Classification

Classification is the process of identifying and grouping objects or ideas into predetermined categories. In machine learning (ML), classification is used in predictive modeling to assign input data with a class label. For example, an email security program/model tasked with identifying spam might use natural language processing (NLP) to classify emails as being "spam" or "not spam."

The input of classification models can be text, images, speech etc., and the output of the model are labels of the different classes/categories. Two basic types of classification are **Binary Classification** and **Multi-class Classification**. In Binary Classification, the model predicts between two different classes. An example: A person detection model that predicts whether a human is present in an image or not (2 labels - 'present', 'not present'). Multi-class Classification models predict between more than two classes. An example: A classification model predicting the different handwritten digits in the MNIST Dataset (10 labels - 0 to 9).

Confusion Matrix

We often use Classification Accuracy as a measure to evaluate how well our model is performing.

$$\text{classification accuracy} = \text{correct predictions} / \text{total predictions}$$

Although classification accuracy is a great place to start, it often encounters problems in practice. The main problem with classification accuracy is that it hides the detail you need to better understand the performance of your classification model. There are two examples where you are most likely to encounter this problem:

1. When your data has more than 2 classes. With 3 or more classes you may get a classification accuracy of 80%, but you don't know if that is because all classes are being predicted equally well or whether one or two classes are being neglected by the model.
2. When your data is imbalanced i.e., you have many more examples/data for one class than the others. You may achieve accuracy of 90% or more, but this is not a good score if 90 records for every 100 belong to one class and you can achieve this score by always predicting the most common class value.

This is where a **Confusion Matrix** helps! A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

Let's consider an example of a Binary Classification to better understand the confusion matrix. Let the model predict whether a photograph contains a dog or not (Labels - **'positive'** indicating there is a dog and **'negative'** indicating there is no dog in the photograph). We have a test dataset of 10 records with expected outcomes and a set of predictions from our classification algorithm.

SINo	**Expected**	**Predicted**
1	Positive	Negative
2	Positive	Negative
3	Positive	Positive
4	Positive	Positive
5	Positive	Positive
6	Negative	Negative
7	Negative	Positive
8	Negative	Negative
9	Negative	Negative
10	Negative	Negative

The confusion matrix would look something like this!

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

		Predicted	
		Positive	Negative
Actual	Positive	3	2
	Negative	1	4

Here,

TP - **True Positive** - # Positive Labels correctly predicted as Positive
FN - **False Negative** - # Positive Labels incorrectly predicted as Negative
TN - **True Negative** - # Negative Labels correctly predicted as Negative
FP - **False Positive** - # Negative Labels incorrectly predicted as Positive

We see that the confusion matrix gives us insight not only into the errors being made by the classifier but more importantly the types of errors that are being made. For example, in the above example, we see that the model makes more errors in predicting when there is actually a dog in the photograph vs when there isn't. It is this breakdown that overcomes the limitation of using classification accuracy alone.

We can also calculate other classification metrics!

Other Evaluation Metrics

Recall/Sensitivity - measure to check correctly positive predicted outcomes out of the total number of positive outcomes.

$$recall = TP / TP + FN$$

Precision - measure to check how many outcomes are actually positive outcomes out of the total positively predicted outcomes.

$$precision = TP / TP + FP$$

Accuracy (same as classification accuracy above)

$$\text{accuracy} = TP + TN \,/\, TP + FN + TN + FN$$

F1 Score - harmonic mean of Precision and Recall and captures the equal contribution of both of them.

$$F1\ \text{score} = 2*\text{precision}*\text{recall} \,/\, \text{precision} + \text{recall}$$

How do we know when to use which metric?

By default we use Accuracy. The others are more problem-specific.

Example 1: a *Stock Market Crash Prediction* model that foresees whether the stock market is going to crash or not. The model aims at reducing the False Negatives (FN) i.e., not predicting a market crash when there actually is one, is bad! Since the Recall metric takes into account FN, we consider Recall in this application.

Example 2: a *Spam Mail Prediction* model that differentiates between spam and safe mails. The model aims at reducing False Positives (FP) otherwise we might miss out on an important mail if its wrongly predicted as spam! Since the Precision metric takes into account FP, we consider Precision in this application.

Neither *precision* nor *recall* is necessarily useful alone, since we rather generally are interested in the overall picture. F1 score takes into account the model's ability over two attributes, which makes it a more robust gauge of model performance. Also, it is able to relay true model performance when the dataset is imbalanced which was one of the disadvantages of the *accuracy* metric!