Supervised Learning for Predictive Analytics

Classification Algorithms

Dr. Unchalisa Taetragool

Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi

Outline

- Basic classification algorithms
  - K-nearest neighbors
  - Decision tree algorithms
  - Model cross-validation techniques
- Probabilistic classification algorithms
  - Logistic regression
  - Naives Bayes classifier

Predictive Modeling as Supervised Segmentation

- How can we segment the population into groups that differ from each other with respect to some quantity of interest?

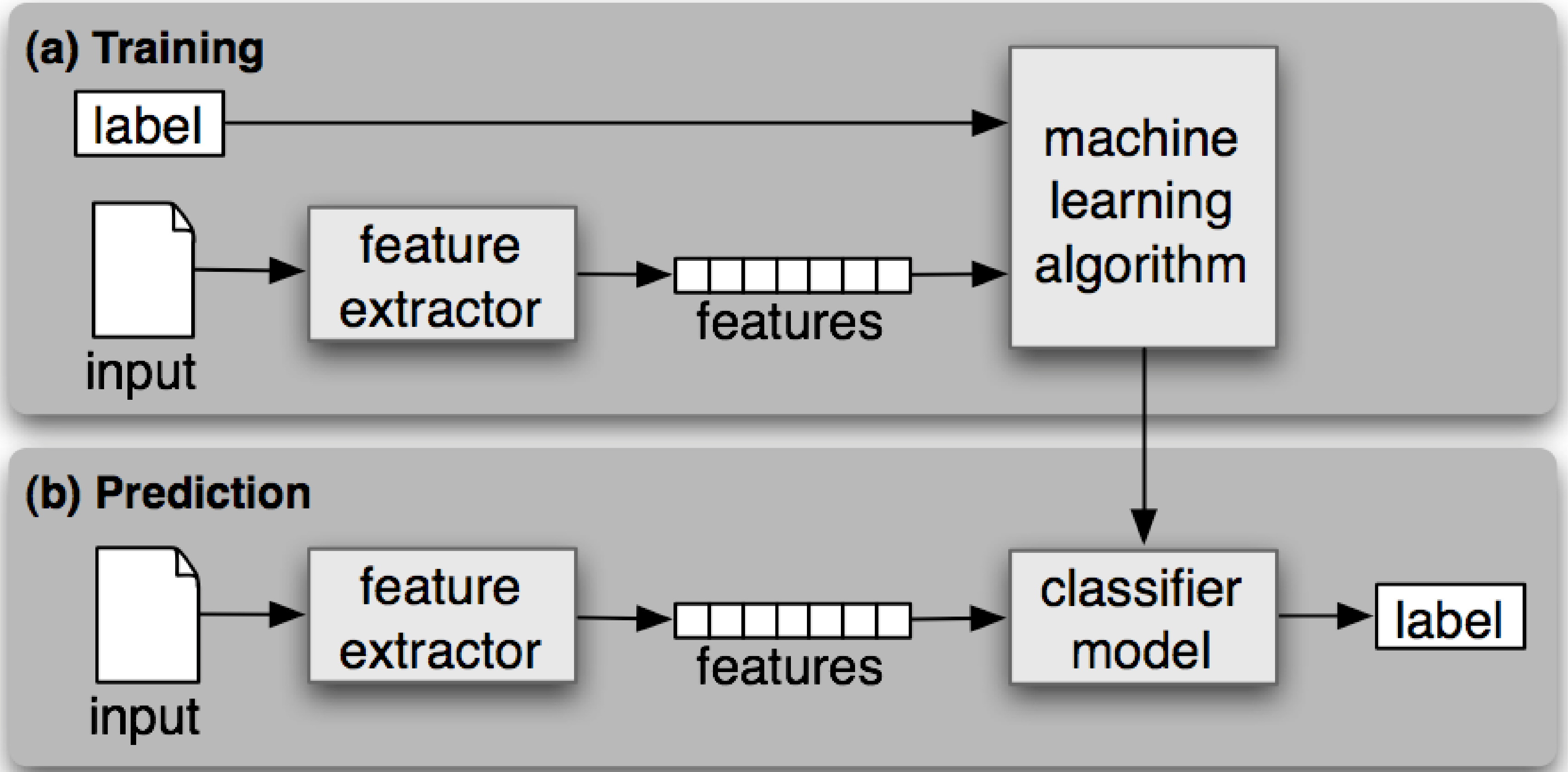
Quantity of interest
=
Things we would like to predict or estimate

Goals of Prediction

- To avoid, such as
  - which customers are likely to leave the company when their contracts expire.
  - Potential customers are likely to write-off (default)
  - Which web pages contain questionable contents
- To target, such as
  - Which consumers are likely to respond to an ads.
  - Which web pages are most appropriate for the search query

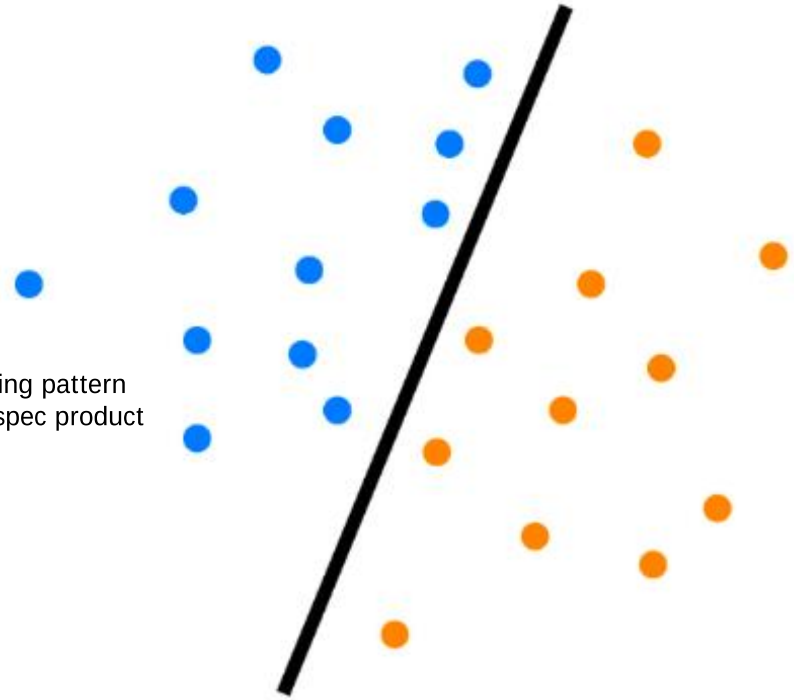
Classification model

there will always have a label because it's supervised learning.



Applications

- Churn prediction
- Targeted marketing
- Risk prediction
- Failure prediction what kind of operating pattern that makes the off-spec product
- Sentiment analysis
- Speech recognition
- Image recognition

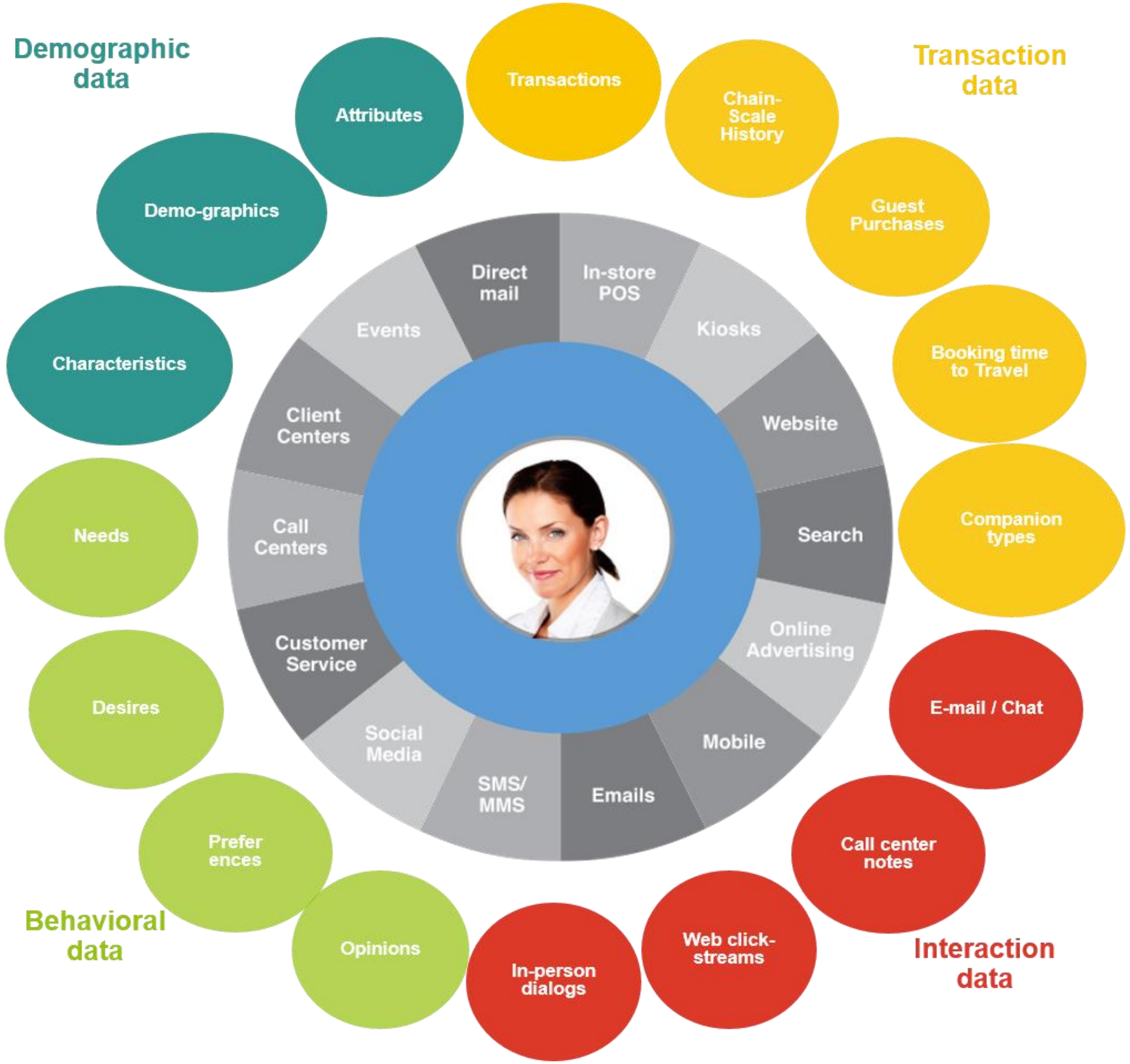


Training data

- Training data contains
  - Label the most importance part
  - Attribute data

- Usually they are historical records

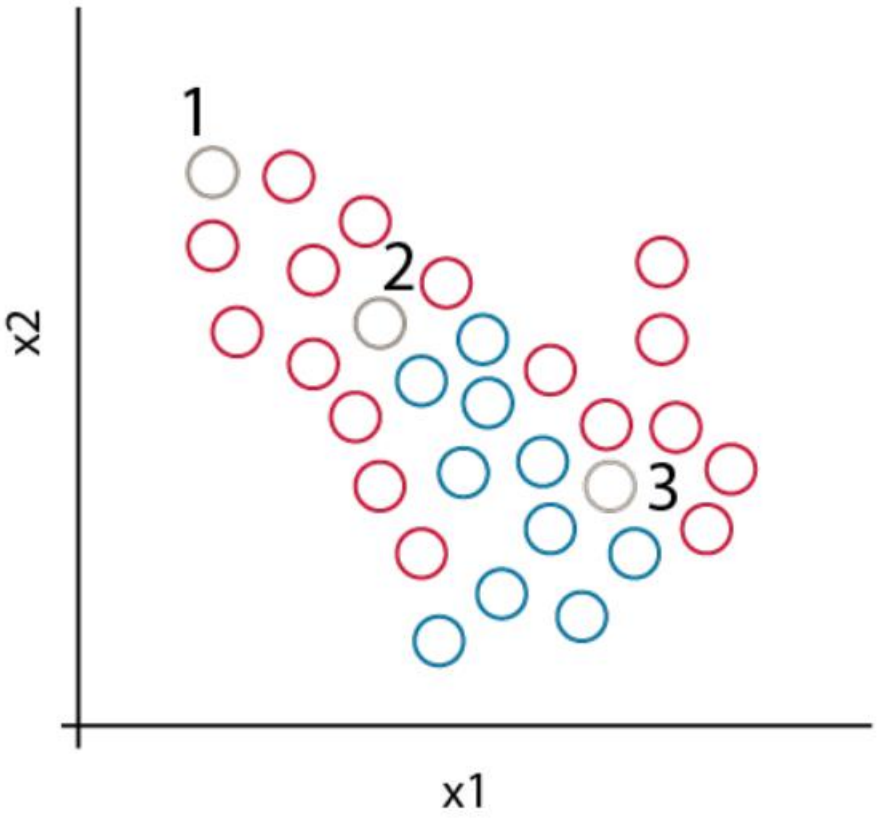
- Determining right labels is very crucial



K-NEAREST NEIGHBOR

Intuition Quiz

Would you classify point 1, 2, 3 as blue or red. Fill in the table.



Point	BLUE or RED
1	Red
2	Red?
3	Blue?

How Decision is Made

- Your source of knowledge is the similarity between two different data points. So you use similarity to make decisions such as classification and regression.
- You make decisions about one data point based on neighboring points.

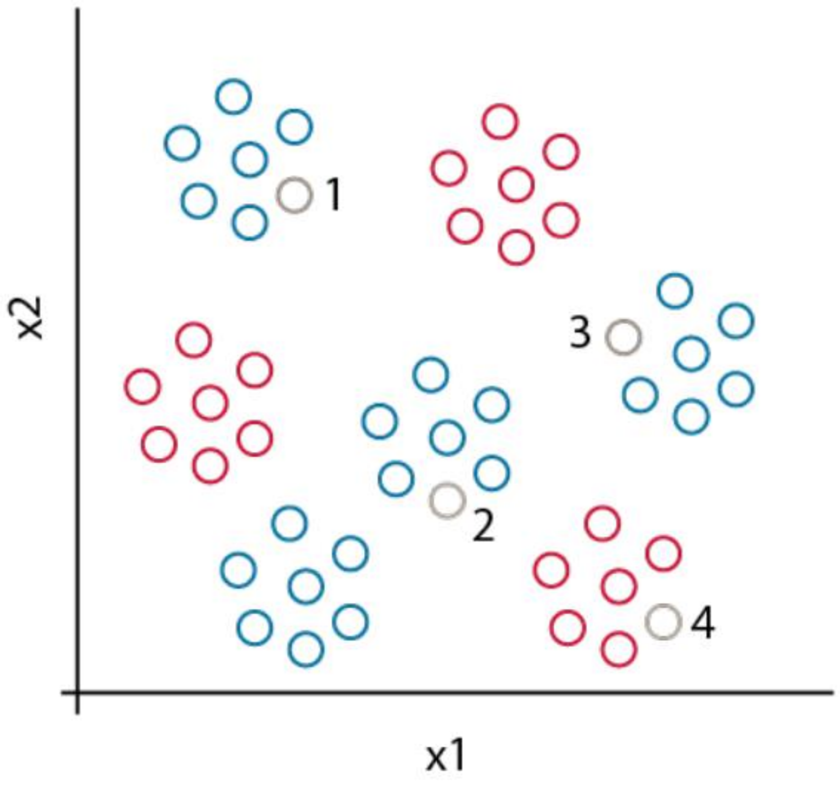
Instance-based Learning (IBL)

- Lazy algorithm: when you see your training set, you do nothing, just store them in the memory.
- When new sample comes you compare the new sample with the existing samples in the memory.
- Examples of algorithms in this family: nearest neighbor, kernel machines. eg, algorithms

IBL - Nearest Neighbor Methods

Nearest neighbor:

when you see a new data point (x'), locate the nearest data point (x) and predict the label of x' to be the same as label of x.



IBL - K-Nearest Neighbor Methods

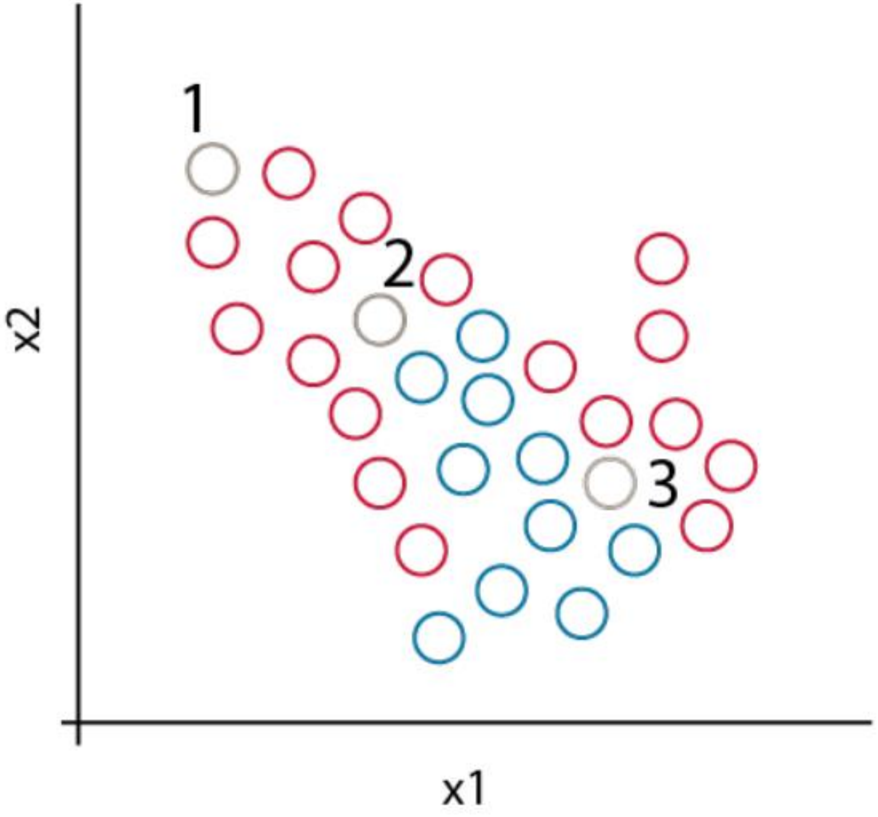
- **K-Nearest neighbor:** locate k nearest neighbors around x'.
  - For classification problem, let k neighbors vote for the right label of x'.
  - For regression problem, average the y values of all neighbors and predict that y as the label of x'.

  Is it kernel regression?

K = the number of interested points

IBL - K-Nearest Neighbor Quiz

Use K-Nearest Neighbor Rule to classify point 1, 2, and 3 with different values of k.



Point	k=1	k=2	k=3	k=4
1				
2				
3				

K-Nearest Neighbor

- Pros:
  - Training takes no time
  - Complex decision boundary is possible
  - Information is not lost
- Cons:
  - Query is slow (the more data the slower)
  - Storage space is huge
  - <u>Easily fooled by irrelevant attributes</u>

Distance and Similarity Metrics

- To determine whether two points are close, we use distance metrics.
- **Distance metrics** are the numerical value that tells you whether two points are close (low value) or far apart (high value).
- There are several ways to define distance metrics, such as Euclidean distance, minkowski distance.
- **Similarity metrics** are the numerical value that tells you whether two points are close (very similar - high value) or far apart (very dissimilar – low value).
- Distance and similarity metrics are important in many ML models such as ‘Support Vector Machine’, ‘K-Nearest Neighbor’, ‘K-Mean Clustering’

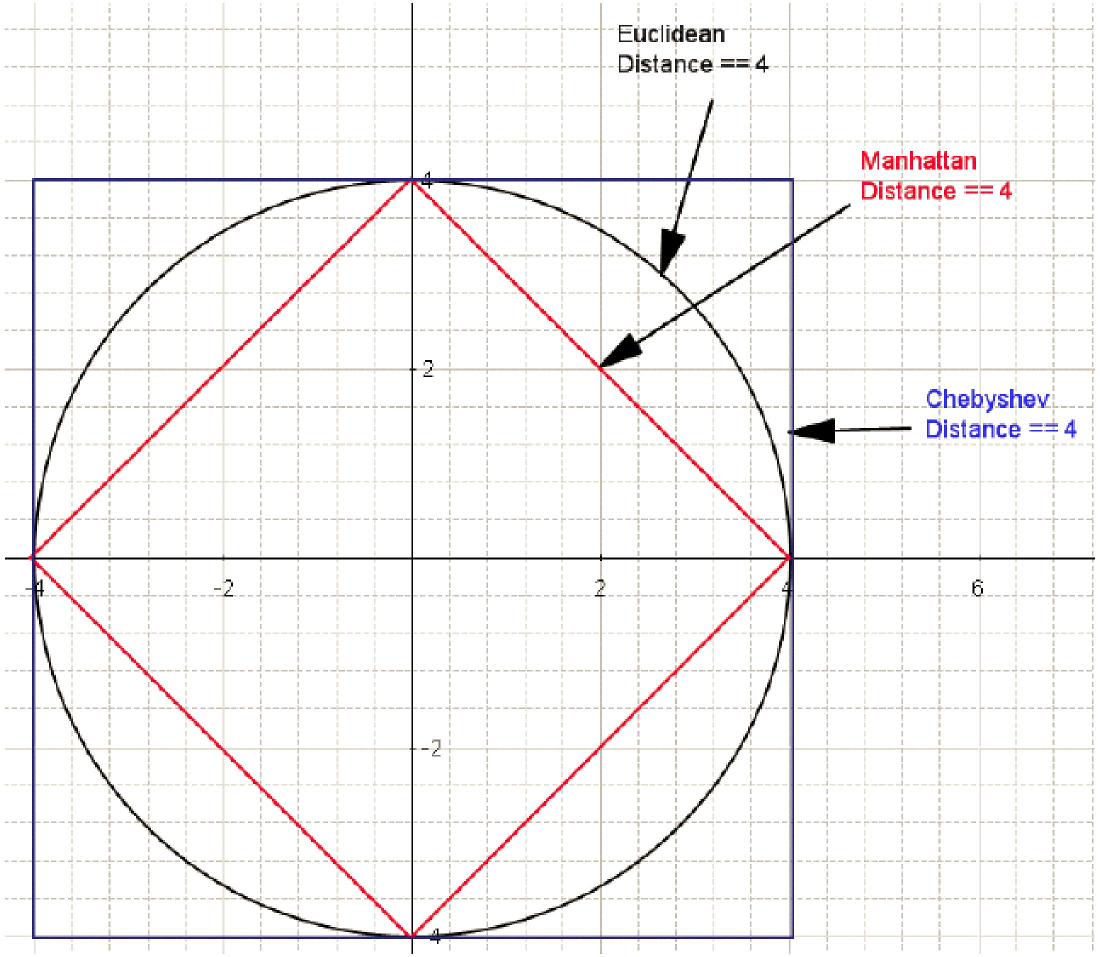
Distance Metrics for Real Value Features

- Euclidean Distance

  sqrt(sum((x - x’)^2))
- Manhattan Distance

  sum(|x - x’|)

ขึ้นกับปัญหาว่าเหมาะสมกับ distance แบบไหน



Distance Metrics for Boolean Features

- Jaccard Distance

Feature	Me	My Dad
Man Barber	F	T
Toyota	T	T
MK	T	T
Water Park	T	F
Temple	F	T
Bar	F	F

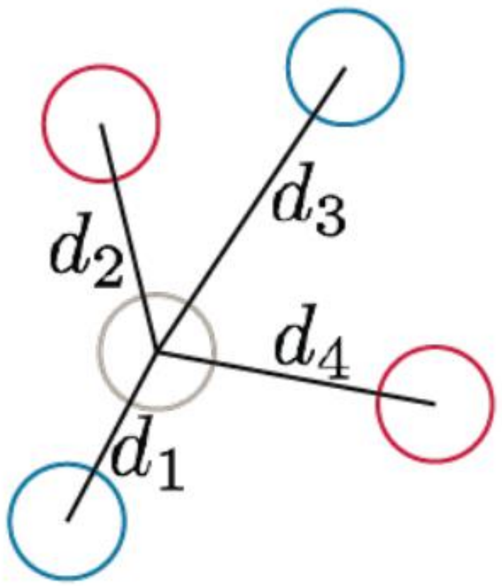
N=6
NTT=2
NTF=1
NFT=2 counting the combination
NFF=1
NNEQ : number of non-equal dimensions
NNEQ = NTF + NFT = 3
NNZ : number of nonzero dimensions
NNZ = NTF + NFT + NTT = 5

NNEQ / NNZ = 3/5 = 0.6

distance of boolean

Using Distances as Weights

- Neighbors who are closer to the target data point should get more say in the voting process.



$$y' = \frac{w_1 y_1 + w_2 y_2 + w_3 y_3 + w_4 y_4}{w_1 + w_2 + w_3 + w_4}$$

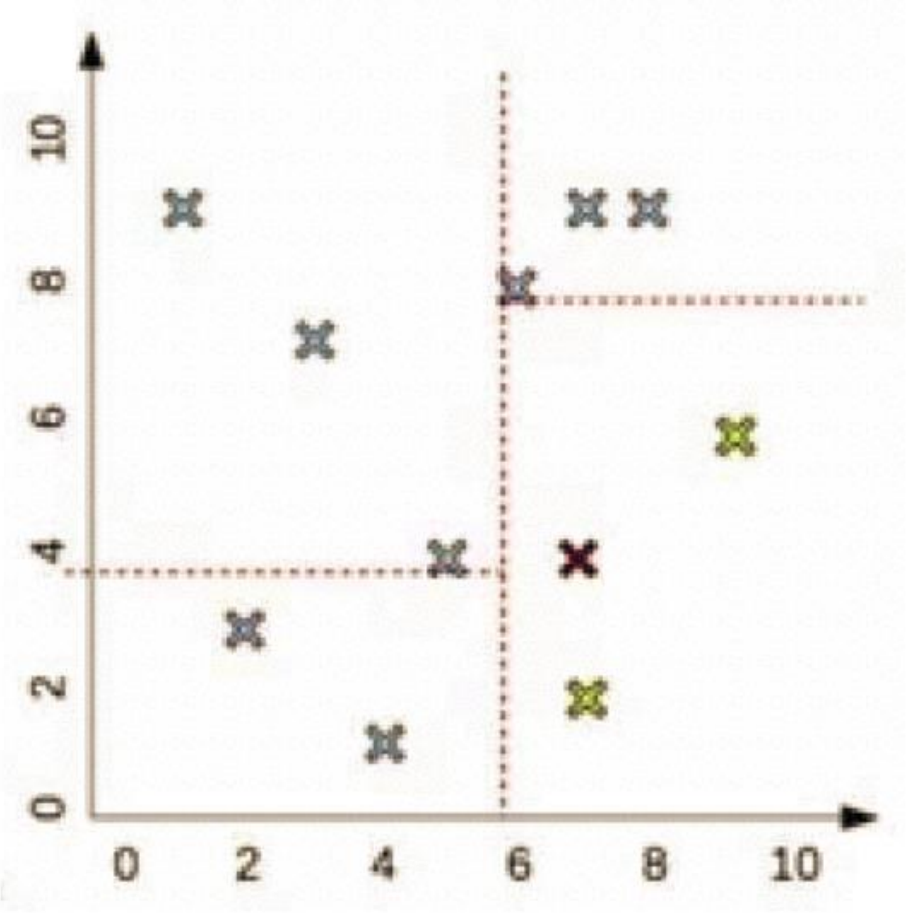
$$w_i = \frac{1}{d_i}$$

Searching for Nearest Neighbors

- Brute force : when a new sample x' appears, calculate the distance between x' and all other points. Consider points with lowest distances for voting.
- Brute force is slowest, but the most accurate.
- If your data is sparse, then brute force is the right way.
- To speed up the search, you can use KD Tree or Ball Tree.

K-D Tree

- Data = [(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)]
- Say we want to search for nearest neighbors of point (7,4)

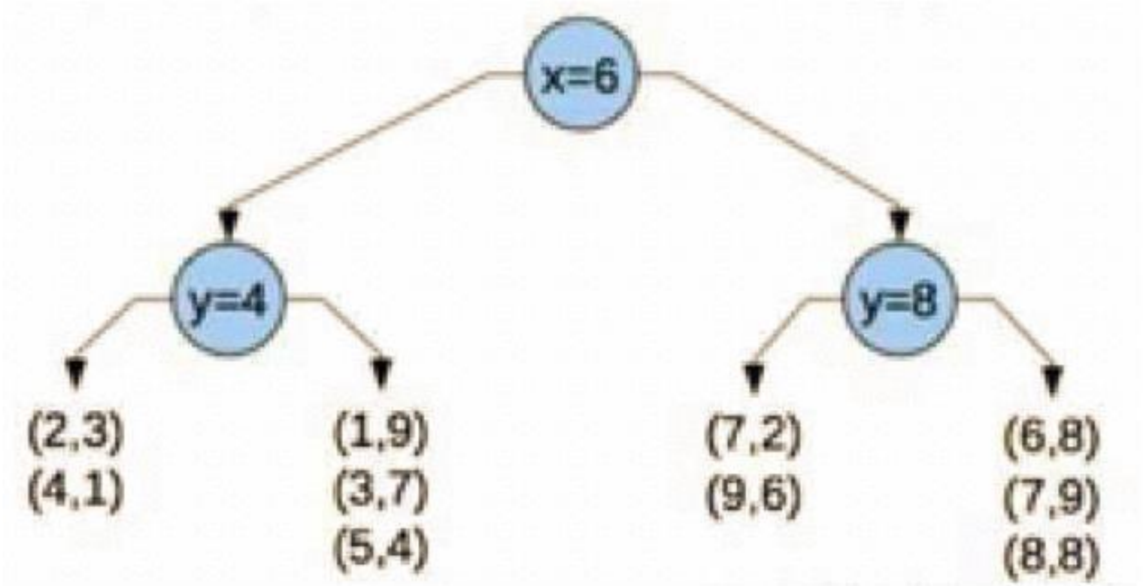


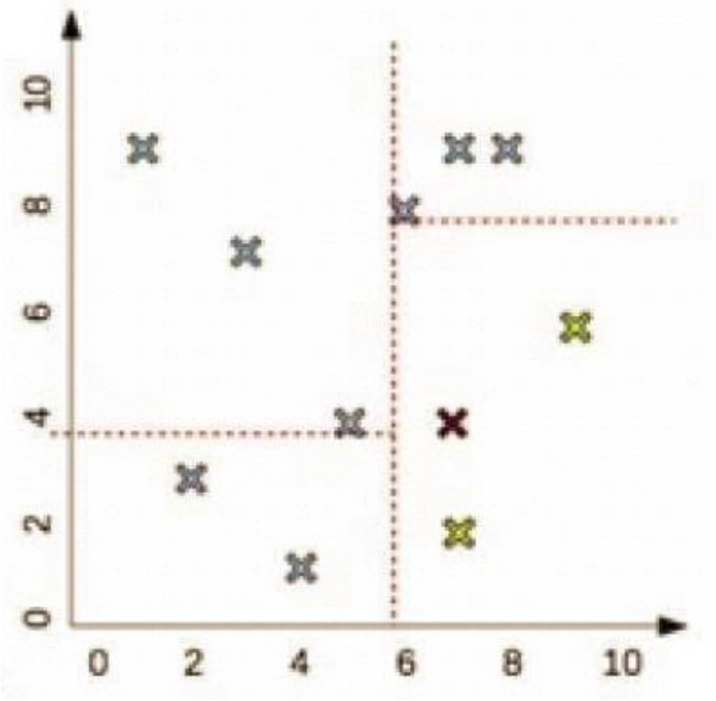
Credit: Victor Lavrenko

ตัวใกล้สุดควรอยู่ใน plane เดียวกันประหยัดเวลาไม่ต้องเทียบ query ใหม่กับทุกจุด

K-D Tree

- First, pick a random dimension (say x1) find median and split data. Repeat for other dimensions.

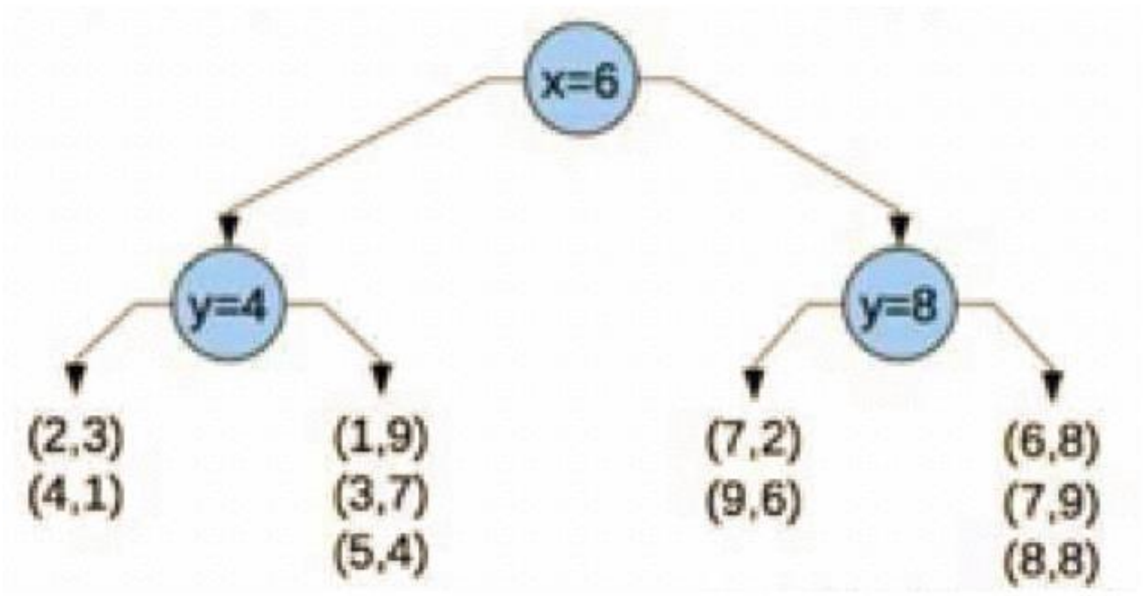


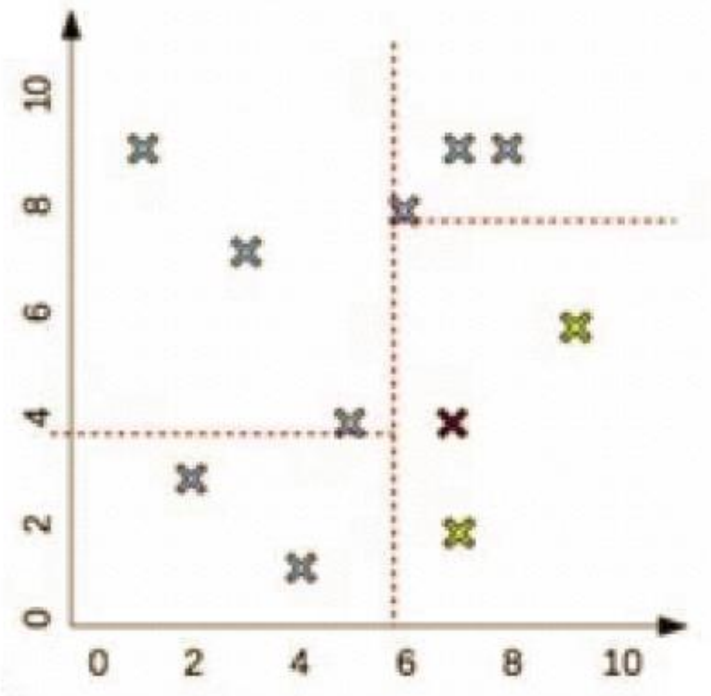


Credit: Victor Lavrenko

K-D Tree

- Find region that contains (7,4) search for neighbors only in that region.



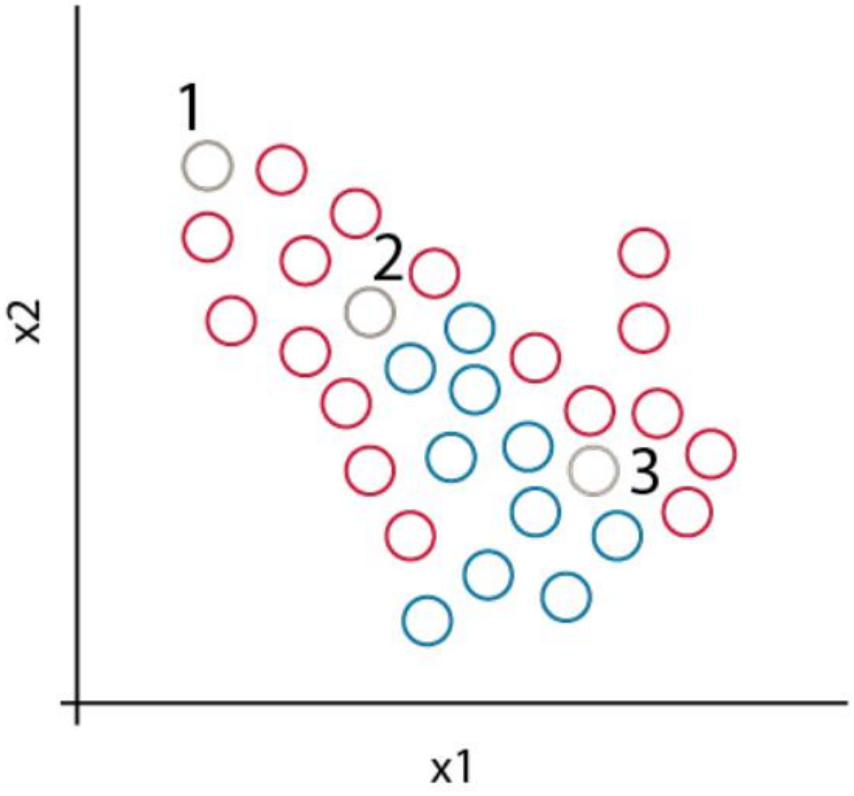


Credit: Victor Lavrenko

How to Avoid Overfitting

k is a hyper parameter

- Use k as an overfitting control.
  - If k is one, you are very susceptible to noise (overfitting).
  - If k is high, you are averaging over really large regions, you lose resolution (under fitting).



How to Avoid Overfitting

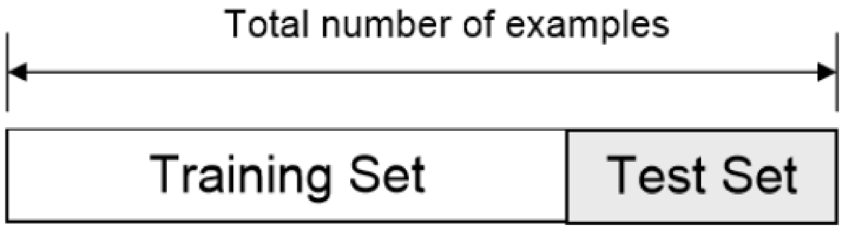
- Remove noisy instances prior to using nearest neighbor algorithm. Remove x if all nearest neighbors of x are in the opposite class.
- Form prototypes. If you observe lots and lots of very similar samples, lump them into a prototype by finding an average over all dimensions.

K-NN: Lab

Do I need to scale the data first?

Holdout method

- Split dataset into two groups
  - Training set: used to train the classifier
  - Test set: used to estimate the error rate of the trained classifier



- Drawback
  - In problems where we have a sparse dataset we may not be able to afford the "luxury" of setting aside a portion of the dataset for testing
  - Since it is a single train-and-test experiment, the holdout estimate of error rate will be misleading if we happen to get an "unfortunate" split

Python: Holdout

- We can now quickly sample a training set while holding out 40% of the data for testing (evaluating) our classifier:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(trainData_X, trainData_Y, test_size = 0.4)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
# Train and test the model
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

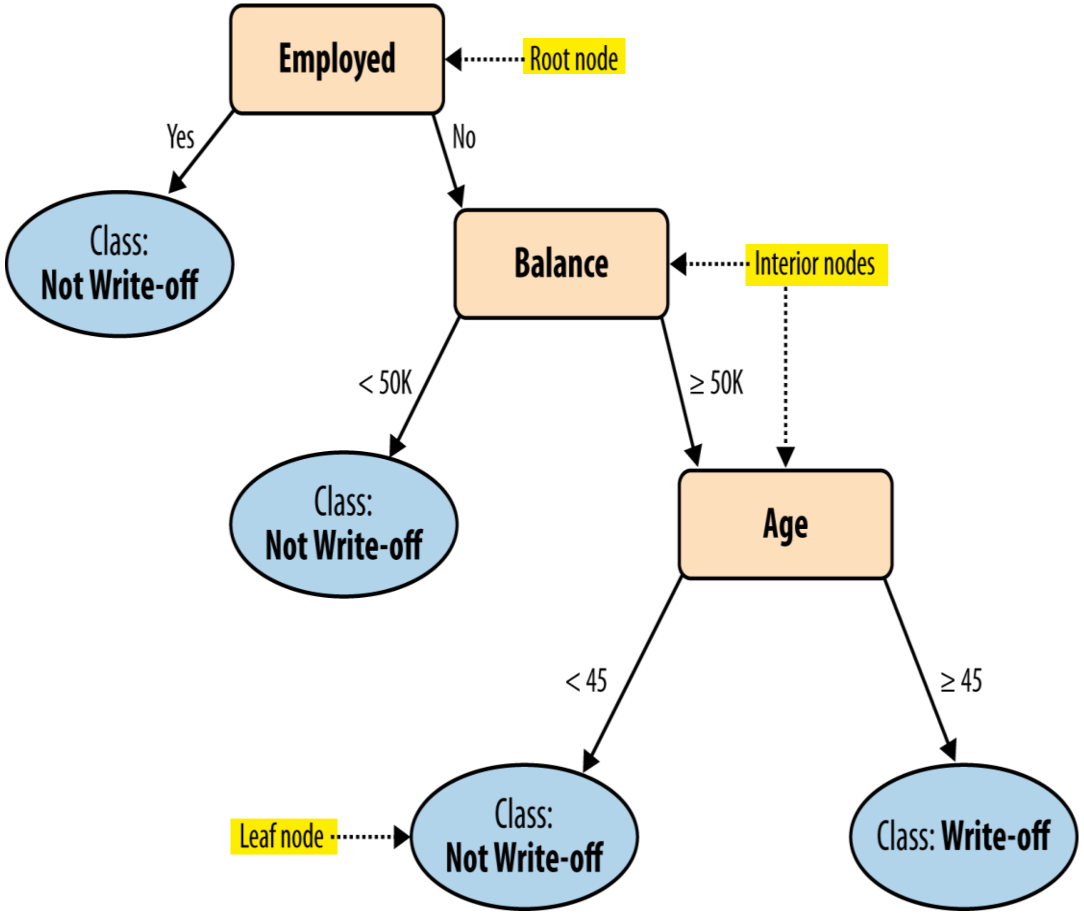
DECISION TREE

rule based method

Basic classifier: decision tree

- A tree consists of nodes: interior and terminal
- Interior node contains a test of an attribute
- Terminal node is a segment

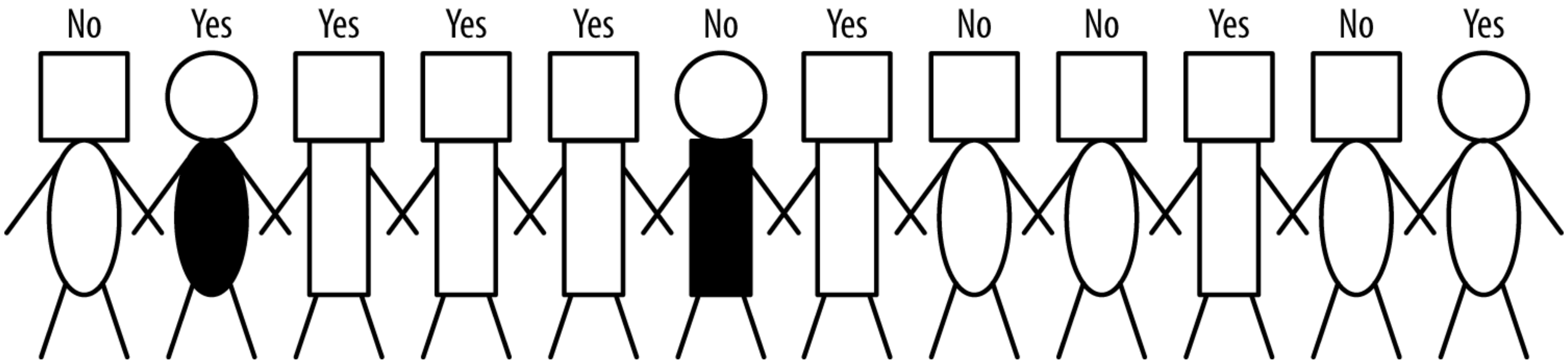
the main focusing is to select the purity variable as much as possible



Which attribute should be used to segment?

- **Fundamental concept**: how can we judge whether a variable contains important information about the target variable? How much?
- **Aims**: automatic selection, ranking
- We will start with considering the selection of the single most informative attribute
- In our example, what variable gives us the most information: Being professional? Age? Place of resident? Income?

Example: Write-off



- Attributes
  - head-shape: square, circular
  - body-shape: rectangular, oval
  - body-color: black, white
- Target variable
  - write-off: Yes, No

Which attribute should be best to segment these people into groups, in a way to distinguish write-off from non-write-off?

Purity

- Technically, we would like the group to be as pure as possible.
- Pure means homogeneous with respect to the target variable
- If some member in the group has a different target then the group is impure
- Comparing
  - G1 = {Y, Y, Y, Y}
  - G2 = {Y, N, N, Y}

In real data, however, we rarely find pure segments.

pure group is the one that has only 1 label (eg. G1)

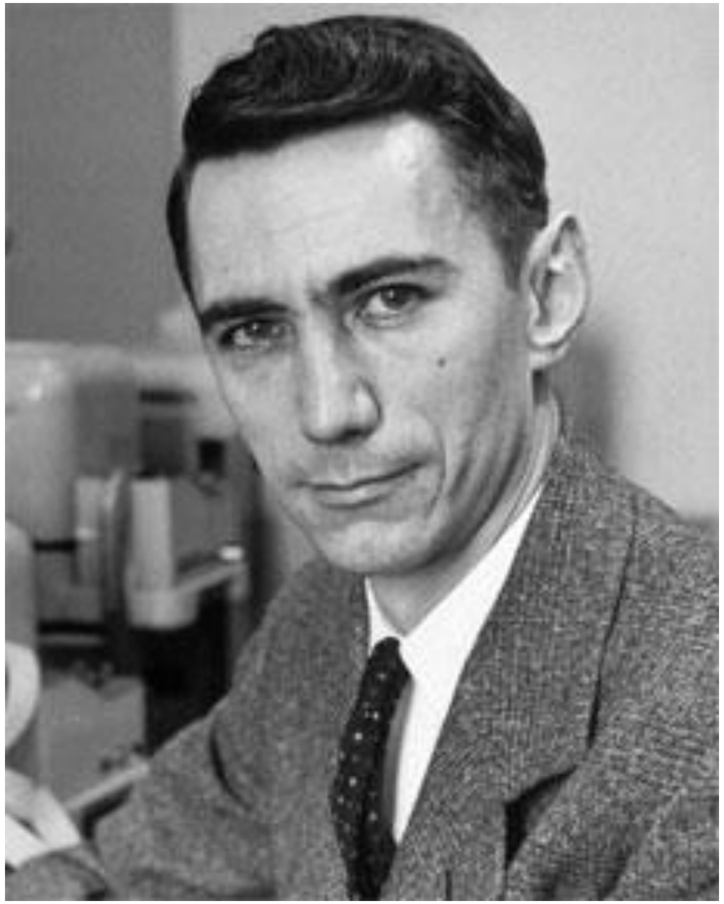
Complications

- Attributes rarely split a group perfectly. Even if one may be pure, the others may not.
- Not all attributes are binary. How do we compare them?
- Some attributes are numeric values. How should we create supervised segmentation using numeric attributes?

metric for decision the purity of the data

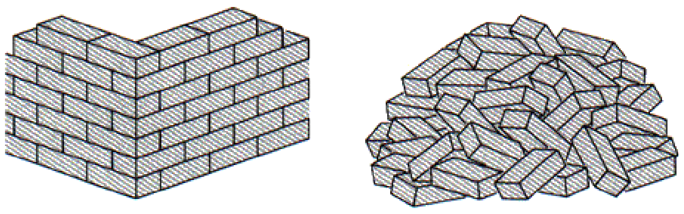
Information Gain and Entropy

- The most common splitting criterion is called information gain, and it is based on a purity measure called entropy
- Both concepts were invented by the pioneer in information theory, Claude Shannon (1948).



Entropy

- Entropy is a measure of disorder that can be applied to a set, such as one of our individual segments
- Disorder corresponds to how mixed (impure) the segment is with respect to the target
- For example, a mixed up segment with lots of write-offs and lots off non-write-offs would have high entropy



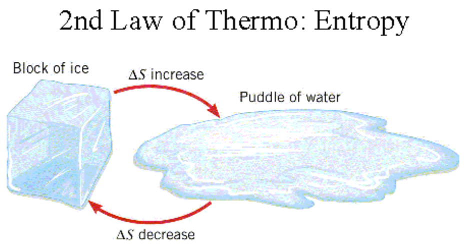
(a) (b)

Entropy Formula

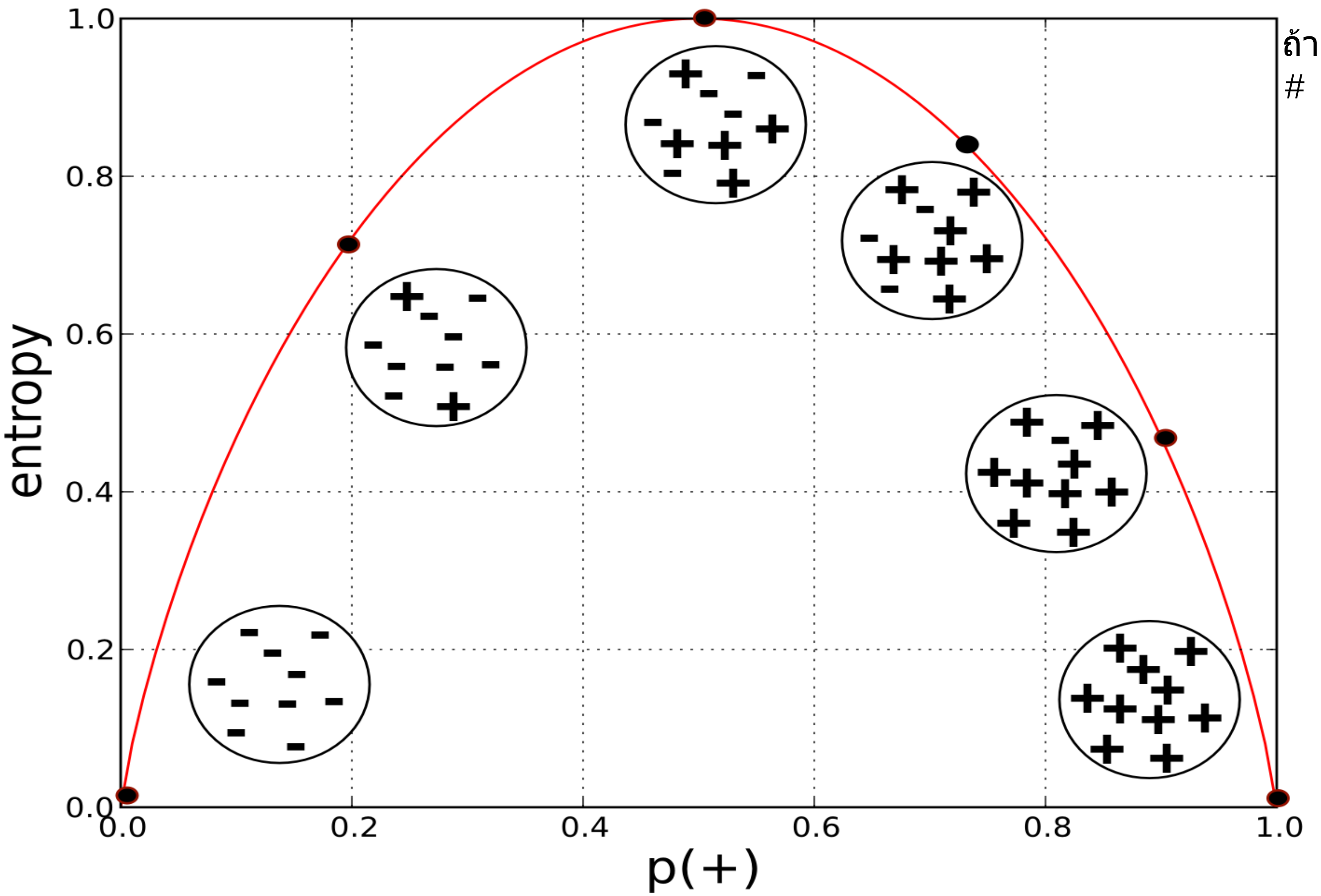
- More technically, entropy is defined as

$$entropy = -p_1\log(p_1) - p_2\log(p_2) - \ldots$$

- This is based on Gibbs entropy in thermodynamics
- Each p_i is the probability (the relative percentage) of property i (e.g. write-offs/non-write-offs) of the target.



Entropy of a two-class set



ถ้า data มีการผสมกันมาก
label X == # label Y

Example

- Consider a set S of 10 people with 7 non-write-off and 3 write-off classes. So:

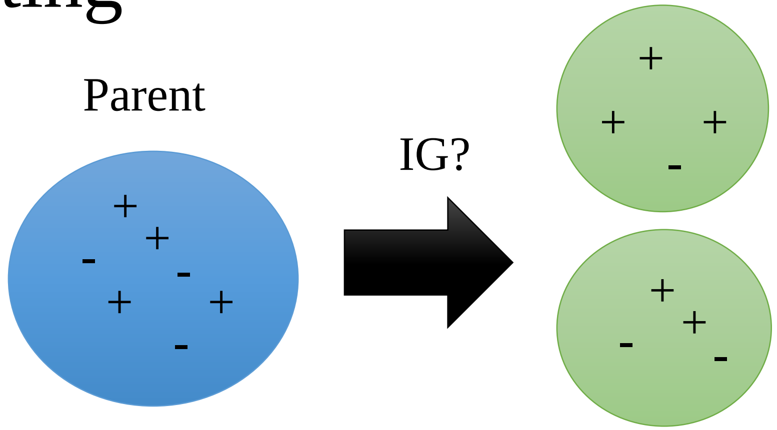
$$p(\text{non-write-off}) = 7/10 = 0.7$$

$$p(\text{write-off}) = 3/10 = 0.3$$

$$\begin{aligned} \text{entropy} &= -[0.7 \times \log_2(0.7) + 0.3 \times \log_2(0.3)] \\ &\approx -[0.7 \times -0.51 + 0.3 \times -1.74] \\ &\approx 0.88 \end{aligned}$$

Information Gain

- Entropy is only part of the story. We would like to measure how informative an attribute is with respect to target: how much gain in information it gives us about the target?
- Information gain (IG) measure how much attribute improves (decreases) entropy over the whole segmentation it creates.
- In our context, IG measures the change in entropy due to further splitting



เลือก IG ที่มากที่สุดมาเป็น group node

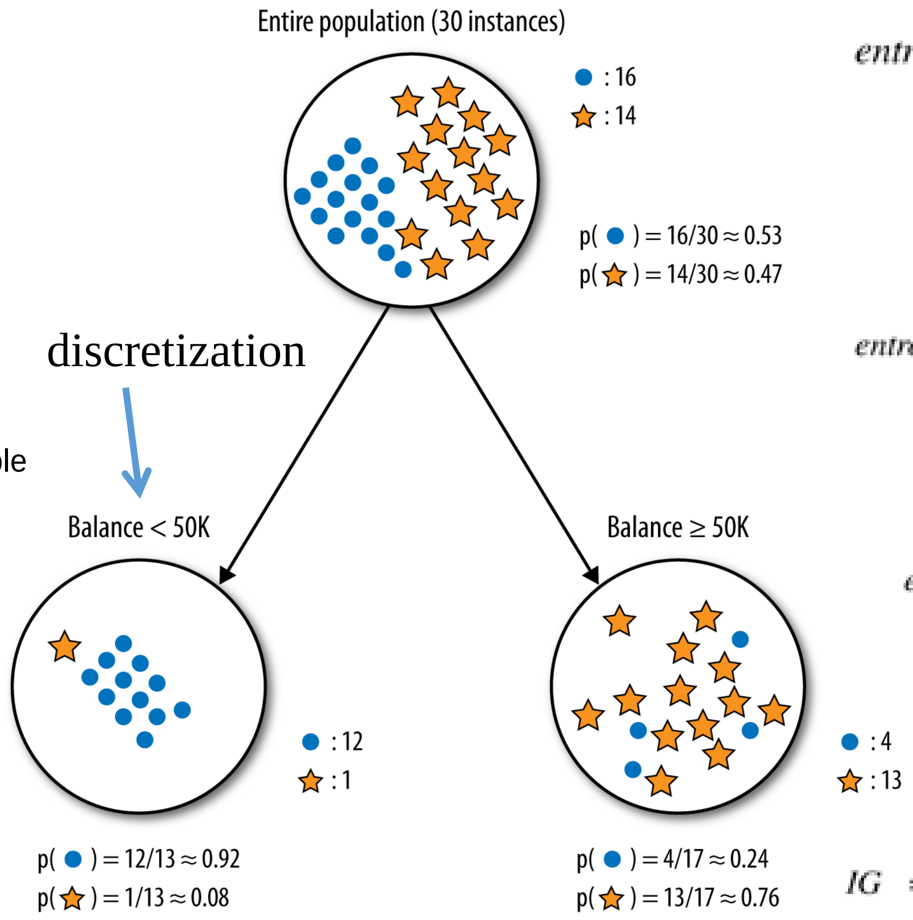
Children

Information Gain Formula

$$\mathrm{IG}(\mathit{parent}, \mathit{children}) = \mathrm{entropy}(\mathit{parent}) - [p(c_1)\times\mathrm{entropy}(c_1) + p(c_2)\times\mathrm{entropy}(c_2) + \ldots]$$

- The entropy for each child (c_i) is weighted by the proportion of instances belonging to that child, $p(c_i)$.

Example 1



discretization

for checking
balance variable

$$
\begin{aligned}
\mathit{entropy}(\mathit{parent}) &= -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)] \\
&\approx -[0.53 \times -0.9 + 0.47 \times -1.1] \\
&\approx 0.99 \quad (\text{very impure})
\end{aligned}
$$

Entropy of the left child is

$$
\begin{aligned}
\mathit{entropy}(\mathit{Balance} < 50K) &= -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)] \\
&\approx -[0.92 \times (-0.12) + 0.08 \times (-3.7)] \\
&\approx 0.39
\end{aligned}
$$

Entropy of the right child is

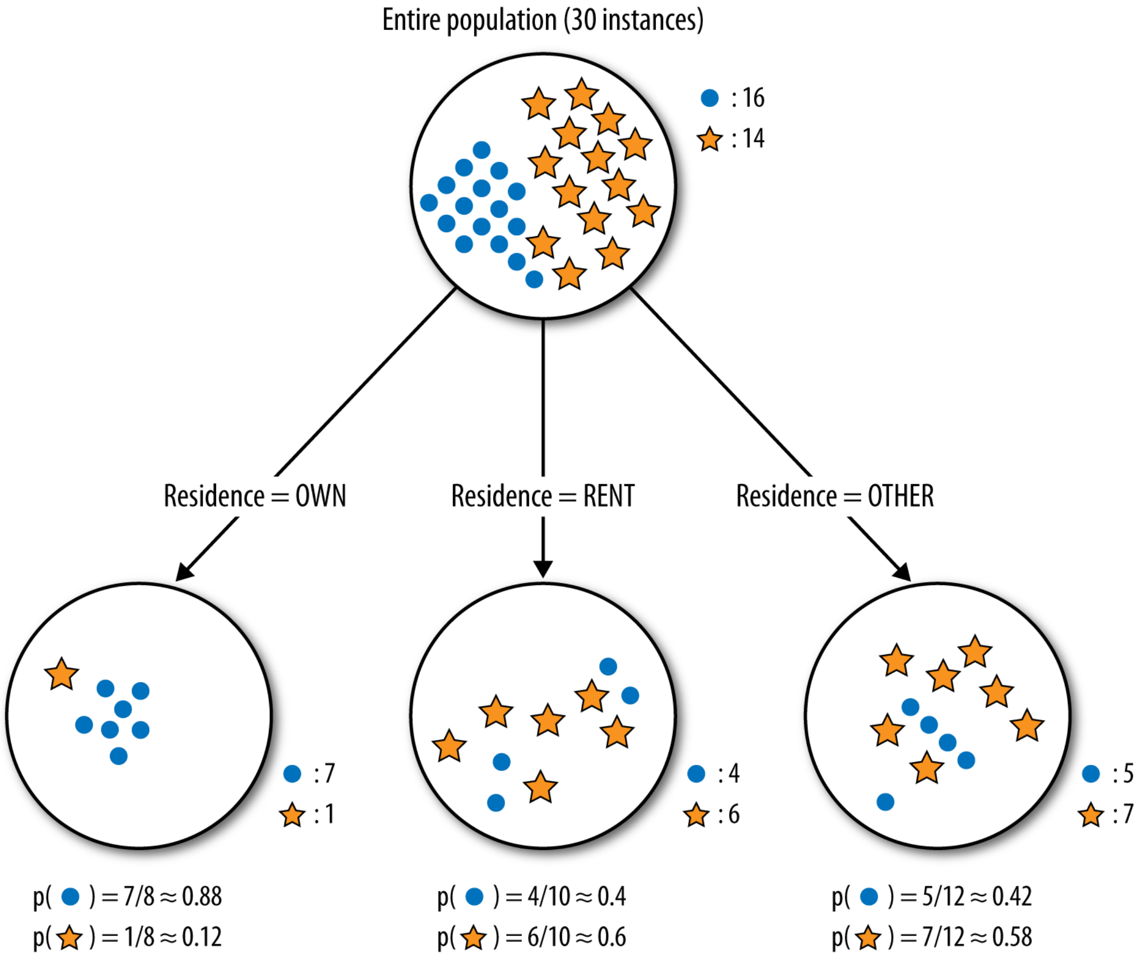
$$
\begin{aligned}
\mathit{entropy}(\mathit{Balance} \geq 50K) &= -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)] \\
&\approx -[0.24 \times (-2.1) + 0.76 \times (-0.39)] \\
&\approx 0.79
\end{aligned}
$$

Information gain is

$$
\begin{aligned}
IG &= \mathit{entropy}(\mathit{parent}) - [p(\text{Balance} < 50\text{K}) \times \mathit{entropy}(\text{Balance} < 50\text{K}) \\
&\qquad + p(\text{Balance} \geq 50\text{K}) \times \mathit{entropy}(\text{Balance} \geq 50\text{K})] \\
&\approx 0.99 - [0.43 \times 0.39 + 0.57 \times 0.79] \\
&\approx 0.37
\end{aligned}
$$

0.43 = 13/30; 0.57 = 17/30

Example 2



Calculations are omitted

$entropy(parent) \approx 0.99$

$entropy(\text{Residence=OWN}) \approx 0.54$

$entropy(\text{Residence=RENT}) \approx 0.97$

$entropy(\text{Residence=OTHER}) \approx 0.98$

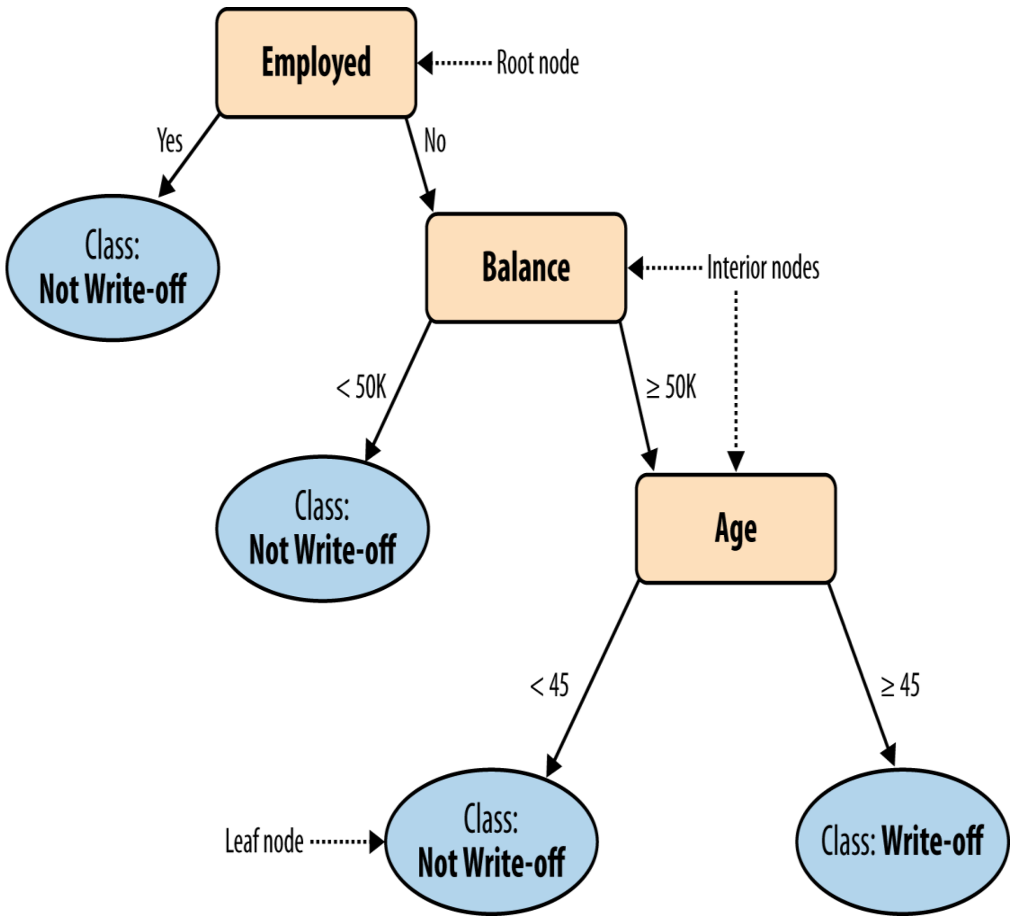
$\text{IG} \approx 0.13$

let us know which variable we should use

Residence variable is less informative than Balance.

Decision Tree

- A tree consists of nodes: interior and terminal
- Interior node contains a test of an attribute
- Terminal node is a segment

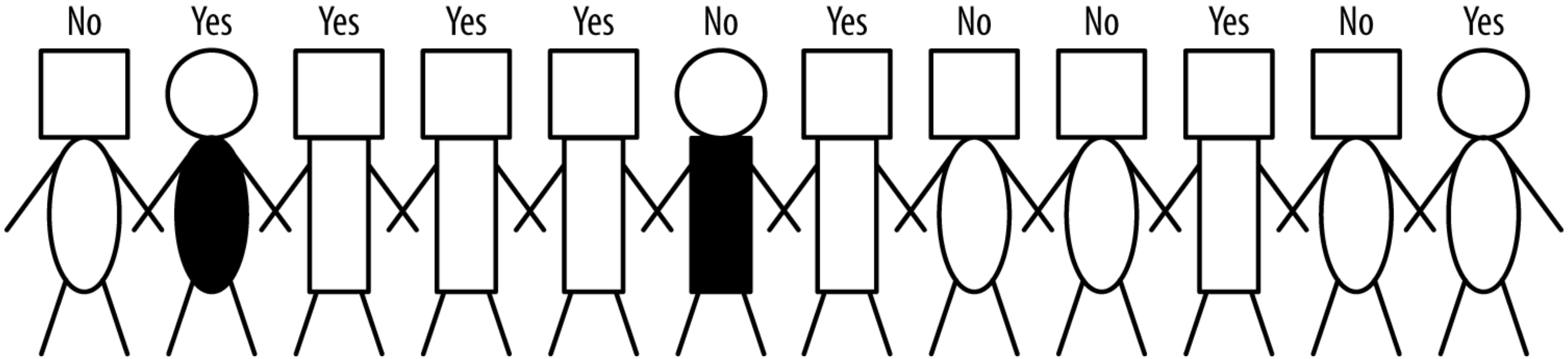


<Claudio, 115000, 40, No>?

Tree Induction

- How do we create a decision tree from data?
- Tree induction takes a divide-and-conquer approach,
  1. starting with the whole dataset
  2. applying variable selection to create subgroups
  3. Recursively repeating step 2 for each subgroup
- Stopping criteria:
  - When the leaf is pure, i.e. the variance of Y is small
  - When the number of samples in the leaf is too small
- We will illustrate this using the write-off example

Example



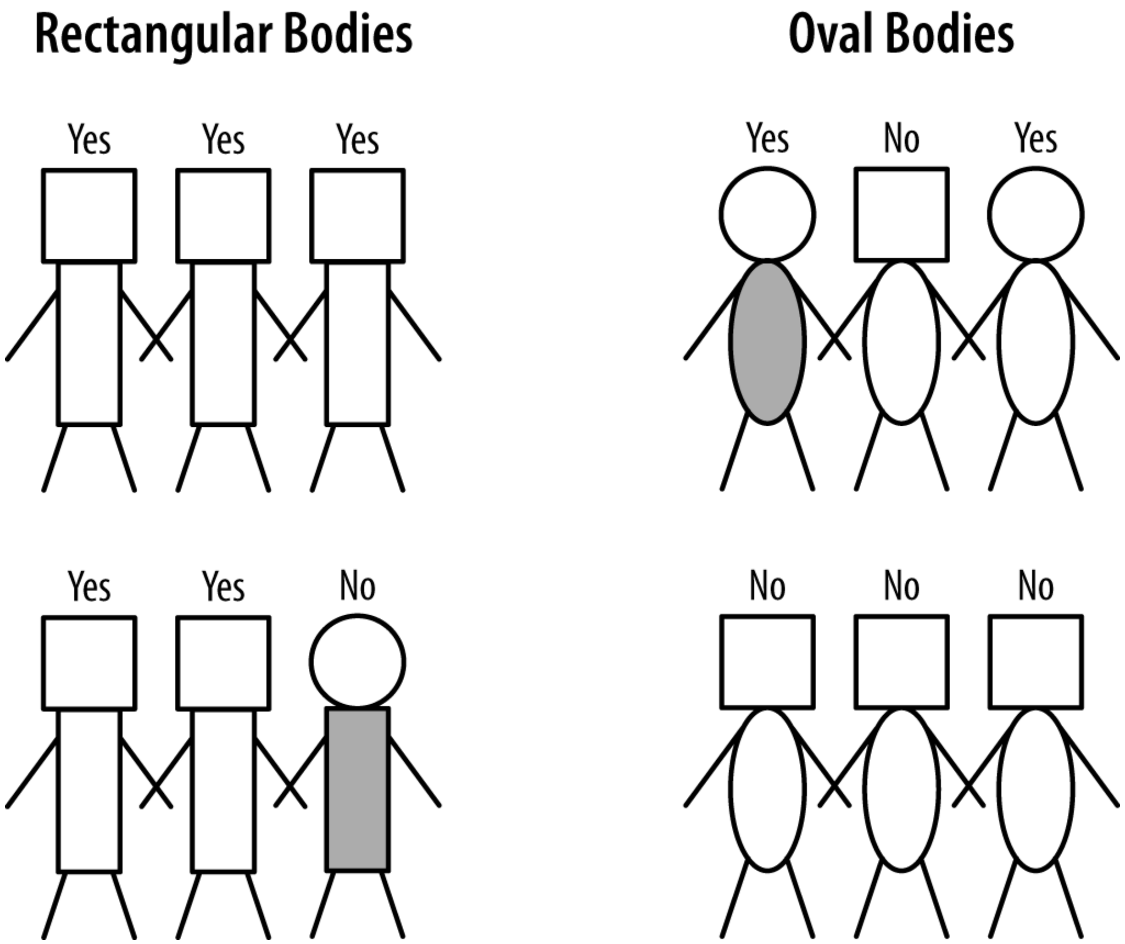
Attributes

- head-shape: square, circular
- body-shape: rectangular, oval
- body-color: gray, white

Target variable

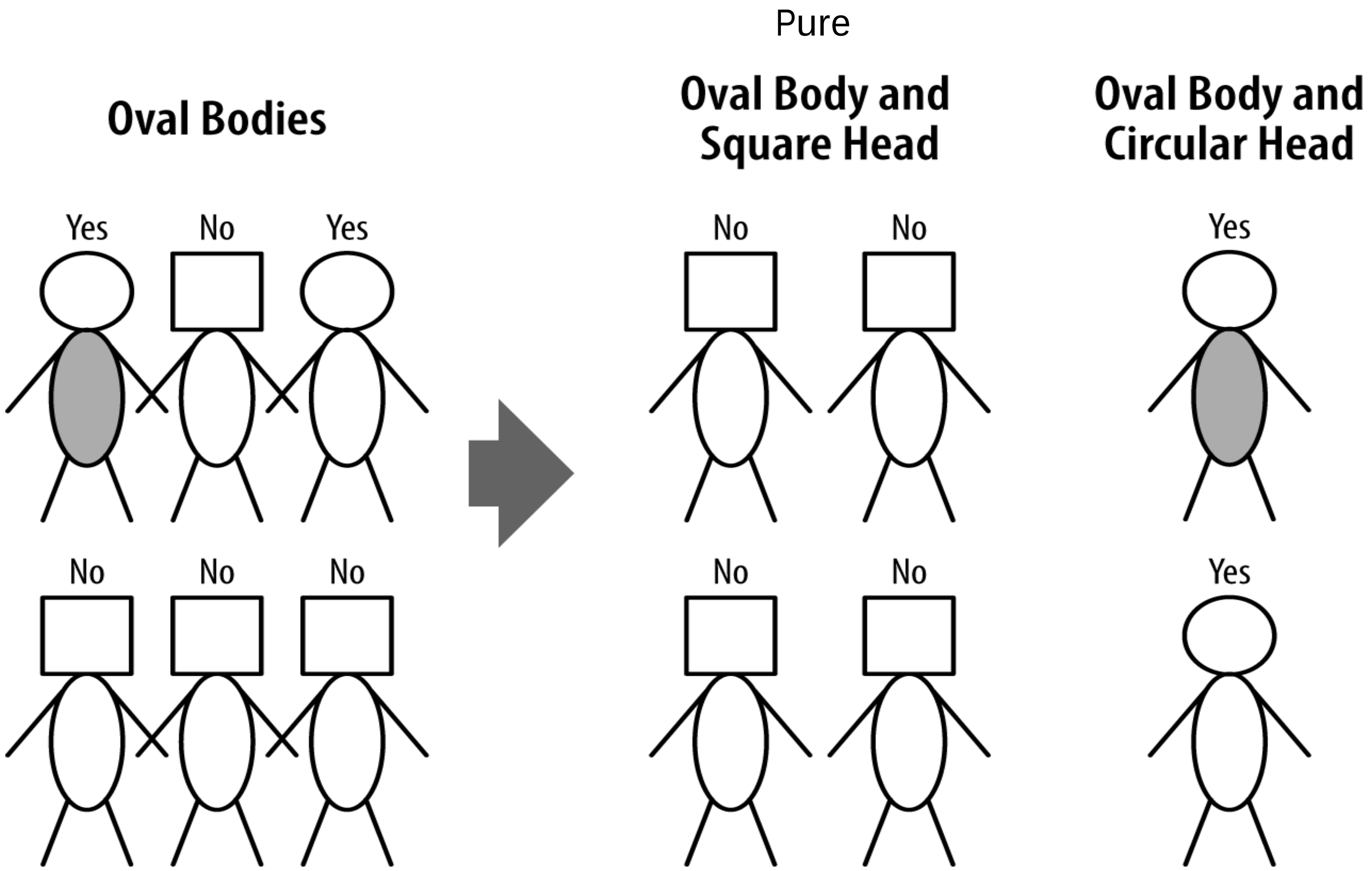
- write-off: Yes, No

First Partitioning: body-shape

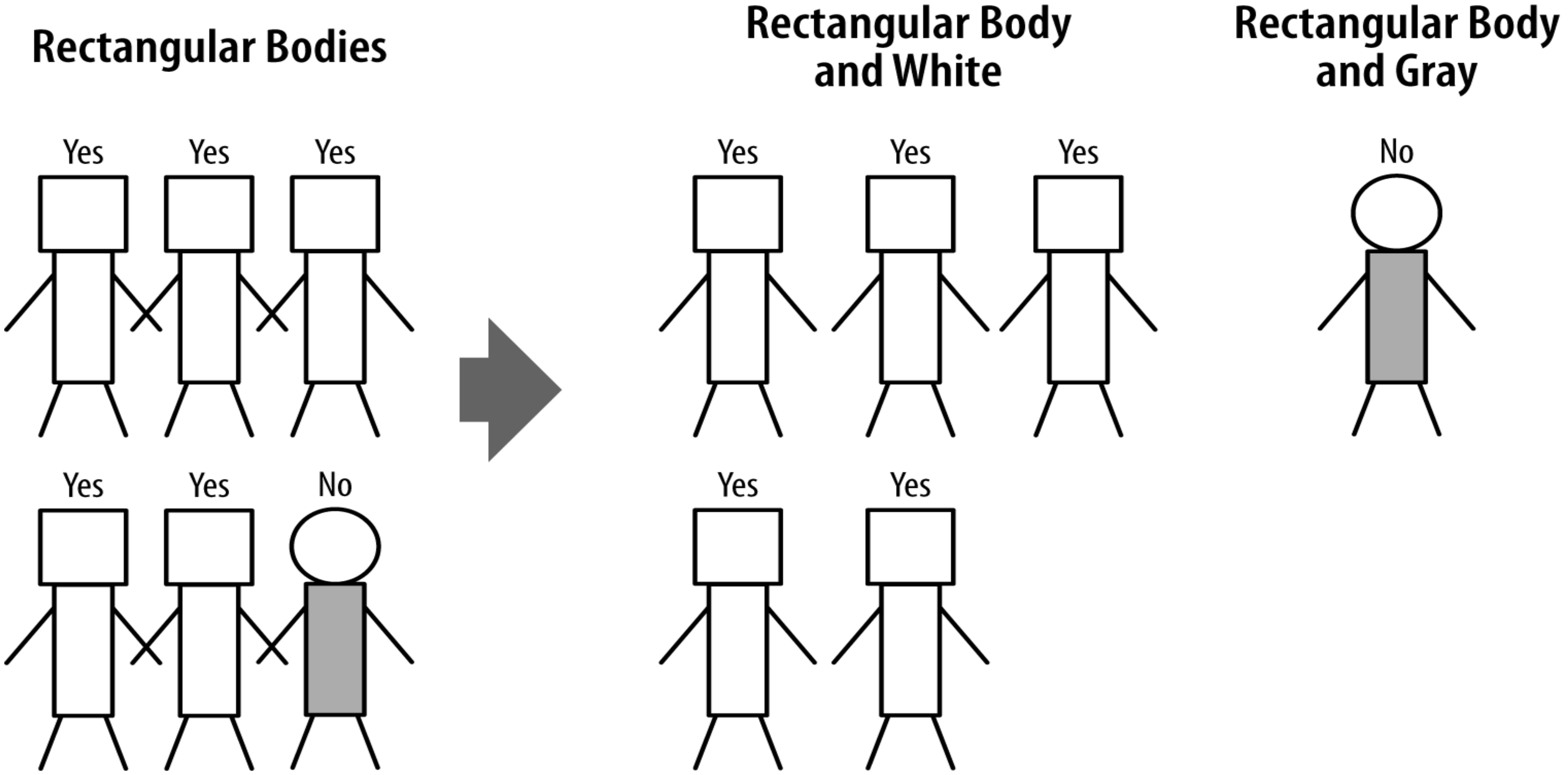


body-shape has the highest IG, so it is selected as the first attribute

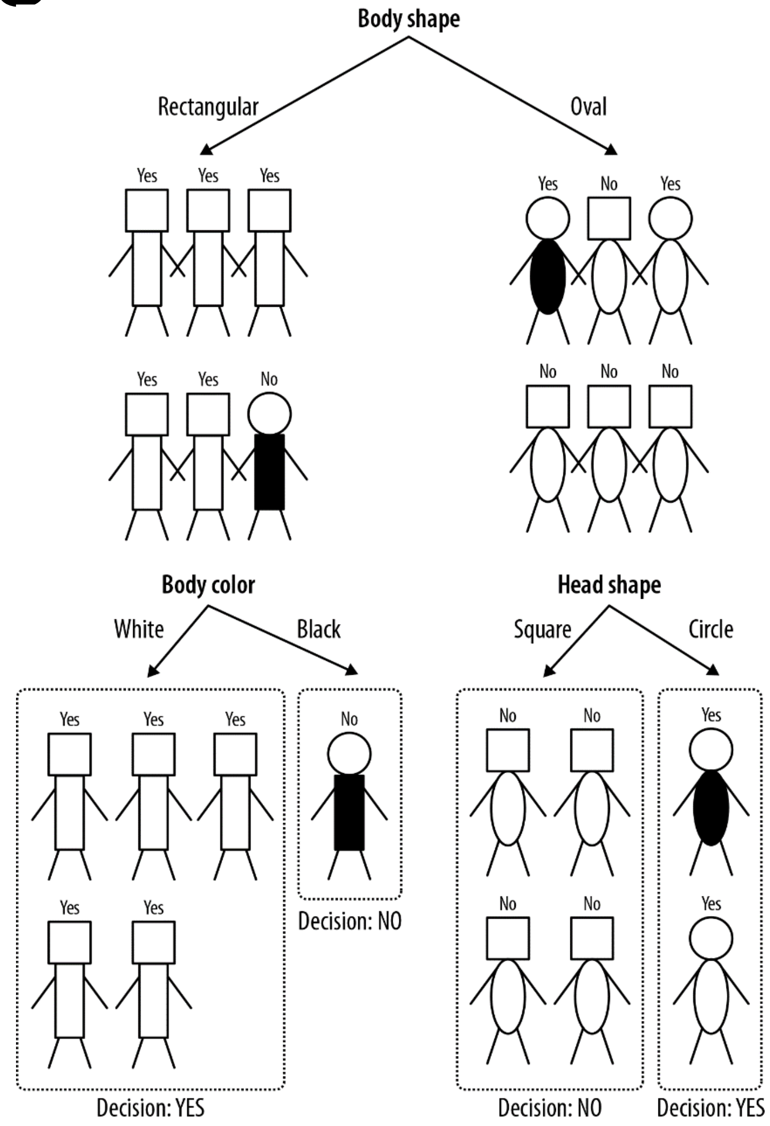
2nd partitioning: oval-body, head-type



3rd partitioning: rectangular-body, body-color



Resulting Decision Tree



Splitting criteria

decision tree could be regression or classification
the IG ovjective fuction could be replaced by other criteria

Regression: residual sum of squares

$$RSS = \sum_{left} (y_i - y_L^*)^2 + \sum_{right} (y_i - y_R^*)^2$$

where y_L^* = mean y-value for left node
y_R^* = mean y-value for right node

Classification: Gini criterion (Similar to Information Gain)

$$Gini = N_L \sum_{k=1,\dots,K} p_{kL} (1- p_{kL}) + N_R \sum_{k=1,\dots,K} p_{kR} (1- p_{kR})$$

where p_{kL} = proportion of class k in left node
p_{kR} = proportion of class k in right node

Classification and Regression Trees

- Grow a binary tree
- At each node, “split” the data into two child nodes
- Splits are chosen using a splitting criterion (e.g. information gain)
- Bottom nodes are terminal nodes
- For regression, the predicted value at a node is the average response variable for all observation in the node
- For classification, the predicted class is the most common class

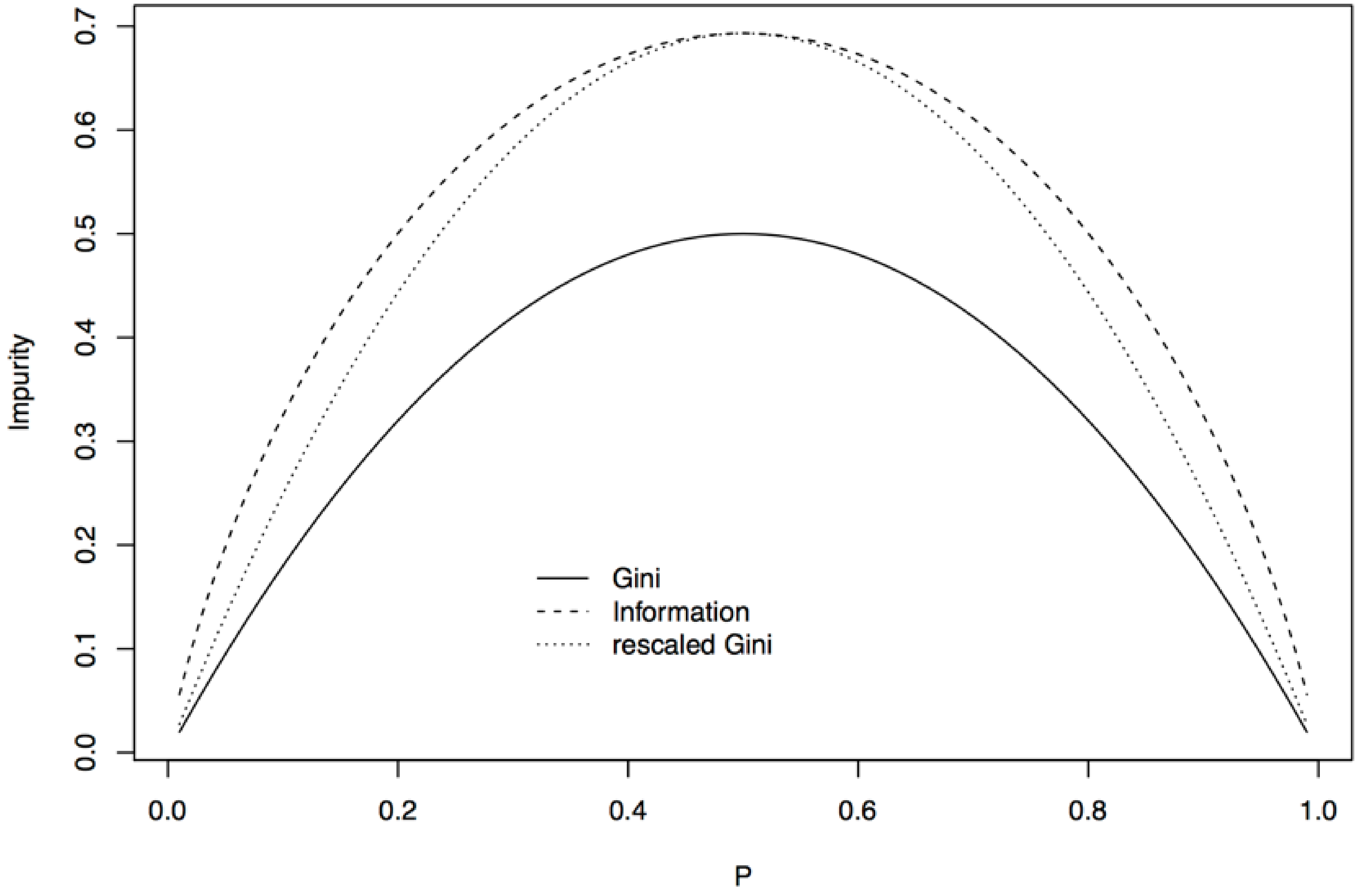


Figure 2: Comparison of Gini and Information impurity for two groups.

Evaluating the classifier

- Mainly evaluate using the *error rate* or *classification accuracy*.

$$\text{accuracy} = \frac{\text{Number of correct decision made}}{\text{Total number of decision made}}$$

$$\text{accuracy} = 1 - \text{error rate}$$

- The accuracy is easy to measure, but sometime not fit to real business problems.
- The accuracy has some well-known problems.
  - The importance of different errors (Unequal Costs and Benefits)
  - Unbalanced class making the algorithms fitting to the dominate class

Confusion Matrix

- Separate out the decisions made by the classifier, making explicit how one class is being confused for another.
- Different errors can then be dealt with separately.
- A 2x2 confusion matrix

ตัวหน้า ทายถูกหรือทายผิด ตัวหลังเป็น โมเดลเราทายว่าเป็นอะไร

	Actual Positive (p)	Actual Negative (n)
The model says "Yes" = positive (y)	True positives	False positives
The model says "No" = not positive (n)	False negatives	True negatives

Counts of the Errors

Counts of the correct decisions

A	Churn	Not churn
Y	500	200
N	0	300

B	Churn	Not churn
Y	500	0
N	200	300

เอามาช่วยแก้ไขปัญหาของ accuracy metrics ในด้านของ unbalanced case or else

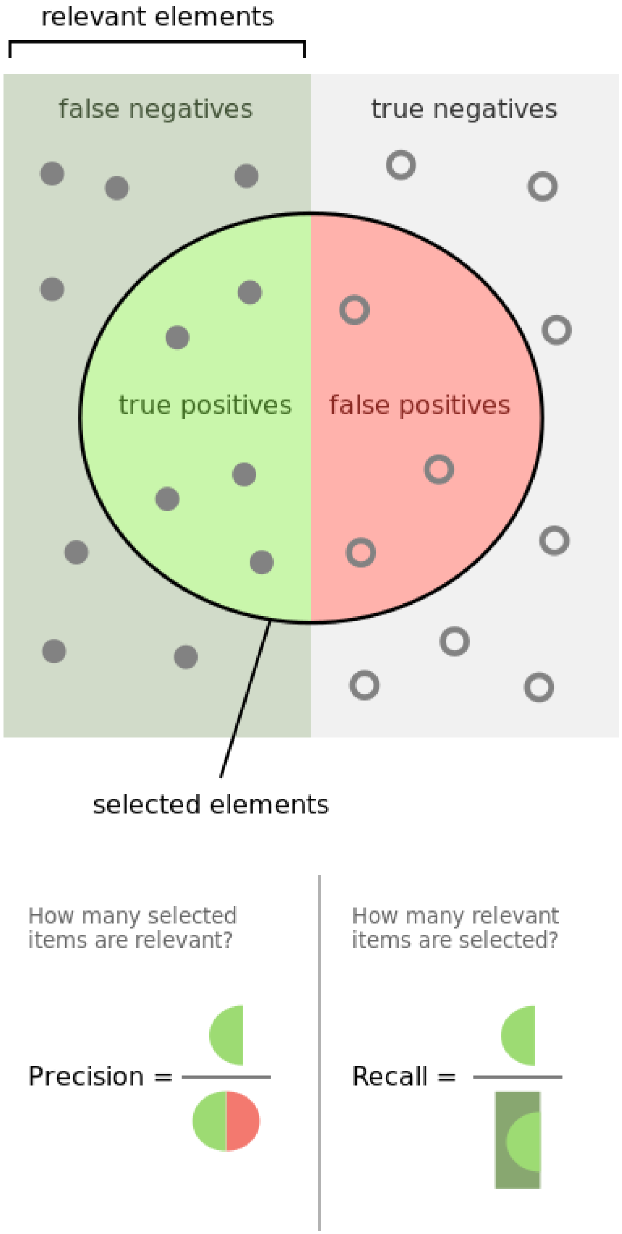
Precision and Recall

	Actual Positive (p)	Actual Negative (n)
The model says “Yes” = positive (y)	True positives	False positives
The model says “No” = not positive (n)	False negatives	True negatives

- Recall (Completeness) = true positive rate = TP/(TP + FN)
- Precision (Exactness) = the accuracy over the cases predicted to be positive, TP/(TP + FP)
- F-measure = the harmonic mean of precision and recall

= the balance between recall and precision

$$= 2 \cdot \frac{precision * recall}{precision + recall}$$



Precision and Recall

- Precision: $\frac{TP}{TP+FP}$ สัดส่วนทายถูกของโมเดล

“Of the responses predicted YES,
how many are actually YES?”

- Recall: $\frac{TP}{TP+FN}$ สัดส่วนทายถูกของชีวิตจริง

“Of the responses that are actually YES,
how many are predicted YES”

Pruning

the larger tree makes the model over fitting

- If the tree is too big, the lower “branches” are modeling noise in the data (“overfitting”).
- The usual paradigm is to grow the trees large and “**prune**” back unnecessary splits.
- Methods for **pruning** trees have been developed. Most use some form of cross validation. Tuning may be necessary.

Classification Strategies

- Binary Classifier: 2 classes
  - e.g. churn, not churn
- Multiclass Classifier: > 2 classes
  - e.g. high, medium, low
- One-vs-The-Rest
  - e.g. to build positive, neutral, negative sentiment prediction. We need (1) positive-vs-not-positive, (2) negative-vs-not-negative.
- One-vs-One
  - Each class is compared to another class

Classification and Regression Trees Disadvantages

- *Accuracy* - current methods, such as support vector machines and ensemble classifiers often have 30% lower error rates than CART.
- *Instability* – if we change the data a little, the tree picture can change a lot. So the interpretation is not as straightforward as it appears.

- Today, we can do better!

Random Forests

Decision Tree: Lab

CROSS-VALIDATION

Issues

- Up until now, we use the training to test the model
- This will eventually lead to an overfitting model
- To avoid this we need to separate the data into training and testing sets
- Or we can sampling the training out of the dataset and train the model and use the whole set to test

Training &Testing: Cross-validation - Why

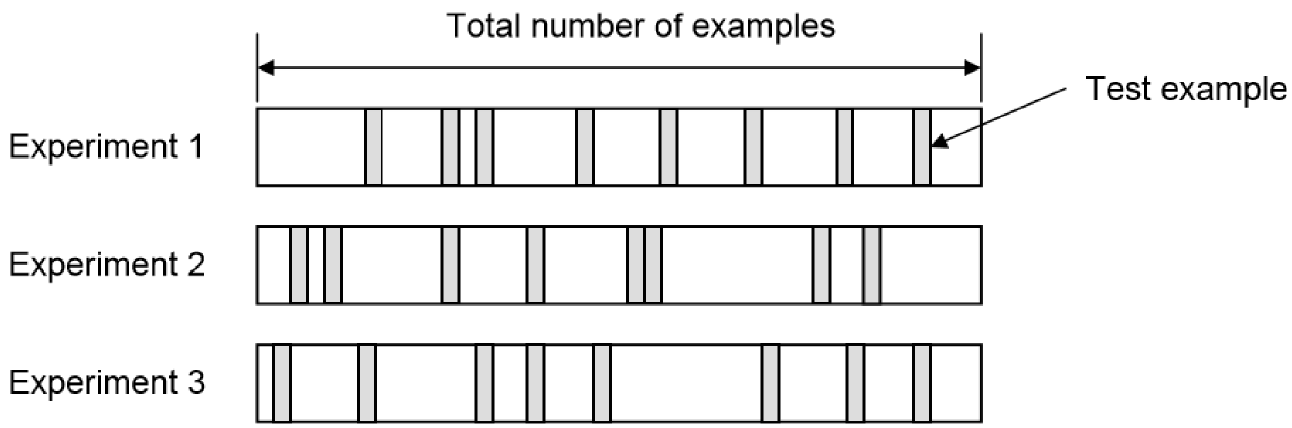
- One may be tempted to use the entire training data to select the "optimal" classifier, then estimate the error rate
- This naïve approach has two fundamental problems

Cons of holdout validation

  - The final model will normally overfit the training data: it will not be able to generalize to new data
    - The problem of overfitting is more pronounced with models that have a large number of parameters
  - The error rate estimate will be overly optimistic (lower than the true error rate)
    - In fact, it is not uncommon to have 100% correct classification on training data

CV: Random subsampling

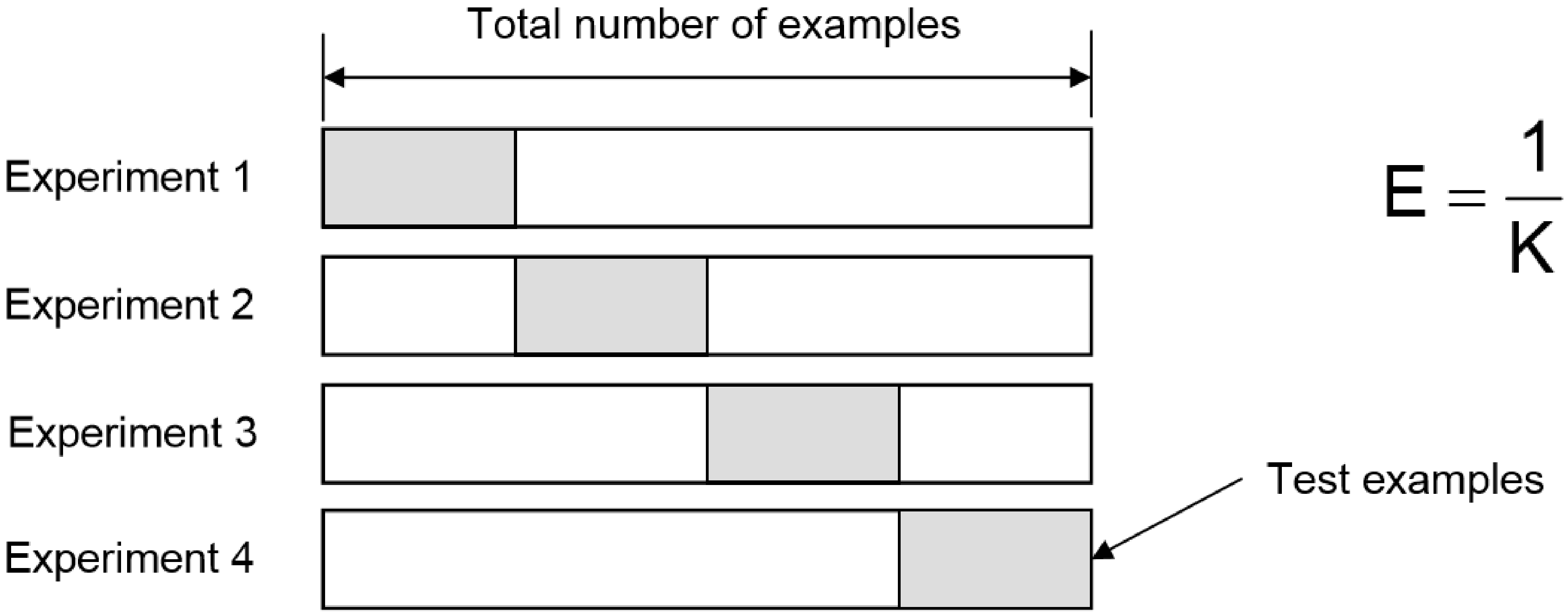
- Performs K data splits of the entire dataset
  - Each data split randomly selects a (fixed) number of examples without replacement
  - For each data split we retrain the classifier from scratch with the training examples and then estimate E_i with the test examples



$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

CV: K-Fold CV

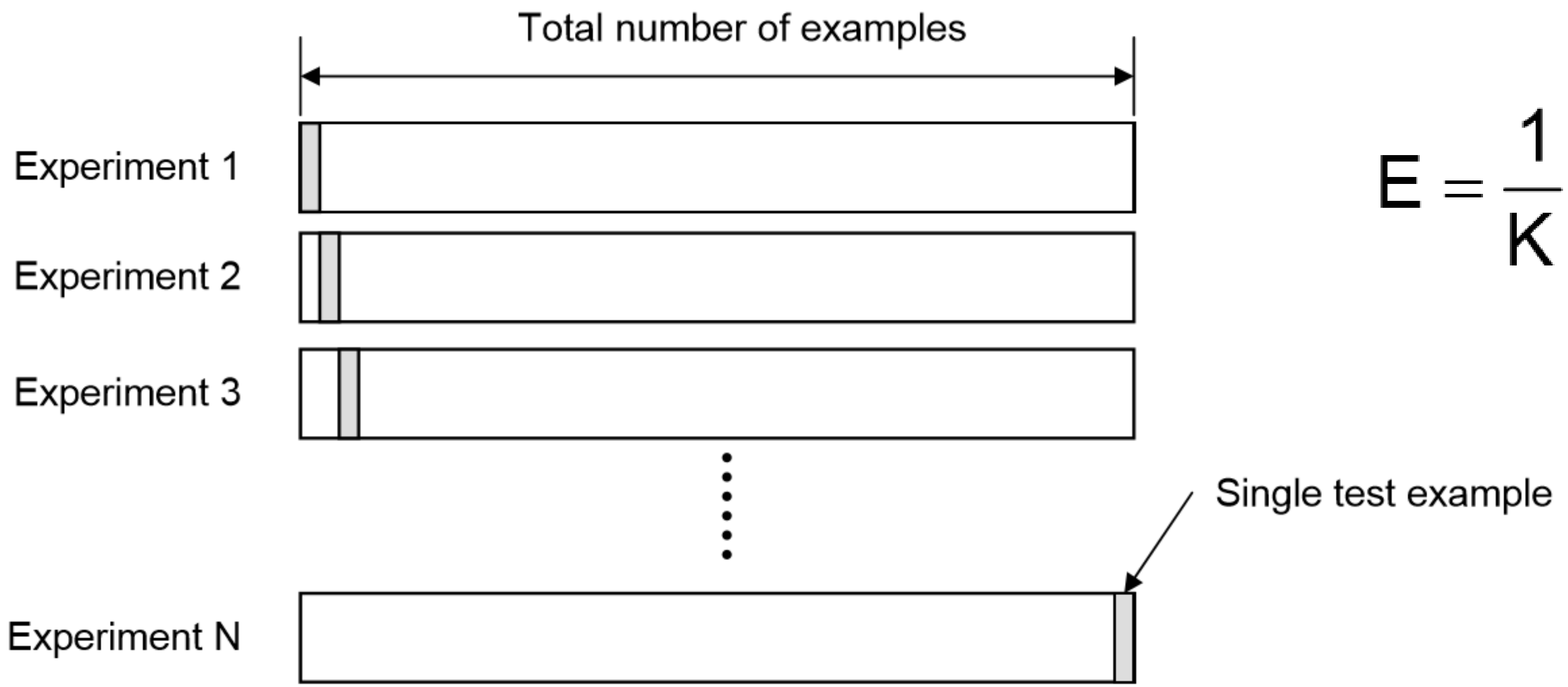
- Create a K-fold partition of the dataset
  - For each of K experiments, use K-1 folds for training and a different fold for testing



$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

CV: Leave-One-Out CV (LOOCV)

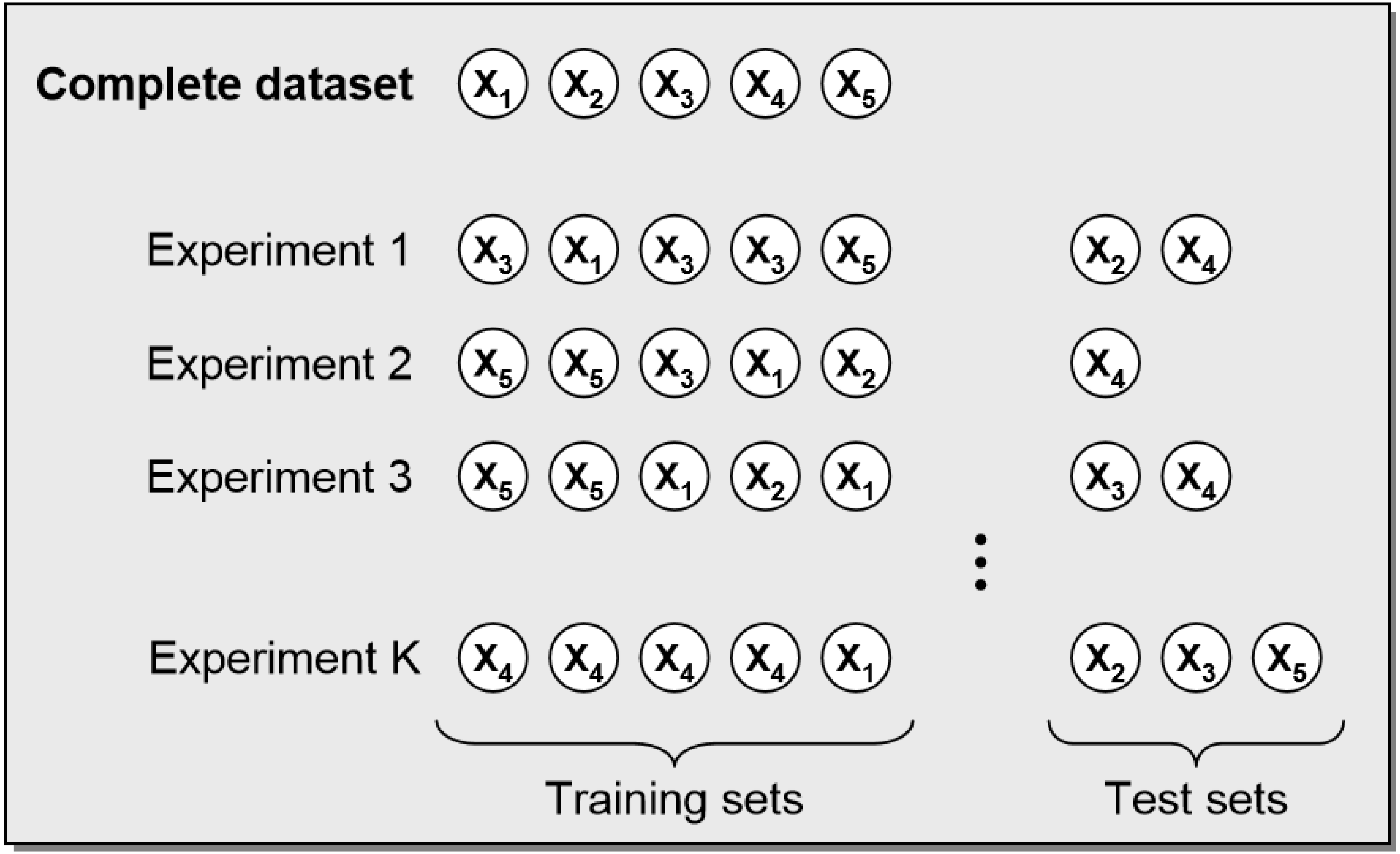
- Leave-one-out is the degenerate case of K-Fold Cross Validation, where K is chosen as the total number of examples



$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

Bootstrap

it's not a cross validation, but it's help to generalize the over fitting model



Cross-validated metrics

- The simplest way to use cross-validation is to call the cross_val_score helper function on the estimator and the dataset.

```
from sklearn.model_selection import cross_val_score
clf = tree.DecisionTreeClassifier()
score = cross_val_score(clf, trainData_X, trainData_Y, cv = 5)
print(score)
print("5-Fold Cross Validation Accuracy : %1.4f" % score.mean())
```

The cross_validate function and multiple metric evaluation

- The cross_validate function differs from cross_val_score in two ways:
  - It allows specifying multiple metrics for evaluation.
  - It returns a dict containing training scores, fit-times and score-times in addition to the test score.
- For single metric evaluation, where the scoring parameter is a string, callable or None, the keys will be ['test_score', 'fit_time', 'score_time']
- And for multiple metric evaluation, the return value is a dict with the following keys ['test_<scorer1_name>', 'test_<scorer2_name>', 'test_<scorer...>', 'fit_time', 'score_time']

cross_validate

```
from sklearn.model_selection import cross_validate


scoring = ['precision','recall']
clf = tree.DecisionTreeClassifier()
scores = cross_validate(clf, trainData_X, trainData_Y, scoring=scoring, cv = 5,
return_train_score= True)
pd.DataFrame(scores)
```

Cross-Validation: Lab

LOGISTIC REGRESSION

Regression so far…

- At this point we have covered:
- Simple linear regression
  - Relationship between numerical response and a numerical or categorical predictor
- Multiple regression
  - Relationship between numerical response and multiple numerical and/or categorical predictors

What we haven't seen is what to do when the predictors are weird (nonlinear, complicated dependence structure, etc.) or when the response is weird (categorical, count data, etc.)

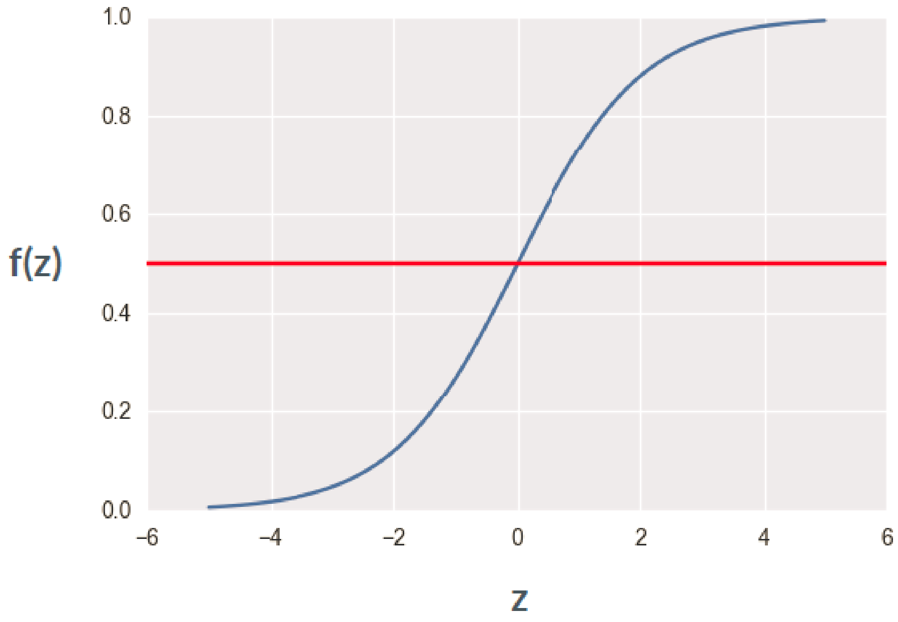
Categorical target

- Categorical target variable has the values in class
- This can be
  - Success/Fail
  - Yes/No
  - Churn/Not Churn
  - Normal/Default
  - Downward/Normal/Upward
    - Upward vs Not-Upward
    - Downward vs Not-Downward

Logistic Regression Model

- Logistic regression is similar to linear regression but used for classification problem

$$h(x) = f(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots)$$



$$f(z) = \frac{1}{1 + e^{-z}}$$

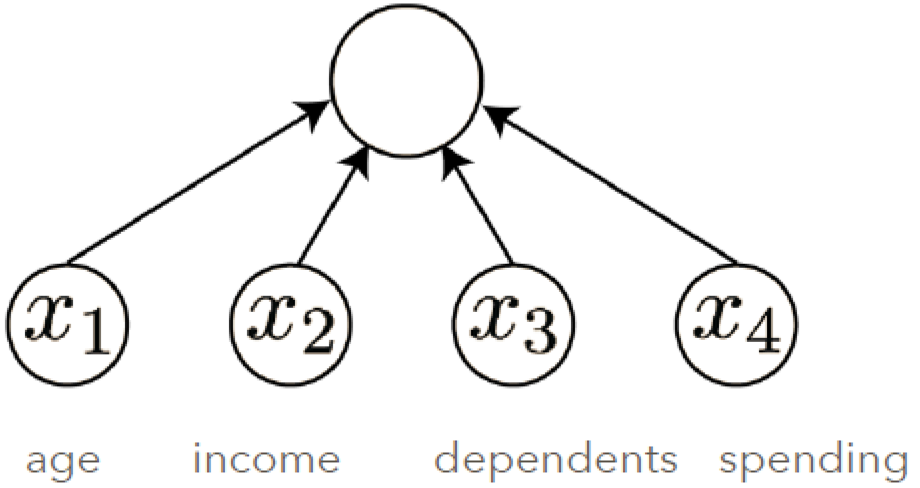
y' = 1, if f(z) > 0.5 or z > 0

y' = 0, if f(z) < 0.5 or z < 0

A Simple Example

Let's look at a simple logistic regression procedure.

$$h = f(\Sigma_j w_j x_j)$$



x1	x2	x3	x4	history
40	50	0	30	1
25	40	2	35	1
18	10	0	12	0
34	22	1	10	1

A Simple Example

We first need to do preprocessing, such as normalization and standardization.

x1	x2	x3	x4	history
40	50	0	30	1
25	40	2	35	1
18	10	0	12	0
34	22	1	10	1

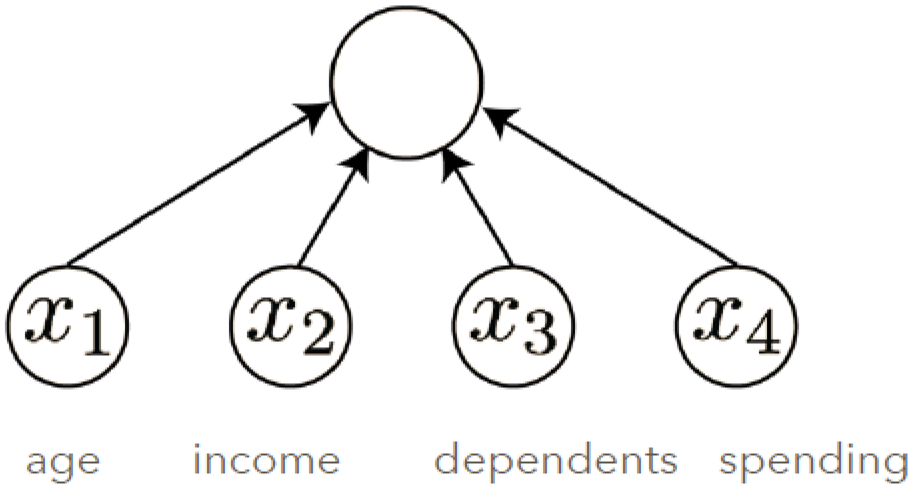
x1	x2	x3	x4	history
0.44	0.63	0	0.6	1
0.28	0.50	0.5	0.7	1
0.20	0.13	0	0.24	0
0.38	0.28	0.25	0.2	1

A Simple Example

Then fit the logistic regression to the data.

Suppose after fitting, here are the weight numbers.

$$h = f(\Sigma_j w_j x_j)$$



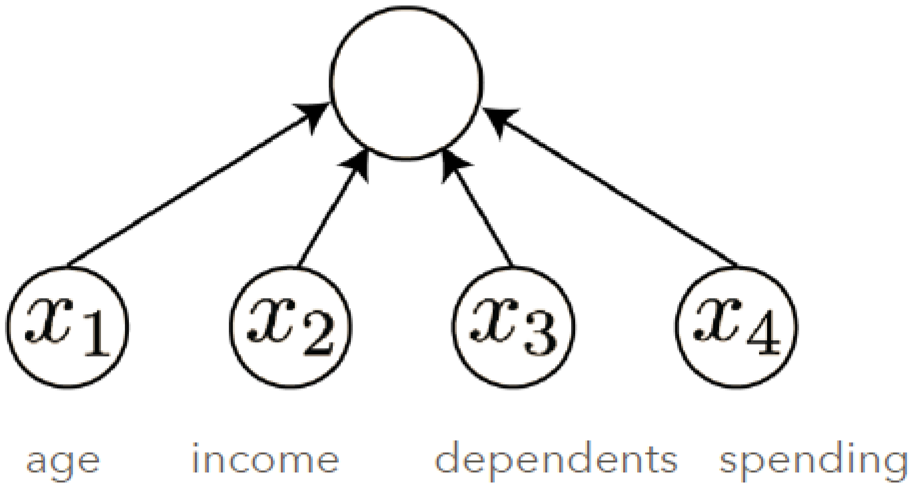
w1	0.7
w2	0.6
w3	-0.1
w4	-0.2

A Simple Example

Let us make prediction for a single customer…

standardized ด้วย mean, std เท่ากับชุดข้อมูลตัวแรก

$$h = f(\Sigma_j w_j x_j)$$



	X	W	X*W
age	0.44	0.7	0.31
income	0.63	0.6	0.38
dependent	0.00	-0.1	0.00
spending	0.60	-0.2	-0.12
		sum:	0.57
		h:	0.64

What the output means

Classification results

	X	W	X*W
age	0.44	0.7	0.31
income	0.63	0.6	0.38
dependent	0.00	-0.1	0.00
spending	0.60	-0.2	-0.12
		sum:	0.57
		h:	0.64

h indicates the probability of customer being good

h = 0.64

64% chance that he will be good

36% chance that he will be bad

Odds

- Odds are another way of quantifying the probability of classes or events, commonly used in gambling, medical (and logistic regression).

$$odds(E) = \frac{P(E)}{P(E^c)} = \frac{P(E)}{1 - P(E)}$$
$$= \frac{x/(x+y)}{y/(x+y)}$$

- The latter is if we are told that the odds of E is *x* to *y*.

Odds: Example 1

- There are 5 pink marbles, 2 blue marbles, and 8 purple marbles
  - Probability of pink

$$\mathrm{P(Pink)} = \frac{5}{5+2+8} = \frac{5}{15} = \frac{1}{3}$$

  - Odds for pink

$$\mathrm{Odds(Pink)} = \frac{5}{2+8} = \frac{5}{10} = \frac{1}{2}$$

Odds: Example 2

- Odds of E is x to y:

$$odds(E) = \frac{x/(x+y)}{y/(x+y)}$$

- What are the odds that a randomly chosen day of the week is a weekend?

Odds: Example 2

- Odds of E is x to y:

$$odds(E) = \frac{x/(x+y)}{y/(x+y)}$$

- What are the odds that a randomly chosen day of the week is a weekend?

$$2{:}5 = (2/7)/(5/7)$$

Assess relationships of categorical variables

- It seems that there are relationships between categorical variables.
- How do we assess them?

Chi-square statistics

- Measures of association provide a means of summarizing the size of the association between two variables.
- One way to determine whether there is a statistical relationship between two variables is to use “the chi square test for independence”

Chi-square statistics

- A cross classification table is used to obtain the expected number of cases under the *assumption* of no relationship between the two variables
- The value of the chi square statistic provides a test whether or not there is a statistical relationship between the variables in the cross classification table.

Chi-square statistics

- Hypothesis for the Chi-square Test of Independence
  - H_0: In the population, the two categorical variables are independent.
  - H_α: In the population, two categorical variables are dependent.

Chi-square: expectation

$$\chi^2 = \sum_{i=1}^{n} \frac{\left(\text{observed - expected}\right)^2}{\text{expected}}$$

Chi-square:
Cross Classification Table (observed)

	no	yes
divorced	4585	622
married	24459	2755
single	10878	1912

Chi-square:
Finding Expected Counts from Observed Counts

$$E = \frac{\text{row total} \times \text{column total}}{\text{sample size}}$$

	no	yes	total
divorced	4585	622	5207
married	24459	2755	27214
single	10878	1912	12790
total	39922	5289	45211

Chi-square: Expected counts

$$E = \frac{\text{row total } \times \text{column total}}{\text{sample size}}$$

	no	yes	total
divorced	$\frac{5207 \times 39922}{45211} = 4597.86$	$\frac{5207 \times 5289}{45211} = 609.14$	5207
married	$\frac{27214 \times 39922}{45211} = 24030.38$	$\frac{27214 \times 5289}{45211} = 3183.62$	27214
single	$\frac{12790 \times 39922}{45211} = 11293.76$	$\frac{12790 \times 5289}{45211} = 1496.24$	12790
total	39922	5289	45211

Chi-square statistics

	no	yes	total
divorced	4585 (4597.86)	622 (609.14)	5207
married	24459 (24030.38)	2755 (3183.624)	27214
single	10878 (11293.76)	1912 (1496.236)	12790
total	39922	5289	45211

Chi-square statistics

	no	yes
divorced	**4585 (4597.86)**	**622 (609.14)**
married	**24459 (24030.38)**	**2755 (3183.624)**
single	**10878 (11293.76)**	**1912 (1496.236)**

Calculate the test statistic by hand:

$$\chi^2 = \sum_{i=1}^{n} \frac{(\text{observed - expected})^2}{\text{expected}}$$

$\chi^2 = 196.5$

with degree of freedom = (2-1)(3-1) = 2

p-value to Hypothesis Testing

- *p*-value is the probability of observing a sample value as extreme as, or more extreme than, the value observed, given that the null hypothesis is true.
- In testing a hypothesis, we can compare the *p*-value to the significance level (α)
- Decision rule using the *p*-value:

$$\text{Reject } H_0 \text{ if } p\text{-value} < \text{significance level}$$

p-value to Hypothesis Testing

TABLE 1

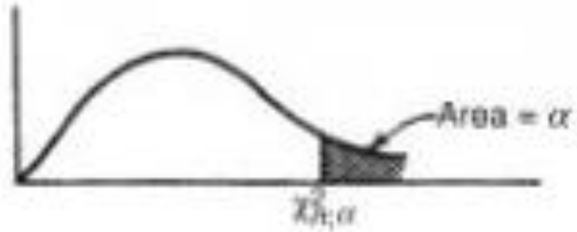
Relationship between Common Language and Hypothesis Testing

COMMON LANGUAGE	STATISTICAL STATEMENT	CONVENTIONAL TEST THRESHOLD
"Statistically significant" "Unlikely due to chance"	The null hypothesis was rejected.	$P < 0.05$
"Not significant" "Due to chance"	The null hypothesis could not be rejected.	$P > 0.05$

Source: http://ecp.acponline.org

Chi-square Table

Value of $\chi^2_{n;\alpha}$ such that $\text{Prob}[\chi^2_n > \chi^2_{n;\alpha}] = \alpha$



n	0.995	0.990	0.975	0.950	0.900	0.10	0.05	0.025	0.010	0.005
1	0.000039	0.00016	0.00098	0.0039	0.0158	2.71	3.84	5.02	6.63	7.88
2	0.0100	0.0201	0.0506	0.103	0.211	4.61	5.99	7.38	9.21	10.60
3	0.0717	0.115	0.216	0.352	0.584	6.25	7.81	9.35	11.34	12.84
4	0.207	0.297	0.484	0.711	1.06	7.78	9.49	11.14	13.28	14.86
5	0.412	0.554	0.831	1.15	1.61	9.24	11.07	12.83	15.09	16.75

Binomial distribution

เช่นการโยนหัวกับก้อย เป็น binomial distribution

- It seems clear that both age and job have an effect on the subscription, how do we come up with a model that will let us explore this relationship?
- Even if we set no to 0 and yes to 1, this isn’t something we can transform our way out of - we need something more.
- One way to think about the problem - we can treat yes and no as successes and failures arising from a binomial distribution where the probability of a success is given by a transformation of a linear model of the predictors.

Generalized linear model

- It turns out that this is a very general way of addressing this type of problem in regression, and the resulting models are called generalized linear models (GLMs). logistic เป็นส่วนหนึ่ง
- Logistic regression is just one example of this type of model.

Generalized linear models

All generalized linear models have the following three characteristics:

1. A probability distribution describing the outcome variable
2. A linear model
   - $\eta = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$
3. A link function that relates the linear model to the parameter of the outcome distribution
   - $g(p) = \eta$ or $p = g^{-1}(\eta)$

GLM: Simple Linear Regression

- *Probability distribution*: normal distribution
- *Linear model*:

$$E(y_i) = y_i = b_0 + b_1 x_i$$

- *Link function*: identity link

$$\eta = g\big(E(y_i)\big) = E(y_i)$$

BIG DATA EXPERIENCE

KMUTT g-able

Logistic regression

probability classification

- Logistic regression is a GLM used to model a binary categorical variable using numerical and categorical predictors.
- We assume a binomial distribution produced the outcome variable and we therefore want to model p the probability of success for a given set of predictors.
- To finish specifying the Logistic model we just need to establish a reasonable link function that connects η to p.
- There are a variety of options but the most commonly used is the logit function.

Logit function สมการของ link function

$$logit(p) = \log\left(\frac{p}{1-p}\right), \text{ for } 0 \leq p \leq 1$$

GLM: Logistic Regression

- *Probability distribution*: binomial distribution
- *Linear model*:

$$\eta = b_0 + b_1 x_1 + \cdots + b_i x_i$$

- *Link function*: logit

$$\eta = \text{logit}(\pi) = \log\left(\frac{\pi}{1-\pi}\right)$$

Logistic regression

- When the ordinal/numeric input associating with categorical dependent variable, logistic regression can be used dependent ต้องเป็น categorical
- Suitable when
  - Number of features is large, or
  - Number of observations is large
- Used in many models, including churn prediction

Precision and Recall

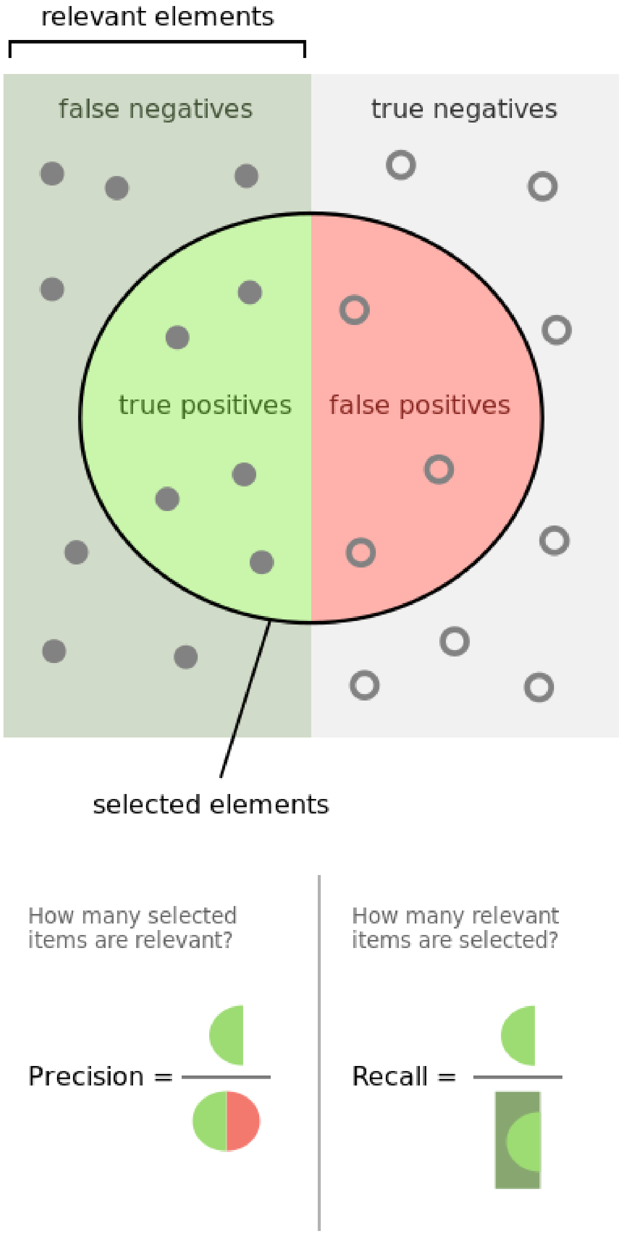
F_score คือการ trade off ระหว่าง precision/recall แต่ในบางวงการก็สนใจแค่บางตัว
เช่น วงการแพทย์สนใจแต่ recall

	Actual Positive (p)	Actual Negative (n)
The model says “Yes” = positive (y)	True positives	False positives
The model says “No” = not positive (n)	False negatives	True negatives

- Precision (Exactness) = the accuracy over the cases predicted to be positive, TP/(TP + FP)
- Recall (Completeness) = true positive rate = TP/(TP + FN)
- F-measure = the harmonic mean of precision and recall

= the balance between recall and precision

$$= 2 \cdot \frac{precision * recall}{precision + recall}$$



Logistic Regression: Lab

NAIVE BAYES CLASSIFIER

What if we have more than 2 categories?

- Sentiment: Positive, Negative, Neutral
- Document topics: Sports, Politics, Business, Entertainment, …

Naive Bayes Classifier

- a classification technique based on **Bayes' Theorem** with an assumption of independence among predictors
- easy to build and particularly useful for very large data sets

Bayes' Theorem

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \qquad \text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

$$P(c|x) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

A Simple Example

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

A Simple Example

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14 \quad P(\text{Play}=No) = 5/14$

A Simple Example

- Test Phase
  - Given a new instance,

    $\mathbf{x}'$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
  - Look up tables

    P(Outlook=*Sunny*|Play=*Yes*) = 2/9 P(Outlook=*Sunny*|Play=*No*) = 3/5

    P(Temperature=*Cool*|Play=*Yes*) = 3/9 P(Temperature=*Cool*|Play==*No*) = 1/5

    P(Huminity=*High*|Play=*Yes*) = 3/9 P(Huminity=*High*|Play=*No*) = 4/5

    P(Wind=*Strong*|Play=*Yes*) = 3/9 P(Wind=*Strong*|Play=*No*) = 3/5

    P(Play=*Yes*) = 9/14 P(Play=*No*) = 5/14
  - MAP rule

P(*Yes*|$\mathbf{x}'$): [P(*Sunny*|*Yes*)P(*Cool*|*Yes*)P(*High*|*Yes*)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053

P(*No*|$\mathbf{x}'$): [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

Given the fact P(*Yes*|$\mathbf{x}'$) < P(*No*|$\mathbf{x}'$), we label $\mathbf{x}'$ to be "*No*".

Naive Bayes Classifier

- Pros:
  - Easy and fast
  - Perform well in multi class prediction
  - When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- Cons:
  - “Zero Frequency” ไม่สามารถคำนวณ situation ที่ไม่เคยเกิดขึ้นได้
    - If categorical variable has a category that was not observed in training data set
    - unable to make a prediction.
    - Solution: smoothing technique e.g. Laplace estimation.
  - On the other side naive Bayes is also known as a bad estimator
    - In real life, it is almost impossible that we get a set of predictors which are completely independent.

BIG DATA
EXPERIENCE

Thank you

Question?

BIG DATA
EXPERIENCE
CENTER

KMUTT g-able