Model Evaluation

XX-161-A-21 – Big Data Analytics

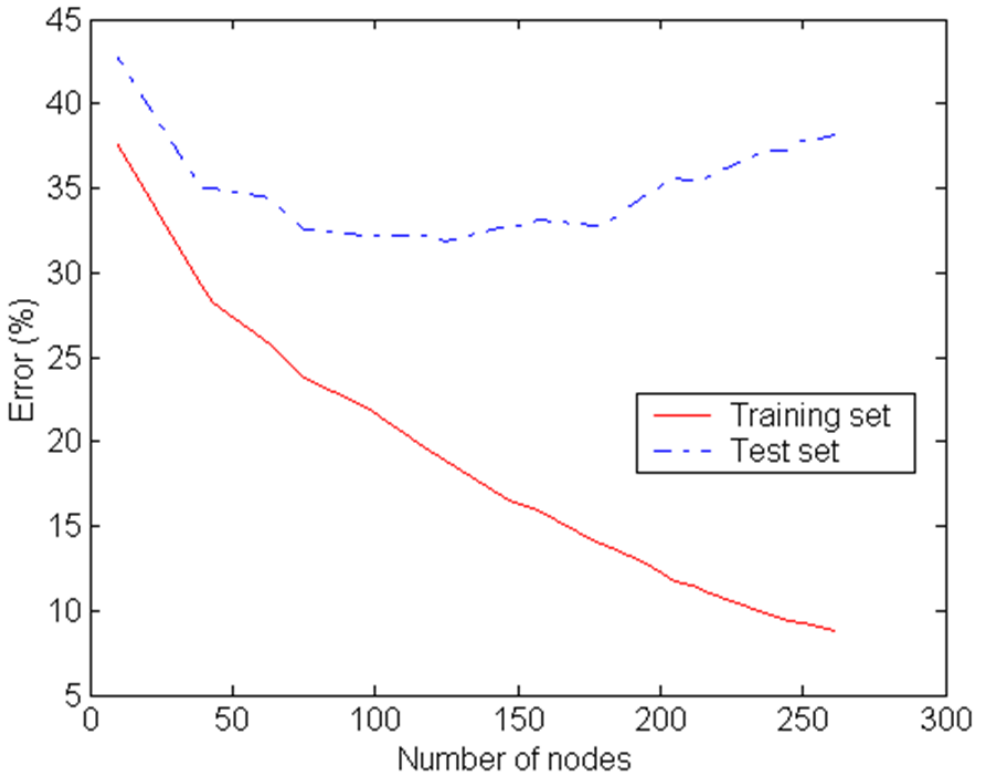
Dr. Jim Scrofani
jwscrofa@nps.edu

Topics at a Glance

- Motivation
- Methods for Performance Evaluation
- Metrics for Performance Evaluation
  - Classifier Models
  - Regression Models
- Methods for Model Comparison
  - ROC
  - Area under the Curve

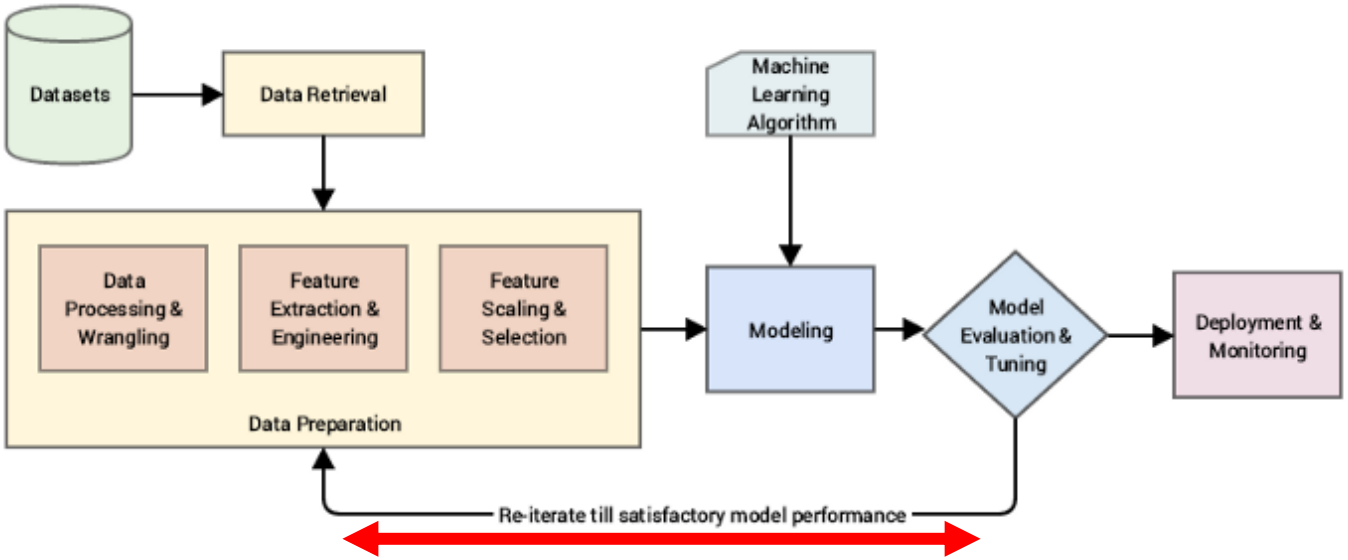
Motivation

- Purpose:
  - To estimate performance of a model on previously unseen data (test set)
  - To create high-quality models



Training and test error rates for learning decision trees of varying sizes

Machine Learning Workflow



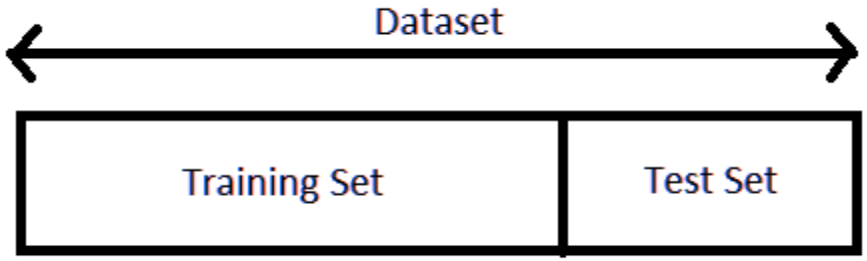
A standard machine learning pipeline (source: Practical Machine Learning with Python, Apress/Springer)

Note the iterative dialogue between ***model evaluation*** *and feature engineering*


THE #1 DATA SCIENTIST EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY MODEL'S TRAINING"
HEY! GET BACK TO WORK!
TRAINING !
OH. CARRY ON.

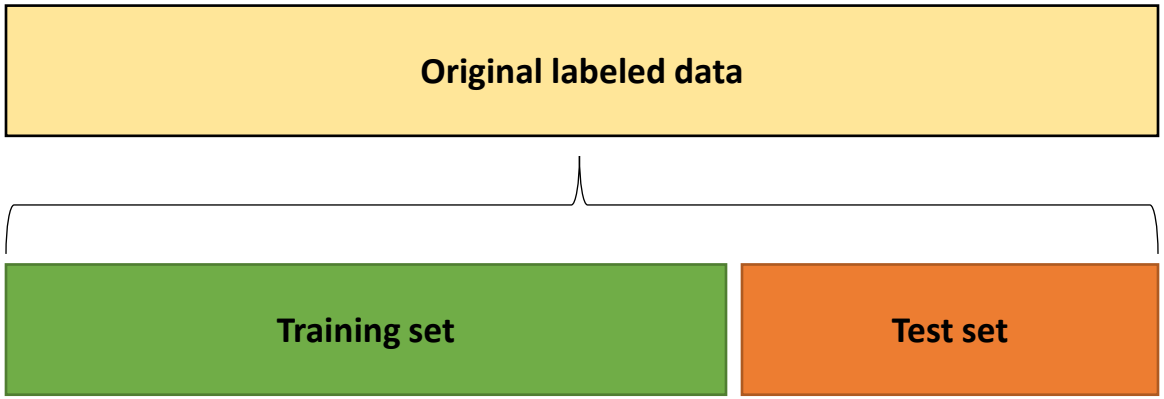
Model Evaluation Methods

- Model evaluation requires the assessment of performance of a learned model on a labeled test set that has not been used at any stage of model selection
- Typically achieved by partitioning dataset of labeled instances D into two disjoint subsets:
  - D_{train} used for model selection and
  - D_{test} used for computing test error rate $\varepsilon_{\text{test}}$



Partitioning Datasets

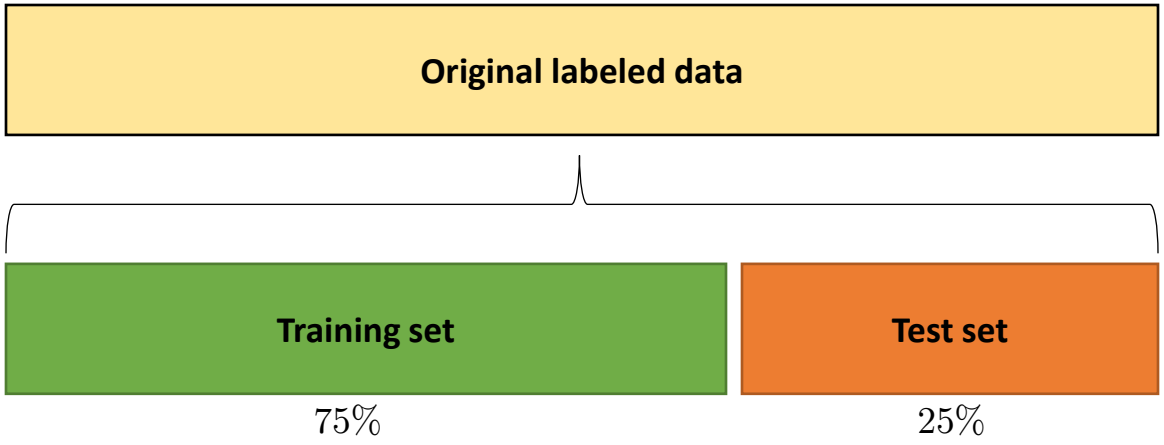
- Two approaches
  - Holdout Method
  - Cross-Validation



Holdout Method

- Most basic method for partitioning dataset
- Labeled dataset is randomly partitioned into two disjoint set
  - Reserve $k\%$ for training and $(100\text{-}k)\%$ for testing
    - *Training set* for model construction
    - *Test set* for model evaluation

$k = 75\%$



Holdout Method

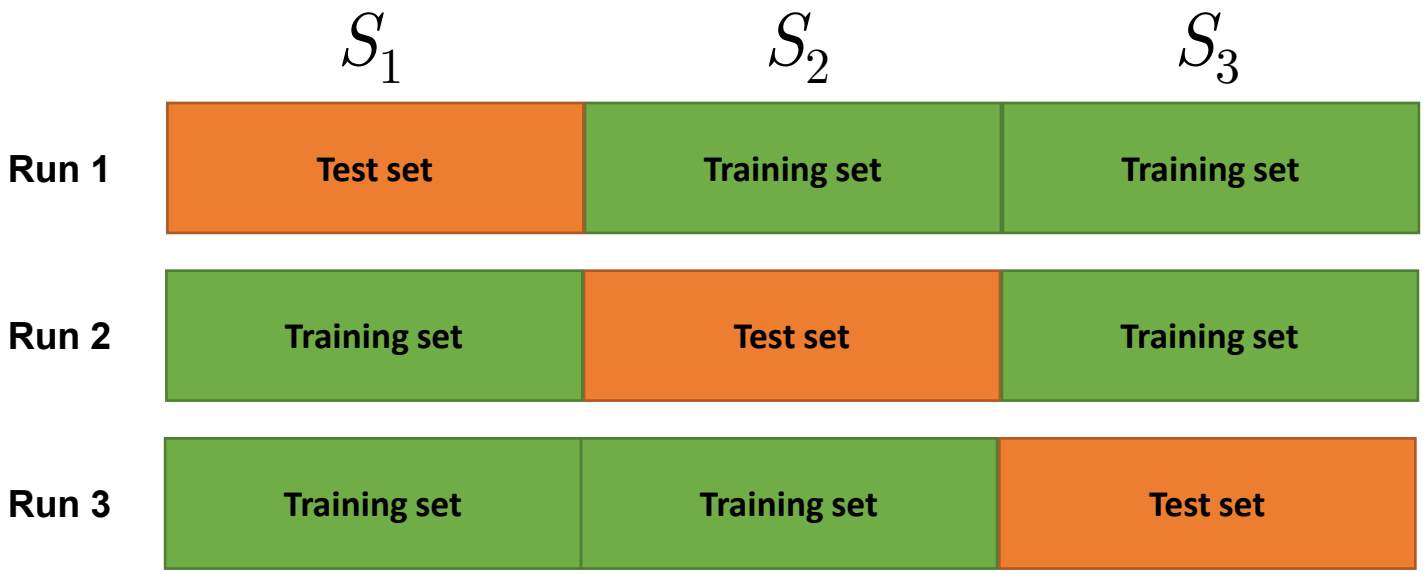
- Variation of holdout method
  - Repeat the holdout method k times
  - Compute test error rate each time
  - Obtain a distribution of error rates to understand variance of $\varepsilon_{\text{test}}$

Cross-validation

- Cross-validation
  - Example: $k = 3$, (see illustration on next slide)
    - Randomly partition the labeled dataset into 3 equally-sized partitions
    - 1st run -- train a model using subsets S_2 and S_3 and test the model on S_1
    - 2nd run -- train a model using subsets S_1 and S_3 and test the model on S_2
    - 3rd run -- train a model using subsets S_1 and S_2 and test the model on S_3

Cross-validation

- Cross-validation
  - Example (cont): $k = 3$, *three-fold cross-validation*



Cross-validation

- Cross-validation
  - Example (cont): $k = 3$, *three-fold cross-validation*
    - Test error rate is computed for each run ($\varepsilon_1, \varepsilon_2, \varepsilon_3$)
    - Overall test error rate obtained by averaging error rate over all runs

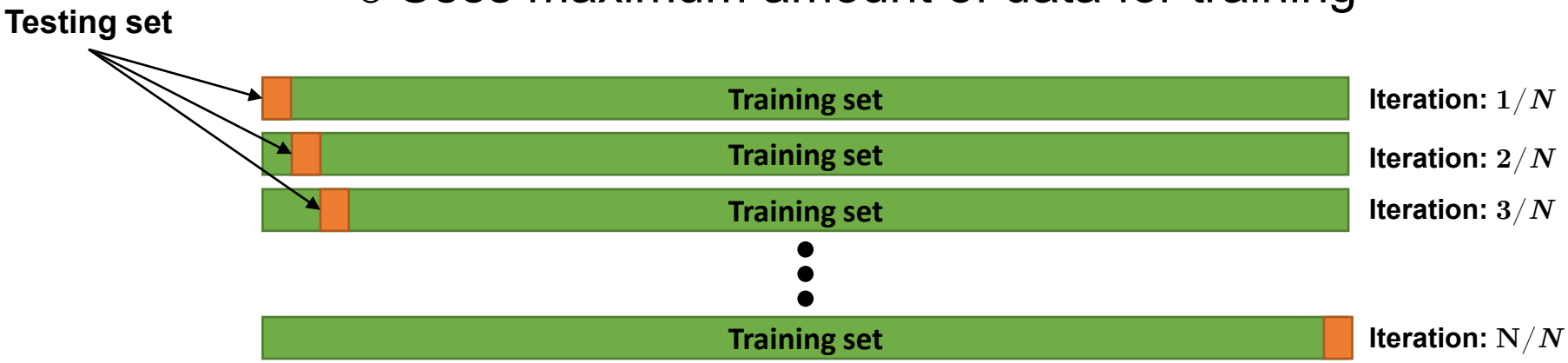
Cross-validation

- Cross-validation
  - Prior example can be generalized to k-fold cross-validation
    - Randomly partition data D into k equal-sized disjoint subsets (*folds*) of size N
  - During the i^{th} run
    - One partition of D is chosen as $D_{\text{test}}(i)$ for testing
    - While the rest are used as $D_{\text{train}}(i)$ for training
    - A model m is learned using $D_{\text{train}}(i)$ and applied on $D_{\text{test}}(i)$ to obtain the test error, $\varepsilon_{\text{sum}}(i)$
    - This procedure is repeated k times
    - Total error rate is computed as

$$\varepsilon_{\text{test}} = \frac{\sum_{i=1}^{k} \varepsilon_{\text{sum}}(i)}{N}$$

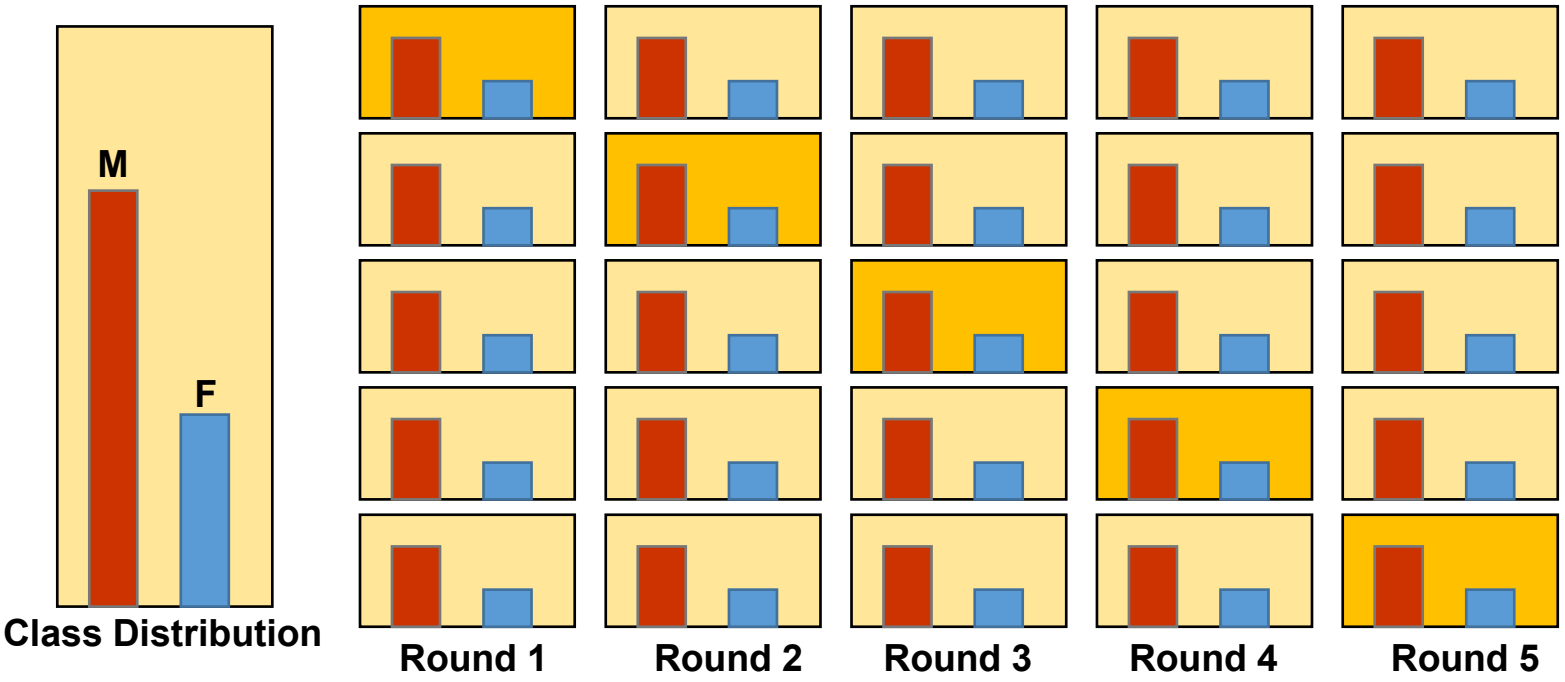
Cross-validation

- Cross-validation (other approaches)
  - *Leave-one-out*: Special case of k-folds where $k = N$, where $N = \#$ of observations in data
    - Every run uses exactly one data instance for testing and the remainder of the data for training
    - Uses maximum amount of data for training



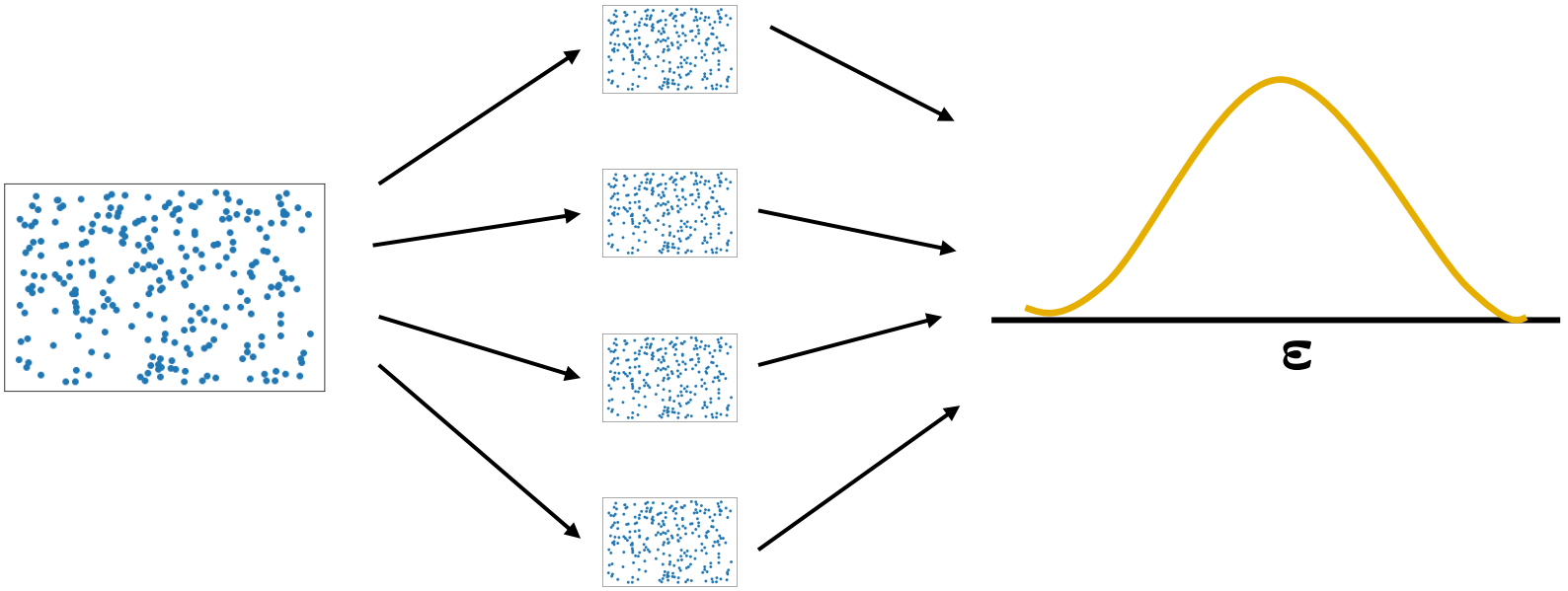
Cross-validation

- Cross-validation (other approaches)
  - *Stratified cross-validation*: folds are stratified so that class distribution in each fold is approx. same as that of original dataset



Bootstrap Method

- Resample available data *with replacements* to generate a number of simulated datasets of the same size as the original dataset
  - New training sets can be used to define the "*bootstrap*" estimates of the error rate



Performance Evaluation Methods

- How to obtain a reliable estimate of performance?
  - Algorithm type
  - Hyperparameters

- Performance of a model may depend on other factors besides the learning alg.
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

Tuning Hyperparameters

- Most training algorithms contain hyperparameters that must be chosen before the training process begins
- It is possible and recommended to search the hyperparameter space for the model that yields the best cross validation score

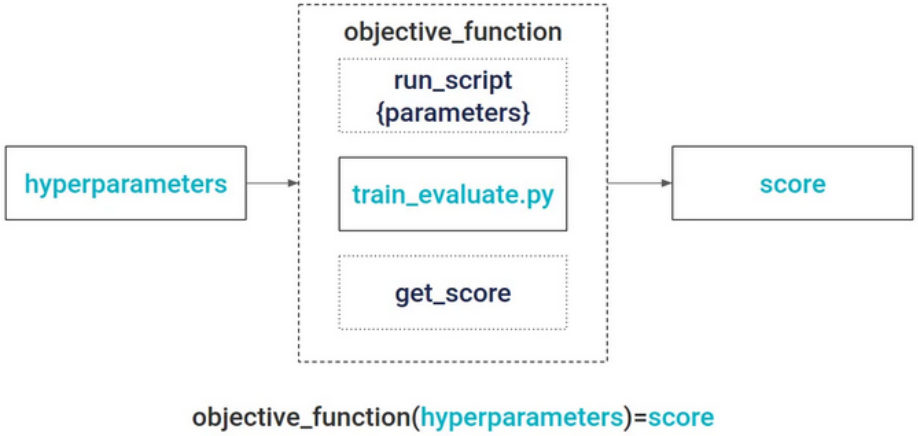
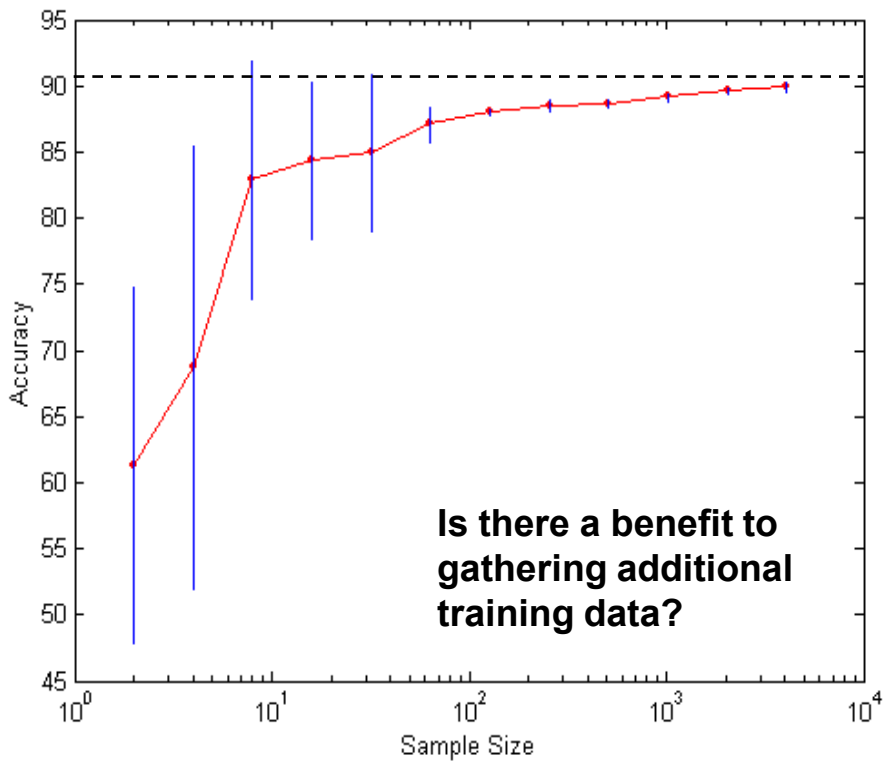


Figure from: https://www.kdnuggets.com/2020/04/hyperparameter-tuning-python.html, last accessed 25 May 2021

Learning Curve



- Learning curve shows how a metric changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, *et al.*)
  - Geometric sampling (Provost, *et al.*)
- Effect of small sample size:
  - Bias in the estimate
  - Variance of estimate

We want to evaluate the effect of the number of observations in the training set on some metric, accuracy, F1, etc.

Predictive Capability

- Focus on the *predictive capability* of a model
  - Rather than on how fast it takes to classify/regress or build models, scalability, etc.
- Confusion matrix most basic approach for representing performance of classifier model on test set
- Regression model performance measured by
  - R^2 Coefficient of Determination
  - Mean Square Error (MSE) / Root Mean Square Error (RMSE)
  - Mean Absolute Error (MAE)

Classifier Evaluation: Confusion Matrix

- Confusion matrix summarizes the number of instances predicted correctly or incorrectly by a classifier
- Various evaluation measures or metrics are derived from the confusion matrix

		Predicted	
		Has Heart Disease	Does Not Have Heart Disease
Actual	Has Heart Disease	True Positives	False Negatives
	Does Not Have Heart Disease	False Positives	True Negatives

Confusion Matrix

Actual value \ Prediction outcome	+	−	total
+	True Positives	False Negatives	P
−	False Positives	True Negatives	N
total	P′	N′	

Confusion Matrix (CM):

- Given m classes, an entry, $CM_{i,j}$ in a confusion matrix indicates # of records in class i that were labeled by the classifier as class j
- May have extra rows/columns to provide totals

Confusion Matrix

Classifier's prediction

Actual value	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	6954	46	7000
buy_computer = no	412	2588	3000
Total	7366	2634	10000

Example of a Confusion Matrix

Classifier correctly assigns 'yes' to 6954 consumers and 'no' to 2588 consumers

Classifier Performance Metrics

- Accuracy or recognition rate: percentage of test set records that are correctly classified

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}}$$

- Error rate:

$$\text{Error rate} = 1 - \text{Accuracy}$$
$$= \frac{\text{FP} + \text{FN}}{\text{Total}}$$

		Prediction outcome		
		+	−	**total**
Actual value	+	True Positives	False Negatives	P
	−	False Positives	True Negatives	N
	total	P′	N′	

24

Classifier Performance Metrics

- True Positive Rate (TPR)
  - Fraction of positive test instances correctly predicted by a classifier
  - Also called *sensitivity* or *recall*
  - A classifier with a high TPR has a high chance of correctly identifying a positive instance of the data

$$\text{TPR or Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{P}}$$

- True Negative Rate (TNR)
  - Fraction of negative test instances correctly predicted by a classifier
  - Also called *specificity*
  - A classifier with a high TPR has a high chance of correctly identifying a positive instance of the data

$$\text{TNR or Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{\text{TN}}{\text{N}}$$

Class Imbalance

- Consider a 2-class problem (10,000 patients)
  - Number of patients that do not have COVID = 9990 (Class 0)
  - Number of patients with COVID = 10 (Class 1)
- If we use a model that predicts everything to be Class 0, a dummy classifier, it appears to be extremely accurate
- Accuracy is misleading because Class 1 is rare

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{0 + 9990}{1000} = 99.9\%$$

		Predicted	
		Has COVID	Does Not Have COVID
Actual	Has COVID	0 **TP**	10 **FN**
	Does Not Have COVID	0 **FP**	9990 **TN**

Class Imbalance

- Precision or Positive Predicted Value (PVV) is useful for highly skewed test sets where the positive predictions, even though small in numbers, are required to be mostly correct

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

# of Positive Predictions

- A classifier that has high precision is likely to have most of its positive predictions correct

Classifier Performance Metrics

- *Precision:* exactness – what % of records that the classifier labeled as positive are actually positive?

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

- *Sensitivity or Recall:* completeness – what % of positive records did the classifier label as positive?
  - Perfect score is 100%
  - Inverse relationship between Precision and Recall

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

- F-measure (**F_1** or ***F*-score**): harmonic mean of Precision and Recall

$$\text{F}_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- F_β: weighted measure of precision and recall

$$\text{F}_\beta = \frac{(1 + \beta^2)\text{Precision} \times \text{Recall}}{\beta^2\text{Precision} + \text{Recall}}$$

Classifier Evaluation Metrics: Example

Actual Class\Predicted Class	Intrusion = yes	Intrusion = no	Total	Recognition (%)
Intrusion = yes	**90** TP	**210** FN	300	30.0 (sensitivity)
Intrusion = no	**140** FP	**9560** TN	9700	98.56 (specificity)
Total	230	9770	10000	96.50 (accuracy)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{90}{230} = 39.13\%$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{90}{300} = 30.00\%$$

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

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

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}}$$

Evaluating Regression Models

- Regression model performance
  - R^2 -- Coefficient of Determination
  - Mean Square Error (MSE) / Root Mean Square Error (RMSE)
  - Mean Absolute Error (MAE)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

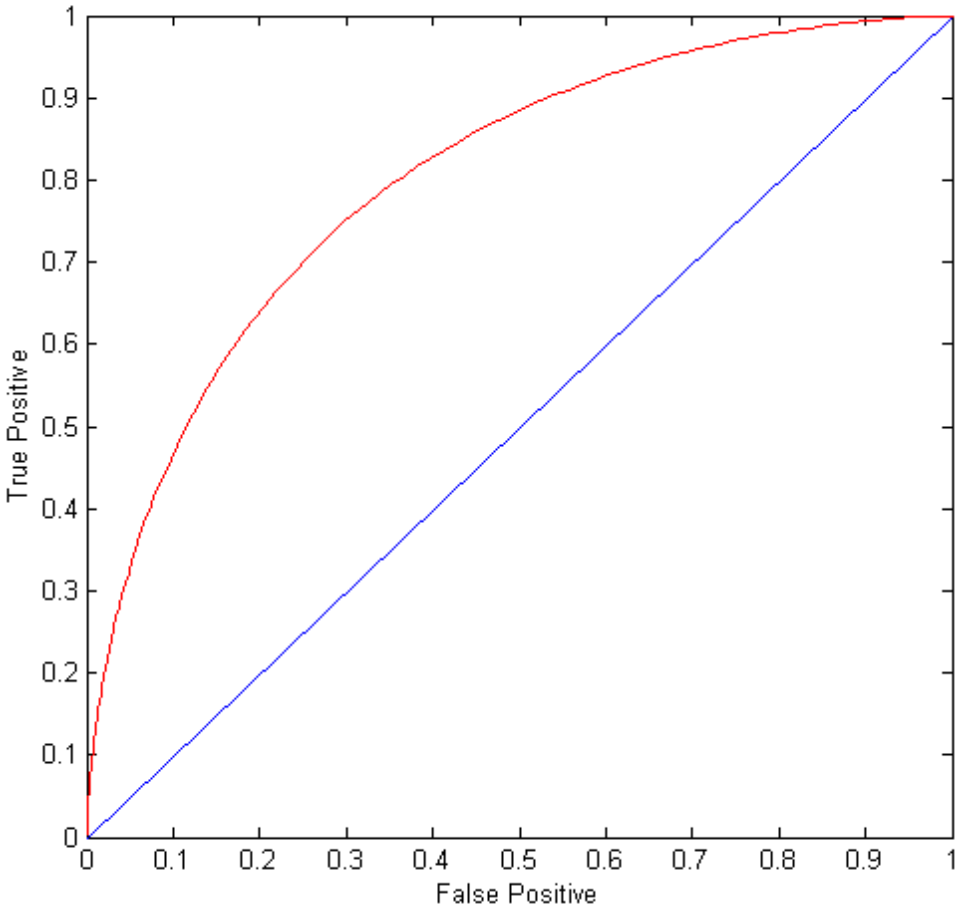
Methods for Model Comparison

Model Comparison

- *Receiver Operating Characteristics (ROC) Curve*: developed from signal detection theory to analyze signals in the presence of noise
  - Characterizes trade-off between detections (TPR) & false alarms (FPR)
- Plots TPR (on the y-axis) against FPR (on the x-axis)
- The area under the ROC curve (AUC) is a measure of the accuracy of the model

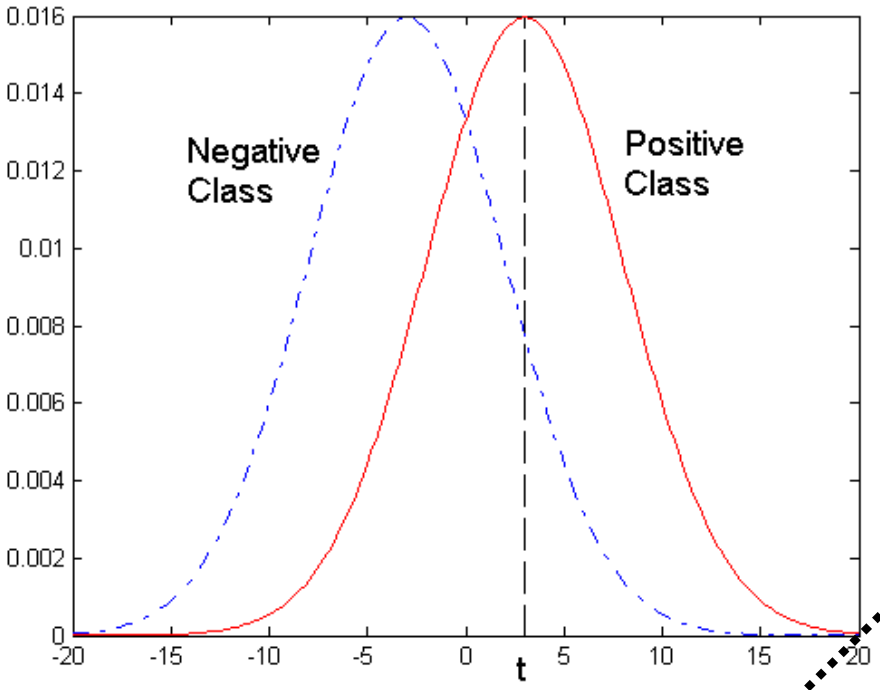
ROC Curve

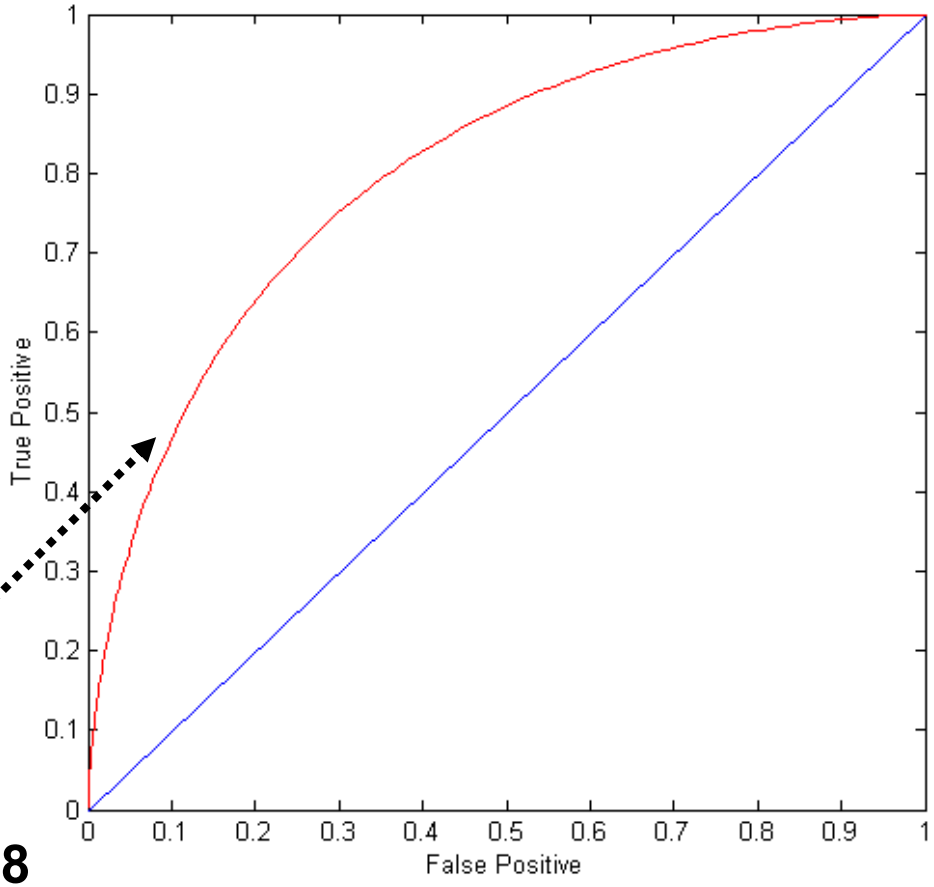
- (TPR, FPR):
  - (0,0): declare everything to be negative class
  - (1,1): declare everything to be positive class
  - (1,0): ideal classifier
- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)
- any points $x > t$ are classified as positive

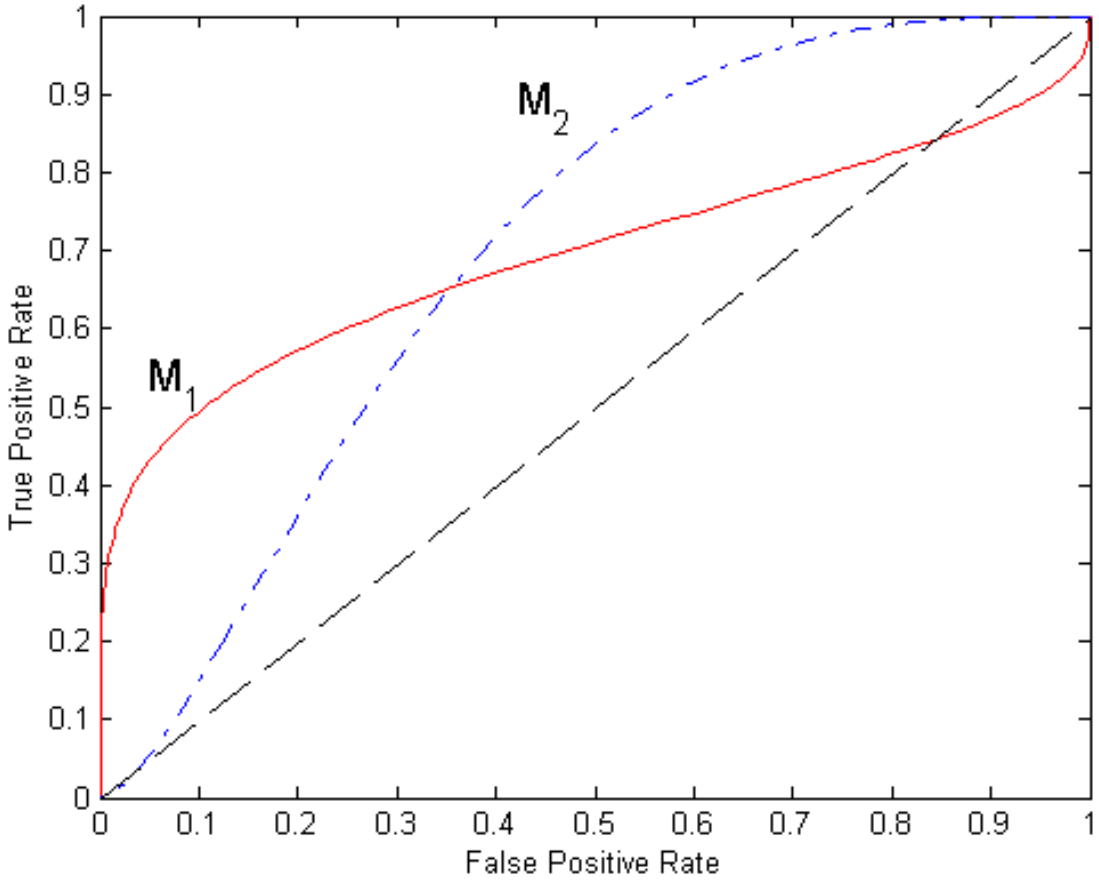




At threshold t:

TP=0.5, FN=0.5, FP=0.12, TN=0.88

Using ROC for Model Comparison



- Neither model consistently outperforms the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- *Area Under the Curve* (AUC-ROC)
  - Ideal: Area = 1
  - Random guess: Area = 0.5

Choose the model with the greatest AUC-ROC

How to Construct a ROC Curve

Instance	Score Threshold (s)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Compute threshold s for each test instance
- Sort the instances according to score in decreasing order
- Apply threshold at each unique value of s
- Count the number of TP, FP, TN, FN at each threshold

How to Construct a ROC Curve

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

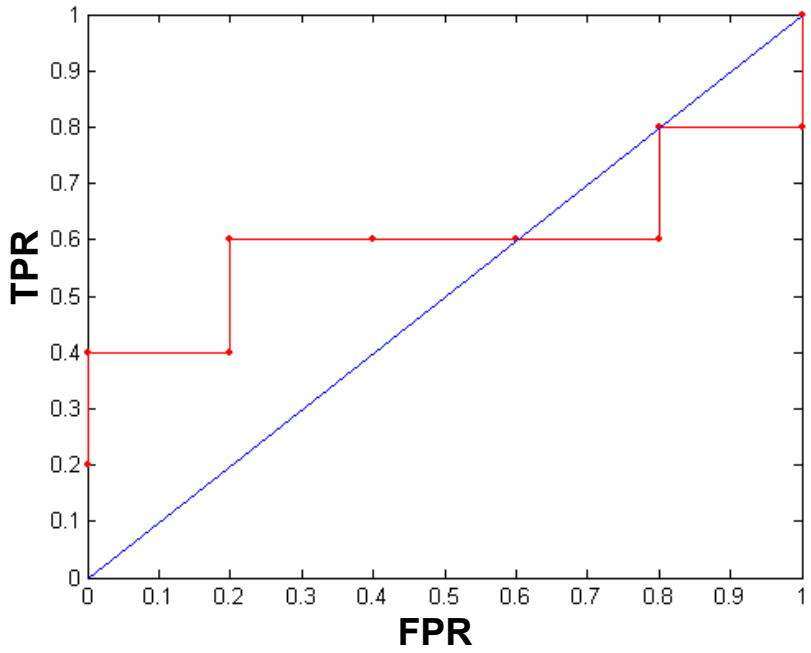
$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

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

For $s = 0.25$, a test observation is classified as positive if its score is greater than or equal to 0.25, TP = 5, FP = 5, TN = 0, FN = 0

Then consider $s = 0.43$, a test observation…, TP = 4, FP = 5, TN = 1, FN = 0

And so on,…



ROC Curve:

Recap

- Motivation
- Methods for Performance Evaluation
- Metrics for Performance Evaluation
  - Classifier Models
  - Regression Models
- Methods for Model Comparison
  - ROC
  - Area under the Curve