Pattern Recognition & Decision Tree

Pasin Manurangsi

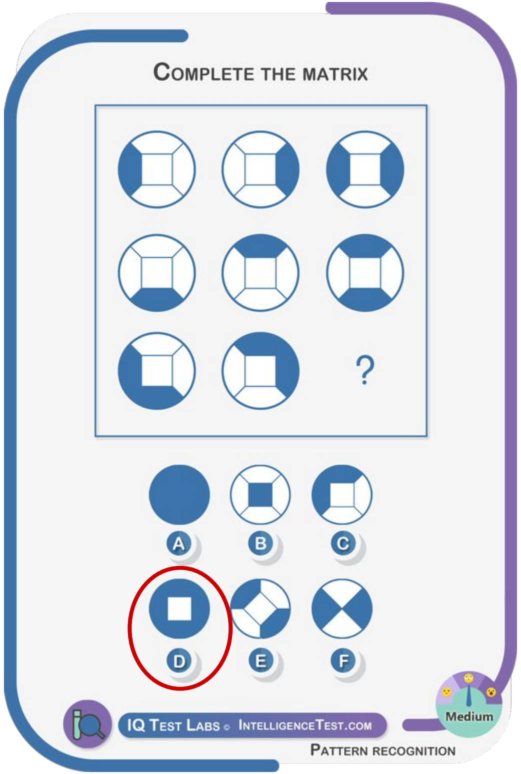
Google Research

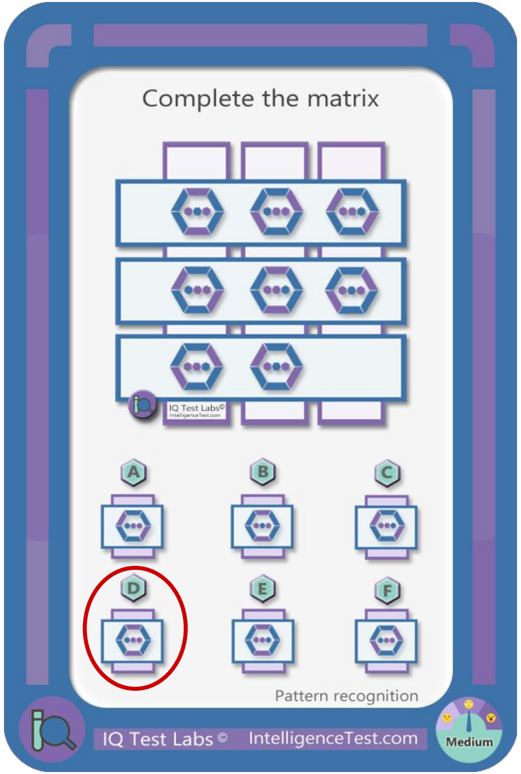
n. any regularly repeated arrangement

Pattern Recognition

n. the fact of knowing someone or something because you have seen or heard him or her or experienced it before

Pattern Recognition Examples: IQ Tests

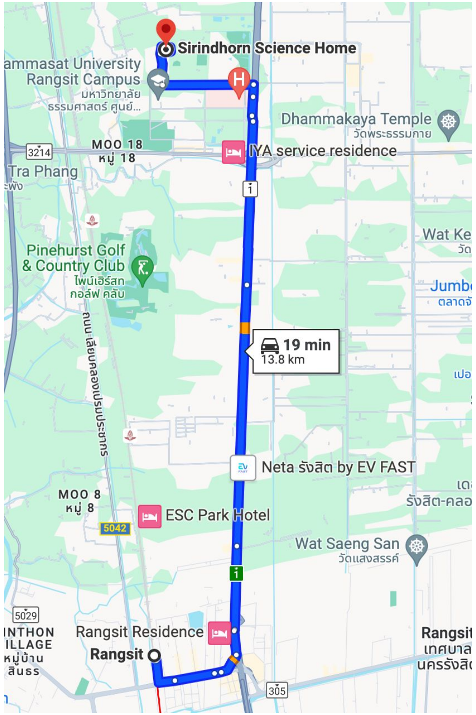




Pattern Recognition Examples: “Real life”

Thrown Away?	Yes	No

Pattern Recognition Examples: “Real life”



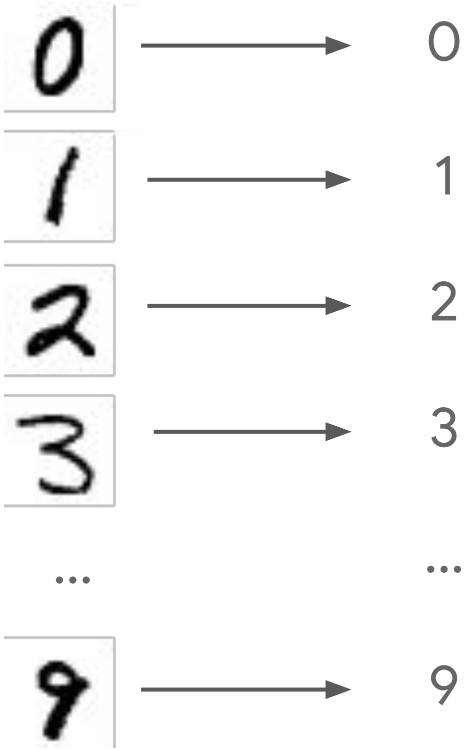
Taxi

- Taxi from A ⇒ B
- Even if we haven’t been from A ⇒ B before
  - Can still estimate the cost
  - Can still estimate travel time

Other examples

- Travel costs / time
- Food, drink costs
- Exam scores

Pattern Recognition Examples



Optical Character Recognition (OCR)

- Takes scanned documents ⇒ Digital version
- Applications
  - Automatic number-plate recognition
  - Automatic invoice / receipt recording
  - Passport recognition and information extraction in airports
  - Traffic-sign recognition

Pattern Recognition Examples

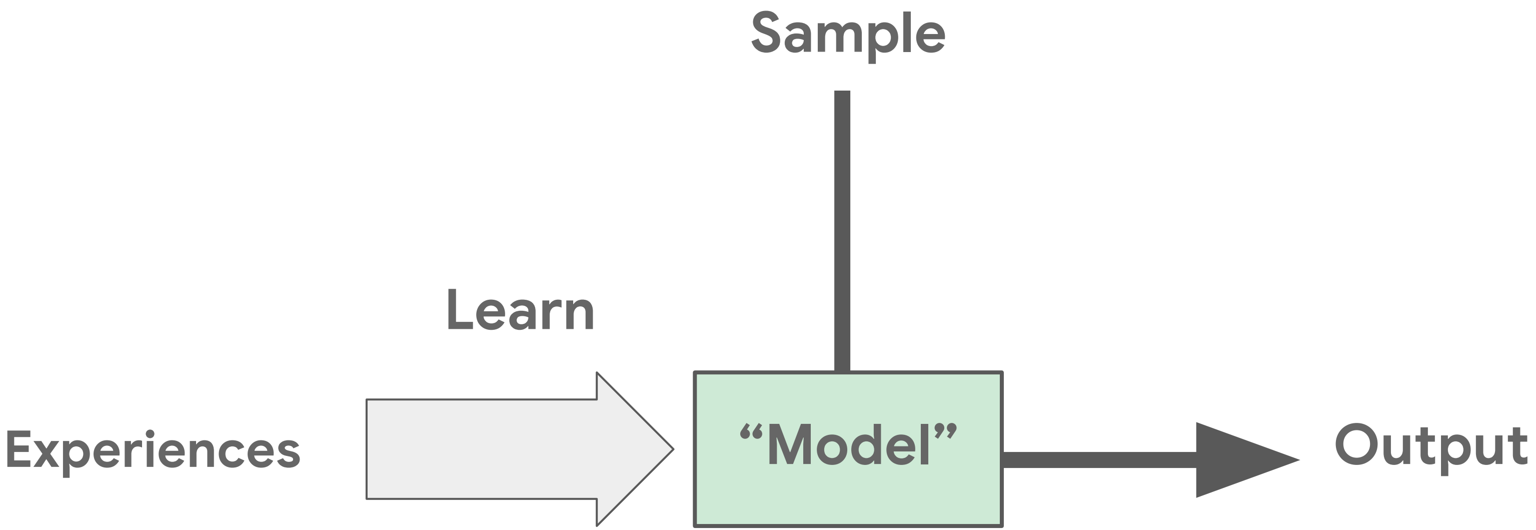
Healthcare

- Diagnosis:
  - Take clinical characteristics ⇒ predicts whether a patient has disease
- Prognosis:
  - Takes physical / biochemical markers
    ⇒ predicts the development of disease / survival rate

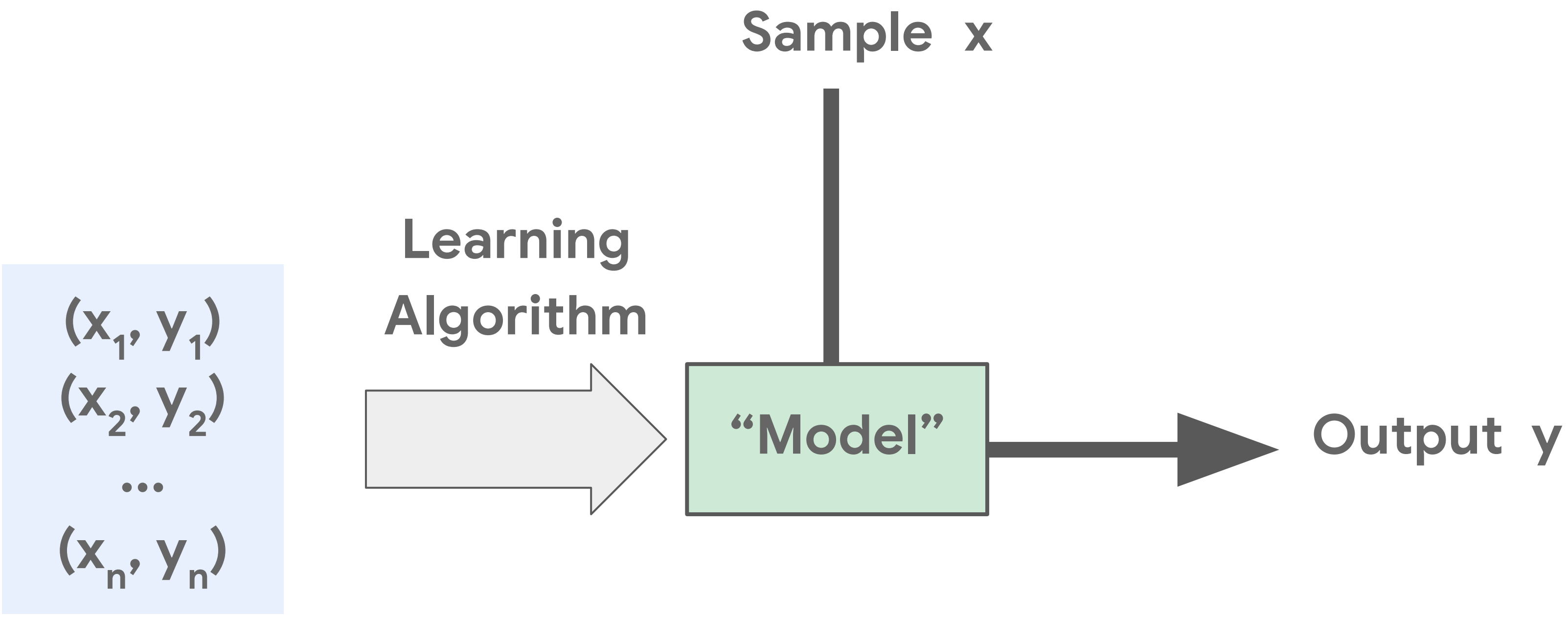
Fraud / Scam detection

- Emails
- SMS
- Social media posts
- Ads
- Financial transactions

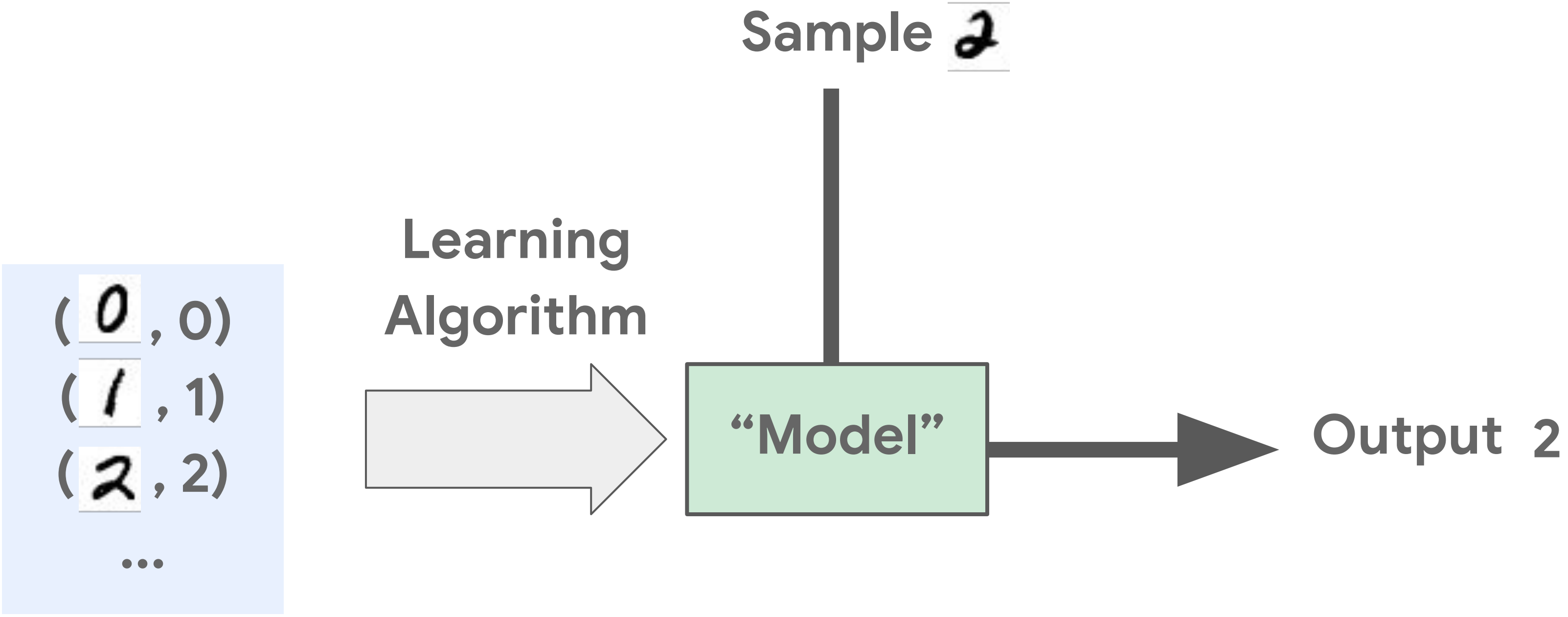
Pattern Recognition: Overview



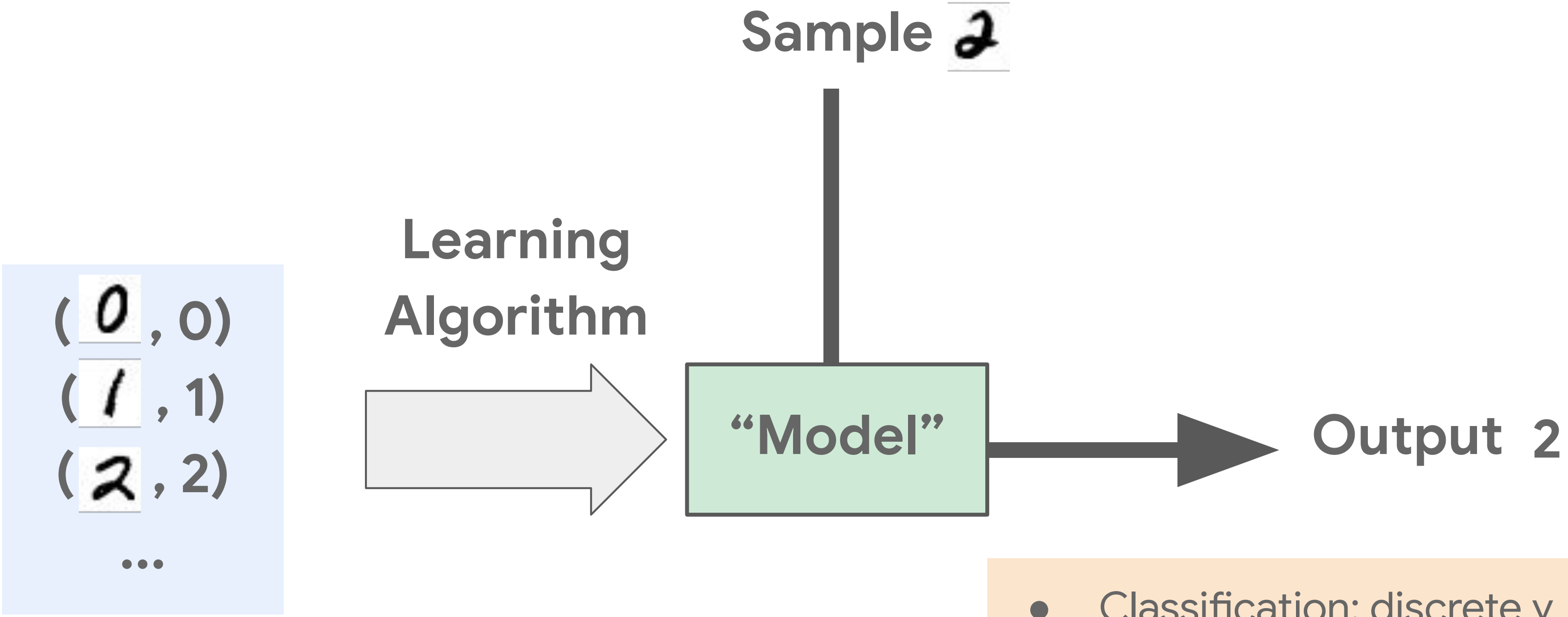
Machine Learning: ***Supervised*** Learning



Machine Learning: ***Supervised*** Learning

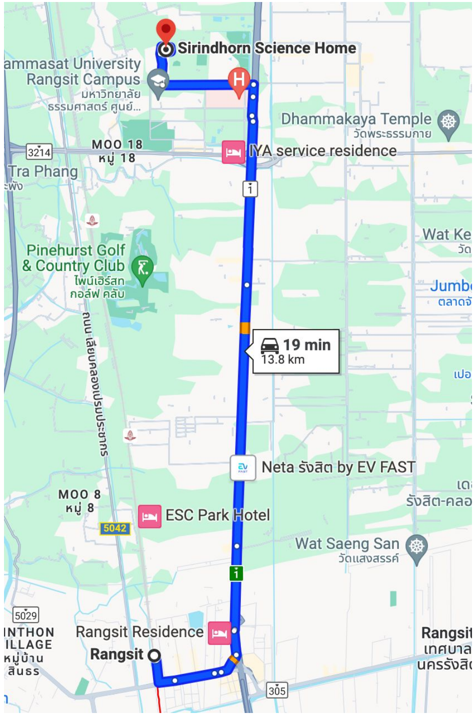


Machine Learning: ***Supervised*** Learning



- Classification: discrete y
- Regression: continuous y

Pattern Recognition Examples: “Real life”



≈ 120 Baht

Taxi

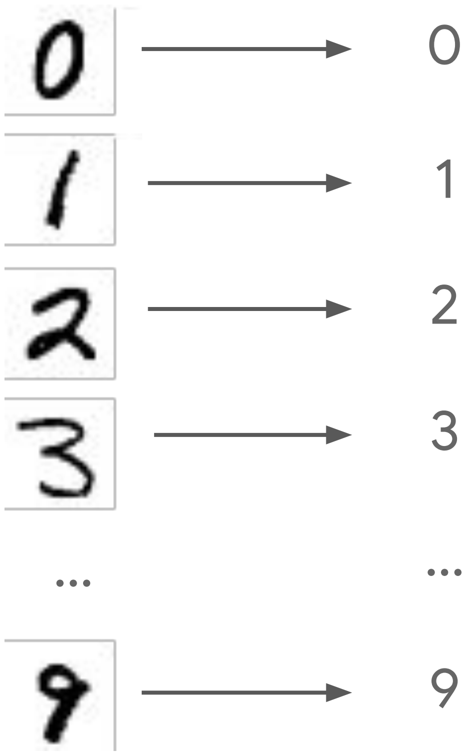
- Taxi from A ⇒ B
- Even if we haven’t been from A ⇒ B before
  - Can still estimate the cost
  - Can still estimate travel time

Other examples

- Travel costs / time
- Food, drink costs
- Exam ranks

Regression

Pattern Recognition Examples

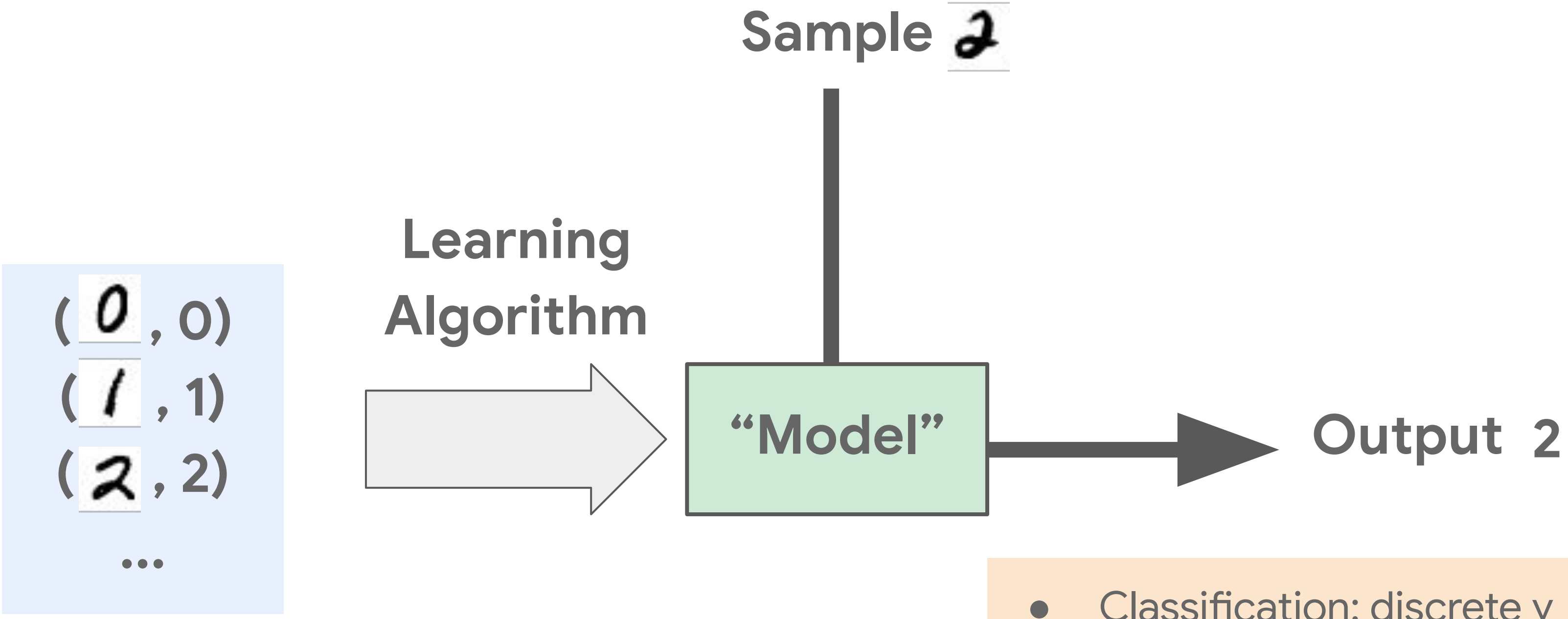


Optical Character Recognition (OCR)

- Takes scanned documents ⇒ Digital version
- Applications
  - Automatic number-plate recognition
  - Automatic invoice / receipt recording
  - Passport recognition and information extraction in airports
  - Traffic-sign recognition

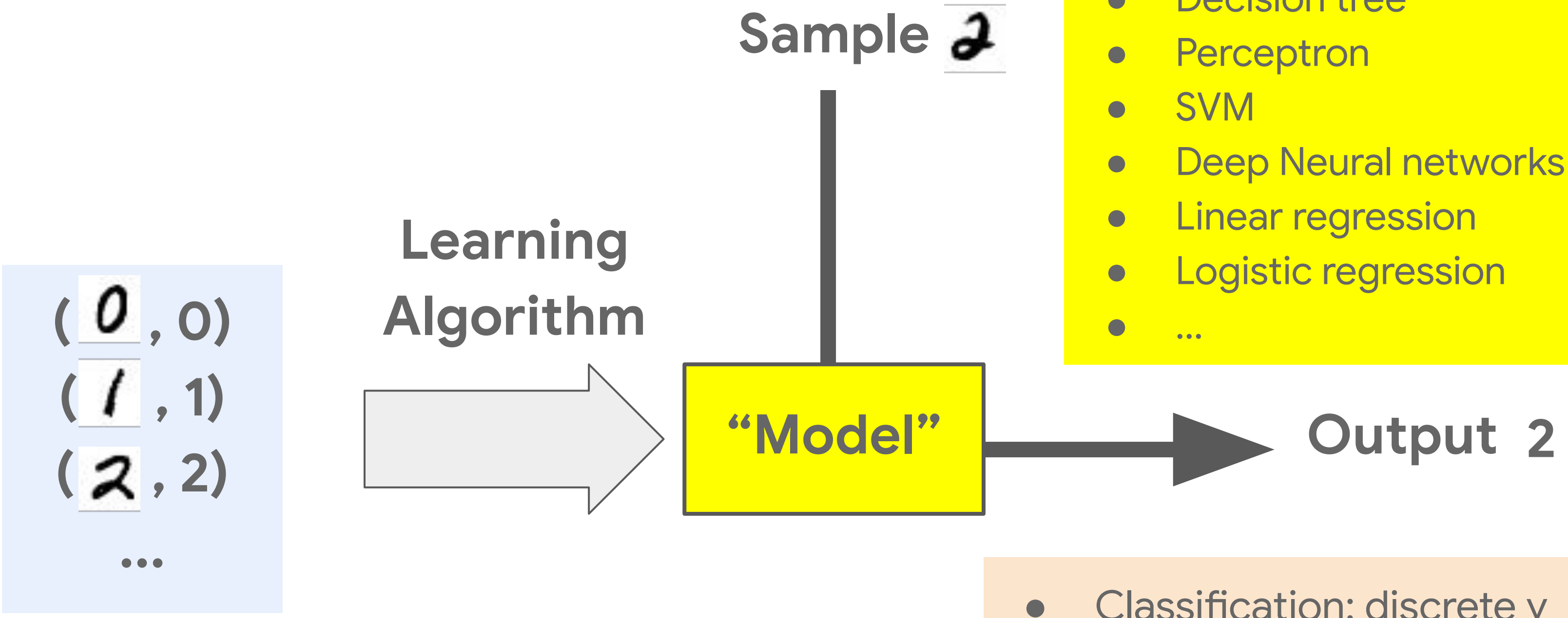
Classification

Machine Learning: ***Supervised*** Learning



- Classification: discrete y
- Regression: continuous y

Machine Learning: ***Supervised*** Learning



- Decision tree
- Perceptron
- SVM
- Deep Neural networks
- Linear regression
- Logistic regression
- ...

- Classification: discrete y
- Regression: continuous y

Machine Learning: ***Supervised*** Learning

Depends on the models
- Stochastic gradient descent (& variants)

Learning Algorithm

(0, 0)
(1, 1)
(2, 2)
...

Sample 2

"Model"

Output 2

- Decision tree
- Perceptron
- SVM
- Deep Neural networks
- Linear regression
- Logistic regression
- ...

- Classification: discrete y
- Regression: continuous y

Other Types of Learning

Supervised Learning
- Labelled Data
- Learn to predict labels

Unsupervised Learning
- Unlabelled Data
- Learn to group data points

Reinforcement Learning
- Learn to take actions
- After each action ⇒ gets reward / penalty

Active Learning
- Can ask a “teacher”
  - E.g. for a label of a data point

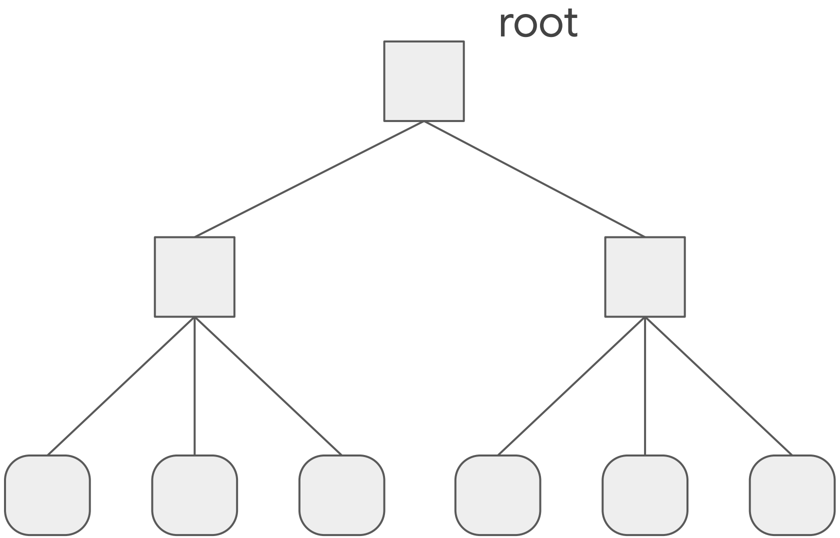
Online Learning
- Data points arrive over time
- Have to make predictions as the data points arrive

Semi-supervised Learning
- Learn from both labelled and unlabelled data points

...

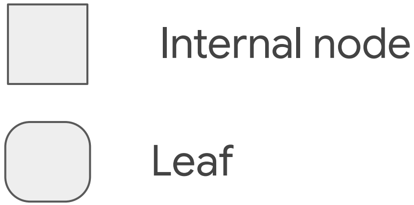
Decision Tree: Basics

Review: Rooted Tree

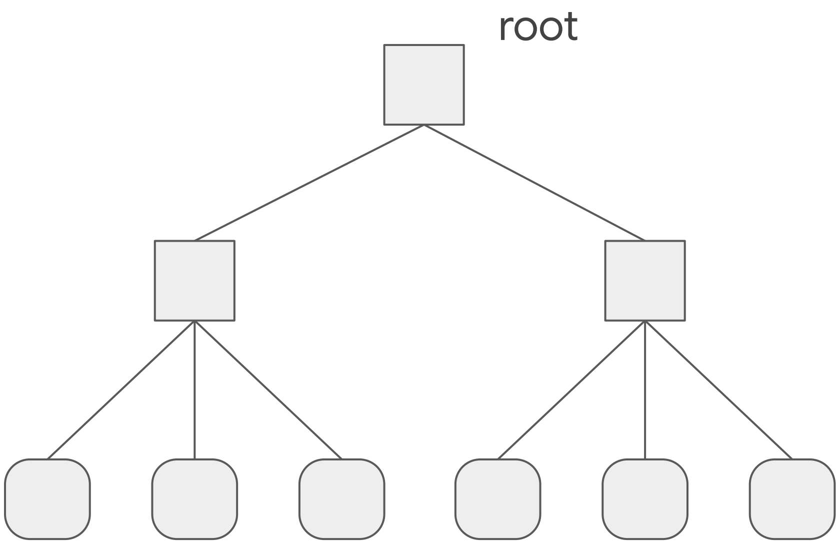


Rooted Tree

- ***Nodes***
- Every node except root has a ***parent***
- ***Internal nodes*** are node that have at least one child
- ***Leaves*** are nodes without a child

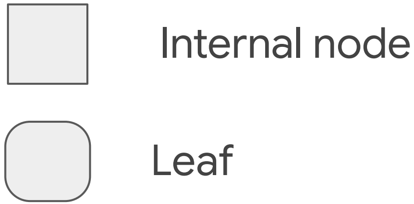


Decision Tree

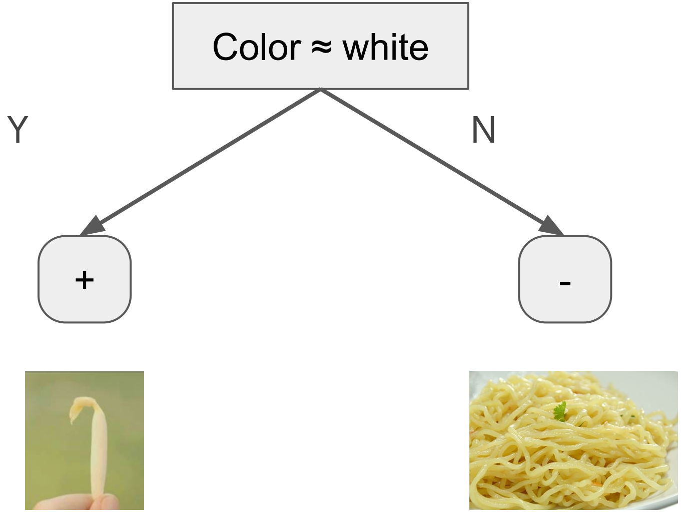


Decision Tree

- ***Internal nodes*** or ***decision nodes*** contain decision rule based on some feature of the input data
- ***Leaves*** or ***end nodes*** contain the labels that the decision tree predicts



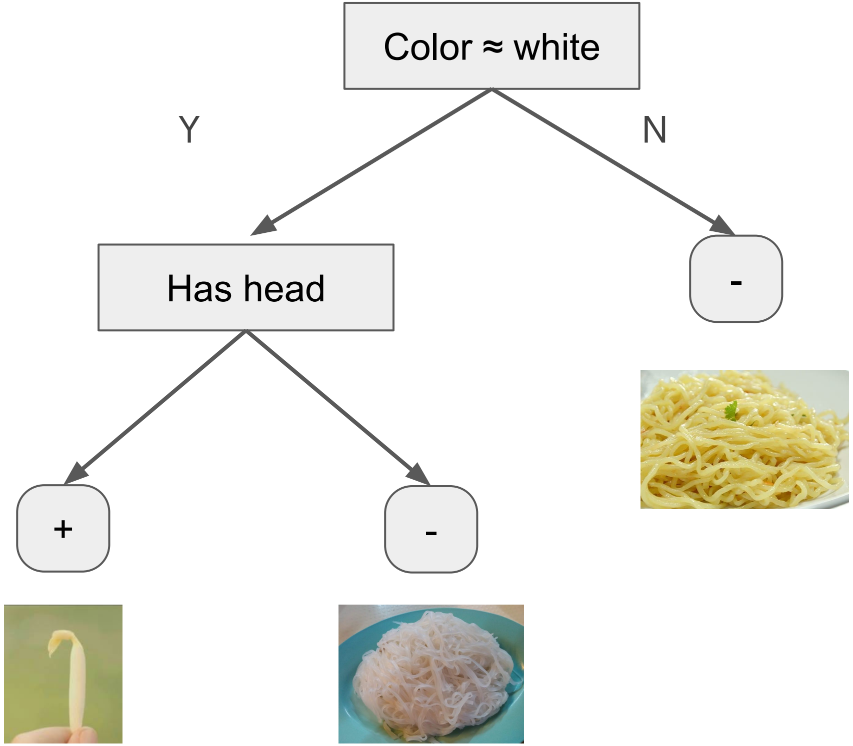
Decision Tree: Example



Example

- Deciding whether an object is a bean sprout

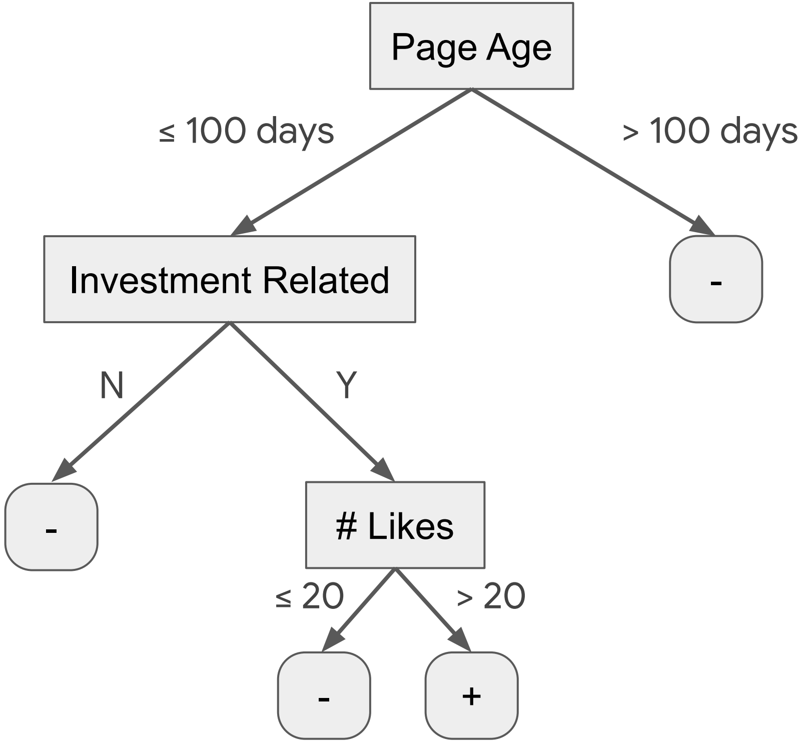
Decision Tree: Example



Example

- Deciding whether an object is a bean sprout

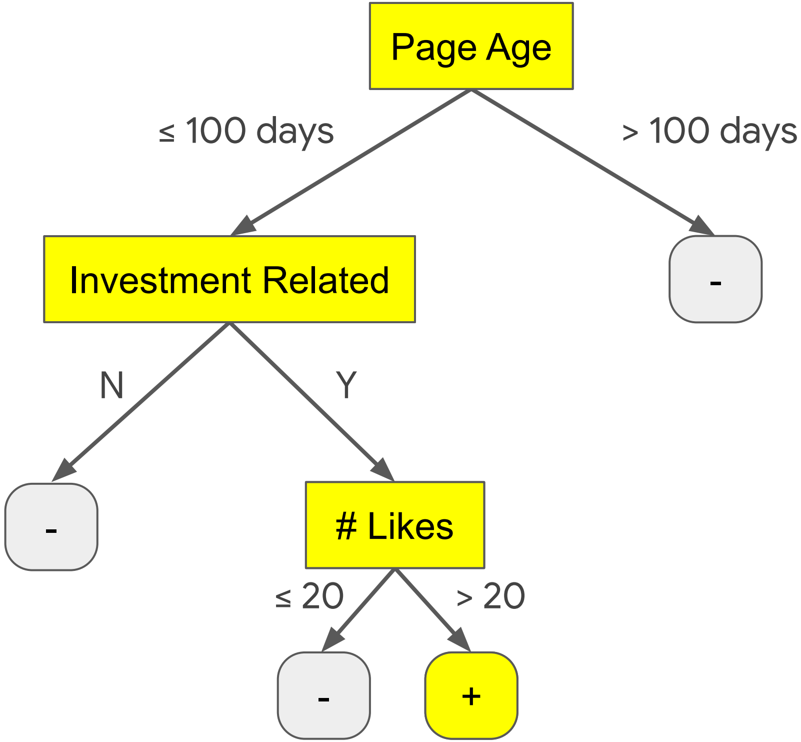
Decision Tree: Example



Example

- Deciding whether a facebook post is a scam
- **Features**:
  - # Post likes
  - Page / profile age
  - Contains external links
  - Investment related

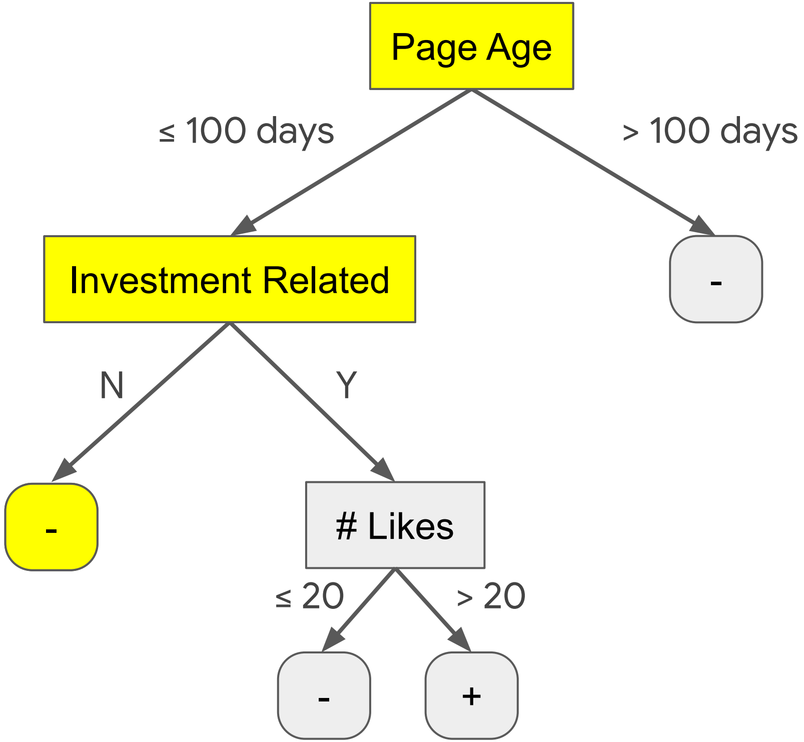
Decision Tree: Example



Post:

- # likes = 25
- Page / profile age = 10d
- Contains external links = N
- Investment related = Y

Decision Tree: Example

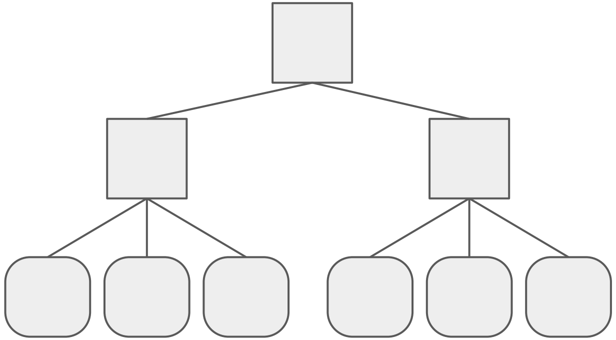


Post:

- # likes = 200
- Page / profile age = 50d
- Contains external links = Y
- Investment related = N

Decision Tree vs Deep Neural Networks

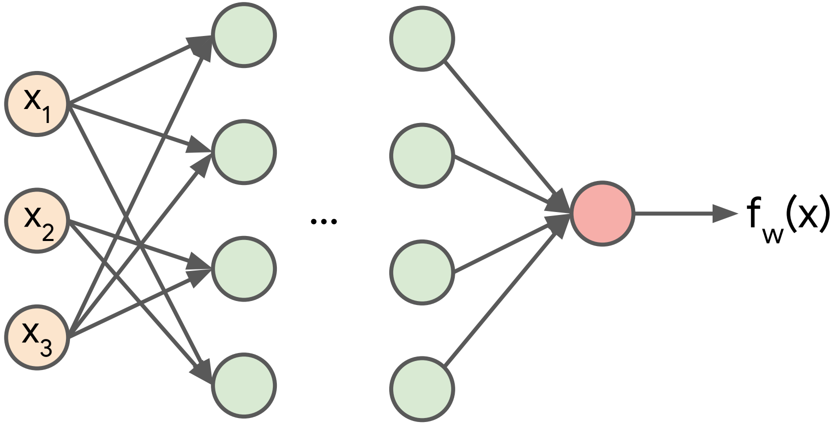
Decision Tree



Advantages

- Prediction is explainable
- Efficient to train & predict
- Easier to detect privacy / abuse issues

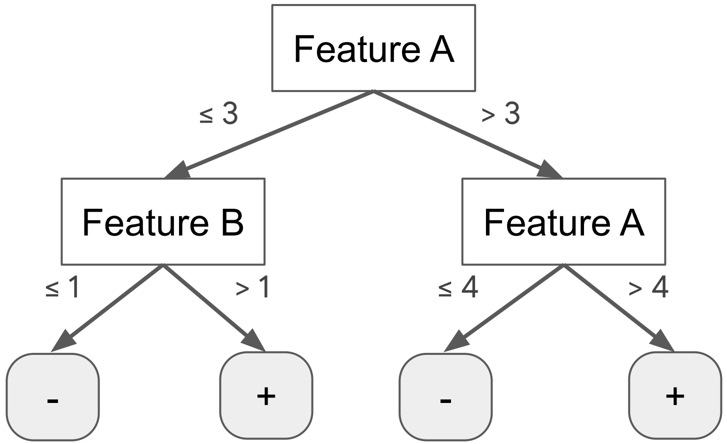
Deep Neural Networks

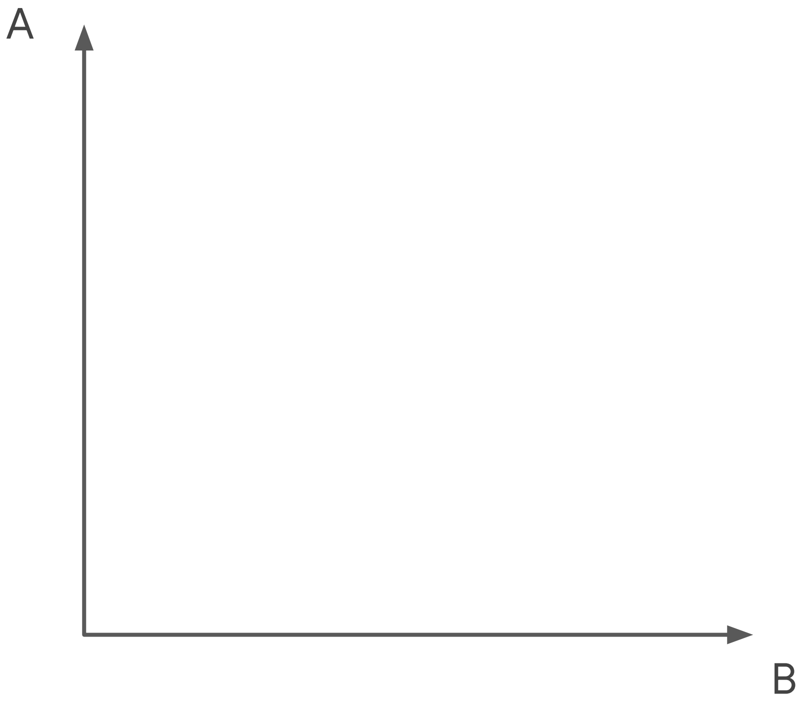


Disadvantages

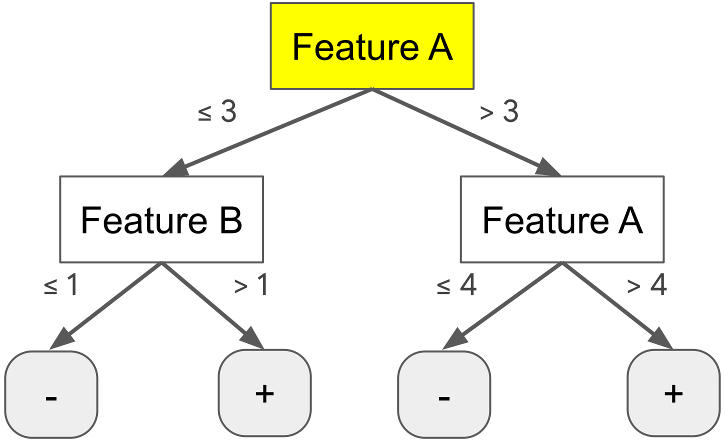
- Prediction can be mysterious
- Requires more resource to train
- Harder to detect privacy / abuse issues

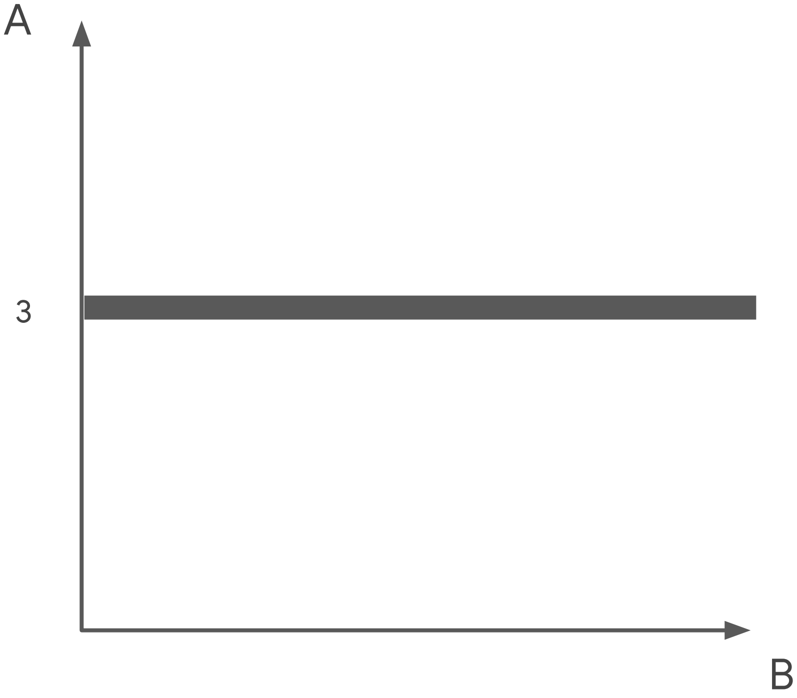
Decision Boundaries



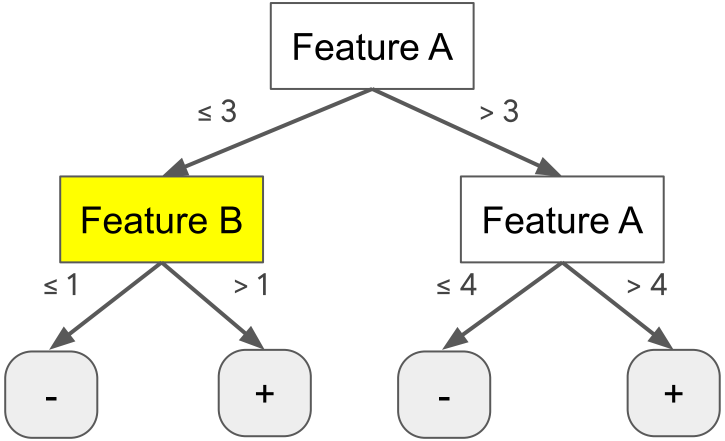


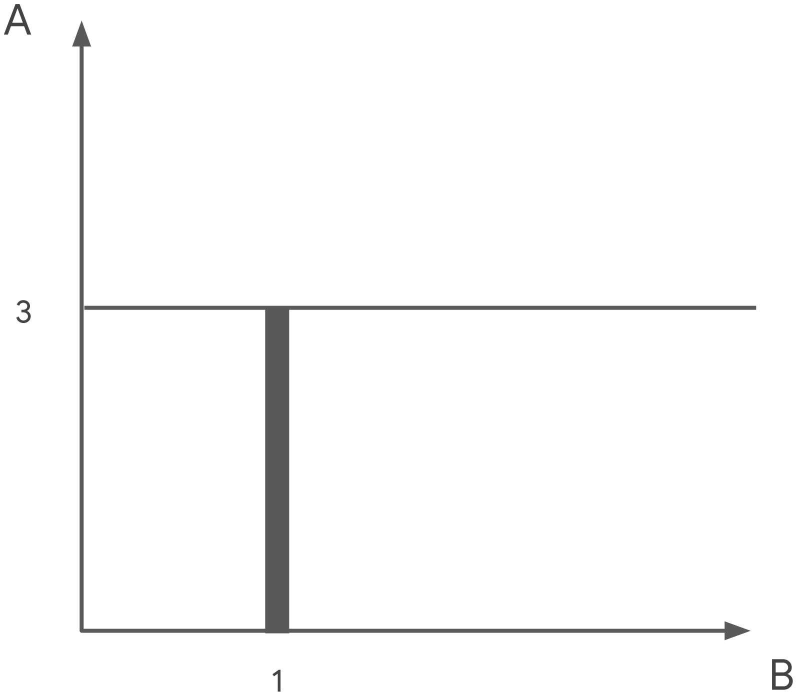
Decision Boundaries



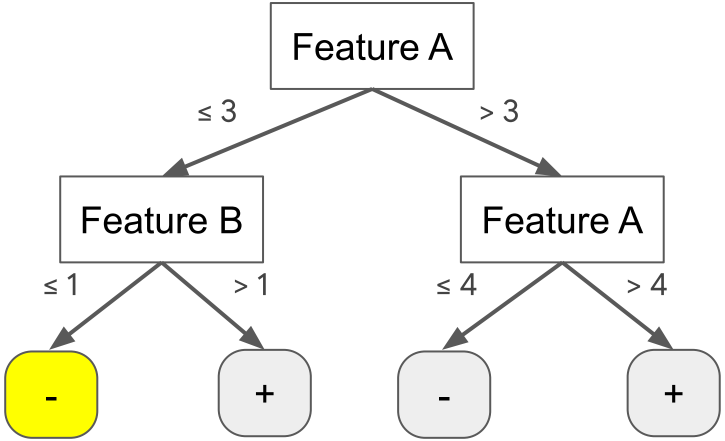


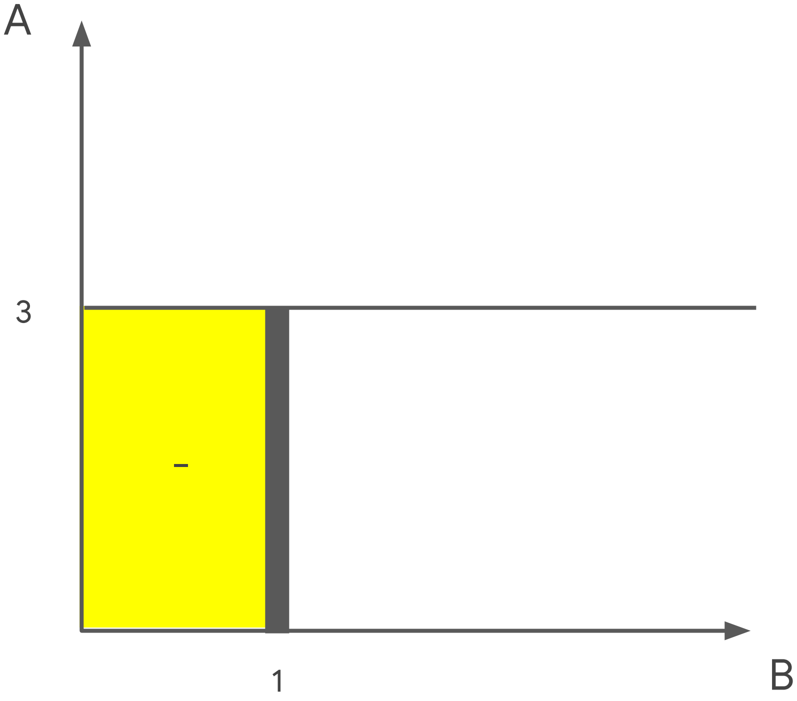
Decision Boundaries



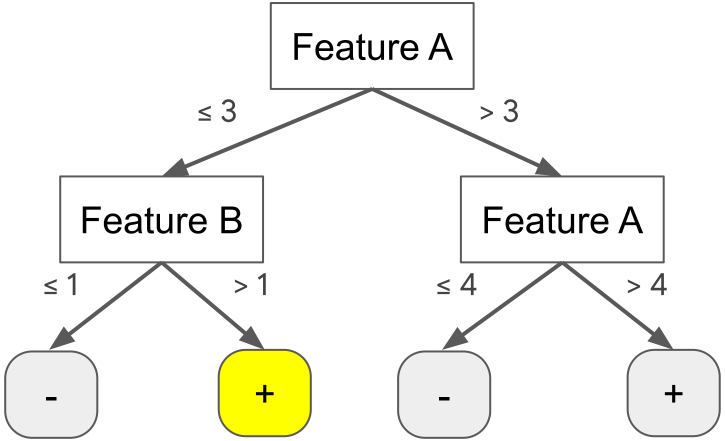


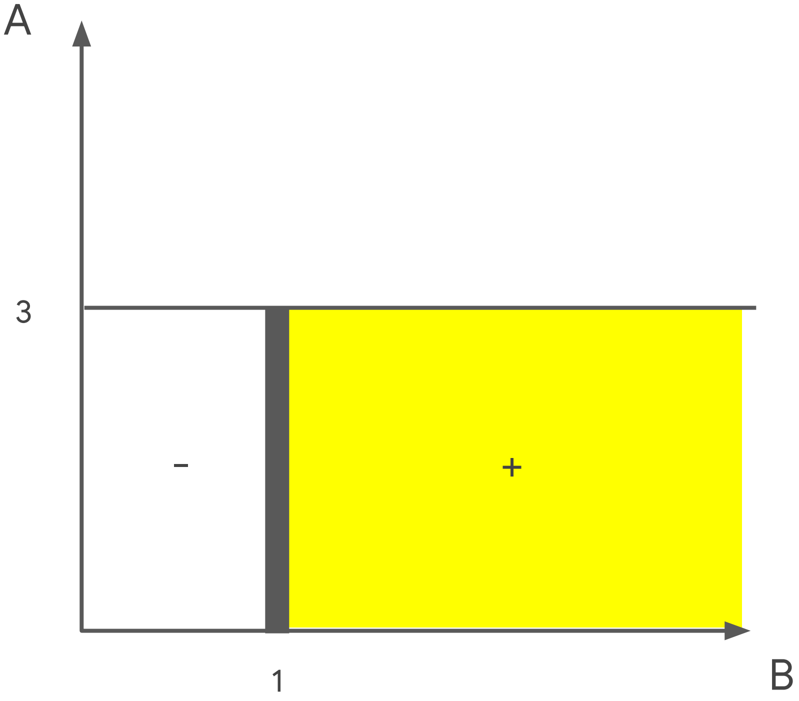
Decision Boundaries



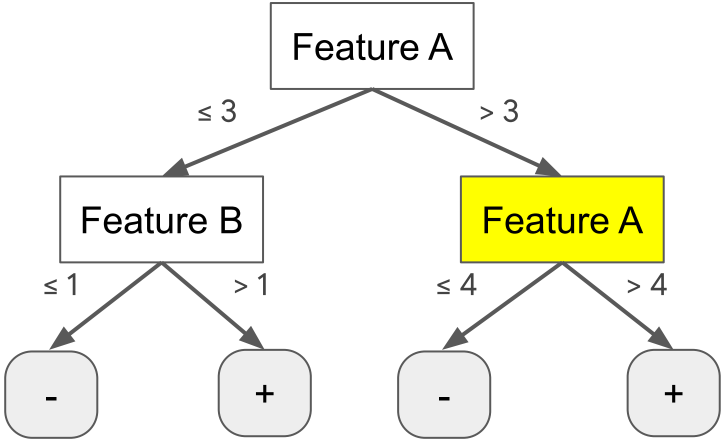


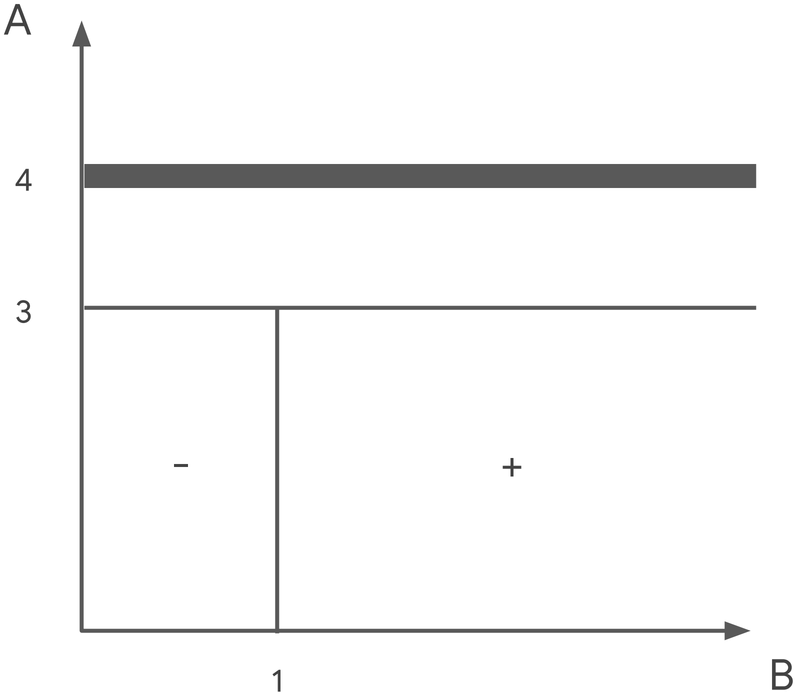
Decision Boundaries



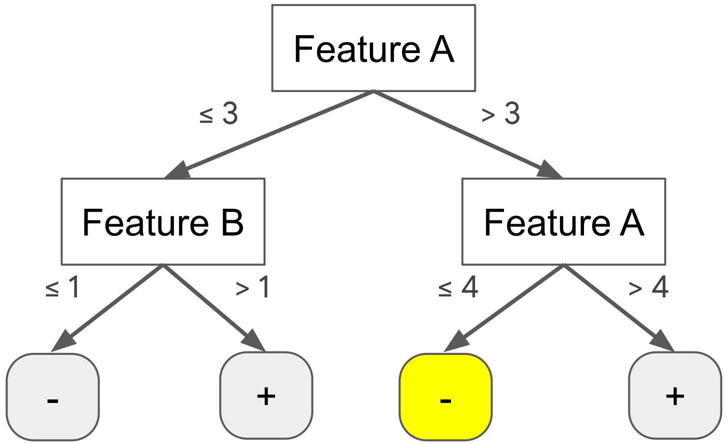


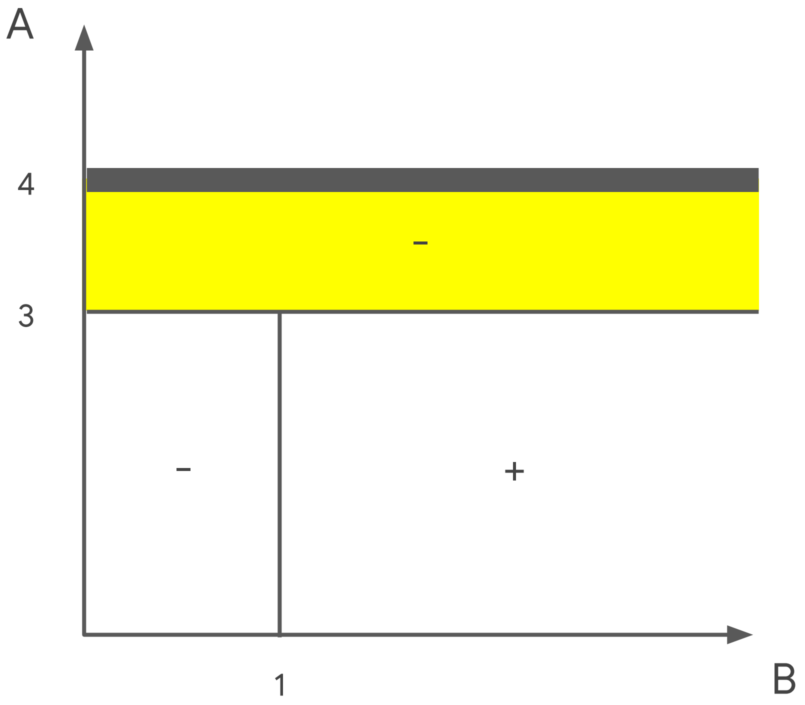
Decision Boundaries



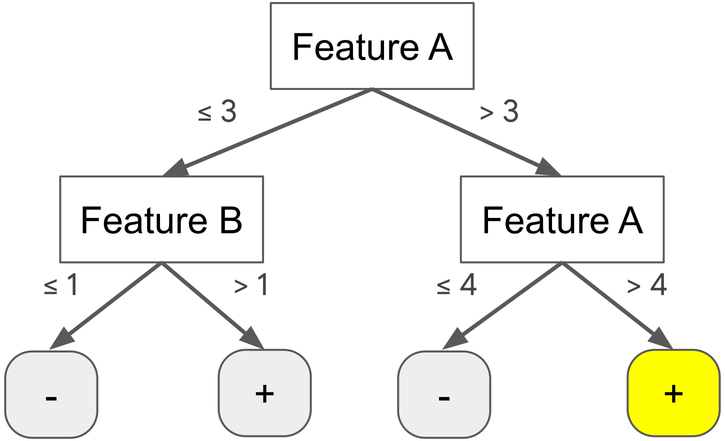


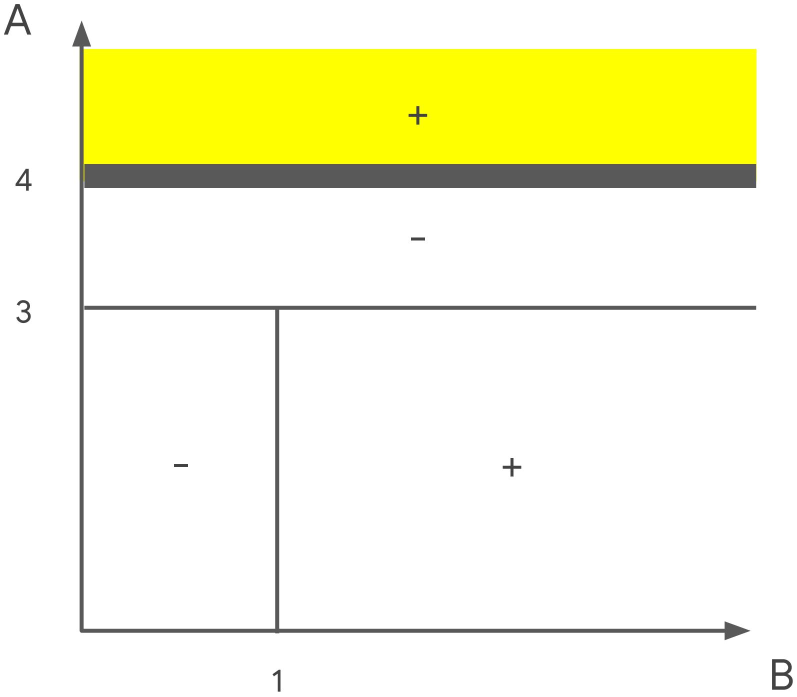
Decision Boundaries



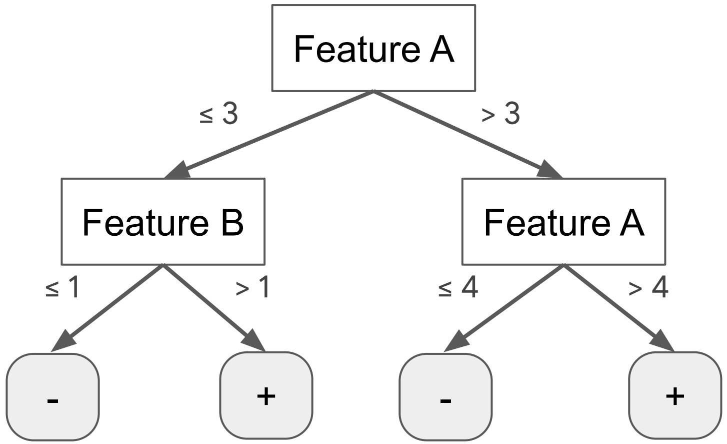


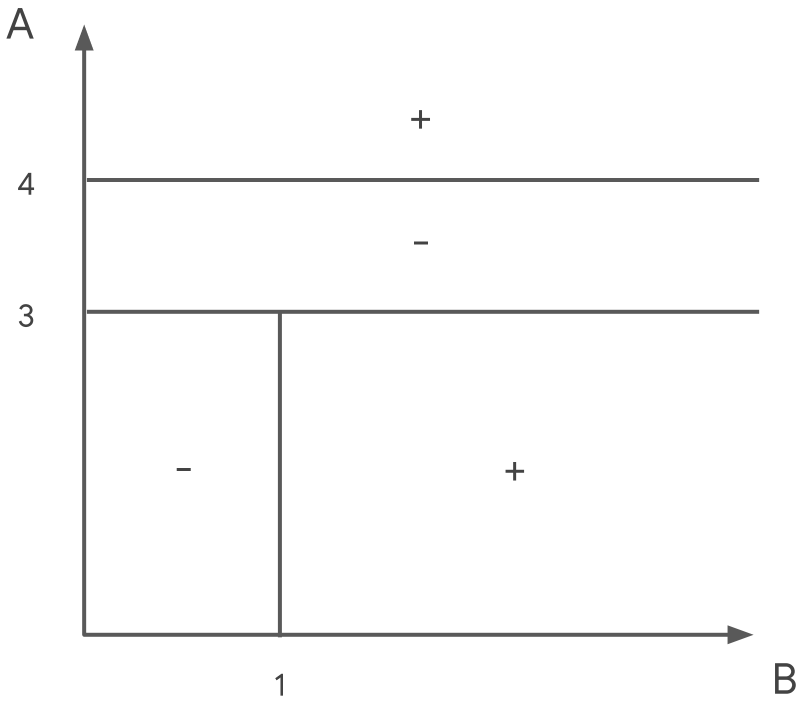
Decision Boundaries





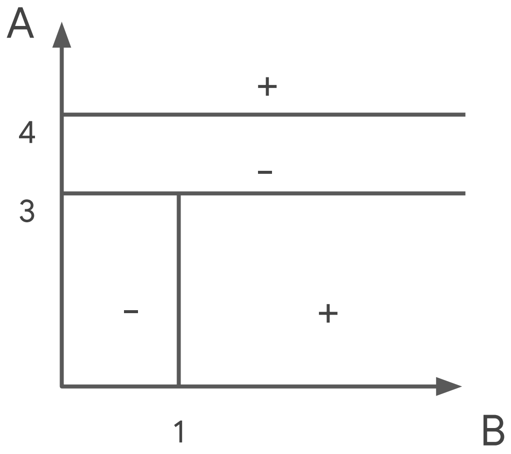
Decision Boundaries





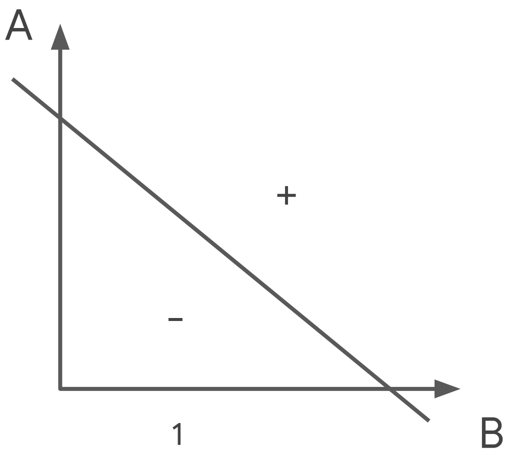
Decision Tree vs Linear Classifiers

Decision Tree DB



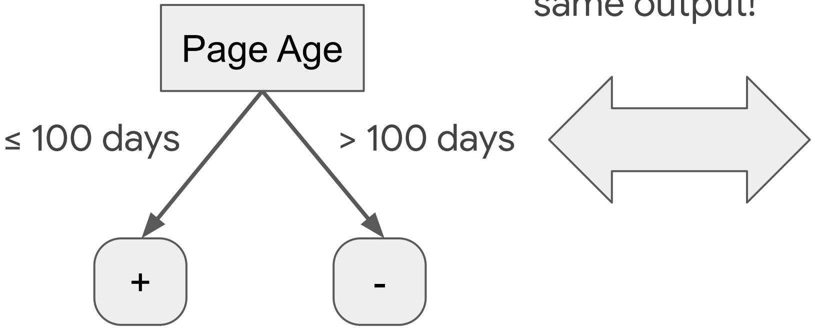
- Can approximate arbitrarily complicated decision boundaries
- But tree size grows if functions are more complicated

Linear Classifiers

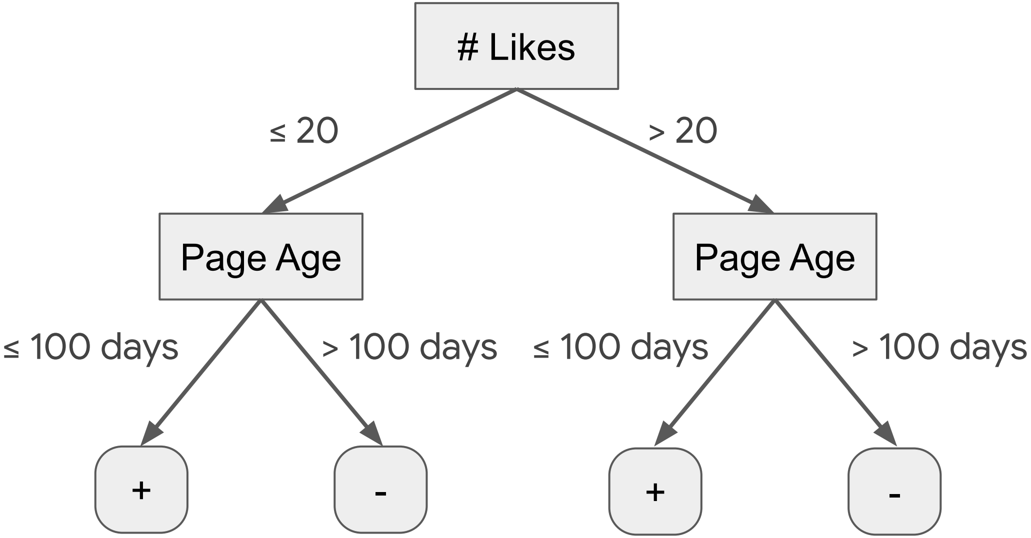


- Only suitable if there is linear relationship between underlying data

Decision Tree: Which Trees to Choose?



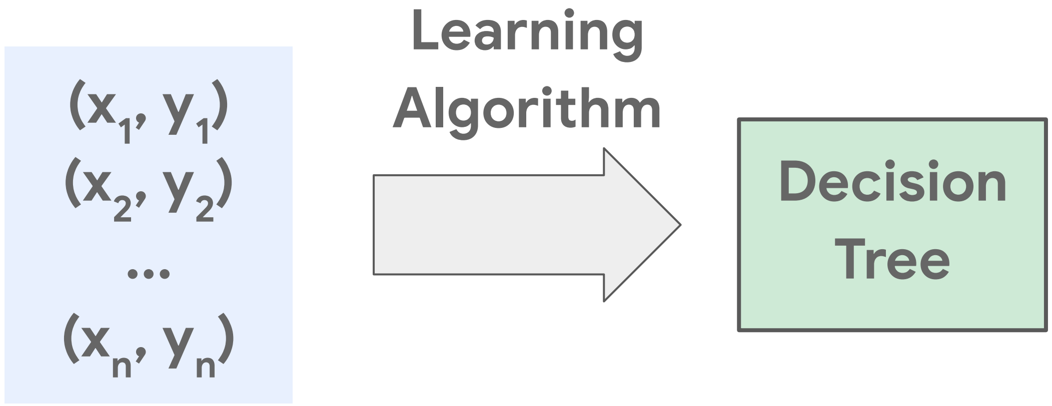
Gives Exactly the same output!



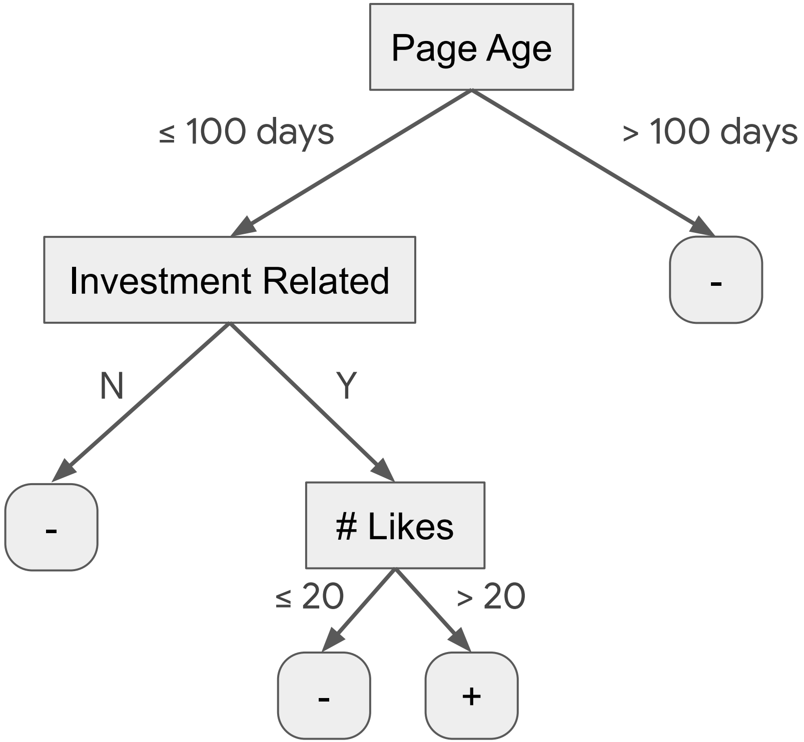
More complicated?

Learning Decision Tree

Learning Decision Tree



Decision Tree: Example



Example

- Deciding whether a facebook post is a scam
- **Features**:
  - # Post likes
  - # Comments
  - Page / profile age
  - Contains typos
  - Contains external links
  - Investment related

Decision Tree

Likes	Com	Age	Typo	Ext	Inv	Scam?
1520	1	5	Y	Y	Y	+
712	50	900	N	N	N	-
10	2	100	N	Y	Y	+
5	20	150	Y	N	N	-
100	100	70	Y	Y	N	+
2	20	7	Y	Y	Y	-

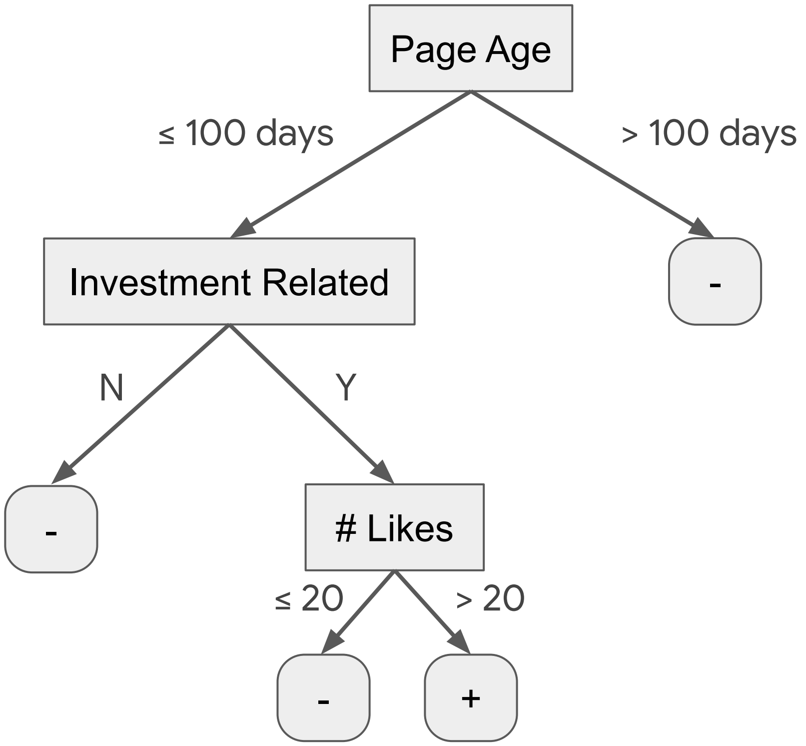
Example

- Deciding whether a facebook post is a scam
- **Features**:
  - # Post likes
  - # Comments
  - Page / profile age
  - Contains typos
  - Contains external links
  - Investment related

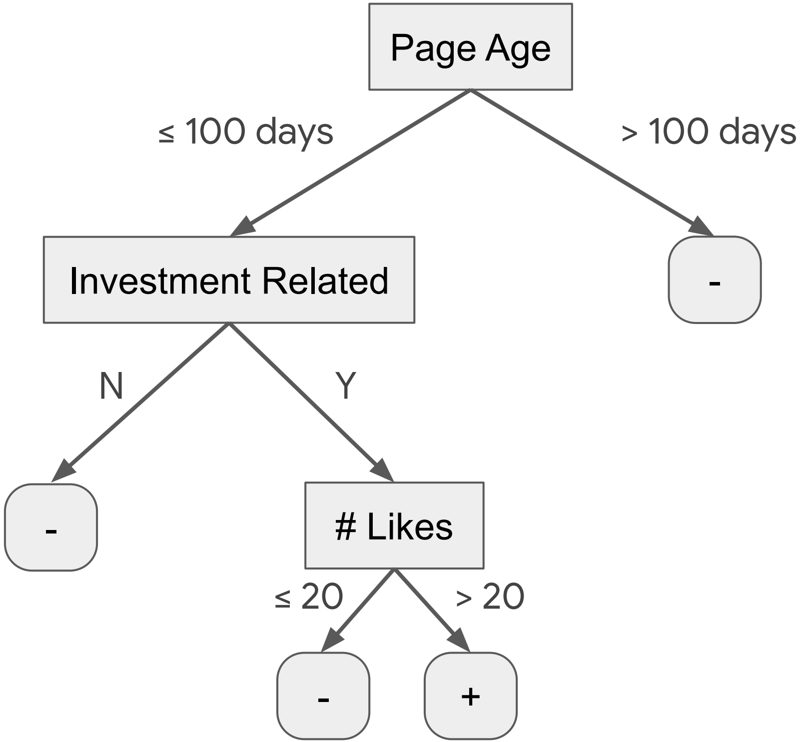
Learning Decision Tree

Likes	Com	Age	Typo	Ext	Inv	Scam?
1520	1	5	Y	Y	Y	+
712	50	900	N	N	N	-
10	2	100	N	Y	Y	+
5	20	150	Y	N	N	-
100	100	70	Y	Y	N	+
2	20	7	Y	Y	Y	-

Learning Algorithm



Learning Decision: Top-Down Algorithms



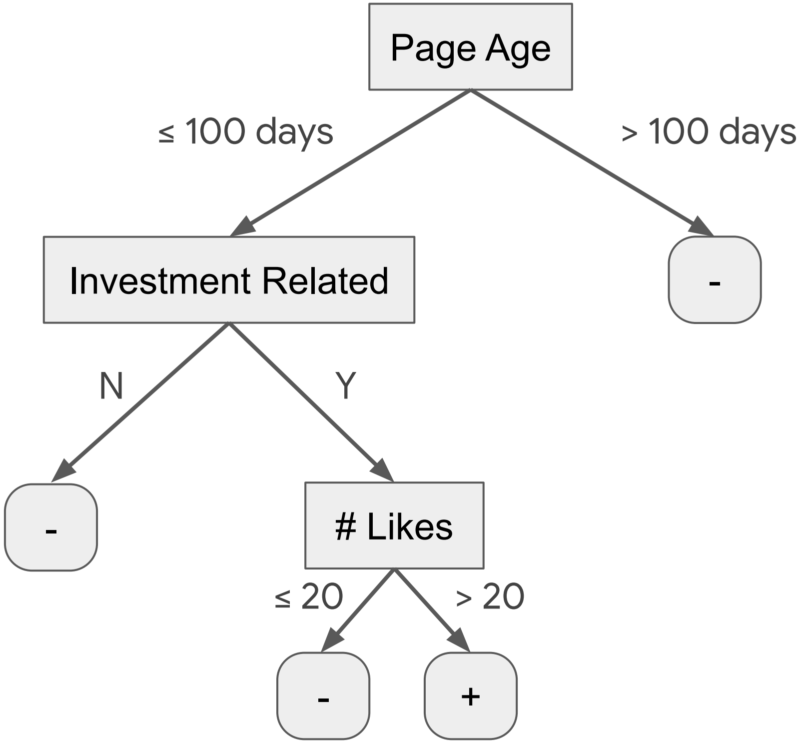
Build the tree in a ***top-down*** manner:

- Start from root
  - Select the “best” split
  - Recurse on left and right

Main Algorithmic Ingredients:

- Splitting Criteria
- Stopping Condition

Learning Decision: Top-Down Algorithms



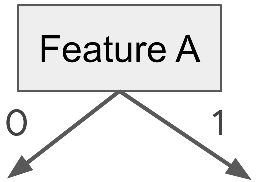
Build the tree in a ***top-down*** manner:

- Start from root
  - Select the “best” split
  - Recurse on left and right

Main Algorithmic Ingredients:

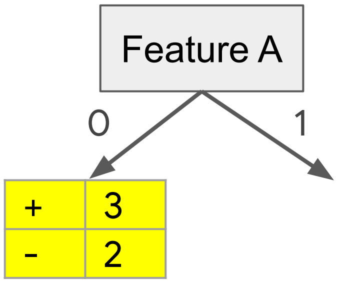
- **Splitting Criteria**
- Stopping Condition

Learning Decision: Top-Down Algorithms



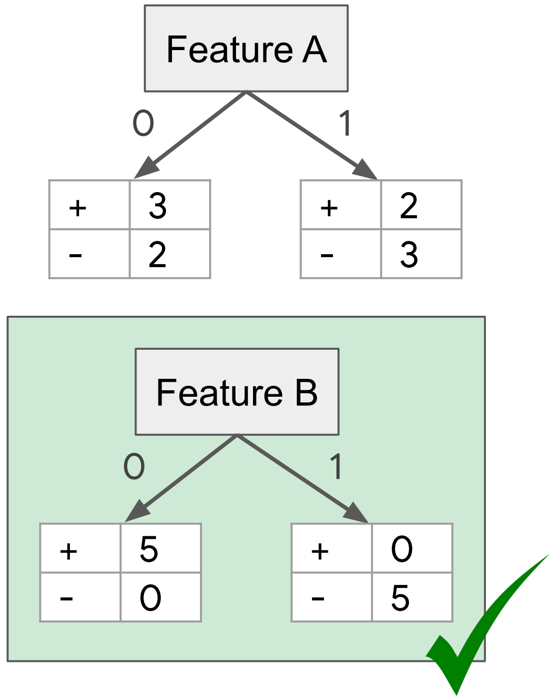
A	B	Label
1	1	+
1	0	-
1	1	+
0	1	+
0	0	-
1	0	-
1	1	+
0	0	-
0	0	-
0	1	+

Learning Decision: Top-Down Algorithms



A	B	Label
1	1	+
1	0	-
1	1	+
0	1	+
0	0	-
1	0	-
1	1	+
0	0	-
0	0	-
0	1	+

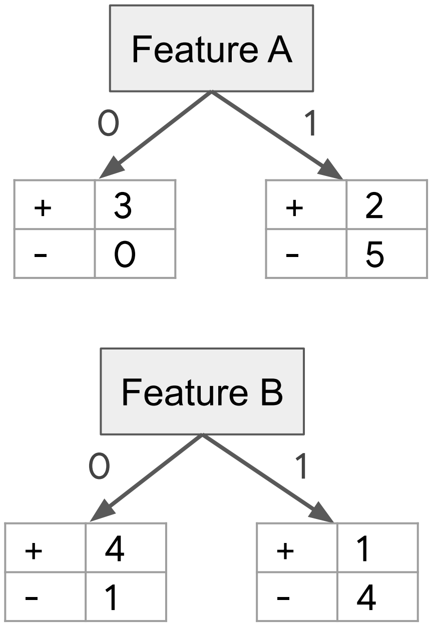
Learning Decision: Top-Down Algorithms



Which is a better split?

A	B	Label
1	1	+
1	0	-
1	1	+
0	1	+
0	0	-
1	0	-
1	1	+
0	0	-
0	0	-
0	1	+

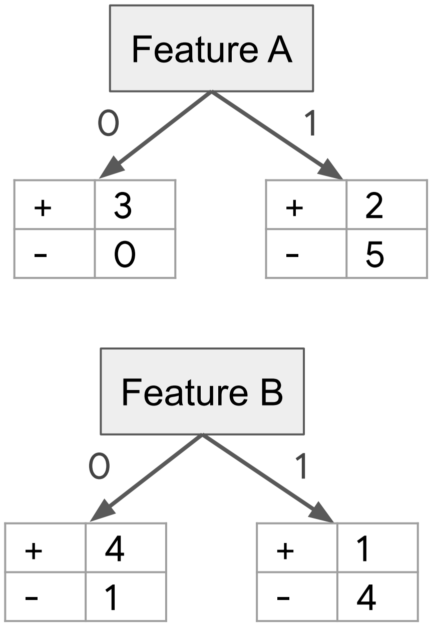
Learning Decision: Top-Down Algorithms



A	B	Label
1	0	+
0	1	-
1	1	+
1	1	+
0	0	-
0	0	-
0	1	+
0	0	-
0	0	-
0	1	+

Which is a better split?

Learning Decision: Top-Down Algorithms



Which is a better split?

Want:

- *“Balancedness”*
- *“Purity”*

Some popular measures

- Misclassification Error
- Gini Index
- Information Gain

Misclassification Error

A	B	Label
1	0	+
0	1	-
1	1	-
1	1	-
0	0	-
0	0	-
0	1	+
0	0	-
0	0	-
0	1	+

What if we have to predict the label without looking at any feature?

Misclassification Error

A	B	Label
*	*	+
*	*	-
*	*	-
*	*	-
*	*	-
*	*	-
*	*	+
*	*	-
*	*	-
*	*	+

Predicts -

Errors
= 10 - 7
= 3

$\mathbf{n}(+) = 3$, $\mathbf{n}(-) = 7$

What if we have to predict the label without looking at any feature?

Best prediction:
predict y that maximizes $n(y)$

Error $= n - \max_y n(y)$

Notations

- $\mathbf{n}$ = total # data points
- $\mathbf{n(y)}$ = # data points labelled y

Misclassification Error

A	B	Label
*	*	1
*	*	2
*	*	3
*	*	4
*	*	1
*	*	3
*	*	4
*	*	9
*	*	4
*	*	4

Predicts **4**

Errors
= 10 - 4
= 6

$\mathbf{n(1)} = 2$, $\mathbf{n(2)} = 1$, $\mathbf{n(3)} = 2$, $\mathbf{n(4)} = 4$, $\mathbf{n(9)} = 1$

What if we have to predict the label without looking at any feature?

Best prediction:
predict y that maximizes $n(y)$

Error $= n - \max_y n(y)$

Notations

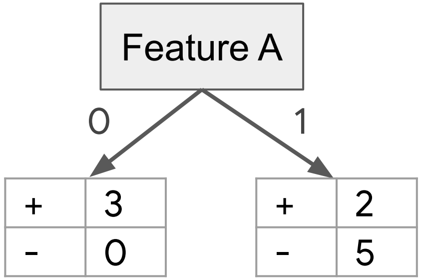
- $\mathbf{n}$ = total # data points
- $\mathbf{n(y)}$ = # data points labelled y

Splitting Criteria I: Misclassification Error

Split Misclass. Error = # of errors if we were to use this feature alone

$= \sum_i$ (misclassification for class i)

$= \sum_i (n_i - \max_y n_i(y))$



$n_0 = 3$
$n_0(+) = 3$
$n_0(-) = 0$

$n_1 = 7$
$n_1(+) = 2$
$n_1(-) = 5$

misclassification	0	2

Misclassification error = 0 + 2 = 2

Notations

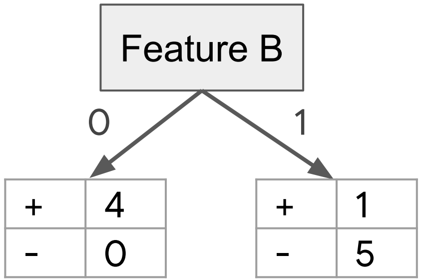
- n_i = # points in class i
- $n_i(y)$ = # data points labelled y in class i

Splitting Criteria I: Misclassification Error

Split Misclass. Error $= \#$ of errors if we were to use this feature alone

$$= \sum_i (\text{misclassification for class } i)$$

$$= \sum_i (n_i - \max_y n_i(y))$$



$n_0 = 4$
$n_0(+) = 4$
$n_0(-) = 0$

$n_1 = 6$
$n_1(+) = 1$
$n_1(-) = 5$

Misclassification error = 0 + 1 = 1

Notations

- n_i = # points in class i
- $n_i(y)$ = # data points labelled y in class i

misclassification	0	1

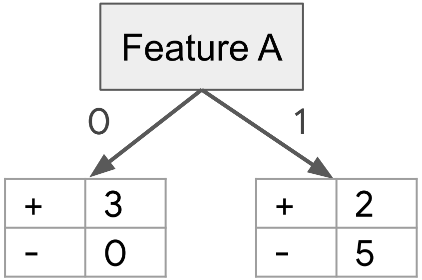
Splitting Criteria I: Misclassification Error

Split Misclass. Error = # of errors if we were to use this feature alone

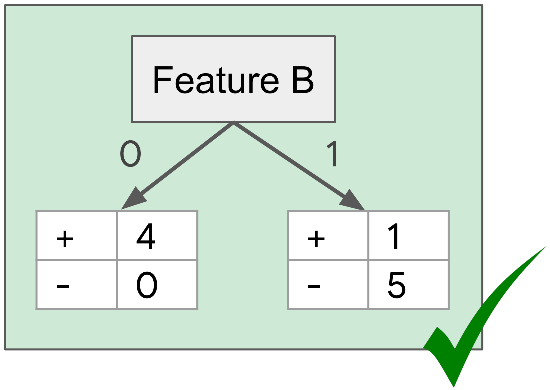
$= \sum_i$ (misclassification for class i)

$= \sum_i (n_i - \max_y n_i(y))$

Criteria I: Select the split to minimize the misclassification error



Misclassification error = 2



Misclassification error = 1

Gini Index

A	B	Label
*	*	+
*	*	-
*	*	-
*	*	-
*	*	-
*	*	-
*	*	+
*	*	-
*	*	-
*	*	+

n(+) = 3, **n(-)** = 7
p(+) = 0.3, **p(-)** = 0.7

Gini Index
= 1 - 0.3^2 - 0.7^2
= 0.42

If we take two random points, what are the chances that their labels disagree?

Gini-Index = $1 - \sum_y p(y)^2$

Notations

- **n** = total # data points
- **n(y)** = # data points labelled y
- **p(y)** = fraction of data points labelled y
  = n(y) / n

Gini Index

A	B	Label
*	*	1
*	*	3
*	*	2
*	*	3
*	*	3
*	*	3
*	*	1
*	*	3
*	*	3
*	*	3

Gini Index
$= 1 - 0.1^2 - 0.1^2 - 0.7^2$
$= 0.49$

$\mathbf{n(1)} = 2, \mathbf{n(2)} = 1, \mathbf{n(3)} = 7$
$\mathbf{p(1)} = 0.2, \mathbf{p(2)} = 0.1, \mathbf{p(3)} = 0.7$

If we take two random points, what are the chances that their labels disagree?

Gini-Index = $1 - \sum_y p(y)^2$

Notations

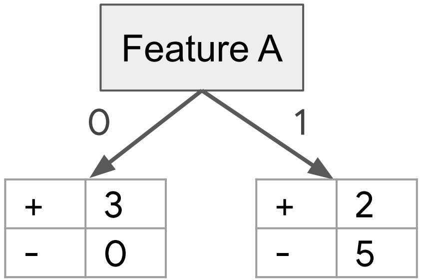
- $\mathbf{n}$ = total # data points
- $\mathbf{n(y)}$ = # data points labelled y
- $\mathbf{p(y)}$ = fraction of data points labelled y
  $= n(y) / n$

Splitting Criteria II: Gini Index

Split Gini Index = expected gini index across all classes

$= \sum_i (n_i / n) \cdot \mathrm{Gini}(\text{class } i)$

$= \sum_i (n_i / n) \cdot (1 - \sum_y p_i(y)^2)$



$n_0 = 3$
$p_0(+) = 1$
$p_0(-) = 0$

$n_1 = 7$
$p_1(+) = 2/7$
$p_1(-) = 5/7$

Gini Index	0	0.41

Split Gini Index = 0 + 0.7 * 0.41 ≈ 0.29

Notations

- n_i = # points in class i
- $n_i(y)$ = # data points labelled y in class i
- $p_i(y)$ = fraction of data points labelled y in class i

Splitting Criteria II: Gini Index

Split Gini Index = expected gini index across all classes

$= \sum_i (n_i / n) \cdot \mathrm{Gini}(\text{class } i)$

$= \sum_i (n_i / n) \cdot (1 - \sum_y p_i(y)^2)$

Feature B

0

+	4
-	0

$n_0 = 4$
$p_0(+) = 1$
$p_0(-) = 0$

1

+	1
-	5

$n_1 = 6$
$p_1(+) = 1/6$
$p_1(-) = 5/6$

Gini Index	0	0.28

Split Gini Index = 0 + 0.6 * 0.28 ≈ 0.17

Notations

- n_i = # points in class i
- $n_i(y)$ = # data points labelled y in class i
- $p_i(y)$ = fraction of data points labelled y in class i

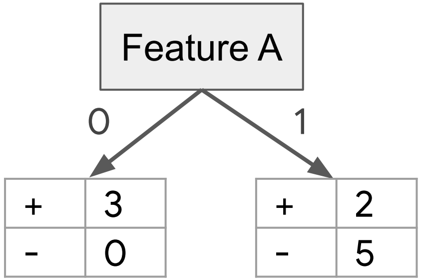
Splitting Criteria II: Gini Index

Split Gini Index = expected gini index across all classes

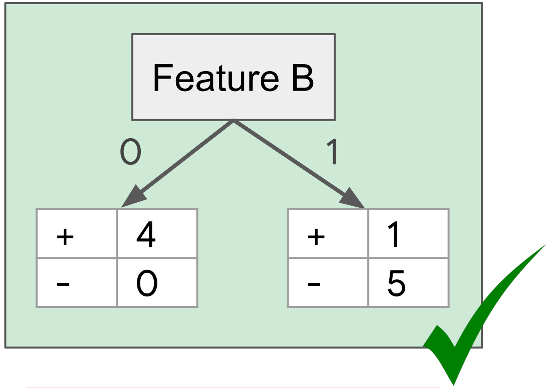
$$= \sum_i (n_i / n) \cdot \mathrm{Gini}(\text{class } i)$$

$$= \sum_i (n_i / n) \cdot (1 - \sum_y p_i(y)^2)$$

Criteria II: Select the split to minimize the split gini index



Split Gini Index = 0.29



Split Gini Index = 0.17

Entropy

A	B	Label
*	*	+
*	*	-
*	*	-
*	*	-
*	*	-
*	*	-
*	*	+
*	*	-
*	*	-
*	*	+

$\mathbf{n(+)} = 3$, $\mathbf{n(\text{-})} = 7$
$\mathbf{p(+)} = 3$, $\mathbf{p(\text{-})} = 0.7$

Entropy
= -0.3 * log(0.3)
-0.7 * log(0.7)
= 0.88

A measure of "uncertainty" of the label

Entropy = $-\sum_y p(y) \cdot \log p(y)$

Notations

- $\mathbf{n}$ = total # data points
- $\mathbf{n(y)}$ = # data points labelled y
- $\mathbf{p(y)}$ = fraction of data points labelled y
  $= n(y) / n$

Entropy

A	B	Label
*	*	1
*	*	3
*	*	2
*	*	3
*	*	3
*	*	3
*	*	1
*	*	3
*	*	3
*	*	3

Entropy
= -0.2 * log(0.2)
-0.1 * log(0.1)
-0.7 * log(0.7)
= 1.15

$\mathbf{n(1)} = 2, \mathbf{n(2)} = 1, \mathbf{n(3)} = 7$
$\mathbf{p(1)} = 0.2, \mathbf{p(2)} = 0.1, \mathbf{p(3)} = 0.7$

A measure of "uncertainty" of the label

Entropy = $-\sum_{y} p(y) \cdot \log p(y)$

Notations

- $\mathbf{n}$ = total # data points
- $\mathbf{n(y)}$ = # data points labelled y
- $\mathbf{p(y)}$ = fraction of data points labelled y
  $= n(y) / n$

Splitting Criteria III: Entropy

Conditional Entropy $= $ expected entropy across all classes

$$= \sum_i (n_i / n) \cdot \text{Entropy(class i)}$$

$$= \sum_i (n_i / n) \cdot \sum_y -p_i(y) \cdot \log p_i(y)$$

Feature A

0 | 1

+	3
-	0

+	2
-	5

$n_0 = 3$
$p_0(+) = 1$
$p_0(-) = 0$

$n_1 = 7$
$p_1(+) = 2/7$
$p_1(-) = 5/7$

Entropy	0	0.86

Conditional Entropy = 0 + 0.7 * 0.86 ≈ 0.60

Notations

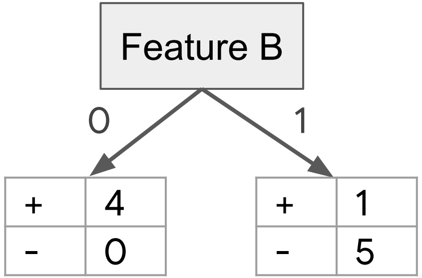
- n_i = # points in class i
- $n_i(y)$ = # data points labelled y in class i
- $p_i(y)$ = fraction of data points labelled y in class i

Splitting Criteria III: Entropy

Conditional Entropy = expected entropy across all classes

$= \sum_i (n_i / n) \cdot \text{Entropy}(\text{class } i)$

$= \sum_i (n_i / n) \cdot \sum_y -p_i(y) \cdot \log p_i(y)$



$n_0 = 4$
$p_0(+) = 1$
$p_0(-) = 0$

$n_1 = 6$
$p_1(+) = 1/6$
$p_1(-) = 5/6$

Entropy	0	0.65

Conditional Entropy = 0 + 0.6 * 0.65 ≈ 0.39

Notations

- n_i = # points in class i
- $n_i(y)$ = # data points labelled y in class i
- $p_i(y)$ = fraction of data points labelled y in class i

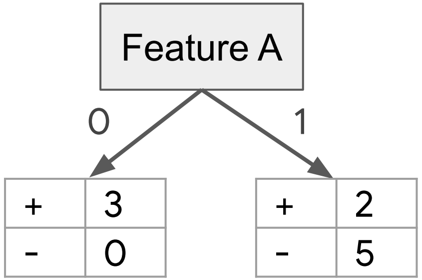
Splitting Criteria III: Entropy

Conditional Entropy = expected entropy across all classes

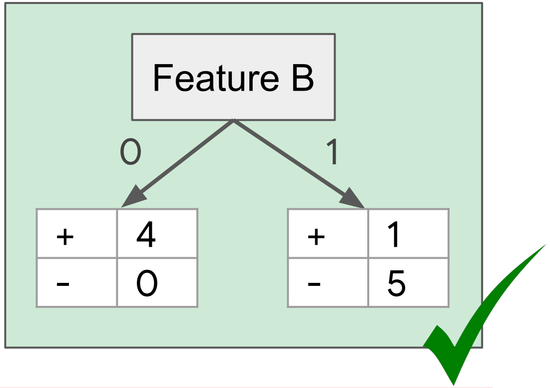
$= \sum_i (n_i / n) \cdot \text{Entropy(class i)}$

$= \sum_i (n_i / n) \cdot \sum_y -p_i(y) \cdot \log p_i(y)$

Criteria III: Select the split to minimize the conditional entropy

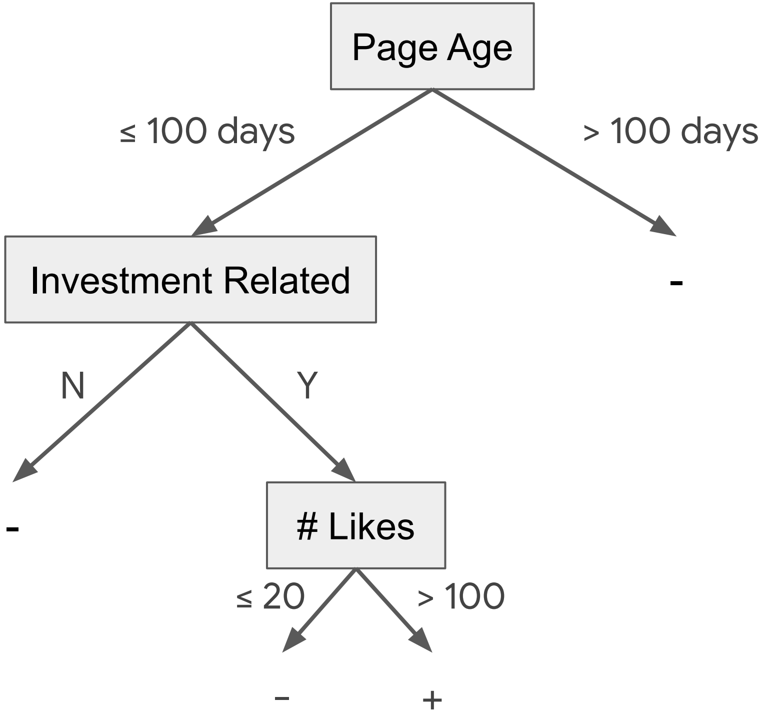


Conditional Entropy = 0.60



Conditional Entropy = 0.39

Learning Decision: Top-Down Algorithms



Build the tree in a ***top-down*** manner:

- Start from root
  - Select the “best” split
  - Recurse on left and right

Main Algorithmic Ingredients:

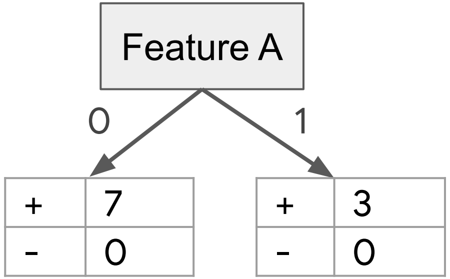
- Splitting Criteria
- **Stopping Condition**

Stopping Criteria

Before Split

+	10
-	0

After Split



Should we split at all?

No?

Stopping Criterion

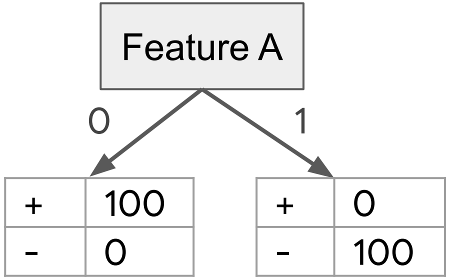
- All labels are the same ("pure")

Stopping Criteria

Before Split

+	100
-	100

After Split



Should we split at all?

Yes?

Stopping Criterion

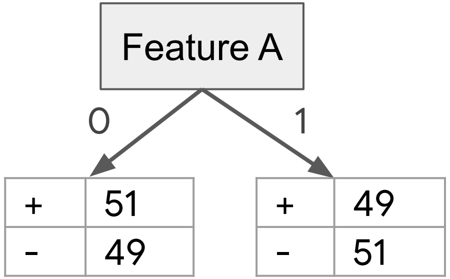
- All labels are the same ("pure")

Stopping Criteria

Before Split

+	100
-	100

After Split



Should we split at all?

Probably No?

Stopping Criterion

- All labels are the same ("pure")
- "Gain" is small
    - E.g. Gini index before and after differ by smaller some threshold
- # Data points are smaller than some threshold
- The depth of the node is larger than a certain threshold

Summary: Splitting & Stopping Criterion

Splitting Criterion

- Minimize one of the following metric (weighted by the group sizes)
- **Misclassification rate:** $1 - \max_y p(y)$
- **Gini Index:** $1 - \sum_y p(y)^2$
- **Entropy:** $-\sum_y p(y) \cdot \log p(y)$

Stopping Criterion

- All labels are the same ("pure")
- "Gain" is small
  - E.g. Gini index before and after differ by smaller some threshold
- # Data points are smaller than some threshold
- The depth of the node is larger than a certain threshold

Summary: Pseudo-code

```
BuildTree(X, Y):
  if (X, Y) meets stopping criteria :
    RETURN EndNode(prediction=Majority(Y))
  else:
    S ← Best splitting criteria for (X, Y)
    v ← new InternalNode(split_criteria=S)
    For each class X_i, Y_i according to S :
        v.add_child(BuildTree(X_i, Y_i))
    RETURN v
```

Summary: Pseudo-code

```
BuildTree(X, Y):
  if (X, Y) meets stopping criteria :
    RETURN EndNode(prediction=Majority(Y))
  else:
    S ← Best splitting criteria for (X, Y)
    v ← new InternalNode(split_criteria=S)
    For each class X_i, Y_i according to S :
        v.add_child(BuildTree(X_i, Y_i))
    RETURN v
```

- Try all possible splits
- For each split, calculates Misclassification Error, Gini Index or Entropy
- Pick the best one

Summary: Pseudo-code

```
BuildTree(X, Y):
  if (X, Y) meets stopping criteria :
    RETURN EndNode(prediction=Majority(Y))
  else:
    S ← Best splitting criteria for (X, Y)
    v ← new InternalNode(split_criteria=S)
    For each class X_i, Y_i according to S:
        v.add_child(BuildTree(X_i, Y_i))
    RETURN v
```

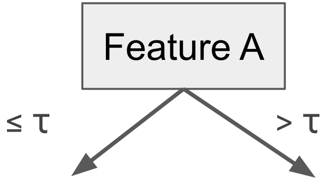
- Try all possible splits
- For each split, calculates Misclassification Error, Gini Index or Entropy
- Pick the best one

If features are discrete, runs in time O(n * (# features))

What if some features are continuous?

Splitting Continuous Feature

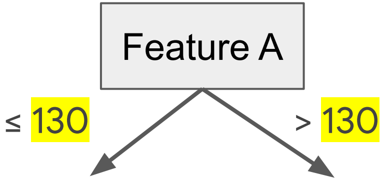
A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+



Try τ = all values that appear in the data

Splitting Continuous Feature

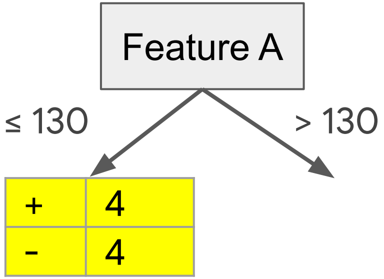
A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+



Try τ = all values that appear in the data

Splitting Continuous Feature

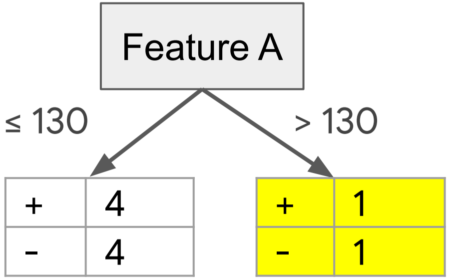
A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+



Try τ = all values that appear in the data

Splitting Continuous Feature

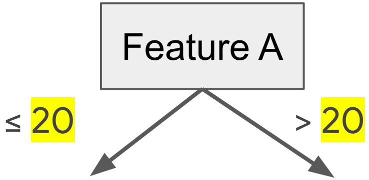
A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+



Try τ = all values that appear in the data

Splitting Continuous Feature

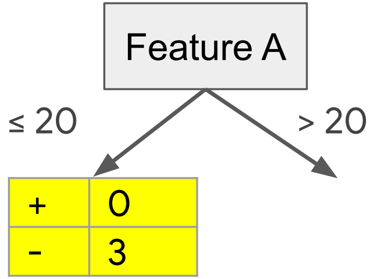
A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+



Try τ = all values that appear in the data

Splitting Continuous Feature

A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+



Try τ = all values that appear in the data

Splitting Continuous Feature

A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+

Feature A

≤ 20

+	0
-	3

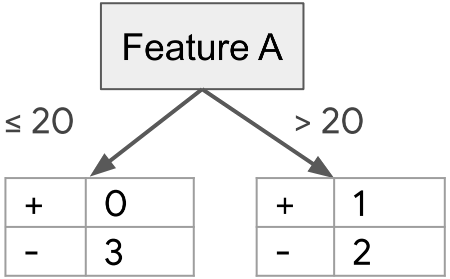
> 20

+	1
-	2

Try τ = all values that appear in the data

Splitting Continuous Feature

A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+



Try τ = all values that appear in the data

$\underbrace{\text{n choices of } \tau \quad + \quad \text{Each } \tau \text{ requires O(n) time to compute the metric}}$

Total running time = $O(n^2)$

Efficiently Splitting Continuous Feature

A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+

Sort based on feature value

A	Label
10	-
15	-
20	-
35	-
40	+
50	+
60	+
130	+
150	-
200	+

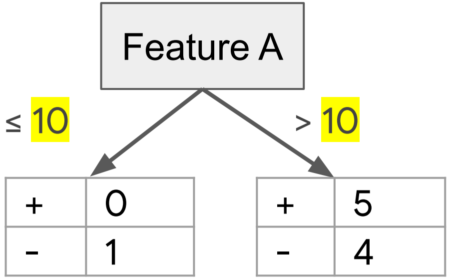
Try τ from small to large

Efficiently Splitting Continuous Feature

A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+

Sort based on feature value

A	Label
10	-
15	-
20	-
35	-
40	+
50	+
60	+
130	+
150	-
200	+



Try τ from small to large

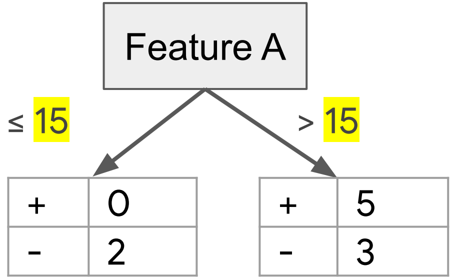
Efficiently Splitting Continuous Feature

A	Label
130	+
20	-
50	+
200	+
10	-
150	-
40	+
15	-
35	-
60	+

Sort based on feature value

O(n log n) time

A	Label
10	-
15	-
20	-
35	-
40	+
50	+
60	+
130	+
150	-
200	+



Try τ from small to large

Each update takes O(1) time

Total running time = O(n log n)

Summary: Pseudo-code

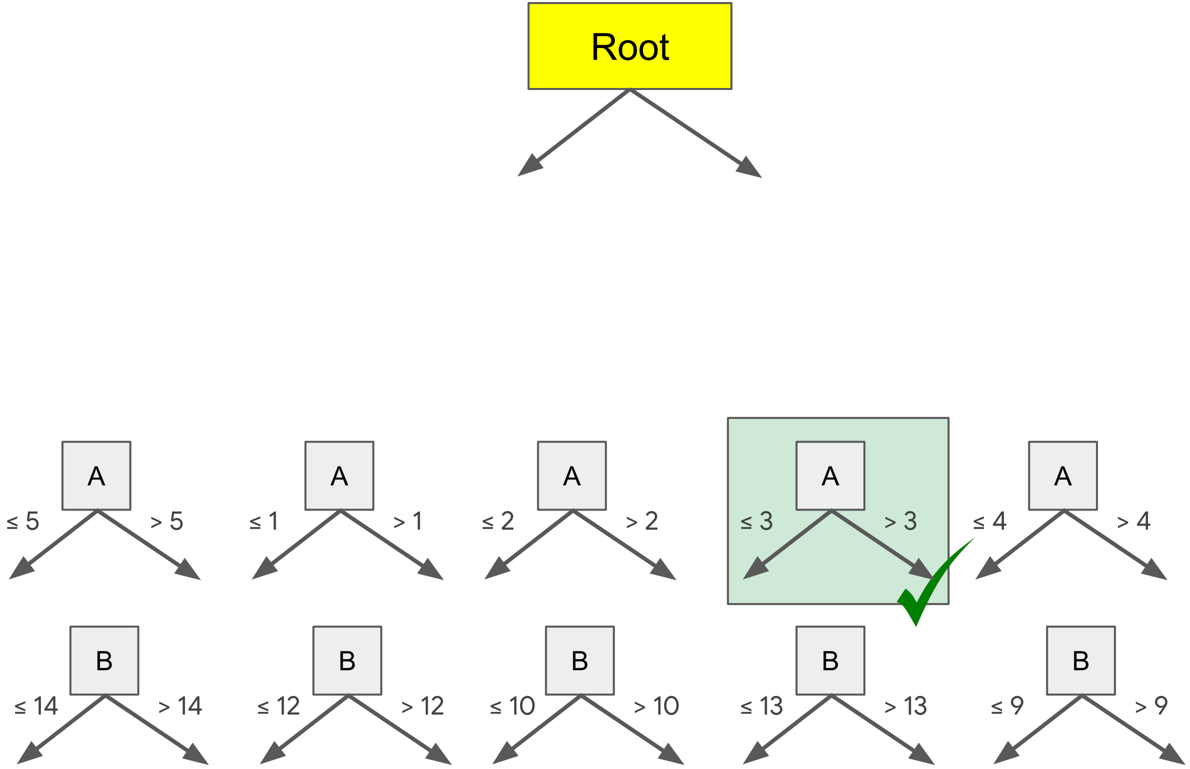
```
BuildTree(X, Y):
  if (X, Y) meets stopping criteria :
    RETURN EndNode(prediction=Majority(Y))
  else:
    S ← Best splitting criteria for (X, Y)
    v ← new InternalNode(split_criteria=S)
    For each class X_i, Y_i according to S:
        v.add_child(BuildTree(X_i, Y_i))
    RETURN v
```

- Try all possible splits
- For each split, calculates Misclassification Error, Gini Index or Entropy
- Pick the best one

If features are discrete, runs in time O(n * (# features))

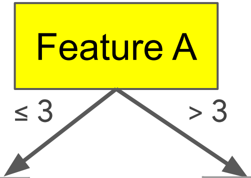
Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



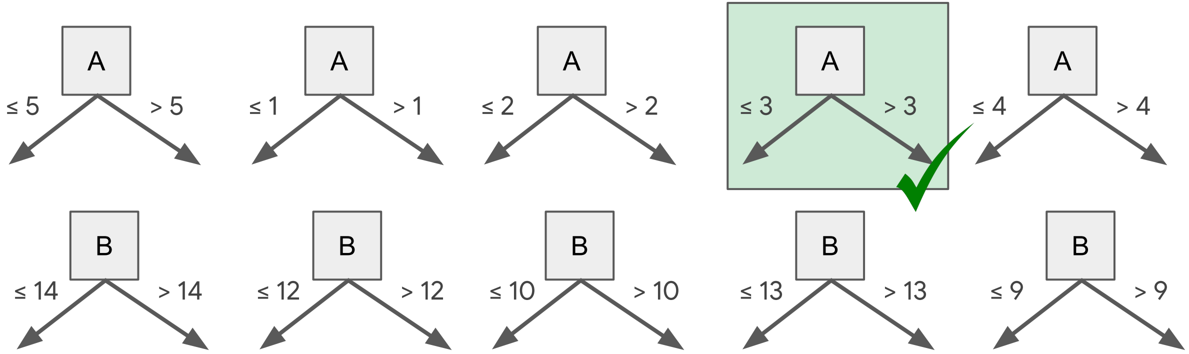
Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



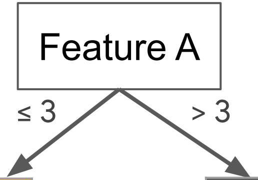
A	B	Label
1	12	+
2	10	-
3	13	+

A	B	Label
5	14	-
6	15	-
4	9	-



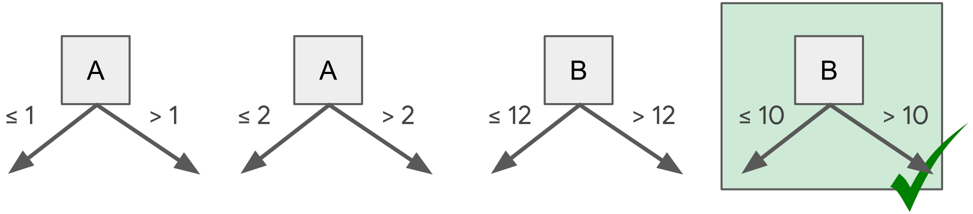
Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



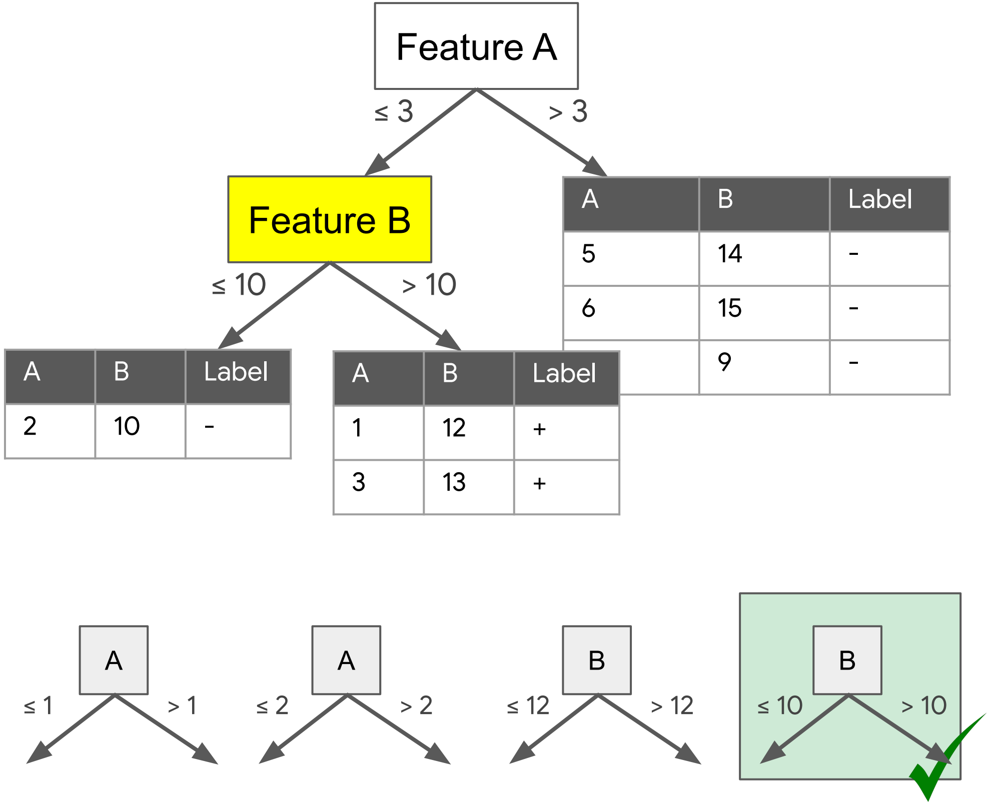
A	B	Label
1	12	+
2	10	-
3	13	+

A	B	Label
5	14	-
6	15	-
4	9	-



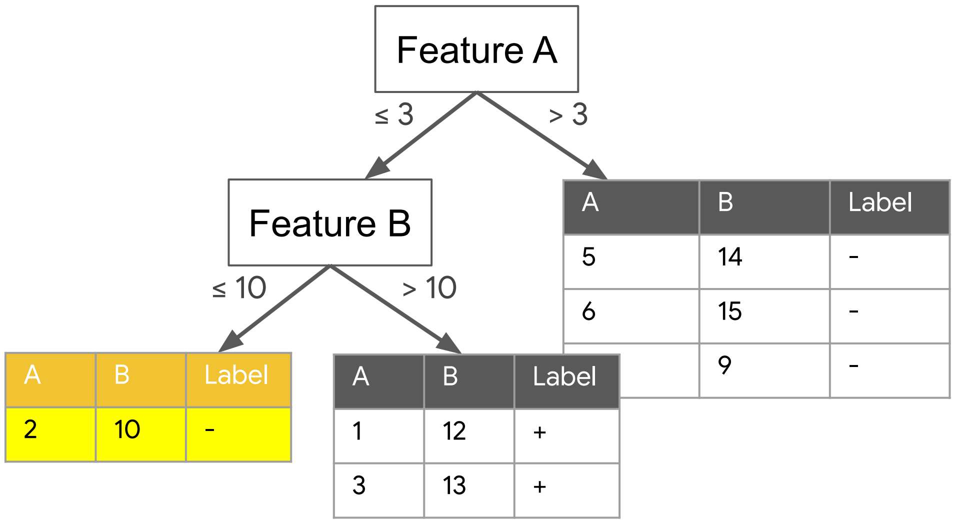
Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



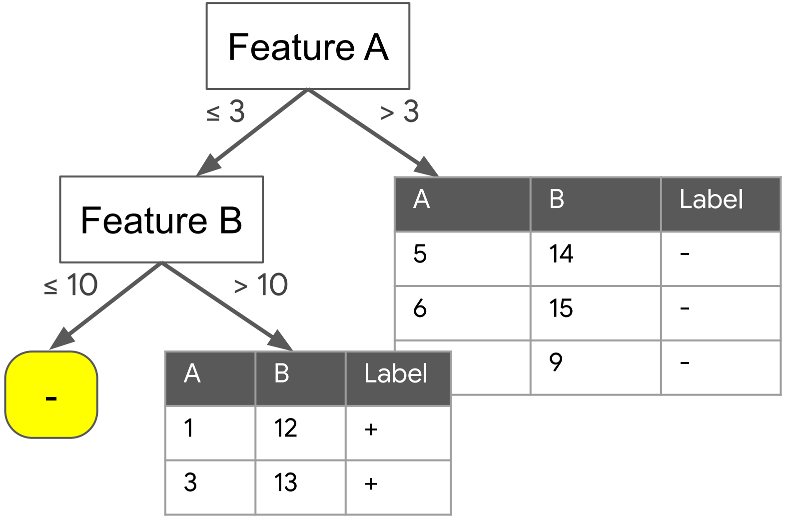
Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



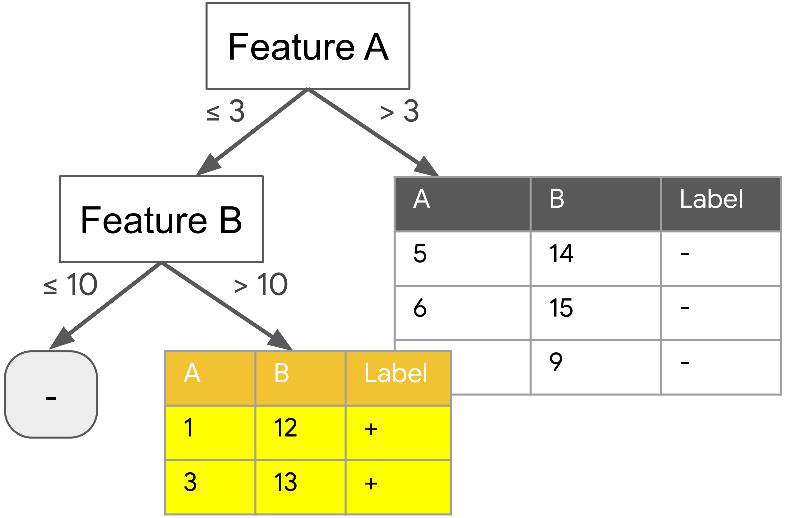
Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



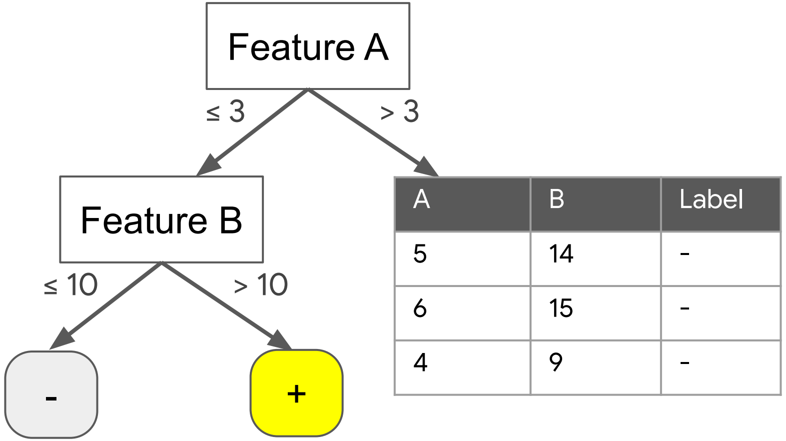
Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



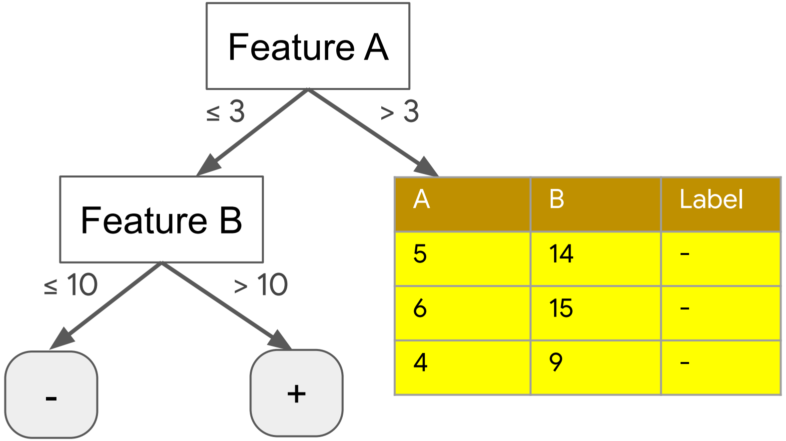
Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



Algorithm Walkthrough

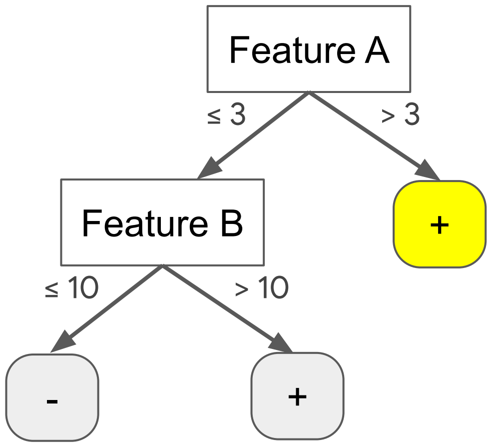
A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



A	B	Label
5	14	-
6	15	-
4	9	-

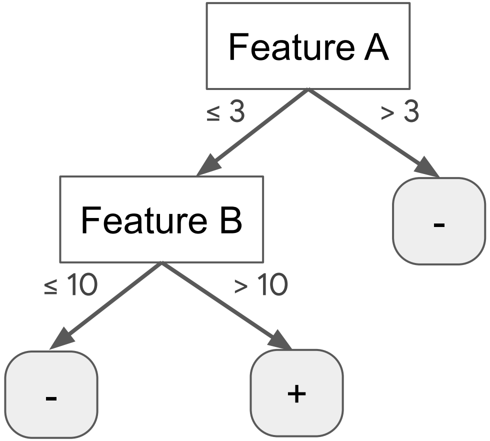
Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



Algorithm Walkthrough

A	B	Label
5	14	-
6	15	-
1	12	+
2	10	-
3	13	+
4	9	-



Colab

Main colab: Link

[Optional] Intro to Pandas: Link

Solution: Link

Free Software for Decision Trees

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.**DecisionTreeClassifier**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : *int or float, default=1*
The minimum number of samples required to be at a leaf node. A split point at any depth will only be

TensorFlow > Resources > Decision Forests > API Reference

Was this helpful?

tfdf.keras.CartModel

View source on GitHub

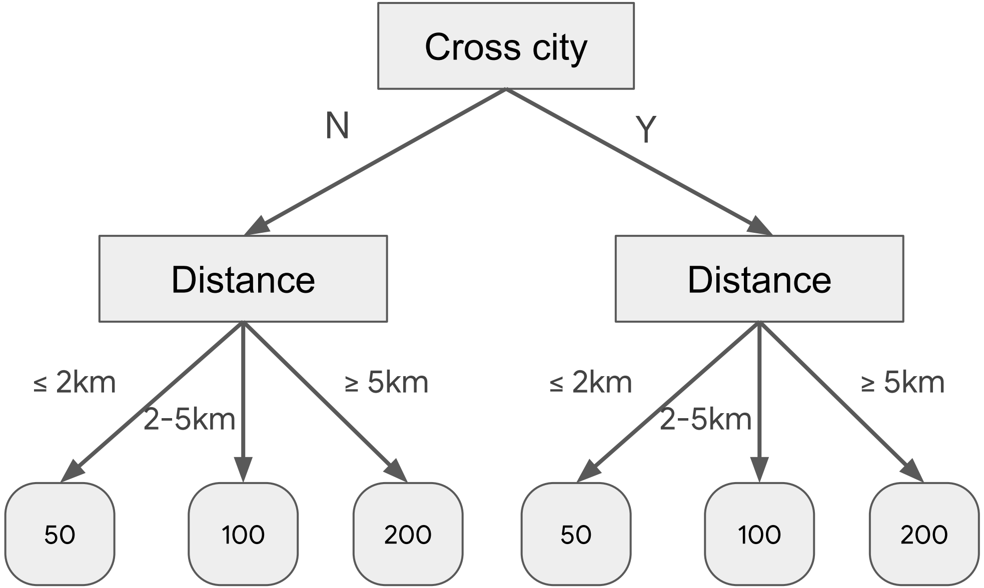
Cart learning algorithm.

Inherits From: `CartModel`, `CoreModel`, `InferenceCoreModel`

```
tfdf.keras.CartModel(
    task: Optional[TaskType] = core.Task.CLASSIFICATION,
    features: Optional[List[core.FeatureUsage]] = None,
    exclude_non_specified_features: Optional[bool] = False,
    preprocessing: Optional['tf.keras.models.Functional'] = None,
    postprocessing: Optional['tf.keras.models.Functional'] = None,
    training_preprocessing: Optional['tf.keras.models.Functional'] = None,
    ranking_group: Optional[str] = None,
    uplift_treatment: Optional[str] = None,
    temp_directory: Optional[str] = None,
    verbose: int = 1,
    hyperparameter_template: Optional[str] = None,
    advanced_arguments: Optional[tfdf.keras.AdvancedArguments] = None,
    num_threads: Optional[int] = None,
    name: Optional[str] = None,
    max_vocab_count: Optional[int] = 2000,
    try_resume_training: Optional[bool] = True,
    check_dataset: Optional[bool] = True,
    tuner: Optional[tfdf.tuner.Tuner] = None,
```

Regression Trees

Regression Tree



Example

- Grab / taxi fee prediction
- **Features:**
  - Cross city
    (e.g. Bangkok ⇒ Pathum Thani)
  - Distance (in km)

Errors for Regression

A	B	Label
*	*	10
*	*	5
*	*	6
*	*	100
*	*	20
*	*	30
*	*	1
*	*	95
*	*	75
*	*	6

What if we have to predict the label without looking at any feature?

Best prediction $\tilde{y}$ depends on error we try to minimize

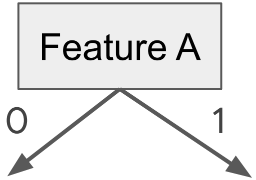
Squared Error $\sum (\tilde{y} - y_i)^2$

Best prediction = Average

Absolute Error $\sum |\tilde{y} - y_i|$

Best prediction = Median

Splitting Criterion



Criteria I: Squared Error

Split Squared Error
= sum squared error across all classes
= $\sum_i$ min-squared-error(class i)

Criteria I: Select the split to minimize the split squared error

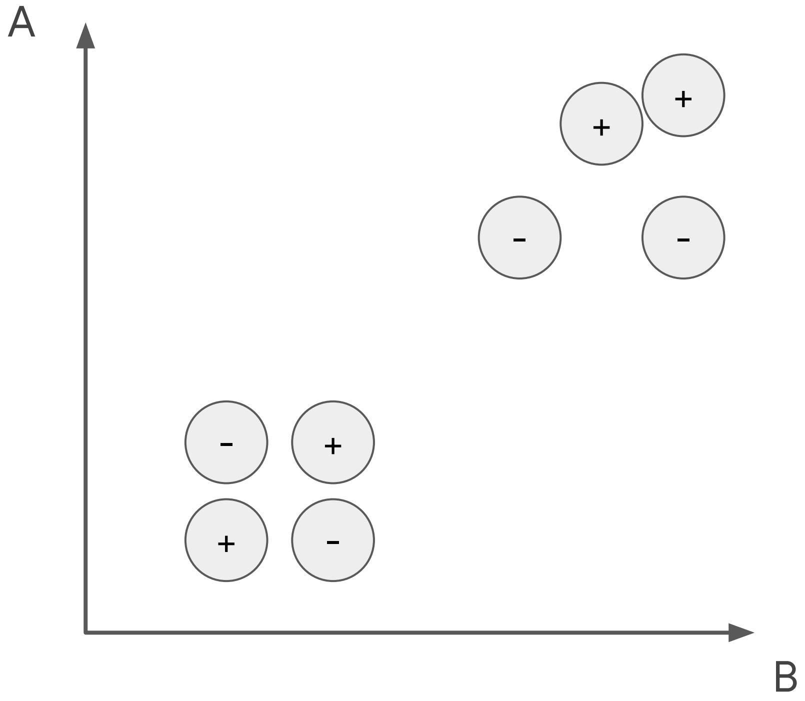
Criteria II: Absolute Error

Split Absolute Error
= sum absolute error across all classes
= $\sum_i$ min-absolute-error(class i)

Criteria II: Select the split to minimize the split absolute error

Advanced Topics

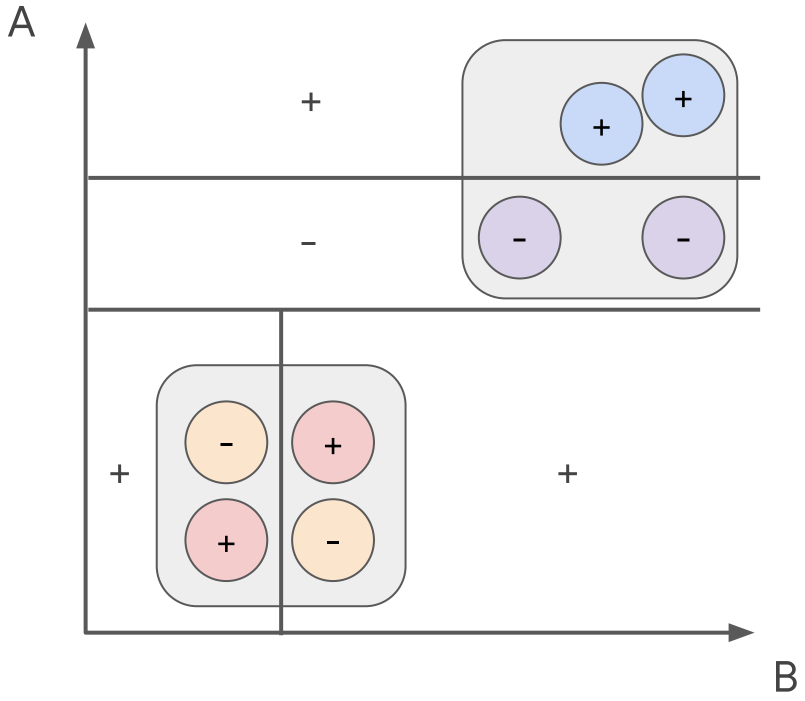
Learning Decision Tree: Bottom-Up Approach



High-level ideas

- Cluster each class to k clusters
- Use agglomerative clustering to build a tree in a bottom up manner
- Try to find split that most closely matches the clusters

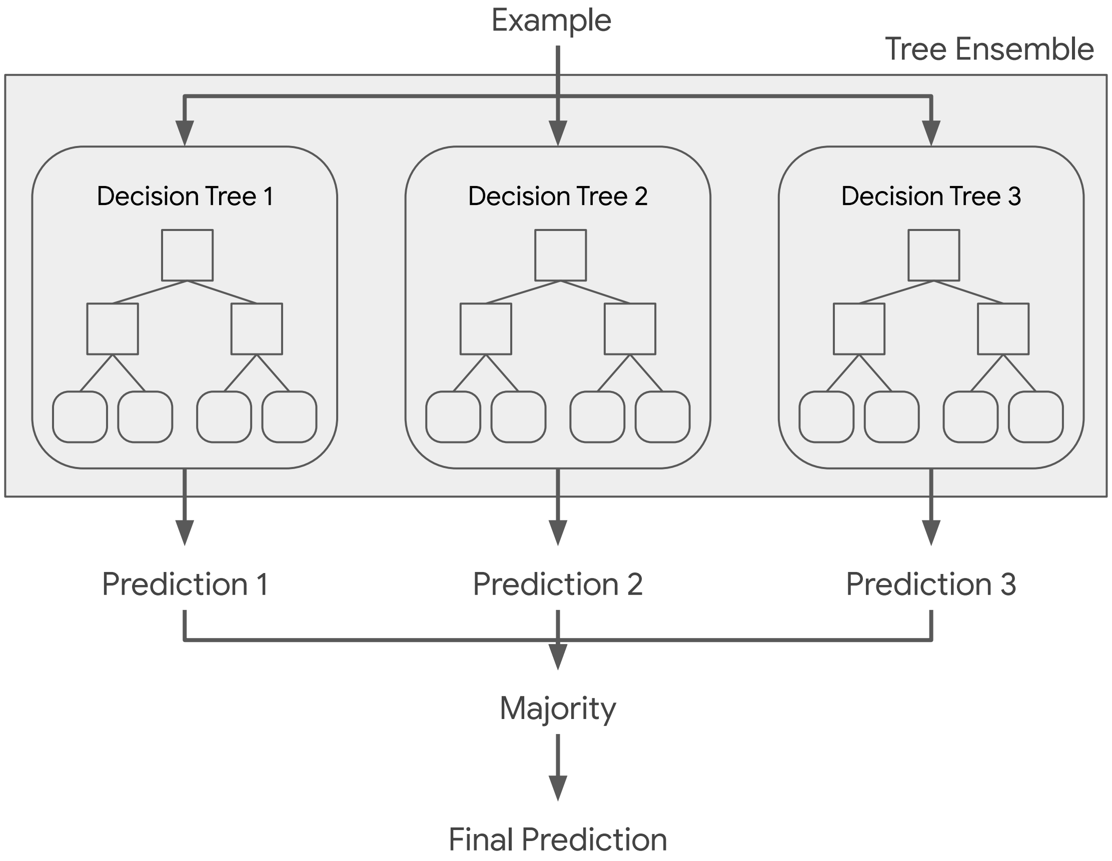
Learning Decision Tree: Bottom-Up Approach



High-level ideas

- Cluster each class to k clusters
- Use agglomerative clustering to build a tree in a bottom up manner
- Try to find split that most closely matches the clusters

Tree Ensembles

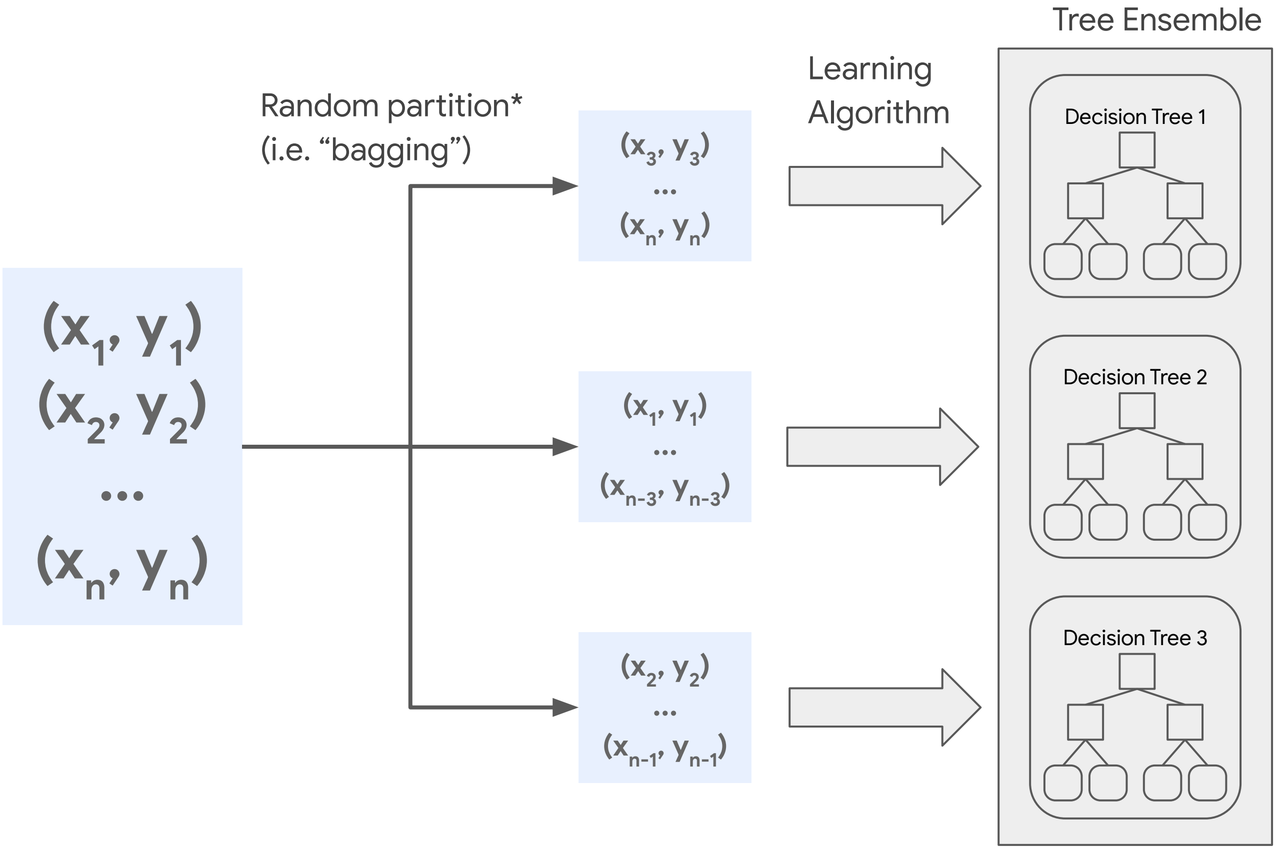


Tree Ensemble

- Multiple decision trees
- To predict:
  - Run prediction on each tree
  - Take the majority of the predictions (or average / median for regression)

- Can improve accuracy
- Prediction harder to explain

Learning Tree Ensembles: **Random Forest**

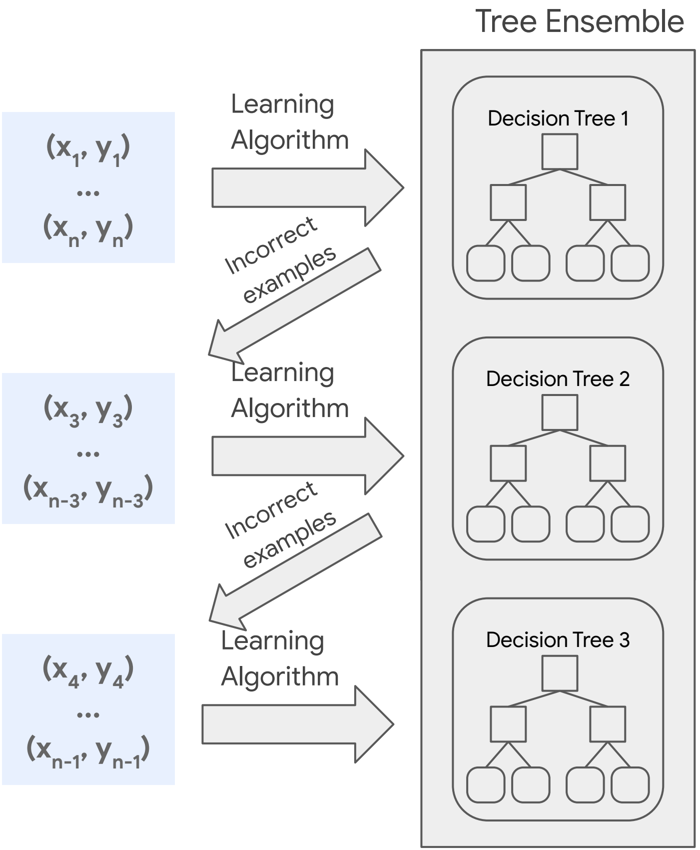


Random Forest

- Randomly “bagging” the training dataset
- Train a decision tree on each bag

- More “stable” than training a single decision tree
- Can improve accuracy

Learning Tree Ensembles: **Adaboost**



Adaboost

- Repeat the following:
    - Training a new decision tree on samples that are incorrect on the model so far

- Can improve accuracy
- Theoretical guarantee: "weak" ⇒ "strong" learner

Free Software for Tree Ensembles

sklearn.ensemble.RandomForestClassifier

class `sklearn.ensemble.`**`RandomForestClassifier`**(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='sqrt'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [sourc

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

For a comparison between tree-based ensemble models see the example Comparing Random Forests and Histogram Gradie Boosting models.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation. Note: Thi parameter is tree-specific.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leav contain less than min_samples_split samples.

sklearn.ensemble.AdaBoostClassifier

class `sklearn.ensemble.`**`AdaBoostClassifier`**(*estimator=None*, ***, *n_estimators=50*, *learning_rate=1.0*, *algorithm='SAMME.R'*, *random_state=None*, *base_estimator='deprecated'*) [source]

An AdaBoost classifier.

An AdaBoost [1] classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

This class implements the algorithm known as AdaBoost-SAMME [2].

Read more in the User Guide.

New in version 0.14.

Parameters:

estimator : *object, default=None*
The base estimator from which the boosted ensemble is built. Support for sample weighting is required, as well as proper `classes_` and `n_classes_` attributes. If `None`, then the base estimator is **`DecisionTreeClassifier`** initialized with `max_depth=1`.

New in version 1.2: `base_estimator` was renamed to `estimator`.

n_estimators : *int, default=50*
The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. Values must be in the range `[1, inf)`.

learning_rate : *float, default=1.0*
Weight applied to each classifier at each boosting iteration. A higher learning rate increases the contribution of each classifier. There is a trade-off between the `learning_rate` and `n_estimators` parameters. Values must be in the range `(0.0, inf)`.

Summary

- Decision trees
  - Classification
  - Regression
- How to learn decision trees
  - Top-down
  - Bottom-up
- Combining multiple decision trees to improve accuracy

Thank you!