**Project on Machine Learning**

Name of the project:

**Mercedes-Benz Greener Manufacturing**

**Done by: Mrinmay Saha**

**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.

**Source Code with graphs an insights:**

#Importing Libaries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

**Importing train & test dataset**

#loading the datasets

train\_data=pd.read\_csv('train.csv')

test\_data=pd.read\_csv('test.csv')

print(train\_data.shape)

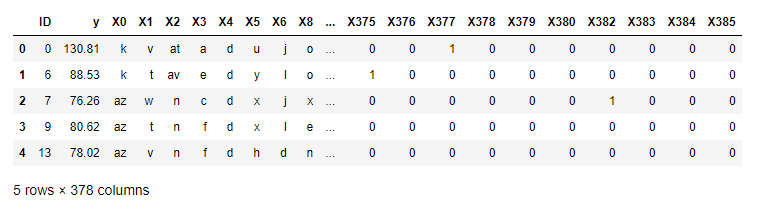
print(test\_data.shape)

Output: (4209, 378)

(4209, 377)

#Checking the dataset

train\_data.head()

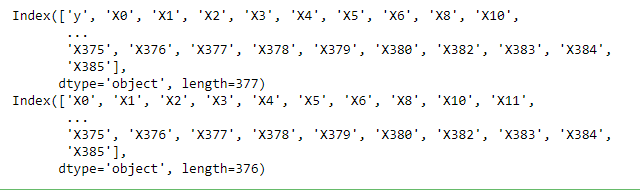
Output:

#drop the ID column as index as it is not needed for prediction

train\_data.drop('ID',inplace=True,axis=1)

test\_data.drop('ID',inplace=True,axis=1)

print(train\_data.columns)

print(test\_data.columns)

Output:

# Columns having variance zero, and remove those variable(s).

Zero\_var\_col = train\_data.var()[train\_data.var()==0].index.values

Zero\_var\_col

Output: array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',

'X293', 'X297', 'X330', 'X347'], dtype=object)

#deleting the columns with zero variance

for zero in Zero\_var\_col:

if zero in train\_data:

train\_data.drop(zero,axis=1,inplace=True)

for zero in Zero\_var\_col:

if zero in test\_data:

test\_data.drop(zero,axis=1,inplace=True)

print(train\_data.shape)

print(test\_data.shape)

Output: (4209, 365)

(4209, 364)

Insight:We can confirm by the shape of the data that the columns with zero variance has been removed.

# Checking the NAN values in the datasets

train\_data.isnull().sum().any()

Output: False

test\_data.isnull().sum().any()

Output: False

Insight:We can conclude from the above observation that the train and the test data do not have any null values.

# Apply label encoder on train & test datasets

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

object\_col=[]

for i in train\_data.columns:

data\_type = train\_data[i].dtype

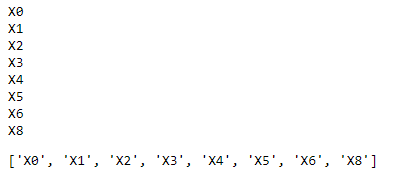
if data\_type == 'object':

print(i)

object\_col.append(i)

object\_col

Output:



for col in object\_col:

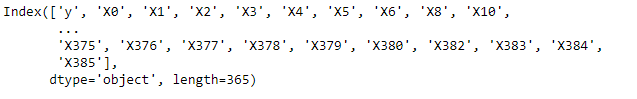
le.fit(train\_data[col].append(test\_data[col]).values)

train\_data[col]= le.transform(train\_data[col])

test\_data[col]= le.transform(test\_data[col])

train\_data.columns

Output:



# cross checking if lables are encoded to numbers

print(train\_data['X0'].unique())

Output: [37 24 46 11 41 49 36 34 45 40 23 32 50 51 9 10 12 52 43 18 15 48 6 0

31 8 30 16 29 1 26 17 35 44 25 22 28 47 4 19 39 38 21 14 3 33 2]

# Perform dimensionality reduction.

#importing libraries

from sklearn.decomposition import PCA

from sklearn.model\_selection import train\_test\_split

pca = PCA(n\_components = 0.98,svd\_solver='full')

#diving the features and the target columns

X = train\_data.drop('y',axis=1)

Y = train\_data['y']

#spillitng the train data set into train and test

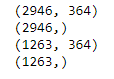
X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.3,random\_state=12)

print(X\_train.shape)

print(Y\_train.shape)

print(X\_test.shape)

print(Y\_test.shape)

Output:

#fitting the train dataset

pca.fit(X)

PCA(n\_components=0.98, svd\_solver='full')

#Transforming other datasets and reducing the dimensions.

pca\_X\_train = pd.DataFrame(pca.transform(X\_train))

pca\_X\_test = pd.DataFrame(pca.transform(X\_test))

pca\_test = pd.DataFrame(pca.transform(test\_data))

pca.n\_components\_

Output: 12

pca.explained\_variance\_ratio\_

Output: 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])

# Predicting using XGBoost

#importing required libaries

from sklearn.metrics import mean\_squared\_error

import xgboost as xgb

#creating the modelmodel = xgb.XGBRegressor(objective ='reg:linear', colsample\_bytree = 0.3, learning\_rate = 0.4, max\_depth = 10, alpha = 6,n\_estimators = 20)

#fitting the data

model.fit(pca\_X\_train,Y\_train)

#predict on the validaiton set

pred\_y\_test = model.predict(pca\_X\_test)

mean\_squared\_error(Y\_test,pred\_y\_test)

Output: 106.12790026771569

pred\_test\_data=model.predict(pca\_test)

pred\_test\_data

Output: array([ 81.70047 , 96.85091 , 92.18459 , ..., 93.24074 , 105.23896 ,89.607704], dtype=float32)

Conclusion:The mean squares error of the model is 106.12790026771569 and from using the Xgboost model now we can predict the test data.