Mercedes-Benz Greener Manufacturing

January 28, 2021

0.1 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 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.

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.
- Find the datasets here.

0.2 Importing packages

[68]:
```
import pandas as pd
import numpy as np

from sklearn.decomposition import PCA
from sklearn import preprocessing

import matplotlib.pyplot as plt
%matplotlib inline
```

0.3 Loading train and test

[69]:
```
train = pd.read_csv("train.csv")
test =pd.read_csv("test.csv")
```

---

[70]:
```
display(train.head())
display(train.sample(3))
display(test.head())
```

```
   ID       y  X0 X1  X2 X3 X4 X5 X6 X8  ...  X375  X376  X377  X378  X379  \
0   0  130.81   k  v  at  a  d  u  j  o  ...     0     0     1     0     0
1   6   88.53   k  t  av  e  d  y  l  o  ...     1     0     0     0     0
2   7   76.26  az  w   n  c  d  x  j  x  ...     0     0     0     0     0
3   9   80.62  az  t   n  f  d  x  l  e  ...     0     0     0     0     0
4  13   78.02  az  v   n  f  d  h  d  n  ...     0     0     0     0     0

   X380  X382  X383  X384  X385
0     0     0     0     0     0
1     0     0     0     0     0
2     0     1     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0

[5 rows x 378 columns]
```

```
        ID       y  X0 X1  X2 X3 X4 X5 X6 X8  ...  X375  X376  X377  X378  \
3400  6785  108.83  ak  v   r  c  d  r  i  i  ...     0     0     1     0
497    958  110.20   a  v   k  d  d  d  e  e  ...     0     1     0     0
2936  5889   98.94  aj  s  as  c  d  p  l  a  ...     1     0     0     0

      X379  X380  X382  X383  X384  X385
3400     0     0     0     0     0     0
497      0     0     0     0     0     0
2936     0     0     0     0     0     0
```

```
[3 rows x 378 columns]

   ID  X0 X1  X2 X3 X4 X5 X6 X8  X10  ...  X375  X376  X377  X378  X379  X380  \
0   1  az  v   n  f  d  t  a  w    0  ...     0     0     0     1     0     0
1   2   t  b  ai  a  d  b  g  y    0  ...     0     0     1     0     0     0
2   3  az  v  as  f  d  a  j  j    0  ...     0     0     0     1     0     0
3   4  az  l   n  f  d  z  l  n    0  ...     0     0     0     1     0     0
4   5   w  s  as  c  d  y  i  m    0  ...     1     0     0     0     0     0

   X382  X383  X384  X385
0     0     0     0     0
1     0     0     0     0
2     0     0     0     0
3     0     0     0     0
4     0     0     0     0

[5 rows x 377 columns]
```

[71]:

```
train.shape,test.shape
```

[71]: ((4209, 378), (4209, 377))

[72]:

```
display(train.describe())
display(test.describe())
```

```
                ID            y          X10     X11          X12  \
count  4209.000000  4209.000000  4209.000000  4209.0  4209.000000
mean   4205.960798   100.669318     0.013305     0.0     0.075077
std    2437.608688    12.679381     0.114590     0.0     0.263547
min       0.000000    72.110000     0.000000     0.0     0.000000
25%    2095.000000    90.820000     0.000000     0.0     0.000000
50%    4220.000000    99.150000     0.000000     0.0     0.000000
75%    6314.000000   109.010000     0.000000     0.0     0.000000
max    8417.000000   265.320000     1.000000     0.0     1.000000

               X13          X14          X15          X16          X17  ...  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000  ...
mean      0.057971     0.428130     0.000475     0.002613     0.007603  ...
std       0.233716     0.494867     0.021796     0.051061     0.086872  ...
min       0.000000     0.000000     0.000000     0.000000     0.000000  ...
25%       0.000000     0.000000     0.000000     0.000000     0.000000  ...
50%       0.000000     0.000000     0.000000     0.000000     0.000000  ...
75%       0.000000     1.000000     0.000000     0.000000     0.000000  ...
max       1.000000     1.000000     1.000000     1.000000     1.000000  ...

              X375         X376         X377         X378         X379  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
```

```
mean      0.318841     0.057258     0.314802     0.020670     0.009503
std       0.466082     0.232363     0.464492     0.142294     0.097033
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       1.000000     0.000000     1.000000     0.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

               X380         X382         X383         X384         X385
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      0.008078     0.007603     0.001663     0.000475     0.001426
std       0.089524     0.086872     0.040752     0.021796     0.037734
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       0.000000     0.000000     0.000000     0.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

[8 rows x 370 columns]
                 ID          X10          X11          X12          X13  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean   4211.039202     0.019007     0.000238     0.074364     0.061060
std    2423.078926     0.136565     0.015414     0.262394     0.239468
min       1.000000     0.000000     0.000000     0.000000     0.000000
25%    2115.000000     0.000000     0.000000     0.000000     0.000000
50%    4202.000000     0.000000     0.000000     0.000000     0.000000
75%    6310.000000     0.000000     0.000000     0.000000     0.000000
max    8416.000000     1.000000     1.000000     1.000000     1.000000

                X14          X15          X16          X17          X18  ...  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000  ...
mean      0.427893     0.000713     0.002613     0.008791     0.010216  ...
std       0.494832     0.026691     0.051061     0.093357     0.100570  ...
min       0.000000     0.000000     0.000000     0.000000     0.000000  ...
25%       0.000000     0.000000     0.000000     0.000000     0.000000  ...
50%       0.000000     0.000000     0.000000     0.000000     0.000000  ...
75%       1.000000     0.000000     0.000000     0.000000     0.000000  ...
max       1.000000     1.000000     1.000000     1.000000     1.000000  ...

               X375         X376         X377         X378         X379  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      0.325968     0.049656     0.311951     0.019244     0.011879
std       0.468791     0.217258     0.463345     0.137399     0.108356
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       1.000000     0.000000     1.000000     0.000000     0.000000
```

```
max       1.000000     1.000000     1.000000     1.000000     1.000000

               X380         X382         X383         X384         X385
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      0.008078     0.008791     0.000475     0.000713     0.001663
std       0.089524     0.093357     0.021796     0.026691     0.040752
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       0.000000     0.000000     0.000000     0.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

[8 rows x 369 columns]
```

[73]:
```
train.isnull().any()
```

[73]:
```
ID       False
y        False
X0       False
X1       False
X2       False
         ...
X380     False
X382     False
X383     False
X384     False
X385     False
Length: 378, dtype: bool
```

[74]:
```
test.isnull().any()
```

[74]:
```
ID       False
X0       False
X1       False
X2       False
X3       False
         ...
X380     False
X382     False
X383     False
X384     False
X385     False
Length: 377, dtype: bool
```

[75]:
```
display(test.dtypes.head(15))
display(test.dtypes.tail(15))
```

```
ID      int64
```

```
X0      object
X1      object
X2      object
X3      object
X4      object
X5      object
X6      object
X8      object
X10      int64
X11      int64
X12      int64
X13      int64
X14      int64
X15      int64
dtype: object
```

```
X370     int64
X371     int64
X372     int64
X373     int64
X374     int64
X375     int64
X376     int64
X377     int64
X378     int64
X379     int64
X380     int64
X382     int64
X383     int64
X384     int64
X385     int64
dtype: object
```

0.4 exploring Train data

[76]:
```
y_train = train["y"].values
y_train
```

[76]:
```
array([130.81,  88.53,  76.26, ..., 109.22,  87.48, 110.85])
```

[77]:
```
features = [col for col in train.columns if 'X' in col]
len(features)
```

[77]:
```
376
```

[78]:
```
train[features].dtypes.value_counts()
```

[78]:
```
int64     368
object      8
dtype: int64
```

0.5 Exploring test data

[79]:
```
test_features = [col for col in test.columns if 'X' in col]
len(test_features)
```

[79]:
```
376
```

[80]:
```
test[test_features].dtypes.value_counts()
```

[80]:
```
int64     368
object      8
dtype: int64
```

[ ]:
```

```

[81]:
```
#removing unusable columns from train and test

usable_columns = list(set(train.columns)-set(["ID","y"]))

y_train = train["y"].values
id_test = test["ID"].values
print("y_train:",y_train)
print("id_test:",id_test)
print("y_train_shape:",y_train.shape)
print("id_test_shape:",id_test.shape)
```

```
y_train: [130.81  88.53  76.26 ... 109.22  87.48 110.85]
id_test: [   1    2    3 ... 8413 8414 8416]
y_train_shape: (4209,)
id_test_shape: (4209,)
```

[82]:
```
X_train = train[usable_columns]
X_test = test[usable_columns]
print("X_train:",X_train.columns)
print("X_test:",X_test.columns)
print("X_train shape:",X_train.shape)
print("X_test shape:",X_test.shape)
```

```
X_train: Index(['X184', 'X291', 'X131', 'X243', 'X126', 'X251', 'X105', 'X316',
'X247',
       'X30',
       ...
```

```
       'X195', 'X364', 'X15', 'X123', 'X130', 'X230', 'X124', 'X59', 'X206',
       'X106'],
      dtype='object', length=376)
X_test: Index(['X184', 'X291', 'X131', 'X243', 'X126', 'X251', 'X105', 'X316',
'X247',
       'X30',
       ...
       'X195', 'X364', 'X15', 'X123', 'X130', 'X230', 'X124', 'X59', 'X206',
       'X106'],
      dtype='object', length=376)
X_train shape: (4209, 376)
X_test shape: (4209, 376)
```

[83]:
```
if X_train.isnull().any() is True:
    print("Missing values in X_Train")
else:
    print("no Missing values in X_train")
```

```
no Missing values in X_train
```

[84]:
```
if X_test.isnull().any() is True:
    print("Missing values in X_Train")
else:
    print("no Missing values in X_test")
```

```
no Missing values in X_test
```

0.6 Dropping the Zero variance columns from train and test data

[85]:
```
for column in usable_columns:
    cardinality = len(np.unique(X_train[column]))
    if cardinality ==1:
        X_train.drop(column,axis=1)
        X_test.drop(column,axis=1)
X_train.head()
```

[85]:
```
   X184  X291  X131  X243  X126  X251  X105  X316  X247  X30  ...  X195  X364  \
0     1     0     1     0     0     0     0     1     0    0  ...     0     0
1     0     0     0     0     0     0     0     1     0    0  ...     0     0
2     0     0     0     0     0     0     0     0     0    0  ...     0     0
3     0     1     0     0     0     0     0     0     0    0  ...     0     0
4     0     0     0     0     0     0     0     0     0    0  ...     0     0

   X15  X123  X130  X230  X124  X59  X206  X106
0    0     0     0     0     0    0     0     0
1    0     0     0     0     0    0     0     0
```

```
2      0       0       0       0       0     0       1       0
3      0       0       0       0       0     0       0       0
4      0       0       0       0       0     0       0       0

[5 rows x 376 columns]
```

0.7 Label encoding to categorical variables in test an dtrain data

[86]:
```python
for f in ["X0","X1","X2","X3","X4","X5","X6","X8"]:
    lbl = preprocessing.LabelEncoder()
    lbl.fit(list(X_train[f].values))
    X_train[f] = lbl.transform(list(X_train[f].values))
```

```
<ipython-input-86-2a0d19a9992f>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train[f] = lbl.transform(list(X_train[f].values))
```

[87]:
```python
X_train[["X0","X1","X2","X3","X4","X5","X6","X8"]].dtypes
```

[87]:
```
X0    int64
X1    int64
X2    int64
X3    int64
X4    int64
X5    int64
X6    int64
X8    int64
dtype: object
```

[88]:
```python
for g in ["X0","X1","X2","X3","X4","X5","X6","X8"]:
    lbl1 = preprocessing.LabelEncoder()
    lbl1.fit(list(X_test[g].values))
    X_test[g] = lbl1.transform(list(X_test[g].values))
```

```
<ipython-input-88-07fe900192b5>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test[g] = lbl1.transform(list(X_test[g].values))
```

[89]:
```
X_test[["X0","X1","X2","X3","X4","X5","X6","X8"]].dtypes
```

[89]:
```
X0    int64
X1    int64
X2    int64
X3    int64
X4    int64
X5    int64
X6    int64
X8    int64
dtype: object
```

[90]:
```
X_train[features].dtypes.value_counts()
```

[90]:
```
int64    376
dtype: int64
```

[91]:
```
X_test[features].dtypes.value_counts()
```

[91]:
```
int64    376
dtype: int64
```

[92]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[93]:
```
pca1 = PCA()
pca1_final_train = pca1.fit_transform(X_train)
pca1_final_test = pca1.fit_transform(X_test)
```

[94]:
```
print(pca1.explained_variance_ratio_)
```

```
[6.63394212e-02 5.03269345e-02 4.57153414e-02 3.75767070e-02
 3.18215977e-02 2.89246157e-02 2.51539240e-02 2.06618035e-02
 1.84389008e-02 1.79941209e-02 1.68205036e-02 1.61593234e-02
 1.58080558e-02 1.53189418e-02 1.51697469e-02 1.48712835e-02
 1.46491490e-02 1.36706593e-02 1.31243083e-02 1.28159562e-02
 1.25472520e-02 1.22316005e-02 1.14636195e-02 1.06621052e-02
 9.67132370e-03 9.09804809e-03 9.05450395e-03 8.58955625e-03
 8.49807882e-03 8.31369273e-03 8.10555591e-03 7.92646977e-03
 7.30411059e-03 7.17049852e-03 6.99520860e-03 6.87960083e-03
 6.70708936e-03 6.35045290e-03 6.21684912e-03 6.06173183e-03
 5.87784265e-03 5.75928614e-03 5.67072824e-03 5.37815991e-03
 5.18551283e-03 5.09914357e-03 5.01372028e-03 4.84597228e-03
```

4.67574449e-03	4.58177883e-03	4.53613387e-03	4.36279045e-03
4.33359558e-03	4.26094041e-03	4.23135429e-03	4.17207207e-03
4.05056371e-03	4.02800841e-03	3.97633339e-03	3.89846715e-03
3.81368647e-03	3.74904277e-03	3.73049306e-03	3.69950969e-03
3.62449006e-03	3.60826517e-03	3.46398321e-03	3.40205604e-03
3.35923344e-03	3.32238906e-03	3.22194444e-03	3.17971188e-03
3.15114993e-03	3.14859055e-03	3.07143844e-03	3.05622002e-03
3.01935139e-03	3.00581114e-03	2.90315855e-03	2.90148089e-03
2.86918224e-03	2.84286136e-03	2.80118531e-03	2.77532741e-03
2.75806525e-03	2.68062792e-03	2.66159817e-03	2.64169140e-03
2.62149485e-03	2.60461639e-03	2.58901057e-03	2.54777747e-03
2.52455614e-03	2.50312014e-03	2.47790123e-03	2.45632139e-03
2.42385499e-03	2.40341530e-03	2.38204128e-03	2.34435082e-03
2.33722011e-03	2.29535619e-03	2.25916618e-03	2.25113681e-03
2.21760645e-03	2.18768291e-03	2.15668943e-03	2.14209412e-03
2.12964433e-03	2.08171443e-03	2.06922156e-03	2.04137843e-03
2.02718273e-03	1.99616414e-03	1.97482821e-03	1.95440912e-03
1.91671559e-03	1.89835378e-03	1.88365323e-03	1.83200252e-03
1.81174526e-03	1.80074982e-03	1.78799974e-03	1.76729831e-03
1.73806047e-03	1.67993706e-03	1.66760993e-03	1.64904824e-03
1.61420926e-03	1.56399350e-03	1.55951636e-03	1.52990956e-03
1.51798411e-03	1.51053990e-03	1.48201907e-03	1.45012409e-03
1.42269722e-03	1.41740901e-03	1.34942658e-03	1.32982925e-03
1.31730047e-03	1.29728201e-03	1.27387802e-03	1.25805131e-03
1.21467877e-03	1.18829574e-03	1.17352838e-03	1.16402666e-03
1.14774792e-03	1.11971082e-03	1.09620645e-03	1.06937874e-03
1.04238715e-03	1.03367002e-03	1.02671001e-03	1.00237228e-03
9.71584996e-04	9.47448343e-04	9.40929300e-04	9.17184858e-04
9.01195503e-04	8.70954039e-04	8.57174008e-04	8.26585188e-04
8.20212515e-04	8.14160456e-04	7.91832057e-04	7.82846854e-04
7.43804208e-04	7.33137596e-04	7.23818969e-04	7.12469091e-04
6.96603045e-04	6.65611928e-04	6.58911313e-04	6.41618882e-04
6.30206647e-04	6.09402723e-04	6.03204337e-04	5.91160828e-04
5.61739453e-04	5.52532993e-04	5.29381571e-04	5.24810515e-04
5.19344346e-04	4.97522524e-04	4.87987631e-04	4.74308543e-04
4.60371283e-04	4.54883297e-04	4.41691403e-04	4.27752279e-04
4.21717140e-04	4.02499320e-04	3.99374354e-04	3.85228354e-04
3.69257900e-04	3.55037601e-04	3.42081001e-04	3.35866730e-04
3.25836751e-04	3.15929689e-04	2.97158501e-04	2.92186567e-04
2.82878040e-04	2.76896256e-04	2.60324878e-04	2.57209562e-04
2.53843395e-04	2.44354895e-04	2.37555222e-04	2.27268498e-04
2.19007219e-04	2.10417753e-04	2.05524021e-04	1.95790941e-04
1.86775801e-04	1.78488532e-04	1.72505451e-04	1.66758532e-04
1.55709304e-04	1.49328344e-04	1.44247245e-04	1.33549250e-04
1.28522759e-04	1.25949014e-04	1.18713064e-04	1.12713875e-04
1.08179073e-04	1.05109773e-04	1.03987183e-04	9.86569143e-05
9.30648764e-05	8.92241671e-05	7.95452024e-05	7.44589218e-05
7.27134971e-05	6.19355711e-05	5.93551256e-05	5.16258290e-05

```
 4.74298817e-05 4.32401019e-05 4.05618601e-05 3.96520640e-05
 3.91156393e-05 3.48875717e-05 3.37026701e-05 3.02296473e-05
 2.65304256e-05 2.49162352e-05 1.83667864e-05 1.79404847e-05
 1.62990808e-05 1.55874247e-05 1.51880204e-05 1.28241878e-05
 9.86532842e-06 9.14397506e-06 8.59022513e-06 7.33795915e-06
 4.63375806e-06 4.55512098e-06 2.79548336e-06 2.79006567e-06
 2.46907063e-06 1.69378498e-06 1.02728198e-06 8.60376643e-07
 6.03716708e-07 5.92608881e-07 5.92266881e-07 5.54155366e-07
 5.02199418e-07 4.52611332e-07 3.73714761e-07 2.08055819e-31
 7.77014225e-33 5.72637011e-33 4.37692130e-33 3.56500399e-33
 3.46891621e-33 2.24833008e-33 1.51787320e-33 1.51548771e-33
 1.33222508e-33 1.29817205e-33 1.04286500e-33 7.41230145e-34
 6.30844286e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.91810135e-34 3.91810135e-34 3.91810135e-34
 3.91810135e-34 3.79764987e-34 1.90094836e-34 1.47205230e-34
 1.35726896e-34 1.17738786e-34 8.20052573e-35 1.55211838e-35]
```

[ ]:

[95]:
```
n_comp = 12
pca = PCA(n_components = n_comp, random_state = 21)
pca2_result_train = pca.fit_transform(X_train)
pca2_result_test = pca.fit_transform(X_test)
```

[96]:
```
print(pca.explained_variance_ratio_)
```

```
[0.06633942 0.05032693 0.04571534 0.03757671 0.03182156 0.02892461
 0.02515386 0.02066116 0.01843474 0.01795676 0.01679762 0.01612837]
```

0.8 Predicting test_df values using XGBoost.

[97]:
```python
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

[98]:
```python
X_train,X_valid,y_train,y_valid =␣
 ↪train_test_split(pca2_result_train,y_train,test_size=0.20,random_state=21)
```

[99]:
```python
d_train = xgb.DMatrix(X_train, label=y_train)
d_valid = xgb.DMatrix(X_valid, label=y_valid)
d_test = xgb.DMatrix(pca2_result_test)
```

[100]:
```python
params = {}
params["objective"] = 'reg:linear'
params["eta"] = 0.02
params["max_depth"] = 6
params["subsample"] = 0.7
params["colsample_size"] = 0.7

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)
```

```
[17:39:51] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/objective/regression_obj.cu:170: reg:linear is now
deprecated in favor of reg:squarederror.
[17:39:51] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.3.0/src/learner.cc:541:
Parameters: { colsample_size } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[0]     train-rmse:98.98949     train-r2:-58.55114      valid-rmse:98.90655
valid-r2:-66.25152
[10]    train-rmse:81.15344     train-r2:-39.02448      valid-rmse:81.02206
valid-r2:-44.12928
```

```
[20]	train-rmse:66.62598	train-r2:-25.97734	valid-rmse:66.46175	
valid-r2:-29.36654
[30]	train-rmse:54.79835	train-r2:-17.24933	valid-rmse:54.59275	
valid-r2:-19.48905
[40]	train-rmse:45.18819	train-r2:-11.40971	valid-rmse:44.94870	
valid-r2:-12.88948
[50]	train-rmse:37.38238	train-r2:-7.49270	valid-rmse:37.12124	
valid-r2:-8.47320
[60]	train-rmse:31.07233	train-r2:-4.86759	valid-rmse:30.79905	
valid-r2:-5.52118
[70]	train-rmse:26.00524	train-r2:-3.10992	valid-rmse:25.73918	
valid-r2:-3.55451
[80]	train-rmse:21.92440	train-r2:-1.92124	valid-rmse:21.68717	
valid-r2:-2.23339
[90]	train-rmse:18.68536	train-r2:-1.12185	valid-rmse:18.49268	
valid-r2:-1.35099
[100]	train-rmse:16.11051	train-r2:-0.57736	valid-rmse:15.97890	
valid-r2:-0.75528
[110]	train-rmse:14.11708	train-r2:-0.21116	valid-rmse:14.05445	
valid-r2:-0.35794
[120]	train-rmse:12.54812	train-r2:0.04309	valid-rmse:12.59403	
valid-r2:-0.09039
[130]	train-rmse:11.33759	train-r2:0.21882	valid-rmse:11.50994	
valid-r2:0.08925
[140]	train-rmse:10.41561	train-r2:0.34070	valid-rmse:10.71759	
valid-r2:0.21033
[150]	train-rmse:9.69779	train-r2:0.42845	valid-rmse:10.15040	
valid-r2:0.29170
[160]	train-rmse:9.15309	train-r2:0.49085	valid-rmse:9.74635	
valid-r2:0.34697
[170]	train-rmse:8.73508	train-r2:0.53629	valid-rmse:9.46903	
valid-r2:0.38360
[180]	train-rmse:8.39644	train-r2:0.57155	valid-rmse:9.27509	
valid-r2:0.40859
[190]	train-rmse:8.14885	train-r2:0.59644	valid-rmse:9.13906	
valid-r2:0.42581
[200]	train-rmse:7.93635	train-r2:0.61722	valid-rmse:9.04401	
valid-r2:0.43769
[210]	train-rmse:7.76197	train-r2:0.63385	valid-rmse:8.99529	
valid-r2:0.44373
[220]	train-rmse:7.60846	train-r2:0.64819	valid-rmse:8.96121	
valid-r2:0.44794
[230]	train-rmse:7.49517	train-r2:0.65859	valid-rmse:8.94731	
valid-r2:0.44965
[240]	train-rmse:7.39046	train-r2:0.66806	valid-rmse:8.93924	
valid-r2:0.45065
[250]	train-rmse:7.28970	train-r2:0.67705	valid-rmse:8.93430	
valid-r2:0.45125
```

```
[260]	train-rmse:7.19950	train-r2:0.68500	valid-rmse:8.92014
valid-r2:0.45299
[270]	train-rmse:7.11905	train-r2:0.69200	valid-rmse:8.92622
valid-r2:0.45225
[280]	train-rmse:7.04576	train-r2:0.69831	valid-rmse:8.93590
valid-r2:0.45105
[290]	train-rmse:6.96334	train-r2:0.70532	valid-rmse:8.94407
valid-r2:0.45005
[300]	train-rmse:6.88008	train-r2:0.71233	valid-rmse:8.94884
valid-r2:0.44947
[310]	train-rmse:6.80763	train-r2:0.71835	valid-rmse:8.95748
valid-r2:0.44840
```

[ ]:

```
[101]: p_test = clf.predict(d_test)
p_test
```

```
[101]: array([ 82.744354, 105.80603 ,  82.08374 , ..., 102.205154, 105.8811  ,
         95.3124  ], dtype=float32)
```

0.9 Creating a result dataframe

```
[102]: sub = pd.DataFrame()
sub["ID"] = id_test
sub["y"] = p_test
sub.to_csv("xgb.csv",index=False)
```