Task :

Reduce the time a Mercedes-Benz spends on the test bench.

Problem Statement :

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.

You are required to reduce the time that cars spend on the test bench. Others will work with a dataset representing different permutations of features in a Mercedes-Benz car to predict the time it takes to pass testing. Optimal algorithms will contribute to faster testing, resulting in lower carbon dioxide emissions without reducing Mercedes-Benz's standards.

Following actions should be performed ----------

1. If for any column(s), the variance is equal to zero, then you need to remove those variable(s).
2. Check for null and unique values for test and train sets.
3. Apply label encoder.
4. Perform dimensionality reduction.
5. Predict your test_df values using XGBoost.

• IMPORT REQUIRED LIBRARIES . . .

In [66]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

• IMPORT & REVIEW THE DATA

In [67]:
```
df_train = pd.read_csv('Benz_Train.csv')
df_test = pd.read_csv('Benz_Test.csv')
```

In [68]:
```
print('Shape of df_train -', df_train.shape)
print('Shape of df_test -', df_test.shape)
```

```
Shape of df_train - (4209, 378)
Shape of df_test - (4209, 377)
```

In [69]:
```
df_train.describe()
```

Out[69]:

	ID	y	X10	X11	X12	X13	X14	
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	4209.000000	4209.000000	4209.0
mean	4205.960798	100.669318	0.013305	0.0	0.075077	0.057971	0.428130	0.0
std	2437.608688	12.679381	0.114590	0.0	0.263547	0.233716	0.494867	0.0
min	0.000000	72.110000	0.000000	0.0	0.000000	0.000000	0.000000	0.0
25%	2095.000000	90.820000	0.000000	0.0	0.000000	0.000000	0.000000	0.0
50%	4220.000000	99.150000	0.000000	0.0	0.000000	0.000000	0.000000	0.0
75%	6314.000000	109.010000	0.000000	0.0	0.000000	0.000000	1.000000	0.0
max	8417.000000	265.320000	1.000000	0.0	1.000000	1.000000	1.000000	1.0

8 rows × 370 columns

In [70]:
```
df_test.describe()
```

Out[70]:

	ID	X10	X11	X12	X13	X14	X15
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000
mean	4211.039202	0.019007	0.000238	0.074364	0.061060	0.427893	0.000713
std	2423.078926	0.136565	0.015414	0.262394	0.239468	0.494832	0.026691
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2115.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	4202.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	6310.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
max	8416.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 369 columns

In [71]:
```
df_train.columns
```

Out[71]:
```
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)
```

In [72]:
```
df_test.columns
```

Out[72]:
```
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)
```

• **Remove Unwanted Columns from Dataset . . .**

In [73]:
```
df_train.drop(['ID'], axis=True)
df_test.drop(['ID'], axis=True)
```

Out[73]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X3
0	az	v	n	f	d	t	a	w	0	0	...	0	0	0	1	0	0	
1	t	b	ai	a	d	b	g	y	0	0	...	0	0	1	0	0	0	
2	az	v	as	f	d	a	j	j	0	0	...	0	0	0	1	0	0	
3	az	l	n	f	d	z	l	n	0	0	...	0	0	0	1	0	0	
4	w	s	as	c	d	y	i	m	0	0	...	1	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4204	aj	h	as	f	d	aa	j	e	0	0	...	0	0	0	0	0	0	
4205	t	aa	ai	d	d	aa	j	y	0	0	...	0	1	0	0	0	0	
4206	y	v	as	f	d	aa	d	w	0	0	...	0	0	0	0	0	0	
4207	ak	v	as	a	d	aa	c	q	0	0	...	0	0	1	0	0	0	
4208	t	aa	ai	c	d	aa	g	r	0	0	...	1	0	0	0	0	0	

4209 rows × 376 columns

1. IF FOR ANY COLUMN(S), THE VARIANCE IS EQUAL TO ZERO, THEN YOU NEED TO REMOVE THOSE VARIABLE(S) . . .

In [74]:
```
df_train.var() [df_train.var()==0]
```

Out[74]:
```
X11     0.0
X93     0.0
X107    0.0
X233    0.0
X235    0.0
X268    0.0
X289    0.0
X290    0.0
X293    0.0
X297    0.0
X330    0.0
X347    0.0
dtype: float64
```

In [75]:
```
df_train = df_train.drop(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X29
```

In [76]:
```
df_train.shape
```

Out[76]:
```
(4209, 366)
```

In [77]:
```
df_test.var() [df_test.var()==0]
```

Out[77]:
```
X257    0.0
X258    0.0
X295    0.0
X296    0.0
X369    0.0
dtype: float64
```

In [78]:
```
df_test = df_test.drop(['X257', 'X258', 'X295', 'X296', 'X369'], axis=True)
```

In [79]:
```
df_test.shape
```

Out[79]:
```
(4209, 372)
```

2. CHECK FOR NULL AND UNIQUE VALUES FOR TEST AND TRAIN SETS

• **Check for Null Values . . .**

In [80]:
```
df_train.isna().any()
```

Out[80]:
```
ID      False
y       False
X0      False
X1      False
X2      False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 366, dtype: bool
```

In [81]:
```
df_test.isna().any()
```

Out[81]:
```
ID      False
X0      False
X1      False
X2      False
X3      False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 372, dtype: bool
```

► Null Values are not present in both the Datasets.

• **Check for Unique Values . . .**

In [89]:
```
df_train.describe(include=['object'])
```

Out[89]:

	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	w	g	j
freq	360	833	1659	1942	4205	231	1042	277

In [82]:

```
df_train_object = df_train.select_dtypes(include=['object'])
df_train_object.head()
```

Out[82]:

	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 [83]:

```
df_test.describe(include=['object'])
```

Out[83]:

	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

In [84]:

```
df_test_object = df_test.select_dtypes(include=['object'])
df_test_object.head()
```

Out[84]:

	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

3. APPLY LABEL ENCODER . . .

In [90]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [91]:

```
for i in df_train_object:
    le=LabelEncoder()
    le.fit_transform(list(df_train_object[i].values) + list(df_train_object[i].value
    df_train_object[i] = le.fit_transform(list(df_train_object[i].values))
```

In [92]:
```
df_train_object.head(10)
```

Out[92]:

	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
5	40	3	25	2	3	11	7	18
6	9	19	25	5	3	10	7	18
7	36	13	16	5	3	10	9	0
8	43	20	16	4	3	10	8	7
9	31	3	14	2	3	10	0	4

4. PERFORM DIMENSIONALITY REDUCTION . . .

In [88]:
```
from sklearn.decomposition import PCA
```

In [ ]:
```

```