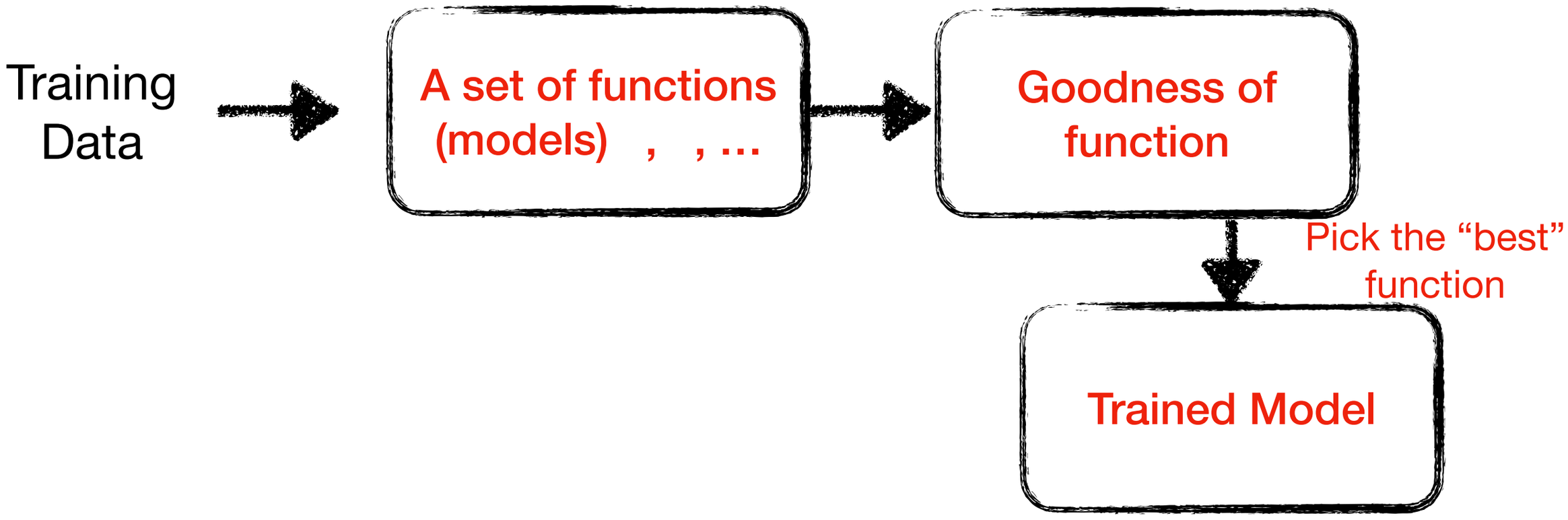
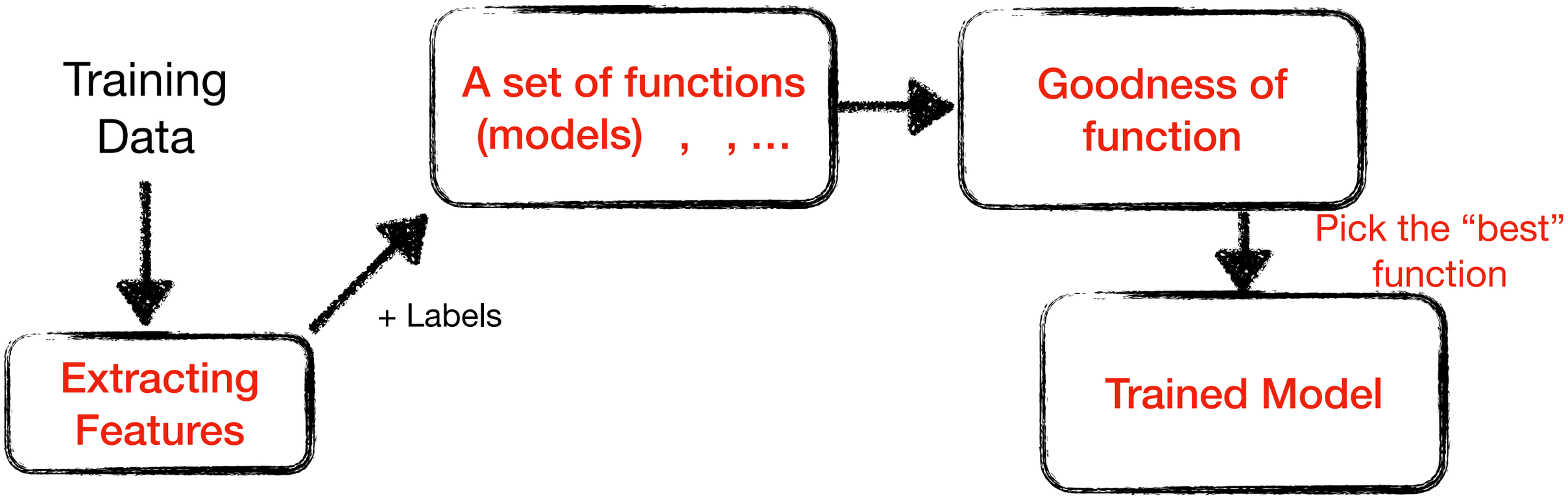
Lecture 2
Feature Extraction

Xu Yuan
University of Louisiana at Lafayette

Data Input = Features Input



Data Input = Feature Input



Real-World Example

When recognizing a person, we compare the face with those stored in memory.

We cannot always remember all the details of a face, but we can still recognize a person.

Previous Example

```
from sklearn import svm

X = [[0, 1],[1, 2],[2, 1],[2, 3],[1, 3],[2, 2]]

y = ['a', 'a', 'b', 'b', 'a', 'b']

clf = svm.SVC()

clf.fit(X, y)

result1 = clf.predict([[3, 1]])

print(result1)

result2 = clf.predict([[0, 2]])

print(result2)
```

Feature representation

```
['b']
['a']
```

Features

- A set of attributes with their values can represent a data record
- Data record = Feature vectors
- With their determinant values, a machine learning model can determine its class

Feature Selection

- **A preprocessing step to choose a subset of original features according to certain criterion**
  - Find the representative data
  - Remove redundant or meaningfulness data
  - Reduce effect of irrelevant data
  - Toward learning accuracy

Relevant

Irrelevant

Redundant

A feature is good if it is relevant to the class task but is not redundant to any of the other relevant features!

Objective of Feature Selection

- **A preprocessing step to choose a subset of original features according to certain criterion**
  - Avoid overfitting and achieve better generalization ability
  - Improve the prediction performance of the predictors
  - Provide a faster and more cost-effective predictors
  - Provide a better understanding of the underlying process that are generating the data

- **Feature Dimensionality Reduction Technique**
  - Principle Component Analysis (PCA)
  - Independent Component Analysis (ICA)
  - Linear Discriminant Analysis (LDA)
  - Local Linear Embedding (LLE)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
  - Autoencoders
  - https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be

Scikit-learn: VarianceThreshold()

- **This feature selector removes all low-variance features**

```
# if not installed, install sklearn
!pip install sklearn

from sklearn.feature_selection import VarianceThreshold
# dataset with three boolean features
X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0],
[0, 1, 1]]
sel = VarianceThreshold(threshold=(.8 * (1 - .8))) # set
threshold value
sel.fit_transform(X) #Reduce X to the selected features
```

```
array([[0, 1],
       [1, 0],
       [0, 0],
       [1, 1],
       [1, 0],
       [1, 1]])
```

Scikit-learn: VarianceThreshold()

- **This feature selector removes all low-variance features**

```
from sklearn.feature_selection import VarianceThreshold
#Sample dataset integer features, two of which are the same
in every sample
X = [[0, 2, 0, 3], [0, 1, 4, 3], [0, 1, 1, 3]]
selector = VarianceThreshold()
selector.fit_transform(X) #Reduce X to the selected
features
```

```
array([[2, 0],
       [1, 4],
       [1, 1]])
```

Scikit-learn: SelectKBest()

- **Select k features according to the highest scores**

```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2
X, y = load_digits(return_X_y=True)
print(X.shape)

X_new = SelectKBest(chi2, k=20).fit_transform(X, y)# use
chi-squared stats and select k = 20 features
print(X_new.shape)
```

```
(1797, 64)
(1797, 20)
```

Scikit-learn: SelectPercentile()

- **Select features according to a percentile (percent of features to keep) of the highest score**

```
from sklearn.datasets import load_digits # load existing
data
from sklearn.feature_selection import SelectPercentile,
chi2
X, y = load_digits(return_X_y=True)
print(X.shape)

X_new = SelectPercentile(chi2,
percentile=10).fit_transform(X, y)# Percent of features to
keep = 10
print(X_new.shape)
```

```
(1797, 64)
(1797, 7)
```

Scikit-learn: GenericUnivariateSelect()

- **Perform univariate feature selection with a configurable strategy**

```
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import
GenericUnivariateSelect, chi2
X, y = load_breast_cancer(return_X_y=True)
print(X.shape)
```

```
transformer = GenericUnivariateSelect(chi2, mode='k_best',
param=20)# k_best, 20 features. The mode can be any from
the set {'percentile', 'k_best', ...}.
X_new = transformer.fit_transform(X, y)
print(X_new.shape)
```

```
(569, 30)
(569, 20)
```

Preview the Data

- **Be familiar with the data before deciding the features**

Friends/Followers

- **Number of Following/Followers**

  Following count and Follower count are important social network features.

Following count

Follower count

Tweet JSON

"follow_request_sent":false,
"followed_by":false,
"followers_count":2720235,
"friends_count":793,
"has_custom_timelines":true,
"is_translator":false,

Friends/Followers

- **Number of Following/Followers**

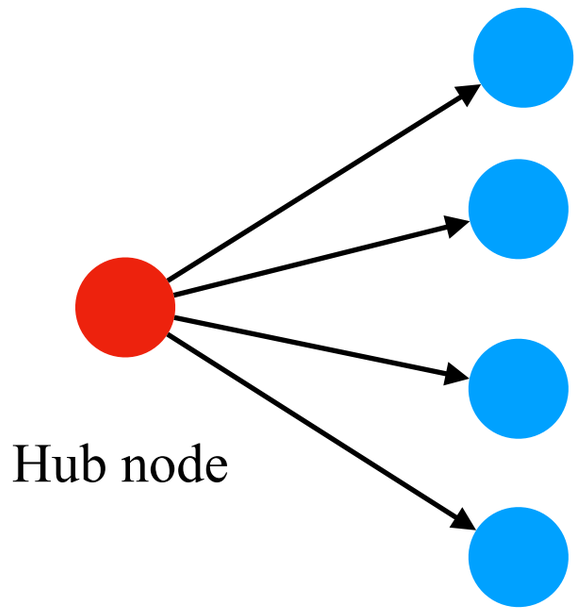
  Different types of accounts.

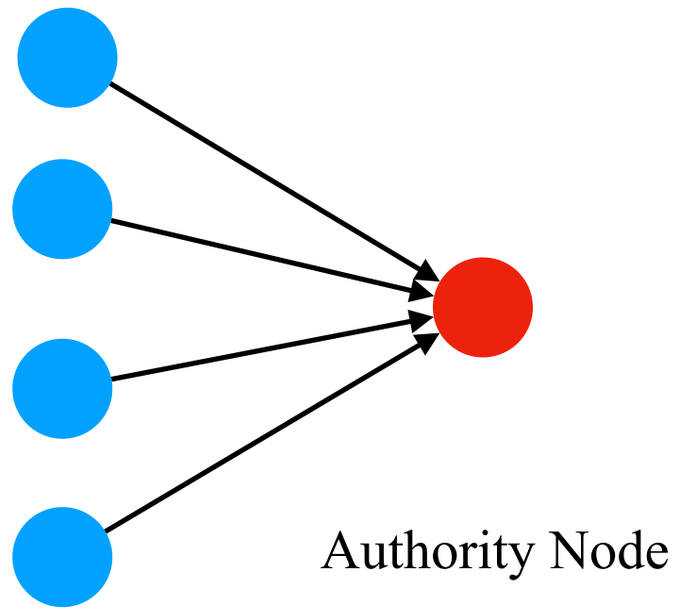
Non-famous Follower count Famous

Friends/Followers

- **Number of Following/Followers**

  Different types of accounts based on Followers count and Following count.

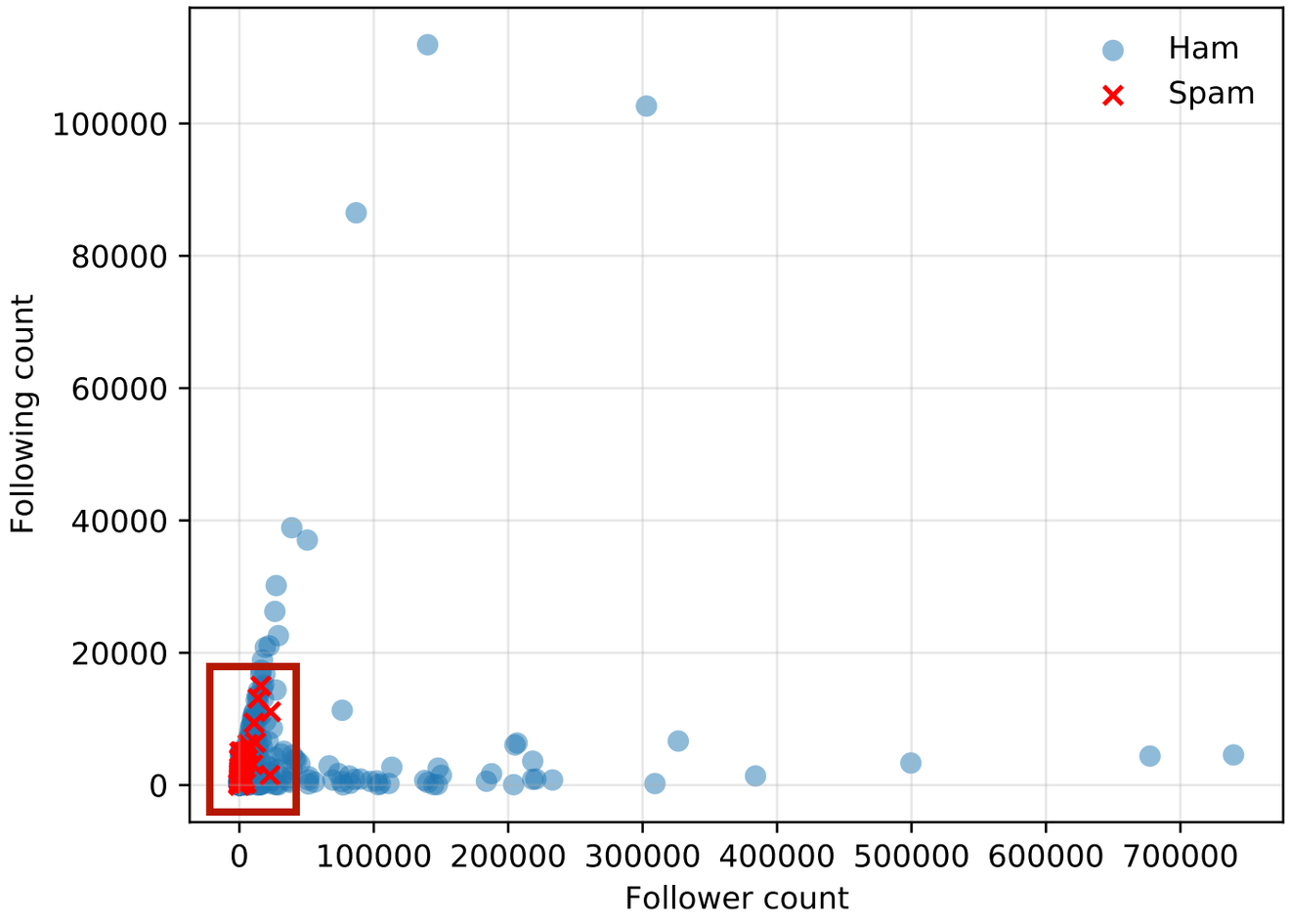




Friends/Followers

- **Number of Following/Followers**

  Followers count and Following count for 1000 randomly sample spams and hams.



Friends/Followers

- **Number of Following/Followers**

  Code example

```
with open(file_path,'r') as read_file:
    for line in read_file:
        tweet_json = json.loads(line)
        tweet_obj = Tweet(line)
        n_friends = tweet_obj.user.get_friends_count()
        n_followers = tweet_obj.user.get_followers_count()
```

```
class Tweet():
    '''
    TwitterTweet class can used to save a tweet
    '''
    def __init__(self, tweet_json):
        self.tweet_json = tweet_json
        self.user = User(tweet_json)
```

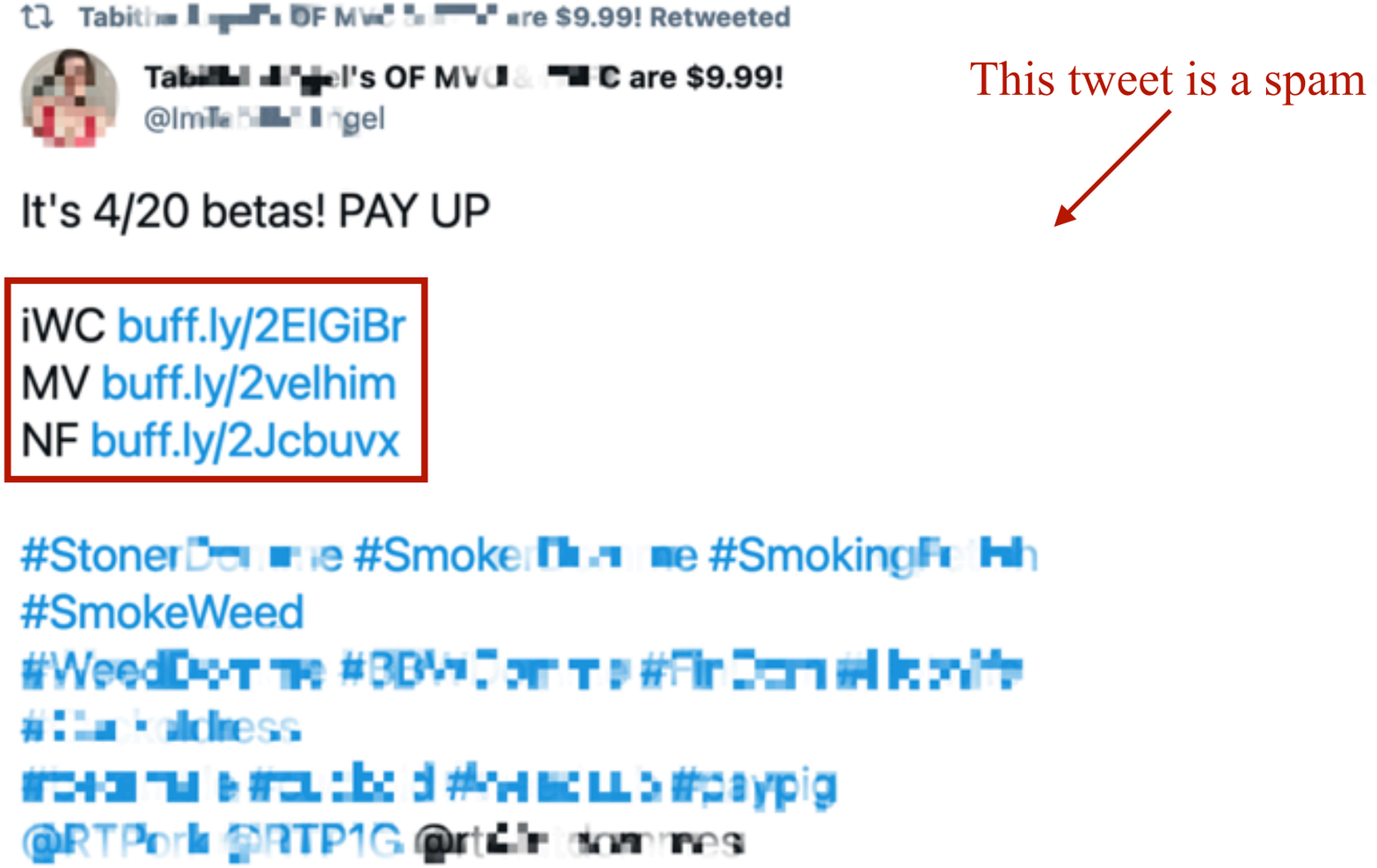
```
def get_followers_count(self):
    '''
    return follower count
    '''
    followers_count = self.user_json['followers_count']
    if followers_count == 0:
        followers_count = 1
    return followers_count
```

```
def get_friends_count(self):
    '''
    return friends count
    '''
    friends_count = self.user_json['friends_count']
    if friends_count == 0:
        friends_count = 1
    return friends_count
```

URLs

- **Number of URLs**

  Spam contains more URLs.



This tweet is a spam

3 URLs

URLs

- **Number of URLs**

3 URLs in JSON

Twitter short URL

External URL, this is a malicious URL

[{'url': 'https://t.co/Bc07wW2S12', 'expanded_url': 'http://onlyfans.com/[redacted]', 'display_url': 'onlyfans.com/mistressxmaya', 'indices': [49, 72]},

{'url': 'https://t.co/zdI7W21W6w', 'expanded_url': 'http://iwantclips.com/stor[redacted]', 'display_url': 'iwantclips.com/store/74741/mi…', 'indices': [73, 96]},

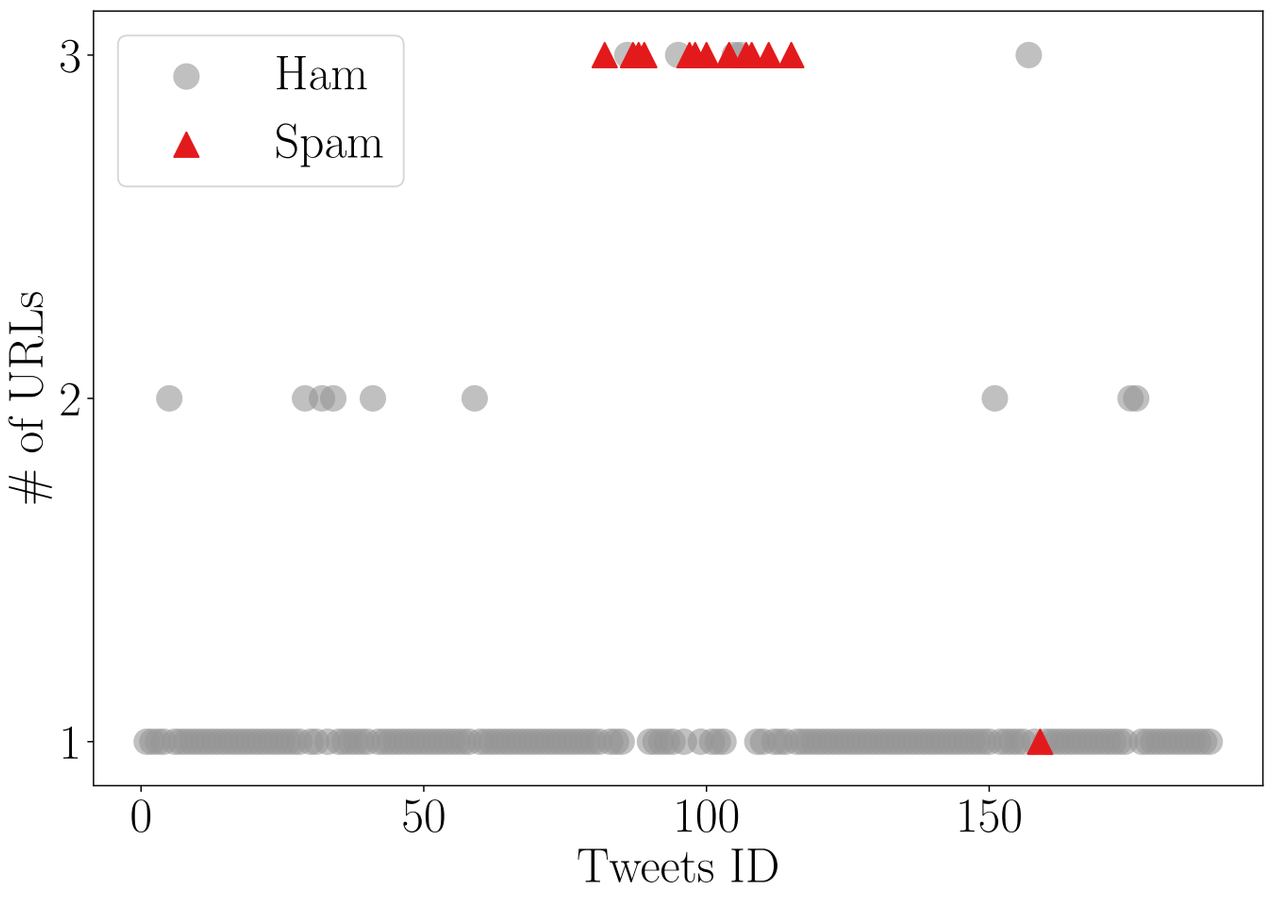
{‘url': 'https://t.co/Z89sz24uyc', 'expanded_url': 'https://twitter.com/i/web/status/987686235010461696', 'display_url': 'twitter.com/i/web/status/9…', 'indices': [116, 139]}]

This account has already been suspended by Twitter!

URLs

- **Number of URLs**

  Spam contains more URLs



URLs

- **Number of URLs**

  Code example

```
with open(file_path,'r') as read_file:
    for line in read_file:
        tweet_json = json.loads(line)
        tweet_obj = Tweet(line)
        n_url = tweet_obj.get_url_count()
```

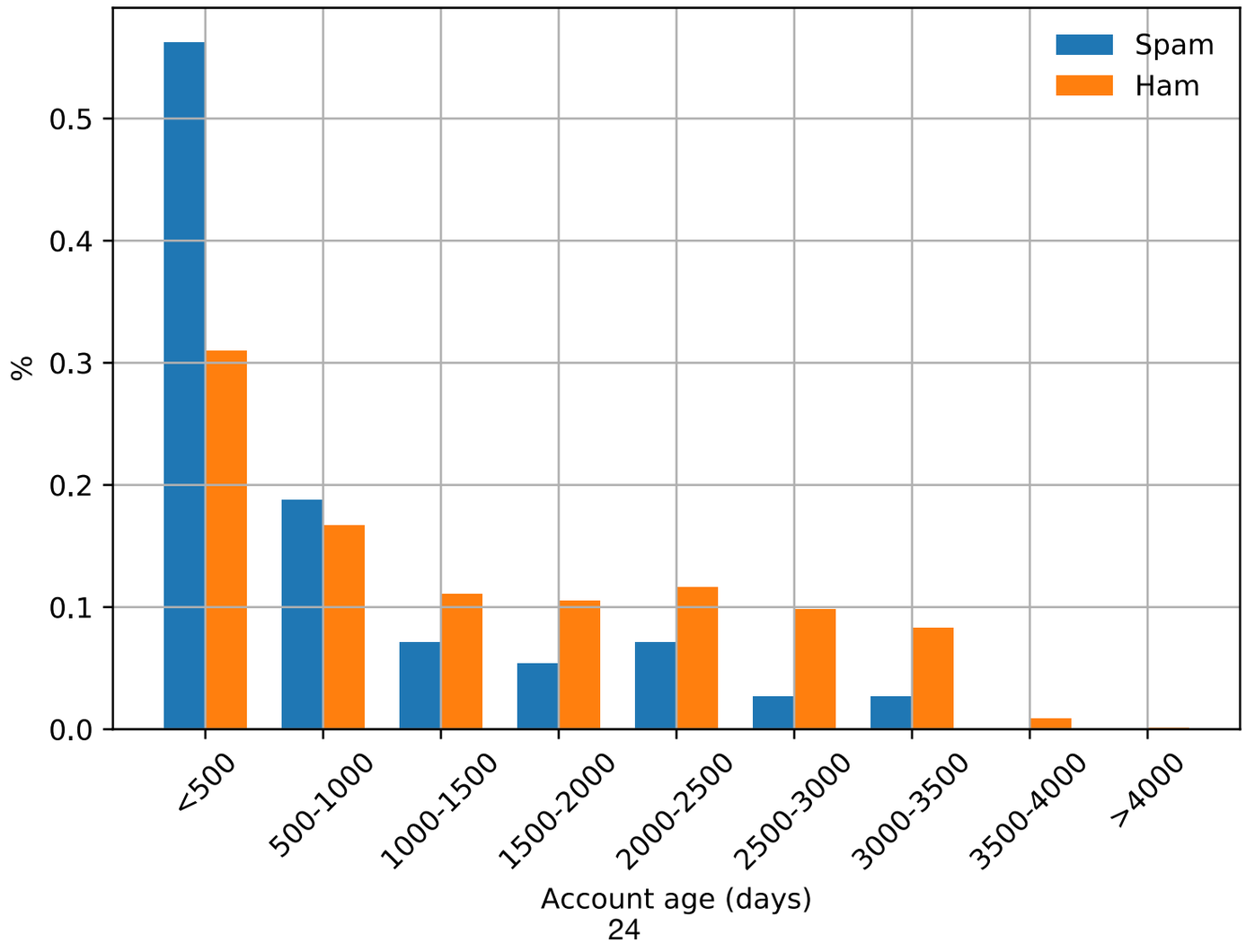
```
class Tweet():
    '''
    TwitterTweet class can used to save a tweet
    '''
    def __init__(self, tweet_json):
        self.tweet_json = tweet_json
        self.user = User(tweet_json)
```

```
    def get_url_count(self):
        '''
        get number of urls
        '''
        urls = self.tweet_json['entities']['urls']
        return len(urls)
```

Account Age

- **Account Age**

  Spam and ham age distribution are different.



Account Age

- **Account Age**

  Code example

```
def get_user_age(self):
    '''
    Age of an account
    get age feature of an account, remember call this function all the time. Time exchange
    '''
    account_start_time = self.json_date_to_os(self.user_json['created_at'])
    now_time = datetime.now()
    account_age = (now_time-account_start_time).days
    if account_age == 0:
        account_age = 1
    return account_age
```

Static Features

- **Account Profile**
  - Friends count, followers count, age, status count, average status, list count, average lists, average favorites, favorites count, verified status, default profile image, screen name length, name length, description length, description emoji count, and description digits count
- **Tweet Content**
  - Tweet status, tweet source platform, hashtag count, mention count, content length content emoji count, and content digits count
- **User Behaviors**
  - Reciprocity count, sender or receiver tweet distribution, sender or receiver source distribution, mention time, average tweet interval, environment score

https://ieeexplore.ieee.org/abstract/document/8809491
https://people.cmix.louisiana.edu/yuan/resources/pdf/19_DSN_Pseudo.pdf

Tweet Features on Lab 2

- **Feature List**

1: User account age — **get_user_age()**

2: The length of user description — **get_description_len()**

3: The number of followers — **get_followers_count()**

4: The number of following — **get_friends_count()**

5: The number of user favorites — **get_user_favorites()**

6: The number of user lists — **get_user_lists()**

7: The number of user statuses — **get_statuses_count()**

8: The number of tweet hashtags — **get_hashtag_count()**

9: The number of tweet mentions — **get_mention_count()**

10: The number of URLs — **get_url_count()**

11: The length of tweet text — **get_text_len()**

12: The number of digits in tweet — **get_text_digits()**

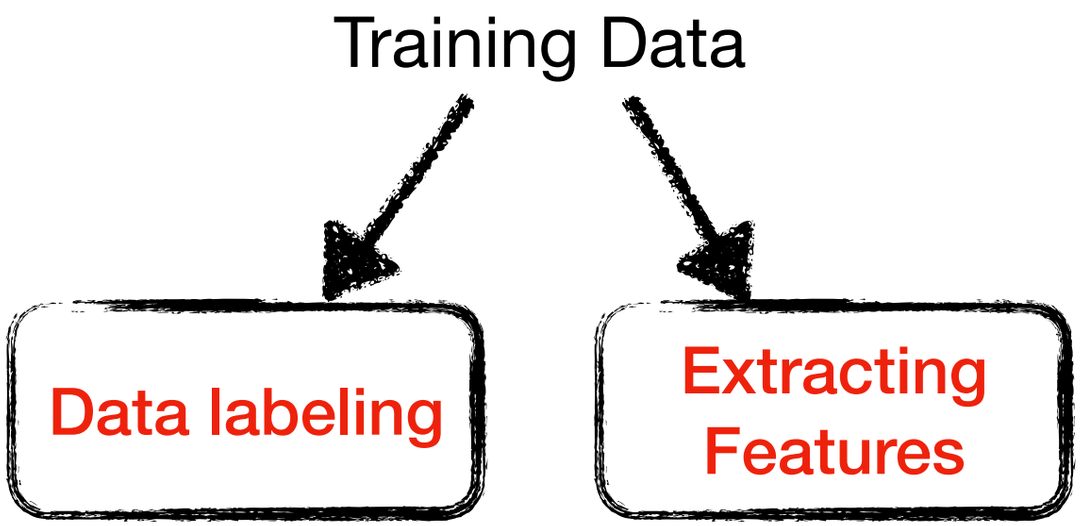
Machine Learning for Classification

Xu Yuan
University of Louisiana at Lafayette

Training

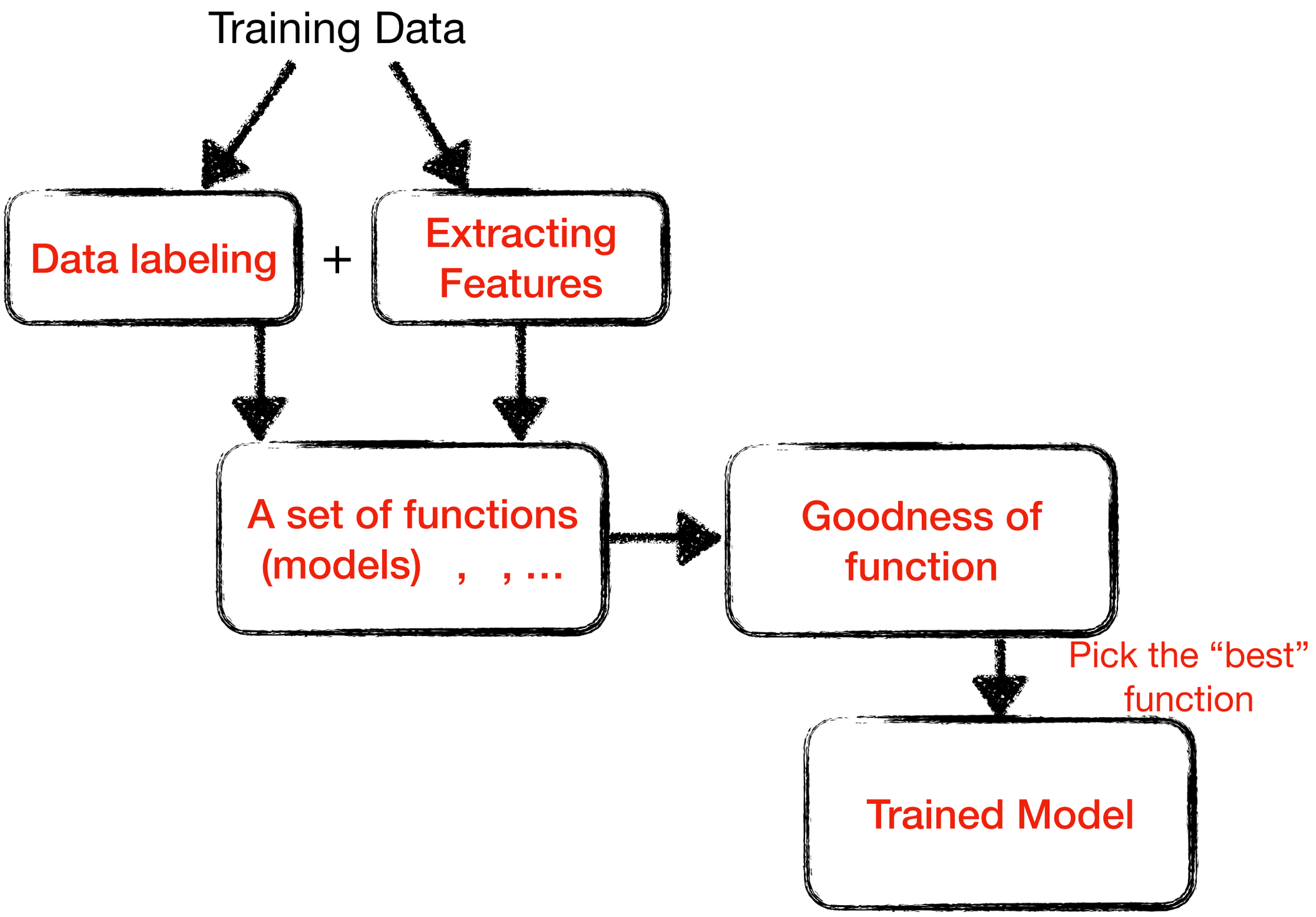
Training Data

Training



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

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)

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

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

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

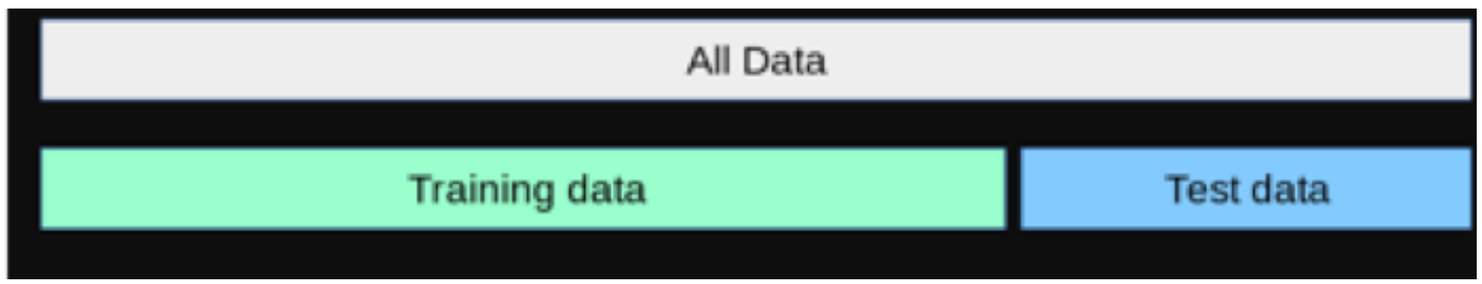
$$\text{F1 Score} = 2 * \frac{77.77\% * 87.5\,\%}{77.77\% + 87.5\,\%} = 82.34\,\%$$

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

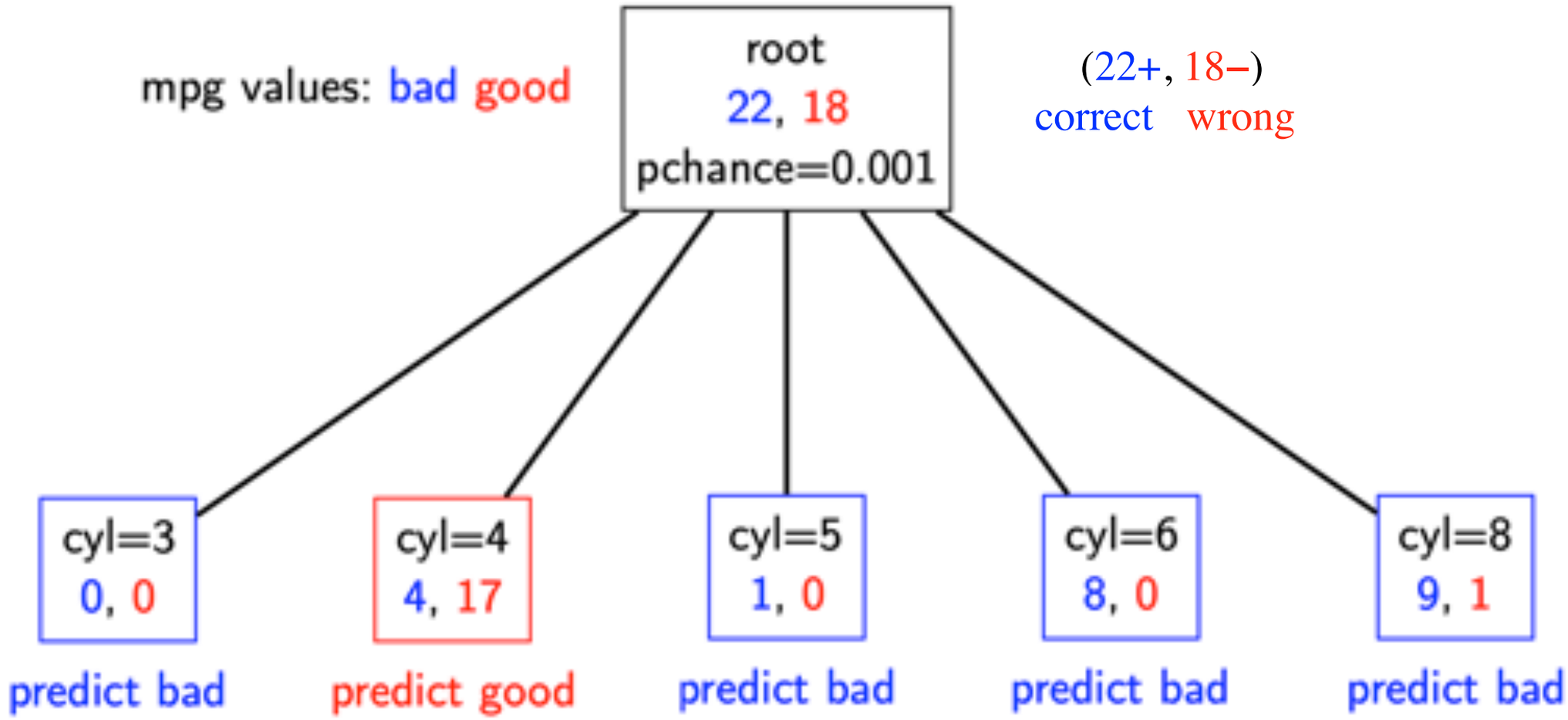
- **What is the simplest tree?**

Table: From the UCI repository

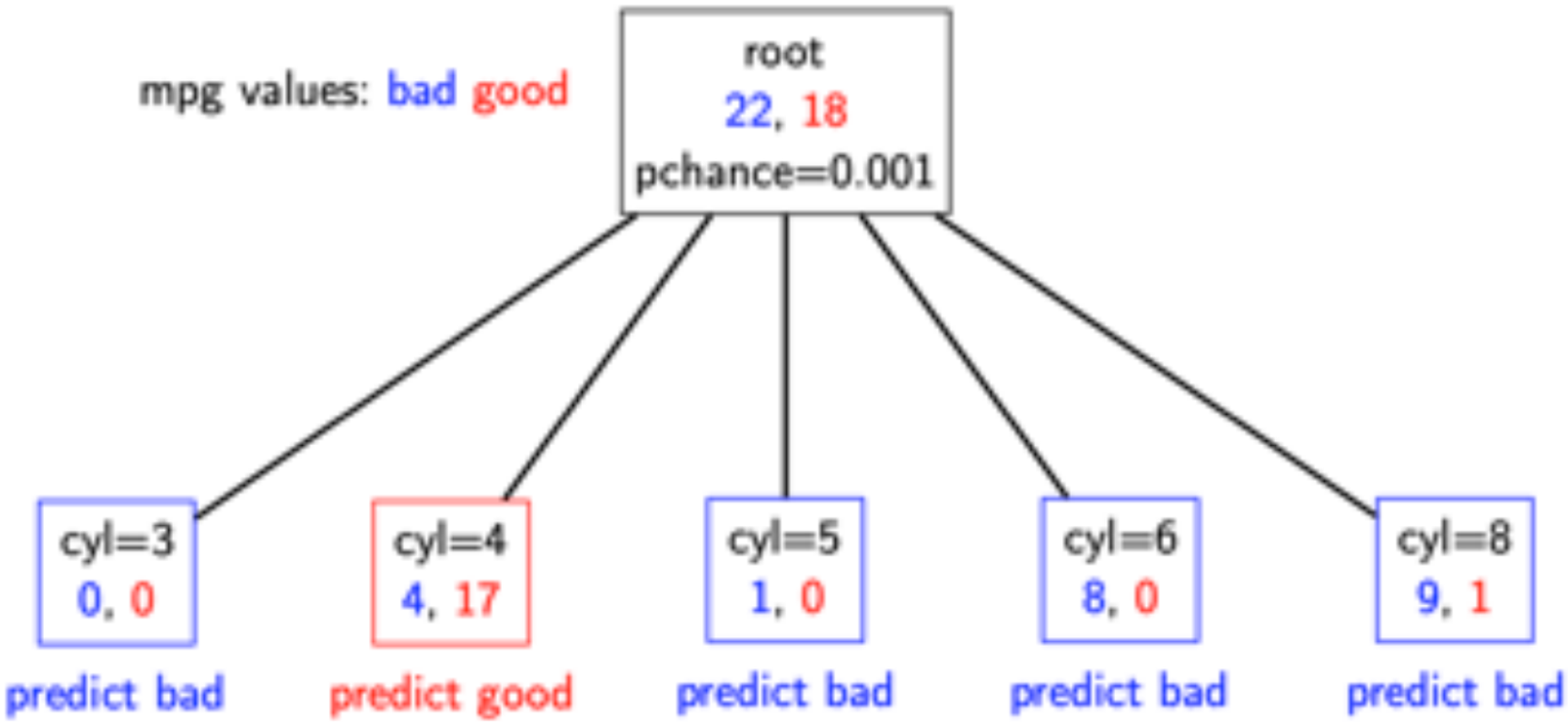
cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	mpg
4	low	low	low	high	75-78	asia	good
6	medium	medium	medium	medium	70-74	america	bad
8	high	high	high	low	70-74	america	bad
4	medium	medium	medium	low	75-78	europe	bad
...	...	...	...	...	...	...	...
4	low	medium	low	medium	75-78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

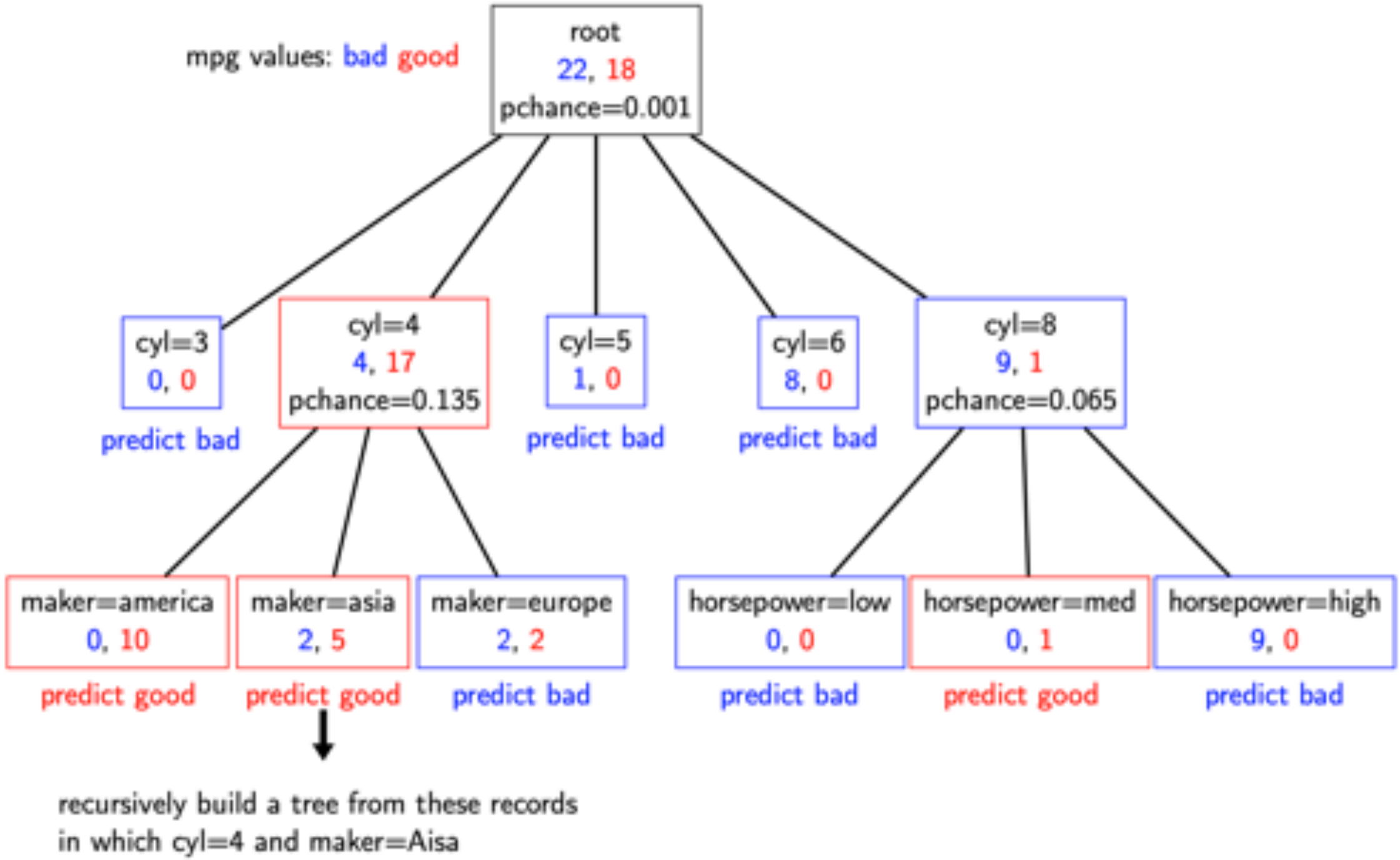


Recursive step

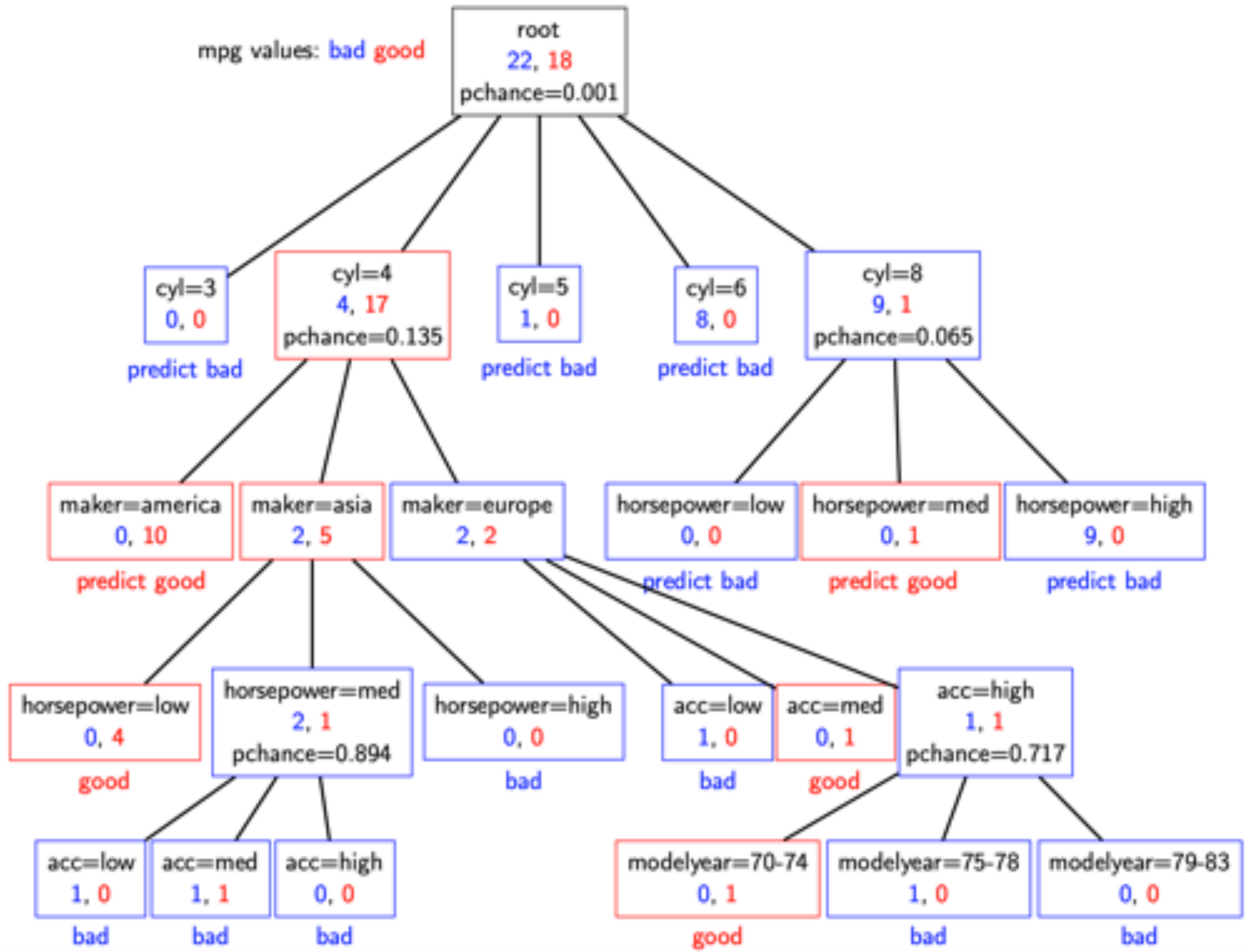


- Take the original dataset
- Partition it according to the values of the attribute we split on
- Build tree from these records (cyl=4, cyl=5, cyl=6, cyl=8)

Second Level of a Decision Tree

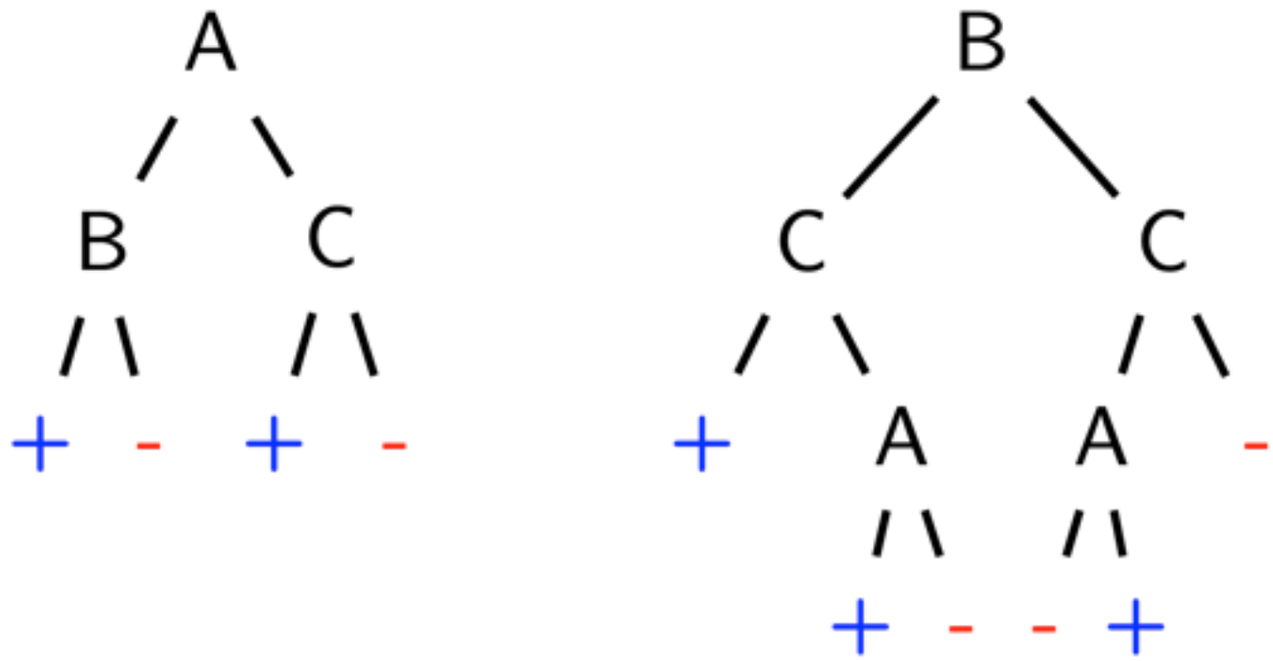


A Full Decision Tree



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)

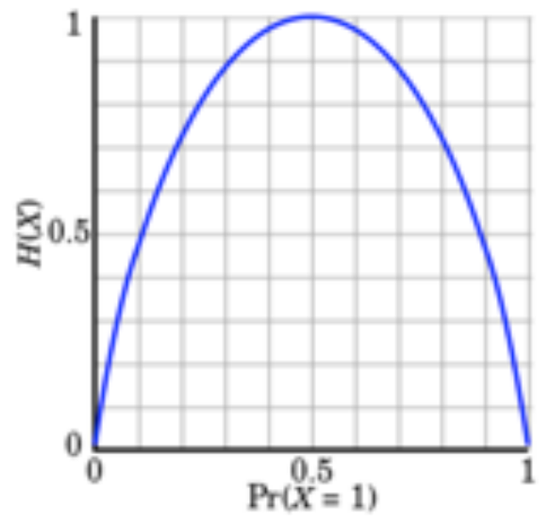


Figure: Entropy of a coin flip

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:

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

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

In this example:

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

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:

$$P(X_1 = T) = 4/6$$
$$P(X_1 = F) = 2/6$$
$$H(Y|X_1) = -4/6(1 \log 1 + 0 \log 0) - 2/6(1/2 \log 1/2 + 1/2 \log 1/2) = 2/6$$

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:

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

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

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

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

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

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right)$$

$$+ \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right)$$

$$+ \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$

$$= 0.694 \text{ bits.}$$

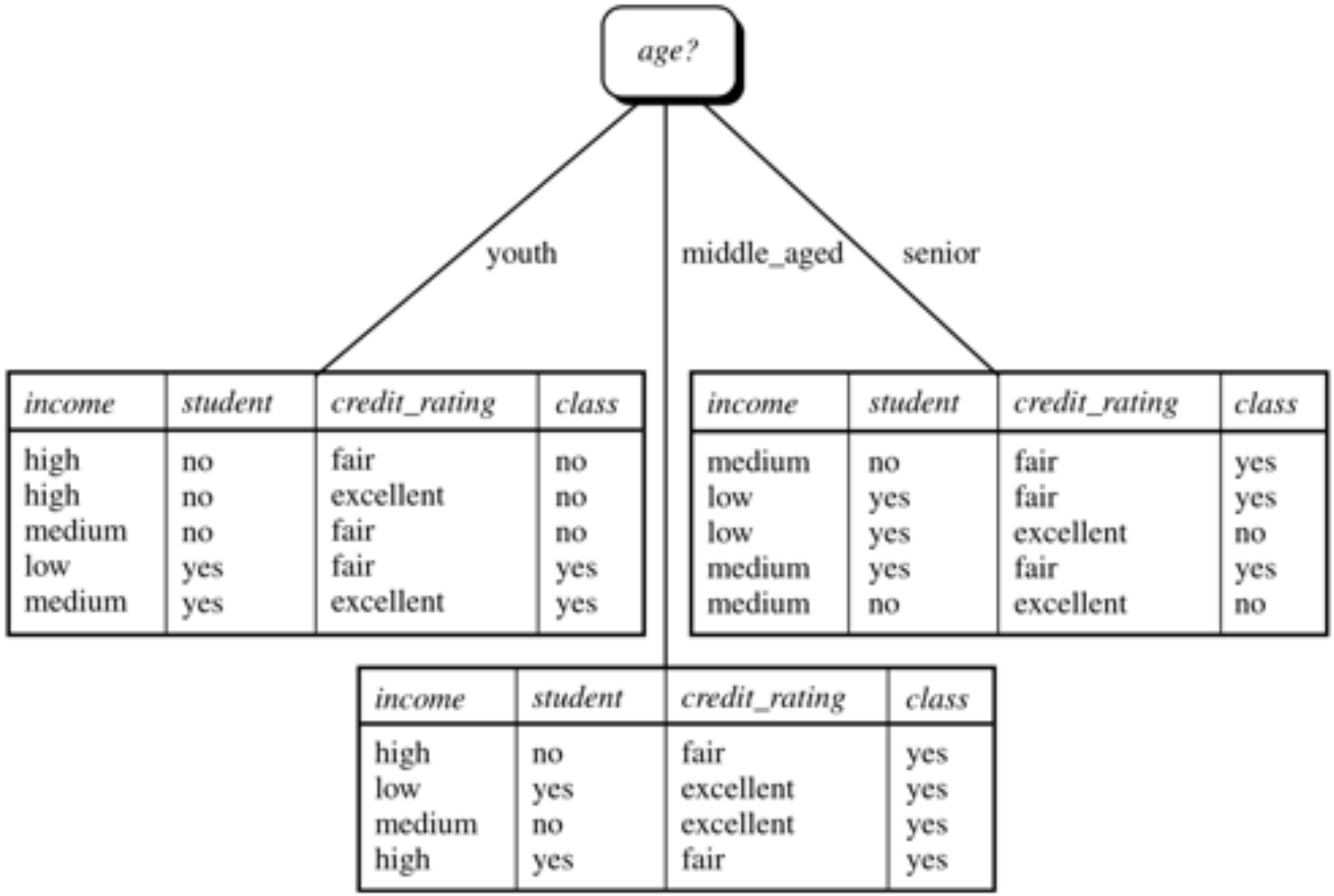
$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

$Gain(income) = 0.029$ bits

$Gain(student) = 0.151$ bits

$Gain(credit\ rating) = 0.048$ bits

Decision Tree

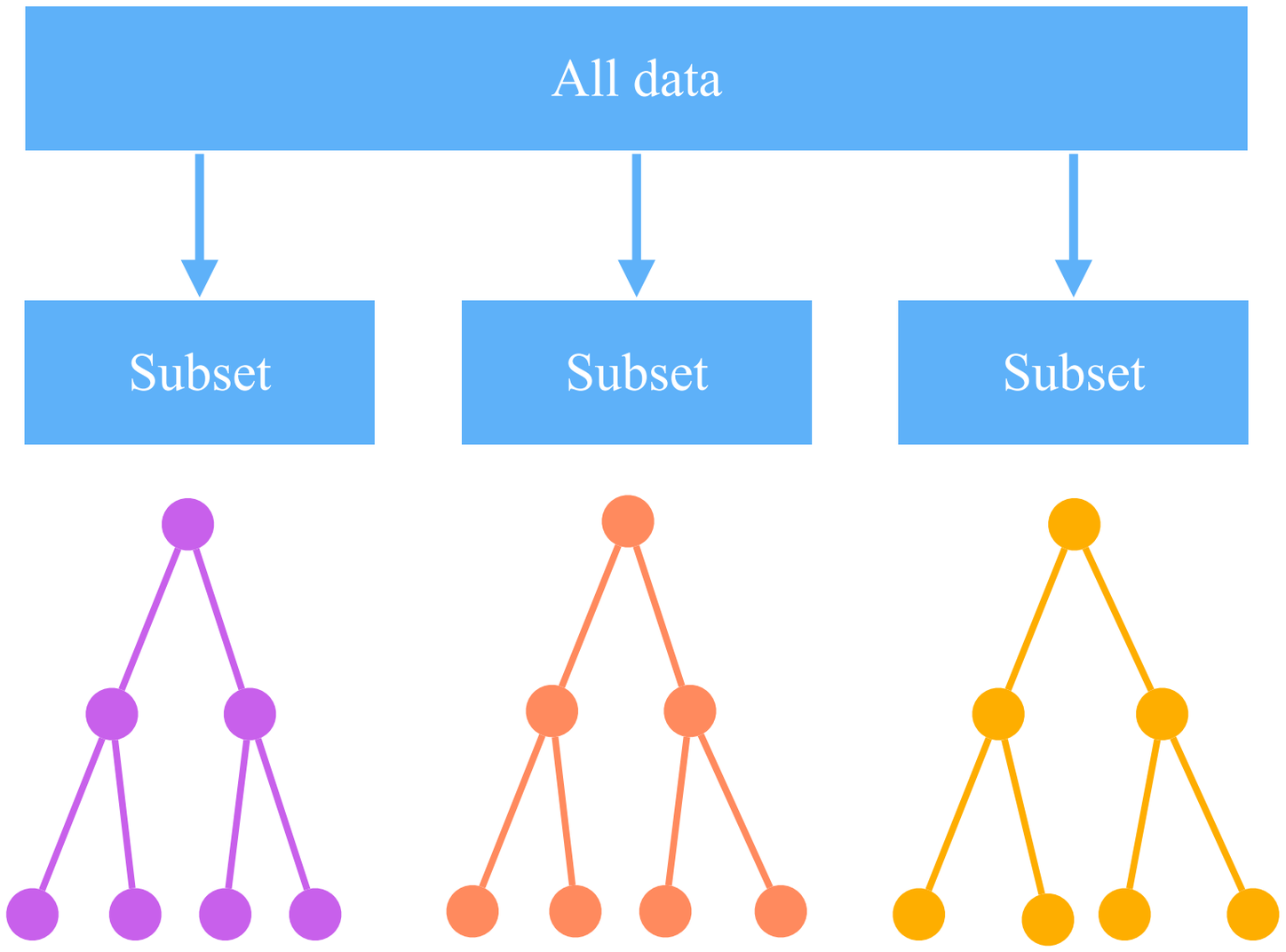


income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

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

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

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**