Mercedes-Benz Greener Manufacturing

Analysis and Model building by

Meet Hariyani

- **Problem Statement Scenario:**
  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 a 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 is the leader in the premium car industry. With a huge selection of features and options, customers can choose the customized Mercedes-Benz of their dreams
- to ensure the safety and reliability of every unique car configuration before they hit the road, the company's engineers have developed a robust testing system. As one of the world's biggest manufacturers of premium cars, safety and efficiency are paramount on Mercedes-Benz's production lines. However, optimizing the speed of their testing system for many possible feature combinations is complex and time-consuming without a powerful algorithmic approach.

- Basic python programing-importing library and loading data

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [2]:
```
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

In [3]:
```
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
```

In [4]:

```python
display(train_df)
```

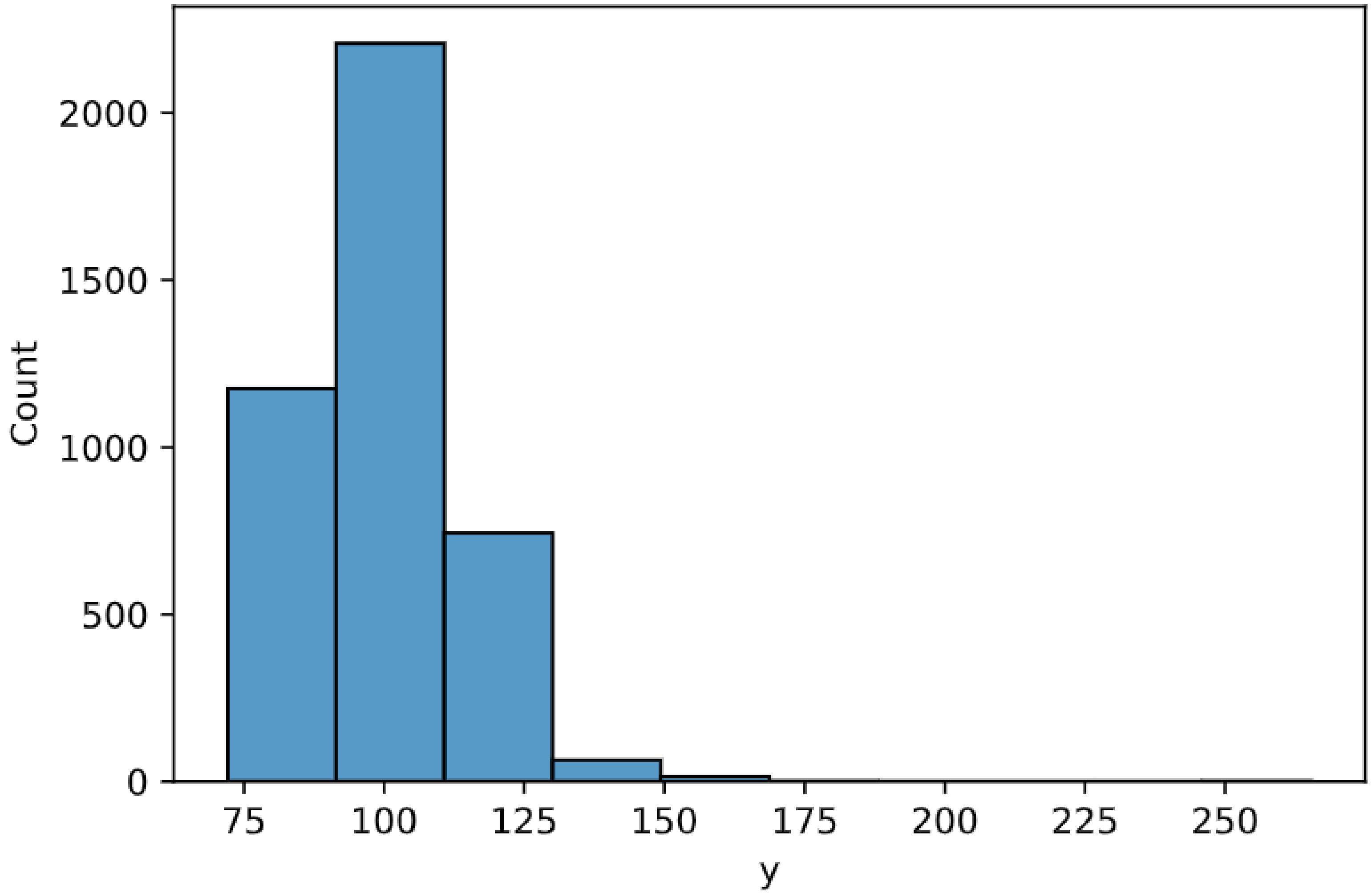
	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	X24	X26	X27	X28	X29	X30	X31	X32	X33
0	0	130.81	k	v	at	a	d	u	j	o	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0
1	6	88.53	k	t	av	e	d	y	l	o	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
2	7	76.26	az	w	n	c	d	x	j	x	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0
3	9	80.62	az	t	n	f	d	x	l	e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0
4	13	78.02	az	v	n	f	d	h	d	n	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8405	107.39	ak	s	as	c	d	aa	d	q	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4205	8406	108.77	j	o	t	d	d	aa	h	h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4206	8412	109.22	ak	v	r	a	d	aa	g	e	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4207	8415	87.48	al	r	e	f	d	aa	l	u	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4208	8417	110.85	z	r	ae	c	d	aa	g	w	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

4209 rows × 378 columns

- Label –Y distribution

```
In [5]: sns.histplot(data=train_df,x='y',bins=10)

Out[5]: <AxesSubplot:xlabel='y', ylabel='Count'>
```



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

In [6]:
```
cols = [c for c in train_df.columns if 'X' in c]
print('Number of features: {}'.format(len(cols)))

print('Feature types:')
train_df[cols].dtypes.value_counts()
```

```
Number of features: 376
Feature types:
```

Out[6]:
```
int64     368
object      8
dtype: int64
```

In [7]:
```
train_df.shape
```

Out[7]:
```
(4209, 378)
```

In [17]:
```
cat_features = train_df.columns[train_df.dtypes == 'object']
cont_features = train_df.columns[train_df.dtypes != 'object']
```

In [9]:
```
zero_var = []
for i in cols:
    typ = train_df[i].dtype
    uniq = len(np.unique(train_df[i]))
    if uniq == 1: zero_var.append(i)
```

In [10]:
```
train_df = train_df.drop(zero_var,axis = 1)
```

In [18]:
```
cat_features
cont_features
```

Out[18]:
```
Index(['ID', 'y', 'X10', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18',
       ...
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384', 'X385'], dtype='object', length=358)
```

In [19]:
```
cat_features
```

Out[19]:
```
Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype='object')
```

- Here removed all the features having zero variance data.

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

```
In [20]: train_df[cat_features].isnull().sum()

Out[20]: X0    0
         X1    0
         X2    0
         X3    0
         X4    0
         X5    0
         X6    0
         X8    0
         dtype: int64

In [23]: train_df[cont_features].isnull().sum()

Out[23]: ID      0
         y       0
         X10     0
         X12     0
         X13     0
         X14     0
         X15     0
         X16     0
         X17     0
         X18     0
```

- Here no null values found in datasets.

```
In [72]: for i in cat_features:
             print(train_df[i].unique())
```

```
['k' 'az' 't' 'al' 'o' 'w' 'j' 'h' 's' 'n' 'ay' 'f' 'x' 'y' 'aj' 'ak' 'am'
 'z' 'q' 'at' 'ap' 'v' 'af' 'a' 'e' 'ai' 'd' 'aq' 'c' 'aa' 'ba' 'as' 'i'
 'r' 'b' 'ax' 'bc' 'u' 'ad' 'au' 'm' 'l' 'aw' 'ao' 'ac' 'g' 'ab']
['v' 't' 'w' 'b' 'r' 'l' 's' 'aa' 'c' 'a' 'e' 'h' 'z' 'j' 'o' 'u' 'p' 'n'
 'i' 'y' 'd' 'f' 'm' 'k' 'g' 'q' 'ab']
['at' 'av' 'n' 'e' 'as' 'aq' 'r' 'ai' 'ak' 'm' 'a' 'k' 'ae' 's' 'f' 'd'
 'ag' 'ay' 'ac' 'ap' 'g' 'i' 'aw' 'y' 'b' 'ao' 'al' 'h' 'x' 'au' 't' 'an'
 'z' 'ah' 'p' 'am' 'j' 'q' 'af' 'l' 'aa' 'c' 'o' 'ar']
['a' 'e' 'c' 'f' 'd' 'b' 'g']
['d' 'b' 'c' 'a']
['u' 'y' 'x' 'h' 'g' 'f' 'j' 'i' 'd' 'c' 'af' 'ag' 'ab' 'ac' 'ad' 'ae'
 'ah' 'l' 'k' 'n' 'm' 'p' 'q' 's' 'r' 'v' 'w' 'o' 'aa']
['j' 'l' 'd' 'h' 'i' 'a' 'g' 'c' 'k' 'e' 'f' 'b']
['o' 'x' 'e' 'n' 's' 'a' 'h' 'p' 'm' 'k' 'd' 'i' 'v' 'j' 'b' 'q' 'w' 'g'
 'y' 'l' 'f' 'u' 'r' 't' 'c']
```

- Categorical features unique data.

```
In [73]: for i in cont_features:
             print(train_df[i].unique())
```

```
[   0    6    7 ... 8412 8415 8417]
[130.81  88.53  76.26 ...  85.71 108.77  87.48]
[0 1]
[0]
[0 1]
[1 0]
[0 1]
[0 1]
[0 1]
[0 1]
[1 0]
[0 1]
```

- Constant features unique data.

- Apply label encoder.

```
In [33]: from sklearn.preprocessing import LabelEncoder
```

```
In [34]: for i in train_df.columns:
             if train_df[i].dtype == 'object':
                 le = LabelEncoder()
                 le.fit(list(train_df[i].values) + list(test_df[i].values))
                 train_df[i] = le.transform(list(train_df[i].values))
                 test_df[i] = le.transform(list(test_df[i].values))
```

- Here, object data typed data are encoded using using sklearn LableEncoder.

In [35]:
```
train_df.head()
```

Out[35]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	X24	X26	X27	X28	X29	X30	X31	X32	X33	X34	X35	X
0	0	130.81	37	23	20	0	3	27	9	14	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	
1	6	88.53	37	21	22	4	3	31	11	14	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	
2	7	76.26	24	24	38	2	3	30	9	23	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	
3	9	80.62	24	21	38	5	3	30	11	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	
4	13	78.02	24	23	38	5	3	14	3	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	

In [36]:
```
test_df.head()
```

Out[36]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	X24	X26	X27	X28	X29	X30	X31	X32	X33	X34	X35	X36
0	1	24	23	38	5	3	26	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	0
1	2	46	3	9	0	3	9	6	24	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
2	3	24	23	19	5	3	0	9	9	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0
3	4	24	13	38	5	3	32	11	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	0
4	5	49	20	19	2	3	31	8	12	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

- Perform dimensionality reduction.

```
In [38]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()

In [39]: from sklearn.decomposition import PCA

In [40]: pca = PCA(n_components=10)

In [41]: pca.fit(train_df)

Out[41]: PCA(n_components=10)

In [42]: x_pca = pca.transform(train_df)

In [45]: x_pca

Out[45]: array([[-4.20592652e+03,  1.01640956e+01,  2.55370854e+01, ...,
                 -2.71103174e+00, -3.48141593e+00,  2.49028559e+00],
                [-4.19990596e+03, -4.96310894e+00, -1.29993620e+01, ...,
                 -4.52329560e+00,  1.37759399e-01,  6.38122725e-01],
                [-4.19890451e+03,  6.18728375e+00, -2.84546239e+01, ...,
                 -2.26974386e+00, -2.06258692e+00,  2.75547112e-01],
                ...,
                [ 4.20599490e+03,  3.23557817e+01, -2.68189962e-01, ...,
                  8.30196749e-01, -2.98590515e+00, -4.77935778e-01],
                [ 4.20900134e+03,  1.94677539e+01, -1.97993826e+01, ...,
                 -4.32208552e+00,  2.46488810e+00,  1.25247467e+00],
                [ 4.21099467e+03, -1.39528399e+01,  1.49205370e+01, ...,
                  5.96116972e-01, -6.02696998e-01,  1.16720152e+00]])
```

- Here dimensionality reduction done using sklearn’s PCA module.

- Predict your test_df values using XGBoost.

```
In [46]: usable_columns = list(set(train_df.columns) - set(['ID', 'y']))
```

```
In [47]: y_train = train_df['y'].values
         id_test = test_df['ID'].values

         x_train = train_df[usable_columns]
         x_test = test_df[usable_columns]
```

```
In [57]: import xgboost as xgb
         from sklearn.metrics import r2_score
         from sklearn.model_selection import train_test_split
```

```
In [58]: x_train, x_valid, y_train, y_valid = train_test_split(x_train, y_train, test_size=0.2, random_state=4242)
```

```
In [59]: d_train = xgb.DMatrix(x_train, label=y_train)
         d_valid = xgb.DMatrix(x_valid, label=y_valid)
         d_test = xgb.DMatrix(x_test)
```

In [60]:

```
params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.02
params['max_depth'] = 4
```

In [61]:

```
def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)
```

In [62]:

```
watchlist = [(d_train, 'train'), (d_valid, 'valid')]
```

In [63]:

```python
clf = xgb.train(params, d_train, 1000, watchlist, early_stopping_rounds=50, feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

```
[15:33:09] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squarederror.
[0]	train-rmse:99.13972	train-r2:-58.34264	valid-rmse:98.25375	valid-r2:-67.62468
[10]	train-rmse:81.18322	train-r2:-38.79276	valid-rmse:80.27140	valid-r2:-44.80403
[20]	train-rmse:66.54102	train-r2:-25.73316	valid-rmse:65.59669	valid-r2:-29.58764
[30]	train-rmse:54.61490	train-r2:-17.00917	valid-rmse:53.63052	valid-r2:-19.44589
[40]	train-rmse:44.91724	train-r2:-11.18141	valid-rmse:43.88424	valid-r2:-12.68986
[50]	train-rmse:37.05078	train-r2:-7.28831	valid-rmse:35.95870	valid-r2:-8.19158
[60]	train-rmse:30.69124	train-r2:-4.68722	valid-rmse:29.52966	valid-r2:-5.19867
[70]	train-rmse:25.57383	train-r2:-2.94878	valid-rmse:24.34061	valid-r2:-3.21158
[80]	train-rmse:21.48163	train-r2:-1.78616	valid-rmse:20.17221	valid-r2:-1.89260
[90]	train-rmse:18.23735	train-r2:-1.00815	valid-rmse:16.85077	valid-r2:-1.01847
[100]	train-rmse:15.69270	train-r2:-0.48685	valid-rmse:14.23308	valid-r2:-0.44006
[110]	train-rmse:13.72521	train-r2:-0.13739	valid-rmse:12.19272	valid-r2:-0.05678
[120]	train-rmse:12.22813	train-r2:0.09720	valid-rmse:10.63564	valid-r2:0.19590
[130]	train-rmse:11.10635	train-r2:0.25524	valid-rmse:9.47612	valid-r2:0.36167
[140]	train-rmse:10.28220	train-r2:0.36167	valid-rmse:8.63676	valid-r2:0.46975
[150]	train-rmse:9.68828	train-r2:0.43328	valid-rmse:8.03852	valid-r2:0.54066
[160]	train-rmse:9.26360	train-r2:0.48188	valid-rmse:7.62438	valid-r2:0.58677
[170]	train-rmse:8.95951	train-r2:0.51534	valid-rmse:7.35496	valid-r2:0.61546
[180]	train-rmse:8.74399	train-r2:0.53837	valid-rmse:7.18265	valid-r2:0.63327
[190]	train-rmse:8.59446	train-r2:0.55403	valid-rmse:7.07482	valid-r2:0.64419
[200]	train-rmse:8.48856	train-r2:0.56495	valid-rmse:7.00931	valid-r2:0.65075
[210]	train-rmse:8.41320	train-r2:0.57264	valid-rmse:6.97180	valid-r2:0.65448
[220]	train-rmse:8.35398	train-r2:0.57864	valid-rmse:6.95254	valid-r2:0.65639
[230]	train-rmse:8.30947	train-r2:0.58311	valid-rmse:6.94854	valid-r2:0.65678
[240]	train-rmse:8.27603	train-r2:0.58646	valid-rmse:6.95118	valid-r2:0.65652
[250]	train-rmse:8.24892	train-r2:0.58917	valid-rmse:6.95731	valid-r2:0.65592
[260]	train-rmse:8.22658	train-r2:0.59139	valid-rmse:6.96898	valid-r2:0.65476
[270]	train-rmse:8.20336	train-r2:0.59369	valid-rmse:6.97885	valid-r2:0.65378
[280]	train-rmse:8.18688	train-r2:0.59532	valid-rmse:6.99017	valid-r2:0.65266
```

```
In [64]: prediction = clf.predict(d_test)
```

```
In [69]: result = pd.DataFrame()
         result['ID'] = id_test
         result['Y'] = prediction
```

```
In [72]: result.head(10)
```

Out[72]:

	ID	Y
0	1	87.989944
1	2	104.646851
2	3	87.957138
3	4	76.859322
4	5	110.633369
5	8	91.903297
6	10	110.751991
7	11	93.979599
8	12	116.240417
9	14	94.083984

- Here Xgboost model used to predict Y label of test data.
- Model was trained with train data set.