Decision Trees (I)

Instructor: Jing Wang

Department of ISOM

Spring 2023

HKUST
BUSINESS SCHOOL
香港科大商學院

Recap: Data Mining Basics

- What is data mining?
- What are variables/target variable?
- What is supervised vs. unsupervised learning?
- What is classification vs. regression?
- What is mining phase vs. using phase?
- What is the general data mining process?

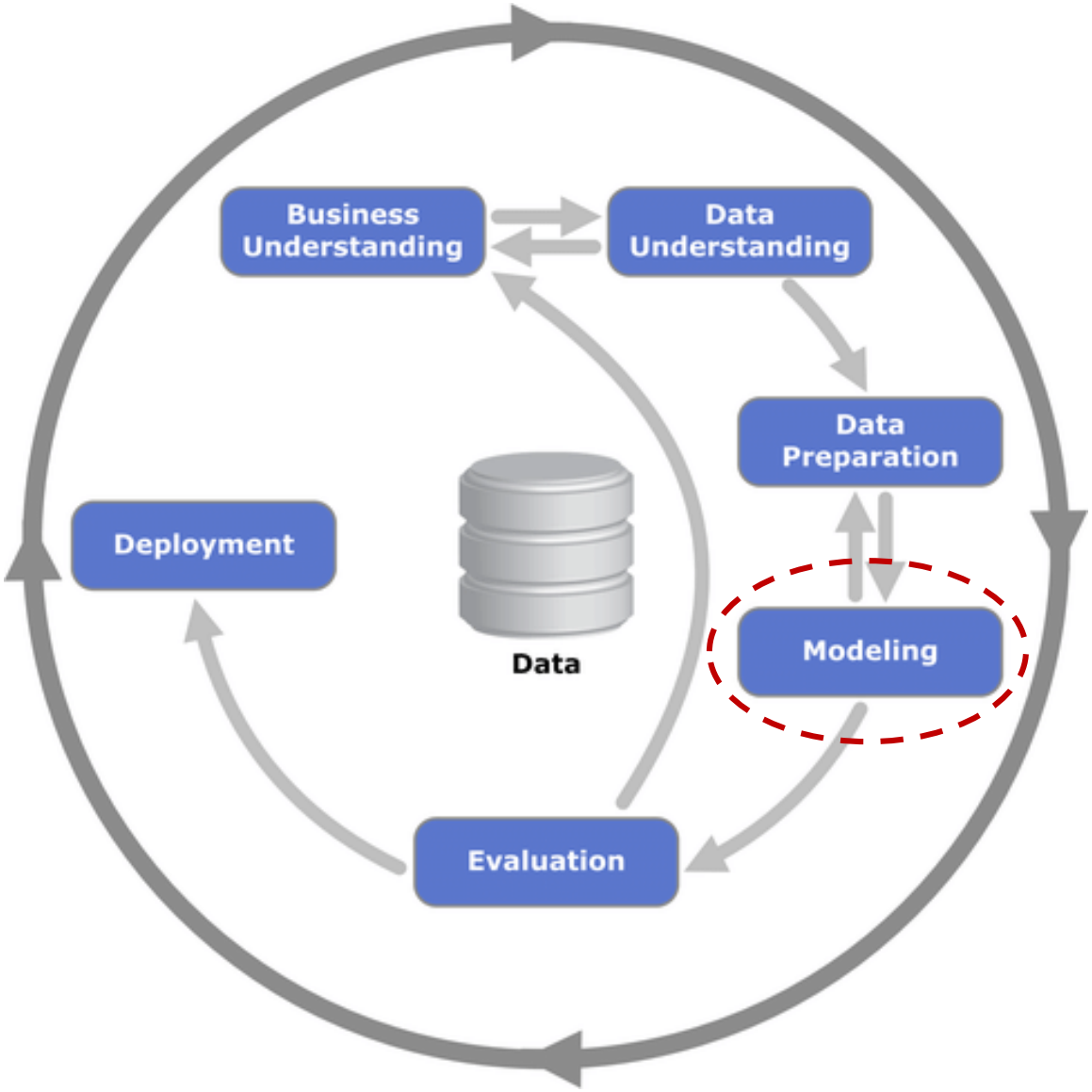
Recap: Data Understanding

- Preliminary investigation of the data to better understand its specific characteristics
  - Help in selecting appropriate data mining algorithms
- Things to look at
  - Class imbalance
  - Dispersion of data attribute values
  - Skewness, outliers, missing values
  - Correlation analysis
- Visualization tools are important
  - Histograms, box plots
  - Scatter plots

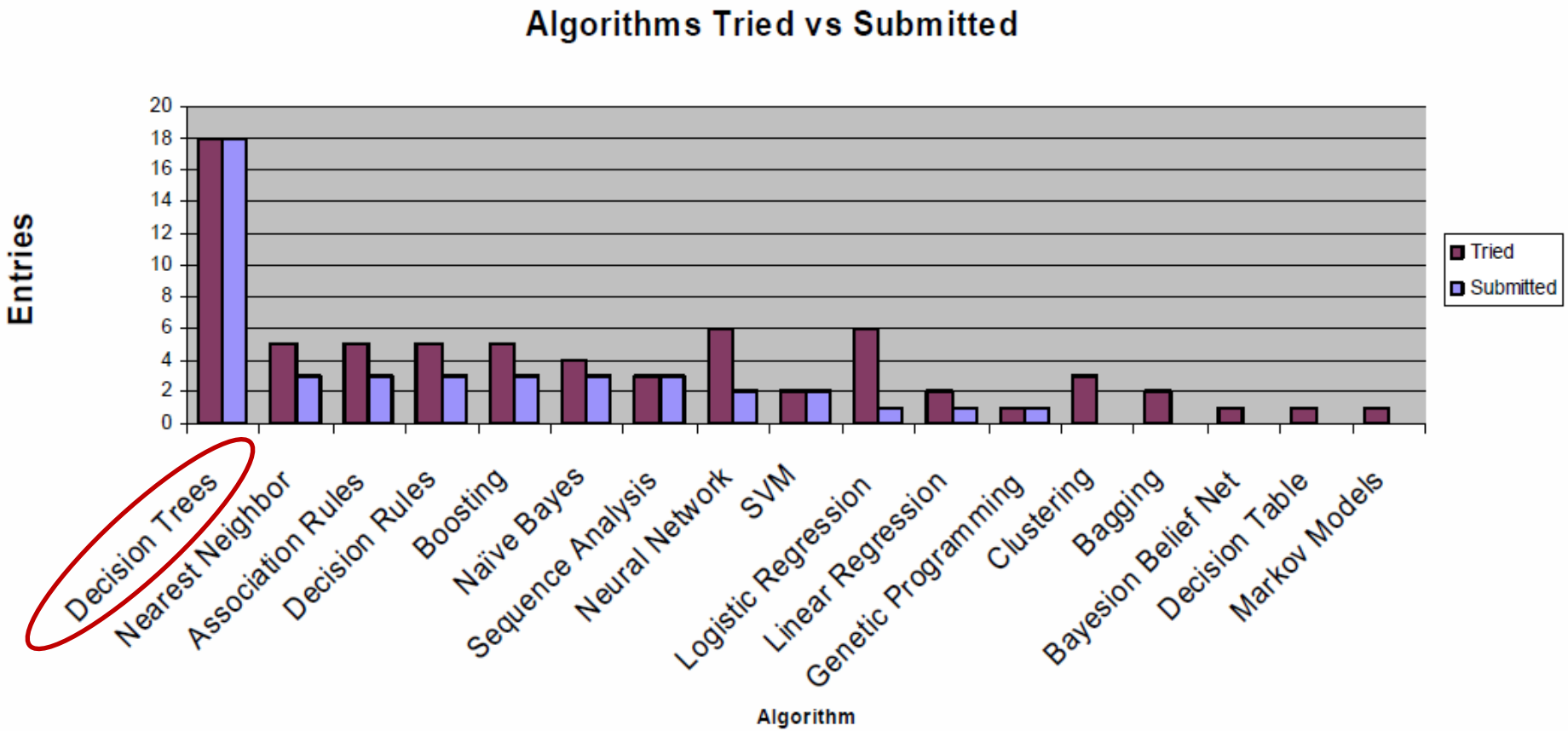
Recap: Data Preprocessing

- Data transformation might be needed
  - Handling missing values
  - Handling categorical variables
  - Feature transformation
    - e.g., log transformation
    - Normalization
  - Feature discretization
  - Feature selection

What Induction Algorithm Shall We Use?



Commonly Used Induction Algorithms



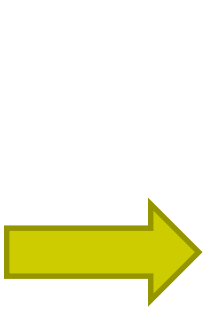
Post-mortem analysis of a popular data mining competition

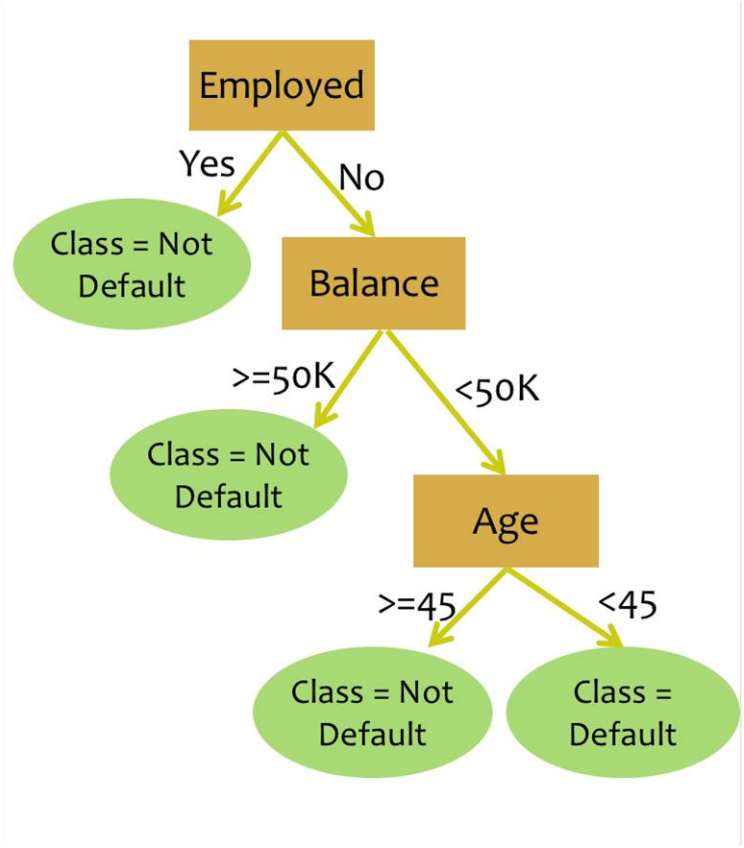
Why Decision Trees?

- Decision trees are one of the most popular data mining tools.
  - Classification trees (used when target variable is categorical)
  - Regression trees (used when target variable is numeric)
- They are easy to understand, implement and use, and computationally cheap.
- The model comprehensibility is important for communication to non-DM-savvy stakeholders.

Classification Tree

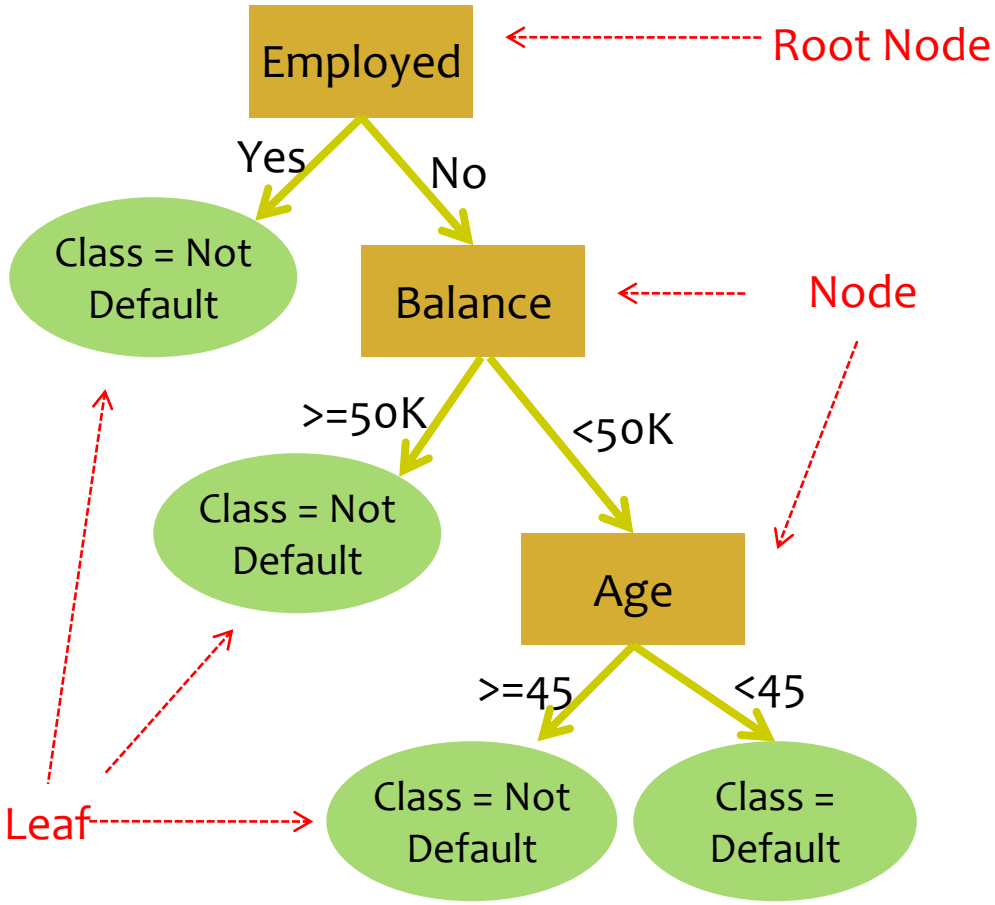
Employed	Balance	Age	Default
Yes	123,000	50	No
No	51,100	40	Yes
No	68,000	55	No
Yes	34,000	46	Yes
Yes	50,000	44	No
No	100,000	50	Yes





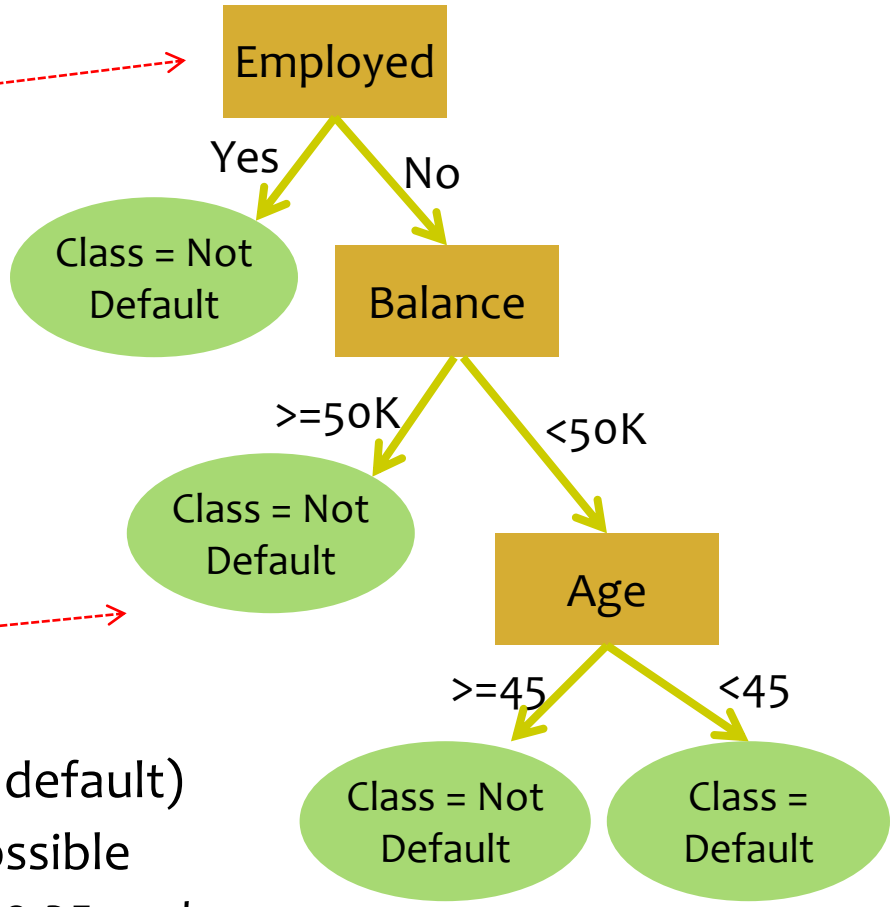
Objective: predicting borrowers who will default on loan payments.

Classification Tree: Upside-Down



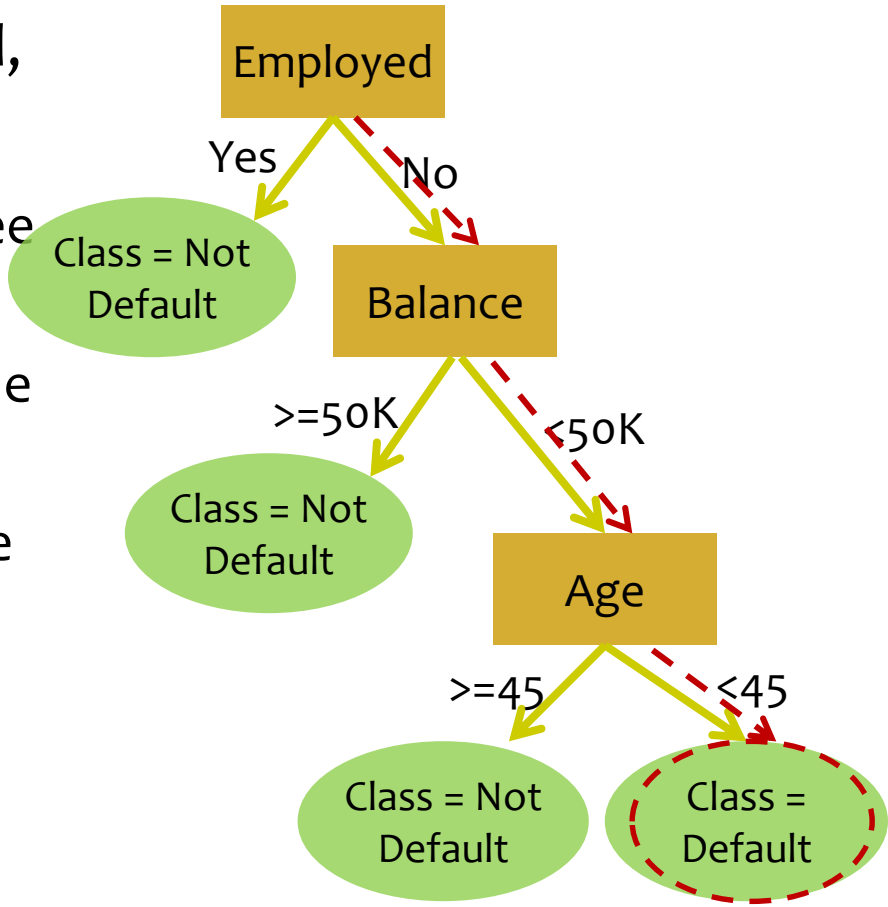
Classification Tree: Divide and Conquer

- "Recursive Partitioning"
- Nodes
  - Each node represents one attribute
  - Tests on nominal attribute: number of splits (branches) is number of possible values or 2 (one value vs. rest)
  - Continuous attributes are discretized
- Leaves
  - A class assignment (e.g., default /not default)
  - Also provide a distribution over all possible classes (e.g., default with probability 0.25, not default with prob. 0.75)

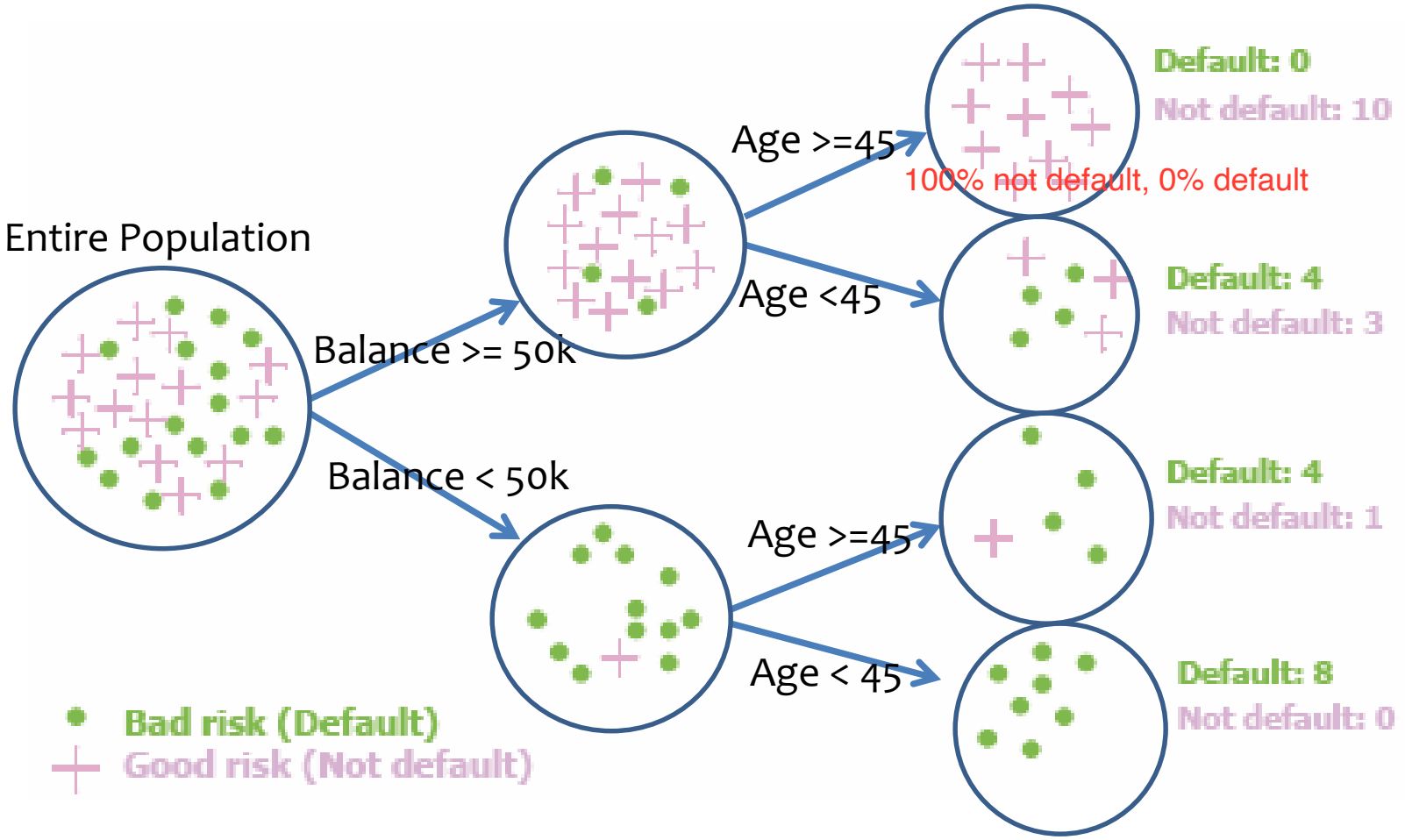


How a Tree is Used for Classification ?

- To determine the class of a new example: e,g., Mark, age 40, retired, balance 38K.
  - The example is routed down the tree according to values of attributes .
  - At each node, a test is applied to one attribute.
  - When a leaf is reached, the example is assigned to a class—or alternatively to a distribution over the possible classes (e.g., default with probability 0.25, not default with prob. 0.75).

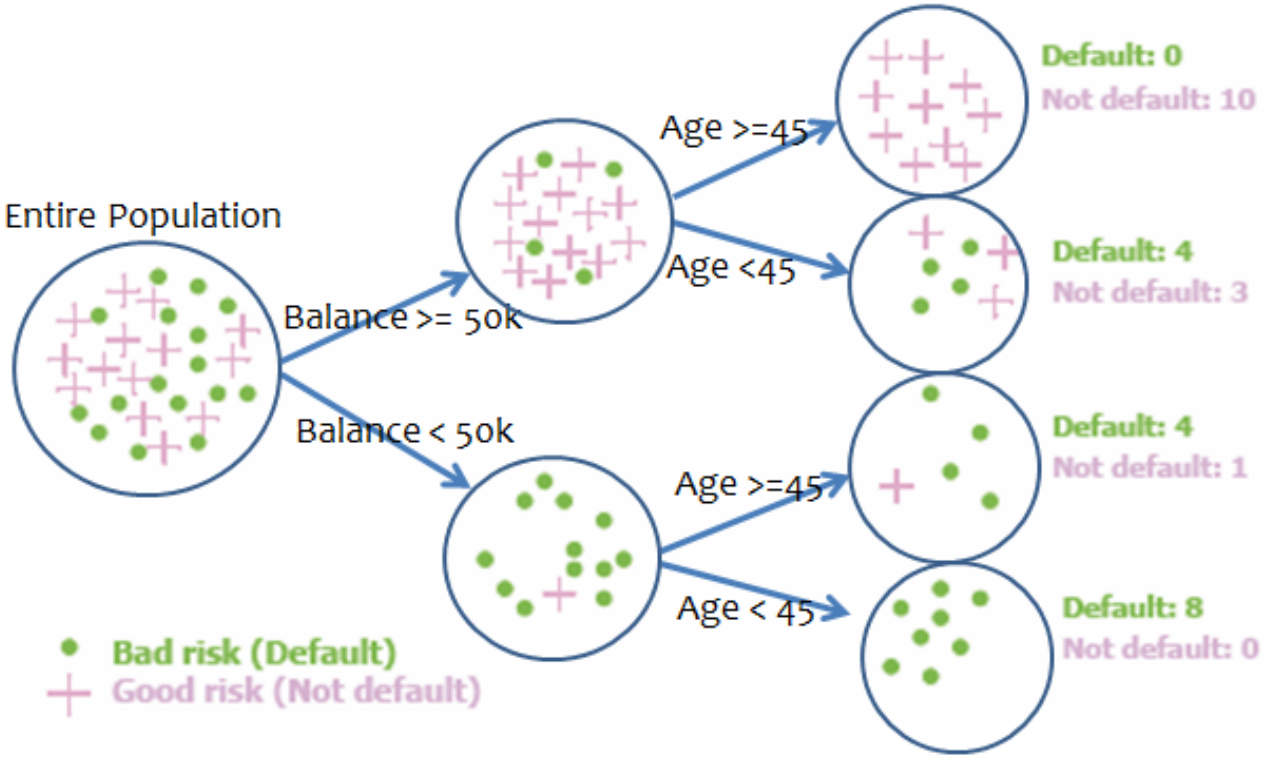


Assigning Probability Estimates



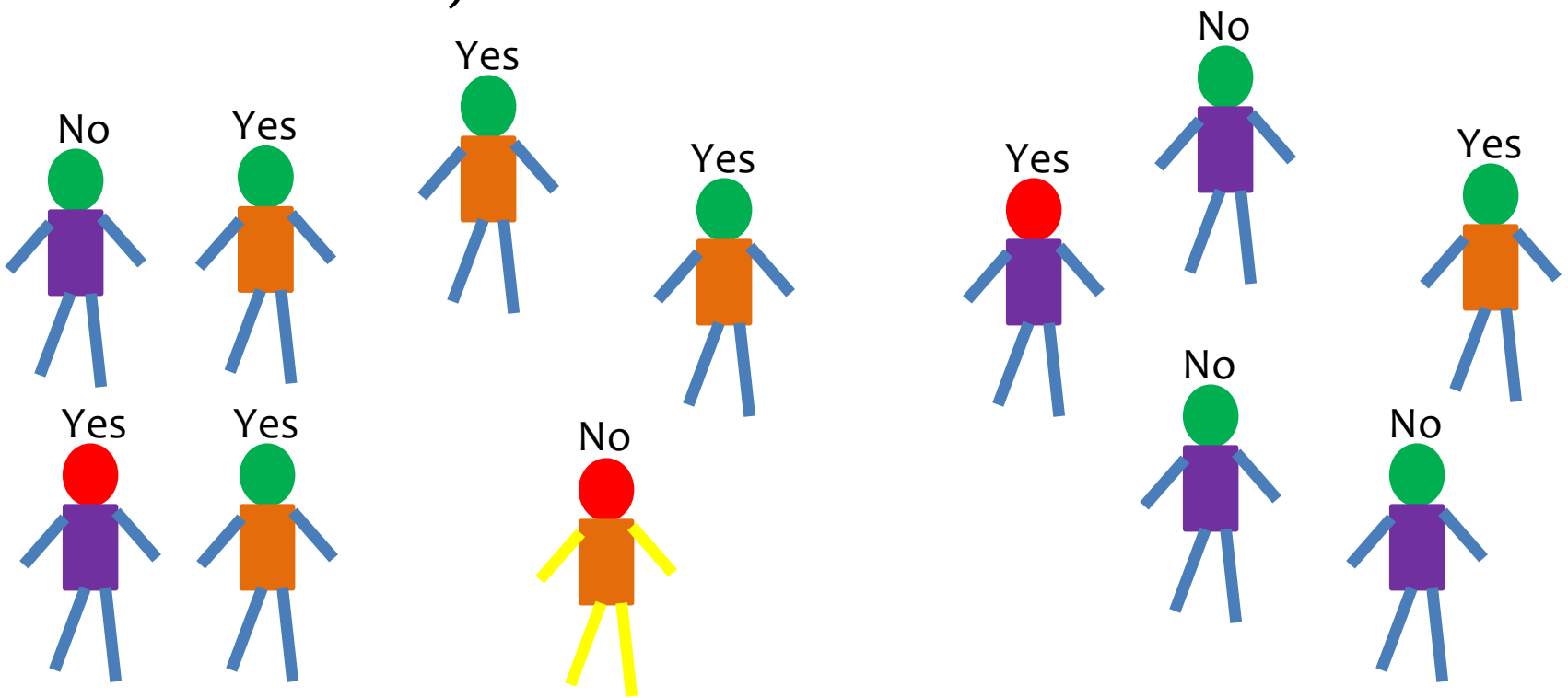
Q: How would you assign estimates of class probability (probability of default/not default) based on such trees?

Exercise: Assume this is the classification tree you learned from the past defaulting data. A new guy, who is 45 and has 20K balance, is applying a credit card issued by your company. *Can you predict if this new guy is gonna default?* How confident are you about your prediction? Another girl is also applying the same credit card. But the only information we have about her is she has 70k balance. *Can you predict if she will default? How sure about that?*



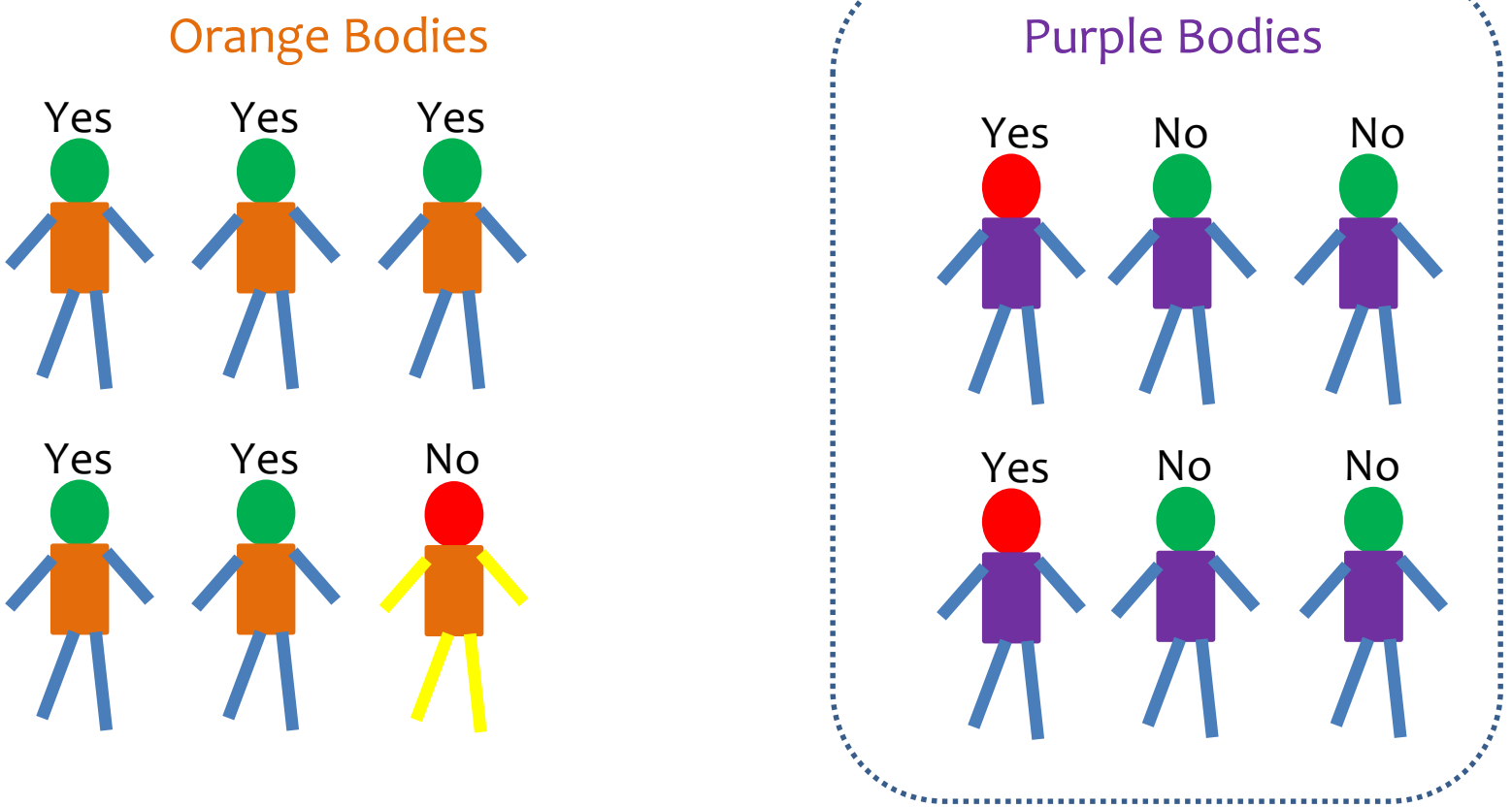
Classification Tree Learning

- **Objective:** based on customer attributes, partition the customers into subgroups that are less impure – with respect to the class (i.e., such that in each group most instances belong to the same class)



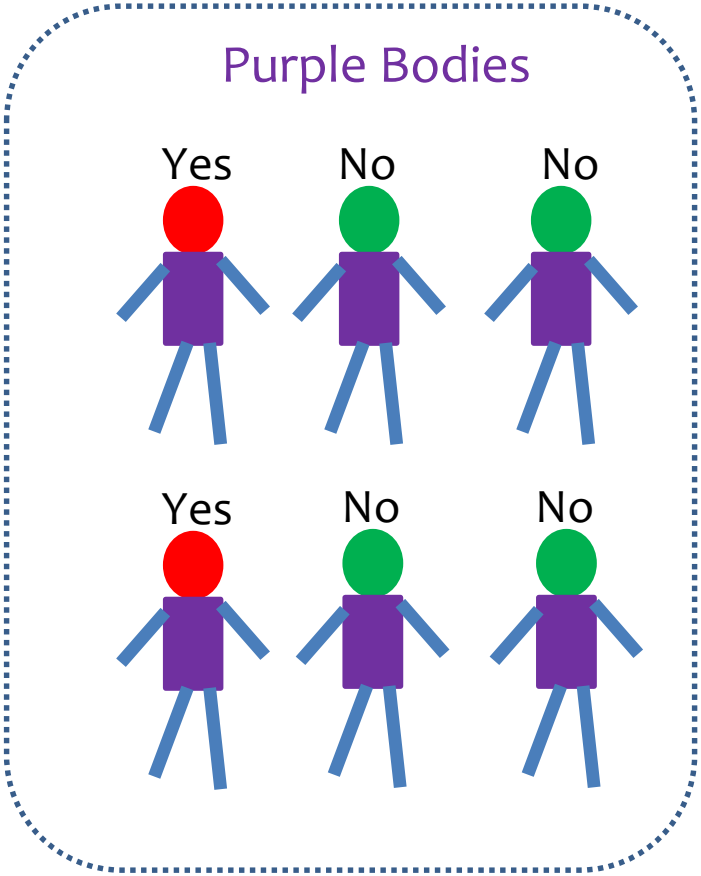
Classification Tree Learning

- Partitioning into "purer" groups

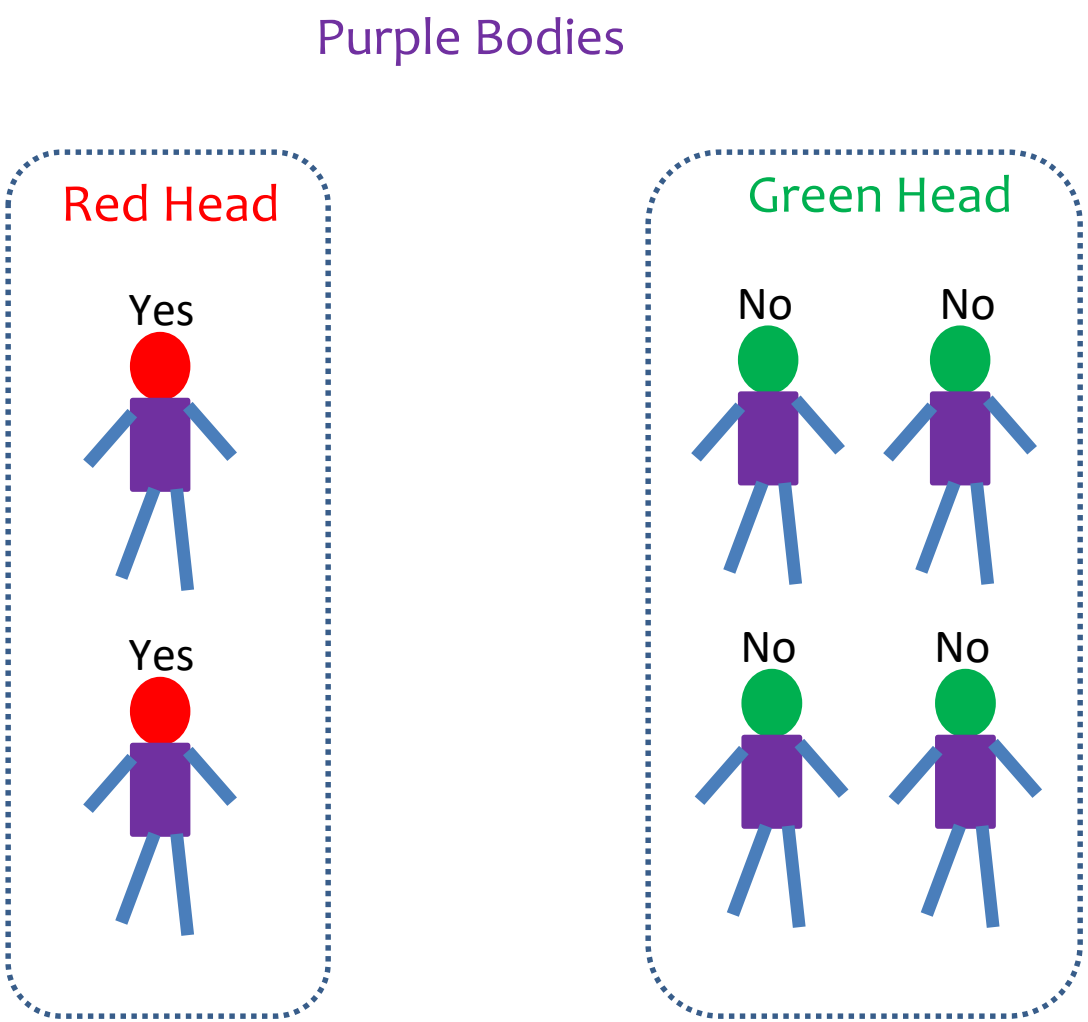


Classification Tree Learning

- Partitioning into "purer" groups recursively

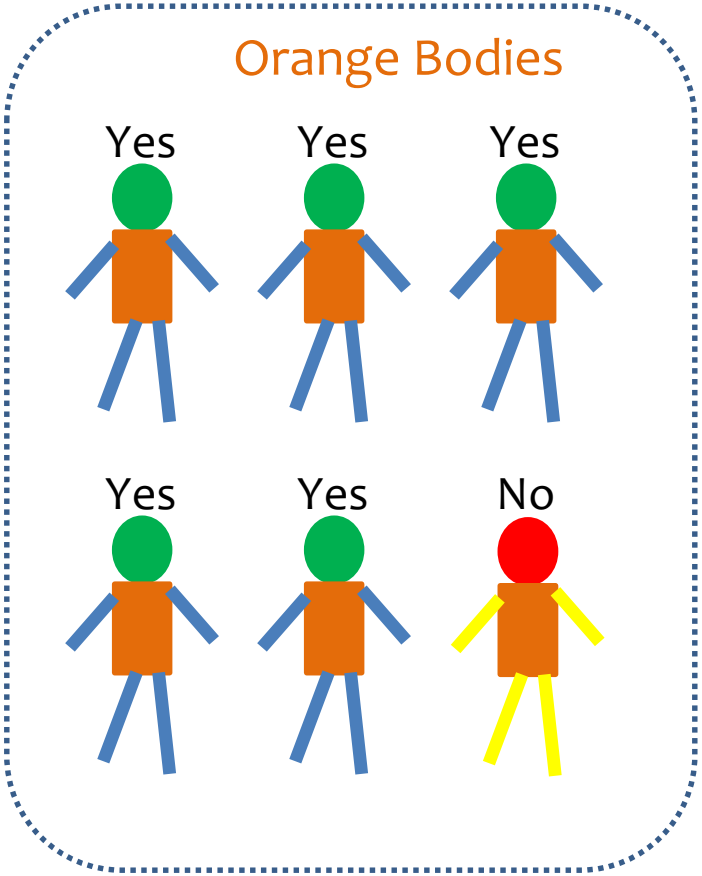


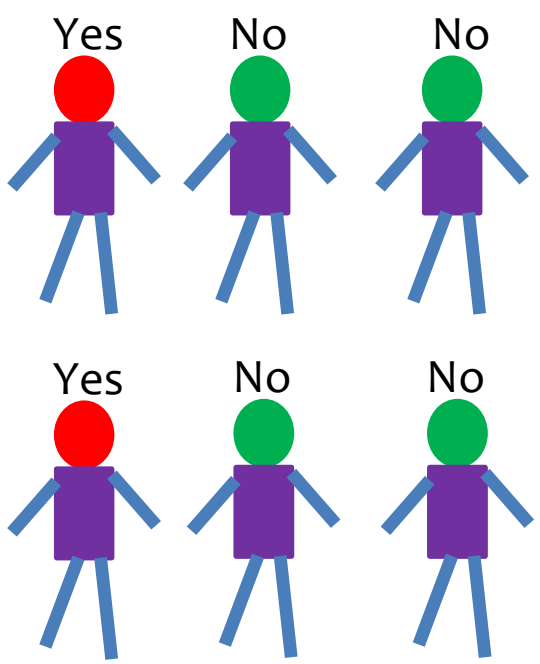
Classification Tree Learning



Classification Tree Learning

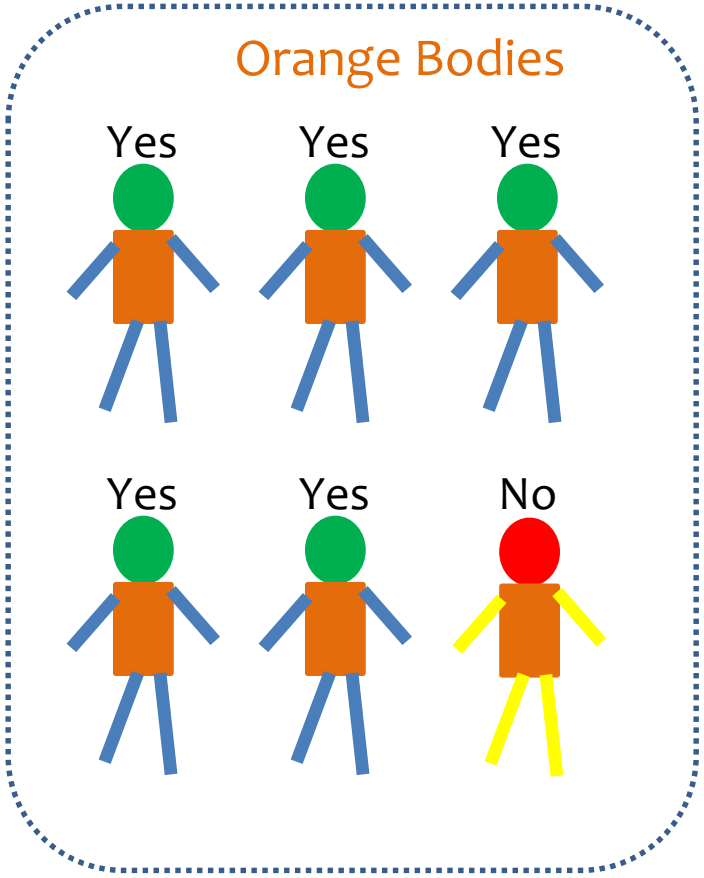
- Partitioning into "purer" groups



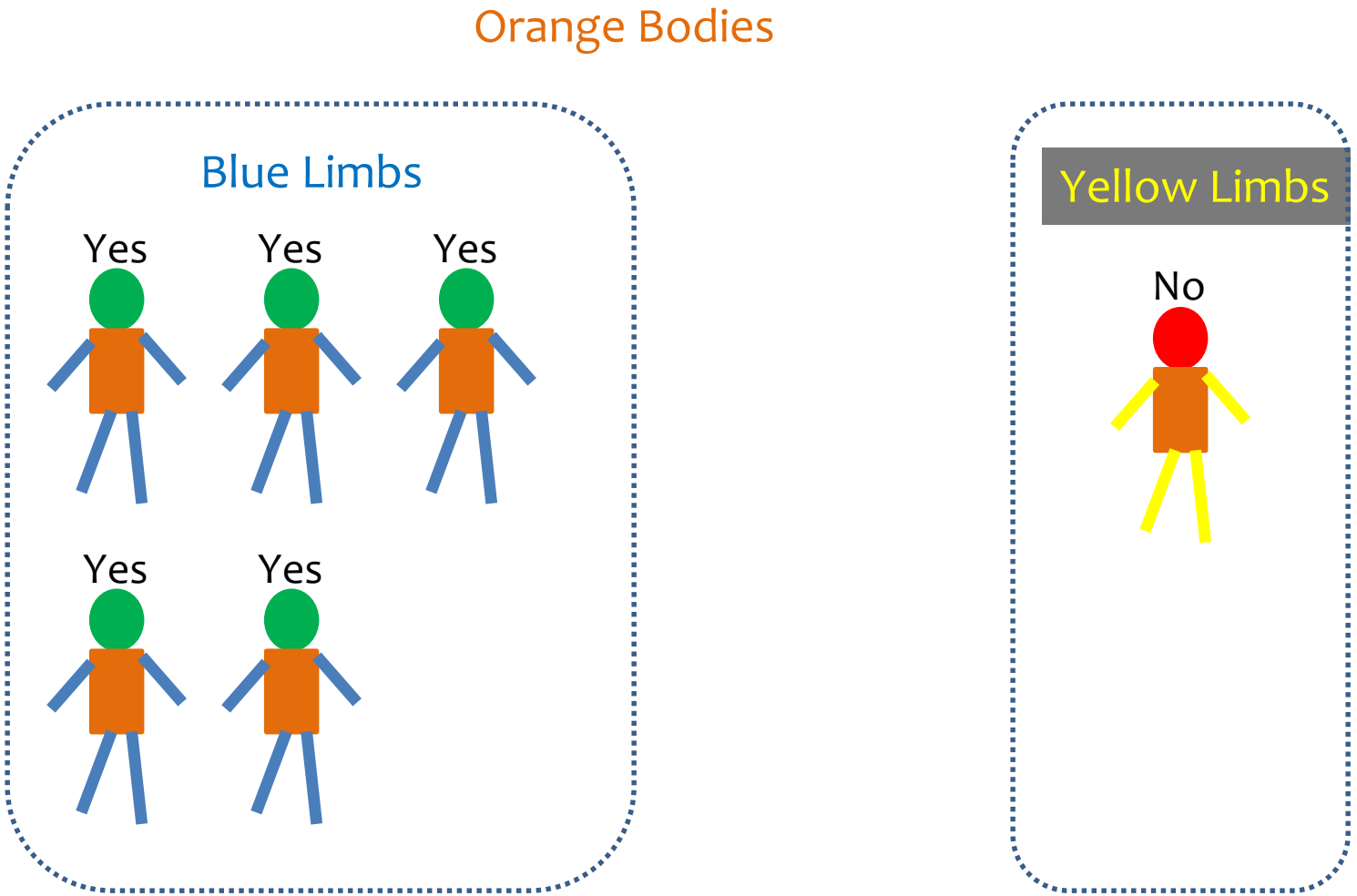


Classification Tree Learning

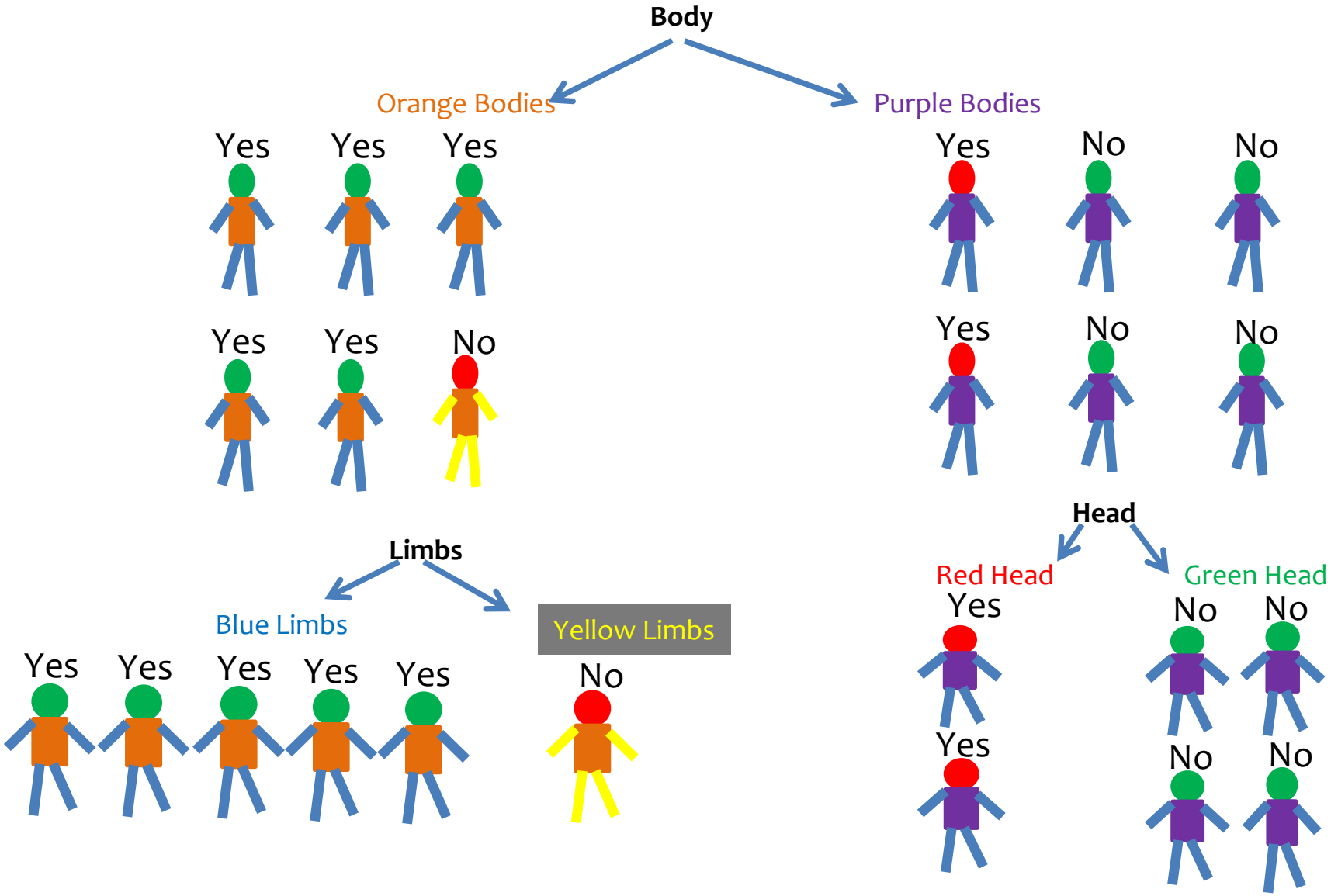
- Partitioning into "purer" groups recursively



Classification Tree Learning

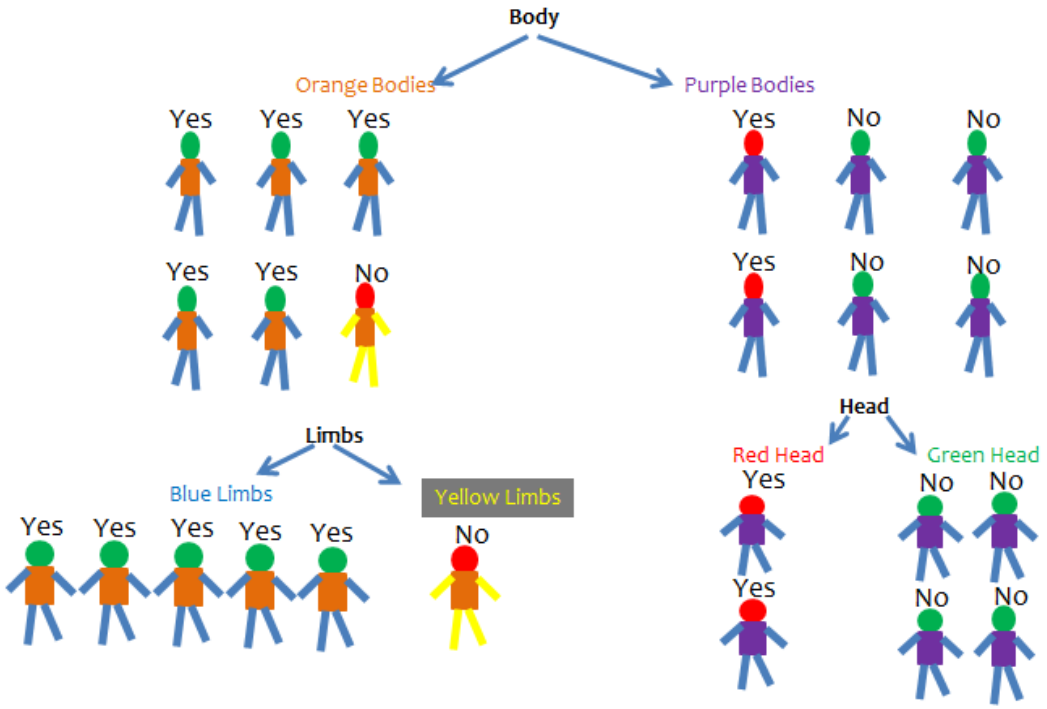


Classification Tree Learning



Summary: Classification Tree Learning

- A tree is constructed by recursively partitioning the examples.
- With each partition, the examples are split into subgroups that are "increasingly pure".



- Let’s play a game. I have someone in my mind, and your job is to guess this person. You can only ask yes/no question.
- This person is an entrepreneur.
- Go!

Next…

- Some important questions without being answered yet:
  - How to automatically choose which attribute to be used to split the data?
  - When to stop the splitting?

How to Choose Which Attribute to Split Over?

- Objectives
  - For each splitting node, choose the attribute that best partitions the population into less impure groups.
  - All else being equal, fewer nodes is better (more comprehensible, easy to use, reduce overfitting)
- Impurity measures: many available but most common one (from information theory) is: entropy.

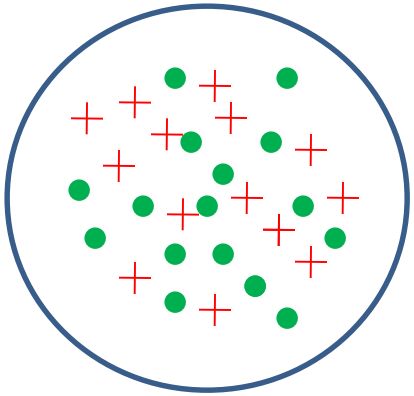
Measure impurity

Entropy

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log_2 p_i$$

p_i is the proportion of class i in the data

High entropy, high uncertainty



- For example: our initial population is composed of 16 cases of class "Default" and 14 cases of class "Not default"
- Entropy (entire population of examples) =

$$-(\frac{14}{30}\log_2\frac{14}{30} + \frac{16}{30}\log_2\frac{16}{30}) = 0.997$$

Entropy Exercise

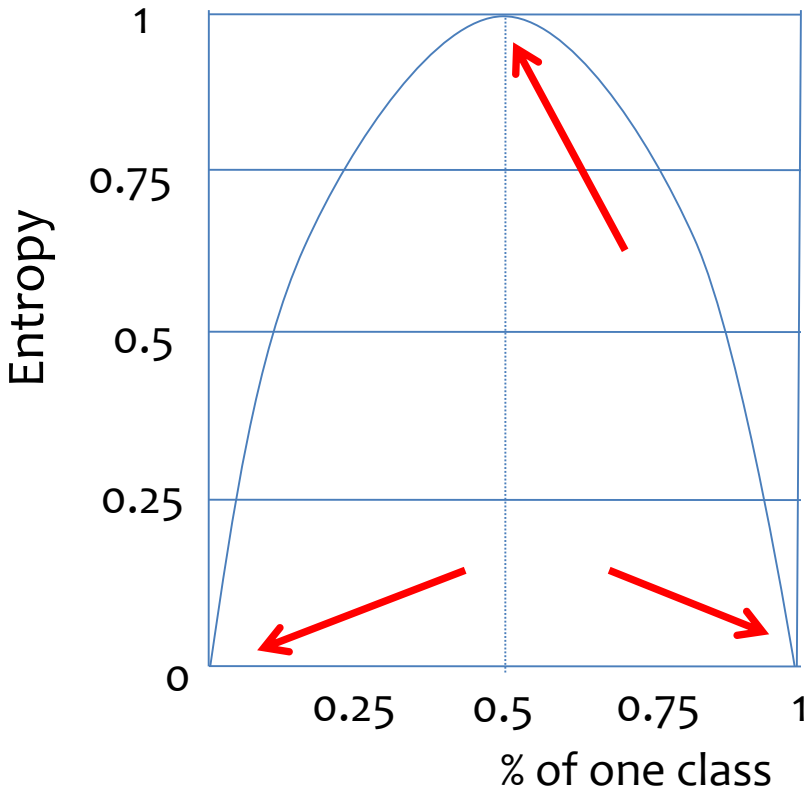
- A dataset is composed of 10 cases of class "Positive" and 10 cases of class "Negative"

  Entropy=? 1

- A dataset is composed of 0 cases of class "Positive" and 20 cases of class "Negative"

  Entropy=? 0

tip: $0 \log_2 0 = 0$



Two-class entropy function

Information Gain (based on Entropy)

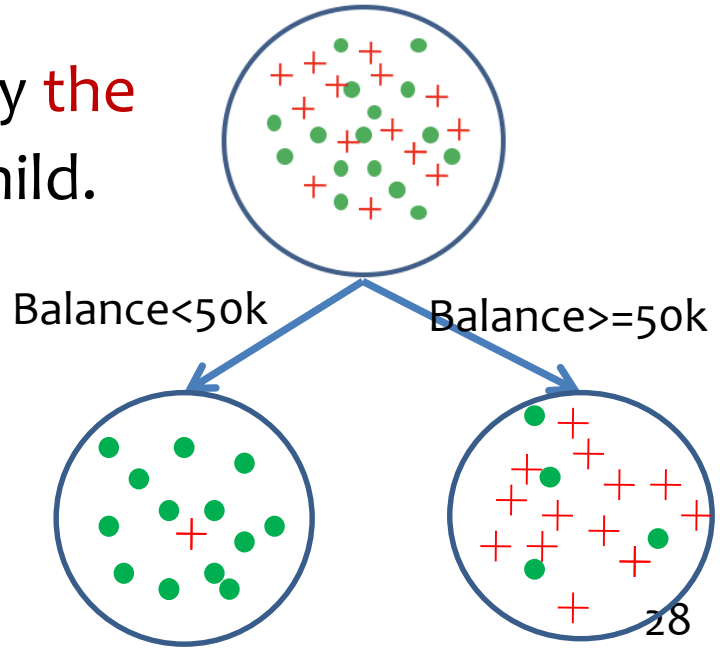
- The information gain is based on the reduction in entropy after a dataset is split on an attribute

Information Gain =

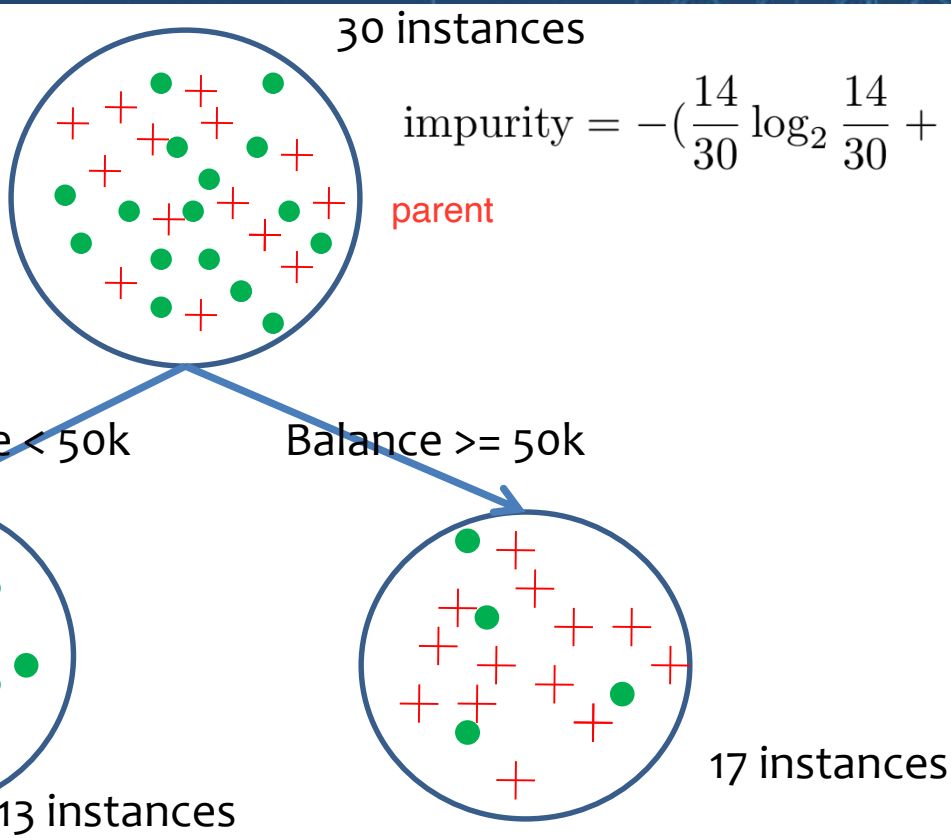
entropy decrease = information gain increase

entropy (parent) – [weighted average] entropy (children)

where weight of each child is given by the proportion of the examples in that child.



Information Gain Example



$$\text{impurity} = -(\frac{14}{30}\log_2\frac{14}{30} + \frac{16}{30}\log_2\frac{16}{30}) = 0.997$$

$$\text{impurity} = -(\frac{12}{13}\log_2\frac{12}{13} + \frac{1}{13}\log_2\frac{1}{13}) = 0.391$$

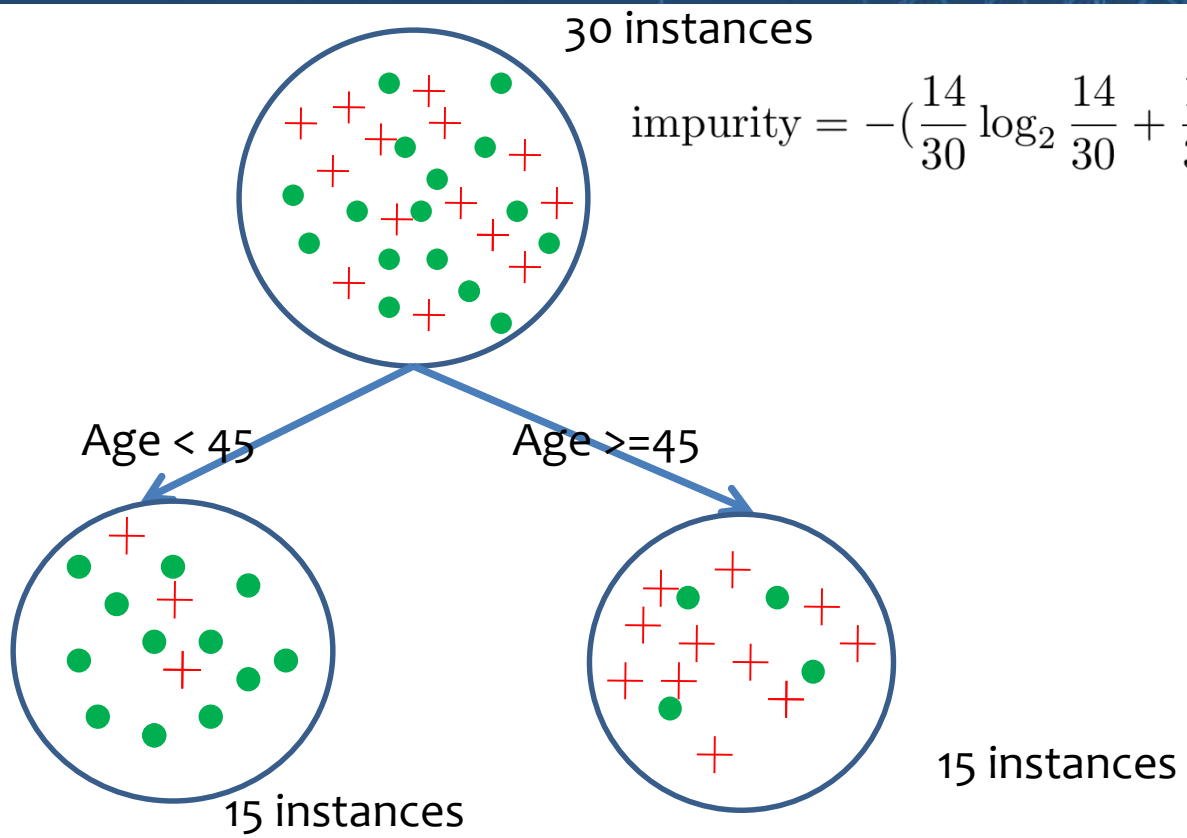
$$\text{impurity} = -(\frac{13}{17}\log_2\frac{13}{17} + \frac{4}{17}\log_2\frac{4}{17}) = 0.787$$

(Weighted) Average Impurity of Children = $(\frac{13}{30} \times 0.391) + (\frac{17}{30} \times 0.787) = 0.615$

Information Gain = 0.997 - 0.615 = 0.382

Information gain
告诉我们split的好不好

What If We Split Over "Age" First?



$$\text{impurity} = -(\frac{14}{30}\log_2\frac{14}{30} + \frac{16}{30}\log_2\frac{16}{30}) = 0.997$$

Impurity = ? 0.72

Impurity = ? 0.84

Exercise!

Information Gain = ? 0.997- (0.72+0.84)/2 = 0.217

recall, gain from first splitting on "Balance" = **0.382**

Our Original Question

- Now, ready to answer anxiously awaiting question: ***How to choose which attribute to split over?***

- Answer: **at each node, choose the attribute that obtains maximum *information gain!***

Decision Tree Algorithm (Full Tree)

- Step 1: Calculate the information gain from splitting over each attribute using the dataset
- Step 2: Split the set into subsets using the attribute for which the information gain is maximum
- Step 3: Make a decision tree node containing that attribute, divide the dataset by its branches and repeat the same process on every node.
- Step 4a: A node with entropy of 0, or cannot partition further, or obtain no information gain from additional split, is a leaf.
- Step 4b: Otherwise, the node needs further splitting.

Decision Trees (II)

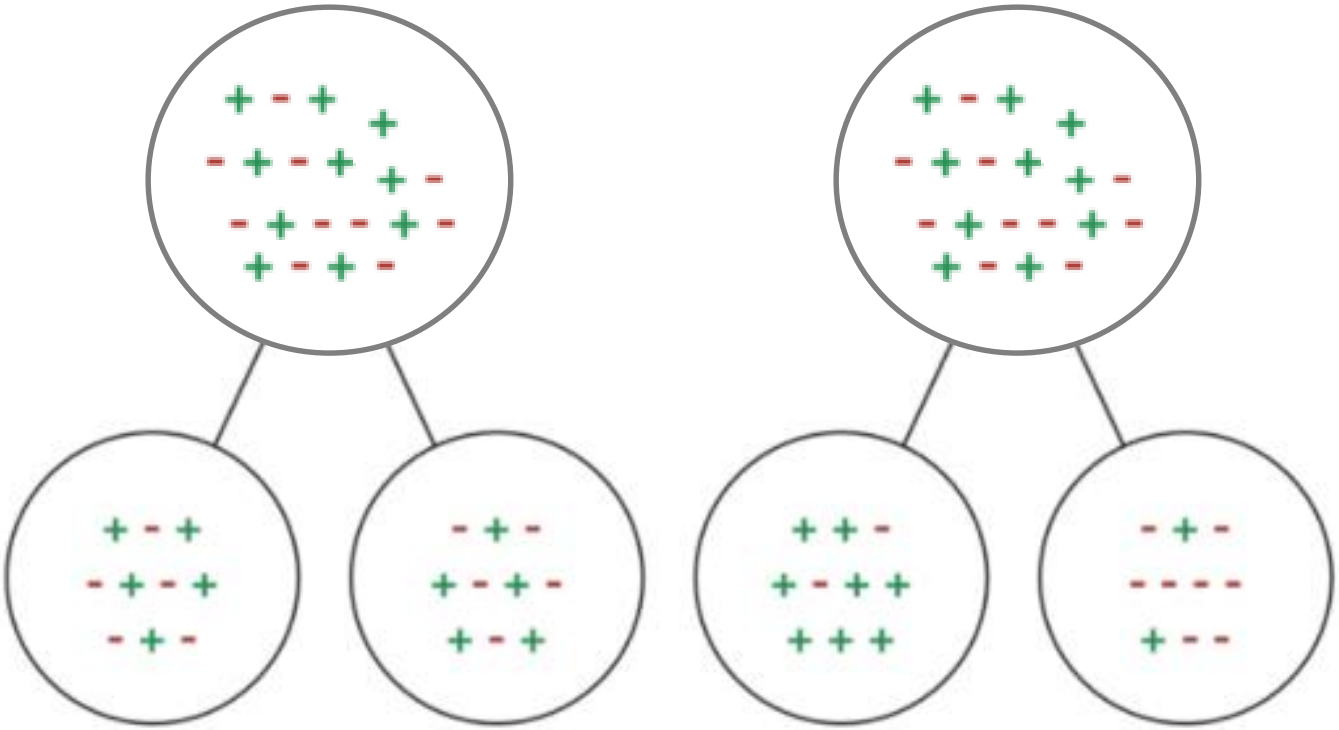
Instructor: Jing Wang

Department of ISOM

Spring 2023

HKUST
BUSINESS SCHOOL
香港科大商學院

Information Gain Exercise



Without calculation, which split (left or right) gives the higher information gain? Right, pure sample

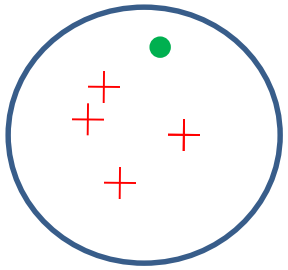
Another Measure: Gini Impurity

- Similar to entropy, the Gini Impurity also measures the impurity of the data.

$$\text{Gini Impurity} = 1 - \sum_{i=1}^{n} p_i^{\,2}$$

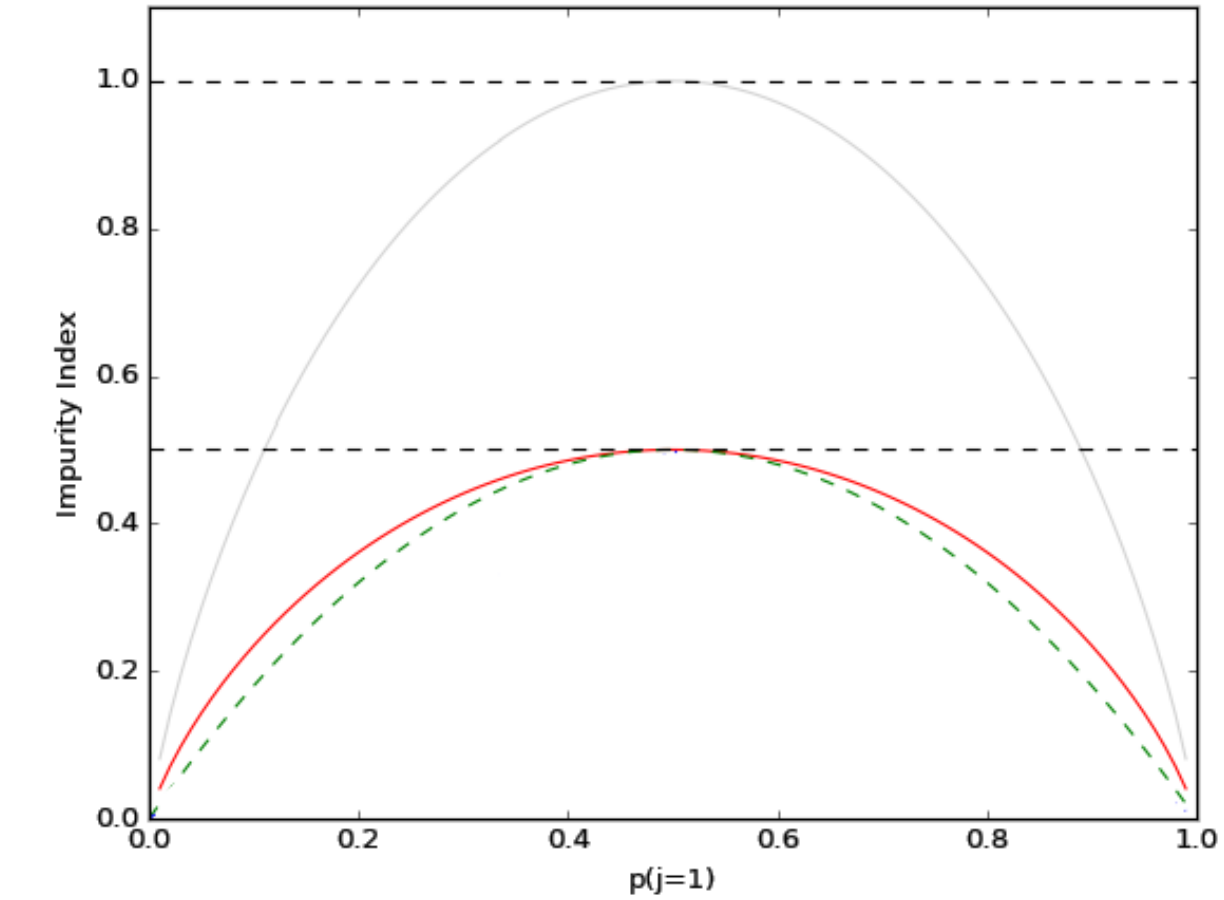
[0,0.5]

p_i is the proportion of class i in the data



Gini impurity = 1- (1/5)^2-(4/5)^2 = 0.32

Gini Impurity vs. Entropy



In terms of calculation, Gini impurity is easier in computation

In terms of performance, we are not sure

Entropy (scaled) = Entropy/2

Information Gain (based on Gini Impurity)

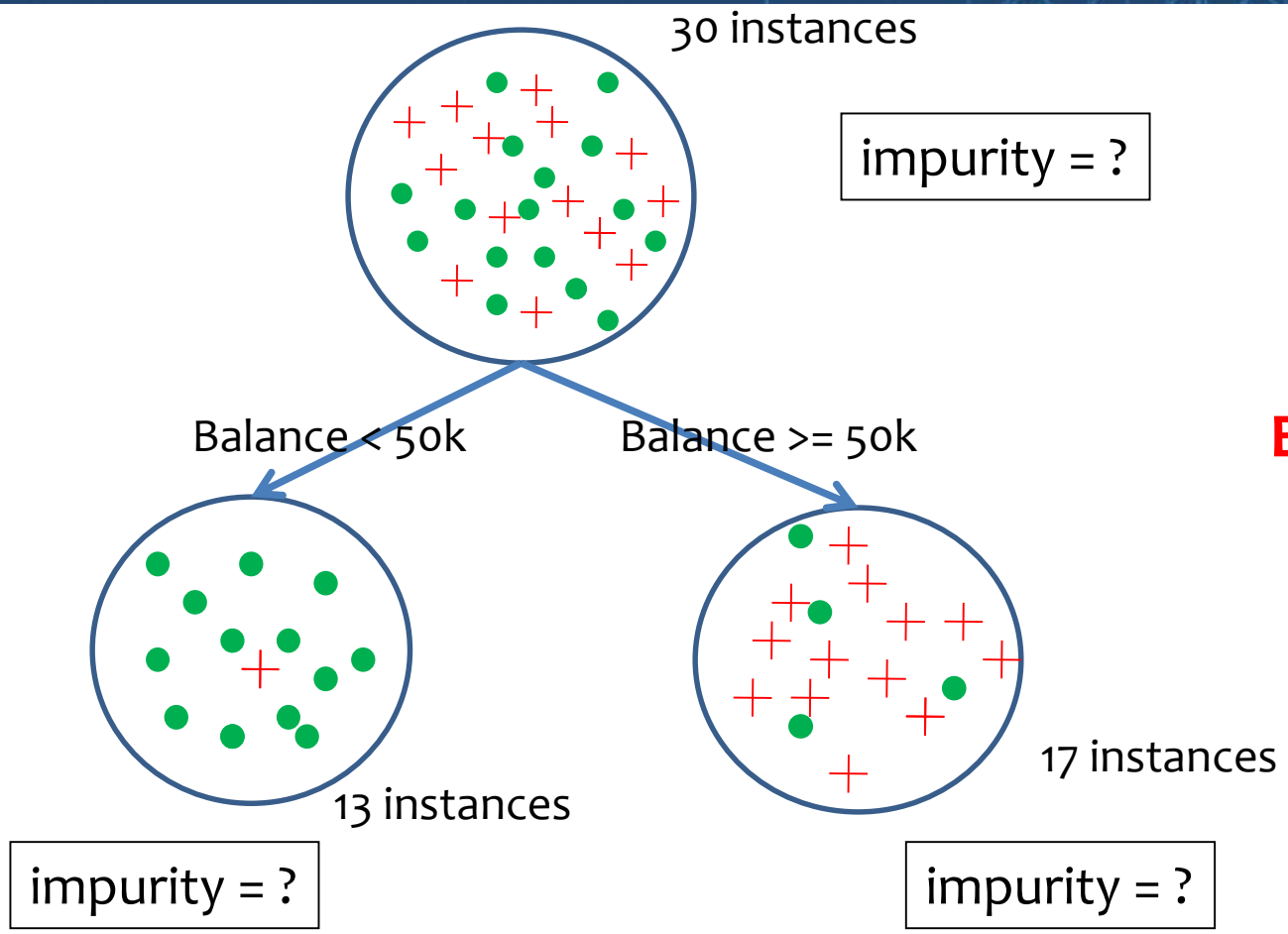
- The information gain (based on Gini impurity) by splitting one attribute is given by:

Information Gain (based on Gini impurity)

= Gini impurity (parent) – [weighted average] Gini impurity (children)

- The attribute that maximizes information gain (based on Gini impurity) is chosen as the splitting attribute.

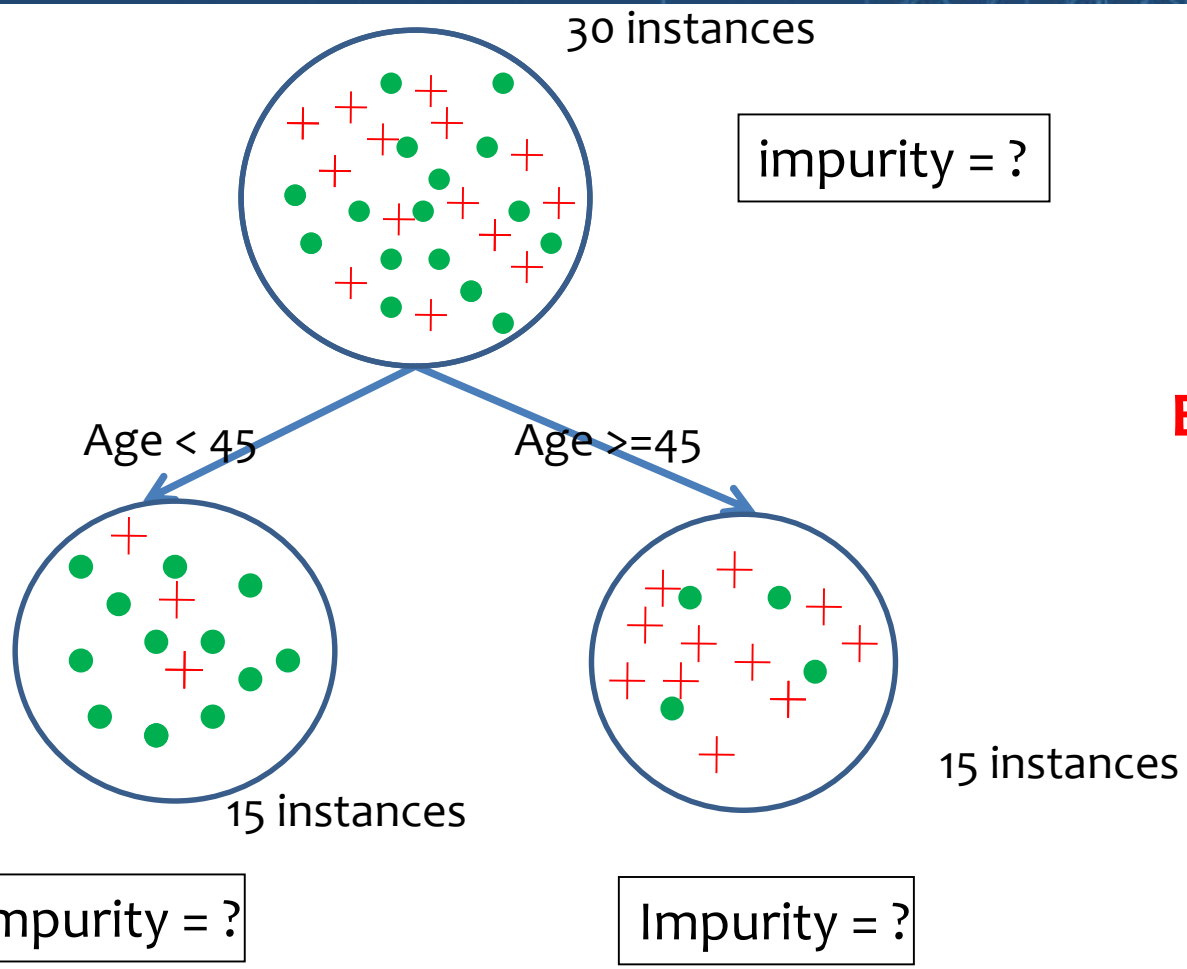
Information Gain (based on Gini Impurity)



Exercise!

Information Gain = ?

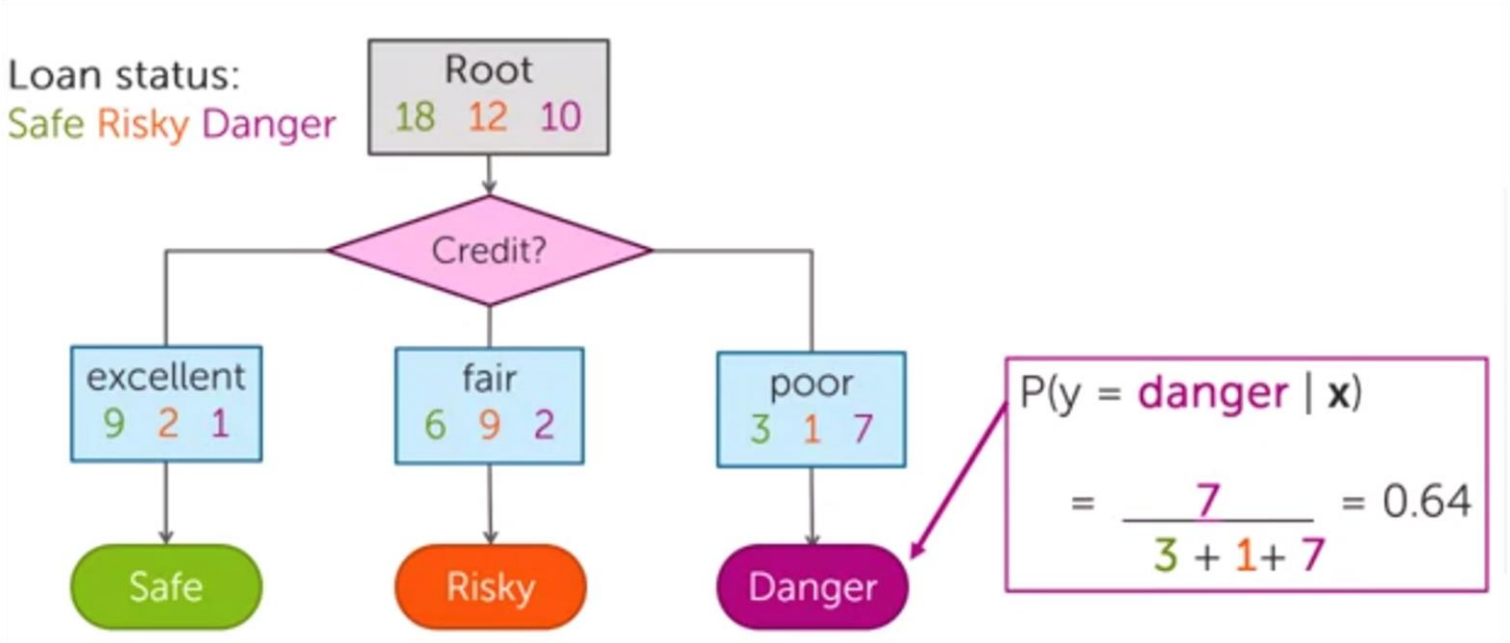
What If We Split Over "Age" First?



Exercise!

Information Gain = ?

What if Outcome Variable Has >2 Categories?



If K (number of classes) >2, same as before, we calculate multi-class entropy or Gini impurity first.

$$\sum_{k=1}^{K} p_k \log_2 p_k$$

$$1 - \sum_{k=1}^{K} p_k^2$$

$$\text{subject to} \sum_{k=1}^{K} p_k = 1$$

What if Outcome Variable is Numeric?

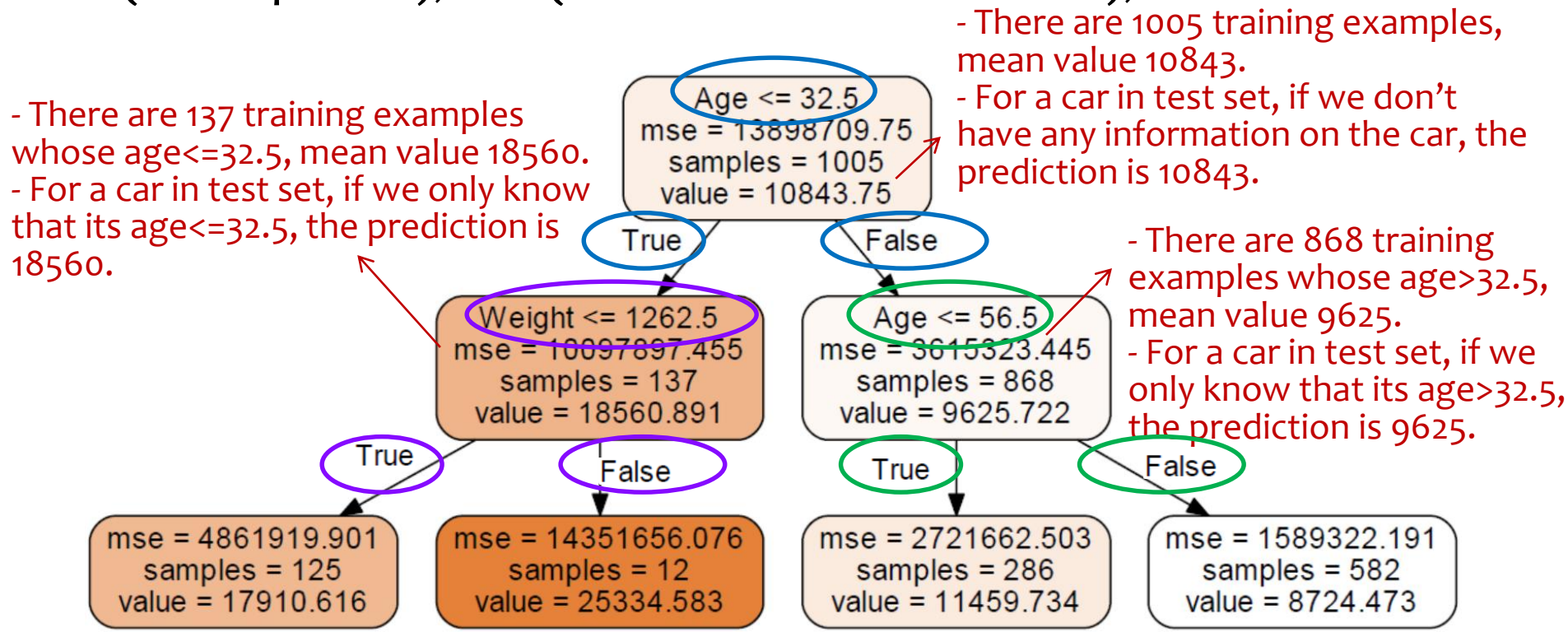
- Regression trees
  - Predicted output = average of the training examples in the subset (leaf)
  - Potential splits are measured on how much they reduce the mean squared error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i is the true value of the target and $\hat{y}_i$ is the predicted value.

Example: A Regression Tree

- Problem: predict the price of used Toyota Corolla cars
- Available attributes: Age (in months), Weight (in kilograms), HP (Horsepower), KM (Accumulated kilometers), …

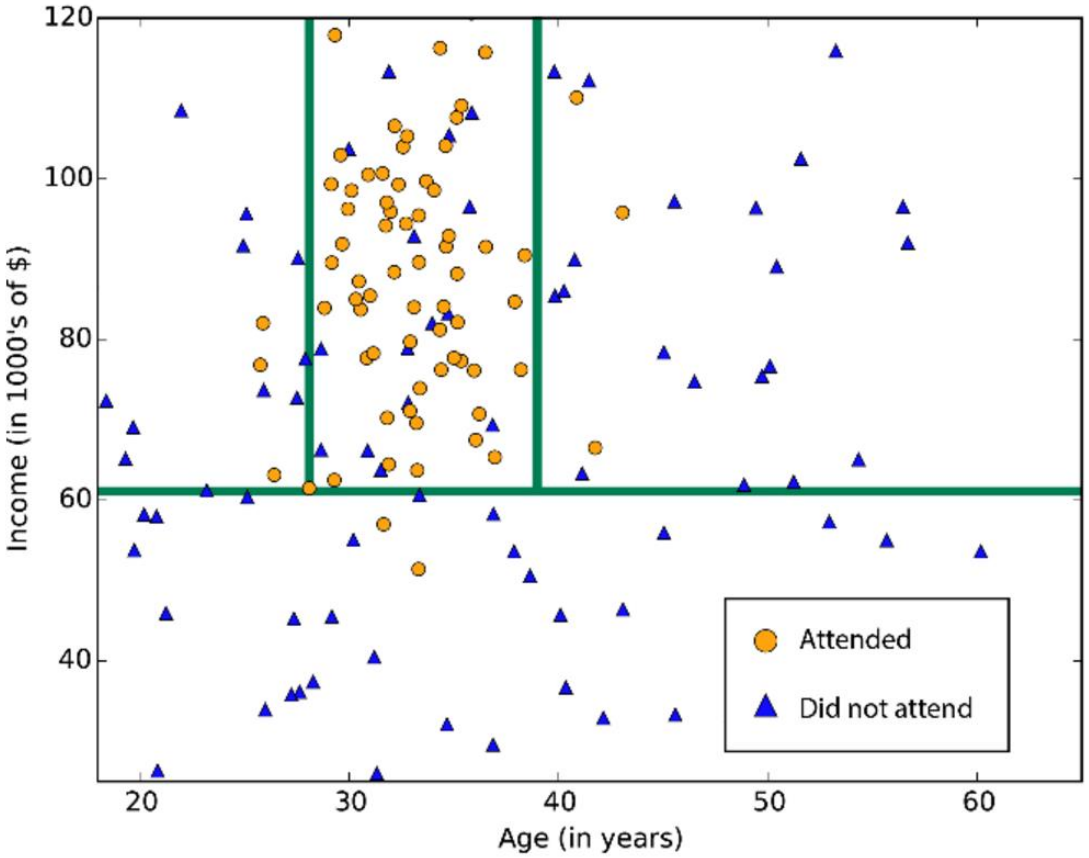


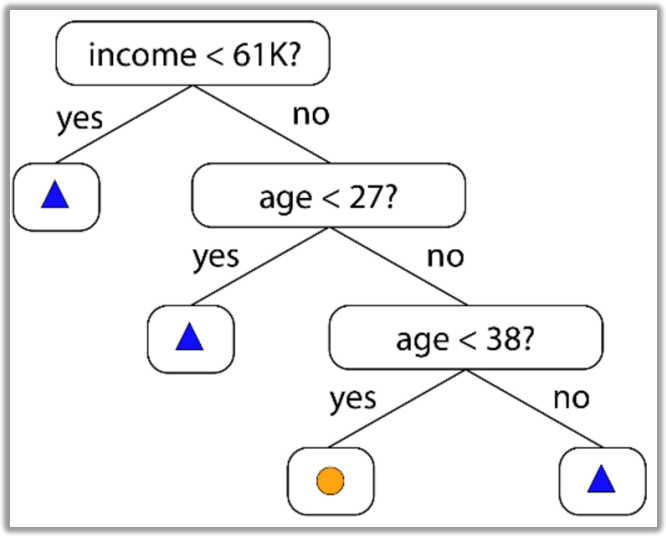
A Note on Handling Continuous Variables

- Handle continuous attribute by splitting into two intervals (can be more) at each node.
- How to find the best threshold to divide?
  - Answer: try them all!!!
  - Use information gain (or other measures) again
  - Sort all the values of an continuous attribute in increasing order {v1, v2, …, vr},
  - One possible threshold between two adjacent values vi and vi+1. Try all possible thresholds and find the one that maximizes the purity

What if we have a lot of unique values (say 100,000)?

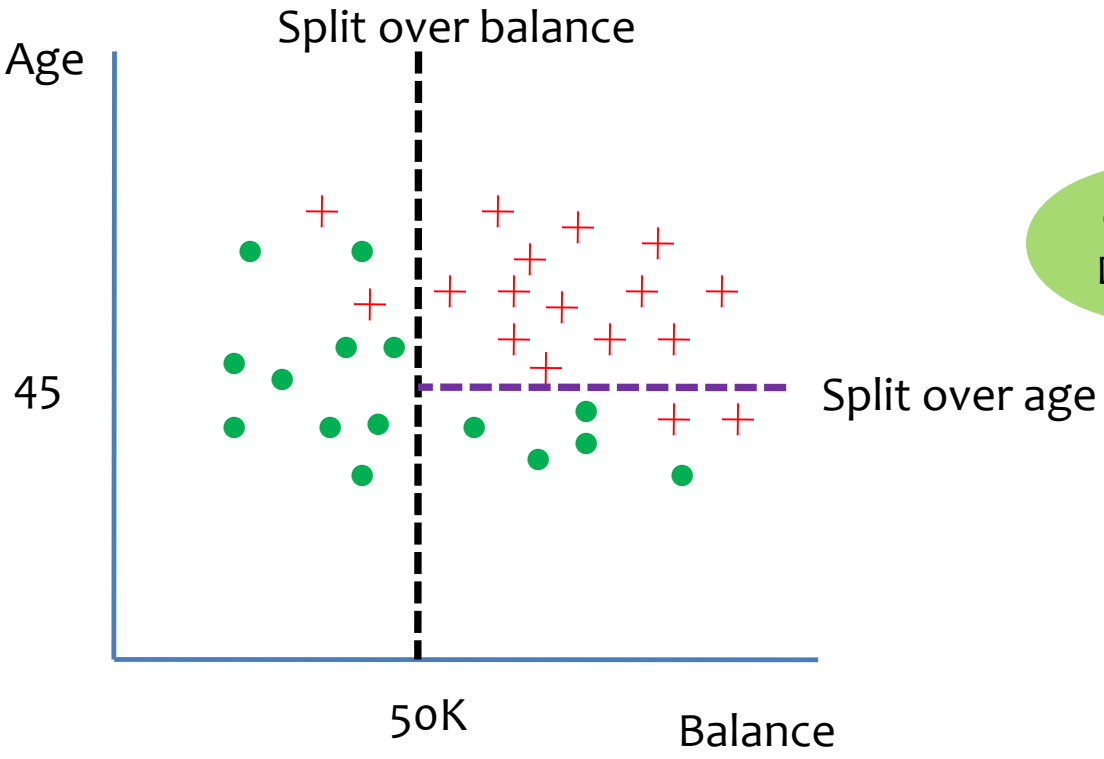
An Example in a Continuous Space

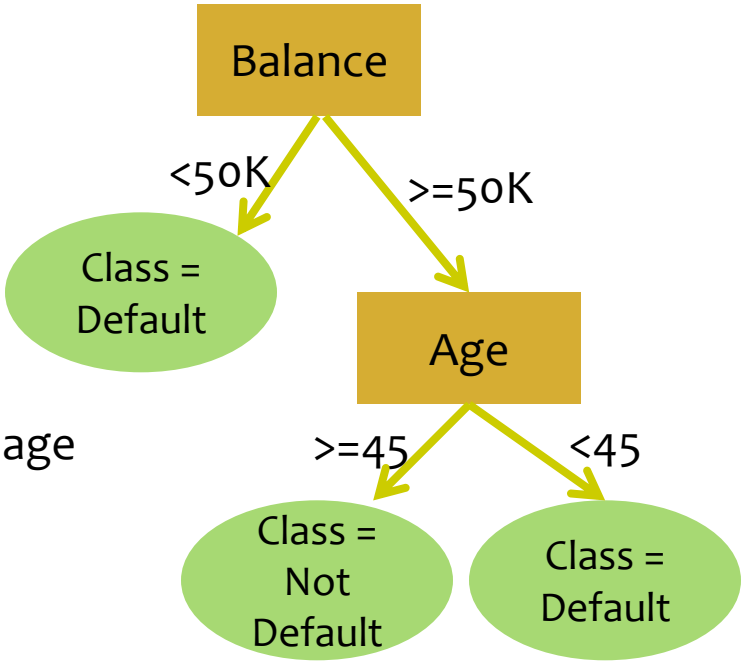




Geometric Interpretation

- Classification tree partitions space of examples with axis-parallel decision boundaries





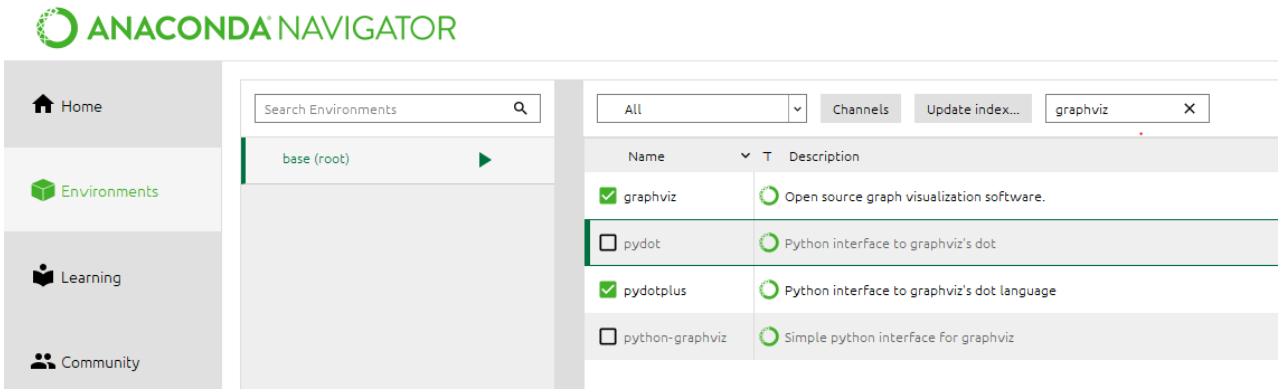
- Bad risk (Default) – 15 cases
- Good risk (Not default) – 17 cases

Lab: Building Decision Tree on Iris Dataset

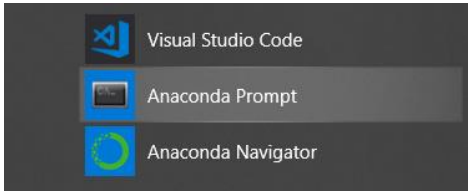
- Files needed
  - Decision Tree-Iris.ipynb (Python file)
  - iris.csv (dataset)

Visualize Trees

- Install new packages of ‘graphviz’ and ‘pydotplus’ in Anaconda



- You may need to launch Anaconda Prompt to do the installation



To install this packages with conda run:

```
conda install -c anaconda graphviz
conda install -c anaconda pydotplus
conda install python-graphviz
```

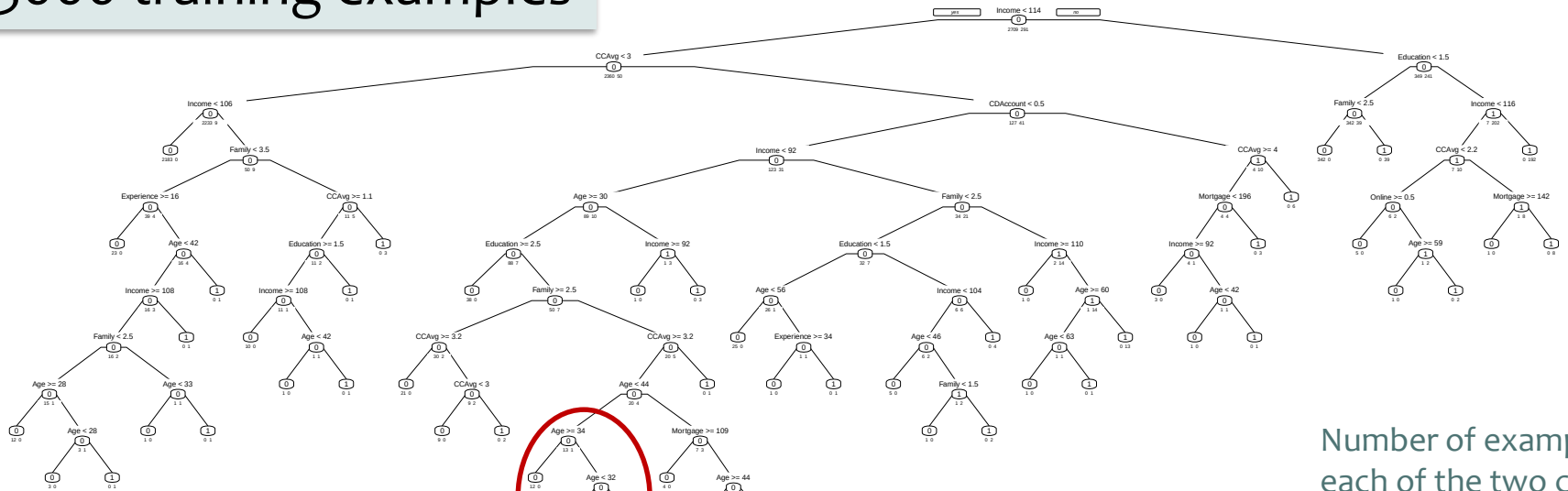
Right Now, We Stop the Partitioning When

- Maximum purity is obtained (i.e., all training examples reaching the node are of the same class)
- There are no remaining attributes for further partitioning (think about two records with exactly same values on explanatory variables but opposite value on target variable -> not possible to achieve maximum purity)
- Additional splits obtain no information gain.

Do we really need to grow the tree fully?

A Fully-Grown Tree

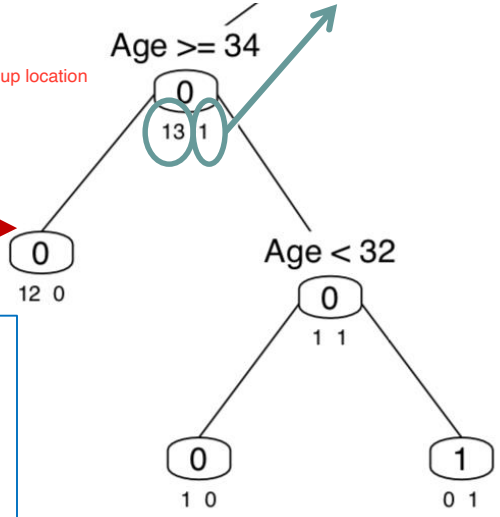
3000 training examples



Number of examples in each of the two classes

0,1 depend by the majority group location
left - majority, 0
right - majority 1

small example in the leaf, not robust
noise raised



A fully-grown tree has training error of close to 0. However, this tree does not learn any pattern, it simply memorize the training data!

A Problem: Overfitting

- A tree may overfit the training data
  - Symptoms: good accuracy on training data but poor on test data
  - Reason:
    - A tree is too deep and have too many branches, some may reflect anomalies due to noise or outliers.

Growing to Purity is Bad (Overfitting)

Humidity	Play Tennis
0.90	0
0.89	1
0.87	0
0.75	0
0.70	1
0.69	1
0.65	1
0.63	1

Hidden information might not be captured

Decent Decision tree rule:

Humidity > 0.70 -> No

otherwise -> Yes

Overfit Decision tree rule:

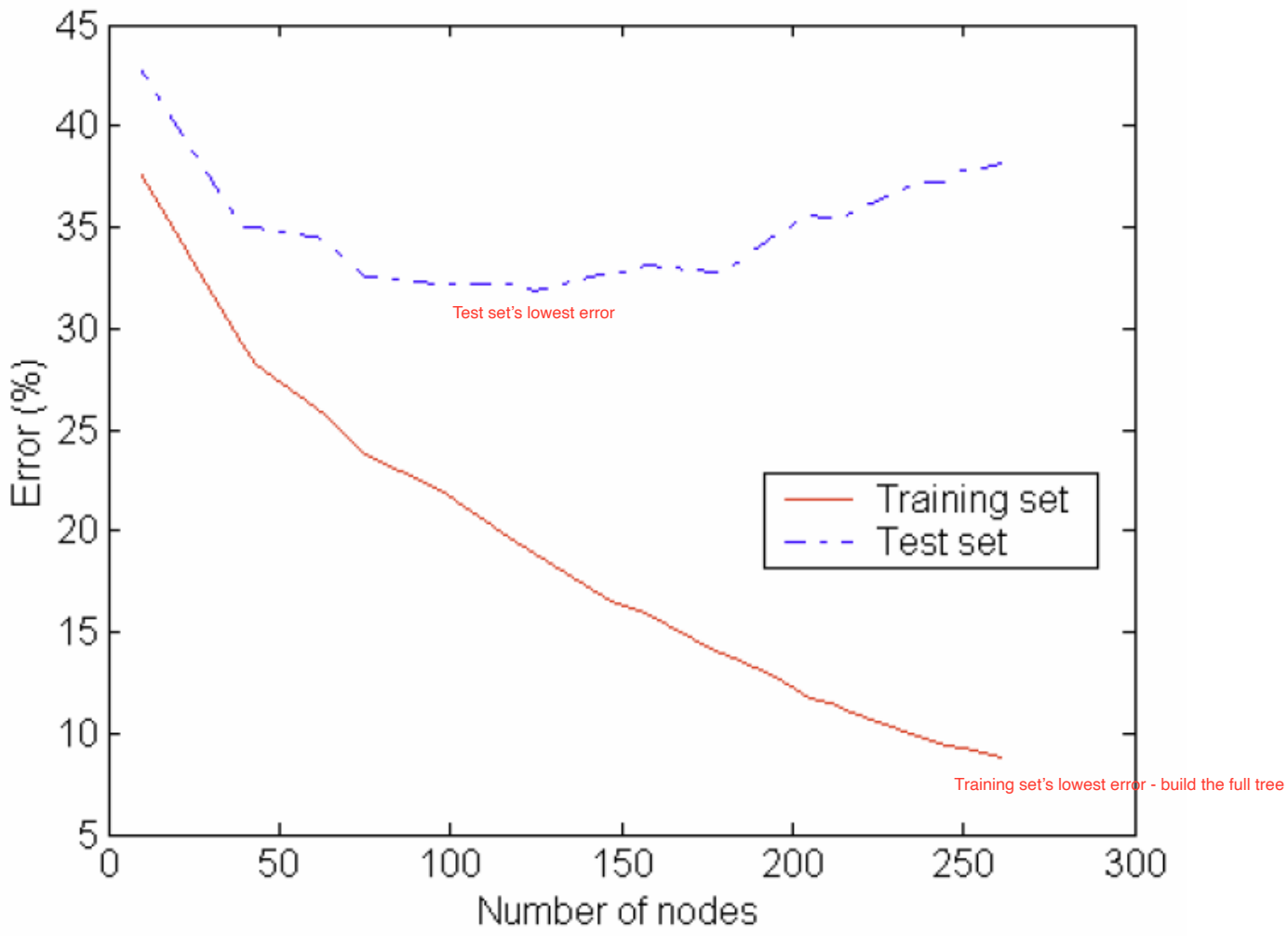
Humidity > 0.89 -> No

Humidity >0.87 AND Humidity <= 0.89 -> Yes

Humidity >0.70 AND Humidity <=0.87 ->No

Humidity <= 0.70 ->Yes

Symptoms of Overfitting



What Is a Good Model?

- A good model must not only fit the training data well but also accurately classify records it has never seen.
- In other words, a good model must have low training error and low generalization error.
  - Training error: the misclassification error rate of the model on training records.
  - Generalization error (can use testing error as a proxy): the expected error rate of the model on previously unseen records.

Overfitting

"Financial professionals must always be aware of the dangers of overfitting a model based on historical data. A common problem is using computer algorithms to search extensive databases of historical market data in order to find patterns. Given enough study, it is often possible to develop elaborate theorems which appear to predict things such as returns in the stock market with close accuracy. However, when applied to data outside of the sample, such theorems may likely prove to be merely an overfitting of a model to what were in reality just chance occurrences. In all cases, it is important to test a model against data which is outside of the sample used to develop it. "

--------Investopedia

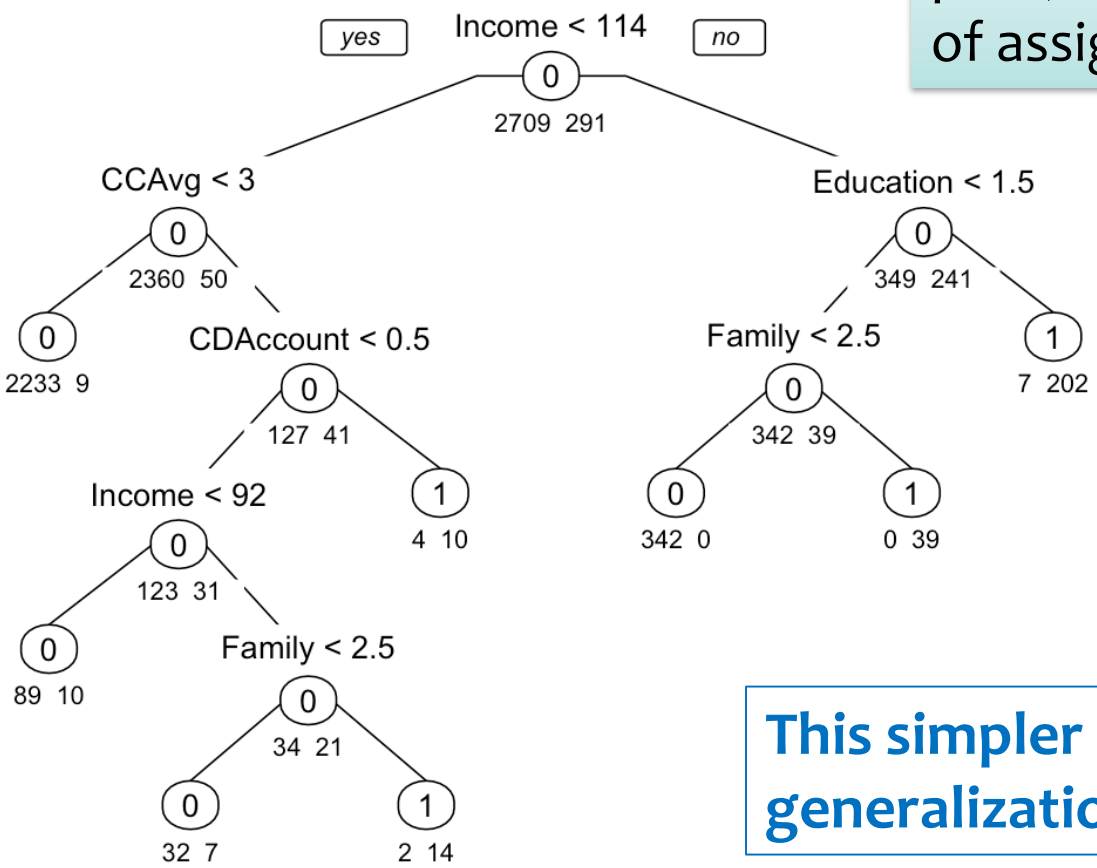
Occam's Razor

- *"Everything should be made as simple as possible, but no simpler."*
- All other things being equal, simple models are preferable to complex ones.
  - A simple model that fits the data is unlikely to be a coincidence.
  - A complex hypothesis that fits the data might be a coincidence.

A Simpler Tree (Leaf Nodes are Not Pure)

3000 training examples

Leaf nodes, which are not pure, represent the probability of assigning class labels.



This simpler tree may have better generalization performance.

How to Avoid Overfitting for Decision Trees?

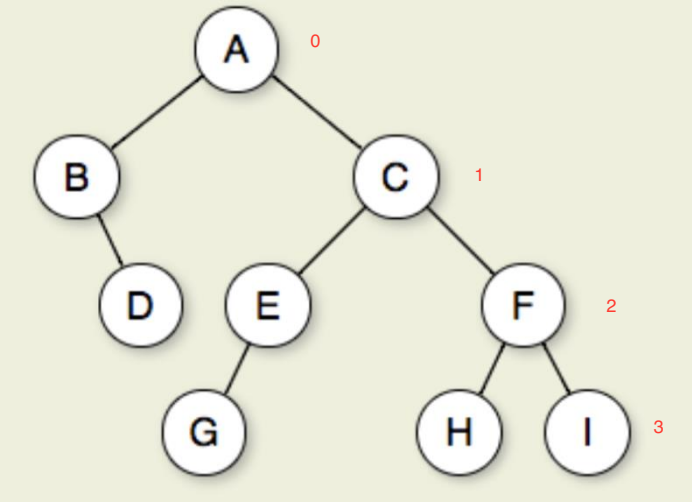
- **Pruning**: replace a sub-tree with a leaf and assign the most popular class to the leaf.
- Approach 1: Pre-pruning -> early stopping.
- Approach 2: Post-pruning -> grow full tree first, then prune the nodes of the decision tree afterwards.

Pre-Pruning: Early Stopping

- Stop growing if
  - the depth of the tree is higher than some user-specified threshold
  - the number of instances is less than some user-specified threshold
- Very simple, not data-driven, but can incorporate domain knowledge.

Depth of a Tree

- Depth of tree
  - The depth of a node is the number of edges from the node to the tree's root node.
  - A root node will have a depth of 0.
  - The depth of a tree is the depth of its deepest node.

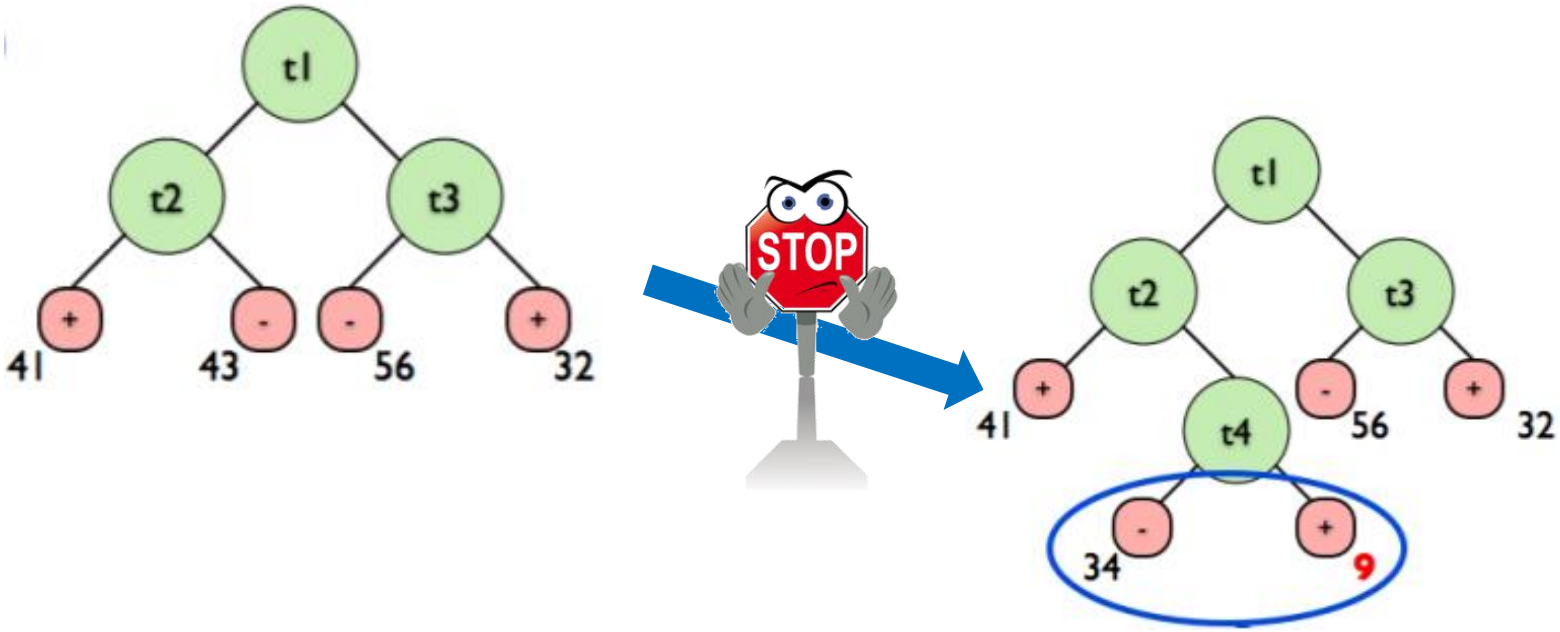


Q: What is the depth of this tree?

3

Limiting Minimal Number of Instances

- Minimal number of instances: 30

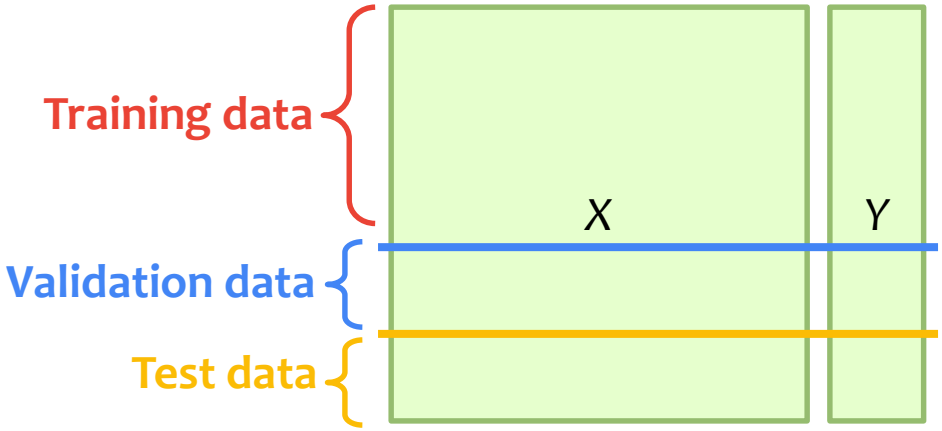


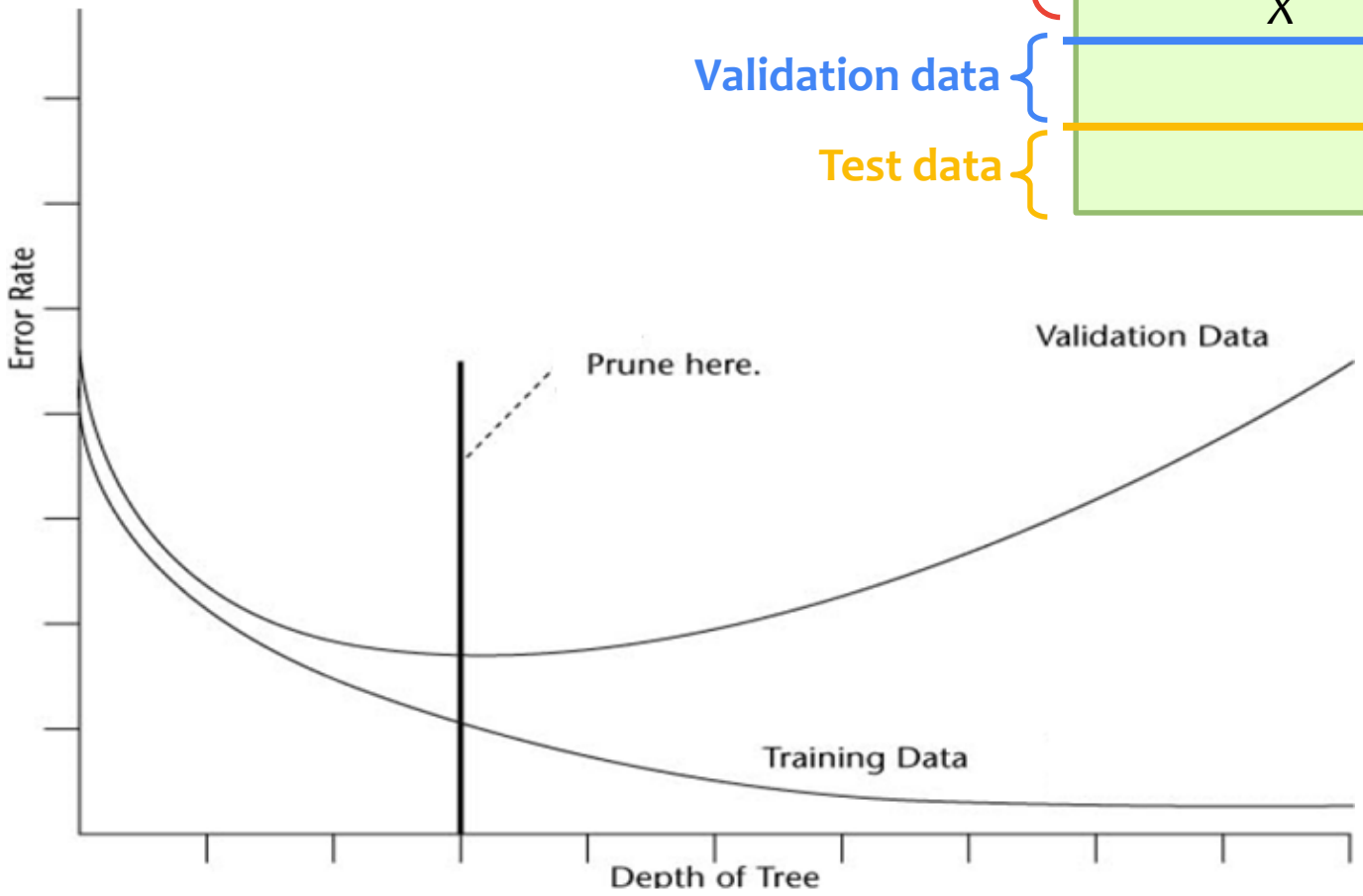
Post-Pruning (Optional)

- Reduced error pruning
  - Split training data into a training and a validation set
    - Validation data is just for parameter tuning, NOT for testing!
  - Grow a full tree on training set
  - Trim the nodes of the decision tree in a bottom-up fashion
    - Eliminate subtree if the resulting pruned tree performs no worse than the original over the validation set.

** Not currently supported by scikit-learn library.*

Reduced Error Pruning





Lab: Overfitting and Pre-pruning (Tree Depth)

- Files needed
  - Overfitting.ipynb (Python file)
  - titanic_train.csv (dataset)

Pros and Cons of Decision Tree Learning

- Pros
  - Simple to understand and interpret
  - Require little data preparation (does not need data normalization for numerical variables, one-hot encoding for categorical variables, etc.)

<but need to do one-hot-encoding if using scikit-learn library>

- Cons
  - Can overfit, thus need pruning steps
  - Tree structure is not very stable*

*A learning algorithm is said to be unstable if it is sensitive to small changes in the training data.

Model Evaluation

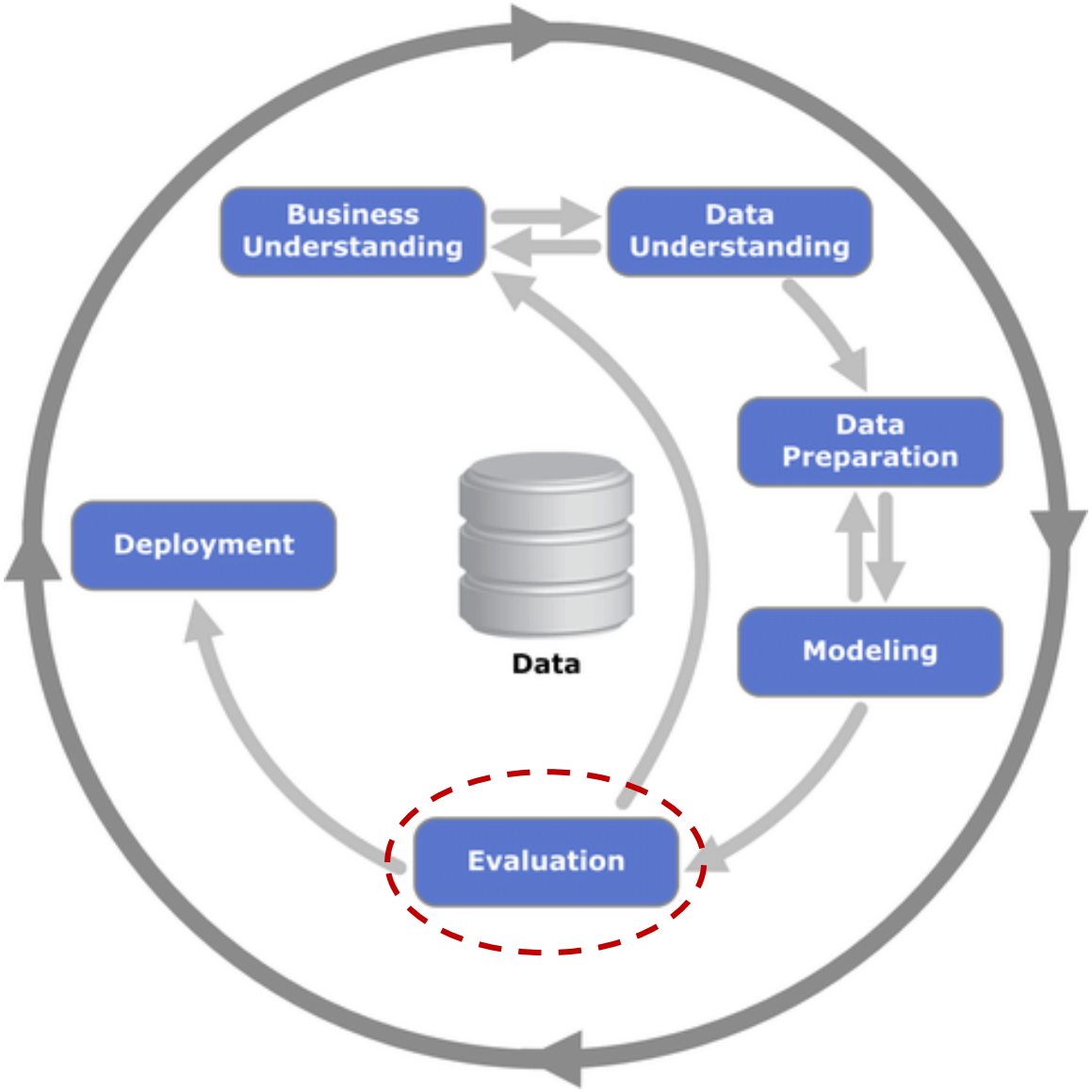
Instructor: Jing Wang

Department of ISOM

Spring 2023

HKUST
BUSINESS SCHOOL
香港科大商學院

Evaluation

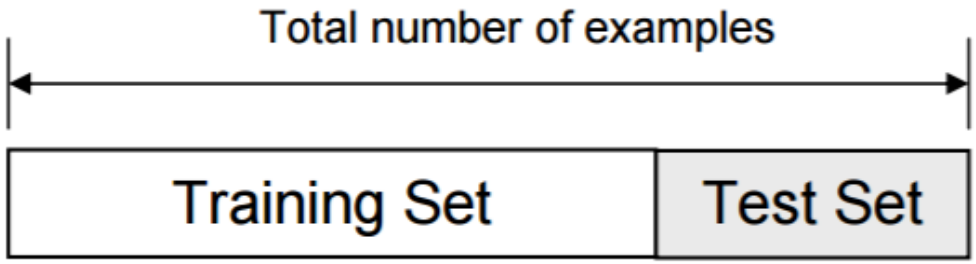

Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data

Evaluation

- We are interested in generalization – *the performance on data not used for training*
  - Holdout validation (one split)
  - *k*-fold cross validation (*k* splits)

Holdout Validation

- Split data into training set and test set (e.g., 70% training set, 30% test set)
- Use training set to train the model
- Use the test set to estimate the error rate of the model



Overestimate/Underestimate

1) What if by accident you selected a particularly easy/hard test set?

2) Do you have an idea of the variation in model accuracy due to the split?

Drawbacks of the Simple Holdout Method

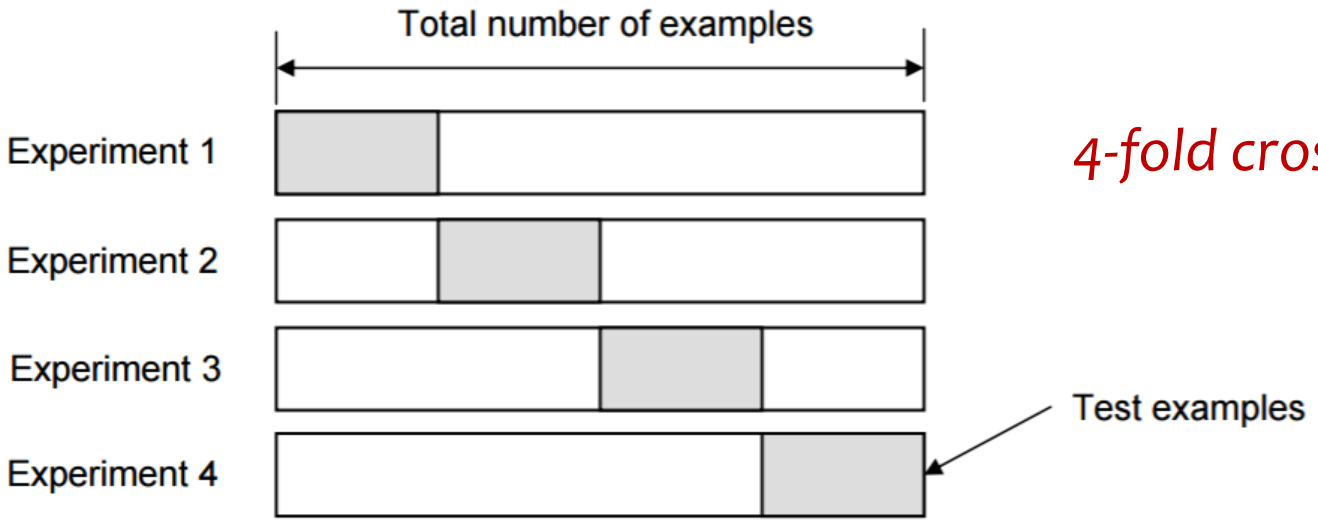
- The holdout estimate of the error can be highly variable across different splits.
  - Since it is a single train-and-test experiment, the holdout estimate of error rate will be misleading if we happen to get an "abnormal" split (e.g., very difficult test set).
  - This is particularly true when we have a small dataset.

Solution: k-fold cross validation!

k-Fold Cross-Validation

- Partition data into *k* folds (randomly) Split K times
- Run experiments *k* times; for each of k experiments, use *k*-1 folds for training and a different fold for testing

3 times



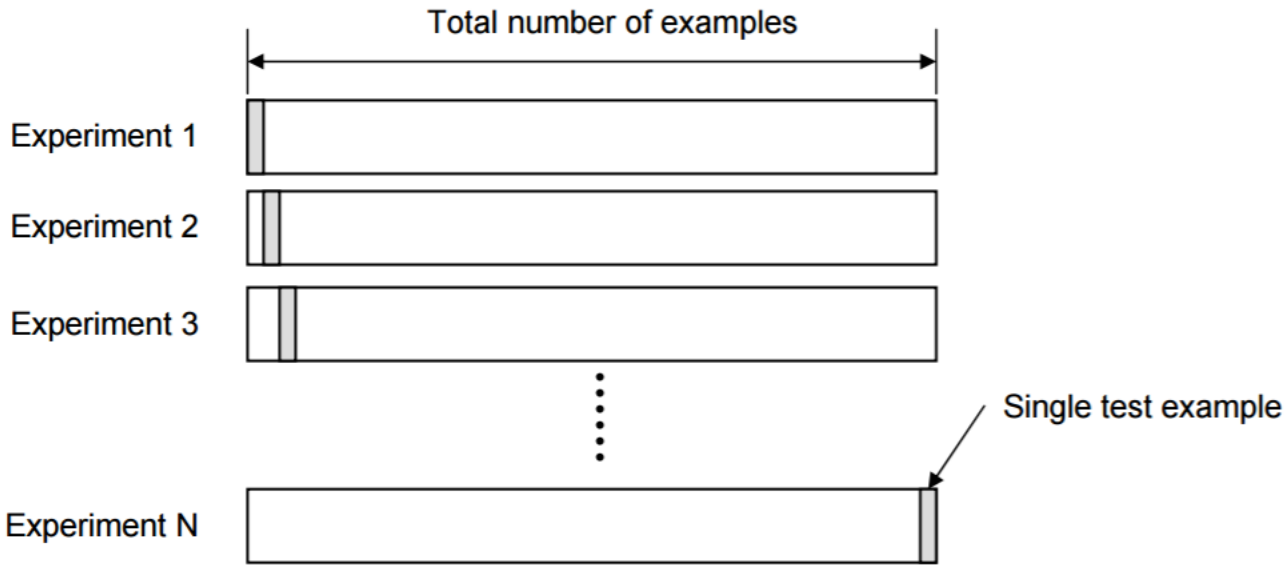
4-fold cross validation

--- Each fold is test set once (test on all data)

--- Can compute average and variance of accuracy measures

Leave-One-Out Cross-validation

- Leave-one-out is a special case of *k*-fold cross validation, where *k* is chosen as the total number of examples.
  - For a dataset with N examples, perform N experiments
  - For each experiment use N-1 examples for training and the remaining example for testing



How Many Folds Are Needed?

- In practice, the choice of the number of folds depends on the size of the dataset
  - For large datasets, even 3-fold cross validation will be quite accurate
  - For very sparse/small datasets, we may have to use more folds in order to train on as many examples as possible (e.g., k=N, leave-one-out cross validation)
- A common choice for k-fold cross validation is k=10

k-Fold Cross-Validation

Under k-fold cross-validation, we will get k decision trees. Which one should we use for deployment?

The purpose of cross-validation is not to come up with the final model. The only use of cross-validation is model evaluation, i.e., for us to get an understanding of how well the model can perform for unseen examples.

Estimate performance

For the deployment, we can use all the examples available to get the best model possible (more data-> better model performance).

Not use any tree to do the deployment

Learner

- A learner or inducer or algorithm
  - A method or algorithm used to generalize a model from a set of examples.

Name	Balance	Age	Default
Mike	123,000	50	No
Mary	51,100	40	Yes
Bill	68,000	55	No
Jim	74,000	46	Yes
Dave	23,000	44	No
Anne	100,000	50	Yes

Learner: induces a pattern from examples

IF Balance >= 50K AND Age > 45
THEN Default = 'no'
ELSE Default = 'yes'

In practice, people use model and learner interchangeably. But they are different.

Model/Learner/Inducer/Algorithm Evaluation

In practice, people use model and learner interchangeably. But they are different.

By saying “model evaluation”, we essentially want to evaluate the performance of the algorithm (with specific hyperparameter values) in the prediction, instead of the performance of any specific model.

Lab: Cross Validation

- Files needed
  - CrossValidation.ipynb (Python file)
  - titanic_cleaned.csv (dataset)

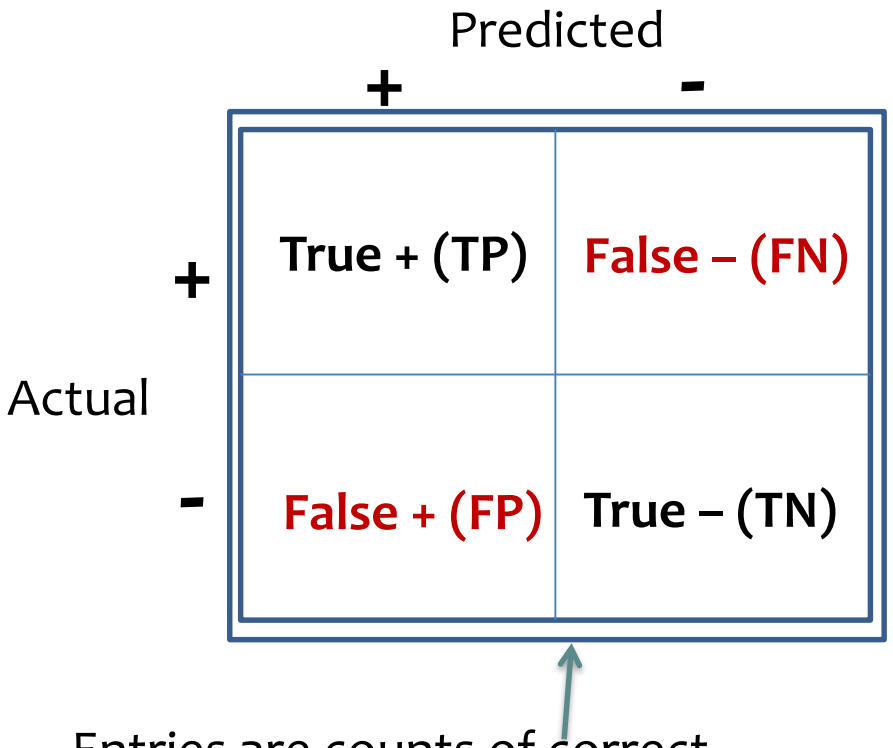
Evaluation Classification Performance

- Naïve rule: majority-class classifier (benchmark model)
  - Classifying everyone as belonging to the majority class.
  - Is used as a baseline or benchmark for evaluating the performance of more complicated classifiers.

Consider direct mail marketing. Suppose the offer is only accepted by 1% of the households. A majority-class classifier simply classifies every household as a non-responder.

Confusion (Classification) Matrix

- interest in test set
  Confusion matrix: a table that is often used to describe the performance of a classification model. Entries are counts of correct and incorrect predictions.



Entries are counts of correct classifications and counts of errors

	Predicted (+)	Predicted (-)
Actual (+)	180	20
Actual (-)	10	90

1. How many records are labeled with + in test set? How many records are labeled with -? 200 positive, 100 negative
2. How many records in test set? 300
3. How many records are correctly predicted? How many records are incorrectly predicted? 270 correct, 30 incorrect

Evaluation Measures for Classification

- Accuracy/Error rate
  - Accuracy: the percentage of correct predictions

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of instances in test set}}$$

  - Error rate: the percentage of incorrect predictions

$$\text{Error rate} = \frac{\text{Number of incorrect predictions}}{\text{Total number of instances in test set}}$$

Accuracy Measure is Limited

- Consider direct mail marketing. Suppose the offer is only accepted by 1% of the households. We build two models, one by decision tree, the other one by majority-class. Which model has higher accuracy?

Decision tree

	Predicted (+)	Predicted (-)
Actual (+)	8	2
Actual (-)	20	970

accuracy = 978/1000

Majority-class

	Predicted (+)	Predicted (-)
Actual (+)	0	10
Actual (-)	0	990

accuracy = 990/1000

+: responder
-: non-responder

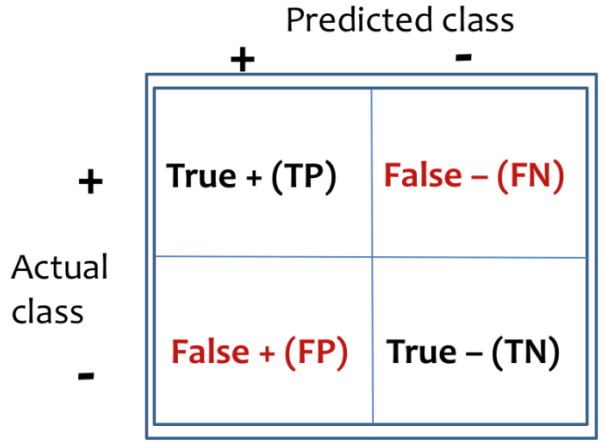
Precision and Recall Measures

- **Precision** is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.
- **Recall** is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

ignores false negative

ignores false positive



Tradeoff between Precision and Recall

- **Precision** = Of all the fishes you caught, what % of them are red fish?

  # red fishes / (# red fishes + # blue fishes)

- **Recall** = Of all the red fishes, what % did you catch?

  # red fishes you caught / # all red fishes

There is an inverse relationship between precision and recall, where it is possible to increase one at the cost of reducing the other.

Exercise

- Still the direct mail marketing example. What is the precision and recall for both models?

Decision tree

	Predicted (+)	Predicted (-)
Actual (+)	8	2
Actual (-)	20	970

Majority-class

	Predicted (+)	Predicted (-)
Actual (+)	0	10
Actual (-)	0	990

Precision: ? 8/28
Recall: ? TP/(TP+FN) = 8/10

+: responder
-: non-responder

Precision: ? Text
Recall: ?

Exercise

Which one is more important for YouTube recommendation, precision or recall?

Precision is more important - Precision: how many recommendation is relevant
Recall: all the targeted audience you want to attract, how many

Which one is more important for COVID-19 detection, precision or recall?

low recall: people have covid but not being captured
low precision: people not have cancer get tested positie

Cost-Sensitive Classification

- What we have discussed so far assumes errors have the same cost.
- But in many business applications, different errors have different costs (actual costs or missed benefits).
  - Wrongly approving a credit card application costs a lot more than wrongly denying an application
  - Wrongly filtering out a good email costs a lot more than wrongly accepting a spam
  - Wrongly diagnosing an ill patient as normal costs a lot more than wrongly diagnosing a normal people as ill.

Asymmetric Misclassification Costs

- Still the direct mail marketing example. Suppose the profit from a responder is $11 and the cost of sending the offer is $1. What will be the cost matrix?

*Assuming the cost of making correct classification is zero. The benefit of making a correct classification would be reflected in the missed benefit of making an incorrect classification.

cost matrix

	Predicted (+)	Predicted (-)
Actual (+)	$0	$11-$1=$10
Actual (-)	$1	$0

loosing the cost

loosing potential profit of $10

+: responder
-: non-responder

Average Misclassification Cost

- A popular performance measure is average misclassification cost, which measures the average cost of misclassification per classified example.

$$\text{Average misclassification cost} = \frac{\#FP*Cost(FP) + \#FN*Cost(FN)}{n}$$

Average Misclassification Cost

- What if the performance measure is average misclassification cost?

cost matrix

	Predicted (+)	Predicted (-)
Actual (+)	$0	$11-$1=$10
Actual (-)	$1	$0

Decision tree

	Predicted (+)	Predicted (-)
Actual (+)	8	2
Actual (-)	20	970

Average misclassification cost=?

40/1000

Majority-class

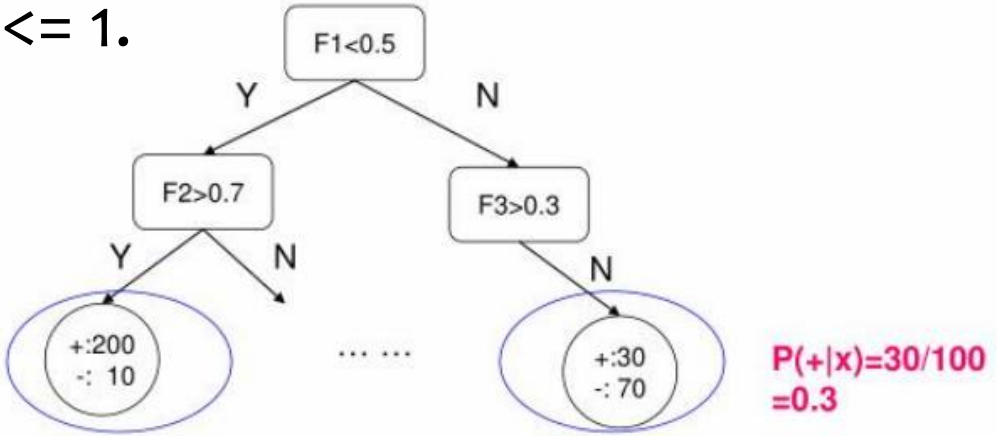
	Predicted (+)	Predicted (-)
Actual (+)	0	10
Actual (-)	0	990

Average misclassification cost=?

100/1000

Probability Estimates (PE)

- So far, we have assumed that the classification output is discrete (i.e., either 0 or 1).
- But most classification algorithms can return probability estimates, i.e., 0 <= P(+) <= 1.



  - Used as an interim step for generating class membership
  - ROC curve

Decision Threshold

- The default decision threshold in 2-class classifiers is 0.5 (i.e., if P(+)>=0.5, predict +; otherwise, predict -.)
- It is possible, however, to use a decision threshold that is either higher or lower than 0.5
  - Typically, the error rate will rise if doing so.

Why would we want to use decision threshold different from 0.5 if they increase the error rate?

Answer: asymmetric misclassification costs! May want to tolerate a higher error rate for the lower-cost misclassification error.

Example for Using Decision Threshold ≠ 0.5

- Still the direct mail marketing example.

cost matrix

	Predicted (+)	Predicted (-)
Actual (+)	$0	$11-$1=$10
Actual (-)	$1	$0

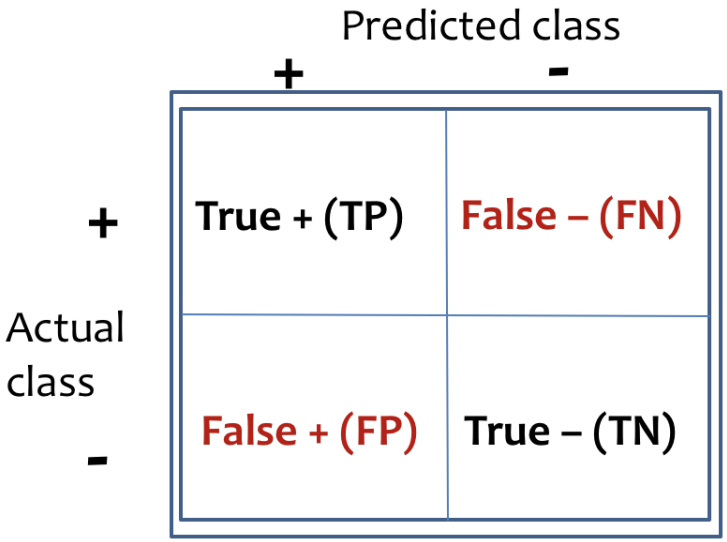
+: responder
-: non-responder

Suppose that P(+) = 0.3, should we classify the customer as responder or non-responder?

Treat as responder - lower misclassification cost = $0.7

ROC Analysis

- Developed in 1950s for signal detection theory to analyze noisy signals
- It provides a better view on classification performance.
- It utilizes all four cells in confusion matrix.



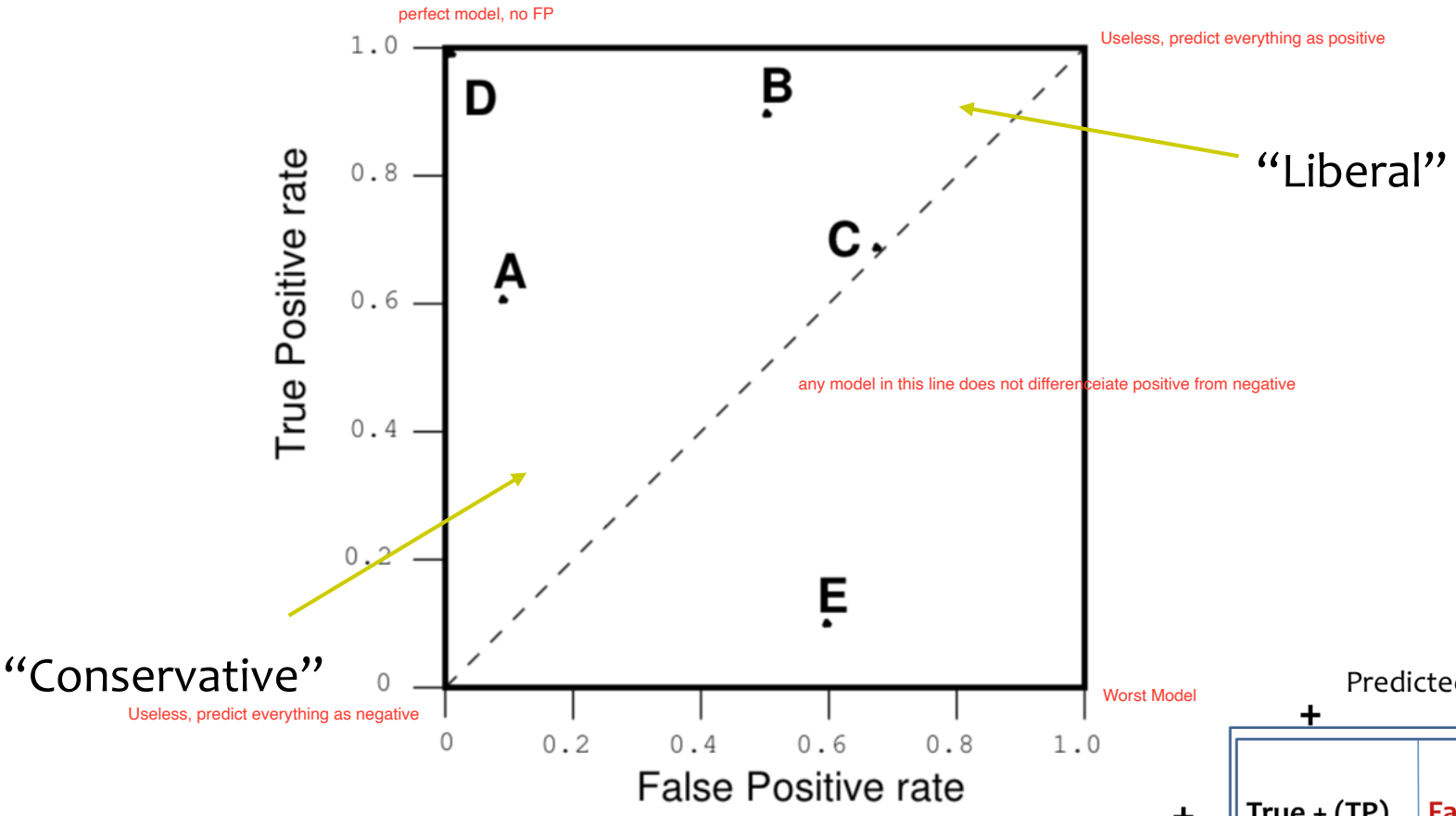
$$\text{True Positive Rate (TPR)}, \text{recall} = \frac{TP}{TP + FN}$$

High TPR is better

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

Low FPR is better

ROC Analysis of A Classifier



Predicted class	+	-
Actual class +	True + (TP)	False – (FN)
Actual class -	False + (FP)	True – (TN)

What do (0, 0), (0,1), (1,0) and (1,1), and the diagonal line on ROC Curve mean?

ROC Analysis

- Still the direct mail marketing example. Can you place two classifiers on the ROC graph?

Decision tree

	Predicted (+)	Predicted (-)
Actual (+)	8	2
Actual (-)	20	970

TPR = ? 8/10

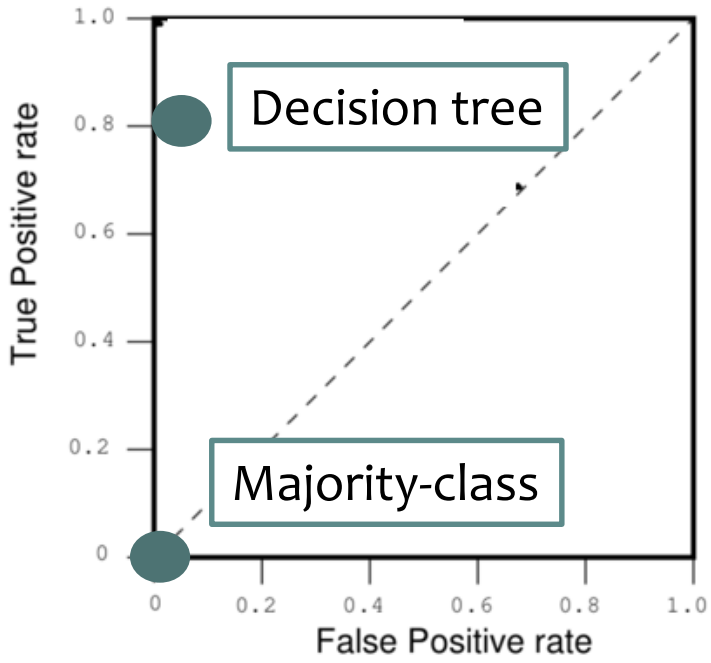
FPR = ? 20/990

Majority-class

	Predicted (+)	Predicted (-)
Actual (+)	0	10
Actual (-)	0	990

TPR = ? 0

FPR = ? 0



Change Decision Threshold

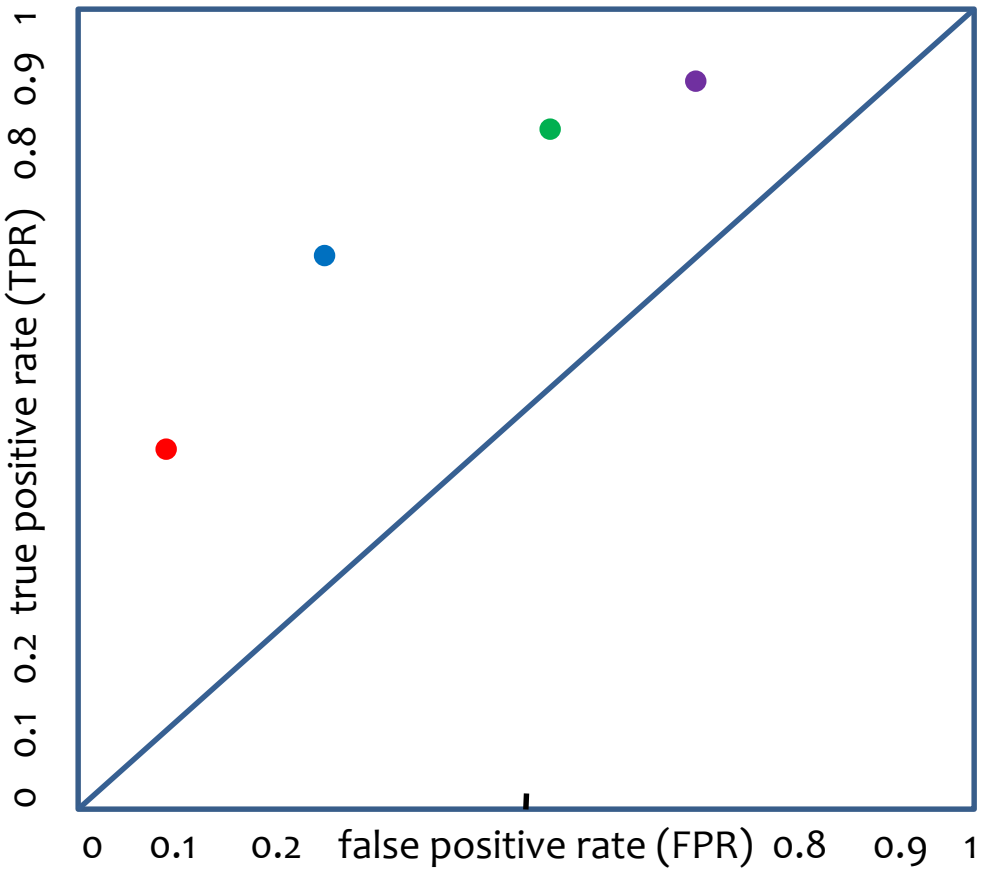
- Given the class probability estimates (PE), changing decision threshold could change TP or FP rate.

4 examples

PE for +	Prediction (0.5 threshold)	Prediction (0.8 threshold)	True value
0.3	-	-	-
0.55	+	-	+
0.75	+	-	-
0.9	+	+	-
True positive rate	1/1	0/1	
False positive rate	2/3	1/3	
Accuracy rate	2/4=0.5	2/4=0.5	

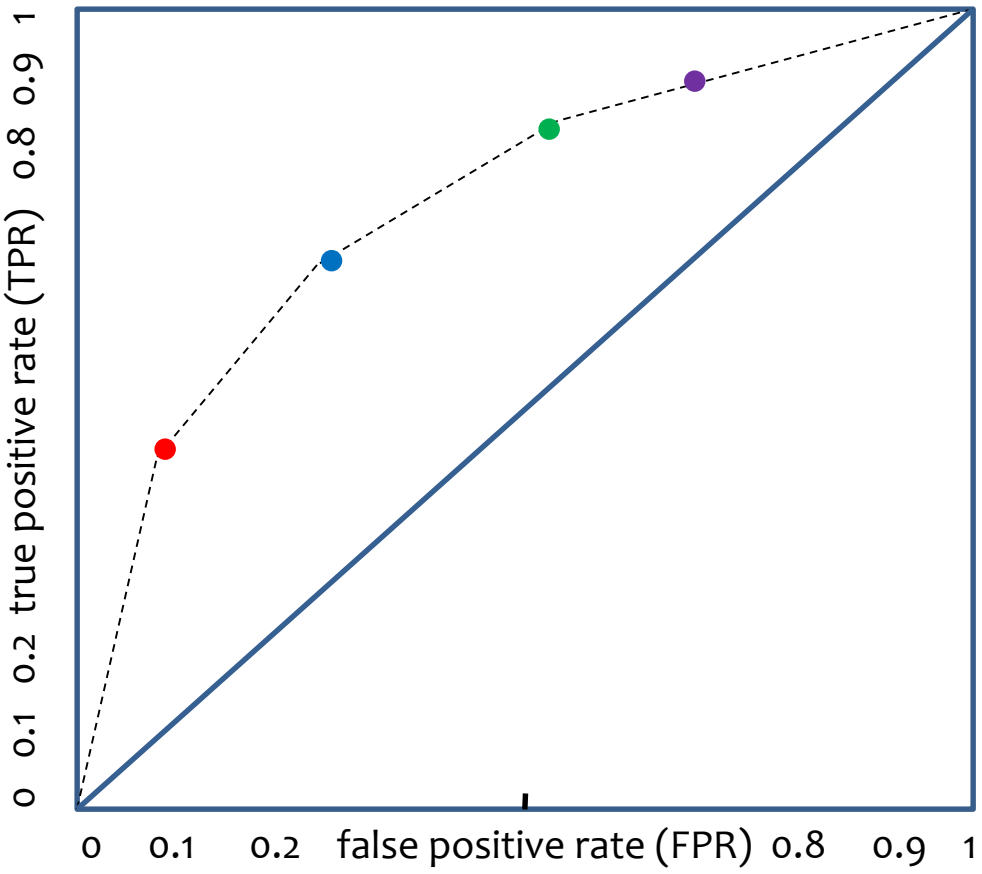
ROC Analysis

- Each decision threshold corresponds to a pair of TPR (on the y-axis) against FPR (on the x-axis), given a particular model
- Changing the decision threshold *may* change the location of the point



ROC Curve

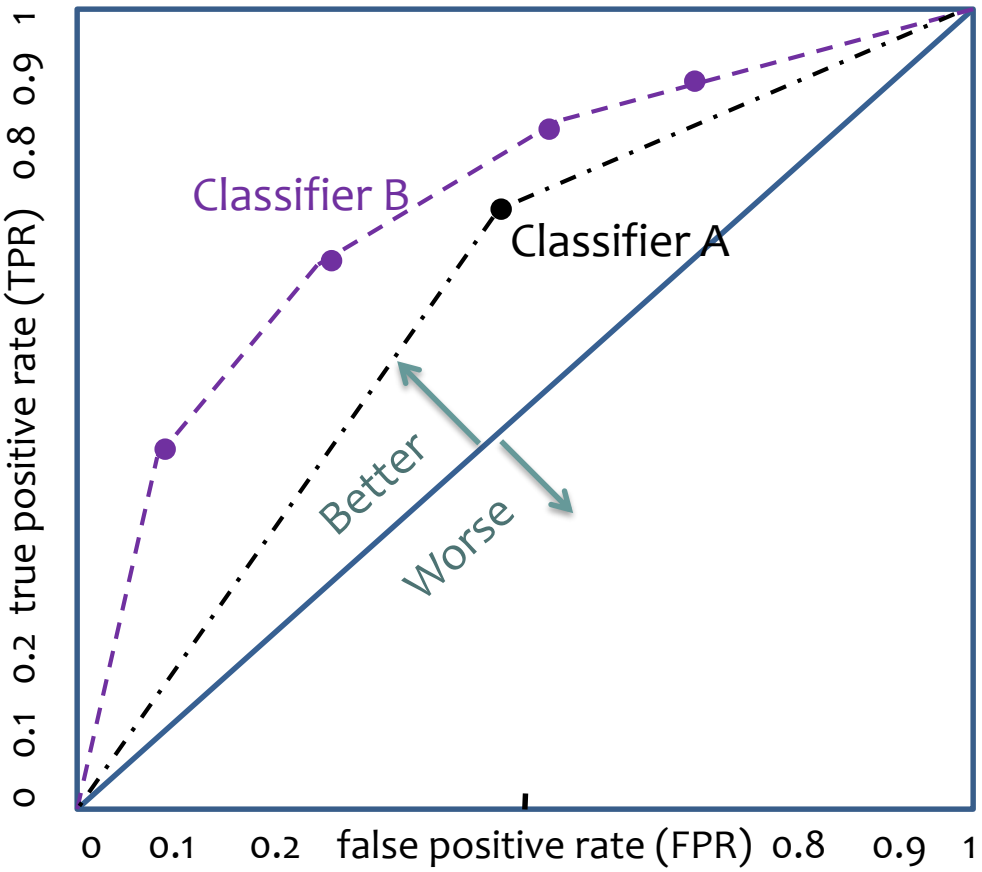
- Connecting dots to get a curve for a model



Threshold = 0.9
Threshold = 0.7
Threshold = 0.5
Threshold = 0.3

ROC curve always starts with (0,0) and ends with (1,1).

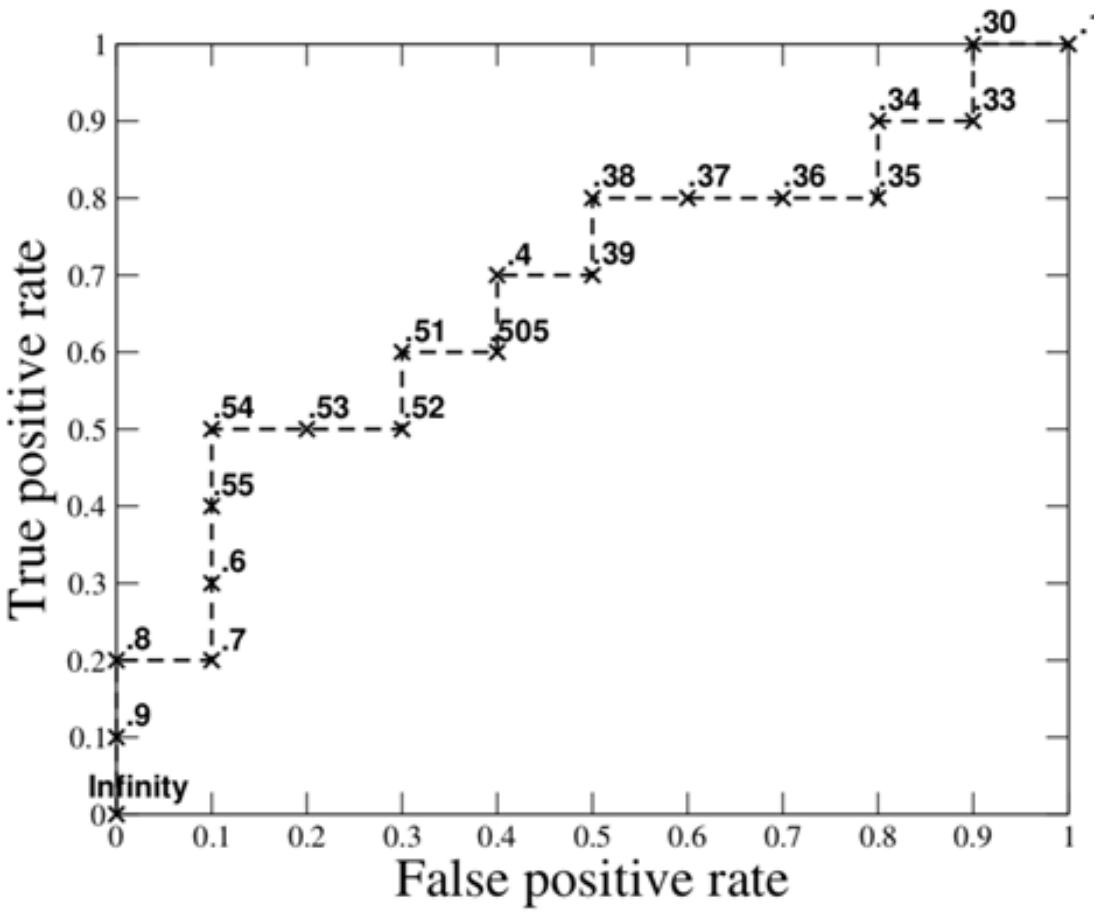
ROC Curve



--- Classifier A:
Discrete Classifier (output value is 0 or 1)

--- Classifier B:
Probabilistic Classifier (output probability estimates)

Drawing an ROC Curve



Actual class

Probability estimate

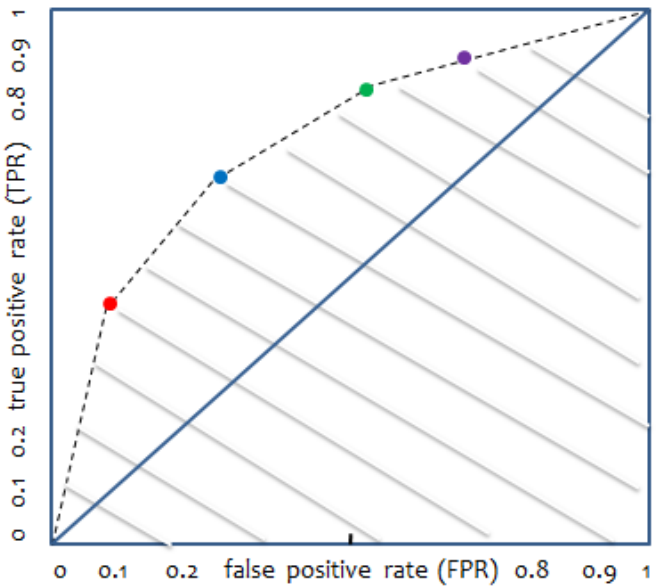
Inst#	Class	Score
1	p	.9
2	p	.8
3	n	.7
4	p	.6
5	p	.55
6	p	.54
7	n	.53
8	n	.52
9	p	.51
10	n	.505
11	p	.4
12	n	.39
13	p	.38
14	n	.37
15	n	.36
16	n	.35
17	p	.34
18	n	.33
19	p	.30
20	n	.1

Lab: Drawing an ROC Curve

- Files needed
  - ROC.ipynb (Python file)

Area under ROC Curve (AUC)

- Performance measure: area under ROC curve (AUC)



- The bigger the AUC, the better the model is
- Measure performance under different decision thresholds.
- AUC is a "deeper" measurement
  - Better measure when uneven costs are unknown
  - Better measure if the class distribution is uneven

Exercise

- What is AUC of a perfect classifier?
- What is AUC of a random classifier (i.e., 30% guessing positive, 70% guessing negative)?

FPR = 0.3 TPR =0.3

Model doesn't differencieate between FPR and TPR

AUC = 0.5

Lab: ROC

- Files needed
  - ROC_titanic.ipynb (Python file)
  - titanic_cleaned.csv (dataset)

Evaluation Regression Performance

- Naïve rule: the average (benchmark model)
  - Using the average outcome value as prediction
  - Is used as a baseline or benchmark for evaluating the performance of more complicated models.

Name	Gender	Age	Spending Amount
Mike	Male	50	$1,370
Mary	Female	40	$1,920
Bill	Male	55	$470
Jim	Male	46	$1,190
Dave	Male	44	$760
Anne	Female	50	$890

Benchmark Prediction:
Spending Amount for Eva (Female, 40) =? Average of the spending amount

Evaluation Measures for Regression

- Mean squared error (MSE)
  - The average of the squares of the differences between predicted values and actual values

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)})^2$$

- Root mean squared error (RMSE)
  - The square root of the average of the squares of the differences between predicted values and actual values

$$\mathbf{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)})^2}$$

- Mean absolute error (MAE)
  - The average of the differences between predicted values and actual values

$$\mathbf{MAE} = \frac{1}{n}\sum_{i=1}^{n}|h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)}|$$