benz-greener-manufacturing-1-2-3

August 14, 2023

[1]:
```python
# Importing library

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing # Import Label Encoder
```

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

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

[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()
```

```
/var/folders/tt/z2svxjk10ljdks5gf4lft4kr0000gq/T/ipykernel_4542/2679125992.py:3:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  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)
```

```
/var/folders/tt/z2svxjk10ljdks5gf4lft4kr0000gq/T/ipykernel_4542/2664506896.py:3:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  (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]:
```python
(train_df.var() == 0).values
```

```
/var/folders/tt/z2svxjk10ljdks5gf4lft4kr0000gq/T/ipykernel_4542/2190880080.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  (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, 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]:
```python
variance_with_zero = train_df.var()[train_df.var()==0].index.values
variance_with_zero
```

```
/var/folders/tt/z2svxjk10ljdks5gf4lft4kr0000gq/T/ipykernel_4542/974452901.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  variance_with_zero = train_df.var()[train_df.var()==0].index.values
```

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

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

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

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

```
(4209, 366)
```

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

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

[11]:
```python
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]:
```python
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]:
```python
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]:
```python
# Find unique records

train_df.nunique()
```

[14]:
```
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 Apply label encoder.

[15]:
```
# Initialize Label Encoder object

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

[15]:
```
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)
```

[16]:
```
# Encode and transform object data to interger

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

[17]:
```
train_df['X0'].unique()
```

[17]:
```
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])
```

[18]:
```
# 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'])
```

[19]:
```
train_df.head()
```

[19]:
```
        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]
```

0.0.4 Perform dimensionality reduction (PCA)

[20]:
```python
from sklearn.decomposition import PCA
```

[21]:
```python
# PCA with 95%

sklearn_pca = PCA(n_components=0.95)
```

[22]:
```python
train_dfwy = train_df.drop('y', axis=1)
```

[23]:
```python
sklearn_pca.fit(train_dfwy)
```

[23]:
```
PCA(n_components=0.95)
```

[24]:
```python
x_train_transformed = sklearn_pca.transform(train_dfwy)
```

[25]:
```python
print(x_train_transformed.shape)
```

```
(4209, 6)
```

[26]:
```python
x_train_transformed=pd.DataFrame(x_train_transformed)
x_train_transformed
```

[26]:

	0	1	2	3	4	5
0	0.614765	-0.133009	15.624460	3.687564	1.359574	-2.691417
1	0.565407	1.560333	17.909581	-0.092902	1.536648	-4.442877
2	16.201713	12.292846	17.633540	0.186308	11.850820	-2.155389
3	16.149998	13.535419	14.898695	-3.140917	-6.832193	-4.290014
4	16.459103	13.175004	4.403096	7.671151	2.139916	3.763860
…	…	…	…	…	…	…
4204	22.161403	-7.184320	-8.659404	10.774860	4.669902	3.527910
4205	6.153949	22.828146	-8.314658	10.303221	-3.089276	0.073621
4206	29.004660	14.860905	-7.753332	11.224415	-5.846985	0.789306
4207	22.972422	1.684824	-9.031248	9.749805	9.449557	-4.355228
4208	-17.283048	-9.951982	-3.719360	18.343096	8.401706	0.509480

[4209 rows x 6 columns]

[27]:
```python
# PCA with 98%

sklearn_pca_98 = PCA(n_components=0.98)
```

[28]:
```python
sklearn_pca_98.fit(train_dfwy)
```

[28]: PCA(n_components=0.98)

[29]:
```python
x_train_transformed_98 = sklearn_pca_98.transform(train_dfwy)
print(x_train_transformed_98.shape)
```

(4209, 14)

[30]:
```python
y=pd.DataFrame(train_df.y)
y
```

[30]:

	y
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

[4209 rows x 1 columns]

0.0.5 Train and Test split on Train dataset

[31]:
```python
x = x_train_transformed
y = y
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.3,random_state=42)
```

[32]:
```python
print(xtrain)
print(xtrain.shape)
```

	0	1	2	3	4	5
370	-4.936054	-0.270541	-1.307269	5.535125	7.026251	0.764564
3392	12.832015	-5.551105	5.313741	-9.335122	4.756201	-2.508884
2208	-3.880524	-1.431836	-2.457911	-6.640973	9.649977	5.031871
3942	-5.201457	-8.389970	16.300885	1.637502	0.647784	0.392336

```
1105  -5.064020   1.288561  -8.691286   9.524665 -11.856796  0.779813
…          …         …         …         …         …         …
3444  -2.083986  -1.334860   6.753467  -5.563931   5.221174 -4.448352
466   14.735468   4.967978   2.638603  10.982170  -2.481212 -2.553799
3092 -14.521561  -9.617927  14.861648   8.878755 -12.087611 -2.146663
3772 -14.407043  -4.817999  15.400567   2.440266 -12.498499 -1.761588
860    1.748483 -11.740613 -13.314692  -4.063811   5.779192 -1.991844

[2946 rows x 6 columns]
(2946, 6)
```

[33]:
```python
print(ytrain)
print(ytrain.shape)
```

```
           y
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

[2946 rows x 1 columns]
(2946, 1)
```

[34]:
```python
print(xtest)
print(xtest.shape)
```

```
              0          1          2          3          4         5
1073  17.841395 -15.245960  -6.299416   4.391821  -0.724654 -2.742346
144   -0.990921 -15.208289   1.040124   2.620286   9.256619 -1.371609
2380  -2.894524   3.927628  -1.575664 -10.600740   3.257902 -3.705331
184   13.849039   1.874404   6.003682   8.912134  -0.749994 -2.604863
2587  20.639205 -15.373246   6.846871   3.755760   4.645073 -1.805657
…          …         …         …         …         …         …
2493   4.689242  -1.803997   7.946399   3.503388  -7.211226 -3.605631
3388  -6.224988   8.120305  14.260077   2.916254   6.657316  3.789372
3997   2.075211 -12.413009   4.970436 -14.636184   6.203056  0.671528
383  -11.941170   2.403183  -8.341066  -3.900772 -11.266806 -0.791802
3364   4.352447  13.481377   4.366949 -11.765672  13.770395  1.687826

[1263 rows x 6 columns]
(1263, 6)
```

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

```
            y
1073    97.94
144     96.41
2380   105.83
184     79.09
2587   108.69
…        …
2493   115.25
3388    88.59
3997    92.90
383     98.24
3364    91.46

[1263 rows x 1 columns]
(1263, 1)
```

1 Perform XGboost

[41]:
```
!pip install xgboost
import xgboost as xgb
from sklearn.metrics import accuracy_score
xgb_classifier=xgb.XGBClassifier()

xtrain=pd.DataFrame(xtrain)
ytrain=pd.DataFrame(ytrain)
xtest=pd.DataFrame(xtest)
ytest=pd.DataFrame(ytest)

label_encoder = preprocessing.LabelEncoder()
ytrain = label_encoder.fit_transform(ytrain.values.ravel())
ytest = label_encoder.fit_transform(ytest.values.ravel())

xgb_classifier.fit(xtrain, ytrain)
y_pred =xgb_classifier.predict(xtest)
print("Accuracy of Model:", accuracy_score(ytest, y_pred))
y_pred
```

```
Requirement already satisfied: xgboost in /opt/anaconda3/lib/python3.9/site-
packages (1.7.6)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.9/site-
packages (from xgboost) (1.20.3)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.9/site-
packages (from xgboost) (1.7.1)
Accuracy of Model: 0.000791765637371338
```

[41]: array([682, 499, 1561, …, 232, 1446,   22])

[ ]: