Big Data and Data Mining

Week 3/4: Classification

Fenerbahce University

Instructors

Assist. Prof. Vecdi Emre Levent

Office: 311

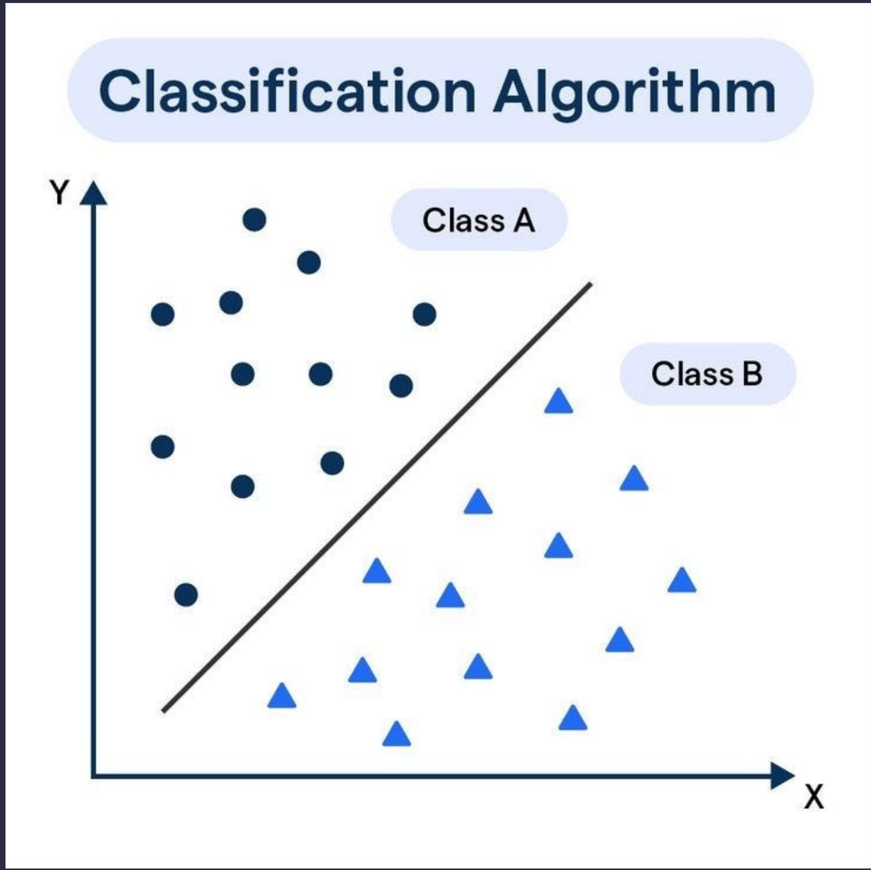
Email : emre.levent@fbu.edu.tr

Classification

- What is classification?
- Issues regarding classification
- Bayesian Classification
- Classification by decision tree induction
- Classification by Neural Networks
- Classification by Support Vector Machines (SVM)
- Instance Based Methods
- Classification accuracy
- Summary

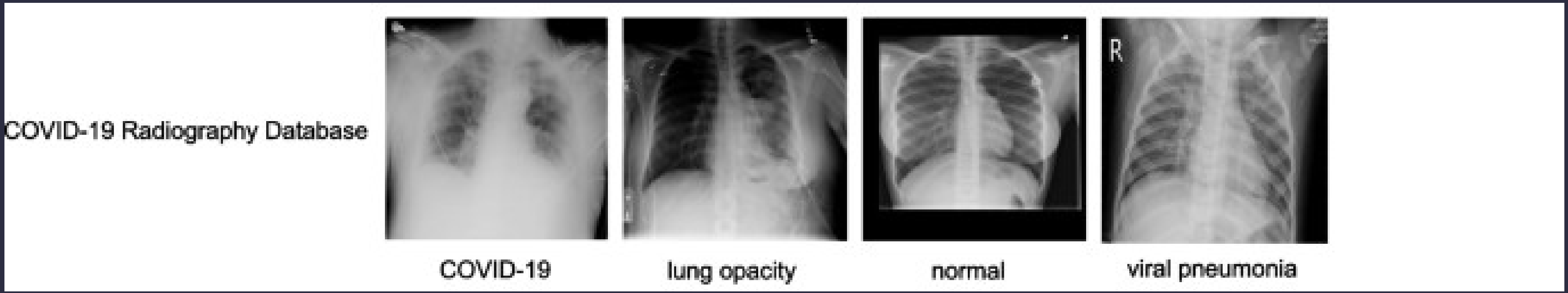
Classification

- Classification:
  - predicts categorical class labels
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data



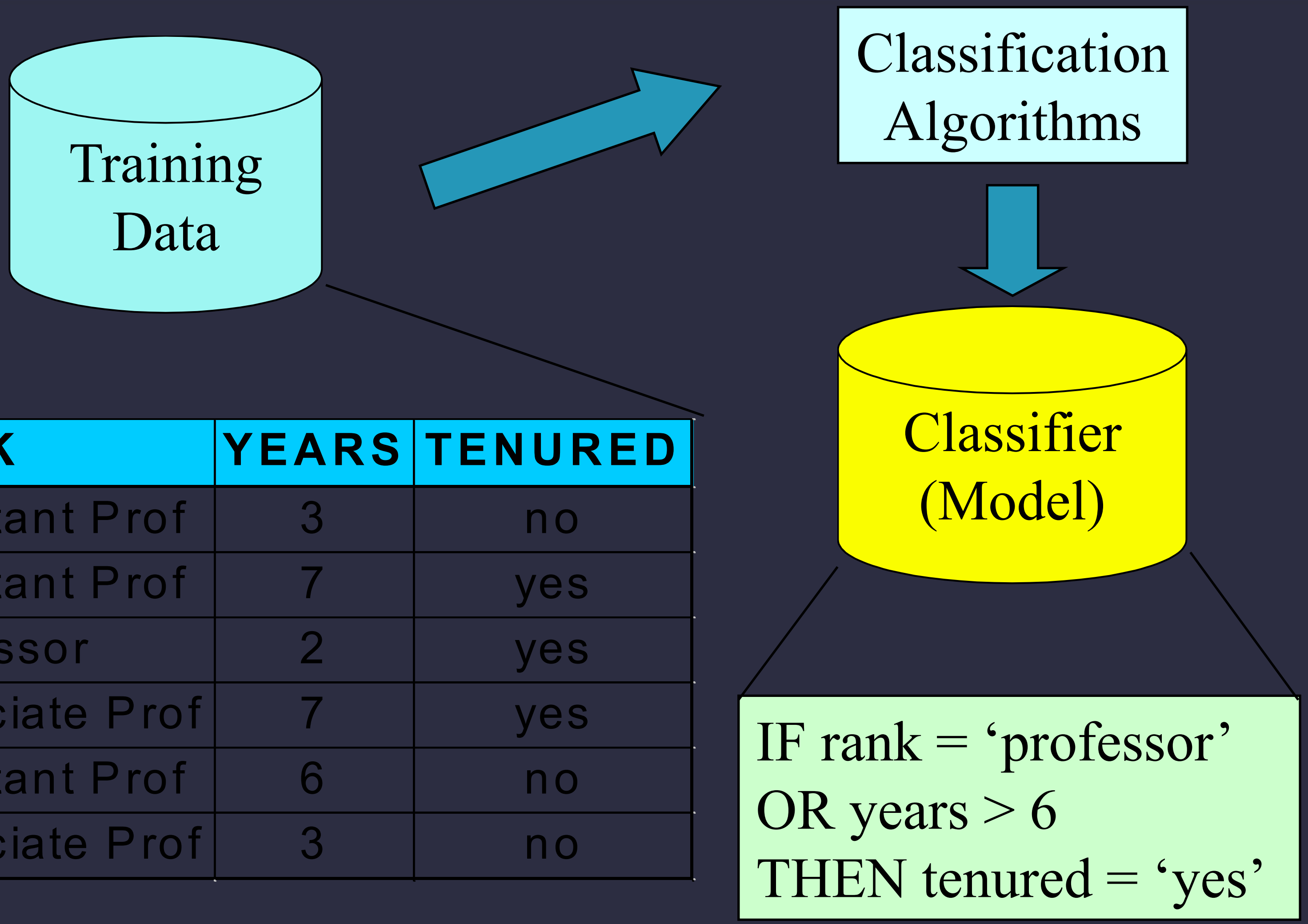
Classification

- Classification:
- Typical Applications
  - credit approval
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis



Classification—A Two-Step Process

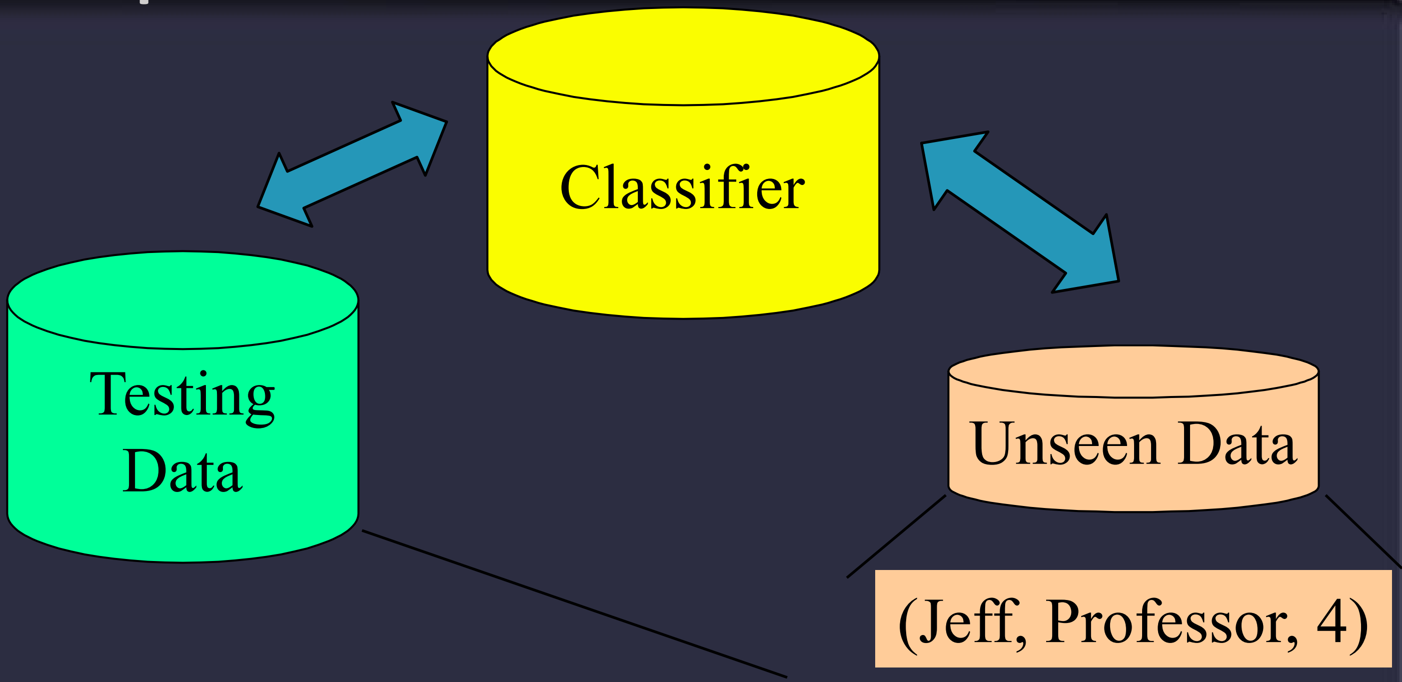
- Model construction: describing a set of predetermined classes
  - Each sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of sample used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formula or AI



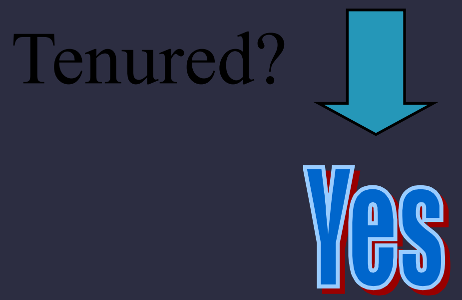
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classification—A Two-Step Process

- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known



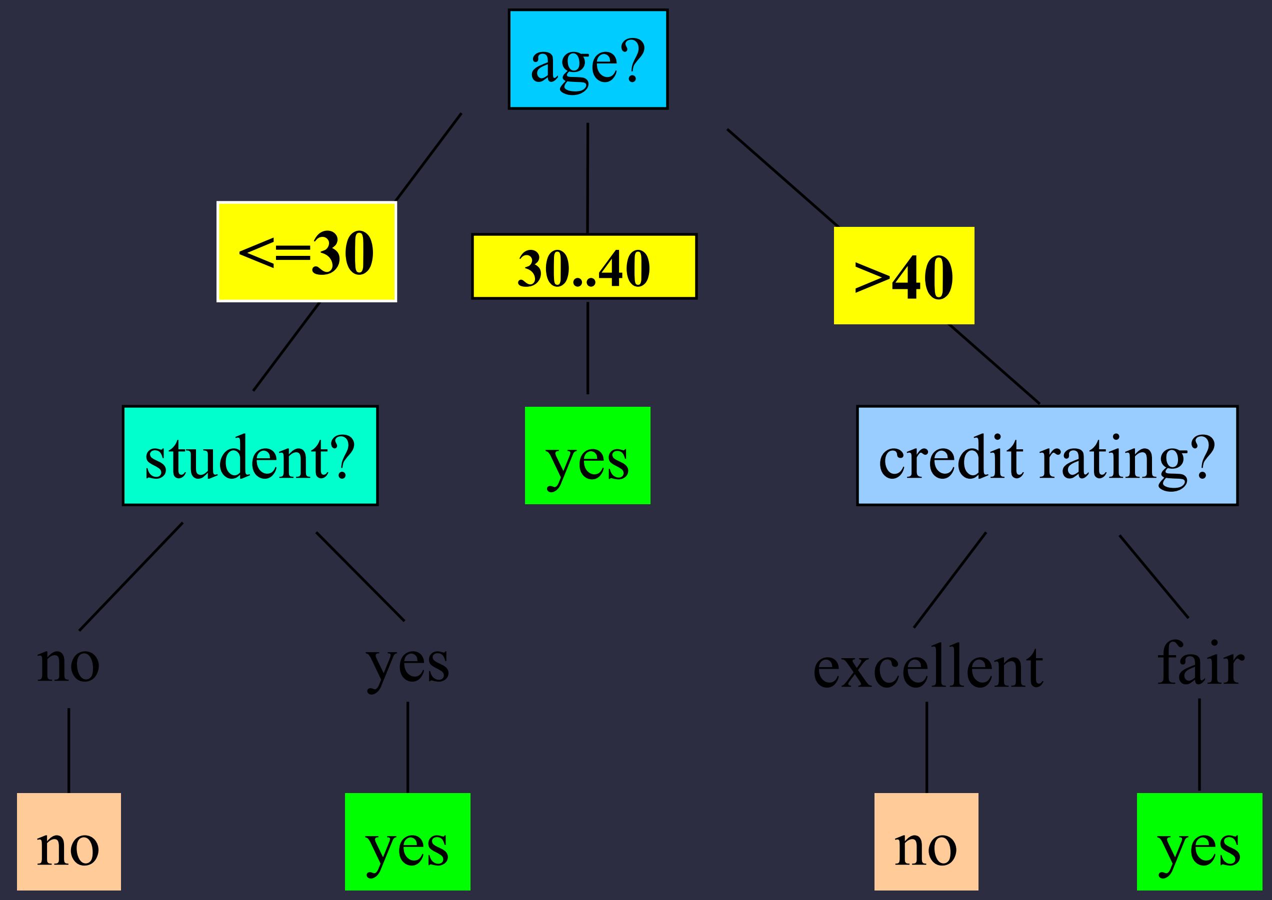
NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes



Dataset for computer buyers

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	
31…40	high	yes	fair	
>40	medium	no	excellent	

A Decision Tree for *"buys_computer"*

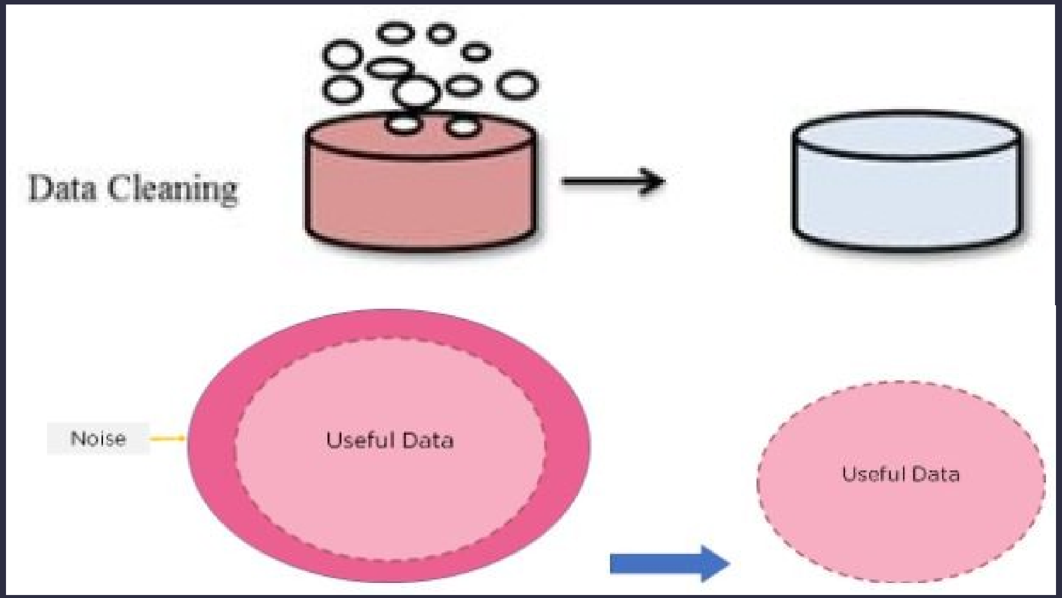


Classification

- What is classification
- Issues regarding classification
- Bayesian Classification
- Classification by decision tree induction
- Classification by Neural Networks
- Classification by Support Vector Machines (SVM)
- Instance Based Methods
- Classification accuracy
- Summary

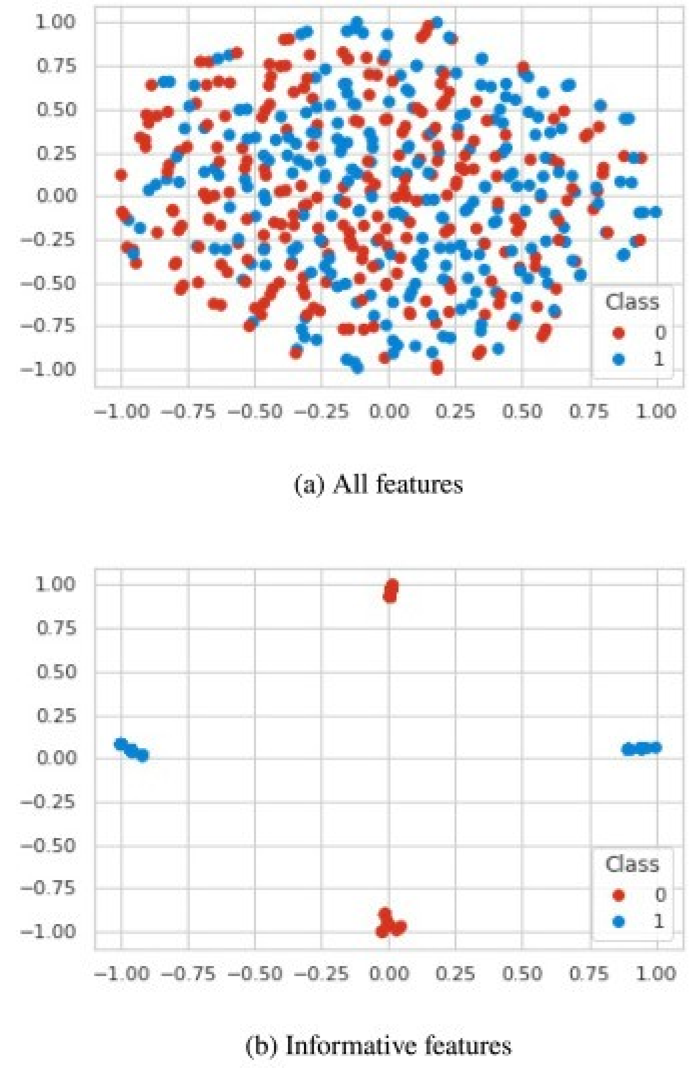
Issues (1): Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values



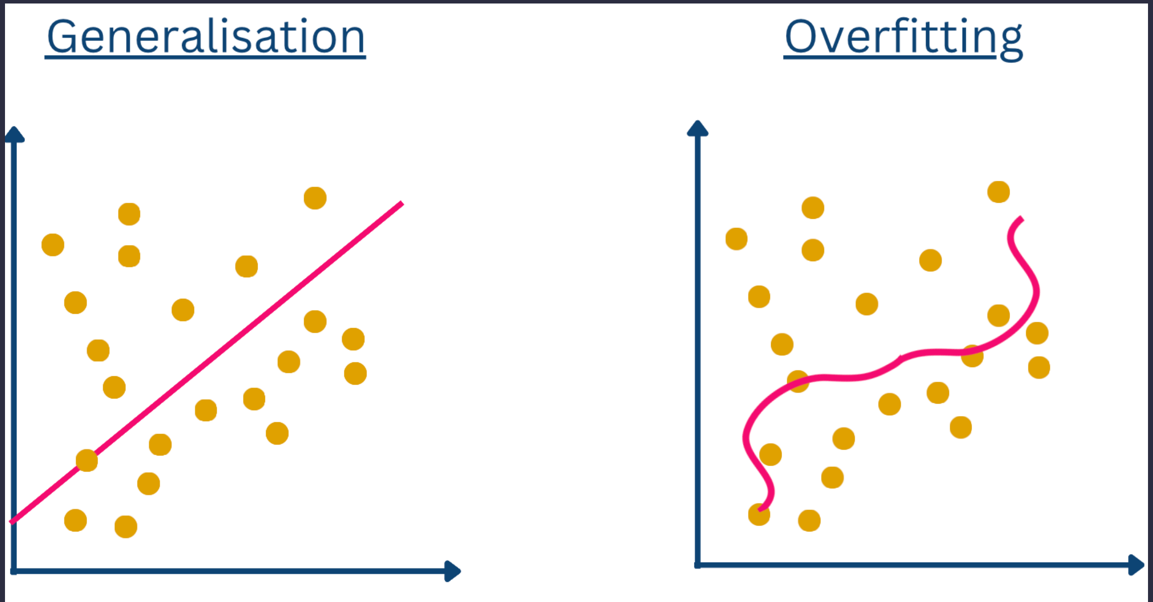
Issues (1): Data Preparation

- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes



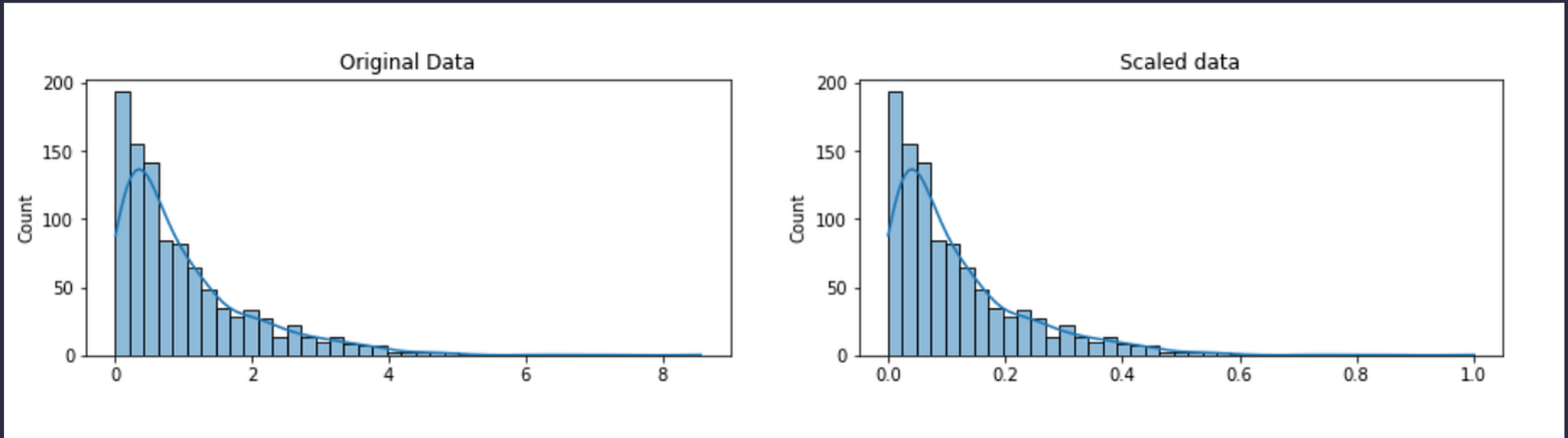
Issues (1): Data Preparation

- Data transformation
  - Generalize and/or normalize data



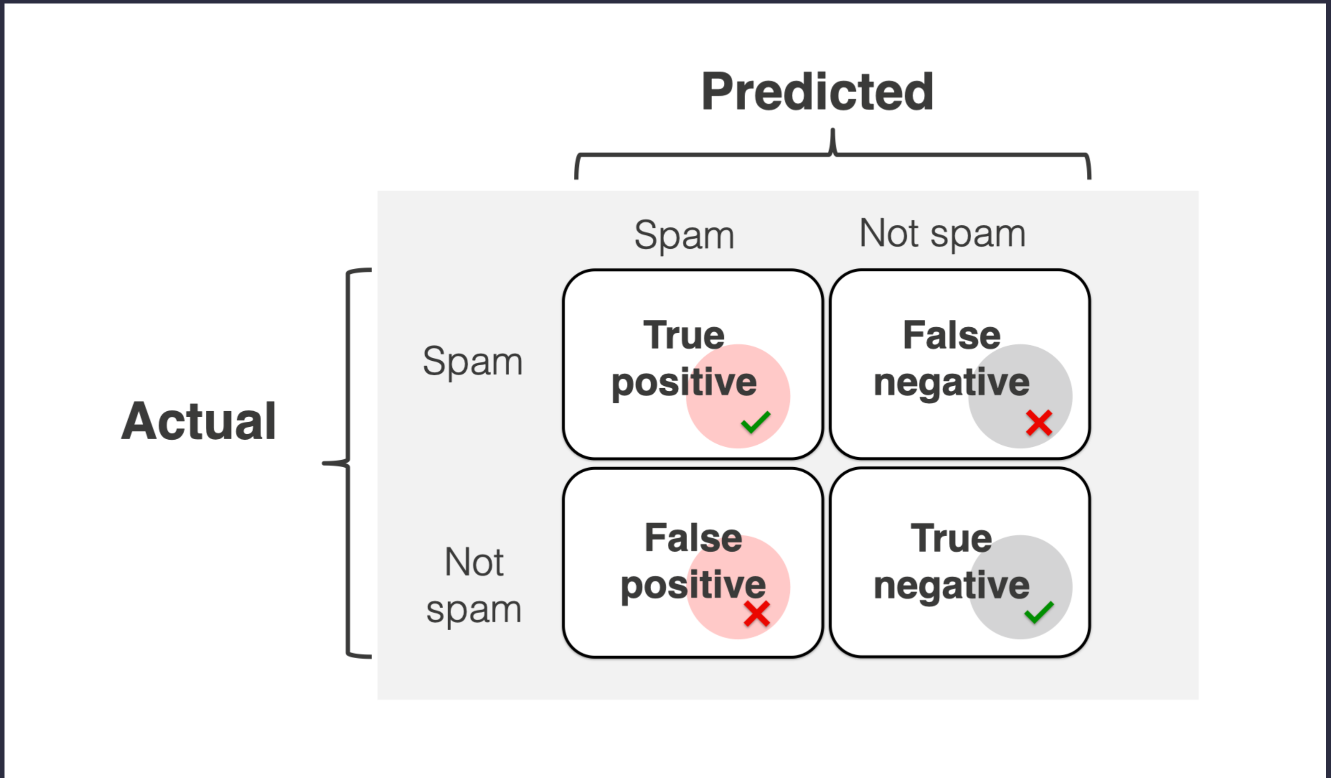
Issues (1): Data Preparation

- Data transformation
  - Generalize and/or normalize data



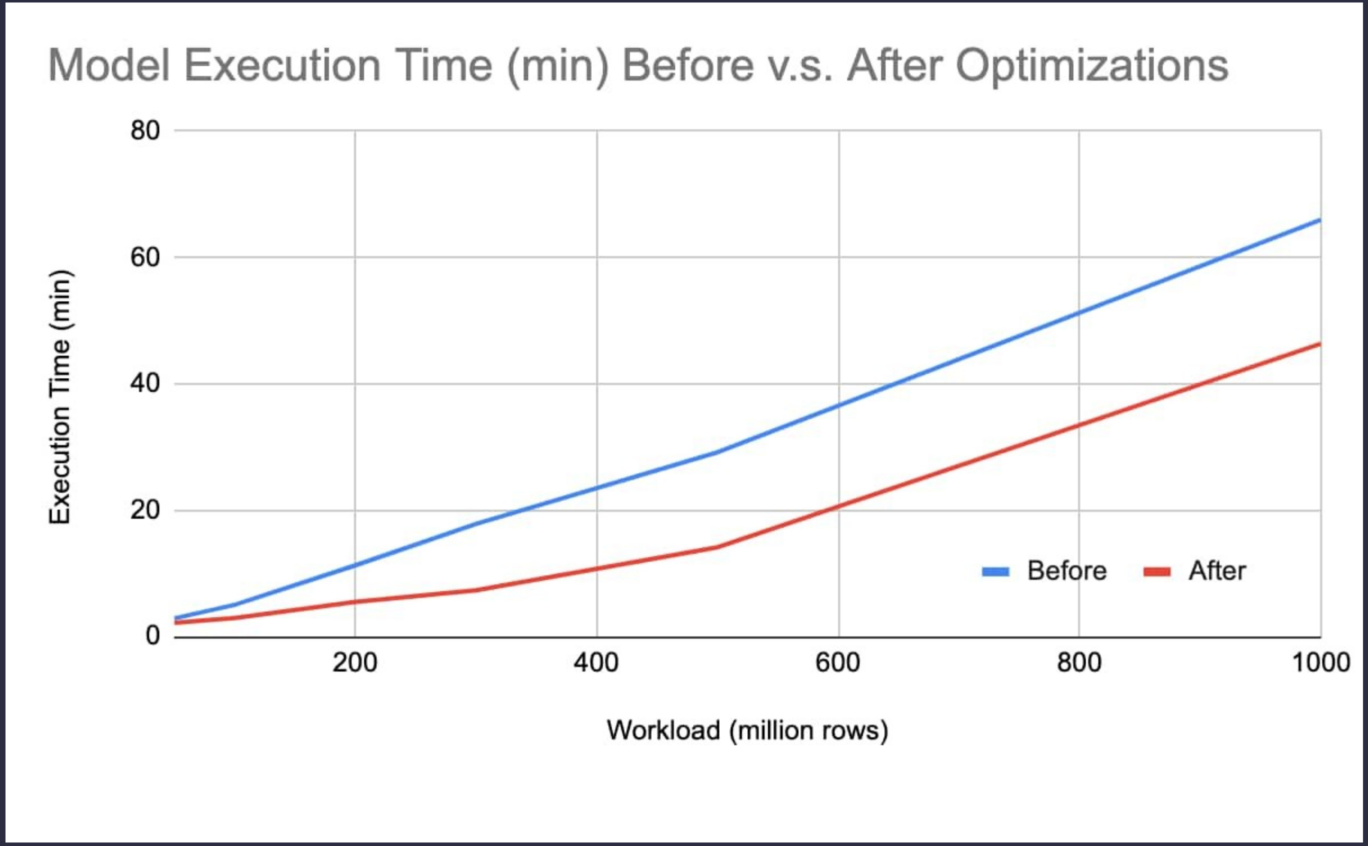
Issues (2): Evaluating Classification Methods

- Predictive accuracy



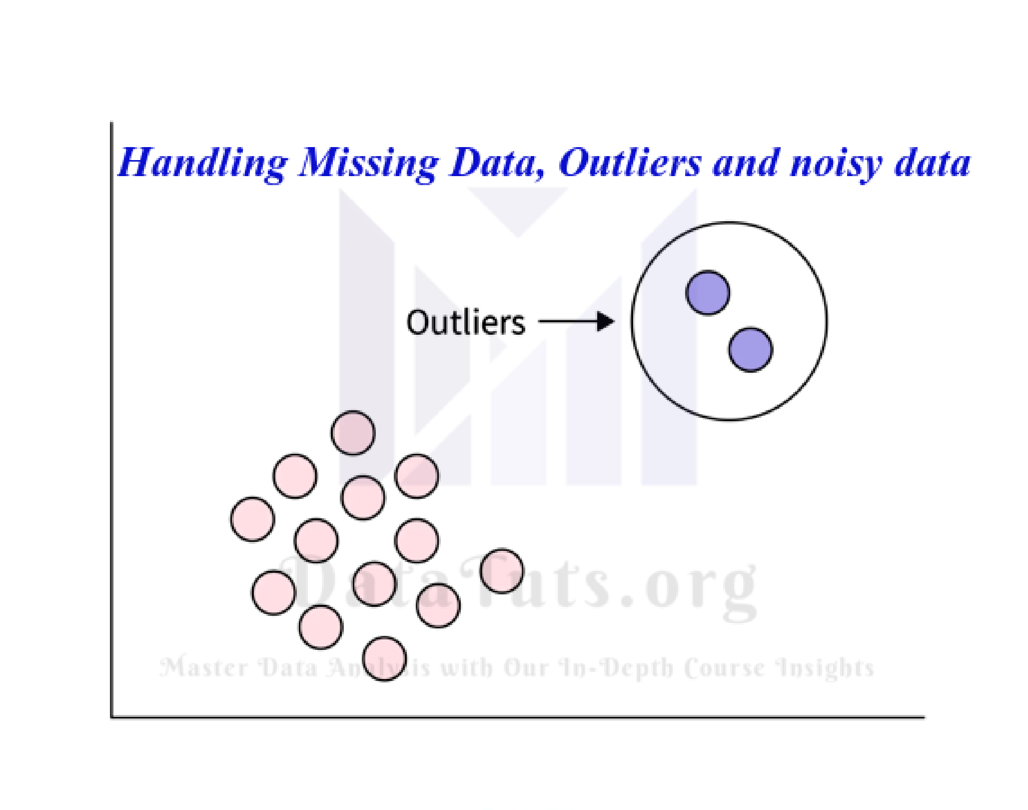
Issues (2): Evaluating Classification Methods

- Speed and scalability
  - time to construct the model
  - time to use the model



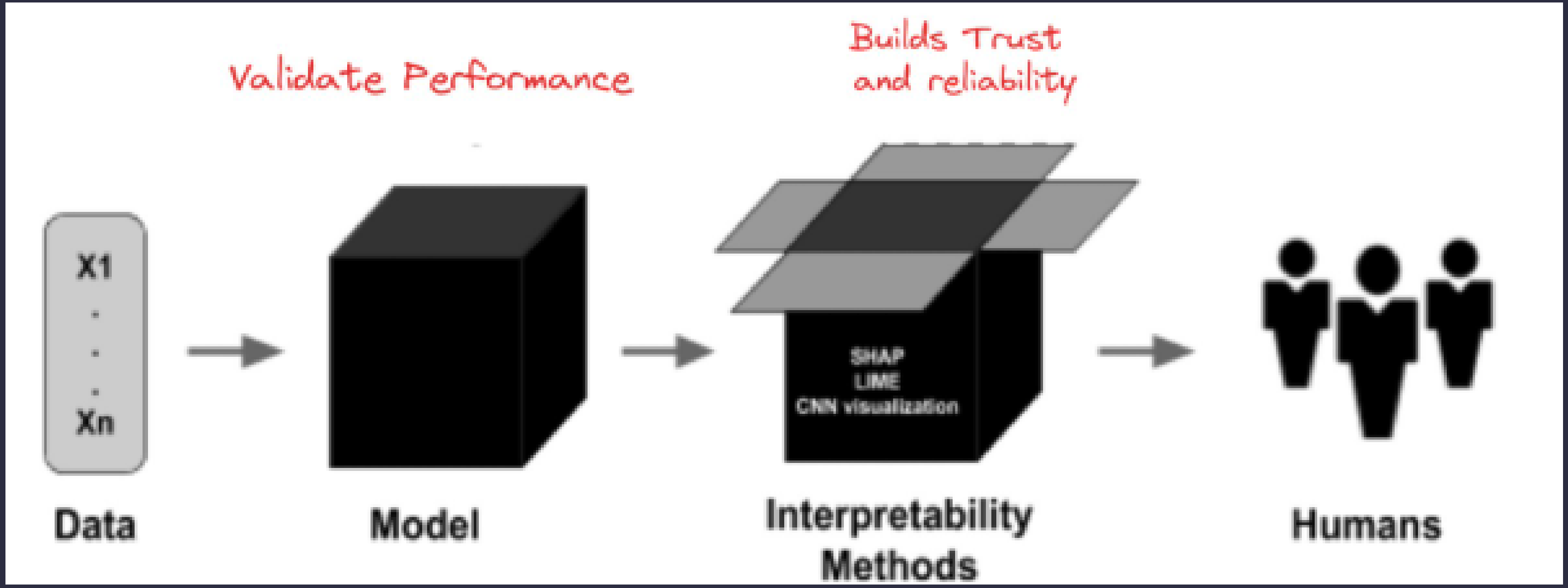
Issues (2): Evaluating Classification Methods

- Robustness
  - handling noise and missing values



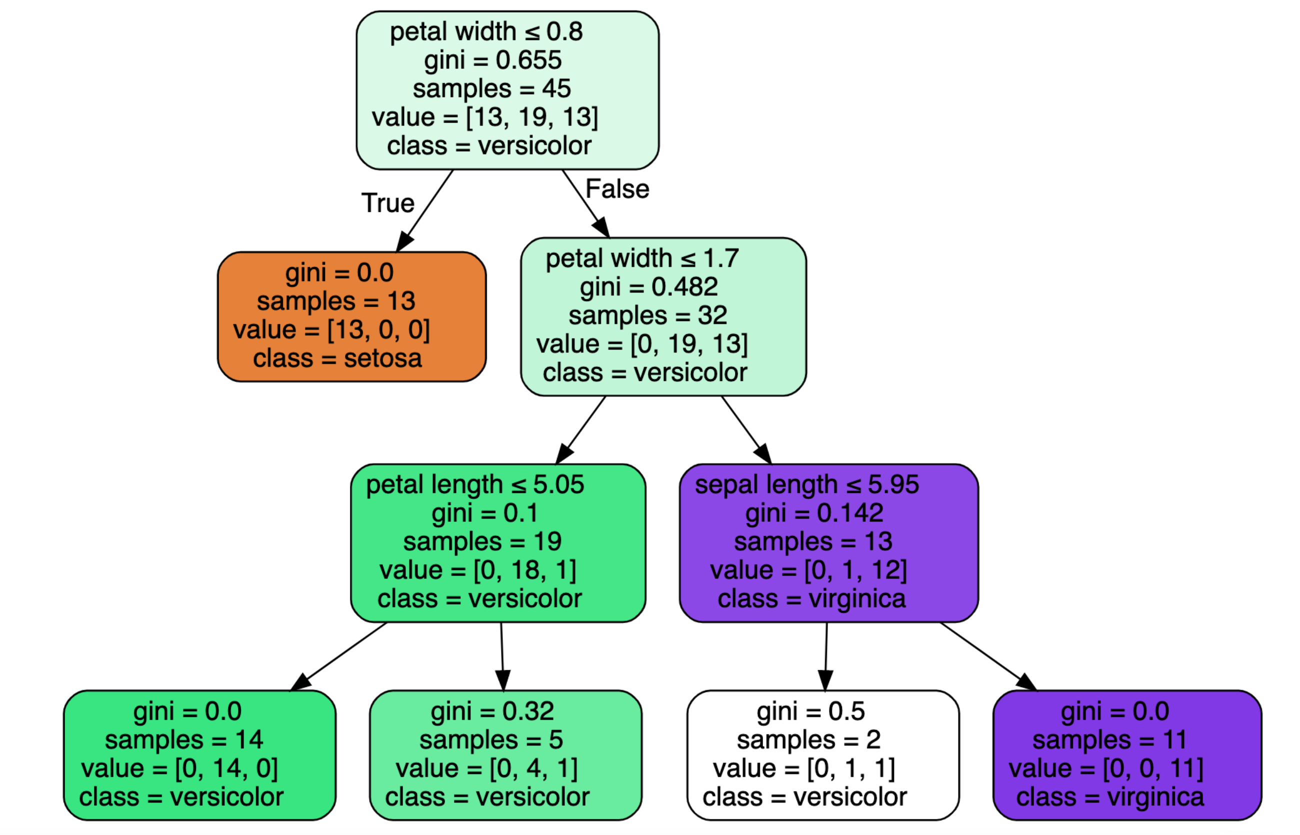
Issues (2): Evaluating Classification Methods

- Interpretability:
  - understanding and insight provided by the model



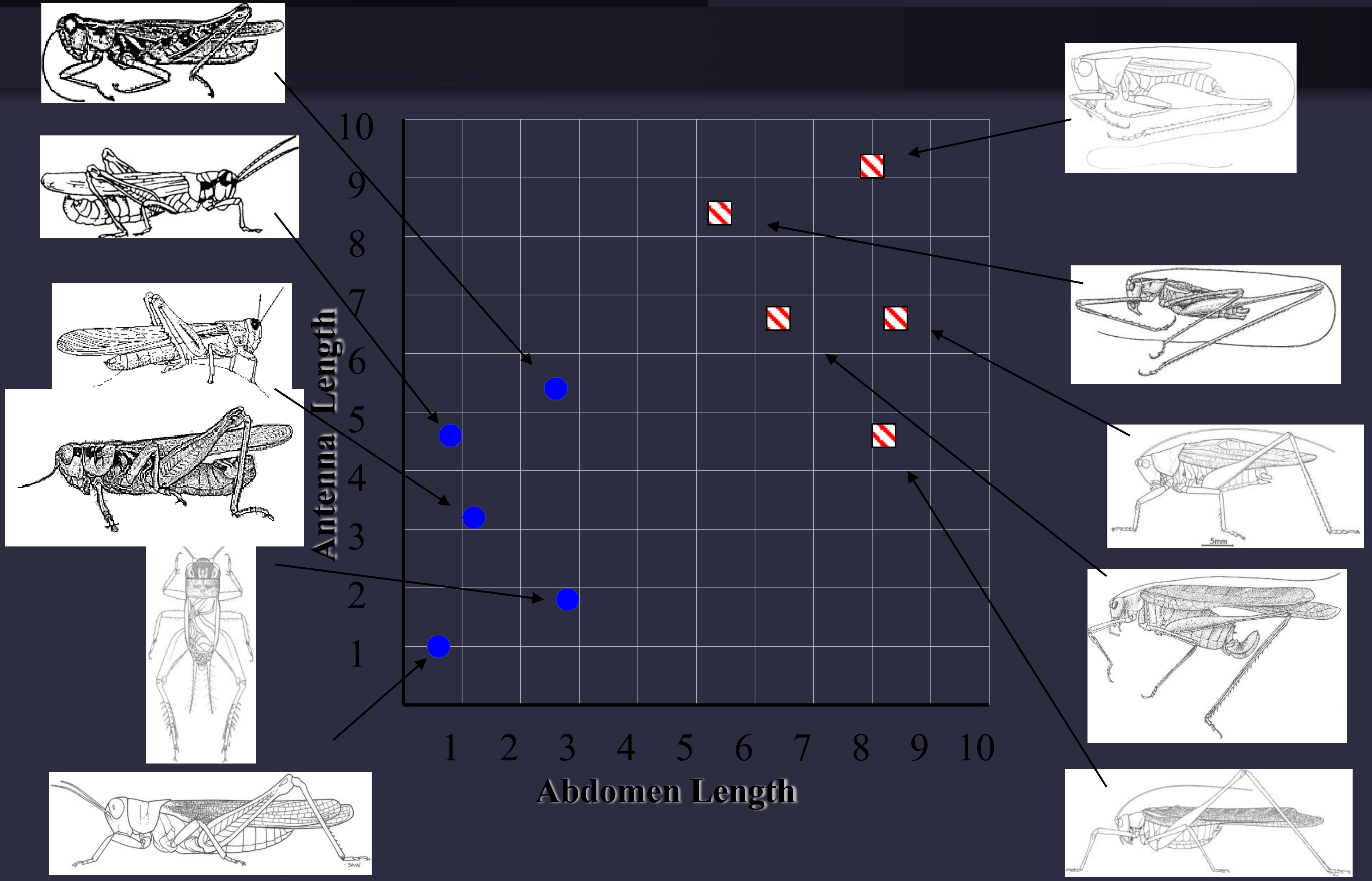
Issues (2): Evaluating Classification Methods

- Goodness of rules
  - decision tree size
  - compactness of classification rules

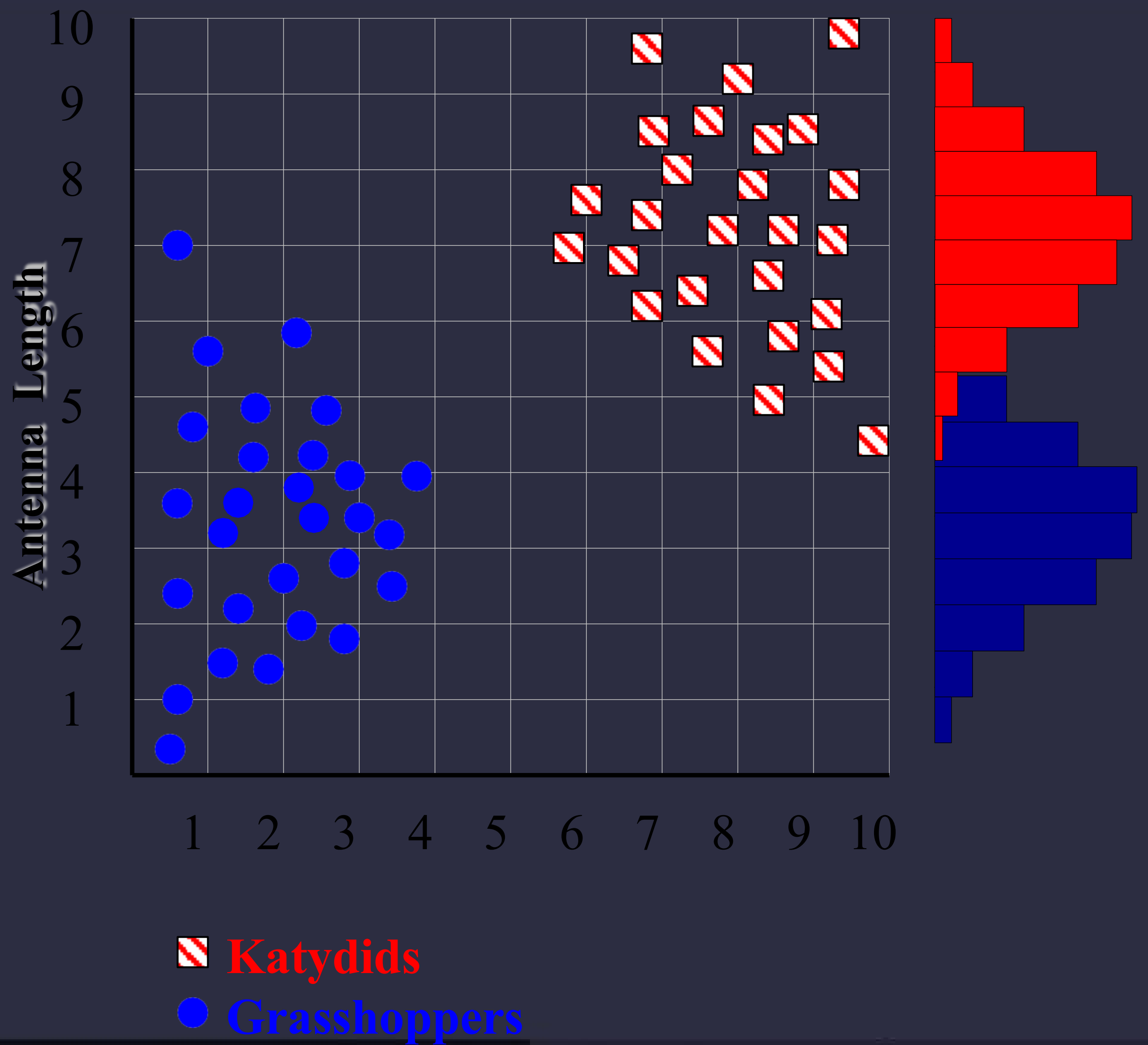


Grasshoppers

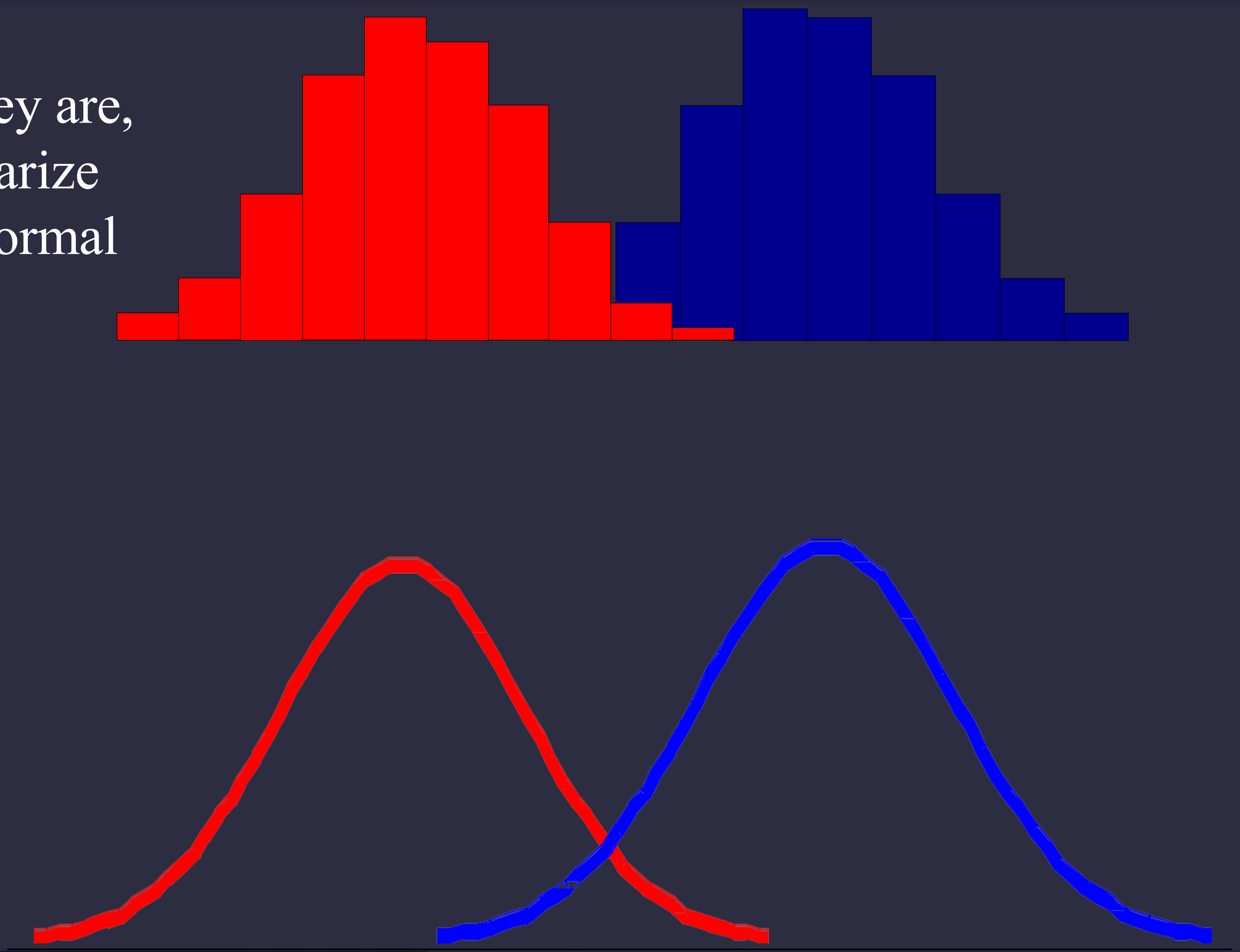
Katydids



With a lot of data, we can build a histogram. Let us just build one for "Antenna Length" for now…



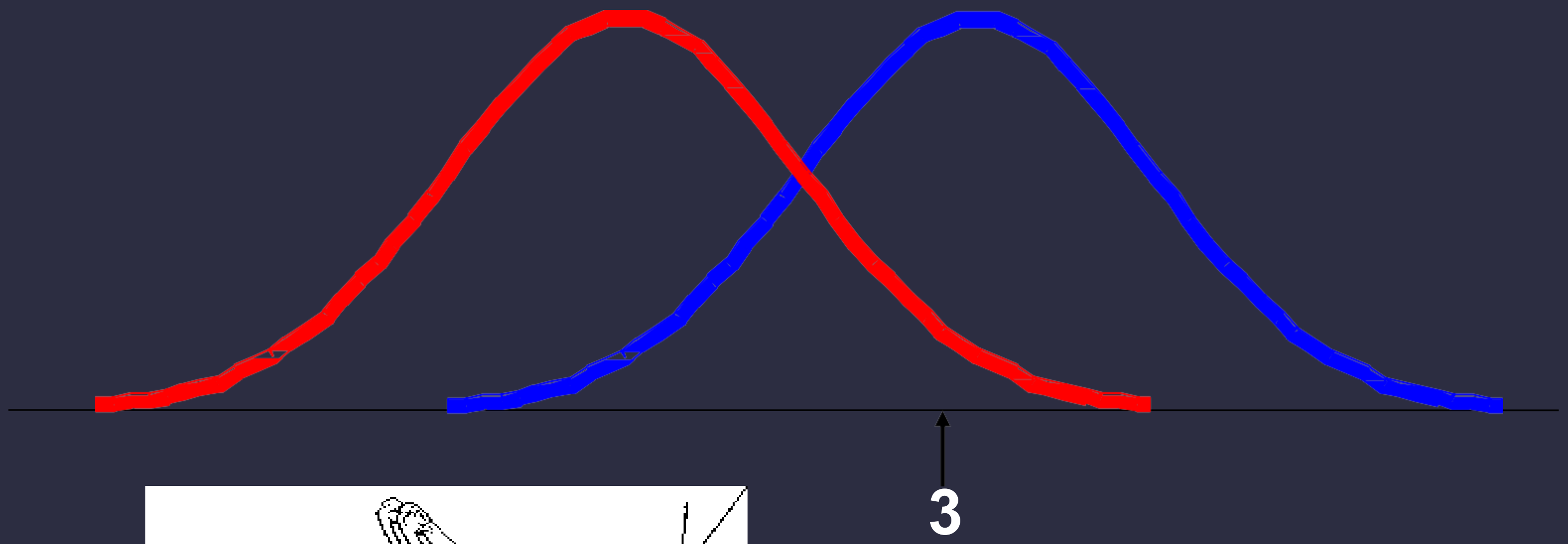
We can leave the histograms as they are, or we can summarize them with two normal distributions.

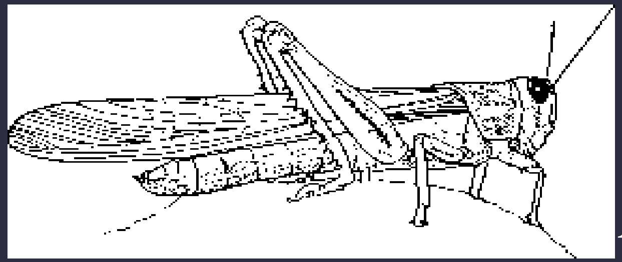


• We want to classify an insect we have found. Its antennae are 3 units long. How can we classify it?

• We can just ask ourselves, give the distributions of antennae lengths we have seen, is it more *probable* that our insect is a **Grasshopper** or a **Katydid**.
• There is a formal way to discuss the most *probable* classification…

$p(c_j | d)$ = probability of class c_j, *given* that we have observed d



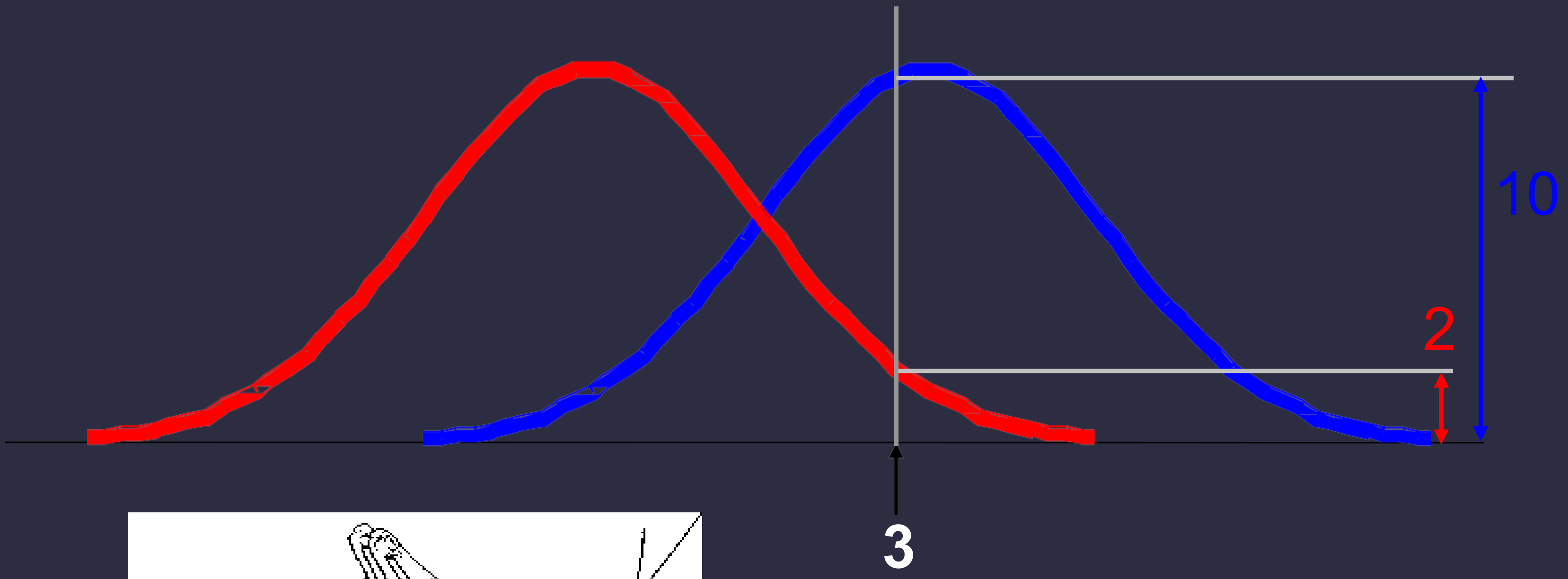


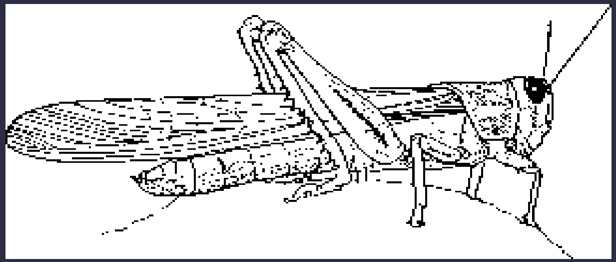
Antennae length is **3**

$p(c_j \mid d)$ = probability of class c_j, given that we have observed d

P(**Grasshopper** | **3**) = 10 / (10 + 2) = 0.833

P(**Katydid** | **3**) = 2 / (10 + 2) = 0.166



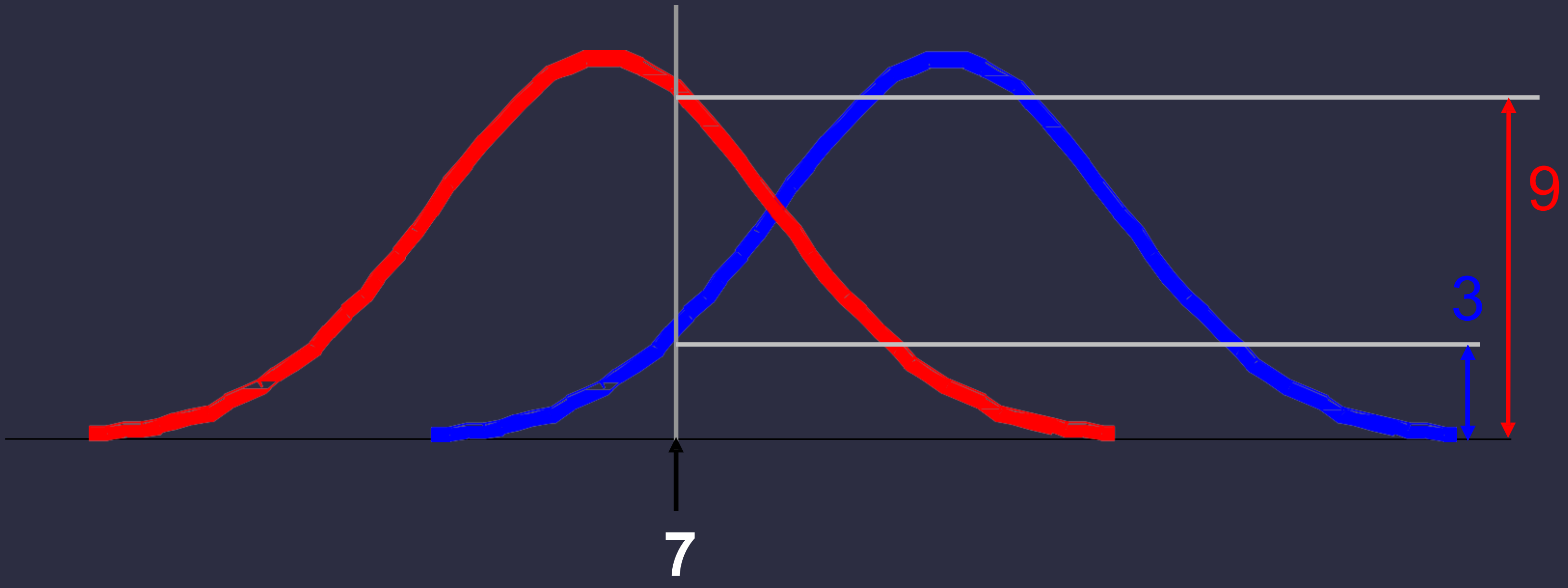


Antennae length is 3

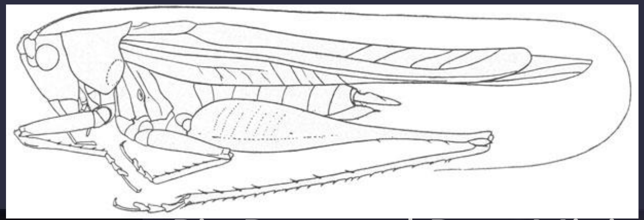
$p(c_j \mid d)$ = probability of class c_j, given that we have observed d

P(**Grasshopper** | **7**) = 3 / (3 + 9) = 0.250

P(**Katydid** | **7**) = 9 / (3 + 9) = 0.750



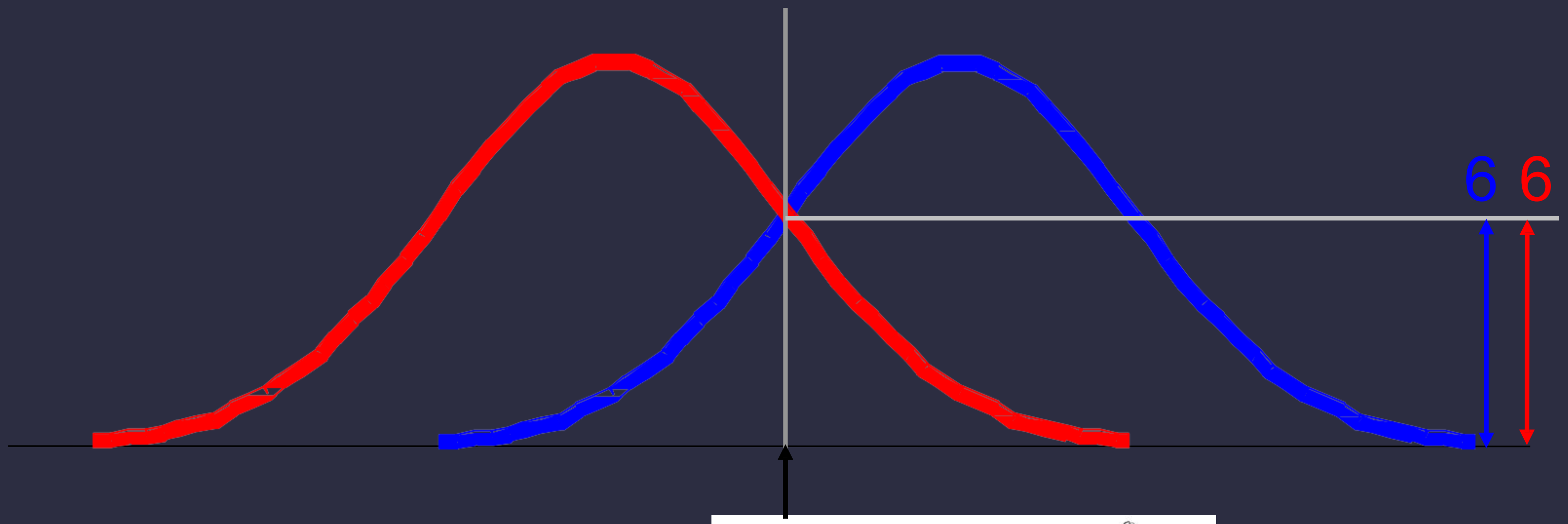
Antennae length is 7

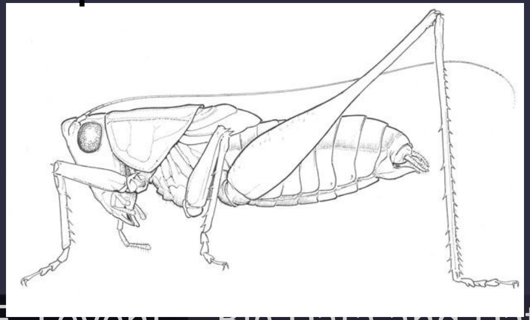


$p(c_j \mid d)$ = probability of class c_j, given that we have observed d

P(**Grasshopper** | **5**) = 6 / (6 + 6) = 0.500

P(**Katydid** | **5**) = 6 / (6 + 6) = 0.500





Antennae length is **5**

Bayes Classifiers

That was a visual intuition for a simple case of the Bayes classifier, also called:

- Idiot Bayes
- Naïve Bayes
- Simple Bayes

Find out the probability of the previously unseen instance belonging to each class, then simply pick the most probable class.

Bayes Classifiers

Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

- $p(c_j \mid d)$ = probability of instance d being in class c_j, This is what we are trying to compute
- $p(d \mid c_j)$ = probability of generating instance d given class c_j,
  We can imagine that being in class c_j, causes you to have feature d with some probability
- $p(c_j)$ = probability of occurrence of class c_j,
  This is just how frequent the class c_j, is in our database
- $p(d)$ = probability of instance d occurring

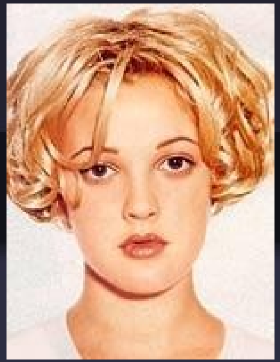
Assume that we have two classes

$c_1 = \text{male}$, and $c_2 = \text{female}$.

We have a person whose sex we do not know, say "*drew*" or *d*.

Classifying *drew* as male or female is equivalent to asking is it more probable that *drew* is male or female, I.e which is greater $p(\text{male} \mid drew)$ or $p(\text{female} \mid drew)$

(Note: "Drew can be a male or female name")



Drew Barrymore

Drew Carey

What is the probability of being called "*drew*" given that you are a male?

What is the probability of being a male?

$$p(\text{male} \mid drew) = \frac{p(drew \mid \text{male})\, p(\text{male})}{p(drew)}$$

What is the probability of being named "*drew*"? (actually irrelevant, since it is that same for all classes)

Is Officer Drew a Male or Female?

Officer Drew

Luckily, we have a small database with names and sex.

We can use it to apply Bayes rule…

Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

Officer Drew

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

$$p(\text{male} \mid drew) = \frac{p(drew \mid \text{male})\, p(\text{male})}{p(drew)}$$

Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

$$p(\text{male} \mid drew) = \frac{1/3 * 3/8}{3/8} = \frac{0.125}{3/8}$$

$$p(\text{female} \mid drew) = \frac{2/5 * 5/8}{3/8} = \frac{0.250}{3/8}$$

Officer Drew is more likely to be a **Female**.

So far we have only considered Bayes Classification when we have one attribute (the "*antennae length*", or the "*name*").
But we may have many features.
How do we use all the features?

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

Name	Over 170CM	Eye	Hair length	Sex
Drew	No	Blue	Short	Male
Claudia	Yes	Brown	Long	Female
Drew	No	Blue	Long	Female
Drew	No	Blue	Long	Female
Alberto	Yes	Brown	Short	Male
Karin	No	Blue	Long	Female
Nina	Yes	Brown	Short	Female
Sergio	Yes	Blue	Long	Male

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \ldots.* p(d_n|c_j)$$

The probability of class c_j generating instance d, equals….

The probability of class c_j generating the observed value for feature 1, multiplied by..

The probability of class c_j generating the observed value for feature 2, multiplied by..

$$p(\text{male} \mid drew) = \frac{p(drew \mid \text{male})\, p(\text{male})}{p(drew)}$$

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \ldots * p(d_n|c_j)$$

$$p(\text{officer drew}|c_j) = p(\text{over\_170}_{\text{cm}} = \text{yes}|c_j) * p(\text{eye} = blue|c_j) * \ldots$$

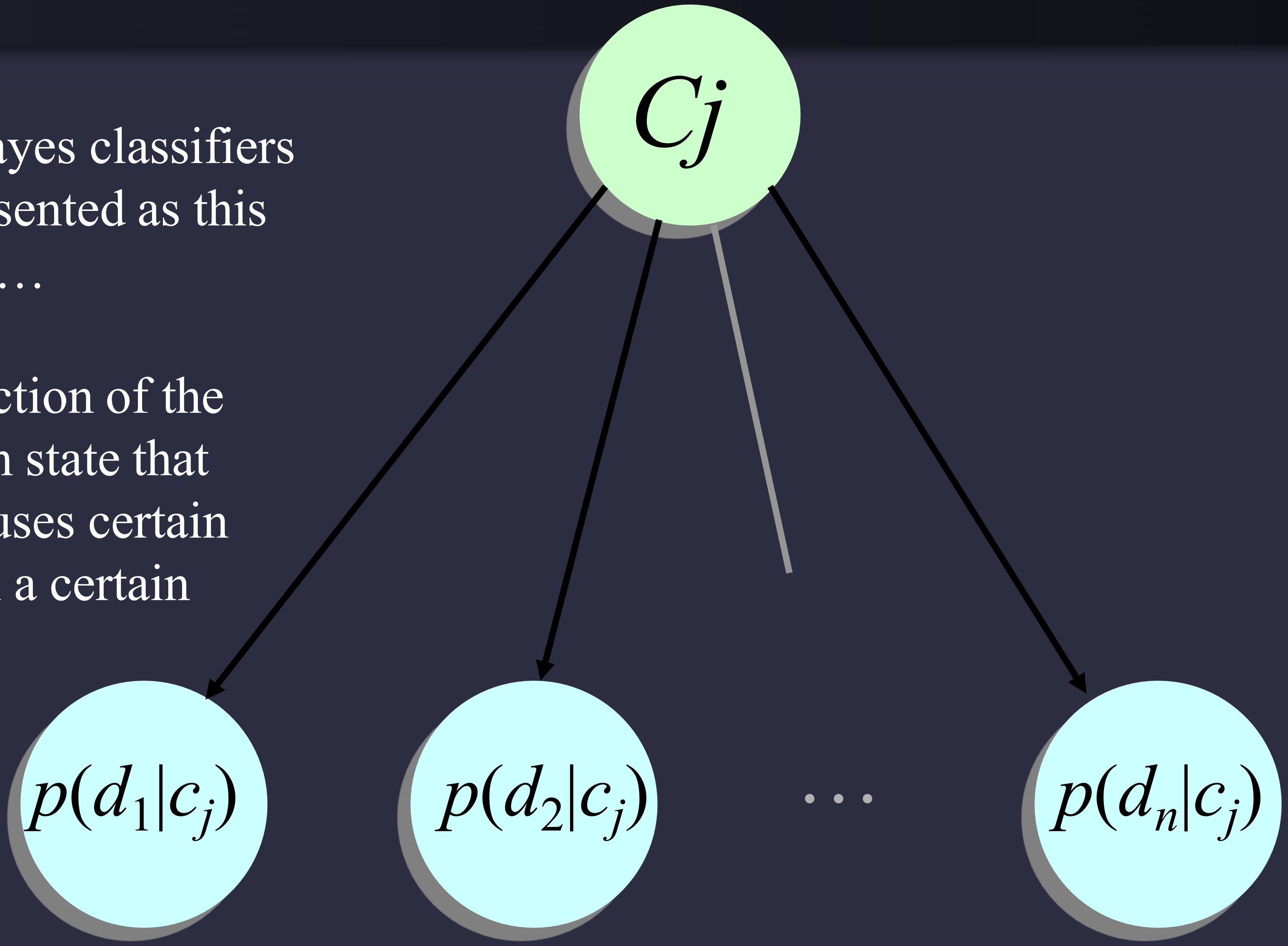
Officer Drew is blue-eyed, over 170_{cm} tall, and has long hair

$$p(\text{officer drew}|\ \text{Female}) = 2/5 \quad * \quad 3/5 \quad * \quad \ldots$$

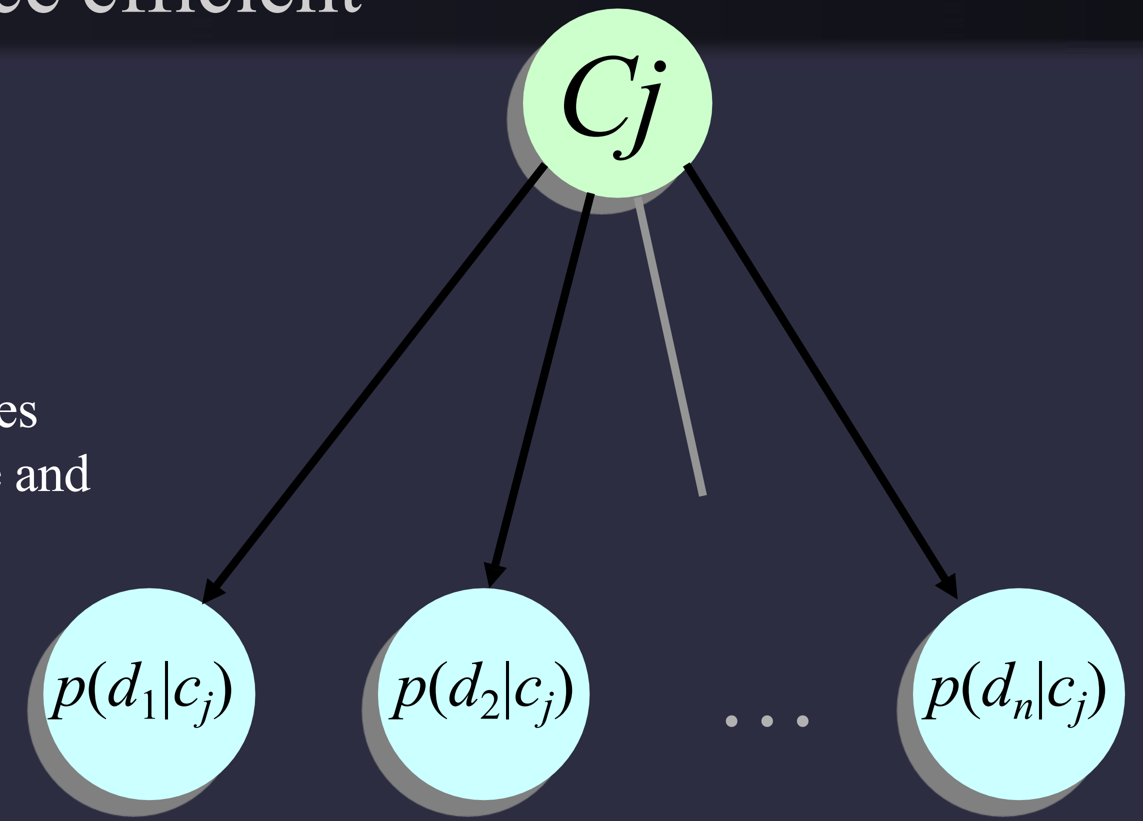
$$p(\text{officer drew}|\ \text{Male}) = 2/3 \quad * \quad 2/3 \quad * \quad \ldots$$

The Naive Bayes classifiers is often represented as this type of graph…

Note the direction of the arrows, which state that each class causes certain features, with a certain probability



Naïve Bayes is fast and space efficient



We can look up all the probabilities with a single scan of the database and store them in a (small) table…

Sex	Over190$_{cm}$	
Male	Yes	0.15
	No	0.85
Female	Yes	0.01
	No	0.99

Sex	Long Hair	
Male	Yes	0.05
	No	0.95
Female	Yes	0.70
	No	0.30

Sex	
Male	
Female	

Naïve Bayes is NOT sensitive to irrelevant features...

Suppose we are trying to classify a persons sex based on several features, including eye color. (Of course, eye color is completely irrelevant to a persons gender)

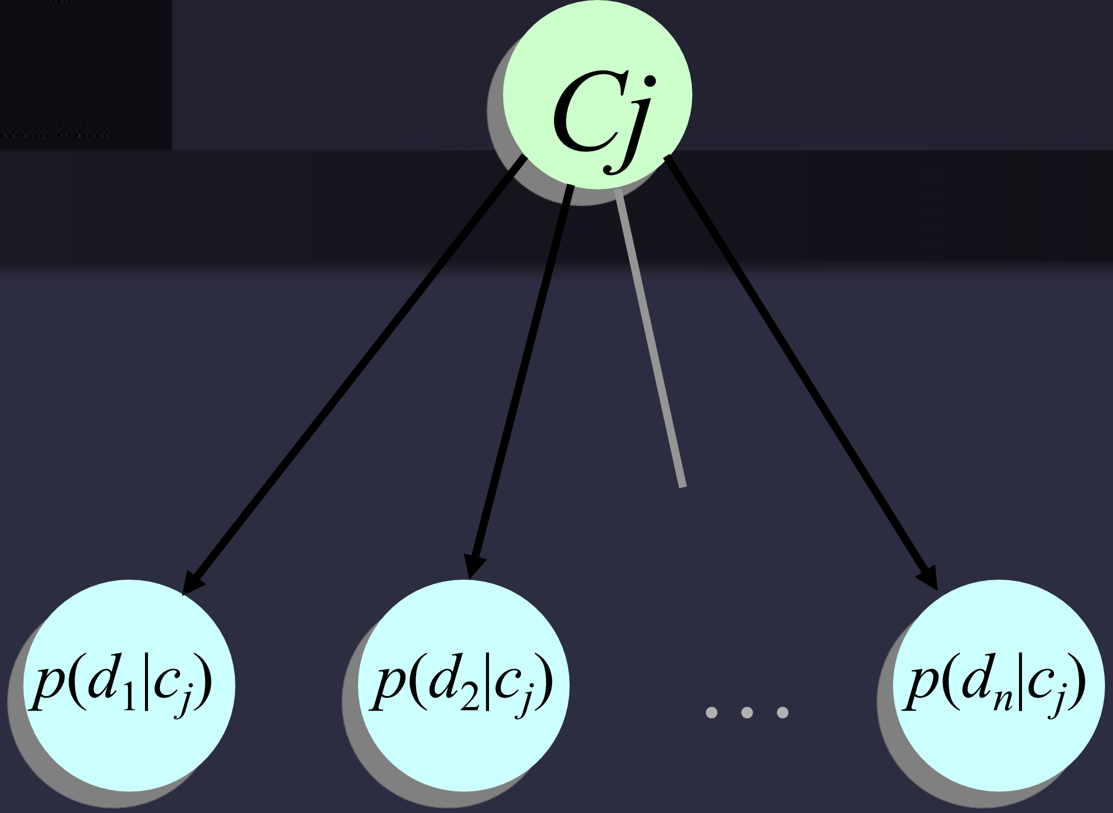
$$p(\text{Jessica}\,|c_j) = p(\text{eye} = \text{brown}|c_j) * p(\text{ wears\_dress} = \text{yes}|c_j) * \ldots.$$

$$p(\text{Jessica}\,|\,\text{Female}) = 9{,}000/10{,}000 \quad * \; 9{,}975/10{,}000 \; * \ldots.$$

$$p(\text{Jessica}\,|\,\text{Male}) = 9{,}001/10{,}000 \quad * \; 2/10{,}000 \; * \ldots.$$

Almost the same!

We can have an arbitrary number of classes, or feature values

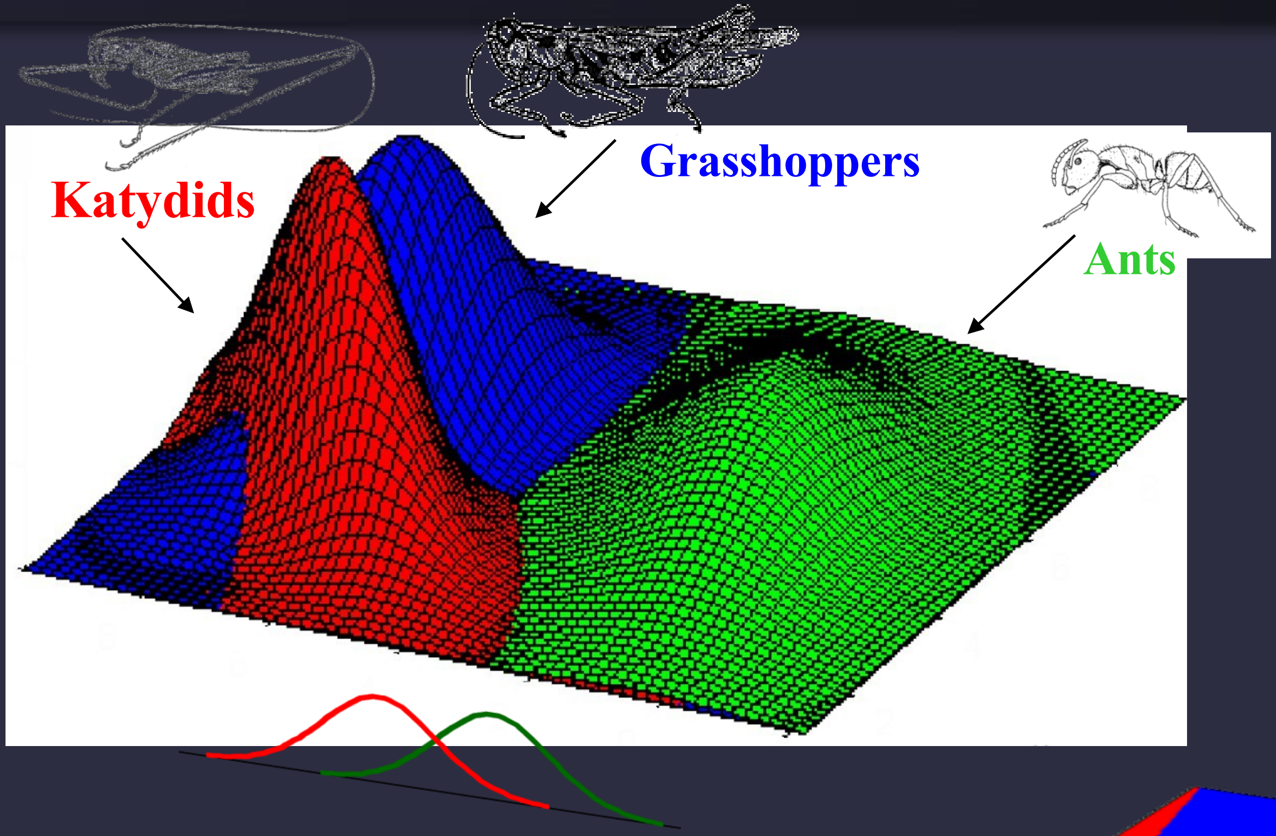


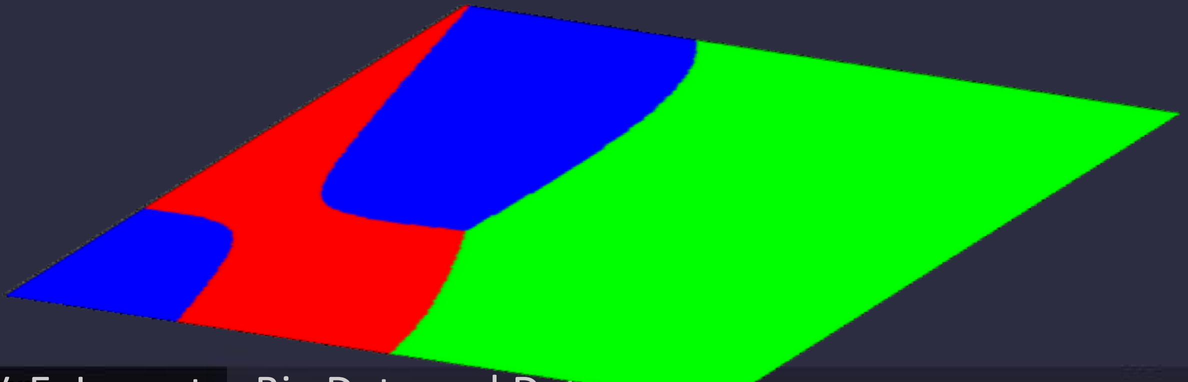
Animal	Mass >10_{kg}	
Cat	Yes	0.15
	No	0.85
Dog	Yes	0.91
	No	0.09
Pig	Yes	0.99
	No	0.01

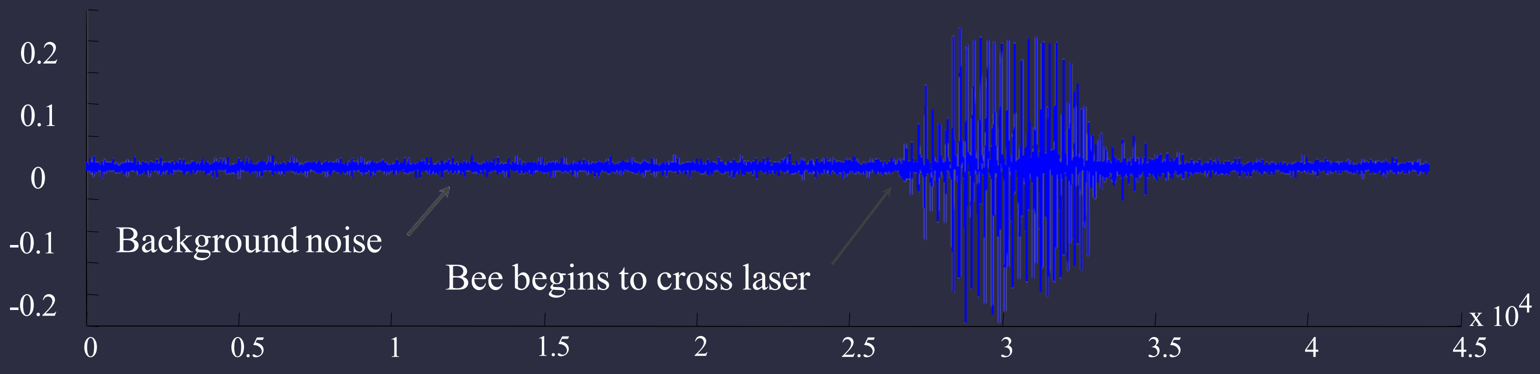
Animal	Color	
Cat	Black	0.33
	White	0.23
	Brown	0.44
Dog	Black	0.97
	White	0.03
	Brown	0.90
Pig	Black	0.04
	White	0.01

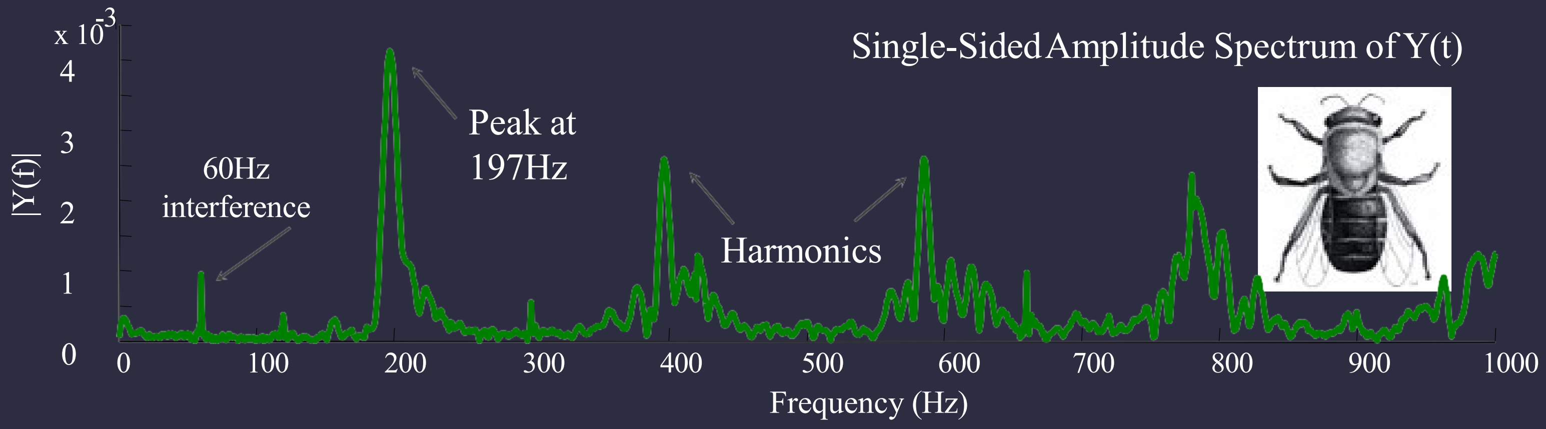
Animal
Cat
Dog
Pig

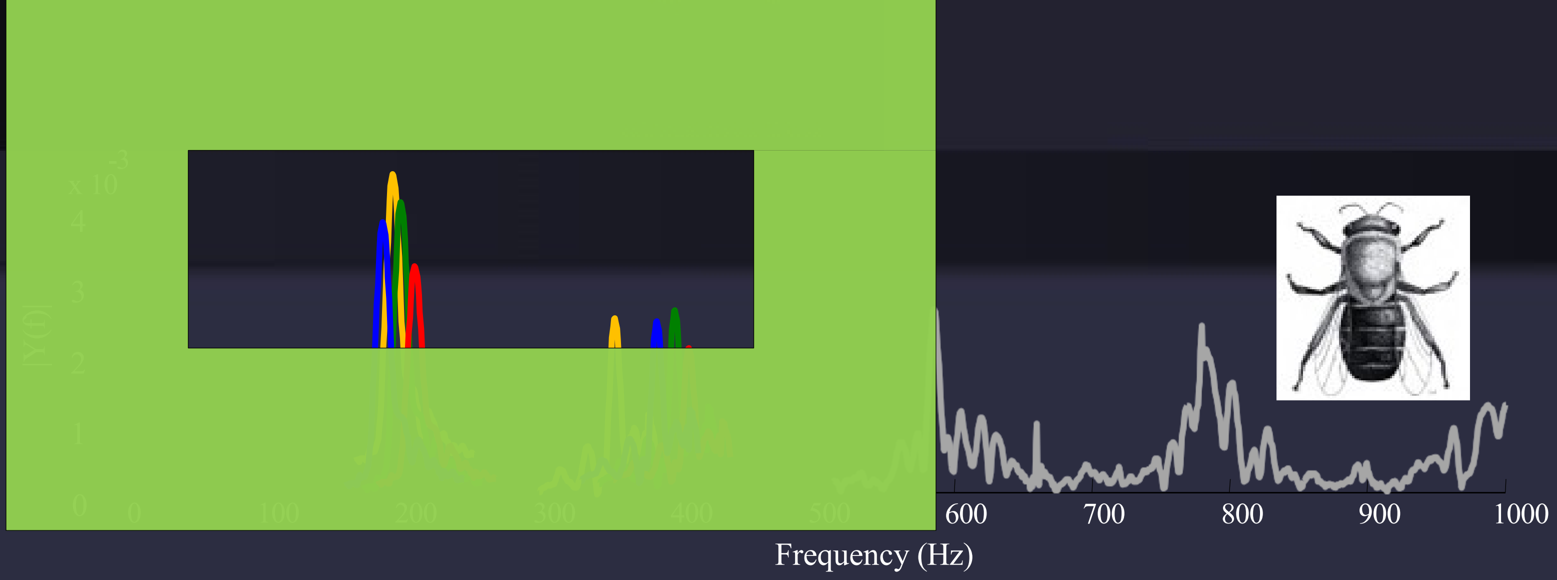
The Naïve Bayesian Classifier has a piecewise quadratic decision boundary



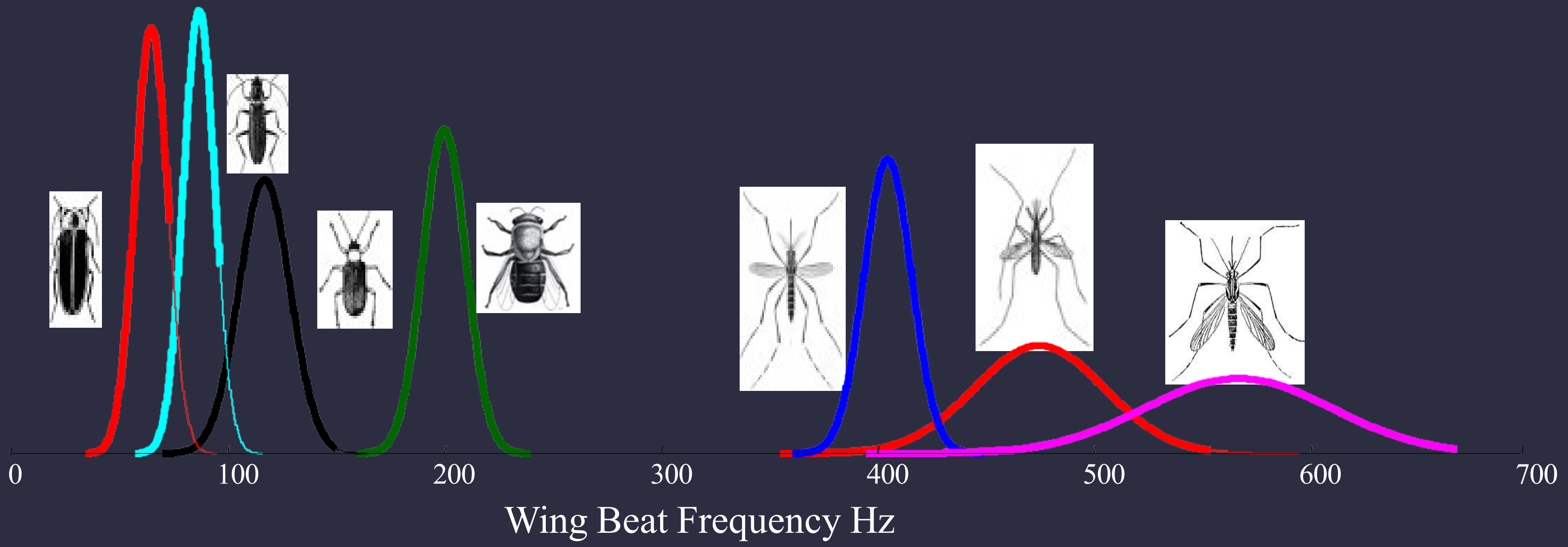


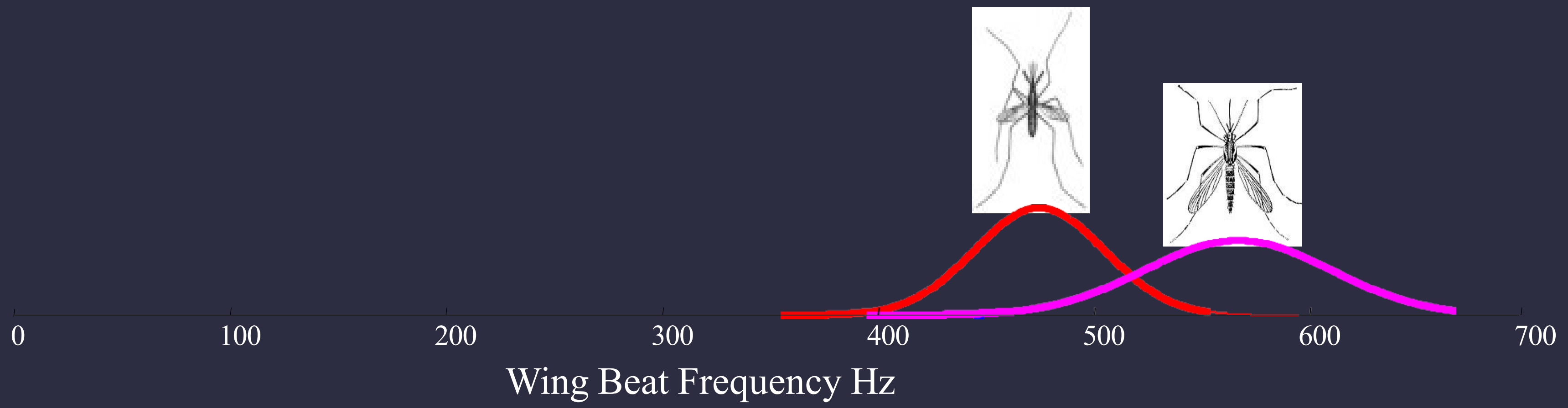

0.2
0.1
0
-0.1
-0.2
Background noise
Bee begins to cross laser
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
x 10^4

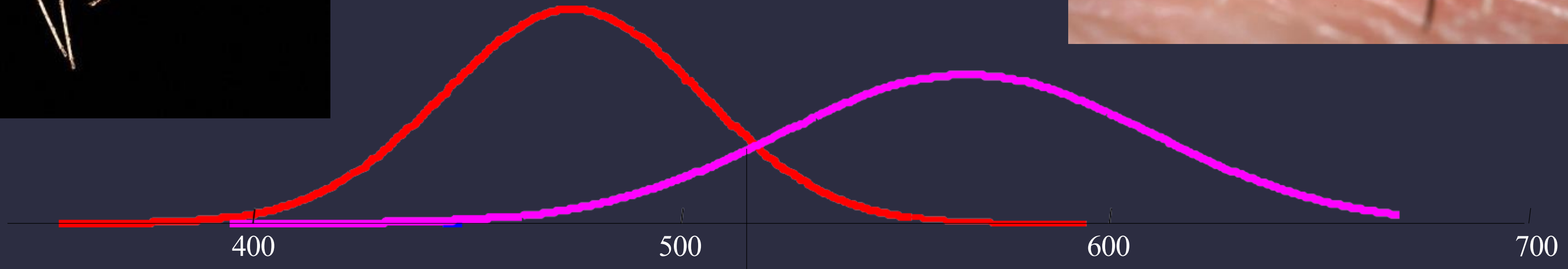

x 10^-3
4
3
2
1
0
|Y(f)|
60Hz interference
Peak at 197Hz
Harmonics
Single-Sided Amplitude Spectrum of Y(t)
0
100
200
300
400
500
600
700
800
900
1000
Frequency (Hz)



$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$


0
100
200
300
400
500
600
700
Wing Beat Frequency Hz


0
100
200
300
400
500
600
700
Wing Beat Frequency Hz

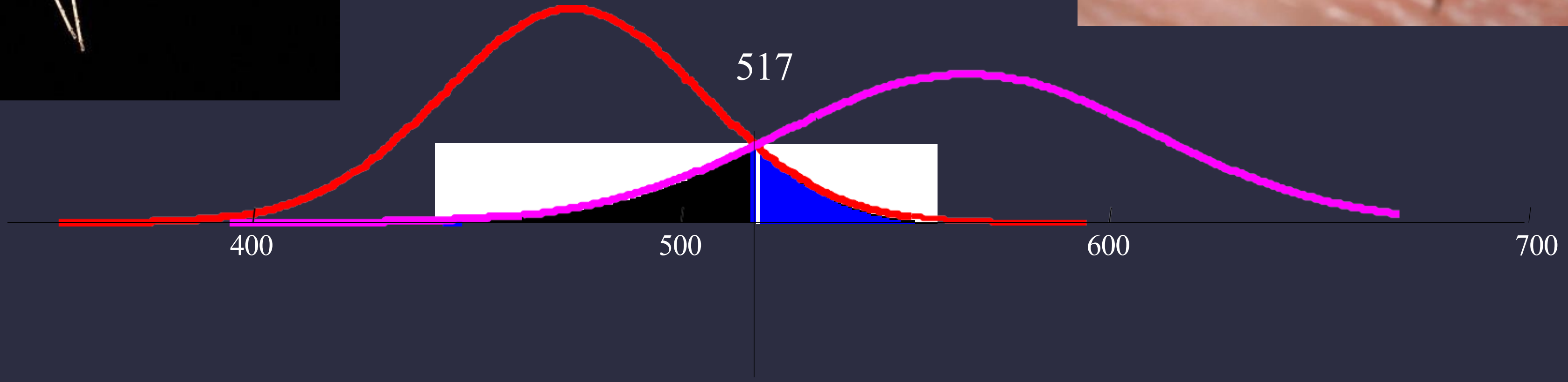


Anopheles stephensi: Female
mean =475, Std = 30

517

Aedes aegyptii : Female
mean =567, Std = 43

If I see an insect with a wingbeat frequency of 500, what is it?



What is the error rate?

12.2% of the area under the pink curve

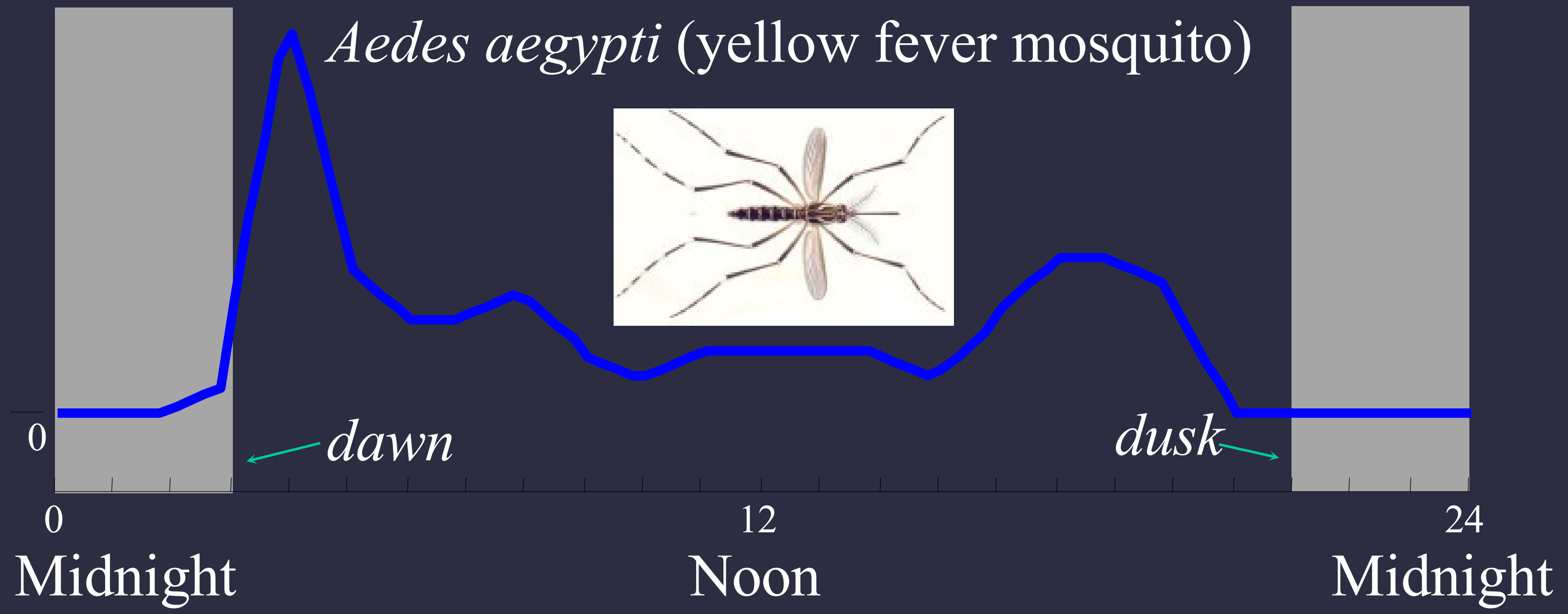
8.02% of the area under the red curve

Can we get more features?

$$\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

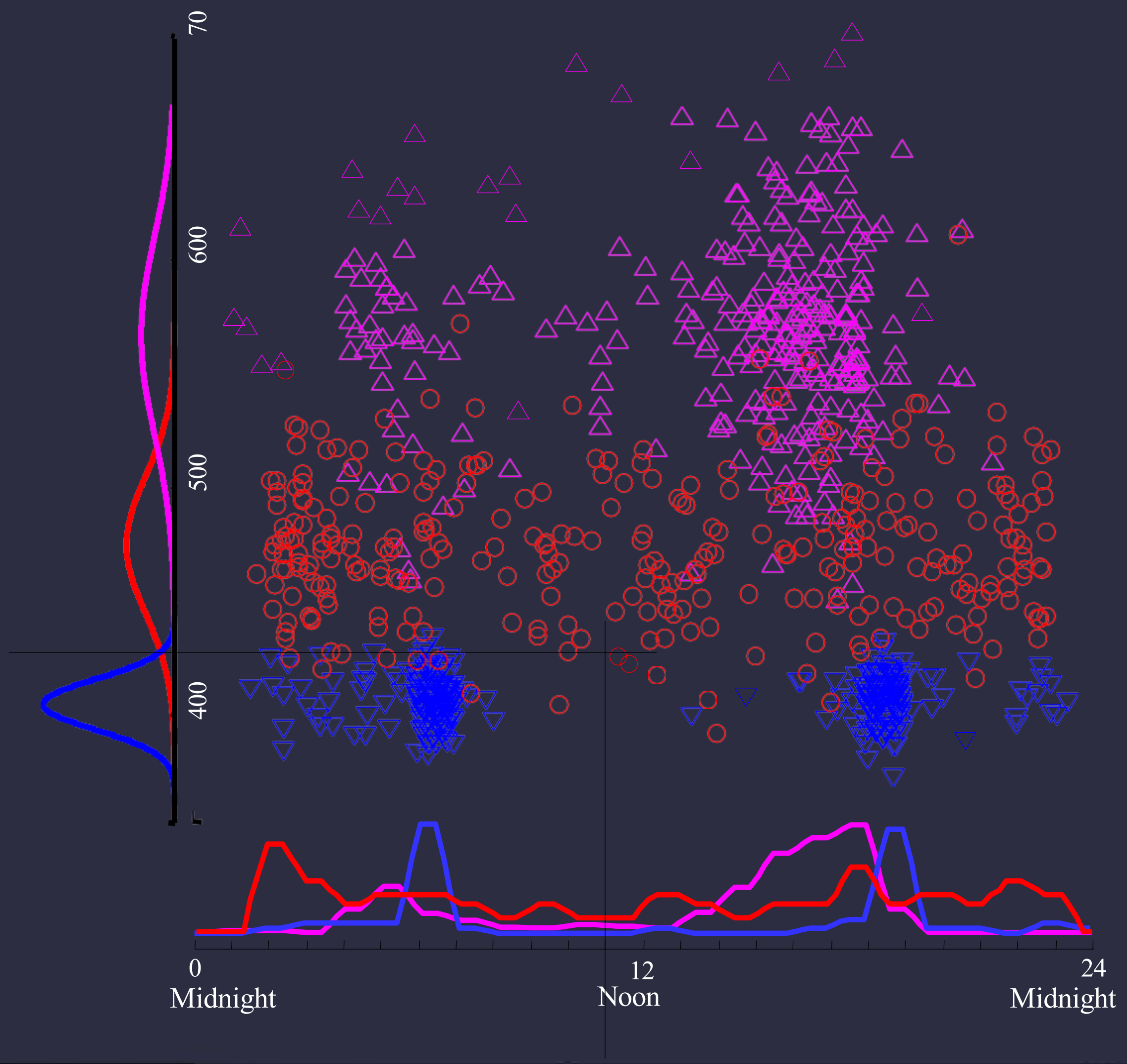
$$P(Anopheles|wingbeat = 500) = \frac{1}{\sqrt{2\pi}\,30} e^{-\frac{(500-475)^2}{2\times 30^2}}$$

Circadian Features



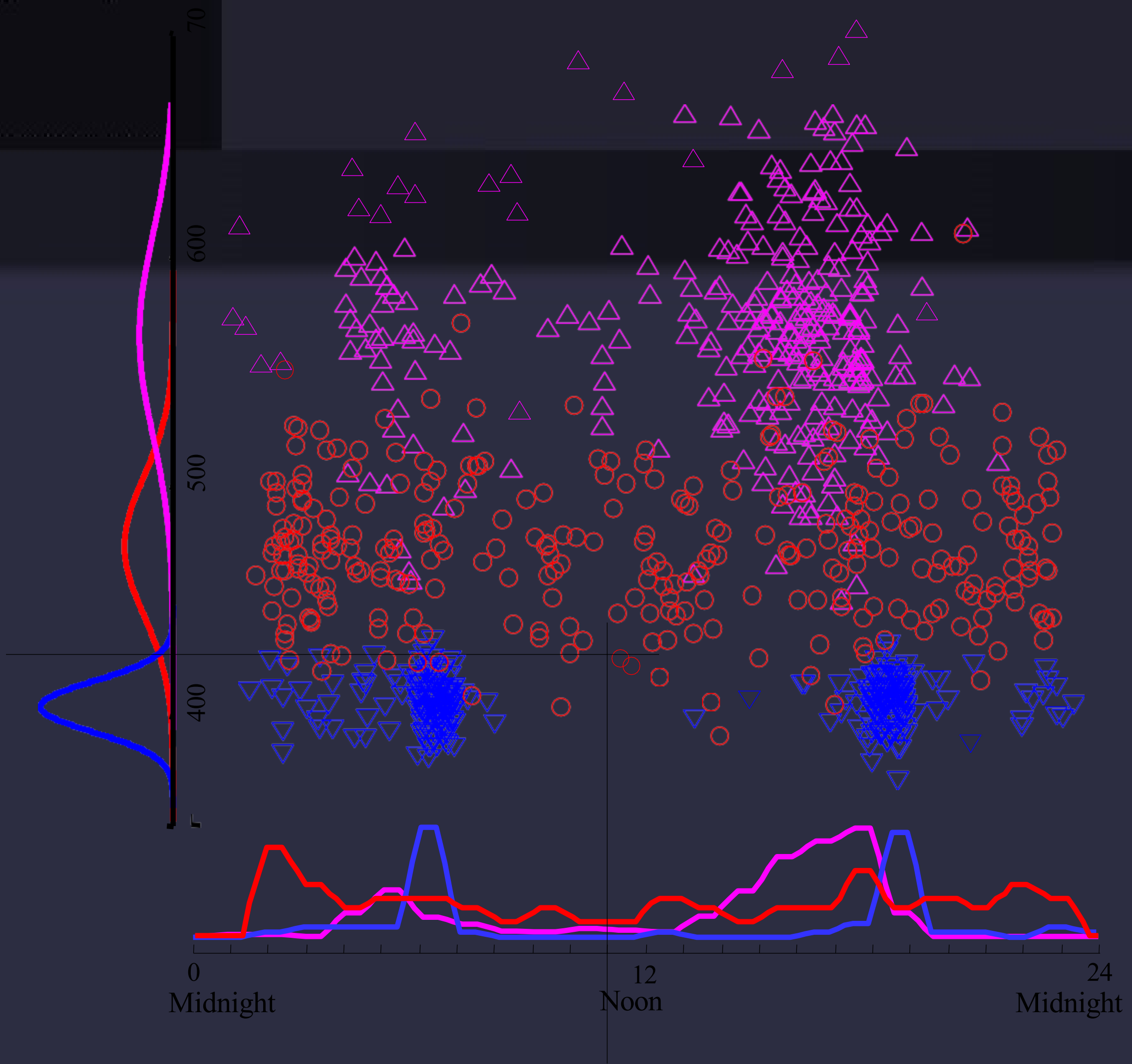
Suppose I observe an insect with a wingbeat frequency of 420Hz **at 11:00am**

What is it?



Suppose I observe an insect with a wingbeat frequency of 420 at 11:00am

What is it?



(Culex | [420Hz,11:00am]) = (6/ (6 + 6 + 0)) * (2/ (2 + 4 + 3)) = 0.111

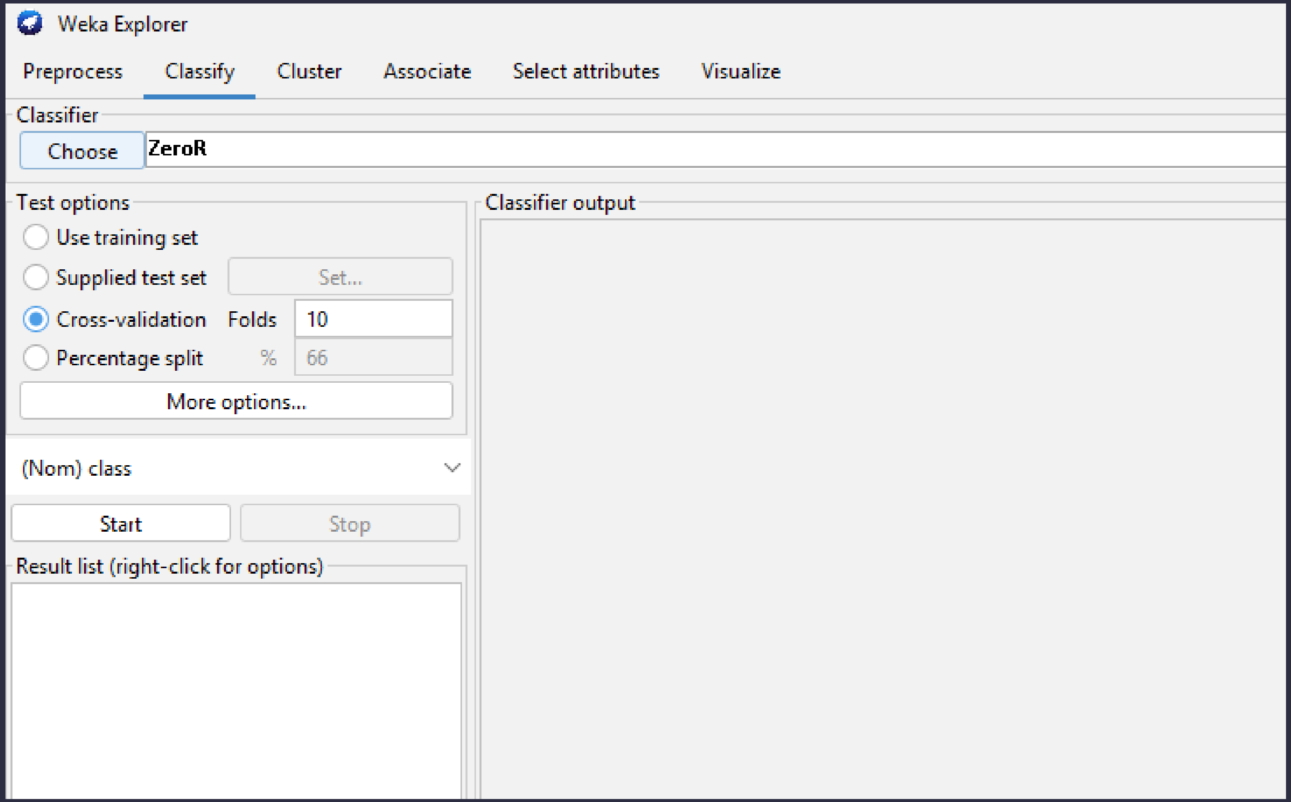
(Anopheles | [420Hz,11:00am]) = (6/ (6 + 6 + 0)) * (4/ (2 + 4 + 3)) = 0.222

(Aedes | [420Hz,11:00am]) = (0/ (6 + 6 + 0)) * (3/ (2 + 4 + 3)) = 0.000

Advantages/Disadvantages of Naïve Bayes

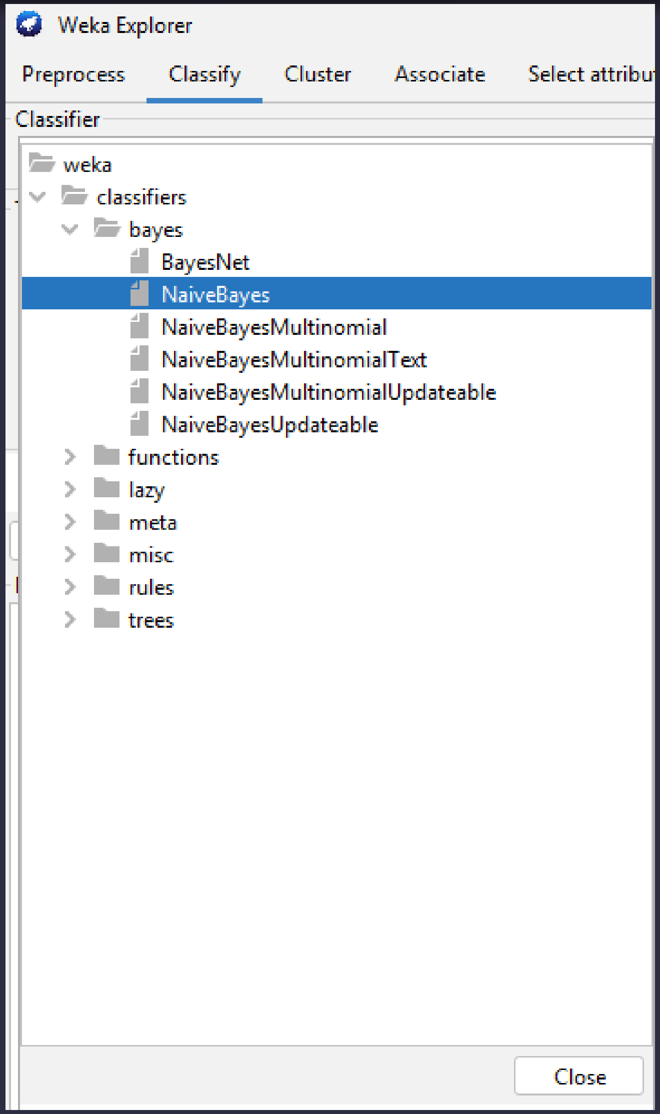
- Advantages:
  - Fast to train (single scan). Fast to classify
  - Not sensitive to irrelevant features
  - Handles real and discrete data
  - Handles streaming data well
- Disadvantages:
  - Assumes independence of features

Load Iris Dataset and goto Classify Tab



Naïve Bayes

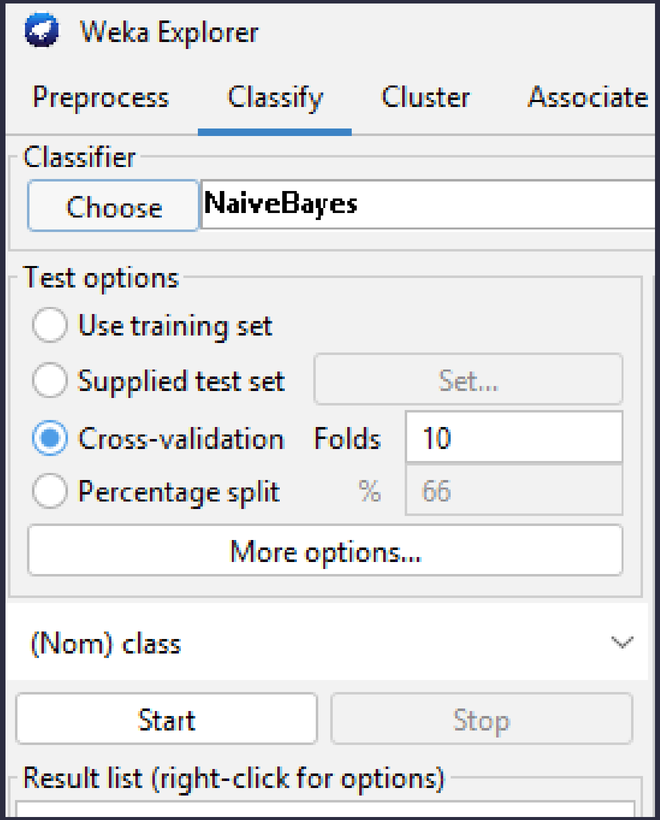
Select Naive Bayes Classifier



Naïve Bayes

Cross-Validation

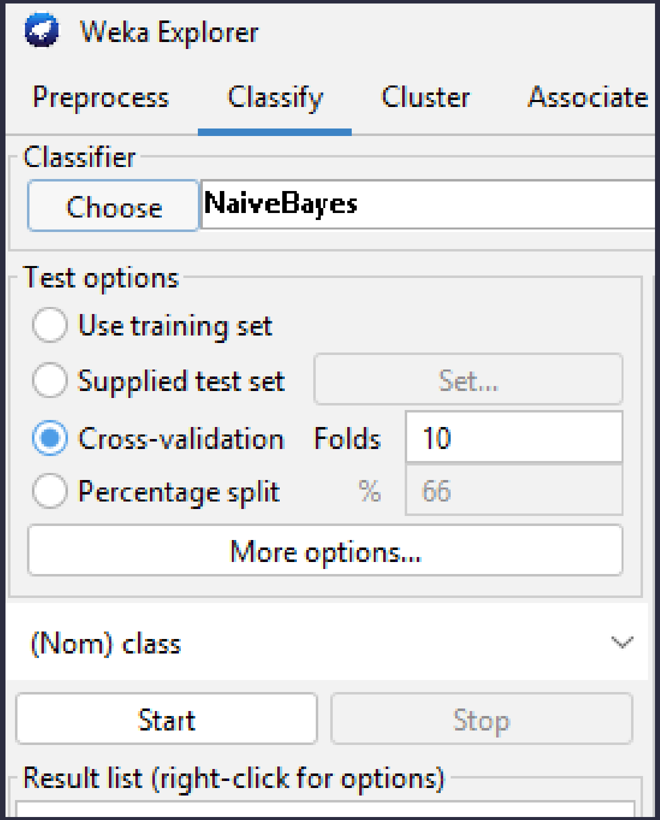
- 10 percent will used for only validation



Naïve Bayes

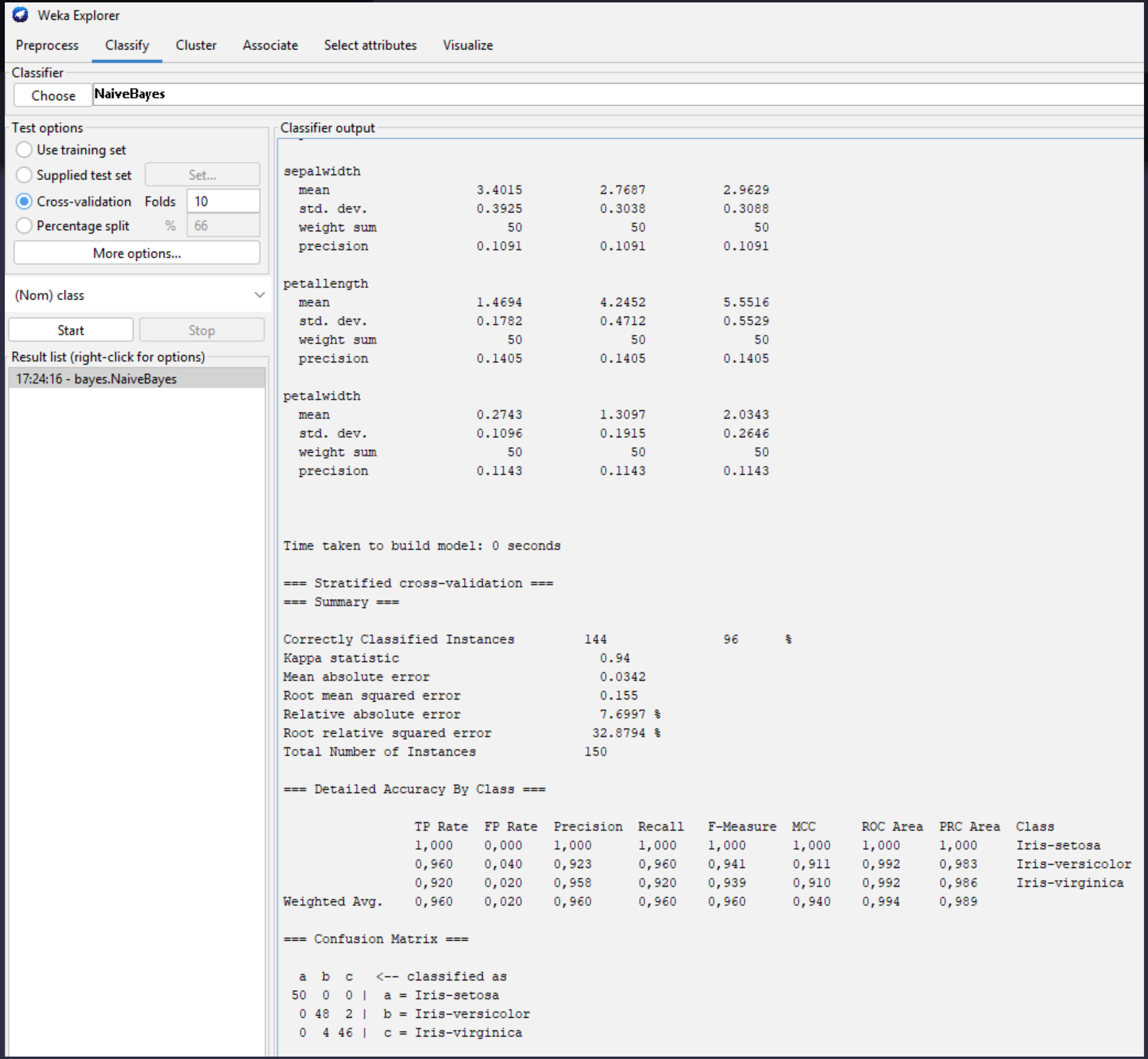
Cross-Validation

- 10 percent will used for only validation



Naïve Bayes

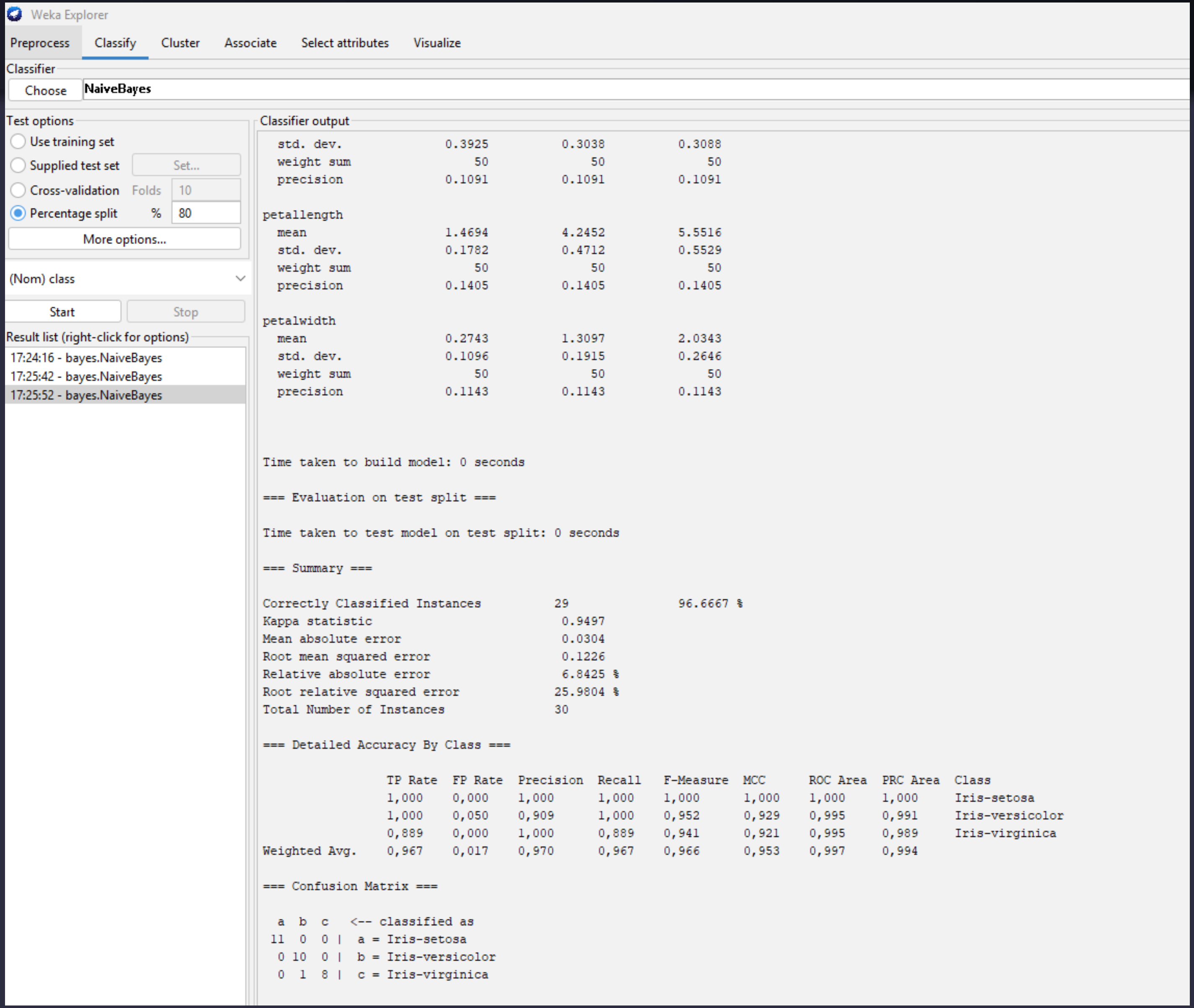
Start



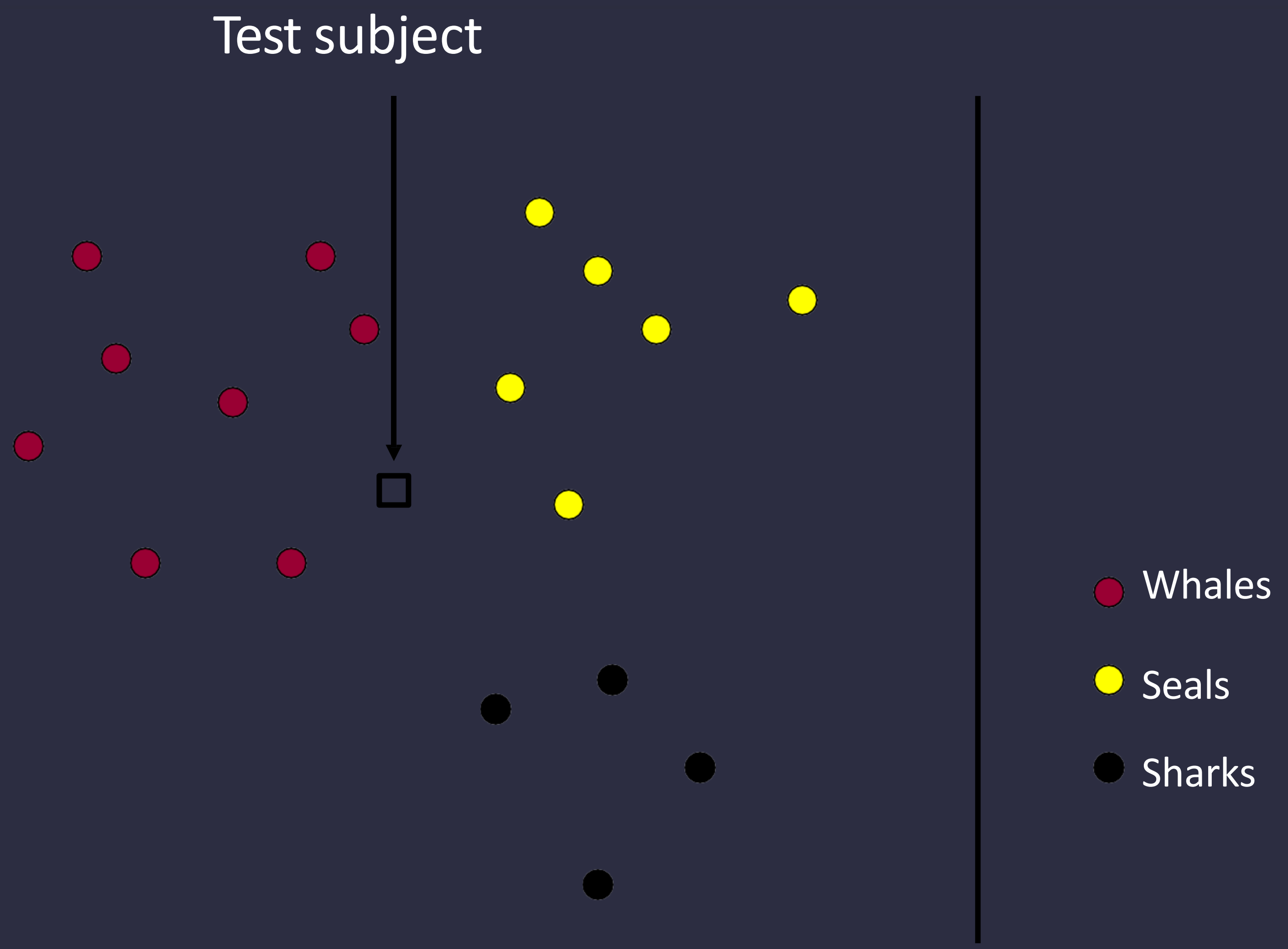
Naïve Bayes

Start

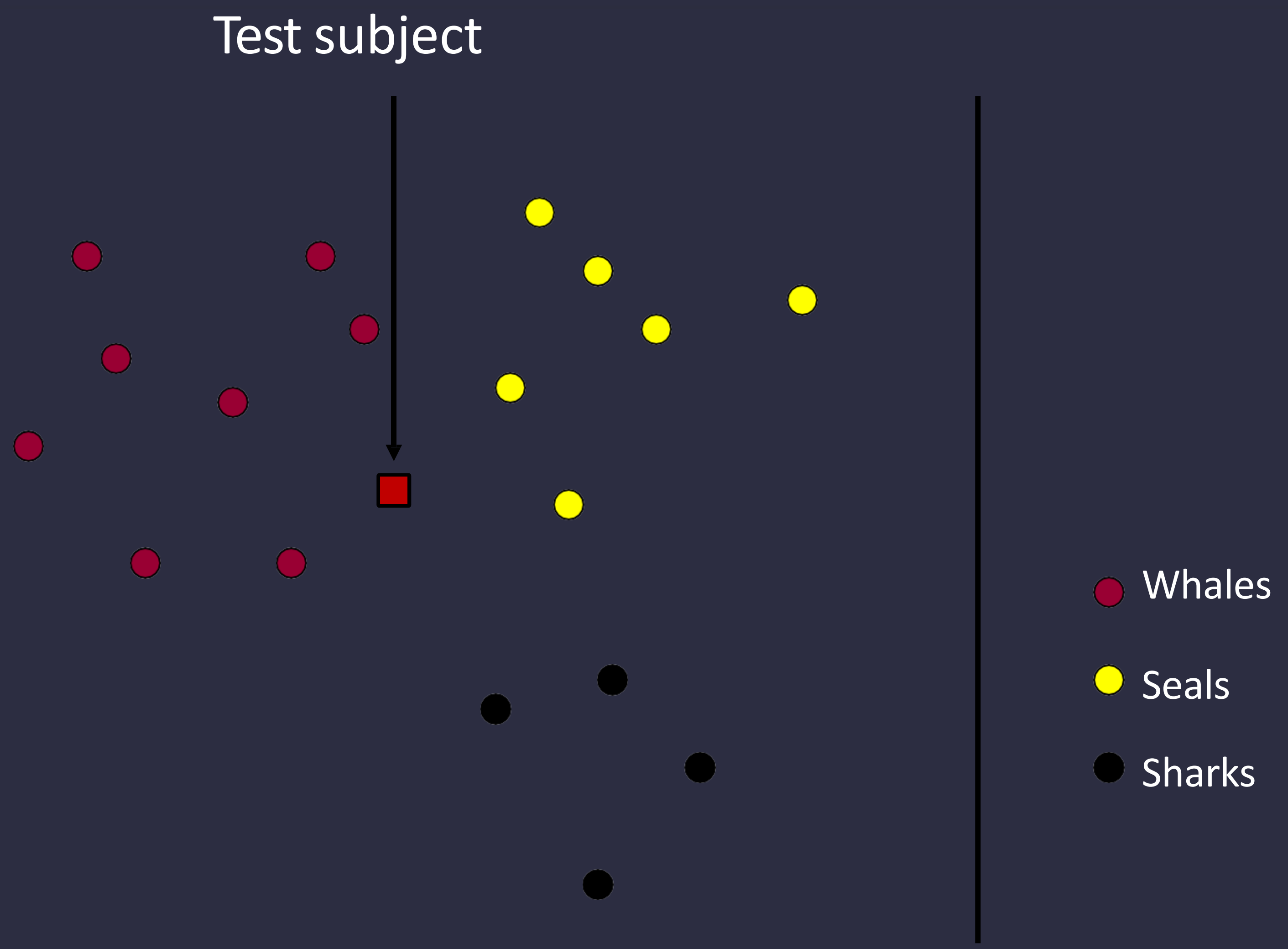
Percentage Split



K-Nearest Neighbor



Nearest Neighbor Classifier



Nearest Neighbor Classification

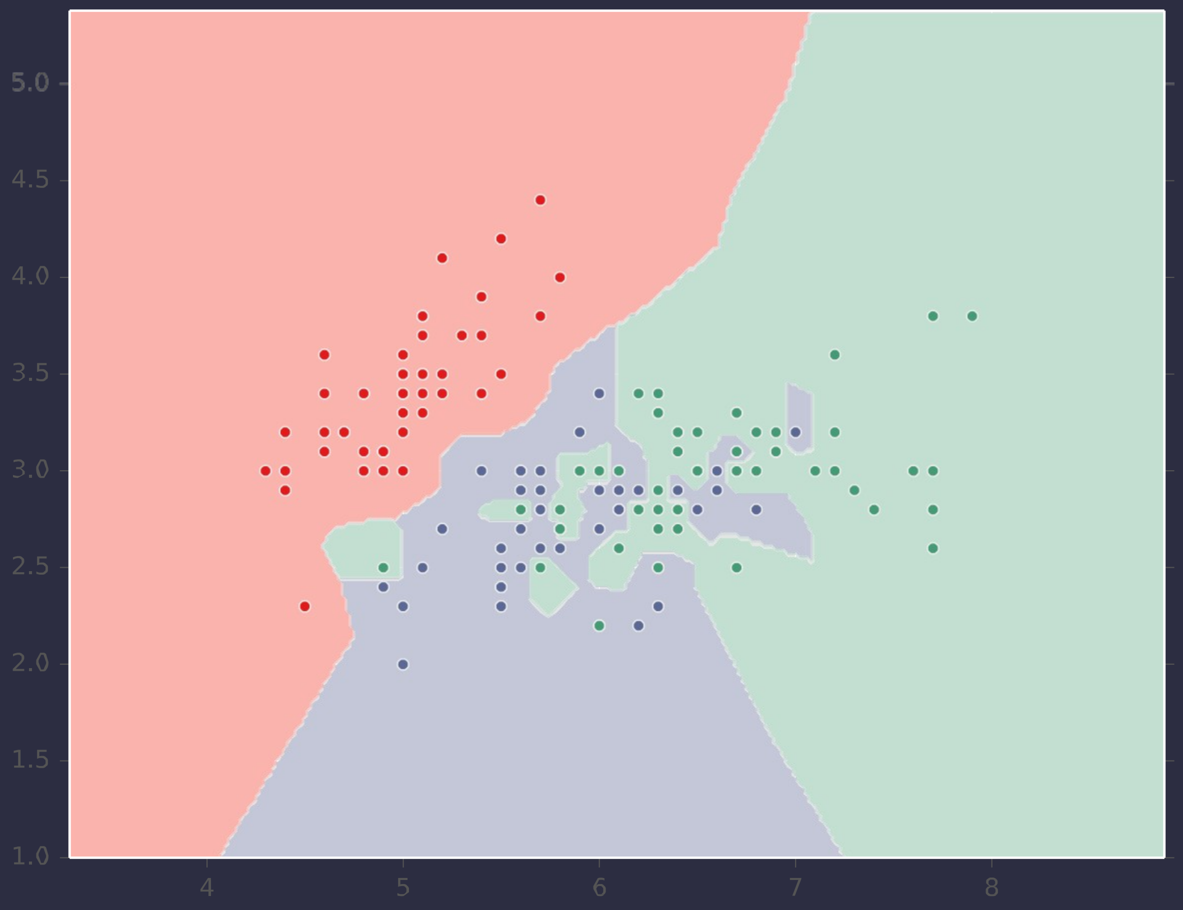
Given a training dataset $\mathcal{D} = \{y^{(n)}, \mathbf{x}^{(n)}\}_{n=1}^{N}$, $y \in \{1, \ldots, C\}$, $\mathbf{x} \in \mathbb{R}^M$

and a test input $\mathbf{x}_{test}$, predict the class label

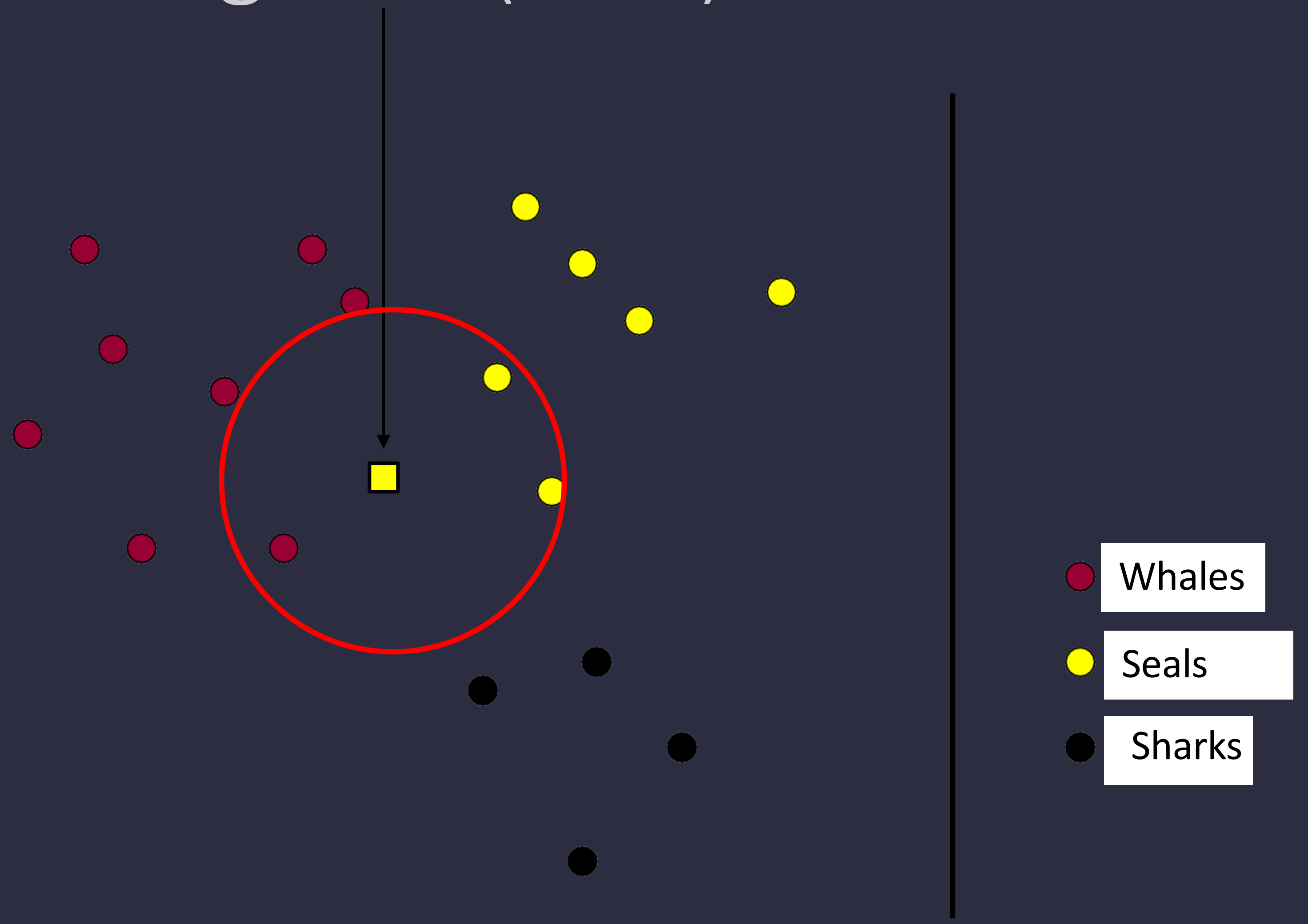
1) Find the closest point in the training data to $\mathbf{x}_{test}$
2) Return the class label of that closest point

Need distance function! What should $d(\mathbf{x}, \mathbf{z})$ be?

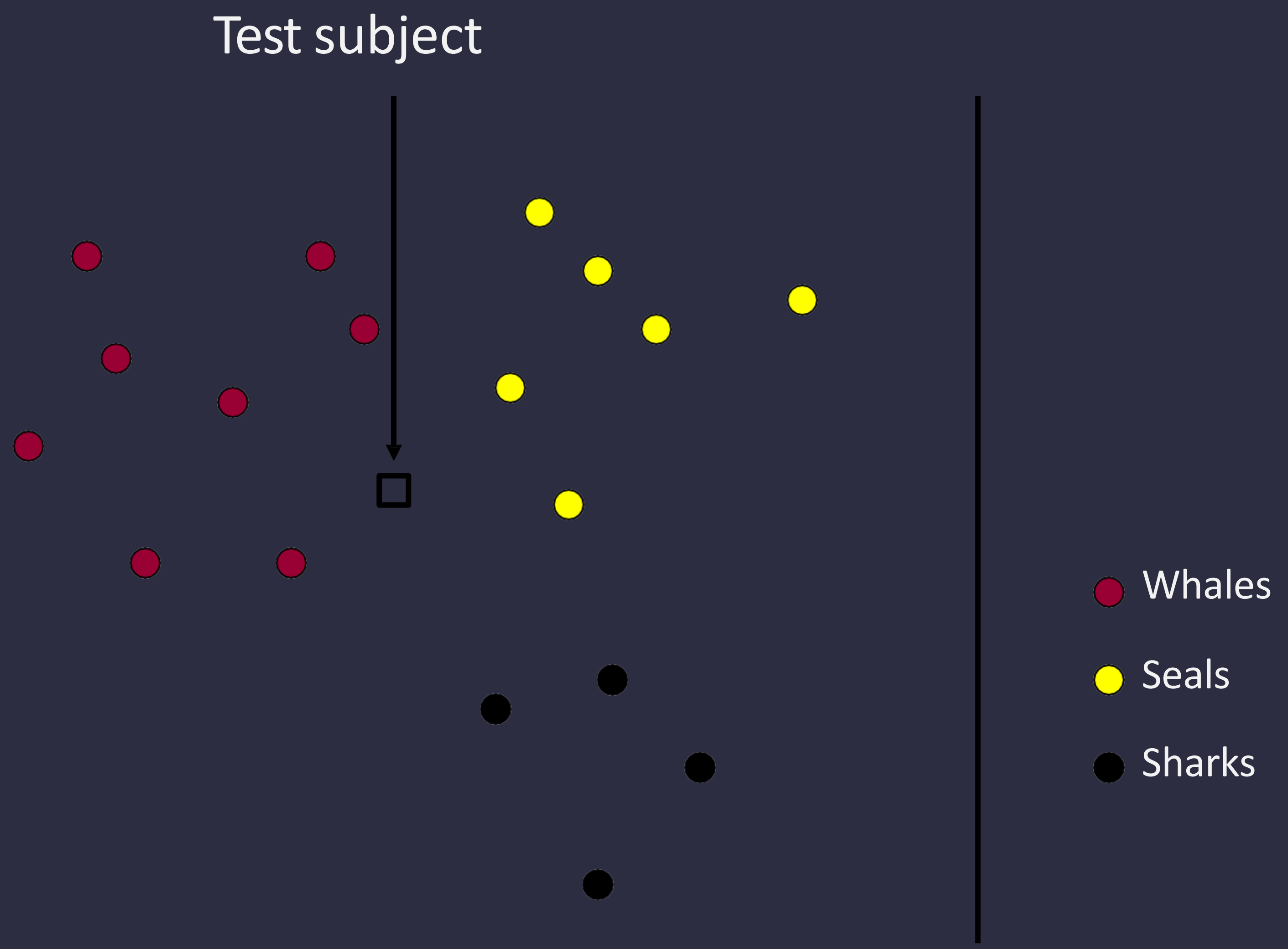
Nearest Neighbor on Fisher Iris Data



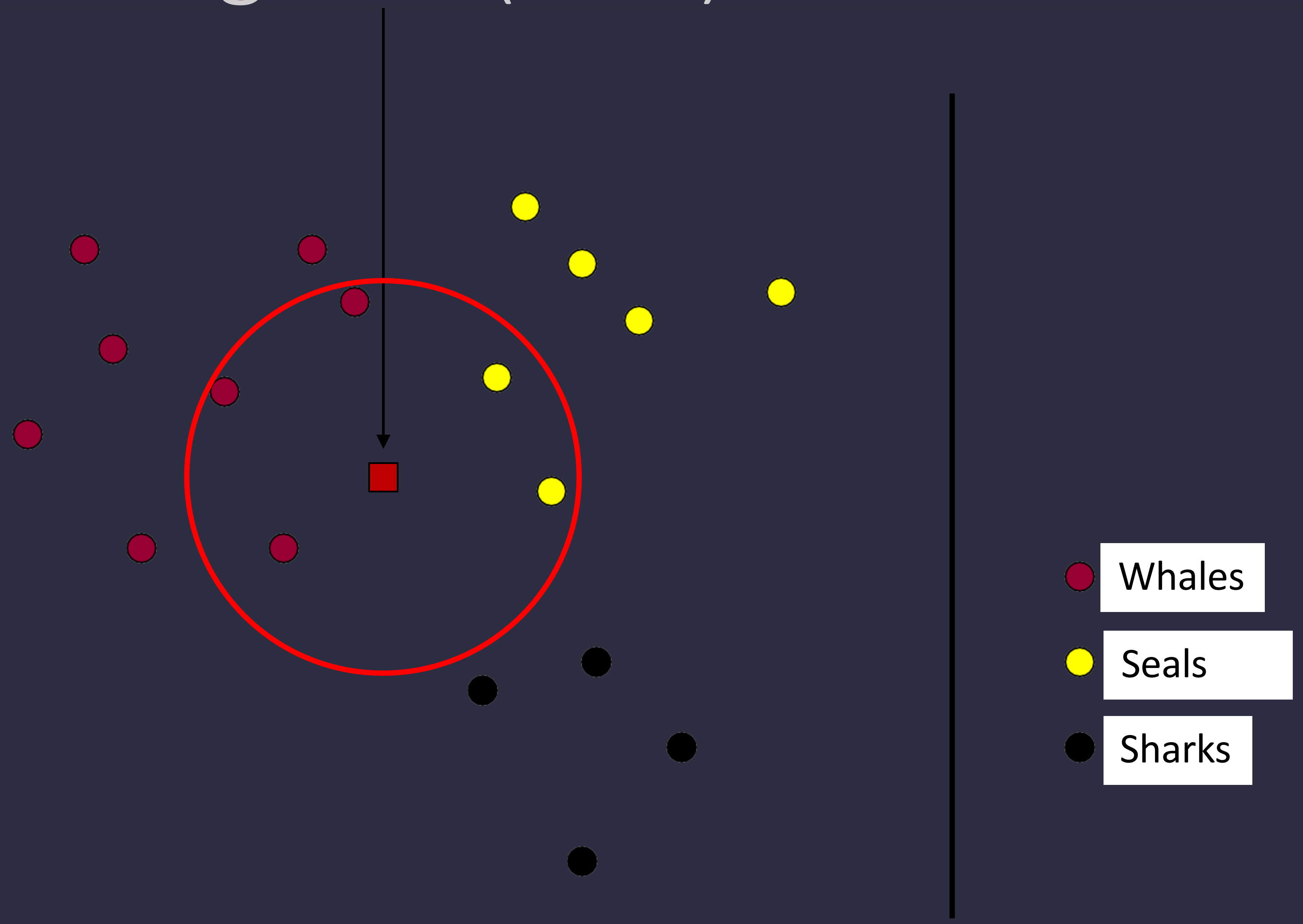
3-Nearest Neighbor (kNN) classifier



kNN classifier (k=5)



5-Nearest Neighbor (kNN) classifier



What is the best k?

How do we choose a learner that is accurate and also generalizes to unseen data?

- Larger k → predicted label is more stable
- Smaller k → predicted label is more affected by individual training points

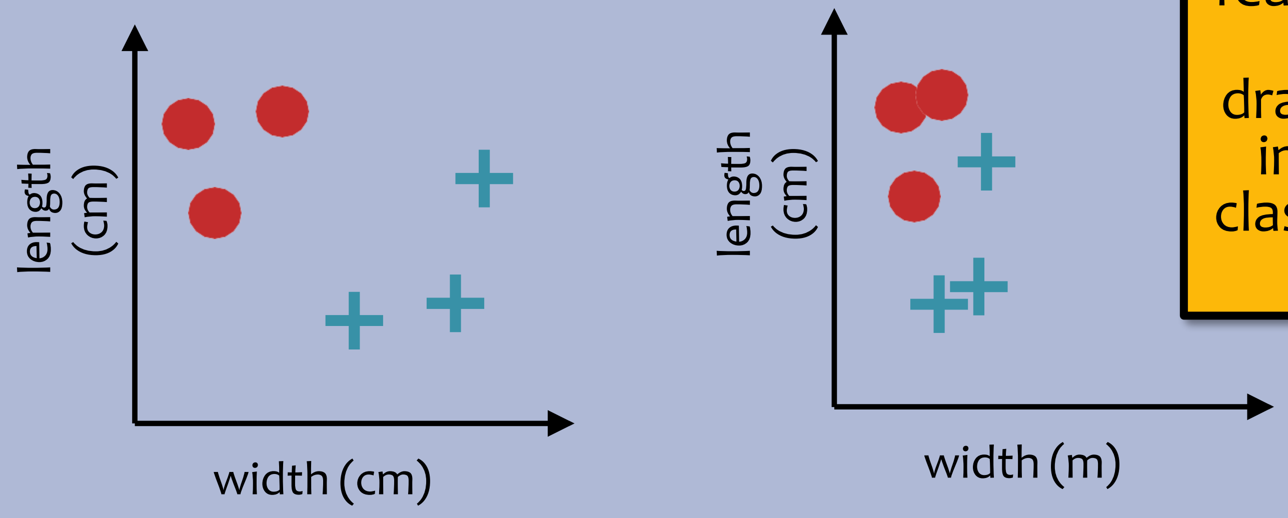
But how to choose k?

k-NN: Details

Inductive Bias:

1. Close points should have similar labels
2. All dimensions are created equally!

Example: two features for k-NN



big problem: feature scale could dramatically influence classification results

KNN Mini Project

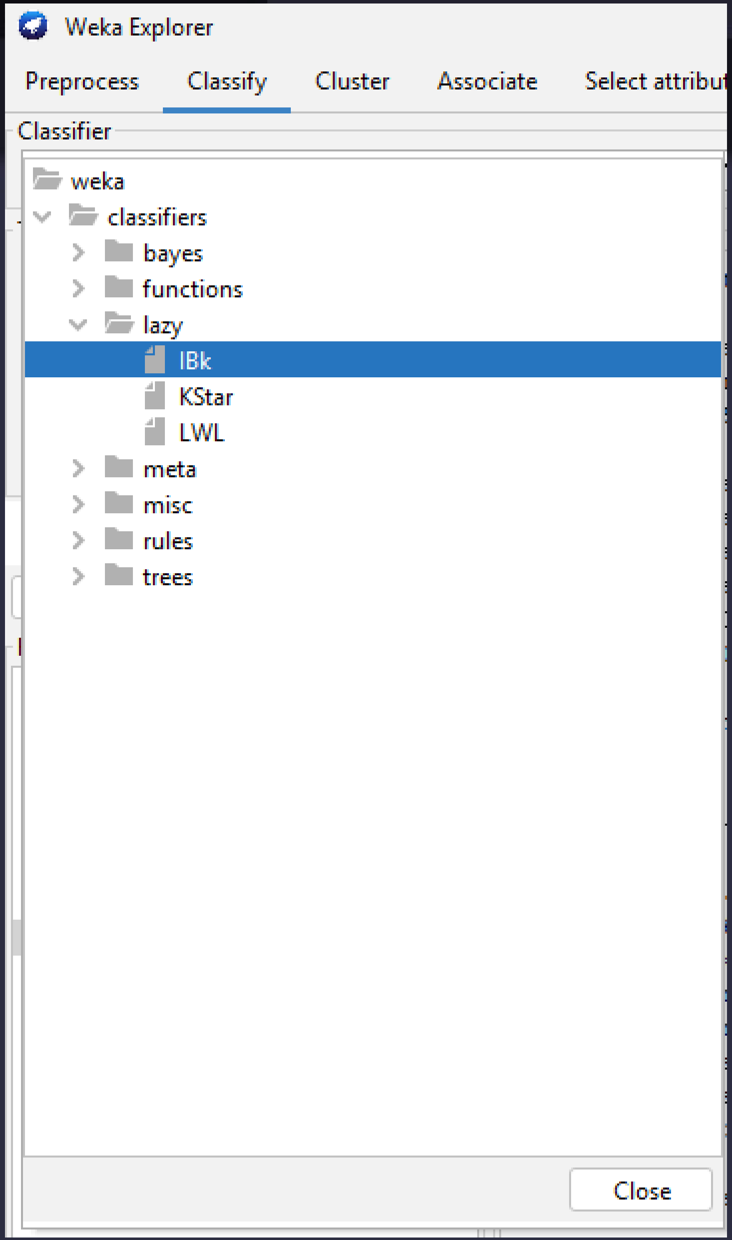
Visit to view project specifications

http://levent.tc/files/courses/big_data/mini_projects/knn/proje1_knn.pdf

Prepare a presentation

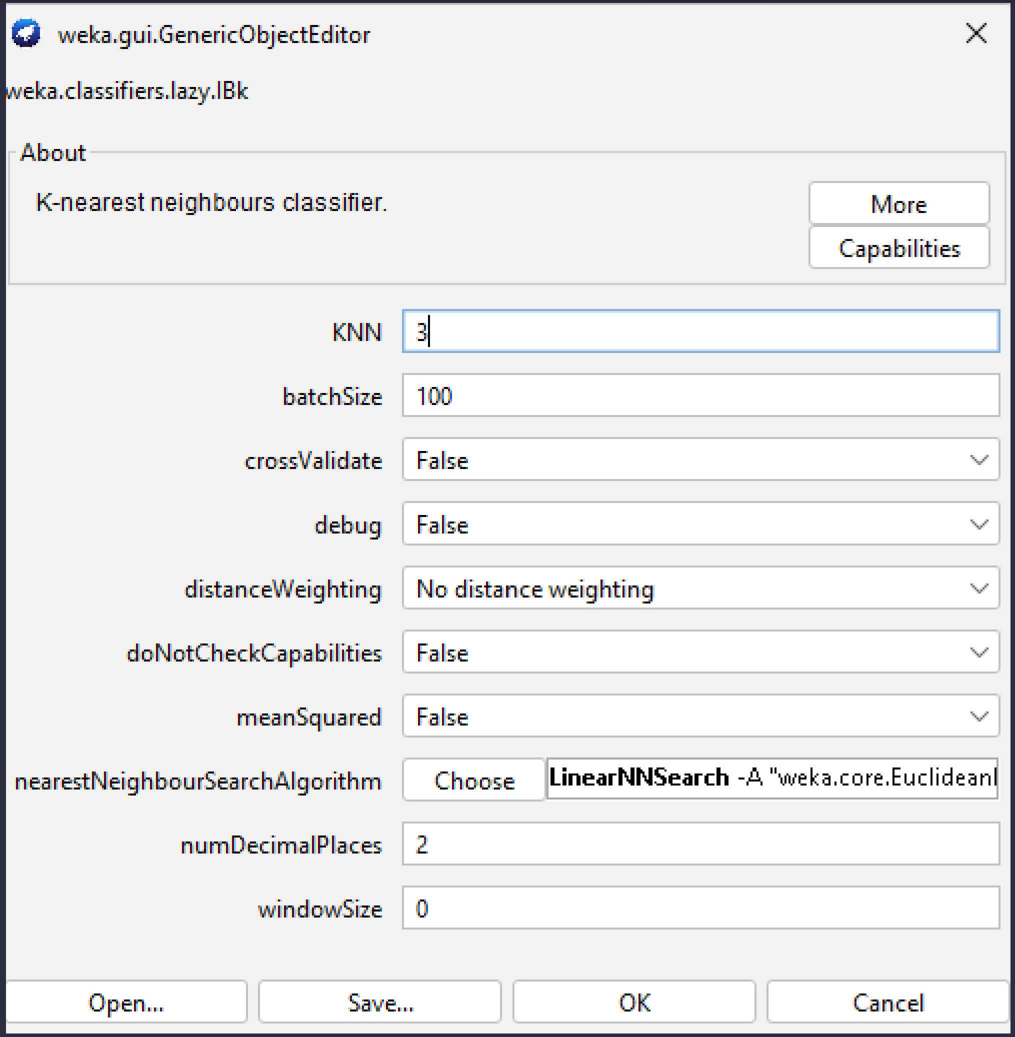
KNN on Weka

Select Classifier



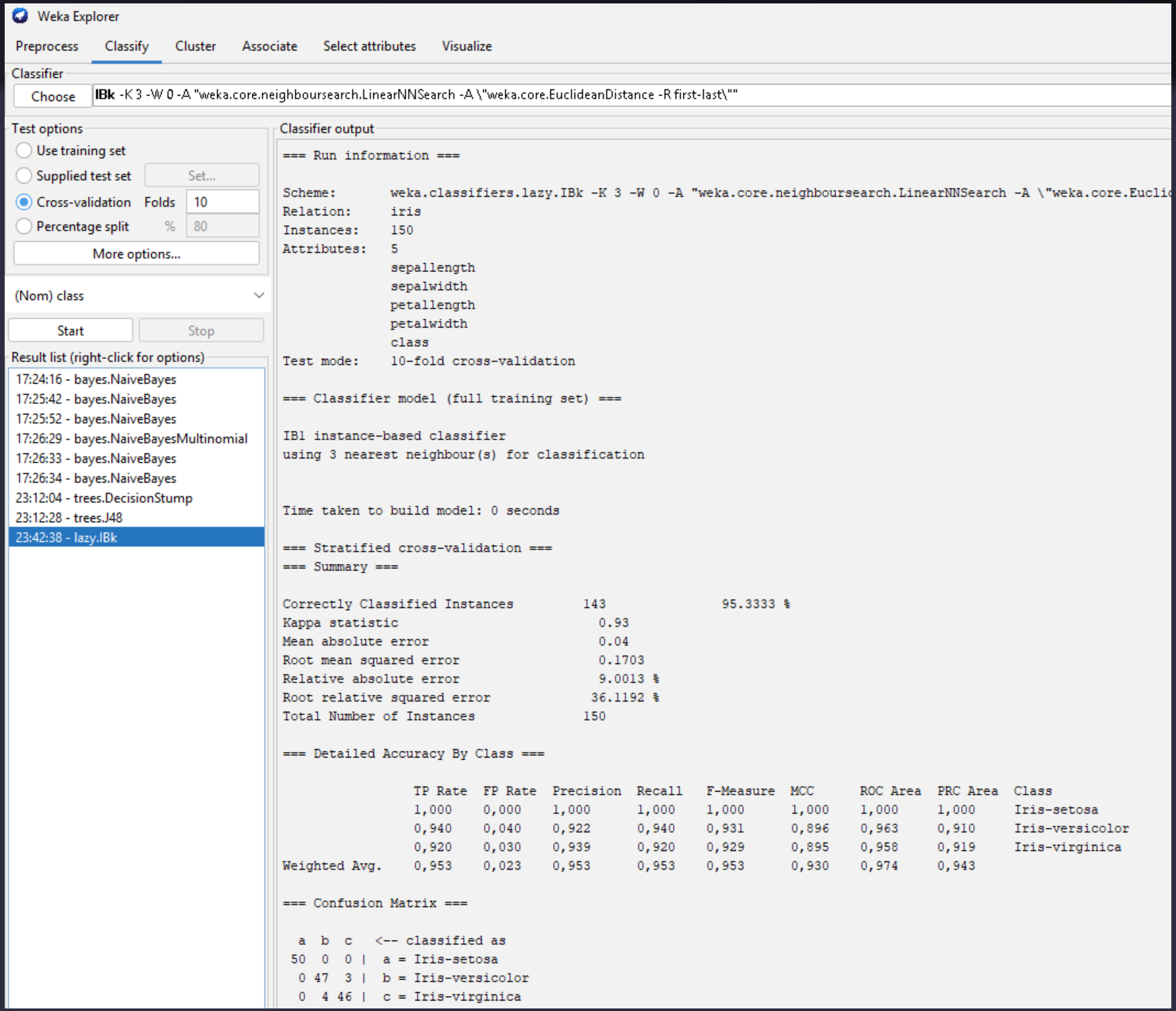
KNN on Weka

Settings



KNN on Weka

Classification



Decision Tree

Problem Setting

- Set of possible instances X
- Set of possible labels Y
- Unknown target function $f : X \text{ -> } Y$
- Set of function hypotheses $H = \{h \mid h : X \text{ -> } Y\}$

Input: Training examples of unknown target function f

$$\{<\mathbf{x}_i, y_>\}_{i=1}^{n} = \{<\mathbf{x}_1, y_{1>}, \ldots, <\mathbf{x}_n, y_{n>}\}$$

Output: Best approximates f

Sample Dataset

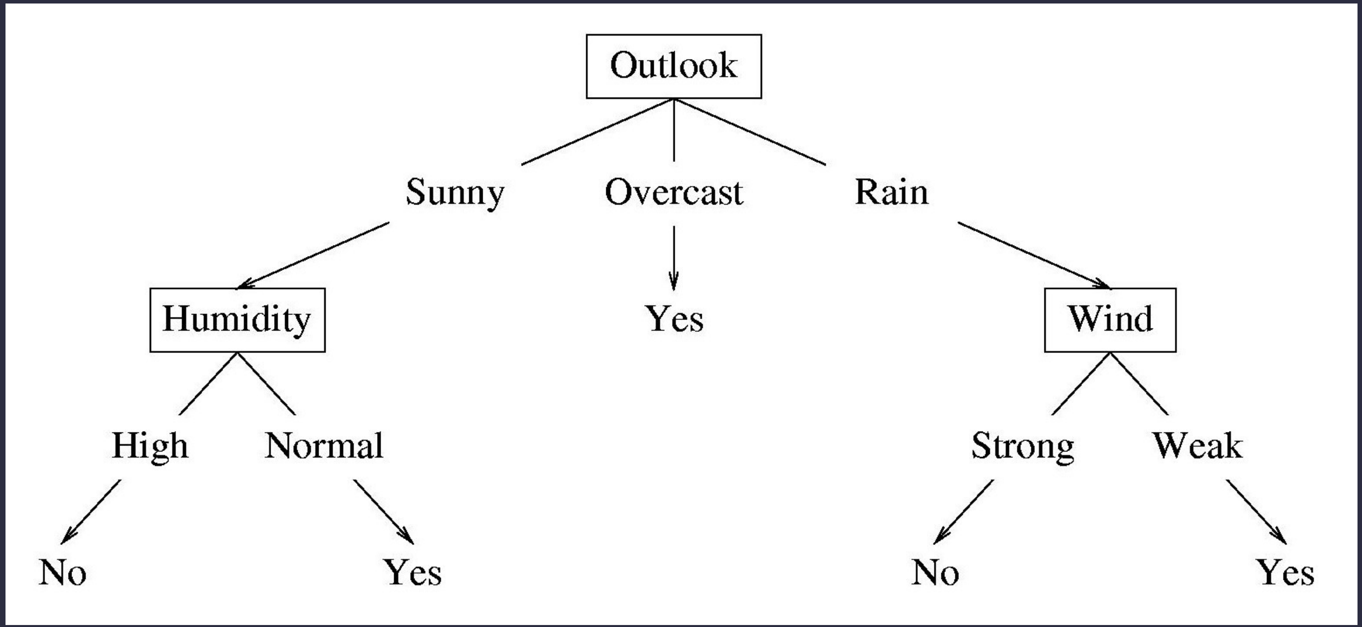
- Columns denote features X_i
- Rows denote labeled instances $\boldsymbol{x}_i, y_i$
- Class label denotes whether a tennis game was played

$\boldsymbol{x}_i, y_i$

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Decision Tree

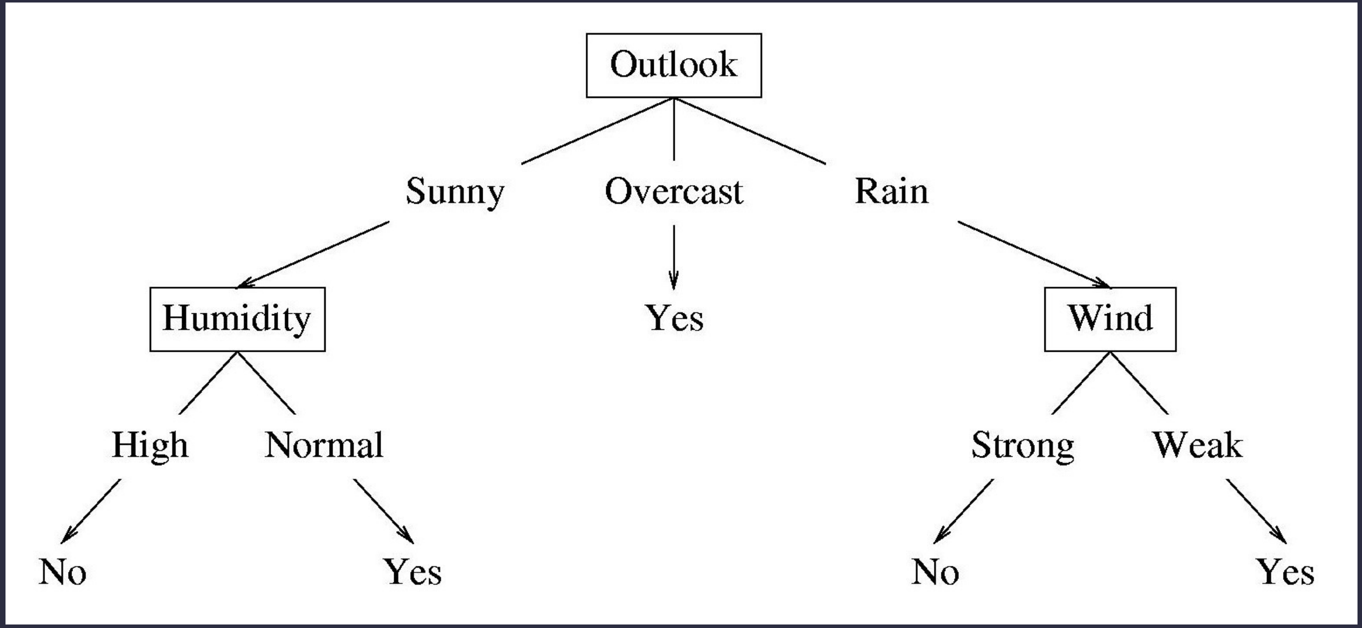
- A possible decision tree for the data:



- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y

Decision Tree

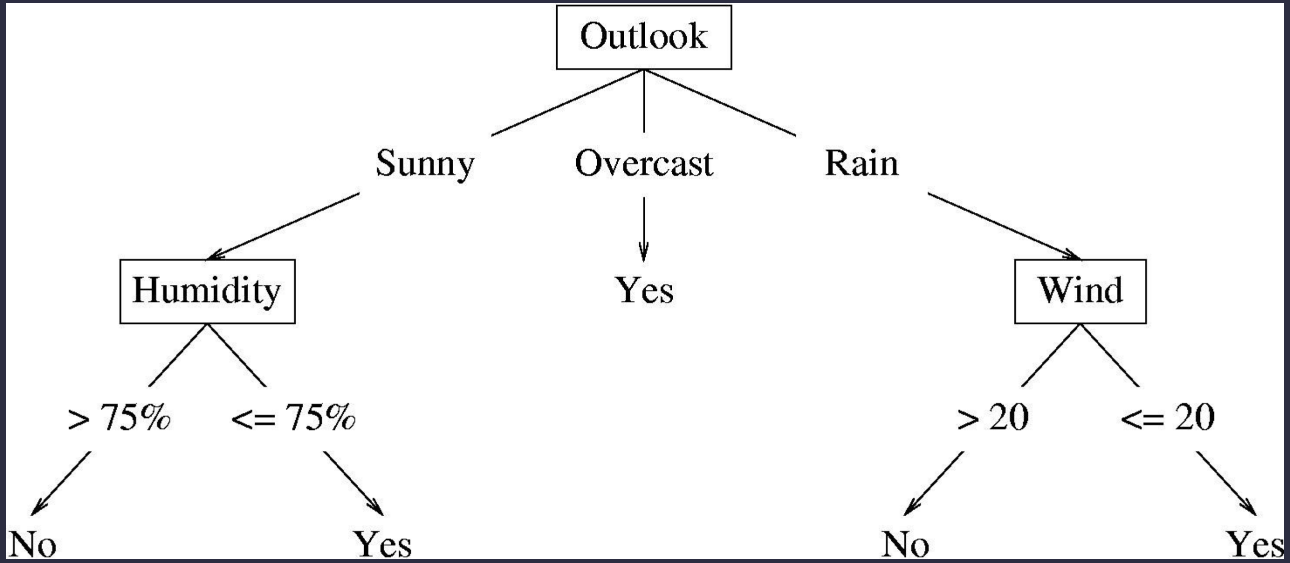
- A possible decision tree for the data:



- What prediction would we make for
  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Decision Tree

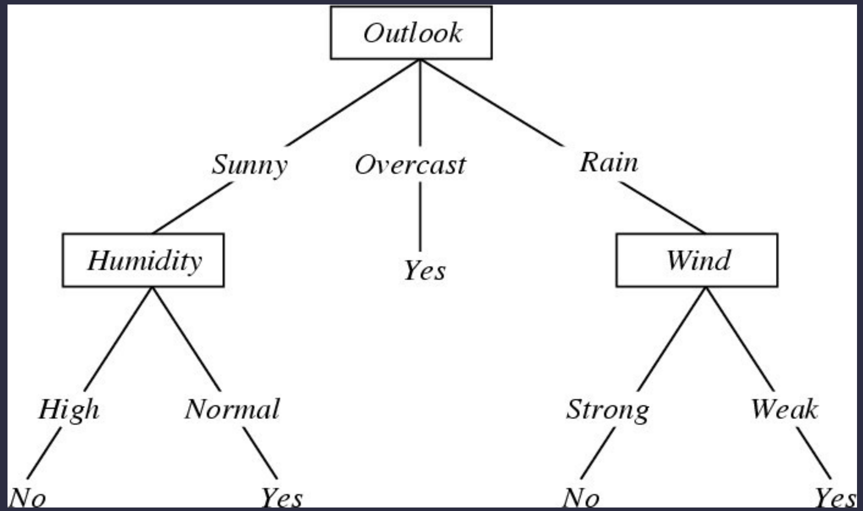
- If features are continuous, internal nodes can test the value of a feature against a threshold



Decision Tree Learning

Problem Setting:

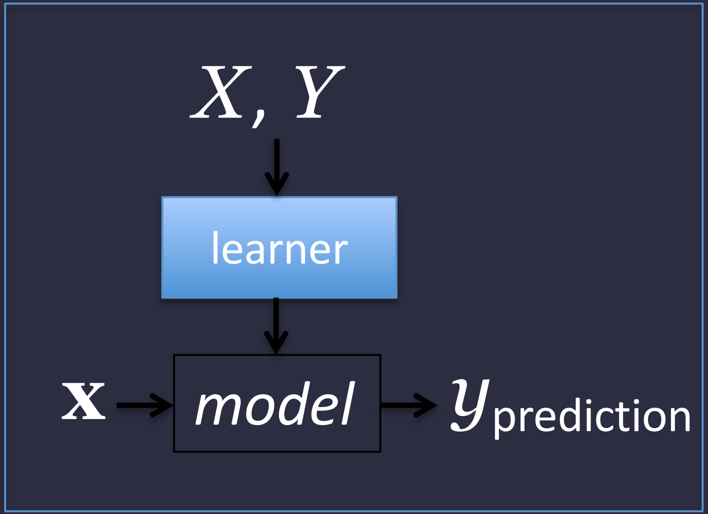
- Set of possible instances X
  - each instance x in X is a feature vector
  - e.g., <*Humidity=low, Wind=weak, Outlook=rain, Temp=hot*>
- Unknown target function $f : X \rightarrow Y$
  - *Y* is discrete valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree
  - trees sorts x to leaf, which assigns y



Stages of (Batch) Machine Learning

Given: labeled training data $X, Y = \{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n}$

- Assumes each $\boldsymbol{x}_i \sim D(X)$ with $y_i = f_{target}(\boldsymbol{x}_i)$



Train the model:

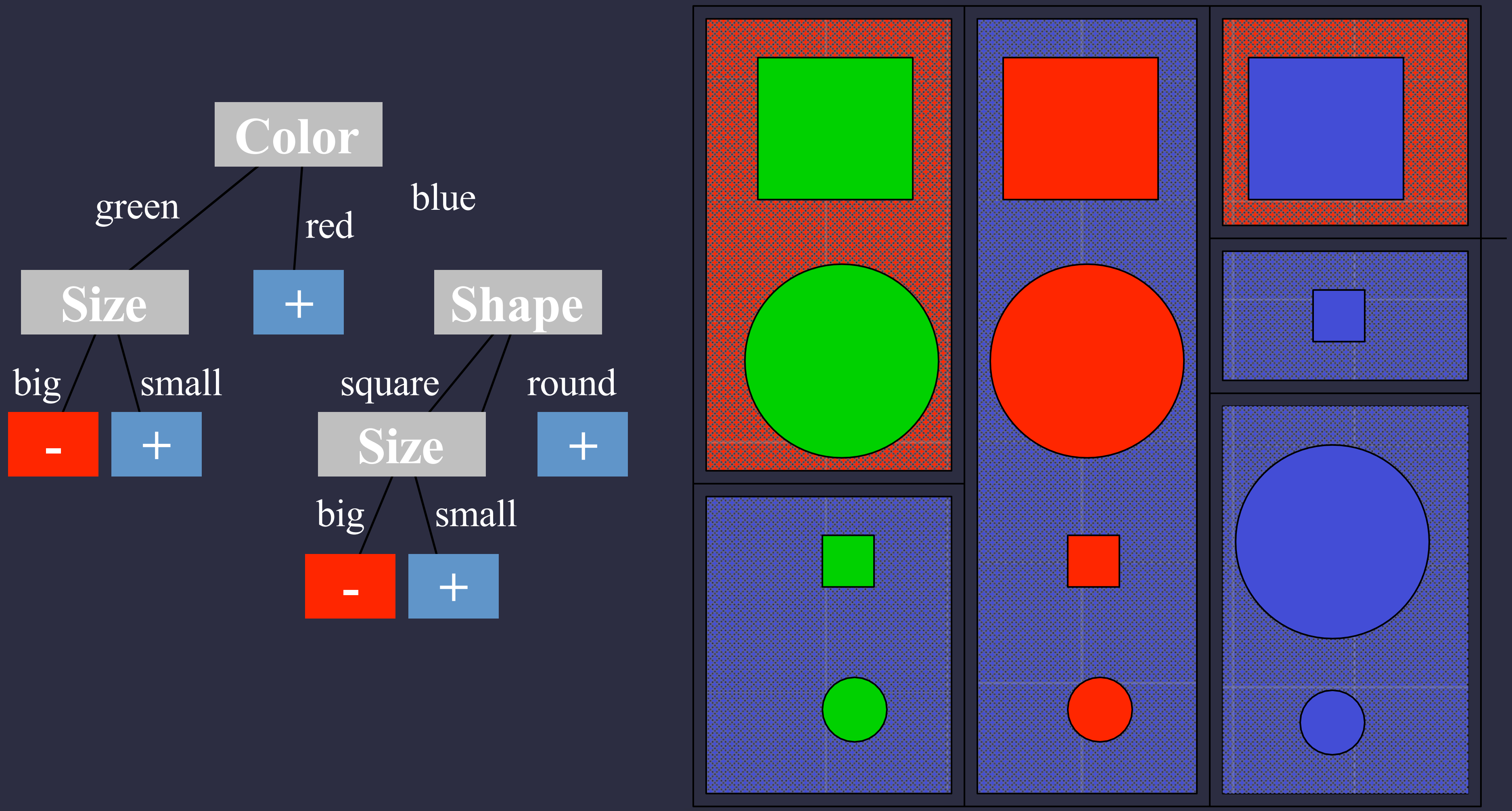
model ← *classifier*.train(X, Y)

Apply the model to new data:

- Given: new unlabeled instance $\boldsymbol{x} \sim D(X)$

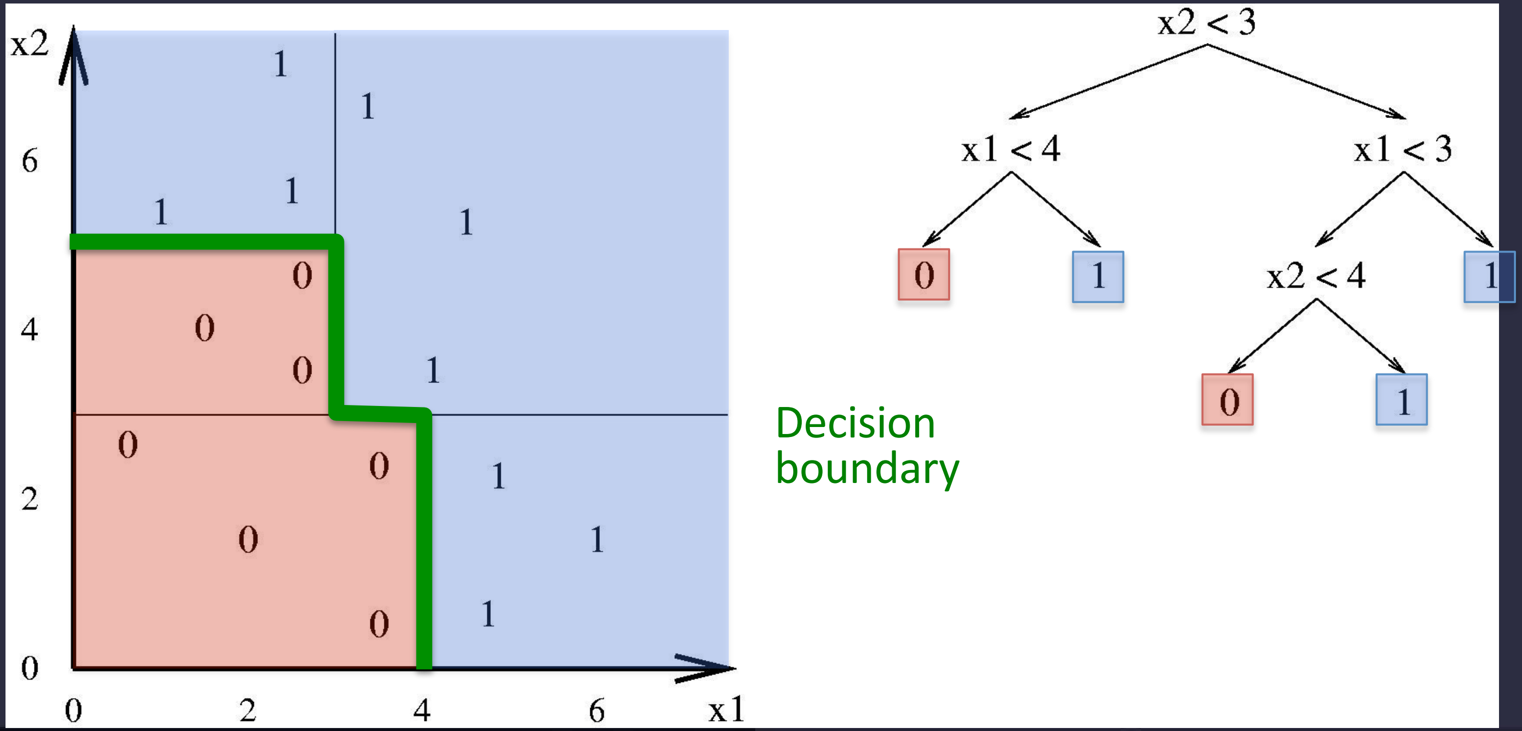
$y_{\text{prediction}}$ ← *model*.predict($\mathbf{x}$)

Decision Tree Induced Partition



Decision Tree – Decision Boundary

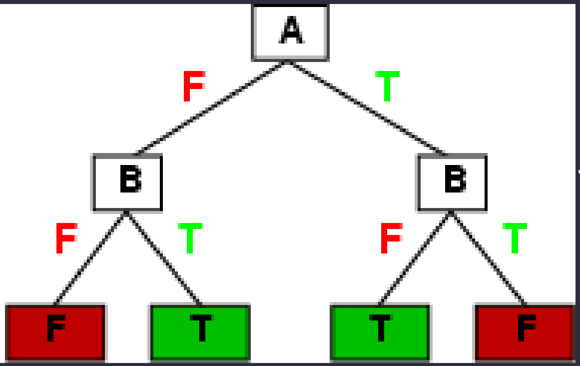
- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - or a probability distribution over labels



Expressiveness

- Decision trees can represent any boolean function of the input attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



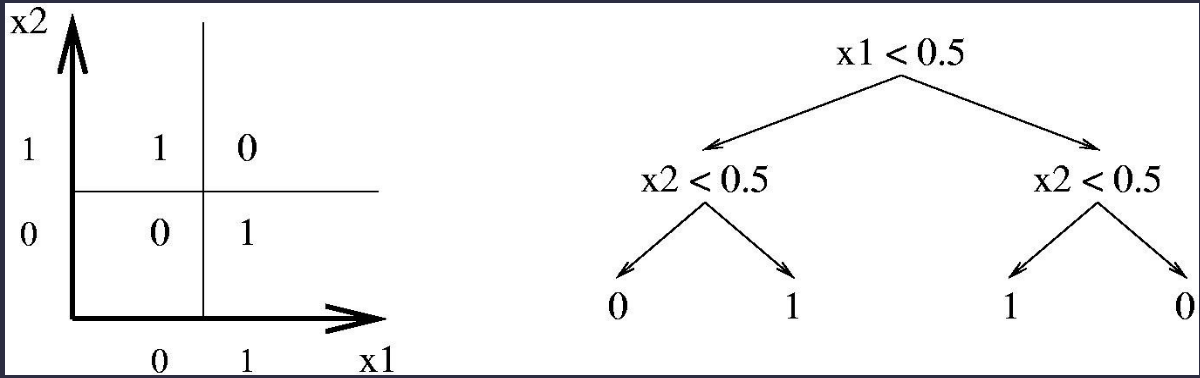
Truth table row → path to leaf

- In the worst case, the tree will require exponentially many nodes

Expressiveness

Decision trees have a variable-sized hypothesis space

- As the #nodes (or depth) increases, the hypothesis space grows
  - Depth 1 (“decision stump”): can represent any boolean function of one feature
  - Depth 2: any boolean fn of two features; some involving three features (e.g., $(x_1\ A\ x_2)\ V\ (\neg x_1\ A\ \neg x_3)$)
  - etc.



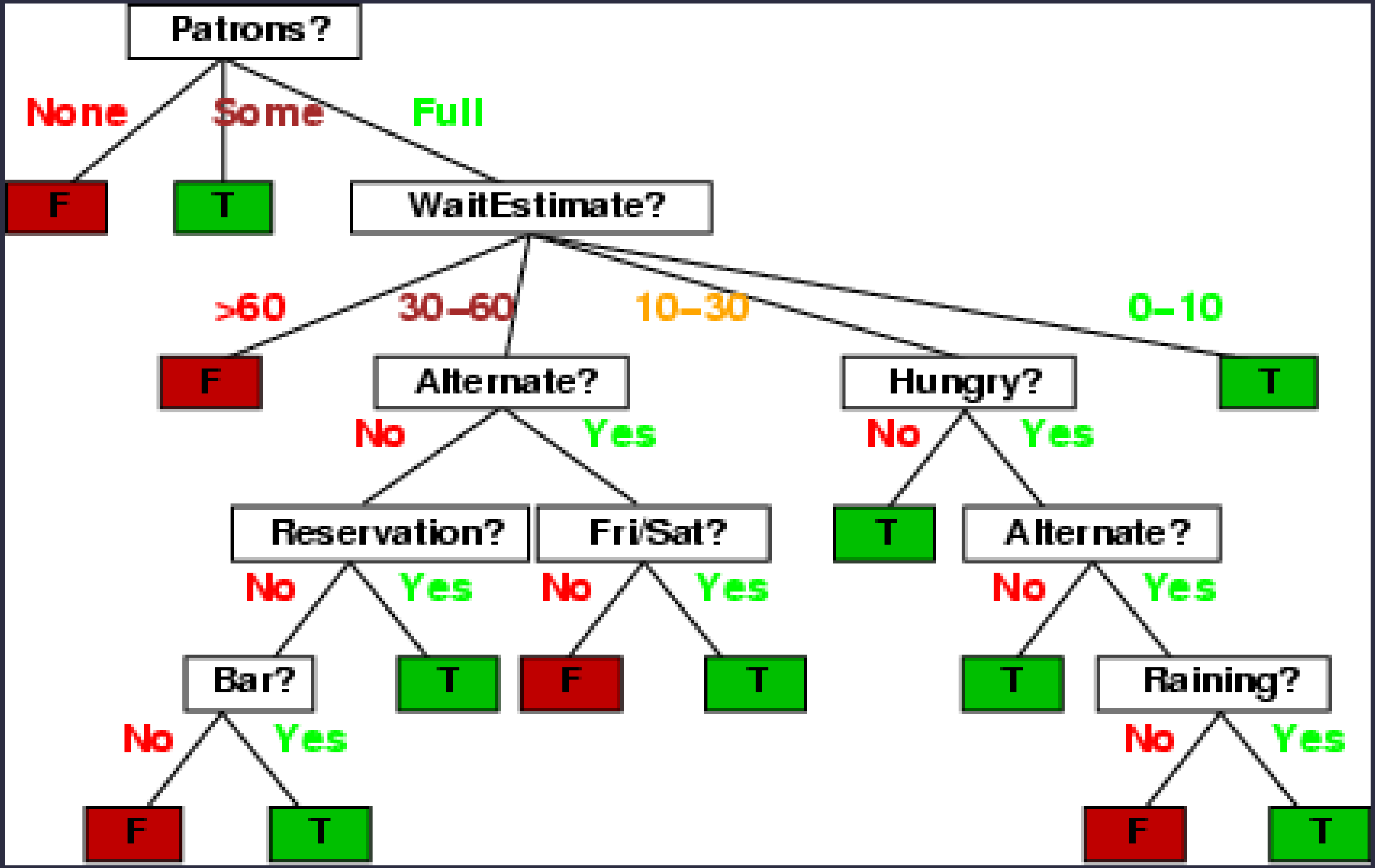
Another Example: Restaurant Domain

Model a patron's decision of whether to wait for a table at a restaurant

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

~7,000 possible cases

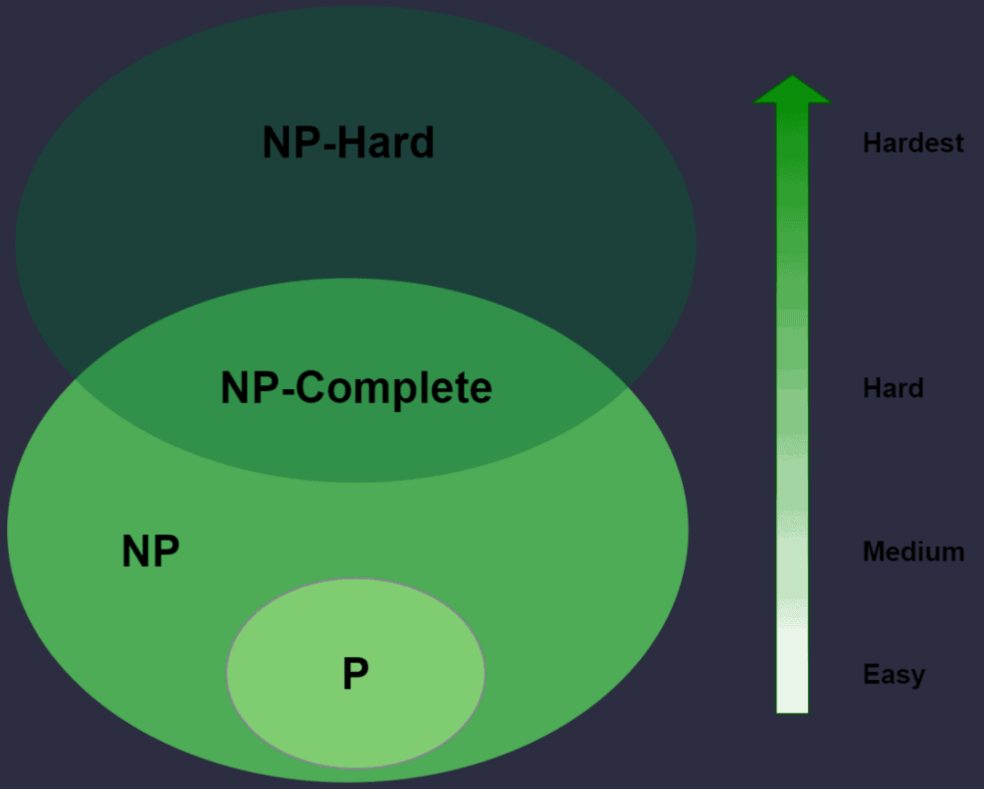
A Decision Tree from Introspection



Is this the best decision tree?

Decision Tree

- The smallest decision tree that correctly classifies all of the training examples is best
  - Finding the provably smallest decision tree is NP-hard
  - ...So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small



Basic Algorithm for Top-Down Induction of Decision Trees

node = root of decision tree

Main loop:

1. *A* ← the "best" decision attribute for the next node.
2. Assign *A* as decision attribute for *node*.
3. For each value of *A*, create a new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop.
   Else, recurse over new leaf nodes.

How do we choose which attribute is best?

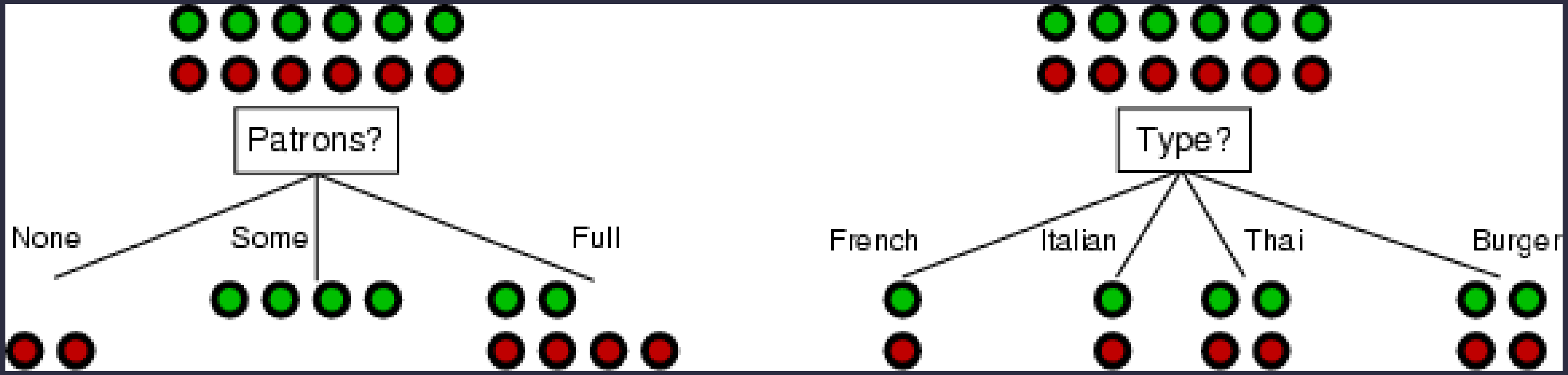
Choosing the Best Attribute

Key problem: choosing which attribute to split a given set of examples

- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the largest expected *information gain*
    - i.e., attribute that results in smallest expected size of subtrees rooted at its children

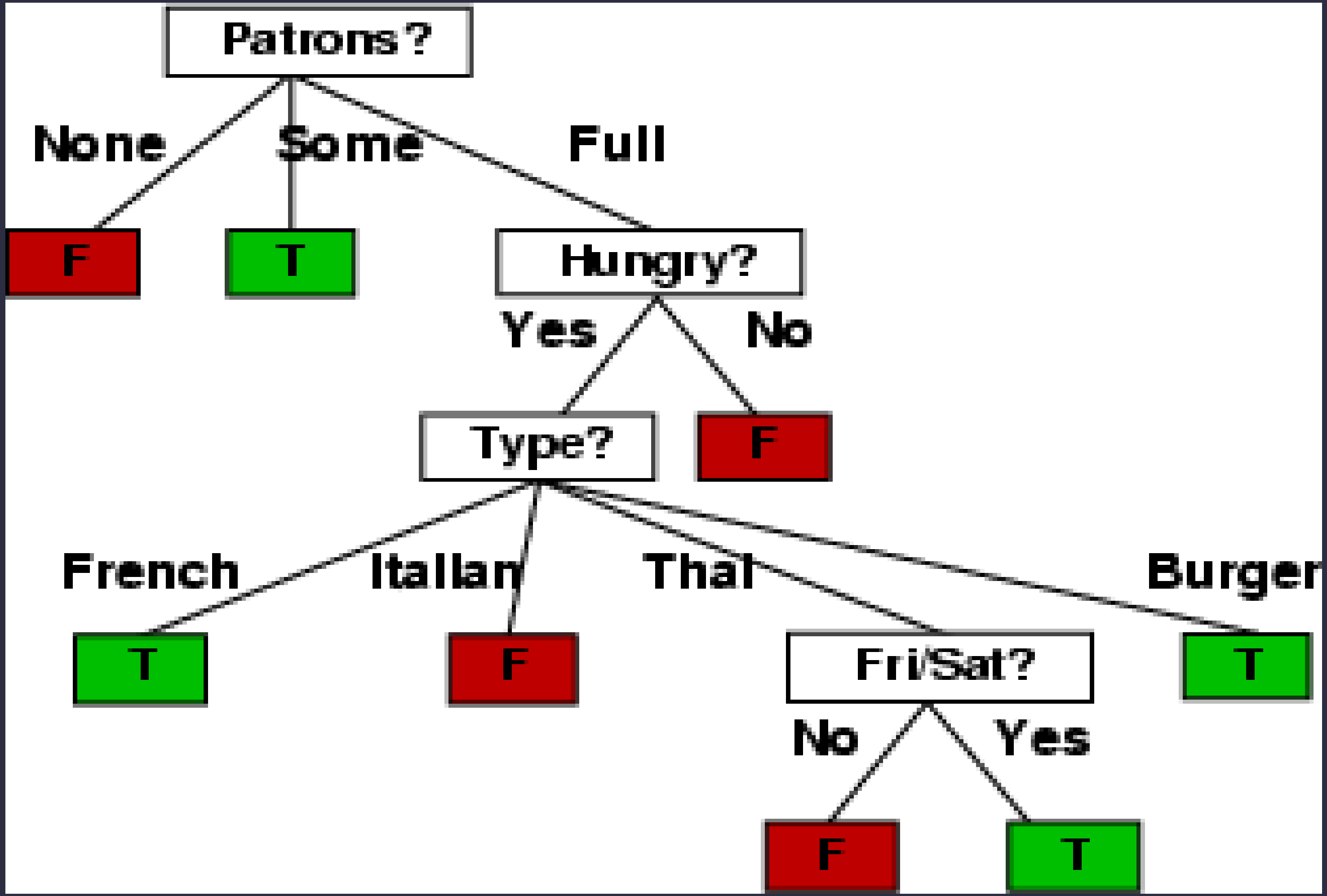
Choosing an Attribute

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

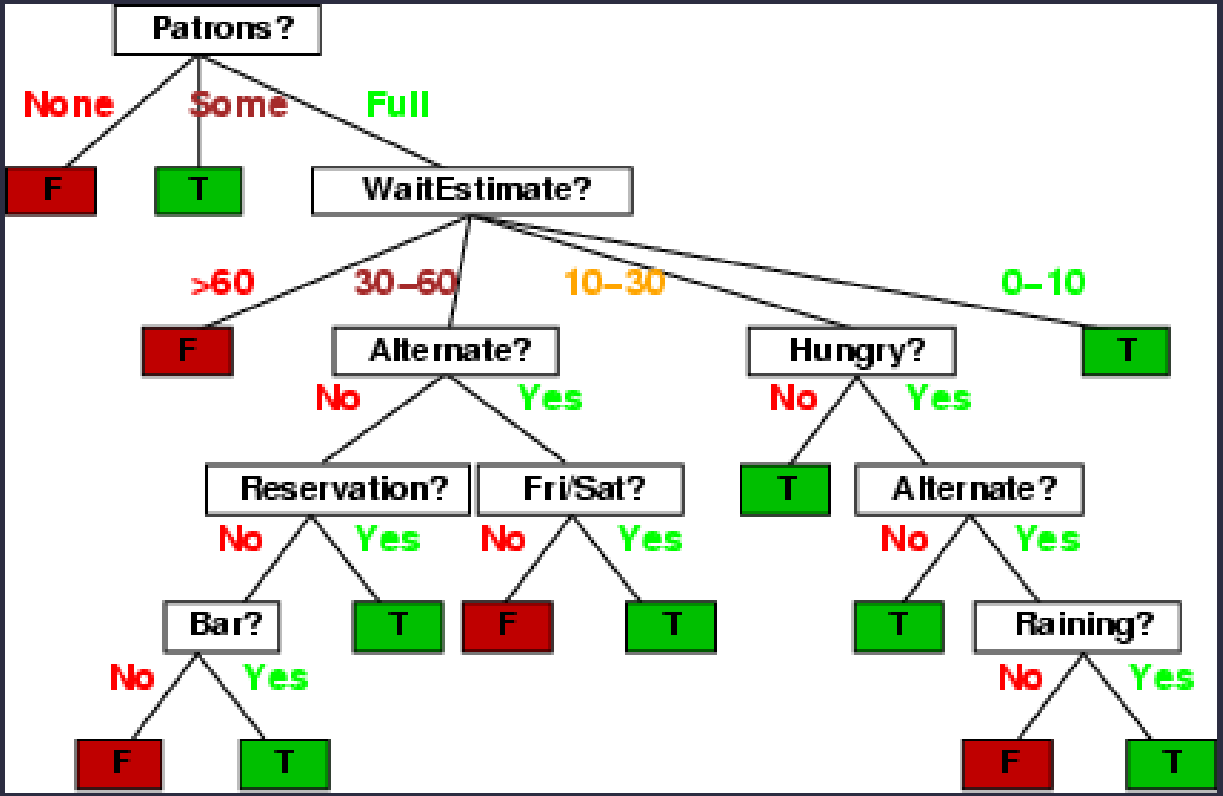


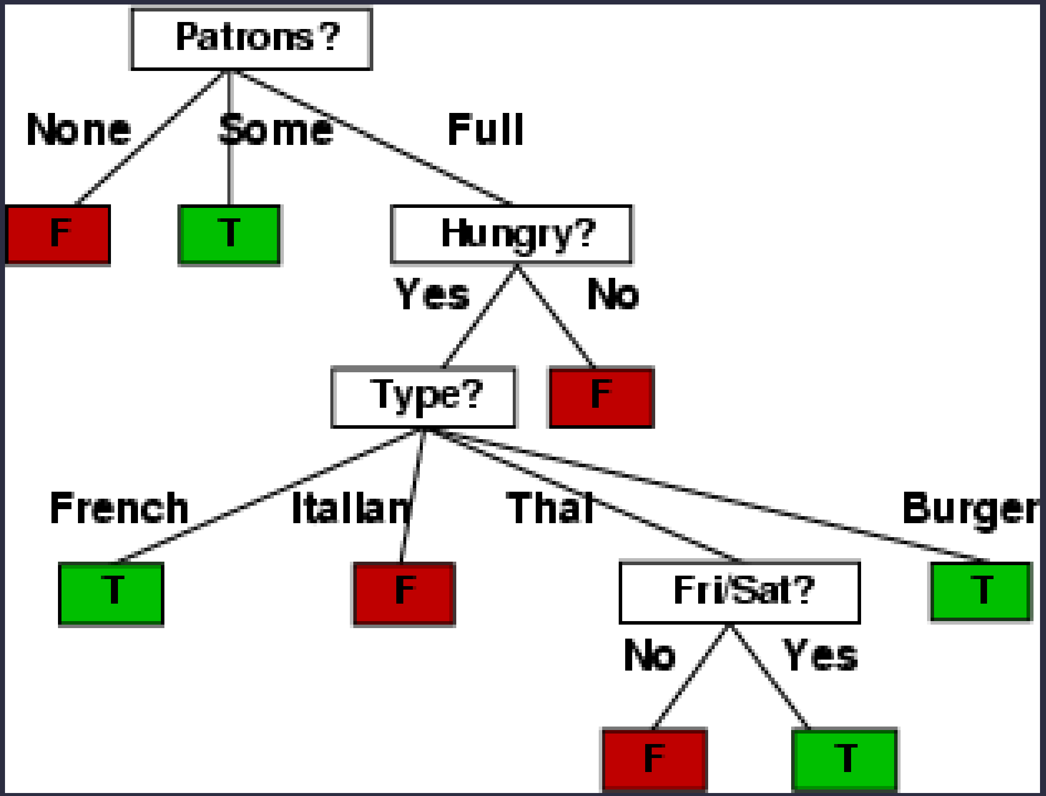
Which split is more informative: *Patrons?* or *Type?*

ID3-induced Decision Tree



Compare the Two Decision Trees

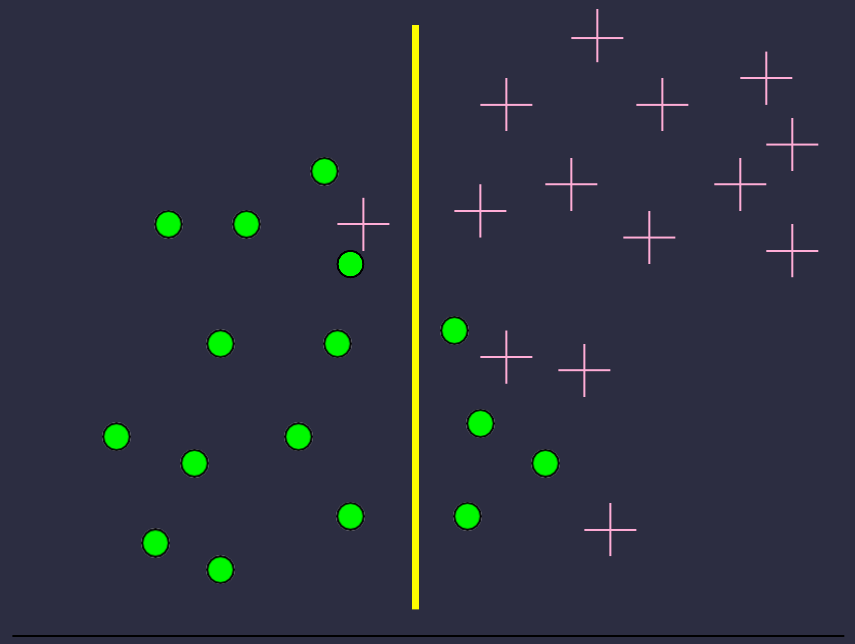




Information Gain

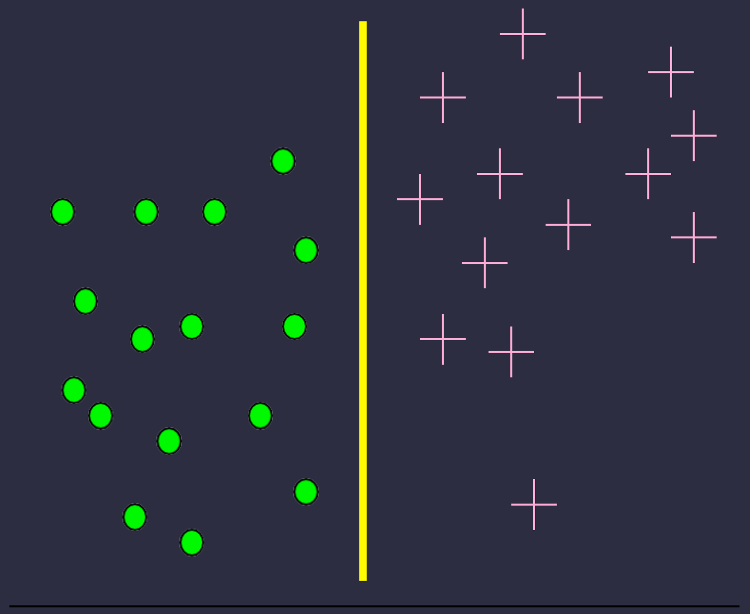
Which test is more informative?

Split over whether Balance exceeds 50K



Less or equal 50K   Over 50K

Split over whether applicant is employed



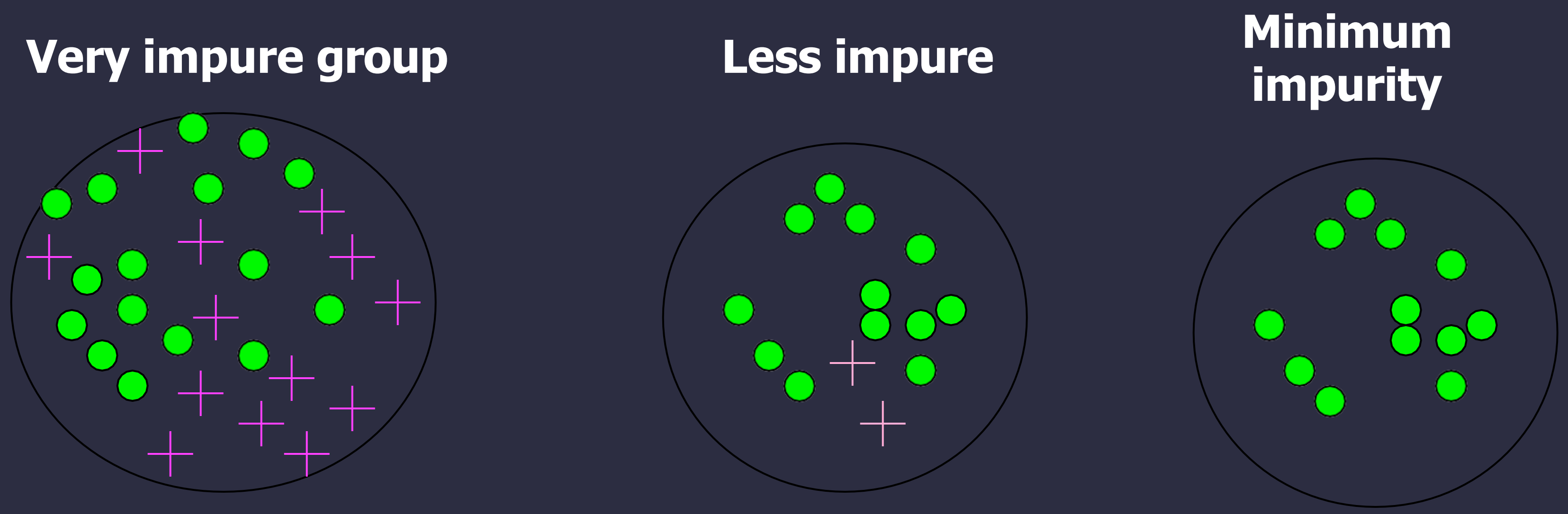
Unemployed   Employed

Information Gain

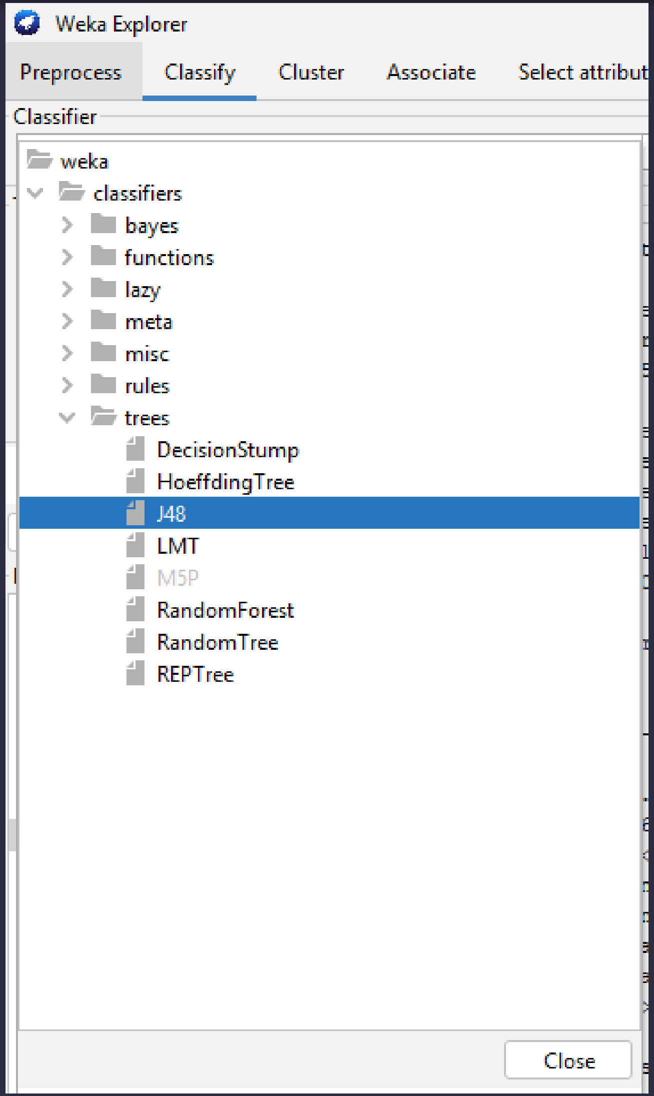
Impurity/Entropy (informal)

– Measures the level of **impurity** in a group of examples

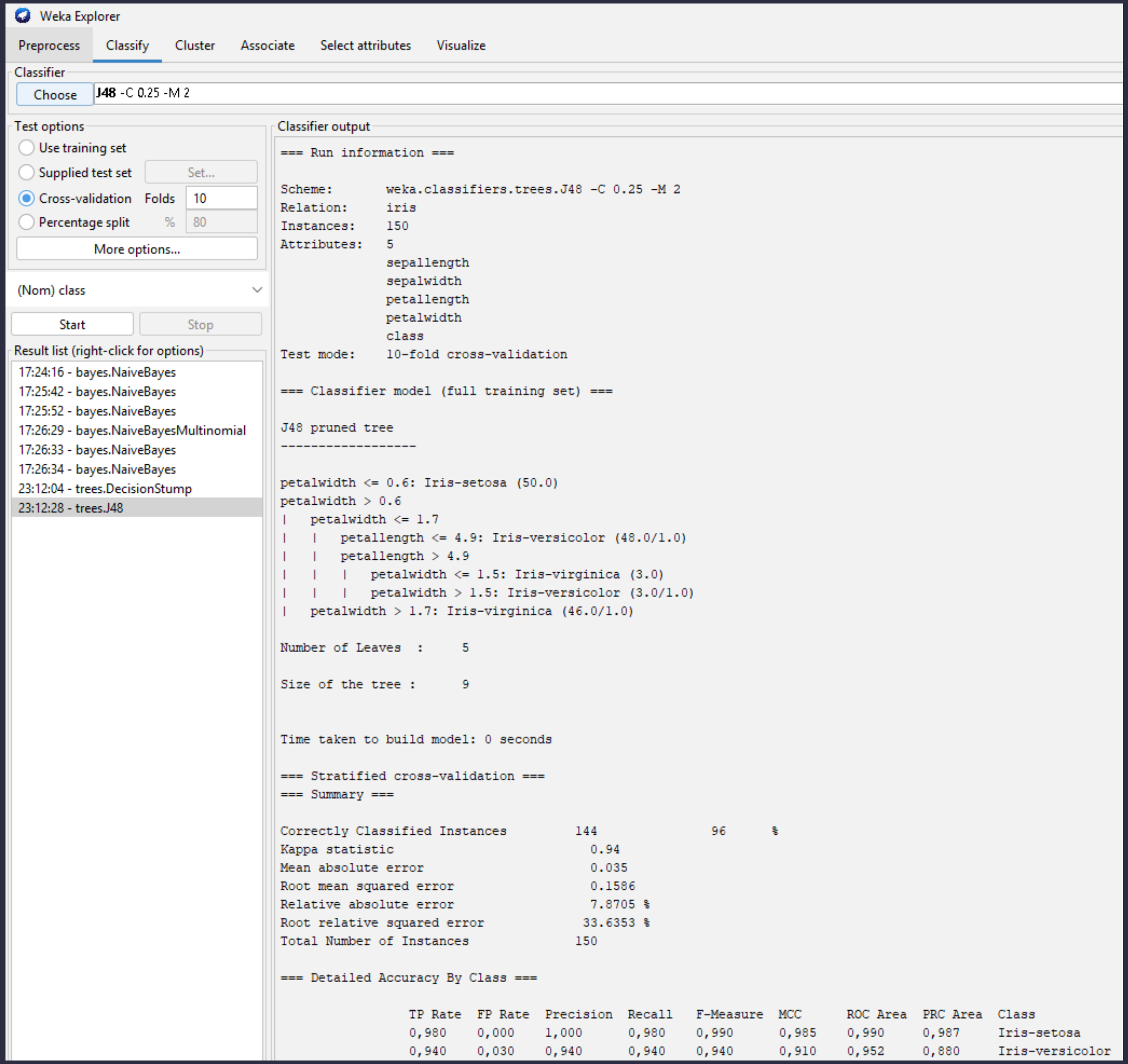
Impurity



Select Decision Tree Based Algorithms



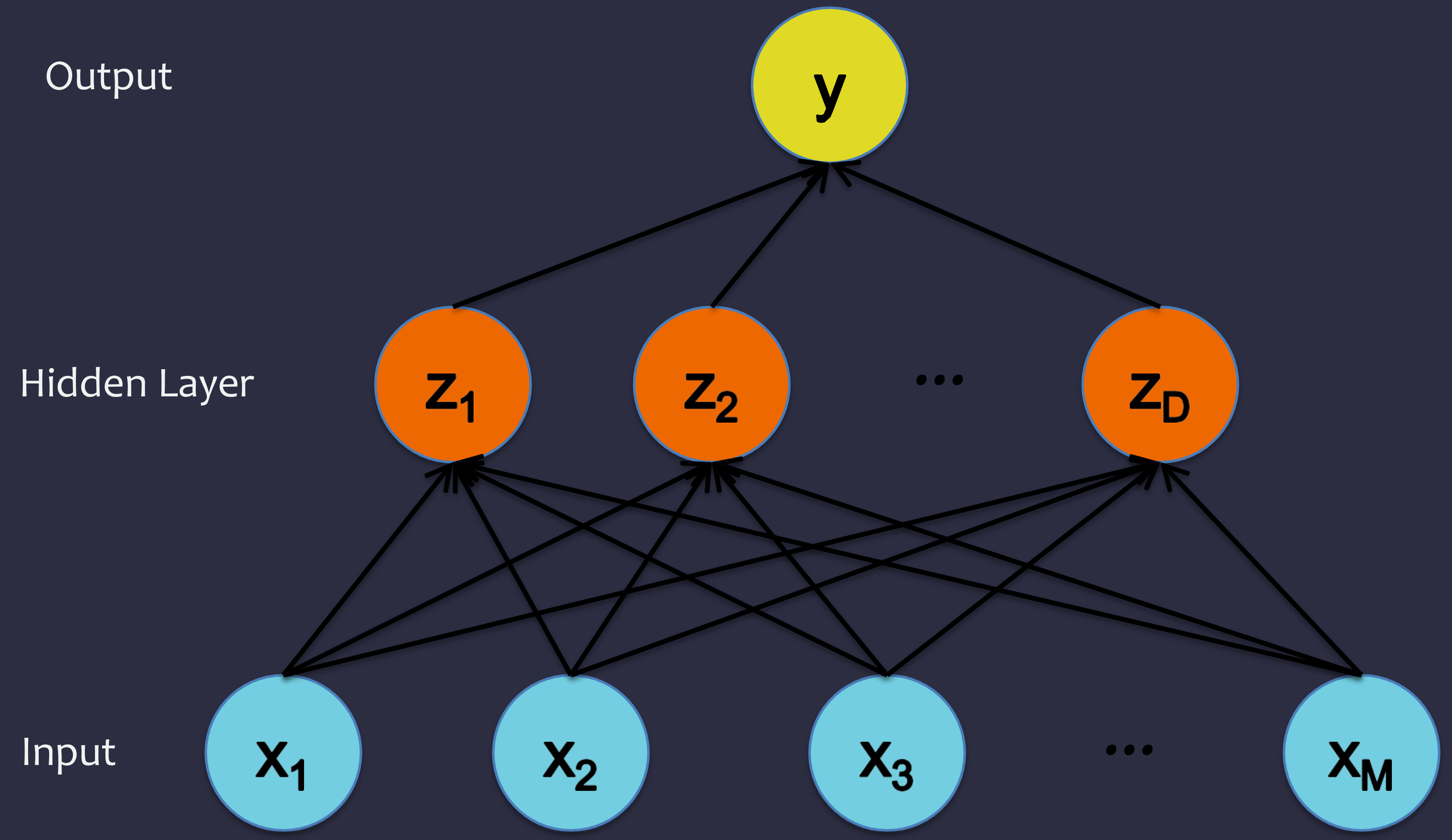
Select Decision Tree Based Algorithms



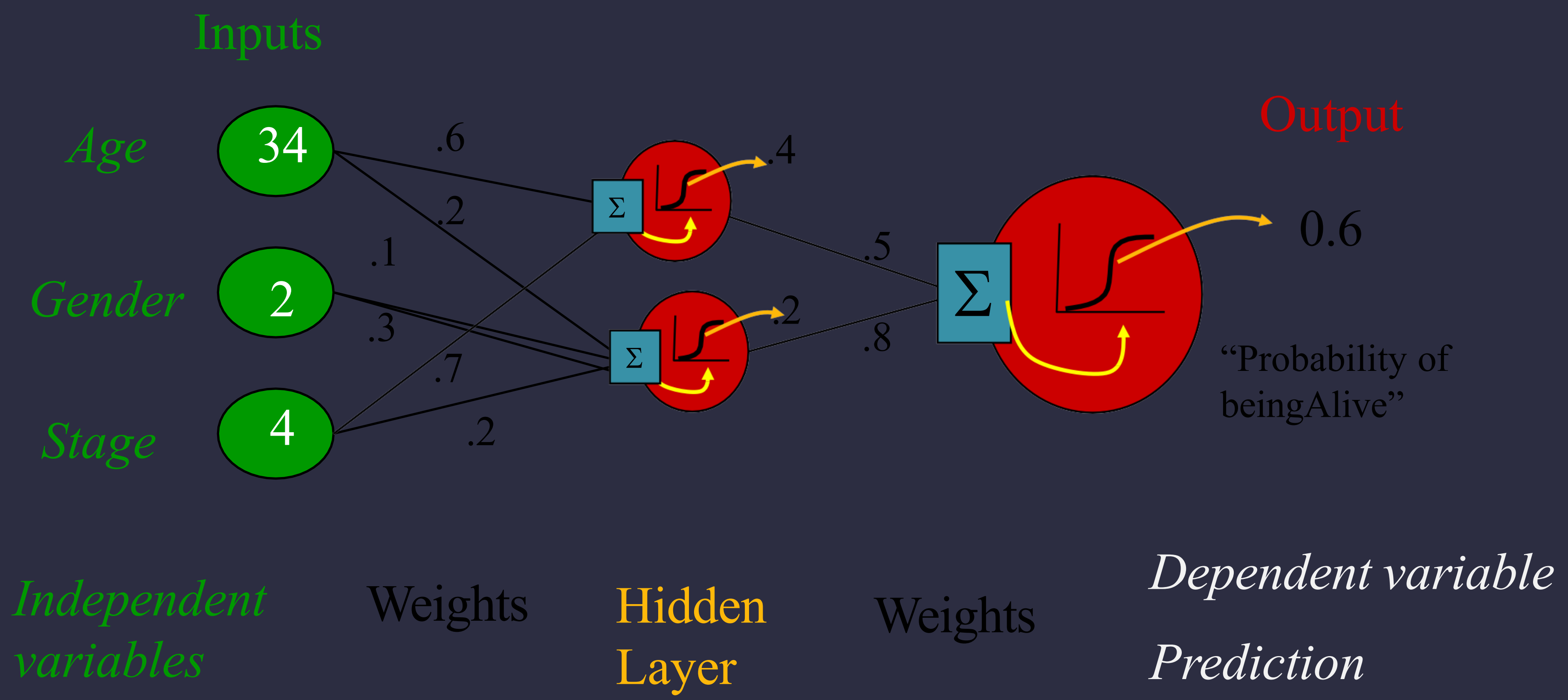
Neural Networks

- Analogy to Biological Systems (Indeed a great example of a good learning system)
- Massive Parallelism allowing for computational efficiency
- The first learning algorithm came in 1959 (Rosenblatt) who suggested that if a target output value is provided for a single neuron with fixed inputs, one can incrementally change weights to learn to produce these outputs using the perceptron learning rule

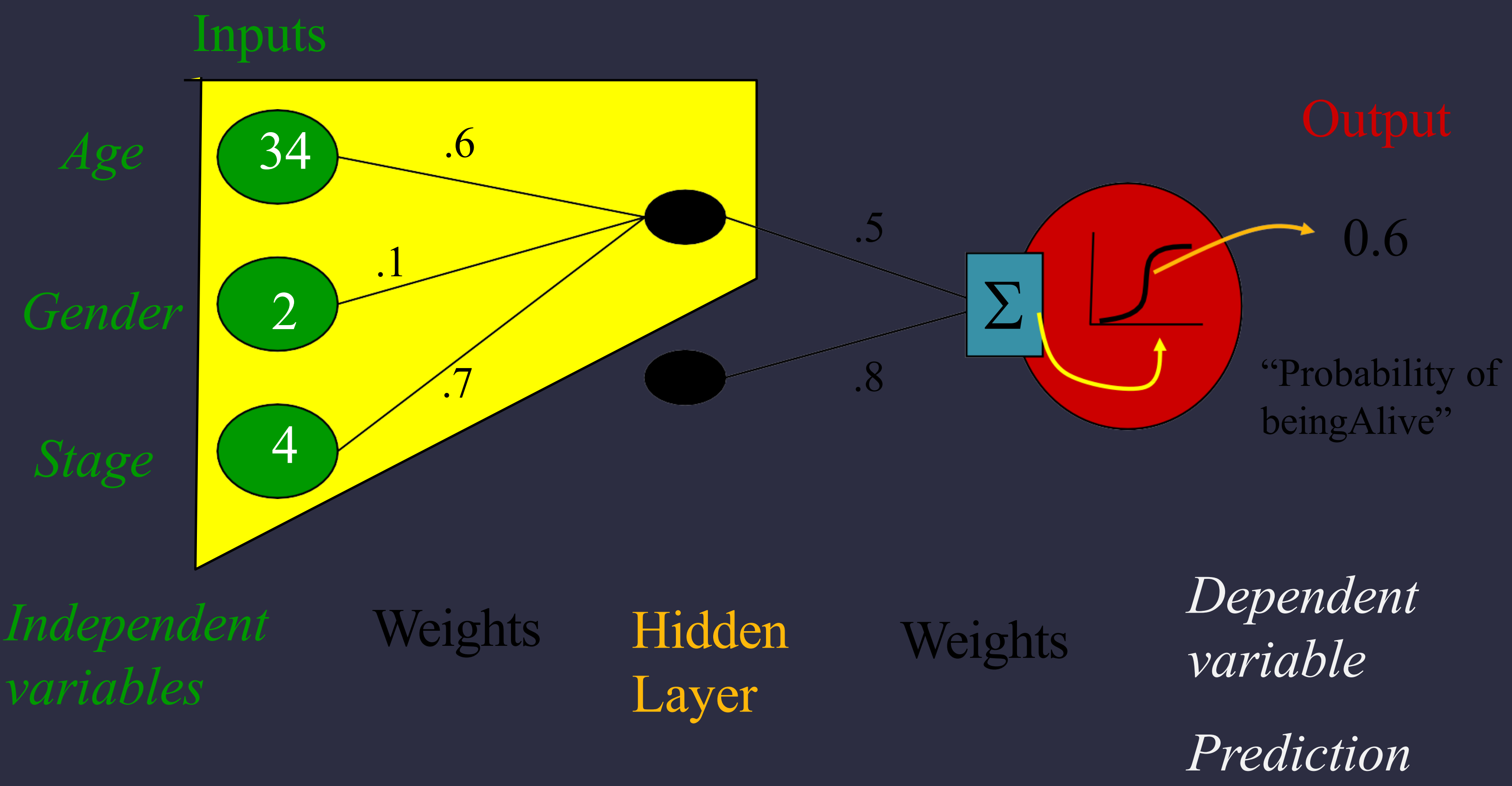
Neural Network

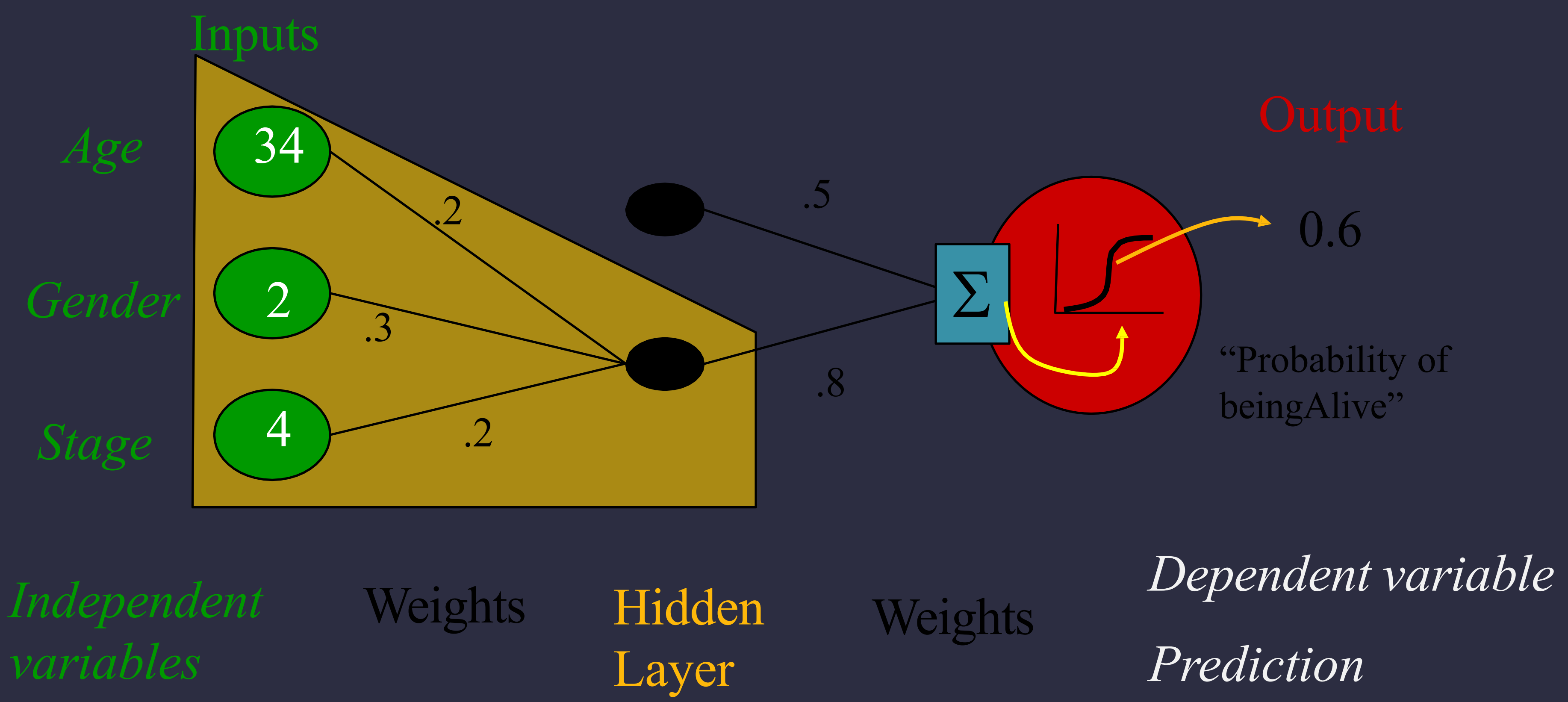


Neural Network Model

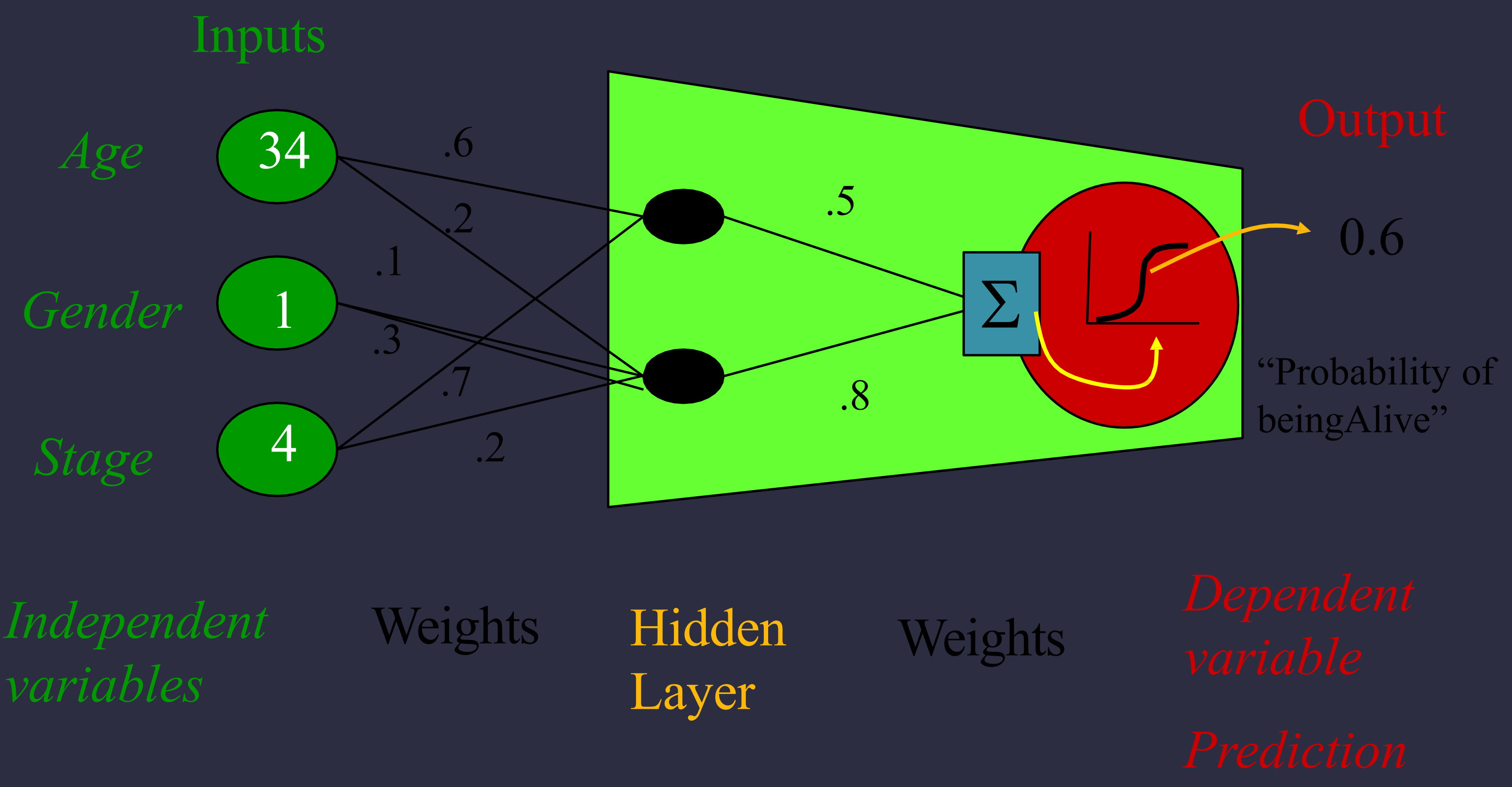


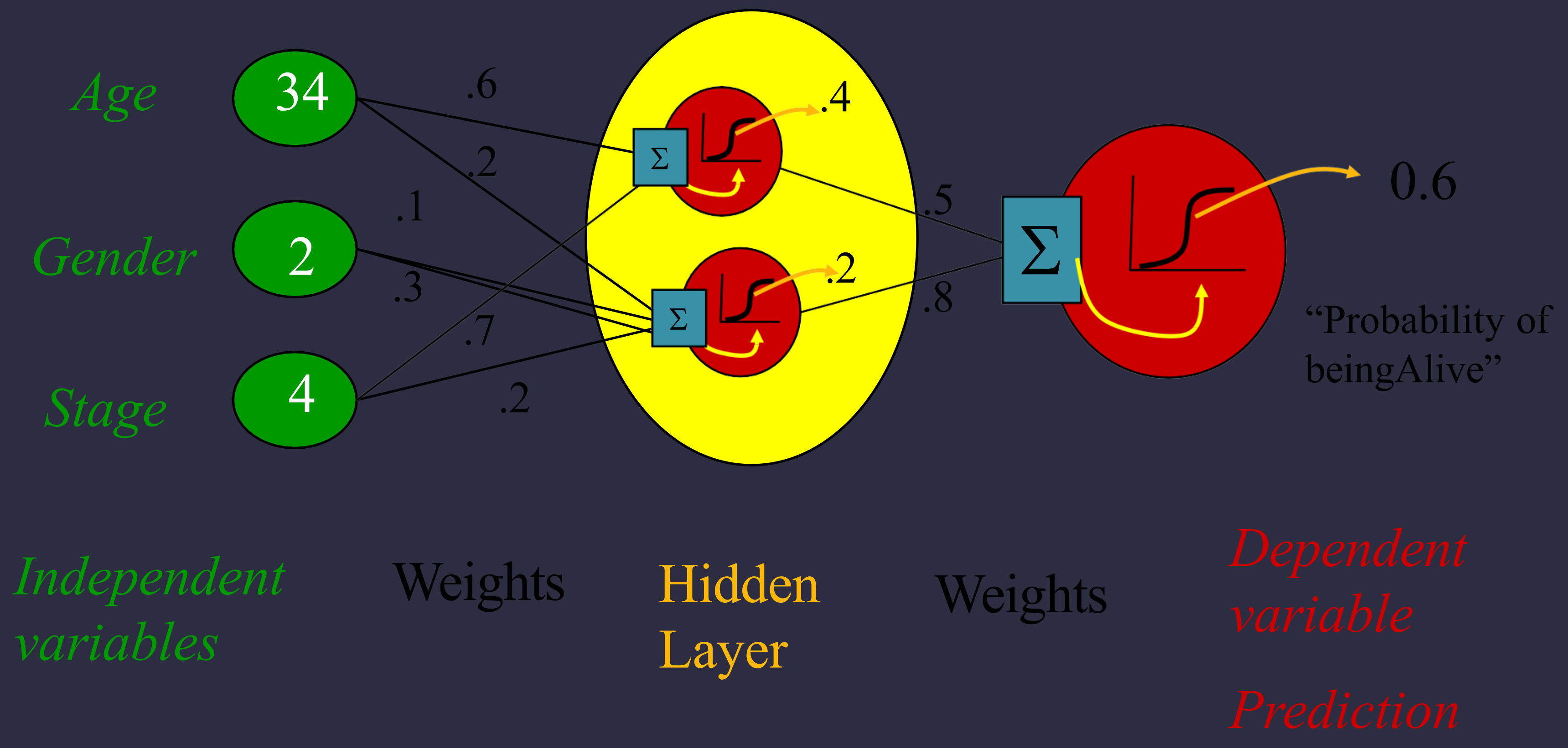
“Combined logistic models”





Inputs
Age
Gender
Stage
34
2
4
.2
.3
.2
.5
.8
Σ
Output
0.6
"Probability of beingAlive"
Independent variables
Weights
Hidden Layer
Weights
Dependent variable
Prediction


Inputs
Age
34
Gender
1
Stage
4
.6
.2
.1
.3
.7
.2
.5
.8
Σ
Output
0.6
"Probability of beingAlive"
Independent variables
Weights
Hidden Layer
Weights
Dependent variable
Prediction



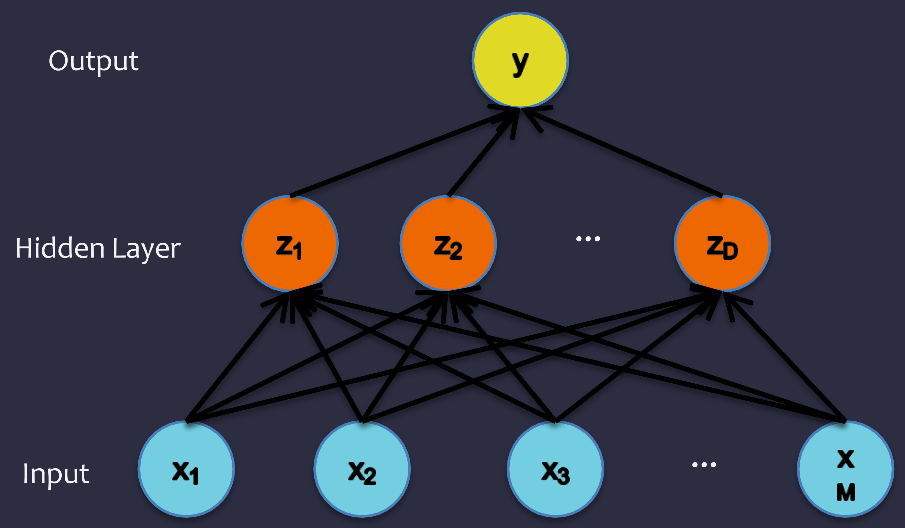
Age
Gender
Stage
34
2
4
.6
.2
.1
.3
.7
.2
Σ
Σ
.4
.2
.5
.8
Σ
0.6
"Probability of beingAlive"
Independent variables
Weights
Hidden Layer
Weights
Dependent variable
Prediction

Neural Networks

– Example: Neural Network w/1 Hidden Layer
– Example: Neural Network w/2 Hidden Layers
– Example: Feed Forward Neural Network

Neural Network

Neural Network for Classification



(E) **Output (sigmoid)**

$$y = \frac{1}{1+\exp(-b)}$$

(D) **Output (linear)**

$$b = \sum_{j=0}^{D} \beta_j z_j$$

(C) **Hidden (sigmoid)**

$$z_j = \frac{1}{1+\exp(-a_j)}, \quad \forall j$$

(B) **Hidden (linear)**

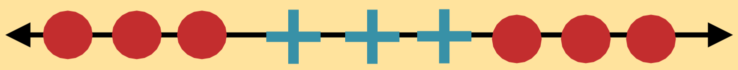
$$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \quad \forall j$$

(A) **Input**

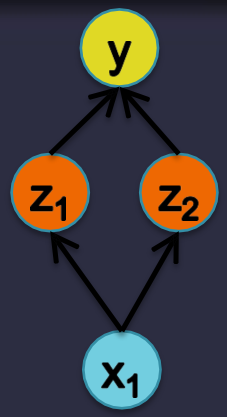
Given $x_i, \quad \forall i$

Neural Network Parameters

- Question:
- Suppose you are training a one-hidden layer neural network with sigmoid activations for binary classification.



True or False: There is a unique set of parameters that maximize the likelihood of the dataset above.



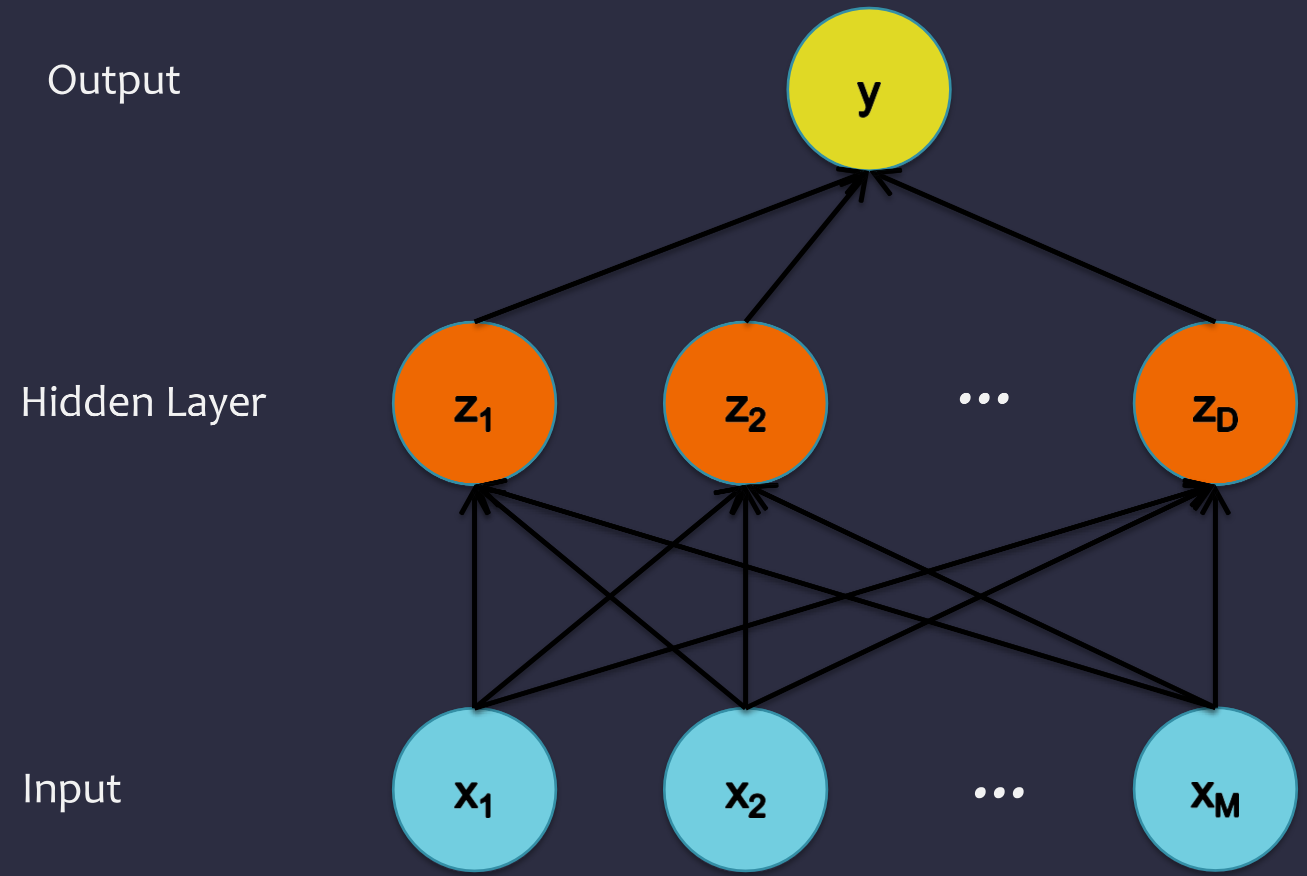
Neural Network Architectures

Even for a basic Neural Network, there are many design decisions to make:

1. # of hidden layers (depth)
2. # of units per hidden layer (width)
3. Type of activation function (nonlinearity)
4. Form of objective function

Building a Neural Net

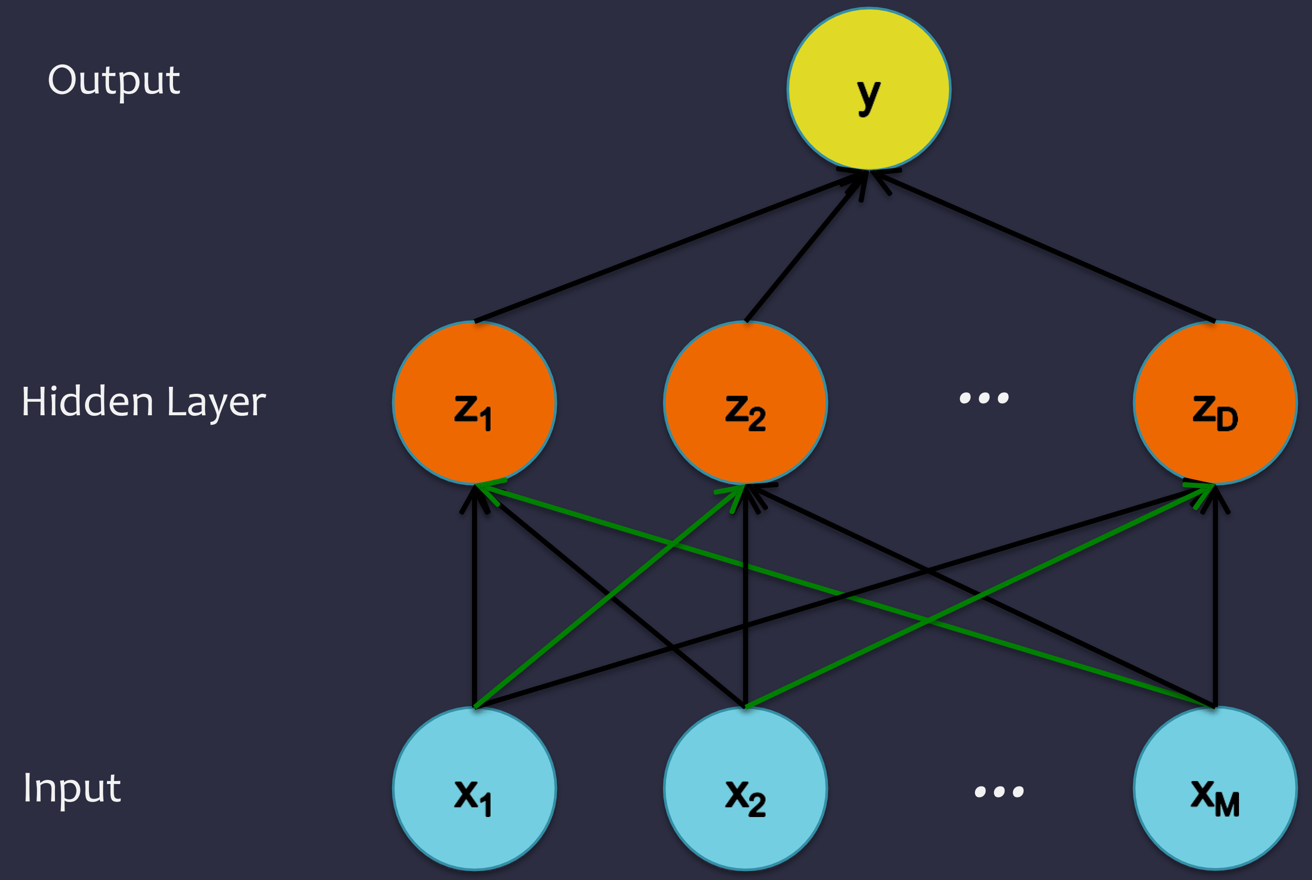
Q: How many hidden units, D, should we use?



D = M

Building a Neural Net

Q: How many hidden units, D, should we use?



D = M

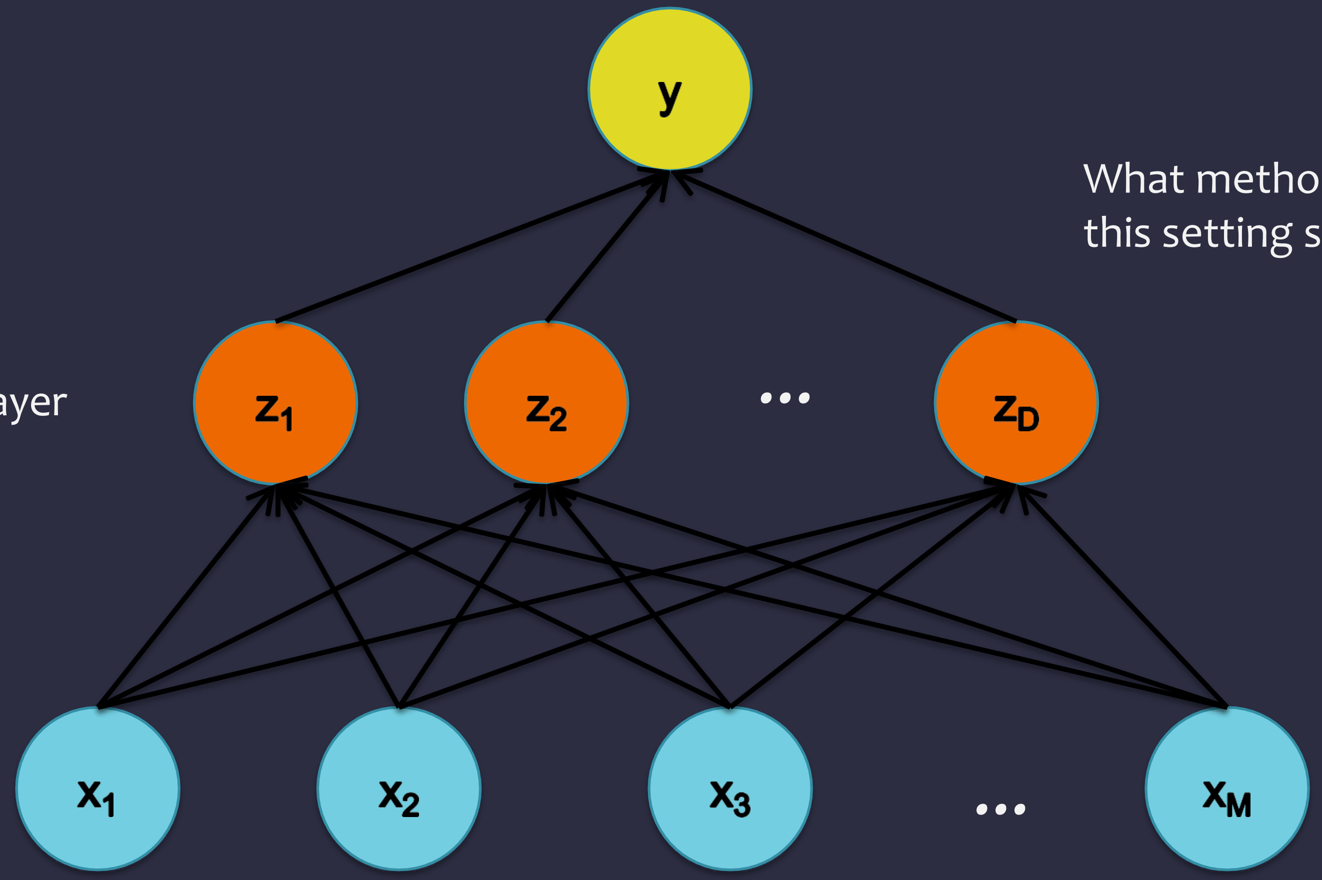
Building a Neural Net

Q: How many hidden units, D, should we use?

Output

Hidden Layer

Input

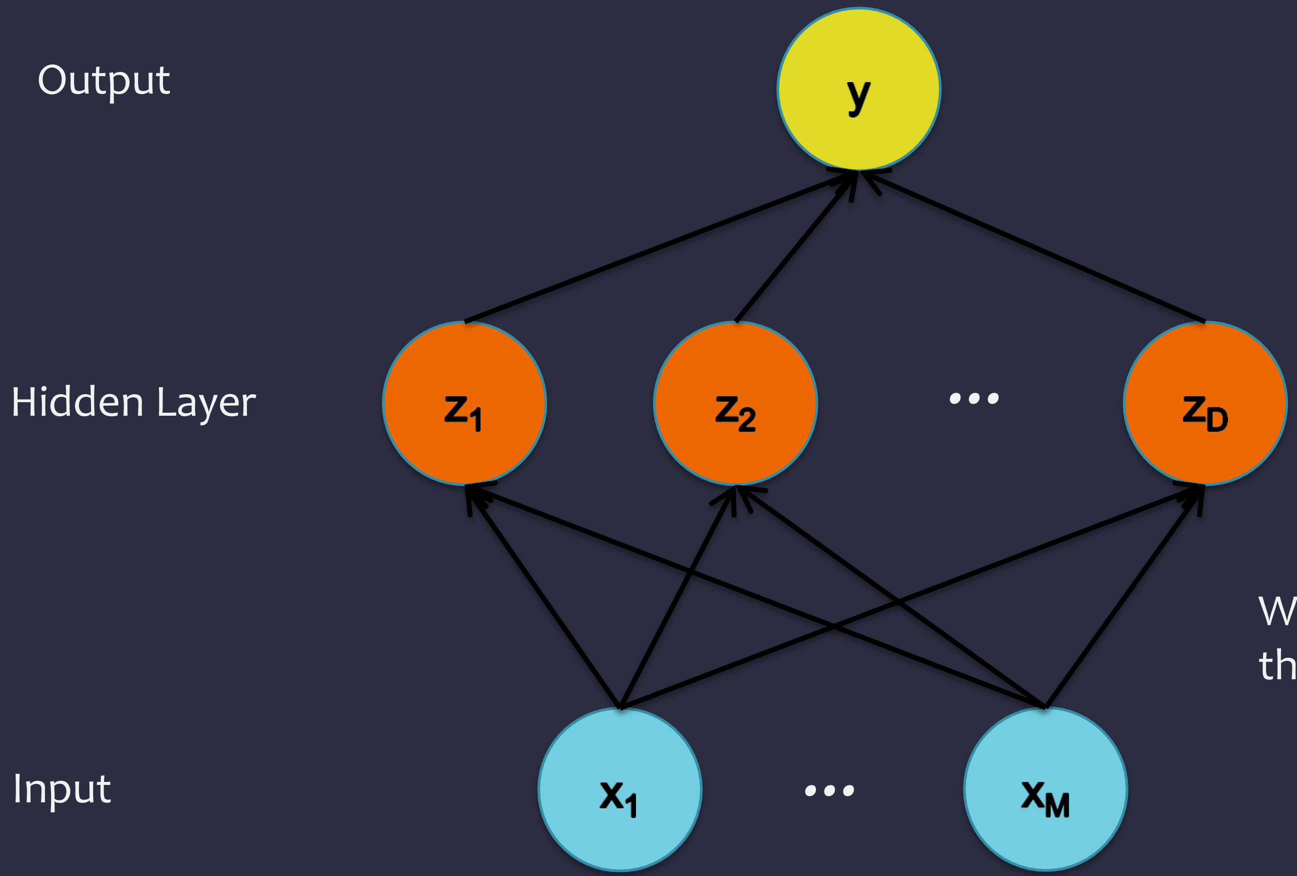


What method(s) is this setting similar to?

$D < M$

Building a Neural Net

Q: How many hidden units, D, should we use?

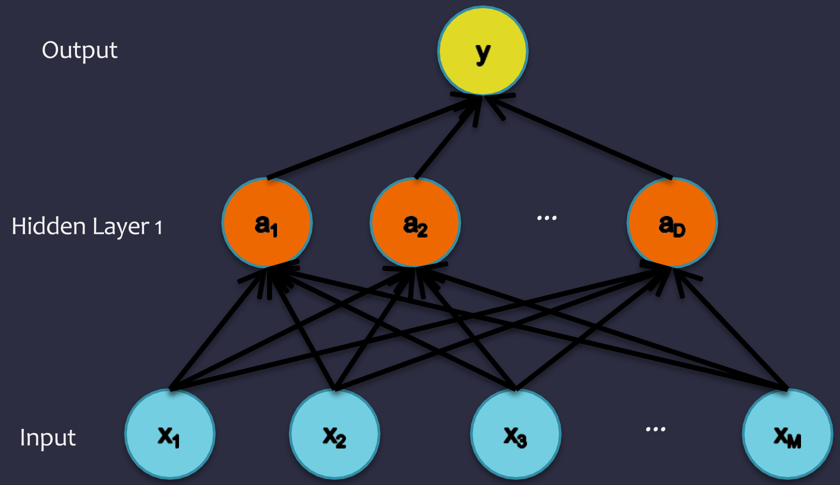


D > M

What method(s) is this setting similar to?

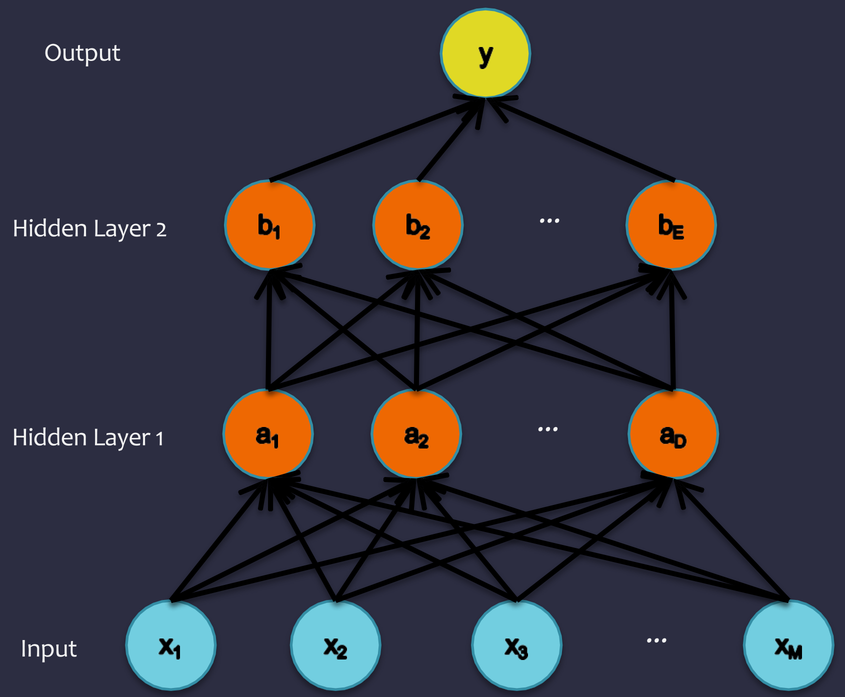
Deeper Networks

Q: How many layers should we use?



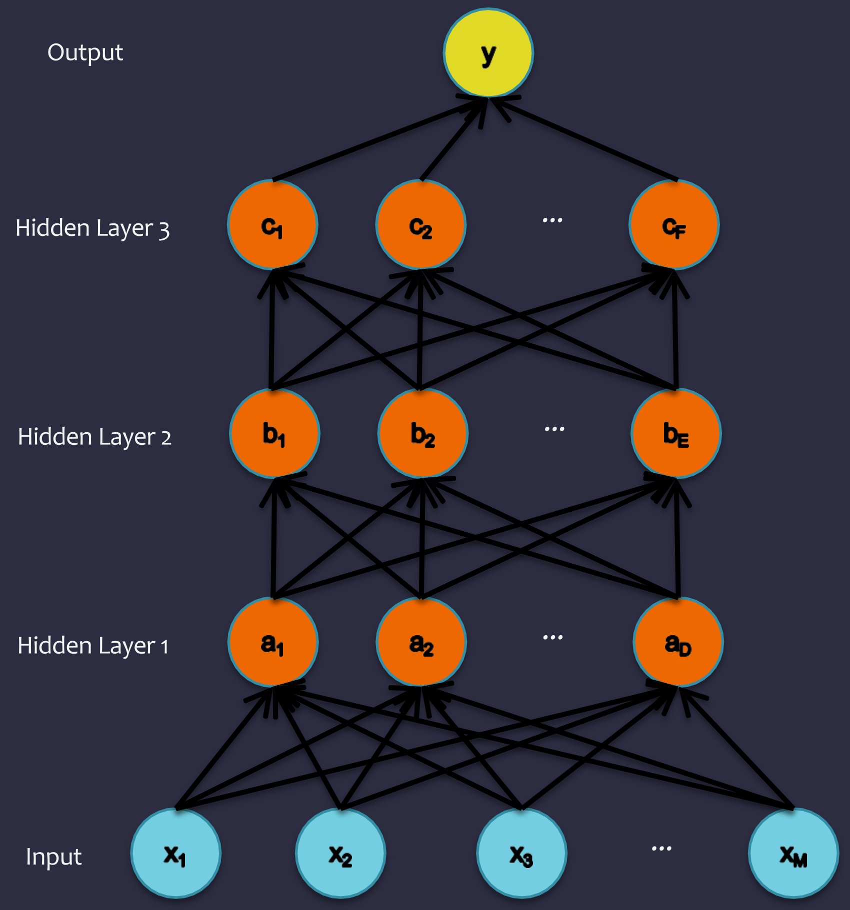
Deeper Networks

Q: How many layers should we use?



Deeper Networks

Q: How many layers should we use?



Deeper Networks

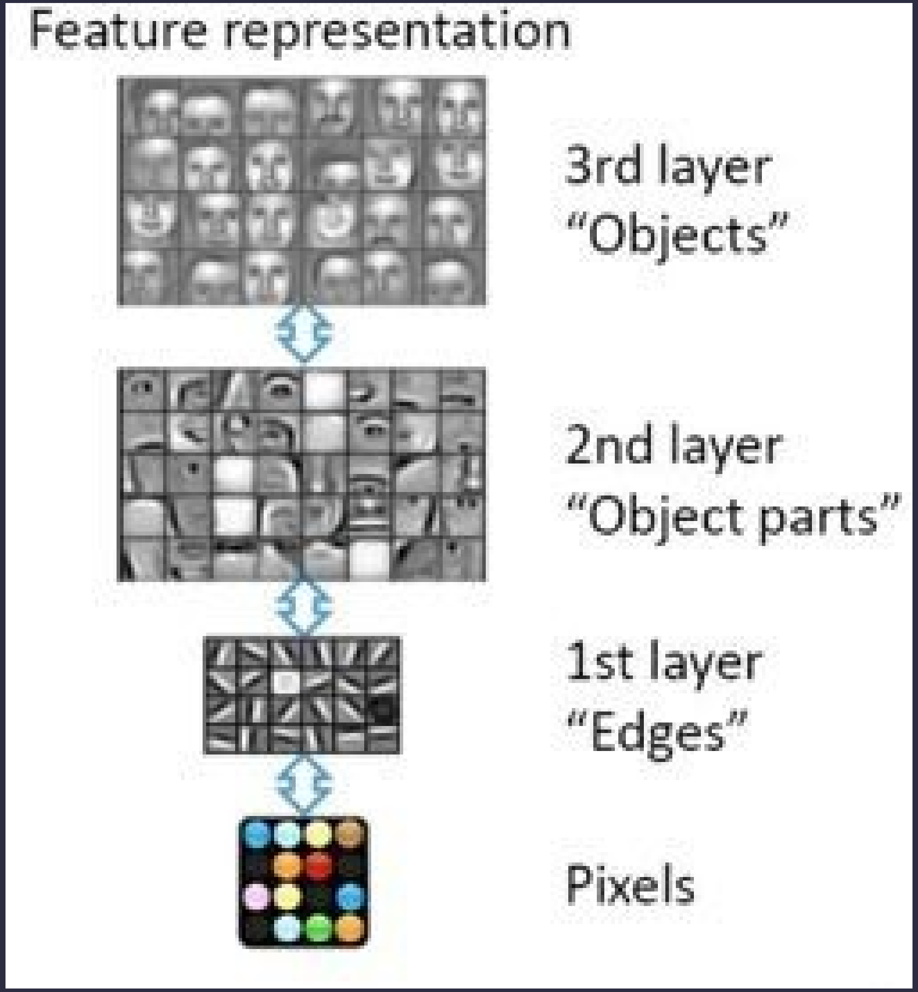
Q: How many layers should we use?

- **Theoretical answer:**
  - A neural network with 1 hidden layer is a **universal function approximator**
  - Cybenko (1989): For any continuous function g(**x**), there exists a 1-hidden-layer neural net $h_\theta(\mathbf{x})$ s.t. $| h_\theta(\mathbf{x}) - g(\mathbf{x}) | < \epsilon$ for all **x**, assuming sigmoid activation functions
- **Empirical answer:**
  - Before 2015 : "Deep networks (e.g. 3 or more hidden layers) are too hard to train"
  - After 2015: "Deep networks are easier to train than shallow networks (e.g. 2 or fewer layers) for many problems"

  Big caveat: You need to know and use the right tricks.

Different Levels of Abstraction

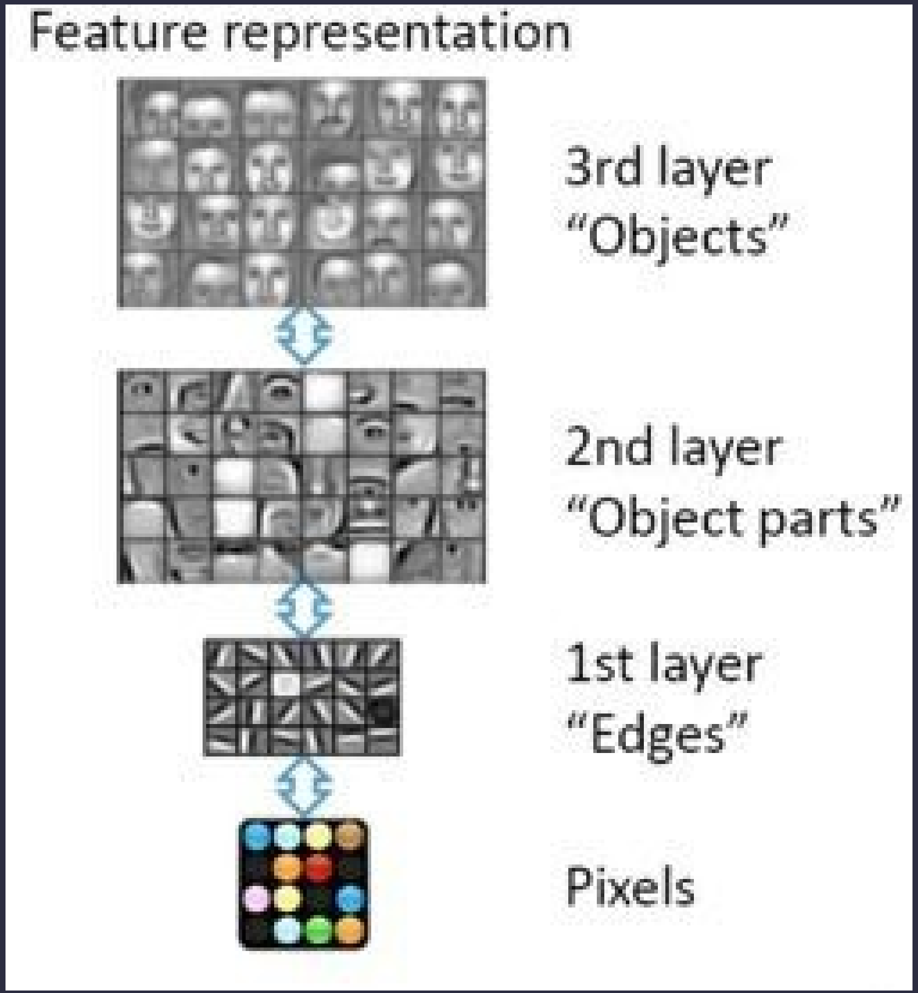
- We don't know the "right" levels of abstraction
- So let the model figure it out!



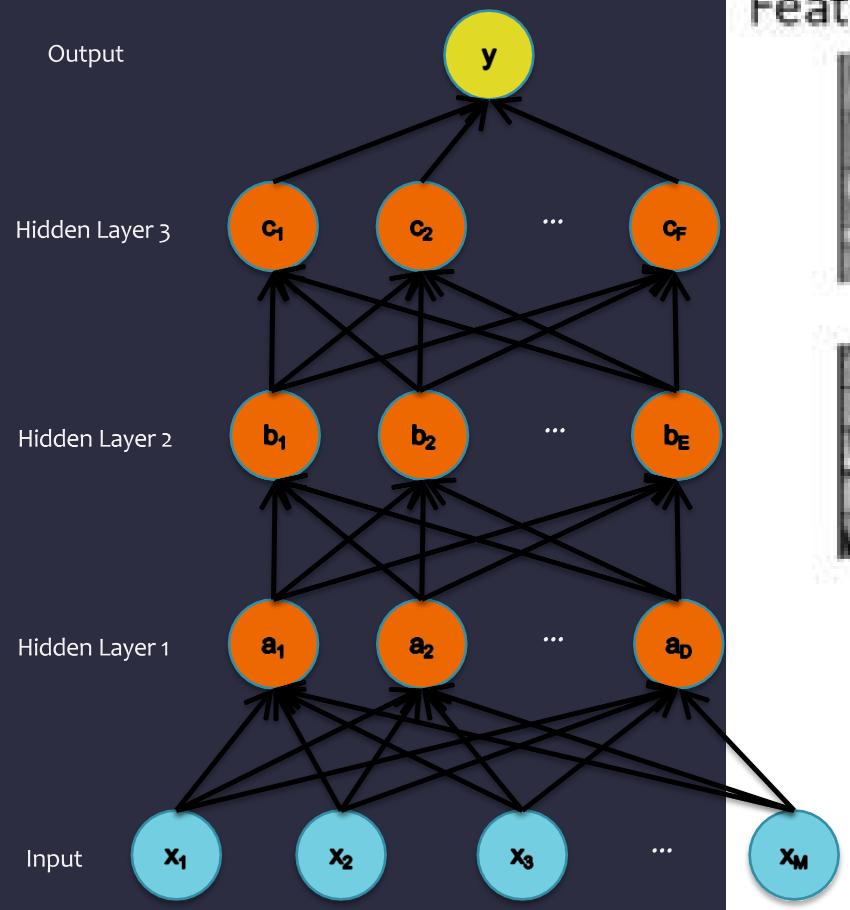
Different Levels of Abstraction

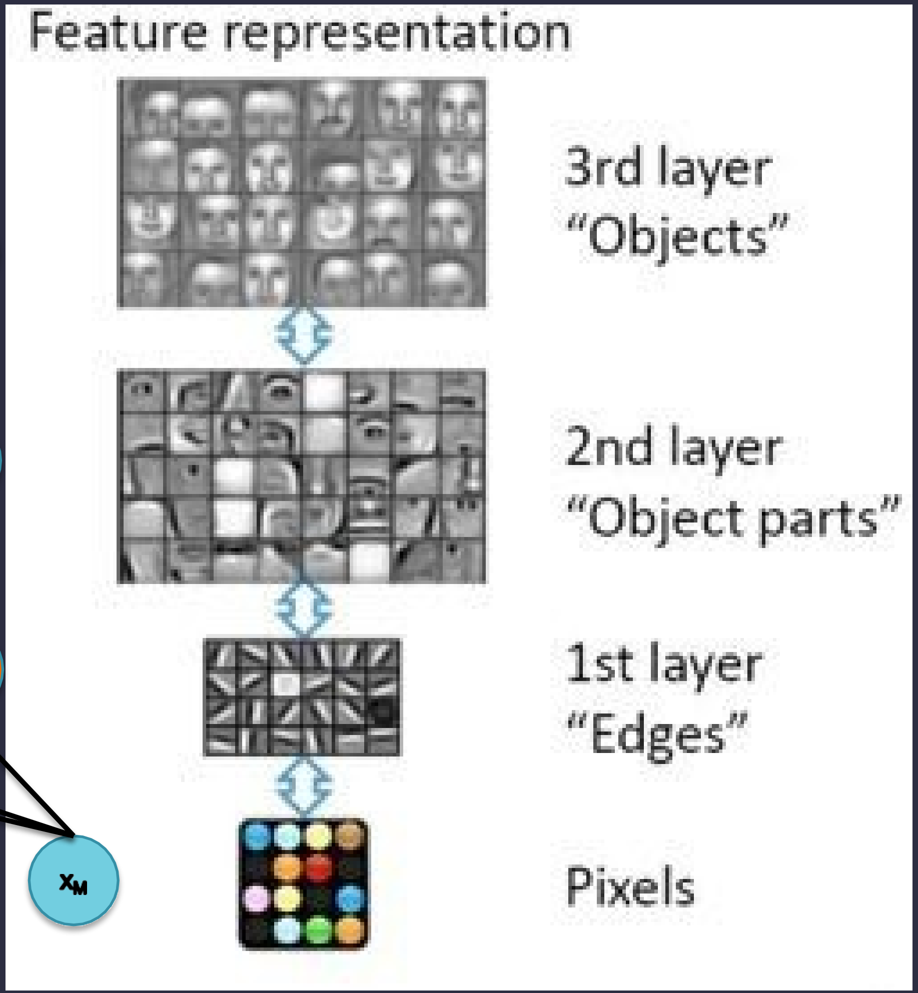
Face Recognition:

- Deep Network can build up increasingly higher levels of abstraction
- Lines, parts, regions



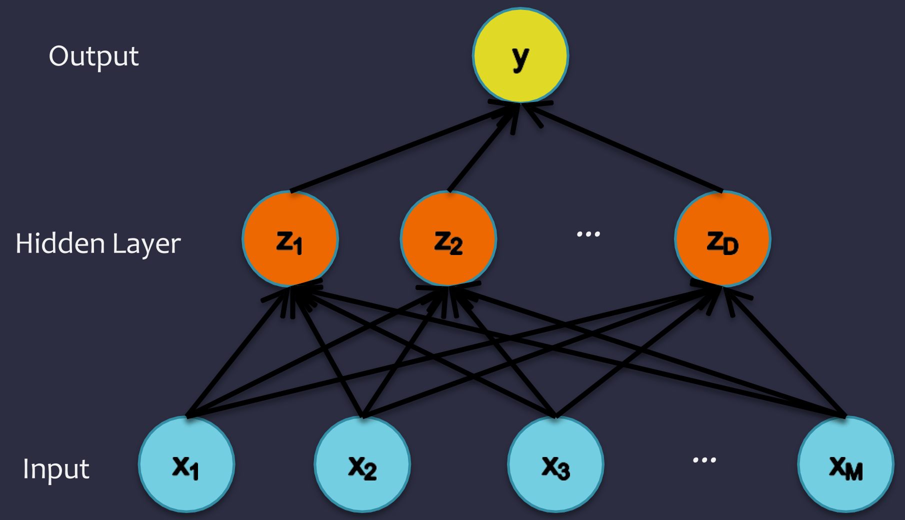
Different Levels of Abstraction





Activation Functions

Neural Network with sigmoid activation functions



(F) Loss
$$J = \frac{1}{2}(y - y^*)^2$$

(E) Output (sigmoid)
$$y = \frac{1}{1+\exp(-b)}$$

(D) Output (linear)
$$b = \sum_{j=0}^{D} \beta_j z_j$$

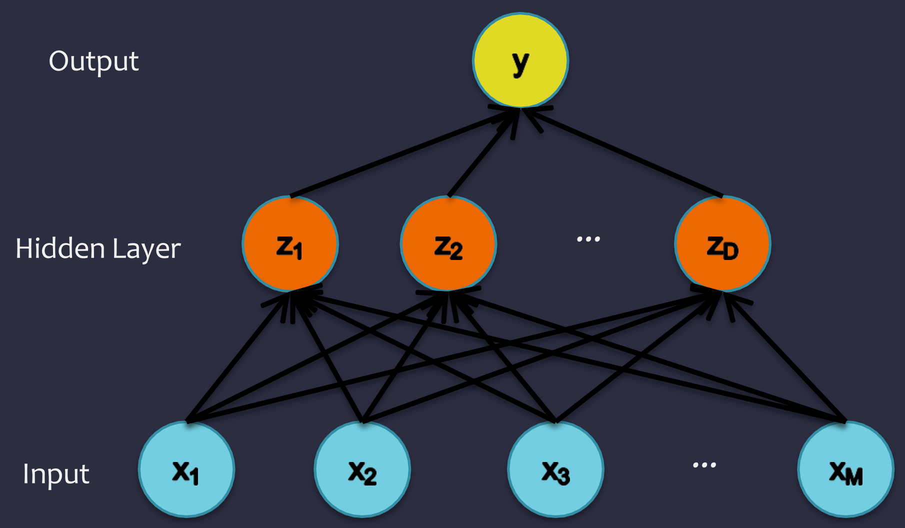
(C) Hidden (sigmoid)
$$z_j = \frac{1}{1+\exp(-a_j)}, \quad \forall j$$

(B) Hidden (linear)
$$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \quad \forall j$$

(A) Input
Given $x_i, \quad \forall i$

Activation Functions

Neural Network with arbitrary nonlinear activation functions



(F) Loss
$J = \frac{1}{2}(y - y^*)^2$

(E) Output (nonlinear)
$y = \sigma(b)$

(D) Output (linear)
$b = \sum_{j=0}^{D} \beta_j z_j$

(C) Hidden (nonlinear)
$z_j = \sigma(a_j), \quad \forall j$

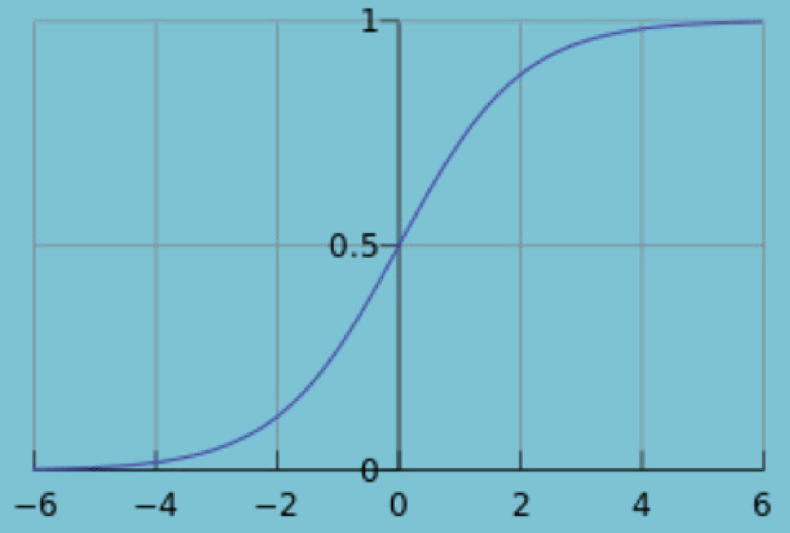
(B) Hidden (linear)
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \quad \forall j$

(A) Input
Given $x_i, \quad \forall i$

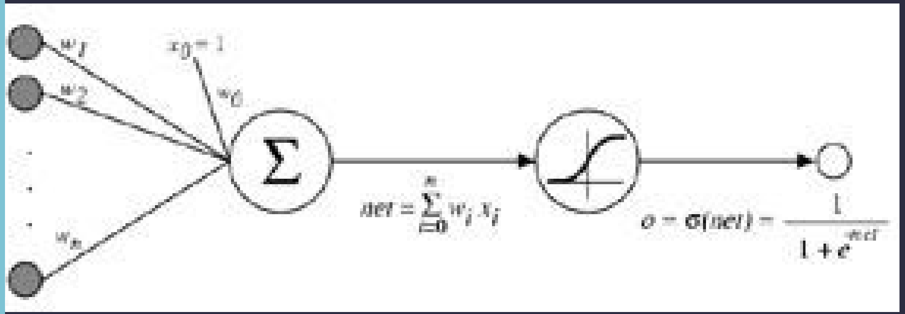
Activation Functions

Sigmoid / Logistic Function

$$\text{logistic}(u) \equiv \frac{1}{1+e^{-u}}$$

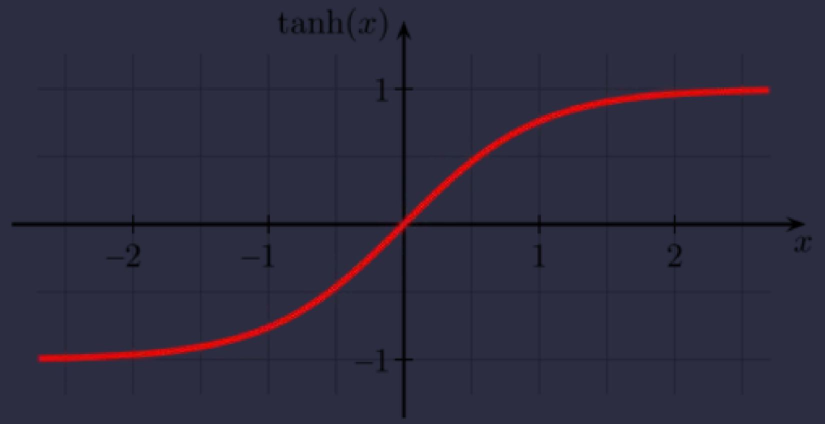


Activation function (nonlinearity) is sigmoid function…



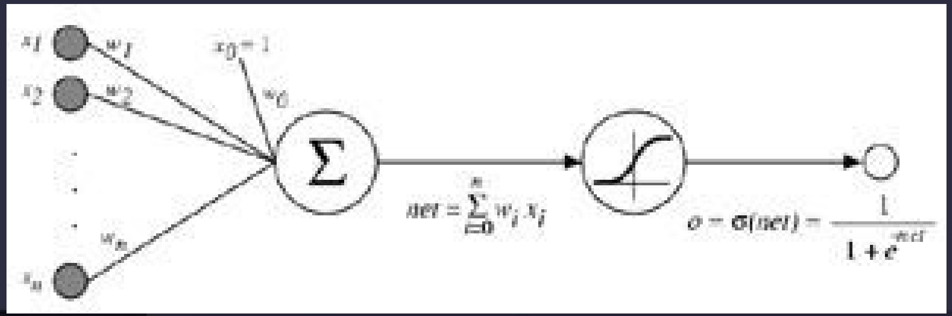
Activation Functions

- A new change: modifying the nonlinearity
  - The logistic is not widely used in modern ANNs



Alternate 1: tanh

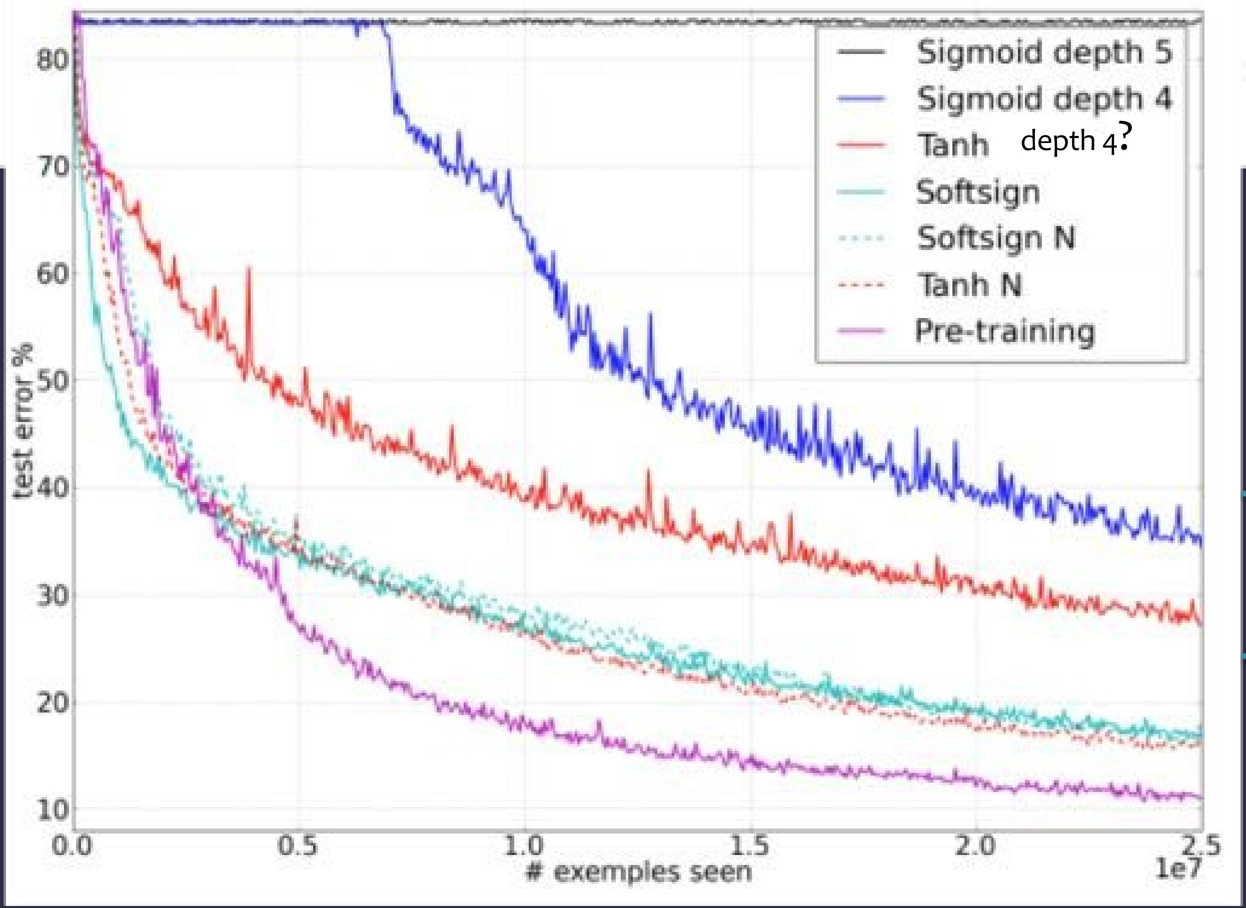
Like logistic function but shifted to range [-1, +1]



Slide from William Cohen

Understanding the difficulty of training deep feedforward neural networks

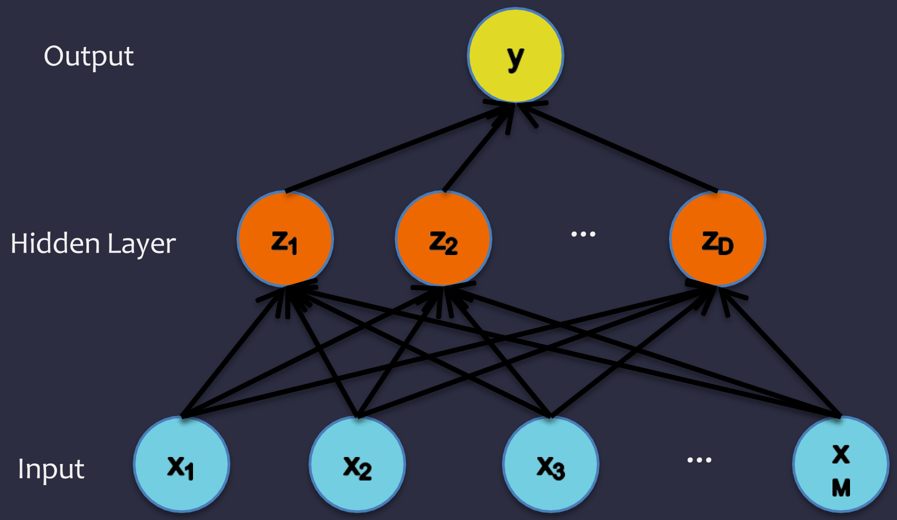
AI Stats 2010



sigmoid
vs.
tanh

Decision Functions

Neural Network for Classification



(E) **Output (sigmoid)**
$$y = \frac{1}{1+\exp(-b)}$$

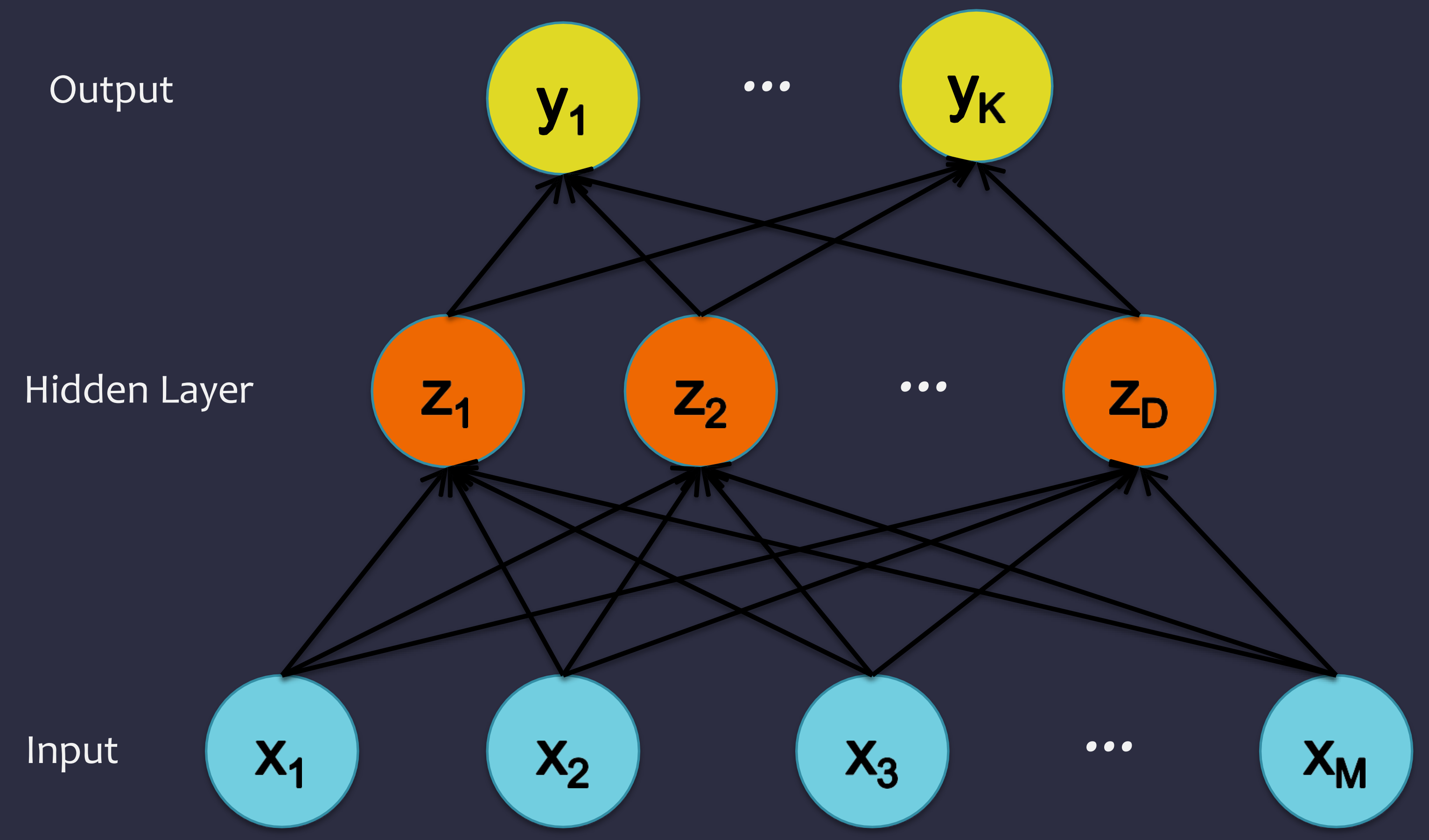
(D) **Output (linear)**
$$b = \sum_{j=0}^{D} \beta_j z_j$$

(C) **Hidden (sigmoid)**
$$z_j = \frac{1}{1+\exp(-a_j)}, \quad \forall j$$

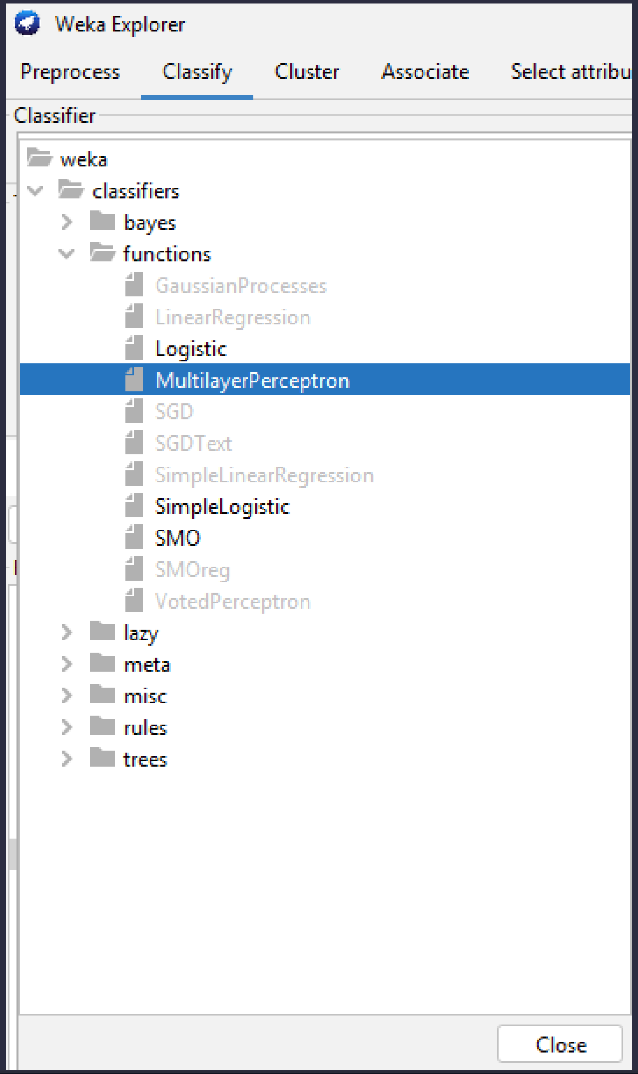
(B) **Hidden (linear)**
$$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \quad \forall j$$

(A) **Input**
Given $x_i, \quad \forall i$

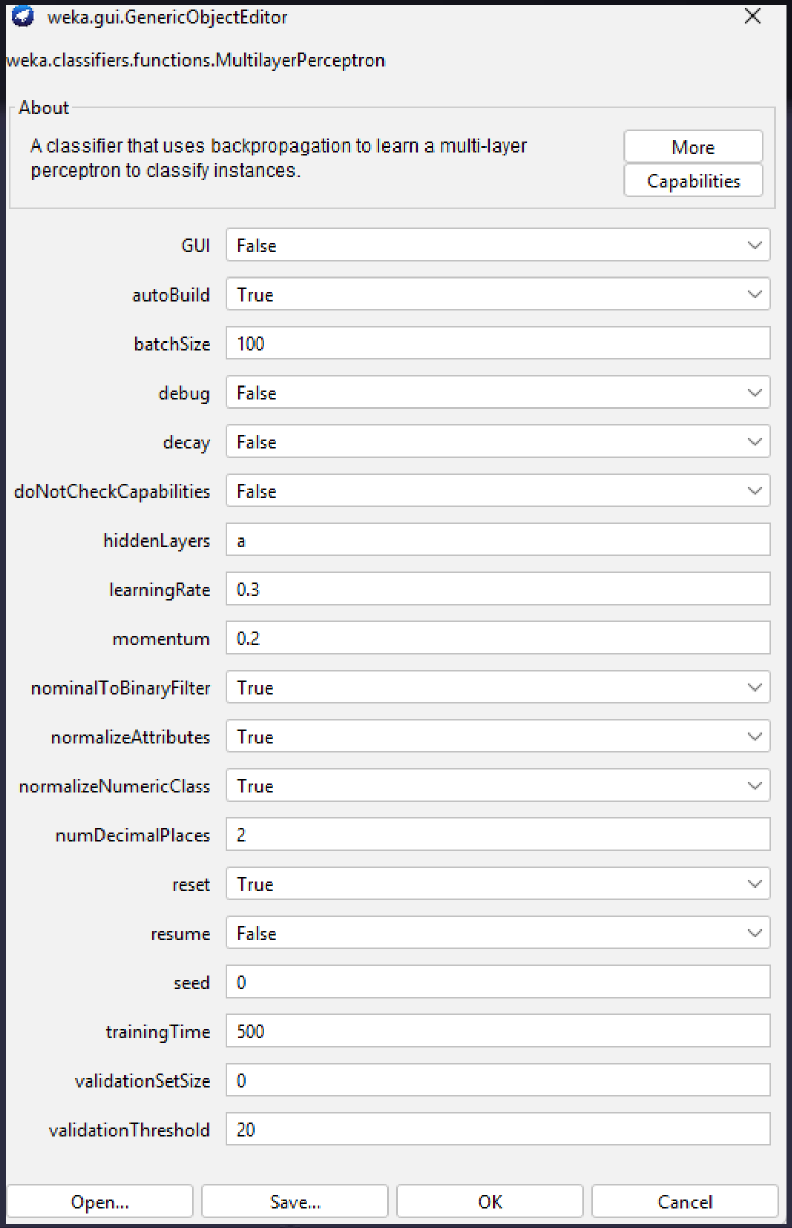
Multi-Class Output



NN on Weka

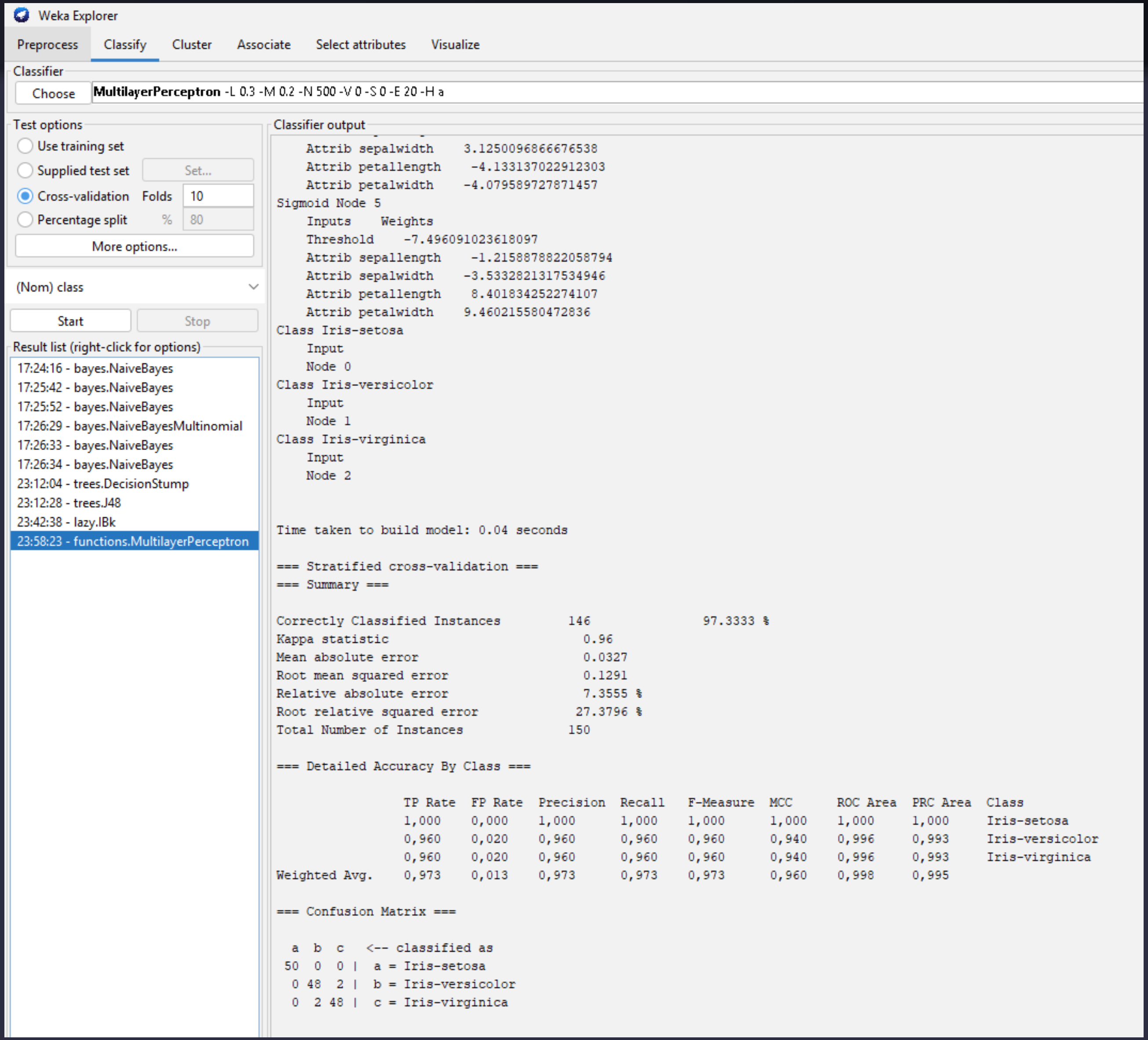


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attribu
Classifier
weka
classifiers
bayes
functions
GaussianProcesses
LinearRegression
Logistic
MultilayerPerceptron
SGD
SGDText
SimpleLinearRegression
SimpleLogistic
SMO
SMOreg
VotedPerceptron
lazy
meta
misc
rules
trees
Close

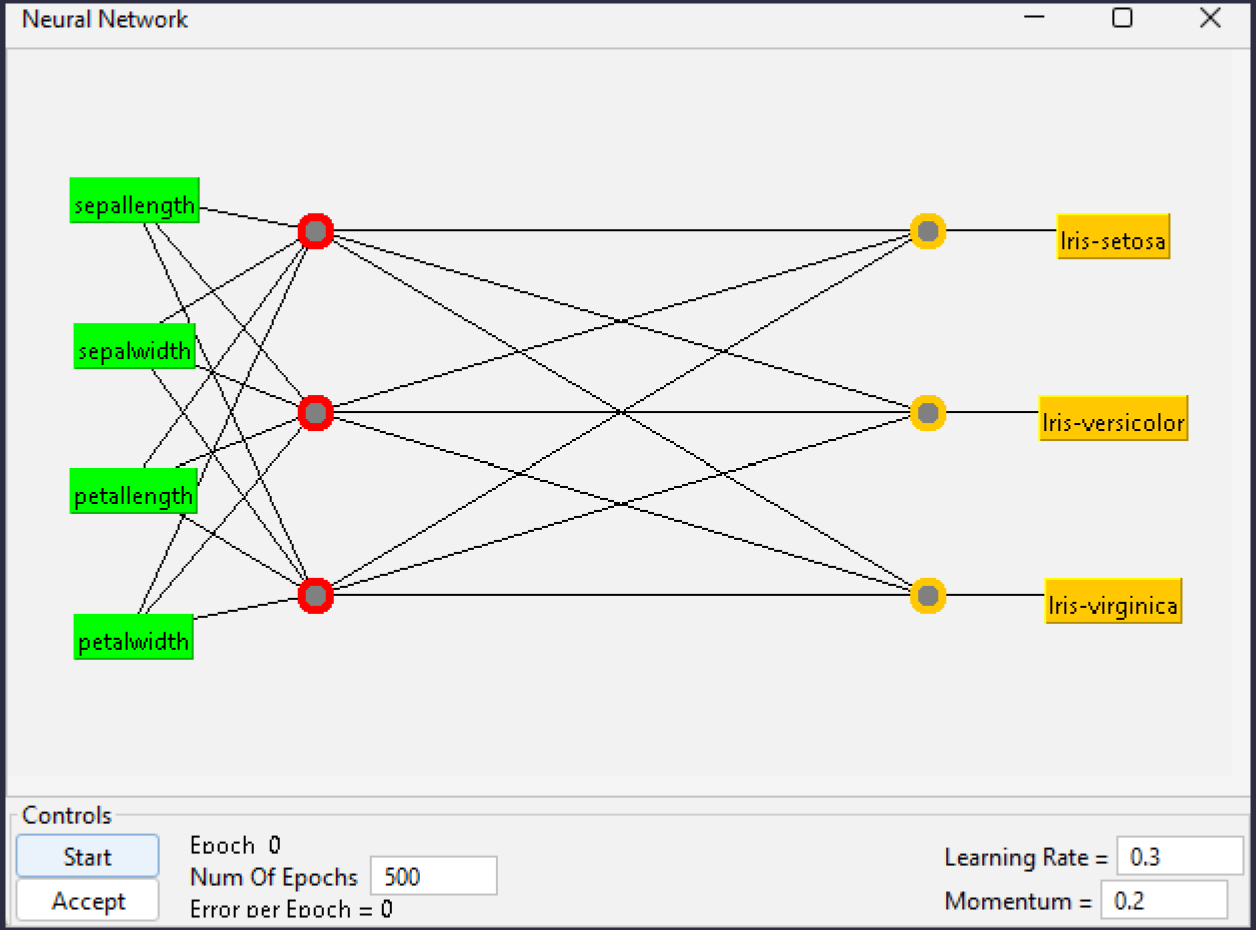


weka.gui.GenericObjectEditor
weka.classifiers.functions.MultilayerPerceptron
About
A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances.
More
Capabilities
GUI False
autoBuild True
batchSize 100
debug False
decay False
doNotCheckCapabilities False
hiddenLayers a
learningRate 0.3
momentum 0.2
nominalToBinaryFilter True
normalizeAttributes True
normalizeNumericClass True
numDecimalPlaces 2
reset True
resume False
seed 0
trainingTime 500
validationSetSize 0
validationThreshold 20
Open...
Save...
OK
Cancel

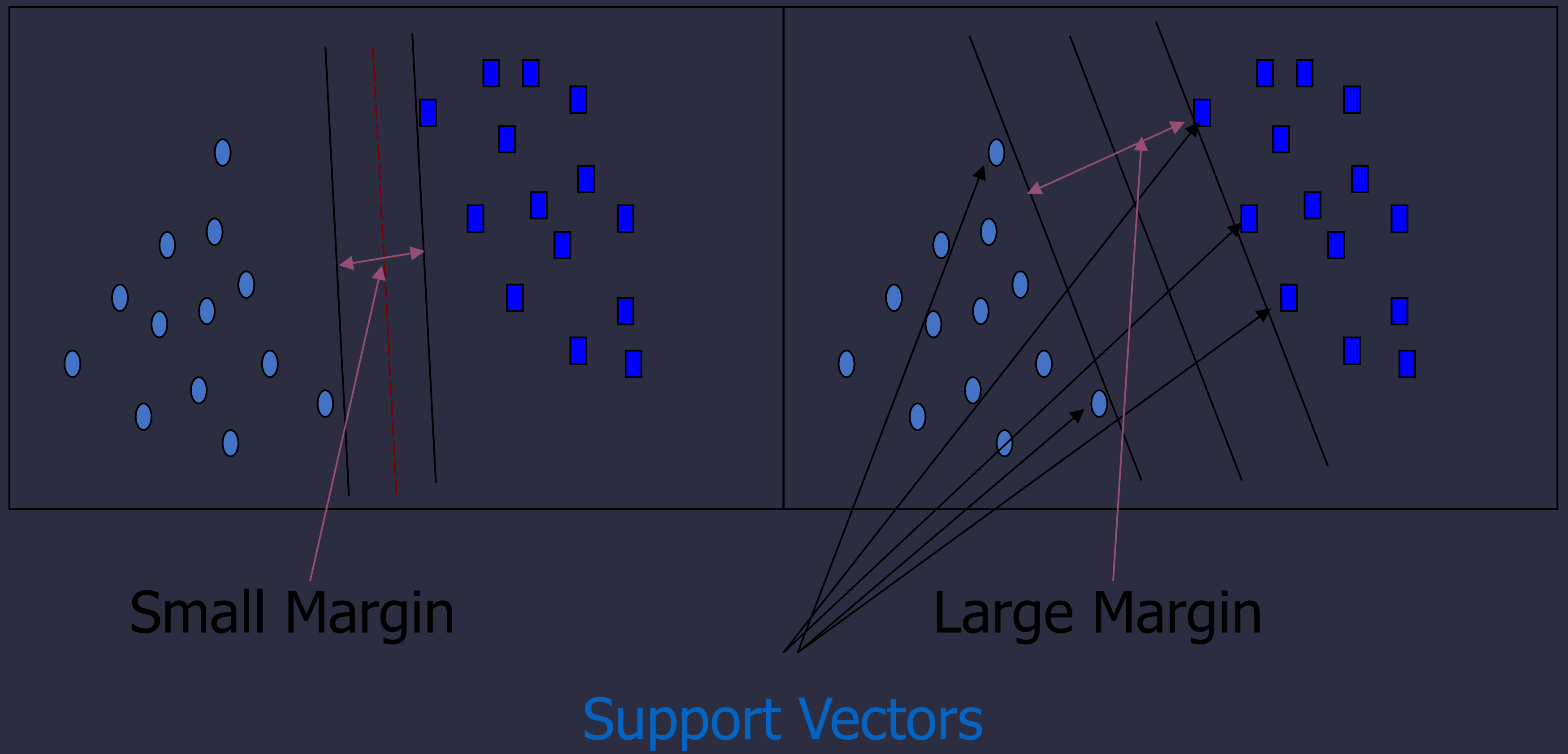
NN on Weka


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Test options
Use training set
Supplied test set
Set...
Cross-validation Folds 10
Percentage split % 80
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
17:24:16 - bayes.NaiveBayes
17:25:42 - bayes.NaiveBayes
17:25:52 - bayes.NaiveBayes
17:26:29 - bayes.NaiveBayesMultinomial
17:26:33 - bayes.NaiveBayes
17:26:34 - bayes.NaiveBayes
23:12:04 - trees.DecisionStump
23:12:28 - trees.J48
23:42:38 - lazy.IBk
23:58:23 - functions.MultilayerPerceptron
Classifier output
Attrib sepalwidth 3.1250096866676538
Attrib petallength -4.133137022912303
Attrib petalwidth -4.079589727871457
Sigmoid Node 5
Inputs Weights
Threshold -7.496091023618097
Attrib sepallength -1.2158878822058794
Attrib sepalwidth -3.5332821317534946
Attrib petallength 8.401834252274107
Attrib petalwidth 9.460215580472836
Class Iris-setosa
Input
Node 0
Class Iris-versicolor
Input
Node 1
Class Iris-virginica
Input
Node 2
Time taken to build model: 0.04 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 146 97.3333 %
Kappa statistic 0.96
Mean absolute error 0.0327
Root mean squared error 0.1291
Relative absolute error 7.3555 %
Root relative squared error 27.3796 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1,000 0,000 1,000 1,000 1,000 1,000 1,000 1,000 Iris-setosa
0,960 0,020 0,960 0,960 0,960 0,940 0,996 0,993 Iris-versicolor
0,960 0,020 0,960 0,960 0,960 0,940 0,996 0,993 Iris-virginica
Weighted Avg. 0,973 0,013 0,973 0,973 0,973 0,960 0,998 0,995
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 48 2 | b = Iris-versicolor
0 2 48 | c = Iris-virginica

NN on Weka


Neural Network
sepallength
sepalwidth
petallength
petalwidth
Iris-setosa
Iris-versicolor
Iris-virginica
Controls
Start
Accept
Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2

SVM – Support Vector Machines

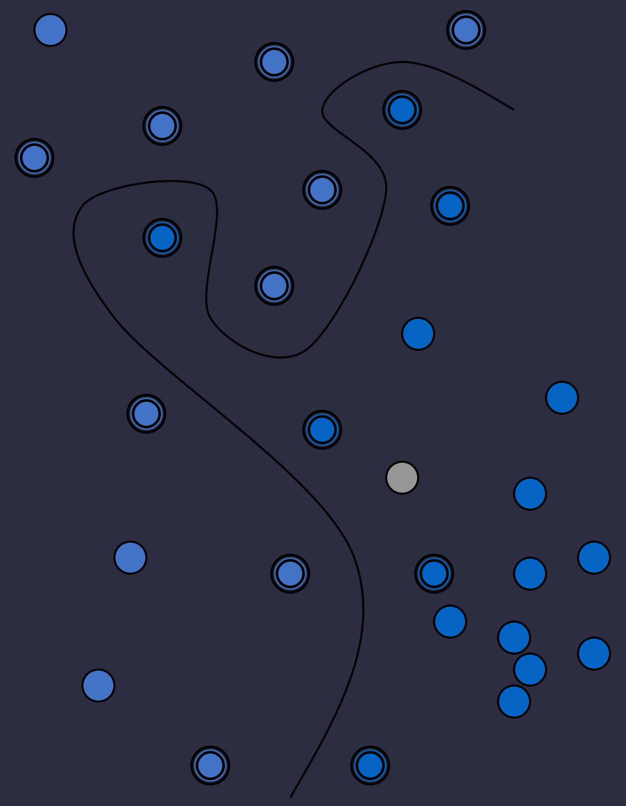


Support vector machine(SVM).

- Classification is essentially finding the best boundary between classes.
- Support vector machine finds the best boundary points called support vectors and build classifier on top of them.
- Linear and Non-linear support vector machine.

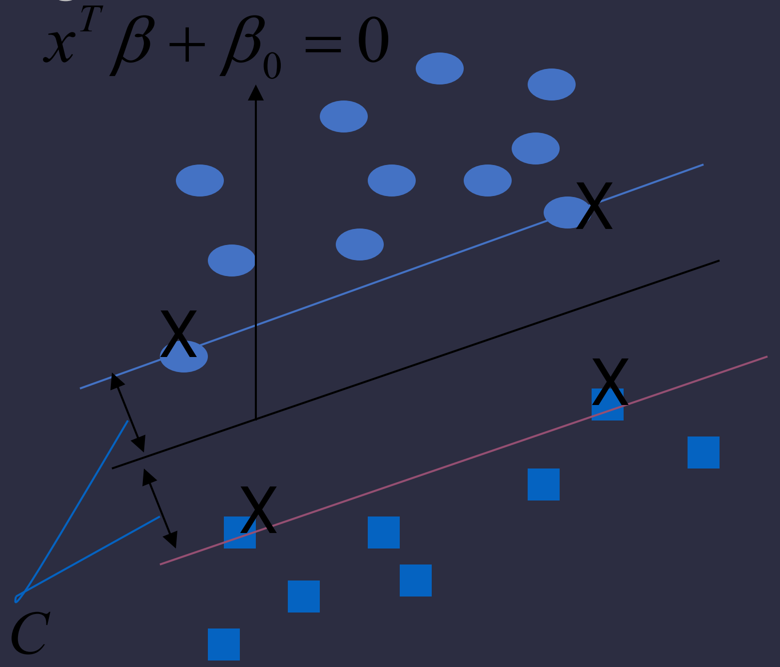
Example of general SVM

The dots with shadow around them are support vectors. Clearly they are the best data points to represent the boundary. The curve is the separating boundary.

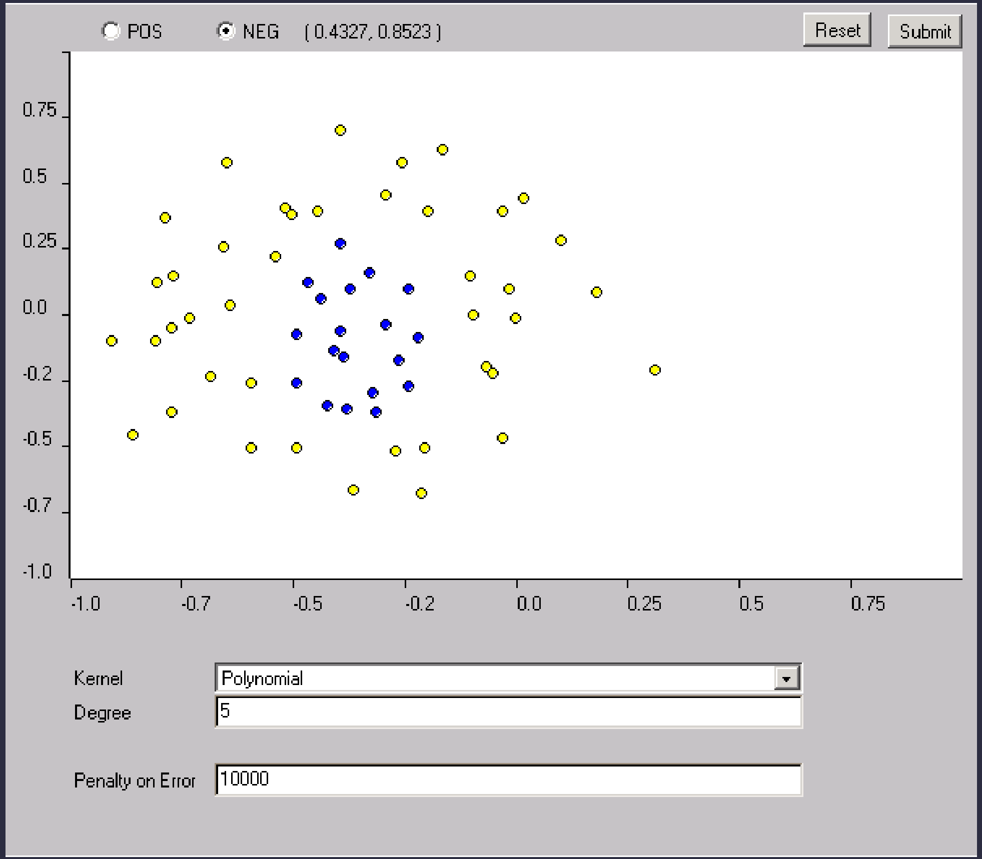


Optimal Hyper plane, separable case.

- In this case, class 1 and class 2 are separable.
- The representing points are selected such that the margin between two classes are maximized.
- Crossed points are support vectors.



Example of Non-linear SVM

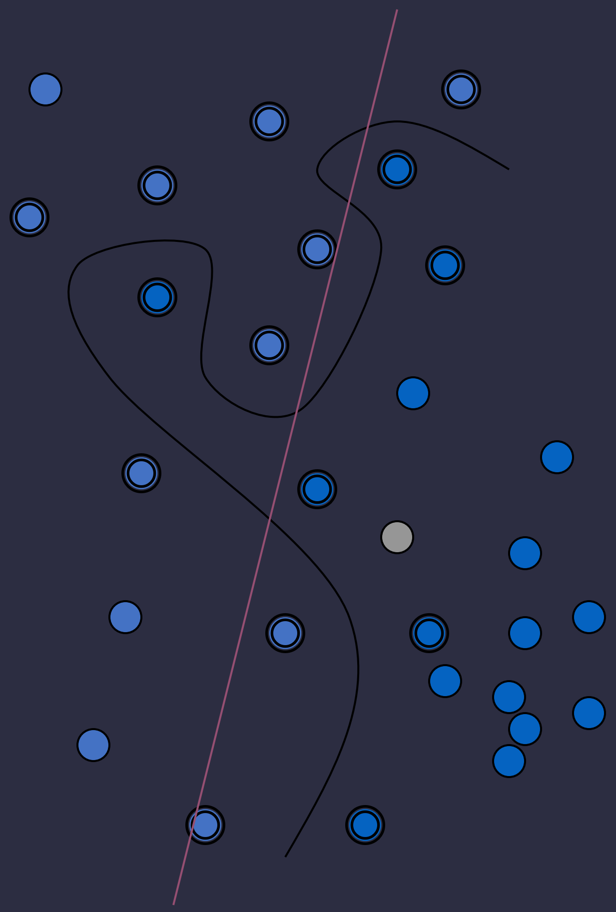


General SVM

This classification problem clearly do not have a good optimal linear classifier.

Can we do better?

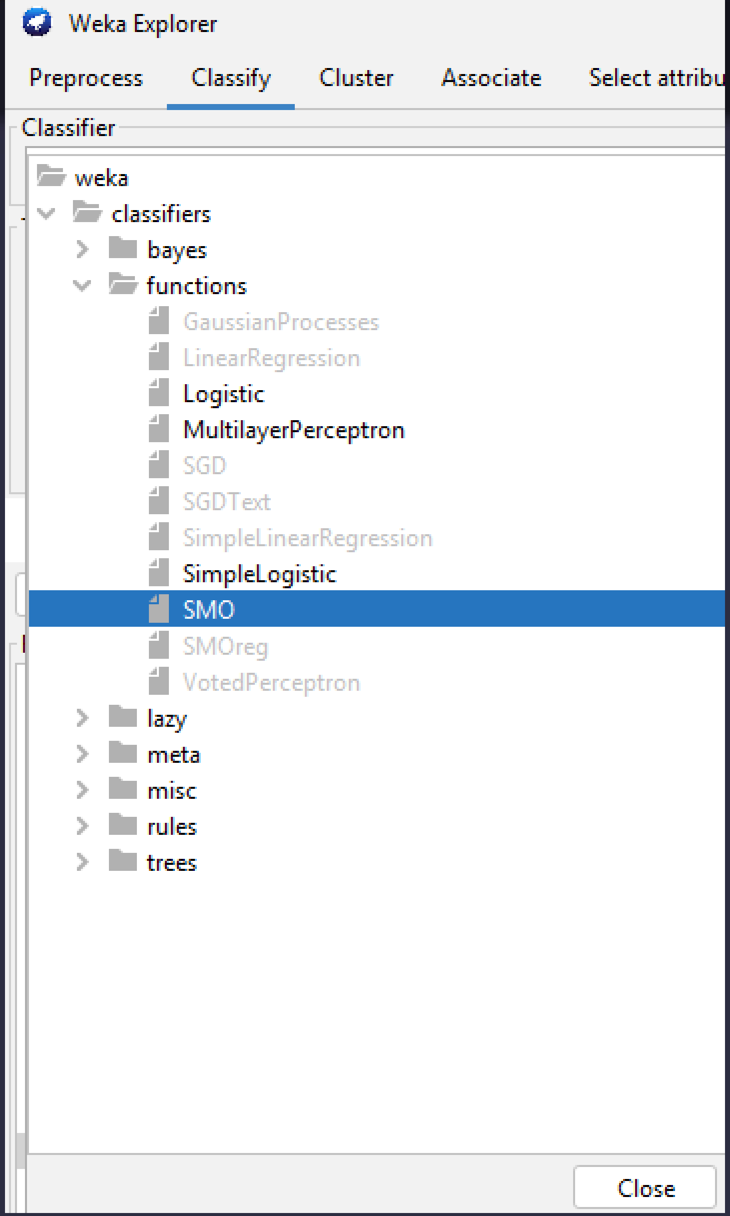
A non-linear boundary as shown will do fine.



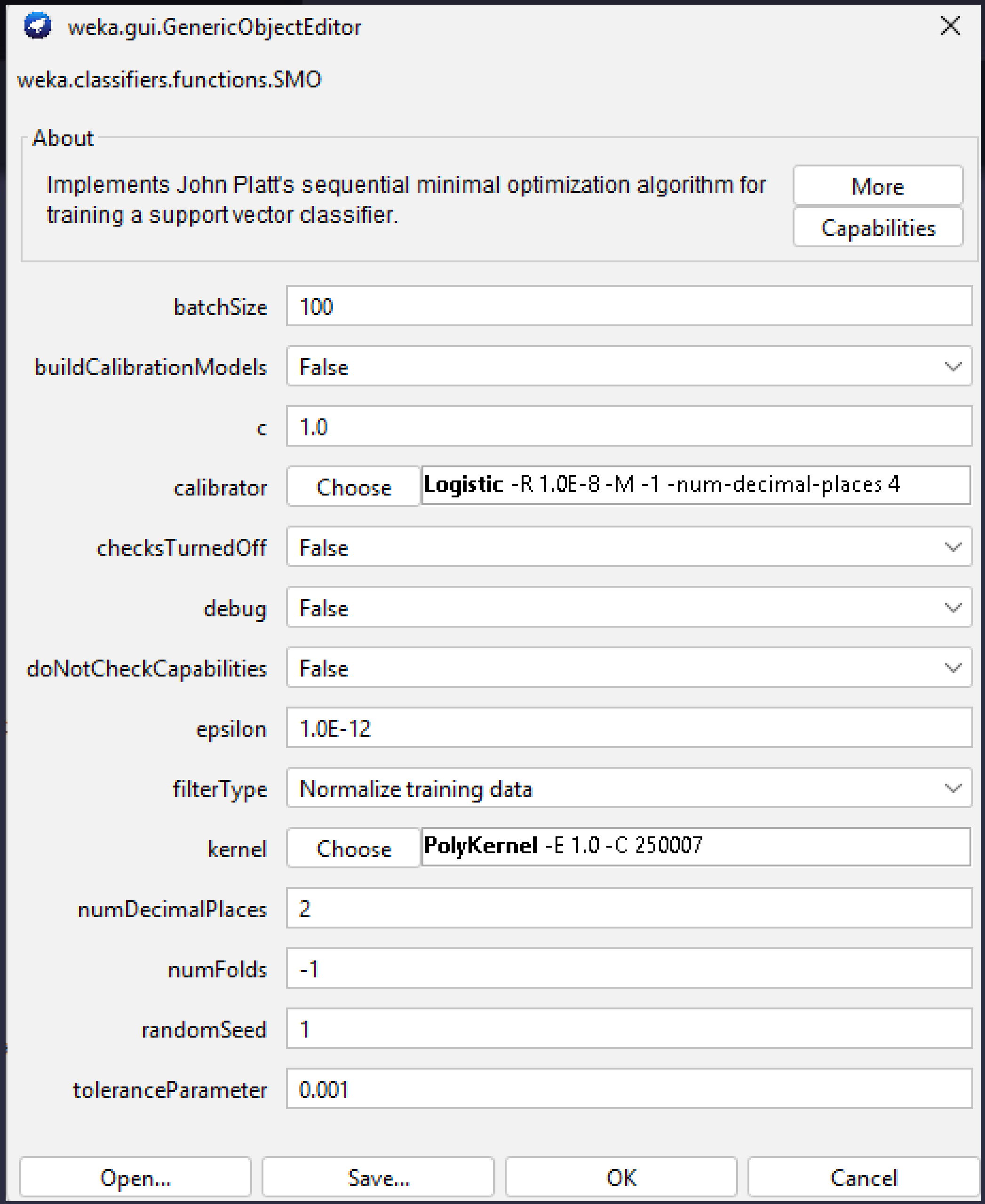
General SVM Cont.

- The idea is to map the feature space into a much bigger space so that the boundary is linear in the new space.
- Generally linear boundaries in the enlarged space achieve better training-class separation, and it translates to non-linear boundaries in the original space.

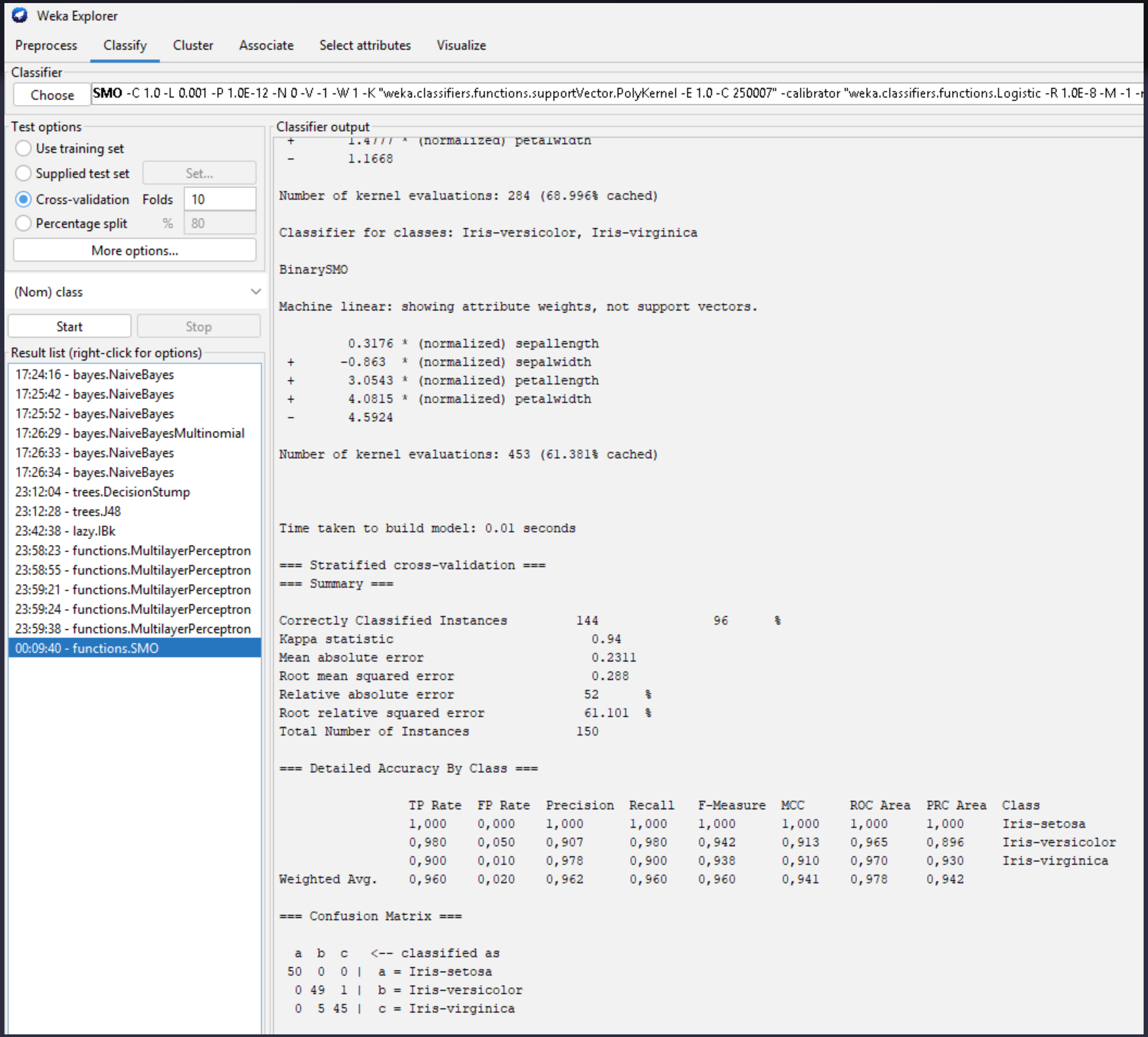
SVM on Veka


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attribu
Classifier
weka
classifiers
bayes
functions
GaussianProcesses
LinearRegression
Logistic
MultilayerPerceptron
SGD
SGDText
SimpleLinearRegression
SimpleLogistic
SMO
SMOreg
VotedPerceptron
lazy
meta
misc
rules
trees
Close

SVM on Veka


weka.gui.GenericObjectEditor
weka.classifiers.functions.SMO
About
Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier.
More
Capabilities
batchSize
100
buildCalibrationModels
False
c
1.0
calibrator
Choose
Logistic -R 1.0E-8 -M -1 -num-decimal-places 4
checksTurnedOff
False
debug
False
doNotCheckCapabilities
False
epsilon
1.0E-12
filterType
Normalize training data
kernel
Choose
PolyKernel -E 1.0 -C 250007
numDecimalPlaces
2
numFolds
-1
randomSeed
1
toleranceParameter
0.001
Open...
Save...
OK
Cancel

SVM on Veka



Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -
Test options
Use training set
Supplied test set
Set...
Cross-validation Folds 10
Percentage split % 80
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
17:24:16 - bayes.NaiveBayes
17:25:42 - bayes.NaiveBayes
17:25:52 - bayes.NaiveBayes
17:26:29 - bayes.NaiveBayesMultinomial
17:26:33 - bayes.NaiveBayes
17:26:34 - bayes.NaiveBayes
23:12:04 - trees.DecisionStump
23:12:28 - trees.J48
23:42:38 - lazy.IBk
23:58:23 - functions.MultilayerPerceptron
23:58:55 - functions.MultilayerPerceptron
23:59:21 - functions.MultilayerPerceptron
23:59:24 - functions.MultilayerPerceptron
23:59:38 - functions.MultilayerPerceptron
00:09:40 - functions.SMO
Classifier output
- 1.1668
Number of kernel evaluations: 284 (68.996% cached)
Classifier for classes: Iris-versicolor, Iris-virginica
BinarySMO
Machine linear: showing attribute weights, not support vectors.
0.3176 * (normalized) sepallength
+ -0.863 * (normalized) sepalwidth
+ 3.0543 * (normalized) petallength
+ 4.0815 * (normalized) petalwidth
- 4.5924
Number of kernel evaluations: 453 (61.381% cached)
Time taken to build model: 0.01 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 144 96 %
Kappa statistic 0.94
Mean absolute error 0.2311
Root mean squared error 0.288
Relative absolute error 52 %
Root relative squared error 61.101 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1,000 0,000 1,000 1,000 1,000 1,000 1,000 1,000 Iris-setosa
0,980 0,050 0,907 0,980 0,942 0,913 0,965 0,896 Iris-versicolor
0,900 0,010 0,978 0,900 0,938 0,910 0,970 0,930 Iris-virginica
Weighted Avg. 0,960 0,020 0,962 0,960 0,960 0,941 0,978 0,942
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 49 1 | b = Iris-versicolor
0 5 45 | c = Iris-virginica

The Problem of Feature Selection

- Large number of features; sometimes greater than 100.
- The number of combinations can be well over a billion!

- Is there a way to search for an optimal set of features in reasonable time and with reasonable computation power?

Different ways to search for this needle

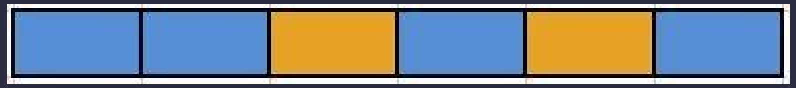
- Evaluate every possible combination to come up with the best combination – the brute force method!
- Step-up/step-down methods that add or remove a feature at a time and evaluate model performance.
- Use genetic algorithms (GA) for searching this huge solution space.

Genetic Algorithms (GA)

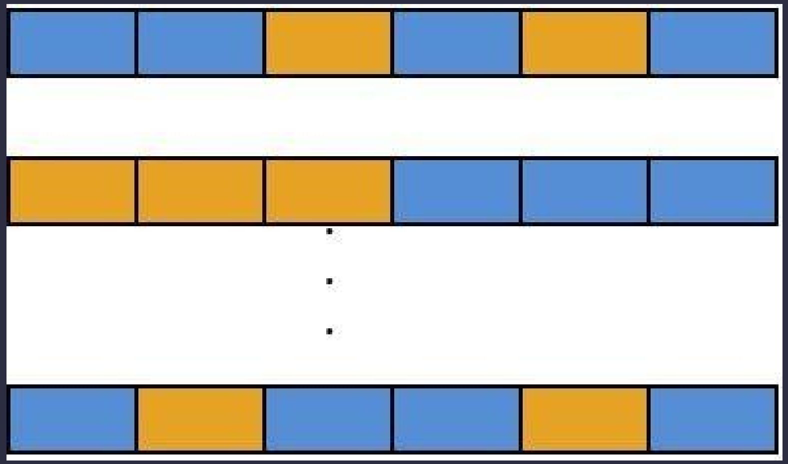
- This is a high level simulation of a biologically inspired adaptive system – evolution.
- Using a simple set of rules, this system can have emergent behaviour that makes it useful for various applications.
- GA have been used in applications such as
  - predicting the structure of proteins
  - training neural networks
- Here, I will talk about the use of GA for searching through the feature space to select an optimal set of features.

Terms associated with GA

- **Chromosome** – a potential solution to the problem. A common way to represent solutions is using binary numbers.

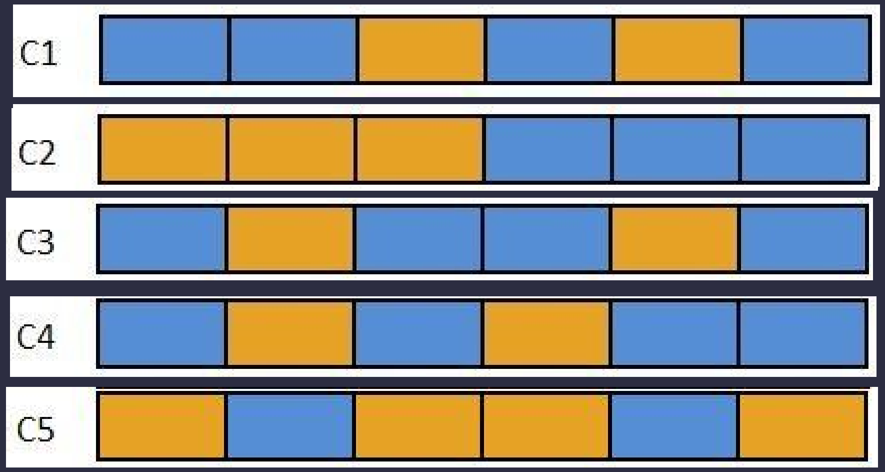


- **Population** – a set of chromosomes belonging to a generation.



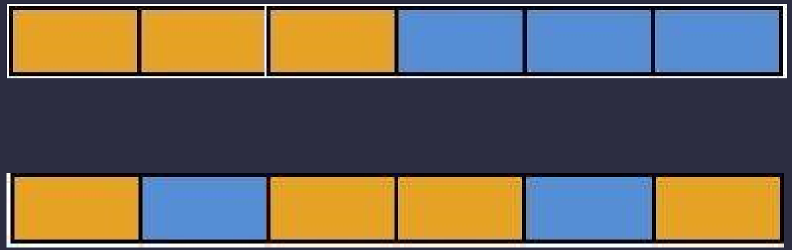
Terms associated with GA

- **Fitness** – a metric to evaluate how well a particular solution solves the problem.
- **Generation** – each iteration of the algorithm.
- **Selection** – a process by which some chromosomes of a population are chosen for generating new solutions.

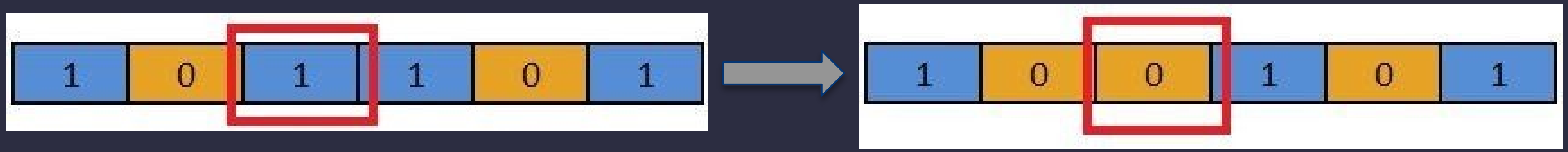


Terms associated with GA

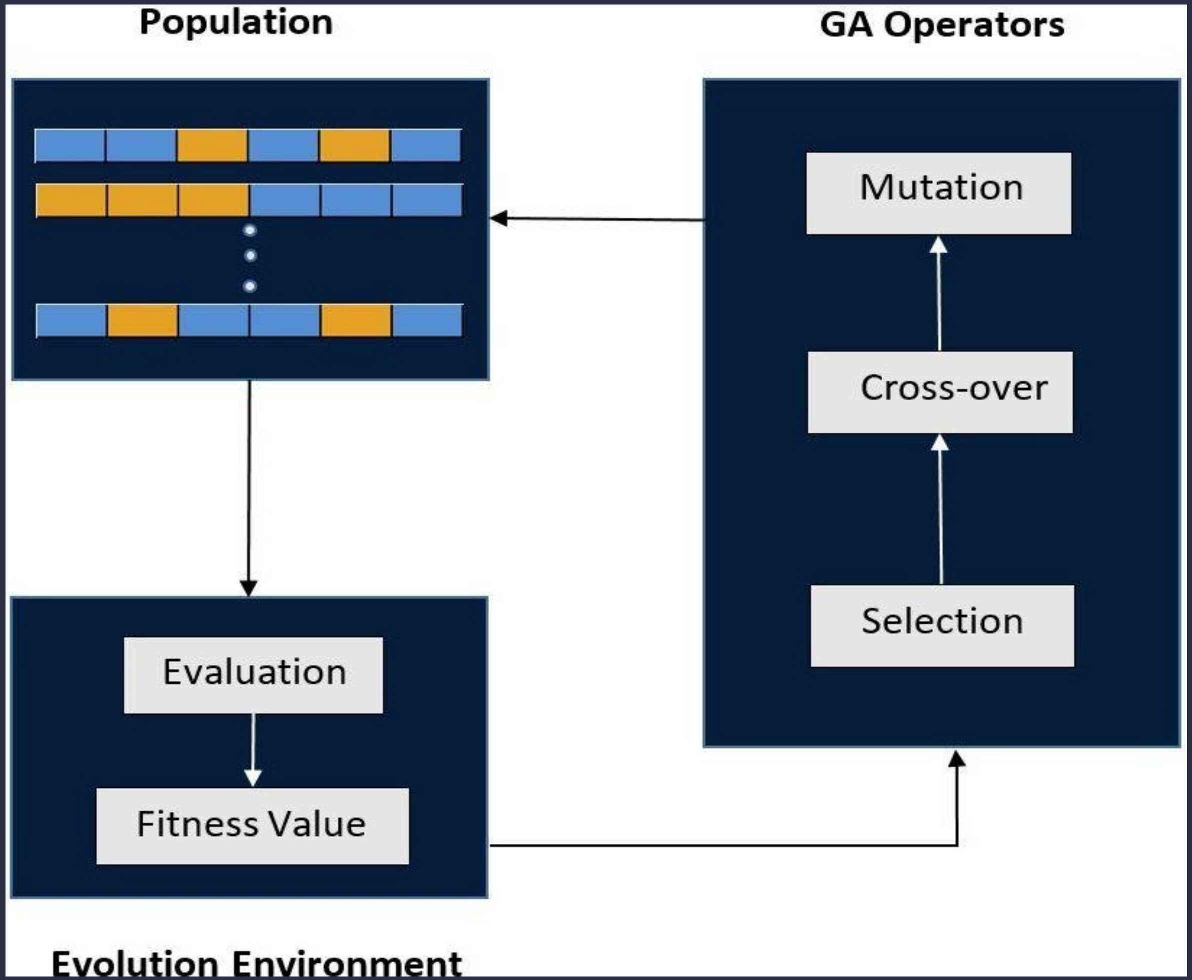
- **Cross-over** – is the process of exchange of information between selected chromosomes.



- **Mutation** – random changes in chromosomes.



Schematic of a GA



Performance