Evaluating Classifiers

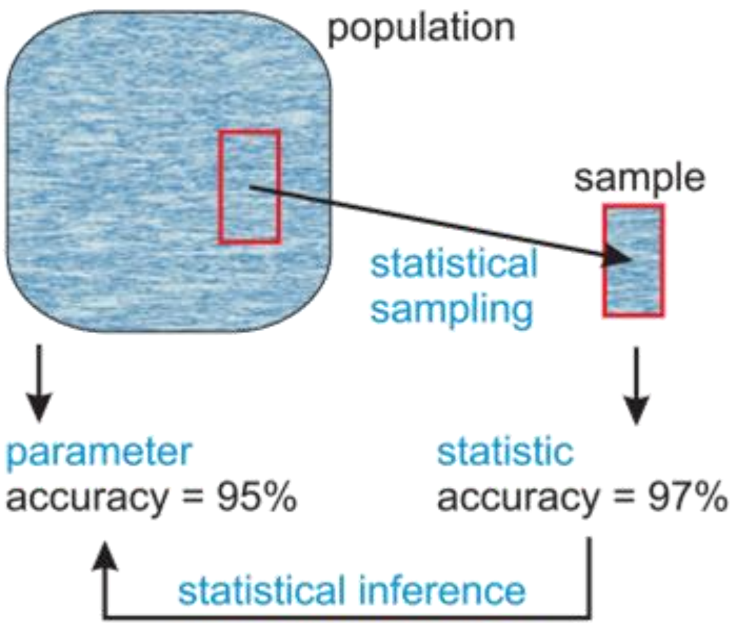
Agenda

- Introduction
- Evaluation Strategies
  - Re-substitution
  - Leave one out
  - Hold out
  - Cross Validation
  - Bootstrap Sampling
- Classifiers Evaluation Metrics
  - Binary classification confusion matrix
    - Sensitivity and specificity
    - ROC curve
    - Precision & Recall
  - Multiclass classification

Introduction

- Evaluation aims at selecting the most appropriate learning schema for a specific problem

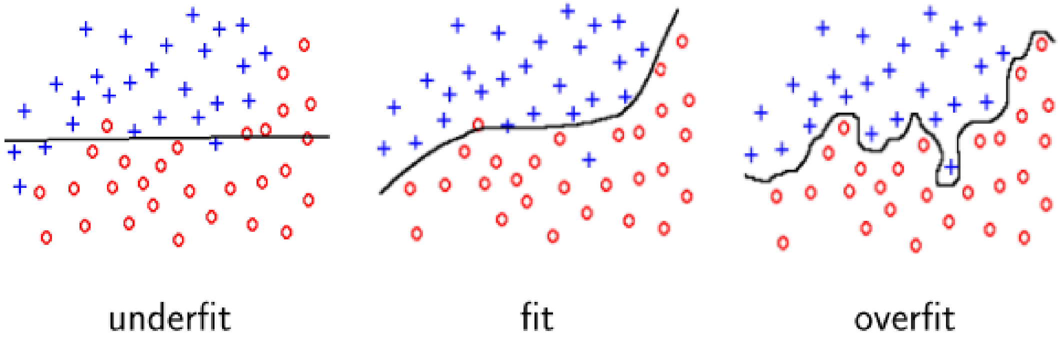
- We evaluate its ability to generalize what it has been learned from the training set on the new unseen instances



Evaluation Strategies

- Different strategies to split the dataset into training ,testing and validation sets.

- Error on the training data is not a good indicator of performance on future data, ***Why***?
  - Because new data will probably **not** be ***exactly*** the same as the training data!
  - The classifier might be fitting the training data too precisely (called ***Over-fitting***)

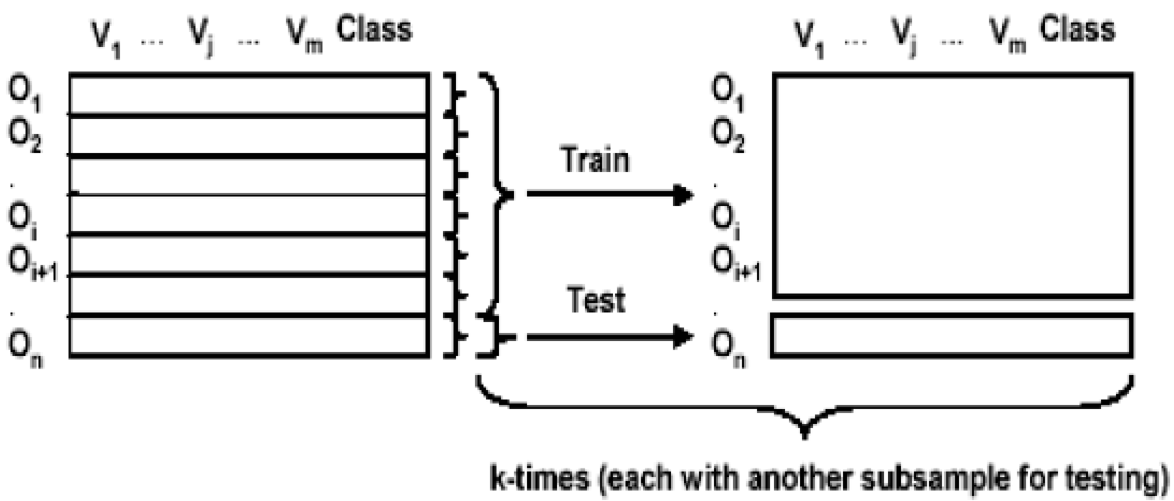


Re-substitution

- Testing the model using the dataset used in training it
- Re-substitution error rate indicates only how good /bad are our results on the TRAINING data; expresses some knowledge about the algorithm used.
- The error on the training data is NOT a good indicator of performance on future data since it does not measure any not yet seen data.

Leave one out

- leave one instance out and train the data using all the other instances
- Repeat that for all instances
- compute the mean error



Hold out

- The dataset into two subsets use one of the training and the other for validation (In Large datasets)

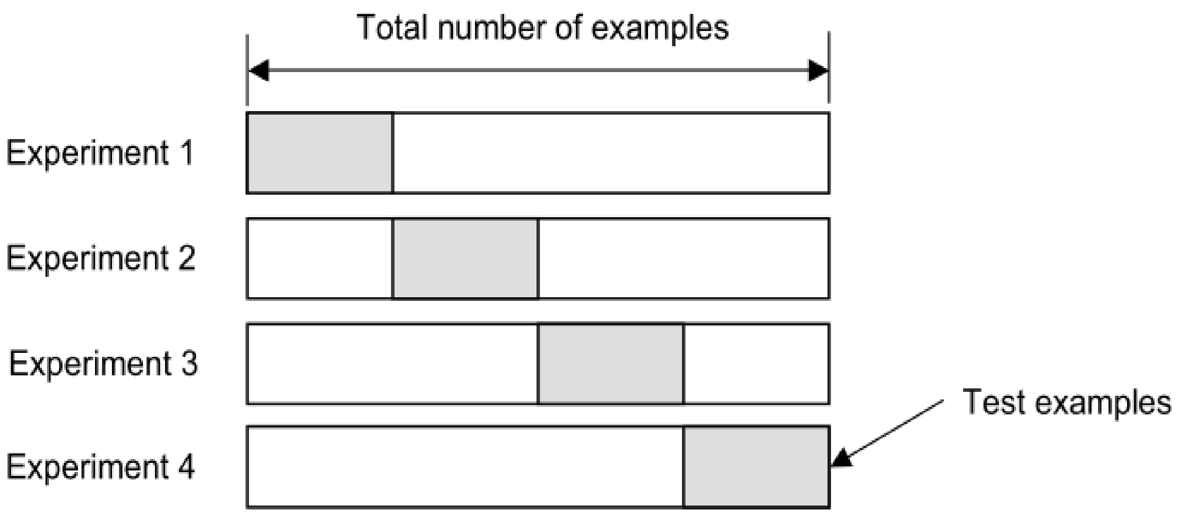
- Common practice is to **train** the classifiers using **2 thirds** of the dataset and **test** it using the **unseen third**.

- ***Repeated holdout method***: process repeated several times with different subsamples

- error rates on the different iterations are averaged to give an overall error rate

K-Fold Cross Validation

- Divide the data to k equal parts (folds),
- One fold is used for testing while the others are used in training,
- The process is repeated for all the folds and the accuracy is averaged.
- 10 folds are most commonly used
- Even better, repeated Cross-validation

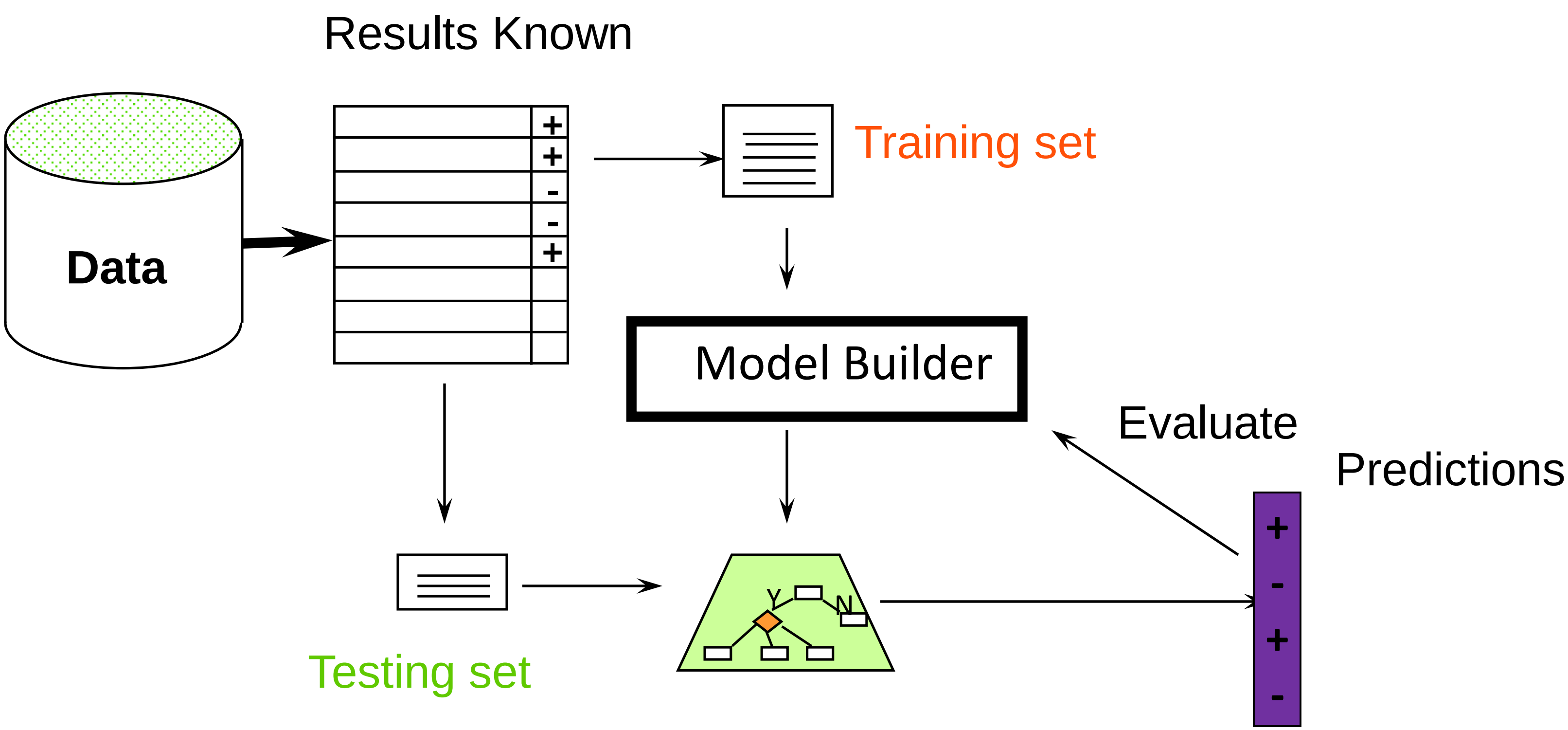


Bootstrap sampling

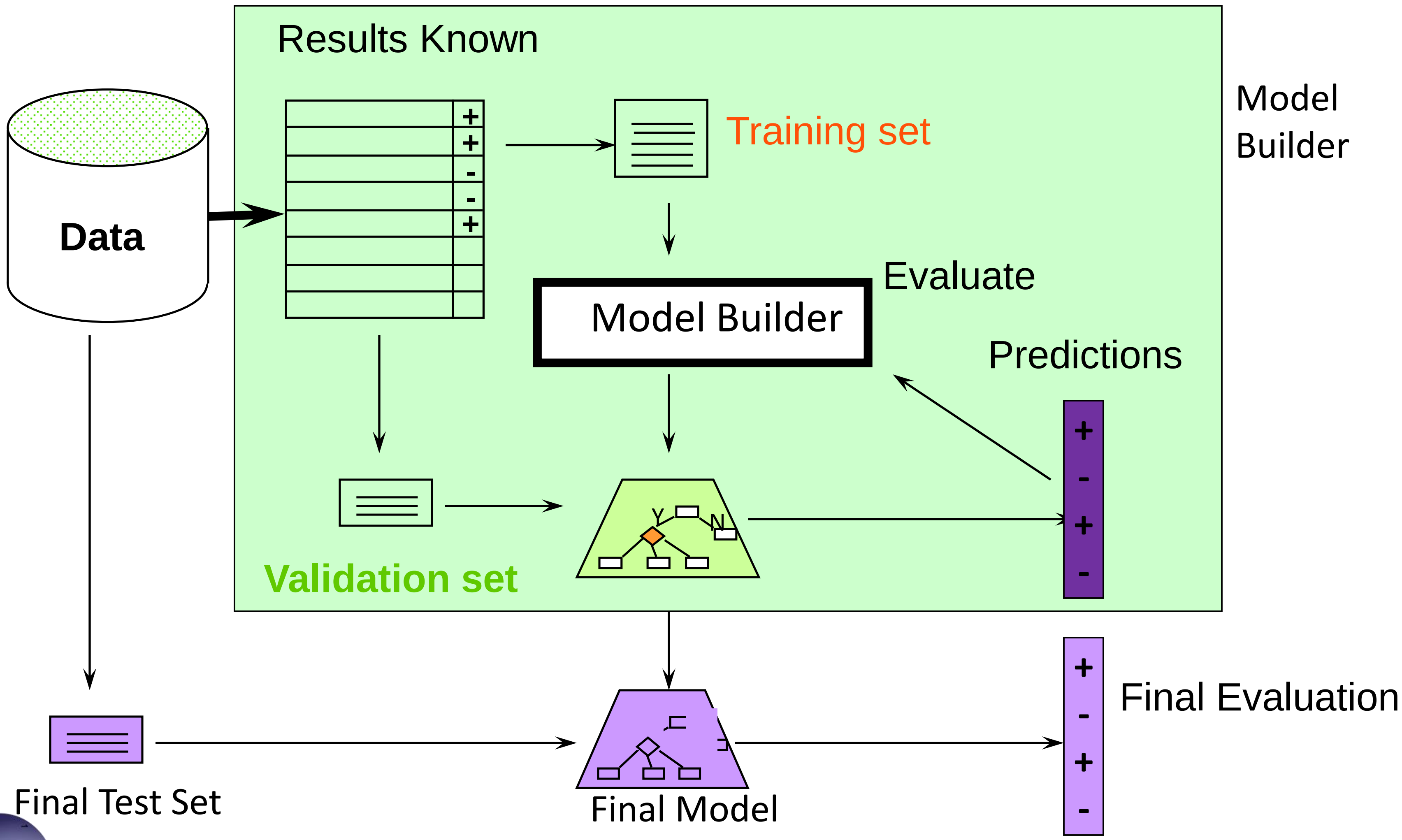
- CV uses sampling without replacement
  - The same instance, once selected, can not be selected again for a particular training/test set

- The bootstrap uses sampling with replacement to form the training set
  - Sample a dataset of n instances n times with replacement to form a new dataset of n instances
  - Use this data as the training set
  - Use the instances from the original dataset that don't occur in the new training set for testing

Are Train/Test sets enough?



Train, Validation, Test split



Metrics

It is extremely important to use **quantitative metrics** for evaluating a machine learning model

- Until now, we relied on the **loss (objective) function value** for regression and classification

- Other metrics can be used to **better evaluate** and understand the model

- **For classification**
  - ✓ Accuracy/Precision/Recall/F1-score, ROC curves,…
- **For regression**
  - ✓ Normalized RMSE, Normalized Mean Absolute Error (NMAE),…

RMSE Metric for Regression

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

$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values

$y_1, y_2, \ldots, y_n$ are observed values

n is the number of observations

Normalized RMSE

$$NRMSE_m(\mathbf{y}, \hat{\mathbf{y}}) = \frac{RMSE(\mathbf{y}, \hat{\mathbf{y}})}{\frac{1}{n}\sum_{i=1}^{n} |y_i|} = \frac{RMSE(\mathbf{y}, \hat{\mathbf{y}})}{mean(|\mathbf{y}|)}$$

Mean Absolute Error (MAE)

$$MAE(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{n}$$

NMAE

$$NMAE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{MAE(\mathbf{y}, \hat{\mathbf{y}})}{\frac{1}{n}\sum_{i=1}^{n} |y_i|} = \frac{MAE(\mathbf{y}, \hat{\mathbf{y}})}{mean(|\mathbf{y}|)}$$

Evaluation Metrics For Classifiers

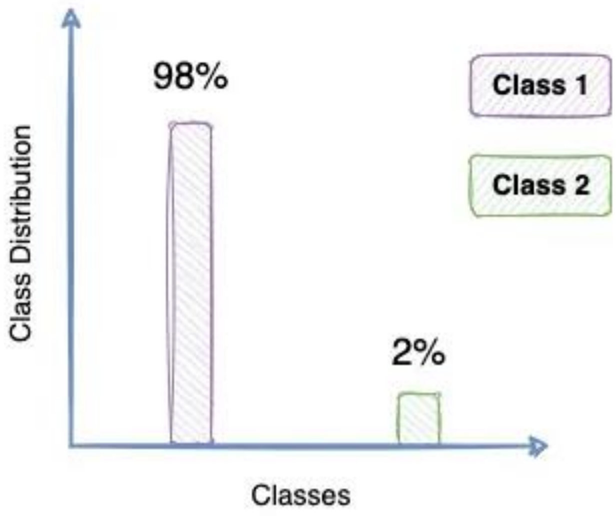
- Accuracy:

$$Accuracy = \frac{number\ of\ correctly\ classified\ instances}{total\ number\ of\ instances} \times 100$$

- It assumes equal cost for all classes
- Misleading in unbalanced datasets
- It doesn't differentiate between different types of errors

- Example -unbalanced classes
  - Cancer Dataset: 10000 instances, 9990 are ***normal***, 10 are ***ill*** **,** If our model classified all instances as ***normal*** accuracy will be 99.9 %
  - Medical diagnosis: 95 % healthy, 5% disease.
  - e-Commerce: 99 % do not buy, 1 % buy.
  - Security: 99.999 % of citizens are not terrorists.

- imbalanced (unbalanced) datasets

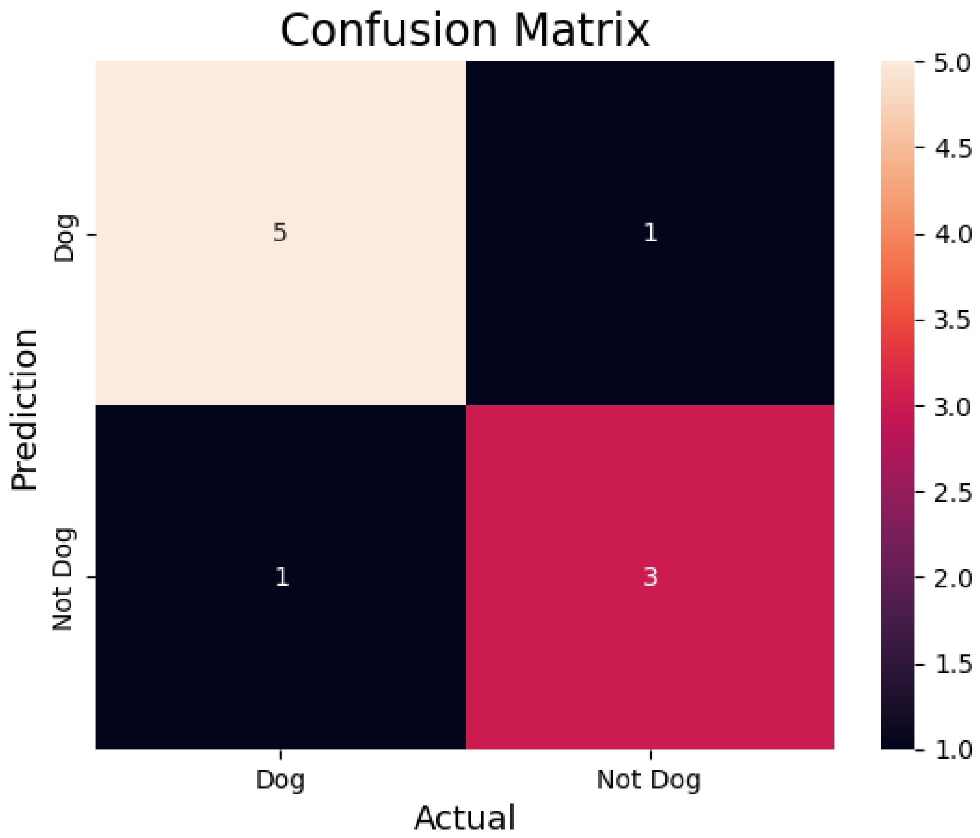


- Other examples : Spam classification

Binary classification Confusion Matrix

		predicted	
		negative	positive
actual examples	negative	*a* TN - True Negative correct rejections	*b* FP - False Positive false alarms type I error
	positive	*c* FN - False Negative misses, type II error overlooked danger	*d* TP - True Positive hits

Binary classification Confusion Matrix



Confusion Matrix

Precision and recall

Suppose that $y = 1$ in presence of a **rare class** that we want to detect

Precision *(How much we are precise in the detection)*

Of all patients where we classified $y = 1$, what fraction actually has the disease?

$$\frac{\text{True Positive}}{\text{\# Estimated Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Recall *(How much we are good at detecting)*

Of all patients that actually have the disease, what fraction did we correctly detect as having the disease?

$$\frac{\text{True Positive}}{\text{\# Actual Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$Success\,rate = \frac{TP + TN}{TP + TN + FP + FN}$$

Confusion matrix

Actual class

Estiamted class	1 (p)	0 (n)
1 (Y)	True positive (TP)	False positive (FP)
0 (N)	False negative (FN)	True negative (TN)

F-measure

- The ***F-measure*** is the harmonic-mean (average of rates) of precision and recall and takes account of both measures.

$$F\ measure = \frac{1}{\frac{1}{Recall} + \frac{1}{Precision}} = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

- It is biased towards all cases except the true negatives

Example:

Dataset:

- Contains 39 instances, 10 attributes
- The class labels are "negative, positive"
- 22 positive & 17 negative instances.

Classifier used: J48 – 10 folds cross validation

Confusion Matrix:

Classified as →	Positive	Negative
Positive	22	0
Negative	17	0

Classifier Accuracy = $\frac{22}{39} \times 100 = 56.4\%$

- TP= 22
- TN= 0
- FP= 17
- FN= 0
- The area under ROC curve: 0.5 in both cases, cause the TP rate = FP rate.
- Precision & Recall
  - Recall = $\frac{22}{22+0} = 1$
  - Precision = $\frac{22}{22+17} = 0.564$
- The *F-measure* = $= \frac{2 \times 22}{2 \times 22+17+0} = 0.7213$

F1-score

It is usually better to compare models by means of one number only. The **F1 – score** can be used to combine precision and recall

	Precision(P)	Recall (R)	Average	F_1 Score
Algorithm 1	0.5	0.4	0.45	$\boxed{0.444}$
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

The best is Algorithm 1

Algorithm 3 classifies always 1

Average says not correctly that Algorithm 3 is the best

$$\text{Average} = \frac{P+R}{2} \qquad F_{\$}\text{score} = 2\,\frac{P \cdot R}{P+R}$$

- $P = 0 \text{ or } R = 0 \Rightarrow F_{\$}\text{score} = 0$
- $P = 1 \text{ and } R = 1 \Rightarrow F_{\$}\text{score} = 1$

Precision vs Recall

- To summarize:
  - A system with high Precision might leave out some actual positives, but what it intends to return is high accuracy on the positive class.
  - On the other hand, a system with high Recall might give you numerous misclassifications of actual negatives, but it almost always will correctly classify the actual positives from the dataset.
  - Which one to choose is a tough decision and entirely depends on the problem you are using machine learning for.

Precision vs Recall

- Generally speaking, optimizing Precision typically reduces Recall and vice versa.

- Therefore, many applications consider a metric derived from both Precision and Recall (called the F-score) to measure the performance of a machine learning situation.

Precision vs Recall-Example 1

- Say you are building a course recommender system. The idea is to recommend courses to students based on their profiles.

- A student's time is crucial; thus, you don't want to waste their valuable time recommending courses that they may not like or are irrelevant to them.

- What would you optimize? Precision or Recall?

Precision vs Recall-Example 1

- Answer:

  - Precision is what you should prefer optimizing for. It is okay not to recommend good courses.

  - However, what your system recommends should be very high quality as students should not spend time watching courses that are irrelevant to them.

Precision vs Recall-Example 2

- Next, assume that you want to shortlist candidates for a job opening and see if they should be selected for an interview.
- As your organization is looking for talented candidates, you don't want to miss out on a candidate that can be a potential hire.

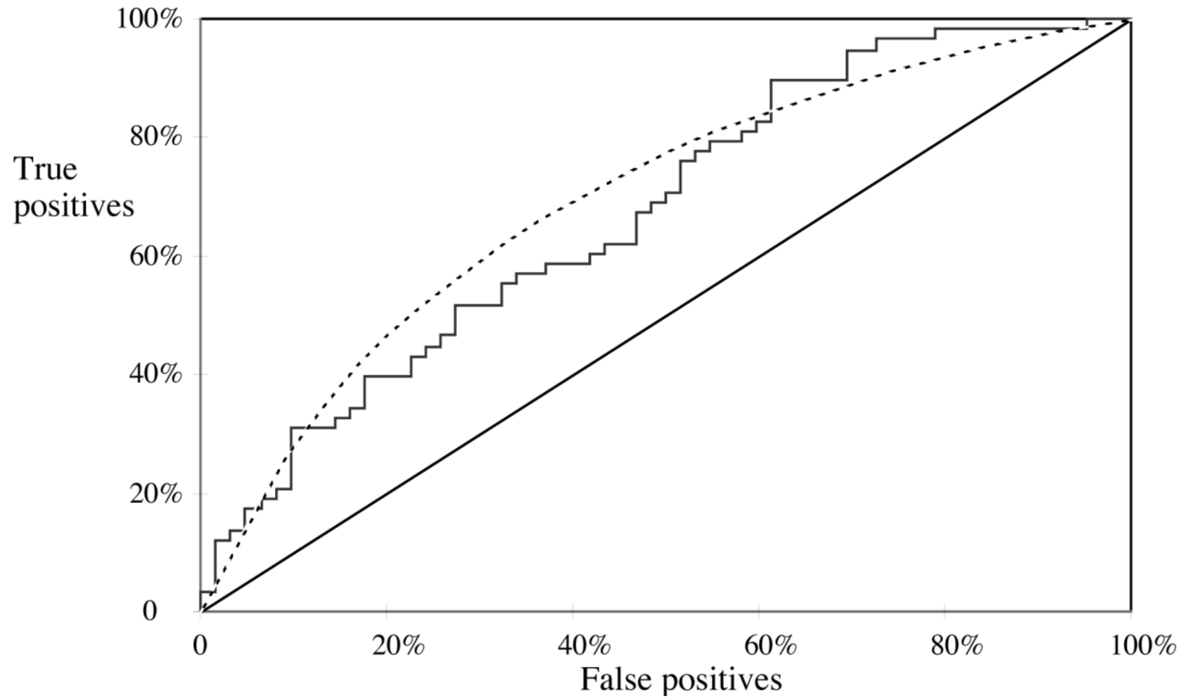
- What would you optimize for now? Precision or Recall?

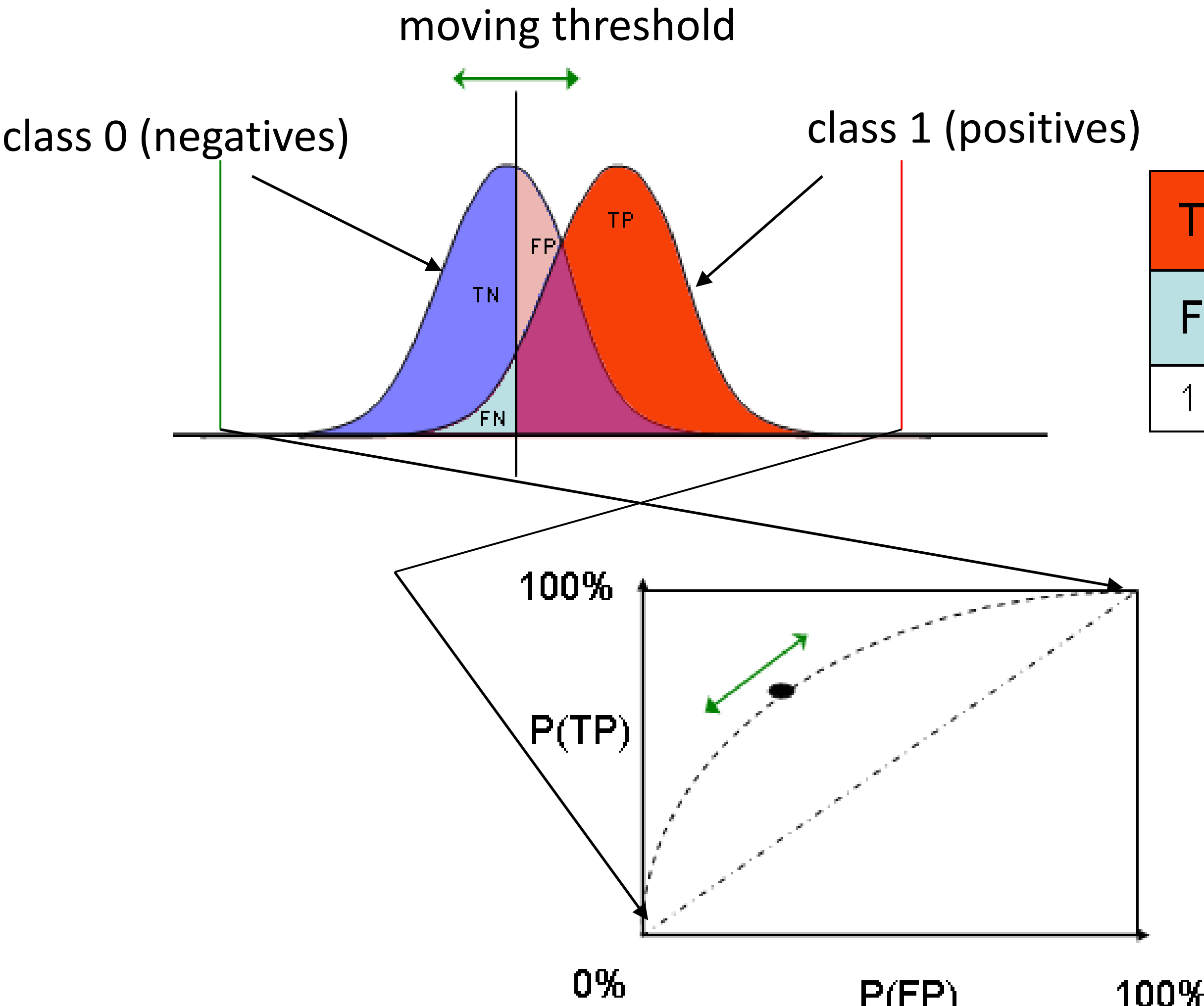
Precision vs Recall-Example 2

- **Answer:**
  - Having inexperienced or irrelevant candidates won’t hurt as long as you catch all the relevant and talented ones.
  - Therefore, Recall is the metric you should optimize for. If you instead optimize for Precision, there is a possibility of excluding talented candidates, which is not desired.

ROC Curves

- Stands for “receiver operating characteristic”
- Used in signal detection to show tradeoff between hit rate and false alarm rate over noisy channel

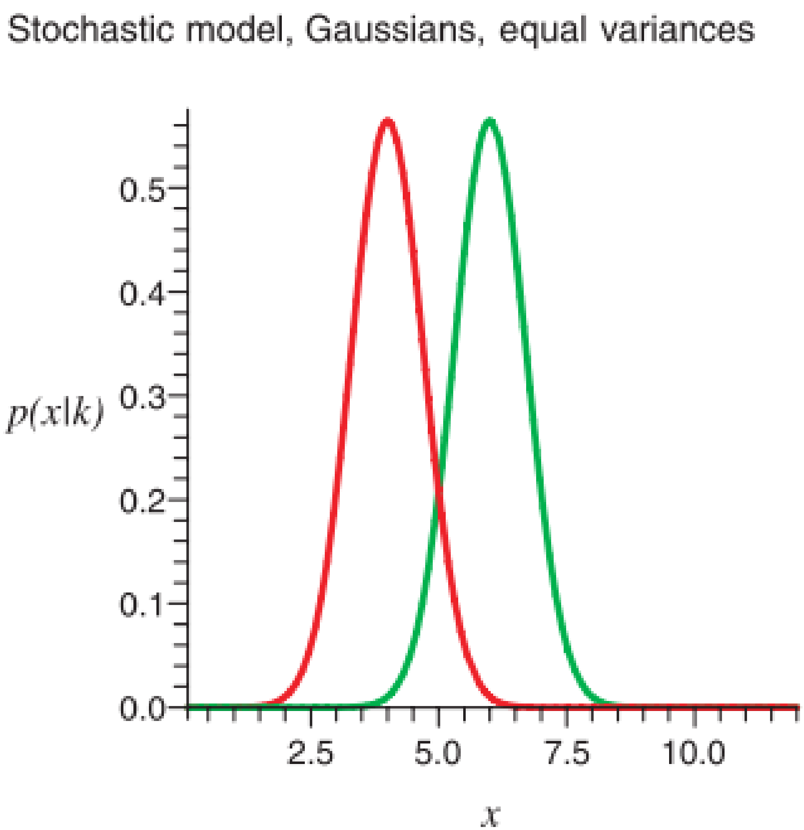


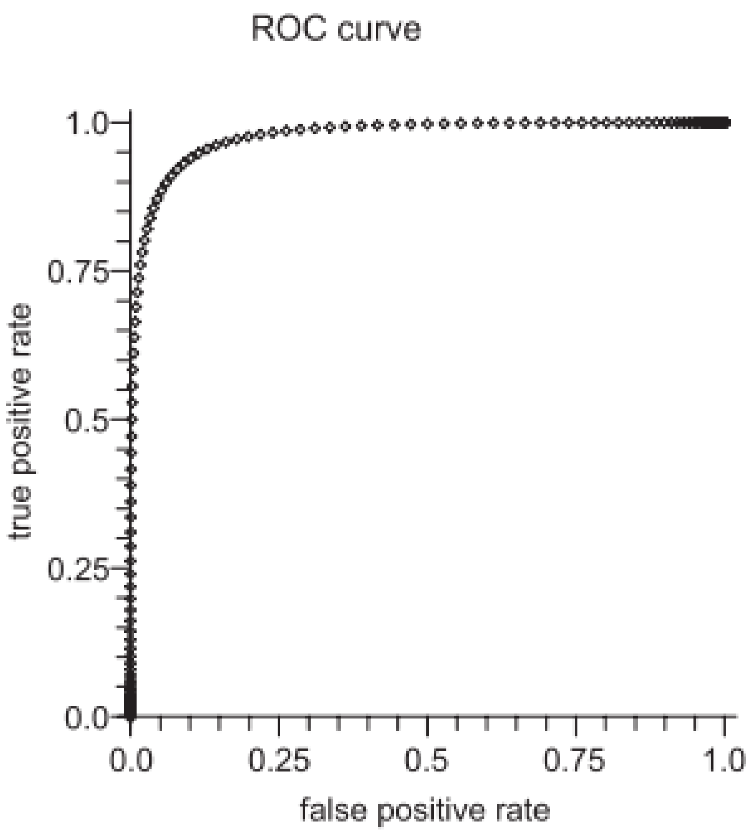


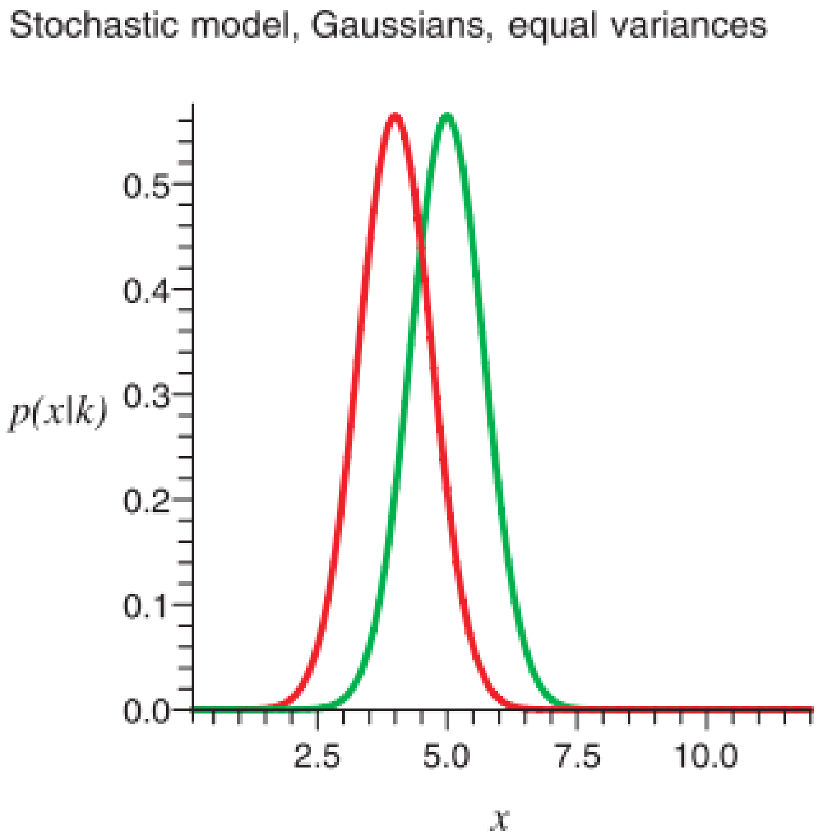
TP	FP
FN	TN
1	1

Identify a threshold in your classifier that you can shift.

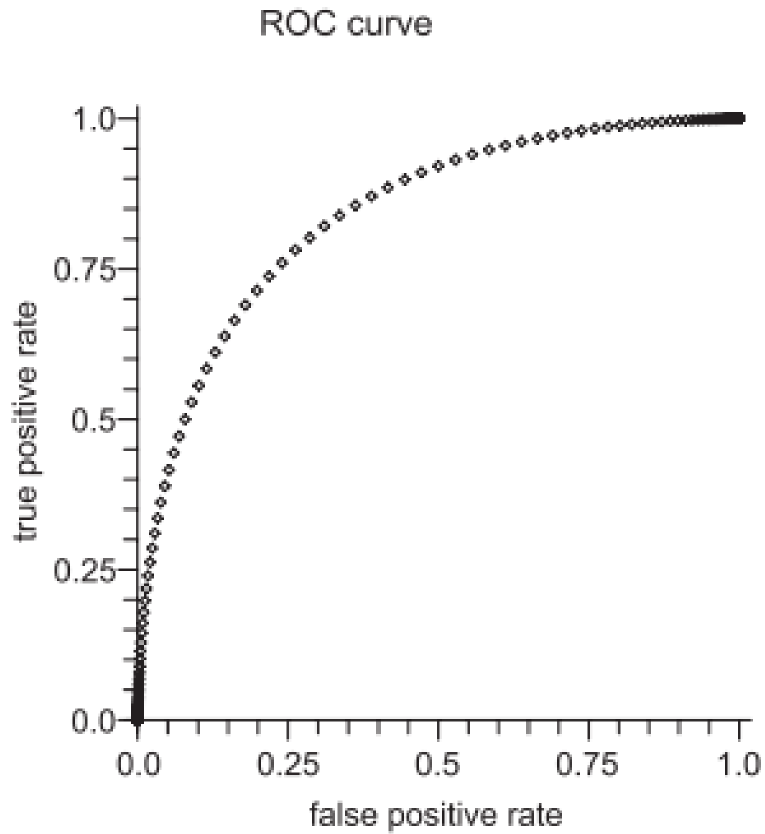
Plot ROC curve while you shift that parameter.


Stochastic model, Gaussians, equal variances
p(x|k)
0.0
0.1
0.2
0.3
0.4
0.5
2.5
5.0
7.5
10.0
x

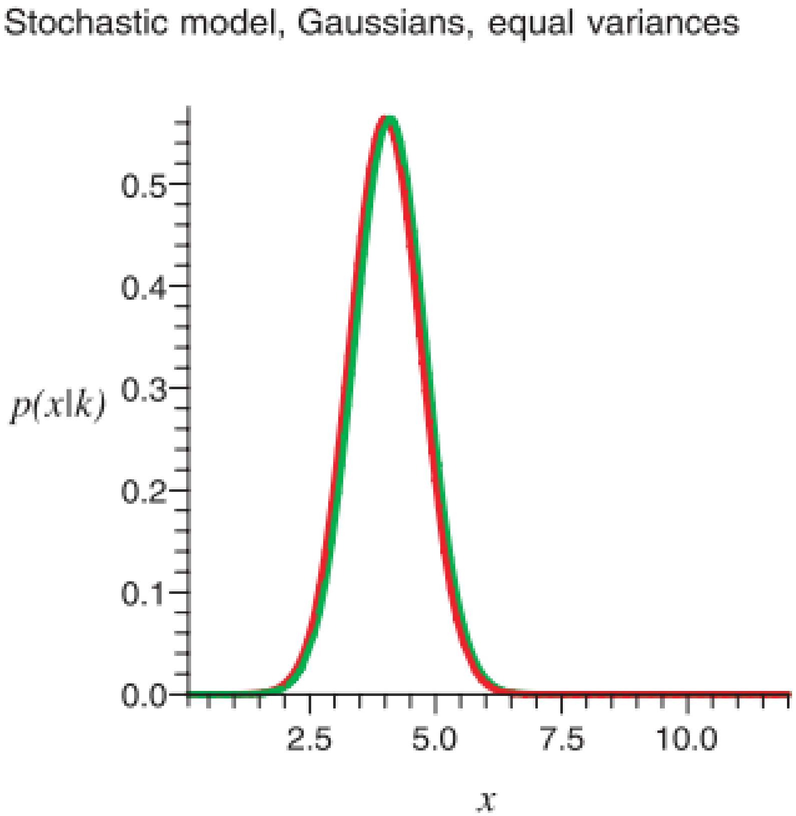

ROC curve
true positive rate
0.0
0.25
0.5
0.75
1.0
0.0
0.25
0.5
0.75
1.0
false positive rate

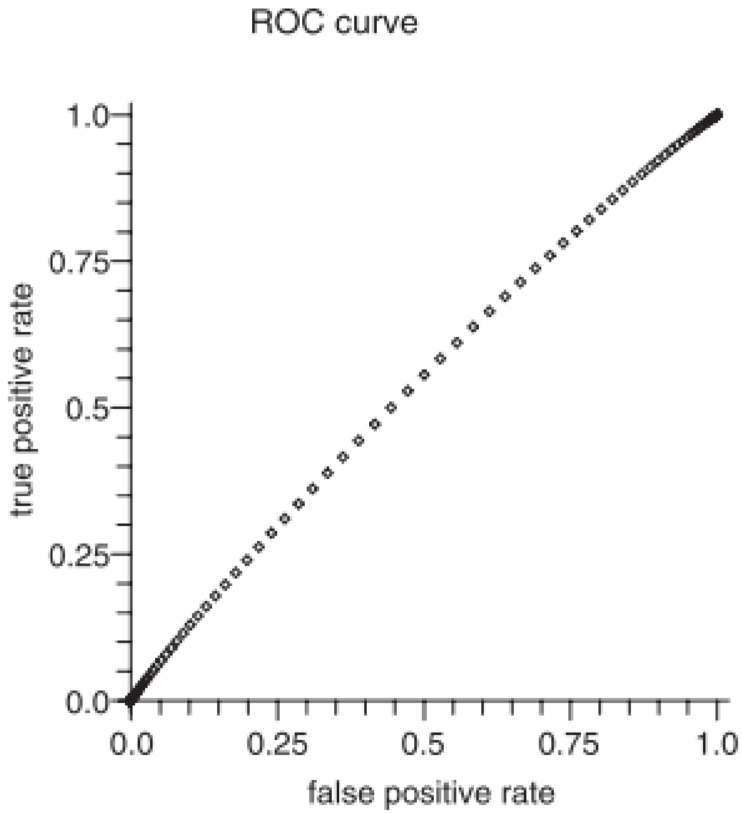


Stochastic model, Gaussians, equal variances
p(x|k)
0.0
0.1
0.2
0.3
0.4
0.5
2.5
5.0
7.5
10.0
x



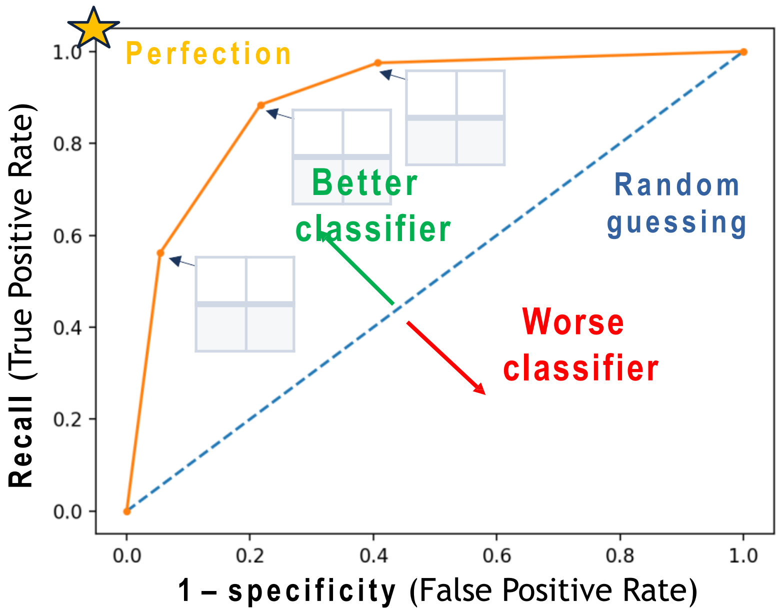
ROC curve
true positive rate
0.0
0.25
0.5
0.75
1.0
false positive rate


Stochastic model, Gaussians, equal variances
p(x|k)
0.5
0.4
0.3
0.2
0.1
0.0
2.5
5.0
7.5
10.0
x


ROC curve
true positive rate
1.0
0.75
0.5
0.25
0.0
0.0
0.25
0.5
0.75
1.0
false positive rate

ROC

ROC curves are a very general way to **represent and compare** the performance of different models (on a binary classification task)



Observations

- (0,0) : classify always negative
- (1,1) : classify always positive
- Diagonal line: random classifier
- Below diagonal line: worse than random classifier
- Different classifiers can be compared

ROC curve (Cnt'd)

- Accuracy is measured by ***the area under the ROC curve***. (AUC) An area of 1 represents a perfect test; an area of .5 represents a worthless test:

  - .90-1 = excellent
  - .80-.90 = good
  - .70-.80 = fair
  - .60-.70 = poor
  - .50-.60 = fail

Multiclass classification

- For **Multiclass prediction** task, the result is usually displayed in confusion matrix where there is a row and a column for each class,
  - Each matrix element shows the number of test instances for which the actual class is the row and the predicted class is the column
  - Good results correspond to large numbers down the diagonal and small values (ideally zero) in the rest of the matrix

Classified as →	a	b	c
A	TP_{aa}	FN_{ab}	FN_{ac}
B	FP_{ab}	TN_{bb}	***FN_{bc}***
C	FP_{ac}	***FN_{cb}***	TN_{cc}

Multiclass classification (Cont'd)

- For example in three classes task {a , b , c} with the confusion matrix below, if we selected a to be the class of interest then

$$\text{True positives for class } \boldsymbol{a} \ = \text{TP}_{\text{aa}}$$

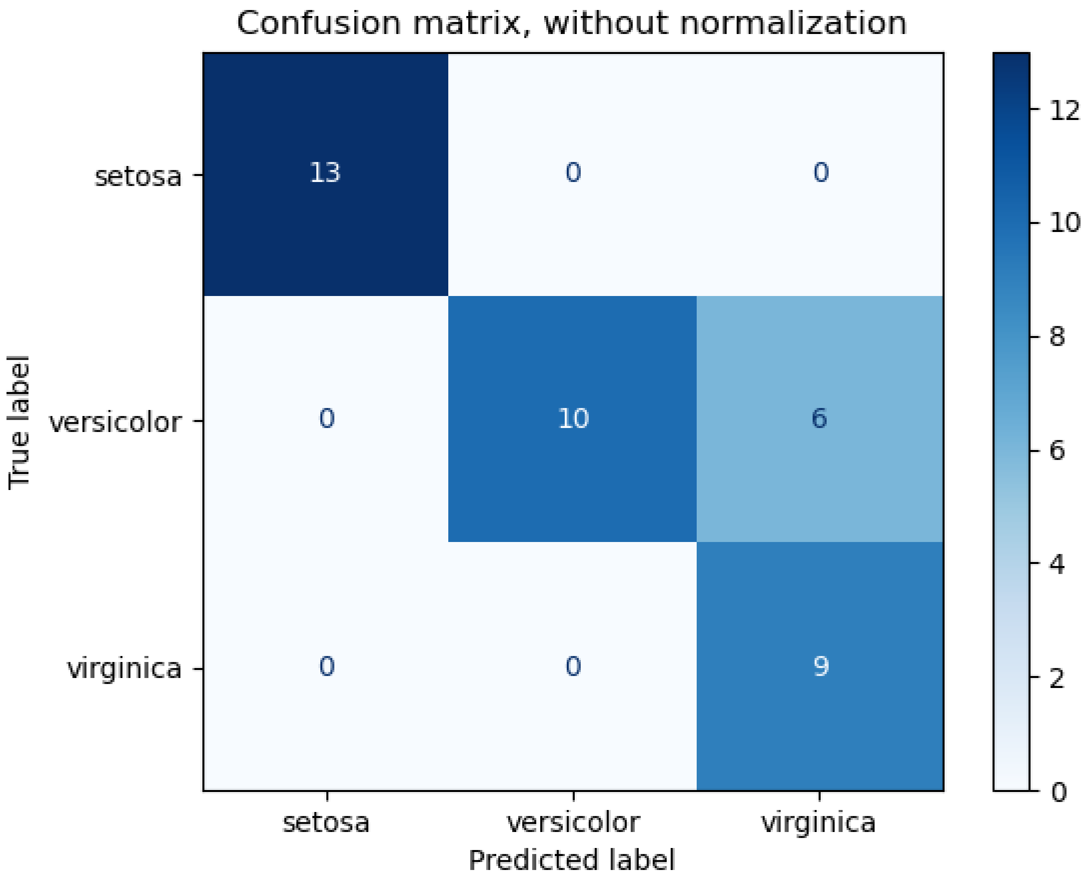
$$\text{True Negatives for class } \boldsymbol{a} \ = \text{TN}_{\text{cc}} + \text{TN}_{\text{bb}}$$

$$\text{False Positives for class } \boldsymbol{a} = \text{FP}_{\text{ab}} + \text{FP}_{\text{ac}}$$

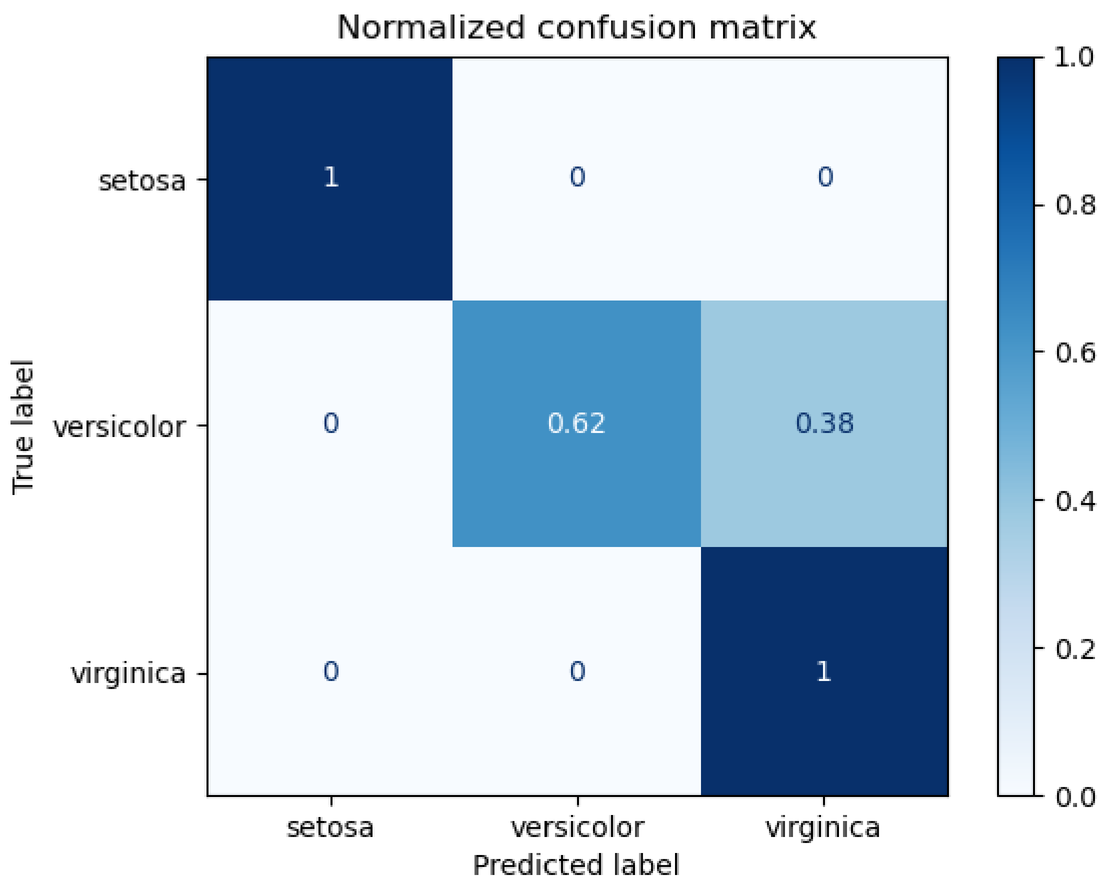
$$\text{False Negatives for class } \boldsymbol{a} = \text{FN}_{\text{ab}} + \text{FN}_{\text{ac}}$$

- Note that we don't care about the values (FNcb & FNbc) as we are considered with evaluating how the classifier is performing with class a, so the misclassifications between the other classes is out of our interest.

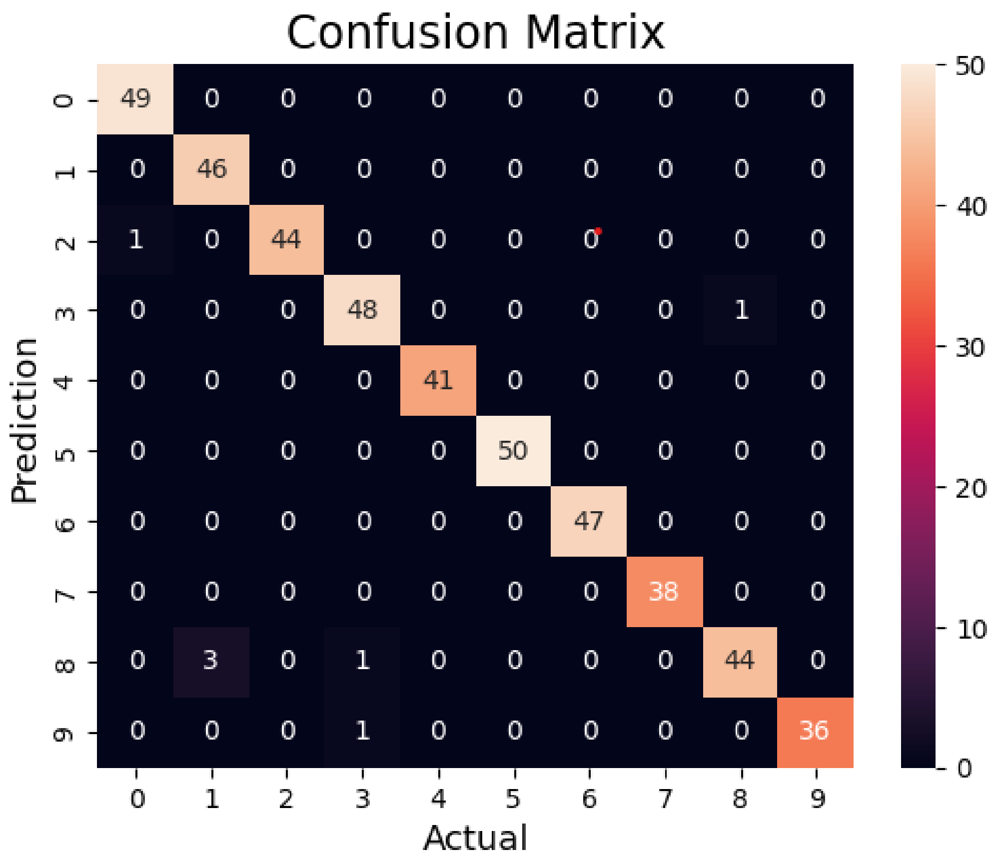
Confusion matrix For Multi-class Classification



Confusion matrix For Multi-class Classification



Confusion matrix For Multi-class Classification



Confusion Matrix for multiclass Classifications

Multiclass classification (Cont'd)

- To calculate overall model performance, we take their weighted average to evaluate the overall performance of the classifier.

- Averaged per category (macro average) :
  - Gives equal weight to each class, including rare ones

$$Recall_{macro} = \frac{\sum_{i=1}^{M} Recall_i}{M}, \quad where\ M\ is\ number\ of\ classes$$

$$Precision_{macro} = \frac{\sum_{i=1}^{M} Precision_i}{M}, \quad where\ M\ is\ number\ of\ classes$$

Multiclass classification (Cont'd)

- Micro Average:
  - Obtained from true positives (TP), false positives (FP), and false negatives (FN) for each class, and F-measure is the harmonic mean of micro-averaged precision and recall
  - Micro average gives equal weight to each sample regardless of its class.
  - They are dominated by those classes with the large number of samples.

$$Recall_{micro} = \frac{\sum_{i=1}^{M} TP_i}{\sum_{i=1}^{M} TP_i + \sum_{i=1}^{M} FN_i}$$

$$Precision_{micro} = \frac{\sum_{i=1}^{M} TP_i}{\sum_{i=1}^{M} TP_i + \sum_{i=1}^{M} FP_i}$$

Example: Area under ROC (AUC)

Dataset	J48	BayesNaive	Multi layer Perceptron	SVM	K Star	CART
approach_1	0.57	0.41	0.52	0.45	0.66	0.58
approach_2	0.48	0.43	0.66	0.5	0.64	0.58
approach_3_10+threshold	0.61	0.63	0.44	0.48	0.53	0.48
approach_3_15+threshold	0.55	0.73	0.59	0.59	0.65	0.54
approach_3_7+threshold	0.62	0.47	0.32	0.39	0.39	0.54
approach_4_10_threshold	0.63	0.49	0.47	0.44	0.55	0.53
approach_4_15_threshold	0.76	0.43	0.44	0.53	0.65	0.53
approach_4_7_threshold	0.35	0.46	0.48	0.51	0.44	0.48

Example: F-Measure

Dataset	J48	BayesNaive	Multi layer Perceptron	SVM	K Star	CART
approach_1	0.4	0.21	0.4	0.03	0.51	0.4
approach_2	0.06	0.31	0.45	0	0.54	0.44
approach_3_10+threshold	0.59	0.52	0.46	0.42	0.49	0.5
approach_3_15+threshold	0.56	0.61	0.54	0.52	0.48	0.53
approach_3_7+threshold	0.61	0.46	0.32	0.37	0.38	0.62
approach_4_10_threshold	0.5	0.36	0.32	0.23	0.29	0.18
approach_4_15_threshold	0.67	0.24	0.35	0.38	0.33	0.18
approach_4_7_threshold	0.07	0.26	0.31	0.27	0.27	0.04