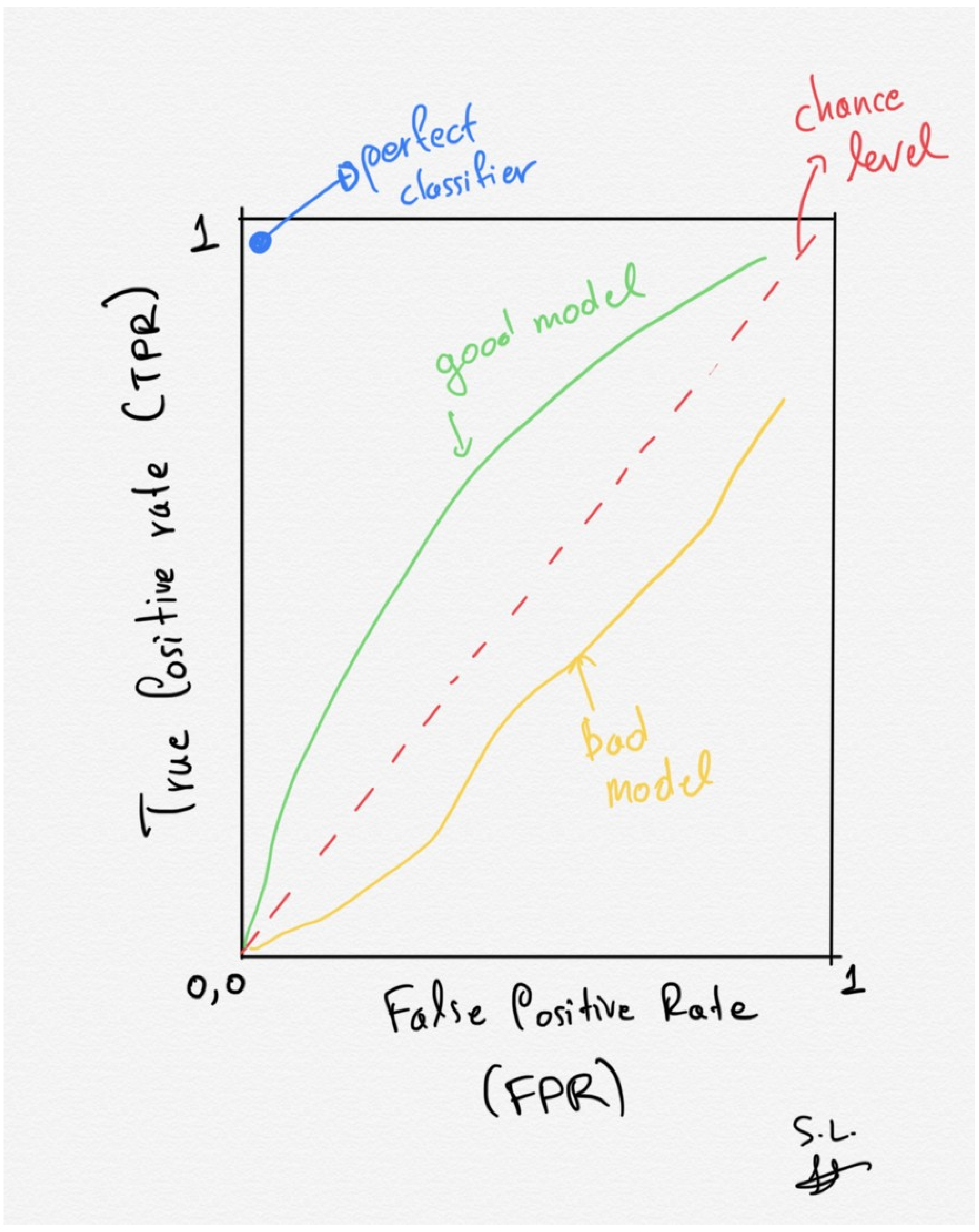
Published in Towards Data Science

Serafeim Loukas
Jun 10, 2020 · 10 min read · Member-only · Listen

ROC Curve explained using a COVID-19 hypothetical example: Binary & Multi-Class Classification tutorial

In this post, I clearly explain what a ROC curve is and how to read it. I use a COVID-19 example to make my point and I also speak about the confusion matrix. Finally, I provide Python code for plotting the ROC and confusion matrix from multi-class classification cases.



Handmade sketch made by the author.

1. Introduction

In 99% of the cases where a machine learning **classification model** is used, people report its **ROC** curve plot (as well as the AUC: area under the ROC) along with other metrics such as the accuracy of the model or the **confusion matrix.**

But **what is a ROC curve? What does it tell us? Why everyone is using them? How is it connected to the confusion matrix?** Continue reading and you will be able to answer all these questions.

1.1. ROC Definition

A **receiver operating characteristic curve (ROC) curve** is a plot that shows the diagnostic ability of a **binary classifier** as its **discrimination threshold** is varied.

Before I dig into the details, we need to understand that this discrimination **threshold** is not the same across different models but instead it is model-specific. For instance, if we have a Support Vector Machine (**SVC**) then this threshold is nothing more than the **bias** term of the **decision boundary** equation. *By varying the bias in a SVM model, we actually just change the position of the decision boundary.* Have a look at my previously published SVM article for more details about the SVM models.

The **ROC** curve is created by plotting the true positive rate (**TPR**) against the false positive rate (**FPR**) at various **threshold** settings. The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning. The false-positive rate is also known as the probability of false alarm and can be calculated as (1−specificity). **It tells us how well our model can distinguish the classes.**

A lot of terminology, right? Hold on a second, I will explain all these terms in the next section using an example that will make you always remember all these terms.

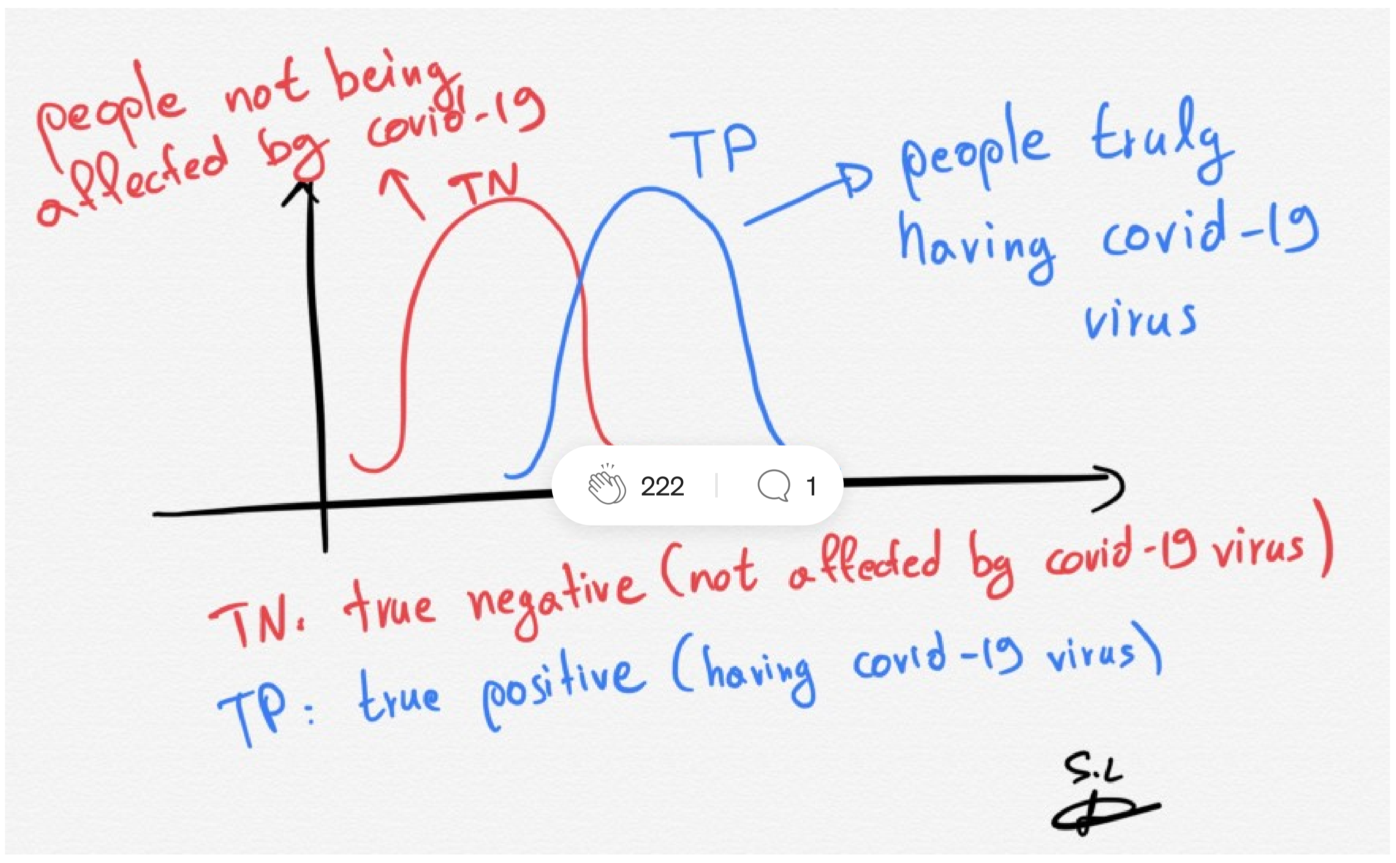
1.2. Terminology clearly explained (TP, TN, FP, FN)

The COVID-19 test example

Let's imagine that we have a **COVID-19 test** that is able within seconds to tell us if one individual is **affected** by the **virus** or **not**. So the **output of the test** can be either **Positive** (affected) or **Negative** (not affected) — we **have a binary classification case.**

Let's also **suppose** that we know the **ground truth** and that we have 2 populations:

- **a)** people that **are really affected** (**TP: True Positives, blue** distribution in the figure below) and
- **b)** people that are **not affected** (**TN: True Negatives, red** distribution in the figure below) — **binary classification case.**

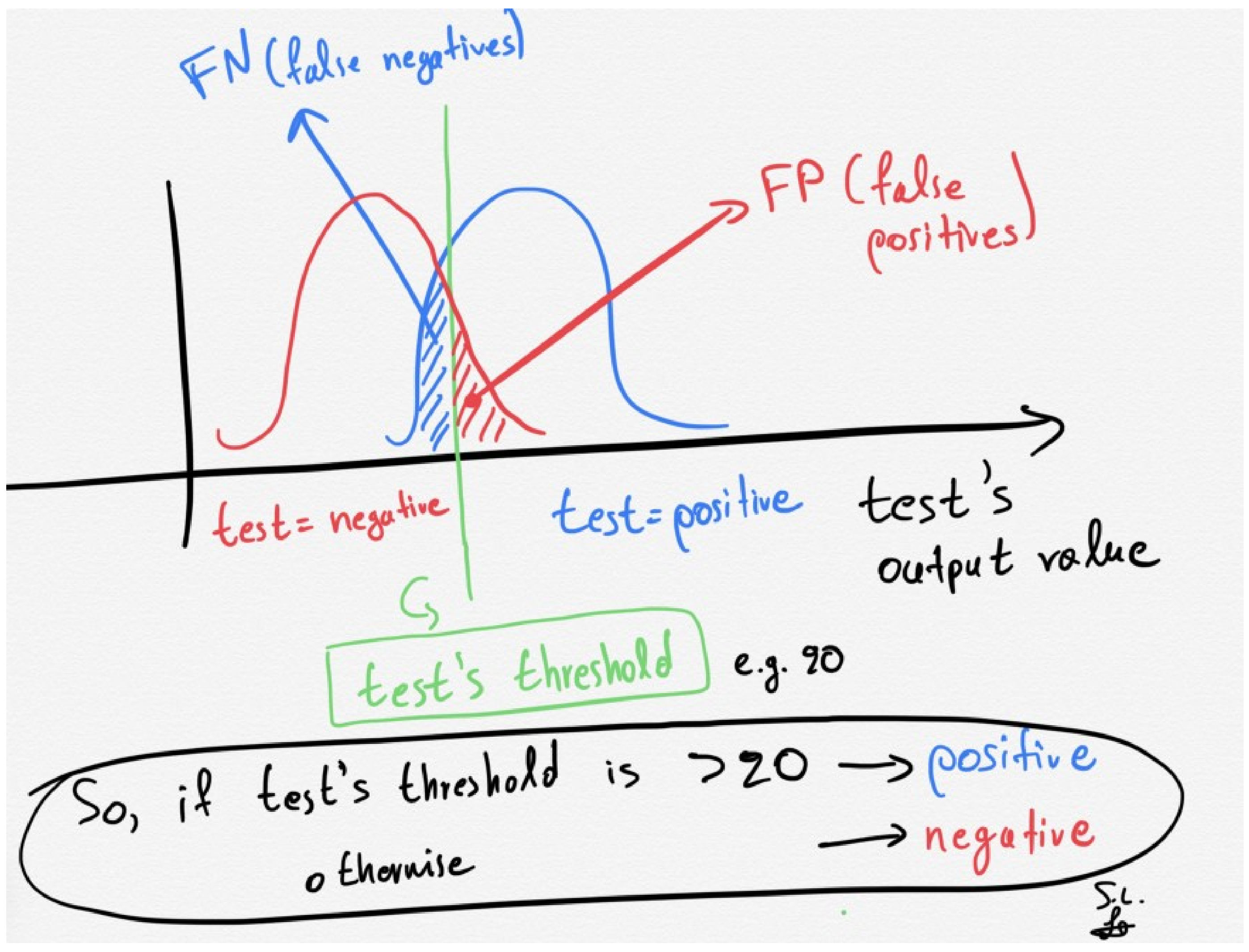

people not being affected by covid-19
TN
TP
people truly having covid-19 virus
TN: true negative (not affected by covid-19 virus)
TP: true positive (having covid-19 virus)
S.L
222
1

Handmade sketch made by the author. An example of 2 populations, one affected by covid-19 and the other not affected, assuming that we really know the ground truth.

So, as said before, we assume that we know the ground truth i.e. we really know who is sick and who is not.

Next, we **use our covid test** and we define a **threshold** (let's say 20). If the **test value** if **above the threshold** then the denote this person as **affected** (positive, covid-affected). On the other hand, if **test value** if **below the threshold** we then denote this person as **non-affected** (negative, covid-free). So, based on the **output of the test**, we can denote a person as **affected** (**positive**, **blue** population) or **not affected** (**negative, red** population).

To digest all these, have a look at the figure below.



Handmade sketch made by the author. An example of 2 populations, one affected by covid-19 and the other not affected, assuming that we really know the ground truth. Additionally, based on the output of the test, we can denote a person as affected (blue population) or not affected (red population).

> But there is a catch! Our test cannot be perfect!

Some people will be **wrongly** *miss-classified* as **positive** (covid-affected, we call this **False Positives** (**FP**)) or **negative** (covid-free, we call this **False Negatives** (**FN**)).

Summary:

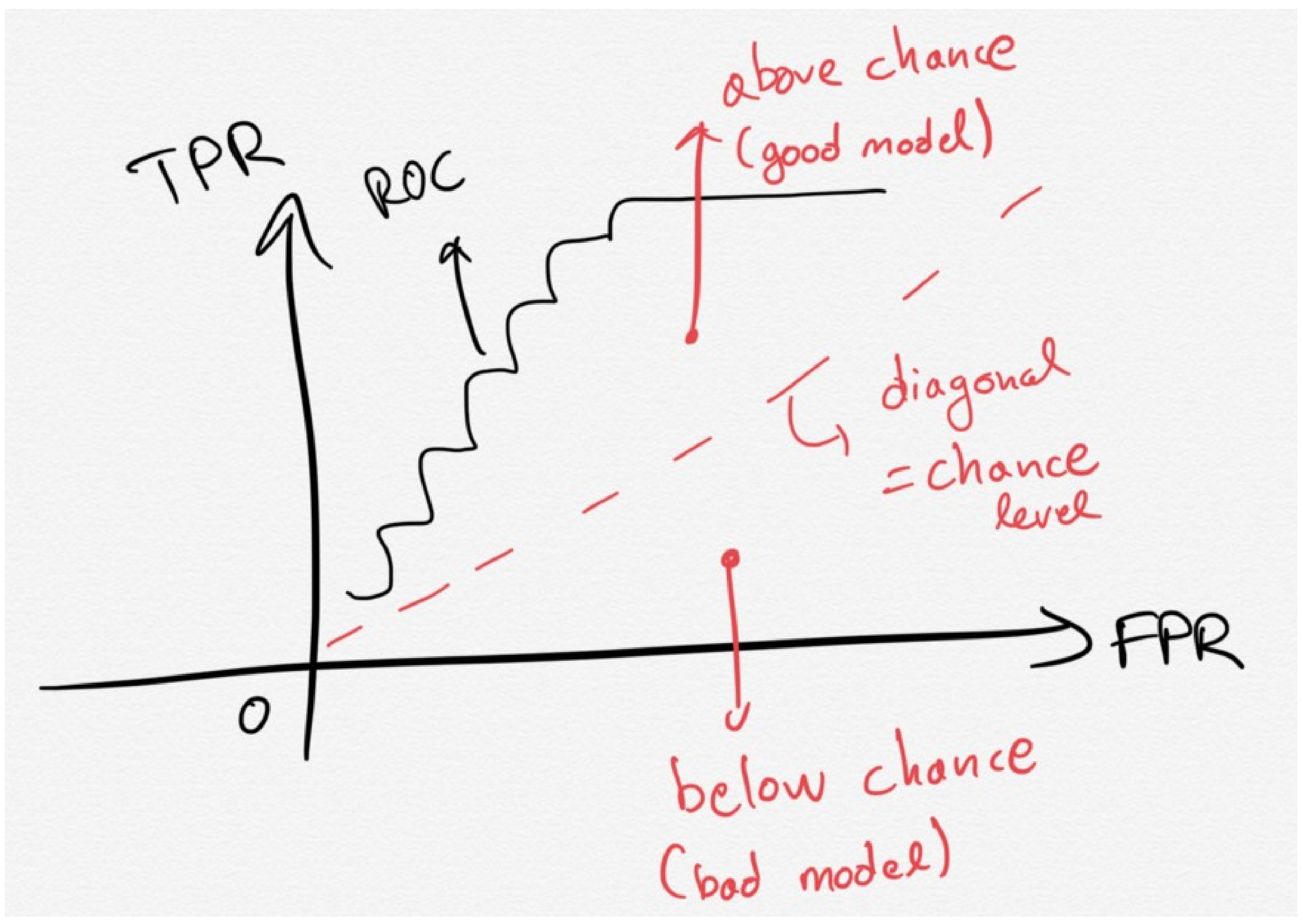
- **True Positives** (**TP**, **blue** distribution) are the people **that truly have** the COVID-19 virus.
- **True Negatives (TN, red** distribution) are the people **that truly DO NOT** have the COVID-19 virus.
- **False Positives (FP)** are the people that **are truly NOT sick** but **based on the test**, they were **falsely** (**False**) denoted as **sick** (**Positives**).
- **False Negatives (FN)** are the people that **are truly sick** but **based on the test**, they were **falsely** (**False**) denoted as **NOT sick** (**Negative**).
- For the **perfect case**, we would want **high values TP** and **TN** and **zero FP** and **FN — this would be the perfect model with the perfect ROC curve.**

1.3. The ROC Curve, AUC & the Confusion Matrix

The ROC

Now that we have understood the terms **TP, TN, FP, FN** let's look back again at the definition of the ROC curve.

> The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. In other words, the ROC curve shows the trade-off of TPR and FPR for different threshold settings of the underlying model.



Handmade sketch made by the author. A ROC curve showing the trade-off between the TPR and FPR for different threshold settings of the underlying model.If the curve is above the diagonal, the model is good and above chance (chance is 50% for a binary case). If the curve is below the diagonal, the model is really bad.

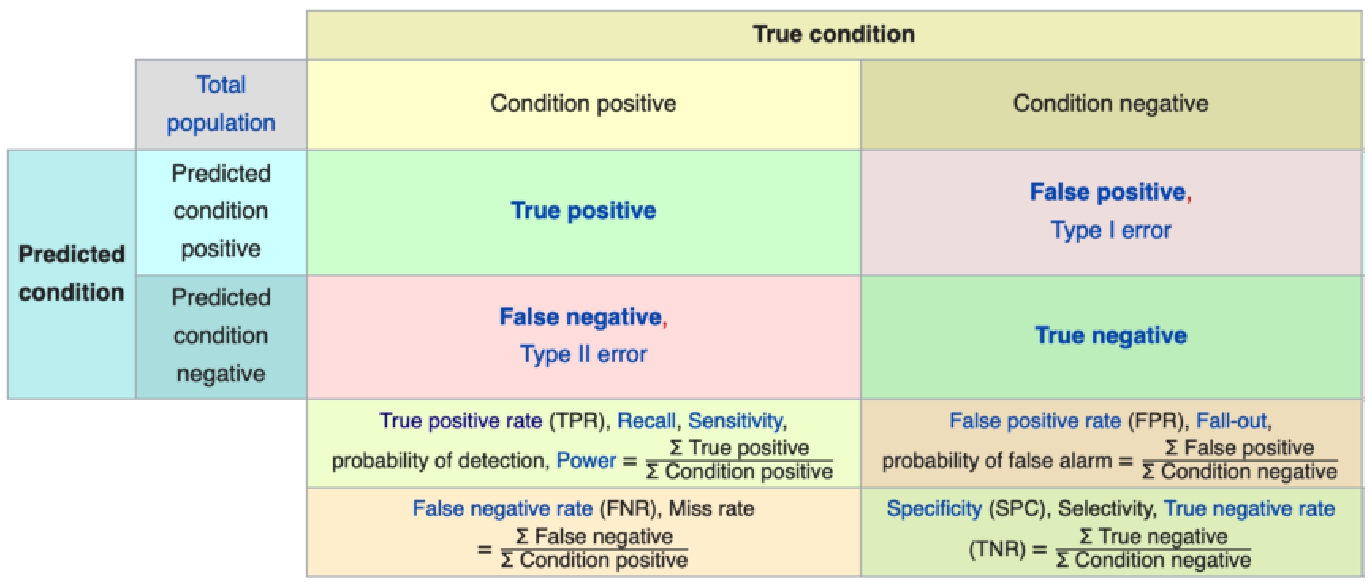
- If the curve is **above** the **diagonal**, the model is **good** and **above** chance (chance is 50% for a binary case). If the **curve is below** the **diagonal**, the **model** is **bad.**

The AUC

The **AUC (area under the curve) indicates** if the curve is **above** or **below** the **diagonal** (chance level). **AUC ranges** in value from **0 to 1**. A model whose predictions are 100% wrong has an **AUC** of 0.0 and one whose predictions are 100% correct has an **AUC** of 1.0.

The Confusion Matrix

Using all the above terms, we can also construct the famous **confusion matrix** that consists of these metrics and then we can compute the **True Positive Rate** and the **False Positive Rate** as shown in the figure below for **a binary classification case.**



		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$

Figure from Wikipedia.

Having estimated the **True Positive Rate** and the **False Positive Rate** (using the formulas from the above table), we can now plot the ROC curve. But one moment!

The **True Positive Rate** and the **False Positive Rate** are just 2 **scalars. How can we really have a curve in the ROC plot?**

> This is achieved by varying some threshold settings. The ROC curve shows the trade-off of TPR and FPR for different thresholds.

For instance, in the case of a **Support Vector Machine** (**SVC**) this **threshold** is nothing more that the **bias** term in the **decision boundary** equation. So, we **would vary this bias** (this would change the position of the decision boundary) and **estimate** the **FPR** and **TPR** for the given values of the **bias.**

To learn everything about SVMs have a look at this post.

1.4. ROC curve & Confusion Matrix for Multi-Class Classification Problems

The **ROC** curve is **only** defined for **binary classification problems**. However, there is a way to integrate it into **multi-class classification problems**. To do so, if we have **N** classes then we will need to define **several** models.

For example, if we have **N=3 classes** then we will need to define the following cases: **case/model 1** for **class 1 vs class 2, case/model 2** for **class 1 vs class 2, and case/model 3** for **class 1 vs class 3.**

Remember that in our Covid-19 test example, we had 2 possible outcomes i.e. affected by the virus (**Positives**) and not affected (**Negatives**). Similarly, in the multi-class cases, we again have to define the Positive and Negative outcomes.

In the multi-class case, for each case the **positive** class is the **second one:**

- *for **case 1**: "**class 1 vs class 2**", the **positive** class is class 2*
- *for **case 2**: "**class 2 vs class 3**", the **positive** class is class 3*
- *for **case** 3: "**class 1 vs class 3**", the **positive** class is class 3*

In other words, we can think of this as follows: We ask the classifier "Is this sample Positive or Negative?" and the classifier will predict the label (**positive** or **negative**). **The ROC will be estimated for each case 1,2,3 independently.**

The same holds for the confusion matrix. We would have one Confusion Matrix **for each case.**

2. Python working example

Now, you should know everything about the ROC curve and the confusion matrix as well as you should be familiar with the terminology such as TP, TN, FP, FN, TPR, FPR. Let's now build a **python** working **example**.

2.1 The classification model

In a previous post, I explained what an **SVC** is so here we will use such a model.

2.2 The dataset

In the iris dataset, we have **3 classes** of flowers and 4 features in total. So the classification problem is not a binary case anymore since we have 3 classes. However, the following code will estimate and plot the ROC curve for our **multi-class classification problem**.

To this end, the model will be used for **class 1 vs class 2, class 2 vs class 3 and class 1 vs class 3.** So, we have **3 cases** at the end and **within each case, the bias will be varied** in order to get the **ROC** curve of the **given case** — so, 3 ROC curves as output.

2.3 Example using Iris data and scikit-learn

The ROC curve & the AUC metric

```
import matplotlib.pyplot as plt
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc
from sklearn.multiclass import OneVsRestClassifier
from itertools import cycle
plt.style.use('ggplot')
```

Let's load the dataset, binarize the labels and split the data into training and test sets (to avoid ovefitting):

```
# Load the iris data
iris = datasets.load_iris()
```

```
X = iris.data
y = iris.target

# Binarize the output
y_bin = label_binarize(y, classes=[0, 1, 2])
n_classes = y_bin.shape[1]

# We split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_bin,
test_size= 0.5, random_state=0)
```

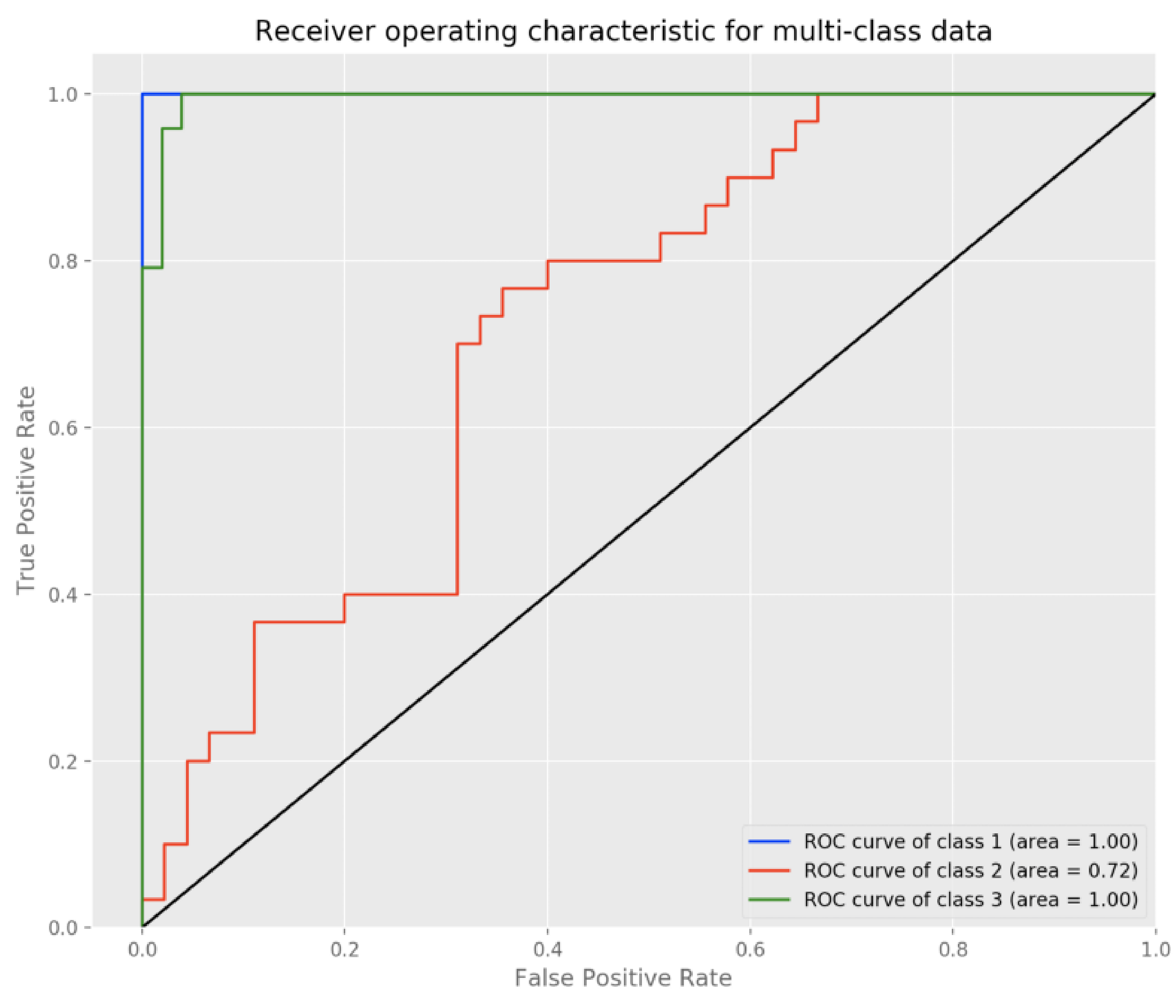
Finally, we build our model (SVC) and we estimate the ROC curve for the 3 cases: **class 1 vs class 2, class 2 vs class 3 and class 1 vs class 3.**

Each time the **positive** class is the **second one** i.e. *for **case 1: "class 1 vs class 2"**, the **positive** class is class 2, for **case 2: "class 2 vs class 3"**, the **positive** class is class 3 etc.*

```
#We define the model as an SVC in OneVsRestClassifier setting.
#this means that the model will be used for class 1 vs class 2, #class
2vs class 3 and class 1 vs class 3. So, we have 3 cases at #the end
and within each case, the bias will be varied in order to #get the ROC
curve of the given case - 3 ROC curves as output.
classifier = OneVsRestClassifier(svm.SVC(kernel='linear',
probability=True, random_state=0))

y_score = classifier.fit(X_train, y_train).decision_function(X_test)

# Plotting and estimation of FPR, TPR
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
colors = cycle(['blue', 'red', 'green'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=1.5, label='ROC curve of
class {0} (area = {1:0.2f})' ''.format(i+1, roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k-', lw=1.5)
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic for multi-class data')
plt.legend(loc="lower right")
plt.show()
```



Output figure produced my the python code presented above.

Reminder: The ROC curve shows the **trade-off** of **TPR** and **FPR** for **different threshold settings** of the underlying model. If the curve is **above** the **diagonal**, the model is **good** and **above** chance (chance is 50% for a binary case). If the **curve** is **below** the **diagonal**, the **model** is **bad.**

The **AUC (area under the curve) indicates** if the curve is **above** or **below** the **diagonal** (chance level).

The Confusion Matrix

Now let's also estimate the **confusion matrix.**

```
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

# Load the iris data
iris = datasets.load_iris()
X = iris.data
y = iris.target

# We split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
0.5, random_state=0)

# the model
classifier_svc = svm.SVC(kernel='linear',random_state=0)

# fit the model using the training set
classifier_svc.fit(X_train, y_train)

# predict the labels/classes of the test set
y_pred = classifier_svc.predict(X_test)
```

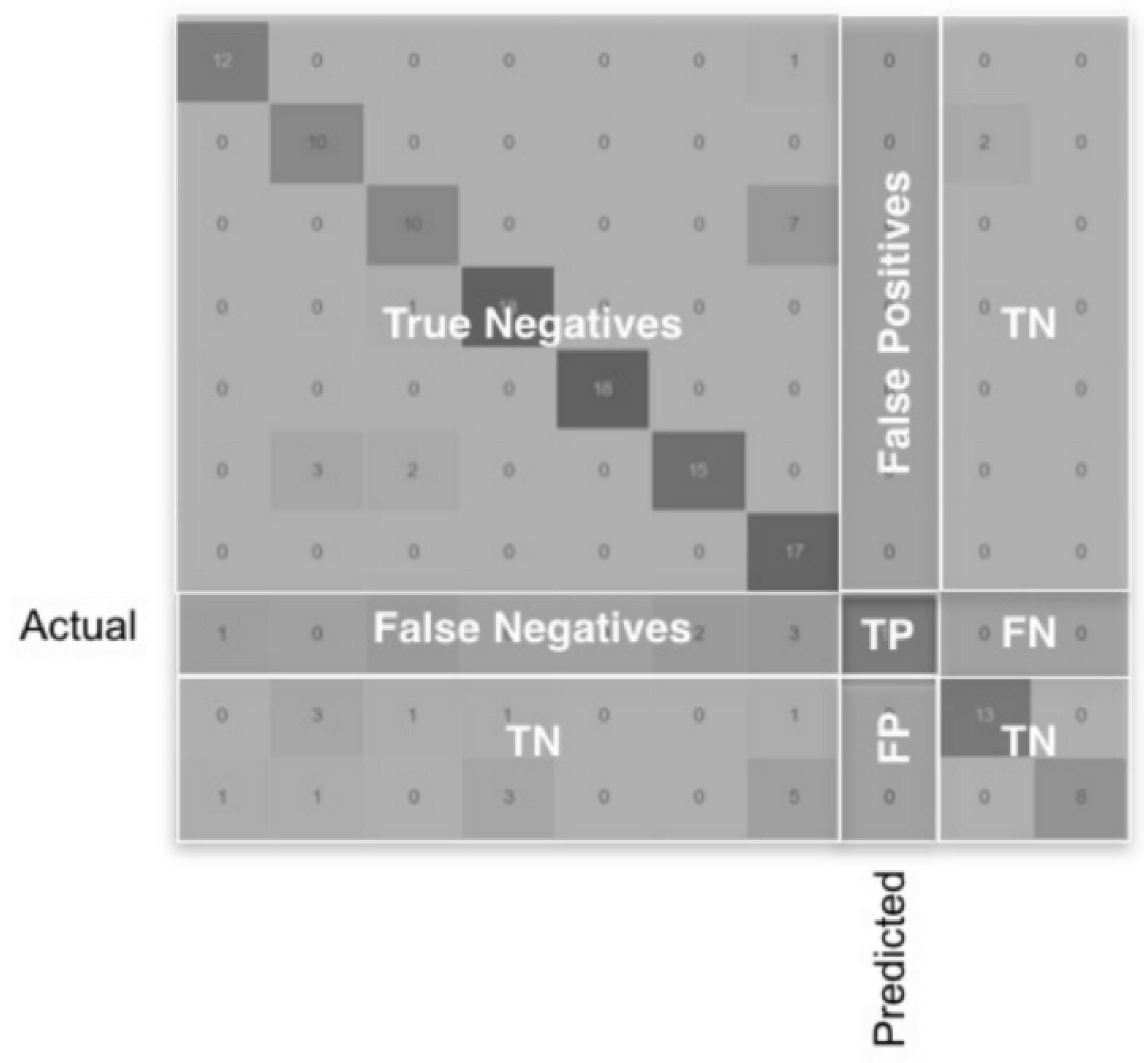
Having the **predicted** `y_pred` and the **ground** truth **labels** `y_test`, estimate the confusion matrix:

```
# build the confusion matrix
cnf_matrix = confusion_matrix(y_test, y_pred)

print(cnf_matrix)
#[[21  0  0]
# [ 0 29  1]
# [ 0  1 23]]
```

We can see that we can **really predict** the labels/class of **all 3 groups** (values mostly on the diagonal meaning high **True Positive** (**TP**) rates).

Reminder: The confusion matrix shows you the TN, TP, FN, FP. Values on the diagonal is the counting of TP so the higher these values the better the predictive ability of the model.



That's all folks! Hope you liked this article!

Latest posts

Time-Series Forecasting: Predicting Stock Prices Using Facebook's Prophet Model
Predict stock prices using a forecasting model publicly available from Facebook: The Prophet
towardsdatascience.com

Support Vector Machines (SVM) clearly explained: A python tutorial for classification problems...
In this article I explain the core of the SVMs, why and how to use them.

Additionally, I show how to plot the support...
towardsdatascience.com

PCA clearly explained — How, when, why to use it and feature importance: A guide in Python
In this post I explain what PCA is, when and why to use it and how to implement it in Python using scikit-learn. Also...
towardsdatascience.com

Everything you need to know about Min-Max normalization in Python
In this post I explain what Min-Max scaling is, when to use it and how to implement it in Python using scikit-learn but...
towardsdatascience.com

How Scikit-Learn's StandardScaler works
In this post I am explaining why and how to apply Standardization using scikit-learn
towardsdatascience.com

Stay tuned & support this effort

If you liked and found this article useful, **follow** me!

Questions? Post them as a comment and I will reply as soon as possible.

References

[1] https://en.wikipedia.org/wiki/Confusion_matrix

[2] https://en.wikipedia.org/wiki/Support_vector_machine

[3] https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[4] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Get in touch with me

- **LinkedIn:** https://www.linkedin.com/in/serafeim-loukas/
- **ResearchGate:** https://www.researchgate.net/profile/Serafeim_Loukas
- **EPFL profile:** https://people.epfl.ch/serafeim.loukas
- **Stack Overflow:** https://stackoverflow.com/users/5025009/seralouk