Machine Learning

|Confusion Matrix

Prof. Dr. Mostafa Elhosseini

https://youtube.com/@ProfElhosseiniTechVerse

Agenda

- Confusion Matrix
- False Positive – Type I errors
- Accuracy
- Precision
- Recall
- Precision & Recall

Confusion Matrix

- To compute the confusion matrix, you first need to **have a set of predictions**, so they can be compared to the **actual targets**.
- You could make predictions on the test set, but let's keep it untouched for now (remember that you want to use the test set only at the very end of your project, once you have a classifier that you are ready to launch).
- Instead, you can use the **cross_val_predict()** function

Confusion Matrix

- The **cross_val_predict** function performs cross-validation and returns the predicted labels or values for each sample in the input data.
- It **splits** the data into multiple folds, **trains** the model on a subset of the folds, and **makes predictions** on the remaining fold. This process is repeated for each fold, and the predictions for all samples are collected.
- The output of cross_val_predict is an **array of predictions**, where each element corresponds to the predicted label or value for a particular sample.
- It is useful when you want to obtain the predicted labels or values for each sample in the dataset, which can be used for further analysis or evaluation.

Confusion Matrix

```
from sklearn.model_selection import cross_val_predict

y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_train_5, y_train_pred)
```

```
array([[53892,   687],
       [ 1891,  3530]])
```

- Each **row** in a confusion matrix represents an **actual class**, while each **column** represents a **predicted class**.

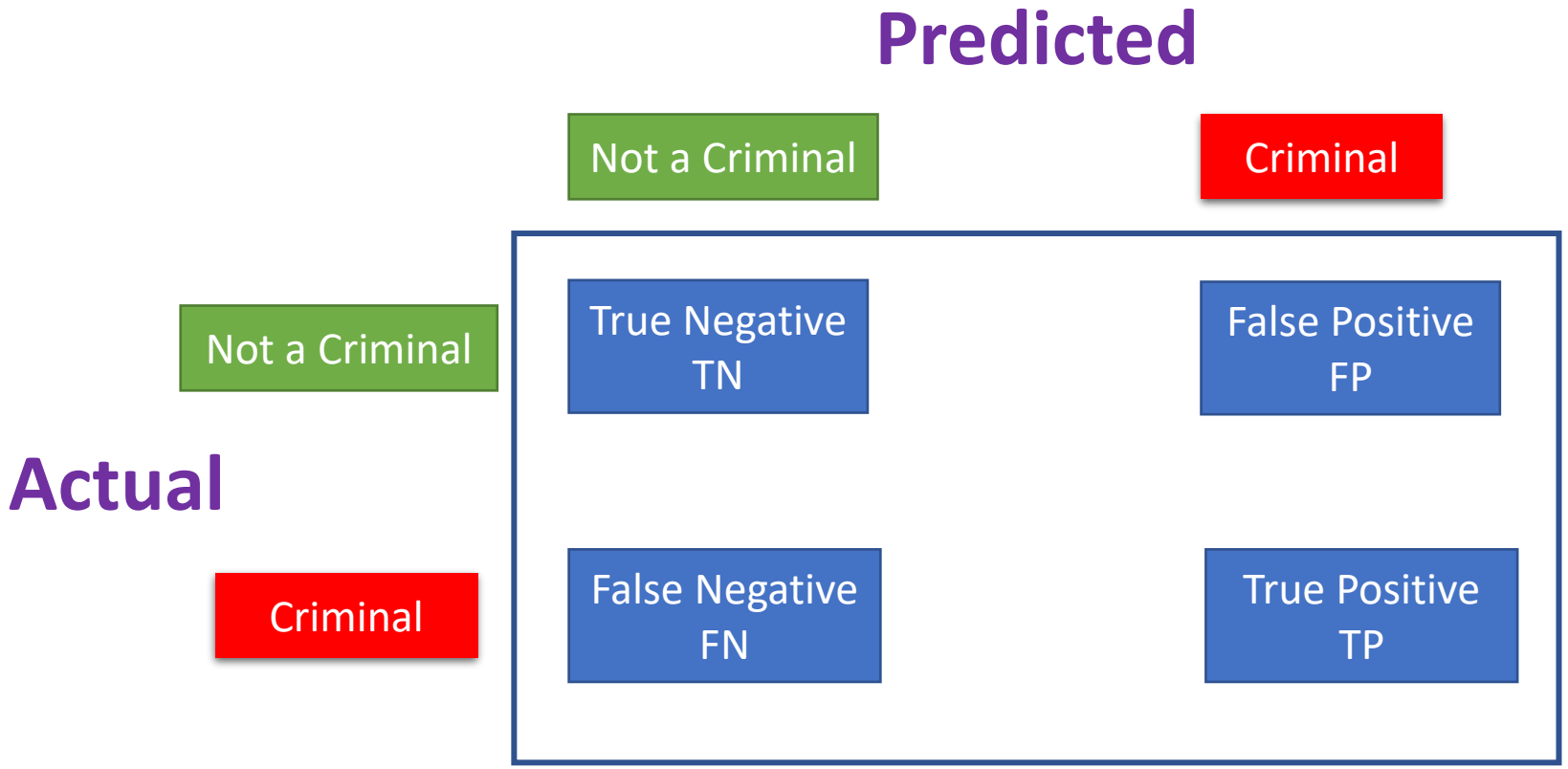
Confusion Matrix

- The **first row** of this matrix considers **non-5 images** (the negative class): 53,892 of them were correctly classified as non 5s (they are called **true negatives**), while the remaining **687** were wrongly classified as 5s (**false positives**, also called **type I errors**).
- The **second row** considers the images of 5s (the **positive class**): 1,891 were wrongly classified as non-5s (**false negatives**, also called **type II errors**), while the remaining 3,530 were correctly classified as 5s (true positives).
- A perfect classifier would have only true positives and true negatives, so its confusion matrix would have nonzero values only on its main diagonal (top left to bottom right)

Confusion Matrix

- **True Positive**
  - When model predicts the positive class and the actual class was also positive. It means the model correctly predicted the positive class
- **True Negative**
  - When the model predicts the negative class and the actual class was also negative. It means the model correctly predicted the negative class.
- **False Positive**
  - model incorrectly predicted negative class as a positive class
- **False Negative**
  - It means the model incorrectly predicted positive class as a negative class

Confusion Matrix



False Positive – Type I errors

- A **False Positive**, also known as a **Type I error**, occurs when a model incorrectly predicts a positive outcome when the actual outcome is negative.
- The term "Type I error" comes from the field of **hypothesis testing** in statistics, where it has a similar meaning.
- In **hypothesis testing**, a Type I error occurs when a **null hypothesis** (H_0) is rejected when it is actually true. The null hypothesis typically represents the default or assumed state, while the **alternative hypothesis (H_1)** represents the claim, or the effect being tested.

False Positive – Type I errors

- In machine learning, we can draw an **analogy** between hypothesis testing and binary classification:
  - The **null hypothesis** (H_o) is equivalent to the **negative class** (e.g., "not spam" or "benign").
  - The alternative hypothesis (H_1) is equivalent to the positive class (e.g., "spam" or "malignant").
- A **false positive** (**Type I error**) in machine learning occurs when the model predicts the positive class (alternative hypothesis) when the true class is actually negative (null hypothesis).

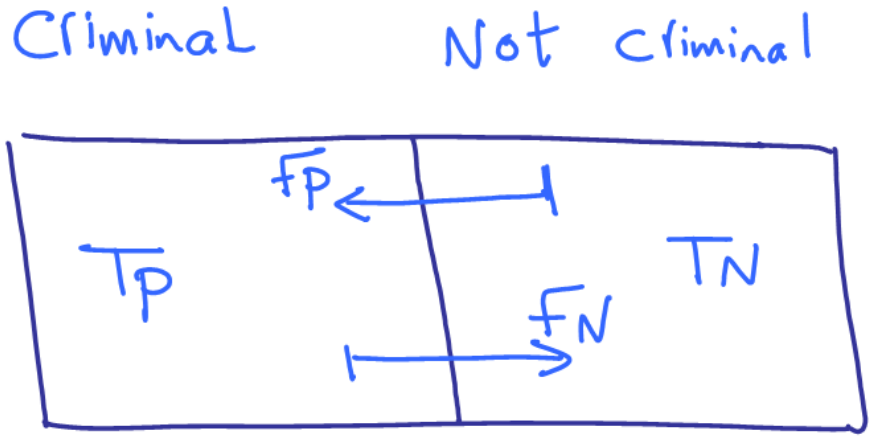
False Positive – Type I errors

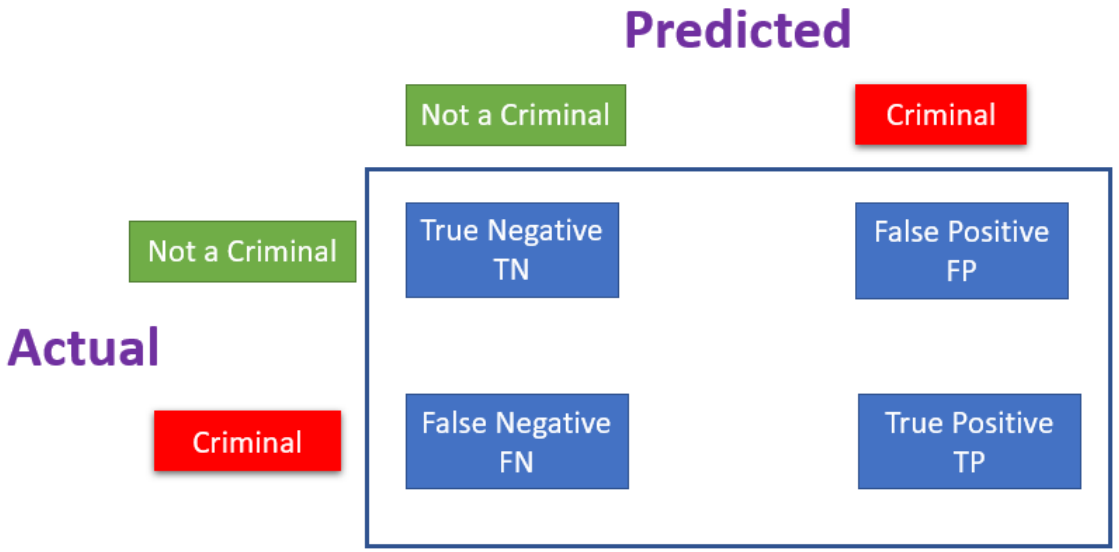
- For example, consider an email spam classifier:
  - H_0 (null hypothesis): The email is not spam (negative class).
  - H_1 (alternative hypothesis): The email is spam (positive class).
- If the classifier predicts that an email is spam (positive class) when it is actually not spam (negative class), this is a false positive or Type I error.

False Positive – Type I errors

- The term "Type I error" is used in machine learning because it aligns with the concept from hypothesis testing, emphasizing the idea of incorrectly rejecting the null hypothesis (i.e., incorrectly predicting the positive class).
- It's important to note that the complimentary error, a false negative or Type II error, occurs when the model incorrectly predicts a negative outcome when the actual outcome is positive (i.e., failing to reject the null hypothesis when the alternative hypothesis is true).

Accuracy

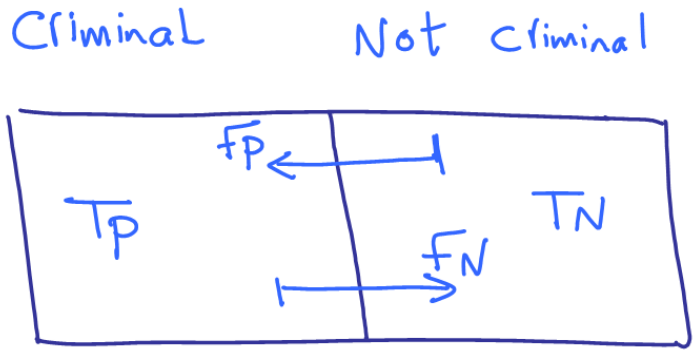




- $\boldsymbol{Accuracy} = \dfrac{TP+TN}{TP+TN+FP+FN}$

Precision

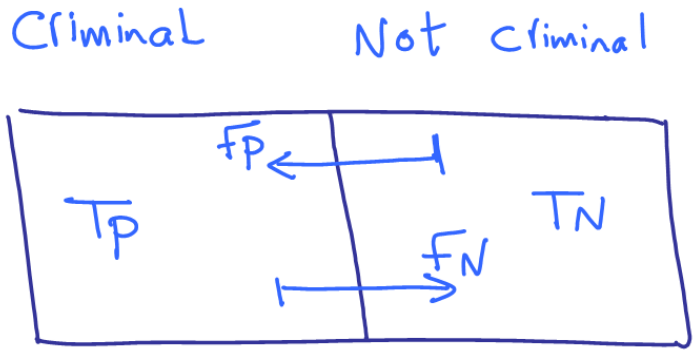
- Precision tells us how many samples were actual positive out of all positive predicted samples
- Precision tells us how many passengers were criminals out of all criminals predicted samples
- Accuracy of the positive predictions
- $Precision = \frac{TP}{TP+FP}$



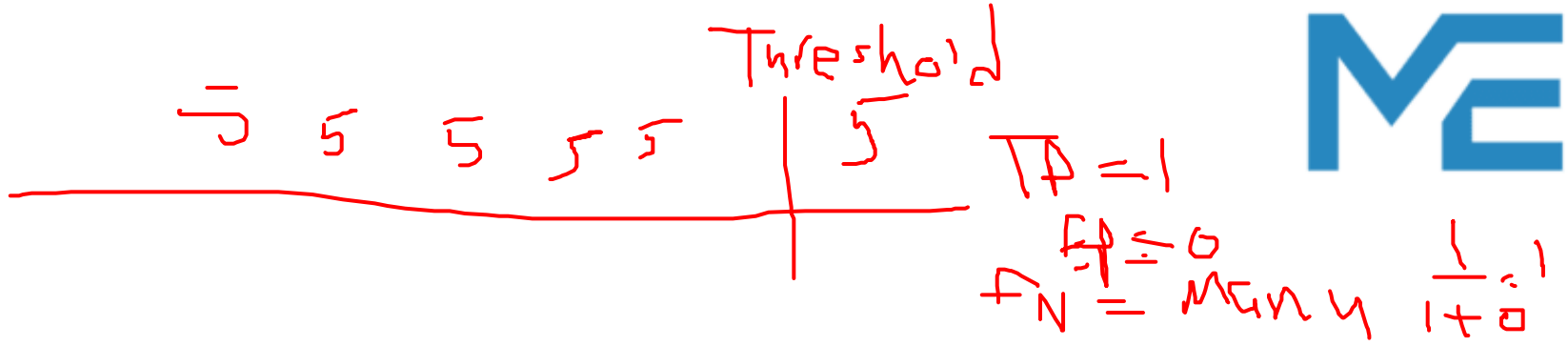
Precision

- A model with high precision indicates
  - Few false positive (“false alarm”)
  - Not many “Not a criminal” were classified as criminal

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}}$$



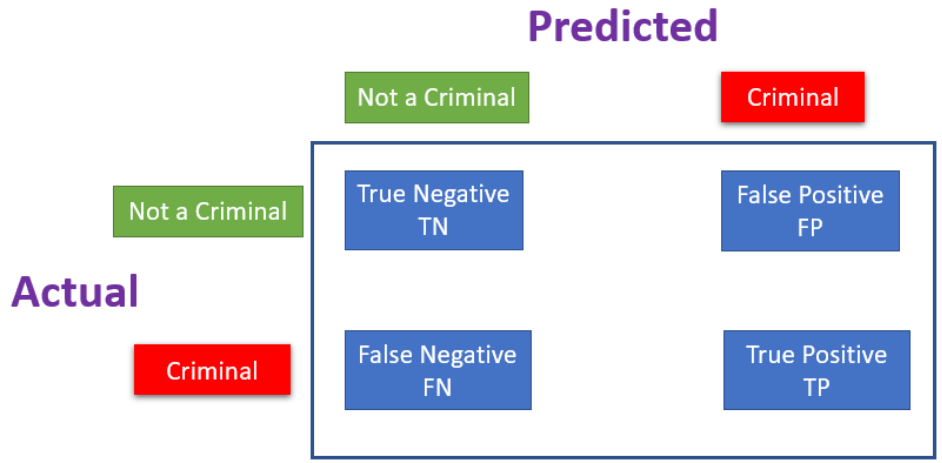
Recall



- A trivial way to have **perfect precision** is to create a classifier that always makes negative predictions, except for one single positive prediction on the instance it's most confident about.
- If this one prediction is correct, then the classifier has 100% precision (precision = 1/1 = 100%). Obviously, such a classifier **would not be very useful**, since it would ignore all but one positive instance. So, precision is typically used along with another metric named recall

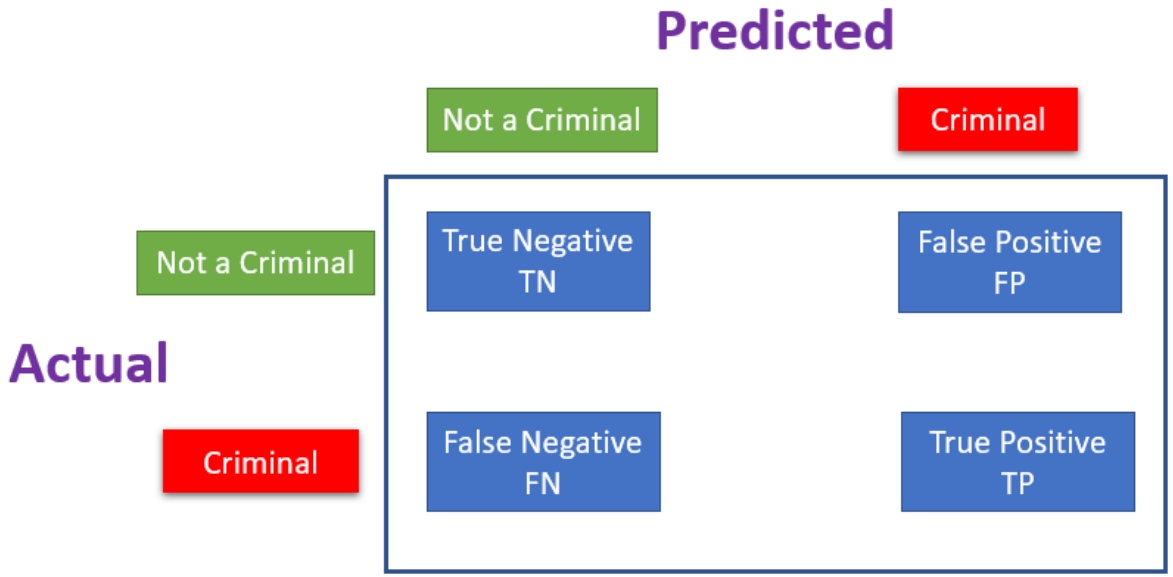
Recall

- Recall tells us how many positive samples were detected out of all actual positive samples
- which is the probability that an actual positive will test positive
- Recall (sensitivity) tells us how many criminals were detected out of all actual criminals
- Sensitivity = True Positive Rate TPR
- $Recall = \frac{TP}{TP+FN}$



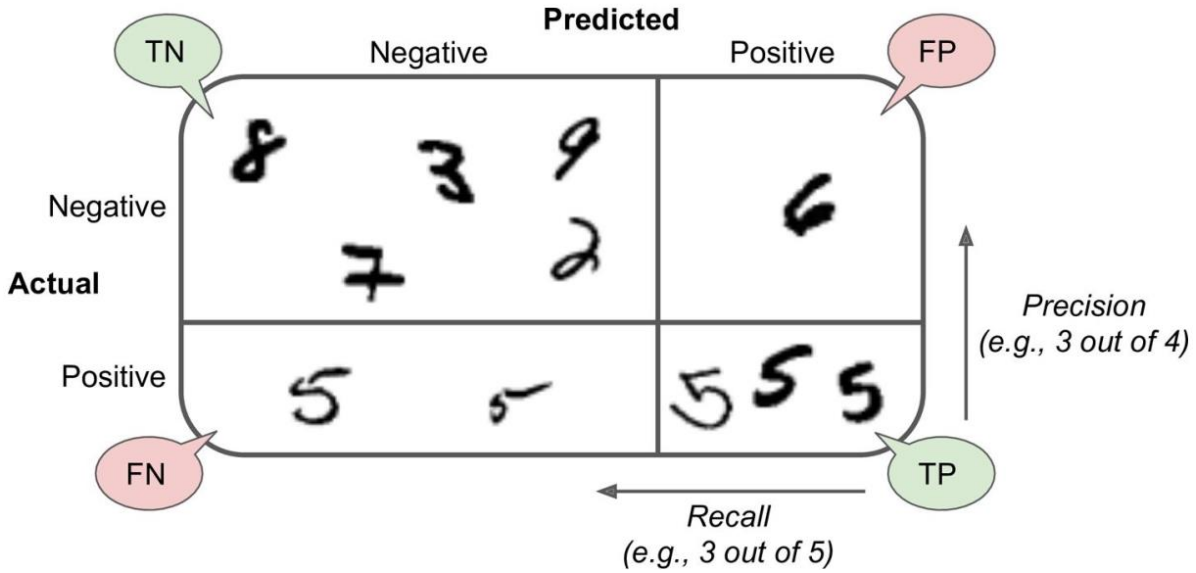
Recall

$$Recall = \frac{TP}{TP + FN}$$



- A High Recall means
  - Correctly classified most criminals

Precision & Recall



- $Precision = \frac{TP}{TP+FP}$
  - Accuracy of the positive predictions
- $Recall = \frac{TP}{TP+FN}$
  - Sensitivity = True Positive Rate TPR

Next Lecture

- Which is more important – Precision / Recall?