Understanding Metrics of Classification

Importing libraries

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_linnerud
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
import itertools

np.random.seed(42)  # for reproducibility
sns.set(rc={"figure.figsize": (8, 8)})
sns.set_style("ticks")
```

Loading linnerud Dataset from sklearn

```
data = load_linnerud()
print(data.DESCR[:760])  # print short description
```

```
.. _linnerrud_dataset:

Linnerrud dataset
-----------------

**Data Set Characteristics:**

    :Number of Instances: 20
    :Number of Attributes: 3
    :Missing Attribute Values: None

The Linnerud dataset is a multi-output regression dataset. It consists of three
exercise (data) and three physiological (target) variables collected from
twenty middle-aged men in a fitness club:

- *physiological* - CSV containing 20 observations on 3 physiological variables:
   Weight, Waist and Pulse.
- *exercise* - CSV containing 20 observations on 3 exercise variables:
   Chins, Situps and Jumps.

.. topic:: References

  * Tenenhaus, M. (1998). La regression PLS: theorie et pratique. Paris:
    Editions Technic.
```

see the targets classes

```
print(f"Types of physical exercise (targets) are {data.target_names}")
```

```
Types of physical exercise (targets) are ['Weight', 'Waist', 'Pulse']
```

Print the target and the features

```
X = data.data  # features
y = data.target  # labels
print(f"Shape of features is {X.shape}, and shape of target is {y.shape}")
```

```
Shape of features is (20, 3), and shape of target is (20, 3)
```

Split the data

```
#Set test_size to an integer smaller than the number of samples
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=10, random_state=42)
```

```
y_train[:10]
```

```
array([[247.,  46.,  50.],
       [193.,  38.,  58.],
       [154.,  33.,  56.],
       [138.,  33.,  68.],
       [189.,  35.,  46.],
```

```
       [154.,  34.,  64.],
       [167.,  34.,  60.],
       [169.,  34.,  50.],
       [193.,  36.,  46.],
       [211.,  38.,  56.]])
```

Training and predicting data

```
classifier = svm.SVC(kernel='linear', probability=True, verbose=True)
```

fit/train the model on our training dataset

```
y_train = y_train.reshape(-1, 1)
```

```
X_train = X_train.reshape(-1, 1)
```

```
y_train_flattened = y_train.ravel()
```

```
classifier.fit(X_train, y_train_flattened)
```

```
[LibSVM]
SVC
SVC(kernel='linear', probability=True, verbose=True)
```

```
X_test = X_test.reshape(-1, 1)
```

```
classifier.fit(X_train, y_train)
y_preds = classifier.predict(X_test)
y_proba = classifier.predict_proba(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a
  y = column_or_1d(y, warn=True)
[LibSVM]
```

reshaping y_proba to a 1D vector

```
y_proba = y_proba[:,1].reshape((y_proba.shape[0],))
```

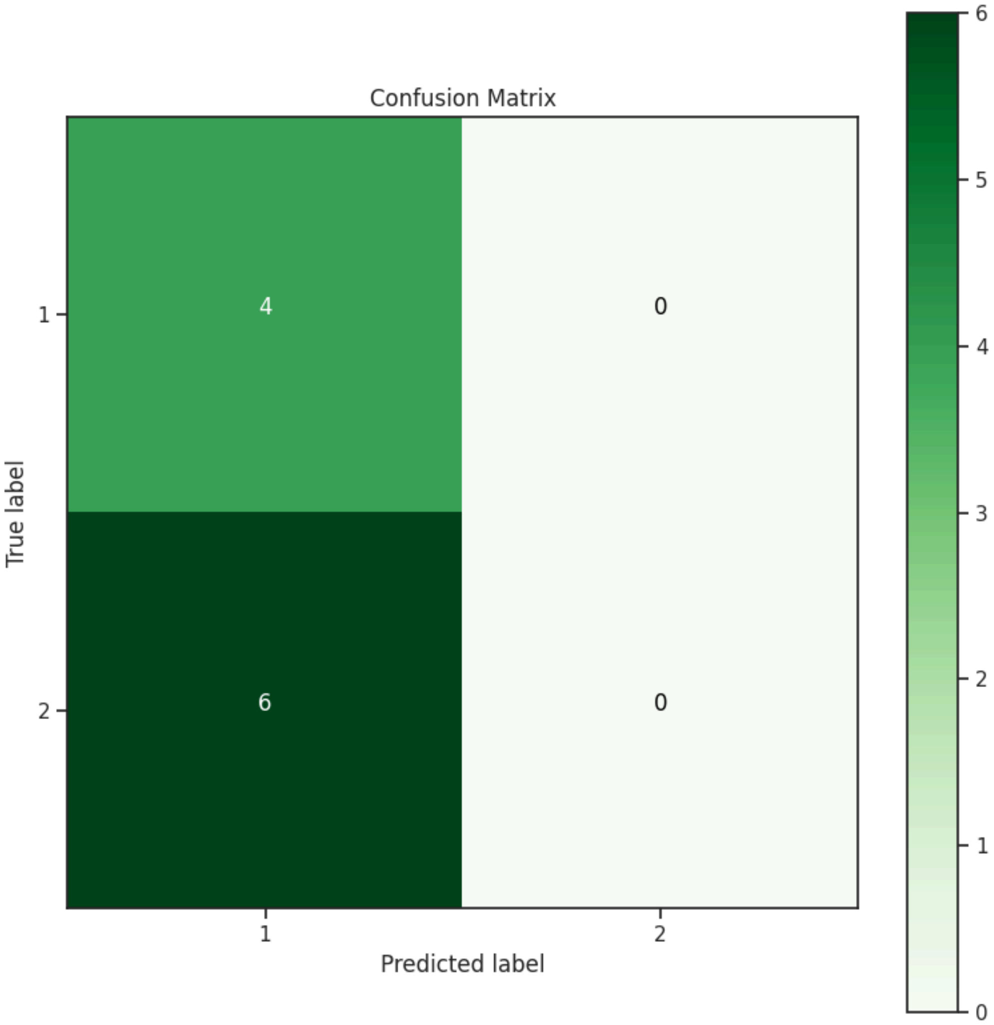
```
y_proba[:5], y_preds[:5], y_test[:5]
```

```
(array([0.03551558, 0.19522111, 0.05863534, 0.03607133, 0.31260209]),
 array([193.,  34.,  46., 193.,  33.]),
 array([[191.,  36.,  50.],
        [157.,  32.,  52.],
        [202.,  37.,  62.],
        [189.,  37.,  52.],
        [176.,  31.,  74.]]))
```

Confusion Matrix

```
import numpy as np
import matplotlib.pyplot as plt
import itertools
from sklearn.datasets import load_linnerud
from sklearn.model_selection import train_test_split
from sklearn.dummy import DummyClassifier
from sklearn.metrics import confusion_matrix

# Load the Linnerud dataset
data = load_linnerud()
X, y = data.data, data.target

# Convert the continuous target values to discrete classes for classification
# For simplicity, we'll classify based on the first target feature (e.g., weight lifting capacity)
y_class = np.digitize(y[:, 0], bins=np.linspace(min(y[:, 0]), max(y[:, 0]), num=4))

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y_class, test_size=10, random_state=42)

# Verify the lengths of the split data
print(f"Length of y_train: {len(y_train)}")
print(f"Length of y_test: {len(y_test)}")

# Train a dummy classifier
clf = DummyClassifier(strategy='most_frequent')
clf.fit(X_train, y_train)

# Make predictions
y_preds = clf.predict(X_test)

# Verify the lengths of y_test and y_preds
print(f"Length of y_preds: {len(y_preds)}")

# Compute the confusion matrix
conf = confusion_matrix(y_test, y_preds)
print("Confusion Matrix:")
print(conf)

# Manually plot the confusion matrix
classes = np.unique(np.concatenate((y_test, y_preds)))
plt.imshow(conf, interpolation='nearest', cmap=plt.cm.Greens)
plt.title("Confusion Matrix")
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)

fmt = 'd'
thresh = conf.max() / 2.
for i, j in itertools.product(range(conf.shape[0]), range(conf.shape[1])):
    plt.text(j, i, format(conf[i, j], fmt),
             horizontalalignment="center",
             color="white" if conf[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

```
Length of y_train: 10
Length of y_test: 10
Length of y_preds: 10
Confusion Matrix:
[[4 0]
 [6 0]]
```



Results from Confusion Matrix

```
# from the confusion matrix
TP = true_pos = 0
TN = true_neg = 4
FP = false_pos = 0
FN = false_neg = 6
```

Some basic metrics

creating a dictionary results

```
results = {}
```

Accuracy

```
metric = "ACC"
results[metric] = (TP + TN) / (TP + TN + FP + FN)
print(f"{metric} is {results[metric]: .3f}")
```

```
ACC is  0.400
```

True Positive Rate

```
# Sensitivity or Recall
metric = "TPR"
results[metric] = TP / (TP + FN)
print(f"{metric} is {results[metric]: .3f}")
```

```
TPR is  0.000
```

True Negative Rate

```
# Specificity
```