0  …     0     0     0     1
3  21  13  34   5   3  27  11  13    0    0  …     0     0     0     1
4  45  20  34   2   3  12   8  12    0    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

[5 rows x 371 columns]
```

0.0.4 Perform dimensionality reduction.

[48]:
```
from sklearn.decomposition import PCA
```

[49]:
```
pca =PCA(n_components=0.95) #95% of PCA(Principal Component Analysis used to␣
 ↪reduce the linearity of dimension)
```

[50]:
```
pca.fit(df_train)
```

[50]:
```
PCA(n_components=0.95)
```

[51]:
```
x_train_95 =pca.transform(df_train)
```

[52]:
```
print("the train set shape",df_train.shape)
print("PCA x train data shape",x_train_95.shape)
```

```
the train set shape (4209, 365)
PCA x train data shape (4209, 6)
```

1 PCA for test Dataset 95%

[53]:
```
pca.fit(df_test)
x_test_95=pca.transform(df_test)
```

[54]:
```
print("the test set shape",df_test.shape)
print("PCA x test data shape",x_test_95.shape)
```

```
the test set shape (4209, 371)
PCA x test data shape (4209, 6)
```

we can see the dimensionality linear reduction using PCmethod has been sucessfully done. lets check for pca =98%

[55]:
```
pca_98=PCA(n_components =0.98)
pca_98.fit(df_train)
x_train_98 =pca_98.transform(df_train)
```

[56]:
```
print("the train set shape",df_train.shape)
print("PCA x train data PCA 95% shape",x_train_95.shape)
print("PCA x train data PCA  98% shape",x_train_98.shape)
```

```
the train set shape (4209, 365)
PCA x train data PCA 95% shape (4209, 6)
PCA x train data PCA  98% shape (4209, 12)
```

2 PCA for test dtaset 98%

[57]:
```
pca_98.fit(df_test)
x_test_98 =pca_98.transform(df_test)
```

[58]:
```
print("the test set shape",df_test.shape)
print("PCA x test data PCA 95% shape",x_test_95.shape)
print("PCA x test data PCA  98% shape",x_test_98.shape)
```

```
the test set shape (4209, 371)
PCA x test data PCA 95% shape (4209, 6)
PCA x test data PCA  98% shape (4209, 13)
```

3 Train and test Split method

[59]:
```
X =df_train.drop('y',axis=1)
y= df_train.y
```

[60]:
```
X.head()
```

[60]:
```
   X0  X1  X2  X3  X4  X5  X6  X8  X10  X12  …  X375  X376  X377  X378  \
0  32  23  17   0   3  24   9  14    0    0  …     0     0     1     0
1  32  21  19   4   3  28  11  14    0    0  …     1     0     0     0
2  20  24  34   2   3  27   9  23    0    0  …     0     0     0     0
3  20  21  34   5   3  27  11   4    0    0  …     0     0     0     0
4  20  23  34   5   3  12   3  13    0    0  …     0     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     1     0     0     0
3     0     0     0     0     0     0
4     0     0     0     0     0     0

[5 rows x 364 columns]
```

[61]:
```
y
```

[61]:
```
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
```

we can see that we have split the train dataset for training and testing.

[62]:
```
x_train,x_test,y_train,y_test =train_test_split(X,y,random_state=4,test_size=0.
 ↪3)
```

[63]:
```
print("the x train ",x_train.shape)
print("the y train ",y_train.shape)
print("the x test ",x_test.shape)
print("the y test ",y_test.shape)
```

```
the x train  (2946, 364)
the y train  (2946,)
the x test  (1263, 364)
the y test  (1263,)
```

let us apply dimensionality reduction on these train and test set

[64]:
```
pca_train =PCA(n_components =0.95)
```

[65]:
```
x_train = pca_train.fit_transform(x_train)
x_train
```

[65]:
```
array([[-14.61088357,  -5.44468509,   0.80104129,  18.74714849,
         -5.65930201,   3.21497912],
       [-15.27453709,   3.28027982, -12.94264609,   3.16512933,
         -7.24110937,   4.02606781],
       [  0.78016352,  17.59226159,  -0.93552596,  11.42292115,
        -10.44001634,  -2.02288687],
       …,
       [ 14.14162758, -16.10180716,  -2.83619905, -12.87851378,
         10.12104015,  -2.31762817],
       [  8.29098768,  22.86626337,  -6.5125823 , -10.48590224,
         -1.19922965,  -0.51539951],
       [ 11.58518722, -12.52254749,  -4.87001745,  14.76108837,
          7.97477676,  -1.51903313]])
```

[ ]:

[66]:
```
pca_test=PCA(n_components=0.95)
pca_test.fit(x_test)
```

[66]:
```
PCA(n_components=0.95)
```

[67]:
```
x_test =pca_test.fit_transform(x_test)
x_test
```

[67]:
```
array([[ -4.31386696,   0.92430474,   9.39805756,   9.74687656,
         -7.7774354 ,  -2.31105816],
       [ -5.03267524,   0.45439079,   5.33359858,   8.75556393,
          3.14899705,  -2.34863649],
       [ 23.33451655,  16.63864391,  -6.68758541, -12.42121238,
         -6.69189448,  -1.66313882],
       …,
       [ 26.87917614,  14.68405351,   7.26613642,  -1.04794556,
         -8.14630772,   4.17370677],
       [  3.35457377, -11.28939448,  10.24108822,  -5.14665291,
          0.69354433,  -4.30753325],
       [ -2.62224304, -14.18635134,  -4.58477466, -13.77830518,
         -3.16576757,   0.67861946]])
```

[68]:
```
x_train.shape
```

[68]: (2946, 6)

[69]:
```
x_test.shape
```

[69]: (1263, 6)

[70]:
```
pca_test.explained_variance_
```

[70]:
```
array([205.58713329, 111.82314965,  71.21523868,  62.4419506 ,
        49.22480611,   8.46646788])
```

[71]:
```
pca_test.explained_variance_ratio_
```

[71]:
```
array([0.3864524 , 0.21019956, 0.13386684, 0.11737525, 0.09253033,
       0.01591484])
```

3.1 Perform XGBoost

Predict your test_df values using XGBoost.

[72]:
```
from sklearn import svm
from sklearn import model_selection
from sklearn.metrics import␣
↪accuracy_score,r2_score,mean_squared_error,precision_score,confusion_matrix,classification_r
import xgboost as xgb
```

[73]:
```
seed =7
num_trees=30
kfold= model_selection.KFold(n_splits=10,random_state=7,shuffle=True)
model =xgb.XGBRegressor( objective ='reg:linear', colsample_bytree = 0.3,␣
↪learning_rate = 0.1,
                        max_depth = 10, alpha = 10,n_estimators=num_trees,␣
↪random_state=7)
result= model_selection.cross_val_score(model,X,y,cv=kfold)
```

```
[07:26:51] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
[07:26:52] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
[07:26:54] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
[07:26:55] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
[07:26:56] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
[07:26:58] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
```

```
is now deprecated in favor of reg:squarederror.
[07:26:59] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
[07:27:01] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
[07:27:02] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
[07:27:03] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
```

[74]:
```
print(result.mean())
```

```
0.45283444057913275
```

lets check the mean squared error and r2_score

[75]:
```
model.fit(x_train,y_train)
```

```
[07:27:04] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
```

[75]:
```
XGBRegressor(alpha=10, base_score=0.5, booster=None, colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.3, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints=None,
             learning_rate=0.1, max_delta_step=0, max_depth=10,
             min_child_weight=1, missing=nan, monotone_constraints=None,
             n_estimators=30, n_jobs=0, num_parallel_tree=1,
             objective='reg:linear', random_state=7, reg_alpha=10, reg_lambda=1,
             scale_pos_weight=1, subsample=1, tree_method=None,
             validate_parameters=False, verbosity=None)
```

[88]:
```
y_pred =model.predict(x_test)
y_pred
```

[88]:
```
array([ 97.59818,  93.51209, 100.91509, …,  99.22959,  93.56673,
        99.0103 ], dtype=float32)
```

Now let us check the mean squared error and r2_score of the data for df_train.

[89]:
```
import math
from math import sqrt
```

[90]:
```
print(sqrt(mean_squared_error(y_pred,y_test)))
```

```
12.069975126714981
```

[91]:
```
print(r2_score(y_pred,y_test))
```

```
-8.419414020856575
```

3.2 Conclusion:

we have seen the each module predict approximately 95% testing speed for Mercedes_Benz to scale the safety and reliability of every unique car models of Mercedes_Benz configuration before hits the road. Therefore dimenstionality reduction helps to create different small scale models of a very large datasets and then helps to ensemble it for prediction by using XGBoost gradient function which wil further predict the algortithm for robust test procedure which includes faster testing, resulting in lower carbon dioxide emissions without reducing Mercedes-Benz's standards.

[ ]:

```

```