Project 1 - Mercedes-Benz Greener Manufacturing

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 the 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 cars are leaders 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, Daimlers engineers have developed a robust testing system. As one of the worlds biggest manufacturers of premium cars, safety and efficiency are paramount on Daimlers 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 Daimlers 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

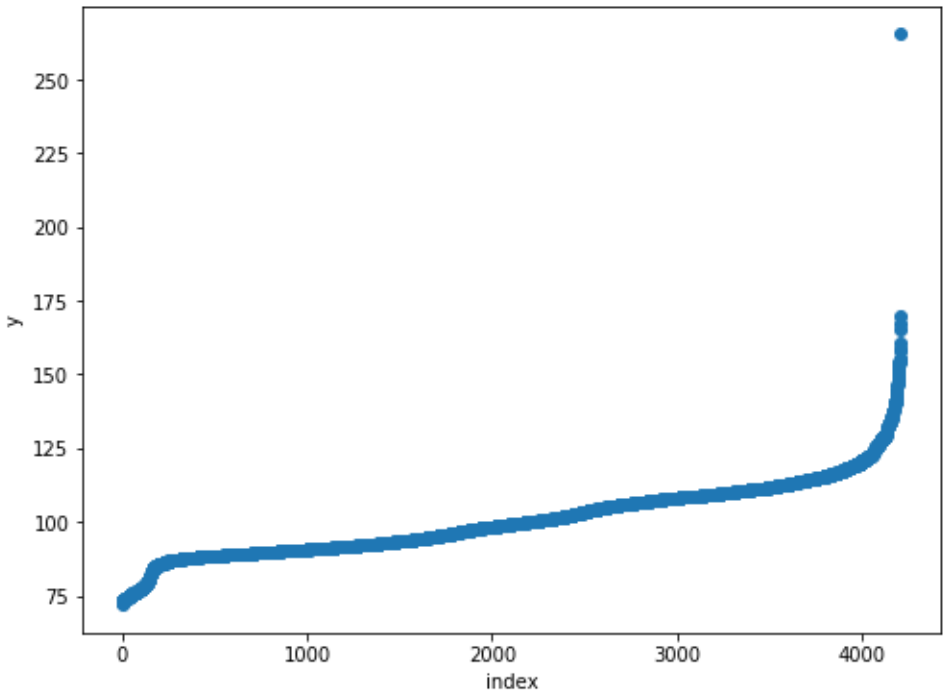
CODE

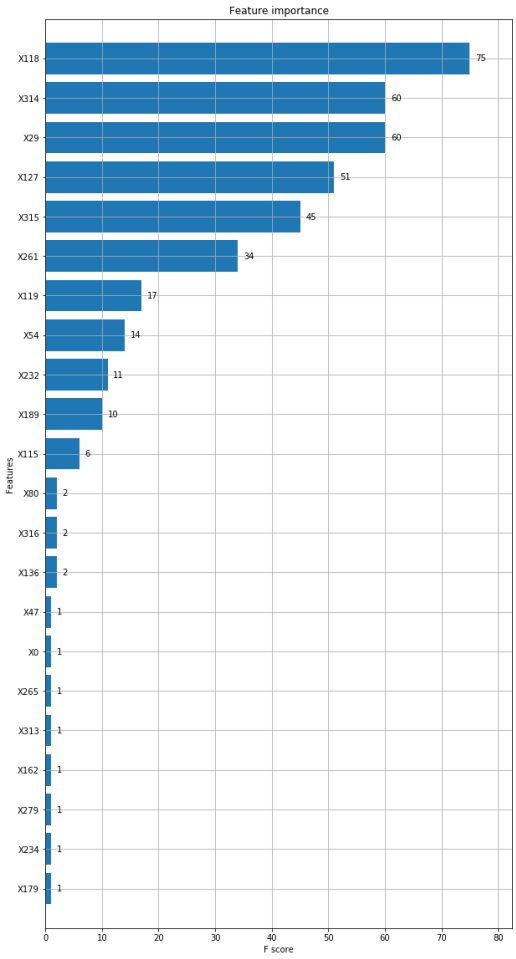
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Jan  4 13:37:09 2020

@author: joydeep
"""
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import numpy as np
import xgboost as xgb
from sklearn.metrics import r2_score

#load test and train data

train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')

# If for any column(s), the variance is equal to zero, then you need
to remove those variable(s).
threshold = 0

train_variance = train_df.var()
index_var = train_df.var() == threshold
index_var=index_var[index_var==True]
index_var_list = index_var.index.tolist()
new_train_df = train_df.drop(axis=1, labels=index_var_list)
new_test_df = test_df.drop(axis=1, labels=index_var_list)

plt.figure(figsize=(8,6))
plt.scatter(range(new_train_df.shape[0]),
np.sort(new_train_df.y.values))
plt.xlabel('index')
plt.ylabel('y')
plt.show()

# Check for null and unique values for test and train sets
missing_train_df = new_train_df.isnull().sum(axis=0).reset_index()
missing_train_df.columns = ['column_name', 'missing_count']
missing_train_df =
missing_train_df.loc[missing_train_df['missing_count']>0]
print("missing Count : ",missing_train_df)
```

```
# Apply label encoder
X = new_train_df.iloc[:,2:]
y = new_train_df.iloc[:,1]

le = LabelEncoder()
X['X0'] = le.fit_transform(X['X0'] )
X['X1'] = le.fit_transform(X['X1'] )
X['X2'] = le.fit_transform(X['X2'] )
X['X3'] = le.fit_transform(X['X3'] )
X['X4'] = le.fit_transform(X['X4'] )
X['X5'] = le.fit_transform(X['X5'] )
X['X6'] = le.fit_transform(X['X6'] )
X['X8'] = le.fit_transform(X['X8'] )

# Perform dimensionality reduction.
X_normalized = StandardScaler().fit_transform(X)
X_normalized=pd.DataFrame(X_normalized,columns=X.columns)

pca = PCA()
x_pca = pca.fit_transform(X_normalized)
x_pca = pd.DataFrame(x_pca)
#print("x_pca : %",x_pca.head())
pca_variance = pca.explained_variance_ratio_

# plot the important features
X_train, X_test, y_train, y_test = train_test_split(X_normalized, y,
test_size=0.2,random_state=1)

xgb_params = {
    'eta': 0.05,
    'max_depth': 6,
    'subsample': 0.7,
    'colsample_bytree': 0.7,
    'objective': 'reg:squarederror',
    'silent': 1,
    'learning_rate':0.01,
    'verbose':True
}
dtrain = xgb.DMatrix(X_train, y_train,
feature_names=X_train.columns.values)
model = xgb.train(dict(xgb_params, silent=0), dtrain,
num_boost_round=100, feval=r2_score, maximize=True)


# plot the important features #
fig, ax = plt.subplots(figsize=(10,20))
```

```
xgb.plot_importance(model, max_num_features=50, height=0.8, ax=ax)
plt.show()


# Predict your test_df values using xgboost
new_test_df = new_test_df.iloc[:,1:]
new_test_df['X0'] = le.fit_transform(new_test_df['X0'] )
new_test_df['X1'] = le.fit_transform(new_test_df['X1'] )
new_test_df['X2'] = le.fit_transform(new_test_df['X2'] )
new_test_df['X3'] = le.fit_transform(new_test_df['X3'] )
new_test_df['X4'] = le.fit_transform(new_test_df['X4'] )
new_test_df['X5'] = le.fit_transform(new_test_df['X5'] )
new_test_df['X6'] = le.fit_transform(new_test_df['X6'] )
new_test_df['X8'] = le.fit_transform(new_test_df['X8'] )

test_df_normalized = StandardScaler().fit_transform(new_test_df)
test_df_normalized=pd.DataFrame(test_df_normalized,columns=new_test_df
.columns)

d_test = xgb.DMatrix(X_test)
predict = model.predict(d_test)
print("Predicted : ", predict)
```

OUTPUT






```
missing Count :  Empty DataFrame
Columns: [column_name, missing_count]
Predicted
:  [49.40946  71.56618  71.56618  65.86166  49.40946  59.85395  59.853
95
59.8367   59.8367   59.89805  49.40946  59.85395  71.56618  59.85395
59.8367   59.96089  71.56618  65.86166  59.8367   66.64076  71.56618
59.96089  59.85395  59.85395  65.86166  71.56618  59.8367   71.86846
59.96089  59.8367   71.56618  71.56618  59.89805  71.56618  59.89805
59.8367   59.8367   49.40946  59.8367   71.56618  71.56618  71.56618
66.18464  59.8367   59.96089  65.86166  59.96089  65.86166  59.8367
59.8367   59.8367   71.56618  59.8367   59.85395  59.8367   65.86166
59.85395  59.8367   59.96089  59.85395  59.96089  59.8367   59.8367
71.56618  59.89805  59.96089  59.8367   71.56618  59.8367   59.8367
71.56618  59.8367   59.89805  59.8367   71.56618  59.96089  65.86166
65.86166  71.56618  59.8367   59.8367   59.8367   59.8367   59.89805
59.96089  59.96089  59.89805  71.56618  71.56618  71.56618  71.56618
71.56618  71.56618  59.85395  59.96089  71.56618  59.85395  59.8367
65.86166  59.8367   71.56618  65.86166  59.96089  71.76429  59.96089
```

59.85395	59.96089	59.8367	71.76429	59.8367	59.8367	71.56618
71.56618	71.56618	59.85395	59.8367	59.96089	59.96089	59.85395
49.40946	59.8367	59.96089	71.86846	59.85395	65.86166	71.56618
71.56618	59.8367	71.56618	59.8367	59.8367	59.89805	59.85395
71.56618	59.89805	71.56618	59.8367	59.85395	59.85395	66.18464
65.86166	65.86166	59.89805	66.18464	59.96089	59.8367	59.85395
66.18464	71.56618	71.56618	59.8367	60.07965	71.56618	71.56618
59.85395	59.89805	59.96089	71.56618	65.86166	65.86166	49.40946
59.8367	59.96089	65.86166	59.8367	59.89805	65.86166	71.56618
59.85395	71.56618	71.56618	59.89805	59.89805	59.8367	65.86166
59.96089	59.96089	59.89805	59.8367	65.86166	59.96089	49.40946
65.86166	59.85395	71.56618	71.56618	59.89805	59.89805	59.89805
59.89805	65.86166	59.96089	71.56618	71.56618	59.85395	59.96089
65.86166	49.40946	59.8367	59.8367	71.56618	71.56618	59.96089
65.86166	71.56618	71.56618	71.56618	59.85395	71.56618	59.89805
59.96089	71.56618	59.8367	71.56618	59.96089	69.92863	59.85395
71.76429	59.96089	59.8367	71.56618	71.76429	59.940872	59.8367
66.18464	59.8367	65.86166	71.56618	59.8367	71.56618	71.56618
59.8367	71.56618	71.56618	71.56618	49.502834	59.89805	71.56618
59.96089	65.86166	65.86166	65.86166	59.8367	65.86166	65.86166
71.56618	59.85395	59.8367	65.86166	71.56618	71.56618	71.56618
71.56618	59.8367	59.85395	59.85395	59.96089	49.40946	59.8367
59.89805	59.96089	71.56618	71.76429	71.56618	59.85395	59.96089
59.8367	59.8367	59.8367	59.96089	49.40946	59.8367	59.96089
59.85395	71.56618	59.96089	59.96089	71.56618	59.85395	59.89805
71.56618	65.86166	71.56618	71.56618	59.8367	59.8367	59.8367
71.56618	65.86166	59.89805	59.96089	59.85395	59.89805	59.96089
59.89805	71.56618	59.89805	69.92863	66.18464	66.64076	65.86166
49.40946	65.86166	59.89805	59.8367	59.8367	59.85395	59.8367
59.85395	59.8367	59.89805	59.85395	59.8367	59.8367	71.56618
65.86166	71.56618	65.86166	71.56618	59.8367	71.56618	71.56618
71.56618	65.86166	59.8367	71.56618	65.86166	59.8367	59.96089
65.86166	49.40946	71.56618	59.89805	59.85395	59.85395	59.8367
59.96089	59.89805	59.8367	59.8367	59.940872	71.56618	49.40946
71.56618	59.89805	59.85395	59.85395	71.56618	59.85395	71.56618
65.86166	59.96089	65.86166	59.8367	59.8367	59.96089	65.86166
59.85395	49.40946	59.96089	59.89805	59.89805	66.18464	59.85395
59.8367	59.8367	71.56618	59.89805	49.40946	59.89805	71.56618
65.86166	59.89805	59.89805	59.89805	71.56618	65.86166	59.96089
71.56618	59.89805	65.86166	59.85395	65.86166	71.56618	59.89805
65.86166	71.56618	71.56618	59.8367	65.86166	71.56618	59.8367
59.85395	59.8367	71.56618	59.8367	71.56618	71.86846	71.56618
59.8367	65.86166	71.76429	59.85395	65.86166	71.56618	71.56618
59.8367	65.86166	65.86166	59.8367	59.85395	71.56618	59.96089
59.96089	59.85395	59.85395	71.56618	59.96089	59.89805	71.56618
65.86166	59.89805	71.56618	59.85395	71.86846	59.89805	65.86166
65.86166	71.56618	71.479614	59.89805	59.89805	59.8367	71.56618
71.56618	71.56618	59.89805	71.56618	71.56618	59.85395	59.8367

71.56618 59.96089 71.56618 59.8367 59.8367 59.85395 59.8367
59.8367 65.86166 71.56618 59.8367 59.96089 71.56618 59.96089
71.56618 59.8367 72.08981 65.86166 59.85395 59.8367 65.86166
71.56618 59.96089 71.56618 59.85395 71.56618 71.56618 59.8367
65.86166 59.8367 59.8367 59.8367 49.40946 59.96089 71.56618
59.96089 59.89805 59.85395 66.18464 59.89805 49.40946 59.8367
59.89805 59.940872 59.8367 71.56618 59.8367 71.56618 72.08981
59.8367 71.56618 65.86166 59.89805 59.8367 59.89805 71.56618
71.56618 71.76429 59.8367 71.56618 65.86166 71.56618 59.8367
65.86166 71.56618 59.89805 59.96089 59.96089 71.76429 71.56618
59.8367 59.8367 71.56618 71.56618 59.8367 71.56618 49.40946
65.86166 71.56618 59.8367 71.56618 59.96089 59.89805 59.940872
59.8367 71.56618 71.56618 71.76429 65.86166 59.89805 59.8367
59.89805 65.86166 65.86166 59.96089 59.8367 71.56618 59.8367
65.86166 59.85395 71.56618 65.86166 65.86166 71.56618 59.85395
71.56618 49.40946 49.40946 71.76429 71.56618 71.76429 59.89805
65.86166 65.86166 59.96089 59.8367 65.86166 65.86166 71.56618
59.96089 71.56618 59.85395 59.8367 59.85395 59.8367 71.56618
59.96089 59.8367 71.56618 59.8367 59.85395 59.89805 49.40946
59.8367 71.56618 59.8367 71.56618 69.92863 71.56618 71.56618
59.96089 59.96089 59.8367 65.86166 59.8367 59.96089 59.89805
59.96089 59.85395 66.18464 59.8367 59.8367 65.86166 59.89805
71.56618 71.56618 71.76429 59.96089 59.85395 59.8367 71.56618
59.89805 59.85395 49.40946 59.96089 59.89805 71.56618 49.40946
59.96089 71.56618 59.85395 49.40946 59.8367 71.56618 59.96089
65.86166 71.56618 49.40946 49.40946 49.40946 65.86166 59.8367
71.56618 71.56618 65.86166 59.8367 65.86166 66.18464 59.89805
59.8367 65.86166 65.86166 59.8367 66.18464 71.56618 71.56618
59.89805 66.18464 71.56618 59.8367 59.89805 59.89805 59.8367
59.8367 59.96089 59.8367 59.96089 71.56618 59.89805 71.56618
59.85395 71.56618 71.56618 59.89805 59.8367 59.8367 71.56618
71.56618 71.56618 71.56618 65.86166 59.8367 65.86166 69.92863
71.56618 65.86166 71.56618 65.86166 59.8367 71.86846 59.96089
71.56618 71.56618 71.56618 59.89805 66.18464 49.40946 59.96089
59.89805 59.8367 59.8367 59.85395 65.86166 49.40946 71.56618
65.86166 71.56618 59.89805 59.85395 49.40946 49.40946 59.96089
59.96089 59.85395 65.86166 59.96089 69.92863 59.89805 59.85395
59.96089 59.85395 71.56618 71.56618 59.89805 59.89805 71.56618
65.86166 59.89805 66.64076 59.8367 49.40946 49.40946 49.40946
71.56618 71.76429 71.56618 59.96089 71.56618 65.86166 71.56618
59.85395 59.8367 71.56618 59.89805 49.40946 71.56618 59.8367
59.96089 71.56618 49.40946 71.56618 59.8367 59.96089 71.56618
71.76429 59.89805 65.86166 59.89805 71.56618 66.18464 71.56618
59.8367 59.8367 59.96089 71.56618 71.56618 66.18464 59.8367
71.76429 71.56618 71.56618 65.86166 59.85395 59.8367 49.502834
59.8367 71.56618 71.56618 71.56618 59.89805 71.56618 59.8367
71.56618 65.86166 71.56618 71.56618 71.56618 59.89805 59.8367
59.85395 59.8367 65.86166 71.56618 59.89805 59.85395 71.56618

```
59.85395  59.96089  65.86166  59.89805  49.40946  71.56618  71.56618
59.96089  65.86166  65.86166  59.96089  59.8367   71.56618  65.86166
59.85395  59.8367   59.8367   71.56618  71.56618  59.8367   59.85395
59.85395  59.85395  59.85395  59.8367   59.85395  59.85395  71.56618
59.85395  71.56618  65.86166  66.18464  71.56618  59.96089  59.85395
59.89805  65.86166  71.56618  71.56618  71.56618  59.8367   71.76429
59.89805  49.40946  59.85395  71.56618  71.56618  65.86166  59.8367
71.56618  65.86166  65.86166  59.8367   59.85395  65.86166  72.08981
59.85395  49.40946  71.56618  59.8367   65.86166  59.8367   71.56618
65.86166  59.89805 ]
```