Mercedes-Benz

December 17, 2021

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: Train_data=pd.read_csv(r'C:
      ↪\Users\LUCKY\Desktop\GRR\Machine-Learning--Projects-master\Projects\Projects␣
      ↪for Submission\Project 1 - Mercedes-Benz Greener Manufacturing\Dataset for␣
      ↪the project\train\train.csv')
```

```
[3]: Train_data
```

```
[3]:         ID       y  X0 X1  X2 X3 X4  X5 X6 X8  …  X375  X376  X377  X378  \
     0        0  130.81   k  v  at  a  d   u  j  o  …     0     0     1     0
     1        6   88.53   k  t  av  e  d   y  l  o  …     1     0     0     0
     2        7   76.26  az  w   n  c  d   x  j  x  …     0     0     0     0
     3        9   80.62  az  t   n  f  d   x  l  e  …     0     0     0     0
     4       13   78.02  az  v   n  f  d   h  d  n  …     0     0     0     0
     …     …       …   .. ..  .. .. ..  .. .. ..  …   …    …    …    …
     4204  8405  107.39  ak  s  as  c  d  aa  d  q  …     1     0     0     0
     4205  8406  108.77   j  o   t  d  d  aa  h  h  …     0     1     0     0
     4206  8412  109.22  ak  v   r  a  d  aa  g  e  …     0     0     1     0
     4207  8415   87.48  al  r   e  f  d  aa  l  u  …     0     0     0     0
     4208  8417  110.85   z  r  ae  c  d  aa  g  w  …     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     1     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 378 columns]
```

[4]:
```
Test_data=pd.read_csv(r'C:
↪\Users\LUCKY\Desktop\GRR\Machine-Learning--Projects-master\Projects\Projects␣
↪for Submission\Project 1 - Mercedes-Benz Greener Manufacturing\Dataset for␣
↪the project\test\test.csv')
```

[5]:
```
Test_data
```

[5]:
```
        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]
```

[6]:
```
Train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 378 entries, ID to X385
dtypes: float64(1), int64(369), object(8)
memory usage: 12.1+ MB
```

[7]:
```
Train_data.var()
```

```
[7]: 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
```

Dropping the Zero Variance columns in Train and Test data

```
[ ]:
```

```
[8]: Train_data1=Train_data.loc[:, (Train_data != Train_data.iloc[0]).any()]
```

```
[9]: Train_data1
```

```
[9]:         ID       y  X0 X1  X2 X3 X4  X5 X6 X8 …   X375  X376  X377  X378  \
     0         0  130.81   k  v  at  a  d   u  j  o …      0     0     1     0
     1         6   88.53   k  t  av  e  d   y  l  o …      1     0     0     0
     2         7   76.26  az  w   n  c  d   x  j  x …      0     0     0     0
     3         9   80.62  az  t   n  f  d   x  l  e …      0     0     0     0
     4        13   78.02  az  v   n  f  d   h  d  n …      0     0     0     0
     …    …     …  .. ..  .. .. ..  .. .. ..  …   …   …   …   …
     4204  8405  107.39  ak  s  as  c  d  aa  d  q …      1     0     0     0
     4205  8406  108.77   j  o   t  d  d  aa  h  h …      0     1     0     0
     4206  8412  109.22  ak  v   r  a  d  aa  g  e …      0     0     1     0
     4207  8415   87.48  al  r   e  f  d  aa  l  u …      0     0     0     0
     4208  8417  110.85   z  r  ae  c  d  aa  g  w …      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     1     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 366 columns]
```

[10]:
```
Train_data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 366 entries, ID to X385
dtypes: float64(1), int64(357), object(8)
memory usage: 11.8+ MB
```

[11]:
```
Test_data1=Test_data.loc[:,(Test_data!=Test_data.iloc[0]).any()]
```

[12]:
```
Test_data1
```

[12]:
```
        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 372 columns]
```

[13]:
```
Test_data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 372 entries, ID to X385
```

```
dtypes: int64(364), object(8)
memory usage: 11.9+ MB
```

```
[ ]:
```

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

```
[14]: Train_data.isnull().values.any()
```

```
[14]: False
```

```
[15]: Test_data.isnull().values.any()
```

```
[15]: False
```

```
[16]: Train_data.nunique()
```

```
[16]: ID      4209
      y       2545
      X0        47
      X1        27
      X2        44
              …
      X380       2
      X382       2
      X383       2
      X384       2
      X385       2
      Length: 378, dtype: int64
```

```
[17]: Test_data.nunique()
```

```
[17]: ID      4209
      X0        49
      X1        27
      X2        45
      X3         7
              …
      X380       2
      X382       2
      X383       2
      X384       2
      X385       2
      Length: 377, dtype: int64
```

```
[ ]:
```

Apply label encoder.

[18]:
```
from sklearn.preprocessing import LabelEncoder
```

[19]:
```
le=LabelEncoder()
```

[20]:
```
Train_data2=Train_data.iloc[:,0:10]
```

[21]:
```
Train_data2
```

[21]:
```
        ID       y  X0 X1  X2 X3 X4  X5 X6 X8
0        0  130.81   k  v  at  a  d   u  j  o
1        6   88.53   k  t  av  e  d   y  l  o
2        7   76.26  az  w   n  c  d   x  j  x
3        9   80.62  az  t   n  f  d   x  l  e
4       13   78.02  az  v   n  f  d   h  d  n
…      …      …    .. ..  .. .. ..  .. .. ..
4204  8405  107.39  ak  s  as  c  d  aa  d  q
4205  8406  108.77   j  o   t  d  d  aa  h  h
4206  8412  109.22  ak  v   r  a  d  aa  g  e
4207  8415   87.48  al  r   e  f  d  aa  l  u
4208  8417  110.85   z  r  ae  c  d  aa  g  w

[4209 rows x 10 columns]
```

[22]:
```
Train_data3=Train_data2.apply(le.fit_transform)
```

[23]:
```
Train_data3
```

[23]:
```
        ID     y  X0  X1  X2  X3  X4  X5  X6  X8
0        0  2466  32  23  17   0   3  24   9  14
1        1   366  32  21  19   4   3  28  11  14
2        2    69  20  24  34   2   3  27   9  23
3        3   133  20  21  34   5   3  27  11   4
4        4   106  20  23  34   5   3  12   3  13
…      …    …    ..  ..  ..  ..  ..  ..  ..  ..
4204  4204  1657   8  20  16   2   3   0   3  16
4205  4205  1766  31  16  40   3   3   0   7   7
4206  4206  1801   8  23  38   0   3   0   6   4
4207  4207   280   9  19  25   5   3   0  11  20
4208  4208  1921  46  19   3   2   3   0   6  22

[4209 rows x 10 columns]
```

[24]:
```
Test_data2=Test_data.iloc[:,0:9]
```

[25]:
```
Test_data2
```

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

      [4209 rows x 9 columns]
```

```
[26]: Test_data3=Test_data2.apply(le.fit_transform)
```

```
[27]: Test_data3
```

```
[27]:         ID  X0  X1  X2  X3  X4  X5  X6  X8
      0        0  21  23  34   5   3  26   0  22
      1        1  42   3   8   0   3   9   6  24
      2        2  21  23  17   5   3   0   9   9
      3        3  21  13  34   5   3  31  11  13
      4        4  45  20  17   2   3  30   8  12
      …     …  ..  ..  ..  ..  ..  ..  ..  ..
      4204  4204   6   9  17   5   3   1   9   4
      4205  4205  42   1   8   3   3   1   9  24
      4206  4206  47  23  17   5   3   1   3  22
      4207  4207   7  23  17   0   3   1   2  16
      4208  4208  42   1   8   2   3   1   6  17

      [4209 rows x 9 columns]
```

Perform dimensionality reduction.

```
[28]: len(Train_data3.columns)
      Train_data3.shape
```

```
[28]: (4209, 10)
```

```
[29]: len(Test_data3.columns)
      Test_data3.shape
```

```
[29]: (4209, 9)
```

PCA for Train_data3

[30]:

```
x=Train_data3.iloc[:,0:10].values

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
```

[31]:

```
from sklearn.decomposition import PCA
pca=PCA()
pca_x=pca.fit_transform(x)
```

[32]:

```
print(pca_x)
len(pca_x)
```

```
[[ 0.38607142  0.92345155  1.5707565  …  1.07507039 -1.6572576
   2.22243546]
 [-0.25581144 -0.06270924 -0.92827036 …  0.43529218 -0.25547872
   2.54594216]
 [-0.24914448  0.76246706 -0.27118973 …  1.03576569  0.60645749
   2.54744973]
 …
 [-0.09526195  2.70326504  1.69630555 …  1.3441078   0.05199277
  -2.20622738]
 [-0.35190887  1.08426847 -1.13468067 …  0.39833108  1.25229046
  -2.28418473]
 [-0.18672818 -0.5450192  -0.15687854 …  0.85056544 -1.60056716
  -2.34400381]]
```

[32]:

```
4209
```

[33]:

```
explained_variance=pca.explained_variance_ratio_
explained_variance
```

[33]:

```
array([0.16753662, 0.14709502, 0.12883563, 0.10384823, 0.10041396,
       0.09705171, 0.08611389, 0.07022931, 0.06425298, 0.03462266])
```

[34]:

```
explained_variance.sum()
cumulative=explained_variance.cumsum()

cumulative
```

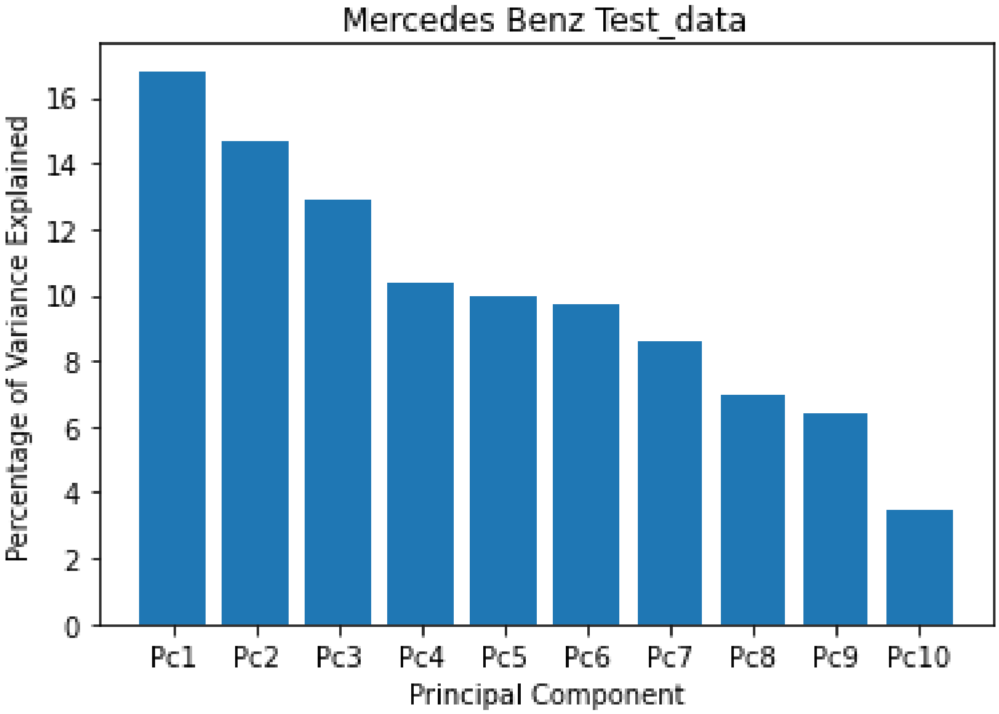
[34]:

```
array([0.16753662, 0.31463163, 0.44346726, 0.54731549, 0.64772945,
       0.74478116, 0.83089505, 0.90112435, 0.96537734, 1.        ])
```

[35]:

```
per_var=np.round(pca.explained_variance_ratio_*100,decimals=1)
labels=['Pc' + str(x) for x in range(1,len(per_var)+1)]
```

[36]:
```
plt.bar(x=range(1, len(per_var)+1), height=per_var, tick_label=labels)
#cumulative = explained_variance.cumm()
#plt.plot(cumulative='green')
plt.ylabel("Percentage of Variance Explained")
plt.xlabel("Principal Component")
plt.title('Mercedes Benz Test_data')
plt.show()
```



PCA for Test_data3

[37]:
```
y=Test_data3.iloc[:,0:9].values

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
y=sc.fit_transform(y)
```

[38]:
```
from sklearn.decomposition import PCA
pca=PCA()
pca_y=pca.fit_transform(y)
```

[39]:
```
print(pca_y)
len(pca_y)
```

```
[[-0.31205274  2.12686537  1.06986954 … -1.53452808 -0.53157011
```

```
  2.02936886]
[ 2.14581904 -1.58406976  0.16594777 …  0.93822321  0.42062354
  0.79468175]
[ 1.82031621  2.20280205  0.43589131 … -0.01433907  0.80288351
  0.03190381]
…
[-0.40110652  0.42521405  1.81087735 … -1.13773022  1.02485349
 -2.40420897]
[-0.57290256  1.2407005  -0.0280556  …  1.65977752  0.66430675
 -2.18207047]
[ 0.38685054 -1.67874018  0.51587481 …  0.54095539 -0.12533595
 -2.37304481]]
```

[39]:

```
4209
```

[40]:

```
explained_variance=pca.explained_variance_ratio_
explained_variance
```

[40]:

```
array([0.18721572, 0.16054922, 0.12943966, 0.11477469, 0.11288262,
       0.10047691, 0.08217049, 0.07397084, 0.03851984])
```

[41]:

```
explained_variance.sum()
cumulative=explained_variance.cumsum()

cumulative
```

[41]:

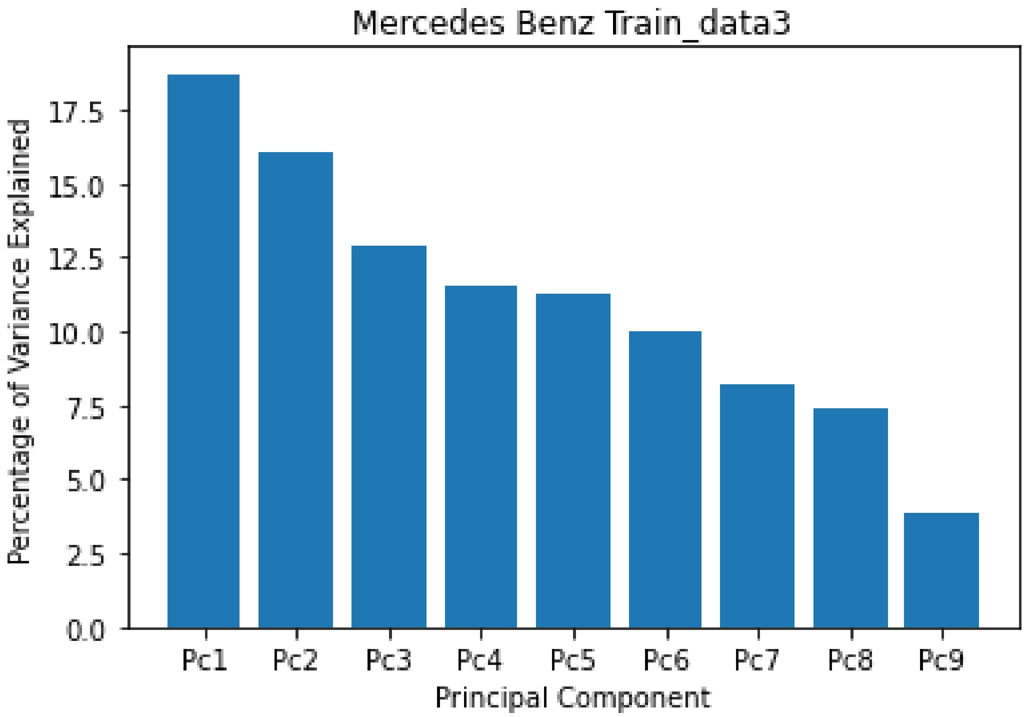
```
array([0.18721572, 0.34776494, 0.4772046 , 0.59197929, 0.70486192,
       0.80533883, 0.88750932, 0.96148016, 1.        ])
```

[42]:

```
per_var=np.round(pca.explained_variance_ratio_*100,decimals=1)
labels=['Pc' + str(y) for y in range(1,len(per_var)+1)]
```

[43]:

```
plt.bar(x=range(1, len(per_var)+1), height=per_var, tick_label=labels)
#cumulative = explained_variance.cumm()
#plt.plot(cumulative='green')
plt.ylabel("Percentage of Variance Explained")
plt.xlabel("Principal Component")
plt.title('Mercedes Benz Train_data3')
plt.show()
```



[ ]:

```
```

XGBOOST for Test data

[87]:

```
Test_data3
```

[87]:

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

[4209 rows x 9 columns]
```

[88]:

```
X=Test_data3.iloc[:,1:].values
```

[89]:

```
y=Test_data3.iloc[:,1].values
```

[90]:
```
X.shape
```

[90]:
```
(4209, 8)
```

[91]:
```
y.shape
```

[91]:
```
(4209,)
```

[92]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)
```

[93]:
```
from xgboost import XGBClassifier
```

[94]:
```
classifier=XGBClassifier()
```

[95]:
```
classifier.fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning:
The use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[18:45:18] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default
evaluation metric used with the objective 'multi:softprob' was changed from
'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
```

[95]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, objective='multi:softprob', predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=None,
              subsample=1, tree_method='exact', validate_parameters=1,
              verbosity=None)
```

[96]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
y_pred=classifier.predict(X_test)
cm=confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)
```

```
[[ 6  0  0 …  0  0  0]
 [ 0  1  0 …  0  0  0]
 [ 0  0  6 …  0  0  0]
 …
 [ 0  0  0 … 52  0  0]
 [ 0  0  0 …  0 57  0]
 [ 0  0  0 …  0  0 90]]
```

```
[96]: 0.994061757719715
```

```
[ ]:
```

```
[ ]:
```