**Project No. 1: Mercedes-Benz Greener**

**Manufacturing**

Step1: Importing library

import numpy as np

import pandas as pd

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

from sklearn import preprocessing

Step2: Read the data from train.csv

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

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

# Find Number of rows and columns

print(train\_df.shape)

print(train\_df.columns)

# Find Number of rows and columns

print(test\_df.shape)

print(test\_df.columns)

# Describe the dataset

train\_df.describe()

Step3: If any column(s) has the variance equal to zero, then you need to remove those variable(s)

train\_df.var()

(train\_df.var() == 0)

(train\_df.var() == 0).values

variance\_with\_zero = train\_df.var()[train\_df.var()==0].index.values

variance\_with\_zero

# Drop zero variance variables

train\_df = train\_df.drop(variance\_with\_zero, axis=1)

print(train\_df.shape)

train\_df = train\_df.drop(['ID'], axis=1)

train\_df.head()

Step4: Check for null and unique values for test and train sets.  
train\_df.isnull().sum().values  
train\_df.isnull().any()  
test\_df.isnull().sum().values

# Find unique records  
train\_df.nunique()

Step5: Filter out the columns having object datatype

object\_datatypes = train\_df.select\_dtypes(include=[object])

object\_datatypes

object\_datatype\_columns = object\_datatypes.columns

object\_datatype\_columns

Step6: Apply label encoder.

label\_encoder = preprocessing.LabelEncoder()

train\_df['X0'].unique()

train\_df['X0'] = label\_encoder.fit\_transform(train\_df['X0'])  
train\_df['X0'].unique()

# Apply same for all columns having object type data

train\_df['X1'] = label\_encoder.fit\_transform(train\_df['X1'])

train\_df['X2'] = label\_encoder.fit\_transform(train\_df['X2'])

train\_df['X3'] = label\_encoder.fit\_transform(train\_df['X3'])

train\_df['X4'] = label\_encoder.fit\_transform(train\_df['X4'])

train\_df['X5'] = label\_encoder.fit\_transform(train\_df['X5'])

train\_df['X6'] = label\_encoder.fit\_transform(train\_df['X6'])

train\_df['X8'] = label\_encoder.fit\_transform(train\_df['X8'])

train\_df.head()

Step7: Perform dimensionality reduction (PCA)

from sklearn.decomposition import PCA

sklearn\_pca = PCA(n\_components=0.95)

sklearn\_pca.fit(train\_df)

x\_train\_transformed = sklearn\_pca.transform(train\_df)

print(x\_train\_transformed.shape)

sklearn\_pca\_98 = PCA(n\_components=0.98)

sklearn\_pca\_98.fit(train\_df)

x\_train\_transformed\_98 = sklearn\_pca\_98.transform(train\_df)

print(x\_train\_transformed\_98.shape)

train\_df.y

Step8: Train and Test split on Train dataset

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

y = train\_df.y

xtrain,xtest,ytrain,ytest = train\_test\_split(X,y,test\_size=0.3,random\_state=42)

print(xtrain)

print(xtrain.shape)

print(ytrain)

print(ytrain.shape)

print(xtest)

print(xtest.shape)

pca\_xtrain = PCA(n\_components=0.95)

pca\_xtrain.fit(xtrain)

pca\_xtrain\_transformed = pca\_xtrain.transform(xtrain)

print(pca\_xtrain\_transformed.shape)

pca\_xtest = PCA(n\_components=0.95)

pca\_xtest.fit(xtest)

pca\_xtest\_transformed = pca\_xtest.transform(xtest)

print(pca\_xtest\_transformed.shape)

print(pca\_xtest.explained\_variance\_)

print(pca\_xtest.explained\_variance\_ratio\_)

Step9: PCA for test\_df dataset

test\_df

test\_object\_datatypes = test\_df.select\_dtypes(include=[object])

test\_object\_datatypes

test\_df['X0'] = label\_encoder.fit\_transform(test\_df['X0'])

test\_df['X1'] = label\_encoder.fit\_transform(test\_df['X1'])

test\_df['X2'] = label\_encoder.fit\_transform(test\_df['X2'])

test\_df['X3'] = label\_encoder.fit\_transform(test\_df['X3'])

test\_df['X4'] = label\_encoder.fit\_transform(test\_df['X4'])

test\_df['X5'] = label\_encoder.fit\_transform(test\_df['X5'])

test\_df['X6'] = label\_encoder.fit\_transform(test\_df['X6'])

test\_df['X8'] = label\_encoder.fit\_transform(test\_df['X8'])

print(test\_df)

print(test\_df.shape)

test\_df = test\_df.drop('ID',axis=1)

pca\_test\_df = PCA(n\_components=0.95)

pca\_test\_df.fit(test\_df)

pca\_test\_df\_transformed = pca\_test\_df.transform(test\_df)

print(pca\_test\_df\_transformed.shape)

print(pca\_test\_df.explained\_variance\_)

print(pca\_test\_df.explained\_variance\_ratio\_)

Step10: Training using xgboost

test\_df = pd.read\_csv('test.csv')

usable\_columns = list(set(train\_df.columns) - set(['ID', 'y']))

ytrain = train\_df['y'].values

id\_test = test\_df['ID'].values

xtrain = train\_df[usable\_columns]

xtest = train\_df[usable\_columns]

for column in usable\_columns:

    cardinality = len(np.unique(xtrain[column]))

    if cardinality == 1:

# Column with only one value is useless so we drop it

        xtrain.drop(column, axis=1)

        xtest.drop(column, axis=1)

    if cardinality > 2:

# Column is categorical

        mapper = lambda x: sum([ord(digit) for digit in x])

        xtrain[column] = xtrain[column].apply(mapper)

        xtest[column] = xtest[column].apply(mapper)

xtrain.head()

Step11: Perform xgboost

import xgboost as xgb

from sklearn.metrics import r2\_score

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

xtrain, xvalid, ytrain, yvalid = train\_test\_split(xtrain, ytrain, test\_size=0.2, random\_state=4242)

d\_train = xgb.DMatrix(xtrain, label=ytrain)

d\_valid = xgb.DMatrix(xvalid, label=yvalid)

d\_test = xgb.DMatrix(xtest)

params = {}

params['objective'] = 'reg:linear'

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, maximize=True, verbose\_eval=10)

Step12: Predict test\_df values using xgboost.

p\_test = clf.predict(d\_test)

sub = pd.DataFrame()

sub['ID'] = ID

sub['y'] = p\_test

sub.to\_csv('xgb.csv', index=False)

sub.head()

# Screenshot of the Output

## Q1. Read the data from train.csv

Step2: Read the data from train.csv

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

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

# Find Number of rows and columns

print(train\_df.shape)

print(train\_df.columns)

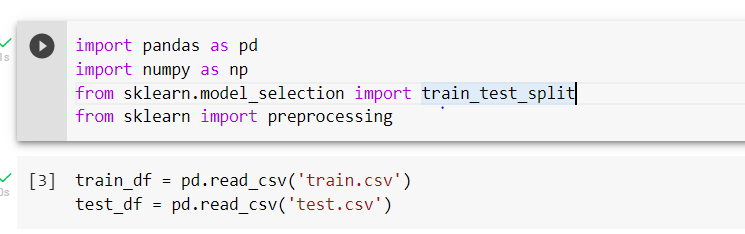
# Find Number of rows and columns

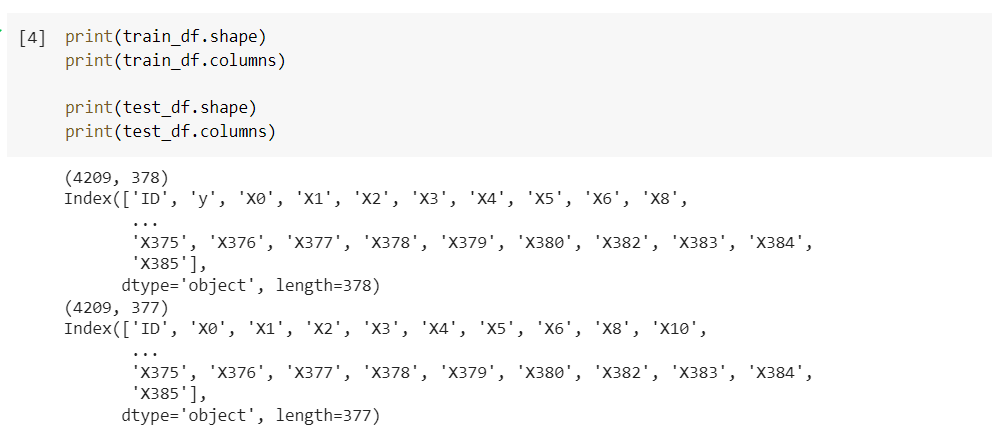
print(test\_df.shape)

print(test\_df.columns)

# Describe the dataset

train\_df.describe()





## Q2. If any column has the variance equal to zero, then you need to remove those variable and Check for null and unique values for test and train sets.

Step4: Check for null and unique values for test and train sets.  
train\_df.isnull().sum().values  
train\_df.isnull().any()  
test\_df.isnull().sum().values

# Find unique records  
train\_df.nunique()

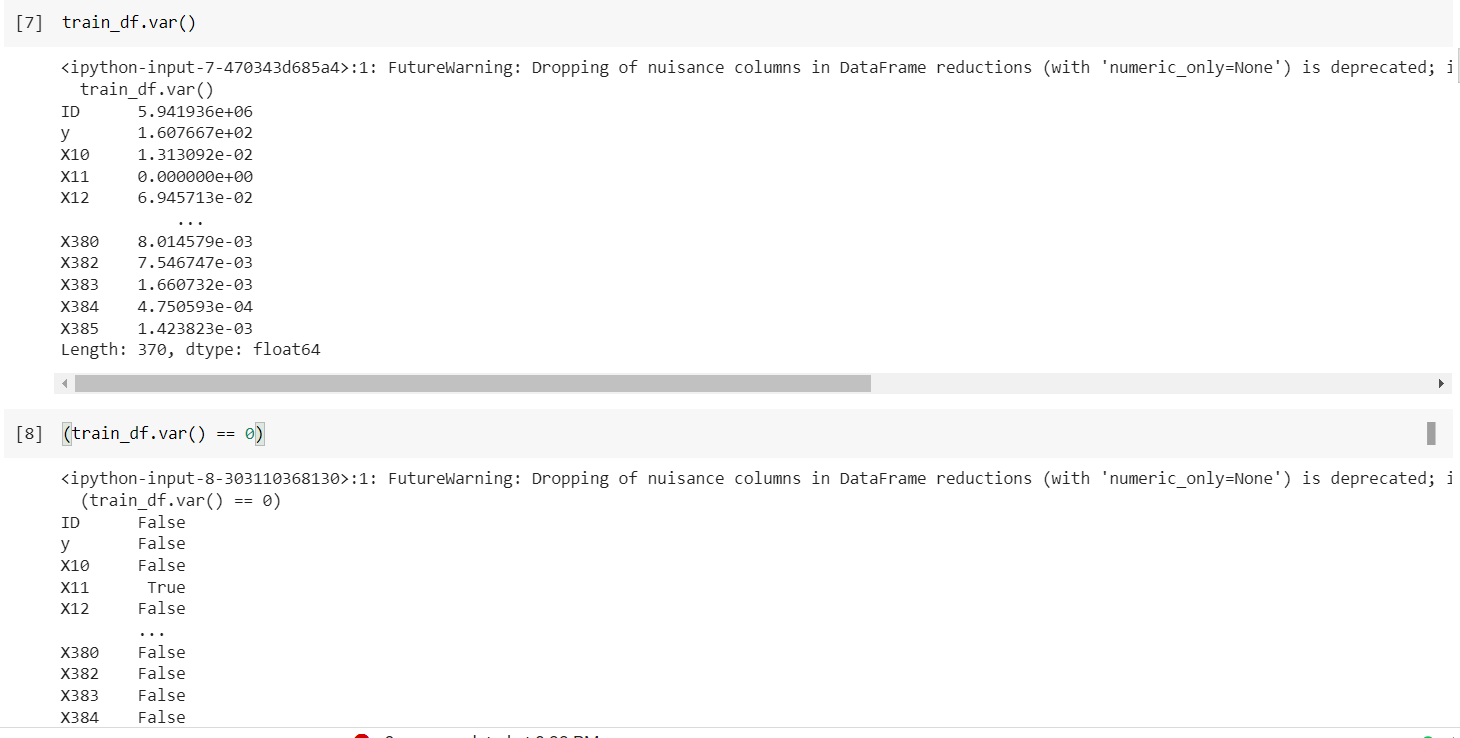
Step5: Filter out the columns having object datatype

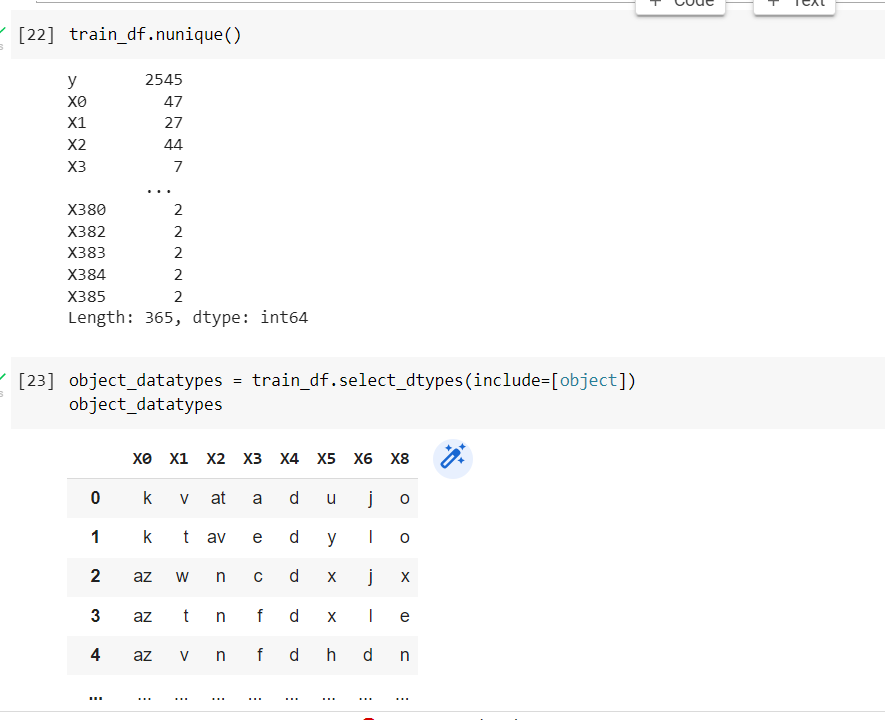
object\_datatypes = train\_df.select\_dtypes(include=[object])

object\_datatypes

object\_datatype\_columns = object\_datatypes.columns

object\_datatype\_columns





**Q3. Apply label encoder**

Step6: Apply label encoder.

label\_encoder = preprocessing.LabelEncoder()

train\_df['X0'].unique()

train\_df['X0'] = label\_encoder.fit\_transform(train\_df['X0'])  
train\_df['X0'].unique()

# Apply same for all columns having object type data

train\_df['X1'] = label\_encoder.fit\_transform(train\_df['X1'])

train\_df['X2'] = label\_encoder.fit\_transform(train\_df['X2'])

train\_df['X3'] = label\_encoder.fit\_transform(train\_df['X3'])

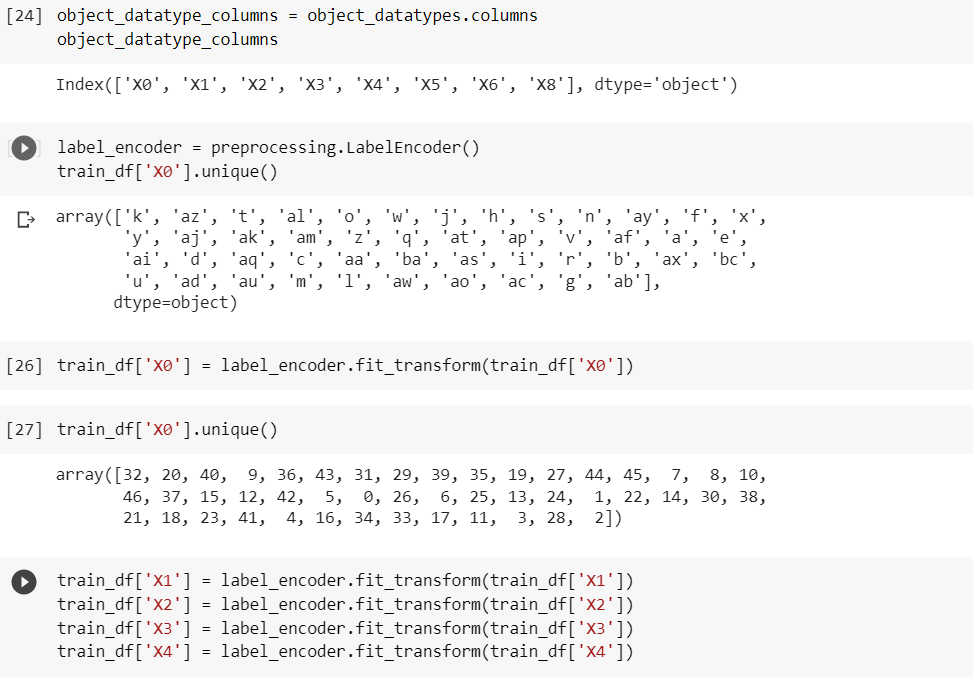
train\_df['X4'] = label\_encoder.fit\_transform(train\_df['X4'])

train\_df['X5'] = label\_encoder.fit\_transform(train\_df['X5'])

train\_df['X6'] = label\_encoder.fit\_transform(train\_df['X6'])

train\_df['X8'] = label\_encoder.fit\_transform(train\_df['X8'])

train\_df.head()



## Q4. Perform dimensionality reduction

Step7: Perform dimensionality reduction (PCA)

from sklearn.decomposition import PCA

sklearn\_pca = PCA(n\_components=0.95)

sklearn\_pca.fit(train\_df)

x\_train\_transformed = sklearn\_pca.transform(train\_df)

print(x\_train\_transformed.shape)

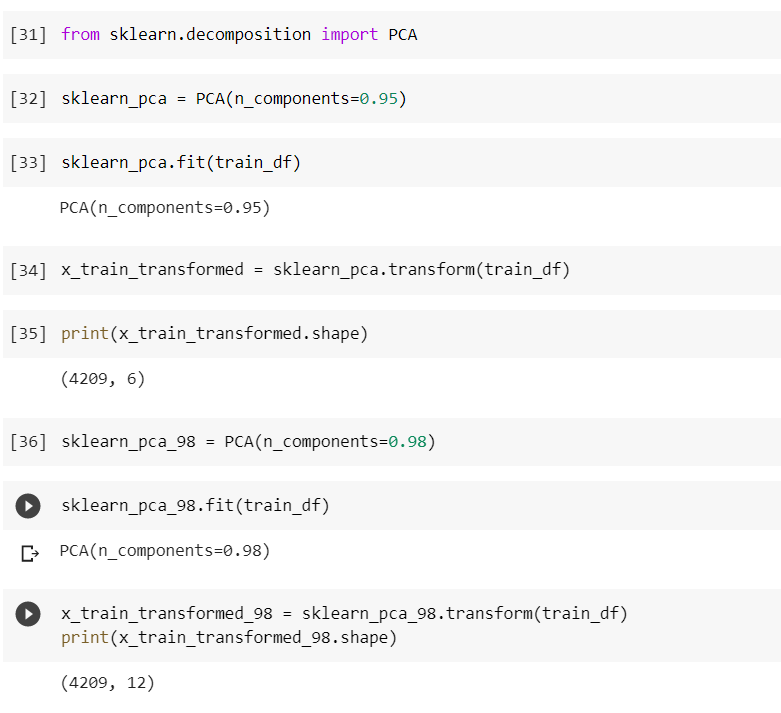
sklearn\_pca\_98 = PCA(n\_components=0.98)

sklearn\_pca\_98.fit(train\_df)

x\_train\_transformed\_98 = sklearn\_pca\_98.transform(train\_df)

print(x\_train\_transformed\_98.shape)

train\_df.y



### **Q5. PCA for test\_df dataset**

Step8: Train and Test split on Train dataset

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

y = train\_df.y

xtrain,xtest,ytrain,ytest = train\_test\_split(X,y,test\_size=0.3,random\_state=42)

print(xtrain)

print(xtrain.shape)

print(ytrain)

print(ytrain.shape)

print(xtest)

print(xtest.shape)

pca\_xtrain = PCA(n\_components=0.95)

pca\_xtrain.fit(xtrain)

pca\_xtrain\_transformed = pca\_xtrain.transform(xtrain)

print(pca\_xtrain\_transformed.shape)

pca\_xtest = PCA(n\_components=0.95)

pca\_xtest.fit(xtest)

pca\_xtest\_transformed = pca\_xtest.transform(xtest)

print(pca\_xtest\_transformed.shape)

print(pca\_xtest.explained\_variance\_)

print(pca\_xtest.explained\_variance\_ratio\_)

Step9: PCA for test\_df dataset

test\_df

test\_object\_datatypes = test\_df.select\_dtypes(include=[object])

test\_object\_datatypes

test\_df['X0'] = label\_encoder.fit\_transform(test\_df['X0'])

test\_df['X1'] = label\_encoder.fit\_transform(test\_df['X1'])

test\_df['X2'] = label\_encoder.fit\_transform(test\_df['X2'])

test\_df['X3'] = label\_encoder.fit\_transform(test\_df['X3'])

test\_df['X4'] = label\_encoder.fit\_transform(test\_df['X4'])

test\_df['X5'] = label\_encoder.fit\_transform(test\_df['X5'])

test\_df['X6'] = label\_encoder.fit\_transform(test\_df['X6'])

test\_df['X8'] = label\_encoder.fit\_transform(test\_df['X8'])

print(test\_df)

print(test\_df.shape)

test\_df = test\_df.drop('ID',axis=1)

pca\_test\_df = PCA(n\_components=0.95)

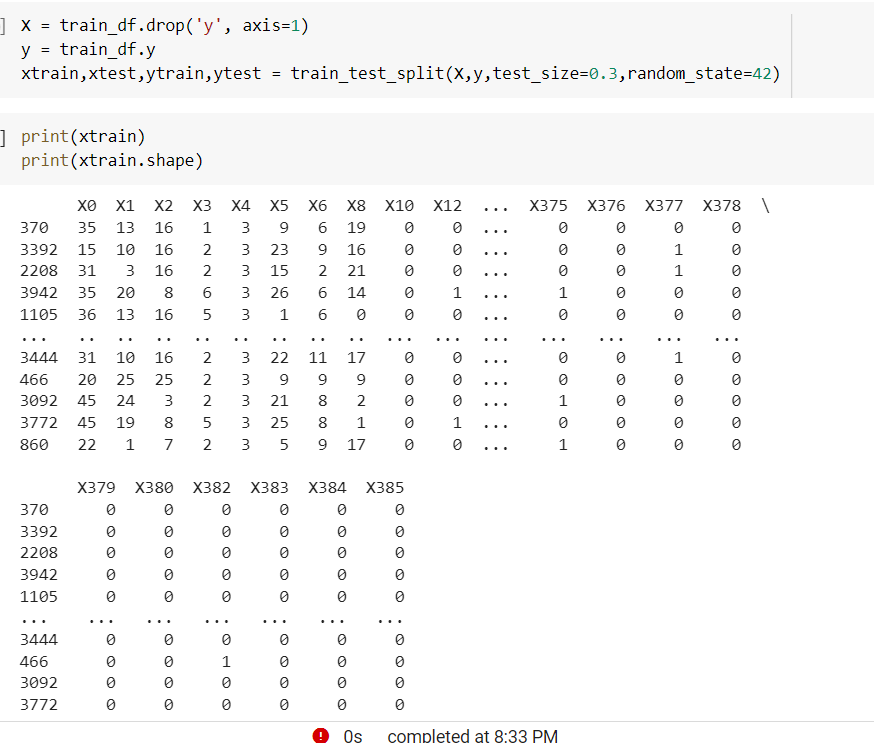
pca\_test\_df.fit(test\_df)

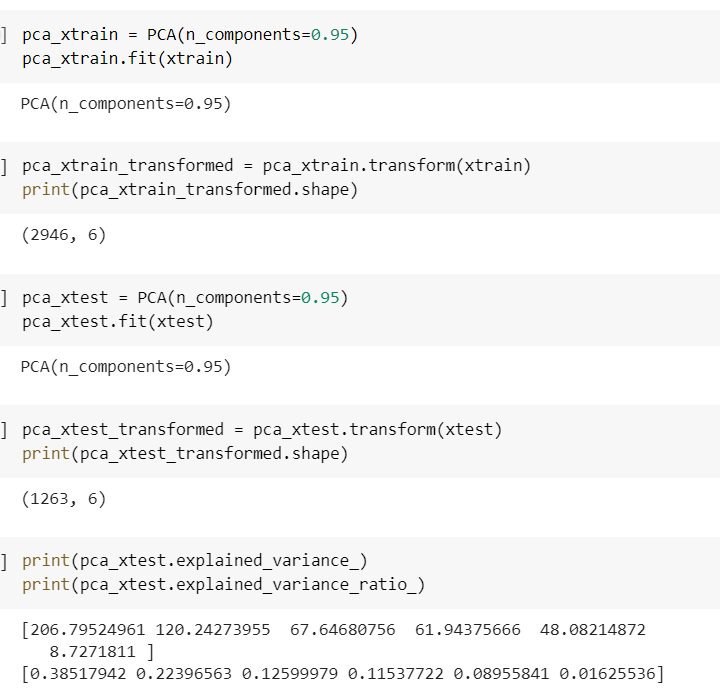
pca\_test\_df\_transformed = pca\_test\_df.transform(test\_df)

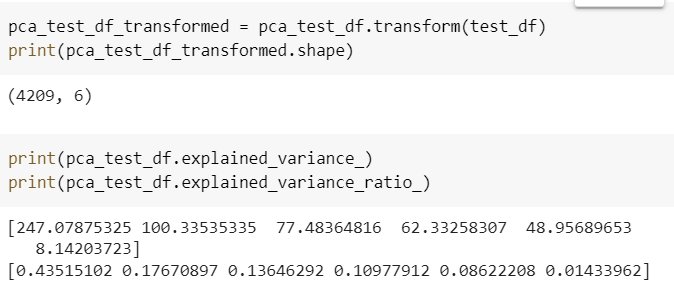
print(pca\_test\_df\_transformed.shape)

print(pca\_test\_df.explained\_variance\_)

print(pca\_test\_df.explained\_variance\_ratio\_)







## Q6. Training using xgboost

Step10: Training using xgboost

test\_df = pd.read\_csv('test.csv')

usable\_columns = list(set(train\_df.columns) - set(['ID', 'y']))

ytrain = train\_df['y'].values

id\_test = test\_df['ID'].values

xtrain = train\_df[usable\_columns]

xtest = train\_df[usable\_columns]

for column in usable\_columns:

    cardinality = len(np.unique(xtrain[column]))

    if cardinality == 1:

# Column with only one value is useless so we drop it

        xtrain.drop(column, axis=1)

        xtest.drop(column, axis=1)

    if cardinality > 2:

# Column is categorical

        mapper = lambda x: sum([ord(digit) for digit in x])

        xtrain[column] = xtrain[column].apply(mapper)

        xtest[column] = xtest[column].apply(mapper)

xtrain.head()

Step11: Perform xgboost

import xgboost as xgb

from sklearn.metrics import r2\_score

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

xtrain, xvalid, ytrain, yvalid = train\_test\_split(xtrain, ytrain, test\_size=0.2, random\_state=4242)

d\_train = xgb.DMatrix(xtrain, label=ytrain)

d\_valid = xgb.DMatrix(xvalid, label=yvalid)

d\_test = xgb.DMatrix(xtest)

params = {}

params['objective'] = 'reg:linear'

params['eta'] = 0.02

params['max\_depth'] = 4

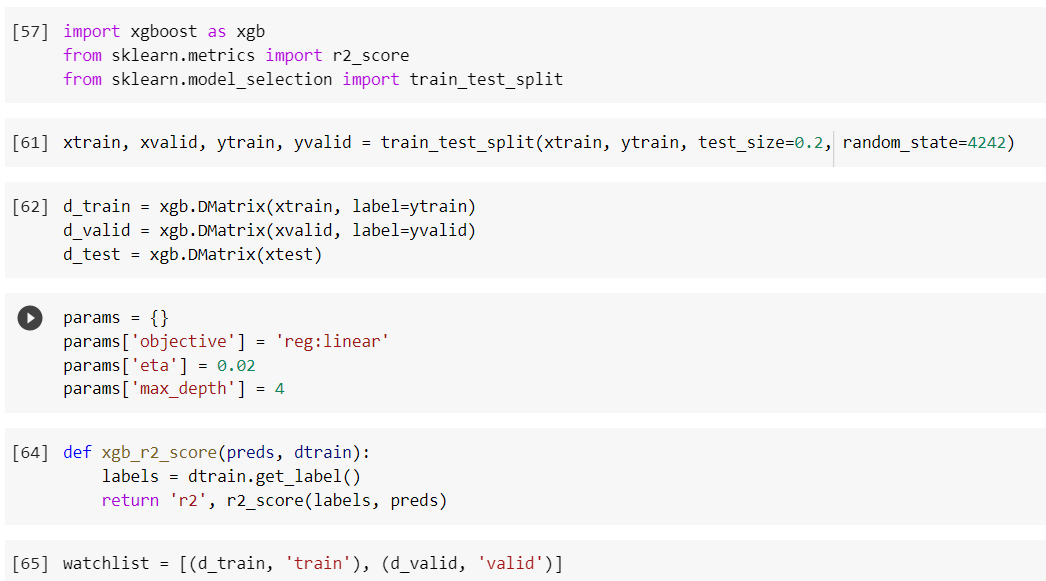
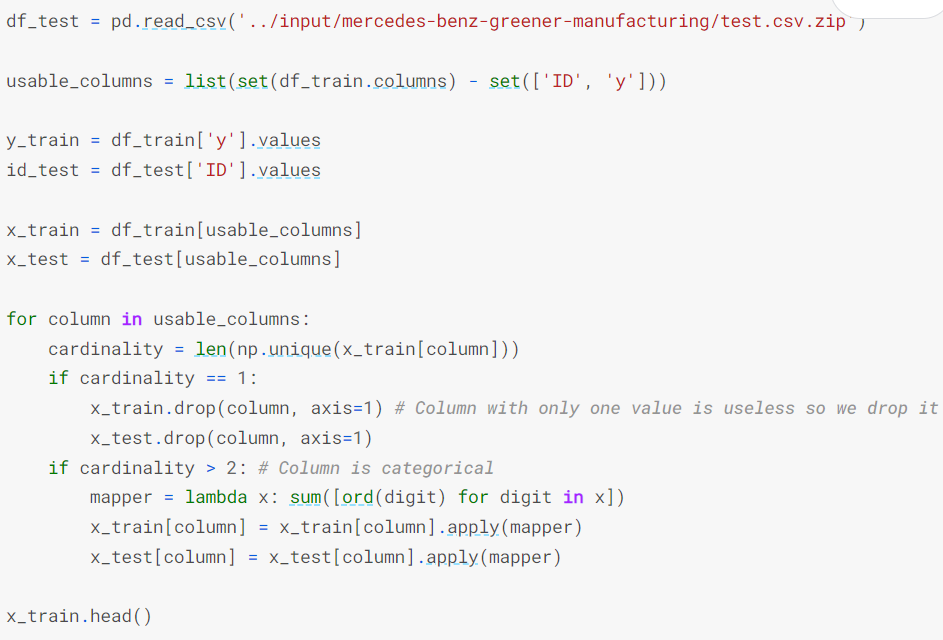
def xgb\_r2\_score(preds, dtrain):

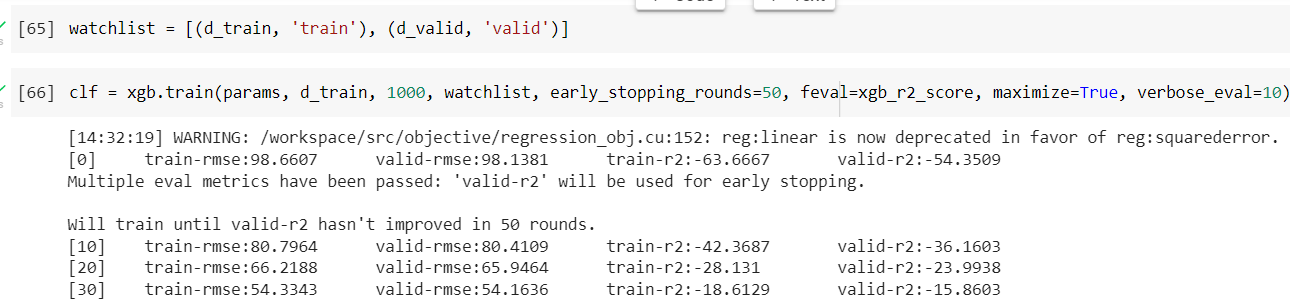
    labels = dtrain.get\_label()

    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, maximize=True, verbose\_eval=10)





## Q7. Predict your test\_df values using xgboost

Step12: Predict test\_df values using xgboost.

p\_test = clf.predict(d\_test)

sub = pd.DataFrame()

sub['ID'] = ID

sub['y'] = p\_test

sub.to\_csv('xgb.csv', index=False)

sub.head()

