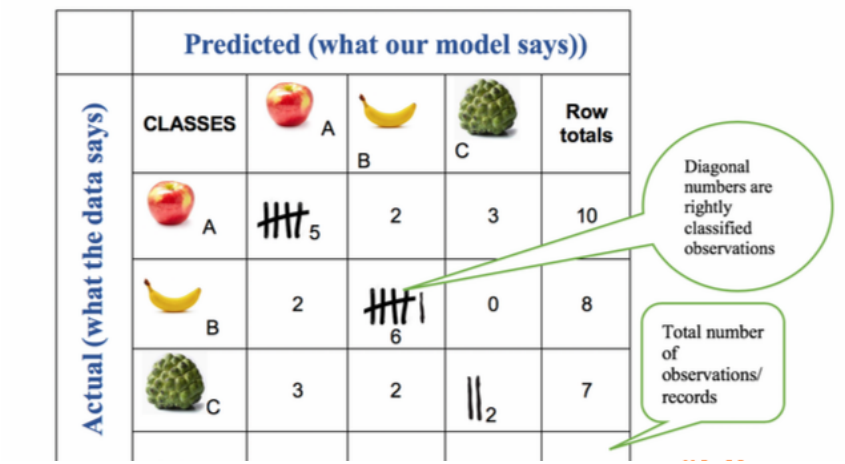
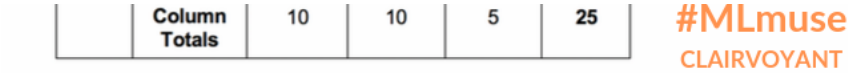
Churning the Confusion out of the Confusion Matrix

This article is about the confusion matrix and its uses in machine learning.

RekhaMolala Follow
Sep 23, 2019 · 7 min read





Picture 1: **CONFUSION MATRIX**

Many beginners in the field of machine learning often get confused with this very important topic and end up writing a piece of code without even understanding its implementation and use. What if we say that this is as simple as A, B, C and we DO NOT need to remember any formulae to calculate the most needed recall, precision, accuracy, etc.? Understanding a confusion matrix gives us the ability to decide which metric is really important for a problem that we might be dealing with and to interpret the performance of a classification model better. This article is all about it.

Confusion Matrix:

A ‘Confusion Matrix’ is a consolidation of the number of times a model gives a correct or an incorrect inference or simply, the number of times a model rightly identifies the truth (actual classes) and the number of times it gets confused in identifying one class from another.

Above picture (*picture 1*) is a representative confusion matrix for a multi-class (here, 3 classes) classification problem which we shall be using for calculating certain metrics in this article.

Applications of confusion matrix

A confusion matrix helps measure the performance of a classification problem with the help of different metrics that can be calculated from it.

As an example, let's assume that our model predicted the fruit types- Apple (A), Banana(B) and Custard apple(C) and gave the confusion matrix (as in picture 1). It gives us the below information about our problem and the model:

- Total number of observations in the data (i.e., fruits) = 25
- Number of A, B, C type fruits the data has = 10, 8, 7 respectively
- Model saw more instances of apples (A = 10) than other fruit types
- The diagonal observations are the true positives of each class i.e., the number of times the model correctly identified A as A, B as B and C as C

- All other non-diagonal observations are incorrect classifications made by the model

Let's rebuild the same confusion matrix in python so that the metric values can be validated at each step.

```
from sklearn import metrics
```

```
actuals = ['A','A','A','A','A','A','A','A','A','A','B','B','B','B','B','B','B','B','C','C','C',
predicted = ['A','A','A','A','A','B','B','C','C','C','A','A','B','B','B','B','B','B','A','A','A
```

```
len(actuals), len(predicted)
```

```
(25, 25)
```

```
conf_matrix = metrics.confusion_matrix(actuals,predicted)
conf_matrix
```

```
array([[5, 2, 3],
       [2, 6, 0],
       [3, 2, 2]])
```

Accuracy:

Tells us how often the model can be correct.

Accuracy of the model = No. of correct predictions by the model / total observations
= Sum (diagonal values of confusion matrix) / Total observations
= (5+6+2) /25
= 13/25 = 0.52

Check in python using sklearn:

Model accuracy

```
metrics.accuracy_score(actuals,predicted)
```

```
0.52
```

```
print('Accuracy = sum of diagonal scores over total observations: ',
      (conf_matrix.trace()/conf_matrix.sum())[0,0])
```

```
Accuracy = sum of diagonal scores over total observations:  0.52
```

Although accuracy is an important metric, it is not always adequate to measure a model's performance and is best relied upon when there is no class imbalance (i.e., when the proportion of instances of all the classes is the same) which rarely is the case in any real scenario.

When there is a class imbalance, accuracy can be misleading too. Let's assume that we have a sample of 100 fruits of which 90 are apples and 5 are bananas and 5 are custard apples. Even if our model predicts all the fruits as apples, the accuracy will be 90% while the truth is that our model is not actually doing a great job!

Hence, the need for other metrics.

Accuracy is calculated for the model as a whole but recall and precision are calculated for individual classes. We use macro or micro or weighted scores

of recall, precision and F1 score of a model for multiclass classification problems. Let's check them out.

Before we manually compute these metrics, let us first get the classification report (using sklearn package) that shows class wise metrics and averaged metrics of the model.

Classification Report showing class-wise recall, precision, F1-score

```
print(metrics.classification_report(actuals,predicted))
```

	precision	recall	f1-score	support
A	0.50	0.50	0.50	10
B	0.60	0.75	0.67	8
C	0.40	0.29	0.33	7
micro avg	0.52	0.52	0.52	25
macro avg	0.50	0.51	0.50	25
weighted avg	0.50	0.52	0.51	25

Recall (also called True Positive Rate or Sensitivity):

As an example, let us calculate recall for class B.

Recall for B would be, from all the actual instances of class B (banana), how often it correctly predicts as B (banana)

Recall for B = out of the total Bs (bananas = 8) in the data, how many Bs did the model identify correctly (6).

So, it would be 6/8 = 0.75 (which is the same as in the classification report above for class B)

Precision:

Precision for class B is: how often is the model correct when it predicts as B?

As earlier, let's calculate precision for class B.

Precision for B: out of the total observations predicted as Bs (10) by the model, how many are correct Bs (6). So, it would be 6/10 = 0.6

F1 — score:

Simply the harmonic mean of precision and recall of a particular class.

F1 score of class B will be (2 x Recall x precision)/(Recall + Precision)
=0.75*0.6*2/(0.75+0.6) = 0.67

For a binary classification problem, we will know our positive class (like spam, fraud, has cancer, etc) and hence we can focus on the scores for the positive class. But, for a multiclass classification problem, apart from the class-wise recall, precision, and f1 scores, we check the macro, micro and weighted average recall, precision and f1 scores of the whole model. These scores help in choosing the best model for the task at hand.

Micro-average scores:

In the Micro-average method, you sum up the individual class's true positives, false positives, and false negatives of the system for different sets and then apply them to get the score. For the fruit's confusion matrix, micro-average recall score is calculated as below (same as in the classification report above):

Micro average recall

sum of true positives of class A,B,C over sum of true positives of class A,B,C and False positives of A,B,C

```
(5+6+2)/((5+6+2)+(5+4+3))
```

```
0.52
```

Macro-average scores:

It is the simple mean of scores of all classes. So, macro- average recall is the mean of the recalls of classes A, B and C.

Weighted average scores:

The sum of the scores of all classes after multiplying their respective class proportions. For example, weighted average recall is calculated as below:

Misclassification Rate (also known as “Error Rate”)

Tells us how often the model can go wrong and is equivalent to 1 minus the Accuracy.

Other important metrics for binary classification problems:

The below metrics are calculated especially for 'binary' classification problems as the false positives and false negatives do not change once we identify our positive class (i.e., a class that we are interested in predicting). Nevertheless, the roots of a confusion matrix come from the 'errors table' of type-1 and type-2 errors.

Let's take the example of a binary classification problem. If we were to predict a fraudulent transaction, with an outcome of 'yes' or 'no', with 'yes' denoting 'fraud' and 'no' denoting 'a non-fraudulent transaction', then yes=fraud will be our positive class as we would be interested in detecting a fraudulent transaction more than a normal transaction.

False Positive Rate (caused by Type I Error): tells us how often the model predicts 'yes' for an actual 'no'. Is it important to keep this error low? It may be a yes or a no and depends on the scenario as illustrated below:

Sometimes, this error might translate to a simple case where a person is predicted to have some bacterial infection while actually that might not be the case. The medication to treat simple bacterial infections might not be very dangerous and is believed to have very mild or no side effects on the patient. So, in such cases, we might not worry much about the Type I error.

But things can get complicated and serious if the same error happens in a scenario where a person *not* suffering from cancer is diagnosed to have cancer. This can be really dangerous and sometimes fatal due to the high doses of radiation and chemotherapy that a patient can be exposed to.

True Negative Rate (or Specificity)is a metric that tells us how often the model predicts 'no' for an actual 'no'. It is equivalent to 1 minus False Positive Rate.

False Negative Rate (caused by Type II Error): Number of items the model wrongly predicted 'no' out of the total actual 'yes'. This metric is especially important in most binary classification problems, as it tells us the frequency with which a positive instance is wrongly identified as negative. For example, if a cancer patient is wrongly diagnosed as not having cancer, that individual would either go undiagnosed or misdiagnosed. Similarly,

identifying a fraudulent transaction as non-fraudulent can cause several serious repercussions for a bank. Hence, whenever we intend our model to be a diagnostic aid, we would always want this metric to be as low as possible.

Ending Notes

When a simple confusion matrix can spit out so much information about our model, why not get it right? Also, the confusion matrix tells us the classes that are more susceptible to misclassification. This hints at generating more key features that can help the model in identifying such classes better. Here are a few other sources for further exploration on the topic.

- http://ceur-ws.org/Vol-710/paper37.pdf
- https://www.researchgate.net/publication/275224157_A_Review_on_Evaluation_Metrics_for_Data_Classification_Evaluations

Machine Learning | Analytics | Confusion Matrix | Classification | Mlmuse