Northeastern University
Khoury College of
Computer Sciences

Naive Bayes & Decision Trees

DS 4400 | Machine Learning and Data Mining I

Zohair Shafi

Spring 2026

Monday | February 23, 2026

Today's Outline

- Naive Bayes
- Decision Trees

Linear Discriminant Analysis (LDA)

- LDA is a **generative** classifier

Linear Discriminant Analysis (LDA)

Discriminative

- LDA is a **generative** classifier
- Instead of directly predicting $\mathbb{P}(Y|X)$ like logistic regression, it models the joint distribution $\mathbb{P}(X, Y)$ by modeling:
  - $\mathbb{P}(X|Y)$: How features are distributed **within each class**
  - $\mathbb{P}(Y)$: Prior probability of each class
- Then it uses Bayes' theorem to compute $\mathbb{P}(Y|X)$ for classification.

$$\mathbb{P}(y|x) = \frac{\mathbb{P}(x|y) \cdot P(y)}{P(x)}$$

Linear Discriminant Analysis (LDA)

- LDA is a **generative** classifier
- **Key idea**:
  - Assume each class generates data from a Gaussian distribution.
  - Find the decision boundary that optimally separates these Gaussian clouds.

Linear Discriminant Analysis (LDA)
Assumptions

- Assumption 1:
  - Class conditional probabilities are Gaussian (normal distribution)
  - $\mathbb{P}(X \mid Y = k) = \mathcal{N}(X \mid \mu_k, \Sigma)$

Linear Discriminant Analysis (LDA)

Assumptions

- Assumption 1:
  - Class conditional probabilities are Gaussian (normal distribution)
  - $\mathbb{P}(X \mid Y = k) = \mathcal{N}(X \mid \mu_k, \Sigma)$
- Assumption 2:
  - All classes share the same co-variance matrix Σ (homoscedasticity)

Linear Discriminant Analysis (LDA)

Assumptions

- Assumption 1:
  - Class conditional probabilities are Gaussian (normal distribution)
  - $\mathbb{P}(X \mid Y = k) = \mathcal{N}(X \mid \mu_k, \Sigma)$
- Assumption 2:
  - All classes share the same co-variance matrix Σ (homoscedasticity)
- Assumption 3:
  - Classes differ only in their means μ_k
- These assumptions lead to linear decision boundaries - hence **Linear** Discriminant Analysis.

Linear Discriminant Analysis (LDA)

Computing Predictions

Gaussian Formula

log

$$\mathbb{P}(Y = k \mid X = x) = \frac{\boxed{\mathbb{P}(X = x \mid Y = k)} \cdot \mathbb{P}(Y = k)}{\boxed{\cancel{\mathbb{P}(X = x)}}}$$

Bayes' Rule

argmax
k

Linear Discriminant Analysis (LDA)

Computing Predictions

Finally:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log P(Y=k)$$

We can also re-write as:

$$\delta_k(x) = \theta_{1_k}^T x + \theta_{0_k}$$

Where:

$$\theta_{1_k} = \Sigma^{-1}\mu_k$$

$$\theta_{0_k} = -\frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log P(Y=k)$$

Naive Bayes Classifier

Gaussian PDF

$\not\downarrow$

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

argmax

Bayes' Formula. $\leftarrow$ $$\mathbb{P}(Y = k \,|\, X = x) = \frac{\boxed{\mathbb{P}(X = x \,|\, Y = k)}\;\mathbb{P}(Y = k)}{\boxed{\cancel{\mathbb{P}(X = x)}}}$$

$\downarrow$

① Discrete

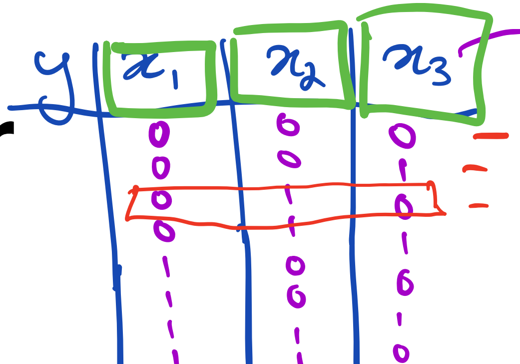
Naive Bayes Classifier

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\boxed{\mathbb{P}(X = x \,|\, Y = k)} \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

- The challenge:
  - Estimating $\mathbb{P}(X \,|\, Y = k)$ for high-dimensional X is difficult.

Naive Bayes Classifier



$X \in m \times 3$

binary .

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

$$\mathbb{P}(Y=k \mid X=x) = \frac{\boxed{\mathbb{P}(X=x \mid Y=k)} \cdot \mathbb{P}(Y=k)}{\mathbb{P}(X=x)}$$

$2^3 = 8$

- The challenge:

$\mathbb{P}(x_1) \cdot \mathbb{P}(x_2) \cdot \mathbb{P}(x_3)$ $\quad 2^{10} =$

  - Estimating $\mathbb{P}(X \mid Y=k)$ for high-dimensional X is difficult.
  - With d features, each taking $\overset{2}{v}$ possible values, we'd need to estimate $\boxed{v^d}$ parameters per class - **exponential in dimensionality**.

Naive Bayes Classifier

Naive Bayes assumes features are conditionally independent given the class

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

$$\mathbb{P}(Y = k \mid X = x) = \frac{\mathbb{P}(X = x \mid Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

- The challenge:
  - Estimating $\mathbb{P}(X \mid Y = k)$ for high-dimensional X is difficult.
  - With d features, each taking v possible values, we'd need to estimate $\boxed{v^d}$ parameters per class - **exponential in dimensionality**.

Naive Bayes Classifier

logistic regression
LDA
} → prob. have meaning
well calibrated.

- Like LDA, Naive Bayes is a generative classifier using Bayes' theorem

$$\mathbb{P}(Y = k \,|\, X = x) = \frac{\mathbb{P}(X = x \,|\, Y = k) \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

argmax
K

- $\mathbb{P}(X = x \,|\, Y = k) = \mathbb{P}(x_1, x_2, x_3, \ldots, x_d \,|\, Y = k) = \Pi_{j=1}^{d} \mathbb{P}(x_j \,|\, Y = k)$
- This is almost always wrong in practice - features are usually correlated.
- But it reduces parameters from exponential to linear: d parameters per class instead of v^d

Main point

Naive Bayes Classifier

$$\mathbb{P}(y|x) = \frac{\boxed{\mathbb{P}(x|y)} \cdot \mathbb{P}(y)}{\mathbb{P}(x)}$$

$\mathbb{P}$("free", "money" | spam)

- Example:
  - For spam classification with words “free” and “money”

    $\prod$

    - Reality: $\underbrace{\mathbb{P}(\text{"free", "money"} \mid \text{spam})} \neq \underbrace{\mathbb{P}(\text{"free"} \mid \text{spam})} \cdot \underbrace{\mathbb{P}(\text{“money"} \mid \text{spam})}$
    - These words are correlated in spam emails
    - Naive Bayes pretends they're **independent**

x_1 | x_2 | x_3

$X \rightarrow$ $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, $\vdots$, $x^{(n)}$

logistic.
each row is independent.

Naive Bayes Classifier

Want to predict:

$$\mathbb{P}(Y = k \mid X = x) = \frac{\boxed{\mathbb{P}(X = x \mid Y = k)} \cdot \mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

for all colums

Naive Assumption:

$$\mathbb{P}(X = x \mid Y = k) = \mathbb{P}(x_1, x_2, x_3, \ldots, x_d \mid Y = k) = \boxed{\Pi_{j=1}^{d}} \mathbb{P}(x_j \mid Y = k)$$

Naive Bayes Classifier

$$\hat{y} = arg\max_k \mathbb{P}(Y = k) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \,|\, Y = k)$$

For numerical stability, we take logs

$$\hat{y} = \underbrace{arg\max_k log(\mathbb{P}(Y = k)) + \sum_{j=1}^{d} log\left(\boxed{\mathbb{P}(x_j \,|\, Y = k)}\right)}$$

$$\log\left[P(Y=k) \cdot \Pi\, P(x_j \,|\, Y=k)\right]$$

$$\log P(Y=k) + \sum \log(P(x_j \,|\, Y=k))]$$

Naive Bayes Classifier

$$\hat{y} = arg\max_{k} \mathbb{P}(Y = k) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \,|\, Y = k)$$

For numerical stability, we take logs

$$\hat{y} = arg\max_{k} log(\mathbb{P}(Y = k)) + \sum_{j=1}^{d} log(\mathbb{P}(x_j \,|\, Y = k))$$

How do you find $\underbrace{\mathbb{P}(x_j \,|\, Y = k)}$?

Naive Bayes Classifier

Gaussian Naive Bayes

For continuous features, assume each feature follows a Gaussian Distribution

$$\mathbb{P}(x_j \mid Y = k) = \frac{1}{\sqrt{2\pi\sigma_{kj}^2}} \cdot e^{-\frac{(x_j - \mu_{kj})^2}{2\sigma_{kj}^2}}$$

Gaussian.

Each class-feature combination has its own mean μ_{kj} and variance σ_{kj}^2

Naive Bayes Classifier

$$\mu_{kj} = \frac{1}{N_k} \sum_{i:y_i=k} x_{ij}$$

x → columns (features) are independent

$$\sigma^2_{kj} = \frac{1}{N_k} \sum_{i:y_i=k} (x_{ij} - \mu_{kj})^2$$

Gaussian Naive Bayes

For continuous features, assume each feature follows a Gaussian Distribution

(1) $\mathbb{P}(y|x) = \frac{\boxed{P(x|y)}\,P(y)}{\cancel{P(x)}}$

(2) argmax K

$$\mathbb{P}(x_j \,|\, Y = k) = \frac{1}{\sqrt{2\pi\sigma^2_{kj}}} \cdot e^{-\frac{(x_j - \mu_{kj})^2}{2\sigma^2_{kj}}}$$

(3) log

Each class-feature combination has its own mean μ_{kj} and variance σ^2_{kj}

(4)

Naive Bayes Classifier

j: "free"
k: spam

Multinomial Naive Bayes

For discrete features/count data like word frequencies

$$\mathbb{P}(x_j \mid Y = k) = \boxed{\theta_{kj}^{x_j}}$$

Where θ_{kj} is the probability of feature j in class k and x_j is the count of feature j

$$\boxed{\theta_{kj}} = \underbrace{\frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}}$$

Naive Bayes Classifier

$p^{y} \cdot (1-p)^{(1-y)}$

$p = \theta_{kj}$

Bernoulli Naive Bayes

For binary data like word frequencies

$$\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j} \cdot (1 - \theta_{kj})^{1 - x_j}$$

Where θ_{kj} is the probability of feature j in class k and x_j is the count of feature j

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total samples in class } k}$$

Naive Bayes Classifier

Example

$P(Y = \text{spam}) = 2/4 = 0.5$

$P(Y = \text{not spam}) = 2/4 = 0.5$

Task

Classify emails as spam or not spam

Training Data

Document (emails)	x_1 = "free"	x_2 = "money"	x_3 = "meeting"	x_4 = "lunch"	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

Task

Classify emails as spam or not spam

$$\mathbb{P}(\textbf{spam}) = \frac{2}{4} = 0.5$$

$$\mathbb{P}(\textbf{not spam}) = \frac{2}{4} = 0.5$$

Training Data

Document	x_1 = "free"	x_2 = "money"	x_3 = "meeting"	x_4 = "lunch"	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

$\mathbb{P}(\text{"free"}|\text{spam}) = \frac{3+2}{8} = \frac{5+1}{8} = \frac{6}{8} = 0$

Task

Classify emails as spam or not spam

Training Data

Document	x_1 = "free"	x_2 = "money"	x_3 = "meeting"	x_4 = "lunch"	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

$$\mathbb{P}(\textbf{spam}) = \frac{2}{4} = 0.5$$

$$\mathbb{P}(\textbf{not spam}) = \frac{2}{4} = 0.5$$

For **spam** class:

Laplace Smoothing

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = \underline{0.75}$

Naive Bayes Classifier
Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

Task
Classify emails as spam or not spam

Training Data

Document	x_1 = "free"	x_2 = "money"	x_3 = "meeting"	x_4 = "lunch"	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

$$\mathbb{P}(\textbf{spam}) = \frac{2}{4} = 0.5$$

$$\mathbb{P}(\textbf{not spam}) = \frac{2}{4} = 0.5$$

$\boxed{\text{For } \textbf{spam} \text{ class:}}$ → Not Spam

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"}|\text{spam}) = \frac{(3+1)}{(8)} = \frac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$

Naive Bayes Classifier

Example

$P(x \mid \text{spam})$
$P(x \mid \text{not spam})$

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

$P(y \mid x)$

Task

Classify emails as spam or not spam

$\boxed{\mathbb{P}(\textbf{spam})} = \frac{2}{4} = 0.5$

$\boxed{\mathbb{P}(\textbf{not spam})} = \frac{2}{4} = 0.5$

Training Data

Document	x_1 = "free"	x_2 = "money"	x_3 = "meeting"	x_4 = "lunch"	Class
1	3	2	0	0	spam
2	2	1	0	0	spam
3	0	0	2	1	not spam
4	0	0	1	2	not spam

For **not spam** class:

- $\mathbb{P}(\text{"free"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

$$\underbrace{\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j}}$$

free money meeting lunch

1 1 0 0

For **spam** class:

- $\mathbb{P}(\text{"free"}|\text{spam}) = \dfrac{(5+1)}{(8)} = \dfrac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"}|\text{spam}) = \dfrac{(3+1)}{(8)} = \dfrac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"}|\text{spam}) = \dfrac{(0+1)}{(8)} = \dfrac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"}|\text{spam}) = \dfrac{(0+1)}{(8)} = \dfrac{1}{8} = 0.125$

New Email: $x =$ "free money"

For **not spam** class:

- $\mathbb{P}(\text{"free"}|\text{not spam}) = \dfrac{(0+1)}{(6)} = \dfrac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"}|\text{not spam}) = \dfrac{(0+1)}{(6)} = \dfrac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"}|\text{not spam}) = \dfrac{(3+1)}{(6)} = \dfrac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"}|\text{not spam}) = \dfrac{(3+1)}{(6)} = \dfrac{4}{6} = 0.66$

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

Naive Bayes Classifier
Example

$$\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j}$$

For **spam** class:

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"}|\text{spam}) = \frac{(3+1)}{(8)} = \frac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$

For **not spam** class:

- $\mathbb{P}(\text{"free"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$

New Email: x = “free money”

Want to find

$$\boxed{\mathbb{P}(spam \mid x)} = \mathbb{P}(Y = spam) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \mid Y = spam)$$

argmax

Naive Bayes Classifier

Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

$$\mathbb{P}(x_j \,|\, Y = k) = \theta_{kj}^{\boxed{x_j}}$$

For **spam** class:

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"}|\text{spam}) = \frac{(3+1)}{(8)} = \frac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$

For **not spam** class:

- $\mathbb{P}(\text{"free"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$

Not spam

1 1

0.5

New Email: $x =$ "free money"

$$\mathbb{P}(spam \,|\, x) = \boxed{\mathbb{P}(Y = spam)} \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \,|\, Y = spam)$$

$$\mathbb{P}(spam \,|\, x) = \boxed{0.5} \cdot (0.75)^{\boxed{1}} \cdot (0.5)^{\boxed{1}} = 0.1875$$

Naive Bayes Classifier
Example

$$\theta_{kj} = \frac{\text{count of feature } j \text{ in class } k}{\text{total count of all features in class } k}$$

$$\mathbb{P}(x_j \mid Y = k) = \theta_{kj}^{x_j}$$

For **spam** class:

- $\mathbb{P}(\text{"free"}|\text{spam}) = \frac{(5+1)}{(8)} = \frac{6}{8} = 0.75$
- $\mathbb{P}(\text{"money"}|\text{spam}) = \frac{(3+1)}{(8)} = \frac{4}{8} = 0.5$
- $\mathbb{P}(\text{"meeting"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$
- $\mathbb{P}(\text{"lunch"}|\text{spam}) = \frac{(0+1)}{(8)} = \frac{1}{8} = 0.125$

For **not spam** class:

- $\mathbb{P}(\text{"free"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"money"}|\text{not spam}) = \frac{(0+1)}{(6)} = \frac{1}{6} = 0.166$
- $\mathbb{P}(\text{"meeting"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$
- $\mathbb{P}(\text{"lunch"}|\text{not spam}) = \frac{(3+1)}{(6)} = \frac{4}{6} = 0.66$

Predicted Class = spam

New Email: x = "free money"

$$\mathbb{P}(spam \mid x) = \mathbb{P}(Y = spam) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \mid Y = spam)$$

$$\mathbb{P}(spam \mid x) = 0.5 \cdot (0.75)^1 \cdot (0.5)^1 = \boxed{0.1875}$$

18%

Arg max

$$\mathbb{P}(not\ spam \mid x) = \mathbb{P}(Y = not\ spam) \cdot \Pi_{j=1}^{d} \mathbb{P}(x_j \mid Y = not\ spam)$$

$$\mathbb{P}(not\ spam \mid x) = 0.5 \cdot (0.166)^1 \cdot (0.166)^1 = \boxed{0.013}$$

≈ 1%

Naive Bayes Classifier

18% ↑

2%

Why does it work?

- Despite violating independence assumption, Naive Bayes often performs surprisingly well.
- Why?
    - Classification only needs **correct ranking**: We don't need accurate probabilities - just need $\mathbb{P}(spam \,|\, X) > \mathbb{P}(not\ \ spam \,|\, X)$ when the email is actually spam.
    - The independence assumption can distort probabilities while preserving the ranking.
    - High bias, low variance tradeoff: The strong assumption **reduces model complexity**, preventing overfitting **especially with limited data**.
    - Conditional independence may approximately hold: Within a class, features are sometimes less correlated than across classes.

Naive Bayes Classifier

18%

1%

Pros

- Extremely fast training (just counting)
- Fast prediction
- Handles high-dimensional data well — linear
- Works with **small training** sets
- Handles missing features naturally
- Easy to implement and interpret
- Often surprisingly accurate

Cons

- Independence assumption is usually wrong
- Probability estimates are unreliable
- Cannot learn feature interactions
- Continuous features require distributional assumptions
- Correlated features are “double-counted”

Classifiers

Comparison	Naive Bayes	Logistic Regression	LDA
Type	Generative	Decision Trees Discriminative	Generative
Assumption	Conditional Independence Between Features	Conditional Independence Between Rows of Data	Gaussian and shared covariance
Training	Closed Form / Counting	Gradient Descent	Closed Form
Data	Better with small data	Better with large data else risk overfitting	Works well across data sizes
Probabilities	Poorly calibrated, care more about correct ranking	Well calibrated	Well calibrated
Missing features	Handles naturally	Requires pre-processing	Requires pre-processing μ Σ

Today’s Outline

- Naive Bayes
- **Decision Trees**

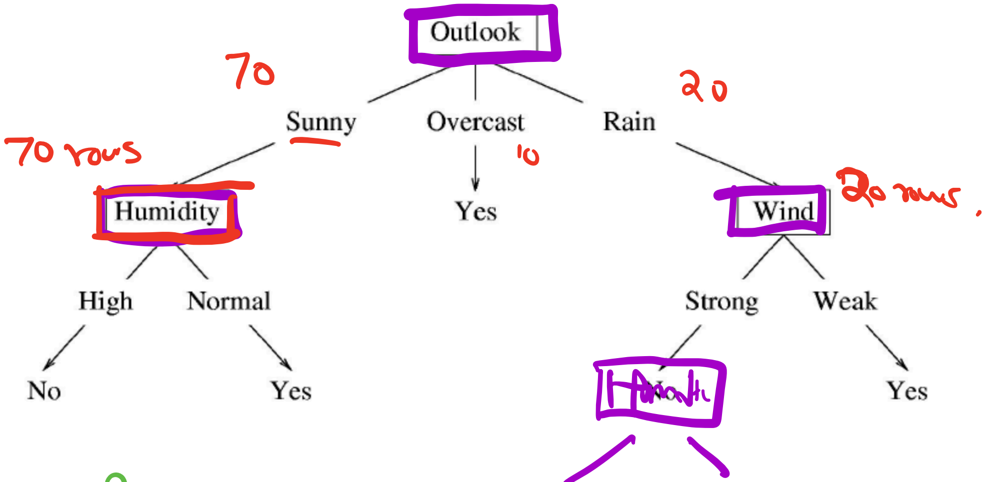
Decision Trees

Play Outside? (y)	Humidity (X1)	Wind (X2)	Outlook (X3)
Yes	Normal	Strong	Sunny
Yes	Low	Weak	Overcast
No	High	Weak	Rain

Decision Trees

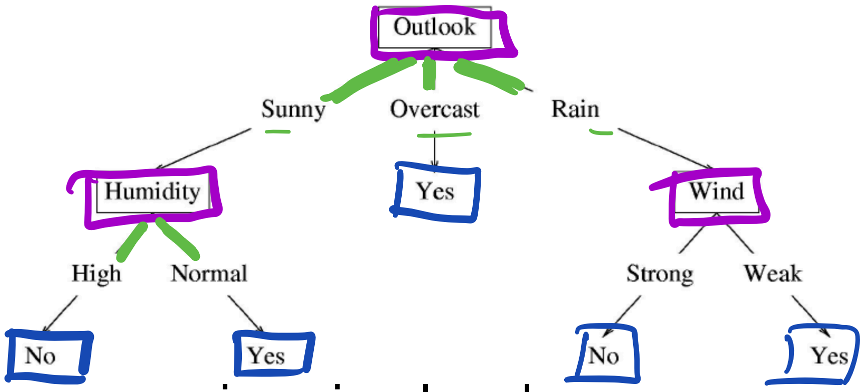
100 rows of data.

Play Outside? (y)	Humidity (X1)	Wind (X2)	Outlook (X3)
Yes	Normal	Strong	Sunny
Yes	Low	Weak	Overcast
No	High	Weak	Rain



Building → Training

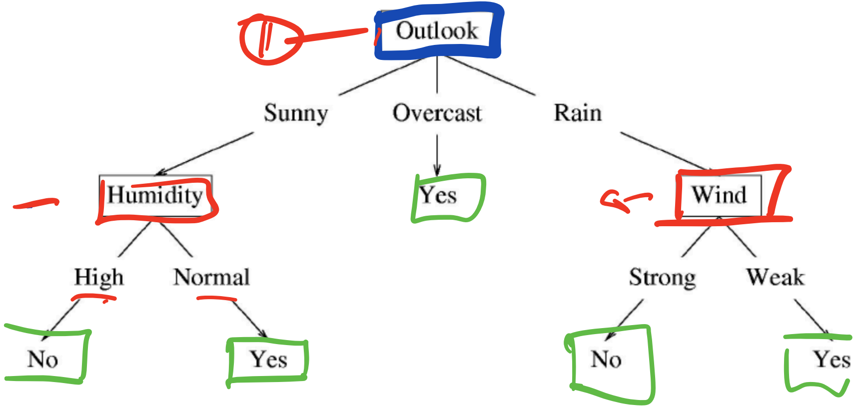
Decision Trees



- Decision trees recursively partition the feature space using simple rules, creating a tree structure that mirrors human decision-making.
- Each internal node is a test on a feature x_i
- Each branch is an outcome of the test (or selects a value for x_i)
- Each leaf is a class prediction Y

Decision Trees

2 hops



- **Root node**: Top node, contains entire dataset
- **Internal node**: Node with children, represents a split/test
- **Leaf node**: Terminal node with no children, makes predictions
- **Depth**: Distance from root to a node
- **Parent/Child**: Nodes directly connected vertically
- **Splitting criterion**: Rule for choosing which feature to split on

Decision Trees

- Learning "optimal", i.e., the simplest and smallest decision trees are an NP-complete problem.
- We resort to a greedy heuristic
  - Start from an empty tree
  - Of all available features x_i, split on the **best feature**
  - Recurse

entropy

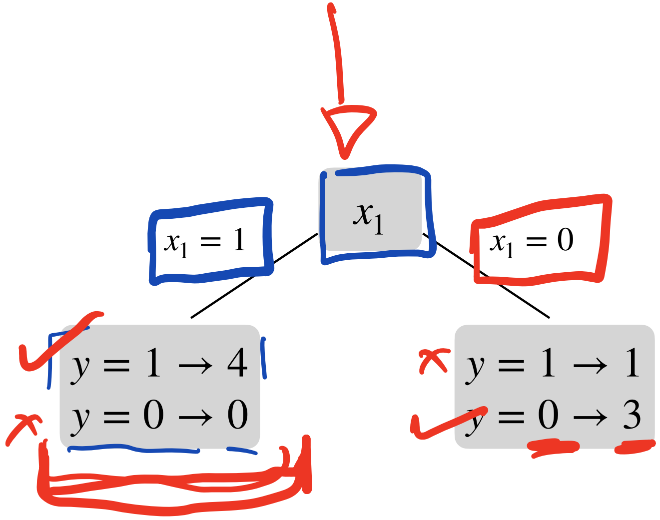
Decision Trees

? ?

y	x_1	x_2
1	1	1
1	1	0
1	1	1
1	1	0
1	0	1
0	0	0
0	0	1
0	0	0

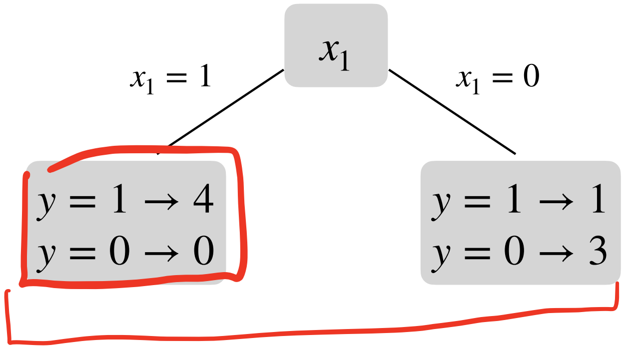
Decision Trees

y	x_1	x_2
1	1	1
1	1	0
1	1	1
1	1	0
1	0	1
0	0	0
0	0	1
0	0	0

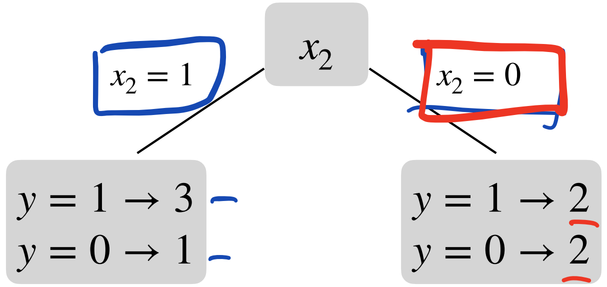


Decision Trees

y	x_1	x_2
1	1	1
1	1	0
1	1	1
1	1	0
1	0	1
0	0	0
0	0	1
0	0	0

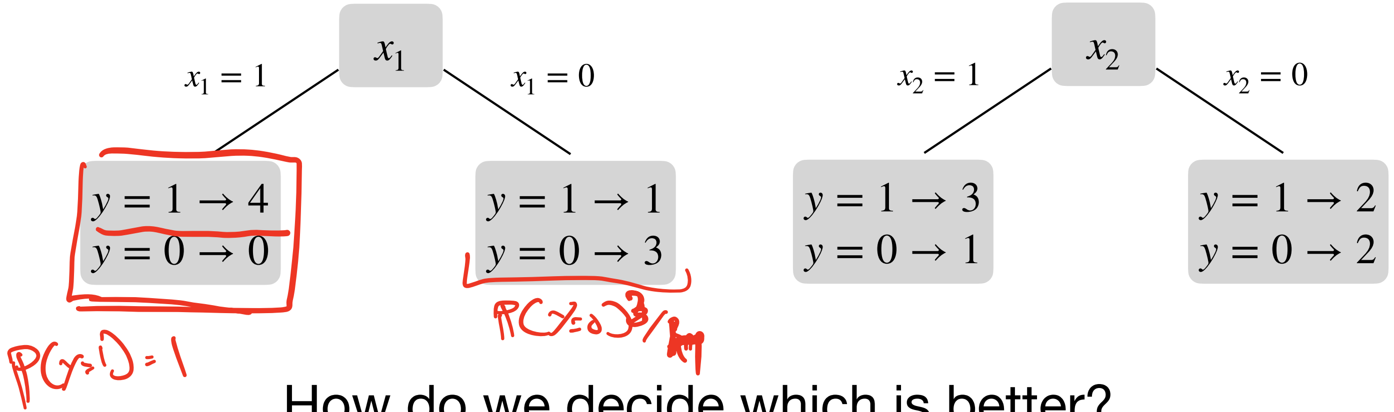


better



Decision Trees

y	x_1	x_2
1	1	1
1	1	0
1	1	1
1	1	0
1	0	1
0	0	0
0	0	1
0	0	0



How do we decide which is better?

Entropy Measures - Information Gain

Use counts at leaf nodes to define probabilities, so we can measure uncertainty

Entropy

- Entropy measures **uncertainty** or **surprise**.
- The **more uncertain** you are about an outcome, the **higher the entropy**.
- **Scenario 1**: A coin that always lands heads.

Entropy

- Entropy measures **uncertainty** or **surprise**.
- The **more uncertain** you are about an outcome, the **higher the entropy**.
- **Scenario 1**: A coin that always lands heads.
  - Before flipping, are you uncertain about the outcome?
  - No. You know it will be heads. There's no surprise, no uncertainty.
  - Entropy = 0 (minimum)

Entropy

- Entropy measures **uncertainty** or **surprise**.
- The **more uncertain** you are about an outcome, the **higher the entropy**.
- **Scenario 2**: A fair coin (50% heads, 50% tails). – highest possible entropy.

Entropy

- Entropy measures **uncertainty** or **surprise**.
- The **more uncertain** you are about an outcome, the **higher the entropy**.
- **Scenario 2**: A fair coin (50% heads, 50% tails).
  - Before flipping, are you uncertain?
  - Yes. You genuinely don't know what will happen. Maximum surprise possible for two outcomes.
  - Entropy = 1 bit (maximum for binary outcome)

Entropy

- Entropy measures **uncertainty** or **surprise**.
- The **more uncertain** you are about an outcome, the **higher the entropy**.
- **Scenario 3**: A biased coin (90% heads, 10% tails).
  - Some uncertainty, but not much. You'd bet on heads and usually be right.
  - Entropy = 0.47

Entropy

- Entropy measures **uncertainty** or **surprise**.
- The **more uncertain** you are about an outcome, the **higher the entropy**.
- **<u>Entropy is maximized when all outcomes are equally likely.</u>**
- For outcomes with probabilities $p_1, p_2, \ldots, p_n$:

n states → 6 states.

$p_1 \cdots p_6$

$$H = -\sum_i p_i log_2(p_i)$$

Entropy

Surprise(x) → Something.

- Why log formulation?
- For outcomes with probabilities $p_1, p_2, ..., p_n$:

$$H = -\sum_i p_i log_2(p_i)$$

<u>1. We want to measure how **surprising** some outcome is</u>

If $p_1 < p_2$, then surprise$(p_1) >$ surprise(p_2)

Entropy

- Why log formulation?
- For outcomes with probabilities $p_1, p_2, \ldots, p_n$:

$$H = -\sum_i p_i log_2(p_i)$$

1. We want to measure how **surprising** some outcome is

If $p_1 < p_2$, then surprise$(p_1) >$ surprise(p_2)

2. Certain events have 0 surprise, i.e., events guaranteed to happen

If $p_1 = 1$, surprise$(p_1) = 0$

Entropy

log satisfies all 3.

p1 < p2. $-\log p_1 > -\log p_2$. 1 ✓

2 $\log(1) = 0$ ✓

3 $\log(ab) = \log a + \log b$,

- Why log formulation?
- For outcomes with probabilities $p_1, p_2, \ldots, p_n$:

$$H = -\sum_i p_i log_2(p_i)$$

1) We want to measure how **surprising** some outcome is

If $p_1 < p_2$, then surprise$(p_1) >$ surprise(p_2)

2) Certain events have 0 surprise, i.e., events guaranteed to happen

If $p_1 = 1$, surprise$(p_1) = 0$

3. Surprise of independent events should add up

2· If p_1 and p_2 are independent, then surprise(p_1, p_2) = surprise(p_1) + surprise(p_1)

The log function hits all three conditions exactly

Entropy

all events.

$\sum_i$ prob . Surprise

- Why log formulation?

- For outcomes with probabilities $p_1, p_2, \ldots, p_n$:

$$\boxed{H = -\sum_i p_i log_2(p_i)}$$

<u>1. We want to measure how **surprising** some outcome is</u>

If $p_1 < p_2$, then surprise$(p_1) >$ surprise(p_2)

<u>2. Certain events have 0 surprise, i.e., events guaranteed to happen</u>

If $p_1 = 1$, surprise$(p_1) = 0$

<u>3. Surprise of independent events should add up</u>

If p_1 and p_2 are independent, then surprise$(p_1, p_2) =$ surprise$(p_1) +$ surprise(p_1)

Entropy

bit 1	bit 2	Outcome
0	0	1
0	1	2
1	0	3
1	1	4

- For outcomes with probabilities $p_1, p_2, \ldots, p_n$:

$$H = -\sum_i p_i log_2(p_i)$$

- Why "Bits" and log_2?
  - Entropy answers: "How many **yes/no** questions do I need to identify the outcome?"
  - Fair coin: 1 question ("Is it heads?") - 1 bit
  - Four equally likely outcomes: 2 questions - 2 bits
    - "Is it in the first half?"
    - "Is it the first of those two?"
- In general, n equally likely outcomes = $log_2(n)$ bits

Entropy

x_1: $\boxed{\text{Outcome}}$

Sunny Rainy Overcast

x_2 $\boxed{\text{Humidity}}$

- Why Entropy Matters in Decision Trees
  - We want splits that **reduce uncertainty** about the class label. y
  - A split that creates pure nodes (all one class) reduces entropy to zero.
    - **Before split:** Mixed classes, high entropy
    - **After good split**: Purer nodes, lower entropy
    - Information gain = entropy reduction

Entropy

$-\sum p_i \log p_i$

- Node 1 has $\underbrace{\text{75\% class A,}}$ $\underbrace{\text{25\% class B:}}$

$$H = -0.75log_2(0.75) - 0.25log_2(0.25) = \underbrace{0.811}\text{ bits}$$

Entropy

- Node 1 has 75% class A, 25% class B:

$$H = -0.75log_2(0.75) - 0.25log_2(0.25) = 0.811 \text{ bits}$$

- Node 2 has 50% class A and 50% class B:

$$H = -0.5log_2(0.5) - 0.5log_2(0.5) = \boxed{1 \text{ bit}}$$

- Node 1 has **lower entropy** (less uncertainty) than node 2.

Measuring Split Quality: Impurity Functions

Uncertainty

Entropy -

- A good split separates classes.
- We measure node **impurity** - how mixed the classes are - and choose splits that maximize impurity reduction.

Measuring Split Quality: Impurity Functions

Gini Impurity 1

- Let p_k be the proportion of class k samples in a $\boxed{\text{node}}$

$$\underbrace{Gini(D) = 1 - \sum_{k=1}^{K} \boxed{p_k}^2 = \sum_{k=1}^{K} p_k(1 - p_k)}$$

Measuring Split Quality: Impurity Functions

Gini Impurity

- Let p_k be the proportion of class k samples in a node

$$Gini(D) = 1 - \sum_{k=1}^{K} p_k^2 = \sum_{k=1}^{K} p_k(1 - p_k)$$

- **Interpretation**: Probability of misclassifying a randomly chosen sample if labeled according to the class distribution.

Measuring Split Quality: Impurity Functions

Gini Impurity

Humidity

High $\boxed{\text{low}}$ → do not play.

- Let p_k be the proportion of class k samples in a node

$$Gini(D) = 1 - \sum_{k=1}^{K} p_k^2 = \sum_{k=1}^{K} p_k(1 - p_k)$$

- **Interpretation**: Probability of misclassifying a randomly chosen sample if labeled according to the class distribution.

- **Properties**:

  - Minimum = 0 when node is pure (all one class)

  - Maximum = $1 - \dfrac{1}{K}$ when classes are equally distributed

  - For binary: max = 0.5 at p = 0.5

x_1

x_i: 1

$\boxed{y=1}$

pure.

Measuring Split Quality: Impurity Functions

Gini Impurity

- Let p_k be the proportion of class k samples in a node

$$Gini(D) = 1 - \sum_{k=1}^{K} p_k^2 = \sum_{k=1}^{K} p_k(1 - p_k)$$

- **Interpretation**: Probability of misclassifying a randomly chosen sample if labeled according to the class distribution.
- **Properties**:
  - Minimum = 0 when node is pure (all one class)
  - Maximum = $1 - \dfrac{1}{K}$ when classes are equally distributed
  - For binary: max = 0.5 at p = 0.5
- **Example (binary classification):**
  - Node with 100% class A: Gini = 1 - 1^2 = **0 (pure)**
  - Node with 50% each: Gini = 1 - 0.5^2 - 0.5^2 = 0.5 (maximum impurity)
  - Node with 90% class A: Gini = 1 - 0.9^2 - 0.1^2 = 0.18

Measuring Split Quality: Impurity Functions

Entropy

- Let p_k be the proportion of class k samples in a node

$$Entropy(D) = -\sum_{k=1}^{K} \underline{p_k} \underline{log_2(p_k)}$$

- **Interpretation:** Expected number of bits needed to encode the class of a random sample.

Measuring Split Quality: Impurity Functions

Entropy

- Let p_k be the proportion of class k samples in a node

$$Entropy(D) = -\sum_{k=1}^{K} p_k log_2(p_k)$$

- **Interpretation:** Expected number of bits needed to encode the class of a random sample.
- **Properties**:
    - Minimum = 0 when node is pure
    - Maximum = $log_2(K)$ when uniform distribution
    - For binary: max = 1 bit at p = 0.5

equal prob. for all classes,

Measuring Split Quality: Impurity Functions

Entropy

- Let p_k be the proportion of class k samples in a node

$$Entropy(D) = -\sum_{k=1}^{K} p_k log_2(p_k)$$

- **Interpretation:** Expected number of bits needed to encode the class of a random sample.
- **Properties**:
  - Minimum = 0 when node is pure
  - Maximum = $log_2(K)$ when uniform distribution
  - For binary: max = 1 bit at p = 0.5

- **Example (binary classification):**
  - Pure node: **Entropy = 0**
  - 50-50 split: Entropy = -0.5log_2(0.5) - 0.5log_2(0.5) = $\boxed{\text{1 bit}}$
  - 90-10 split: Entropy = -0.9 log_2(0.9) - 0.1 log_2(0.1) ≈ 0.47 bits

Measuring Split Quality: Impurity Functions

Information Gain

- We want splits that reduce **impurity** (i.e., Gini, Entropy).
- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{classes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$

Gini or Entropy.

Measuring Split Quality: Impurity Functions

Information Gain

- We want splits that reduce **impurity** (i.e., Gini, Entropy).
- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \underbrace{\sum_{k \in \text{classes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)}_{\text{weighted average impurity of child nodes}}$$

Measuring Split Quality: Impurity Functions

Information Gain

y = Play or don't play — predict.

x_0 — wind

x_1 — humidity

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$

100 Days
60 Play, 40 Don't Play

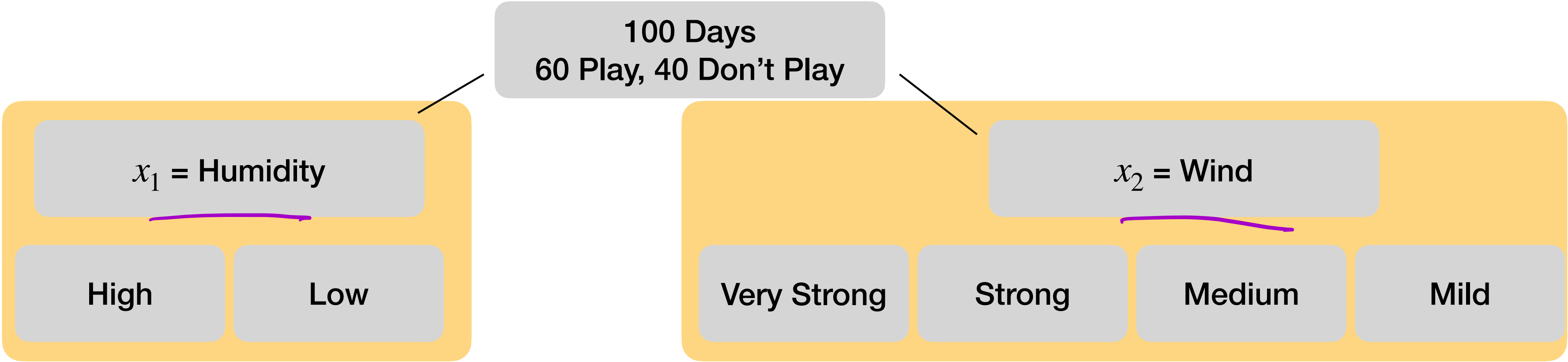
Measuring Split Quality: Impurity Functions

Information Gain

Step 1: Decide between Wind and Humidity

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$



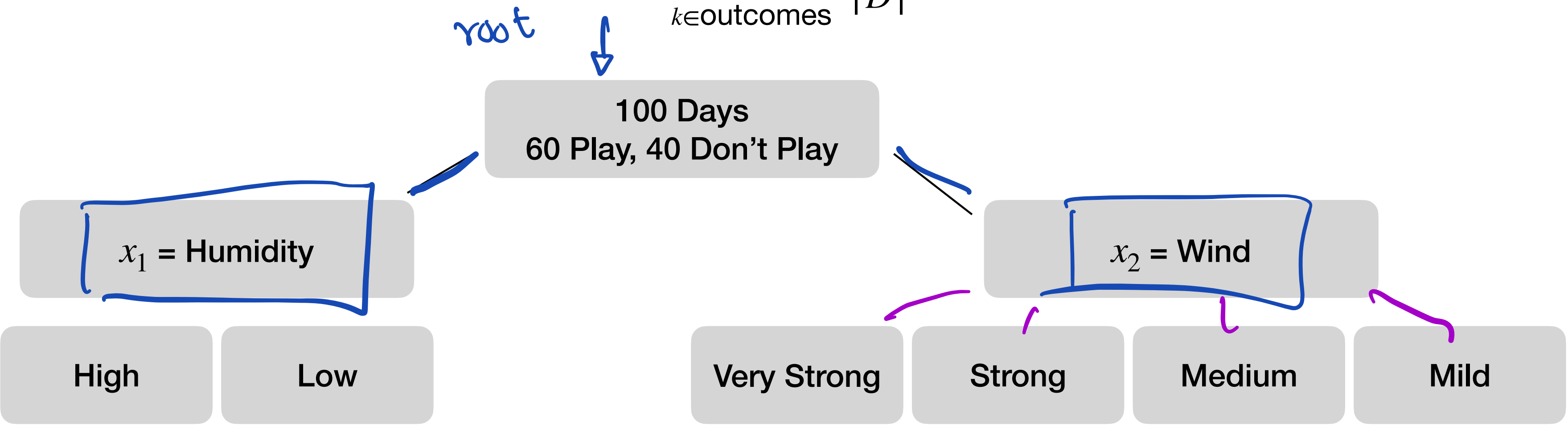
Measuring Split Quality: Impurity Functions

Information Gain

Step 2: Assuming we pick Wind, how to then split the data again?

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$



Split each into its own branch?

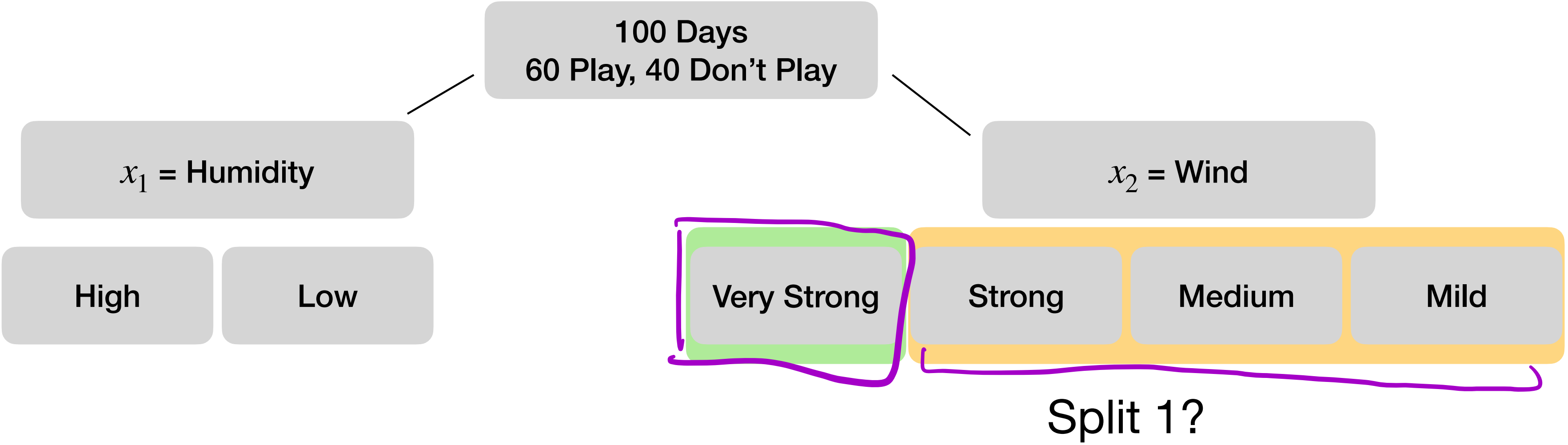
Measuring Split Quality: Impurity Functions

Information Gain

Step 2: Assuming we pick Wind, how to then split the data again?

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$



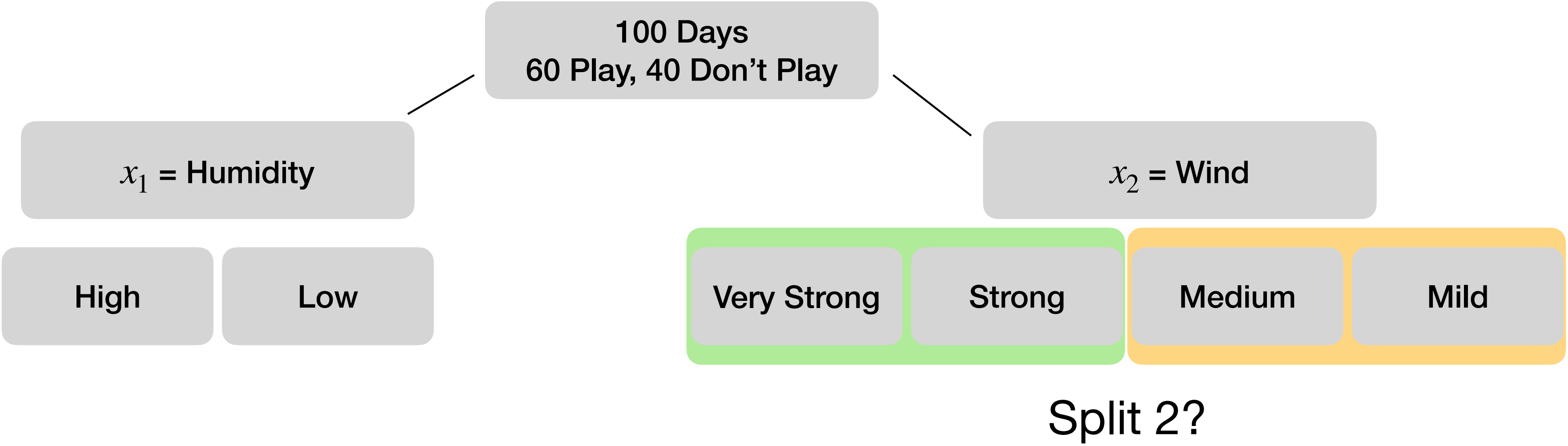
Measuring Split Quality: Impurity Functions

Information Gain

Step 2: Assuming we pick Wind, how to then split the data again?

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$



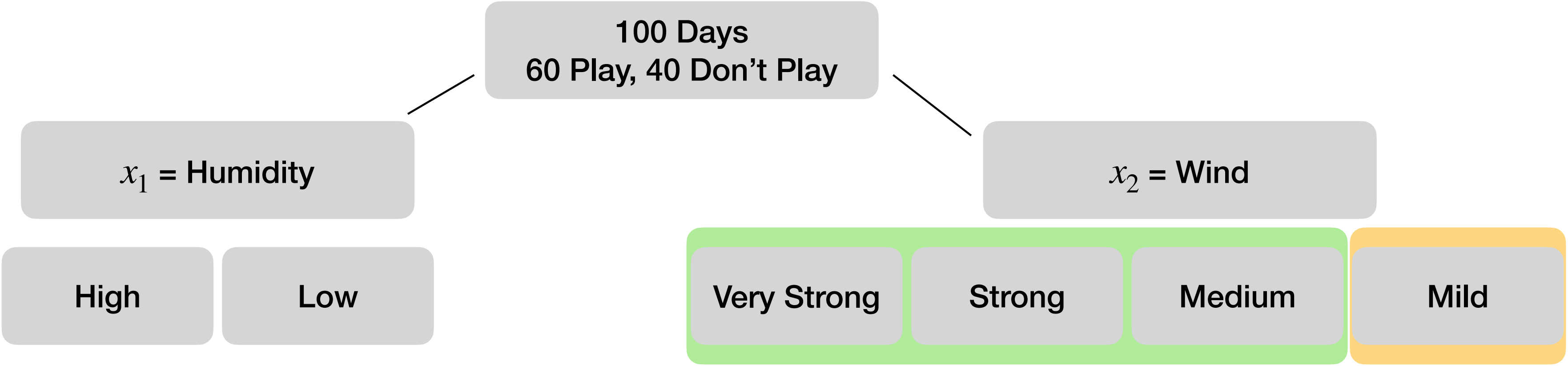
Measuring Split Quality: Impurity Functions

Information Gain

Step 2: Assuming we pick Wind, how to then split the data again?

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$



Split 3?

Measuring Split Quality: Impurity Functions

Information Gain

- Information gain measures this reduction:

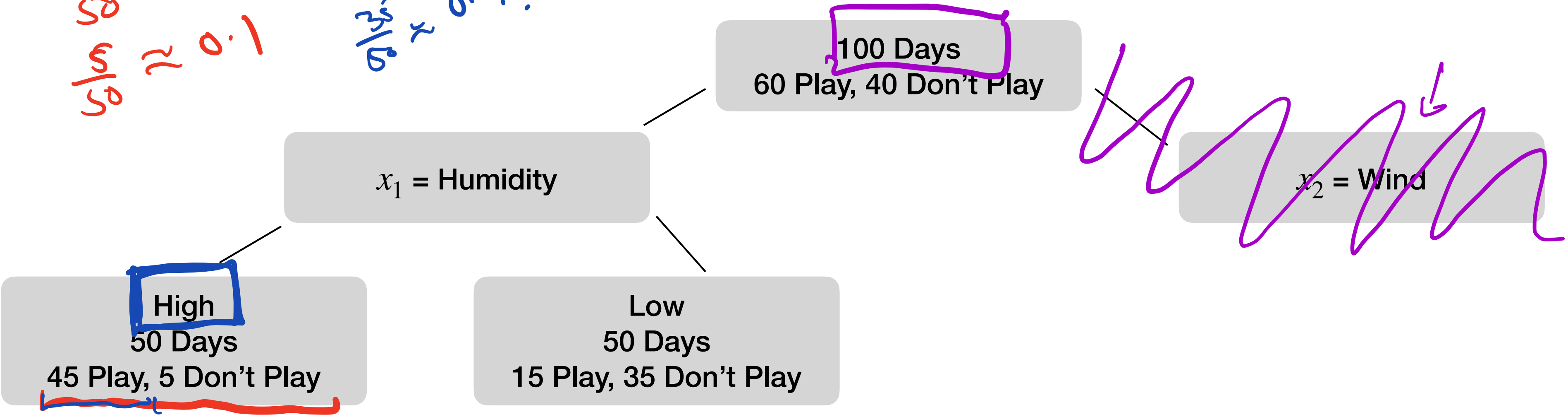
$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$

$\frac{45}{50} \approx 0.9$

$\frac{5}{50} \approx 0.1$

$\frac{15}{50} \approx 0.3$

$\frac{35}{50} \approx 0.7$



$H = -0.9log_2(0.9) - 0.1log_2(0.1) = 0.469\ bits$

$H = -0.3log_2(0.3) - 0.7log_2(0.7) = 0.881\ bits$

Measuring Split Quality: Impurity Functions

Information Gain

$|D_k| \leftarrow$ size of this set

- Information gain measures this reduction:

$$Gain(D, split) = \boxed{Impurity(D)} - \boxed{\sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)}$$

100 Days
60 Play, 40 Don't Play

x_1 = Humidity

x_2 = Wind

0.5

High
50 Days
45 Play, 5 Don't Play

Low
50 Days
15 Play, 35 Don't Play

Weighted Child Entropy =
$\boxed{0.5 \cdot 0.469} + \boxed{0.5 \cdot 0.881} = \boxed{0.675 \ bits}$

$H = -0.9 log_2(0.9) - 0.1 log_2(0.1) = \boxed{0.469 \ bits}$

$H = -0.3 log_2(0.3) - 0.7 log_2(0.7) = \boxed{0.881 \ bits}$

Measuring Split Quality: Impurity Functions

Information Gain

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$

100 Days
60 Play, 40 Don't Play

Impurity of Root Node =
$-0.6log_2(0.6) - 0.4log_2(0.4) = \boxed{0.971\ bits}$

x_1 = Humidity

x_2 = Wind

High
50 Days
45 Play, 5 Don't Play

Low
50 Days
15 Play, 35 Don't Play

Weighted Child Entropy =
$0.5 \cdot 0.469 + 0.5 \cdot 0.881 = 0.675\ bits$

$H = -0.9log_2(0.9) - 0.1log_2(0.1) = 0.469\ bits$

$H = -0.3log_2(0.3) - 0.7log_2(0.7) = 0.881\ bits$

Measuring Split Quality: Impurity Functions

Information Gain

Information Gain = 0.971 - 0.675 = **0.296 bits**

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$

100 Days
60 Play, 40 Don't Play

Impurity of Root Node =
$-0.6log_2(0.6) - 0.4log_2(0.4) = 0.971 \ bits$

x_1 = Humidity

x_2 = Wind

High
50 Days
45 Play, 5 Don't Play

Low
50 Days
15 Play, 35 Don't Play

Weighted Child Entropy =
$0.5 \cdot 0.469 + 0.5 \cdot 0.881 = 0.675 \ bits$

$H = -0.9log_2(0.9) - 0.1log_2(0.1) = 0.469 \ bits$

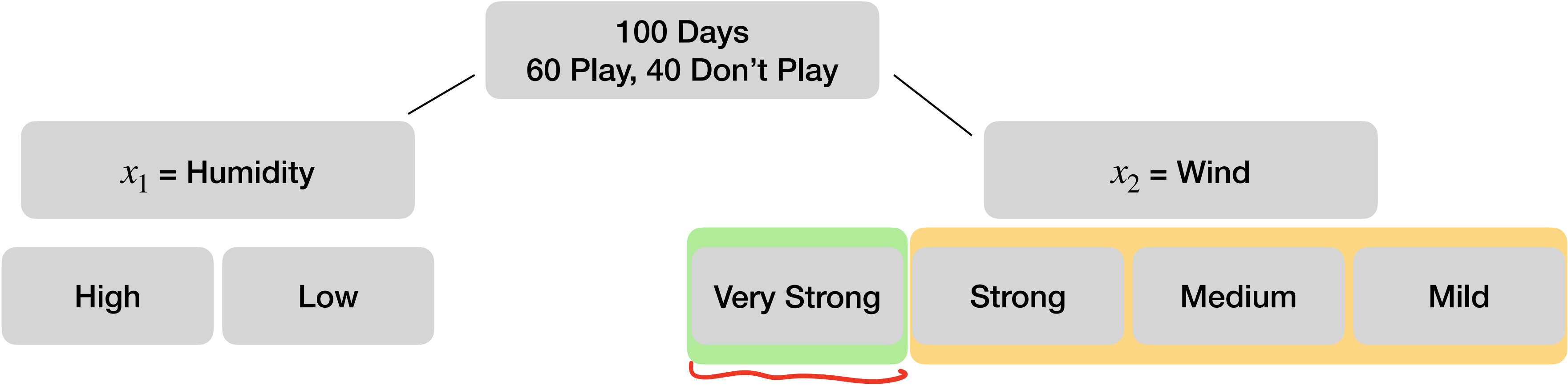
$H = -0.3log_2(0.3) - 0.7log_2(0.7) = 0.881 \ bits$

Measuring Split Quality: Impurity Functions

Information Gain

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$



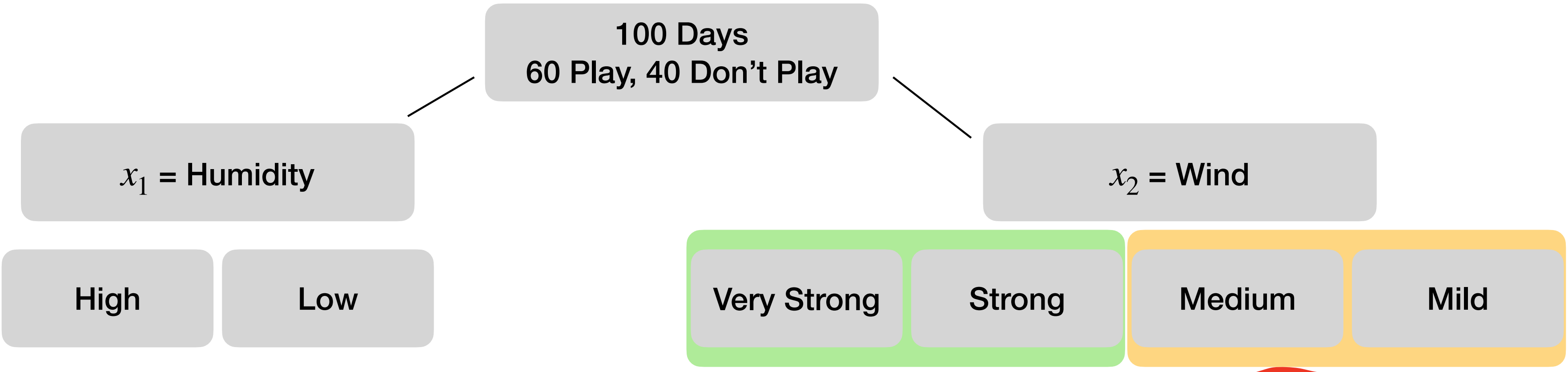
Split 1 - IG = 0.3

Measuring Split Quality: Impurity Functions

Information Gain

- Information gain measures this reduction:

$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$



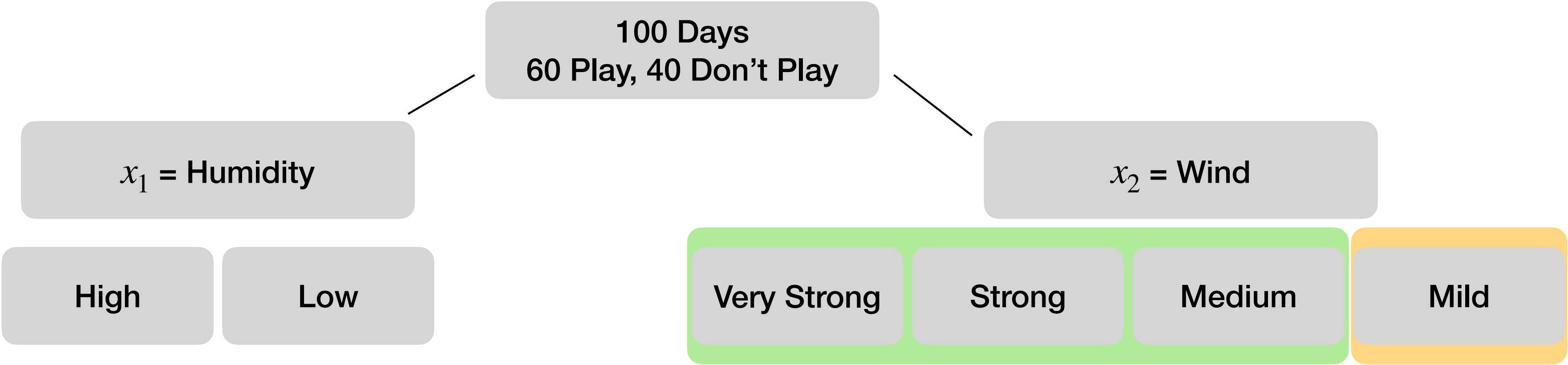
Split 2 - IG = 0.5

Measuring Split Quality: Impurity Functions

Information Gain

- Information gain measures this reduction:

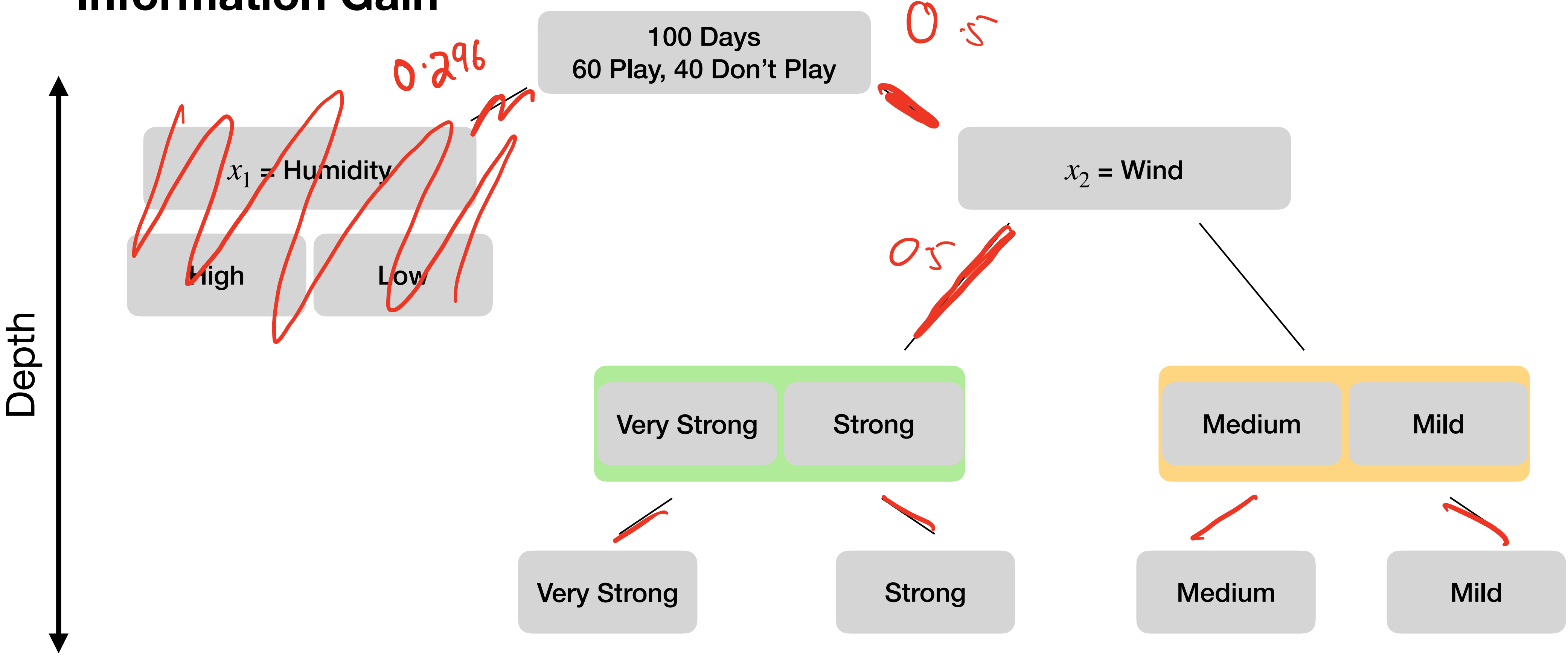
$$Gain(D, split) = Impurity(D) - \sum_{k \in \text{outcomes}} \frac{|D_k|}{|D|} \cdot Impurity(D_k)$$



Split 3 - IG = 0.1

Measuring Split Quality: Impurity Functions

Information Gain



Measuring Split Quality: Impurity Functions

Information Gain

14 Days
8 Play, 4 Don’t Play

Play	Humidity	Wind
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	High	Weak
1	High	Weak
1	High	Weak
1	High	Weak
0	High	Strong
0	High	Strong
0	High	Strong
0	High	Strong

Measuring Split Quality: Impurity Functions

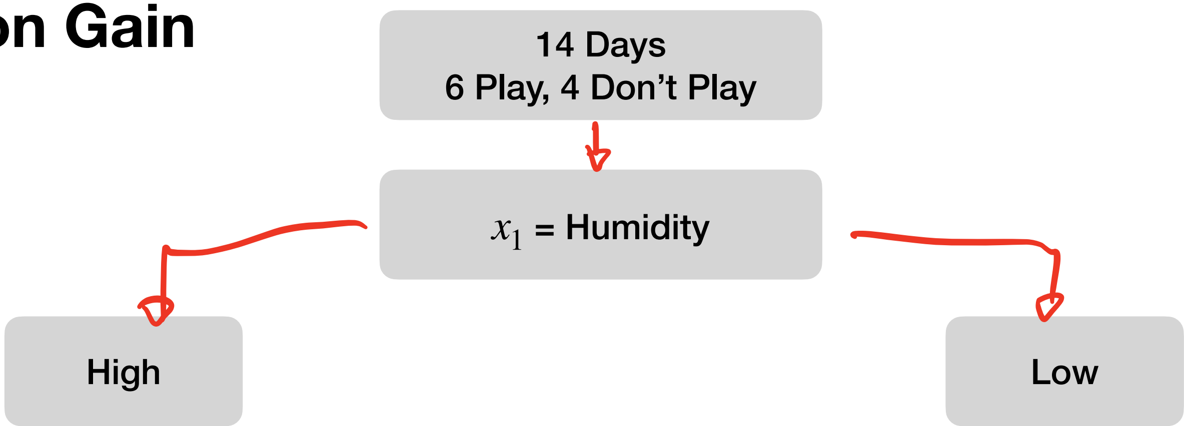
Information Gain

14 Days
6 Play, 4 Don’t Play

Play	Humidity	Wind
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	High	Weak
1	High	Weak
1	High	Weak
1	High	Weak
0	High	Strong
0	High	Strong
0	High	Strong
0	High	Strong

Measuring Split Quality: Impurity Functions

Information Gain



Play	Humidity	Wind
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	High	Weak
1	High	Weak
1	High	Weak
1	High	Weak
0	High	Strong
0	High	Strong
0	High	Strong
0	High	Strong

Measuring Split Quality: Impurity Functions

Information Gain

Play	Humidity	Wind
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	High	Weak
1	High	Weak
1	High	Weak
1	High	Weak
0	High	Strong
0	High	Strong
0	High	Strong
0	High	Strong

Measuring Split Quality: Impurity Functions

Information Gain

14 Days
6 Play, 4 Don’t Play

x_1 = Humidity

High

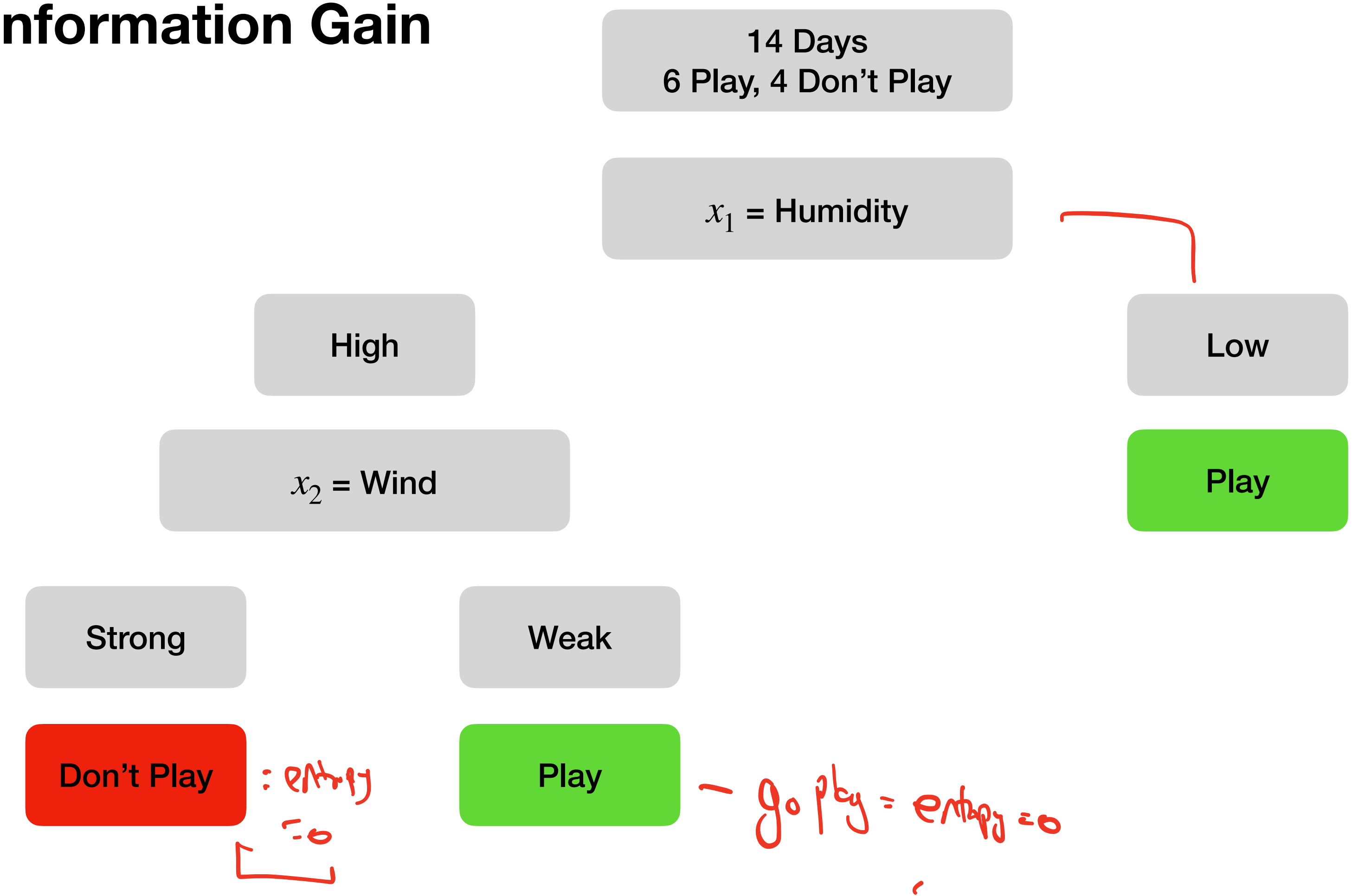
Low

Play

Play	Humidity	Wind
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	High	Weak
1	High	Weak
1	High	Weak
1	High	Weak
0	High	Strong
0	High	Strong
0	High	Strong
0	High	Strong

Measuring Split Quality: Impurity Functions

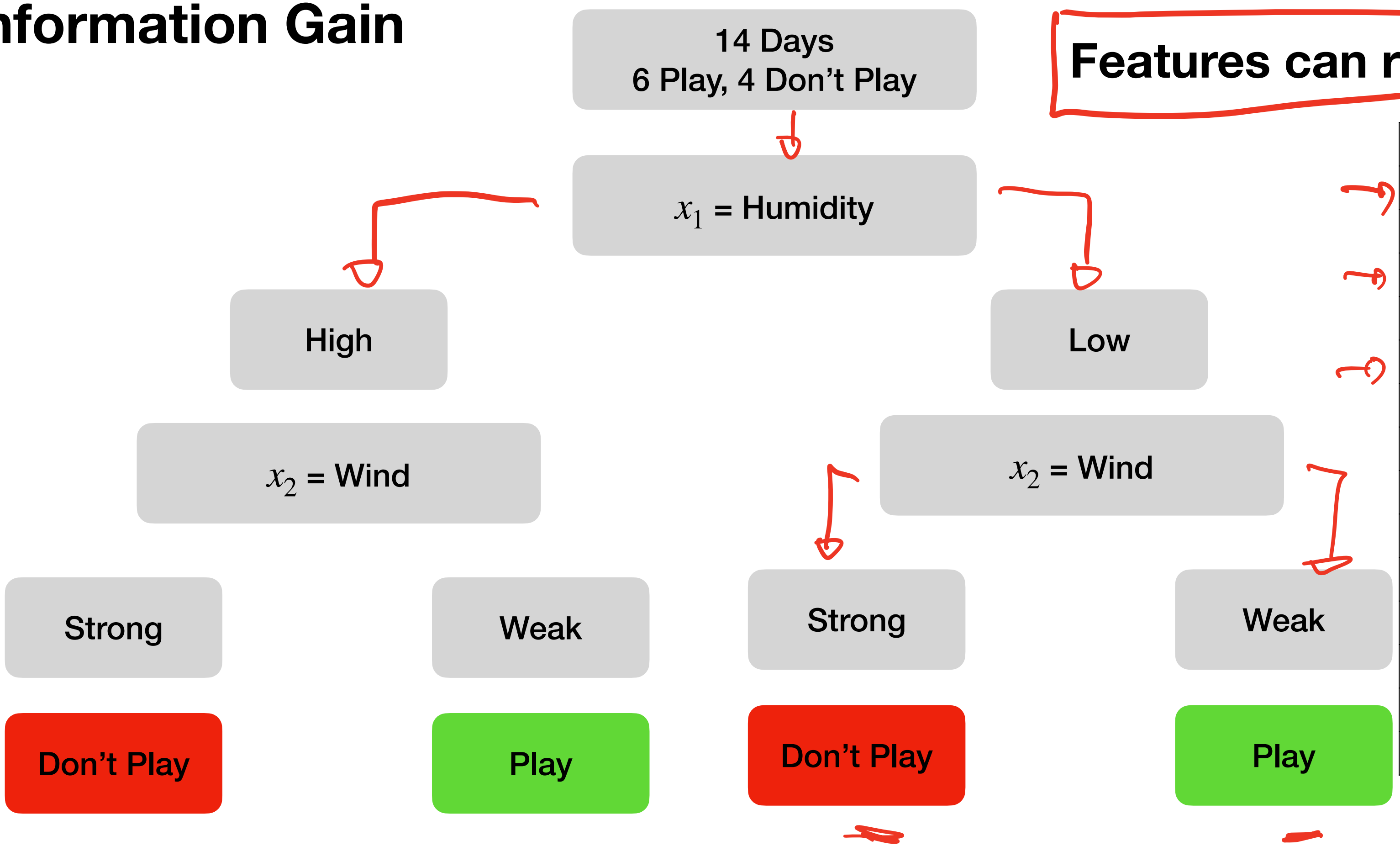
Information Gain



Play	Humidity	Wind
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	Low	Strong
1	Low	Weak
1	High	Weak
1	High	Weak
1	High	Weak
1	High	Weak
0	High	Strong
0	High	Strong
0	High	Strong
0	High	Strong

Measuring Split Quality: Impurity Functions

Information Gain



Features can repeat too!

Play	Humidity	Wind
0	Low	Strong
1	Low	Weak
0	Low	Strong
1	Low	Weak
0	Low	Strong
1	Low	Weak
1	High	Weak
1	High	Weak
1	High	Weak
1	High	Weak
0	High	Strong
0	High	Strong
0	High	Strong
0	High	Strong

Next Class

- Brief recap of decision trees
- Overfitting and underfitting in decision trees
- Regression Trees
- Bagging and Boosting