//

Machine Learning X

maXbox Starter 77 – Unified Machine Learning

!!!!!!!!!!!!!! final !!!!!!!!!!!!!!

Welcome to the Machine Learning concerning Confusion Matrix

In terms of the courtroom example, a type I error (false positive) corresponds to convicting an innocent defendant.

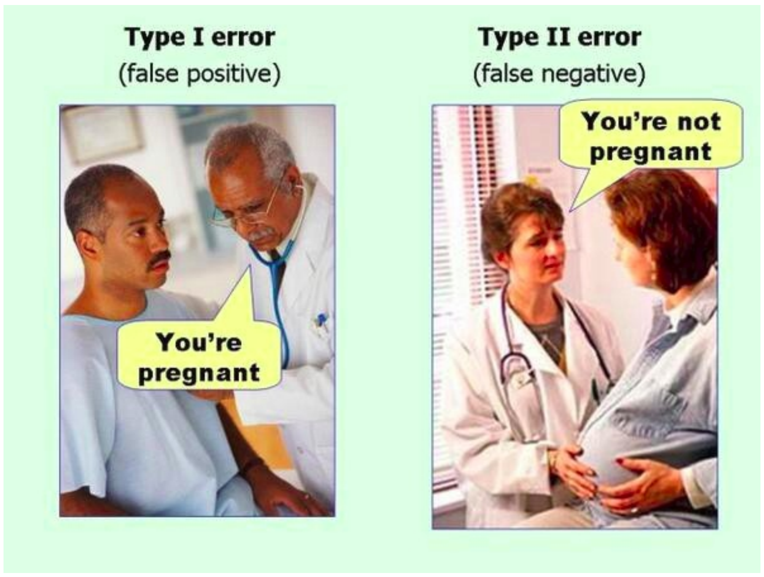
In terms of the courtroom example, a type II error (false negative) corresponds to acquitting a criminal.

.NET | WinUI | Angular | Azure | Blazor | ML.NET | SharePoint
S&S
10.2020
Deutschland 9,80 €
Österreich 10,80 €
Schweiz 19,50 sFr
windows .developer
www.windowsdeveloper.de
Machine Learning ganz praktisch
Maschinelles Lernen mit ML.NET
Der Weg von den Daten zur Vorhersage
Modernes WinUI
Cross-Platform-Apps mit der Uno Platform
Neues in Angular 10
Ruhe vor dem Sturm?
Blazor erhält Zustände
State Management in Blazor WebAssembly und Server
▶ 8
Azure Web Apps
Private Endpoints in der Public Cloud
▶ 56
IoT im DevOps-Zeitalter
Mit vollautomatisierten Pipelines zum Erfolg
▶ 82

A Machine Learning model cannot take this into account because errors for it have symmetrical costs, rarely in real life do we have equal costs for type I error and type II error, and whenever the costs are different, we need to choose the right threshold.

Statistically speaking, we want that our sample keeps the probability distribution of the population under a reasonable significance level. In other words, if we take a look at the histogram of the sample, it must be the same as the histogram of the population.

An example of this compromise exists in our domain, media monitoring (at least traditionally), many customers expect an almost perfect recall. They never want to miss an relevant article about the subjects they are interested in. However precision is not as important (albeit highly desirable), if you get a bit of noise in the article feed with false positives to sort out, you are usually fine.



“A sensitive test will correctly identify people with a disease. Sensitivity measures correct positive results. A specific test will accurately identify people without disease. Specificity measures correct negatives.”

“If a test is 90% sensitive (recall), it will correctly identify 90% of people who ARE infected — called a true positive,”. “However, 10% of people who are infected and tested would get a false negative result – they have the virus, but the test said they don’t.”

“If a test is 90% specific (precision), it will correctly identify 90% of people who are NOT infected – registering a true negative. “However, 10% of people who are not infected will test positive for the virus and receive a false positive.”

False positives of Covid:
https://www.spectator.co.uk/article/how-many-covid-diagnoses-are-false-positives-

First we need a library with modules. **ImageAI** is a Python library built to empower developers to build applications and systems with self-contained deep learning and Computer Vision capabilities using a few lines of straight forward code. But to use ImageAI you need to install a few dependencies namely:

- TensorFlow
- OpenCV
- Keras and ImageAI itself to install with $ pip3 install imageAI

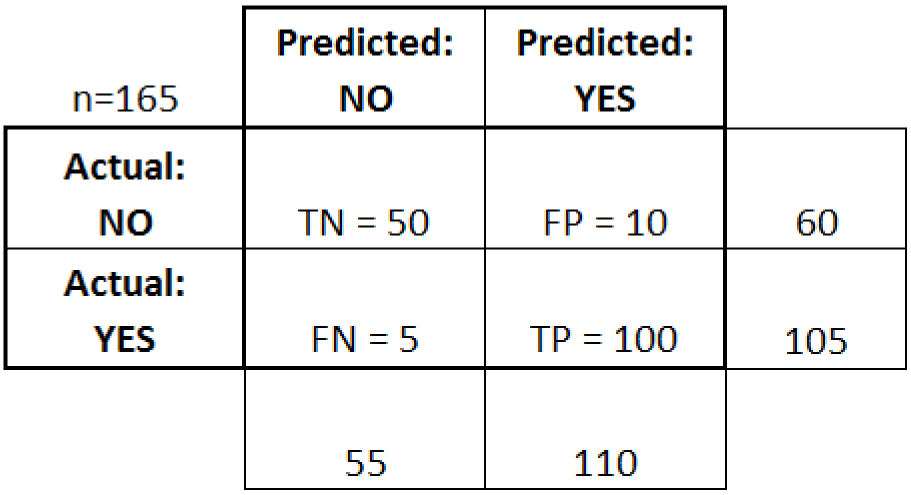
Now download the TinyYOLOv3 model file (33.9 MB) that contains a pretrained classification model that will be used for object detection:

https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON24/ImageDetector/yolo-tiny.h5/download

Then we need 3 necessary folders.
- input\
- models\yolo-tiny.h5
- output

Now put an image for detection in the input folder, for example: manmachine.jpg

Open now your preferred text editor for writing Python code (in my case maXbox) and create a new file *detector3.py* or some valid file name.



n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

In line 2 we import *ObjectDetection* class from the **ImageAI** library.

```
from imageai.Detection import ObjectDetection
```

As the next thing we create an instance of the class *ObjectDetection*, as shown in line 5 above:

```
detector = ObjectDetection()
```

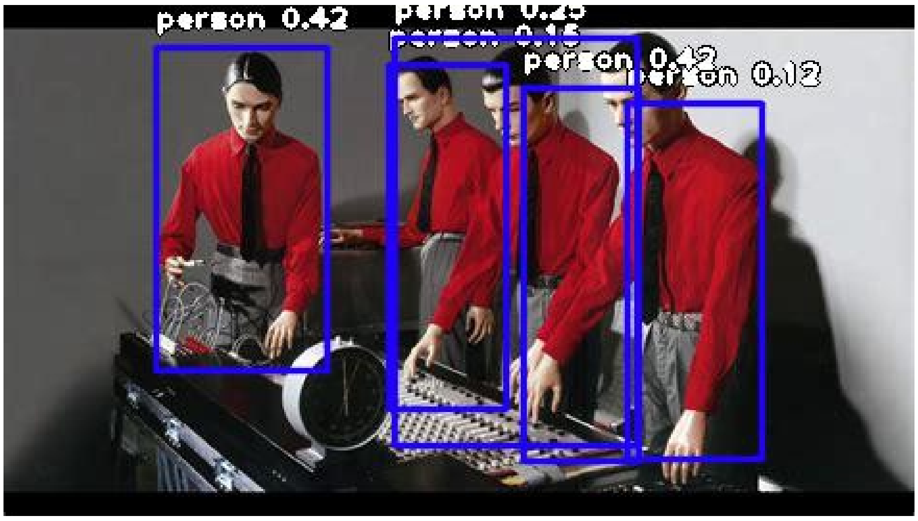
It goes on with the declaration of the previous created paths:

```
model_path = "./models/yolo-tiny.h5"
input_path = "./input/manmachine.jpg"
output_path = "./output/manmachineout.jpg"
```

In this tutorial, as I mentioned we'll be using the pre-trained TinyYOLOv3 model, so I use the *setModelTypeAsTinyYOLOv3*() function to load our model:

```
#using the pre-trained TinyYOLOv3 model,
detector.setModelTypeAsTinyYOLOv3()
detector.setModelPath(model_path)
```

```
#loads model from path specified above using the setModelPath() class method.

detector.loadModel()
```



To detect only some of the objects above, I will need to call the *CustomObjects* method and set the name of the object(s) we want to detect to through. The rest are False by default.
In our example, we detect customized only person. But it detects a fifth person as a combination of two persons sort of a man-machine in it ;-)).

```
custom= detector.CustomObjects(person=True,boat=True,laptop=True,bottle=True)
detections = detector.detectCustomObjectsFromImage(custom_objects=custom, \
                    input_image=input_path, output_image_path=output_path,\
                                             minimum_percentage_probability=10)
```

Another reason for the custom instance is that I can define the threshold to find things. Unlike normal *detectObjectsFromImage*() function, this needs an extra parameter which is "custom_object" which accepts the dictionary returned by the *CustomObjects*() function. In the sample below, we set the detection function to report only detections we want:

```
for eachItem in detections:
    print(eachItem["name"] , " : ", eachItem["percentage_probability"])
```

```
person  :  11.582126468420029
person  :  15.317463874816895
person  :  25.40428638458252
person  :  41.63033664226532
person  :  42.076510190963745
integrate image detector compute ends...
elapsedSeconds:= 19.9037681927473

Stop with Return Key--->
```

So we get 5 objects as persons with color frames and corresponding probability. But we only have 4 persons, there must be 1 false positive; on the other side there's no false negative as a 100% recall. Funny but really effective man-machine detection!

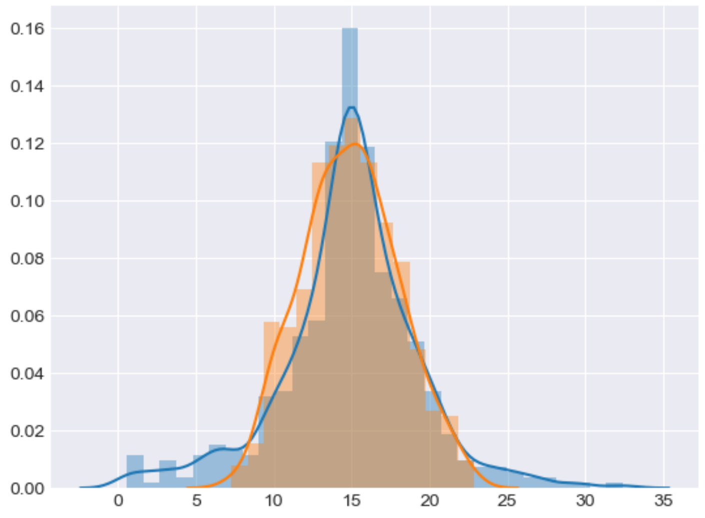
This is the function (detectCustomObjectsFromImage) that performs object detection task after the model as loaded. It can be called many times to detect objects in any number of images.

Script detector2.py

```
# ImageAI is a Python library built to empower Computer Vision
from imageai.Detection import ObjectDetection
#Using TensorFlow backend.

detector = ObjectDetection()

model_path = "./models/yolo-tiny.h5"
input_path = "./input/manmachine.jpg"
output_path = "./output/manmachineout.jpg"

#using the pre-trained TinyYOLOv3 model,
detector.setModelTypeAsTinyYOLOv3()
detector.setModelPath(model_path)

detector.loadModel()

#detection = detector.detectObjectsFromImage(input_image=input_path, \
#                                              output_image_path=output_path)

custom= detector.CustomObjects(person=True,boat=True, laptop=True,bottle=True)
detections = detector.detectCustomObjectsFromImage(custom_objects=custom, \
                  input_image=input_path, output_image_path=output_path,\
                                          minimum_percentage_probability=10)

for eachItem in detections:
    print(eachItem["name"] , " : ", eachItem["percentage_probability"])

print('image detector compute ends...')

#https://stackabuse.com/object-detection-with-imageai-in-python/
#https://github.com/OlafenwaMoses/ImageAI/releases/download/1.0/yolo-tiny.h5
#https://imageai.readthedocs.io/en/latest/detection/index.html
```

There are 80 possible objects that you can detect with the ObjectDetection class, and they are as seen below (not ordered).

```
  person, bicycle, car, motorcycle, airplane, bus,train, truck, boat, traffic
light, fire hydrant, stop_sign, parking meter, bench, bird, cat,dog, horse,
sheep, cow, elephant, bear, zebra, giraffe,  backpack,umbrella, handbag, tie,
suitcase,  frisbee, skis, snowboard, sports ball, kite,baseball bat, baseball
glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, fork,
knife, spoon, bowl,  banana, apple, sandwich,range,broccoli, carrot, hot dog,
pizza, donot, cake, chair, couch, potted plant, bed, dining table, toilet, tv,
laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink,
refrigerator, book, clock, vase, scissors, teddy bear, hair dryer, toothbrush.
```



The script and data can be found:

http://www.softwareschule.ch/examples/992_detector21_wget_integrate2.htm

http://www.softwareschule.ch/examples/detector2.htm
https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON24/ImageDetector/

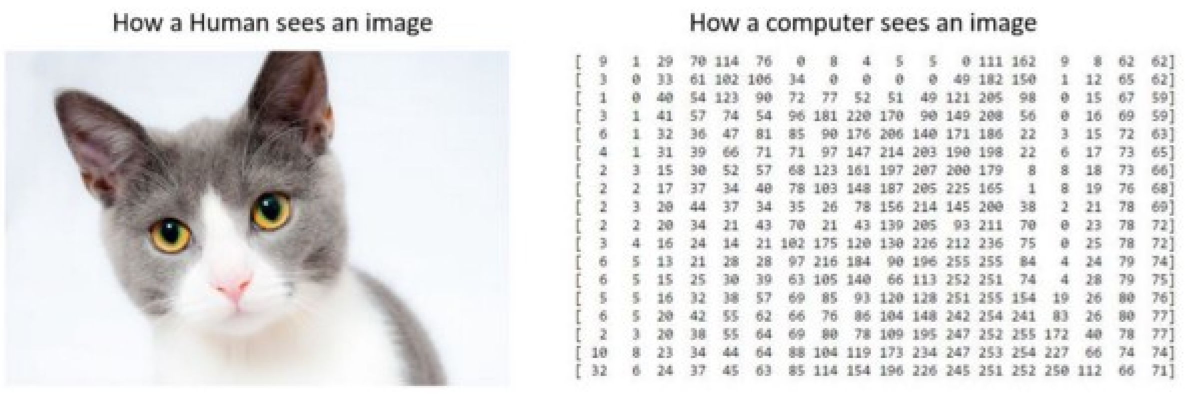
http://www.softwareschule.ch/examples/classifier_compare2confusion2.py.txt

Author: Max Kleiner

Ref:
    http://www.softwareschule.ch/box.htm
    https://scikit-learn.org/stable/modules/
    https://realpython.com/python-histograms/

    https://imageai.readthedocs.io/en/latest/detection/index.html

Doc:
    https://maxbox4.wordpress.com



```
[   9   1  29  70 114  76   0   8   4   5   5   0 111 162   9   8  62  62]
[   3   0  33  61 102 106  34   0   0   0   0  49 182 150   1  12  65  62]
[   1   0  40  54 123  90  72  77  52  51  49 121 205  98   0  15  67  59]
[   3   1  41  57  74  54  96 181 220 170  90 149 208  56   0  16  69  59]
[   6   1  32  36  47  81  85  90 176 206 140 171 186  22   3  15  72  63]
[   4   1  31  39  66  71  71  97 147 214 203 190 198  22   6  17  73  65]
[   2   3  15  30  52  57  68 123 161 197 207 200 179   8   8  18  73  66]
[   2   2  17  37  34  40  78 103 148 187 205 225 165   1   8  19  76  68]
[   2   3  20  44  37  34  35  26  78 156 214 145 200  38   2  21  78  69]
[   2   2  20  34  21  43  70  21  43 139 205  93 211  70   0  23  78  72]
[   3   4  16  24  14  21 102 175 120 130 226 212 236  75   0  25  78  72]
[   6   5  13  21  28  28  97 216 184  90 196 255 255  84   4  24  79  74]
[   6   5  15  25  30  39  63 105 140  66 113 252 251  74   4  28  79  75]
[   5   5  16  32  38  57  69  85  93 120 128 251 255 154  19  26  80  76]
[   6   5  20  42  55  62  66  76  86 104 148 242 254 241  83  26  80  77]
[   2   3  20  38  55  64  69  80  78 109 195 247 252 255 172  40  78  77]
[  10   8  23  34  44  64  88 104 119 173 234 247 253 254 227  66  74  74]
[  32   6  24  37  45  63  85 114 154 196 226 245 251 252 250 112  66  71]
```