



**POLITECNICO**  
MILANO 1863

# Data Analysis for Smart Agriculture

- Classification -

*Prof. Matteo Matteucci – [matteo.matteucci@polimi.it](mailto:matteo.matteucci@polimi.it)*

# Classification in a nutshell

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Gentoo	Biscoe	45.8	14.2	219.0	4700.0	Female
1	Gentoo	Biscoe	50.8	15.7	226.0	5200.0	Male
2	Chinstrap	Dream	46.9	16.6	192.0	2700.0	Female
3	Adelie	Torgersen	41.4	18.5	202.0	3875.0	Male
4	Adelie	Torgersen	34.6	21.1	198.0	4400.0	Male

The target variable can assume one value out of a finite set of **K classes**

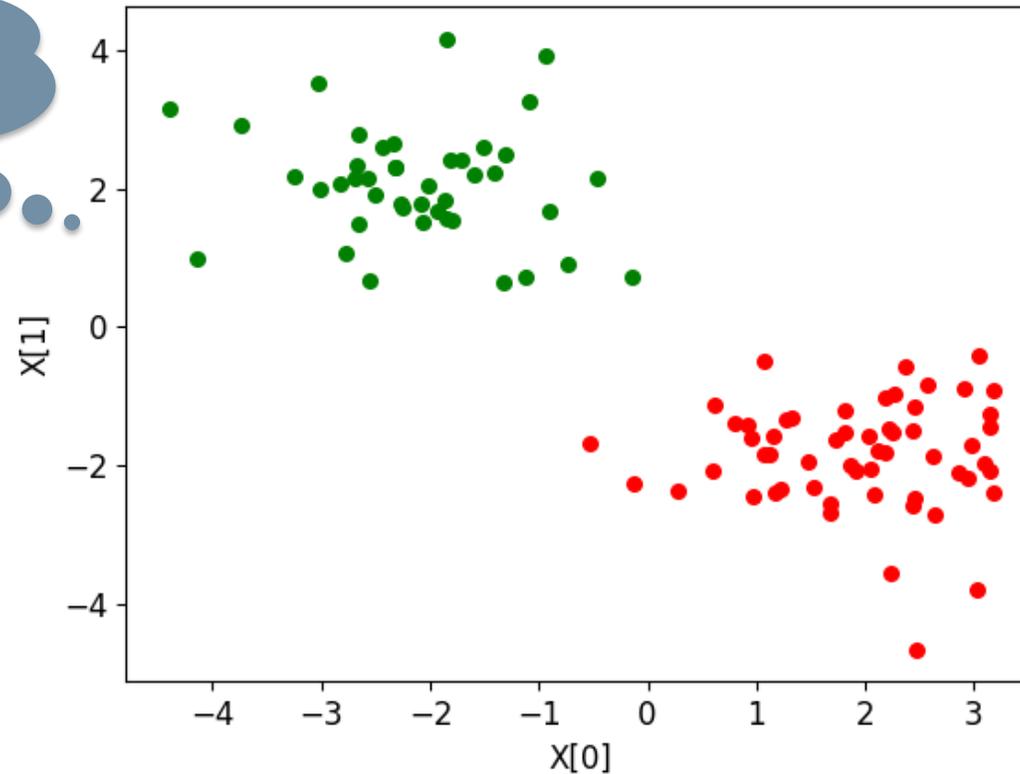
- Binary Classification:  $K=2$ , i.e.,  $\mathcal{C}=\{c_1, c_2\}$ . Usually,  $\{0, 1\}$ , or  $\{-1, 1\}$
- Multi-Class Classification:  $K>2$

A classifier has to guess the right one using a **vector of features** in order to minimize the overall number of errors, i.e., the error rate.

# It all starts from some data

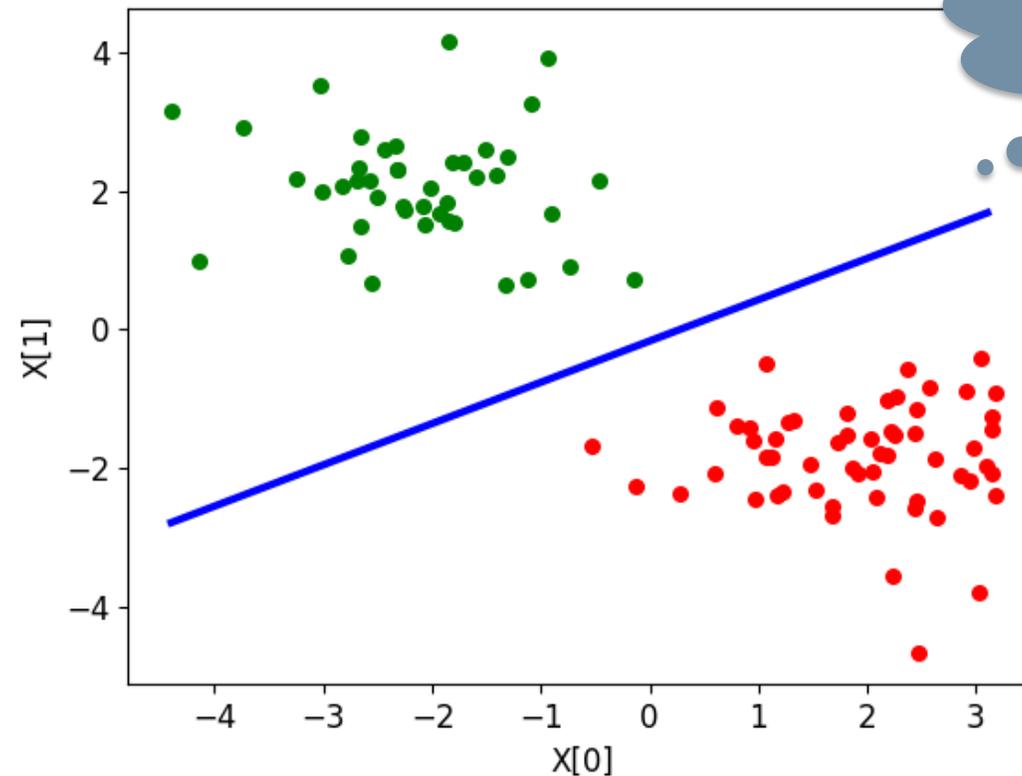
In classification, data have been labeled according to a certain concept (good/bad, positive/negative, etc.) by a supervisor

*Vector space model.*



# It all starts from some data

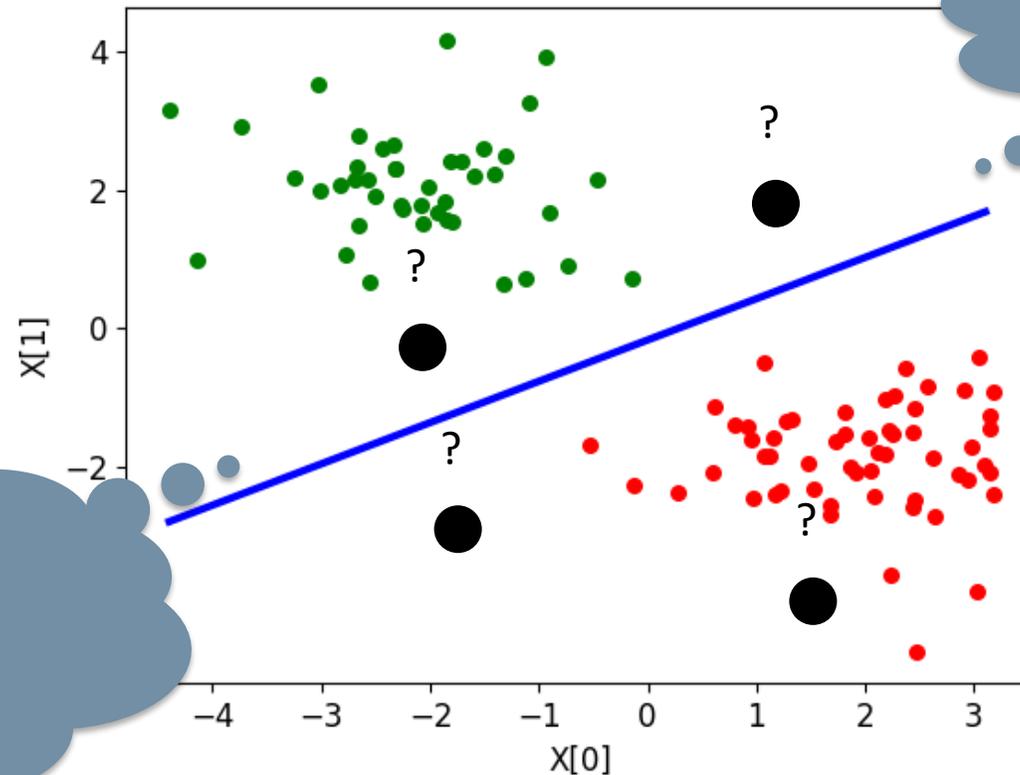
In classification, data have been labeled according to a certain concept (good/bad, positive/negative, etc.) by a supervisor



*The model finds a decision boundary between the classes.*

# It all starts from some data ... and a quest!

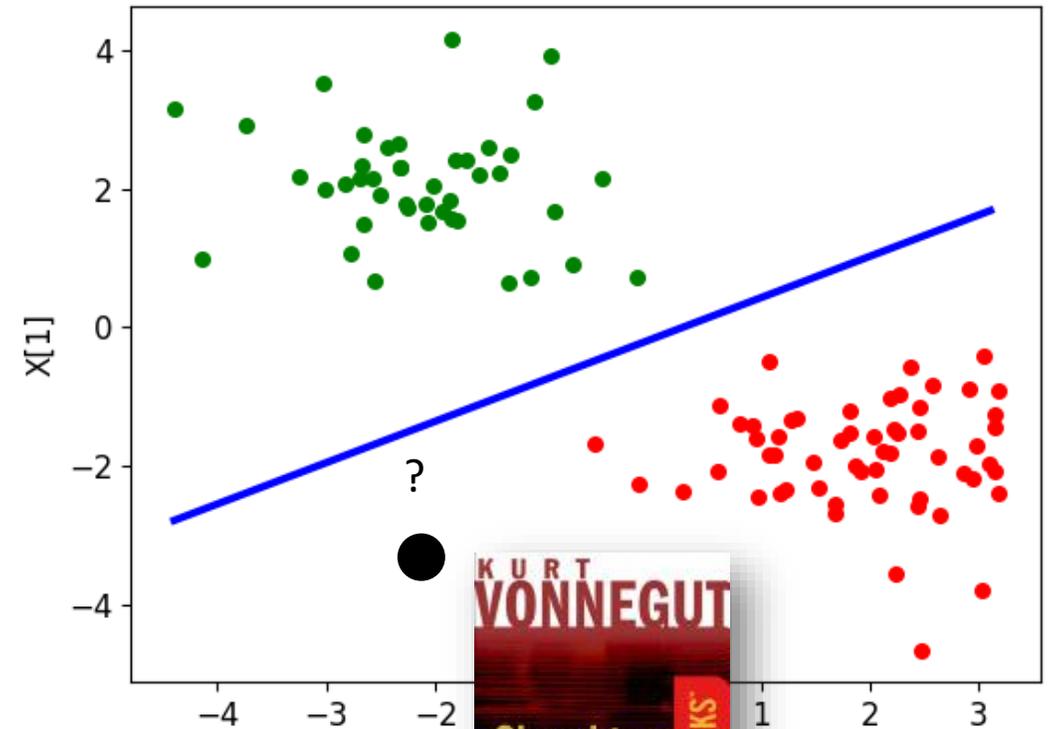
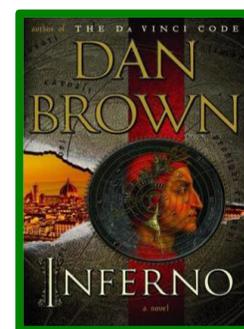
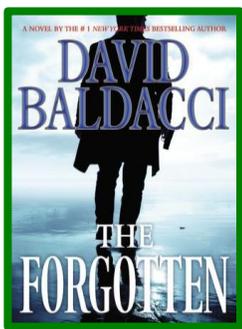
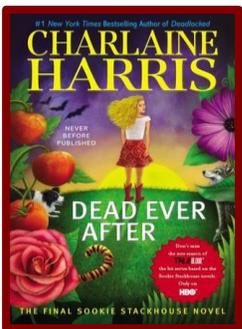
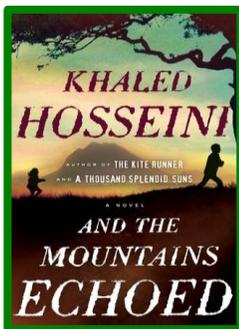
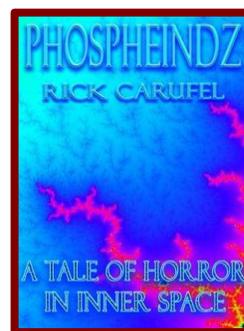
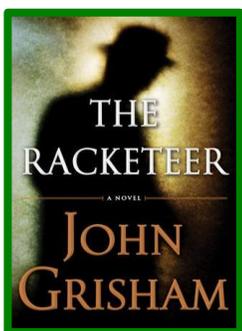
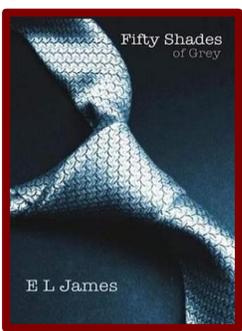
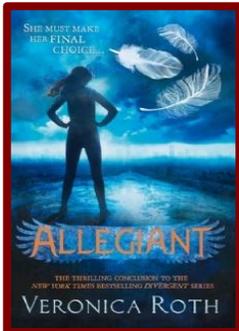
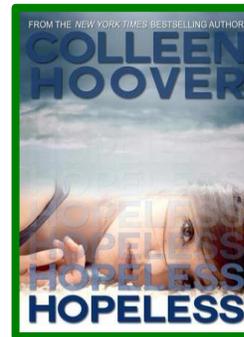
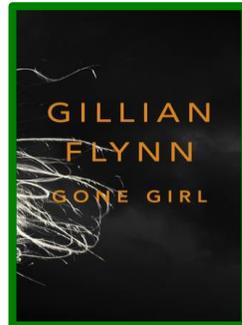
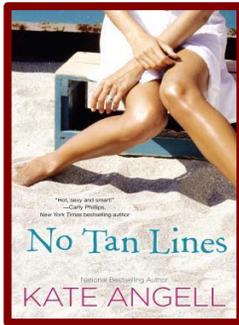
In classification, data have been labeled according to a certain concept (good/bad, positive/negative, etc.) by a supervisor



*The model finds a decision boundary between the classes.*

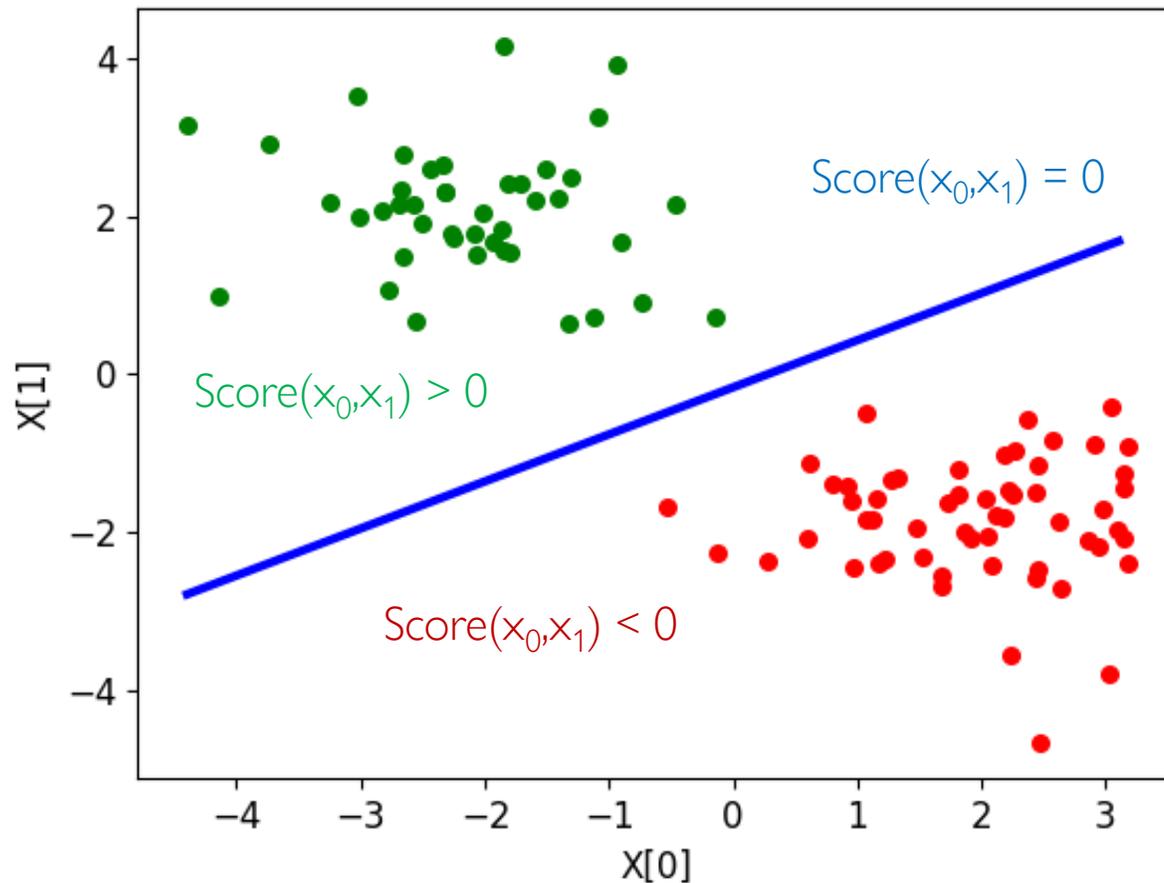
*Classification  
=  
Model Building +  
Model Usage*

# Data and Points



# Linear Classifiers: The Idea

To predict the labels we check whether the value of  $\text{Score}(x_0, x_1)$  is positive or negative and we label the examples accordingly



$$\text{Score}(\vec{x}_i) = \sum_{j=0}^D w_j h_j(\vec{x}_i)$$

$$\hat{y}_i = \text{sign}(\text{Score}(\vec{x}_i)) = \begin{cases} +1 & \text{if } \text{Score}(\vec{x}_i) \geq 0 \\ -1 & \text{if } \text{Score}(\vec{x}_i) < 0 \end{cases}$$

# Linear Classifiers: The Idea

To predict the labels we check whether the value of  $\text{Score}(x_0, x_1)$  is positive or negative and we label the examples accordingly

We define a score function similar to the one used in linear regression,

$$\text{Score}(\vec{x}_i) = \sum_{j=0}^D w_j h_j(\vec{x}_i)$$

The label is determined by the sign of the score value,

$$\hat{y}_i = \text{sign}(\text{Score}(\vec{x}_i)) \\ = \begin{cases} +1 & \text{if } \text{Score}(\vec{x}_i) \geq 0 \\ -1 & \text{if } \text{Score}(\vec{x}_i) < 0 \end{cases}$$

# Logistic Regression

Instead of computing the label, it computes the probability of assigning a class to an example, that is,

$$P(y_i|\vec{x}_i)$$

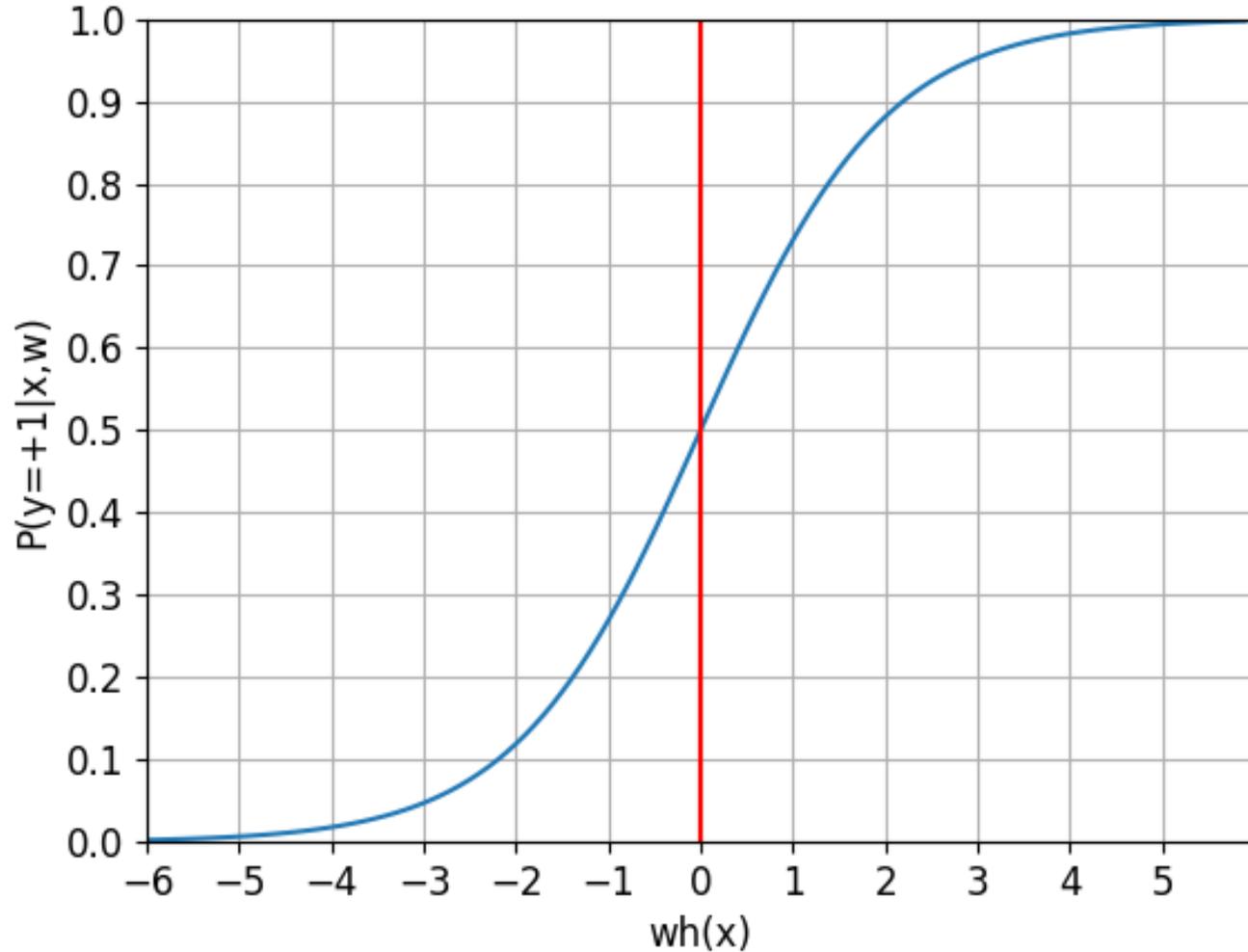
For this purpose, logistic regression assumes that,

$$P(\hat{y}_i = +1|\vec{x}_i) = \frac{1}{1 + e^{-Score(\vec{x}_i)}}$$

By making the score computation explicit and using  $h$  to identify all the feature transformation  $h_j$  we get,

$$P(\hat{y}_i = +1|\vec{x}_i, \vec{w}) = \frac{1}{1 + e^{-\vec{w}h(\vec{x}_i)}}$$

# Logistic Regression



$$P(\hat{y}_i = +1 | \vec{x}_i, \vec{w}) = \frac{1}{1 + e^{-\vec{w}h(\vec{x}_i)}}$$

# Logistic Regression

Logistic Regression search for the weight vector that corresponds to the highest likelihood

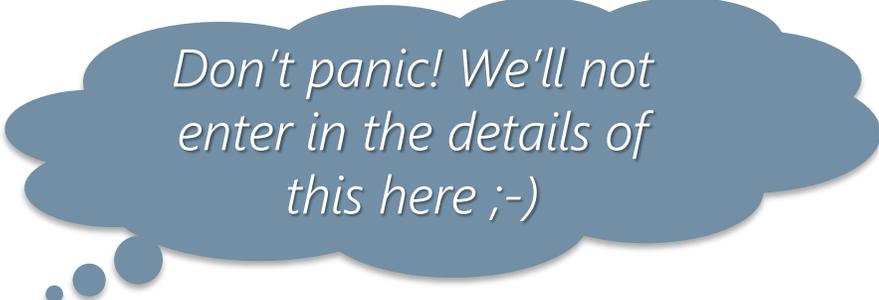
$$\ell(\vec{w}) = \prod_{I=0}^N P(y_i | \vec{x}_i, \vec{w})$$

Thus it performs a gradient ascent on the log likelihood function

$$\ell\ell(\vec{w}) = \ln \ell(\vec{w})$$

Which updates weight  $j$  using,

$$\frac{\partial \ell\ell}{\partial w_j} = \sum_{I=1}^N h_j(\vec{x}_i) (1[y_i = +1] - P(y = +1 | \vec{x}_i, \vec{w}))$$



*Don't panic! We'll not enter in the details of this here ;-)*

# Logistic Regression

To classify  $X$ , we select the class  $Y$  that maximizes probability  $P(Y=y | X)$  or

$$\frac{P(y = -1|x)}{P(y = +1|x)} > 1$$

which is equivalent to check

$$e^{-\vec{w}h(\vec{x})} > 1$$

By taking the natural logarithm of both sides, we obtain a linear classification rule that assign the label  $Y=-1$  if

$$\sum_{i=0}^D w_i h_i(x) < 0$$

*Logistic Regression is a linear classifier and Scores are related to class probabilities.*

# What if Variables Are Categorical?

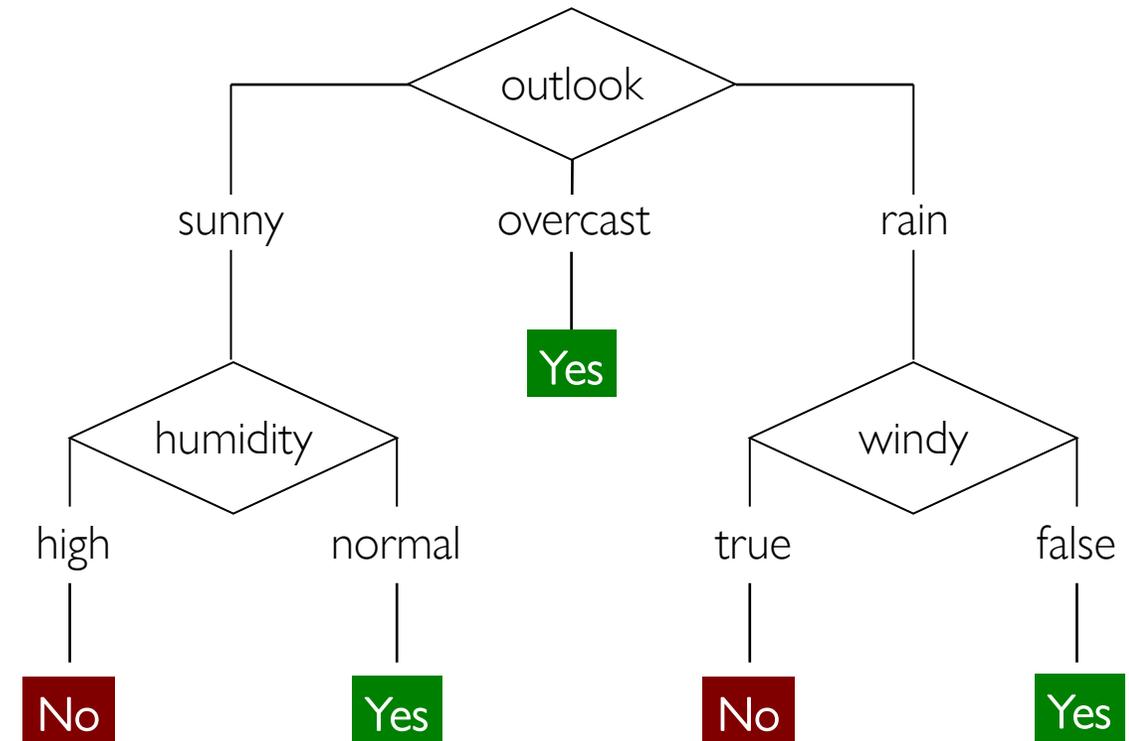
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

*One Hot Encoding*

overcast	rainy	sunny
0	0	1
0	0	1
1	0	0
0	1	0
0	1	0
0	1	0
1	0	0
0	0	1
0	0	1
0	1	0
0	0	1
1	0	0
1	0	0
0	1	0

# Decision Trees

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



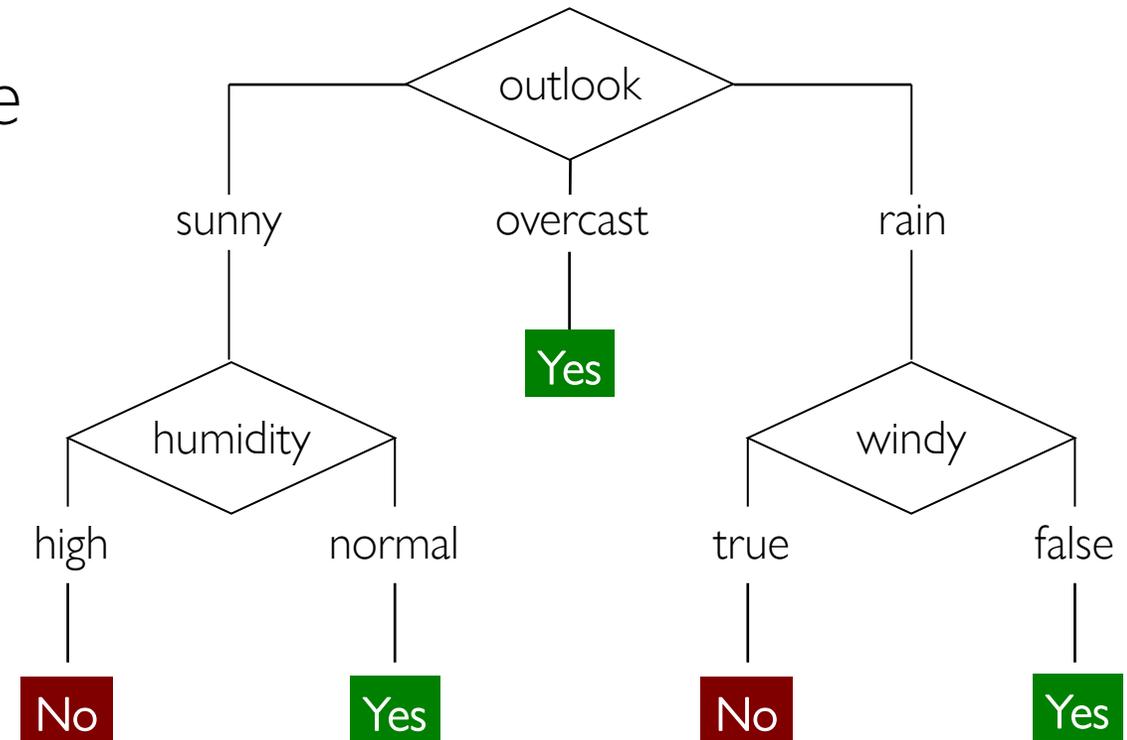
# What is a Decision Tree?

An internal node is a test on an attribute

A branch represents an outcome of the test, e.g., outlook=windy

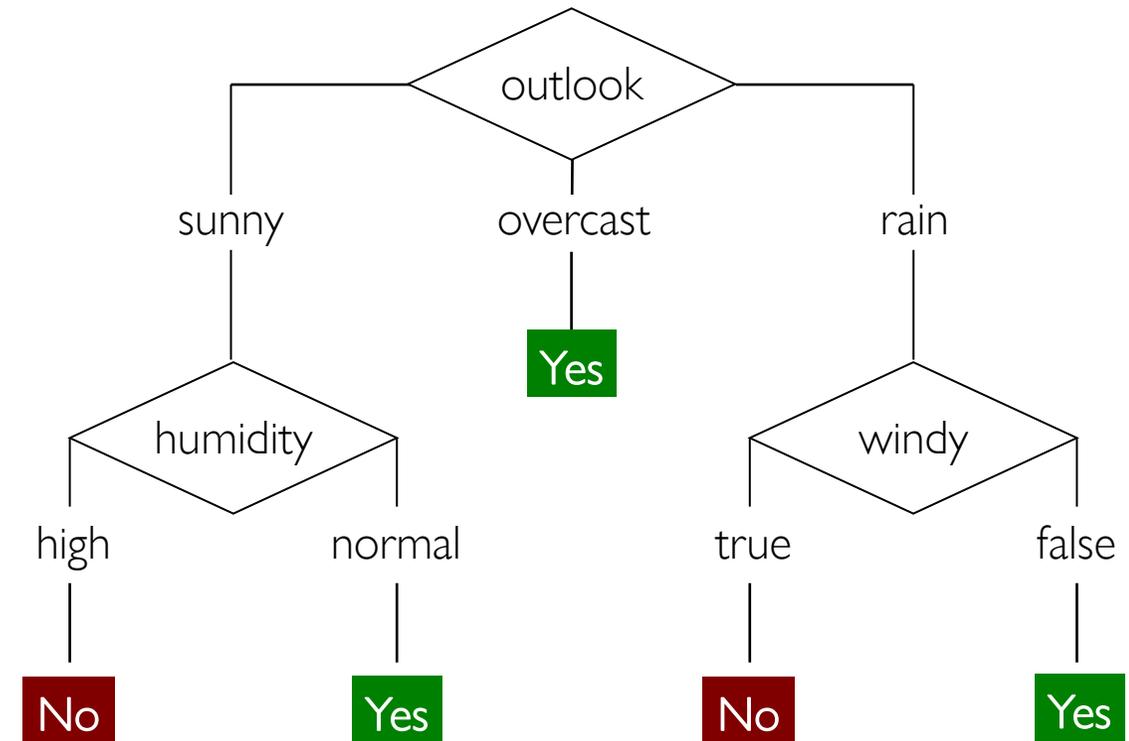
A leaf node represents a class label or class label distribution

A new case is classified by following a matching path to a leaf node



# Decision Trees

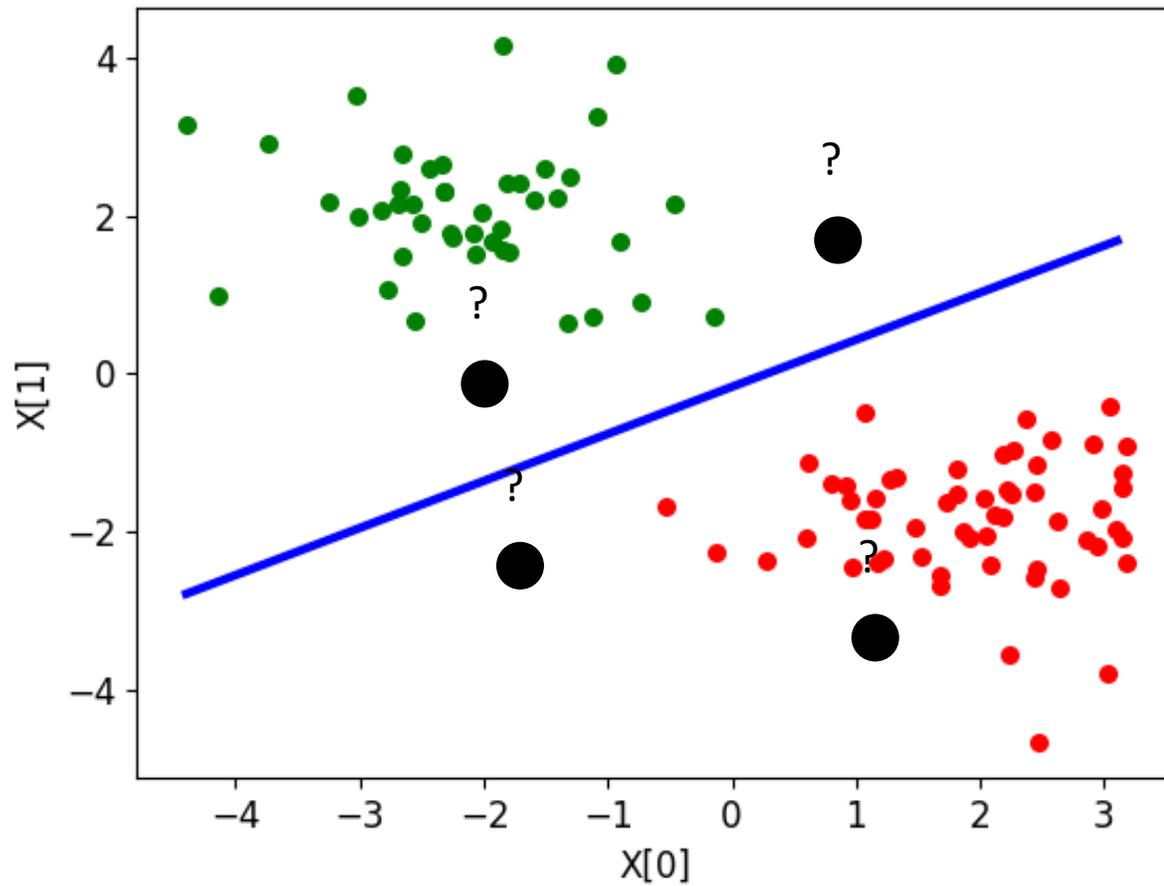
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



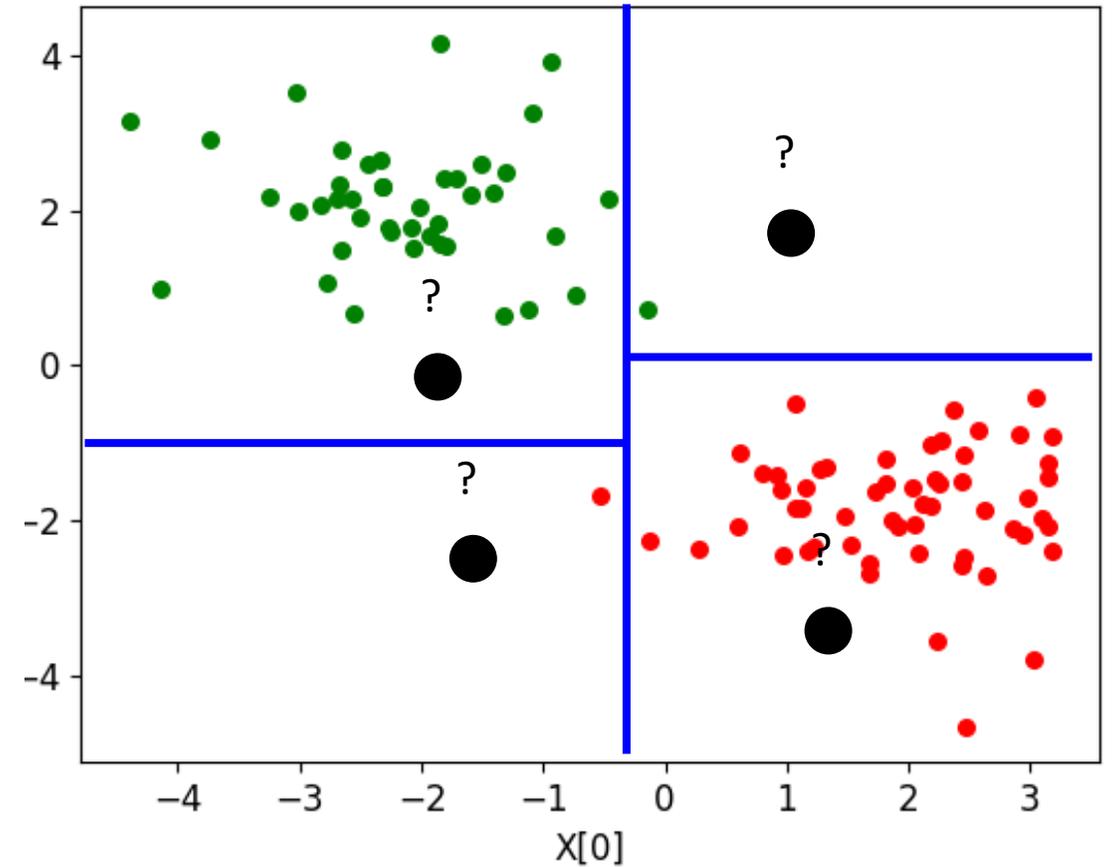
*Sequence of questions to get to the class*

# Geometric Interpretation

## Logistic Regression

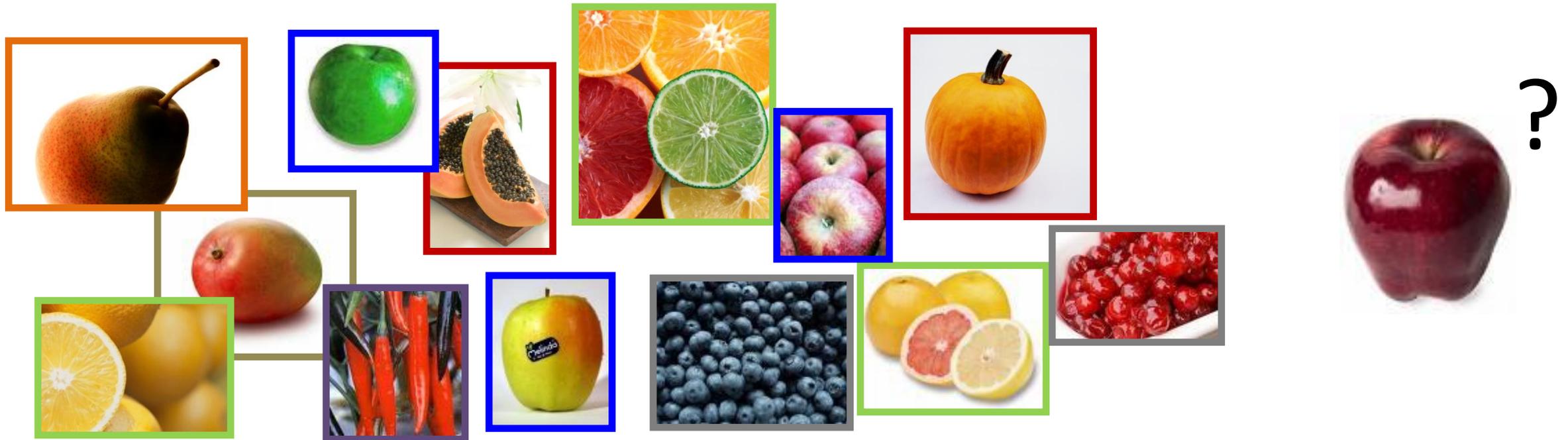


## Decision Tree



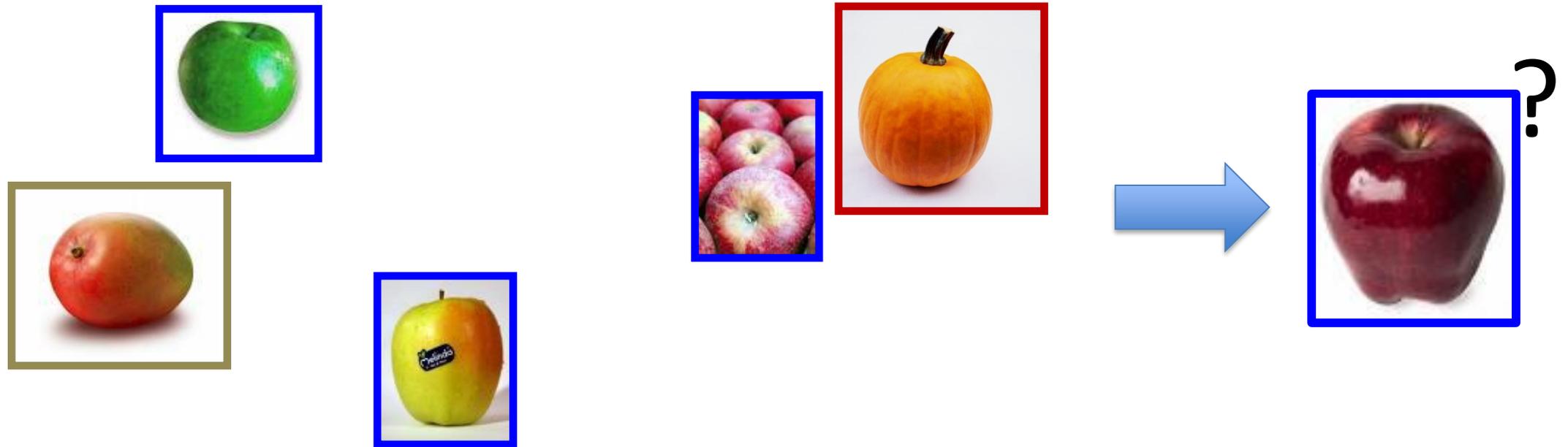
# Nearest Neighbors Classifier

To decide the label for an unseen example, consider the  $k$  examples (5) from the training data that are more similar to the unknown one



# Nearest Neighbors Classifier

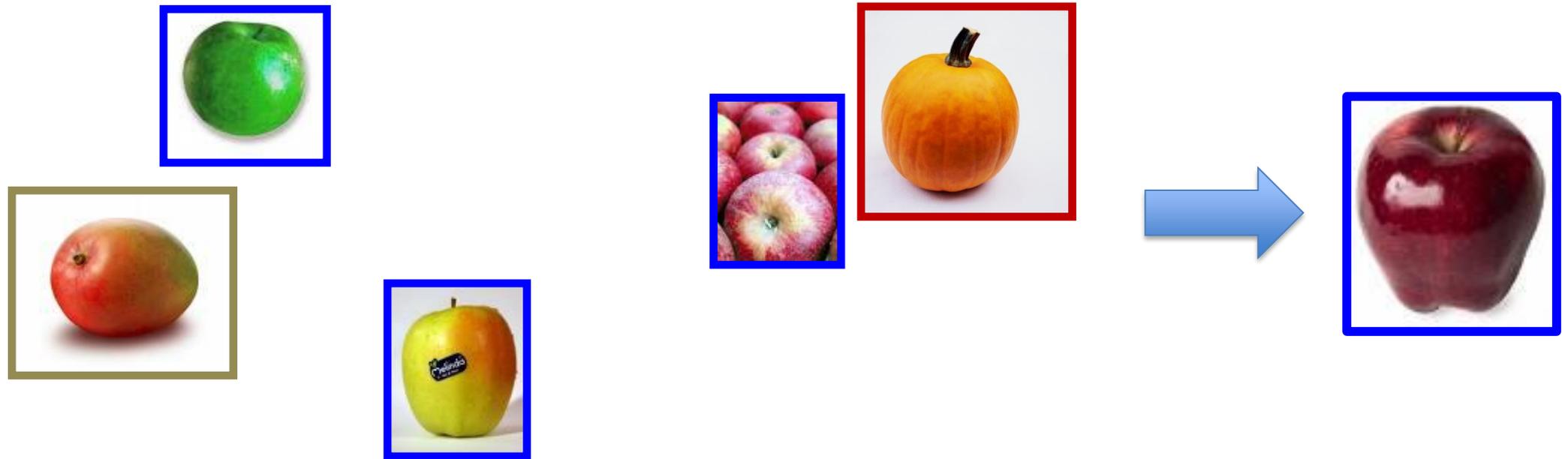
To decide the label for an unseen example, consider the  $k$  examples (5) from the training data that are more similar to the unknown one



Classify the unknown example using the most frequent class

# Nearest Neighbors Classifier

To decide the label for an unseen example, consider the  $k$  examples (5) from the training data that are more similar to the unknown one



Classify the unknown example using the most frequent class

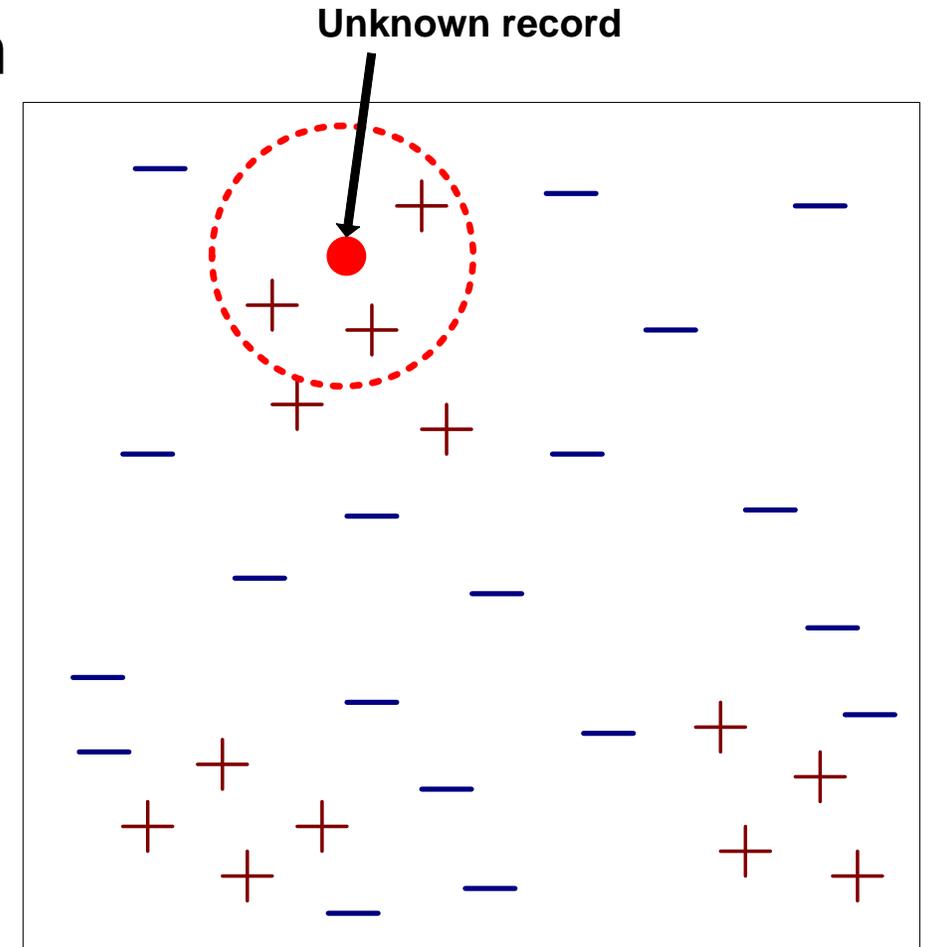
# Nearest Neighbors Classifier

To decide the label for an unseen example via Nearest Neighbors classification you need:

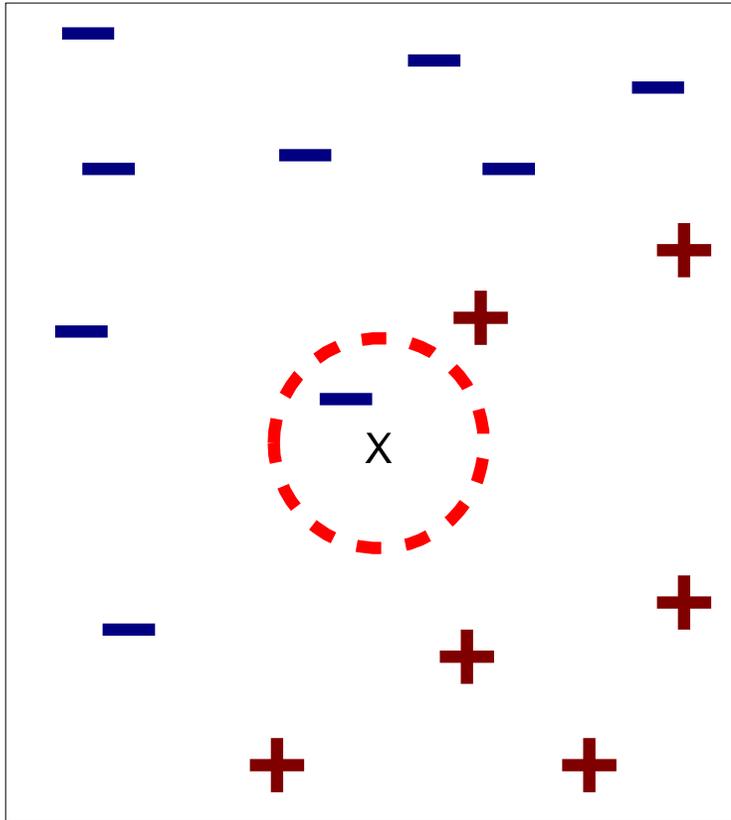
- The training dataset
- Similarity function (or distance metric)
- The value of  $k$ , of nearest neighbors to retrieve

Classification is then:

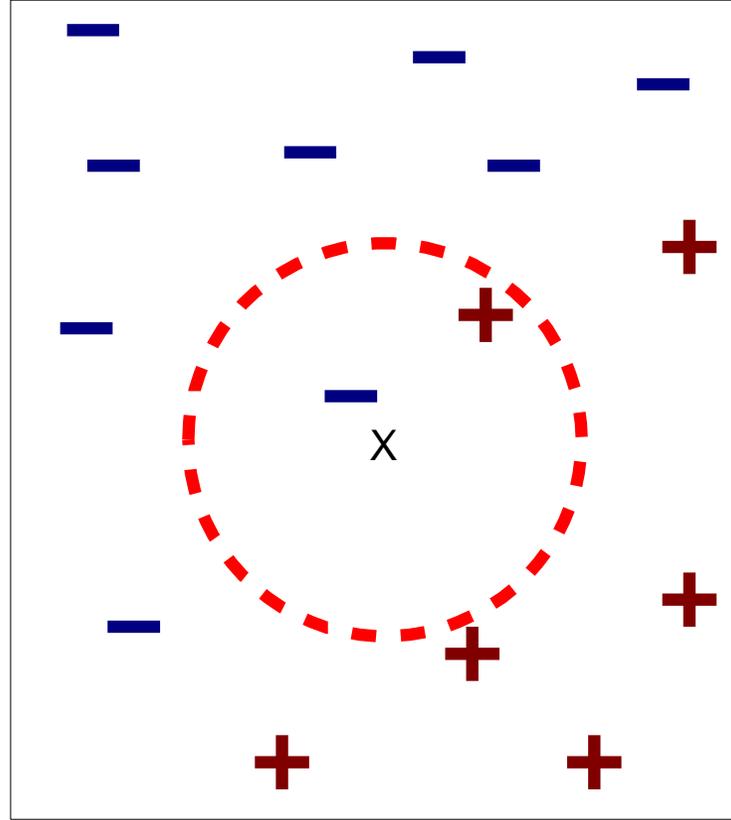
- Compute distance to other training records
- Identify the  $k$  nearest neighbors
- Use labels of nearest neighbors to determine the class of unknown record (e.g., by majority vote)



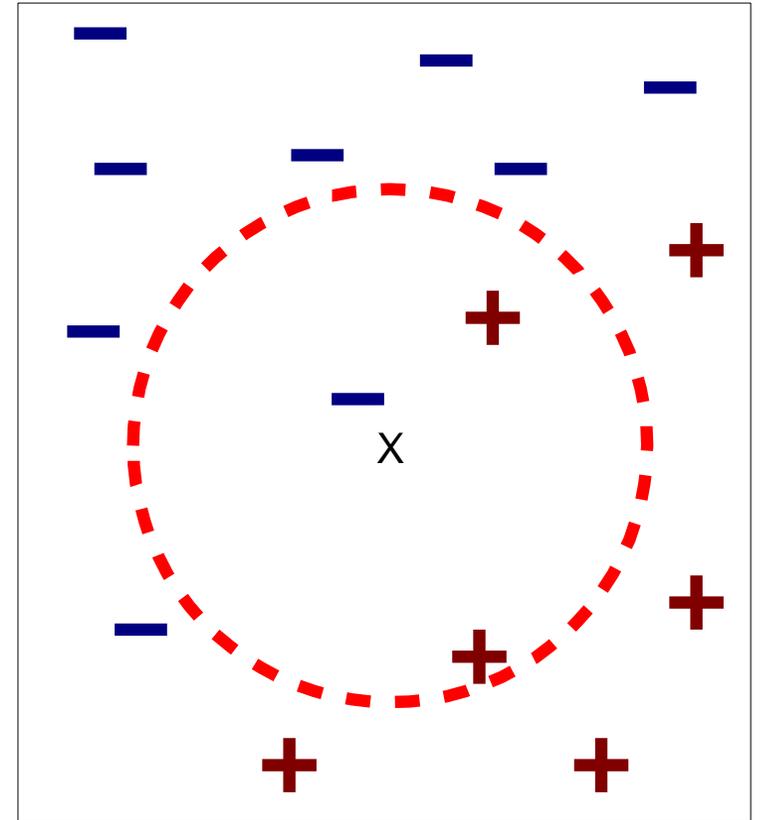
# K-Nearest Neighbors Classifier



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

# Instance Based Learning (Lazy Learners)

Store the training records only, no model is computed

Use the training records to predict an unknown class label

Att <sub>1</sub>	...	Att <sub>n</sub>	Class

Att <sub>1</sub>	...	Att <sub>n</sub>

*It might be slow without indexing!*

# What Similarity?

Euclidian distance is the typical function used to compute the similarity between two examples

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

To determine the class from nearest neighbor list

- Take the majority vote of class labels among the k neighbors
- Or weight the vote according to distance (e.g.,  $w = 1/d^2$ )

Other popular metrics are the Normalized Distance or the city-block (Manhattan) metric, i.e., the sum of absolute differences

# Normalization and Other Issues

Different attributes are measured on different scales so attributes might need to be normalized:

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$$

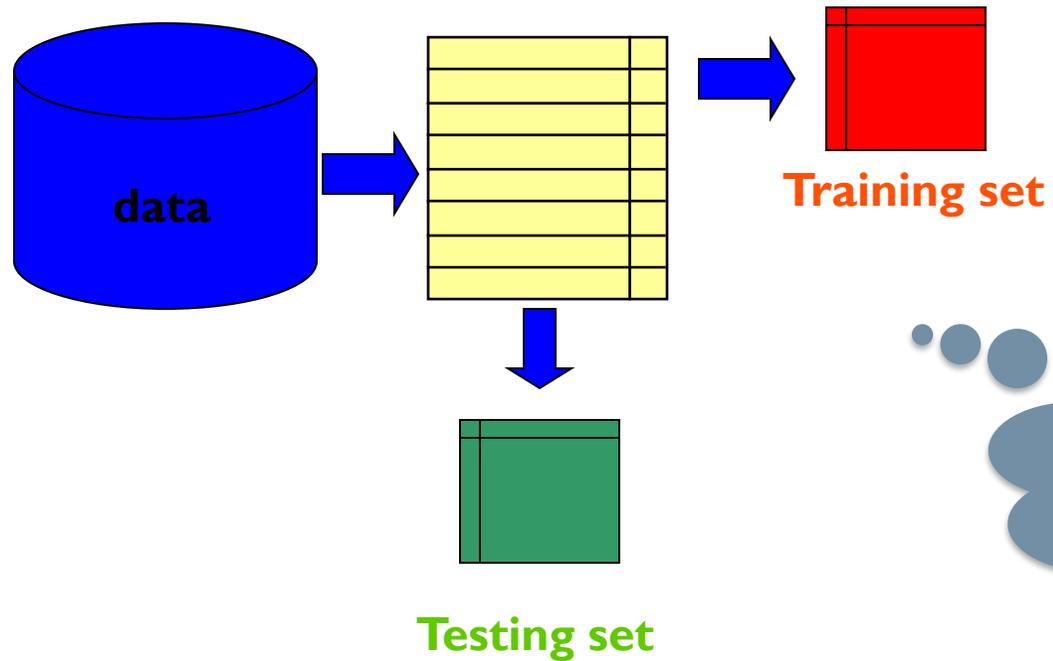
$$a_i = \frac{v_i - \text{Avg}(v_i)}{\text{StDev}(v_i)}$$

$v_i$  is the actual value of attribute  $I$ , and  $a_i$  is the normalized value

For nominal attributes, the distance is either 0 or 1

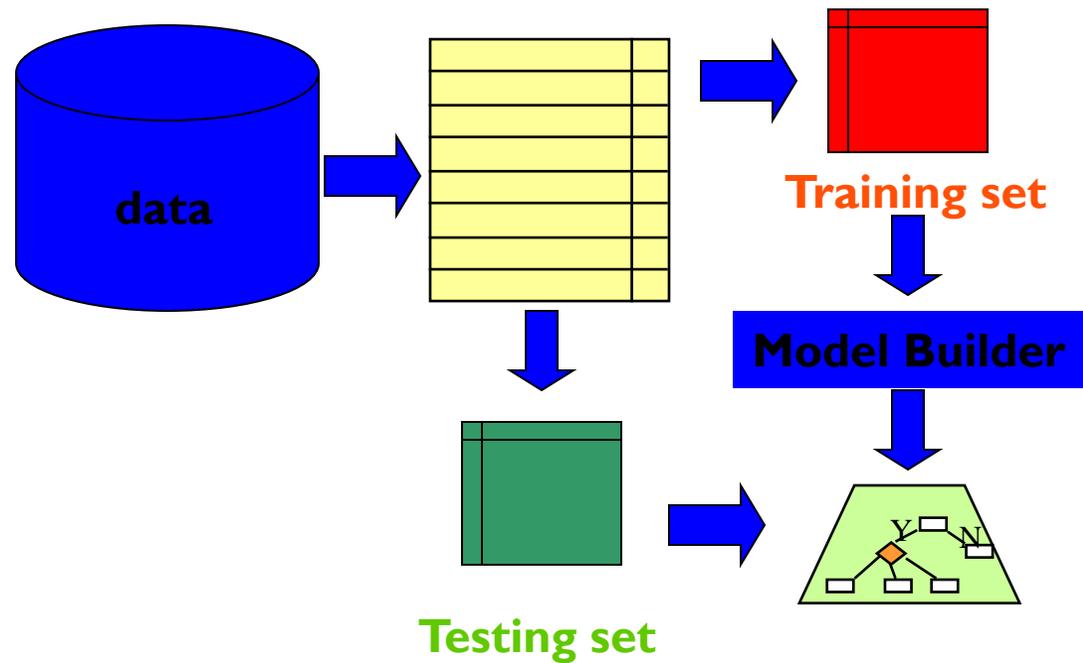
Missing values are usually assumed to be maximally distant (given normalized attributes)

# Fair Evaluation and Hyperparameter Tuning

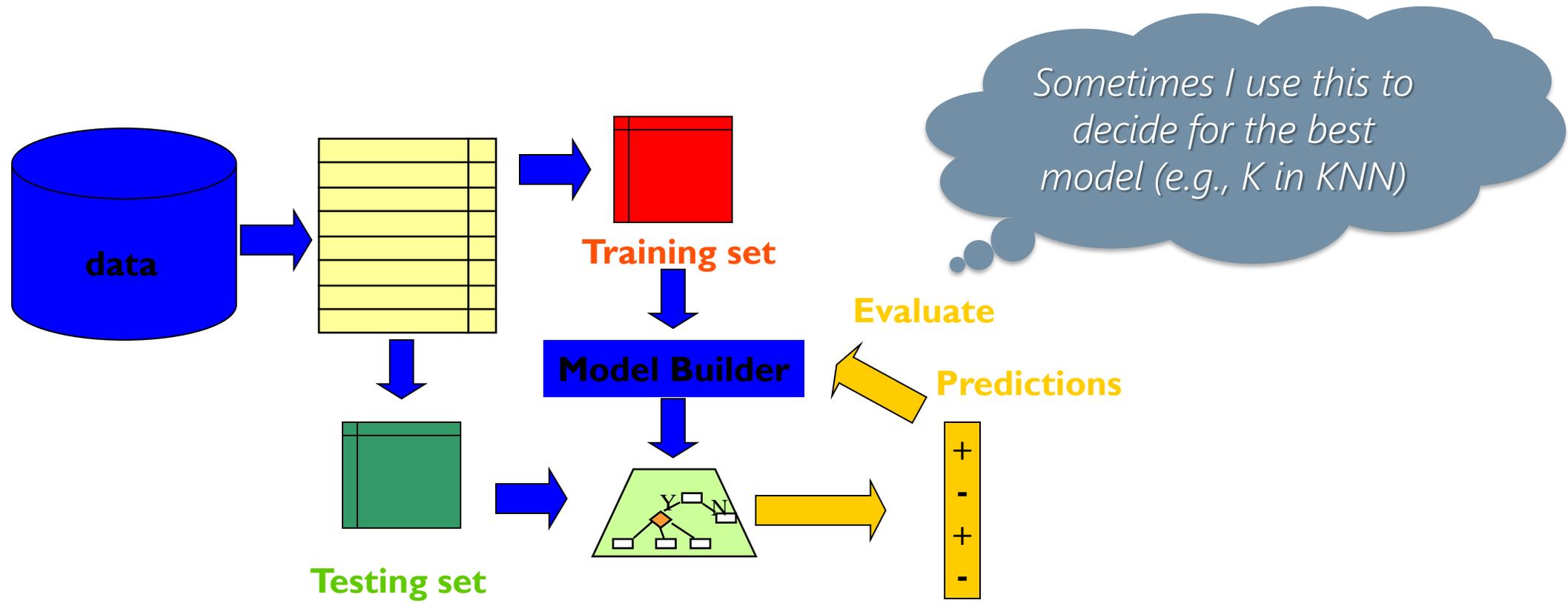


*Do not forget about stratified sampling!*

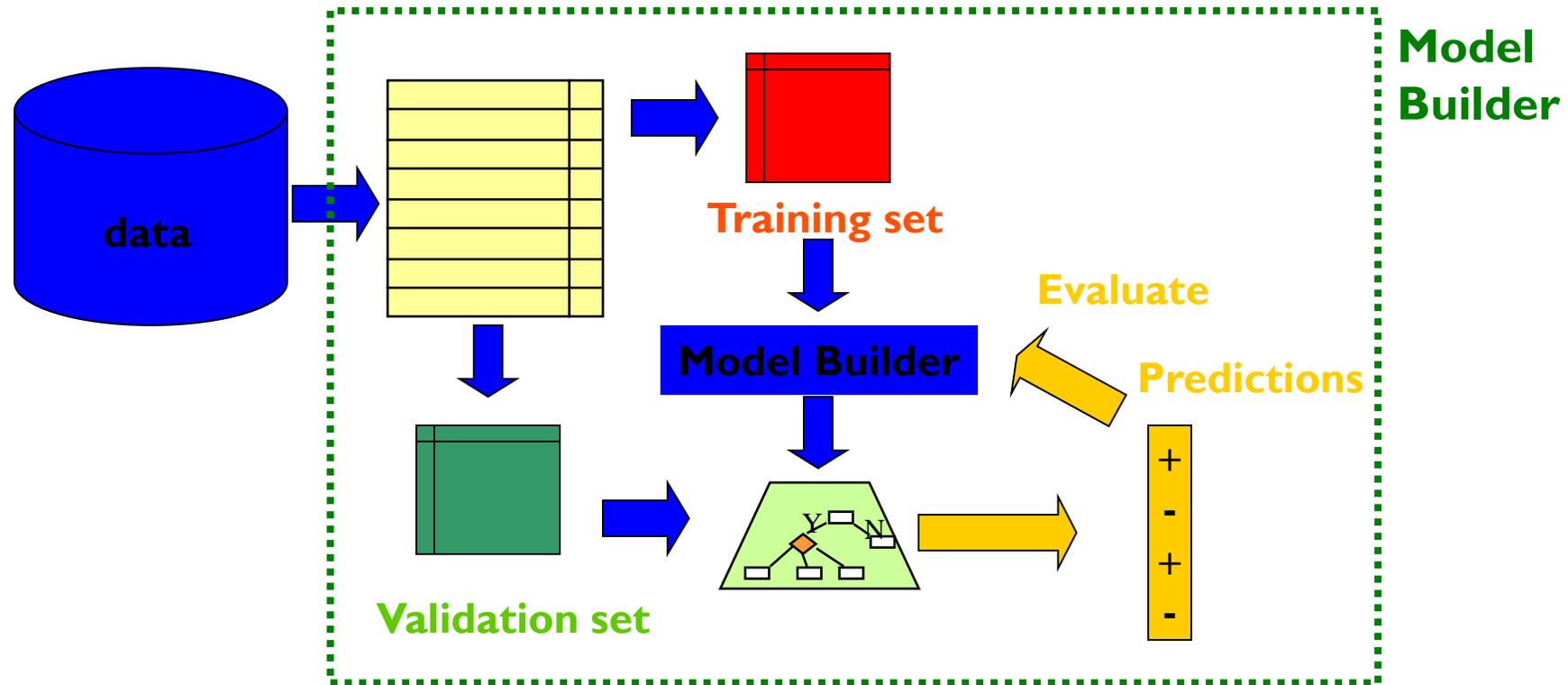
# Fair Evaluation and Hyperparameter Tuning



# Fair Evaluation and Hyperparameter Tuning

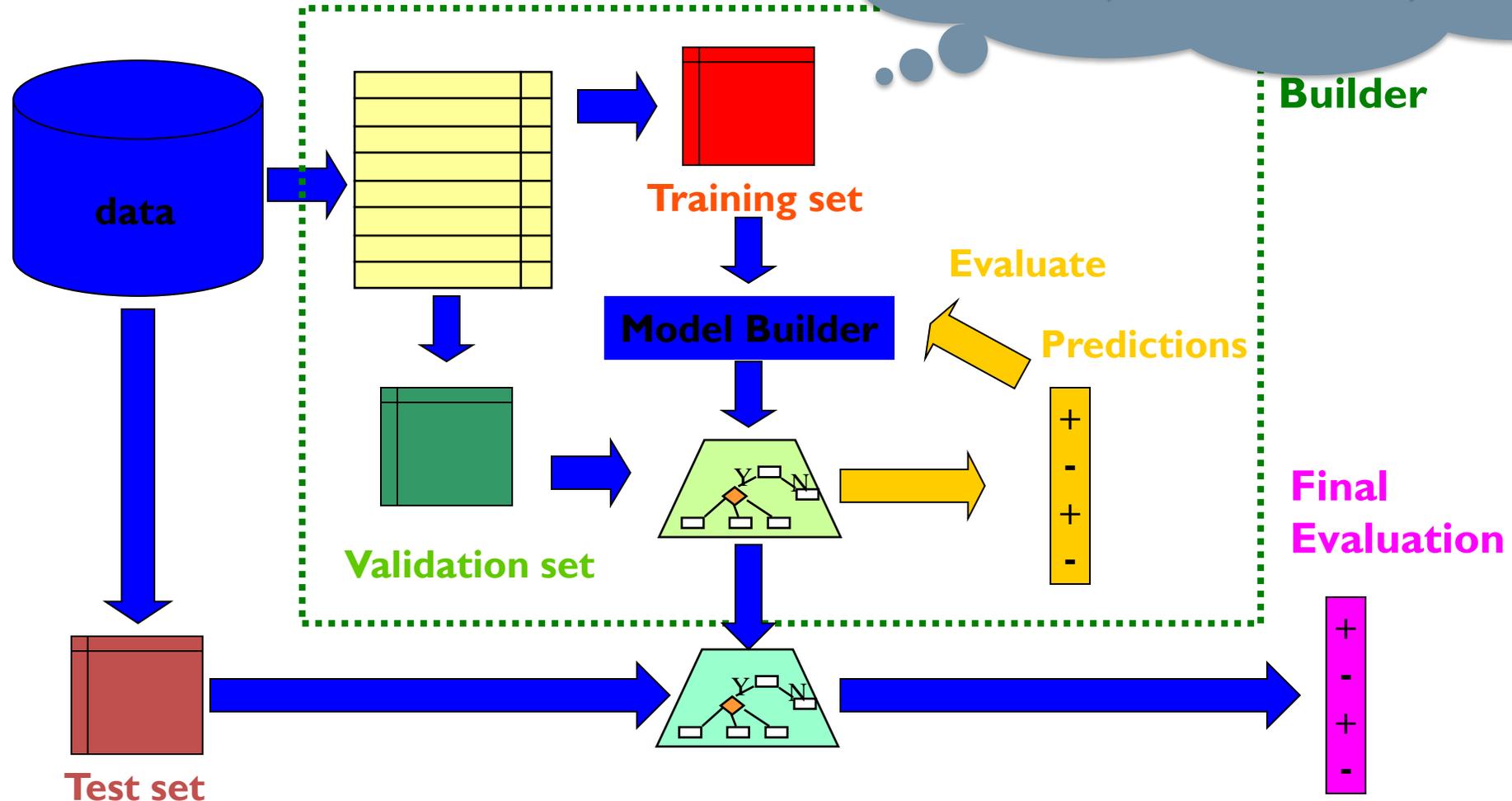


# Fair Evaluation and Hyperparameter Tuning



# Fair Evaluation and Hyperparameter Tuning

*You can repeat this according to a Cross-Validation schema (or leave one out).*



# Evaluation of Classification Algorithms

When evaluating a classifier we should look at the Confusion Matrix

	PREDICTED CLASS		
	Yes	No	
TRUE CLASS	Yes	a (TP)	b (FN)
	No	c (FP)	d (TN)

a: TP (true positive)

c: FP (false positive)

b: FN (false negative)

d: TN (true negative)

# Evaluation of Classification Algorithms

When evaluating a classifier we should look at the Confusion Matrix

	PREDICTED CLASS	
	Yes	No
TRUE CLASS	Yes	a (TP)      b (FN)
	No	c (FP)      d (TN)

Most widely used metric is

$$Accuracy = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

# Accuracy is Sometimes Misleading

Consider a 2-class problem

- Number of Class 0 examples = 9990
- Number of Class 1 examples = 10

If model predicts everything to be class 0,

Accuracy is  $9990/10000 = 99.9\%$

Accuracy is misleading because model does not detect any class 1

# Cost Matrix

We can thus resort to the Cost Matrix combined with Confusion Matrix

	PREDICTED CLASS		
	Yes	No	
TRUE CLASS	Yes	$C(TP)$	$C(FN)$
	No	$C(FP)$	$C(TN)$

$C(x)$ : Cost of misclassifying examples of type  $x$

# Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
	C(.)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Model M <sub>1</sub>	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Accuracy = 80%  
Cost = 3910

Model M <sub>2</sub>	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 90%  
Cost = 4255

# Precision and Recall

Alternatives to accuracy, from information retrieval and search engine

## Precision

- Focuses on the percentage of examples that have been classified as positive examples and are actually positive
- In the information retrieval context represents the percentage of actually good documents that have been shown as a result.

## Recall

- Focuses on the percentage of positively classified examples with respect to the number of existing good documents
- In the information retrieval context, recall represents the percentage of good documents shown with respect to the existing ones.

# Precision and Recall

$$\textit{Precision}(p) = \frac{TP}{TP + FP} = \frac{a}{a + c}$$

$$\textit{Recall}(r) = \frac{TP}{TP + FN} = \frac{a}{a + b}$$

$$\textit{F1 - measure} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

Precision is biased towards C(TP) & C(FP)

Recall is biased towards C(TP) & C(FN)

F1-measure is biased towards all except C(TN), it is high when both precision and recall are reasonably high

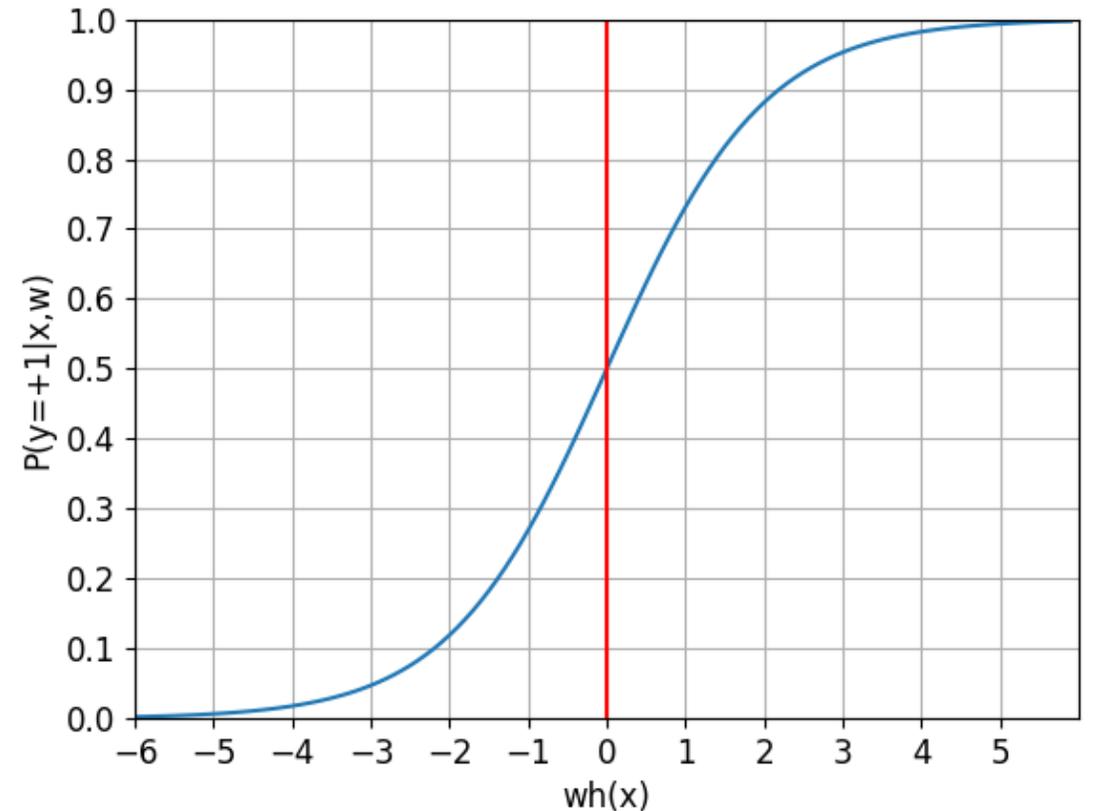
# Probabilistic Classifiers

Logistic regression returns a probability  $P(y_i|\vec{x}_i)$

Given an example  $x_i$  its predicted class is the label with the largest probability which is equivalent to using a threshold of 0.5 to decide which class to assign.

We can use a different threshold and label as positive examples when

$$P(+1|x) > 0.75$$



# Influencing Precision and Recall

Suppose we use a near one threshold to classify positive examples, then, we will classify as positives only examples for which we are very confident

- Precision will be high since we are not likely produce false positives
- Recall will be low since we are likely to produce more false negatives

Suppose we use a near zero threshold to classify positive examples, then, we will classify everything as positives (this is an optimistic classifier)

- Precision will be low; we are going to generate maximum number of false positives
- Recall will be high since by classifying everything as positive we are going to generate the maximum number of false negatives

# Precision Recall Curves

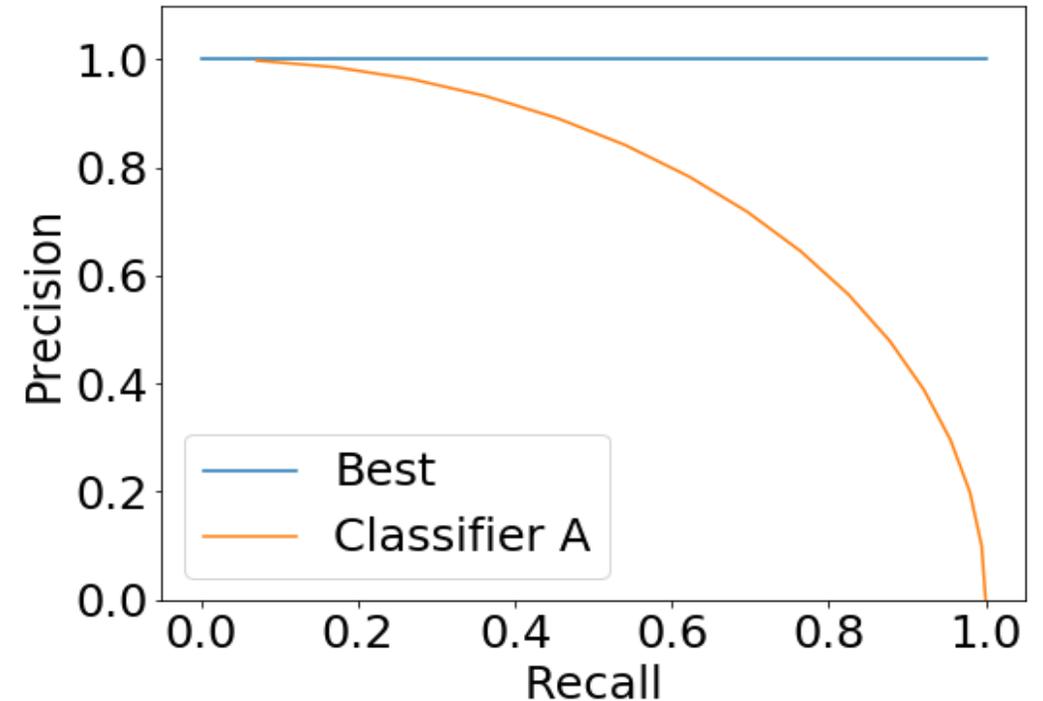
Plot precision as a function of recall for varying threshold values

The best classifier would be the one that has always a precision equal to one (but never happens)

More in general classifiers will show different shapes

How to decide among more classifiers?

- Use the area under the curve (the nearer to one, the better)
- Use F1 measure

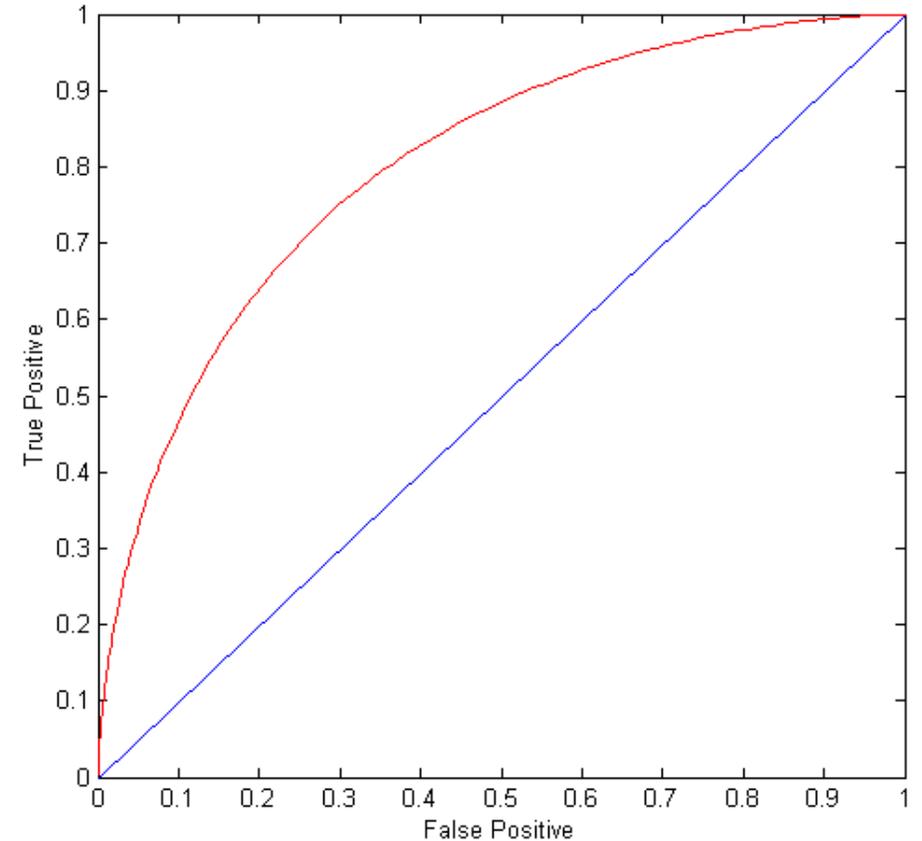


# Receiver Operating Characteristics (ROC)

Plot the True Positive Rate ( $TPR=TP/(TP+FN)$ ) against the False Positive Rate ( $FPR=FP/(TN+FP)$ )

Performance of each classifier represented as a point on the ROC curve

Changing the classification threshold, sample distribution, or cost matrix changes the location of the point



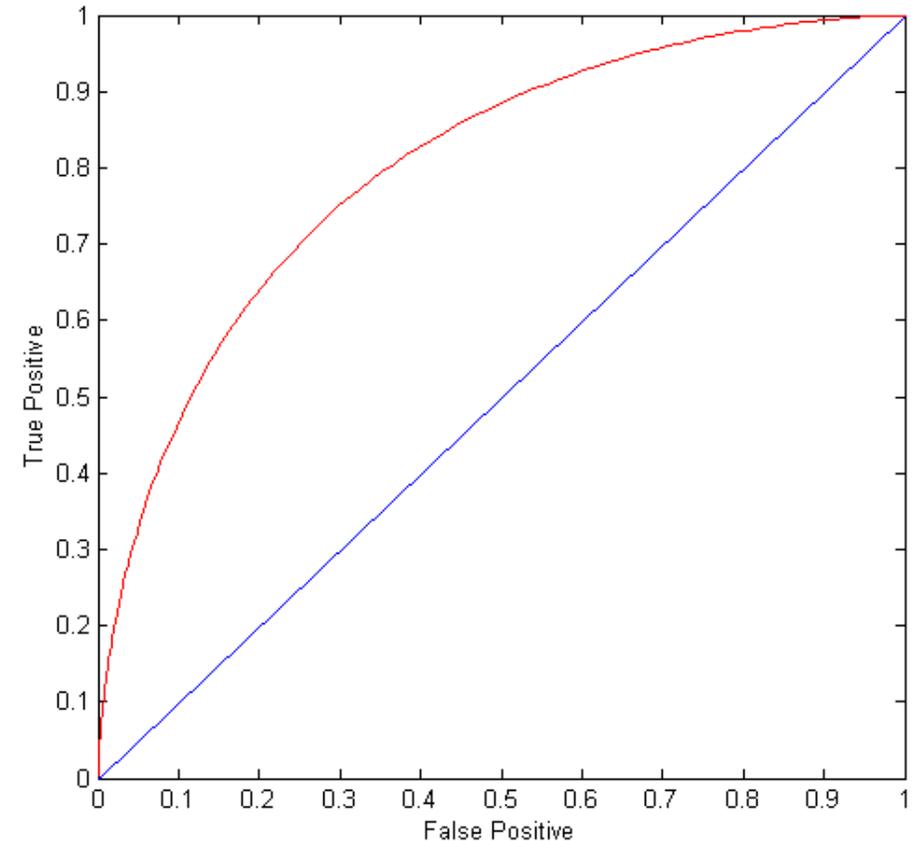
# Receiver Operating Characteristics (ROC)

Plot the True Positive Rate ( $TPR=TP/(TP+FN)$ ) against the False Positive Rate ( $FPR=FP/(TN+FP)$ )

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (0,1): ideal

Diagonal line:

- Random guessing
- Below diagonal line, prediction is opposite of the true class



# Receiver Operating Characteristics (ROC)

Plot the True Positive Rate ( $TPR=TP/(TP+FN)$ ) against the False Positive Rate ( $FPR=FP/(TN+FP)$ )

- No model consistently outperform the other
- M1 is better for small FPR
- M2 is better for large FPR

Area Under the ROC curve

- Ideal, area = 1
- Random guess, area = 0.5

