**Mercedes-Benz Greener Manufacturing Project**

* First, we will import all the required libraries

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**%matplotlib inline**

* Then we will import Train and Test data sets using pandas and read\_csv

**df\_train= pd.read\_csv('train.csv')**

**df\_test= pd.read\_csv('test.csv')**

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

* For this we will use dtypes, describe() functions, then we will use np.var() for variance
* Then we will use For Loop on train and test dataframe along with iteritems() and append() and print Columns with variance is equal 0:
* Then will drop columns from Train and Test dataset with variance is equal to zero

**Code:**

**df\_train.dtypesdf\_train.describe()**

**train\_data = np.var(df\_train, axis=0)**

**train\_data**

**test\_data = np.var(df\_test, axis=0)**

**test\_data**

**train\_name=[]**

**for i in train\_data.iteritems():**

**if(i[1]==0):**

**train\_name.append(i[0])**

**print('Columns with variance is equal 0:')**

**print(train\_name)**

**test\_name=[]**

**for i in test\_data.iteritems():**

**if(i[1]==0):**

**test\_name.append(i[0])**

**print('Columns with variance is equal 0:')**

**print(test\_name)**

**df\_train.drop(train\_name, axis=1, inplace=True)**

**df\_test.drop(test\_name, axis=1, inplace=True)**

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

* For this we will use isnull().sum() to check for null values
* Then we will use desc.columns and use for loops to iterate between Index and get the unique values using unique() function

**Code:**

**for i,j in zip(df\_train.columns, df\_train.isnull().sum()):**

**if(j!=0):**

**print(i)**

**for i,j in zip(df\_test.columns, df\_test.isnull().sum()):**

**if(j!=0):**

**print(i)**

**train\_desc = df\_train.describe(include = 'O')**

**df\_train.describe(include = 'O')**

**train\_desc.columns**

**test\_desc = df\_test.describe(include='O')**

**df\_train.describe(include='O')test\_desc.columns**

**for i in ['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']:**

**print('df\_train')**

**print(i, df\_train[i].unique())**

**print('df\_test')**

**print(i, df\_test[i].unique())**

**3.Apply label encoder**

* We will import **LabelEncoder** from **sklearn.preprocessing**

**Code:**

**df\_train\_x = df\_train.drop(['ID','y'], axis=1)**

**df\_train\_y = df\_train['y']**

**print('df\_train')**

**print(i, df\_train[i].unique())**

**print('df\_test')**

**print(i, df\_test[i].unique())**

**df\_test\_x = df\_test.drop(['ID'], axis=1)**

**print(df\_test\_x.shape)**

**print(df\_test\_x.shape)**

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

**Code:**

**from sklearn.preprocessing import LabelEncoder**

**le= LabelEncoder()**

**for i in train\_desc.columns:**

**df\_train\_x[i] = le.fit\_transform(df\_train\_x[i])**

**for i in test\_desc.columns:**

**df\_test\_x[i] = le.fit\_transform(df\_test\_x[i])**

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

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(df\_train\_x, df\_train\_y,␣**

**,→test\_size=0.2)**

**print(x\_train.shape, y\_train.shape)**

**print(x\_test.shape, y\_test.shape)**

**4.Perform dimensionality reduction**

**Code:**

**from sklearn.decomposition import PCA**

**from xgboost import XGBRegressor**

**from sklearn.metrics import accuracy\_score**

**# PCA on train dataset**

**pca = PCA(n\_components=0.99)**

**x\_train\_trans = pca.fit\_transform(x\_train)**

**x\_test\_trans = pca.transform(x\_test)**

**print(x\_train.shape)**

**print(x\_train\_trans.shape)**

**print(x\_test.shape)**

**print(x\_test\_trans.shape)**

**# PCA on test dataset**

**pca = PCA(n\_components=0.99)**

**df\_train\_trans = pca.fit\_transform(df\_train\_x)**

**print(df\_train\_x.shape)**

**print(df\_train\_trans.shape)**

**5.Predict your test\_df values using XGBoost**

**Code:**

**xgb = XGBRegressor()**

**xgb.fit(x\_train\_trans, y\_train)**

**y\_pred = xgb.predict(x\_test\_trans)**

**xgb.predict(x\_test\_trans)**

**print('Train Accuracy:', xgb.score(x\_train\_trans, y\_train))**

**print('Test Accuracy:', xgb.score(x\_test\_trans, y\_test))**

**# Prediction on Testing File**

**xgb.predict(df\_train\_trans)**