mle001

February 2, 2022

1 Machine Learning Evaluation Metrics

1.1 Augusto Damasceno

- augustodamasceno@protonmail.com
- augustodamasceno.org

1.2 This lab is part of the project adlabs.

1.2.1 See https://github.com/augustodamasceno/adlabs/

1.3 ADlabs Licenses

1.4 Libraries

[1]:
```
# Numpy
import numpy as np
# Pyplot is a state-based interface to matplotlib. It provides a MATLAB-like␣
↪way of plotting.
import matplotlib.pyplot as plt
# Display plots that are the output of running code cells
%matplotlib inline
# Pandas
import pandas as pd
# Train and Test Subsets
from sklearn.model_selection import train_test_split
# Breast Cancer Wisconsin Dataset from Scikit-Learn Toy datasets
from sklearn.datasets import load_breast_cancer
# MLP Classifier
from sklearn.neural_network import MLPClassifier
# Accuracy Metric
from sklearn.metrics import accuracy_score
# Precision Metric
from sklearn.metrics import precision_score
# Recall Metric
```

```python
from sklearn.metrics import recall_score
# F1 Metric
from sklearn.metrics import f1_score
# Confusion Matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
# Cross-validation Metric
from sklearn.model_selection import cross_val_score
```

1.5 Breast Cancer Wisconsin Dataset

Data Set Characteristics:

Number of Instances:	569
Number of Attributes:	30 numeric, predictive attributes and the class

Attribute Information:	• radius (mean of distances from center to points on the perimeter)
	• texture (standard deviation of gray-scale values)
	• perimeter
	• area
	• smoothness (local variation in radius lengths)
	• compactness (perimeter^2 / area - 1.0)
	• concavity (severity of concave portions of the contour)
	• concave points (number of concave portions of the contour)
	• symmetry
	• fractal dimension ("coastline approximation" - 1)
	
	The mean, standard error, and "worst" or largest (mean of the three worst/largest values) of these features were computed for each image, resulting in 30 features. For instance, field 0 is Mean Radius, field 10 is Radius SE, field 20 is Worst Radius.
	
	• **class:**
	    ○ WDBC-Malignant
	    ○ WDBC-Benign

[2]:
```python
data = load_breast_cancer()
```

[3]:
```python
# X = data.data
# y = data.target
X, y = load_breast_cancer(return_X_y=True)
```

[4]:
```python
print('Features:\n')
for feature in data.feature_names:
    print('\t', feature)
```

```
Features:

        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
```

[5]:
```python
print('Targets:\n\t', data.target_names)
```

```
Targets:
         ['malignant' 'benign']
```

[6]:
```python
# print(data.DESCR)
```

1.6 Training and Test Subsets

[7]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
 ↪3,shuffle=True, random_state=None)
```

[8]:
```python
X_train.shape
```

[8]: (398, 30)

[9]:
```python
X_test.shape
```

[9]: (171, 30)

[10]:
```python
y_train.shape
```

[10]: (398,)

```
[11]: y_test.shape
```

[11]: (171,)

1.7 Model Training

```
[12]: clf = MLPClassifier(random_state=None, max_iter=50*X_train.shape[0]).
        ↪fit(X_train, y_train)
```

1.8 Model Prediction

```
[13]: y_pred = clf.predict(X_test)
```

1.9 Model Evaluation with Accuracy

1.10 "All correct predictions"

$\frac{TP+TN}{TP+FP+TN+FN}$

```
[14]: accuracy_score(y_test, y_pred)
```

[14]: 0.9707602339181286

1.11 Model Evaluation with Precision

1.12 "It's really true when it says so."

$\frac{TP}{TP+FP}$

```
[15]: precision_score(y_test, y_pred, average='binary')
```

[15]: 0.9557522123893806

1.13 Model Evaluation with Recall

1.14 "How much predicts positive compare to all real positives."

$\frac{TP}{TP+FN}$

```
[16]: recall_score(y_test, y_pred, average='binary')
```

[16]: 1.0

1.15 Model Evaluation with Specificity

1.16 "How much predicts negative compare to all real negatives."

$\frac{TN}{TN+FP}$

[17]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
specificity = tn / (tn+fp)
specificity
```

[17]: 0.9206349206349206

1.17 Model Evaluation with F1

1.18 "Harmonic mean of precision and recall."

$\frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$

[18]:
```
f1_score(y_test, y_pred, average='binary')
```

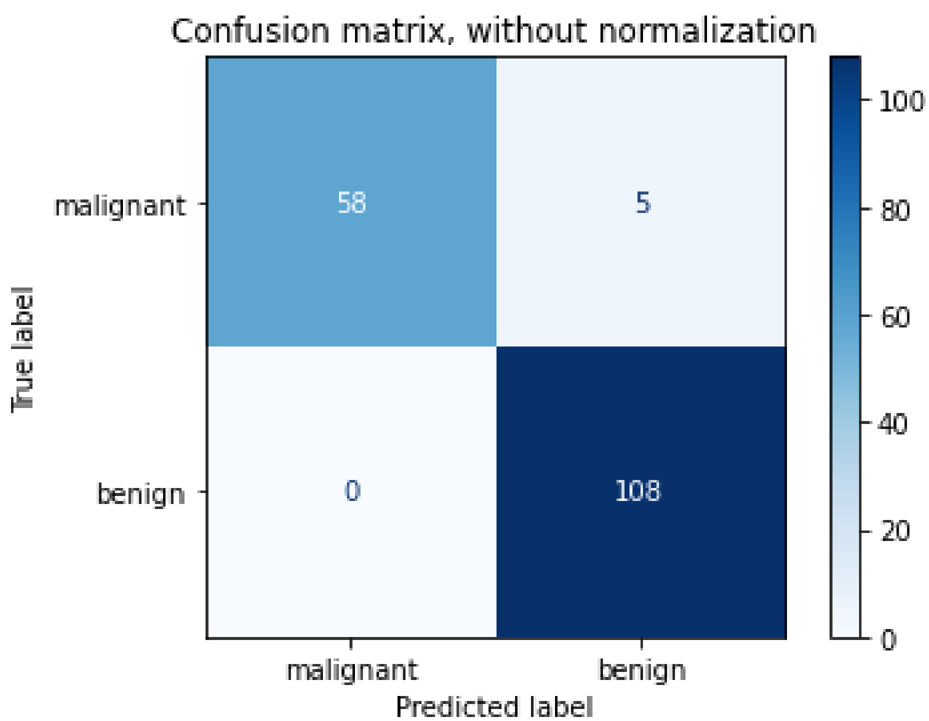
[18]: 0.9773755656108597

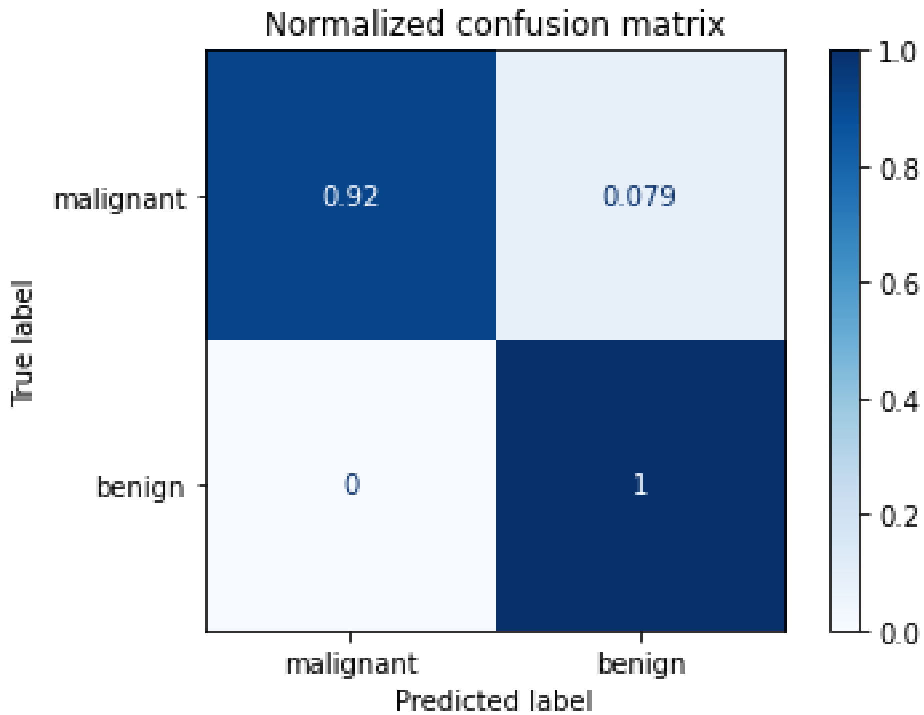
1.19 Model Evaluation with Confusion Matrix

[19]:
```
# Code from this cell in reference 15.

np.set_printoptions(precision=2)
class_names = data.target_names

titles_options = [
    ("Confusion matrix, without normalization", None),
    ("Normalized confusion matrix", "true"),
]
for title, normalize in titles_options:
    disp = ConfusionMatrixDisplay.from_estimator(
        clf,
        X_test,
        y_test,
        display_labels=class_names,
        cmap=plt.cm.Blues,
        normalize=normalize,
    )
    disp.ax_.set_title(title)

    tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
    print("tn={}\nfp={}\nfn={}\ntp={}\n".format(tn, fp, fn, tp))
    print(title)
    print(disp.confusion_matrix)
```

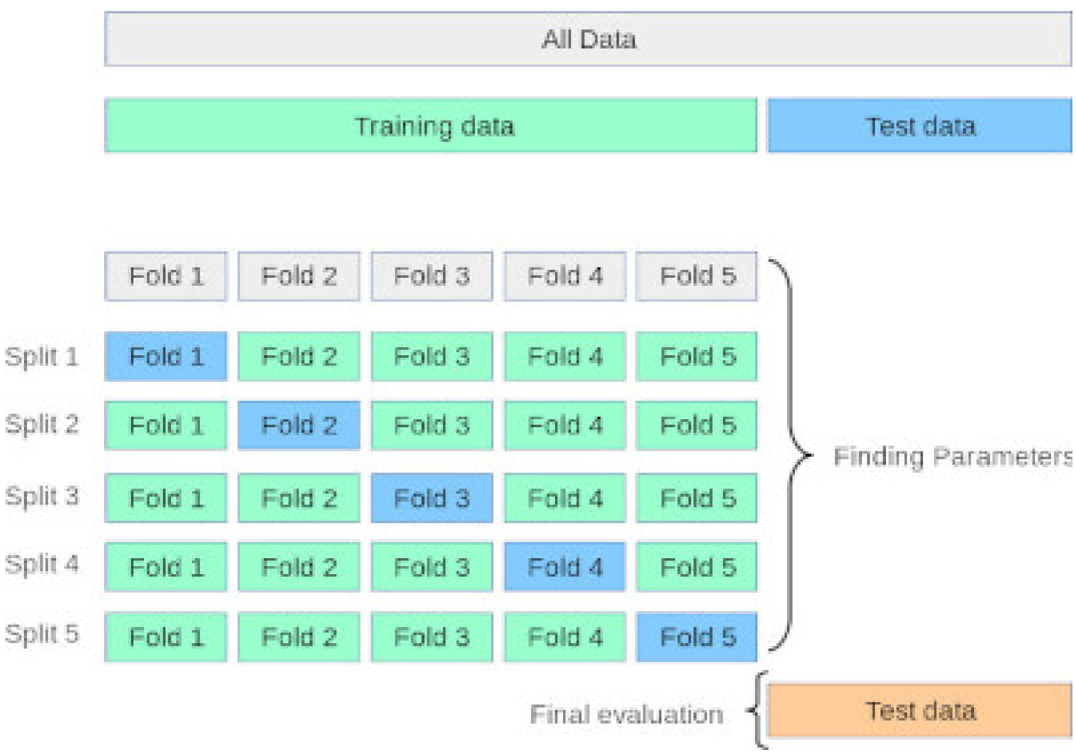
```
plt.show()
```

```
tn=58
fp=5
fn=0
tp=108

Confusion matrix, without normalization
[[ 58   5]
 [  0 108]]
tn=58
fp=5
fn=0
tp=108

Normalized confusion matrix
[[0.92 0.08]
 [0.   1.  ]]
```





1.20 Model Evaluation with Cross-validation



[20]:
```
scores = cross_val_score(clf, X, y, cv=5)
scores
```

[20]:
```
array([0.95, 0.92, 0.95, 0.96, 0.92])
```

[21]:
```
np.mean(scores)
```

[21]:
```
0.9384567613724576
```

[22]:
```
np.std(scores)
```

[22]:
```
0.01484694607160666
```

2 References

- 1. Python DOC

- 2. Numpy DOC
- 3. matplotlib.pyplot
- 4. ipython magic functions
- 5. Pandas DOC
- 6. Breast Cancer Wisconsin Dataset
- 7. sklearn.model_selection.train_test_split
- 8. sklearn.neural_network.MLPClassifier
- 9. sklearn.metrics.accuracy_score
- 10. sklearn.metrics.precision_score
- 11. sklearn.metrics.recall_score
- 12. sklearn.metrics.f1_score
- 13. Harmonic Mean
- 14. sklearn.metrics.confusion_matrix
- 15. https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
- 16. Cross-validation