Project 1: Mercedes-Benz Greener Manufacturing

Submitted by Ayub S Bijapur

In [2]:
```
# 1.Import Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.feature_selection import VarianceThreshold
variance = VarianceThreshold(threshold=0)
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder
```

In [3]:
```
pwd()
```

Out[3]:
```
'C:\\Users\\AYUB BIJAPUR\\Machine Learning\\Project 1'
```

In [5]:
```
import os
```

In [6]:
```
os.getcwd()
```

Out[6]:
```
'C:\\Users\\AYUB BIJAPUR\\Machine Learning\\Project 1'
```

In [7]:
```
os.chdir("D:\\Simplilearn Ayub\\Courses 1-8\\(6) Machine Learning\\Projects\\Merce
```

In [8]:
```
os.getcwd()
```

Out[8]:
```
'D:\\Simplilearn Ayub\\Courses 1-8\\(6) Machine Learning\\Projects\\Mercedes-Benz
Greener Manufacturing\\Datasets'
```

In [9]:
```
# 2.Load Data

mb_trn = pd.read_csv('Train.csv')
mb_tst = pd.read_csv('Test.csv')
```

In [10]:
```
print(mb_trn)
```

```
        ID       y  X0 X1  X2 X3 X4  X5 X6 X8  ...  X375  X376  X377  X378  \
0        0  130.81   k  v  at  a  d   u  j  o  ...     0     0     1     0
1        6   88.53   k  t  av  e  d   y  l  o  ...     1     0     0     0
2        7   76.26  az  w   n  c  d   x  j  x  ...     0     0     0     0
3        9   80.62  az  t   n  f  d   x  l  e  ...     0     0     0     0
4       13   78.02  az  v   n  f  d   h  d  n  ...     0     0     0     0
...    ...     ...  .. ..  .. .. ..  .. .. ..  ...   ...   ...   ...   ...
4204  8405  107.39  ak  s  as  c  d  aa  d  q  ...     1     0     0     0
4205  8406  108.77   j  o   t  d  d  aa  h  h  ...     0     1     0     0
4206  8412  109.22  ak  v   r  a  d  aa  g  e  ...     0     0     1     0
4207  8415   87.48  al  r   e  f  d  aa  l  u  ...     0     0     0     0
4208  8417  110.85   z  r  ae  c  d  aa  g  w  ...     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     1     0     0     0
3        0     0     0     0     0     0
4        0     0     0     0     0     0
...    ...   ...   ...   ...   ...   ...
4204     0     0     0     0     0     0
4205     0     0     0     0     0     0
4206     0     0     0     0     0     0
4207     0     0     0     0     0     0
4208     0     0     0     0     0     0

[4209 rows x 378 columns]
```

In [11]:
```
print(mb_tst)
```

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

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

[4209 rows x 377 columns]
```

In [13]:
```
 ## ANALYSIS ON TRAIN DATA:
```

In [12]:
```
# The target of train data is "y". Also the ID column is unnecessary so we remove t

mb_new_trn = mb_trn.drop(["y","ID"], axis = 1)
print(mb_new_trn)
```

```
      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
...   .. ..  .. .. ..  .. .. ..  ...  ...  ...   ...   ...   ...   ...   ...
4204  ak  s  as  c  d  aa  d  q    0    0  ...     1     0     0     0     0
4205   j  o   t  d  d  aa  h  h    0    0  ...     0     1     0     0     0
4206  ak  v   r  a  d  aa  g  e    0    0  ...     0     0     1     0     0
4207  al  r   e  f  d  aa  l  u    0    0  ...     0     0     0     0     0
4208   z  r  ae  c  d  aa  g  w    0    0  ...     1     0     0     0     0

      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
...    ...   ...   ...   ...   ...
4204     0     0     0     0     0
4205     0     0     0     0     0
4206     0     0     0     0     0
4207     0     0     0     0     0
4208     0     0     0     0     0

[4209 rows x 376 columns]
```

In [19]:
```
mb_new_tst = mb_tst.drop(["ID"], axis = 1)
print(mb_new_tst)
```

```
      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
...   ..  ..  .. .. ..  .. .. ..  ...  ...  ...   ...   ...   ...   ...   ...
4204  aj   h  as  f  d  aa  j  e    0    0  ...     0     0     0     0     0
4205   t  aa  ai  d  d  aa  j  y    0    0  ...     0     1     0     0     0
4206   y   v  as  f  d  aa  d  w    0    0  ...     0     0     0     0     0
4207  ak   v  as  a  d  aa  c  q    0    0  ...     0     0     1     0     0
4208   t  aa  ai  c  d  aa  g  r    0    0  ...     1     0     0     0     0

      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
...    ...   ...   ...   ...   ...
4204     0     0     0     0     0
4205     0     0     0     0     0
4206     0     0     0     0     0
4207     0     0     0     0     0
4208     0     0     0     0     0

[4209 rows x 376 columns]
```

Task 1: If for any column(s), the variance is equal to zero, then you need to remove those variable(s)

In [14]:
```
# Performing var on train data:
```

In [15]:
```
new_train = mb_new_trn.var().sort_values().head(20)
new_train
```

Out[15]:
```
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
X288    0.000238
X39     0.000238
X33     0.000238
X190    0.000238
X270    0.000238
dtype: float64
```

In [16]:
```
train_data_without_zero_var = variance.fit_transform(mb_new_trn.iloc[:,10:])
train_data_without_zero_var
```

Out[16]:
```
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)
```

In [17]:
```
train_data_without_zero_var.shape
## (Reduction in shape indiactes that the data with zero var discarded and hence re
```

Out[17]:
```
(4209, 355)
```

In [20]:
```
new_test = mb_new_tst.var().sort_values().head(20)
new_test
```

Out[20]:
```
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
X105    0.000238
X124    0.000238
X259    0.000238
X372    0.000238
X190    0.000238
dtype: float64
```

In [21]:
```
test_data_without_zero_var = variance.fit_transform(mb_new_tst.iloc[:,10:])
test_data_without_zero_var
```

Out[21]:
```
array([[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, 1, 1, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [22]:
```
test_data_without_zero_var.shape
## (Reduction in shape indiactes that the data with zero var discarded and hence re
```

Out[22]:
```
(4209, 361)
```

Task 2: Check Null and Unique Values for test and train dataset

In [32]:
```
# Checking Null values
train_data.isnull().sum().any()
```

Out[32]:
```
False
```

In [33]:
```
# Checking Null values
test_data.isnull().sum().any()
```

Out[33]:
```
False
```

In [34]:
```
# Checking Unique values
train_data.nunique().sum().any()
```

Out[34]:
```
True
```

In [36]:
```
train_data.nunique().sum()
```

Out[36]:
```
7661
```

In [38]:
```
# Checking Unique values
test_data.nunique().sum().any()
```

Out[38]:
```
True
```

In [39]:
```
test_data.nunique().sum()
```

Out[39]:
```
5136
```

Task 3: Apply Label Encoder

In [23]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, OrdinalEncoder
```

In [31]:
```
labeled_data_trn = mb_new_trn.iloc[:,0:8]
print(labeled_data_trn.head())
```

```
   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
```

In [33]:
```
ler = labeled_data_trn.apply(label().fit_transform)
print(ler)
```

```
      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
...   ..  ..  ..  ..  ..  ..  ..  ..
4204   8  20  16   2   3   0   3  16
4205  31  16  40   3   3   0   7   7
4206   8  23  38   0   3   0   6   4
4207   9  19  25   5   3   0  11  20
4208  46  19   3   2   3   0   6  22

[4209 rows x 8 columns]
```

In [34]:
```
ler.var()
```

Out[34]:
```
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
```

In [27]:
```
train_data = pd.DataFrame(train_data_without_zero_var)
print(train_data.head())
```

```
   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]
```

In [35]:
```
final_train = pd.concat([ler,train_data],axis=1)
print(final_train.head())
```

```
   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]
```

In [36]:
```
labeled_data_tst = mb_new_tst.iloc[:,0:8]
print(labeled_data_tst.head())
```

```
   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
```

In [37]:
```
les = labeled_data_tst.apply(label().fit_transform)
print(les)
```

```
      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
...   ..  ..  ..  ..  ..  ..  ..  ..
4204   6   9  17   5   3   1   9   4
4205  42   1   8   3   3   1   9  24
4206  47  23  17   5   3   1   3  22
4207   7  23  17   0   3   1   2  16
4208  42   1   8   2   3   1   6  17

[4209 rows x 8 columns]
```

In [38]:
```
les.var()
```

Out[38]:
```
X0    231.684235
X1     73.008823
X2    104.598050
X3      3.157646
X4      0.006171
X5     75.119478
X6      8.225523
X8     49.229574
dtype: float64
```

In [39]:
```
test_data = pd.DataFrame(test_data_without_zero_var)
print(test_data.head())
```

```
   0    1    2    3    4    5    6    7    8    9    ...  351  352  353  354  \
0    0    0    0    0    0    0    0    0    0    0  ...    0    0    0    1
1    0    0    0    0    0    0    0    1    0    0  ...    0    0    1    0
2    0    0    1    0    0    0    0    0    0    0  ...    0    0    0    1
3    0    0    0    0    0    0    0    0    0    0  ...    0    0    0    1
4    0    0    1    0    0    0    0    0    0    0  ...    1    0    0    0

   355  356  357  358  359  360
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 361 columns]
```

In [40]:
```
final_test = pd.concat([les,test_data],axis=1)
print(final_test.head())
```

```
   X0  X1  X2  X3  X4  X5  X6  X8  0  1  ...  351  352  353  354  355  356  \
0  21  23  34   5   3  26   0  22  0  0  ...    0    0    0    1    0    0
1  42   3   8   0   3   9   6  24  0  0  ...    0    0    1    0    0    0
2  21  23  17   5   3   0   9   9  0  0  ...    0    0    0    1    0    0
3  21  13  34   5   3  31  11  13  0  0  ...    0    0    0    1    0    0
4  45  20  17   2   3  30   8  12  0  0  ...    1    0    0    0    0    0

   357  358  359  360
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 369 columns]
```

Task 4: Perform dimensionality reduction.

In [30]:
```
from sklearn.model_selection import train_test_split
train_target = mb_trn.y
```

In [51]:
```
X_train, X_test, y_train, y_test = train_test_split(final_train, train_target,trai
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2525, 363)
(1684, 363)
(2525,)
(1684,)
```

In [54]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=0.95)
```

In [43]:
```
X_train = pca.fit_transform(X_train)
X_test = pca.fit_transform(X_test)
```

In [44]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(2525, 6)
(1684, 6)
```

In [52]:
```
# For test data

X_train, X_test, y_train, y_test = train_test_split(final_test, train_target,train_
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2525, 369)
(1684, 369)
(2525,)
(1684,)
```

Task 5:Predict your test_df values using XGBoost.

In [67]:
```
 from sklearn import svm
from sklearn.metrics import r2_score, mean_squared_error
from xgboost import XGBRegressor
xgbr = XGBRegressor(random_state=42)
```

In [68]:
```
model = xgbr.fit(X_train, y_train)
```

In [69]:
```
ypred_test = xgbr.predict(X_test)
ypred_test
```

Out[69]:
```
array([ 97.46753,  96.96668, 101.3332 , ..., 105.21336,  98.45503,
       110.08113], dtype=float32)
```

In [70]:
```
 print(r2_score(y_test,ypred_test))
```

```
-0.2326796019223032
```

In [71]:
```
MSE = mean_squared_error(y_test,ypred_test)
print(MSE)
```

```
202.97277772796755
```

In [72]:
```
RMSE = MSE*(1/2.0)
print (RMSE)
```

```
101.48638886398378
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: