Upcoming Deadlines

- **Project Team Formation**
  - If you have not yet formed a team, please email us
  - We will randomly assign teams if they are not formed by Wednesday (9/27)

- **HW 2 due Wednesday (9/27) at 8pm**

- Quiz 2 due Thursday (9/28) at 8pm

Recap

- Linear regression for regression
  - Bias-variance tradeoff
  - Regularization
  - Cross-validation
  - Optimization

- Logistic regression for classification
  - Maximum likelihood framework
  - Different evaluation metrics

Lecture 7: kNNs

CIS 4190/5190

Fall 2023

Parametric vs. Non-Parametric Learning

- The algorithms we have seen so far are **parametric**
  - Assume the model family has the form $\{ f_\beta \mid \beta \in \mathbb{R}^d \}$
- Not all model families have this form!
- **Non-parametric models**
  - Very high capacity model families that can fit “arbitrary” functions

k-Nearest Neighbors (kNN)

- **Classification:** Given a new input x:
  - **Step 1:** Find k nearest neighbors $\{i_1, \ldots, i_k\}$ in the training dataset

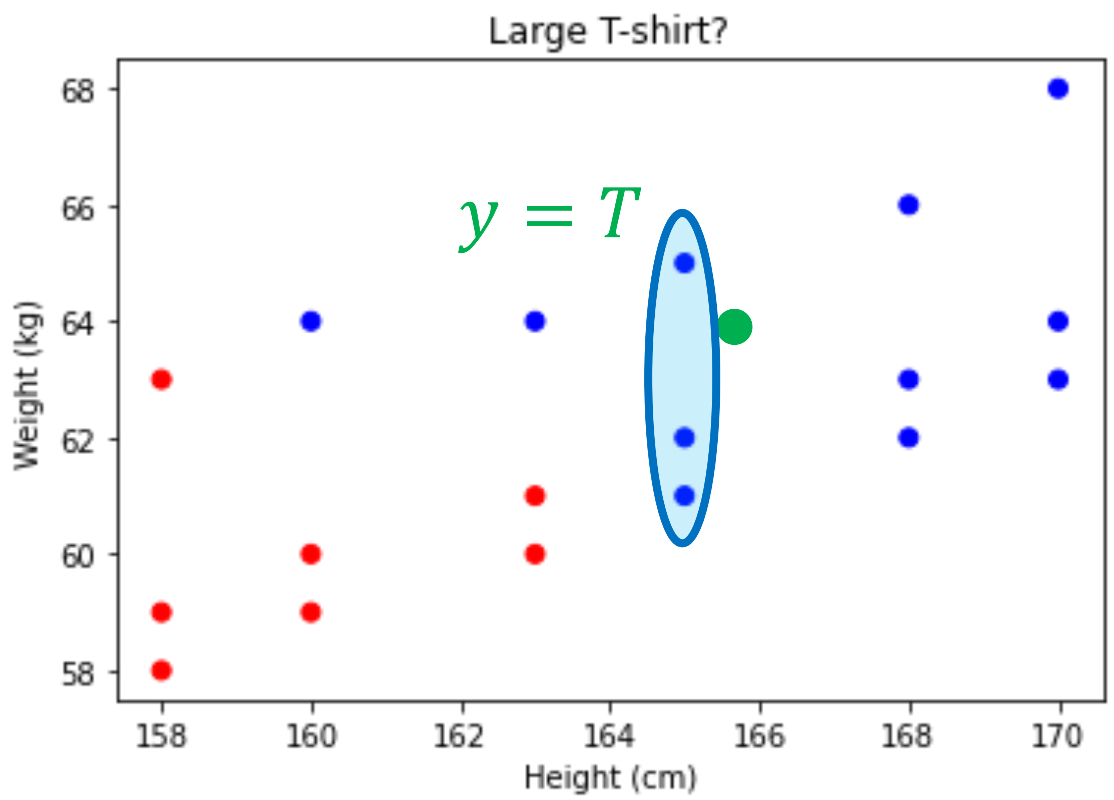
$$i_1, \ldots, i_k = k\mathrm{ArgMin}_{i\in\{1,\ldots,n\}} \operatorname{dist}(x, x_i)$$

  - **Step 2:** Return the majority label (i.e., label that occurs most frequently):

$$y = \mathrm{Majority}\left\{y_{i_1}, \ldots, y_{k_k}\right\}$$

Example: T-Shirt Size

$$k = 3$$



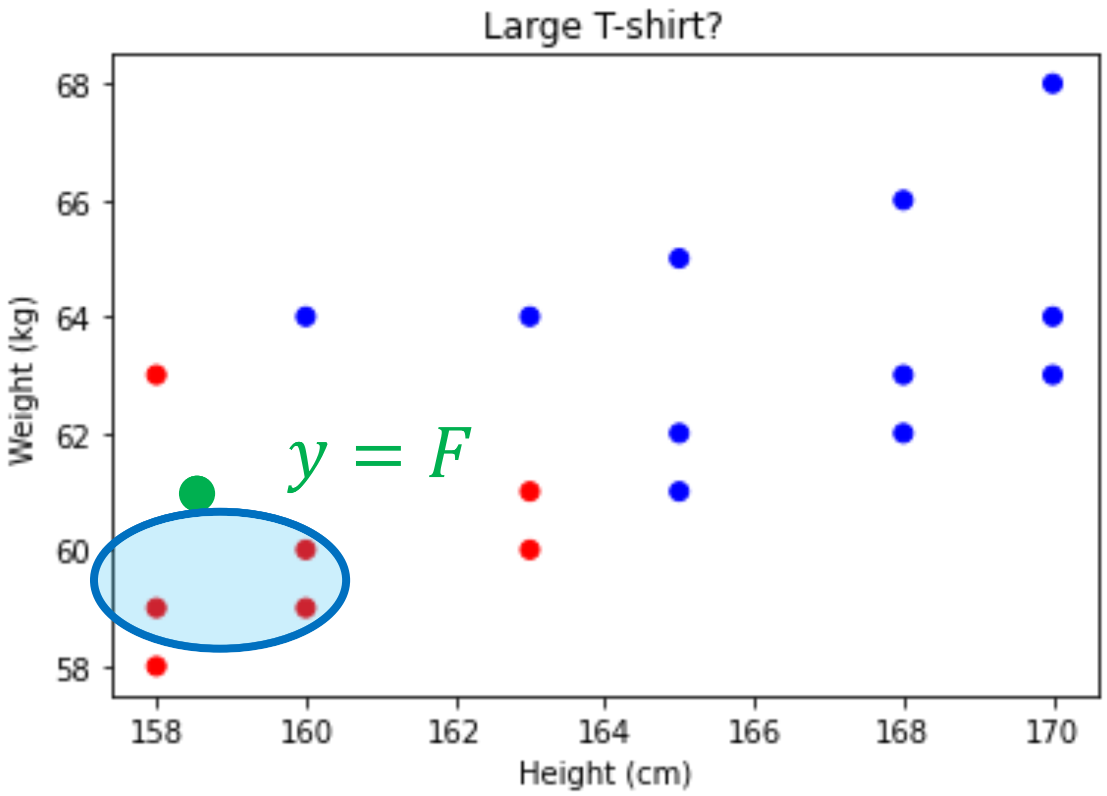
Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Inputs x_i Labels y_i

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

Example: T-Shirt Size

$k = 3$

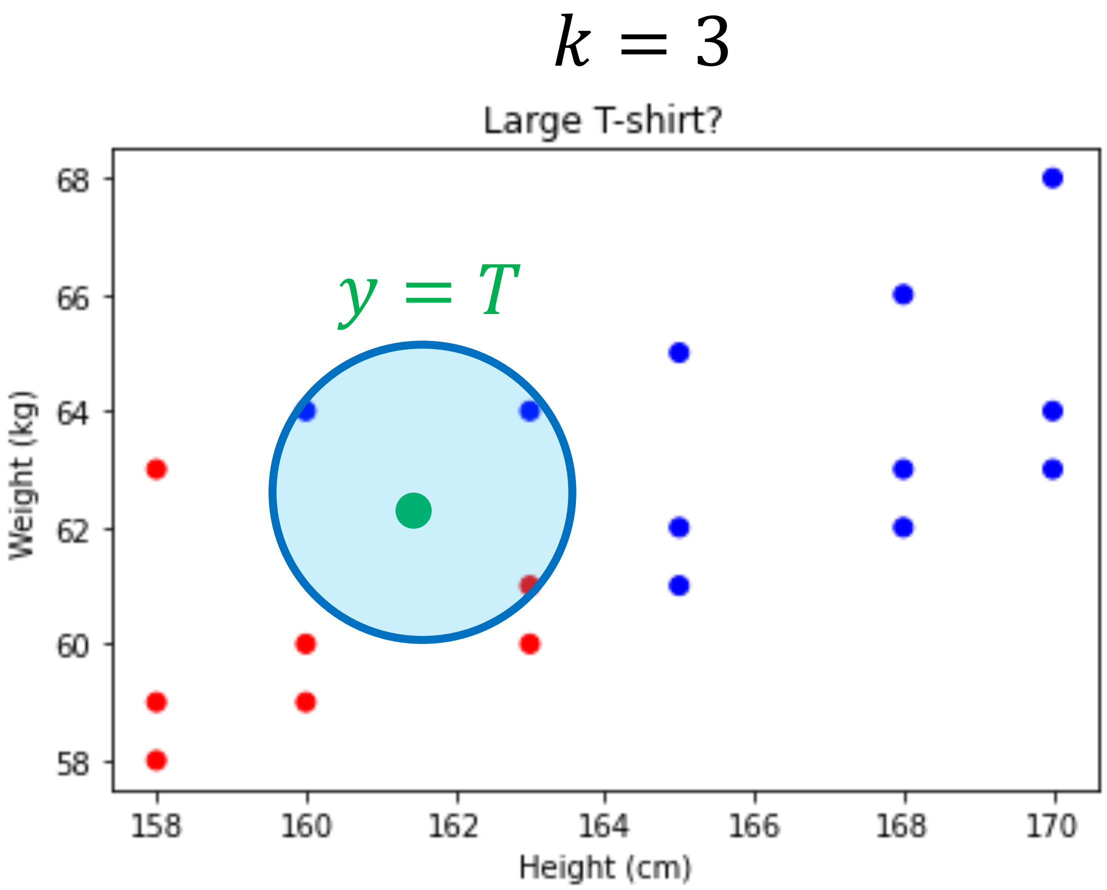


Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Inputs x_i Labels y_i

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

Example: T-Shirt Size



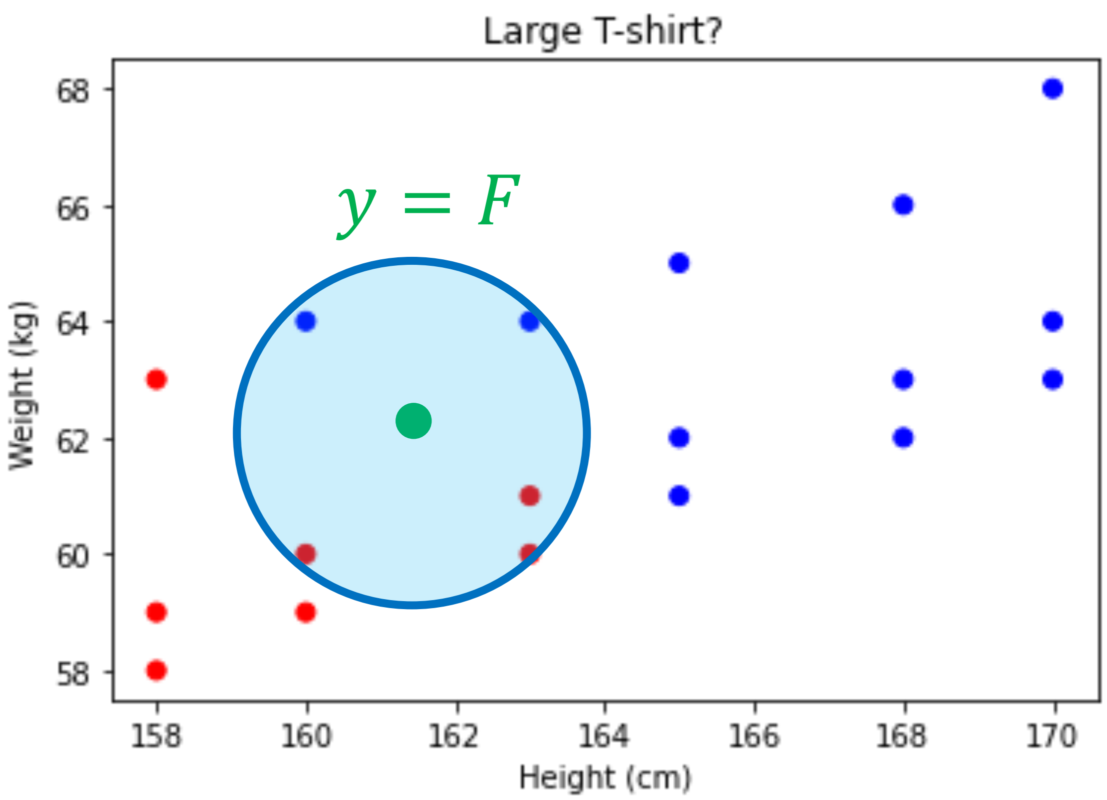
Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Inputs x_i Labels y_i

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

Example: T-Shirt Size

$k = 5$



Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Inputs x_i Labels y_i

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

k-Nearest Neighbors (kNN)

- **Regression:** Given a new input x:
  - **Step 1:** Find k nearest neighbors $\{i_1, \ldots, i_k\}$ in the training dataset

$$i_1, \ldots, i_k = k\mathrm{ArgMin}_{i \in \{1,\ldots,n\}} \, \mathrm{dist}(x, x_i)$$

  - **Step 2:** Return the average label (i.e., label that occurs most frequently):

$$y = \mathrm{Average}\,\{y_{i_1}, \ldots, y_{k_k}\}$$

- We can use this approach to get probabilities for classification

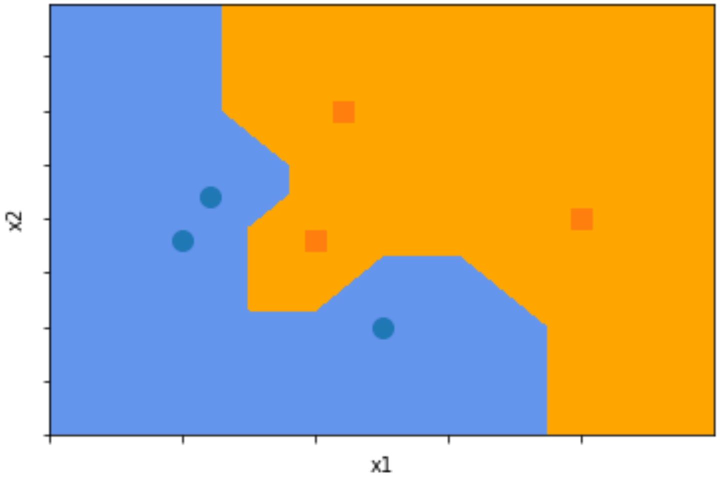
k-Nearest Neighbors (kNN)

- **General framework**
    - **Design decision 1:** What notion of “distance” to use? (e.g., L_2 distance)
    - **Design decision 2:** How to aggregate labels? (e.g., majority or average)

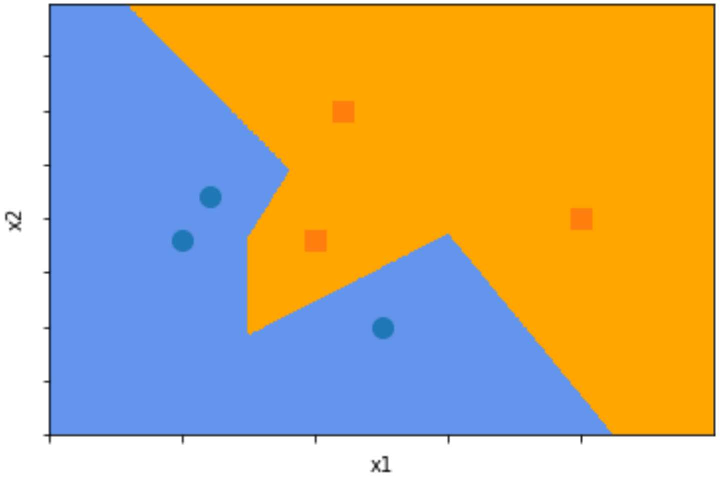
Choice of Distance Function

- $\boldsymbol{L_1}$ **distance:** $\text{dist}(x, x') = \|x - x'\|_1 = \sum_{j=1}^{d} \left| x_j - x_j' \right|$
- $\boldsymbol{L_2}$ **distance:** $\text{dist}(x, x') = \|x - x'\|_2 = \left(\sum_{j=1}^{d} \left(x_j - x_j' \right)^2 \right)^{\frac{1}{2}}$
- $\boldsymbol{L_\infty}$ **distance:** $\text{dist}(x, x') = \|x - x'\|_\infty = \max_{j \in \{1,\ldots,d\}} \left| x_j - x_j' \right|$
- $\boldsymbol{L_p}$ **distance:** $\text{dist}(x, x') = \|x - x'\|_p = \left(\sum_{j=1}^{d} \left| x_j - x_j' \right|^p \right)^{\frac{1}{p}}$

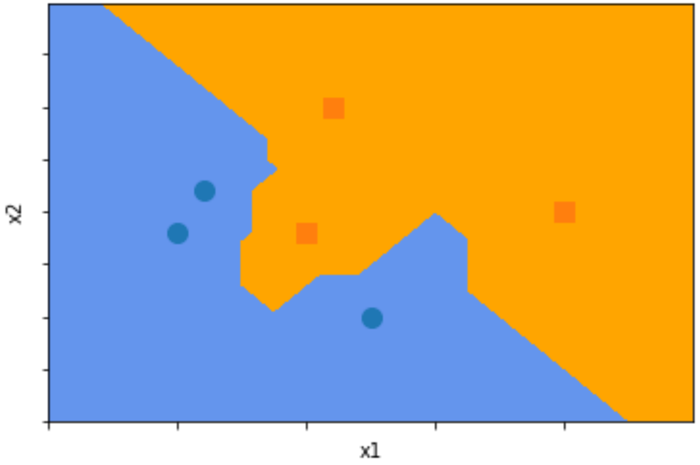
Choice of Distance Function



L_1 distance



L_2 distance



L_∞ distance

Distances for Strings

- **Hamming distance:** Number of characters that are different
  - **Example:** ABCDE vs. AGDDF → Hamming distance $= 3$
  - Assumes strings have the same length
- **Edit distance:** Number of insert/delete/replace operations needed to transform one string into the other
  - **Example:** ROBOT vs. BOT → Edit distance $= 2$
  - Can be computed using dynamic programming

Distances for Strings

- **Jaccard distance** between sets $1 - \frac{|A \cap B|}{|A \cup B|}$
  - Apply to set of $\boldsymbol{n}$**-grams** (i.e., n-character substrings)
  - **Example:** ROBOT vs. BOT yields

$$A = \{R, RO, ROB, OBO, BOT, OT, T\}$$
$$B = \{B, BO, BOT, OT, T\}$$

  - → Jaccard distance $= 1 - \frac{3}{9} = \frac{2}{3}$

Loss Minimization Framework

- **What is the model family?**
  - What are the “parameters”?

Loss Minimization Framework

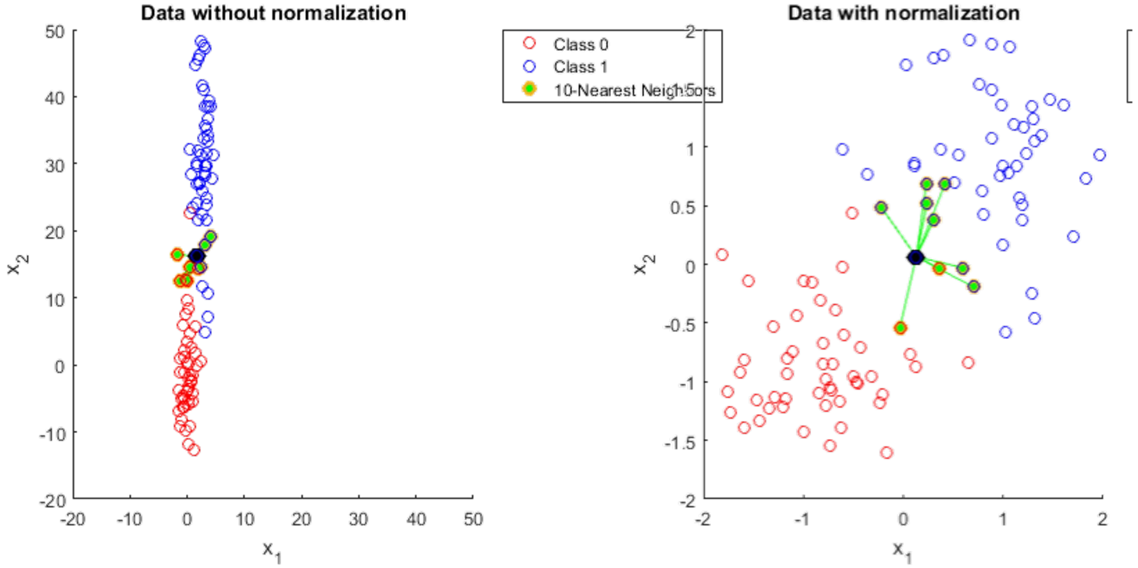
- **What is the model family?**
  - What are the “parameters”? The training dataset Z!
  - The model family is the set of kNN functions induced by Z
  - In general, “non-parametric” means the number of “parameters” scales with the number of training examples n

- **What is the loss function?**
  - kNN does not directly minimize a loss function
  - But, evaluate using standard losses (e.g., accuracy or MSE)

Feature Standardization

- We saw that feature standardization is not necessary for vanilla linear regression but is necessary for regularized linear regression

- It is very important for kNN!



https://stats.stackexchange.com/questions/287425/why-do-you-need-to-scale-data-in-knn

Curse of Dimensionality

- **Example:** Predict acceleration of an object being pushed by a robot

- Features:
  - $x_1 =$ mass
  - $x_2 =$ Force
  - $x_3 =$ color of object
  - $x_4 =$ what the operator ate for breakfast that morning

- When more irrelevant variables, distance function becomes dominated by irrelevant dimensions in x
  - Amount of data needed by kNN scales exponentially in dimension

Curse of Dimensionality

- Adding more dimensions makes a lot of things counterintuitive

- **Example:** The percentage of the volume of a d-dimensional sphere with radius r, that lies beyond ℓ_2 distance $0.99r$ from the center is:
  - 3% if $d = 3$
  - 63% if $d = 100$
  - 99.99% if $d = 1000$
  - **Intuition:** Volume inside radius scales as roughly 0.99^d

- For kNN, nearest neighbors become very far apart, and of similar distance, making it an **unreliable predictor**

Scalability

Scaling: Naively, must compute n distances between pairs of d-dimensional vectors to compute kNN

Scalability

- **Indexing**
  - Use kd-trees and other multidimensional indices to capture the training data
  - Each lookup is O(log *n*) but on disk
- **Parallelism**
  - Use multiple cores, and compare against in-memory data or kd trees
  - E.g., PANDA, LBL
- **Approximation**
  - Compare against a sample, not all of the training data
  - See, e.g., https://www.kaggle.com/code/pawanbhandarkar/knn-vs-approximate-knn-what-s-the-difference/notebook

kNNs

- **Strengths**
  - Very simple algorithm
  - Nonparametric, can learn complex decision boundaries

- **Weaknesses**
  - Requires an enormous (exponential) amount of data in high dimensions
  - Evaluating the model at test time scales with size of training data

- **Modern usage:** Look up nearest examples according to “learned” features (in the context of deep learning)

Lecture 8: Decision Trees

CIS 4190/5190

Fall 2023

Decision Trees

- Much more practical nonparametric learning algorithm
    - **Idea:** Can fit complex decision boundaries, but learns simpler ones first
- **Interpretable models:** Humans can look at the decision tree and understand what it is doing

Example: Diabetes Prediction

Over the years, I've collected data from lots of patients, recording their physical information, their demographic information, habits, and done their lab work to diagnose diabetes. I'm wondering now: from all this data, could I model the risk of other people with similar characteristics having diabetes given all this other information about them? And would your applied ML class be able to help? I've attached the data here for you to take a look.

Eventually, we'll want to explain our findings to patients, and point out any behavioral changes that would mitigate their risk for diabetes. Even if the risk factors we find are non-modifiable, insurance companies would be interested in understanding and estimating this risk. Either way, it'd be great to have something that we can understand and interpret well!

Example: Diabetes Prediction

SEQN	RIDAGEYR	BMXWAIST	BMXHT	LBXTC	BMXLEG	BMXWT	BMXBMI	RIDRETH1	BPQ020	ALQ120Q	DMDEDUC2	RIAGENDR	INDFMPIR	LBXGH	DIABETIC
73557	69.0	100.0	171.3	167.0	39.2	78.3	26.7	Non-Hispanic Black	yes	1.0	high school graduate / GED	male	0.84	13.9	yes
73558	54.0	107.6	176.8	170.0	40.0	89.5	28.6	Non-Hispanic White	yes	7.0	high school graduate / GED	male	1.78	9.1	yes
73559	72.0	109.2	175.3	126.0	40.0	88.9	28.9	Non-Hispanic White	yes	0.0	some college or AA degree	male	4.51	8.9	yes
73562	56.0	123.1	158.7	226.0	34.2	105.0	41.7	Mexican American	yes	5.0	some college or AA degree	male	4.79	5.5	no
73564	61.0	110.8	161.8	168.0	37.1	93.4	35.7	Non-Hispanic White	yes	2.0	college graduate or above	female	5.0	5.5	no
73566	56.0	85.5	152.8	278.0	32.4	61.8	26.5	Non-Hispanic White	no	1.0	high school graduate / GED	female	0.48	5.4	no
73567	65.0	93.7	172.4	173.0	40.0	65.3	22.0	Non-Hispanic White	no	4.0	9th-11th grade	male	1.2	5.2	no
73568	26.0	73.7	152.5	168.0	34.4	47.1	20.3	Non-Hispanic White	no	2.0	college graduate or above	female	5.0	5.2	no
73571	76.0	122.1	172.5	167.0	35.5	102.4	34.4	Non-Hispanic White	yes	2.0	college graduate or above	male	5.0	6.9	yes
73577	32.0	100.0	166.2	182.0	36.5	79.7	28.9	Mexican American	no	20.0	Less than 9th grade	male	0.29	5.3	no
73581	50.0	99.3	185.0	202.0	42.8	80.9	23.6	Other or Multi-Racial	no	0.0	college graduate or above	male	5.0	5.0	no
73585	28.0	90.3	175.1	198.0	40.5	92.2	30.1	Other or Multi-Racial	no	4.0	some college or AA degree	male	2.26	5.0	no
73589	35.0	94.6	172.9	192.0	39.1	78.3	26.2	Non-Hispanic White	no	2.0	high school graduate / GED	male	1.74	5.5	no
73595	58.0	114.8	175.3	165.0	40.1	96.0	31.2	Other Hispanic	no	1.0	some college or AA degree	male	3.09	7.7	no
73596	57.0	117.8	164.7	151.0	35.3	104.0	38.3	Other or Multi-Racial	yes	1.0	college graduate or above	female	5.0	5.9	no
73600	37.0	122.9	185.1	189.0	48.1	126.2	36.8	Non-Hispanic Black	yes	2.0	high school graduate / GED	male	0.63	6.2	yes
73604	69.0	96.6	156.9	203.0	37.0	59.5	24.2	Non-Hispanic White	no	1.0	some college or AA degree	female	2.44	5.4	no
73607	75.0	130.5	169.6	161.0	36.5	111.9	38.9	Non-Hispanic White	yes	0.0	high school graduate / GED	male	1.08	5.0	no
73610	43.0	102.6	176.8	200.0	38.8	90.2	28.9	Non-Hispanic White	no	5.0	college graduate or above	male	2.03	4.9	no
73613	60.0	113.6	163.8	203.0	41.6	104.9	39.1	Non-Hispanic Black	yes	2.0	9th-11th grade	female	5.0	6.1	no
73614	55.0	90.9	167.9	256.0	43.5	60.9	21.6	Non-Hispanic White	no	0.0	high school graduate / GED	female	1.29	5.0	no
73615	65.0	100.3	145.9	166.0	30.0	55.4	26.0	Other Hispanic	yes	1.0	Less than 9th grade	female	1.22	6.3	yes

Inputs x_i Labels y_i

Aside: Data Dictionaries

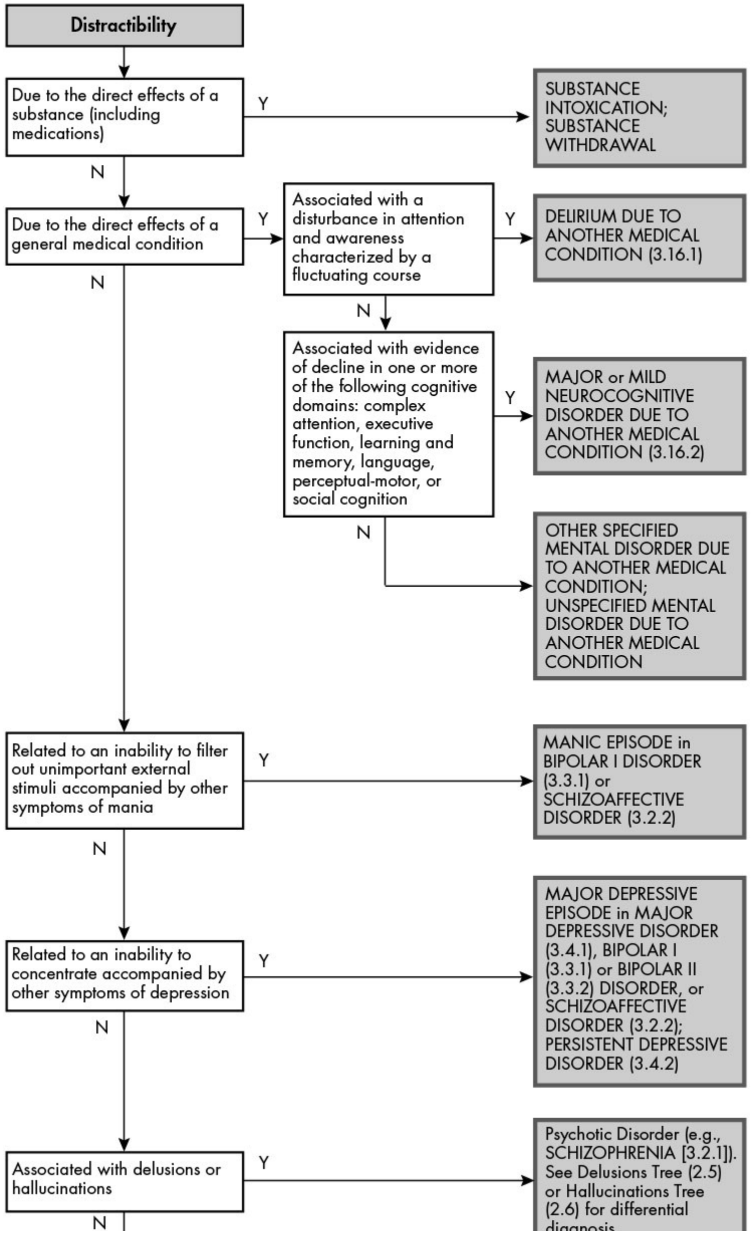
- Datasets are often accompanied by a **data dictionary**
  - Describes each column in the dataset
  - Important to understand dataset before doing any machine learning!
  - E.g., which columns have missing values

- The dictionary for our data: https://wwwn.cdc.gov/nchs/nhanes/Default.aspx

Decision Trees

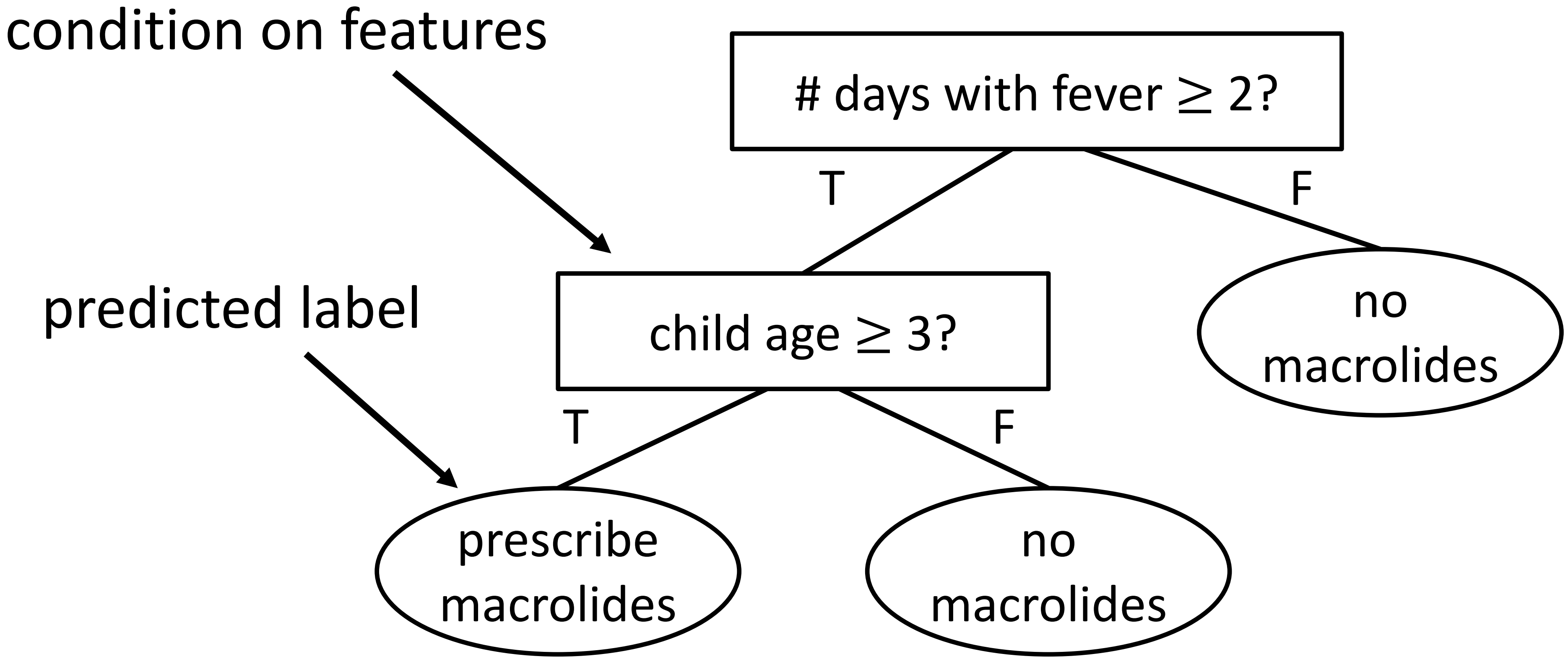
- A kind of flowchart based on tests
  - Commonly used in medicine

- “Explainable”, easy to mentally evaluate



APA DSM Library

Decision Trees



Decision tree example from: Martignon and Monti. (2010). Conditions for risk assessment as a topic for probabilistic education. *Proceedings of the Eighth International Conference on Teaching Statistics* (ICOTS8).

Decision Trees

- Binary tree

- Each **internal node** has a Boolean condition that is a function of x
  - Typically reference a single feature x_j
  - **Real-valued feature:** Condition $1(x_j \geq t)$ (where $t \in \mathbb{R}$)
  - **Categorical feature:** Condition $1(x_j = t)$ (where $t \in \{1, \dots, k_j\}$ is a category)

- Each **leaf node** is a label
  - Can be either regression or classification
  - Can also be a probability distribution

Intuition: Dataset Splitting

- Internal nodes split the dataset

ColorOfCoat	TypeOfHorse
black	thoroughbred
bay	Arabian
black	thoroughbred
chestnut	quarter
black	Arabian

N=5; 3 classes

ColorOfCoat = “black”

ColorOfCoat	TypeOfHorse
black	thoroughbred
black	thoroughbred
black	Arabian

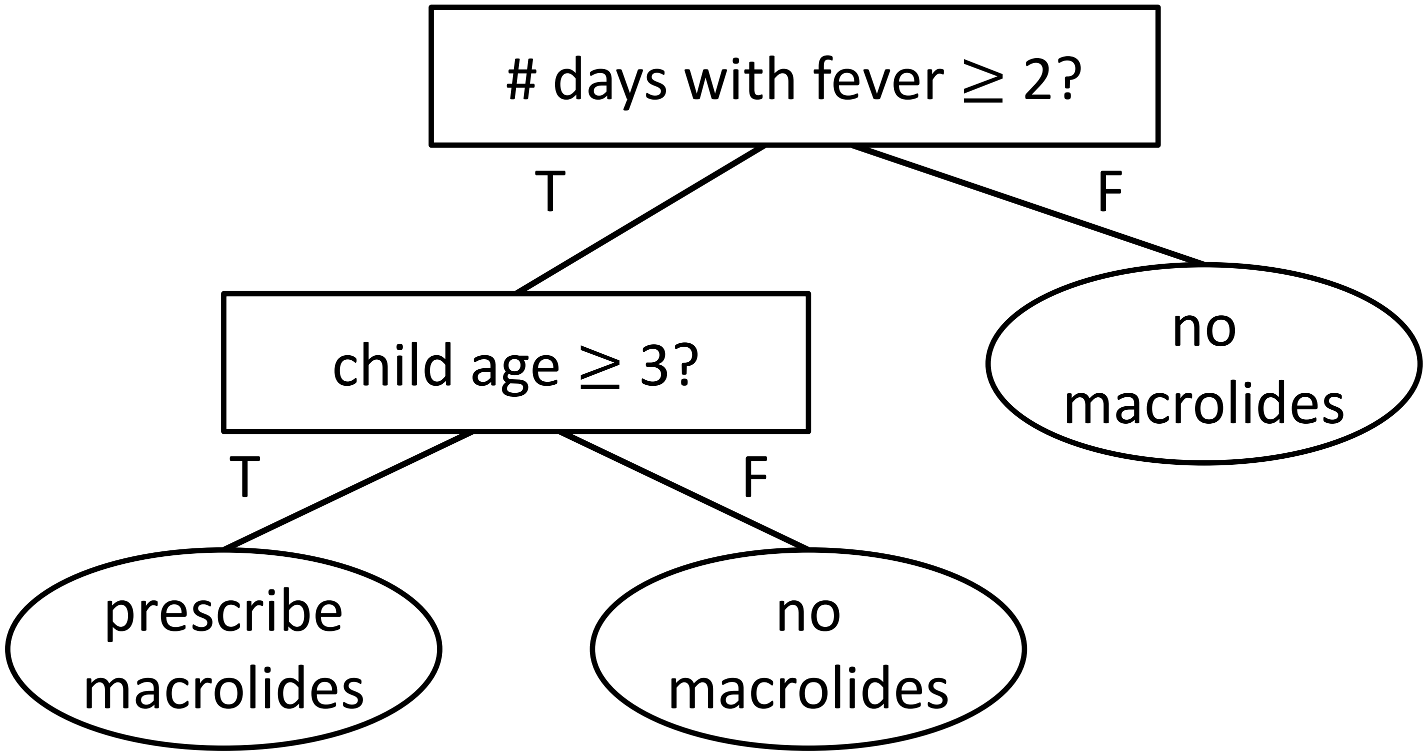
N=3; 2 classes

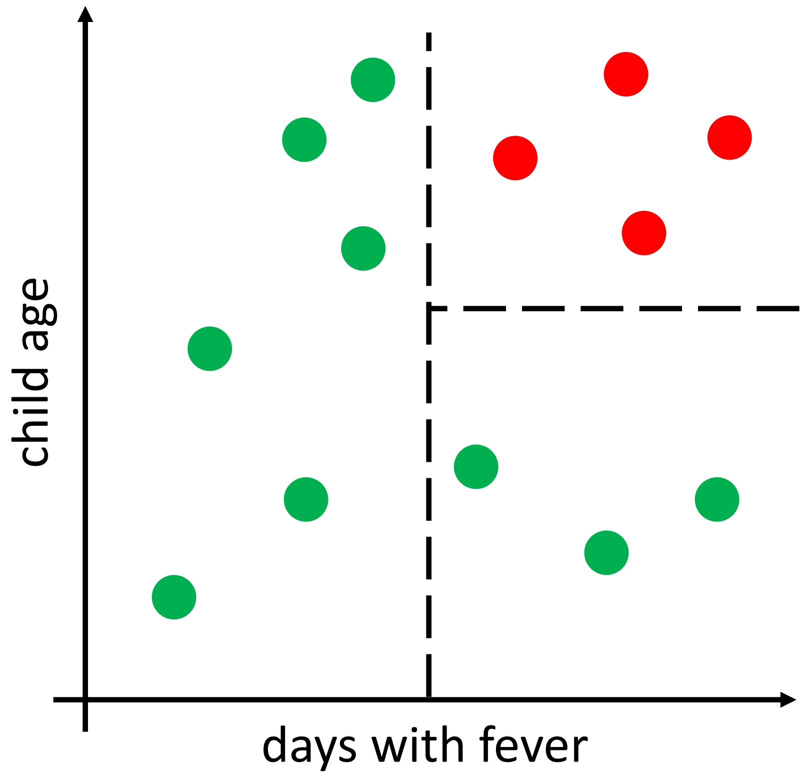
ColorOfCoat	TypeOfHorse
bay	Arabian
chestnut	quarter

N=2; 2 classes

Visualizing the Model Family

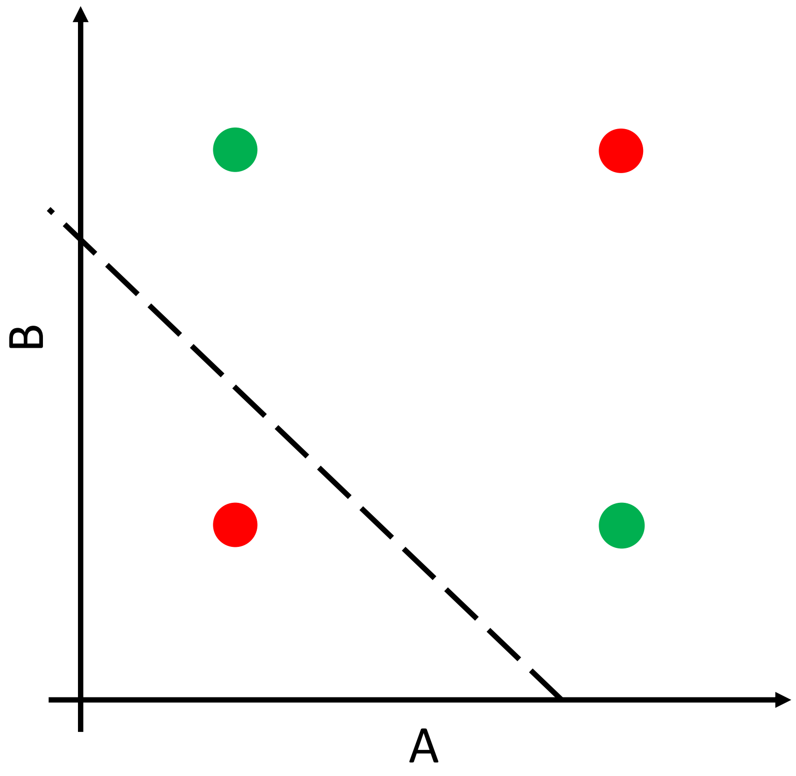
- Axis-aligned decision boundaries





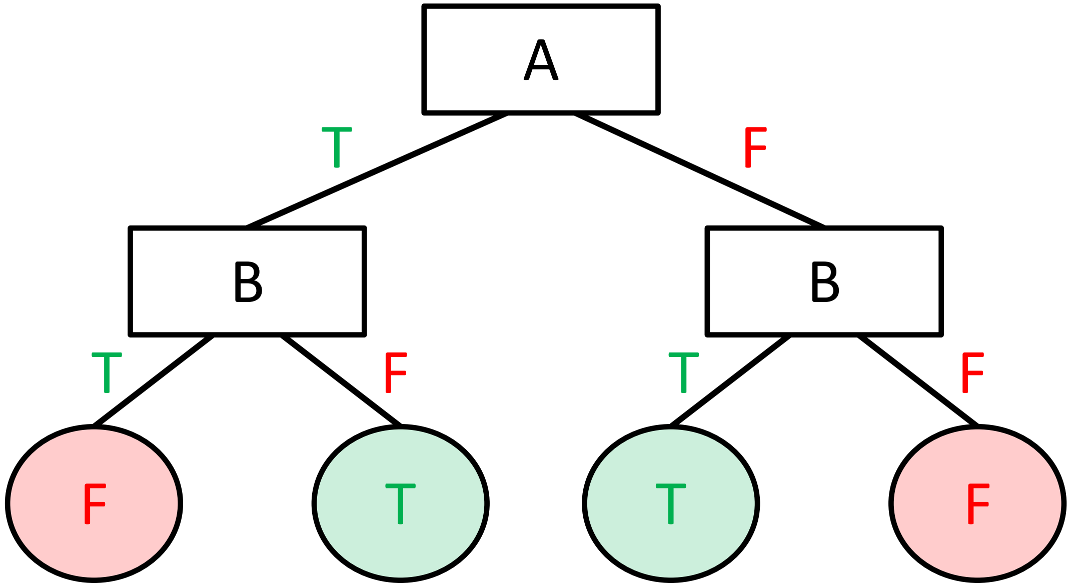
Decision Trees and XOR

A	B	A XOR B
T	T	F
T	F	T
F	T	T
F	F	F



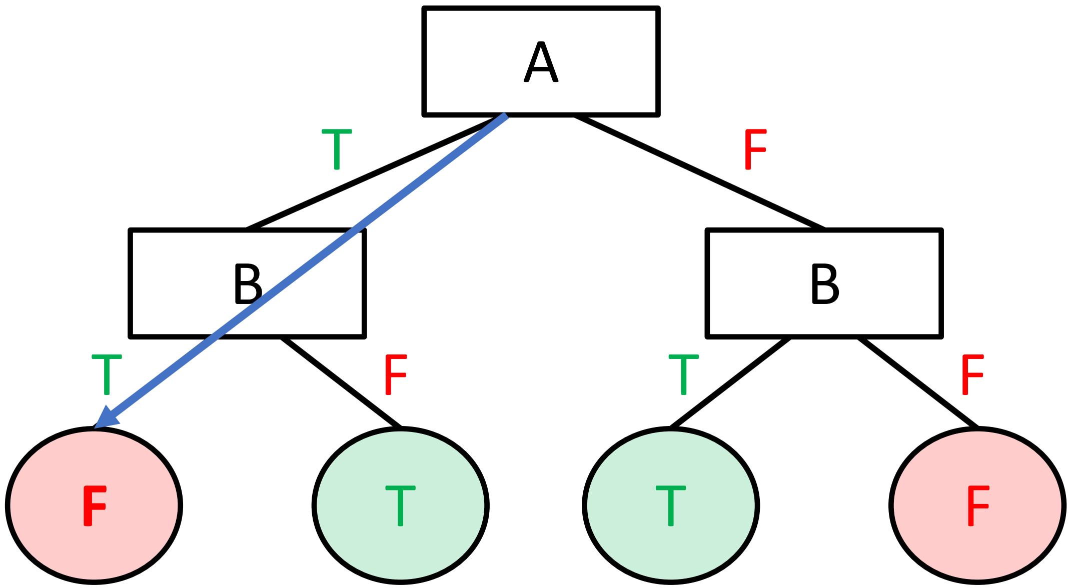
Decision Trees and XOR

A	B	A XOR B
T	T	F
T	F	T
F	T	T
F	F	F



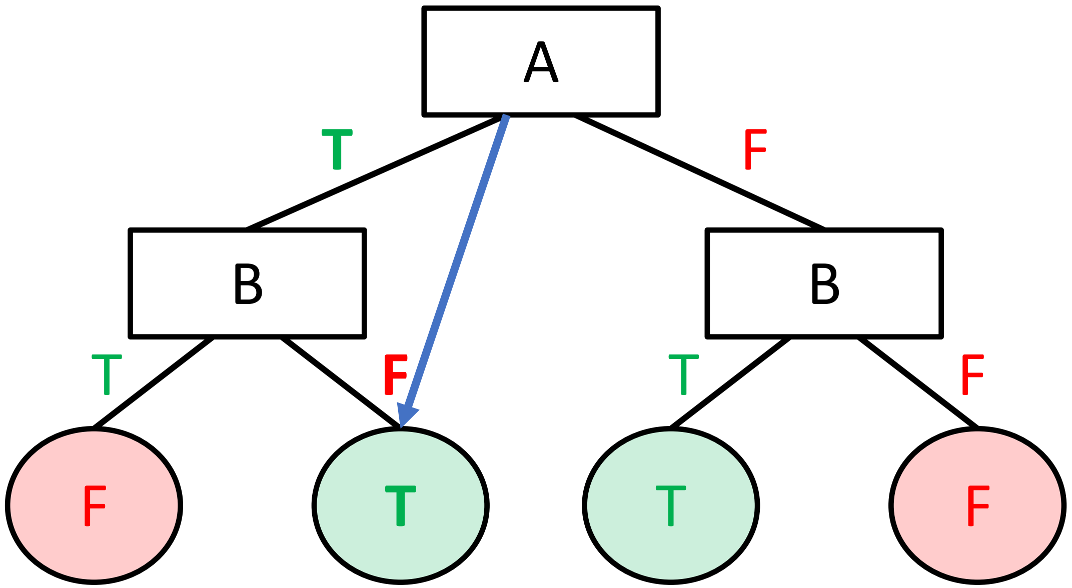
Decision Trees and XOR

A	B	A XOR B
T	**T**	**F**
T	F	T
F	T	T
F	F	F



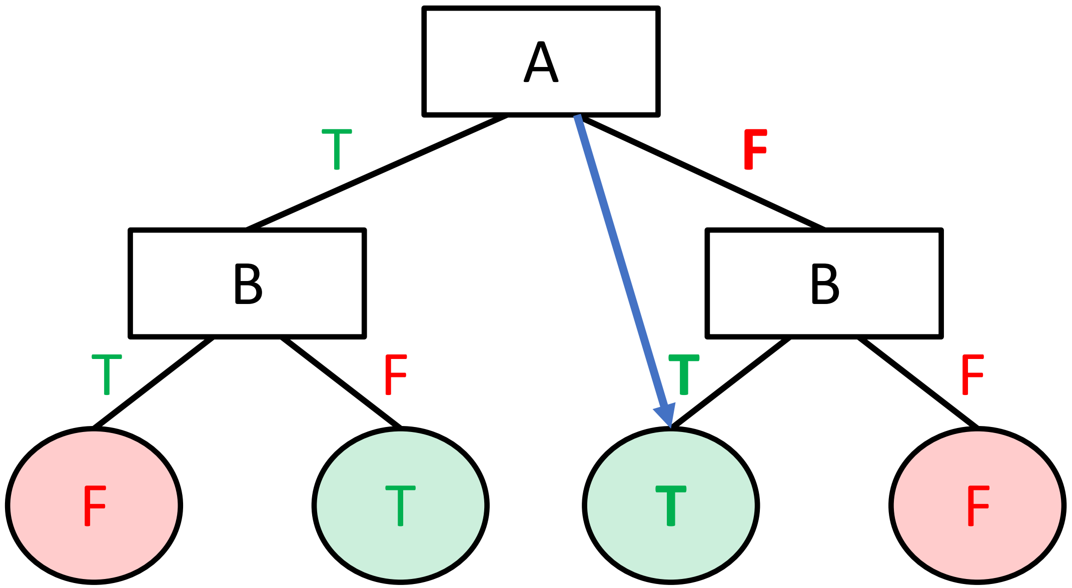
Decision Trees and XOR

A	B	A XOR B
T	T	F
T	F	T
F	T	T
F	F	F



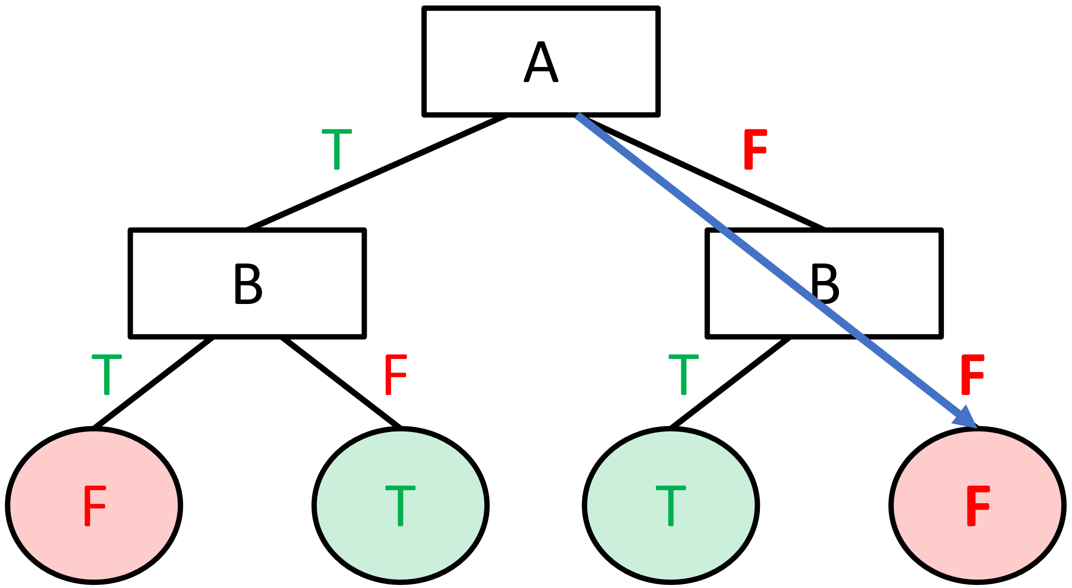
Decision Trees and XOR

A	B	A XOR B
T	T	F
T	F	T
F	T	T
F	F	F

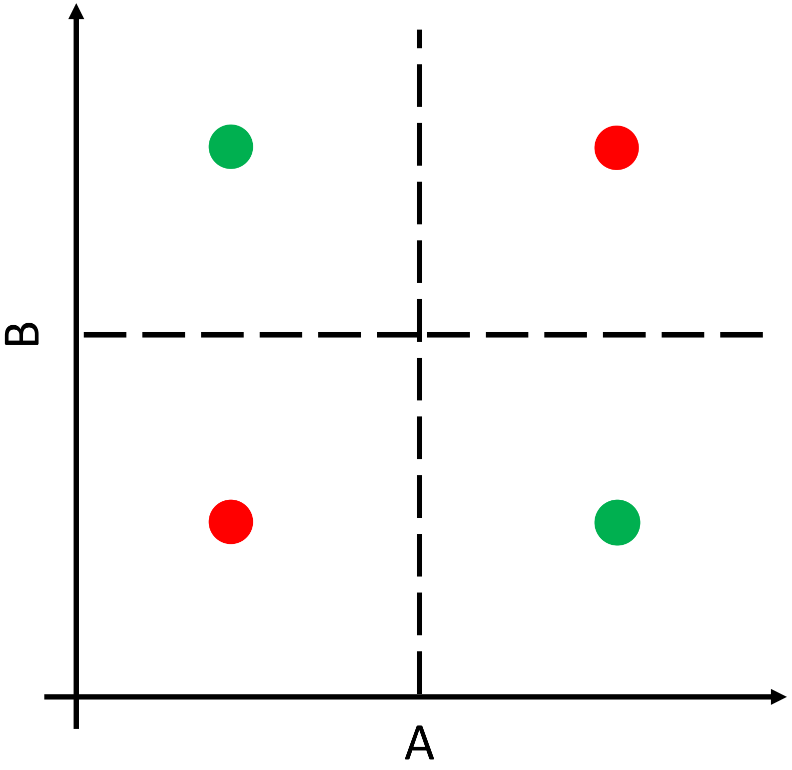


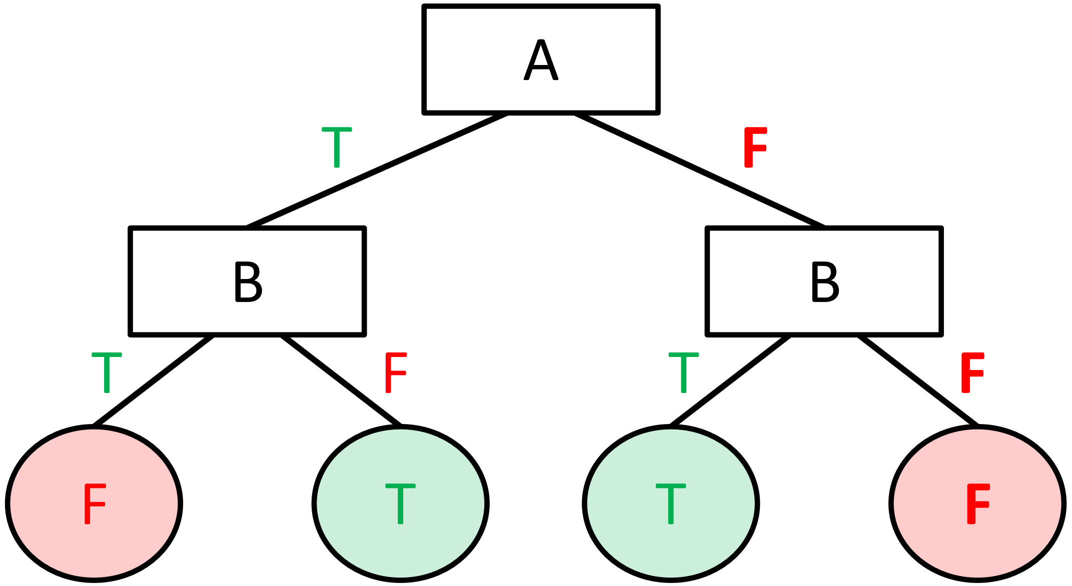
Decision Trees and XOR

A	B	A XOR B
T	T	F
T	F	T
F	T	T
F	**F**	**F**



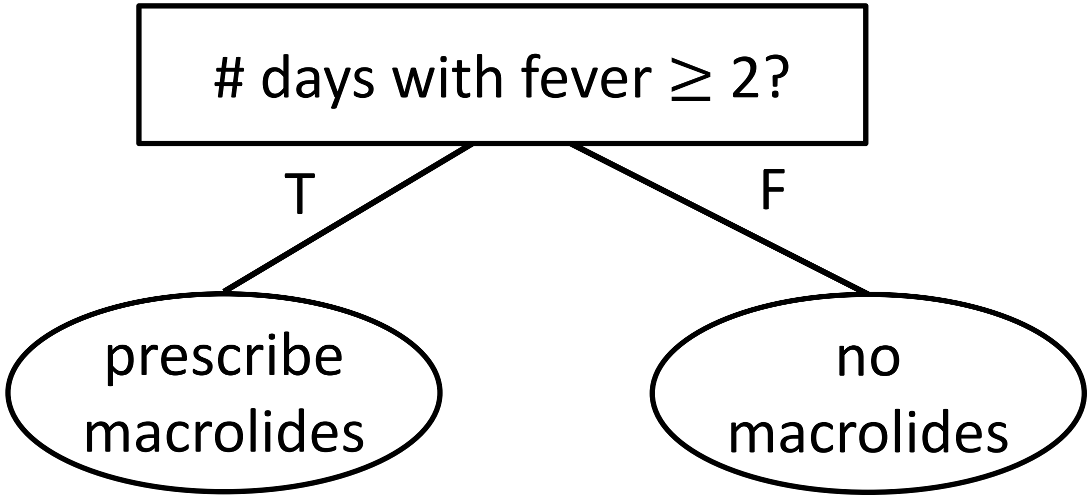
Decision Trees and XOR

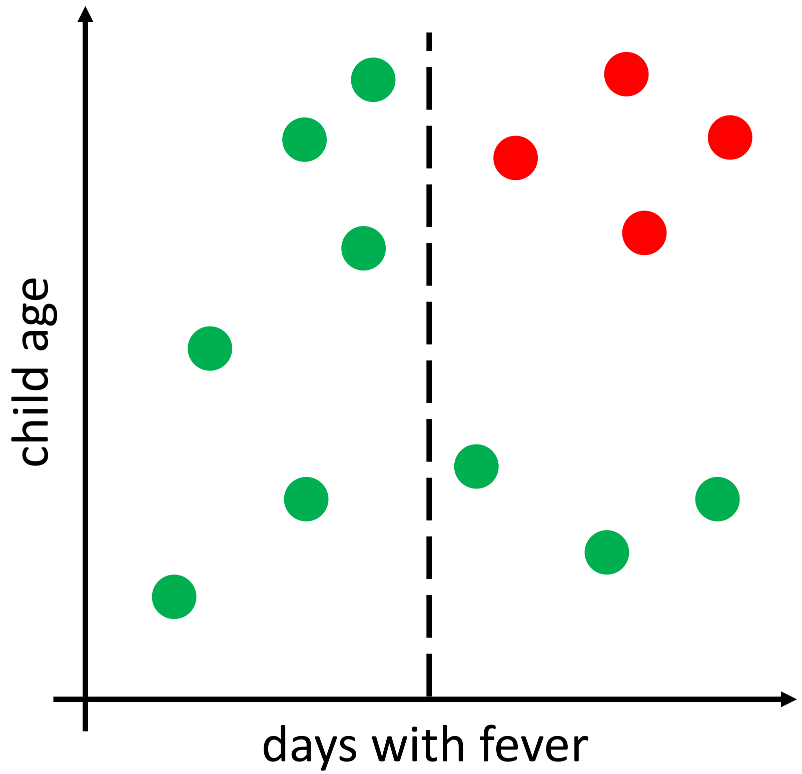




Decision Boundary and Depth

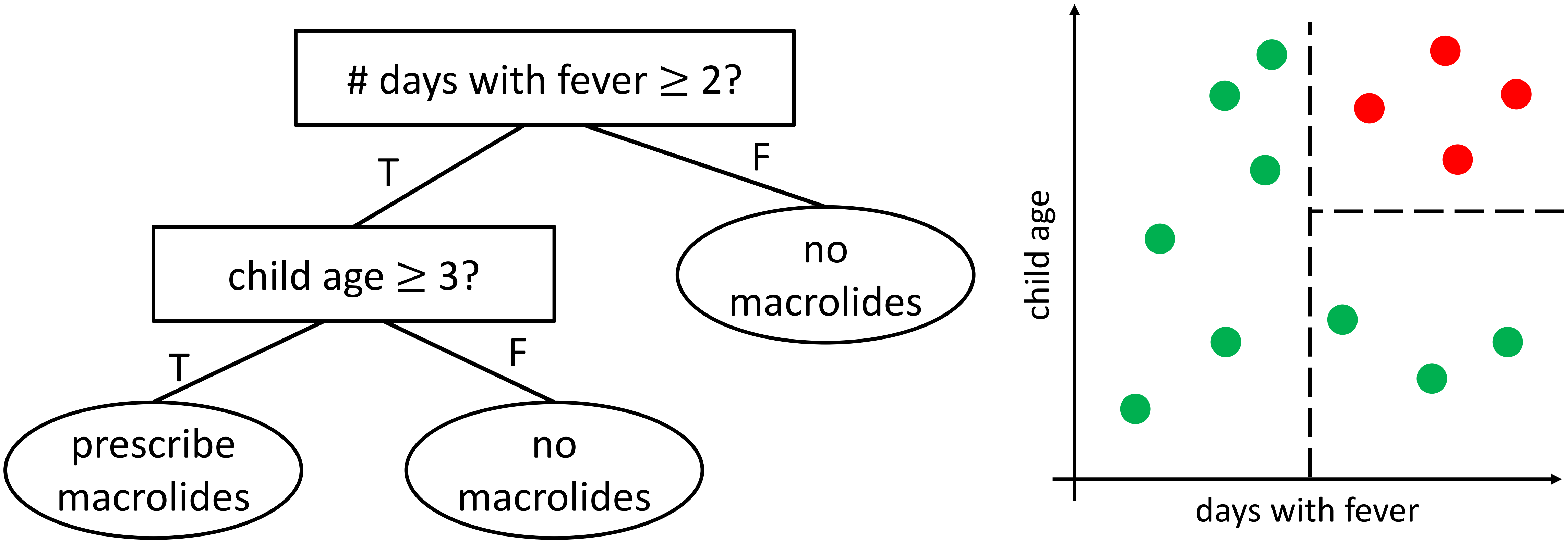
- Complexity increases with depth





Decision Boundary and Depth

- Complexity increases with depth

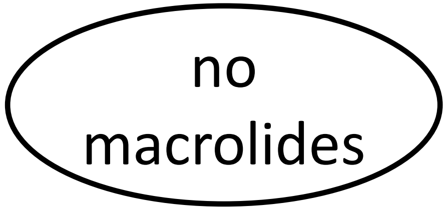


Learning Algorithm

- Similar to kNN, traditional decision tree learning algorithms do not fit in the loss minimization framework
  - Computing the optimal decision tree is NP complete
  - Recent work has tried to devise more efficient algorithms

- Instead, they are heuristically constructed in a top-down fashion

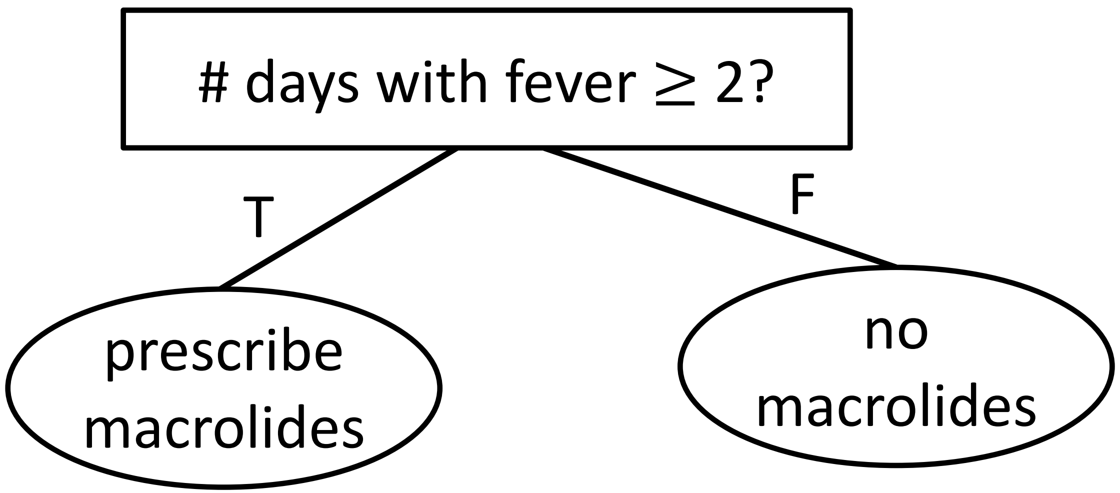
Learning Algorithm



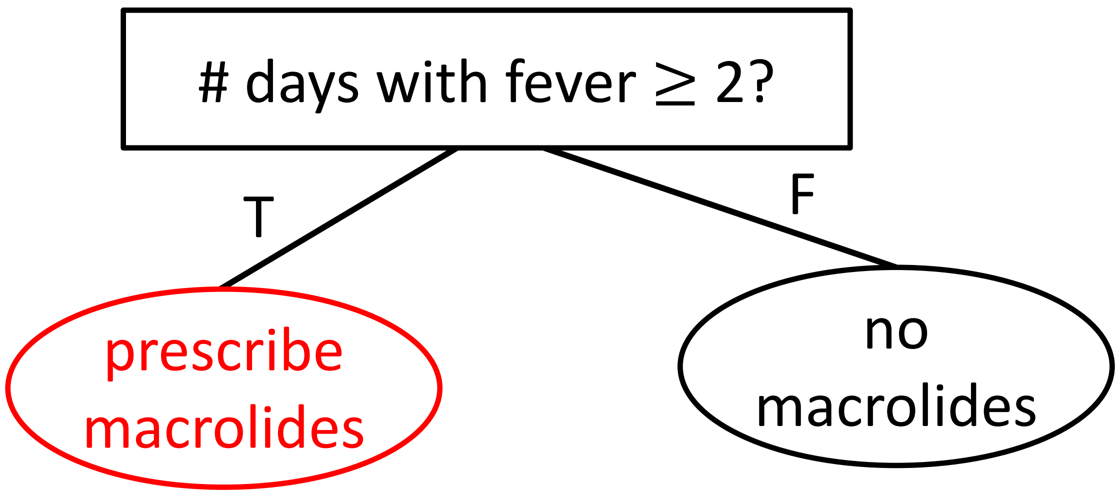
Learning Algorithm

no macrolides

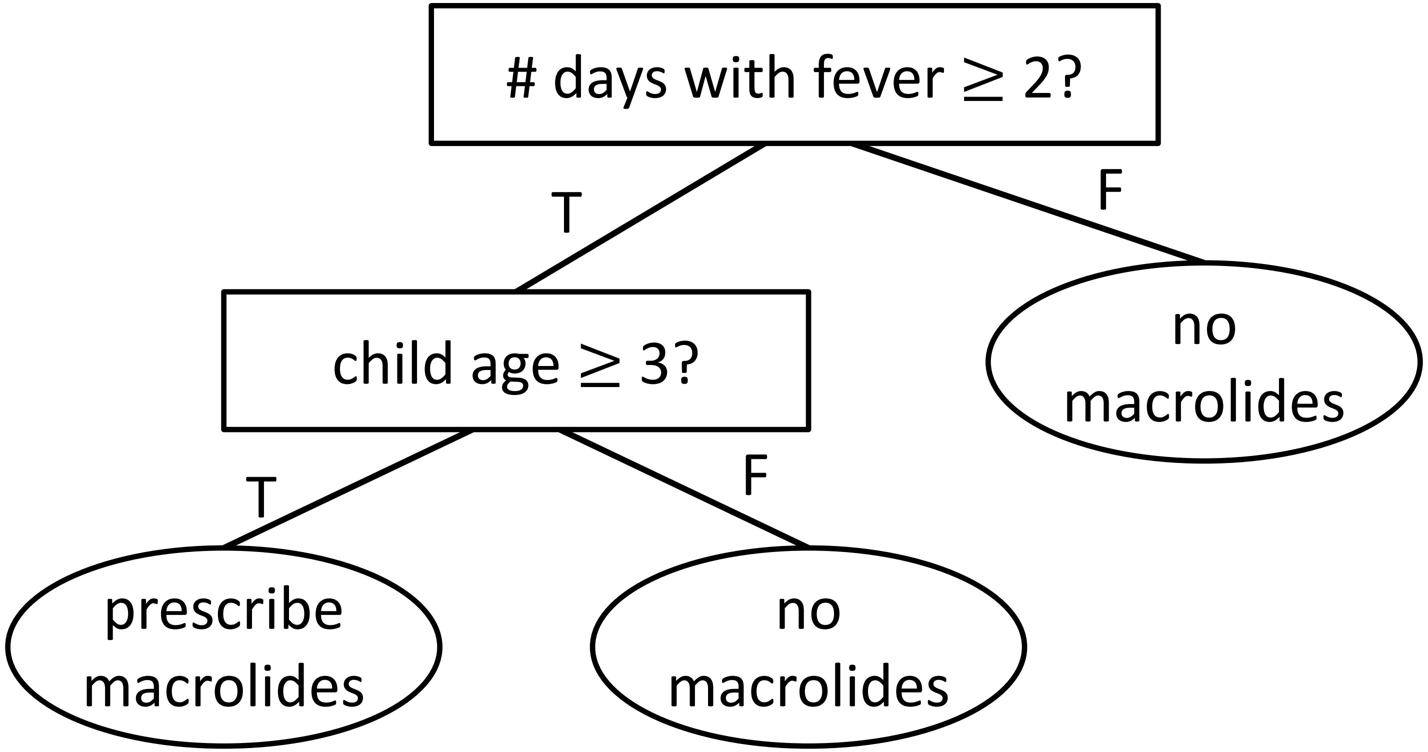
Learning Algorithm



Learning Algorithm



Learning Algorithm



Learning Algorithm

- Let $Z[C] = \{\, (x, y) \in Z \mid C(x, y) \,\}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

    return LeafNode(y)

  $(j, t) \leftarrow \mathrm{BestSplit}(Z)$

  $T_{\text{left}} \leftarrow \mathrm{LearnTree}\big(Z[x_j \geq t]\big)$

  $T_{\text{right}} \leftarrow \mathrm{LearnTree}\big(Z[x_j < t]\big)$

  return $\mathrm{InternalNode}\big(j, t, T_{\text{left}}, T_{\text{right}}\big)$

Z

Learning Algorithm

- Let $Z[C] = \{ (x, y) \in Z \mid C(x, y) \}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

    return LeafNode(y)

  $\boxed{(j, t) \leftarrow \text{BestSplit}(Z)}$

  $T_{\text{left}} \leftarrow \text{LearnTree}\big(Z[x_j \geq t]\big)$

  $T_{\text{right}} \leftarrow \text{LearnTree}\big(Z[x_j < t]\big)$

  return InternalNode$\big(j, t, T_{\text{left}}, T_{\text{right}}\big)$

Z

$\boxed{\text{\# days with fever} \geq 2?}$

Learning Algorithm

- Let $Z[C] = \{\, (x, y) \in Z \mid C(x, y) \,\}$ be the subset of Z where C holds

def LearnTree(Z):

    if all labels in Z are the same and equal y:

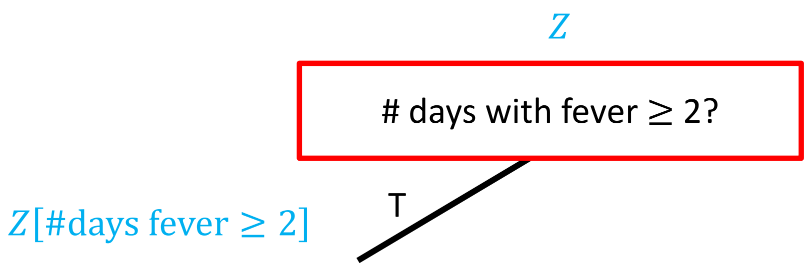
        return LeafNode(y)

    $(j, t) \leftarrow \text{BestSplit}(Z)$

    $T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])$

    $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

    return InternalNode$(j, t, T_{\text{left}}, T_{\text{right}})$



Learning Algorithm

- Let $Z[C] = \{\,(x, y) \in Z \mid C(x, y)\,\}$ be the subset of Z where C holds

def LearnTree(Z):

    if all labels in Z are the same and equal y:

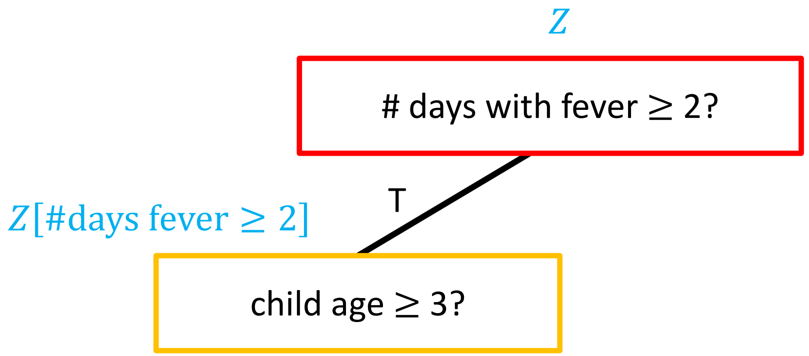
        return LeafNode(y)

    $(j, t) \leftarrow \text{BestSplit}(Z)$

    $T_{\text{left}} \leftarrow \text{LearnTree}\big(Z[x_j \geq t]\big)$

    $T_{\text{right}} \leftarrow \text{LearnTree}\big(Z[x_j < t]\big)$

    return $\text{InternalNode}\big(j, t, T_{\text{left}}, T_{\text{right}}\big)$



Learning Algorithm

- Let $Z[C] = \{\, (x, y) \in Z \mid C(x, y) \,\}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

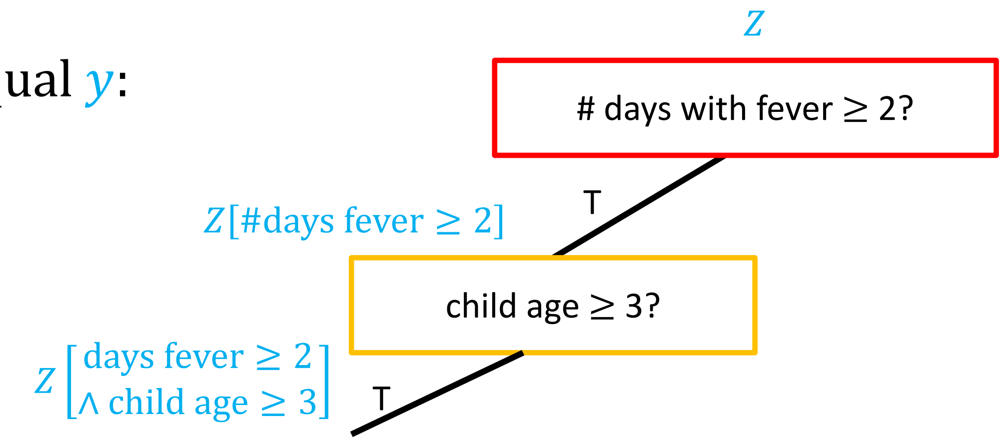
    return LeafNode(y)

  $(j, t) \leftarrow$ BestSplit(Z)

  $T_{\text{left}} \leftarrow \text{LearnTree}\left(Z[x_j \geq t]\right)$

  $T_{\text{right}} \leftarrow \text{LearnTree}\left(Z[x_j < t]\right)$

  return InternalNode$\left(j, t, T_{\text{left}}, T_{\text{right}}\right)$



Learning Algorithm

- Let $Z[C] = \{\,(x,y) \in Z \mid C(x,y)\,\}$ be the subset of Z where C holds

def LearnTree(Z):

    if all labels in Z are the same and equal y:

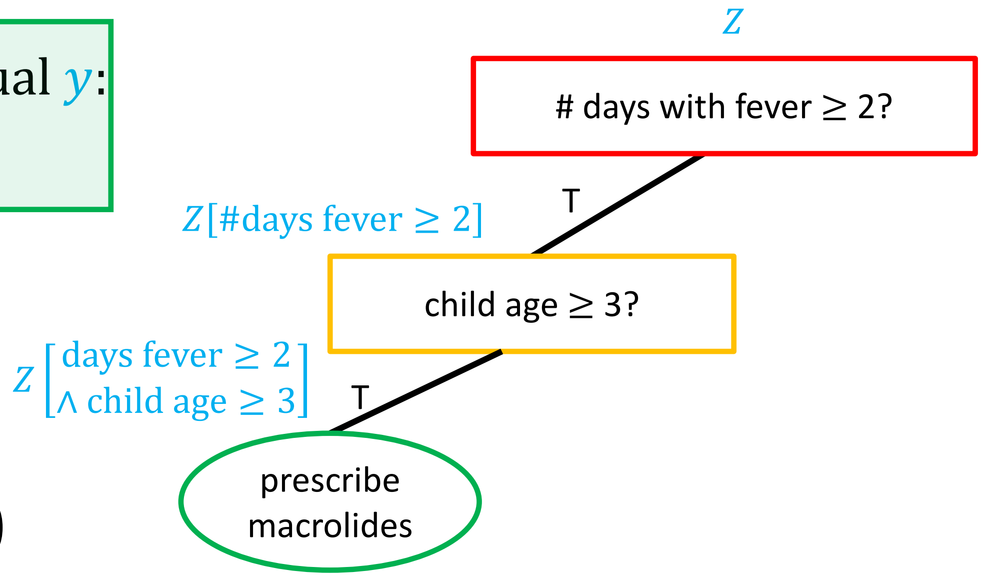
        return LeafNode(y)

    $(j,t) \leftarrow \mathrm{BestSplit}(Z)$

    $T_{\mathrm{left}} \leftarrow \mathrm{LearnTree}(Z[x_j \geq t])$

    $T_{\mathrm{right}} \leftarrow \mathrm{LearnTree}(Z[x_j < t])$

    return $\mathrm{InternalNode}(j, t, T_{\mathrm{left}}, T_{\mathrm{right}})$



Learning Algorithm

- Let $Z[C] = \{ (x, y) \in Z \mid C(x, y) \}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

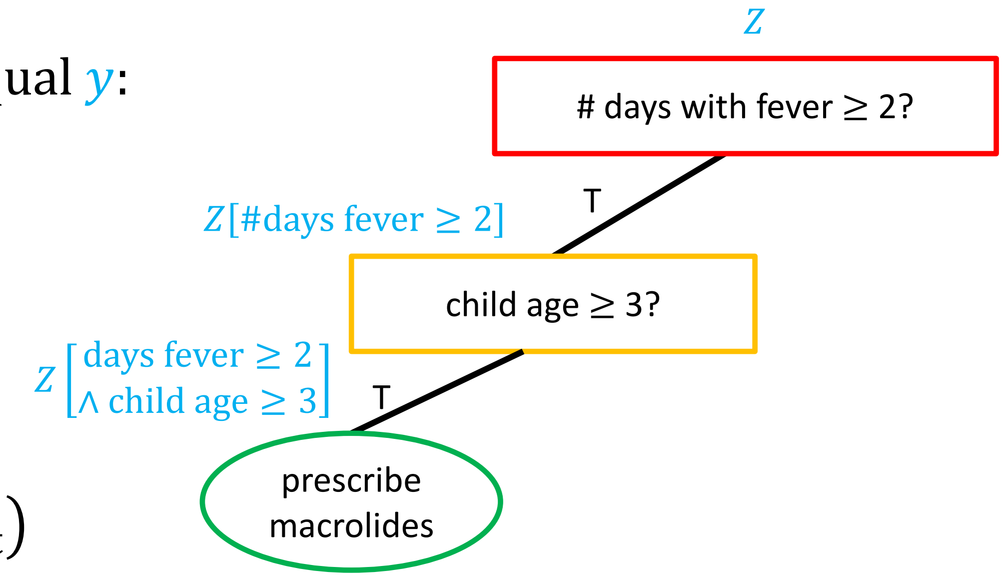
    return LeafNode(y)

  $(j, t) \leftarrow$ BestSplit(Z)

  $\boxed{T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])}$

  $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

  return InternalNode$(j, t, T_{\text{left}}, T_{\text{right}})$



Learning Algorithm

- Let $Z[C] = \{\,(x,y) \in Z \mid C(x,y)\,\}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

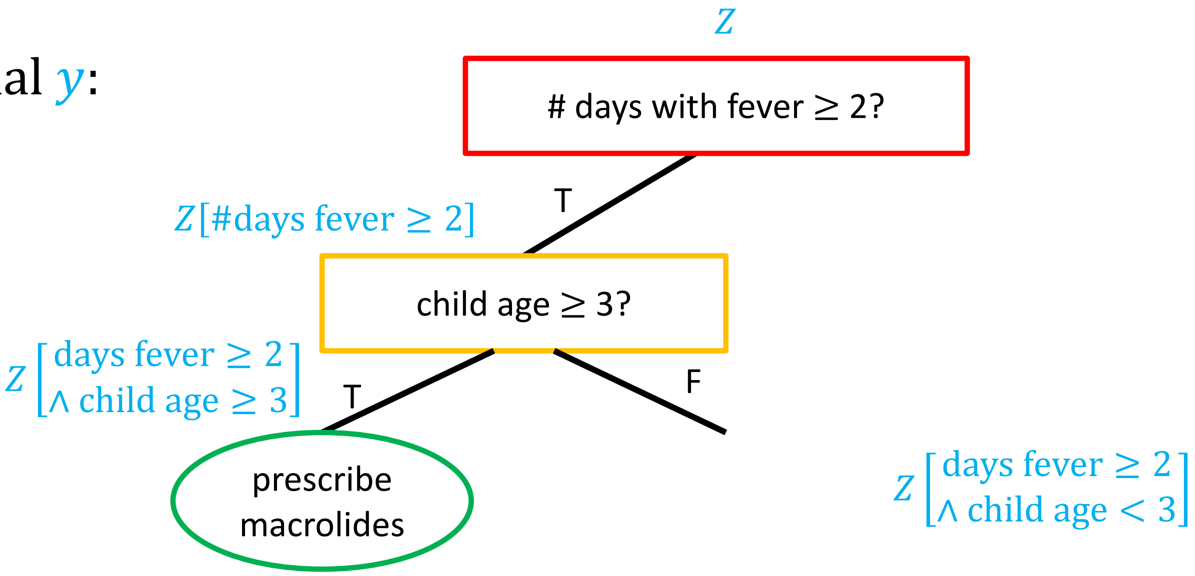
    return LeafNode(y)

  $(j,t) \leftarrow \text{BestSplit}(Z)$

  $T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])$

  $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

  return InternalNode$(j, t, T_{\text{left}}, T_{\text{right}})$



Learning Algorithm

- Let $Z[C] = \{\, (x, y) \in Z \mid C(x, y) \,\}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

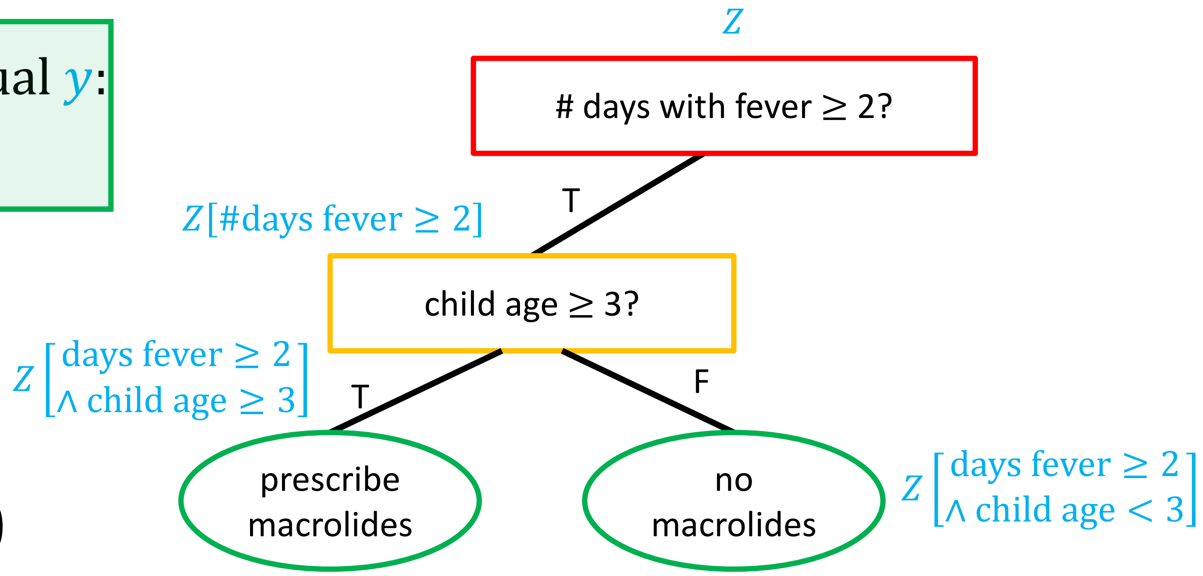
    return LeafNode(y)

  $(j, t) \leftarrow$ BestSplit(Z)

  $T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])$

  $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

  return InternalNode$(j, t, T_{\text{left}}, T_{\text{right}})$



Learning Algorithm

- Let $Z[C] = \{ (x, y) \in Z \mid C(x, y) \}$ be the subset of Z where C holds

def LearnTree(Z):

    if all labels in Z are the same and equal y:

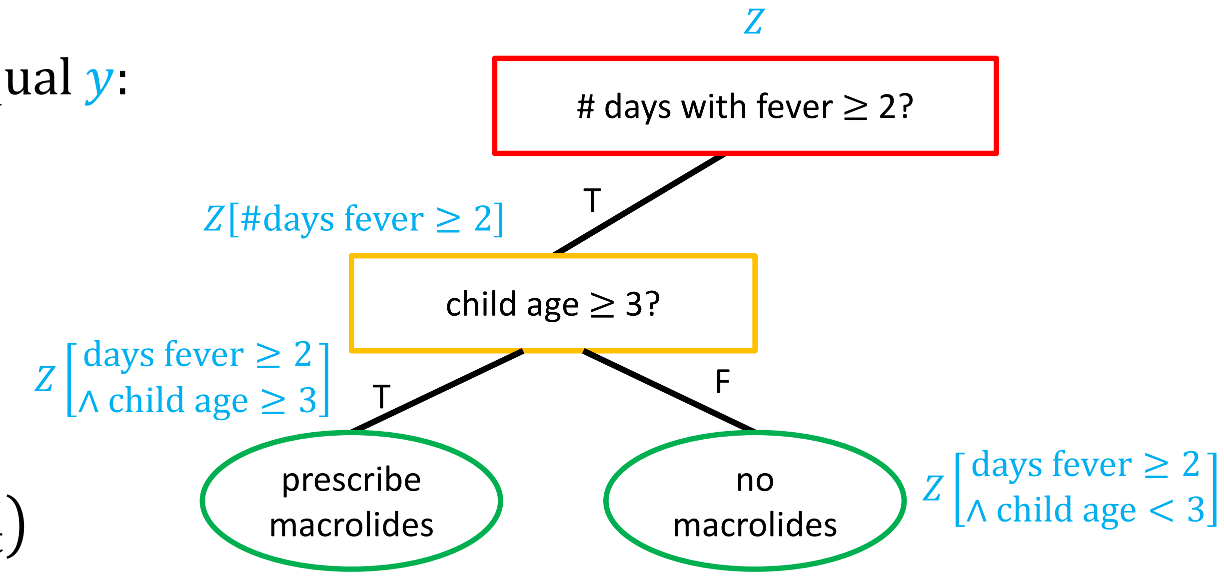
        return LeafNode(y)

    $(j, t) \leftarrow \text{BestSplit}(Z)$

    $T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])$

    $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

    return $\text{InternalNode}(j, t, T_{\text{left}}, T_{\text{right}})$



Learning Algorithm

- Let $Z[C] = \{\,(x, y) \in Z \mid C(x, y)\,\}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

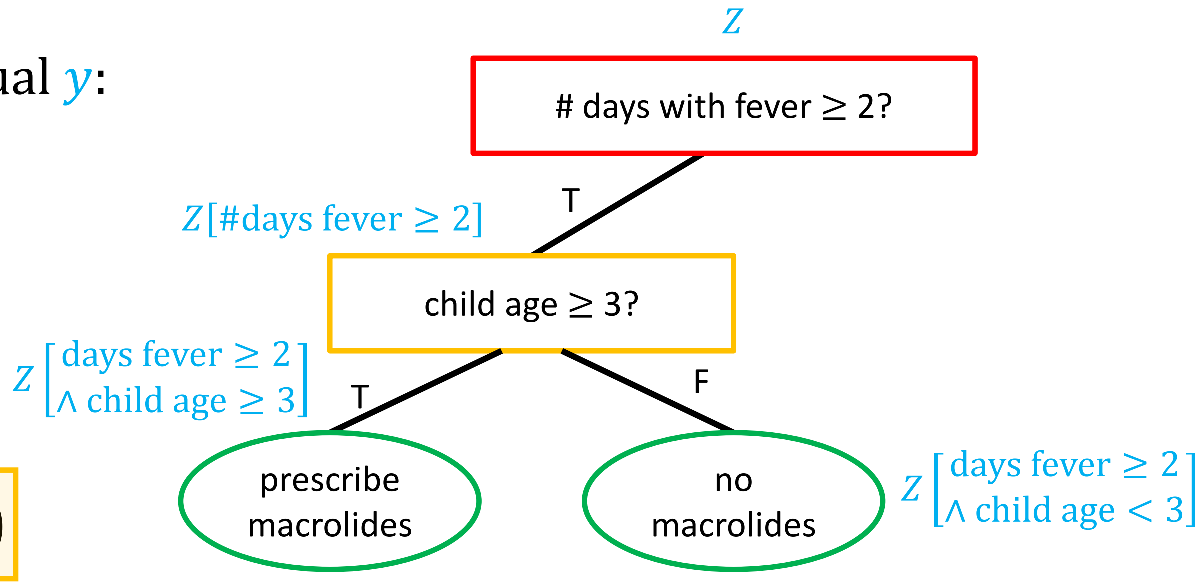
    return LeafNode(y)

  $(j, t) \leftarrow$ BestSplit(Z)

  $\boxed{T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])}$

  $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

  $\boxed{\textbf{return}\ \text{InternalNode}(j, t, T_{\text{left}}, T_{\text{right}})}$



Learning Algorithm

- Let $Z[C] = \{\, (x, y) \in Z \mid C(x, y) \,\}$ be the subset of Z where C holds

def LearnTree(Z):

    if all labels in Z are the same and equal y:

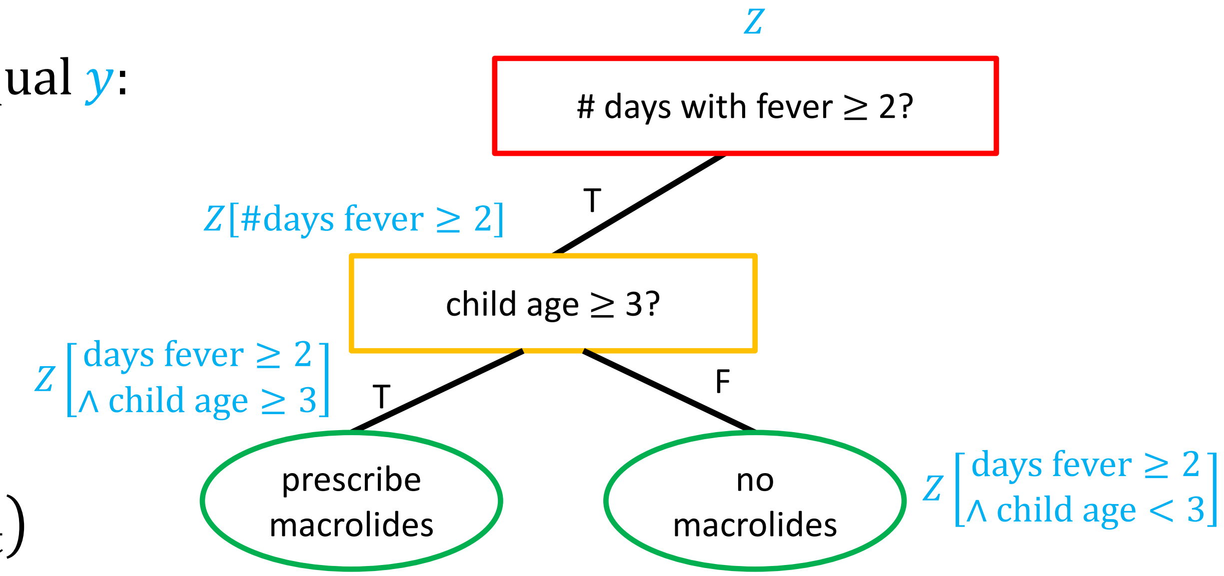
        return LeafNode(y)

    $(j, t) \leftarrow \text{BestSplit}(Z)$

    $\boxed{T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])}$

    $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

    return InternalNode$(j, t, T_{\text{left}}, T_{\text{right}})$



Learning Algorithm

- Let $Z[C] = \{ (x, y) \in Z \mid C(x, y) \}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

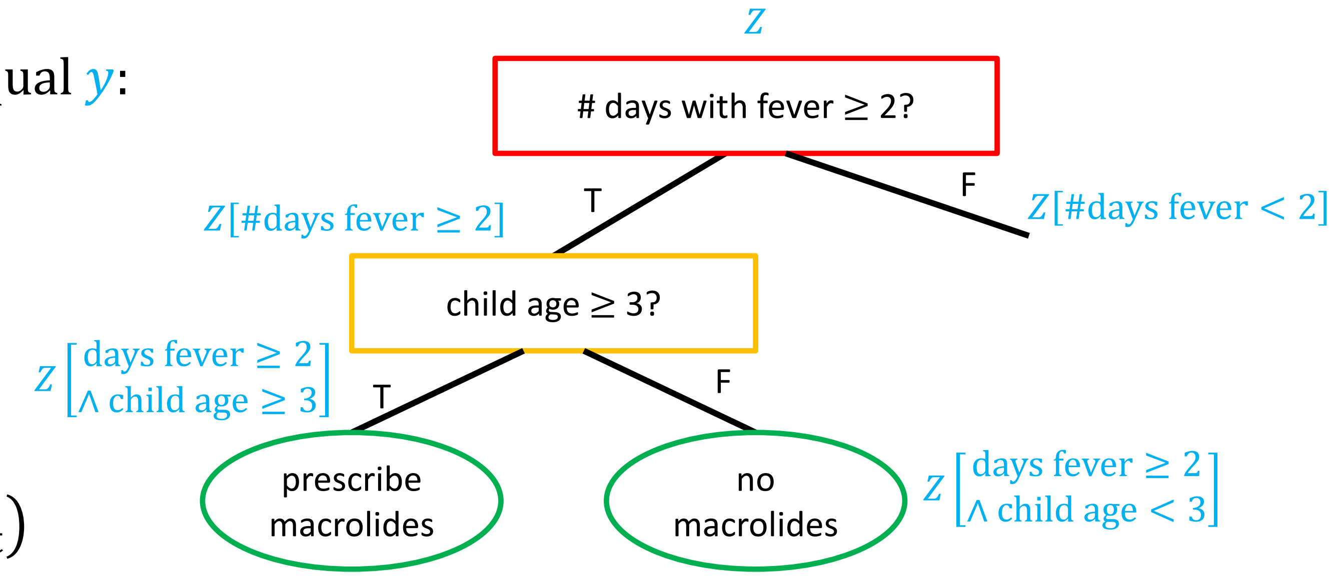
    return LeafNode(y)

  $(j, t) \leftarrow$ BestSplit(Z)

  $T_{\text{left}} \leftarrow$ LearnTree$\left(Z\left[x_j \geq t\right]\right)$

  $T_{\text{right}} \leftarrow$ LearnTree$\left(Z\left[x_j < t\right]\right)$

  return InternalNode$\left(j, t, T_{\text{left}}, T_{\text{right}}\right)$



Learning Algorithm

- Let $Z[C] = \{ (x, y) \in Z \mid C(x, y) \}$ be the subset of Z where C holds

def LearnTree(Z):

    if all labels in Z are the same and equal y:

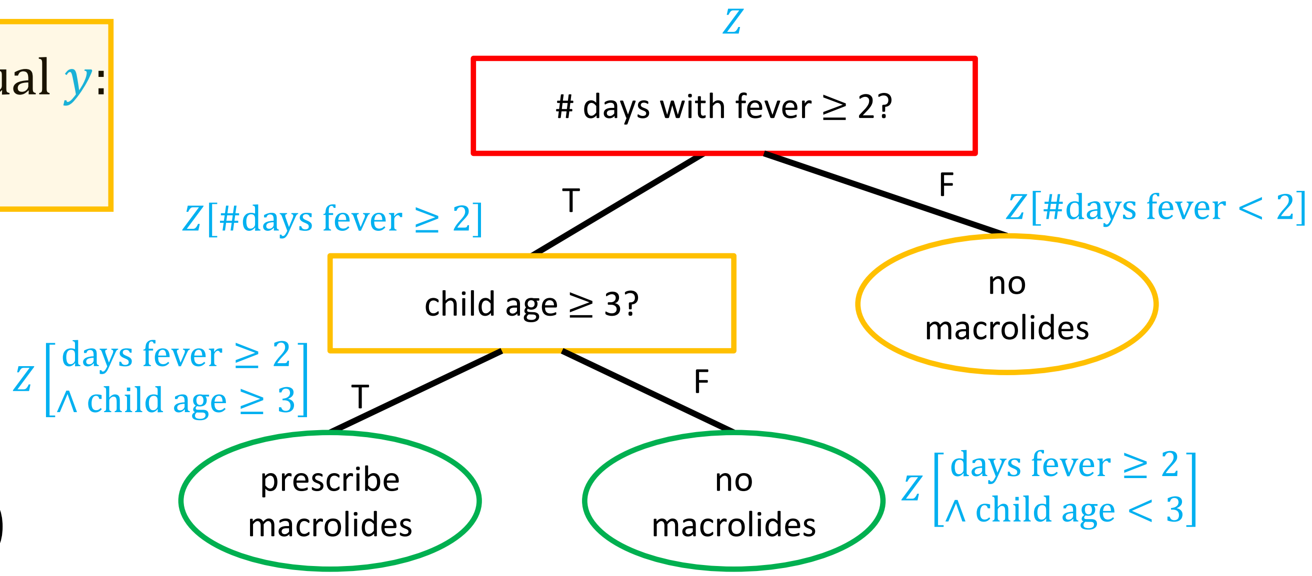
        return LeafNode(y)

    $(j, t) \leftarrow \text{BestSplit}(Z)$

    $T_{\text{left}} \leftarrow \text{LearnTree}\left(Z\left[x_j \geq t\right]\right)$

    $T_{\text{right}} \leftarrow \text{LearnTree}\left(Z\left[x_j < t\right]\right)$

    return $\text{InternalNode}\left(j, t, T_{\text{left}}, T_{\text{right}}\right)$



Learning Algorithm

- Let $Z[C] = \{\, (x, y) \in Z \mid C(x, y) \,\}$ be the subset of Z where C holds

def LearnTree(Z):

    if all labels in Z are the same and equal y:

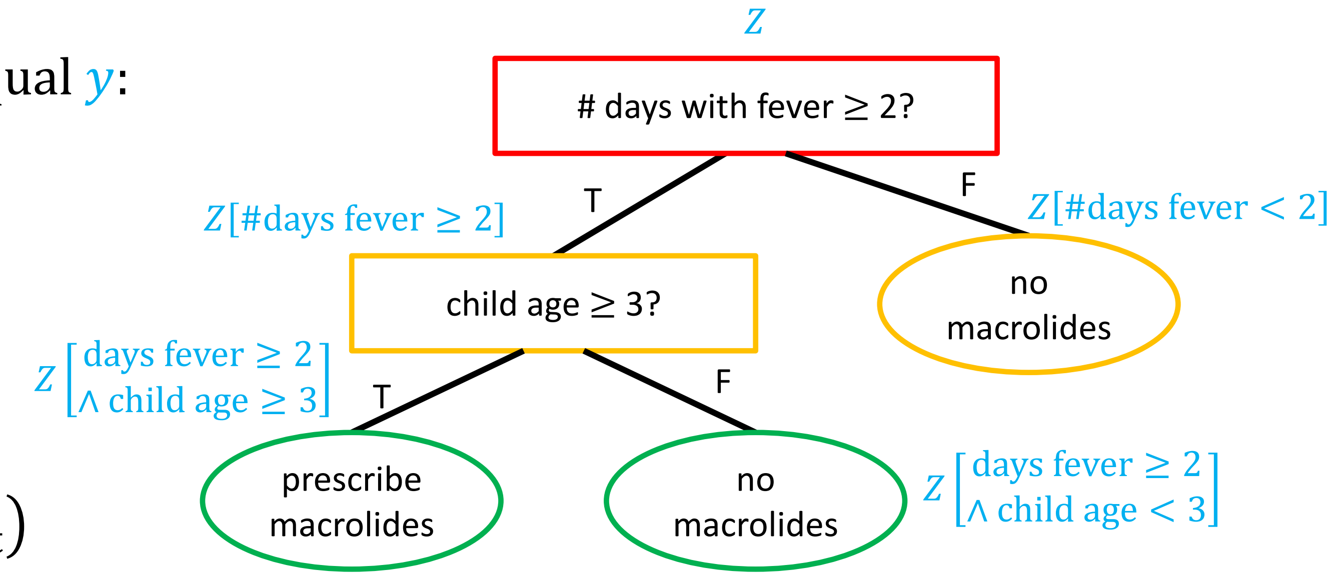
        return LeafNode(y)

    $(j, t) \leftarrow \mathrm{BestSplit}(Z)$

    $T_{\mathrm{left}} \leftarrow \mathrm{LearnTree}(Z[x_j \geq t])$

    $\boxed{T_{\mathrm{right}} \leftarrow \mathrm{LearnTree}(Z[x_j < t])}$

    return $\mathrm{InternalNode}(j, t, T_{\mathrm{left}}, T_{\mathrm{right}})$



Learning Algorithm

- Let $Z[C] = \{\, (x, y) \in Z \mid C(x, y) \,\}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

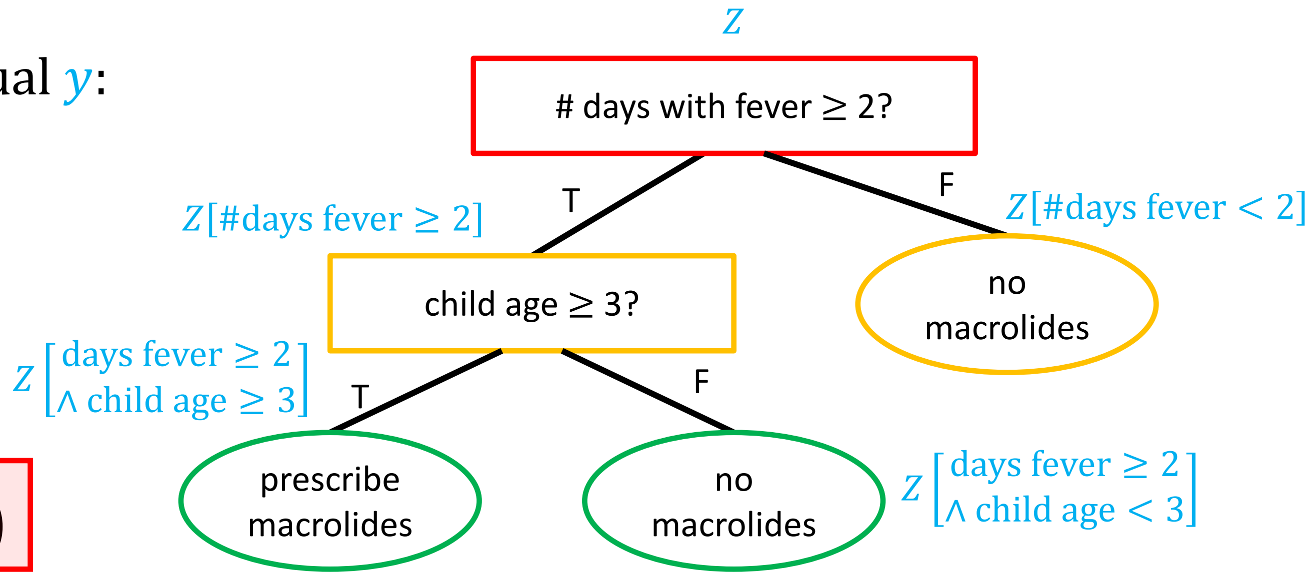
    return LeafNode(y)

  $(j, t) \leftarrow \text{BestSplit}(Z)$

  $T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])$

  $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

  $\boxed{\textbf{return } \text{InternalNode}(j, t, T_{\text{left}}, T_{\text{right}})}$



Learning Algorithm

- Let $Z[C] = \{ (x, y) \in Z \mid C(x, y) \}$ be the subset of Z where C holds

def $\text{LearnTree}(Z)$:

    if all labels in Z are the same and equal y:

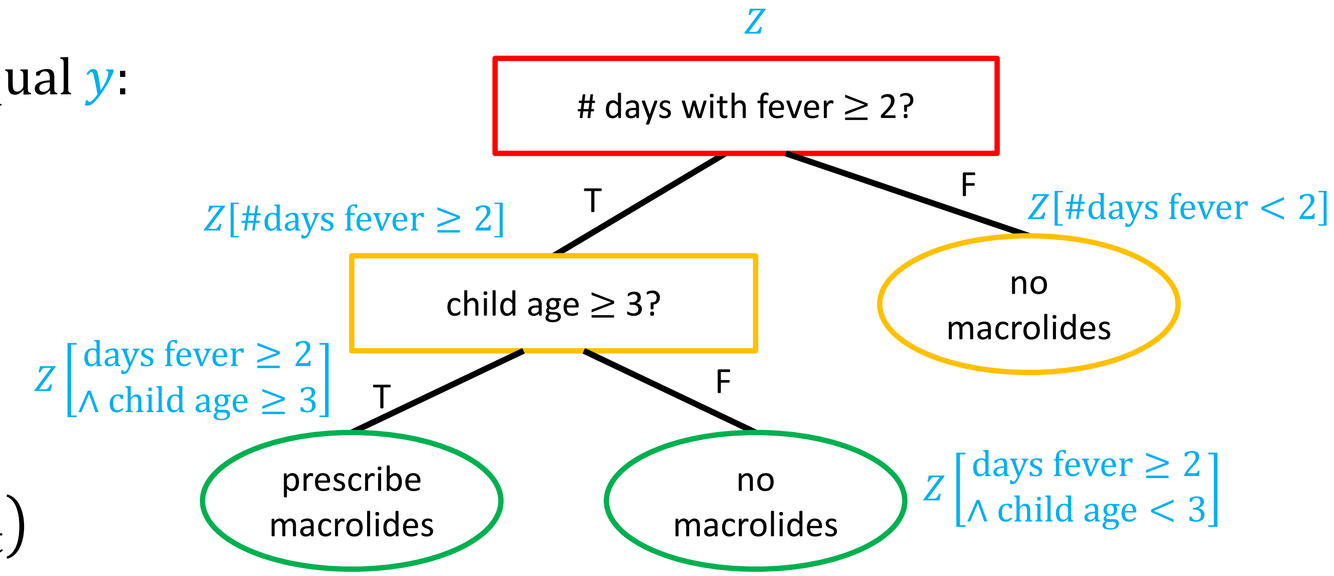
        return $\text{LeafNode}(y)$

    $(j, t) \leftarrow \text{BestSplit}(Z)$

    $T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])$

    $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

    return $\text{InternalNode}(j, t, T_{\text{left}}, T_{\text{right}})$



Learning Algorithm

- Let $Z[C] = \{\, (x, y) \in Z \mid C(x, y) \,\}$ be the subset of Z where C holds

def LearnTree(Z):

  if all labels in Z are the same and equal y:

    return LeafNode(y)

  $(j, t) \leftarrow \text{BestSplit}(Z)$

  $T_{\text{left}} \leftarrow \text{LearnTree}(Z[x_j \geq t])$

  $T_{\text{right}} \leftarrow \text{LearnTree}(Z[x_j < t])$

  return InternalNode$(j, t, T_{\text{left}}, T_{\text{right}})$

How to choose the best split?