Decision Trees

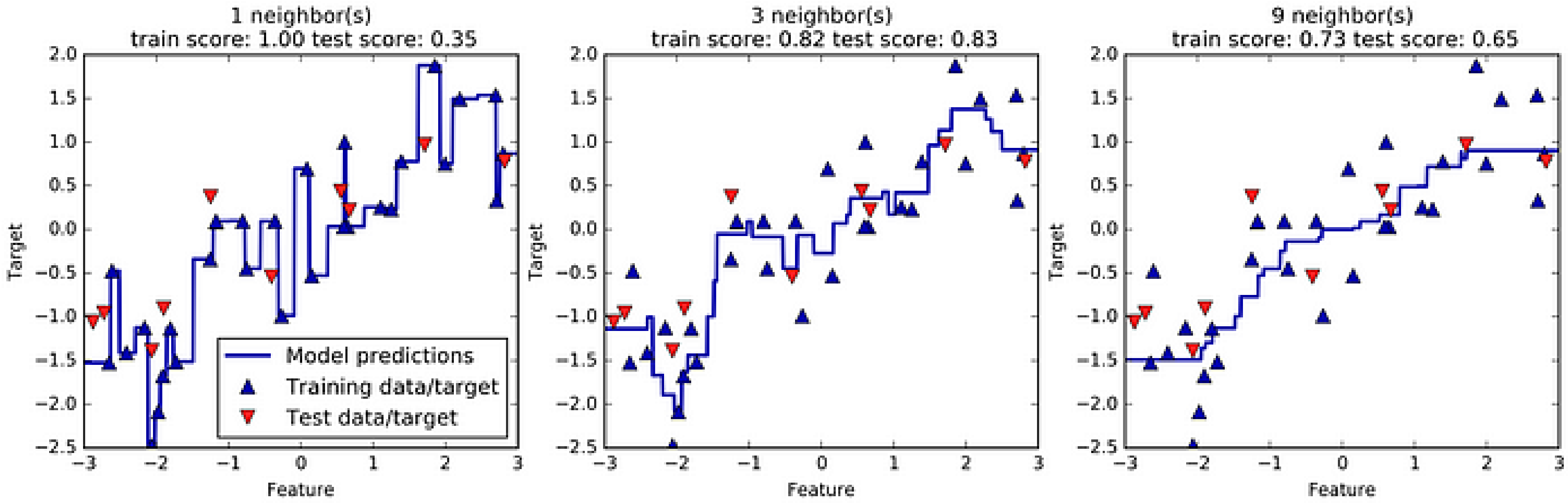
Adopted from slides by Alexander Ihler and

Jure Leskovec, Anand Rajaraman, Jeff Ullman, http://www.mmds.org

Supervised Learning

- **Given** examples of a function $(X, Y = F(X))$
- **Find** function $\hat{Y} = h(X)$ to estimate $F(X)$
  - Continuous $h(X)$: Regression
  - Discrete $h(X)$: Classification

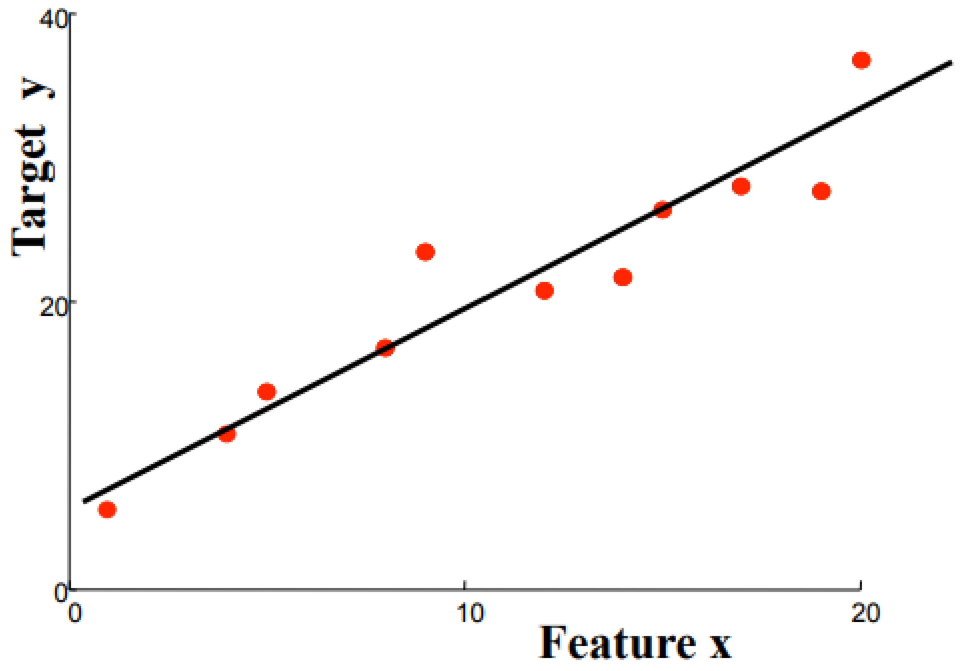
K-Nearest neighbor regression



ref : Andreas C.Muller and Sarah Guido. 2017. Introduction to machine learning with pyhton

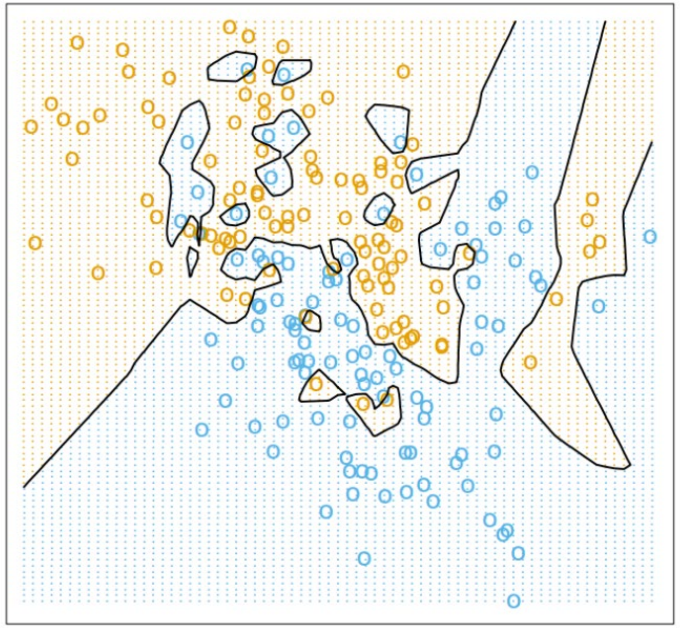
Linear regression

- $h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

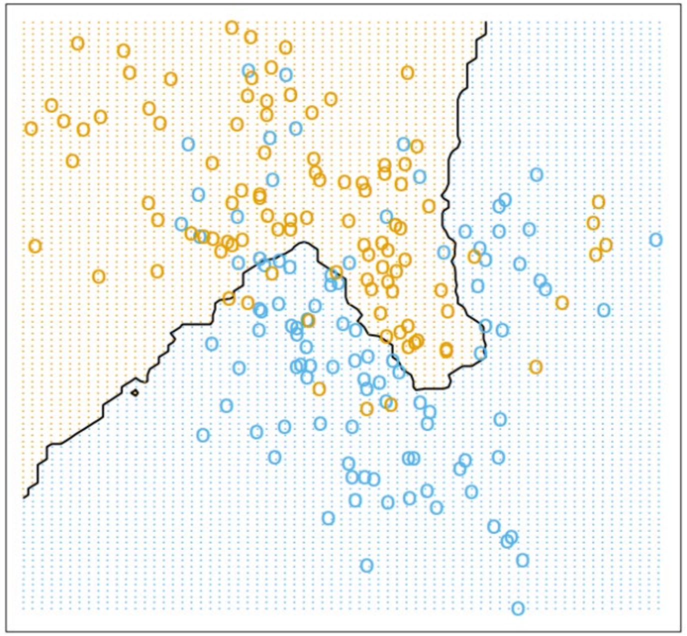


k-nearest neighbor classifier

$k = 1$



$k = 15$

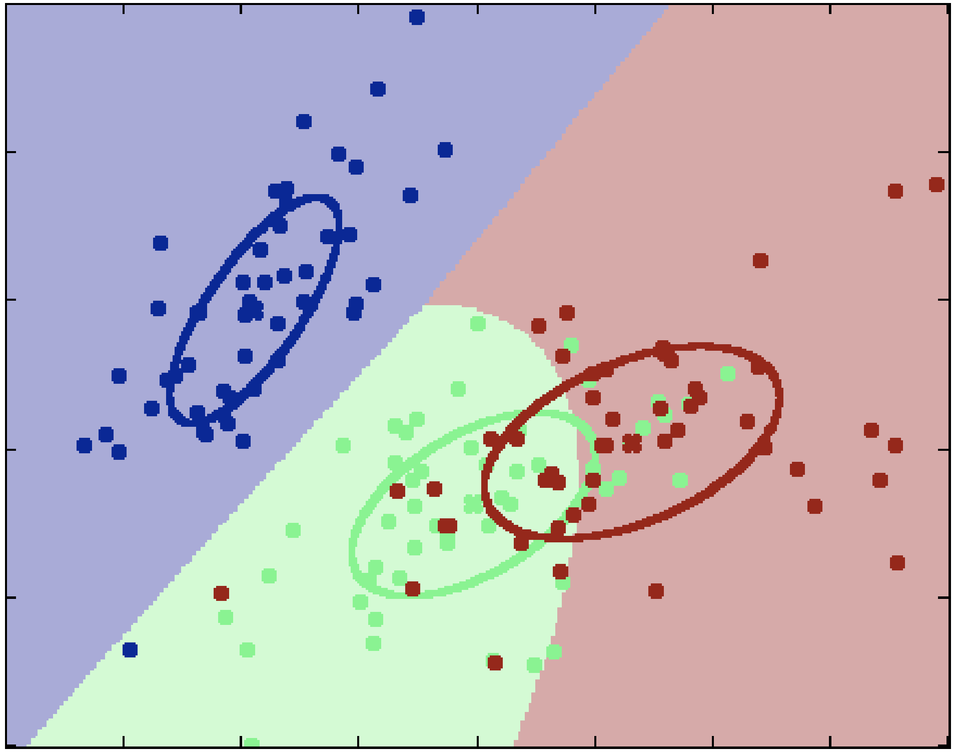


Example: Gaussian Bayes for Iris Data

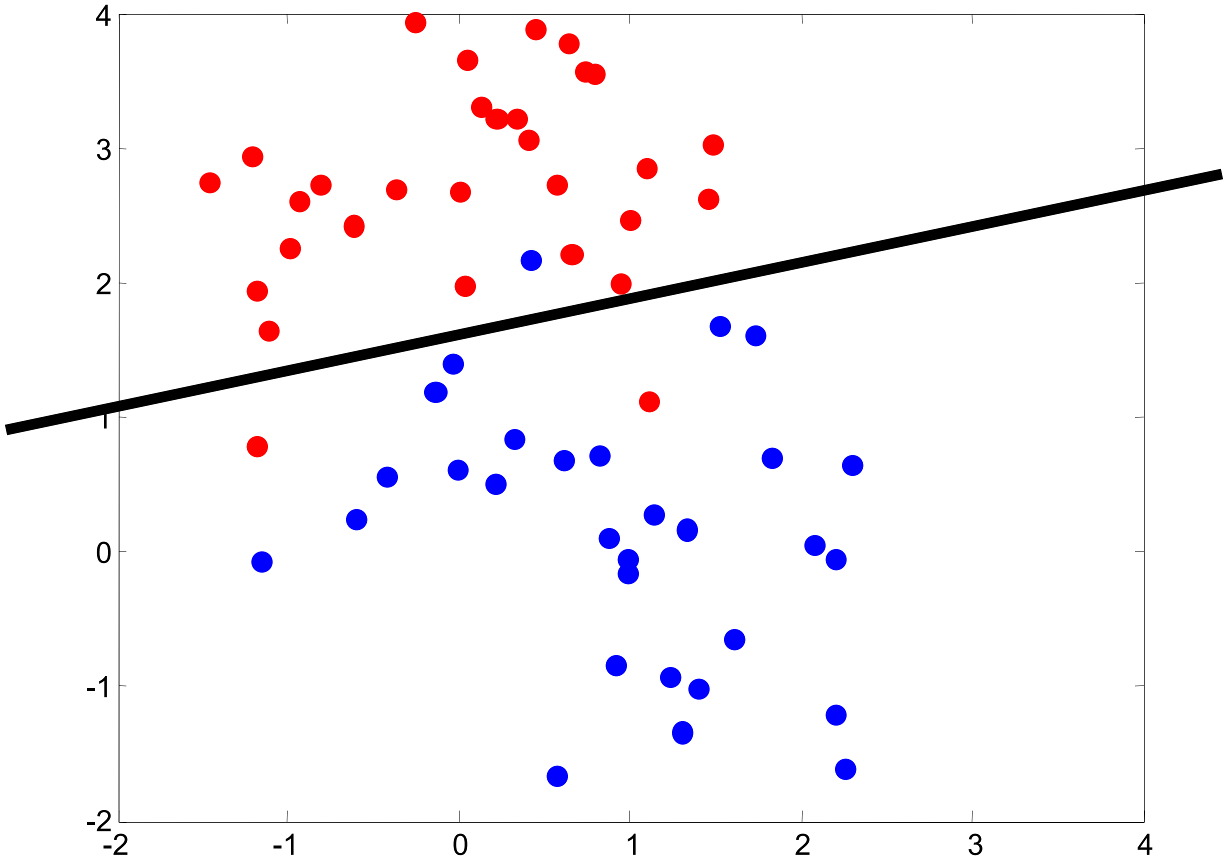
- Fit Gaussian distribution to each class {0,1,2}

$$p(y) = \text{Discrete}(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3})$$

$$p(x_1, x_2 | y = 0) = \mathcal{N}(x \,;\, \mu_0, \Sigma_0)$$
$$p(x_1, x_2 | y = 1) = \mathcal{N}(x \,;\, \mu_1, \Sigma_1)$$
$$p(x_1, x_2 | y = 2) = \mathcal{N}(x \,;\, \mu_2, \Sigma_2)$$

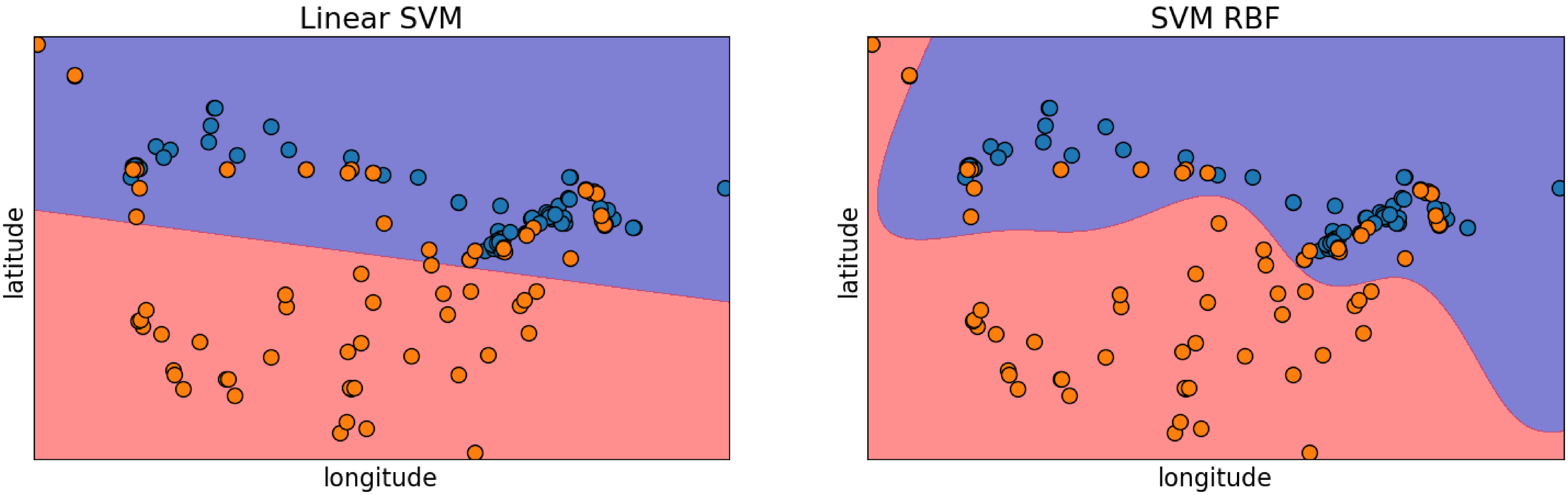


Linear Classifier



(c) Alexander Ihler

Support Vector Machine

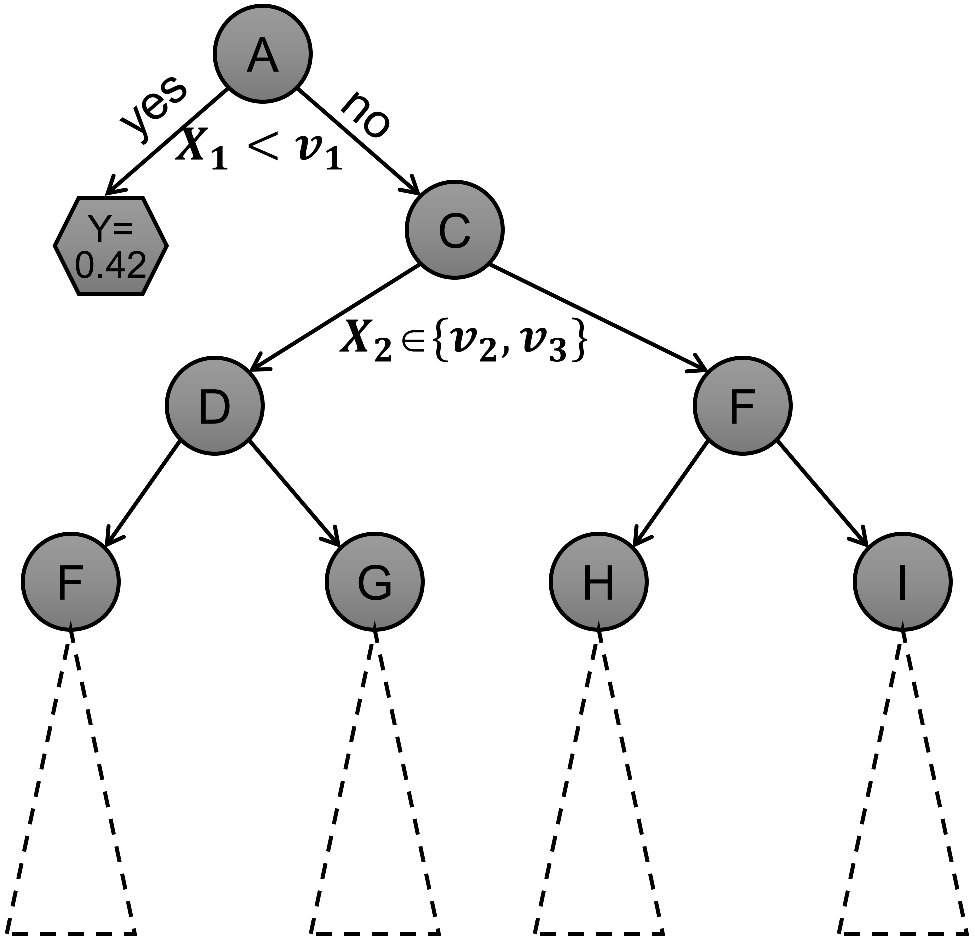


Decision Trees

- **Input attributes:**
  - $\boldsymbol{n}$ features/attributes: $x_1, x_2, \ldots\ x_n$
  - Each x_j has **domain** $\mathbf{O_j}$
    - **Categorical:** $\mathbf{O_j}$ = {red, blue}
    - **Numerical:** $\mathbf{H_j}$ = (0, 10)
  - $\boldsymbol{Y}$ is output variable with domain $\boldsymbol{O_Y}$:
    - **Categorical:** Classification, **Numerical:** Regression
- **Data D:**
  - $\boldsymbol{m}$ examples $(\boldsymbol{x}^{(i)}, y^{(i)})$ where $\boldsymbol{x}^{(i)}$ is the feature vector, $y^{(i)}$ is the output variable
- **Task:**
  - Given an input data vector $\boldsymbol{x}$ predict y

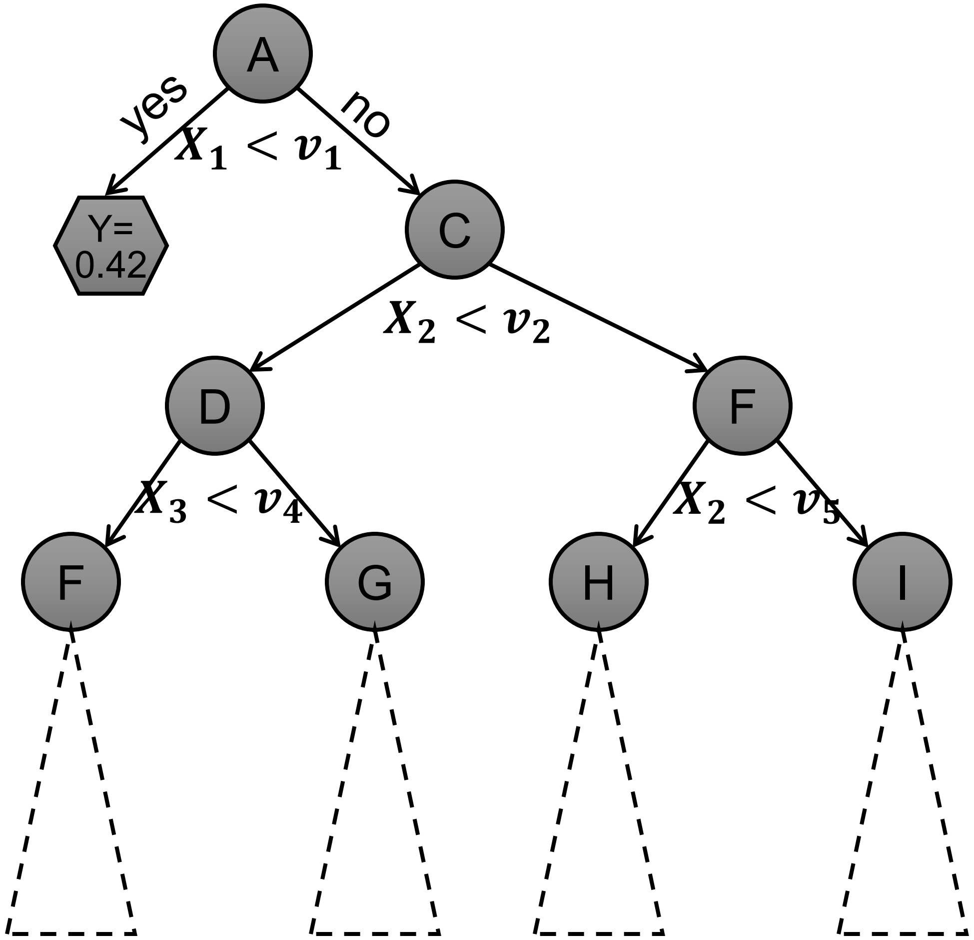
Decision Trees

- A **Decision Tree** is a tree-structured plan of a set of attributes to test in order to predict the output



Decision Trees (1)

- **Decision trees:**
  - Split the data at each internal node
  - Each leaf node makes a prediction
- **Today we focus on:**
  - Binary splits: $X_j < v$

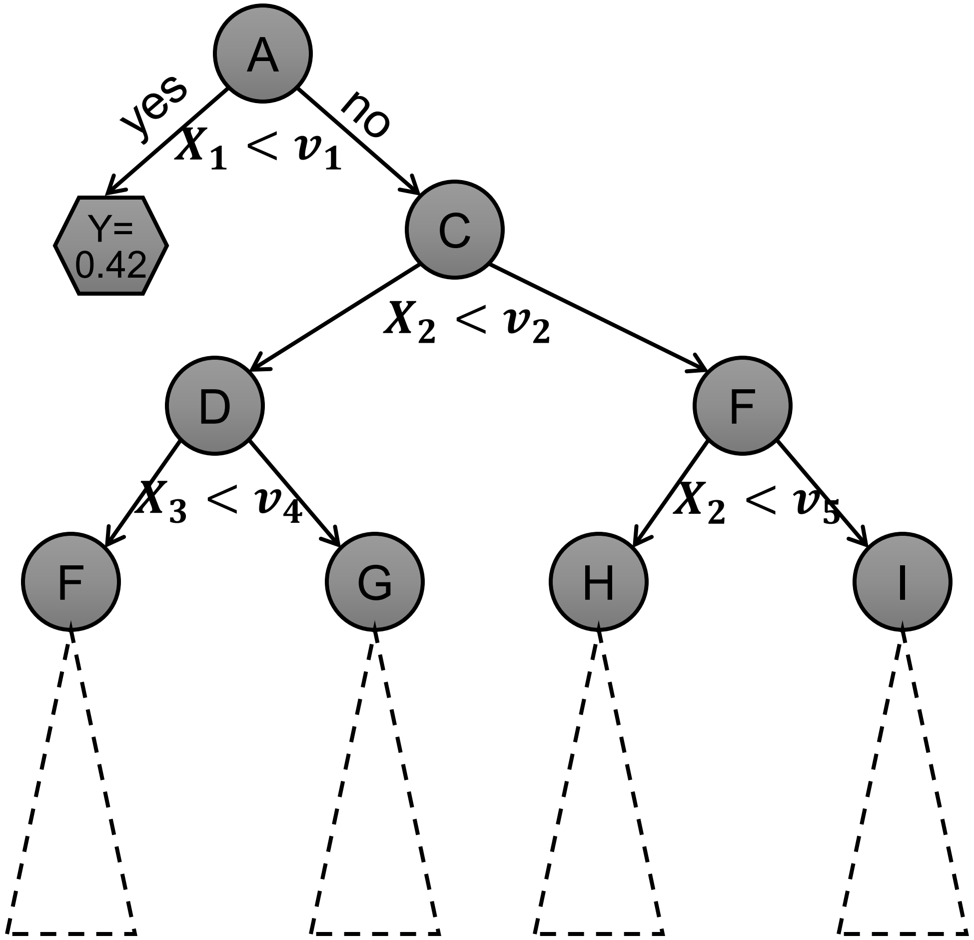


How to make predictions?

- **Input:** Example $\boldsymbol{x}$
- **Output:** Predicted $\hat{y}$

- "Drop" $\boldsymbol{x}$ down the tree until it hits a leaf node

- Predict the value stored in the leaf that $\boldsymbol{x}$ hits

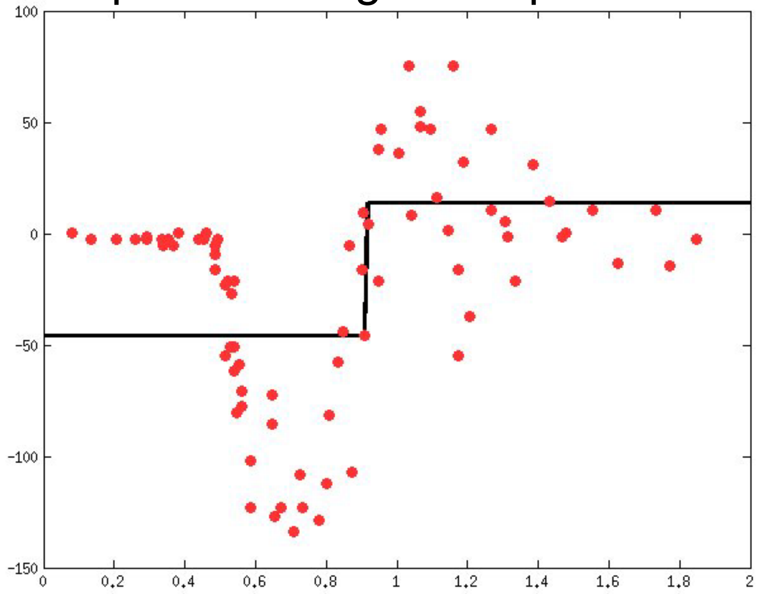


Decision trees for regression

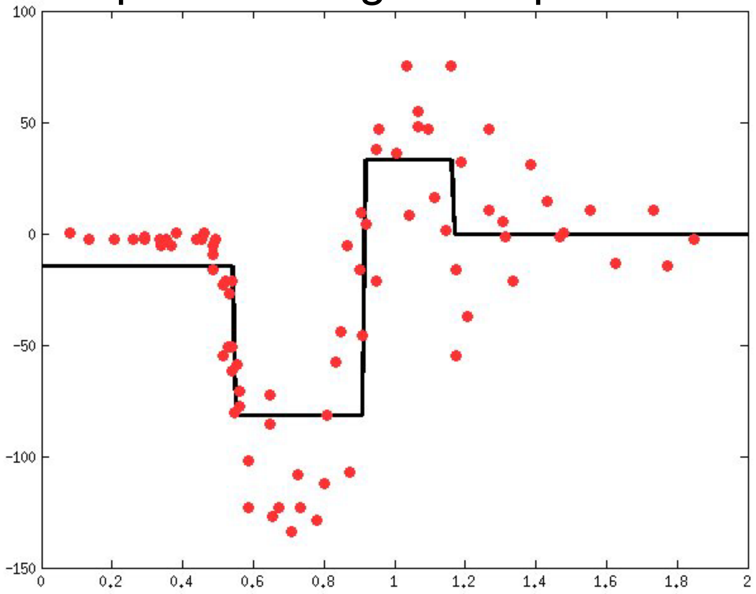
- Predict real valued numbers at leaf nodes

- Examples on a single scalar feature:

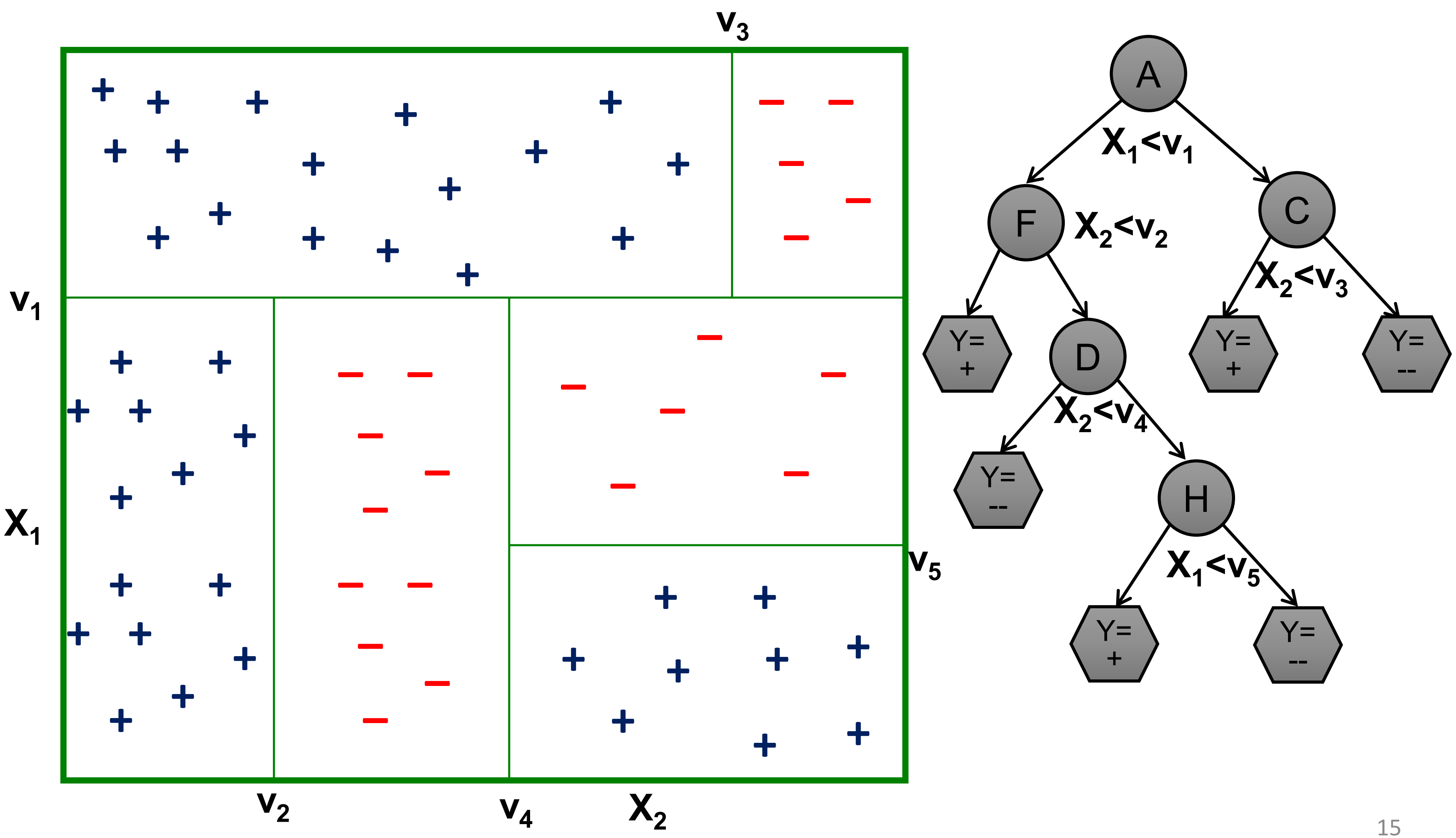
Depth 1 = 2 regions & predictions



Depth 2 = 4 regions & predictions …



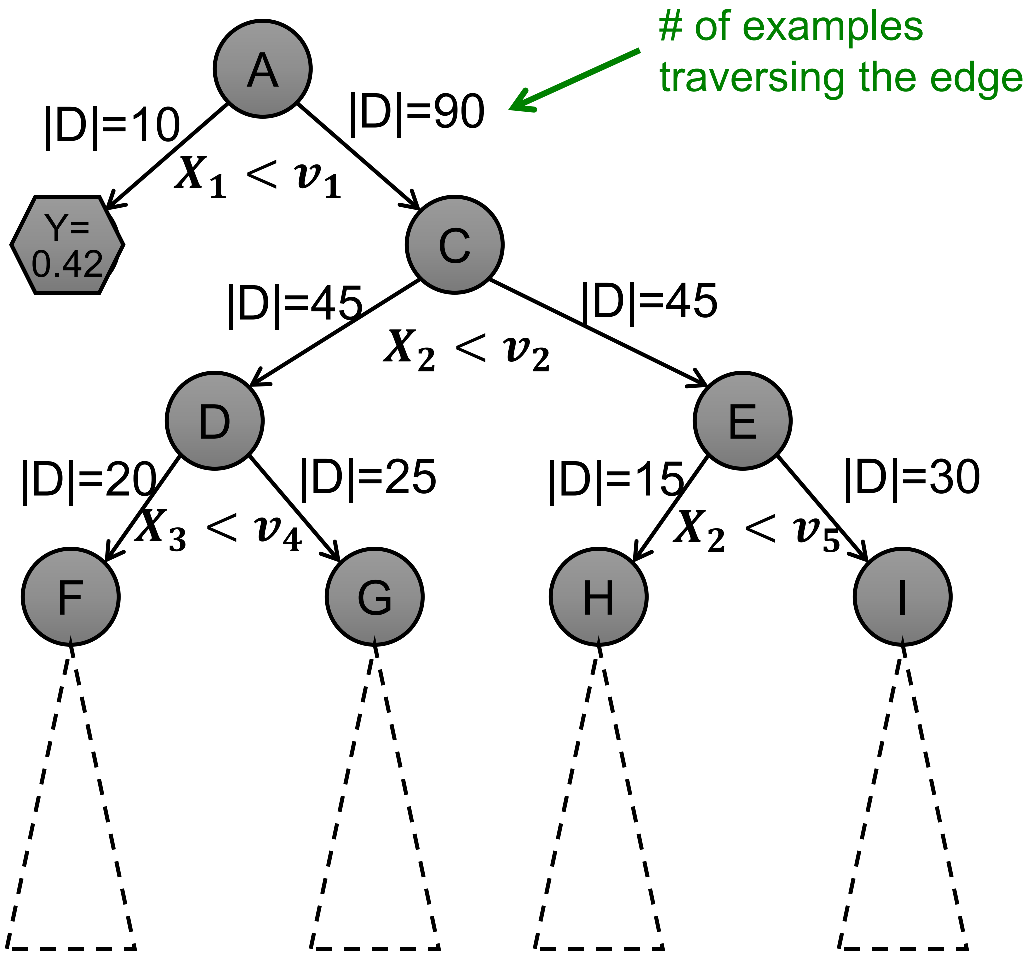
Decision Trees for classification



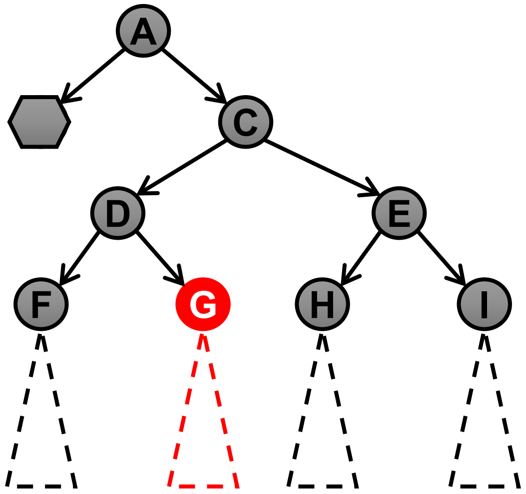
Learning Decision Tress

How to construct a tree?

- **Training dataset D*, |D*|=100 examples**

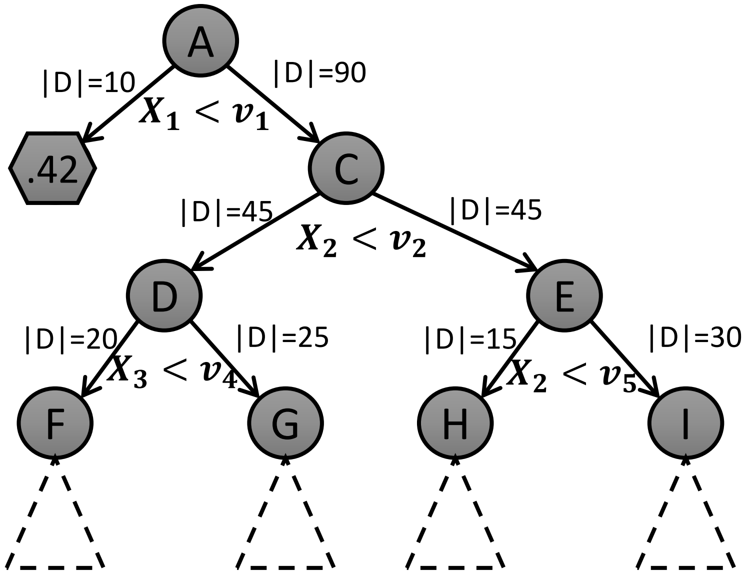


How to construct a tree?



- Imagine we are currently at some node ***G***
  - Let D_G be the data that reaches ***G***
- **There is a decision we have to make: Do we continue building the tree?**
  - **If yes**, which variable and which value do we use for a **split**?
    - Continue building the tree recursively
  - **If not**, how do we make a prediction?
    - We need to build a **"predictor node"**

How to construct a tree?



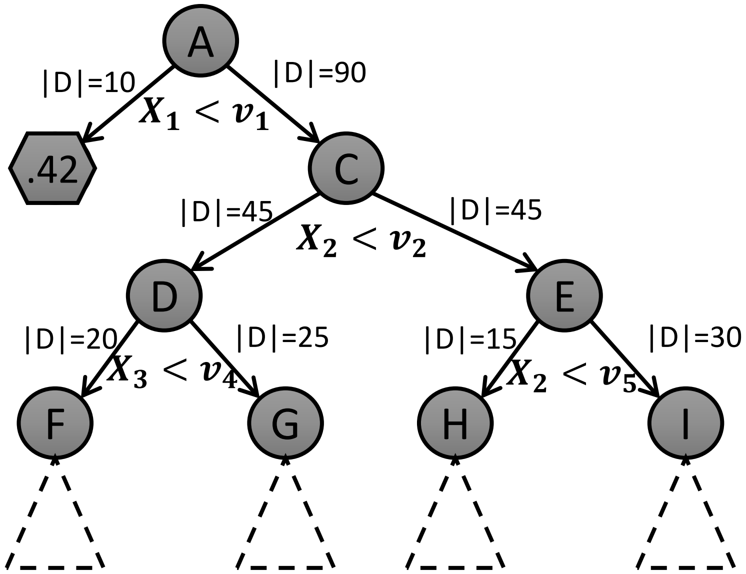
(1) How to split? Pick attribute & value that optimizes some criterion

- **Information Gain**
  - **Measures how much a given attribute X tells us about the class Y**
  - **IG(Y | X)** : We must transmit **Y** over a binary link. How many bits on average would it save us if both ends of the line knew **X**?

When to stop?

(2) When to stop?

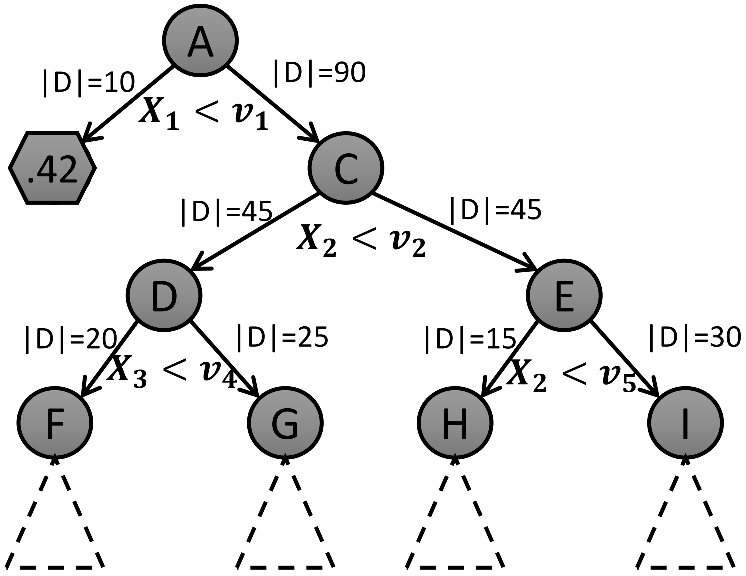
- Many different heuristic options
- **Two ideas:**
  - **(1) When the leaf is "pure"**
    - The target variable does not vary too much: ***Var(y) < ε***
  - **(2) When # of examples in the leaf is too small**
    - For example, ***|D|≤100***



How to predict?

(3) How to predict?

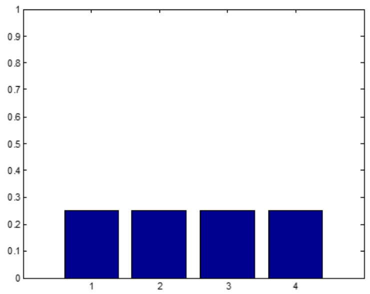
- **Many options**
  - **Regression:**
    - Predict average ***y*** of the examples in the leaf
    - Build a linear regression model on the examples in the leaf
  - **Classification:**
    - Predict most common ***y*** of the examples in the leaf



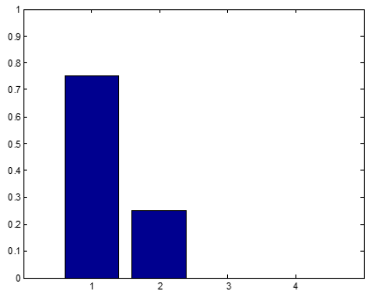
Information Gain?

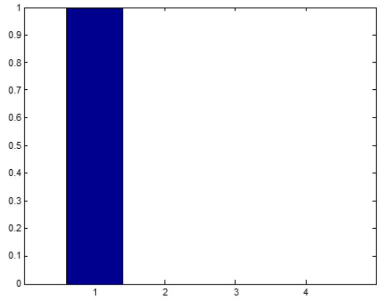
$$H(Y) = -\sum_y P(y)\, log\, P(y)$$

- **Entropy:**
  - What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from **X**'s distribution?
- **The entropy of Y:** $H(Y) = -\sum_{j=1}^{m} p_j\, log\, p_j$
  - **"High Entropy":** **Y** is from a uniform (boring) distribution
    - A histogram of the frequency distribution of values of **Y** is **flat**
  - **"Low Entropy":** **Y** is from a varied (peaks/valleys) distrib.
    - A histogram of the frequency distribution of values of **Y** would have many lows and one or two highs



Max entropy





Min entropy

Information Gain

- **Def: Information Gain**
  - $\boldsymbol{IG(Y|X)}$ = I must transmit **Y**. **How many bits on average would it save me if both ends of the line knew X?**

$$\boldsymbol{IG(Y|X) = H(Y) - H(Y \mid X)}$$

Example

$$H(Y) = -\sum_y P(y)\, log\, P(y)$$

- **Suppose I want to predict Y and I have input X**
  - $X1$ = College Major
  - $X2$ = …
  - Y = Likes "Gladiator"

X1	X2	Y
Math	…	Yes
History		No
CS		Yes
Math		No
Math		No
CS		Yes
Math		Yes
History		No

- **From this data we estimate**
  - $P(Y = Yes) = 0.5$
- **Note:**
  - $H(Y) = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2}) = 1$

$$H(Y|X=x) = -\sum_y P(y|x) \log P(y|x)$$

Example

- **Suppose I want to predict Y and I have input X**
  - $X1$ = College Major
  - $X2$ = …
  - Y = Likes "Gladiator"

X1	X2	Y
Math	…	Yes
History		No
CS		Yes
Math		No
Math		No
CS		Yes
Math		Yes
History		No

- **Def: Specific Conditional Entropy**
  - **H(Y | X1=v)** = The entropy of **Y** among only those records in which **X1** has value **v**
  - **Example:**
    - $H(Y|X1 = Math) = 1$
    - $H(Y|X1 = History) = 0$
    - $H(Y|X1 = CS) = 0$

$$H(Y|X) = -\sum_x P(x) \sum_y P(y|x) \log P(y|x)$$

Example

- **Suppose I want to predict Y and I have input X**
  - $X1$ = College Major
  - $X2$ = …
  - Y = Likes "Gladiator"

X1	X2	Y
Math	…	Yes
History		No
CS		Yes
Math		No
Math		No
CS		Yes
Math		Yes
History		No

- **Def: Conditional Entropy**
  - $H(Y \mid X)$ = The average specific conditional entropy of **Y**
    - = if you choose a record at random what will be the conditional entropy of **Y**, conditioned on that row's value of **X**
    - = Expected number of bits to transmit **Y** if both sides will know the value of **X**
  - $= \sum_j P(X = v_j) H(Y|X = v_j)$

Example

- **Suppose I want to predict $\boldsymbol{Y}$ and I have input $\boldsymbol{X}$**

X1	X2	Y
Math	…	Yes
History		No
CS		Yes
Math		No
Math		No
CS		Yes
Math		Yes
History		No

- $\boldsymbol{H(Y \mid X)}$ $=$ The average specific conditional entropy of $\boldsymbol{Y}$

$$= \sum_j P(X = v_j) H(Y | X = v_j)$$

- **Example:**

V_j	$P(X1=v_j)$	$H(Y\|X1=v_j)$
Math	0.5	1
History	0.25	0
CS	0.25	0

- **So:** H(Y|X1)=0.5*1+0.25*0+0.25*0 = **0.5**

Example

- Suppose I want to predict Y and I have input X

X1	X2	Y
Math	…	Yes
History		No
CS		Yes
Math		No
Math		No
CS		Yes
Math		Yes
History		No

- **Def: Information Gain**
  - $IG(Y|X)$ = I must transmit **Y**. **How many bits on average would it save me if both ends of the line knew X?**

$$IG(Y|X) = H(Y) - H(Y \mid X)$$

- **Example:**
  - H(Y) = 1
  - H(Y|X1) = 0.5
  - Thus **IG(Y|X1) = 1 – 0.5 = 0.5**

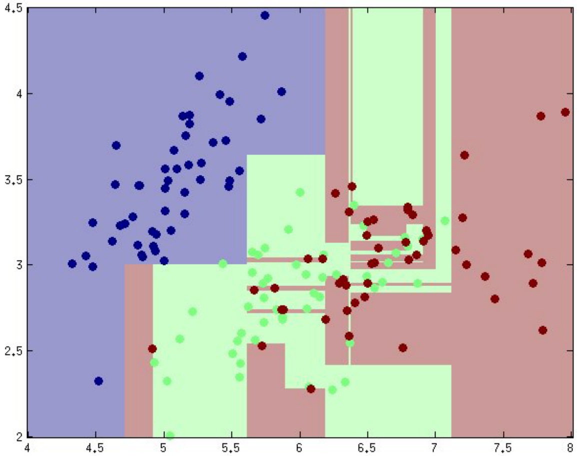
How to build decision tree

- Choose the feature and value that “decrease the entropy most = give us the largest information gain”
- Algorithms to build decision trees:
  - ID3, C4.5, …

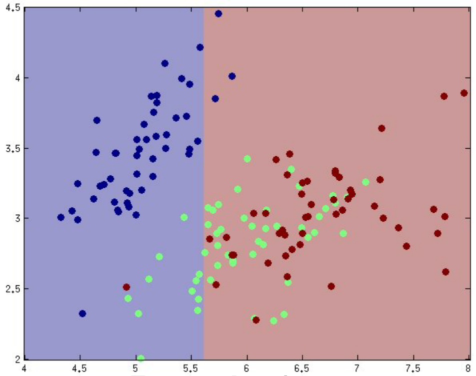
```
FUNCTION ID3(Examples, Target_Attribute, Attributes)
    CREATE a new node Root
    IF all Examples have the same Target_Attribute value THEN
        RETURN Root with label = that value
    END IF
    IF Attributes is empty THEN
        RETURN Root with label = most common Target_Attribute value in Examples
    END IF
    Best_Attribute = Attribute with highest Information_Gain(Examples, Target_Attribute, Attributes)
    Root.decision_attribute = Best_Attribute
    FOR each possible value v of Best_Attribute
        ADD a new branch below Root for value v
        Examples_v = {e ∈ Examples | e.Best_Attribute = v}
        IF Examples_v is empty THEN
            ADD leaf node with label = most common Target_Attribute value in Examples
        ELSE
            ADD subtree ID3(Examples_v, Target_Attribute, Attributes – {Best_Attribute})
        END IF
    END FOR
    RETURN Root
END FUNCTION
```

Controlling complexity

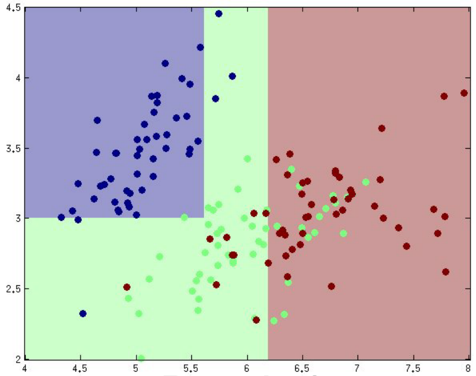
- Maximum depth cutoff



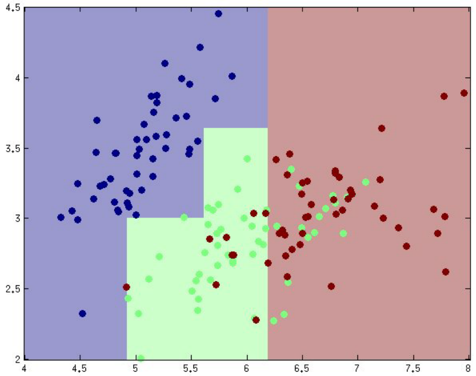
No limit



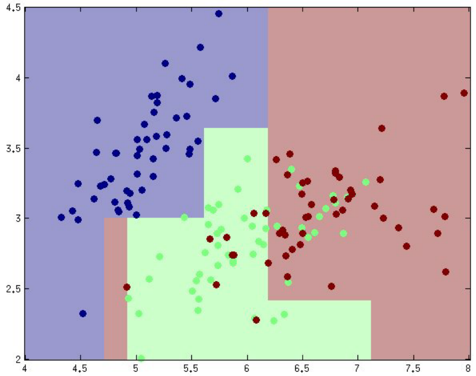
Depth 1



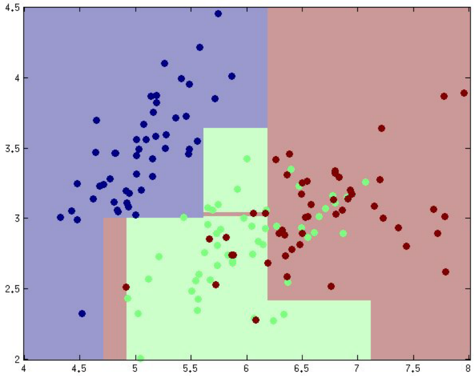
Depth 2



Depth 3



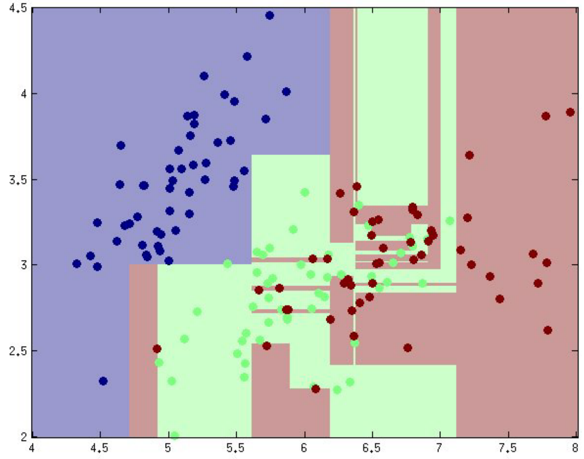
Depth 4



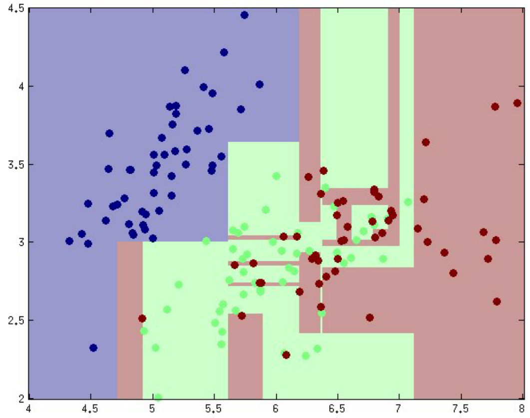
Depth 5

Controlling complexity

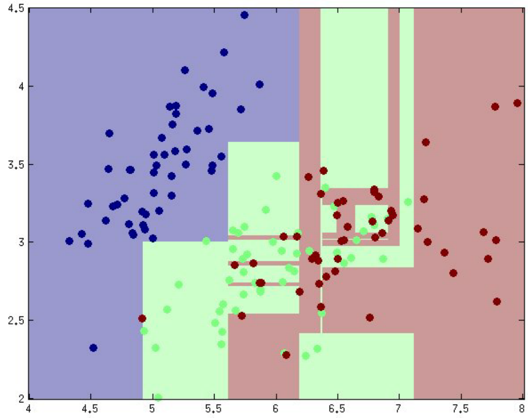
- Minimum # parent data



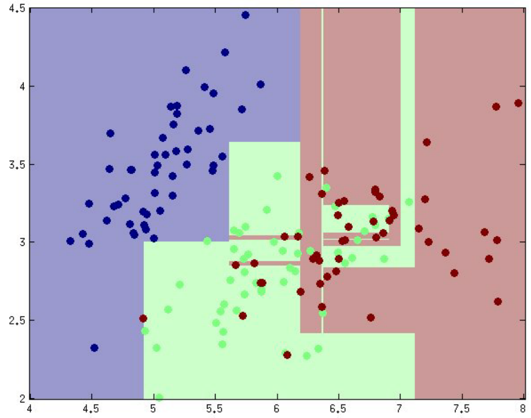
minParent 1



minParent 3



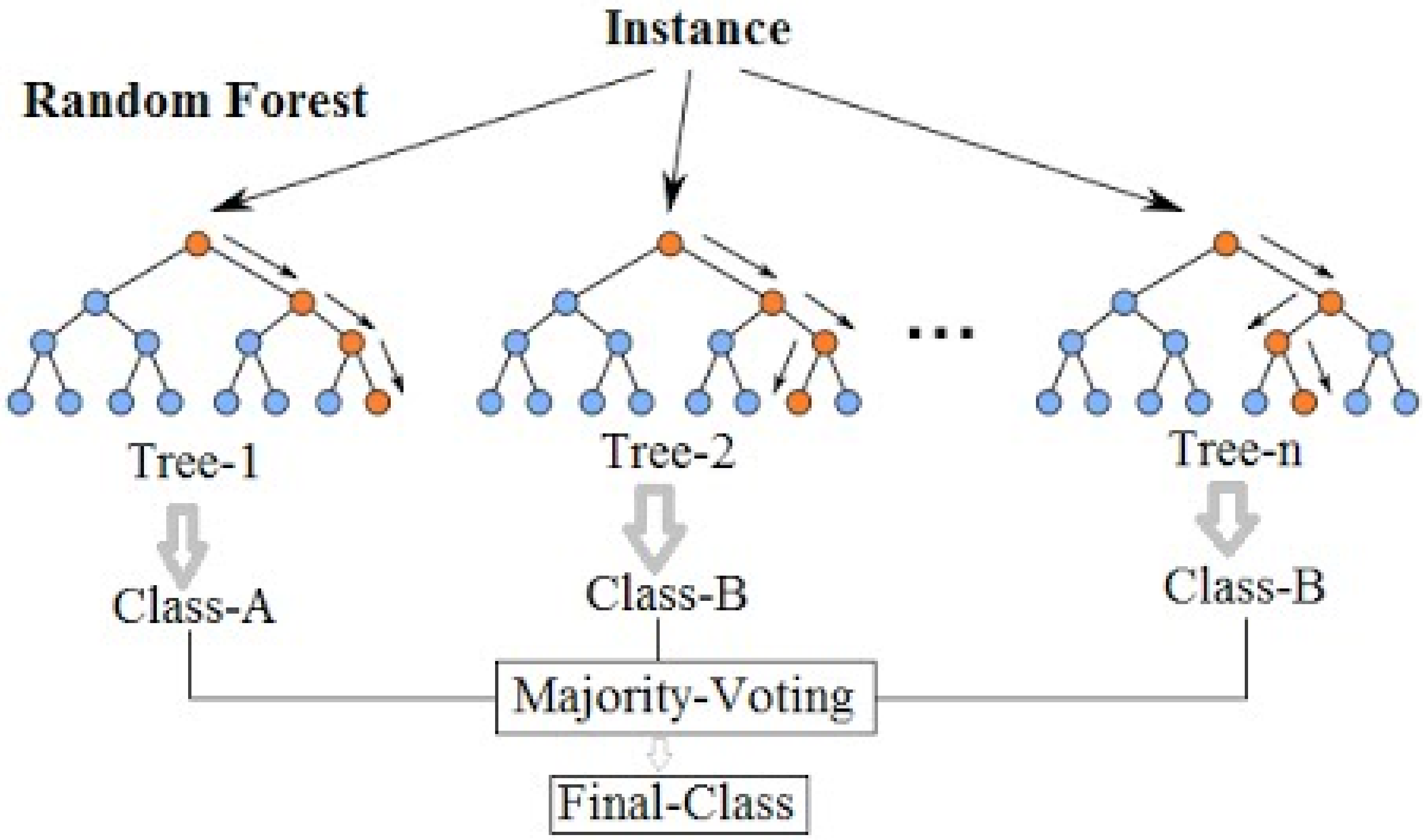
minParent 5



minParent 10

- Alternate (similar): min # of data per leaf

Improvement: Random Forests



Summary

- Decision trees
  - Flexible functional form
  - At each level, pick a variable and split condition
  - At leaves, predict a value

- Learning decision trees
  - Score all splits & pick best
    - Information gain, Gini index
  - Stopping criteria

- Complexity depends on depth
  - Decision stumps: very simple classifiers