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

Problem Statement Scenario:

Mercedes-Benz engineers have developed a robust testing system, to ensure the safety and reliability of every unique car configuration before they hit the road. However, optimizing the speed of their testing system for many possible feature combinations is complex and time-consuming without a powerful algorithmic approach. Optimal algorithms will contribute to faster testing, resulting in lower carbon dioxide emissions without reducing Daimler'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

In [1]:
```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [2]:
```
# read data into a DataFrame
df_train_data = pd.read_csv('train.csv')
df_test_data = pd.read_csv('test.csv')
```

In [3]:
```
print('Train dataset: \n')
df_train_data.head()
```

```
Train dataset:
```

Out[3]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X3
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	0	
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	0	0	
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	0	
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	0	0	
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	0	0	

5 rows × 378 columns

In [4]:
```
print('Test dataset: \n')
df_test_data.head()
```

```
Test dataset:
```

Out[4]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	0	0

5 rows × 377 columns

Data Preparation : Drop column ID as this is not relevant to prediction

In [5]:
```
df_train_data = df_train_data.drop(['ID'], axis = 1)
df_test_data = df_test_data.drop(['ID'], axis = 1)
```

In [6]:
```
print('Train dataset: \n')
df_train_data.head()
```

```
Train dataset:
```

Out[6]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X
0	130.81	k	v	at	a	d	u	j	o	0	...	0	0	1	0	0	0	0	0	
1	88.53	k	t	av	e	d	y	l	o	0	...	1	0	0	0	0	0	0	0	
2	76.26	az	w	n	c	d	x	j	x	0	...	0	0	0	0	0	0	1	0	
3	80.62	az	t	n	f	d	x	l	e	0	...	0	0	0	0	0	0	0	0	
4	78.02	az	v	n	f	d	h	d	n	0	...	0	0	0	0	0	0	0	0	

5 rows × 377 columns

In [7]:
```
print('Test dataset: \n')
df_test_data.head()
```

```
Test dataset:
```

Out[7]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382	X383	X38
0	az	v	n	f	d	t	a	w	0	0	...	0	0	0	1	0	0	0	0	
1	t	b	ai	a	d	b	g	y	0	0	...	0	0	1	0	0	0	0	0	
2	az	v	as	f	d	a	j	j	0	0	...	0	0	0	1	0	0	0	0	
3	az	l	n	f	d	z	l	n	0	0	...	0	0	0	1	0	0	0	0	
4	w	s	as	c	d	y	i	m	0	0	...	1	0	0	0	0	0	0	0	

5 rows × 376 columns

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

In [8]:
```
columns_with_zero_var = df_train_data.var()[df_train_data.var()==0].index.values
columns_with_zero_var
```

Out[8]:
```
array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'], dtype=object)
```

In [9]:
```
df_train_data = df_train_data.drop(['X11','X93','X107','X233','X235','X268','X28
9','X290','X293','X297','X330','X347'], axis = 1)
df_test_data = df_test_data.drop(['X11','X93','X107','X233','X235','X268','X289
','X290','X293','X297','X330','X347'], axis = 1)
```

In [10]:
```
df_train_data.shape
```

Out[10]:
```
(4209, 365)
```

In [11]:
```
df_test_data.shape
```

Out[11]:
```
(4209, 364)
```

Task 2. Check for null and unique values for test and train sets

In [12]:
```
#check for null values in train dataset
np.sum(df_train_data.isnull().sum())
```

Out[12]:
```
0
```

In [13]:
```
#check for null values in test dataset
np.sum(df_test_data.isnull().sum())
```

Out[13]:
```
0
```

In [14]:
```
#check for unique values in train dataset
np.sum(df_train_data.nunique().sum())
```

Out[14]: 3452

In [15]:
```
#check for unique values in train dataset
np.sum(df_test_data.nunique().sum())
```

Out[15]: 908

Both train & test dataset does not contain any null values. Whereas, they got 3452 & 908 unique values respectively

Task 3. Apply label encoder

In [16]:
```
# find the columns having datatype as object
object_columns = df_train_data.describe(include=[object]).columns.values
object_columns
```

Out[16]: array(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype=object)

In [17]:
```
le = LabelEncoder()
for col in object_columns:
    le.fit(df_train_data[col].append(df_test_data[col]).values)
    df_train_data[col] = le.transform(df_train_data[col])
    df_test_data[col] = le.transform(df_test_data[col])
```

In [18]:
```
# create X and y
X = df_train_data.drop(['y'], axis=1)
y = df_train_data.y

#create train and test split
from sklearn import model_selection
xtrain,xtest,ytrain,ytest = model_selection.train_test_split(X,y,test_size=0.3,r
andom_state=1)
```

Task 4. Perform dimensionality reduction

In [19]:
```
from sklearn.decomposition import PCA as sklearnPCA
pca = sklearnPCA(0.98, svd_solver='full')
sklearn_pca = pca.fit(X)
```

In [20]:
```
sklearn_pca.n_components_
```

Out[20]: 12

Using Principal component analysis (PCA), decomposition of the data i.e., 364 components with 98% variance to project it to a lower dimensional space is 12 components

In [21]:
```
sklearn_pca.explained_variance_ratio_
```

Out[21]:
```
array([0.40868988, 0.21758508, 0.13120081, 0.10783522, 0.08165248,
       0.0140934 , 0.00660951, 0.00384659, 0.00260289, 0.00214378,
       0.00209857, 0.00180388])
```

Task 5. Predict your test_df values using xgboost

In [22]:
```
pca_xtrain = pd.DataFrame(sklearn_pca.transform(xtrain))
pca_xtest = pd.DataFrame(sklearn_pca.transform(xtest))
pca_df_test_data = pd.DataFrame(sklearn_pca.transform(df_test_data))
```

In [23]:
```
import xgboost as xgb
xgb_regressor_model = xgb.XGBRegressor(objective = 'reg:squarederror', learning_
rate= 0.1 )
xgb_regressor_model.fit(pca_xtrain, ytrain)
```

Out[23]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.1, max_delta_step=0,
             max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=
0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact
',
             validate_parameters=1, verbosity=None)
```

In [24]:
```
xgb_regressor_model_predicted_y_test = xgb_regressor_model.predict(pca_xtest)
print(sqrt(mean_squared_error(ytest, xgb_regressor_model_predicted_y_test)))
```

```
8.555776313214766
```

In [25]:
```
xgb_RFregressor_model = xgb.XGBRFRegressor(objective = 'reg:squarederror', learn
ing_rate= 1)
xgb_RFregressor_model.fit(pca_xtrain, ytrain)
```

Out[25]:
```
XGBRFRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
               colsample_bytree=1, enable_categorical=False, gamma=0, gpu_id=-
1,
               importance_type=None, interaction_constraints='',
               learning_rate=1, max_delta_step=0, max_depth=6,
               min_child_weight=1, missing=nan, monotone_constraints='()',
               n_estimators=100, n_jobs=8, num_parallel_tree=100,
               objective='reg:squarederror', predictor='auto', random_state=0,
               reg_alpha=0, scale_pos_weight=1, tree_method='exact',
               validate_parameters=1, verbosity=None)
```

In [26]:
```
xgb_RFregressor_model_predicted_y_test = xgb_RFregressor_model.predict(pca_xtes
t)
print(sqrt(mean_squared_error(ytest, xgb_RFregressor_model_predicted_y_test)))
```

```
9.064875902434391
```

Compare to XGBRFRegressor model XGBRegressor model have better prediction