Data Mining 2

Topic 03 : Review of Model Building

Lecture 03 : Classification Models

Dr Kieran Murphy

Department of Computing and Mathematics, Waterford Institute of Technology.
(Kieran.Murphy@setu.ie)

Spring Semester, 2023

Outline

- Confusion matrix
- Precision, Accuracy, Recall and Specificity
- ROC and AUC

A Non-perfect Test — Type I and Type II Errors

Consider an imperfect test with two outcomes, there are four possible outcomes:

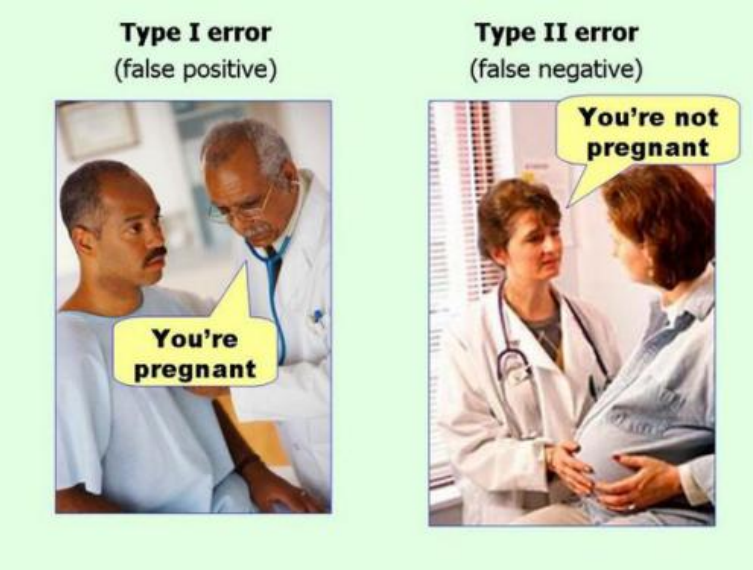
Confusion Matrix

Actual \ Predicted	Negative	Positive	
Negative	✔ True Negative (*TN*)	Type I error False Positive (*FP*)	N
Positive	Type II error False Negative (*FN*)	✔ True Positive (*TP*)	P
	$\hat{N}$	$\hat{P}$	T

- If the test is applied to $T = P + N = \hat{P} + \hat{N}$ observations / subjects / instances then we have four independent quantities *TP*, *TN*, *FP*, and *FN*.
- How do we combines these quantities into a single metric ?
- The fraction of correct results seems like a good idea

$$\text{accuracy} = \frac{TP + TN}{P + N}$$

But what happens, if we are testing for an rare event? Maximising accuracy will result in the test always returning negative.



- Ideally we want the probability of either error to be zero but that may not be possible.
- Depending on the conditions we often modify the test to reduce probability of the type of error we don't want at the expense of increasing the probability of the other — think court case vs medical condition.

Confusion matrix (Contingency table) Metrics I

Accuracy — how well model is trained and perform in general

Actual \ Predicted	Negative	Positive	
Negative	✔ True Negative (*TN*)	Type I error False Positive (*FP*)	*N*
Positive	Type II error False Negative (*FN*)	✔ True Positive (*TP*)	*P*
	$\hat{N}$	$\hat{P}$	*T*

$$\textbf{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}}$$

(How often is the classifier correct?)

- False negative rate (FNR) $= \frac{FN}{\mathrm{P}} = 1 - TPR$
- **Sensitivity** = **Recall** = True positive rate (TPR) $= \frac{TP}{\mathrm{P}} = 1 - FNR$

  (Of positive cases that exist how many did we mark positive?)

Recall — important when the costs of false negatives are high

- **Specificity** $= \frac{\mathrm{TN}}{\mathrm{N}} = 1 - \mathrm{FPR}$

  (When it's actually no, how often does we predict no?)

  (Of cases that are negative, how many did we mark negative?)
- False positive rate (FPR) = false acceptance $= \frac{FP}{\mathrm{N}} = 1 - \text{Specificity}$
- **Precision** = positive predictive value (PPV) $= \frac{TP}{\hat{\mathrm{P}}} = \frac{TP}{\mathrm{TP} + \mathrm{FP}}$

  (Of cases that we marked positive, how many were correct?)

Precision — important when the costs of false positives are high

Confusion matrix (Contingency table) Metrics II

F_1 Score

The F-measure or balanced F-score (F_1 score, is a special case of the F_β score) is the harmonic mean of precision and recall:

$$F_1 = 2\left[\frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}\right] = 2\left[\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}\right]$$

A word of Caution ...

Consider the three binary classifiers A, B and C

	A		B		C	
	F	T	F	T	F	T
F	0	0.1	0.1	0	0.1	0
T	0	0.9	0.1	0.8	0.12	0.78

Metric	**A**	**B**	**C**	(best)
Accuracy	0.9	0.9	0.88	**AB**
Precision	0.9	1.0	1.0	**BC**
Recall	1.0	0.888	0.8667	**A**
F-score	0.947	0.941	0.9286	**A**

Clearly classifier *A* is useless since it always predicts label T regardless of the input. Also, B is slightly better than C (lower off- diagonal total).
Yet look at the performance metrics – B is never the clear winner.

We use some metrics because they are easy to understand, and not because they always give the "correct" result.

Mutual Information is a Better Metric

The mutual information between predicted and actual label (case) is defined

$$I(\hat{y}, y) = \sum_{\hat{y}=\{0,1\}} \sum_{y=\{0,1\}} p(\hat{y}, y) \log \frac{p(\hat{y}, y)}{p(\hat{y})p(y)}$$

where $p(\hat{y}, y)$ is the joint probability distribution function.
This gives the intuitively correct rankings $B > C > A$

Metric	**A**	**B**	**C**
Accuracy	0.9	0.9	0.88
Precision	0.9	1.0	1.0
Recall	1.0	0.888	0.8667
F-score	0.947	0.941	0.9286
Mutual information	**0**	**0.1865**	**0.1735**

Receiver Operating Characteristic (ROC) I

Many classification systems have thresholds and parameters that can vary their performance*. In such cases, it can be useful to look at the TPR and FPR as the threshold/parameters are changed. One can seek to determine the best possible values for the threshold/parameters by finding a particular TPR vs FPR ratio. This is done by plotting an ROC curve.
For example, consider the Titanic survivor dataset, a part of the output from a classifier is given bellow:

Output from classifier: class probabilities

	Survived	Died
15	0.092	0.908
16	0.904	0.096
17	0.646	0.354
18	0.740	0.260
19	0.460	0.540

Sorted probabilities

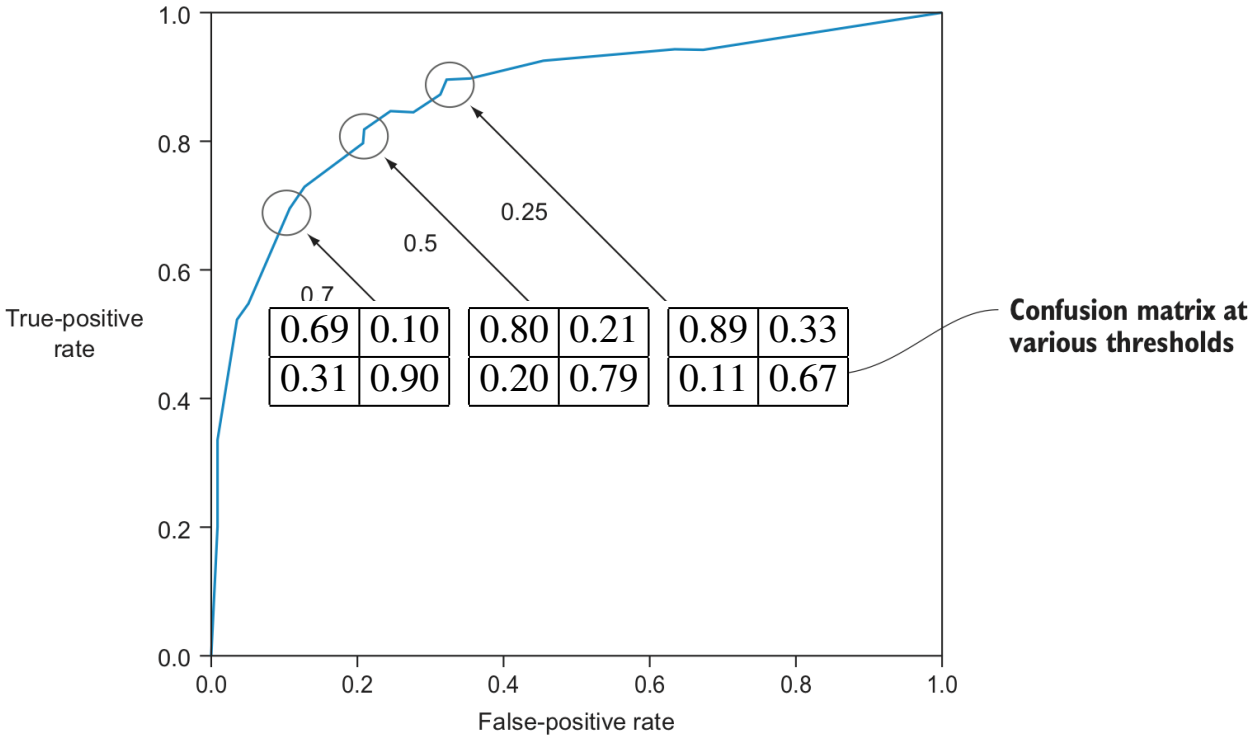
	Survived	Died
308	0.705	0.295
215	0.703	0.297
217	0.700	0.300
54	0.698	0.302
169	0.698	0.302

Threshold: "survived" probabilities > 0.7

After sorting the full table by decreasing survival probability, we can set a threshold and consider all rows above this threshold as survived.

*Think logistic regression – at what probability value do we switch from predicting 0 to predicting 1?

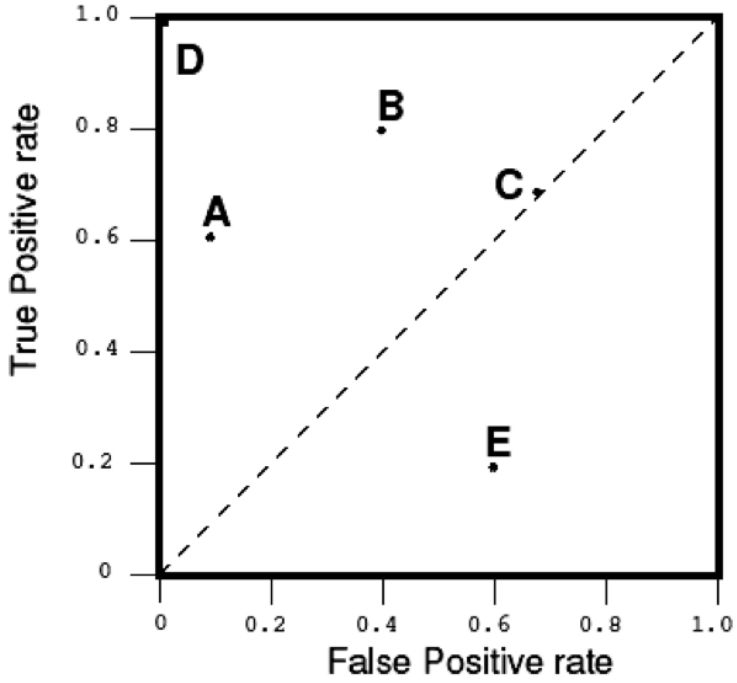
Receiver Operating Characteristic (ROC) II



The ROC curve defined by calculating the confusion matrix and ROC metrics at various threshold points from 0 to 1. By convention, we plot the false-positive rate on the x-axis and the true-positive rate on the y-axis.

ROC Space

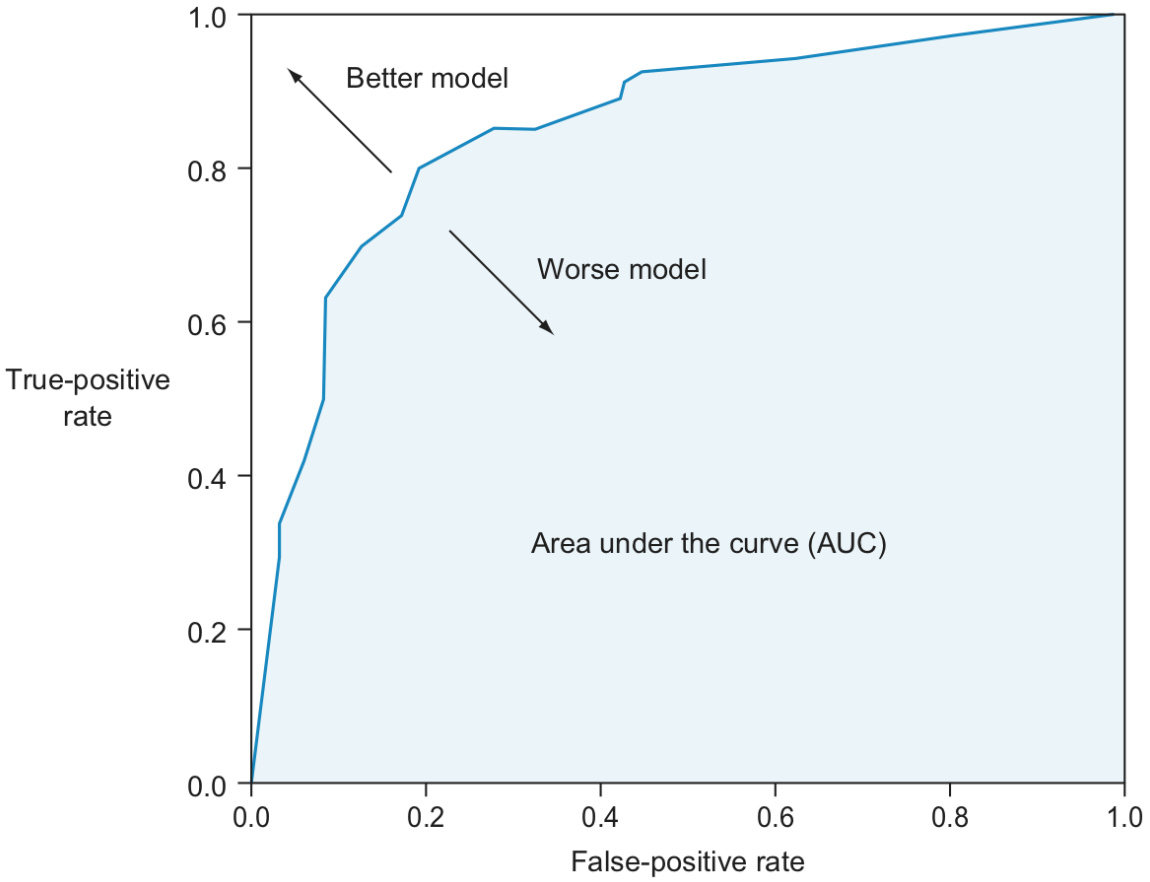
- Each point in ROC represents a classifier.
- Lower left point $(0, 0)$ represents the strategy of never issuing a positive classification: such a classier commits no false positive errors but also gains no true positives
- Upper right corner $(1, 1)$ represents the opposite strategy, of unconditionally issuing positive classifications.
- Point $(0, 1)$ represents perfect classification D's performance is perfect as shown.
- Informally, one point in ROC space is better than another if it is to the northwest of the first: TPR is higher, FPR is lower, or both.
- The diagonal line $y = x$ represents the strategy of randomly guessing a class.



An ROC graph with five classifiers: A,. . . , E.

Area Under an ROC Curve (AUC)

Rather than looking at individual points (based on particular threshold values) we can compare classifiers by using the area under the ROC.



The AUC of a classifier is equivalent to the probability that the classier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

The bigger AUC the better

Micro Average vs Macro Average Performance

In a multi-class classifier we have more than two classes (iris dataset):

- The ROC is computed for each class using the one-vs-all — for each class, we denote the particular class as the positive class, and everything else as the negative class, and we draw the ROC curves as usual.
- To combine the metrics for individual classes to get an overall system metrics, we can apply either

  Micro-Average Method

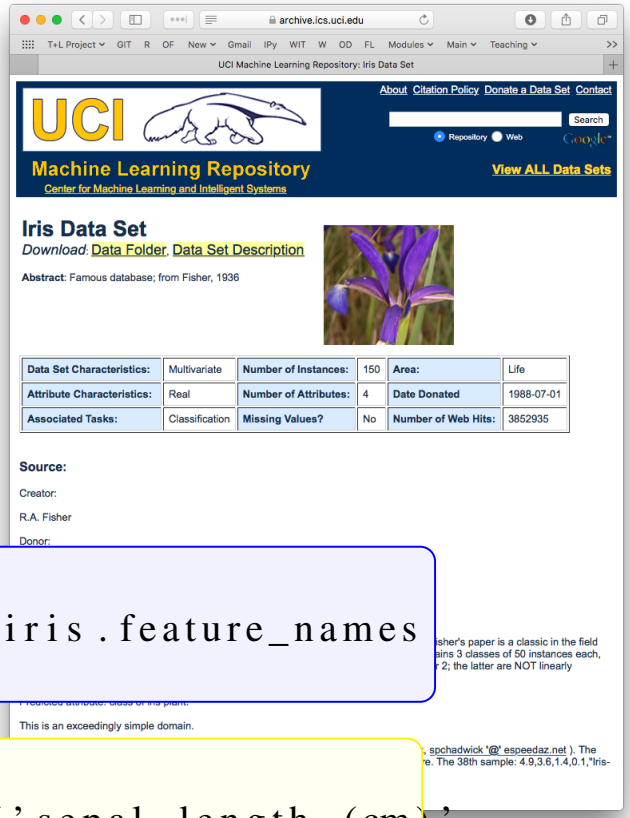
  Sum up the individual true positives, false positives, and false negatives of the system for different classes and then apply totals to get the statistics.

  Macro-average Method

  Average the precision and recall of the system on different classes.

See `classification_report` from `sklearn.metrics`.

Example: IRIS Dataset — Load



1

```
from sklearn import datasets
iris = datasets.load_iris()

df = pd.DataFrame(iris.data)
df.columns = iris.feature_names
df['target'] = iris.target_names[iris.target]
df.sample(4)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
17	5.1	3.5	1.4	0.3	setosa
80	5.5	2.4	3.8	1.1	versicolor
97	6.2	2.9	4.3	1.3	versicolor
99	5.7	2.8	4.1	1.3	versicolor

2

```
iris.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

The data set contains, four numeric features, 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Example: IRIS Dataset — Preprocess Data

We will cover some classifiers in a moment, but for now just treat the classifiers (`LogisticRegression`) as a black box and focus on the general process:

Extract the data (features and target)

The IRIS dataset has 4 features, but to simplify visualisation we are only going to use the first two[†] ('sepal length' and 'sepal width'):

```
3
dataset_name = "IRIS"
X, y, target_names = iris.data[:,:2], iris.target, iris.target_names
```

Split dataset into `train` and `test`

We will keep 40% of the data for testing. Setting the parameter `random_state` to a value means that we will get a random — but still reproducible — split.

```
4
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, random_state=666)
```


[†] Python for Data Science — Cheat Sheet Numpy Basics

Example: IRIS Dataset — Fit Model and Predict

Select classifier

Scikit-learn supports a large set of classifiers, and aims to have a consistent interface to all. First import classifier and create instance ...

5
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=500)
```

Train model

Then we train (fit) the classifier/model using only the features (`X_train`) and targets (`y_train`) from the train dataset ...

6
```
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=500)
```

Predict

Now that model is trained, we can use it to generate predictions, using the features (`X_test`) from the test dataset ...

7
```
y_pred = model.predict(X_test)
```

Example: IRIS Dataset — Evaluate I

Scoring and confusion matrix

We could just compute the score using whatever metric we have picked . . .

8
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.8333333333333334
```

But this needs context, and even if "good" it can hide critical flaws. Lets look at the confusion matrix . . .

9
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

```
array([[18,  1,  0],
       [ 0, 17,  6],
       [ 0,  3, 15]])
```

or, to get a nicer output, convert to a DataFrame . . .

10
```
df_cm = pd.crosstab(target_names[y_test],target_names[y_pred])
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'
df_cm
```

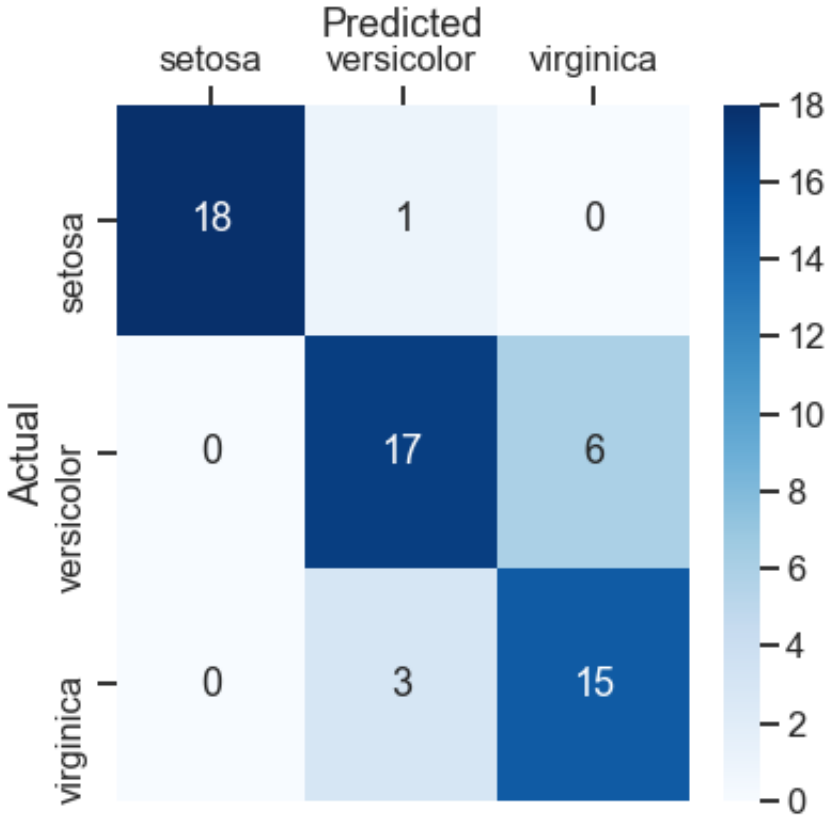
Predicted	setosa	versicolor	virginica
Actual			
setosa	18	1	0
versicolor	0	17	6

Example: IRIS Dataset — Evaluate II

The confusion matrix is fundamental in evaluating a classifier, so find a presentation/visualisation that you like and use it. Here I have a heat map representation that I tend to use.

Predicted	setosa	versicolor	virginica
Actual			
setosa	18	1	0
versicolor	0	17	6
virginica	0	3	15

The first class `setosa` was only misclassified once, while the classifier had more difficulty between the second two classes.




```
11
plt.figure(figsize=(6,6))
g = sns.heatmap(df_cm, annot=True, cmap="Blues")
g.xaxis.set_ticks_position("top")
g.xaxis.set_label_position('top')
```

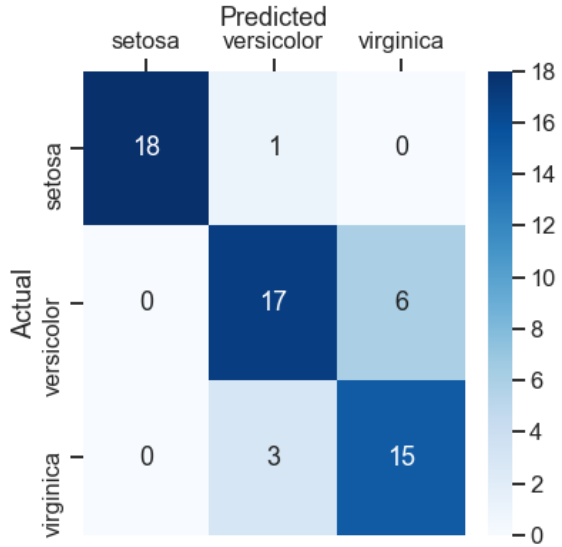
Example: IRIS Dataset — Evaluate III

The classification report, constructed from the confusion matrix, summaries the most common metrics per class and for overall averages . . .

```
12
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, target_names=target_names))
```

precision (setosa) = $TP/\hat{P} = 18/(18+0+0) = 1$

recall (setosa) = $TP/P = 18/(18+1+0) = 0.95$




```
                precision    recall  f1-score   support

        setosa       1.00      0.95      0.97        19
    versicolor       0.81      0.74      0.77        23
     virginica       0.71      0.83      0.77        18

      accuracy                           0.83        60
     macro avg       0.84      0.84      0.84        60
  weighted avg       0.84      0.83      0.84        60
```

accuracy = $(TP+TN)/(P+N) = (18+17+15)/60 = 0.83$

f1-score (virginica) = $2 \Big/ \left(\frac{1}{\text{precision}} + \frac{1}{\text{recall}}\right) = 2 \Big/ \left(\frac{1}{0.71} + \frac{1}{0.83}\right) = 0.77$