swetha-pca

May 8, 2023

[2]:
```
#Importing required libraries
import numpy as np
import pandas as pd
```

[3]:
```
ctg_df=pd.read_csv("ctg_data1.csv")
```

[4]:
```
ctg_df.head()
```

[4]:
```
     b     e  AC  FM  UC  DL  DS  DP  DR   LB  …  C  D  E  AD  DE  LD  FS  \
0  240   357   0   0   0   0   0   0   0  120  … -1 -1 -1  -1  -1  -1   1
1    5   632   4   0   4   2   0   0   0  132  … -1 -1 -1   1  -1  -1  -1
2  177   779   2   0   5   2   0   0   0  133  … -1 -1 -1   1  -1  -1  -1
3  411  1192   2   0   6   2   0   0   0  134  … -1 -1 -1   1  -1  -1  -1
4  533  1147   4   0   5   0   0   0   0  132  … -1 -1 -1  -1  -1  -1  -1

   SUSP  CLASS  NSP
0    -1      9    2
1    -1      6    1
2    -1      6    1
3    -1      6    1
4    -1      2    1

[5 rows x 42 columns]
```

[5]:
```
ctg_df.dtypes
```

[5]:
```
b             int64
e             int64
AC            int64
FM            int64
UC            int64
DL            int64
DS            int64
DP            int64
DR            int64
LB            int64
AC.1        float64
```

```
FM.1          float64
UC.1          float64
DL.1          float64
DS.1          float64
DP.1          float64
ASTV            int64
MSTV          float64
ALTV            int64
MLTV          float64
Width           int64
Min             int64
Max             int64
Nmax            int64
Nzeros          int64
Mode            int64
Mean            int64
Median          int64
Variance        int64
Tendency        int64
A               int64
B               int64
C               int64
D               int64
E               int64
AD              int64
DE              int64
LD              int64
FS              int64
SUSP            int64
CLASS           int64
NSP             int64
dtype: object
```

[6]:
```
#Checking for null values
ctg_df.isna().sum()
```

[6]:
```
b           0
e           0
AC          0
FM          0
UC          0
DL          0
DS          0
DP          0
DR          0
LB          0
AC.1        0
```

```
FM.1        0
UC.1        0
DL.1        0
DS.1        0
DP.1        0
ASTV        0
MSTV        0
ALTV        0
MLTV        0
Width       0
Min         0
Max         0
Nmax        0
Nzeros      0
Mode        0
Mean        0
Median      0
Variance    0
Tendency    0
A           0
B           0
C           0
D           0
E           0
AD          0
DE          0
LD          0
FS          0
SUSP        0
CLASS       0
NSP         0
dtype: int64
```

[7]:
```
ctg_df.dropna()
```

[7]:
```
         b     e  AC  FM  UC  DL  DS  DP  DR   LB  …   C   D   E  AD  DE  LD  \
0      240   357   0   0   0   0   0   0   0  120  …  -1  -1  -1  -1  -1  -1
1        5   632   4   0   4   2   0   0   0  132  …  -1  -1  -1   1  -1  -1
2      177   779   2   0   5   2   0   0   0  133  …  -1  -1  -1   1  -1  -1
3      411  1192   2   0   6   2   0   0   0  134  …  -1  -1  -1   1  -1  -1
4      533  1147   4   0   5   0   0   0   0  132  …  -1  -1  -1  -1  -1  -1
…      …     …    ..  ..  ..  ..  ..  ..  …    …  .. .. ..  ..  ..  ..
2121  2059  2867   0   0   6   0   0   0   0  140  …  -1  -1   1  -1  -1  -1
2122  1576  2867   1   0   9   0   0   0   0  140  …  -1  -1   1  -1  -1  -1
2123  1576  2596   1   0   7   0   0   0   0  140  …  -1  -1   1  -1  -1  -1
2124  1576  3049   1   0   9   0   0   0   0  140  …  -1  -1   1  -1  -1  -1
2125  2796  3415   1   1   5   0   0   0   0  142  …  -1  -1  -1  -1  -1  -1
```

```
      FS  SUSP  CLASS  NSP
0      1    -1      9    2
1     -1    -1      6    1
2     -1    -1      6    1
3     -1    -1      6    1
4     -1    -1      2    1
…     ..   …     …    …
2121  -1    -1      5    2
2122  -1    -1      5    2
2123  -1    -1      5    2
2124  -1    -1      5    2
2125  -1    -1      1    1

[2126 rows x 42 columns]
```

[8]:
```
ctg_df.isna().sum()
```

[8]:
```
b           0
e           0
AC          0
FM          0
UC          0
DL          0
DS          0
DP          0
DR          0
LB          0
AC.1        0
FM.1        0
UC.1        0
DL.1        0
DS.1        0
DP.1        0
ASTV        0
MSTV        0
ALTV        0
MLTV        0
Width       0
Min         0
Max         0
Nmax        0
Nzeros      0
Mode        0
Mean        0
Median      0
Variance    0
```

```
Tendency    0
A           0
B           0
C           0
D           0
E           0
AD          0
DE          0
LD          0
FS          0
SUSP        0
CLASS       0
NSP         0
dtype: int64
```

[9]:
```
Features=ctg_df.drop('NSP', axis=1)
Label=ctg_df['NSP']
```

PCA

[10]:
```
# mean Centering the data
Features_meaned = Features - np.mean(Features , axis = 0)
Features_meaned
```

[10]:
```
                b            e        AC        FM        UC        DL  \
0     -638.439793 -1345.877234 -2.722484 -7.241298 -3.659925 -1.570085
1     -873.439793 -1070.877234  1.277516 -7.241298  0.340075  0.429915
2     -701.439793  -923.877234 -0.722484 -7.241298  1.340075  0.429915
3     -467.439793  -510.877234 -0.722484 -7.241298  2.340075  0.429915
4     -345.439793  -555.877234  1.277516 -7.241298  1.340075 -1.570085
…             …            …         …         …         …         …
2121  1180.560207  1164.122766 -2.722484 -7.241298  2.340075 -1.570085
2122   697.560207  1164.122766 -1.722484 -7.241298  5.340075 -1.570085
2123   697.560207   893.122766 -1.722484 -7.241298  3.340075 -1.570085
2124   697.560207  1346.122766 -1.722484 -7.241298  5.340075 -1.570085
2125  1917.560207  1712.122766 -1.722484 -6.241298  1.340075 -1.570085

            DS        DP   DR         LB  …         B         C         D  \
0    -0.003293 -0.126058  0.0 -13.303857  … -0.544685 -0.049859 -0.076199
1    -0.003293 -0.126058  0.0  -1.303857  … -0.544685 -0.049859 -0.076199
2    -0.003293 -0.126058  0.0  -0.303857  … -0.544685 -0.049859 -0.076199
3    -0.003293 -0.126058  0.0   0.696143  … -0.544685 -0.049859 -0.076199
4    -0.003293 -0.126058  0.0  -1.303857  …  1.455315 -0.049859 -0.076199
…          …         …  …         …  …         …         …         …
2121 -0.003293 -0.126058  0.0   6.696143  … -0.544685 -0.049859 -0.076199
2122 -0.003293 -0.126058  0.0   6.696143  … -0.544685 -0.049859 -0.076199
2123 -0.003293 -0.126058  0.0   6.696143  … -0.544685 -0.049859 -0.076199
```

```
2124 -0.003293 -0.126058  0.0    6.696143  … -0.544685 -0.049859 -0.076199
2125 -0.003293 -0.126058  0.0    8.696143  … -0.544685 -0.049859 -0.076199

             E        AD        DE        LD        FS      SUSP     CLASS
0    -0.067733 -0.312324 -0.237065 -0.100659  1.935089 -0.185325  4.490122
1    -0.067733  1.687676 -0.237065 -0.100659 -0.064911 -0.185325  1.490122
2    -0.067733  1.687676 -0.237065 -0.100659 -0.064911 -0.185325  1.490122
3    -0.067733  1.687676 -0.237065 -0.100659 -0.064911 -0.185325  1.490122
4    -0.067733 -0.312324 -0.237065 -0.100659 -0.064911 -0.185325 -2.509878
…          …        …        …        …        …        …        …
2121  1.932267 -0.312324 -0.237065 -0.100659 -0.064911 -0.185325  0.490122
2122  1.932267 -0.312324 -0.237065 -0.100659 -0.064911 -0.185325  0.490122
2123  1.932267 -0.312324 -0.237065 -0.100659 -0.064911 -0.185325  0.490122
2124  1.932267 -0.312324 -0.237065 -0.100659 -0.064911 -0.185325  0.490122
2125 -0.067733 -0.312324 -0.237065 -0.100659 -0.064911 -0.185325 -3.509878

[2126 rows x 41 columns]
```

[11]:
```python
# Calculate the co-variance matrix of the mean-centered data.
cov_matrix = np.cov(Features_meaned , rowvar = False)
```

[12]:
```python
#Calculating Eigenvalues and Eigenvectors of the covariance matrix
eigen_values , eigen_vectors = np.linalg.eigh(cov_matrix)
```

[13]:
```python
#sort the eigenvalues in descending order
sorted_index = np.argsort(eigen_values)[::-1]

sorted_eigenvalue = eigen_values[sorted_index]
#similarly sort the eigenvectors
sorted_eigenvectors = eigen_vectors[:,sorted_index]
sorted_eigenvectors
```

[13]:
```
array([[ 6.91839404e-01,  7.21512407e-01,  1.75656916e-02, …,
          0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 7.22033857e-01, -6.91484578e-01, -1.52148770e-02, …,
          2.42108531e-16, -1.19944670e-16,  9.70778671e-16],
       [ 5.35372127e-05, -5.53091262e-03,  1.35371709e-02, …,
          3.08198656e-12, -4.15940569e-12, -7.86838897e-11],
       …,
       [-3.88395017e-05,  1.12232968e-04, -1.40543871e-03, …,
         -4.51674774e-01,  1.90457665e-01, -2.39697854e-03],
       [-7.85665562e-05,  4.22719070e-05, -3.33119538e-03, …,
         -5.54477252e-01,  1.58982377e-01, -2.16854955e-03],
       [-1.95401290e-04, -1.98217218e-05,  9.55309887e-03, …,
          2.05604956e-01,  6.29505765e-02, -4.56858038e-04]])
```

[14]:
```python
# select the first n eigenvectors, n is desired dimension
# of our final reduced data.

n_components = 30 #you can select any number of components.
eigenvector_subset = sorted_eigenvectors[:,0:n_components]
eigenvector_subset
```

[14]:
```
array([[ 6.91839404e-01,  7.21512407e-01,  1.75656916e-02, …,
        -3.54733462e-05, -3.58086842e-05, -1.13214907e-04],
       [ 7.22033857e-01, -6.91484578e-01, -1.52148770e-02, …,
         7.41245718e-05,  5.58820374e-05,  9.34508112e-05],
       [ 5.35372127e-05, -5.53091262e-03,  1.35371709e-02, …,
         3.87838214e-03,  1.41788419e-02, -5.82796889e-03],
       …,
       [-3.88395017e-05,  1.12232968e-04, -1.40543871e-03, …,
         3.68522925e-01,  3.75361381e-01, -2.92472624e-01],
       [-7.85665562e-05,  4.22719070e-05, -3.33119538e-03, …,
         1.43153772e-02,  3.86644752e-02, -2.58845884e-01],
       [-1.95401290e-04, -1.98217218e-05,  9.55309887e-03, …,
         3.88068380e-02,  6.43857288e-02,  4.79739804e-02]])
```

[15]:
```python
#Transform the data
Features_reduced = np.dot(eigenvector_subset.transpose(),Features_meaned.
 ↪transpose()).transpose()
Features_reduced
```

[15]:
```
array([[-1.41343472e+03,  4.70134580e+02,  2.84244658e+01, …,
         4.15078967e-01,  9.06630922e-01, -2.25026432e-01],
       [-1.37755124e+03,  1.08605023e+02,  5.95305431e+01, …,
        -4.15074824e-02, -4.22331452e-02,  1.84028598e-01],
       [-1.15241176e+03,  1.31088223e+02,  6.10420703e+01, …,
         4.02103965e-02, -1.72973592e-01,  1.41992917e-01],
       …,
       [ 1.12739797e+03, -1.14737308e+02, -2.45558023e+01, …,
         3.12989095e-01, -3.28107836e-03, -1.13333542e-01],
       [ 1.45447463e+03, -4.27942448e+02, -3.29994207e+01, …,
         3.70680658e-01, -4.59279370e-02, -8.61247641e-02],
       [ 2.56282951e+03,  2.00210681e+02, -4.24806990e+01, …,
        -5.89768074e-02, -8.03310712e-02, -2.10165288e-02]])
```

[16]:
```python
PCA_df = pd.DataFrame(Features_reduced)
PCA_df
```

[16]:
```
             0           1          2          3          4          5  \
0 -1413.434724  470.134580  28.424466  13.879196  40.820047 -46.972007
1 -1377.551238  108.605023  59.530543  25.883117 -29.940244  28.647525
2 -1152.411762  131.088223  61.042070  25.135022 -29.413593  28.519969
```

3	-692.298332	14.835719	53.288634	22.533661	-9.411187	36.527373
4	-640.389127	133.974232	54.899883	22.053800	-12.337565	38.486316
…	…	…	…	…	…	…
2121	1657.277161	47.326853	-55.520818	-10.339265	-29.489633	-20.718990
2122	1323.070040	-302.087589	-29.906180	0.127539	-21.426965	-5.432812
2123	1127.397969	-114.737308	-24.555802	0.747021	-25.360678	-6.057273
2124	1454.474628	-427.942448	-32.999421	-0.257821	-19.233663	-7.715140
2125	2562.829512	200.210681	-42.480699	-12.576243	-14.140811	-11.918309

	6	7	8	9	…	20	21 \
0	-34.539355	-31.888888	-24.626308	8.053120	…	0.093018	-0.810424
1	6.920000	4.199390	26.167438	0.827025	…	0.922525	0.144430
2	6.723026	3.926497	26.920672	-1.677391	…	1.043045	0.109449
3	2.915169	-15.857135	8.636404	-10.282103	…	0.228416	-0.304752
4	0.657737	-14.152976	8.526533	-6.546051	…	-0.246582	-0.227238
…	…	…	…	…	…	…	…
2121	-17.590565	22.660287	2.243139	3.793377	…	-0.016704	-0.093432
2122	-24.794924	8.820563	-7.776421	1.622849	…	-0.054152	-0.235634
2123	-26.694728	12.285514	-9.679601	3.296160	…	-0.014438	-0.190680
2124	-25.851610	5.120339	-4.884823	1.412911	…	-0.044336	-0.258925
2125	-35.517205	7.239103	0.122317	0.780335	…	0.273635	0.255655

	22	23	24	25	26	27	28 \
0	-0.721120	0.299470	-0.078584	-0.709594	0.296404	0.415079	0.906631
1	-0.304961	-0.209206	0.042727	-0.135705	0.117000	-0.041507	-0.042233
2	-0.288802	-0.171645	0.007562	-0.168075	0.059803	0.040210	-0.172974
3	-0.283725	-0.168691	-0.076990	-0.273580	0.210667	0.035501	-0.375974
4	0.363436	0.132460	-0.147433	-0.122192	0.279740	0.220051	-0.021162
…	…	…	…	…	…	…	…
2121	-0.921338	0.694898	0.318349	1.329368	0.459549	0.166742	-0.134555
2122	-0.874839	0.691758	0.068500	1.378358	0.377367	0.364705	-0.015705
2123	-0.853806	0.710446	0.062186	1.426263	0.441196	0.312989	-0.003281
2124	-0.911995	0.655145	0.062835	1.390439	0.316014	0.370681	-0.045928
2125	0.158093	-0.197516	0.316936	0.094122	-0.071173	-0.058977	-0.080331

	29
0	-0.225026
1	0.184029
2	0.141993
3	0.123455
4	-0.088130
…	…
2121	-0.086301
2122	-0.105195
2123	-0.113334
2124	-0.086125
2125	-0.021017

```
[2126 rows x 30 columns]
```

[17]:
```
from sklearn.model_selection import train_test_split # Import train_test_split␣
 ↪function
from sklearn import metrics
```

[18]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(PCA_df, Label, test_size=0.
 ↪3, random_state=1)
```

DECISION TREE

[19]:
```
# Import Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred_train = clf.predict(X_train)
```

[20]:
```
print("Decision Tree Model Accuracy with training data (in %):",metrics.
 ↪accuracy_score(y_train, y_pred_train)*100)
```

```
Decision Tree Model Accuracy with training data (in %): 99.93279569892472
```

[21]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=8)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

[22]:
```
print("Decision Tree model accuracy(in %):",metrics.accuracy_score(y_test,␣
 ↪y_pred)*100)
```

```
Decision Tree model accuracy(in %): 95.7680250783699
```

Naive Bayes

[23]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

```
y_pred_train = gnb.predict(X_train)

print('Gaussian Naive Bayes Training-set accuracy(in %):', metrics.
↪accuracy_score(y_train, y_pred_train)*100)
```

```
Gaussian Naive Bayes Training-set accuracy(in %): 95.83333333333334
```

[24]:
```
# making predictions on the testing set
y_pred = gnb.predict(X_test)

# comparing actual response values (y_test) with predicted response values
print("Gaussian Naive Bayes model accuracy(in %):", metrics.
↪accuracy_score(y_test, y_pred)*100)
```

```
Gaussian Naive Bayes model accuracy(in %): 95.61128526645768
```

Random Forest

[25]:
```
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
```

[26]:
```
# creating a RF classifier
rfclf = RandomForestClassifier(n_estimators = 100)

# Training the model on the training dataset
rfclf.fit(X_train, y_train)
y_pred_train = rfclf.predict(X_train)

print('Training-set accuracy(in %):', metrics.accuracy_score(y_train,␣
↪y_pred_train)*100)
```

```
Training-set accuracy(in %): 99.93279569892472
```

[27]:
```
# performing predictions on the test dataset
y_pred = rfclf.predict(X_test)

# using metrics module for accuracy calculation
print("Random Forest model accuracy(in %): ", metrics.accuracy_score(y_test,␣
↪y_pred)*100)
```

```
Random Forest model accuracy(in %):  98.43260188087774
```

SVM

[28]:
```
#Import svm model
from sklearn import svm

#Create a svm Classifier
```

```
svmclf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
svmclf.fit(X_train, y_train)

y_pred_train = svmclf.predict(X_train)

print('Training-set accuracy(in %):', metrics.accuracy_score(y_train,␣
↪y_pred_train)*100)
```

```
Training-set accuracy(in %): 99.32795698924731
```

[29]:

```
#Predict the response for test dataset
y_pred = svmclf.predict(X_test)

# using metrics module for accuracy calculation
print("SVM model accuracy(in %): ", metrics.accuracy_score(y_test, y_pred)*100)
```

```
SVM model accuracy(in %):  98.90282131661442
```

[ ]:

```

```