

Datasets and Metrics

Neural Networks for Machine Learning Applications

Spring 2023

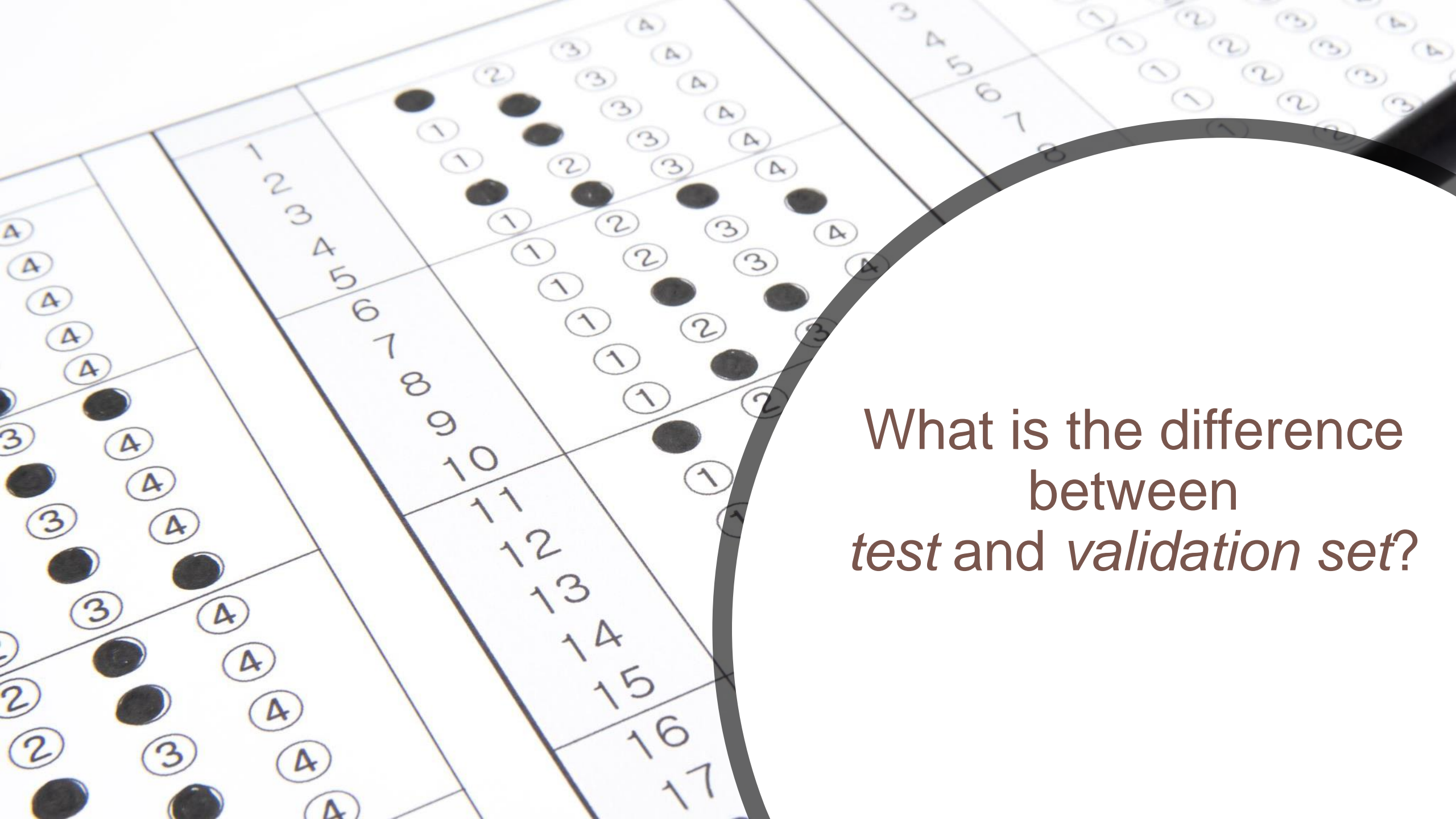
Sakari Lukkarinen

Metropolia University of Applied Sciences



Contents

- Dataset
 - Training, validation and test set
 - Supervised learning process
 - Why we need separate validation and test sets
 - How should we split the original data into datasets
 - 80/20 rule
- Metrics
 - Accuracy
 - Sensitivity and Specificity
 - Precision and recall
 - Confusion matrix
 - ROC curve
 - Metrics in scikit-learn
 - See also
- Example
 - UCI Heart disease dataset

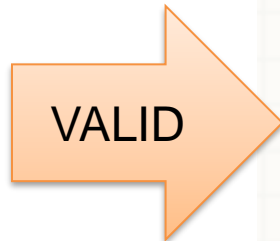
The image shows a close-up, slightly angled view of a 15-puzzle grid. The grid is composed of numbered tiles, with numbers 1 through 15 visible. Some tiles are black, indicating they are in their goal positions. A semi-transparent circular overlay is positioned on the right side of the image, containing the text "What is the difference between test and validation set?".

What is the difference
between
test and *validation* set?

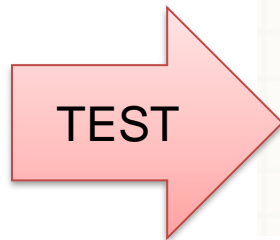
Supervised learning



TRAIN



VALID



TEST

While performing machine learning you do the following:

1. Training phase: you present your data from your "gold standard" and train your model, by pairing the input with expected output.
2. Validation/Test phase: in order to estimate how well your model has been trained (that is dependent upon the size of your data, the value you would like to predict, input etc) and to estimate model properties (mean error for numeric predictors, classification errors for classifiers, recall and precision for IR-models etc.)
3. Application phase: now you apply your freshly-developed model to the real-world data and get the results. Since you normally don't have any reference value in this type of data (otherwise, why would you need your model?), you can only speculate about the quality of your model output using the results of your validation phase.



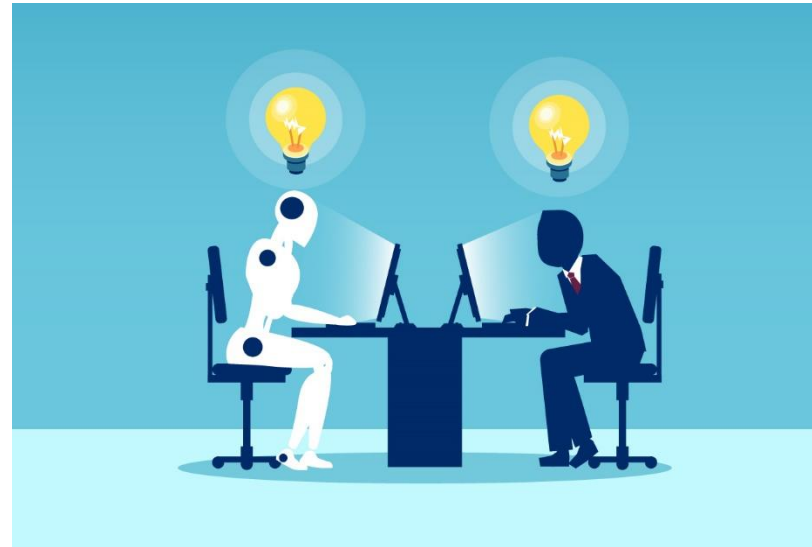
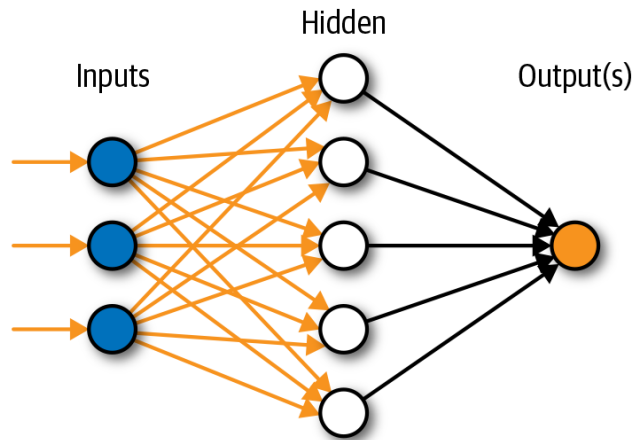
Why we need
separate validation
and test sets?

Why they can't be the
same?

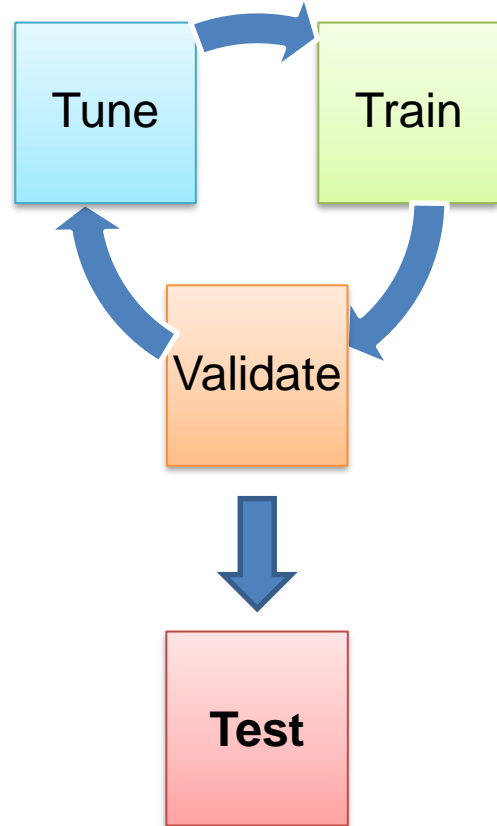
Why separate test and validation sets? The error rate estimate of the final model on validation data will be biased (smaller than the true error rate) since the validation set is used to select the final model. After assessing the final model on the test set, **YOU MUST NOT** tune the model any further!

source : Introduction to Pattern Analysis, Ricardo Gutierrez-Osuna Texas A&M University, Texas A&M University

Artificial Neural Network



Setting up development and test sets



We usually define:

- **Training set** — Which you run your learning algorithm on.
- **Dev (development) set** — Which you use to tune parameters, select features, and make other decisions regarding the learning algorithm. Sometimes also called the **hold-out cross validation set**.
- **Test set** — which you use to evaluate the performance of the algorithm, but not to make any decisions regarding what learning algorithm or parameters to use.

Once you define a dev set (development set) and test set, your team will try a lot of ideas, such as different learning algorithm parameters, to see what works best. The dev and test sets allow your team to quickly see how well your algorithm is doing.

In other words, **the purpose of the dev and test sets are to direct your team toward the most important changes to make to the machine learning system.**

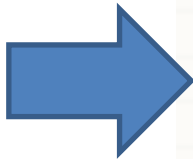
How should we split the original data into the datasets?

How much data should be used for training, validation and test?

First thoughts of partitioning

The validation phase is often split into two parts:

1. In the first part you just look at your models and select the best performing approach using the validation data (=validation)
2. Then you estimate the accuracy of the selected approach (=test).



Hence the separation to 50/25/25.

In case if you don't need to choose an appropriate model from several rivaling approaches, you can just re-partition your set that you basically have only training set and test set, without performing the validation of your trained model. I personally partition them 70/30 then.

See also [Why only three partitions?](#)



80/20 rule

Steps

1. Split your data into training and testing (80/20).
2. Split the training data into training and validation (80/20).

Results

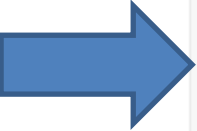
- Training set (64%)
 - used for training the model
- Validation set (16%)
 - used for tuning the parameters of the classifier during development
- Test set (20%)
 - used only to assess the final performance

Example – Heart disease prediction system

Splitting data

Separation of labels and features


In [34]:



```
x=df.drop(['HeartDiseaseorAttack', 'Education', 'Income'], axis=1)
y=df['HeartDiseaseorAttack']
```

Splitting data into train and test datasets

In [35]:



```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
y_train.value_counts()
```

Out[35]:

```
0.0    160844
1.0     16732
Name: HeartDiseaseorAttack, dtype: int64
```

[heart diseases prediction system | Kaggle](#)


Practise

- Open the [heart diseases prediction system](#)
- Find from Import libraries section the code for
 - `... import train_test_split`
 - Copy that to your Notebook
- Find the Splitting data section
 - Copy the separation of labels and features and splitting into train and test dataset code to your Notebook
- How do you split the data also for validation dataset?
 - Tip: You need to call second second time `train_test_split`, but now to apply it for the remaining train set



PART 2 - METRICS

Datasets and Metrics



How do we
evaluate
the
performance of
our model?



Most common metrics

Classification metrics

- **Accuracy**, precision, recall, fscore
- **Sensitivity and specificity**
- ROC Curve and area under ROC

Methods for classification prediction results

- **Confusion Matrix**
- **Classification Report**

Regression metrics

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- R^2 (R-squared)

A case problem – binary classifier

Background

We have data for 100 heart disease patients.

We know their diagnoses and have a bundle of measured data (blood pressure, cholesterol values, age, weight, ...).

We have trained our model to predict the diagnose and we want to know how well our model works.

How to calculate the accuracy, sensitivity and specificity of our model?

Case #	Diagnose (Disease = 1) (Healthy = 0)	Model (Disease = 1) (Healthy = 0)
1	1	1
2	1	0
3	0	0
4	0	0
5	0	1
6	1	1
7	0	0
8	0	1
....
100	1	0

Confusion (error) matrix

		Predicted condition	
		Cancer	Non-cancer
Actual condition	Total 8 + 4 = 12		
	Cancer	6	2
	Non-cancer	1	3

Individual Number	1	2	3	4	5	6	7	8	9	10	11	12
Actual Classification	1	1	1	1	1	1	1	1	0	0	0	0
Predicted Classification	0	0	1	1	1	1	1	1	1	0	0	0

In the field of [machine learning](#) and specifically the problem of [statistical classification](#), a **confusion matrix**, also known as an **error matrix**, is a specific table layout that allows visualization of the performance of an algorithm.

[Confusion matrix \(Wikipedia\)](#)

How to calculate the confusion matrix manually

1. Count how many of the cases have

- True positives
 - Diagnose = 1, Model = 1
- False positives
 - Diagnose = 0, Model = 1
- False negatives
 - Diagnose = 1, Model = 0
- True negatives
 - Diagnose = 0, Model = 0

2. Arrange the counts into matrix form.

3. Add row and column sums.

	Test (+)	Test (-)	SUM
Diagnose (+)	42	18	60
Diagnose (-)	28	12	40
SUM	70	30	100

	Test (+)	Test (-)	SUM
Diagnose (+)	TP	FN	TP+FN
Diagnose (-)	FP	TN	FP+TN
SUM	TP+FP	FN+TN	TP+FP+FN+TN

Calculate the metrics (manual)

- **Accuracy**

- Sum of (TP + TN) divided by all cases (TP+FP+FN+TN)
- $(42 + 12)/100 = 0.54$

- **Sensitivity**

- TP divided by all Diagnose+
- $42/(42+18) = 42/60 = 0.70$

- **Specificity**

- TN divided by all Diagnose-
- $12/(28 + 12) = 12/40 = 0.30$

	Test (+)	Test (-)	SUM
Diagnose (+)	42	18	60
Diagnose (-)	28	12	40
SUM	70	30	100

Ideal classifier

Ideal classifier has only True Positives and True Negatives

- there are not false test results

- Accuracy
 - $(60 + 40) / 100 = 1.00$
- Sensitivity (=Recall)
 - $60 / (60 + 0) = 60 / 60 = 1.00$
- Specificity
 - $40 / (0 + 40) = 40 / 40 = 1.00$

	Test (+)	Test (-)	SUM
Diagnose (+)	60	0	60
Diagnose (-)	0	40	40
SUM	60	40	100



What do different
metrics mean?

How do we interpret
these values?

Accuracy

Accuracy is also used as a statistical measure of how well a [binary classification](#) test correctly identifies or excludes a condition. That is, the accuracy is the proportion of true results (both [true positives](#) and [true negatives](#)) among the total number of cases examined.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / \text{Total}$$

Source: [Accuracy in binary classification \(Wikipedia\)](#)

Binary classification

Binary or **binomial classification** is the task of [classifying](#) the elements of a given [set](#) into two groups (predicting which group each one belongs to) on the basis of a [classification rule](#).

A typical example:

- [Medical testing](#) to determine if a patient has certain disease or not – the classification property is the presence of the disease.

Condition

Disease (+, 1)

Healthy (no disease) (-, 0)

Source: [Binary classification \(Wikipedia\)](#)

Medical test

A **medical test** is a kind of [medical procedure](#) performed to [detect](#), [diagnose](#), or [monitor](#) diseases, disease processes, susceptibility, and determine a course of treatment.

Source: [Medical test \(Wikipedia\)](#)



Medical screening tests

Screenings are tests that look for *diseases before you have symptoms*. Screening tests can find diseases early, when they're easier to treat.

Some conditions that are commonly screened for

- Breast and cervical cancer in women
- Prostate cancer in men
- Colorectal cancer
- Diabetes
- High blood pressure
- High cholesterol
- Osteoporosis

Sensitivity and specificity

Sensitivity and ***specificity*** are statistical measures of the performance of a [binary classification test](#)

Sensitivity (also called the **true positive rate**, the [recall](#), or **probability of detection**) measures the *percentage (%) of sick people who are correctly identified by the test having the condition.*

Specificity (also called the **true negative rate**) measures the *percentage (%) of healthy people who are correctly identified by the test as not having the condition.*

Source: [Sensitivity and specificity \(Wikipedia\)](#)

Sensitivity and specificity

Total	Test (+)	Test (-)
True (+)	TP	FN
True (-)	FP	TN

How many relevant items are selected?
e.g. How many sick people are correctly identified as having the condition.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{True (+)}}$$

How many negative selected elements are truly negative?
e.g. How many healthy people are identified as not having the condition.

$$\text{Specificity} = \frac{\text{TN}}{\text{True (-)}}$$

Precision and recall

In [pattern recognition](#), [information retrieval](#) and [binary classification](#),

precision (also called [positive predictive value](#)) is the fraction of relevant instances (true positive) among the retrieved instances

- for example model predicts: “Disease”

recall (also known as [sensitivity](#)) is the fraction of relevant instances (true positive) that have been retrieved over the total amount of relevant instances

- for example, true condition/diagnose is “Disease”

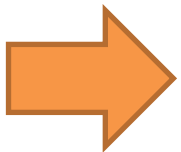
Both precision and recall are therefore based on an understanding and measure of [relevance](#).

Source: [Precision and recall \(Wikipedia\)](#)


Precision and recall

True condition
= Diagnose

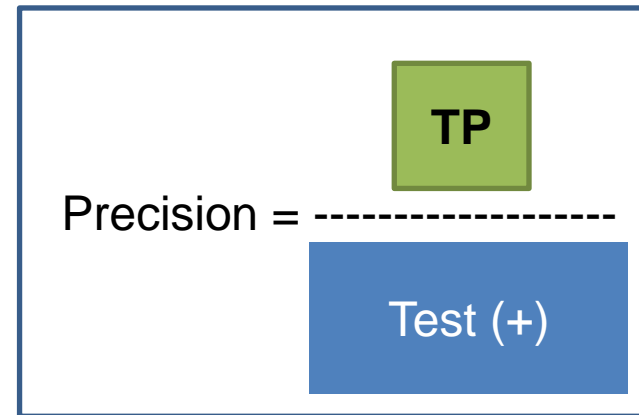
Test says:
"Disease" (+)
"Healthy" (-)



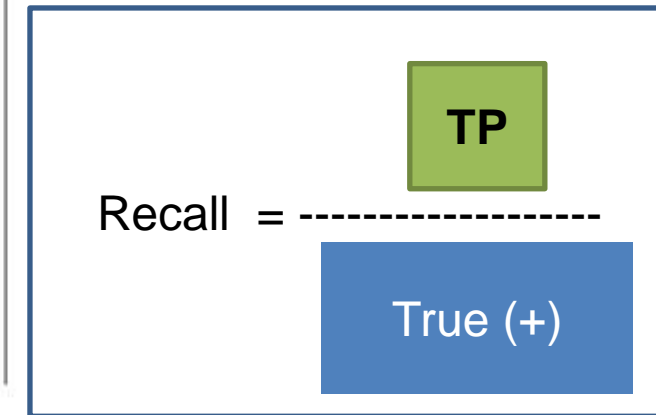
Total	Test (+)	Test (-)
True (+)	TP	FN
True (-)	FP	TN



How many selected
items are relevant?



How many relevant
items are selected?



NOTE:
Recall = Sensitivity !!!!

Practise

- Open the Notebook [Case 1. Sampling and metrics.](#)
- Study how
 - the dataset is read in
 - The evaluation and demonstration of metrics
- Apply them to your copy of Case 1. Template.
- Learn more
 - [3.3.2 Classification metrics — scikit-learn 1.2.0 documentation](#)

Confusion matrix [edit]

Main article: *Confusion matrix*

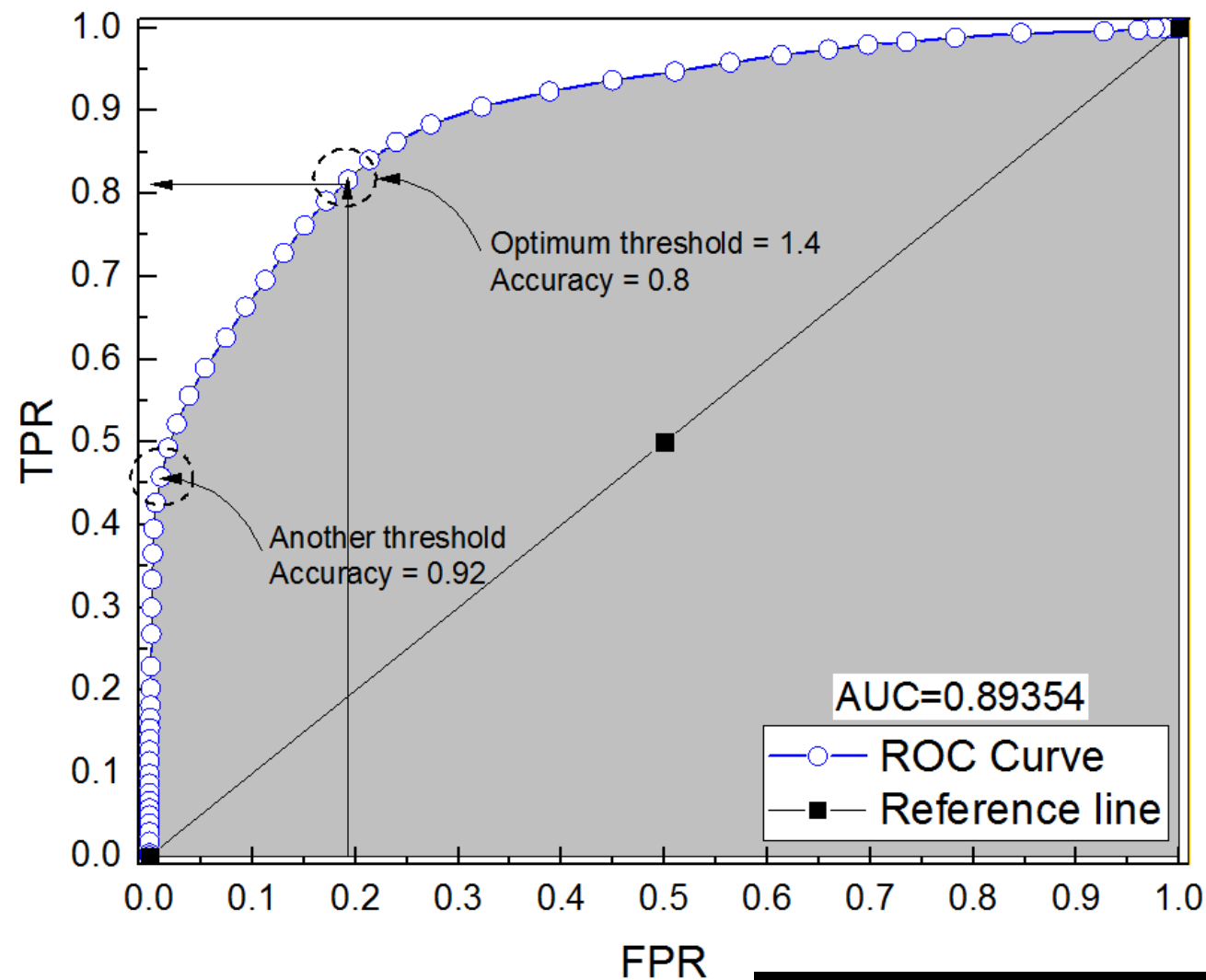
The relationship between sensitivity, specificity, and similar terms can be understood using the following table. Consider a group with **P** positive instances and **N** negative instances of some condition. The four outcomes can be formulated in a 2×2 *contingency table* or *confusion matrix*, as well as derivations of several metrics using the four outcomes, as follows:

Sources: [23][24][25][26][27][28][29][30] view · talk · edit

		Predicted condition			
Total population = P + N		Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) = TPR + TNR – 1	Prevalence threshold (PT) = $\frac{\sqrt{\text{TPR} \times \text{FPR}} - \text{FPR}}{\text{TPR} - \text{FPR}}$
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power = $\frac{\text{TP}}{\text{P}} = 1 - \text{FNR}$	False negative rate (FNR), miss rate = $\frac{\text{FN}}{\text{P}} = 1 - \text{TPR}$
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection	False positive rate (FPR), probability of false alarm, fall-out = $\frac{\text{FP}}{\text{N}} = 1 - \text{TNR}$	True negative rate (TNR), specificity (SPC), selectivity = $\frac{\text{TN}}{\text{N}} = 1 - \text{FPR}$
Prevalence = $\frac{\text{P}}{\text{P} + \text{N}}$		Positive predictive value (PPV), precision = $\frac{\text{TP}}{\text{PP}} = 1 - \text{FDR}$	False omission rate (FOR) = $\frac{\text{FN}}{\text{PN}} = 1 - \text{NPV}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Negative likelihood ratio (LR–) = $\frac{\text{FNR}}{\text{TNR}}$
Accuracy (ACC) = $\frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$		False discovery rate (FDR) = $\frac{\text{FP}}{\text{PP}} = 1 - \text{PPV}$	Negative predictive value (NPV) = $\frac{\text{TN}}{\text{PN}} = 1 - \text{FOR}$	Markedness (MK), deltaP (Δp) = PPV + NPV – 1	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR–}}$
Balanced accuracy (BA) = $\frac{\text{TPR} + \text{TNR}}{2}$		F ₁ score = $\frac{2\text{PPV} \times \text{TPR}}{\text{PPV} + \text{TPR}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$	Fowlkes–Mallows index (FM) = $\sqrt{\text{PPV} \times \text{TPR}}$	Matthews correlation coefficient (MCC) = $\frac{\sqrt{\text{TPR} \times \text{TNR} \times \text{PPV} \times \text{NPV}}}{\sqrt{\text{FNR} \times \text{FPR} \times \text{FOR} \times \text{FDR}}}$	Threat score (TS), critical success index (CSI), Jaccard index = $\frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}$

ROC CURVE

Metrics



This Photo by Unknown author is licensed under [CC BY-SA](#).

What are ROC curve and
area under ROC?

Case – Heart Disease

Example:

Our machine learning model gives prediction of the *probability of having disease* (continuous value between 0...1)

What is the best *decision point (= threshold)* for making the binary decision?

threshold = 0.30, 0.50 or 0.70?

test_output = 1.0*(model_prediction >= threshold)

Case #	Diagnose (Disease = 1) (Healthy = 0)	Model (Disease = 1) (Healthy = 0)
1	1	0.875
2	1	0.321
3	0	0.008
4	0	0.120
5	0	0.652
6	1	0.988
7	0	0.432
8	0	0.537
....
100	1	0.492

How to calculate ROC (manual)

1. Calculate the binary output for the first decision point
2. Calculate the metrics
 - True_positive_rate = Sensitivity
 - False_negative_rate = 1 – Specificity
3. Repeat the steps 1 & 2 for all decision points
4. Draw a graph
 1. X = False_negative_rate
 2. Y = True_positive_rate
5. Decide which threshold gives the best outcome for your research goals.

Case #	Dg (+)	Model	Th = 0.30	Th = 0.50	Th = 0.70
1	1	0.875	1	1	1
2	1	0.321	1	0	0
3	0	0.008	0	0	0
4	0	0.120	0	0	0
5	0	0.652	1	1	0
6	1	0.988	1	1	1
7	0	0.432	1	0	0
8	0	0.537	1	1	0
....			
100	1	0.492	1	0	0

Example – ROC curve and Area under ROC

In this example three different models and their ability to predict heart disease are compared.

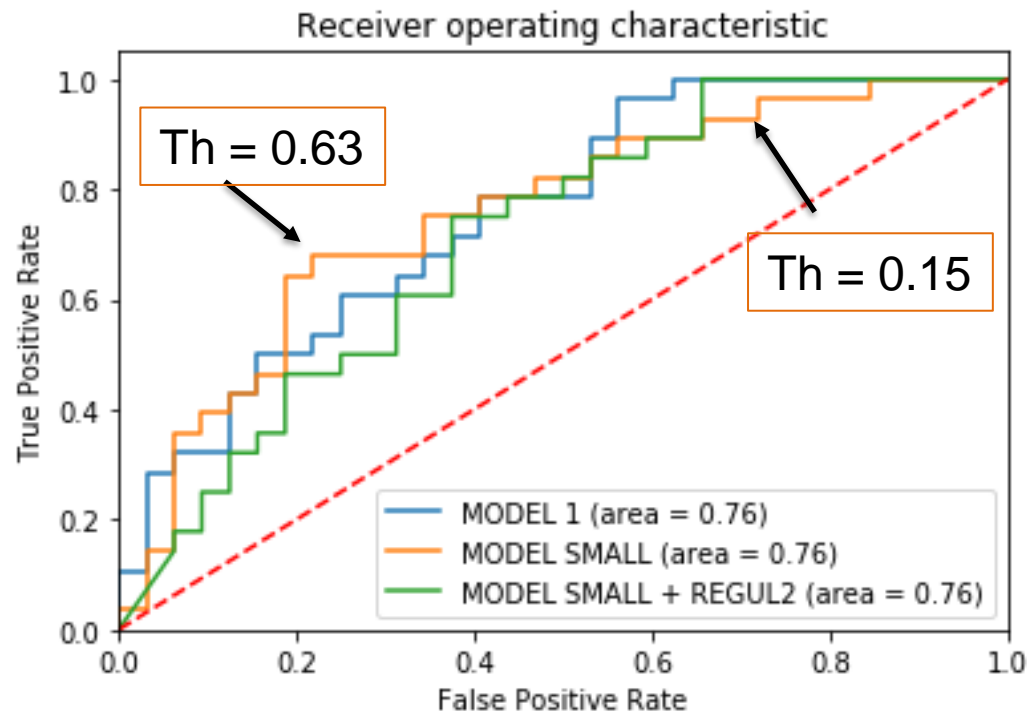
Each point in the graph represents different decision point on the probability output of the model.

Area under ROC represents how much of the unit square (1 x 1) the ROC curve covers. The higher the area the better.

- Area under ROC = 1 is ideal model.
- Area under ROC = 0.5 random result

Dashed red diagonal represent the results of totally random process.

- We give randomly probabilities for each case between 0 and 1 and then calculate the metrics.



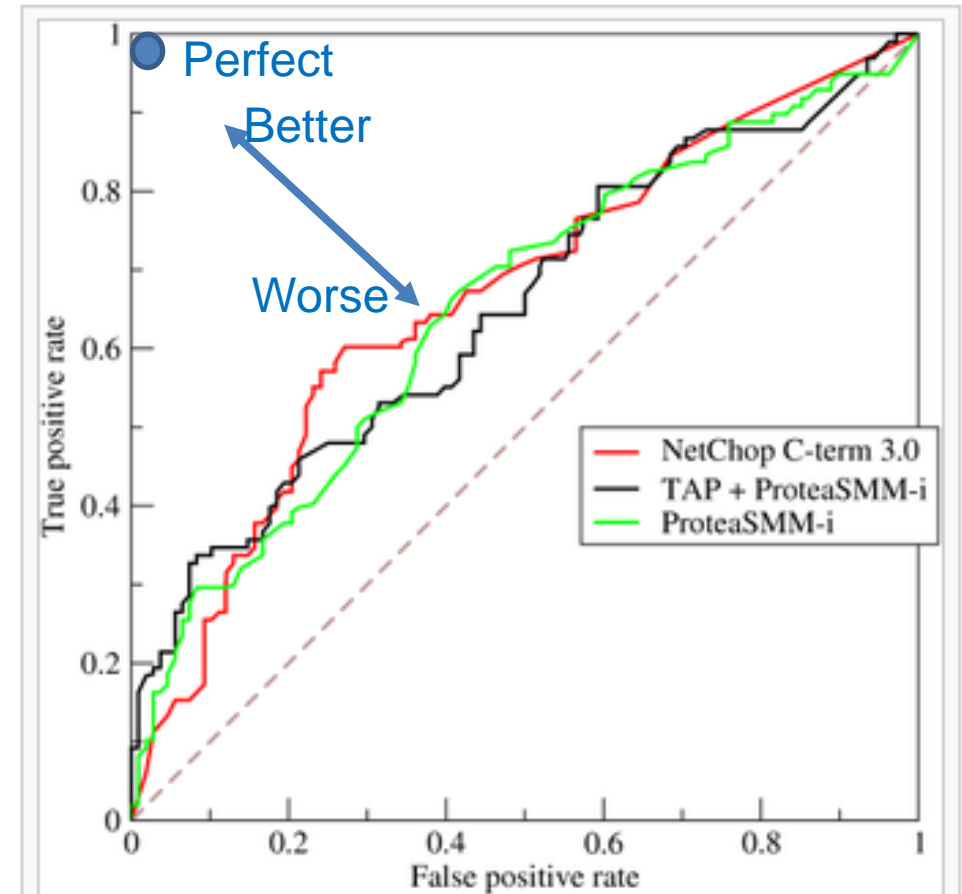
ROC curve

In statistics, a **receiver operating characteristic curve**, i.e. **ROC curve**, is a [graphical plot](#) that illustrates the diagnostic ability of a [binary classifier](#) system as its discrimination threshold is varied.

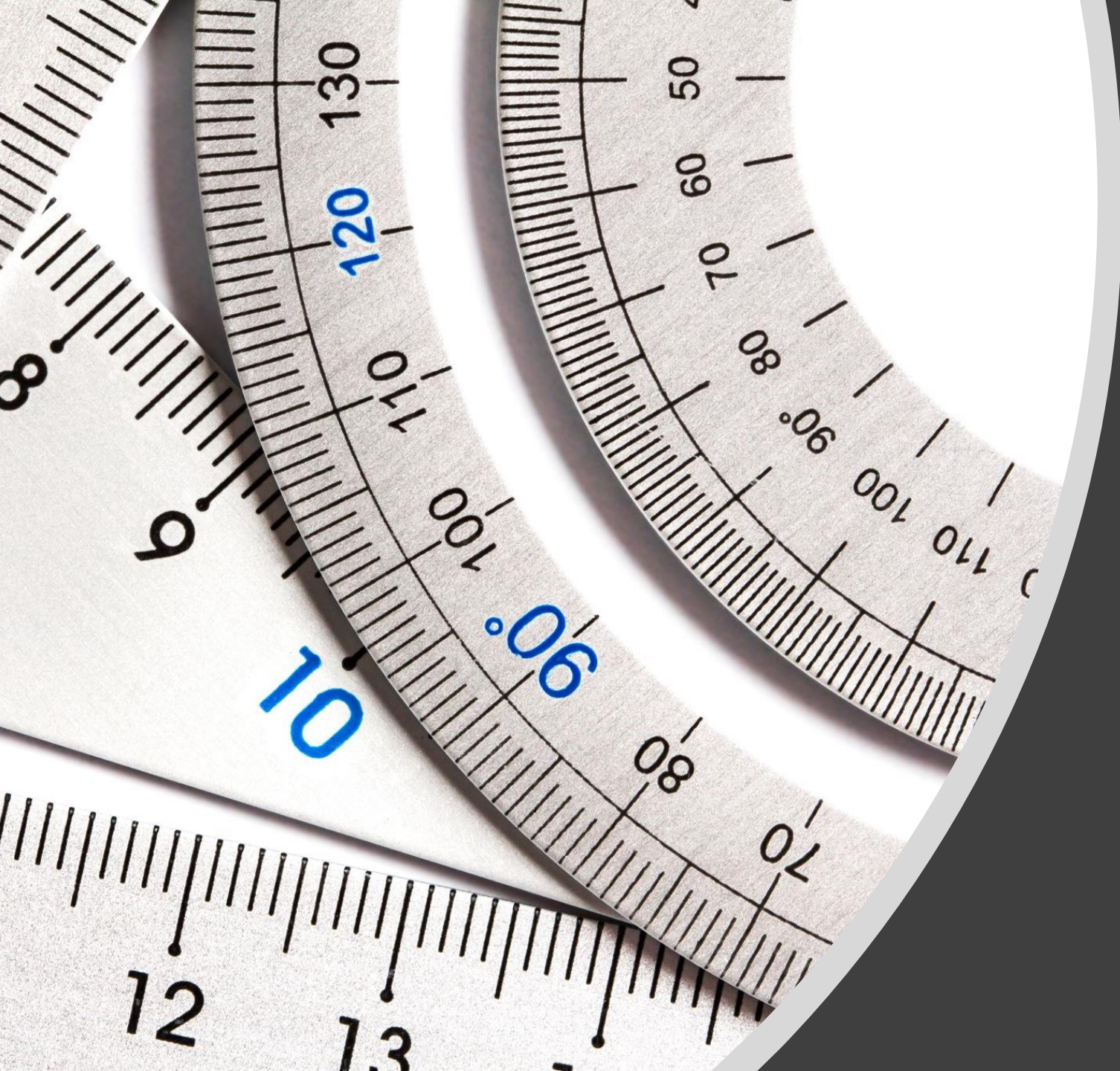
True positive rate = sensitivity

False positive rate = 1 - specificity

Source: [Receiver operating characteristics \(Wikipedia\)](#)



ROC curve of three predictors of peptide cleaving in the proteasome.



METRICS IN SCIKIT- LEARN

3.3.2. Classification metrics

The `sklearn.metrics` module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values. Most implementations allow each sample to provide a weighted contribution to the overall score, through the `sample_weight` parameter.

Some of these are restricted to the binary classification case:

<code>precision_recall_curve</code> (<code>y_true</code> , <code>probas_pred</code>)	Compute precision-recall pairs for different probability thresholds
<code>roc_curve</code> (<code>y_true</code> , <code>y_score</code> [, <code>pos_label</code> , ...])	Compute Receiver operating characteristic (ROC)

Others also work in the multiclass case:

<code>cohen_kappa_score</code> (<code>y1</code> , <code>y2</code> [, <code>labels</code> , <code>weights</code> , ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
<code>confusion_matrix</code> (<code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute confusion matrix to evaluate the accuracy of a classification
<code>hinge_loss</code> (<code>y_true</code> , <code>pred_decision</code> [, <code>labels</code> , ...])	Average hinge loss (non-regularized)
<code>matthews_corrcoef</code> (<code>y_true</code> , <code>y_pred</code> [, ...])	Compute the Matthews correlation coefficient (MCC)

http://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics

Some also work in the multilabel case:

<code>accuracy_score</code> (y_true, y_pred[, normalize, ...])	Accuracy classification score.
<code>classification_report</code> (y_true, y_pred[, ...])	Build a text report showing the main classification metrics
<code>f1_score</code> (y_true, y_pred[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
<code>fbeta_score</code> (y_true, y_pred, beta[, labels, ...])	Compute the F-beta score
<code>hamming_loss</code> (y_true, y_pred[, labels, ...])	Compute the average Hamming loss.
<code>jaccard_similarity_score</code> (y_true, y_pred[, ...])	Jaccard similarity coefficient score
<code>log_loss</code> (y_true, y_pred[, eps, normalize, ...])	Log loss, aka logistic loss or cross-entropy loss.
<code>precision_recall_fscore_support</code> (y_true, y_pred)	Compute precision, recall, F-measure and support for each class
<code>precision_score</code> (y_true, y_pred[, labels, ...])	Compute the precision
<code>recall_score</code> (y_true, y_pred[, labels, ...])	Compute the recall

Confusion matrix (scikit-learn version)

Scikit-learn (machine learning)

True condition (diagnosis)	Model prediction		
	Total	0	1
Healthy (-)	0	TN	FP
Disease (+)	1	FN	TP

Medical literature

	True condition (diagnosis)		
		Disease (+)	Healthy (-)
Test or model	Total	1	0
	1	TP	FP
	0	FN	TN

Store the values to a variable and reorder them !!!

```
cm = confusion_matrix(true_label, pred_label)
```

Precision, recall, fscore and support

Calculate precision, recall, fscore and support

P, R, F, S = [precision recall fscore support](#)(true_label, pred_label)

Support is the number of occurrences of each class in true_label

- In binary classification problem support contains the number of true (+) and true (-) cases

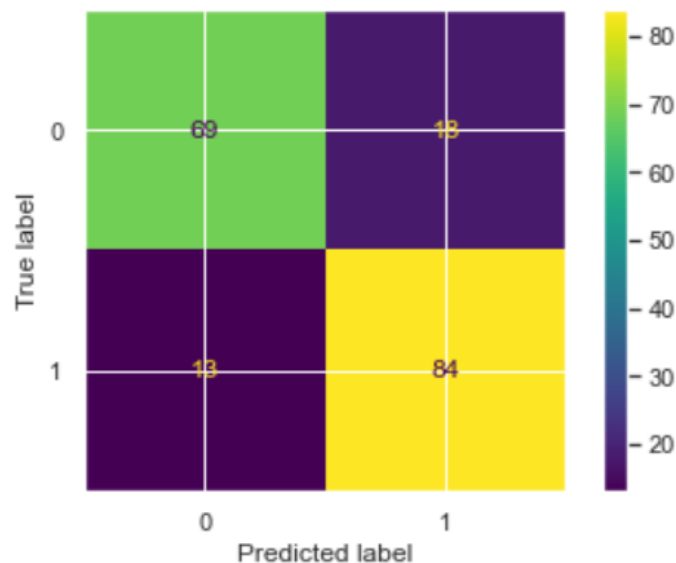
```
[4] from sklearn.metrics import precision_recall_fscore_support
y_true = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2]
y_pred = [1, 1, 1, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2]
P, R, F, S = precision_recall_fscore_support(y_true, y_pred)
print('Precision:', P)
print('Recall:    ', R)
print('FScore:   ', F)
print('Support:  ', S)
```

```
↳ Precision: [0.71428571 0.44444444]
Recall:      [0.5          0.66666667]
FScore:      [0.58823529 0.53333333]
Support:     [10  6]
```

```
from sklearn.metrics import classification_report, roc_curve, roc_auc_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
C = confusion_matrix(test_labels, pred_labels)
disp = ConfusionMatrixDisplay(C)
disp.plot()
print(C)
print("True positives:", C[1, 1])
print("False negatives:", C[0, 1])
print("False positives:", C[1, 0])
print("True negatives:", C[0, 0])
```

```
[[69 18]
 [13 84]]
True positives: 84
False negatives: 18
False positives: 13
True negatives: 69
```



Classification report

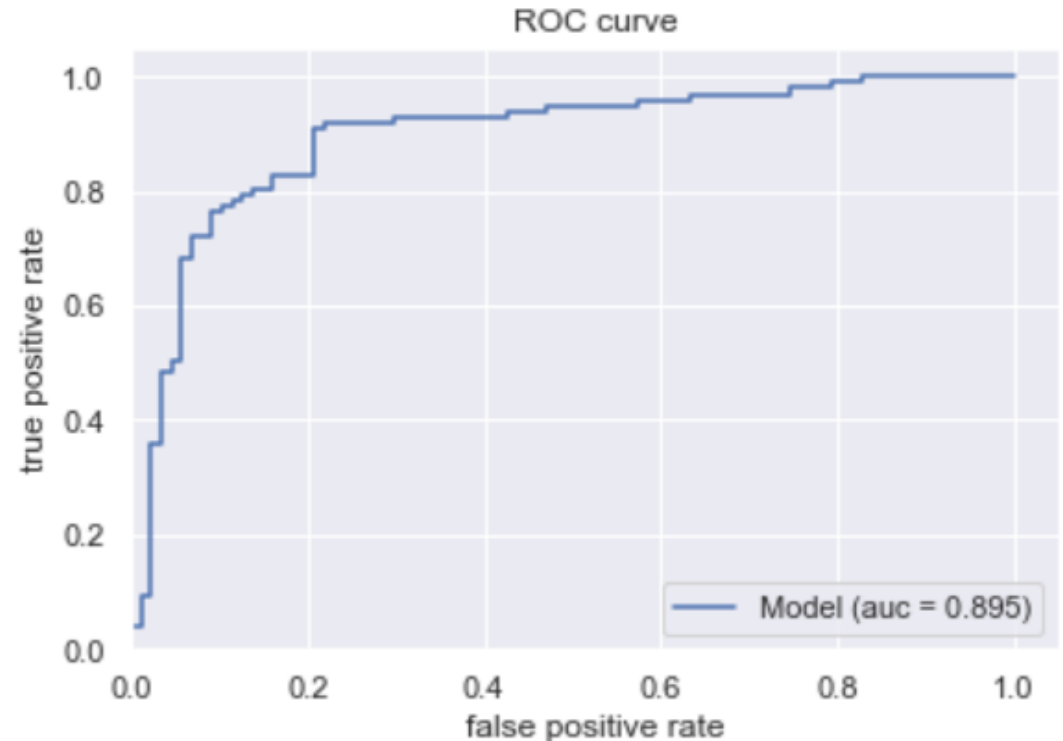
[Classification report](#) builds a text report showing the main classification metrics

```
CR = classification_report(test_labels, pred_labels)
print(CR)
```

	precision	recall	f1-score	support
0	0.84	0.79	0.82	87
1	0.82	0.87	0.84	97
accuracy			0.83	184
macro avg	0.83	0.83	0.83	184
weighted avg	0.83	0.83	0.83	184

ROC curve example with scikit-learn

```
fpr, tpr, th = roc_curve(test_labels, pred_values)
auc = roc_auc_score(test_labels, pred_values)
plt.plot(fpr, tpr, label = "Model (auc = {:.3f})".format(auc))
legend(loc = 4)
plt.title('ROC curve')
plt.xlabel('false positive rate')
plt.ylabel('true positive rate')
plt.xlim(0,)
plt.ylim(0,);
```



See also

[tf.keras.metrics](#)

- [BinaryAccuracy](#)
- [FalseNegatives](#)
- [FalsePositives](#)
- [TrueNegatives](#)
- [TruePositives](#)
- [Precision](#)
- [Recall](#)
- [AUC](#)