4. Datasets and Metrics

Neural Networks for Health Technology Applications
Spring 2020
Sakari Lukkarinen
Helsinki Metropolia University of Applied Sciences

Contents

- Dataset
  - Training, development (=validation) and test set
  - 60-20-20 rule
- Metrics
  - Accuracy
  - Medical tests
    - Sensitivity and specificity
  - Precision and recall
  - Confusion matrix
  - ROC curve
  - Metrics in scikit-learn
    - Confusion matrix
    - Precision, recall, fscore, and support
    - Classification report

What is the difference between test set and validation set?

มันคือ final test .test ครั้งสุดท้าย

ใช้ตรวจสอบระกว่างการทำงาน

What is the difference between test set and validation set?

307

220

I found this confusing when I use the neural network toolbox in Matlab.
It divided the raw data set into three parts:

1. training set
2. validation set
3. test set

I notice in many training or learning algorithm, the data is often divided into 2 parts, the training set and the test set.

My questions are:

1. what is the difference between validation set and test set?
2. Is the validation set really specific to neural network? Or it is optional.
3. To go further, is there a difference between validation and testing in context of machine learning?

machine-learning validation

https://stats.stackexchange.com/questions/19048/what-is-the-difference-between-test-set-and-validation-set

Supervised learning

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.

https://stats.stackexchange.com/questions/19048/what-is-the-difference-between-test-set-and-validation-set

Why separate test and validation sets?

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-OsunaTexas A&M University, Texas A&M University

https://stats.stackexchange.com/questions/19048/what-is-the-difference-between-test-set-and-validation-set

How to split the datasets?

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 this question.

To summarize

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**.

From: Andrew Ng. (2019). Setting up development and test sets. In Machine Learning Yearning (draft).

Datasets summary

- Training set (50-70%)
  - a set of examples used for learning
- Validation set (20-30%)
  - a set of examples used for tune the parameters of the classifier
- Test set (20-30 %)
  - a set of examples used only to assess the final performance

How do we evaluate
the performance of our model?

Metrics To Evaluate Machine Learning Algorithms in Python
by Jason Brownlee on May 25, 2016 in Python Machine Learning
Photo by Ferrous Büller, **CC BY-SA 2.0**

Metrics (scikit-learn)

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)

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.

Accuracy = (TP + TN) / Total

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

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

    (-) = No disease (Healthy)

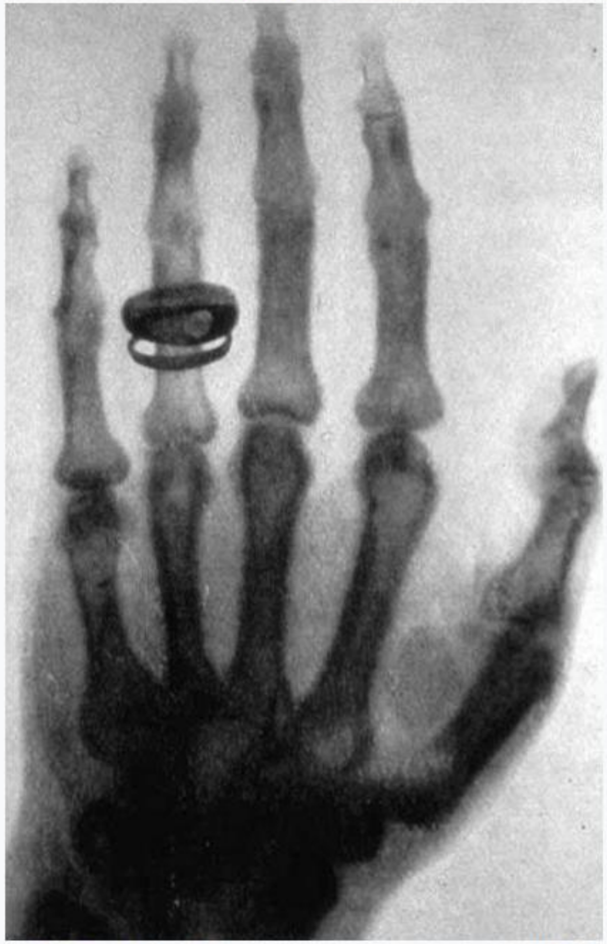
Source: Binary classification (Wikipedia)

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 test



X-ray of a hand. X-rays are a common medical test.

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

https://medlineplus.gov/healthscreening.html

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

True condition (diagnosis)

	Disease (+)	**Healthy (-)**
Total	**True (+)**	**True (-)**
Test (+)	**TP**	FP
Test (-)	FN	**TN**

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

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 peple are identified as not having the condition.

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

Source: [Sensitivity and specificity (Wikipedia)]

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 (model says: “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 (true condition 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 (diagnosis)

	Disease (+)	**Healthy (-)**
Total	**True (+)**	**True (-)**
Test (+)	**TP**	FP
Test (-)	FN	TN

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

How many selected items are relevant?

$$\text{Precision} = \frac{\textbf{TP}}{\text{Test (+)}}$$

How many relevant items are selected?

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

NOTE:
Recall = Sensitivity !!!!

Confusion (error) matrix

		Actual class		
		Cat	Dog	Rabbit
Predicted class	Cat	5	2	0
	Dog	3	3	2
	Rabbit	0	1	11

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.

Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa).

The name stems from the fact that it makes it easy to see if the system is confusing two classes.

Source: Confusion matrix (Wikipedia)

Confusion matrix [edit]

Let us 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 follows:

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$	F_1 score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$		

Source: Sensitivity and specificity (Wikipedia)

Terminology and derivations from a confusion matrix
(number of) positive samples (P)
(number of) negative samples (N)
(number of) true positive (TP)
eqv. with hit
(number of) true negative (TN)
eqv. with correct rejection
(number of) false positive (FP)
eqv. with false alarm, Type I error
(number of) false negative (FN)
eqv. with miss, Type II error

True positive (TP)
How many diseased persons got (true)"positive" label from the test.

True negative (TN)
How many healthy persons got (true) "negative" label from the test.

False positive (FP) (Type I error)
How many healthy persons got (false) "positive" label from the test.

False negative (FN) (Type I error)
How many diseased person got (false) "healthy" label from the test.

True positive, Power	**False positive**, Type I error
False negative, Type II error	**True negative**

sensitivity or **true positive rate (TPR)**

eqv. with hit rate, recall

$$TPR = TP/P = TP/(TP + FN)$$

specificity (SPC) or true negative rate

$$SPC = TN/N = TN/(TN + FP)$$

precision or positive predictive value (PPV)

$$PPV = TP/(TP + FP)$$

negative predictive value (NPV)

$$NPV = TN/(TN + FN)$$

fall-out or false positive rate (FPR)

$$FPR = FP/N = FP/(FP + TN) = 1 - SPC$$

false negative rate (FNR)

$$FNR = FN/(TP + FN) = 1 - TPR$$

false discovery rate (FDR)

$$FDR = FP/(TP + FP) = 1 - PPV$$

Sensitivity (true positive rate, recall, hit rate)

How well the model finds the (true) diseased person.

TPR = TP / (TP + FN)

Specificity (SPC, true negative rate)

How well the test finds the (true) healthy person.

SPC = TN / (TN + FP)

Precision (positive prediction value)

How well the model predicts the (true) disease.

PPV = TP / (TP + FP)

True positive, Power	**False positive**, Type I error
False negative, Type II error	**True negative**

accuracy (ACC)

$$ACC = (TP + TN)/(TP + FP + FN + TN)$$

F1 score

is the harmonic mean of precision and sensitivity

$$F1 = 2TP/(2TP + FP + FN)$$

Matthews correlation coefficient (MCC)

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Informedness

$$TPR + SPC - 1$$

Markedness

$$PPV + NPV - 1$$

Sources: Fawcett (2006) and Powers (2011).[2][3]

Accuracy

How many times the model gives the right diagnosis (either disease or healthy).

F1 score

The harmonic mean (~average) of **precision** and **recall** (sensitivity).

		True condition			
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$ F_1 score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

Definition (classification context) [edit]

For classification tasks, the terms *true positives*, *true negatives*, *false positives*, and *false negatives* (see Type I and type II errors for definitions) compare the results of the classifier under test with trusted external judgments. The terms *positive* and *negative* refer to the classifier's prediction (sometimes known as the *expectation*), and the terms *true* and *false* refer to whether that prediction corresponds to the external judgment (sometimes known as the *observation*).

Let us define an experiment from *P* positive instances and *N* negative instances for some condition. The four outcomes can be formulated in a 2×2 contingency table or confusion matrix, as follows:

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

Source: Precision and recall (Wikipedia)

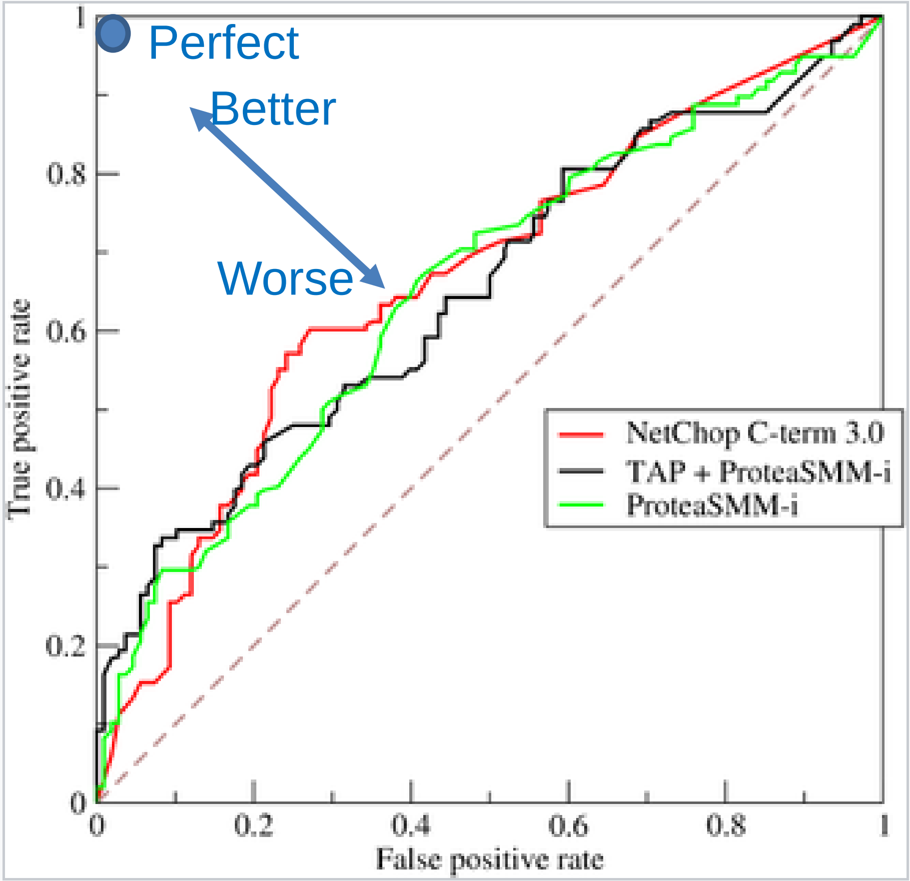
ROC curve

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.

The sklearn.metrics module includes score functions, performance metrics and pairwise metrics and distance computations.

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:

`precision_recall_curve` (y_true, probas_pred)	Compute precision-recall pairs for different probability thresholds
`roc_curve` (y_true, y_score[, pos_label, …])	Compute Receiver operating characteristic (ROC)

Others also work in the multiclass case:

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

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

Some also work in the multilabel case:

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

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

Confusion matrix (scikit-learn version)

NOTE!!! Scikit-learn gives the *transposed* confusion matrix, e.g. the rows and columns are changed:

Scikit-learn (machine learning)

		Test says: "Disease" (+)	Test says: "Healthy" (-)
True condition (diagnosis)	**Total**	**Test (+)**	**Test (-)**
Disease (+)	**True (+)**	**TP**	FN
Healthy (-)	**True (-)**	FP	TN

Medical literature

		True condition (diagnosis): Disease (+)	True condition (diagnosis): Healthy (-)
Test says	**Total**	**True (+)**	**True (-)**
"Disease" (+)	**Test (+)**	**TP**	FP
"Healthy" (-)	**Test (-)**	FN	TN

Solution is to use numpy's transpose function: `Cm = transpose(Cm)`

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 occurences 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, 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]
```

Classification report

Classification report builds a text report showing the main classification metrics

```
[1] from sklearn.metrics import classification_report
    y_true = [0, 1, 2, 2, 2]
    y_pred = [0, 0, 2, 2, 1]
    target_names = ['class 0', 'class 1', 'class 2']
    print(classification_report(y_true, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3

   micro avg       0.60      0.60      0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
```