Mercedes-Benz Greener Manufacturing

July 29, 2020

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.

[1]:
```
# Importing library

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing # Import Label Encoder
```

[2]:
```
# Read csv
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
```

```
print(train_df.shape) # Find Number of rows and columns
print(train_df.columns)

print(test_df.shape) # Find Number of rows and columns
print(test_df.columns)

train_df.head() # Show first 5 records
```

```
(4209, 378)
Index(['ID', 'y', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=378)
(4209, 377)
Index(['ID', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=377)
```

[2]:

	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]

[3]:
```
# Describe the dataset i.r.t its data Distribution

train_df.describe()
```

[3]:

	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]
```

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

[4]:
```
# Check the variance

train_df.var()
```

[4]:
```
ID      5.941936e+06
y       1.607667e+02
X10     1.313092e-02
X11     0.000000e+00
X12     6.945713e-02
            …
X380    8.014579e-03
X382    7.546747e-03
X383    1.660732e-03
X384    4.750593e-04
X385    1.423823e-03
Length: 370, dtype: float64
```

[5]:
```
# Find out the variance is equal to zero for any columns

(train_df.var() == 0)
```

[5]:
```
ID      False
y       False
X10     False
X11      True
X12     False
        …
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 370, dtype: bool
```

[6]:
```
(train_df.var() == 0).values
```

[6]:
```
array([False, False, False,  True, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
```

```
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False,  True, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False,  True,  True, False, False,
        True, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False])
```

[7]:
```
variance_with_zero = train_df.var()[train_df.var()==0].index.values
variance_with_zero
```

[7]:
```
array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'], dtype=object)
```

[8]:
```
# Drop zero variance variables

train_df = train_df.drop(variance_with_zero, axis=1)
```

[9]:
```
print(train_df.shape)
```

```
(4209, 366)
```

[10]:
```
# As ID column is irrelevant for our prediction hence we drop this column

train_df = train_df.drop(['ID'], axis=1)
```

[11]:
```
train_df.head()
```

[11]:
```
          y  X0 X1  X2 X3 X4 X5 X6 X8  X10  …  X375  X376  X377  X378  X379  \
0  130.81   k  v  at  a  d  u  j  o    0  …     0     0     1     0     0
1   88.53   k  t  av  e  d  y  l  o    0  …     1     0     0     0     0
2   76.26  az  w   n  c  d  x  j  x    0  …     0     0     0     0     0
3   80.62  az  t   n  f  d  x  l  e    0  …     0     0     0     0     0
4   78.02  az  v   n  f  d  h  d  n    0  …     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 365 columns]
```

0.0.2 Check for null and unique values for test and train sets.

[12]:
```
train_df.isnull().sum().values
```

[12]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

[13]:
```
train_df.isnull().any()
```

[13]:
```
y       False
X0      False
X1      False
X2      False
X3      False
        …
```

```
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 365, dtype: bool
```

[14]:
```
test_df.isnull().sum().values
```

[14]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0])
```

[15]:
```
# Find unique records

train_df.nunique()
```

[15]:
```
y       2545
X0        47
X1        27
X2        44
X3         7
        …
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 365, dtype: int64
```

0.0.3 Filter out the columns having object datatype

[16]:
```
object_datatypes = train_df.select_dtypes(include=[object])
object_datatypes
```

[16]:
```
      X0 X1  X2 X3 X4  X5 X6 X8
0      k  v  at  a  d   u  j  o
1      k  t  av  e  d   y  l  o
2     az  w   n  c  d   x  j  x
3     az  t   n  f  d   x  l  e
4     az  v   n  f  d   h  d  n
…   .. ..  .. .. ..  .. .. ..
4204  ak  s  as  c  d  aa  d  q
4205   j  o   t  d  d  aa  h  h
4206  ak  v   r  a  d  aa  g  e
4207  al  r   e  f  d  aa  l  u
4208   z  r  ae  c  d  aa  g  w

[4209 rows x 8 columns]
```

[17]:
```
object_datatype_columns = object_datatypes.columns
object_datatype_columns
```

[17]:
```
Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype='object')
```

0.0.4 Apply label encoder.

[18]:
```
# Initialize Label Encoder object

label_encoder = preprocessing.LabelEncoder()
train_df['X0'].unique()
```

[18]:
```
array(['k', 'az', 't', 'al', 'o', 'w', 'j', 'h', 's', 'n', 'ay', 'f', 'x',
       'y', 'aj', 'ak', 'am', 'z', 'q', 'at', 'ap', 'v', 'af', 'a', 'e',
       'ai', 'd', 'aq', 'c', 'aa', 'ba', 'as', 'i', 'r', 'b', 'ax', 'bc',
       'u', 'ad', 'au', 'm', 'l', 'aw', 'ao', 'ac', 'g', 'ab'],
      dtype=object)
```

[19]:
```
# Encode and transform object data to interger

train_df['X0'] = label_encoder.fit_transform(train_df['X0'])
```

[20]:
```
train_df['X0'].unique()
```

[20]:
```
array([32, 20, 40,  9, 36, 43, 31, 29, 39, 35, 19, 27, 44, 45,  7,  8, 10,
       46, 37, 15, 12, 42,  5,  0, 26,  6, 25, 13, 24,  1, 22, 14, 30, 38,
       21, 18, 23, 41,  4, 16, 34, 33, 17, 11,  3, 28,  2])
```

[21]:
```
# 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'])
```

[22]:
```
train_df.head()
```

[22]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	…	X375	X376	X377	X378	\
0	130.81	32	23	17	0	3	24	9	14	0	…	0	0	1	0	
1	88.53	32	21	19	4	3	28	11	14	0	…	1	0	0	0	
2	76.26	20	24	34	2	3	27	9	23	0	…	0	0	0	0	
3	80.62	20	21	34	5	3	27	11	4	0	…	0	0	0	0	
4	78.02	20	23	34	5	3	12	3	13	0	…	0	0	0	0	

	X379	X380	X382	X383	X384	X385
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0

[5 rows x 365 columns]

0.0.5 Perform dimensionality reduction (PCA)

[23]:
```
from sklearn.decomposition import PCA
```

[24]:
```
# PCA with 95%

sklearn_pca = PCA(n_components=0.95)
```

[25]:
```
sklearn_pca.fit(train_df)
```

[25]:
```
PCA(copy=True, iterated_power='auto', n_components=0.95, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

[26]:
```
x_train_transformed = sklearn_pca.transform(train_df)
```

[27]:
```
print(x_train_transformed.shape)
```

```
(4209, 6)
```

[28]:
```
# PCA with 98%

sklearn_pca_98 = PCA(n_components=0.98)
```

[29]:
```
sklearn_pca_98.fit(train_df)
```

[29]:
```
PCA(copy=True, iterated_power='auto', n_components=0.98, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

[30]:
```
x_train_transformed_98 = sklearn_pca_98.transform(train_df)
print(x_train_transformed_98.shape)
```

```
(4209, 12)
```

[31]:
```
train_df.y
```

[31]:
```
0       130.81
1        88.53
2        76.26
3        80.62
4        78.02
         …
4204    107.39
4205    108.77
4206    109.22
4207     87.48
4208    110.85
Name: y, Length: 4209, dtype: float64
```

0.0.6 Train and Test split on Train dataset

[32]:
```
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)
```

[33]:
```
print(xtrain)
print(xtrain.shape)
```

```
      X0  X1  X2  X3  X4  X5  X6  X8  X10  X12  …  X375  X376  X377  X378  \
370   35  13  16   1   3   9   6  19    0    0  …     0     0     0     0
3392  15  10  16   2   3  23   9  16    0    0  …     0     0     1     0
2208  31   3  16   2   3  15   2  21    0    0  …     0     0     1     0
```

```
3942  35  20   8   6   3  26   6  14    0    1  …     1     0     0     0
1105  36  13  16   5   3   1   6   0    0    0  …     0     0     0     0
…   ..  ..  ..  ..  ..  ..  ..  ..  …  …  …   …   …   …   …
3444  31  10  16   2   3  22  11  17    0    0  …     0     0     1     0
466   20  25  25   2   3   9   9   9    0    0  …     0     0     0     0
3092  45  24   3   2   3  21   8   2    0    0  …     1     0     0     0
3772  45  19   8   5   3  25   8   1    0    1  …     0     0     0     0
860   22   1   7   2   3   5   9  17    0    0  …     1     0     0     0

      X379  X380  X382  X383  X384  X385
370      0     0     0     0     0     0
3392     0     0     0     0     0     0
2208     0     0     0     0     0     0
3942     0     0     0     0     0     0
1105     0     0     0     0     0     0
…    …   …   …   …   …   …
3444     0     0     0     0     0     0
466      0     0     1     0     0     0
3092     0     0     0     0     0     0
3772     0     0     0     0     0     0
860      0     0     0     0     0     0

[2946 rows x 364 columns]
(2946, 364)
```

[34]:
```
print(ytrain)
print(ytrain.shape)
```

```
370      95.13
3392    117.36
2208    109.01
3942     93.77
1105    103.41
         …
3444    109.42
466      78.25
3092     92.18
3772     91.92
860      87.71
Name: y, Length: 2946, dtype: float64
(2946,)
```

[35]:
```
print(xtest)
print(xtest.shape)
```

```
      X0  X1  X2  X3  X4  X5  X6  X8  X10  X12  …  X375  X376  X377  X378  \
1073   9  16   7   5   3   6   9  11    0    0  …     0     0     0     0
144   27  13   3   5   3  13   8  22    0    0  …     0     0     0     0
```

```
2380  31   1  21   2   3  18  11  14    1    0  …     1     0     0     0
184   20  25  22   2   3  13   9  11    0    0  …     0     0     0     0
2587   8  23   8   3   3  17   8  17    0    0  …     0     0     0     0
…    ..  ..  ..  ..  ..  ..  ..  ..  …  …  …   …   …   …   …
2493  27  20  16   2   3  18  10   5    0    0  …     0     0     1     0
3388  40  19  24   5   3  23   3  19    0    0  …     0     0     0     0
3997  22   3   7   0   3  26   6  18    0    0  …     0     0     1     0
383   40   1  16   6   3   9   8   0    0    0  …     1     0     0     0
3364  27   4  33   2   3  23   6  24    0    0  …     0     0     1     0

      X379  X380  X382  X383  X384  X385
1073     0     0     0     0     0     0
144      0     0     0     0     0     0
2380     0     0     0     0     0     0
184      0     0     1     0     0     0
2587     0     0     0     0     0     0
…    …   …   …   …   …   …
2493     0     0     0     0     0     0
3388     0     0     0     0     0     0
3997     0     0     0     0     0     0
383      0     0     0     0     0     0
3364     0     0     0     0     0     0

[1263 rows x 364 columns]
(1263, 364)
```

[36]:
```
# PCA with 95% for xtrain

pca_xtrain = PCA(n_components=0.95)
pca_xtrain.fit(xtrain)
```

[36]:
```
PCA(copy=True, iterated_power='auto', n_components=0.95, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

[37]:
```
pca_xtrain_transformed = pca_xtrain.transform(xtrain)
print(pca_xtrain_transformed.shape)
```

```
(2946, 6)
```

[38]:
```
# PCA with 95% for xtest

pca_xtest = PCA(n_components=0.95)
pca_xtest.fit(xtest)
```

[38]:
```
PCA(copy=True, iterated_power='auto', n_components=0.95, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

[39]:
```
pca_xtest_transformed = pca_xtest.transform(xtest)
print(pca_xtest_transformed.shape)
```

```
(1263, 6)
```

[40]:
```
print(pca_xtest.explained_variance_)
print(pca_xtest.explained_variance_ratio_)
```

```
[206.79524961 120.24273955  67.64680756  61.94375666  48.08214872
   8.7271811 ]
[0.38517942 0.22396563 0.12599979 0.11537722 0.08955841 0.01625536]
```

0.0.7 PCA for test_df dataset

[41]:
```
test_df
```

[41]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	…	X375	X376	X377	X378	\
0	1	az	v	n	f	d	t	a	w	0	…	0	0	0	1	
1	2	t	b	ai	a	d	b	g	y	0	…	0	0	1	0	
2	3	az	v	as	f	d	a	j	j	0	…	0	0	0	1	
3	4	az	l	n	f	d	z	l	n	0	…	0	0	0	1	
4	5	w	s	as	c	d	y	i	m	0	…	1	0	0	0	
…	…	..	..	..	..	..	..	..	..	…	…	…	…	…	…	
4204	8410	aj	h	as	f	d	aa	j	e	0	…	0	0	0	0	
4205	8411	t	aa	ai	d	d	aa	j	y	0	…	0	1	0	0	
4206	8413	y	v	as	f	d	aa	d	w	0	…	0	0	0	0	
4207	8414	ak	v	as	a	d	aa	c	q	0	…	0	0	1	0	
4208	8416	t	aa	ai	c	d	aa	g	r	0	…	1	0	0	0	

	X379	X380	X382	X383	X384	X385
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
…	…	…	…	…	…	…
4204	0	0	0	0	0	0
4205	0	0	0	0	0	0
4206	0	0	0	0	0	0
4207	0	0	0	0	0	0
4208	0	0	0	0	0	0

```
[4209 rows x 377 columns]
```

[42]:
```
test_object_datatypes = test_df.select_dtypes(include=[object])
test_object_datatypes
```

[42]:
```
      X0  X1  X2 X3 X4  X5 X6 X8
0     az   v   n  f  d   t  a  w
1      t   b  ai  a  d   b  g  y
2     az   v  as  f  d   a  j  j
3     az   l   n  f  d   z  l  n
4      w   s  as  c  d   y  i  m
…   ..  ..  .. .. ..  .. .. ..
4204  aj   h  as  f  d  aa  j  e
4205   t  aa  ai  d  d  aa  j  y
4206   y   v  as  f  d  aa  d  w
4207  ak   v  as  a  d  aa  c  q
4208   t  aa  ai  c  d  aa  g  r

[4209 rows x 8 columns]
```

[43]:
```
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'])
```

[44]:
```
print(test_df)
print(test_df.shape)
```

```
        ID  X0  X1  X2  X3  X4  X5  X6  X8  X10  …  X375  X376  X377  X378  \
0        1  21  23  34   5   3  26   0  22    0  …     0     0     0     1
1        2  42   3   8   0   3   9   6  24    0  …     0     0     1     0
2        3  21  23  17   5   3   0   9   9    0  …     0     0     0     1
3        4  21  13  34   5   3  31  11  13    0  …     0     0     0     1
4        5  45  20  17   2   3  30   8  12    0  …     1     0     0     0
…    …  ..  ..  ..  ..  ..  ..  ..  ..  …  …   …   …  …  …
4204  8410   6   9  17   5   3   1   9   4    0  …     0     0     0     0
4205  8411  42   1   8   3   3   1   9  24    0  …     0     1     0     0
4206  8413  47  23  17   5   3   1   3  22    0  …     0     0     0     0
4207  8414   7  23  17   0   3   1   2  16    0  …     0     0     1     0
4208  8416  42   1   8   2   3   1   6  17    0  …     1     0     0     0

      X379  X380  X382  X383  X384  X385
0        0     0     0     0     0     0
1        0     0     0     0     0     0
2        0     0     0     0     0     0
3        0     0     0     0     0     0
4        0     0     0     0     0     0
…    …   …   …   …   …   …   …
```

```
4204     0     0     0     0     0     0
4205     0     0     0     0     0     0
4206     0     0     0     0     0     0
4207     0     0     0     0     0     0
4208     0     0     0     0     0     0

[4209 rows x 377 columns]
(4209, 377)
```

[45]:
```
test_df = test_df.drop('ID',axis=1)
```

[46]:
```
# PCA with 95% for test_df

pca_test_df = PCA(n_components=0.95)
pca_test_df.fit(test_df)
```

[46]:
```
PCA(copy=True, iterated_power='auto', n_components=0.95, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

[47]:
```
pca_test_df_transformed = pca_test_df.transform(test_df)
print(pca_test_df_transformed.shape)
```

```
(4209, 6)
```

[48]:
```
print(pca_test_df.explained_variance_)
print(pca_test_df.explained_variance_ratio_)
```

```
[247.07875325 100.33535335  77.48364816  62.33258307  48.95689653
   8.14203723]
[0.43515102 0.17670897 0.13646292 0.10977912 0.08622208 0.01433962]
```

[49]:
```
y
```

[49]:
```
0       130.81
1        88.53
2        76.26
3        80.62
4        78.02
         ...
4204    107.39
4205    108.77
4206    109.22
4207     87.48
4208    110.85
Name: y, Length: 4209, dtype: float64
```

0.0.8 Perform XGboost

[78]:
```
from sklearn import svm
from sklearn import model_selection
import xgboost as xgb
```

[80]:
```
model = xgb.XGBRegressor(objective="reg:linear",learning_rate=0.1)
model.fit(pca_xtrain, ytrain) # I am getting a small error here, unable to␣
 ↪solve.Please help me with soliution.
y_pred = model.predict(pca_x_test)
y_pred
model.predict(pca_test_df)
```

[ ]:
```
** End **
```