Evaluating Model Performance & Hyperparameter Tuning

Courtesy: Thanks to Dikscha Sapra for DPhi Data Science Bootcamp

Why evaluate Performance?

- To understand how good is our model
- To compare it with other models
- To generalise how good our model will perform on new data

How to evaluate your Model?

CLASSIFICATION

DISCRETE VARIABLES

Why do we need different methods for both of them?

REGRESSION

CONTINUOUS VARIABLES

CLASSIFICATION ACCURACY

- Most basic metrics to evaluate our model

- Intuition: $$\frac{\text{All correctly predicted values}}{\text{All predicted values}} \times 100$$

Actual Labels:

Predicted Labels:

Task at hand - Separate Yellow and pink balls

Colors predicted by our model

Correctly predicted = 4. Total = 6
Accuracy = 66.7%

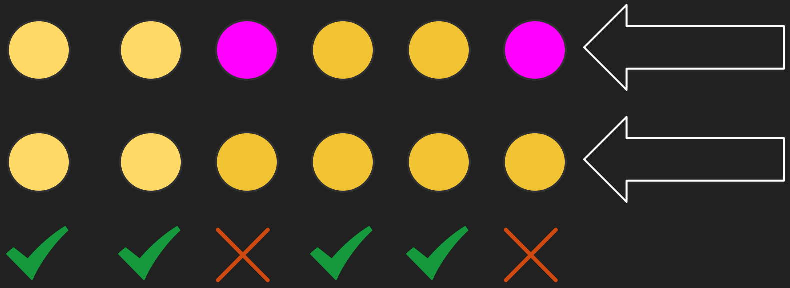
Why Accuracy is not a good metric?

Baseline Model

1. Base Model with which we are comparing our performance
2. Several ways to consider a Baseline Model
3. We are considering a model which classifies all labels as that of majority class

Actual Labels:

Predicted Labels:



Task at hand - Separate Red and Yellow balls

Majority class is Yellow (4/6), so all are predicted yellow

Accuracy = 4/6 *100 = 66.7

Our model is not better than the baseline model (Which is technically just a nonsense model!)

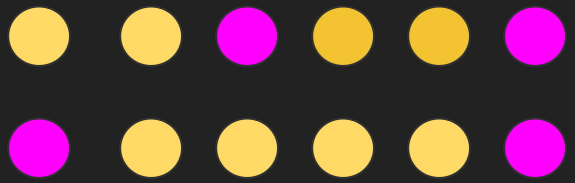
So what to do?

Confusion Matrix:

For the sake of generalisation, let us call yellow as positive labels, and pink as negative labels.

Actual Labels:

Predicted Labels:



FN TP FP TP TP TN

Break down the data into four categories

a. Actual =Positive, Predicted = Positive (True Positive)

b. Actual =Positive, Predicted = Negative (False Negative)

c. Actual =Negative, Predicted = Negative (True Negative)

d. Actual = Negative, Predicted = Positive (False Positive)

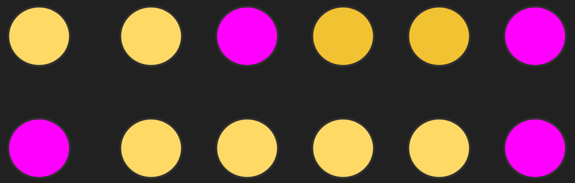
		Actual	
		Positive	Negative
Predicted	Positive	(TP) 3	(FP) 1
	Negative	(FN) 1	(TN) 1

Confusion Matrix:

For the sake of generalisation, let us call yellow as positive labels, and pink as negative labels.

Actual Labels:

Predicted Labels:



AIM

- ➢ Maximise TP, TN
- ➢ Minimise FP, FN

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Other Metrics to Evaluate our Models

Sensitivity

Also called as Recall

True Positive Rate

Correctly guessed as positives compared to total number of positives

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Other Metrics to Evaluate our Models

Specificity

Also called as True Negative Rate

Correctly guessed as negatives compared to total number of negatives

$$\text{Specificity} = \frac{TN}{TN+FP}$$

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Other Metrics to Evaluate our Models

Precision

Also called as Positive Predictive Value

Correctly guessed as positives
compared to total guessed as positives

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

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Other Metrics to Evaluate our Models

F1-Score

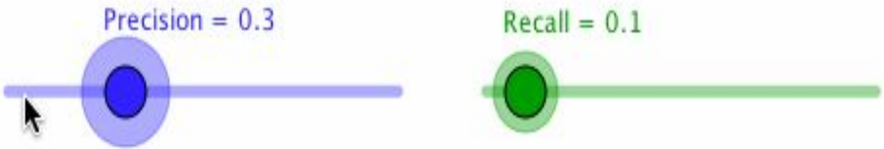
Harmonic Mean of Precision and Recall

Penalises False negatives and false positives.

Mostly used for uneven class distribution

$$\text{Precision} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$$

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN



$$F1 = \frac{2 * precision * recall}{precision + recall}$$

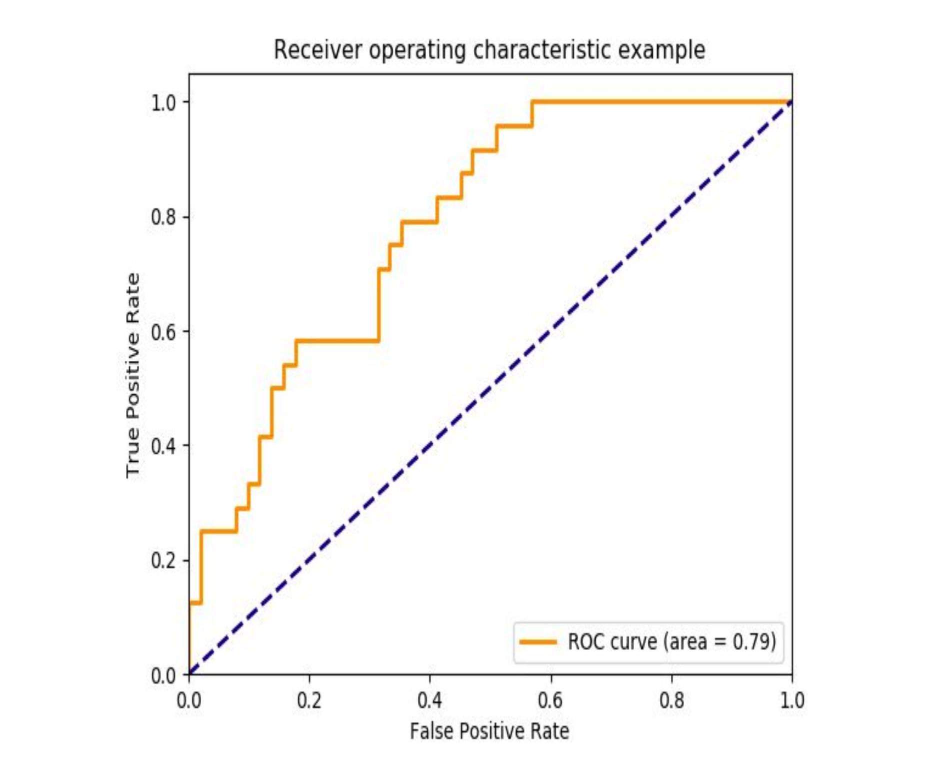
$$F1 = \frac{2 \times 0.3 \times 0.1}{0.3 + 0.1} \qquad \therefore F1 = 0.15$$

ROC-AUC

Receiver Operating Characteristics - Area under the Curve

(CLICK HERE TO EXPERIMENT)

- ➢ Threshold based evaluation metrics
- ➢ Also called Precision Recall Curve
- ➢ Tells the optimal threshold to select, depending on the true and the false positive rate



REGRESSION METRICS

MEAN SQUARED ERROR

- It is simply the average of the squared difference between the target value and the value predicted by the regression model.

- As it squares the differences, it penalizes even a small error which leads to over-estimation of how bad the model is.

- MSE or Mean Squared Error is one of the most preferred metrics for regression tasks.

$$MSE = \frac{1}{n} \Sigma \left(y - \hat{y} \right)^2$$

REGRESSION METRICS

ROOT MEAN SQUARED ERROR

- RMSE is the square root of the averaged squared difference between the target value and the value predicted by the model.
- It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors.
- This implies that RMSE is useful when large errors are undesired.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2}$$

REGRESSION METRICS

MEAN ABSOLUTE ERROR

- The MAE is more robust to outliers and does not penalize the errors as extremely as mse
- MAE is the absolute difference between the target value and the value predicted by the model.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - y_j|$$

What Metrics to use When?

Depends on the Dataset

Classification

- **Fraud Detection:** Every Non-Fraud transaction that gets classified as Fraud does not bear that heavy a cost, at the maximum - the person will get one extra phone call to check whether the transaction is fraudulent or not. BUT! Every fraud transaction that goes undetected and unchecked - will incur a huge cost!
  - THEREFORE False positives are not as important as False Negatives
  - SPECIFICITY >> SENSITIVITY

What Metrics to use When?

Depends on the Dataset

Classification

- **Disease Detection:** If a healthy person is falsely detected, it is problematic since they may undergo unnecessary surgery/treatment. If a diseased person is falsely detected as healthy, the disease may progress further to an advanced stage.
    - THEREFORE False positives and False Negatives both important
    - SPECIFICITY and SENSITIVITY are both important

What Metrics to use When?

Depends on the Dataset

Regression

- **MSE:** When we want to penalize even small errors
- **MSE:** When we want to penalize outliers
- **MAE:** When we do not want of penalise outliers that much
- **RMSE:** Useful when large errors are undesired.

Good Rule of Thumb

If unsure, or in general - Report all of these metrics!

Most of these are provided in sklearn.

CROSS VALIDATION

- We know how our model performs on seen data, but how do we be sure on how it performs on new data?
- What if less data is available - which makes it difficult to separate data for training and testing
- What if our training and testing was sampled in such a way that there is a certain bias which causes the testing dataset to perform better than it would on new and unknown data?

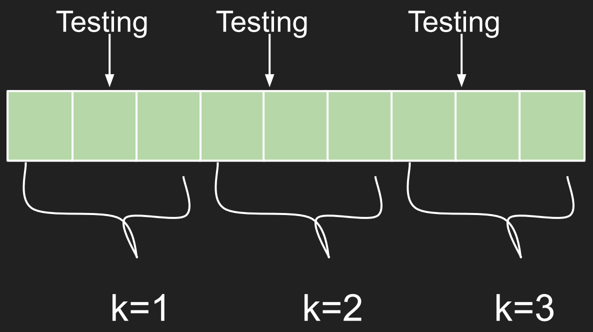
Answer: Cross Validate your data

There are many different types of cross validation that exist, however we will be discussing the two most common types.

WIDELY USED CROSS VALIDATION METHODS

k-Fold Cross Validation

Divide Data into K Folds
Let's take k=3 for example



Leave One Out CV (LOOCV)

All but one sample is chosen as training set

WIDELY USED CROSS VALIDATION METHODS

k-Fold Cross Validation

- Used on larger datasets

Leave One Out CV (LOOCV)

- Used on smaller datasets
- Computationally Expensive
- Best results

LOOCV is a special case of k-Fold CV

When k=size(data)-1, only one row is the testing set, others are training which is nothing but LOOCV!

HYPERPARAMETER TUNING

➢ For any classifier/regressor that we use, there are a lot of parameters and hyperparameters.

How to find the best values for each of these parameters? Testing out all of these will take a huge amount of time!

Solution:
GridSearchCV in sklearn

GRID SEARCH CV

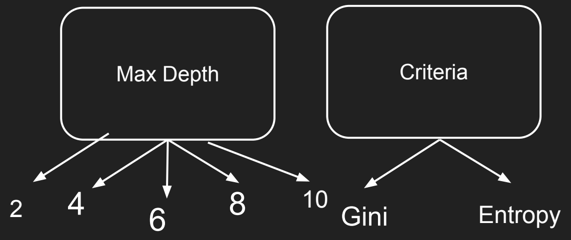
- ➢ Input all different values you want to check the performance on.

- ➢ Input the metrics you want to optimize the results on (Accuracy, Sensitivity, Specificity, AUC-ROC)

- ➢ Cross Validate results however many times required

Try it using sklearn!

GRID SEARCH CV



Total iterations to be carried out = 5*2 = 10 times

Performance of each will be ranked on the basis of what scoring metrics is provided

Return best model hyperparameters

- PROBLEM?
  - Computationally too expensive!
  - A simple dataset with 500 rows can take upto hours to compute the best results (depending on the model).
  - Even addition of a single new value of one hyperparameter will increase the computation time by a lot (order of total number of hyper parameter values).
  - Solution - RandomisedSearchCV

RANDOMISED SEARCH CV

- ➢Input all different values you want to check the performance on.
- ➢Input number of iterations you want (searches the grid only that many number of times)
- ➢Faster than GridSearch, since it does not search the entire sample space
- ➢Finds close to optimal solution
- ➢Works good on larger data sets.
- ➢One can use Randomised Search to find close to optimal solution, and then tune it further on Grid Search CV.