Beginner-friendly ML school: Performance Metrics

Viviana Acquaviva
(CUNY)
vacquaviva@citytech.cuny.edu

Beyond accuracy

Imagine building an algorithm to look for a rare objects (e.g. a variable star).
The data you have have 1,000 instances (stars).
Of those, 10 belong to the "interesting" class.
(the data set is imbalanced: the target values are not distributed uniformly).
You feel lazy and submit an algorithm that says that there are no variable stars.

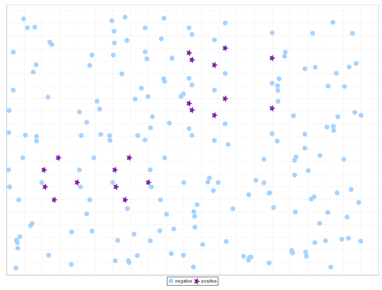
What is its accuracy (% of correct predictions)?

Picking a performance metric

"The accuracy paradox"

Accuracy = % of correct predictions

meaningless in unbalanced data set

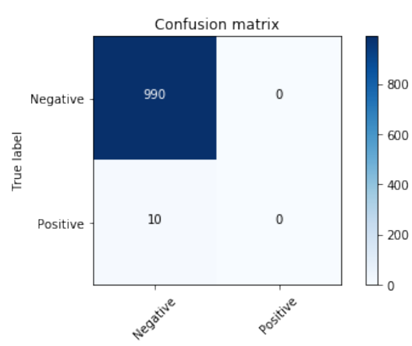


For our dataset :
Accuracy = 990/1000 = 99.0%!

Evaluating classifiers performance: beyond accuracy

For a binary classifier where the "true" or desired class members are defined to be positive, every metric is enclosed by four numbers:

TP = True Positives	**0**
TN = True Negatives	**990**
FP = False Positives	**0**
FN = False Negatives	**10**



Accuracy = % of correct predictions **TP+TN/(TP+TN+FP+FN)**

Alternative metrics

precision = percentage of correct positive classifications TP/(TP + FP)

recall = percentage of "caught" positive instances = TP/(TP+FN)

For our lazy classifier

TP = 0; TN = 990; FP = 0; FN = 10

Will have undefined precision and 0 recall
so we'll be able to know that something is amiss.

Alternative metrics

Often we (astronomers) talk of precision
as purity, or 1 – precision as contamination

Recall is IMO best visualized as completeness

A useful one is F1 = weighted avg of precision/recall

The best metric can only be decided by you on the basis
of the science you want to do!

A few words about ROC and AUC

ROC (receiving operator characteristic) is
a plot of TPR (True Positive Rate) vs FPR (False Positive Rate)

TPR = TP/(TP+FN) = recall

FPR = FP/(TN+FP)
(swap F->T in recall)

How can this be a plot?

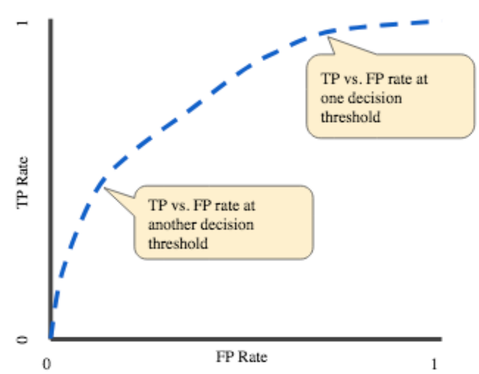
The trick is to calculate it for different thresholds
of what it means for an object to belong to the positive class
(as opposed to the “standard” 0.5)

A few words about ROC and AUC

ROC (receiving operator characteristic) is
a plot of TPR (True Positive Rate) vs FPR (False Positive Rate)

TPR = TP/(TP+FN) = recall

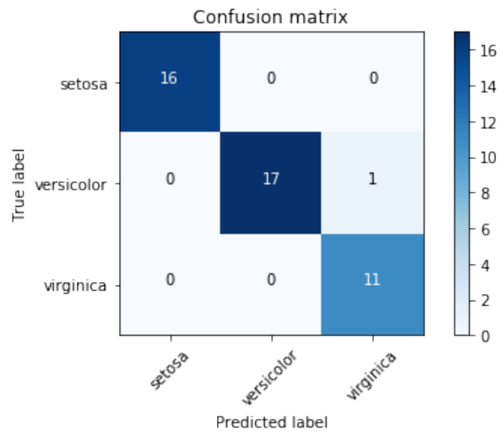
FPR = FP/(TN+FP)
(swap F->T in recall)



AUC =
area under
the curve

What happens for non-binary classifiers?

The confusion matrix can be built normally



Note: in this case accuracy is still the % of correct predictions, but precision and recall require you to define a "positive" class and do one-vs-all (plus choose micro/macro averaging if you want one number)

Summary of how to build a ML model and what we'll see today

1. **Choose a class of model (aka a machine learning algorithm) by importing the appropriate estimator class from Scikit-Learn.**
2. **Choose model hyperparameters by instantiating this class with desired values. Alternatively: optimize hyperparameters (later).**
3. **Arrange data into a features matrix and target vector, if necessary.**
4. **Split the learning set into training/test using k fold cross validation.**
5. **Fit the model to your data by calling the ``fit()`` method.**
6. **Apply the Model to new data: predict labels for unknown data using the ``predict()`` method.**
7. **Estimate the performance (averaged over the k folds) by using one of the metrics in the "metrics" method of the model instance (accuracy, precision, recall … or custom).**
8. **Figure out what is not working out: Check training vs test score, learning curves, diagnose high variance vs high bias and decide how to move forward.**