UNIVERSITY of LOUISIANA LAFAYETTE

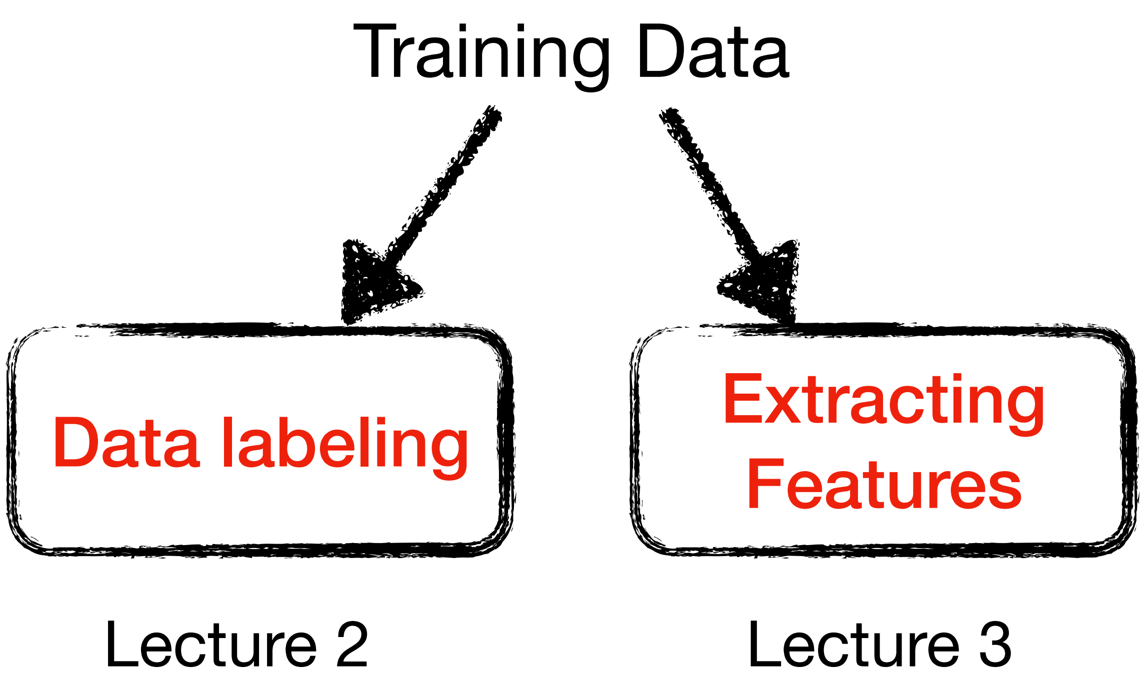
Lecture 4
Machine Learning for Classification

Xu Yuan
University of Louisiana at Lafayette

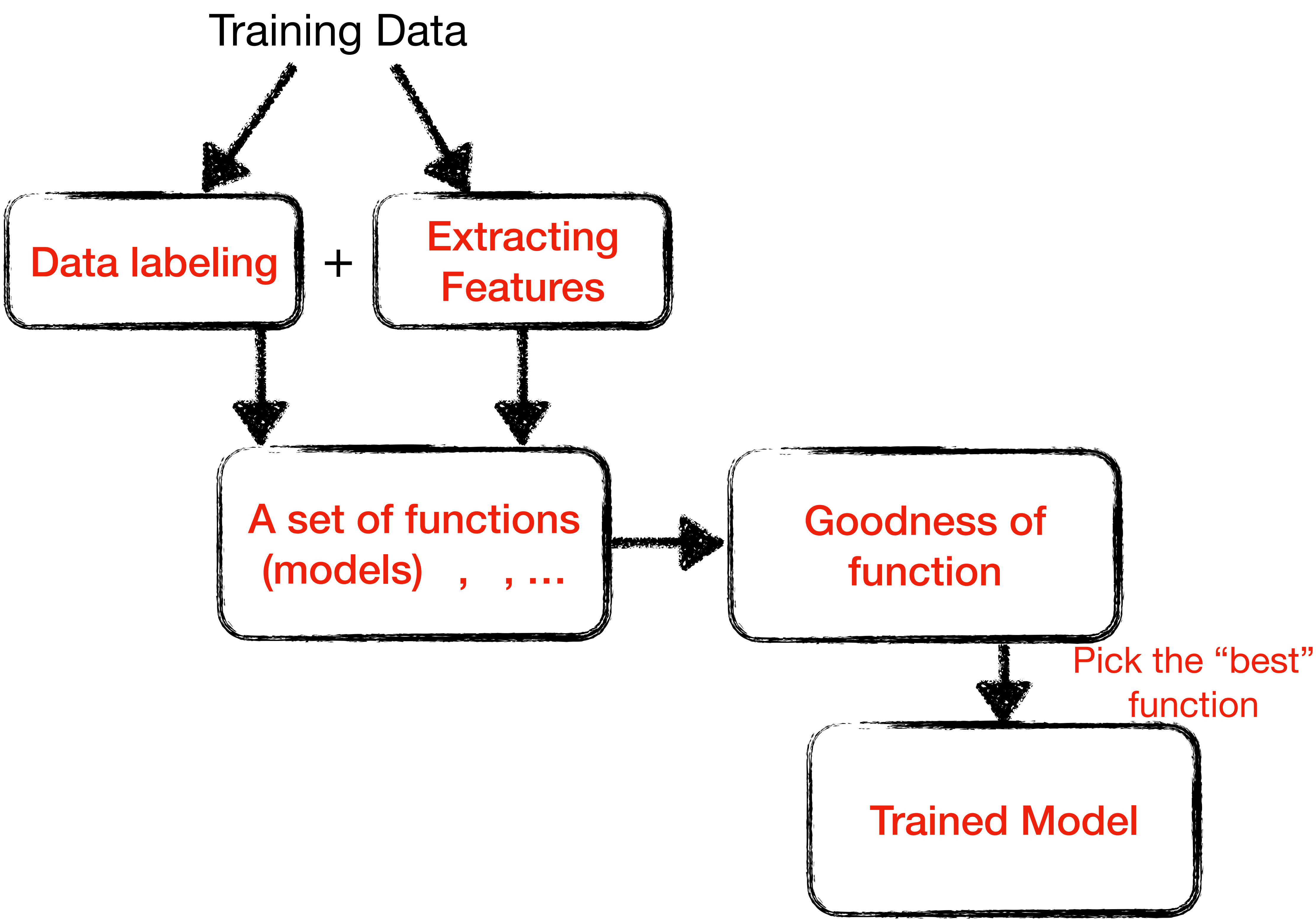
Training

Training Data

Training


Training Data
Data labeling
Extracting Features
Lecture 2
Lecture 3

Training


Training Data
Data labeling
+
Extracting Features
A set of functions (models) , , ...
Goodness of function
Pick the "best" function
Trained Model

Traditional Machine Learning Algorithms

- **Support Vector Machines (SVM)**
- **Random Forest**
- **K-Nearest Neighbors**
- **Decision Tree**

Classification

How to evaluate the performance of trained model?

Classification Results

- **True Positive (TP)**
  - The number of spams that are classified as spams
- **False Positive (FP)**
  - The number of non-spams that are classified as spams
- **False Negative (FN)**
  - The number of spams that are classified as non-spams
- **True Negative (TN)**
  - The number of non-spams that are classified as non-spams

Performance Metrics

- **Four Metrics**
  - Accuracy = $\dfrac{TP + TN}{TP + TN + FP + FN}$ — The percentage of the correctly classified spams and non-spams in the dataset
  - Precision = $\dfrac{TP}{TP + FP}$ — The percentage of the real spams in the classified spams
  - Recall = $\dfrac{TP}{TP + FN}$ — The percentage of the truly classified spams in the real spams
  - F1 Score = $2 * \dfrac{Precision * Recall}{Precision + Recall}$

An Example

- **100 Tweets: 40 spams and 60 non-spams**
    - After classification: 45 spams =35 real spam + 10 non-spams, 55 non-spams = 50 non-spams + 5 spams

45 Spams:
35 real spams and 10 non-spam

55 non-spams:
50 non-spams and 5 spams

An Example

- **100 Tweets: 40 spams and 60 non-spams**
  - After classification: 45 spams =35 real spam + 10 non-spams, 55 non-spams = 50 non-spams + 5 spams

45 Spams:
35 real spams and 10 non-spam

55 non-spams:
50 non-spams and 5 spams

True Positive (TP) False Positive (FP) True Negative (TN) False Negative (FN)

An Example

- **100 Tweets: 40 spams and 60 non-spams**
  - After classification: 45 spams =35 real spam + 10 non-spams, 55 non-spams = 50 non-spams + 5 spams

45 Spams:
35 real spams and 10 non-spam

55 non-spams:
50 non-spams and 5 spams

True Positive (TP) False Positive (FP) True Negative (TN) False Negative (FN)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{35 + 50}{35 + 10 + 50 + 5} = 85\,\%$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{35}{35 + 10} = 77.777\,\%$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{35}{35 + 5} = 87.5\,\%$$

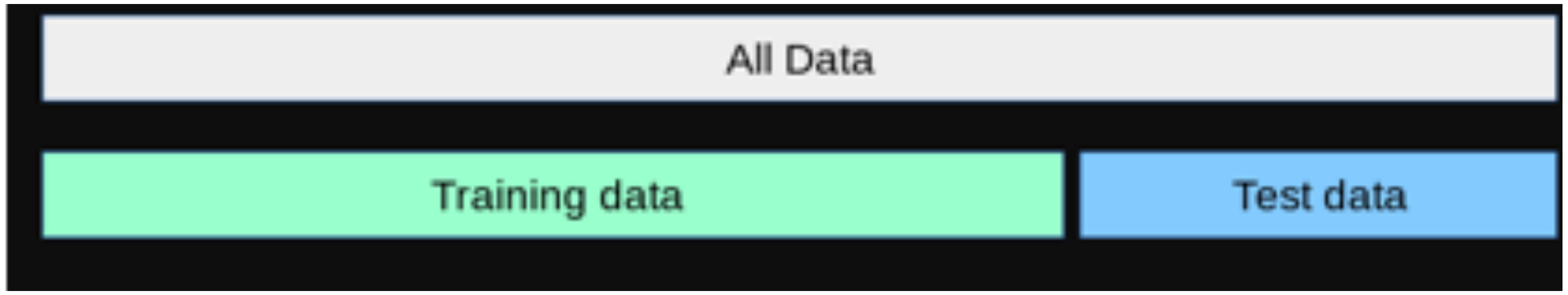
$$\text{F1 Score} = 2 * \frac{77.77\% * 87.5\,\%}{77.77\% + 87.5\,\%} = 82.34\,\%$$

Prediction

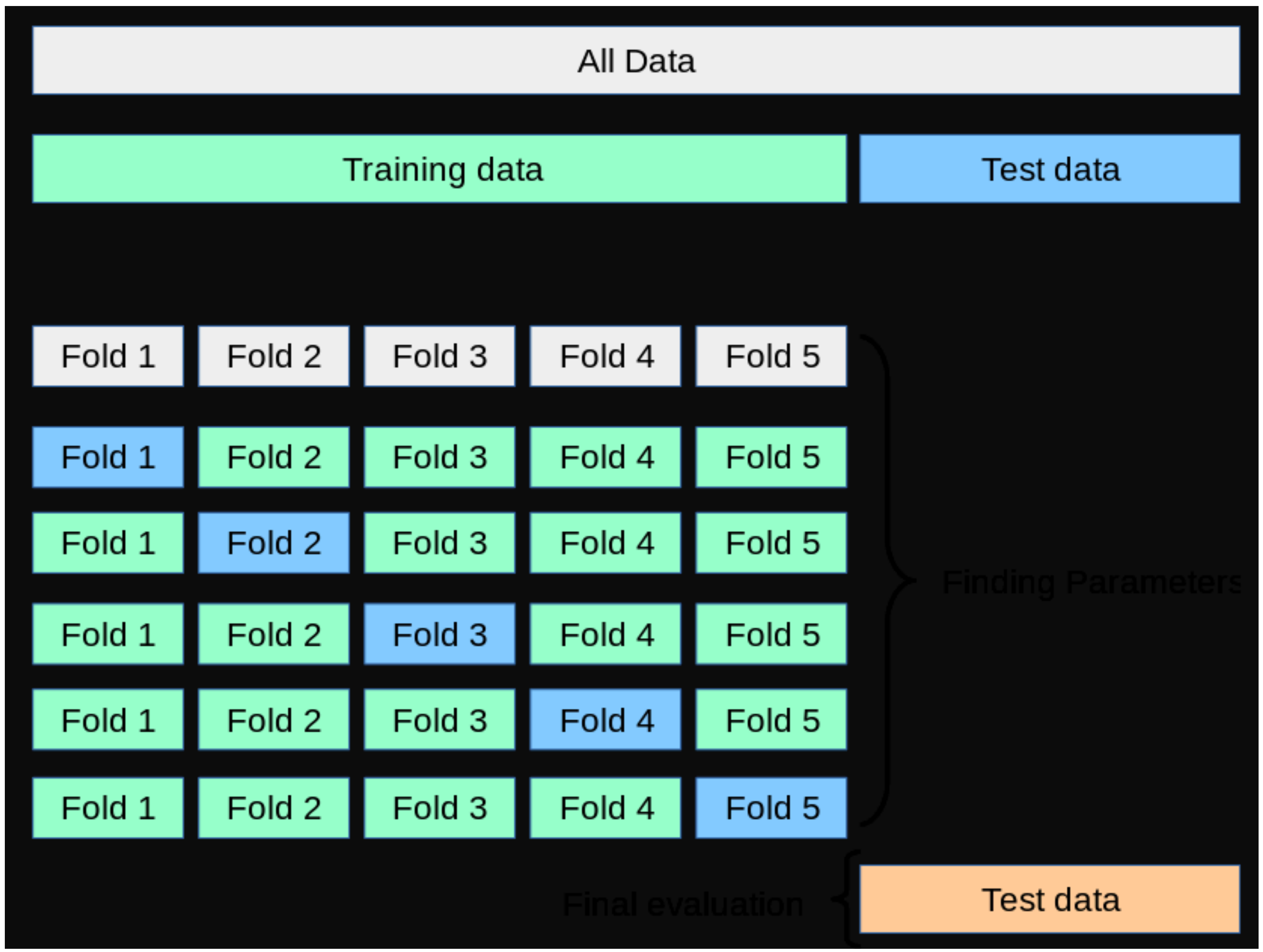
- **How to validate the correctness of your classification**
  - On testing data directly?

In real world, no ground observation for comparison!

- **The strategy is to label a large dataset**
  - Partition the labeled ground truth as training + testing



5-fold Cross-Validation



Decision Tree

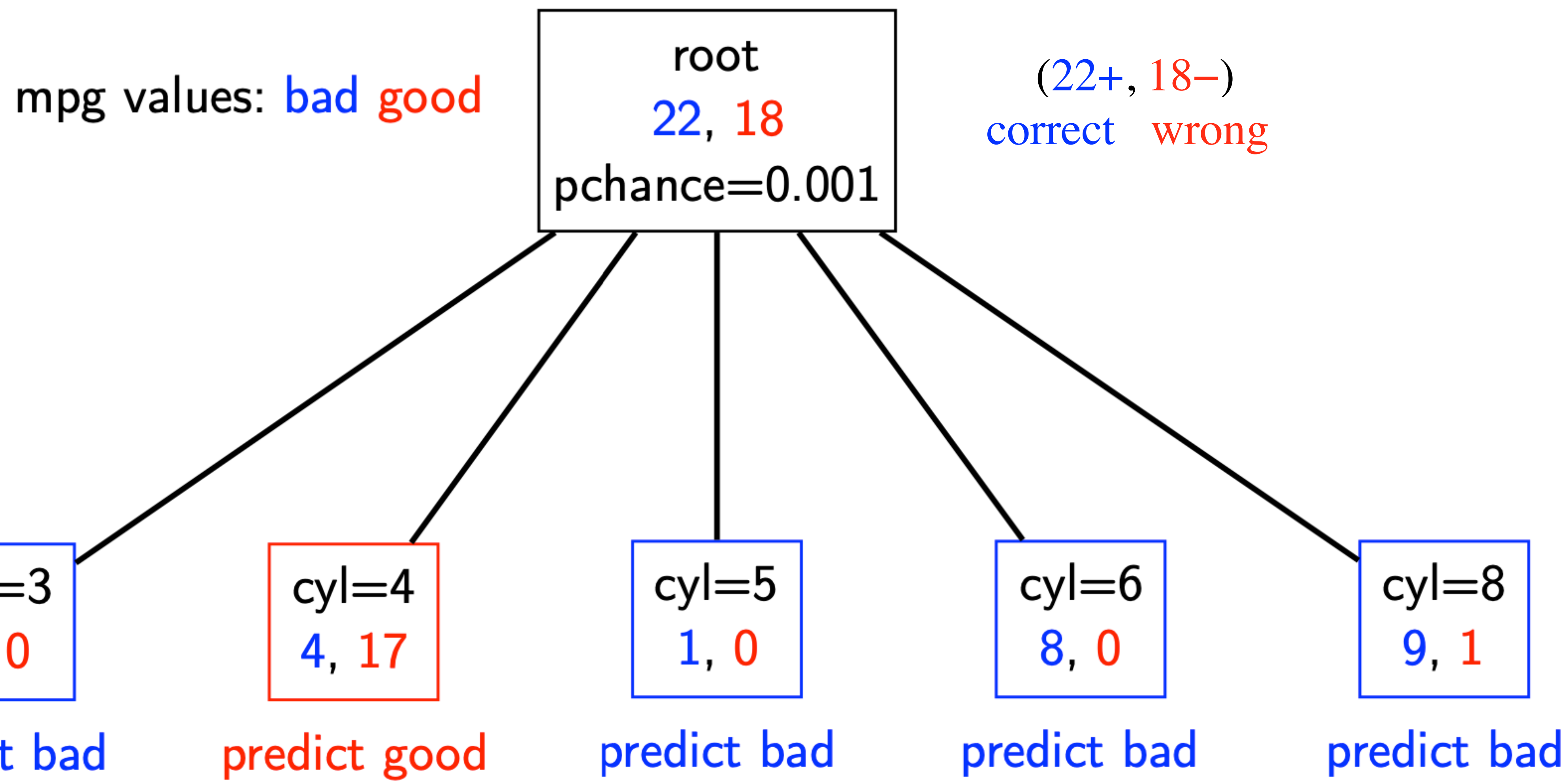
- **What is the simplest tree?**

Table: From the UCI repository

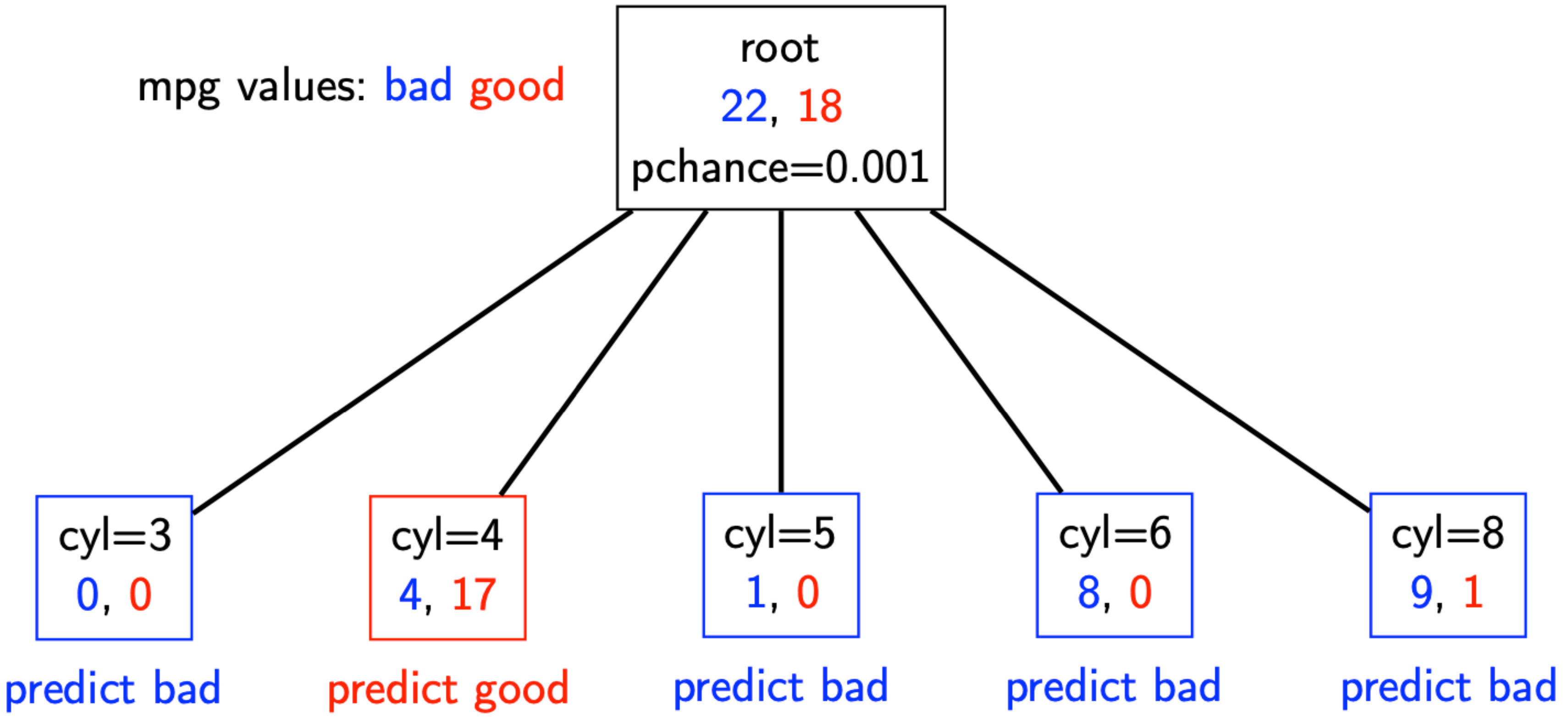
cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	mpg
4	low	low	low	high	75-78	asia	good
6	medium	medium	medium	medium	70-74	america	bad
8	high	high	high	low	70-74	america	bad
4	medium	medium	medium	low	75-78	europe	bad
...	...	...	...	...	...	...	...
4	low	medium	low	medium	75-78	europe	good

- Predict mpg=bad
- Is this a good tree? Total we get $(22+, 18-)$, which means we are correct on 22 examples and wrong on 18 examples.

A Simple Decision Tree

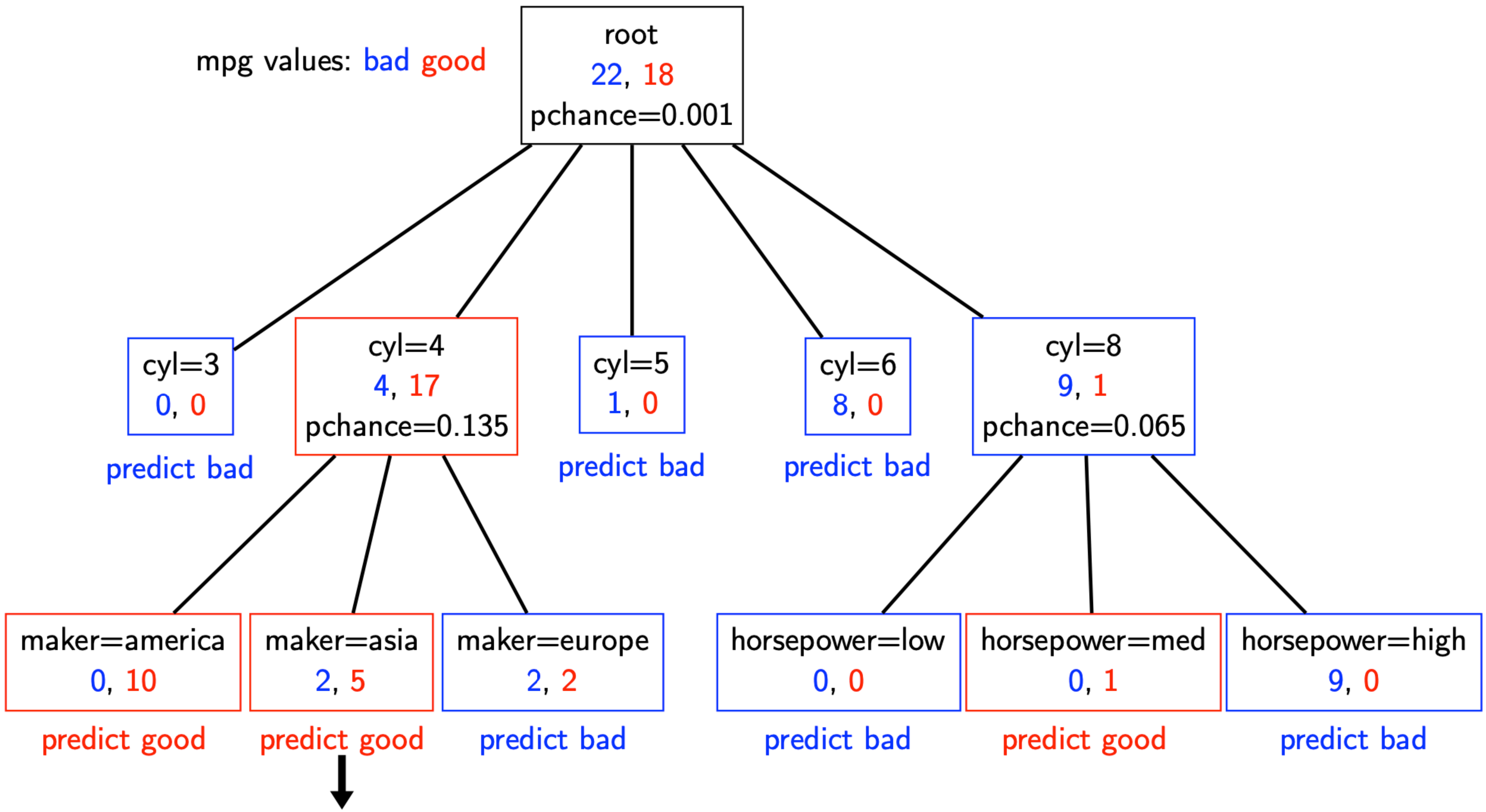


Recursive step



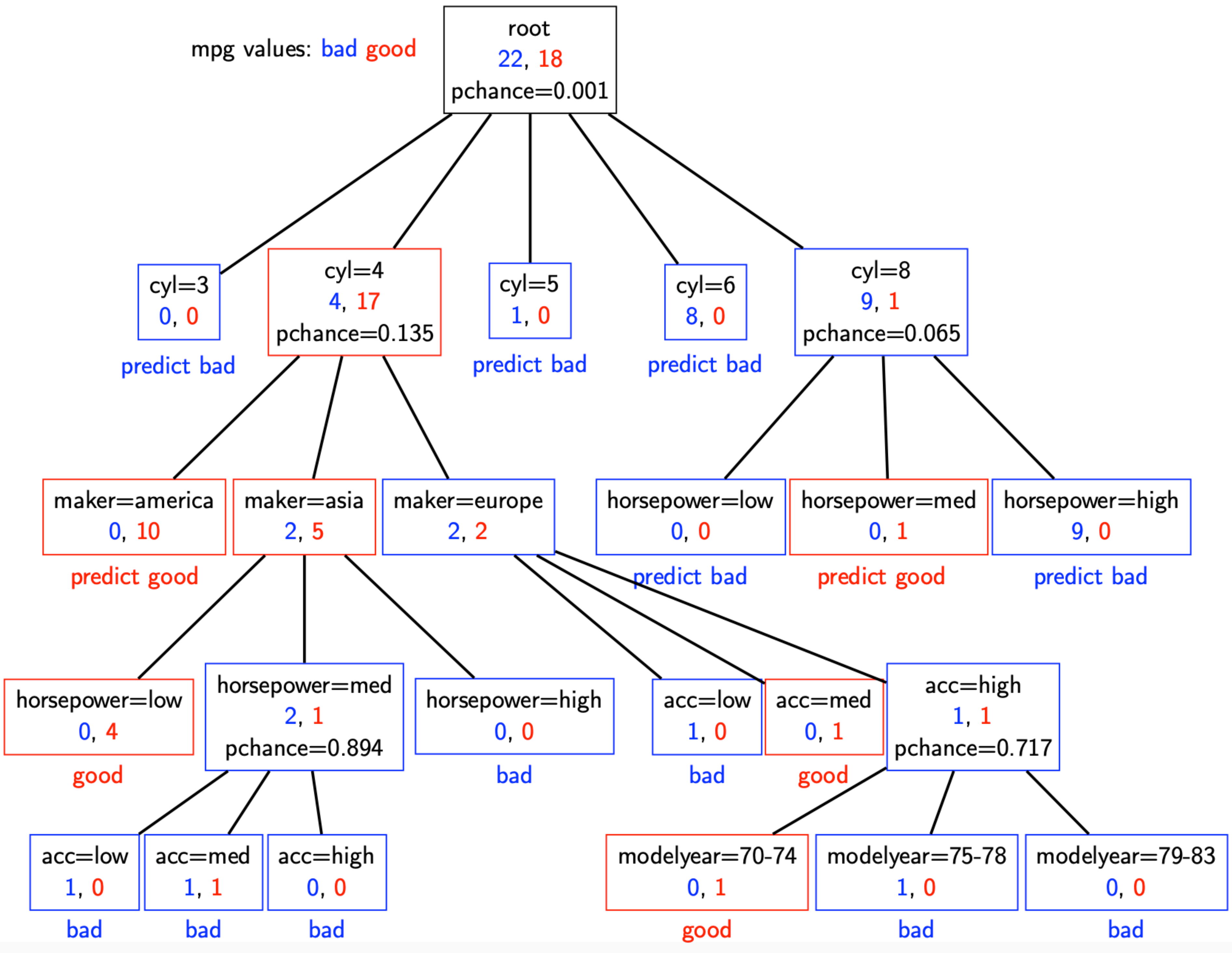
- Take the original dataset
- Partition it according to the values of the attribute we split on
- Build tree from these records (cyl=4, cyl=5, cyl=6, cyl=8)

Second Level of a Decision Tree



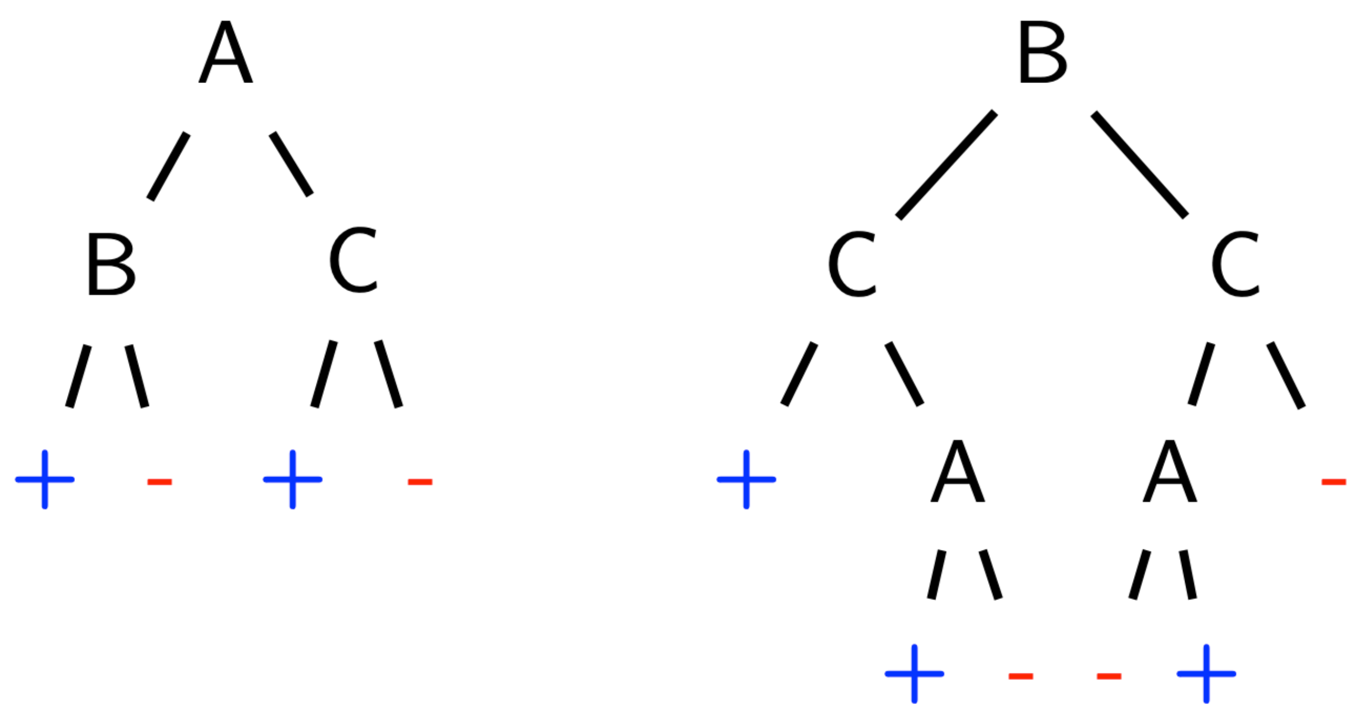
recursively build a tree from these records
in which cyl=4 and maker=Aisa

A Full Decision Tree



Decision Tree

- **Many trees can represent the same concept**



Is there a better method?

Entropy

- Entropy $H(Y)$ of a random variable Y:

$$H(Y) = -\sum_{i=1}^{K} P(Y = y_i) \log P(Y = y_i)$$

- More uncertainty, more entropy!
- Information theory interpretation: $H(Y)$ is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)

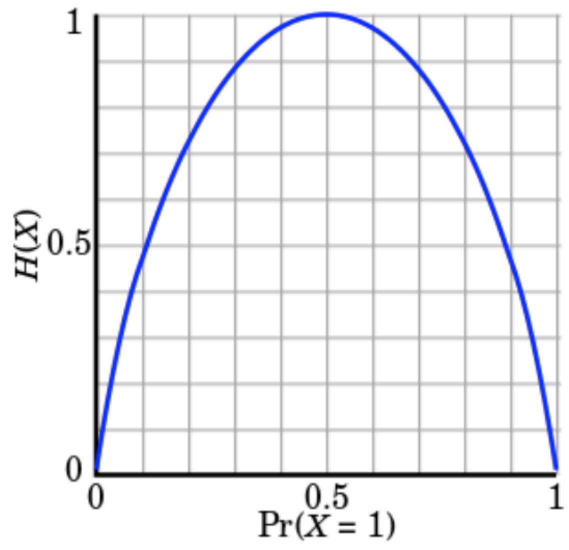


Figure: Entropy of a coin flip

Entropy

- High Entropy
  - Y is from a uniform like distribution
  - Flat histogram
  - Values sampled from it are less predictable
- Low Entropy
  - Y is from a varied distribution(peaks and valleys)
  - Histogram has many lows and highs
  - Values sampled from it are more predictable

Entropy Example

Entropy:

$$H(Y) = -\sum_{i=1}^{K} P(Y = y_i) \log P(Y = y_i)$$

In this example:

$$P(Y = T) = 5/6$$
$$P(Y = F) = 1/6$$
$$\begin{aligned} H(Y) &= -5/6 \log 5/6 - 1/6 \log 1/6 \\ &= 0.65 \end{aligned}$$

X_1	X_2	**Y**
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Conditional entropy

Conditional entropy $H(Y|X)$ of a random variable Y conditioned on a random variable X:

$$H(Y|X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{K} P(Y = y_i | X = x_j) \log P(Y = y_i | X = x_j)$$

In this example:

$$\begin{aligned} P(X_1 = T) &= 4/6 \\ P(X_1 = F) &= 2/6 \\ H(Y|X_1) &= -4/6(1 \log 1 + 0 \log 0) \\ &\quad -2/6(1/2 \log 1/2 + 1/2 \log 1/2) \\ &= 2/6 \end{aligned}$$

X_1	X_2	**Y**
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Information Gain

Used by the ID3, C4.5 and C5.0 tree-generation algorithms.
Decrease in entropy (uncertainty) after splitting:

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

In this example:

X_1	X_2	**Y**
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$$\begin{aligned} IG(X_1) &= H(Y) - H(Y|X_1) \\ &= 0.65 - 0.33 \end{aligned}$$

We prefer the split ($IG(X_1) > 0$)

Decision Tree

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Decision Tree

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

$$\begin{aligned} Info_{age}(D) = {} & \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ & + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) \\ & + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ & = 0.694 \text{ bits.} \end{aligned}$$

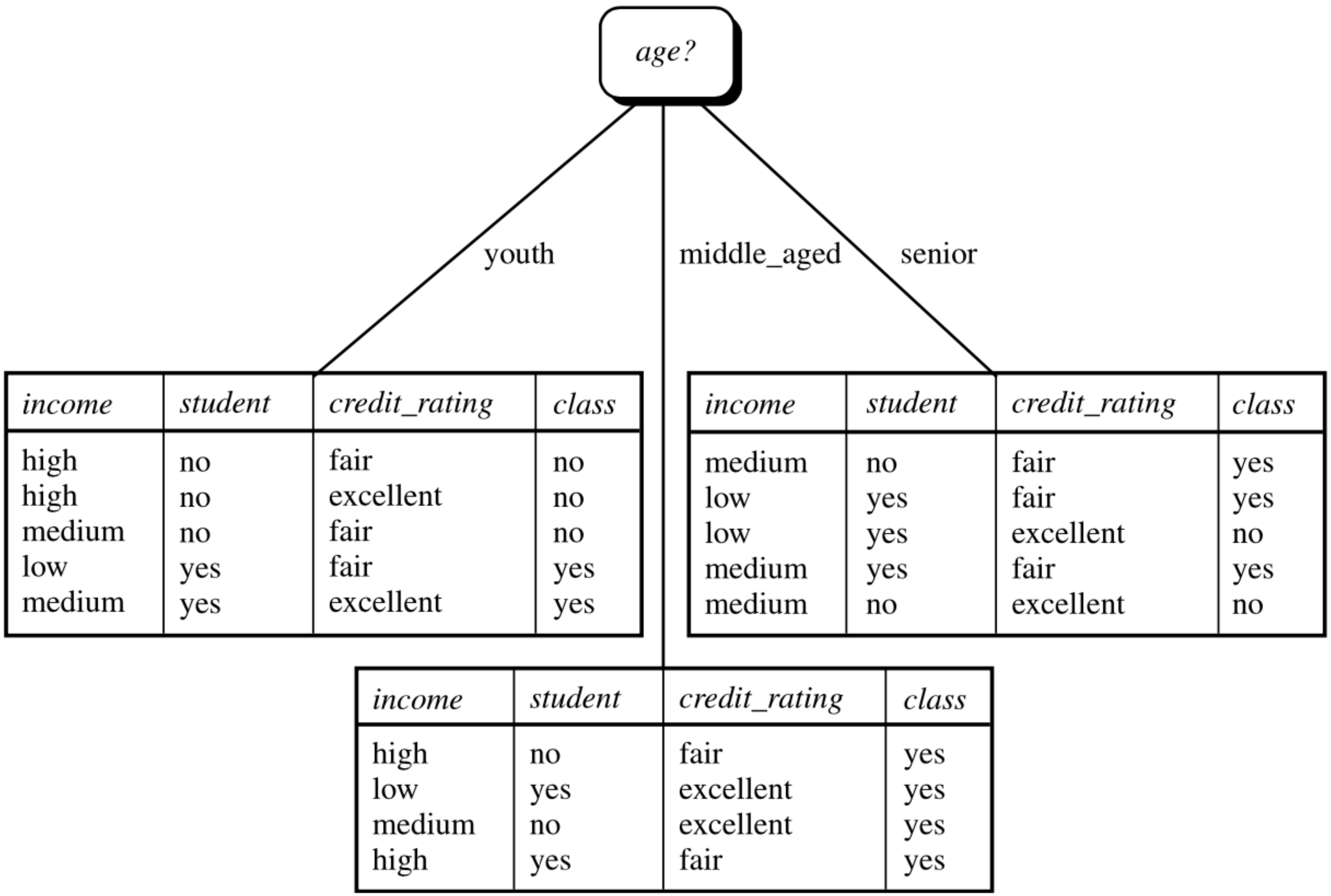
$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

$Gain(income) = 0.029$ bits

$Gain(student) = 0.151$ bits

$Gain(credit\ rating) = 0.048$ bits

Decision Tree

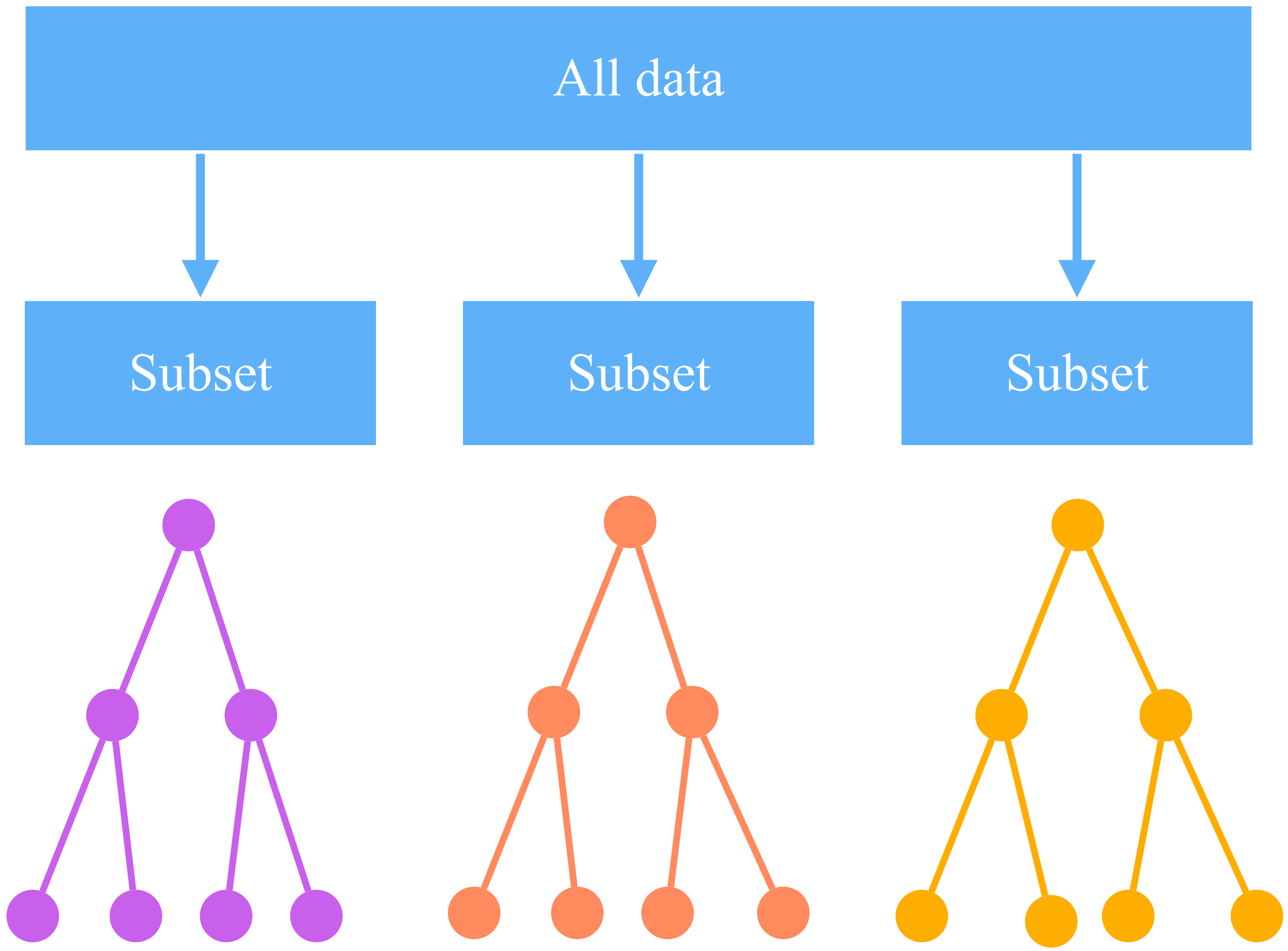


income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

Random Forest

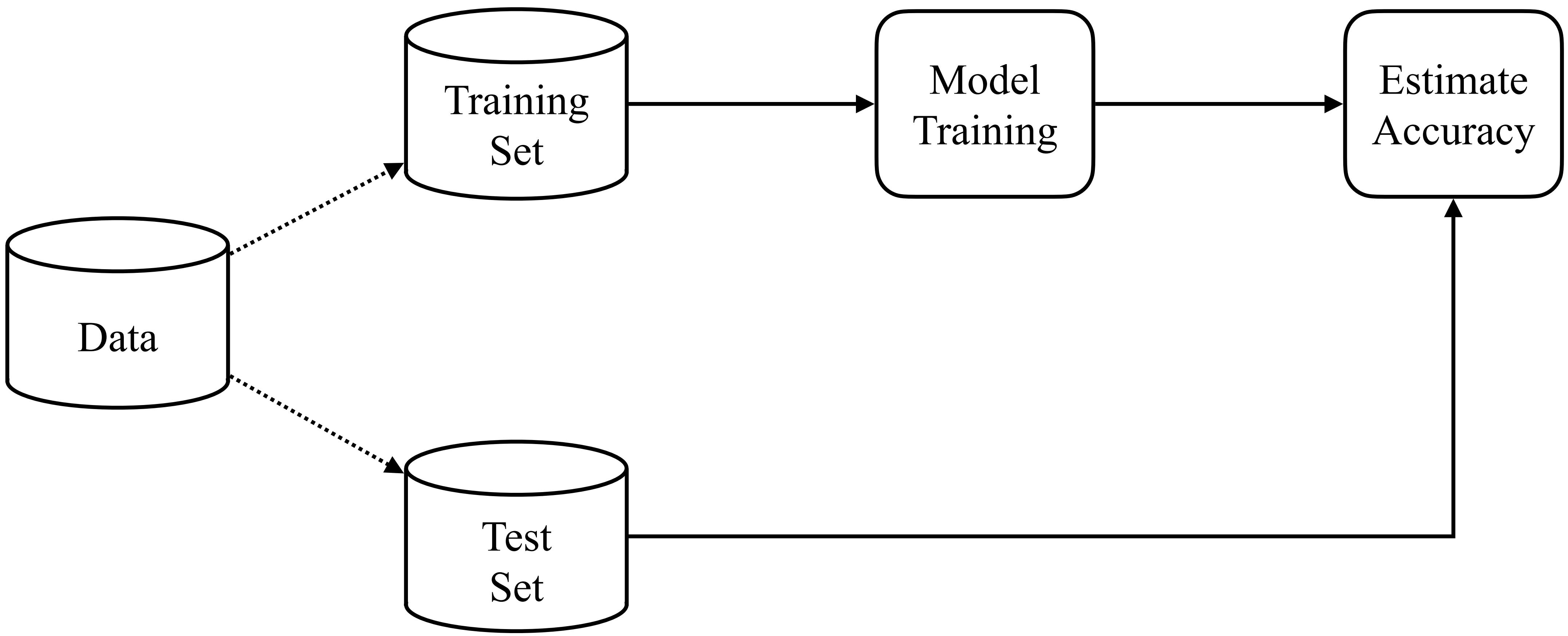


Scikit-learn Implementation

```python
# SVM
from sklearn import svm
clf = svm.SVC()
clf.fit(X, Y)

# Random Forest
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
rf.fit(X, Y)

# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X, Y)

# Decision Tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0)
clf.fit(X, Y)
```

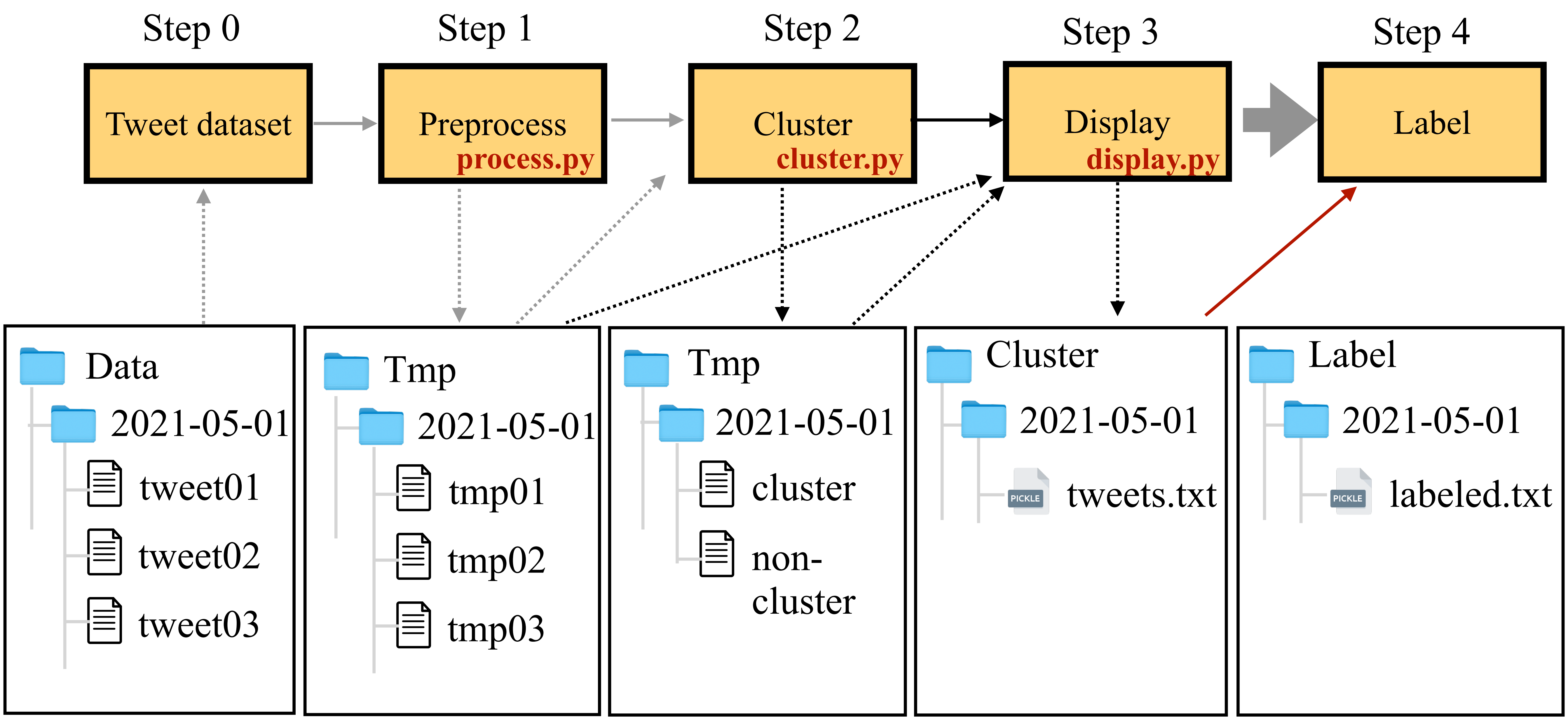
Hands-on

- **Model**



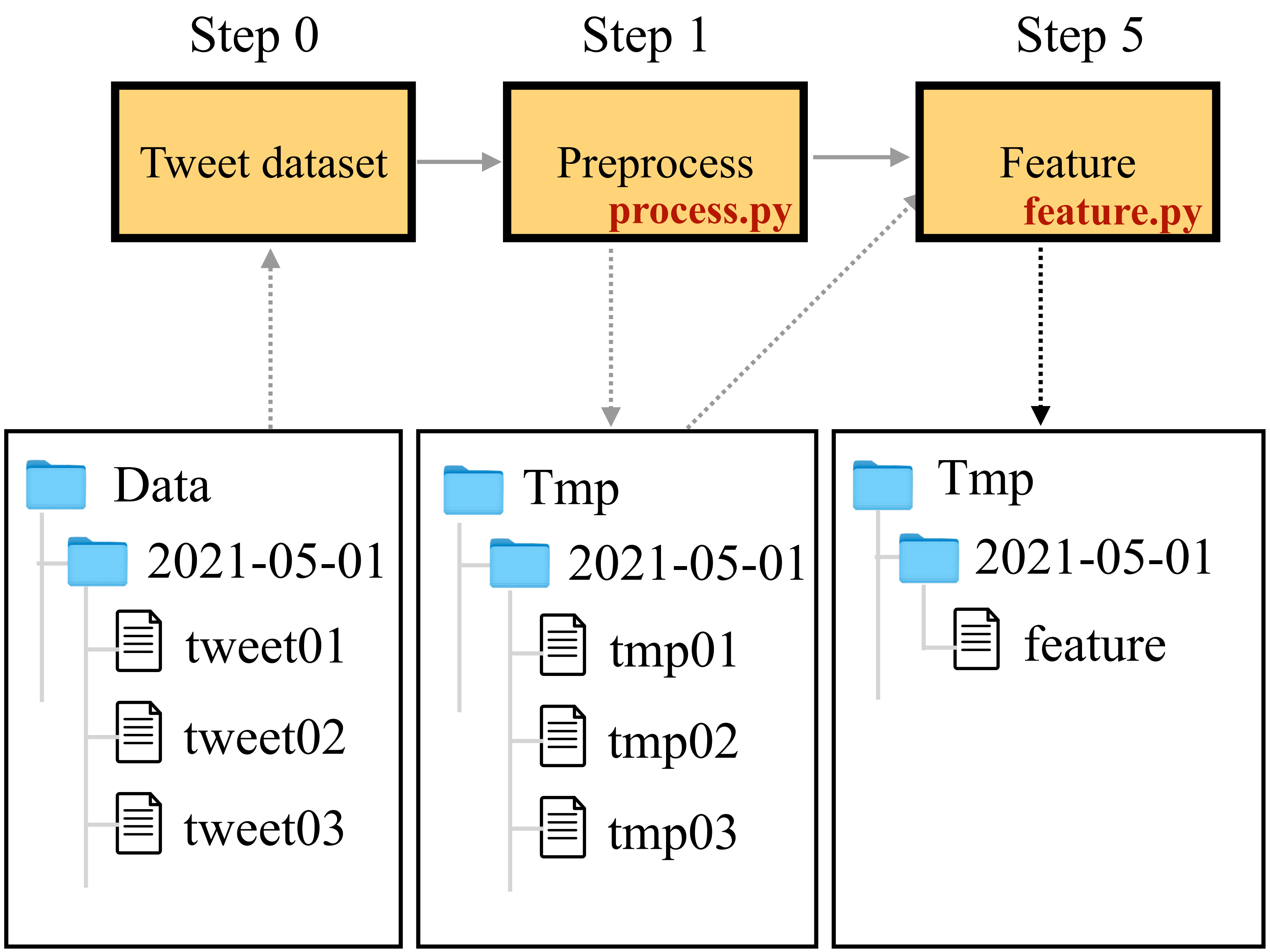
Hands-on

- **Clustering Tweets**



Hands-on

- **Feature extraction**



Hands-on

- **Feature extraction**