HW2_P3P4

October 9, 2022

[1]:
```
"""
    Edward Pascual-Bautista
    ECGR 4105 HW2
"""
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[2]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

[3]:
```
breast = load_breast_cancer()

breast_data = breast.data
breast_data.shape
```

[3]:
```
(569, 30)
```

[4]:
```
breast_input = pd.DataFrame(breast_data)
breast_input.head()
```

[4]:
```
      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]
```

[5]:
```
breast_labels = breast.target
breast_labels.shape
```

[5]:
```
(569,)
```

[6]:
```
labels = np.reshape(breast_labels,(569,1))
final_breast_data = np.concatenate([breast_data,labels],axis = 1)
final_breast_data.shape
```

[6]:
```
(569, 31)
```

[7]:
```
breast_dataset = pd.DataFrame(final_breast_data)
breast_dataset
```

[7]:
```
        0      1       2       3        4        5        6        7       8  \
0    17.99  10.38  122.80  1001.0  0.11840  0.27760  0.30010  0.14710  0.2419
1    20.57  17.77  132.90  1326.0  0.08474  0.07864  0.08690  0.07017  0.1812
2    19.69  21.25  130.00  1203.0  0.10960  0.15990  0.19740  0.12790  0.2069
3    11.42  20.38   77.58   386.1  0.14250  0.28390  0.24140  0.10520  0.2597
4    20.29  14.34  135.10  1297.0  0.10030  0.13280  0.19800  0.10430  0.1809
..     …      …      …       …        …        …        …        …       …
564  21.56  22.39  142.00  1479.0  0.11100  0.11590  0.24390  0.13890  0.1726
565  20.13  28.25  131.20  1261.0  0.09780  0.10340  0.14400  0.09791  0.1752
566  16.60  28.08  108.30   858.1  0.08455  0.10230  0.09251  0.05302  0.1590
567  20.60  29.33  140.10  1265.0  0.11780  0.27700  0.35140  0.15200  0.2397
568   7.76  24.54   47.92   181.0  0.05263  0.04362  0.00000  0.00000  0.1587

           9  …     21      22      23       24       25      26      27  \
0    0.07871  …  17.33  184.60  2019.0  0.16220  0.66560  0.7119  0.2654
1    0.05667  …  23.41  158.80  1956.0  0.12380  0.18660  0.2416  0.1860
2    0.05999  …  25.53  152.50  1709.0  0.14440  0.42450  0.4504  0.2430
3    0.09744  …  26.50   98.87   567.7  0.20980  0.86630  0.6869  0.2575
4    0.05883  …  16.67  152.20  1575.0  0.13740  0.20500  0.4000  0.1625
..       …  …    …      …       …        …        …       …       …
564  0.05623  …  26.40  166.10  2027.0  0.14100  0.21130  0.4107  0.2216
565  0.05533  …  38.25  155.00  1731.0  0.11660  0.19220  0.3215  0.1628
566  0.05648  …  34.12  126.70  1124.0  0.11390  0.30940  0.3403  0.1418
567  0.07016  …  39.42  184.60  1821.0  0.16500  0.86810  0.9387  0.2650
568  0.05884  …  30.37   59.16   268.6  0.08996  0.06444  0.0000  0.0000
```

```
         28       29   30
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
..      …        …    …
564  0.2060  0.07115  0.0
565  0.2572  0.06637  0.0
566  0.2218  0.07820  0.0
567  0.4087  0.12400  0.0
568  0.2871  0.07039  1.0

[569 rows x 31 columns]
```

[8]:
```
features = breast.feature_names

feature_labels = np.append(features, 'label')

breast_dataset.columns = feature_labels

breast_dataset.head()
```

[8]:
```
   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]
```

[9]:
```
"""breast_dataset['label'].replace(0, 'Benign',inplace=True)
breast_dataset['label'].replace(1, 'Malignant',inplace=True)

breast_dataset.head()"""
```

[9]:
```
"breast_dataset['label'].replace(0,
'Benign',inplace=True)\nbreast_dataset['label'].replace(1,
'Malignant',inplace=True)\n\nbreast_dataset.head()"
```

[10]:
```
X = breast.data
y = breast.target
```

[11]:
```
X.shape
```

[11]:
```
(569, 30)
```

[12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8,␣
 ↪test_size = 0.2, random_state = 0)
X_train.shape
```

[12]:
```
(455, 30)
```

[13]:
```
sc = StandardScaler()

X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

[14]:
```
from sklearn.linear_model import LogisticRegression

classifer = LogisticRegression(random_state = 0)
classifer.fit(X_train_std, y_train)
```

[14]:
```
LogisticRegression(random_state=0)
```

[15]:
```
y_pred = classifer.predict(X_test_std)
```

[16]:
```
from sklearn import metrics

print("Accuracy: ", metrics.accuracy_score(y_test, y_pred))
print("Precision: ", metrics.precision_score(y_test,y_pred))
print("Recall: ", metrics.recall_score(y_test, y_pred))
```

```
Accuracy:  0.9649122807017544
Precision:  0.9701492537313433
Recall:  0.9701492537313433
```

[17]:
```
from sklearn.metrics import confusion_matrix

cnf_matrix = confusion_matrix(y_test, y_pred)
cnf_matrix
```

[17]:
```
array([[45,  2],
       [ 2, 65]], dtype=int64)
```

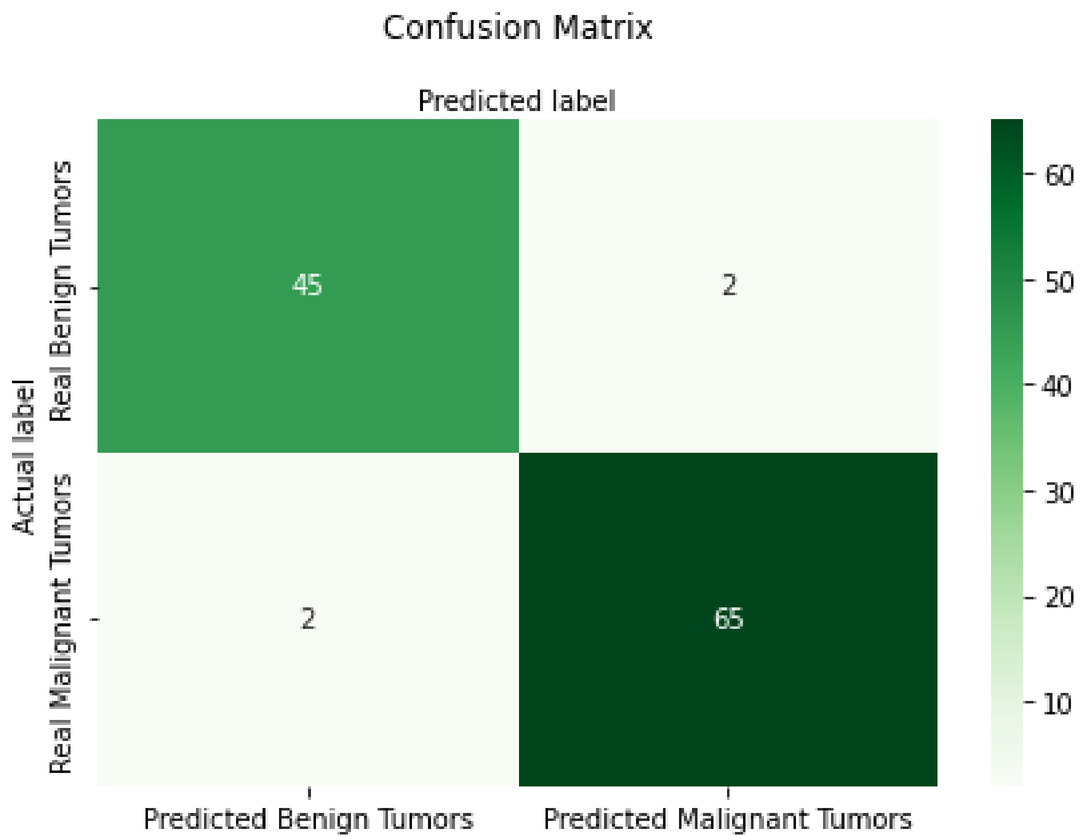
[18]:
```
import seaborn as sns

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(cnf_matrix), annot = True, cmap = plt.cm.Greens, fmt =␣
 ↪'g', yticklabels = ["Real Benign Tumors", "Real Malignant Tumors"],␣
 ↪xticklabels = ["Predicted Benign Tumors", "Predicted Malignant Tumors"])
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]: classifer_weight = LogisticRegression(penalty='l1', class_weight =␣
       ↪'balanced',solver = 'liblinear', random_state = 0)
      classifer_weight.fit(X_train_std, y_train)
```

```
[19]: LogisticRegression(class_weight='balanced', penalty='l1', random_state=0,
                         solver='liblinear')
```

```
[20]: y_pred = classifer_weight.predict(X_test_std)
```

```
[21]: print("Accuracy: ", metrics.accuracy_score(y_test, y_pred))
      print("Precision: ", metrics.precision_score(y_test,y_pred))
      print("Recall: ", metrics.recall_score(y_test, y_pred))
```

```
Accuracy:  0.956140350877193
Precision:  0.9558823529411765
Recall:  0.9701492537313433
```

```
[22]: from sklearn.model_selection import KFold
      from sklearn.model_selection import cross_val_score
```

[23]:

```
kfold = KFold(n_splits = 5, random_state = 0, shuffle = True)

model = LogisticRegression(solver = 'liblinear')

results = cross_val_score(model, X, y, cv = kfold)

print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

```
Accuracy: 95.434% (2.737%)
```

[24]:

```
kfold = KFold(n_splits = 10, random_state = 0, shuffle = True)

model = LogisticRegression(solver = 'liblinear')

results = cross_val_score(model, X, y, cv = kfold)

print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

```
Accuracy: 95.432% (3.858%)
```

[25]:

```
kfold = KFold(n_splits = 5, random_state = 0, shuffle = True)

model = LogisticRegression(penalty = 'l1', class_weight ='balanced', solver =␣
 ↪'liblinear')

results = cross_val_score(model, X, y, cv = kfold)

print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

```
Accuracy: 95.435% (2.680%)
```

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

[26]:

```
kfold = KFold(n_splits = 10, random_state = 0, shuffle = True)

model = LogisticRegression(penalty = 'l1', class_weight ='balanced', solver =␣
 ↪'liblinear')

results = cross_val_score(model, X, y, cv = kfold)

print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

```
Accuracy: 95.432% (3.695%)
```

[ ]:

```

```