DESCRIPTION

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

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.

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:

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.

In [1]:
```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.feature_selection import VarianceThreshold
variance=VarianceThreshold(threshold=0)
from sklearn.preprocessing import LabelEncoder
label=LabelEncoder
```

In [2]:
```
train=pd.read_csv('train.csv')
```

In [3]:
```
train.head(10)
```

Out[3]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X38
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	18	92.93	t	b	e	c	d	g	h	s	...	0	0	1	0	0	0	
6	24	128.76	al	r	e	f	d	f	h	s	...	0	0	0	0	0	0	
7	25	91.91	o	l	as	f	d	f	j	a	...	0	0	0	0	0	0	
8	27	108.67	w	s	as	e	d	f	i	h	...	1	0	0	0	0	0	
9	30	126.99	j	b	aq	c	d	f	a	e	...	0	0	1	0	0	0	

10 rows × 378 columns

In [4]:
```
test=pd.read_csv('test.csv')
```

In [5]:
```
test.head(10)
```

Out[5]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0
5	8	y	aa	ai	e	d	x	g	s	0	...	1	0	0	0	0	0	0
6	10	x	b	ae	d	d	x	d	y	0	...	0	0	0	0	0	1	0
7	11	f	s	ae	c	d	h	d	a	0	...	0	0	1	0	0	0	0
8	12	ap	l	s	c	d	h	j	n	0	...	0	0	0	0	0	0	0
9	14	o	v	as	f	d	g	f	v	0	...	0	0	0	0	0	0	0

10 rows × 377 columns

In [6]:
```
train.describe()
```

Out[6]:

	ID	y	X10	X11	X12	X13	X14	
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	4209.000000	4209.000000	4209
mean	4205.960798	100.669318	0.013305	0.0	0.075077	0.057971	0.428130	0
std	2437.608688	12.679381	0.114590	0.0	0.263547	0.233716	0.494867	0
min	0.000000	72.110000	0.000000	0.0	0.000000	0.000000	0.000000	0
25%	2095.000000	90.820000	0.000000	0.0	0.000000	0.000000	0.000000	0
50%	4220.000000	99.150000	0.000000	0.0	0.000000	0.000000	0.000000	0
75%	6314.000000	109.010000	0.000000	0.0	0.000000	0.000000	1.000000	0
max	8417.000000	265.320000	1.000000	0.0	1.000000	1.000000	1.000000	1

8 rows × 370 columns

In [7]:
```
test.describe()
```

Out[7]:

	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 [8]:
```
train.isnull().sum()
```

Out[8]:
```
ID      0
y       0
X0      0
X1      0
X2      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 378, dtype: int64
```

In [9]:
```
train_target=["y"]
train_data=train.drop(["y","ID"], axis=1)
```

In [10]:
```
train_data.head(10)
```

Out[10]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	0	0
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	0	0
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	0	1
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	0	0
4	az	v	n	f	d	h	d	n	0	0	...	0	0	0	0	0	0	0
5	t	b	e	c	d	g	h	s	0	0	...	0	0	1	0	0	0	0
6	al	r	e	f	d	f	h	s	0	0	...	0	0	0	0	0	0	0
7	o	l	as	f	d	f	j	a	0	0	...	0	0	0	0	0	0	0
8	w	s	as	e	d	f	i	h	0	0	...	1	0	0	0	0	0	0
9	j	b	aq	c	d	f	a	e	0	0	...	0	0	1	0	0	0	0

10 rows × 376 columns

In [11]:
```
train_data.var().sort_values().head(15)
```

Out[11]:
```
C:\Users\alwar\AppData\Local\Temp\ipykernel_7444\2491115096.py:1: FutureWarning: Dr
opping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid columns
before calling the reduction.
  train_data.var().sort_values().head(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
dtype: float64
```

In [12]:
```
train_data_without_zero_var=variance.fit_transform(train_data.iloc[:,9:])
```

In [13]:
```
train_data_without_zero_var
```

Out[13]:
```
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 [14]:
```
labeled_data=train_data.iloc[:,0:8]
```

In [15]:
```
labeled_data.head()
```

Out[15]:

	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 [16]:
```
labeled_data.nunique()
```

Out[16]:
```
X0    47
X1    27
X2    44
X3     7
X4     4
X5    29
X6    12
X8    25
dtype: int64
```

In [17]:
```
labeled_data1=labeled_data.apply(label().fit_transform)
```

In [18]:
```
labeled_data1.head()
```

Out[18]:

	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

In [19]:
```
labeled_data1.var()
```

Out[19]:
```
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 [20]:
```
train_data_Zero_var_final=pd.DataFrame(train_data_without_zero_var)
```

In [21]:
```
train_data_Zero_var_final
```

Out[21]:

	0	1	2	3	4	5	6	7	8	9	...	345	346	347	348	349	350	351	352	353	354
0	0	1	0	0	0	0	1	0	0	1	...	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	...	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	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	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	0	0	1	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0
4205	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
4206	1	1	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
4207	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4208	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0	0	0	0	0

4209 rows × 355 columns

In [22]:
```
final_train_data=pd.concat([labeled_data1,train_data_Zero_var_final], axis=1)
```

In [23]:
```
final_train_data.head()
```

Out[23]:

	X0	X1	X2	X3	X4	X5	X6	X8	0	1	...	345	346	347	348	349	350	351	352	353	3
0	32	23	17	0	3	24	9	14	0	1	...	0	0	1	0	0	0	0	0	0	
1	32	21	19	4	3	28	11	14	0	0	...	1	0	0	0	0	0	0	0	0	
2	20	24	34	2	3	27	9	23	0	0	...	0	0	0	0	0	0	1	0	0	
3	20	21	34	5	3	27	11	4	0	0	...	0	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	0	0	0	

5 rows × 363 columns

In [24]:
```
final_train_data.isnull().any()
```

Out[24]:
```
X0      False
X1      False
X2      False
X3      False
X4      False
        ...
350     False
351     False
352     False
353     False
354     False
Length: 363, dtype: bool
```

In [25]:
```
test=test.drop(['ID'],axis=1)
```

In [26]:
```
test.head()
```

Out[26]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382
0	az	v	n	f	d	t	a	w	0	0	...	0	0	0	1	0	0	0
1	t	b	ai	a	d	b	g	y	0	0	...	0	0	1	0	0	0	0
2	az	v	as	f	d	a	j	j	0	0	...	0	0	0	1	0	0	0
3	az	l	n	f	d	z	l	n	0	0	...	0	0	0	1	0	0	0
4	w	s	as	c	d	y	i	m	0	0	...	1	0	0	0	0	0	0

5 rows × 376 columns

In [27]:
```
test.nunique()
```

Out[27]:
```
X0      49
X1      27
X2      45
X3       7
X4       4
        ..
X380     2
X382     2
X383     2
X384     2
X385     2
Length: 376, dtype: int64
```

In [28]:
```
test.isnull().any()
```

Out[28]:
```
X0      False
X1      False
X2      False
X3      False
X4      False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 376, dtype: bool
```

In [29]:
```
test.var().sort_values().head(15)
```

```
C:\Users\alwar\AppData\Local\Temp\ipykernel_7444\1038450595.py:1: FutureWarning: Dr
opping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid columns
before calling the reduction.
  test.var().sort_values().head(15)
```

Out[29]:
```
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
dtype: float64
```

In [30]:
```
test_without_zero_var=variance.transform(test.iloc[:,9:])
```

In [31]:
```
test_without_zero_var
```

Out[31]:
```
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 [32]:
```
test_without_zero_var_final=pd.DataFrame(test_without_zero_var)
```

In [33]:
```
test_without_zero_var_final.head()
```

Out[33]:

	0	1	2	3	4	5	6	7	8	9	...	345	346	347	348	349	350	351	352	353	354
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	...	0	0	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0

5 rows × 355 columns

In [34]:
```
labeled_data1=test.iloc[:,0:8]
```

In [35]:
```
labeled_data1.head(10)
```

Out[35]:

	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
5	y	aa	ai	e	d	x	g	s
6	x	b	ae	d	d	x	d	y
7	f	s	ae	c	d	h	d	a
8	ap	l	s	c	d	h	j	n
9	o	v	as	f	d	g	f	v

In [36]:
```
test_label=labeled_data1.apply(label().fit_transform)
```

In [37]:
```
test_label.head(10)
```

Out[37]:

	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
5	47	1	8	4	3	29	6	18
6	46	3	4	3	3	29	3	24
7	29	20	4	2	3	14	3	0
8	12	13	38	2	3	14	9	13
9	38	23	17	5	3	13	5	21

In [38]:
```
test_data_final=pd.concat([test_label, test_without_zero_var_final], axis=1)
```

In [39]:
```
test_data_final.head(10)
```

Out[39]:

	X0	X1	X2	X3	X4	X5	X6	X8	0	1	...	345	346	347	348	349	350	351	352	353	3
0	21	23	34	5	3	26	0	22	0	0	...	0	0	0	1	0	0	0	0	0	
1	42	3	8	0	3	9	6	24	0	0	...	0	0	1	0	0	0	0	0	0	
2	21	23	17	5	3	0	9	9	0	0	...	0	0	0	1	0	0	0	0	0	
3	21	13	34	5	3	31	11	13	0	0	...	0	0	0	1	0	0	0	0	0	
4	45	20	17	2	3	30	8	12	0	0	...	1	0	0	0	0	0	0	0	0	
5	47	1	8	4	3	29	6	18	0	0	...	1	0	0	0	0	0	0	0	0	
6	46	3	4	3	3	29	3	24	0	0	...	0	0	0	0	0	1	0	0	0	
7	29	20	4	2	3	14	3	0	0	1	...	0	0	1	0	0	0	0	0	0	
8	12	13	38	2	3	14	9	13	0	0	...	0	0	0	0	0	0	0	0	0	
9	38	23	17	5	3	13	5	21	0	0	...	0	0	0	0	0	0	0	0	0	

10 rows × 363 columns

Dimensionality Reduction

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
final_train_data.shape
```

In [ ]:
```
train_target.shape
```

In [ ]:
```
x_train, x_test, y_train, y_test= train_test_split(final_train_data, train_target,
```

In [ ]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

In [ ]:
```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
```

In [ ]:
```
x_train=pca.fit_transform(x_train)
x_test=pca.transform(x_test)
test_data_final=pca.transform(test_data_final)
```

XGBoost

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

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

In [ ]:
```
ypred_test=model.predict(x_test)
ypred_test
```

In [ ]:
```
ypred_train=model.predict(x_train)
ypred_train
```

In [ ]:
```
print(r2_score(ypred_train, y_train))
```

In [ ]:
```
print(mean_squared_error(ypred_train, y_train))
```

In [ ]:
```
test_data_final_prediction=model.predict(test_data_final)
test_data_final_prediction
```

In [ ]:
```
prediction=pd.DataFrame({'ytest': y_test, 'ypred': ypred_test})
```

In [ ]:
```
plt.plot(prediction['ytest'], color='red')
plt.plot(prediction['ypred'], color='blue')
plt.show()
```