ECGR_4105_HW2

October 9, 2022

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import classification_report
from sklearn.datasets import load_breast_cancer
```

[2]:
```
diabset = pd.read_csv(r'C:\Users\homer\OneDrive\Documents\School␣
↪Folder\diabetes.csv')
```

[3]:
```
diabset.head()
```

[3]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[4]:
```
diabset.shape
```

[4]: (768, 9)

[5]:
```
diab_index = diabset.index.values
diab_index.shape
```

[5]: (768,)

[6]:
```
diab_labels = np.reshape(diab_index,(768,1))
```

[7]:
```
diab_data = np.concatenate([diabset,diab_labels],axis=1)
```

[8]:
```
diab_data.shape
```

[8]: (768, 10)

[9]:
```
diab_dataset = pd.DataFrame(diab_data)
```

[10]:
```
diab_dataset.head()
```

[10]:

	0	1	2	3	4	5	6	7	8	9
0	6.0	148.0	72.0	35.0	0.0	33.6	0.627	50.0	1.0	0.0
1	1.0	85.0	66.0	29.0	0.0	26.6	0.351	31.0	0.0	1.0
2	8.0	183.0	64.0	0.0	0.0	23.3	0.672	32.0	1.0	2.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0.0	3.0
4	0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0	1.0	4.0

[11]:
```
diab_X = diab_dataset.values[:, 9]
diab_Y = diab_dataset.values[:, 8]
```

[12]:
```
diab_X_train, diab_X_test, diab_Y_train, diab_Y_test = train_test_split(diab_X,␣
↪diab_Y, test_size=0.2, random_state=42)
```

[13]:
```
sc = StandardScaler()
diab_X_reshape = diab_X_train.reshape(-1, 1)
diab_X_std = sc.fit_transform(diab_X_reshape)
diab_Xtest_reshape = diab_X_test.reshape(-1, 1)
diab_Xtest_std = sc.transform(diab_Xtest_reshape)
```

[14]:
```
diab_logreg = LogisticRegression(solver ='liblinear', random_state=0)
diab_logreg.fit(diab_X_std, diab_Y_train)
```

[14]: LogisticRegression(random_state=0, solver='liblinear')

[15]:
```
diab_Y_pred = diab_logreg.predict(diab_Xtest_std)
```

[16]:
```
diab_cnf_matrix = confusion_matrix(diab_Y_test, diab_Y_pred)
diab_cnf_matrix
```

[16]:
```
array([[99,  0],
       [55,  0]], dtype=int64)
```

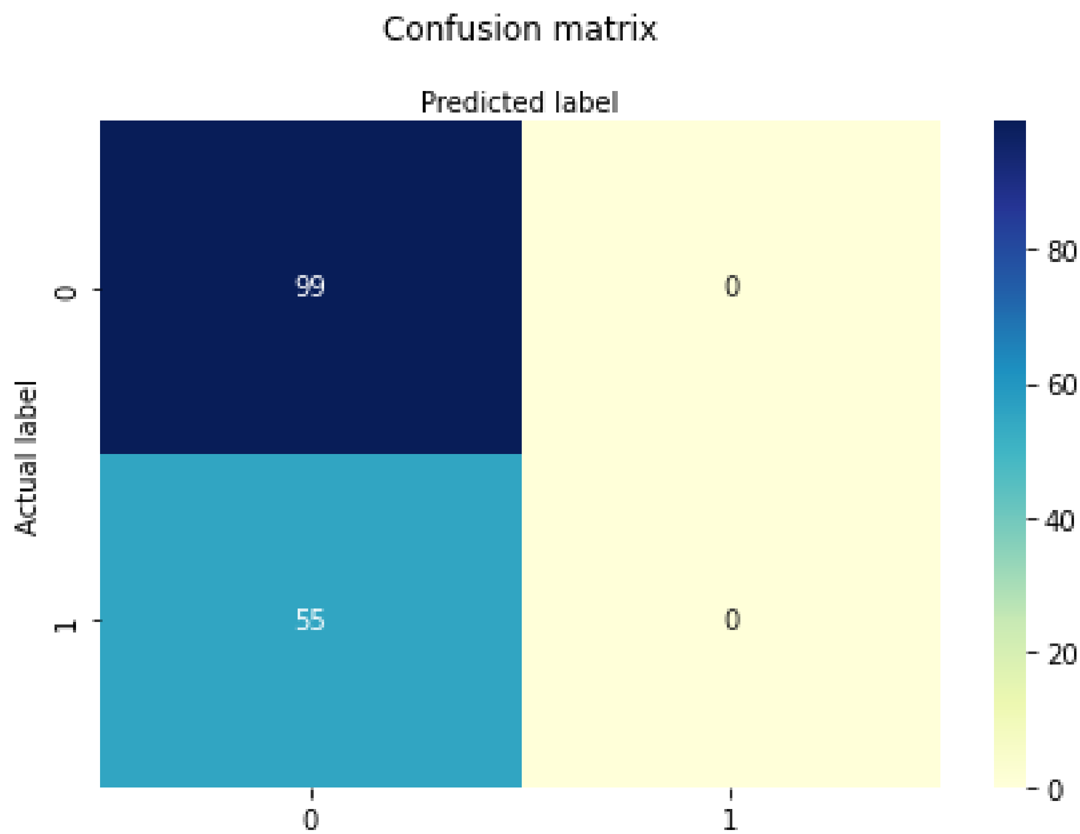
[17]:
```
print("Accuracy:",metrics.accuracy_score(diab_Y_test, diab_Y_pred))
print("Precision:",metrics.precision_score(diab_Y_test, diab_Y_pred))
print("Recall:",metrics.recall_score(diab_Y_test, diab_Y_pred))
```

```
Accuracy: 0.6428571428571429
Precision: 0.0
Recall: 0.0
```

```
C:\Users\homer\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[18]:
```
class_names = [0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(diab_cnf_matrix), annot=True, cmap="YlGnBu", fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

[18]:
```
Text(0.5, 257.44, 'Predicted label')
```



[19]:
```
diab_kfold = KFold(n_splits=5, random_state=42, shuffle=True)
model = LogisticRegression(solver='liblinear')
results = cross_val_score(model, diab_X.reshape(-1, 1), diab_Y, cv=diab_kfold)
print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

```
Accuracy: 65.106% (3.703%)
```

[20]:
```
diab_kfold = KFold(n_splits=10, random_state=42, shuffle=True)
model = LogisticRegression(solver='liblinear')
results = cross_val_score(model, diab_X.reshape(-1, 1), diab_Y, cv=diab_kfold)
print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

```
Accuracy: 65.096% (4.779%)
```

[21]:
```
cancer = load_breast_cancer()
```

[22]:
```
cancer_data = cancer.data
cancer_data.shape
```

[22]:
```
(569, 30)
```

[23]:
```
cancer_input = pd.DataFrame(cancer_data)
cancer_input.head()
```

[23]:
```
       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]
```

[24]:
```
cancer_labels = cancer.target
```

[25]:
```
cancer_labels.shape
```

[25]: (569,)

[26]:
```
labels = np.reshape(cancer_labels, (569,1))
```

[27]:
```
final_cancer_data = np.concatenate([cancer_data,labels],axis=1)
```

[28]:
```
final_cancer_data.shape
```

[28]: (569, 31)

[29]:
```
cancer_dataset = pd.DataFrame(final_cancer_data)
```

[30]:
```
features = cancer.feature_names
features
```

[30]:
```
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')
```

[31]:
```
features_labels = np.append(features, 'label')
```

[32]:
```
cancer_dataset.columns = features_labels
```

[33]:
```
cancer_dataset.head()
```

[33]:
```
   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]
```

[34]:
```python
cancer_dataset['label'].replace(0, 'Benign', inplace=True)
cancer_dataset['label'].replace(1, 'Malignant', inplace=True)
```

[35]:
```python
cancer_dataset.tail()
```

[35]:
```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
564        21.56         22.39          142.00     1479.0          0.11100
565        20.13         28.25          131.20     1261.0          0.09780
566        16.60         28.08          108.30      858.1          0.08455
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263

     mean compactness  mean concavity  mean concave points  mean symmetry  \
564           0.11590         0.24390              0.13890         0.1726
565           0.10340         0.14400              0.09791         0.1752
566           0.10230         0.09251              0.05302         0.1590
567           0.27700         0.35140              0.15200         0.2397
568           0.04362         0.00000              0.00000         0.1587

     mean fractal dimension  …  worst texture  worst perimeter  worst area  \
564                 0.05623  …          26.40           166.10      2027.0
565                 0.05533  …          38.25           155.00      1731.0
566                 0.05648  …          34.12           126.70      1124.0
567                 0.07016  …          39.42           184.60      1821.0
568                 0.05884  …          30.37            59.16       268.6

     worst smoothness  worst compactness  worst concavity  \
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000

     worst concave points  worst symmetry  worst fractal dimension      label
564                0.2216          0.2060                  0.07115     Benign
565                0.1628          0.2572                  0.06637     Benign
566                0.1418          0.2218                  0.07820     Benign
567                0.2650          0.4087                  0.12400     Benign
568                0.0000          0.2871                  0.07039  Malignant

[5 rows x 31 columns]
```

[36]:
```
cancer_X = cancer_dataset.iloc[:,0:29].values
cancer_Y = cancer_dataset.iloc[:,30].values
```

[37]:
```
cancer_X_train, cancer_X_test, cancer_Y_train, cancer_Y_test =␣
↪train_test_split(cancer_X, cancer_Y, test_size=0.2, random_state=42)
```

[38]:
```
sc_X = StandardScaler()
cancer_X_trainstd = sc_X.fit_transform(cancer_X_train)
cancer_X_teststd = sc_X.transform(cancer_X_test)
```

[39]:
```
cancerClass = LogisticRegression(random_state=42)
cancerClass.fit(cancer_X_trainstd, cancer_Y_train)
```

[39]:
```
LogisticRegression(random_state=42)
```

[40]:
```
cancer_Y_pred = cancerClass.predict(cancer_X_teststd)
```

[41]:
```
cancer_Y_pred[0:9]
```

[41]:
```
array(['Malignant', 'Benign', 'Benign', 'Malignant', 'Malignant',
       'Benign', 'Benign', 'Benign', 'Malignant'], dtype=object)
```

[42]:
```
cancer_cnf_matrix = confusion_matrix(cancer_Y_test, cancer_Y_pred)
cancer_cnf_matrix
```

[42]:
```
array([[42,  1],
       [ 1, 70]], dtype=int64)
```

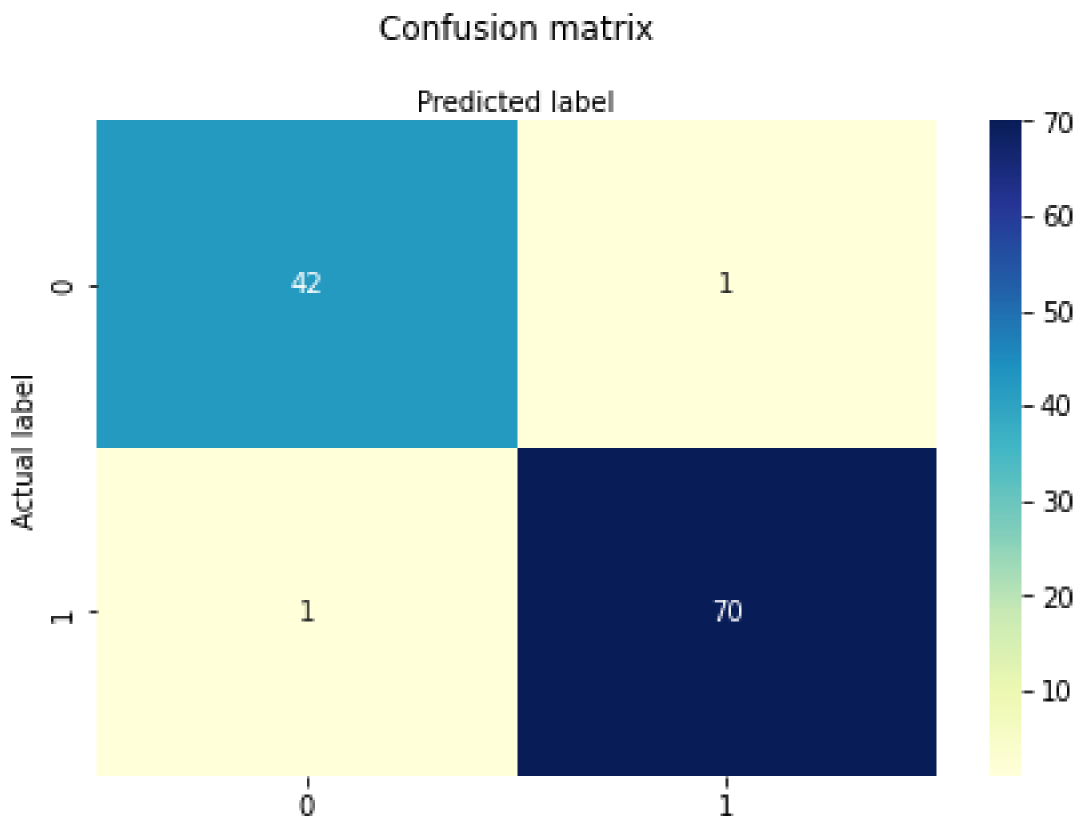
[43]:
```
print("Accuracy:",metrics.accuracy_score(cancer_Y_test, cancer_Y_pred))
print("Precision:",metrics.precision_score(cancer_Y_test, cancer_Y_pred,␣
↪pos_label="Benign"))
print("Recall:",metrics.recall_score(cancer_Y_test, cancer_Y_pred,␣
↪pos_label="Benign"))
```

```
Accuracy: 0.9824561403508771
Precision: 0.9767441860465116
Recall: 0.9767441860465116
```

[44]:
```
class_names = [0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(cancer_cnf_matrix), annot=True, cmap="YlGnBu", fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
```

```
plt.xlabel('Predicted label')
```

[44]: Text(0.5, 257.44, 'Predicted label')



[48]:
```
C = [10, 1, .1, .001]
for c in C:
    clf = LogisticRegression(penalty ='l1', C=c, solver='liblinear')
    clf.fit(cancer_X_trainstd, cancer_Y_train)
    print('C:', c)
    print('Training accuracy: ', clf.score(cancer_X_trainstd, cancer_Y_train))
    print('Test accuracy: ', clf.score(cancer_X_teststd, cancer_Y_test))
    print(' ')
```

```
C: 10
Training accuracy:  0.9912087912087912
Test accuracy:  0.956140350877193

C: 1
Training accuracy:  0.989010989010989
Test accuracy:  0.9736842105263158
```

```
C: 0.1
Training accuracy:  0.9802197802197802
Test accuracy:  0.9649122807017544

C: 0.001
Training accuracy:  0.37142857142857144
Test accuracy:  0.37719298245614036
```

[49]:
```python
cancer_kfold = KFold(n_splits=5, random_state=42, shuffle=True)
model = LogisticRegression(solver='liblinear')
results = cross_val_score(model, cancer_X, cancer_Y, cv=cancer_kfold)
print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

```
Accuracy: 94.723% (2.675%)
```

[50]:
```python
cancer_kfold = KFold(n_splits=10, random_state=42, shuffle=True)
model = LogisticRegression(solver='liblinear')
results = cross_val_score(model, cancer_X, cancer_Y, cv=cancer_kfold)
print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

```
Accuracy: 94.897% (3.477%)
```

[51]:
```python
for c in C:
    cancer_kfold = KFold(n_splits=5, random_state=42, shuffle=True)
    model = LogisticRegression(penalty ='l1', C=c, solver='liblinear')
    results = cross_val_score(model, cancer_X, cancer_Y, cv=cancer_kfold)
    print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.
 ↪std()*100.0))
```

```
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

```
Accuracy: 96.479% (2.097%)
```

```
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
```

```
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

```
Accuracy: 94.898% (3.219%)
Accuracy: 93.316% (3.760%)
Accuracy: 91.559% (3.039%)
```

```
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

[52]:
```
for c in C:
    cancer_kfold = KFold(n_splits=10, random_state=42, shuffle=True)
    model = LogisticRegression(penalty ='l1', C=c, solver='liblinear')
    results = cross_val_score(model, cancer_X, cancer_Y, cv=cancer_kfold)
    print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.
 ↪std()*100.0))
```

```
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

```
Accuracy: 95.949% (2.966%)
```

```
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
```

```
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

```
Accuracy: 94.721% (3.346%)
```

```
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
C:\Users\homer\anaconda3\lib\site-packages\sklearn\svm\_base.py:1206:
```

```
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

```
Accuracy: 93.318% (4.636%)
Accuracy: 91.557% (5.090%)
```

[ ]:

```
```