Problem_1

October 24, 2022

[16]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.naive_bayes import GaussianNB
import seaborn as sns
from sklearn.datasets import load_breast_cancer
```

[17]:
```
breast = load_breast_cancer()
breast_data = breast.data
breast_data.shape
```

[17]:
```
(569, 30)
```

[18]:
```
breast_input = pd.DataFrame(breast_data)
breast_input.head()
```

[18]:
```
       0      1       2       3        4        5       6        7       8   \
0  17.99  10.38  122.80  1001.0  0.11840  0.27760  0.3001  0.14710  0.2419
1  20.57  17.77  132.90  1326.0  0.08474  0.07864  0.0869  0.07017  0.1812
2  19.69  21.25  130.00  1203.0  0.10960  0.15990  0.1974  0.12790  0.2069
3  11.42  20.38   77.58   386.1  0.14250  0.28390  0.2414  0.10520  0.2597
4  20.29  14.34  135.10  1297.0  0.10030  0.13280  0.1980  0.10430  0.1809

         9  …     20     21      22      23      24      25      26      27  \
0  0.07871  …  25.38  17.33  184.60  2019.0  0.1622  0.6656  0.7119  0.2654
1  0.05667  …  24.99  23.41  158.80  1956.0  0.1238  0.1866  0.2416  0.1860
2  0.05999  …  23.57  25.53  152.50  1709.0  0.1444  0.4245  0.4504  0.2430
3  0.09744  …  14.91  26.50   98.87   567.7  0.2098  0.8663  0.6869  0.2575
4  0.05883  …  22.54  16.67  152.20  1575.0  0.1374  0.2050  0.4000  0.1625

       28       29
0  0.4601  0.11890
```

```
1  0.2750  0.08902
2  0.3613  0.08758
3  0.6638  0.17300
4  0.2364  0.07678

[5 rows x 30 columns]
```

[19]:
```
breast_labels = breast.target
breast_labels.shape
```

[19]:
```
(569,)
```

[20]:
```
labels = np.reshape(breast_labels,(569,1))
```

[21]:
```
final_breast_data = np.concatenate([breast_data,labels],axis=1)
final_breast_data.shape
```

[21]:
```
(569, 31)
```

[22]:
```
breast_dataset = pd.DataFrame(final_breast_data)
features = breast.feature_names
features
```

[22]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

[23]:
```
features_labels = np.append(features,'label')
breast_dataset.columns = features_labels
breast_dataset.head()
```

[23]:
```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
```

```
2          0.15990          0.1974               0.12790          0.2069
3          0.28390          0.2414               0.10520          0.2597
4          0.13280          0.1980               0.10430          0.1809

   mean fractal dimension  …  worst texture  worst perimeter  worst area  \
0                 0.07871  …          17.33           184.60      2019.0
1                 0.05667  …          23.41           158.80      1956.0
2                 0.05999  …          25.53           152.50      1709.0
3                 0.09744  …          26.50            98.87       567.7
4                 0.05883  …          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   worst symmetry  worst fractal dimension  label
0          0.4601                  0.11890    0.0
1          0.2750                  0.08902    0.0
2          0.3613                  0.08758    0.0
3          0.6638                  0.17300    0.0
4          0.2364                  0.07678    0.0

[5 rows x 31 columns]
```

[24]:
```python
breast_dataset.shape
```

[24]:
```
(569, 31)
```

[25]:
```python
X = breast_dataset.values[:,0:30]
print('X =', X[0:5])
```

```
X = [[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
```

```
 3.613e-01 8.758e-02]
[1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
 1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
 9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
 2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
 6.638e-01 1.730e-01]
[2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
 1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
 1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
 1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
 2.364e-01 7.678e-02]]
```

[26]:
```
Y = breast_dataset.values[:,30]
print('Y =', Y[0:5])
```

```
Y = [0. 0. 0. 0. 0.]
```

[27]:
```
# Splitting the datasets to training and validation sets.

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state = 0)
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(455, 30) (114, 30) (455,) (114,)
```

[28]:
```
# Feature scaling between 0 and 1 for independent variables using␣
 ↪Standardization.

from sklearn.preprocessing import StandardScaler, MinMaxScaler
sc_X = StandardScaler()
X_train_sc = sc_X.fit_transform(X_train)
X_test_sc = sc_X.fit_transform(X_test)
```

[29]:
```
print('New X_train =', X_train_sc[0:5])
```

```
New X_train = [[-1.15036482 -0.39064196 -1.12855021 -0.95876358  0.3109837
-0.5959945
  -0.80259612 -0.80249002  0.29453906  0.0942515  -0.4950523   1.48720153
  -0.51448782 -0.49154005  0.28149837 -0.60451206 -0.46900701 -0.61170002
   0.05798237 -0.35763702 -1.0431756   0.21353282 -1.0360446  -0.84880771
   0.34249851 -0.73009743 -0.81232053 -0.75798367 -0.01614761 -0.38503402]
 [-0.93798972  0.68051405 -0.94820146 -0.82152548 -0.60963604 -0.90986721
  -0.66066905 -0.89871612  0.75493453 -0.42547082 -0.33381757  0.75941203
  -0.28751805 -0.42127695 -0.1620797  -0.20486693 -0.05029632 -0.20309076
  -0.25469005 -0.39139463 -0.71565415  1.06684183 -0.68992205 -0.66869703
  -0.09553745 -0.53786647 -0.37504806 -0.60687023  0.09669004 -0.38615797]
 [ 0.574121   -1.03333557  0.51394098  0.40858627 -0.10616078 -0.36301886
  -0.41799048 -0.08844569 -0.27182044 -0.57522132 -0.57672579 -1.05784511
  -0.53856037 -0.38708923 -1.07211882 -0.72057496 -0.42362791 -0.49218988
```

```
 -0.67484362 -0.80147288  0.29761532 -0.97781783  0.26213665  0.11388819
 -0.52472419 -0.52086645 -0.18298917 -0.02371948 -0.20050207 -0.75144254]
[-0.54721953 -0.3160221  -0.57762185 -0.5666148   0.5866618  -0.64933105
 -0.80529827 -0.50006514  0.33107838  0.54056672 -0.12822568  0.55622207
 -0.20400103 -0.33234693 -0.55285085 -0.75888143 -0.64891421  0.60156561
  0.20454757 -0.11596321 -0.70132509 -0.75792666 -0.73573673 -0.65896593
 -0.81674816 -1.03492082 -1.09163333 -0.85254451 -1.07618575 -0.54688318]
[-0.52739786  0.79124029 -0.5615634  -0.52357067 -1.05144646 -1.0175317
 -0.90514905 -0.93580596 -0.9697215  -0.42693897 -0.62882784 -0.13092944
 -0.61323441 -0.46658092 -0.67149038 -0.74401623 -0.71006335 -1.20449751
 -0.54293494 -0.50302491 -0.42702588  1.05863694 -0.42242341 -0.44095517
 -0.30349391 -0.46725101 -0.72456516 -0.78311815  0.31124049 -0.08212882]]
```

[30]:
```python
# Construct the logistic regression's report and confusion matrix
model = LogisticRegression(solver = 'liblinear')
model.fit(X_train_sc, Y_train)
predicted = model.predict(X_test_sc)
matrix = confusion_matrix(Y_test, predicted)
report = classification_report(Y_test, predicted)
print("Confusion Matrix: \n",matrix)
print("\n")
print("Classification Report: \n",report)
```

```
Confusion Matrix:
 [[44  3]
 [ 2 65]]


Classification Report:
               precision    recall  f1-score   support

         0.0       0.96      0.94      0.95        47
         1.0       0.96      0.97      0.96        67

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

[31]:
```python
# Constructing the Naive Bayes's report and confusion matrix.
model = GaussianNB()
model.fit(X_train_sc, Y_train)
predicted = model.predict(X_test_sc)
print("Confusion Matrix: \n",metrics.confusion_matrix(Y_test, predicted))
print("\n")
print("Classification Report: \n",metrics.classification_report(Y_test,
 ↪predicted))
```

```
Confusion Matrix:
```

```
 [[43  4]
 [ 3 64]]


Classification Report:
              precision    recall  f1-score   support

         0.0       0.93      0.91      0.92        47
         1.0       0.94      0.96      0.95        67

    accuracy                           0.94       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.94      0.94      0.94       114
```

Problem_2_3

October 24, 2022

[118]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.datasets import load_breast_cancer
```

[119]:
```
breast = load_breast_cancer()
breast_data = breast.data
breast_data.shape
```

[119]:
```
(569, 30)
```

[120]:
```
breast_input = pd.DataFrame(breast_data)
breast_input.head()
```

[120]:
```
       0      1       2       3        4        5       6        7       8  \
0  17.99  10.38  122.80  1001.0  0.11840  0.27760  0.3001  0.14710  0.2419
1  20.57  17.77  132.90  1326.0  0.08474  0.07864  0.0869  0.07017  0.1812
2  19.69  21.25  130.00  1203.0  0.10960  0.15990  0.1974  0.12790  0.2069
3  11.42  20.38   77.58   386.1  0.14250  0.28390  0.2414  0.10520  0.2597
4  20.29  14.34  135.10  1297.0  0.10030  0.13280  0.1980  0.10430  0.1809

         9  …     20     21      22      23      24      25      26      27  \
0  0.07871  …  25.38  17.33  184.60  2019.0  0.1622  0.6656  0.7119  0.2654
1  0.05667  …  24.99  23.41  158.80  1956.0  0.1238  0.1866  0.2416  0.1860
2  0.05999  …  23.57  25.53  152.50  1709.0  0.1444  0.4245  0.4504  0.2430
3  0.09744  …  14.91  26.50   98.87   567.7  0.2098  0.8663  0.6869  0.2575
4  0.05883  …  22.54  16.67  152.20  1575.0  0.1374  0.2050  0.4000  0.1625
```

```
       28       29
0  0.4601  0.11890
1  0.2750  0.08902
2  0.3613  0.08758
3  0.6638  0.17300
4  0.2364  0.07678

[5 rows x 30 columns]
```

[121]:
```
breast_labels = breast.target
breast_labels.shape
```

[121]:
```
(569,)
```

[122]:
```
labels = np.reshape(breast_labels,(569,1))
```

[123]:
```
final_breast_data = np.concatenate([breast_data,labels],axis=1)
final_breast_data.shape
```

[123]:
```
(569, 31)
```

[124]:
```
breast_dataset = pd.DataFrame(final_breast_data)
features = breast.feature_names
features
```

[124]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

[125]:
```
features_labels = np.append(features,'label')
breast_dataset.columns = features_labels
breast_dataset.head()
```

[125]:
```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
```

```
0           0.27760          0.3001            0.14710         0.2419
1           0.07864          0.0869            0.07017         0.1812
2           0.15990          0.1974            0.12790         0.2069
3           0.28390          0.2414            0.10520         0.2597
4           0.13280          0.1980            0.10430         0.1809

   mean fractal dimension  …  worst texture  worst perimeter  worst area  \
0                 0.07871  …          17.33           184.60      2019.0
1                 0.05667  …          23.41           158.80      1956.0
2                 0.05999  …          25.53           152.50      1709.0
3                 0.09744  …          26.50            98.87       567.7
4                 0.05883  …          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   worst symmetry  worst fractal dimension  label
0          0.4601                  0.11890    0.0
1          0.2750                  0.08902    0.0
2          0.3613                  0.08758    0.0
3          0.6638                  0.17300    0.0
4          0.2364                  0.07678    0.0

[5 rows x 31 columns]
```

[126]:
```
breast_dataset.shape
```

[126]:
```
(569, 31)
```

[127]:
```
X = breast_dataset.values[:,0:30]
print('X =', X[0:5])
```

```
X = [[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
```

```
 6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
 2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
 3.613e-01 8.758e-02]
[1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
 1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
 9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
 2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
 6.638e-01 1.730e-01]
[2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
 1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
 1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
 1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
 2.364e-01 7.678e-02]]
```

[128]:
```
Y = breast_dataset.values[:,30]
print('Y =', Y[0:5])
```

```
Y = [0. 0. 0. 0. 0.]
```

[129]:
```
# Using PCA feature extraction to simplify the features.
pca = PCA(n_components = 10)
principalComponents = pca.fit_transform(X)
principalDF = pd.DataFrame(data = principalComponents, columns =␣
 ↪['1','2','3','4','5','6','7','8','9','10'])
#,'11','12','13','14','15'])
principalDF.head()
```

[129]:
```
             1           2          3          4          5         6  \
0  1160.142574 -293.917544  48.578398  -8.711975  32.000486  1.265415
1  1269.122443   15.630182 -35.394534  17.861283  -4.334874 -0.225872
2   995.793889   39.156743  -1.709753   4.199340  -0.466529 -2.652811
3  -407.180803  -67.380320   8.672848 -11.759867   7.115461  1.299436
4   930.341180  189.340742   1.374801   8.499183   7.613289  1.021160

          7         8         9        10
0  0.931337  0.148167  0.745463  0.589359
1 -0.046037  0.200804 -0.485828 -0.084035
2 -0.779745 -0.274026 -0.173874 -0.186994
3 -1.267304 -0.060555 -0.330639 -0.144155
4 -0.335522  0.289109  0.036087 -0.138502
```

[130]:
```
# Splitting the datasets to training and validation sets.

X_train, X_test, Y_train, Y_test = train_test_split(principalDF, Y, test_size=0.
 ↪2, random_state = 0)
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(455, 10) (114, 10) (455,) (114,)
```

[131]:
```python
# Feature scaling between 0 and 1 for independent variables using␣
↪Standardization.

from sklearn.preprocessing import StandardScaler, MinMaxScaler
sc_X = StandardScaler()
X_train_sc = sc_X.fit_transform(X_train)
X_test_sc = sc_X.fit_transform(X_test)
```

[132]:
```python
print('New X_train =', X_train_sc[0:5])
```

```
New X_train = [[-0.88568257 -0.40583219  0.39750989  0.33360589 -0.82403734
0.79717288
   1.36486082  0.93326731  0.37094003  0.63542118]
 [-0.71552247 -0.55110359  0.2621112  -0.81275164 -0.99567678  1.61036554
   0.43205142 -0.11031695 -0.69576898  0.11803613]
 [ 0.19358313  1.02938885 -0.94593214  0.69047135  0.91199831 -0.05307128
   0.21796358 -0.46967881 -0.37321323  0.21184242]
 [-0.63876355  0.30664196  0.30820425  0.44447528  0.05508024 -0.90020891
  -0.73136923  0.77187283 -0.15680726  0.40727644]
 [-0.46747526 -0.30053312 -0.16289155 -0.7795932  -0.91315631  0.80585734
   0.37304117 -1.08723135 -0.39521306 -0.06307839]]
```

[133]:
```python
# Problem #2
# Construct the logistic regression's report and confusion matrix
model = LogisticRegression(solver = 'liblinear')
model.fit(X_train_sc, Y_train)
predicted = model.predict(X_test_sc)
matrix = confusion_matrix(Y_test, predicted)
report = classification_report(Y_test, predicted)
print("Confusion Matrix: \n",matrix)
print("\n")
print("Classification Report: \n",report)
```

```
Confusion Matrix:
 [[44  3]
 [ 2 65]]


Classification Report:
               precision    recall  f1-score   support

         0.0       0.96      0.94      0.95        47
         1.0       0.96      0.97      0.96        67

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

[134]:
```
# Problem #3
# Constructing the Naive Bayes's report and confusion matrix.
model = GaussianNB()
model.fit(X_train_sc, Y_train)
predicted = model.predict(X_test_sc)
print("Confusion Matrix: \n", metrics.confusion_matrix(Y_test, predicted))
print("\n")
print("Classification Report: \n", metrics.classification_report(Y_test,␣
 ↪predicted))
```

```
Confusion Matrix:
 [[42  5]
 [10 57]]


Classification Report:
               precision    recall  f1-score   support

         0.0       0.81      0.89      0.85        47
         1.0       0.92      0.85      0.88        67

    accuracy                           0.87       114
   macro avg       0.86      0.87      0.87       114
weighted avg       0.87      0.87      0.87       114
```