Project Description (Mercedes-Benz Greener Manufacturing):

“””

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 of the Project:

```
#importing libraries

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from pandas.api.types import is_numeric_dtype
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import r2_score

#loading datasets

test_path=input("enter path for test dataset: ")
test_datasetpath=test_path.replace("\\",'/')
train_path=input("enter path for train dataset: ")
train_datasetpath=train_path.replace("\\",'/')
test=pd.read_csv(test_datasetpath)
train=pd.read_csv(train_datasetpath)
t=list(set(train.columns)-set(['ID','y']))
test_df=pd.DataFrame(test[t])
train_df=pd.DataFrame(train[t])

#checking for null values for both test and train datsets
```

```
print("Checking for null values in the train and test datasets:")
print("-----------------------------------------------------------")
if(test_df.isnull().values.any()):
  print("there are missing values in the test dataset")
  train_df.dropna()
  print("removed the missing values")
else:
  print("there are no missing values in the test dataset")
if(train_df.isnull().values.any()):
  print("there are missing values in the train dataset")
  train_df.dropna()
  print("removed the missing values")

else:
  print("there are no missing values in the train dataset")



#performing label encoder for test dataset and train datasets

print("\n")
print("Performing Label Encoder on the datasets:")
print("-----------------------------------------")
lb=LabelEncoder()
c=0
for i in test_df.columns:
  if not(is_numeric_dtype(test_df[i])):
    test_df[i]=lb.fit_transform(test_df[i])
```

```
for i in test_df.columns:
  if not(is_numeric_dtype(test_df[i])):
    print("label encoder not performed for ",i)
    c=c+1
if(c==0):
  print("label encoder operation is done successfully on test dataset and all the columns are tranformed to numeric")
else:
  print("label encoder operation failed on test dataset")

for i in train_df.columns:
  if not(is_numeric_dtype(train_df[i])):
    train_df[i]=lb.fit_transform(train_df[i])
for i in train_df.columns:
  if not(is_numeric_dtype(train_df[i])):
    print("label encoder not performed for ",i)
if(c==0):
  print("label encoder operation is done successfully on train dataset and all the columns are tranformed to numeric")
else:
  print("label encoder operation failed on train dataset")

#checking for variance is zero , if zero dropping the variable in test and train datasets

print("\n")
print("checking for variance is zero , if zero dropping the variable in test and train datasets:")
print("----------------------------------------------------------------------------------------")
```

```
for i in test_df.columns:
  if(test_df[i].var()==0):
    test_df[i].drop
    print(i)
print("these columns in test datset are removed")
for i in train_df.columns:
  if(train_df[i].var()==0):
    train_df[i].drop
    print(i)
print("these columns in train dataset are removed")
print("\n")

#performing dimensionality reduction

sklearn_pca=PCA(n_components=12)
x=train_df
x_test=test_df
y=train['y'].values
sklearn_pca.fit(x)
x_train_transformed=sklearn_pca.transform(x)
x_test_transformed=sklearn_pca.transform(x_test)
x_train,x_valid,y_train,y_valid=train_test_split(x_train_transformed,y)


#XGBoost Model

print("applying XGBoost Model:")
```

```
d_train=xgb.DMatrix(x_train,label=y_train)

d_valid=xgb.DMatrix(x_valid,label=y_valid)

d_test=xgb.DMatrix(x_test_transformed)


params={}

params['eta']=0.02

params['max_depth']=4


def xgb_r2_score(preds, dtrain):

    labels = dtrain.get_label()

    return 'r2', r2_score(labels, preds)


watchlist=[(d_train,'train'),(d_valid,'valid')]


clf=xgb.train(params,d_train,1000,watchlist,early_stopping_rounds=50,feval=xgb_r2_score,ma
ximize=True,verbose_eval=10)


#predict test_df values


y_predict=clf.predict(d_test)

print("\n")

print("predicted values:")

print("------------------")

print(y_predict)
```

Screenshots of the Output:

```
C:\Users\Swetha Thanjavur\Documents\AI\Machine Learning>python Mercedes-Benz.py
enter path for test dataset: C:\Users\Swetha Thanjavur\Documents\AI\Machine Learning\test.csv
enter path for train dataset: C:\Users\Swetha Thanjavur\Documents\AI\Machine Learning\train.csv
Checking for null values in the train and test datasets:
-------------------------------------------------------
there are no missing values in the test dataset
there are no missing values in the train dataset


Performing Label Encoder on the datasets:
-----------------------------------------
label encoder operation is done successfully on test dataset and all the columns are tranformed to numeric
label encoder operation is done successfully on train dataset and all the columns are tranformed to numeric


checking for variance is zero , if zero dropping the variable in test and train datasets:
-----------------------------------------------------------------------------------------
X296
X295
X258
X257
X369
these columns in test datset are removed
X107
X297
X11
X347
X293
X268
X289
X233
X330
X93
X235
X290
these columns in train dataset are removed
```

```
predicted values:
------------------
[ 77.35219  99.42287  78.80486 ...  96.68241 109.9597   94.5818 ]
```