Mercedes-Benz Greener Manufacturing

Since the first automobile, the Benz Patent Motor Car in 1886, Mercedes-Benz has stood for important automotive innovations. These include the passenger safety cell with the crumple zone, the airbag, and intelligent assistance systems. Mercedes-Benz applies for nearly 2000 patents per year, making the brand the European leader among premium carmakers. Mercedes-Benz cars are leaders in the premium car industry. With a huge selection of features and options, customers can choose the customized Mercedes-Benz of their dreams.

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

- Check for null and unique values for test and train sets
- Apply label encoder.
- Perform dimensionality reduction.
- Predict your test_df values using xgboost

Importing the libraries

In [43]:

```
import time
import random
from math import *
import operator
import pandas as pd
import numpy as np

# import plotting libraries
import matplotlib
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
%matplotlib inline

import seaborn as sns
sns.set(style="white", color_codes=True)
sns.set(font_scale=1.5)

# import the ML algorithm
import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression
#from pandas.core import datetools

# import libraries for model validation
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

# import libraries for metrics and reporting
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn import metrics
from statsmodels.tools.eval_measures import rmse
import warnings
warnings.filterwarnings('ignore')
```

In [44]:

```
#import os
#os.getcwd()
# os.chdir('/Users/ds/Desktop/Project/Mercedes_Benz_Greener_Manufacturing')
```

In [45]:

```
pwd
```

Out[45]:

```
'/Users/ds/Desktop/Project/Mercedes_Benz_Greener_Manufacturing'
```

Importing the Titanic dataset

In [46]:

```
df = pd.read_csv('train.csv')
```

Data understanding and Exploration

In [47]:

```
df.head()
```

Out[47]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0

5 rows × 378 columns

In [48]:

```
df.shape
```

Out[48]:

```
(4209, 378)
```

In [49]:

```
df.columns
```

Out[49]:

```
Index(['ID', 'y', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8',
       ...
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X38
3', 'X384',
       'X385'],
      dtype='object', length=378)
```

In [50]:

```
df.describe()
```

Out[50]:

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

8 rows × 370 columns

Checking for the variance of all features and removing the feature which is equal to zero

In [51]:

```
df.var()==0
```

Out[51]:

```
ID      False
y       False
X10     False
X11      True
X12     False
X13     False
X14     False
X15     False
X16     False
X17     False
X18     False
X19     False
X20     False
X21     False
X22     False
X23     False
X24     False
X26     False
X27     False
X28     False
X29     False
X30     False
X31     False
X32     False
X33     False
X34     False
X35     False
X36     False
X37     False
X38     False
        ...
X355    False
X356    False
X357    False
X358    False
X359    False
X360    False
X361    False
X362    False
X363    False
X364    False
X365    False
X366    False
X367    False
X368    False
X369    False
X370    False
X371    False
X372    False
X373    False
X374    False
X375    False
X376    False
X377    False
X378    False
X379    False
X380    False
X382    False
X383    False
```

```
X384    False
X385    False
Length: 370, dtype: bool
```

Droping the feature whose variance is equal to zero

In [52]:

```
df.drop(['X11'],axis =1,inplace =True)
```

Checking for null values

In [53]:

```
df.isnull().sum()
```

Out[53]:

```
ID      0
y       0
X0      0
X1      0
X2      0
X3      0
X4      0
X5      0
X6      0
X8      0
X10     0
X12     0
X13     0
X14     0
X15     0
X16     0
X17     0
X18     0
X19     0
X20     0
X21     0
X22     0
X23     0
X24     0
X26     0
X27     0
X28     0
X29     0
X30     0
X31     0
       ..
X355    0
X356    0
X357    0
X358    0
X359    0
X360    0
X361    0
X362    0
X363    0
X364    0
X365    0
X366    0
X367    0
X368    0
X369    0
X370    0
X371    0
X372    0
X373    0
X374    0
X375    0
X376    0
X377    0
X378    0
X379    0
X380    0
X382    0
X383    0
```

```
X384    0
X385    0
Length: 377, dtype: int64
```

count of the unique values in each feature

In [54]:

```
df.nunique().head(10)
```

Out[54]:

```
ID    4209
y     2545
X0      47
X1      27
X2      44
X3       7
X4       4
X5      29
X6      12
X8      25
dtype: int64
```

info gives the information of the data types, no. of columns,no. of rows

In [55]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 377 entries, ID to X385
dtypes: float64(1), int64(368), object(8)
memory usage: 12.1+ MB
```

In [56]:

```
df.dtypes
```

Out[56]:

```
ID        int64
y       float64
X0       object
X1       object
X2       object
X3       object
X4       object
X5       object
X6       object
X8       object
X10       int64
X12       int64
X13       int64
X14       int64
X15       int64
X16       int64
X17       int64
X18       int64
X19       int64
X20       int64
X21       int64
X22       int64
X23       int64
X24       int64
X26       int64
X27       int64
X28       int64
X29       int64
X30       int64
X31       int64
          ...
X355      int64
X356      int64
X357      int64
X358      int64
X359      int64
X360      int64
X361      int64
X362      int64
X363      int64
X364      int64
X365      int64
X366      int64
X367      int64
X368      int64
X369      int64
X370      int64
X371      int64
X372      int64
X373      int64
X374      int64
X375      int64
X376      int64
X377      int64
X378      int64
X379      int64
X380      int64
X382      int64
X383      int64
```

```
X384      int64
X385      int64
Length: 377, dtype: object
```

In [57]:

```
df.shape
```

Out[57]:

```
(4209, 377)
```

In [58]:

```
y = df.iloc[:,1].values
```

In [59]:

```
X = df.iloc[:,2:377].values
```

In [60]:

```
X.shape
```

Out[60]:

```
(4209, 375)
```

Label Enncoder

Converting the nomial features object data types into numerical

In [61]:

```
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
encoder = LabelEncoder()
X[:,0] = encoder.fit_transform(X[:,0])
X[:,1] = encoder.fit_transform(X[:,1])
X[:,2] = encoder.fit_transform(X[:,2])
X[:,3] = encoder.fit_transform(X[:,3])
X[:,4] = encoder.fit_transform(X[:,4])
X[:,5] = encoder.fit_transform(X[:,5])
X[:,6] = encoder.fit_transform(X[:,6])
X[:,7] = encoder.fit_transform(X[:,7])
```

In [62]:

```
X
```

Out[62]:

```
array([[32, 23, 17, ..., 0, 0, 0],
       [32, 21, 19, ..., 0, 0, 0],
       [20, 24, 34, ..., 0, 0, 0],
       ...,
       [8, 23, 38, ..., 0, 0, 0],
       [9, 19, 25, ..., 0, 0, 0],
       [46, 19, 3, ..., 0, 0, 0]], dtype=object)
```

In [63]:

```
#X = pd.DataFrame(X)
```

In [64]:

```
X.shape
```

Out[64]:

```
(4209, 375)
```

PCA dimensionality reduction

In [65]:

```
from sklearn.decomposition import PCA
```

In [66]:

```
X_centered = X - X.mean(axis=0)
```

In [67]:

```
pca = PCA(n_components=3)
pca.fit(X_centered)
```

Out[67]:

```
PCA(copy=True, iterated_power='auto', n_components=3, random_state=N
one,
  svd_solver='auto', tol=0.0, whiten=False)
```

In [68]:

```
X_pca = pca.transform(X_centered)
```

In [69]:

```
X_pca.shape
```

Out[69]:

```
(4209, 3)
```

In [70]:

```
pca.components_
```

Out[70]:

```
array([[-9.31138471e-01,  2.45494127e-01,  2.57713643e-01, ...,
         2.84628944e-06, -9.56497803e-06, -2.86842355e-05],
       [ 2.63046285e-01, -1.92186472e-02,  9.59720661e-01, ...,
        -9.23858305e-05, -5.16075133e-07, -1.32382564e-05],
       [ 1.40346165e-01,  5.51248020e-01,  2.15461586e-03, ...,
        -1.13659802e-04,  4.64816456e-05,  1.61546453e-04]])
```

In [71]:

```
pca.explained_variance_
```

Out[71]:

```
array([204.02462081, 113.83096499,  70.58204097])
```

In [72]:

```
pca.explained_variance_ratio_
```

Out[72]:

```
array([0.38334782, 0.21388033, 0.13261866])
```

Splitting the dataset into traning and testing

70% of the dataset is goes for training

30% of the dataset is goes for testing

In [73]:

```
%time
# Splitting the dataset
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_pca,y,test_size =0.3,random_s
tate=1)
```

```
CPU times: user 4 µs, sys: 1 µs, total: 5 µs
Wall time: 10 µs
```

In [74]:

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2946, 3)
(1263, 3)
(2946,)
(1263,)
```

Random Forest Regressor

In [79]:

```
%time
# Importing the RF model from scikit learn ensemble
from sklearn.ensemble import RandomForestRegressor
classifier = RandomForestRegressor()
model = classifier.fit(X_train,y_train)
model
```

```
CPU times: user 6 µs, sys: 3 µs, total: 9 µs
Wall time: 15.3 µs
```

Out[79]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=Non
e,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=Non
e,
           oob_score=False, random_state=None, verbose=0, warm_start
=False)
```

In [80]:

```
# Predicting the values on independent variables testing dataset
y_pred = classifier.predict(X_test)
```

In [81]:

```
# Model evaluation metrics for regression

print('Mean Abs Error   MAE    : ', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Sq  Error MSE      : ', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Sq Error RMSE : ', np.sqrt(metrics.mean_squared_error(y_test, y
_pred)))
print('r2 value                : ', metrics.r2_score(y_test, y_pred))
```

```
Mean Abs Error   MAE    :  7.171028496022322
Mean Sq  Error MSE      :  102.07154443307036
Root Mean Sq Error RMSE :  10.103046294710836
r2 value                :  0.3188832345478644
```

XGBoost Model Regressor

In [82]:

```
%time
# Importing the XGBoost model from scikit learn ensemble
from xgboost import XGBRegressor
classifier = XGBRegressor(n_estimator =1000)
classifier.fit(X_train,y_train)
```

```
CPU times: user 6 µs, sys: 1e+03 ns, total: 7 µs
Wall time: 13.8 µs
```

Out[82]:

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, importance_type='gain',
       learning_rate=0.1, max_delta_step=0, max_depth=3,
       min_child_weight=1, missing=None, n_estimator=1000,
       n_estimators=100, n_jobs=1, nthread=None, objective='reg:line
ar',
       random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=
1,
       seed=None, silent=True, subsample=1)
```

In [83]:

```
# Predicting the values on independent variables testing dataset
y_pred = classifier.predict(X_test)
```

In [84]:

```
# Model evaluation metrics for regression

print('Mean Abs Error   MAE    : ', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Sq  Error MSE      : ', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Sq Error RMSE : ', np.sqrt(metrics.mean_squared_error(y_test, y
_pred)))
print('r2 value                : ', metrics.r2_score(y_test, y_pred))
```

```
Mean Abs Error   MAE    :  7.5001429227827465
Mean Sq  Error MSE      :  98.57131012135072
Root Mean Sq Error RMSE :  9.92830852267146
r2 value                :  0.34224007004951906
```

The Champion model out of all Models is XGBoost