CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

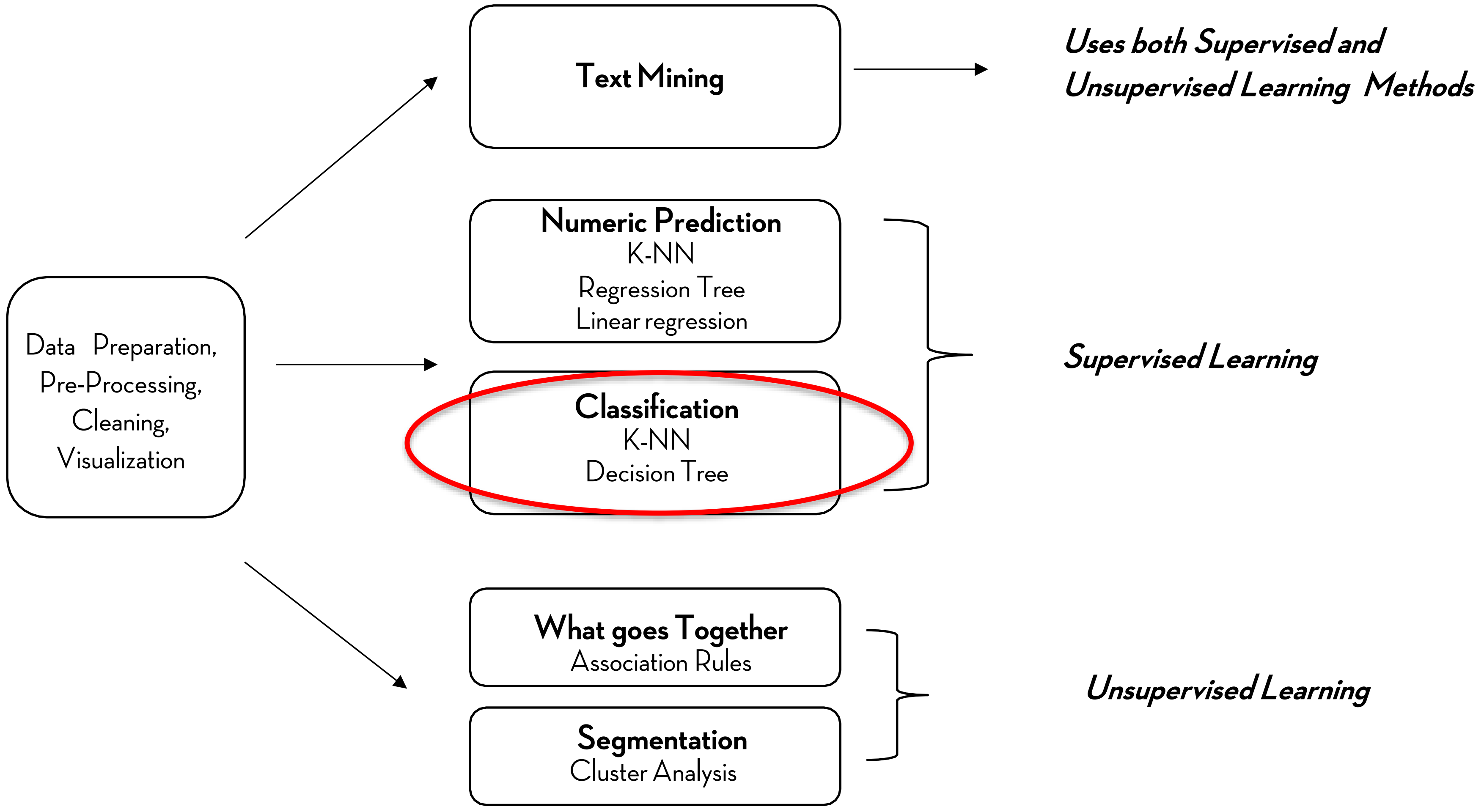
IDSC 4444 (004) Predictive Analytics: Classification

Zihong Huang
Information & Decision Sciences
Carlson School of Management
huan0707@umn.edu

Agenda

- ❑ Classification Definitions and Steps
- ❑ Classifiers
  - k-NN
  - Decision Tree
- ❑ Evaluating Performance
  - Confusion Matrix

An Overview



Data Preparation, Pre-Processing, Cleaning, Visualization
Text Mining
Uses both Supervised and Unsupervised Learning Methods
Numeric Prediction
K-NN
Regression Tree
Linear regression
Classification
K-NN
Decision Tree
Supervised Learning
What goes Together
Association Rules
Segmentation
Cluster Analysis
Unsupervised Learning

Predictive Data Mining

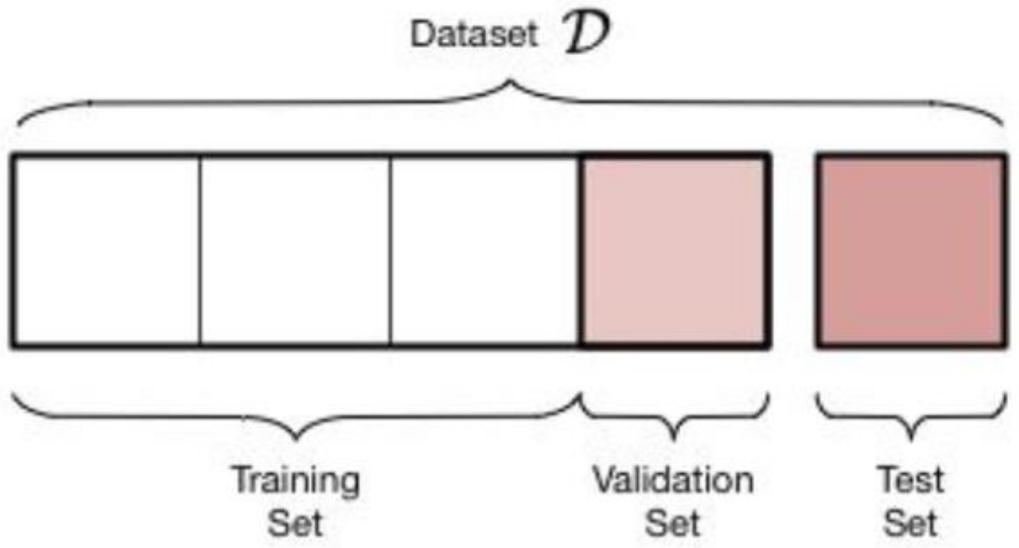
- Unlike exploratory data analytics (Association Rules, Cluster Analysis), in predictive analytics we know exactly what you are looking for
  - We are trying to predict certain well-defined outcome using existing data
  - Among the attributes we have in the data, we have to specify **dependent (outcome) variables** and **independent variables (attributes/features)**
- **Classification:** predict categorical values of the outcome variable using other available attributes, e.g.
  - Techniques: **k-NN (Nearest-Neighbors), Decision Trees**, etc.
- **Numeric Prediction:** predicts continuous/numeric values of an outcome variable, e.g.
  - Techniques: K-Nearest-Neighbors, Regression Trees, etc

Classification: Some Definitions

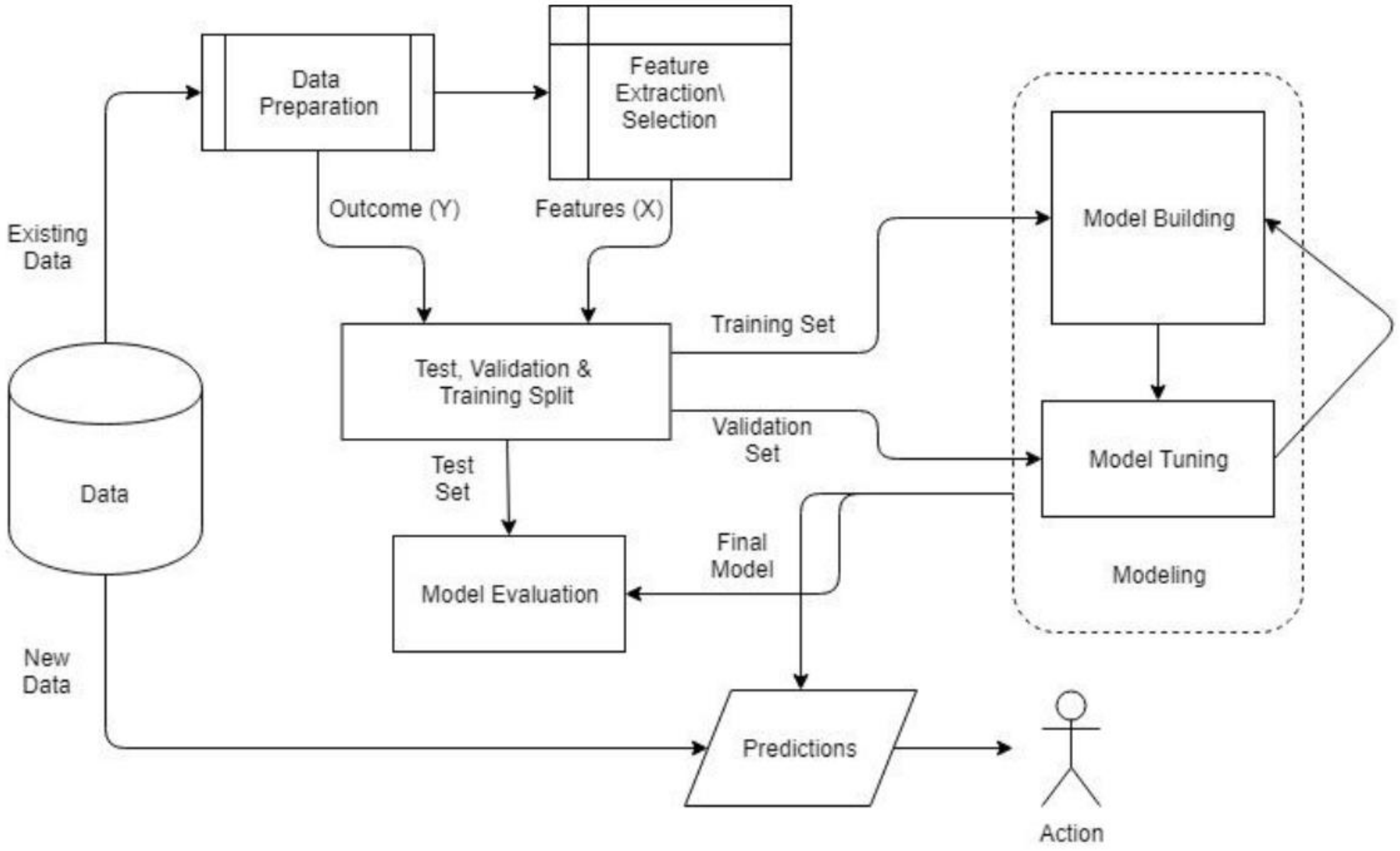
- ❑ **Observation/Record**: **our unit of observation** (example, a customer), defined by the collection of available attributes (variables)
- ❑ **The attributes** that refer to a given observation, are now divided into:
  - ○ **One dependent variable (Y) / Outcome Variable, Class variable**:
    - ✓ This variable represents the outcome we want to predict
  - ○ **Independent Variables (X) /Predictors, Covariates:**
    - ✓ The attributes of an observation used to predict the outcome variable Y
    - ✓ Collectively known as **Features vector** or **Covariates vector**

Training -Validation - Test

- ❑ Randomly split your labeled data into three parts:
  - ○ **Training data:** to build your model
  - ○ **Validation data:** to tune the performance of your model
  - ○ **Test data:** to test the performance of your model
- ❑ It works because your model, built on training data, has not "seen" the validation data
  - ○ If your models perform well on validation data, it is likely to also perform well for new, unseen data when you actually deploy it.
  - ○ Sometimes, Validation might be skipped



The Prediction Process



k-NN (k- Nearest Neighbors)

- Do not confuse with k-Means in Clustering!
- **Main idea:** For a given observation, identify "nearby" (similar) observations
  - Use distance metrics seen in Clustering (Euclidian is the favorite)
- k-NN will classify the observation according to the predominant class among the identified nearest neighbors observations
  - We need to decide k, how many neighbors we want to consider.

	(Age)	(Income)	(Gender)	Purchase/Not Purchase Product
A	25	55000	M	No Purchase
B	32	120000	M	Purchase
C	43	150000	F	Purchase

Labeled Data:
We know the Y ("class")

New observation to classify

D	40	130000	F	???

Euclidian Distance

d(D, A)	2.8489
d(D, B)	1.9545
d(D, C)	0.5285

K = 1,?
K = 3, ?

k-NN Procedure

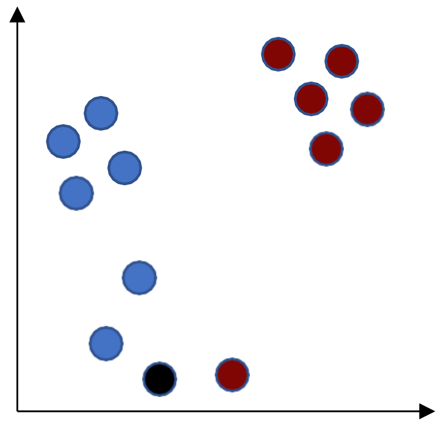
- ❑ Choose a specific k and a distance measure (usually, Euclidian)
  - Remember, in this case, k is the number of neighbors you want to look at
- ❑ Normalize data if needed
- ❑ For every unlabeled observation, identify the k-nearest existing labeled observations
- ❑ Classify the unlabeled observation as the majority class among the k-nearest
- ❑ In case of a tie, randomly choose a class
  - E.g., if k is an even number

Choosing k value

- k is the number of neighbors considered when classifying a new observation
- Usually, choose k that has lowest error rate in validation data (or, equivalently, the greatest accuracy)
  - **Error:** classifying an observation as belonging to one class when it belongs to another
  - **Error rate:** % of misclassified observations out of the total number of observations in the validation data
  - **Accuracy**: 1 – error rate

k-NN Summary

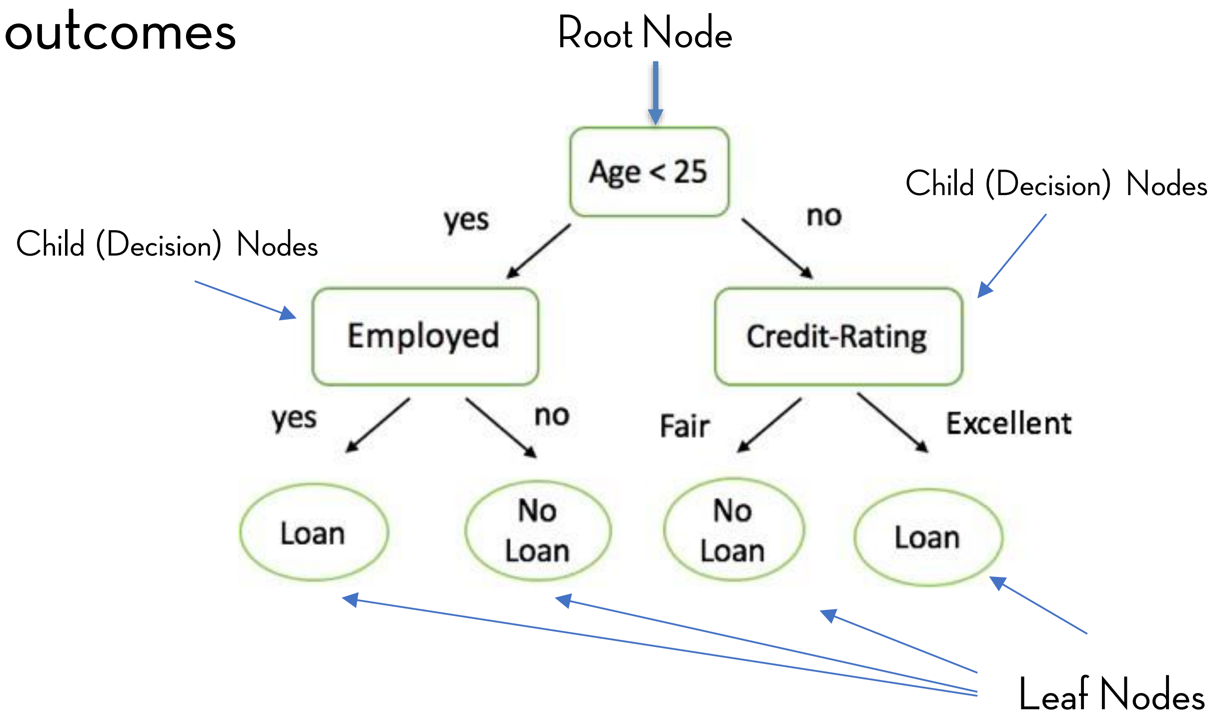
- k-NN is known as a type of "Lazy" classifier
  - No "real" model building – your labeled data is your model
- Enhancement:
  - **Weighted distance**, e.g., give more importance (weight) to closer neighbors
- **Strengths:**
  - Easy to implement and use
  - Do not make assumptions about statistical/distributional characteristics of the data (other methods do)
- **Weakness:** bad for big dataset
  - Slower learner, cannot be used for real-time prediction.



Decision Trees

- Also known as *Classification* Trees
- Goal: To classify data into predefined classes based on attributes
  - The output is a Decision Tree which is a set of decision rules organized as a tree.
- Nodes indicate decisions, leaves indicate outcomes

Ind.	Age	Employed	Credit-rating	Loan
A	23	Yes	Fair	Yes
B	28	No	Excellent	Yes
C	22	No	Fair	No
D	35	No	Fair	No
E	21	Yes	Fair	No
F	22	Yes	Fair	Yes
G	33	No	Excellent	Yes

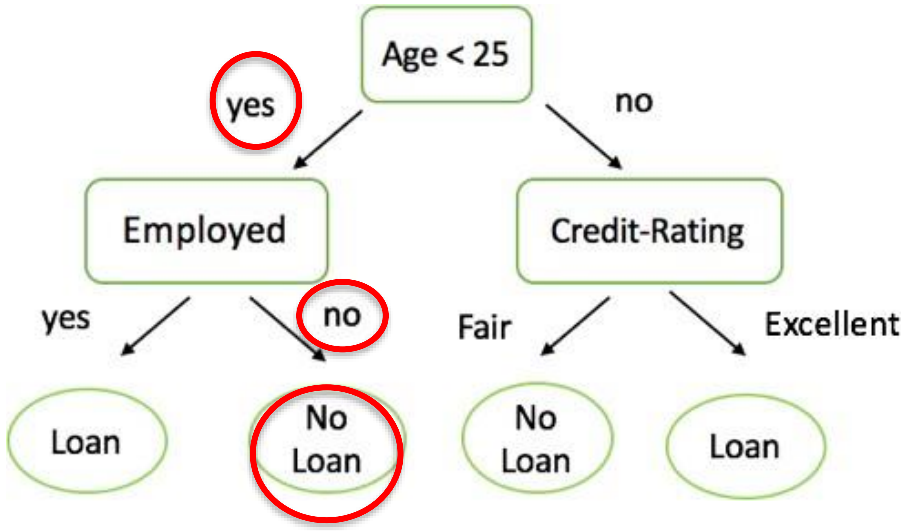


Decision Trees

- To classify a new record, the attributes are tested against the decision tree
- Example:

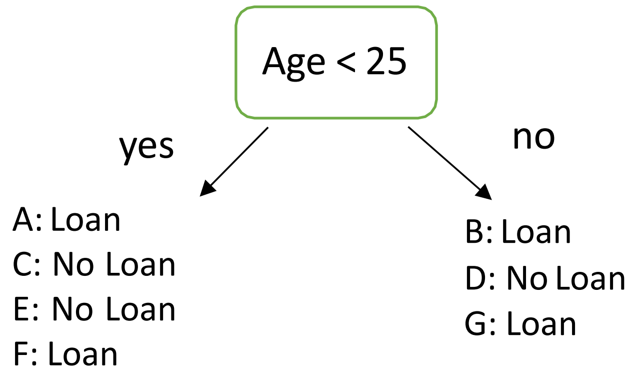
Ind.	Age	Employed	Credit-rating	Loan
ZZ	23	No	Fair	??

- Test on Age: Age < 25: Yes
- Test on Employed: No
- Reach Leaf node (class): No Loan
- Customer ZZ is unlikely to get a Loan



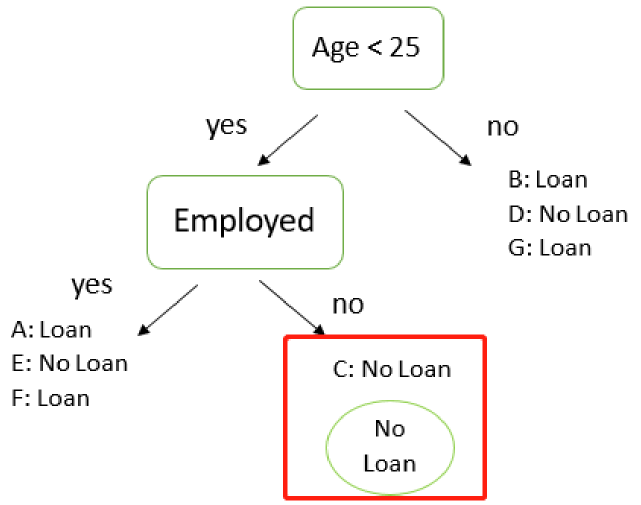
How is Decision Tree Built?

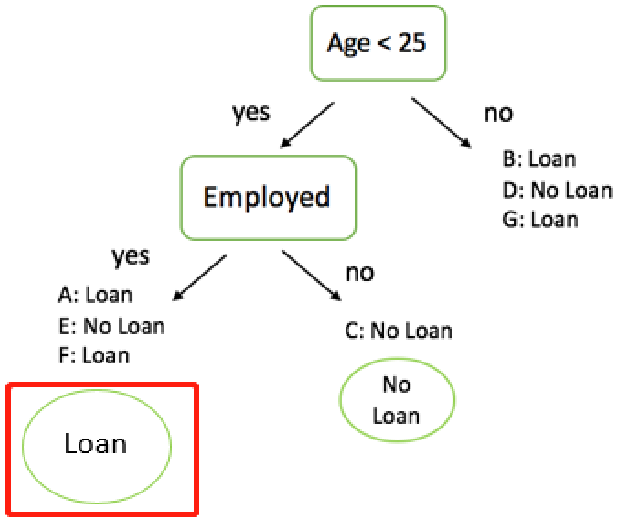
- The decision tree is built (trained) from the training data
- Data are split (partitioned) based on the attributes/predictor variables
- **Recursive Partitioning**
  - Pick one of the **attributes,** Xi, e.g., Age
  - Pick a **value** of Xi (let us call it s_i) that divides the training data into portions (not necessarily equal), one with Xi >= s_i and the other with Xi < s_i
  - Measure **how "pure"** each of the resulting proportion is
    - ✓ "Pure" means the partition contains records of mostly one class
  - Repeat the process until find Xi and s_i to **maximize purity of the partitions**



When do We Stop?

- Stopping criteria (when to stop splitting nodes?)
  - All data points associated with a node are from the same class
    - ✓ The node becomes a leaf node of that class
  - There are no remaining attributes to further split the data
    - ✓ E.g., no attribute further increases "purity" much
    - ✓ Use "majority vote" to label that node



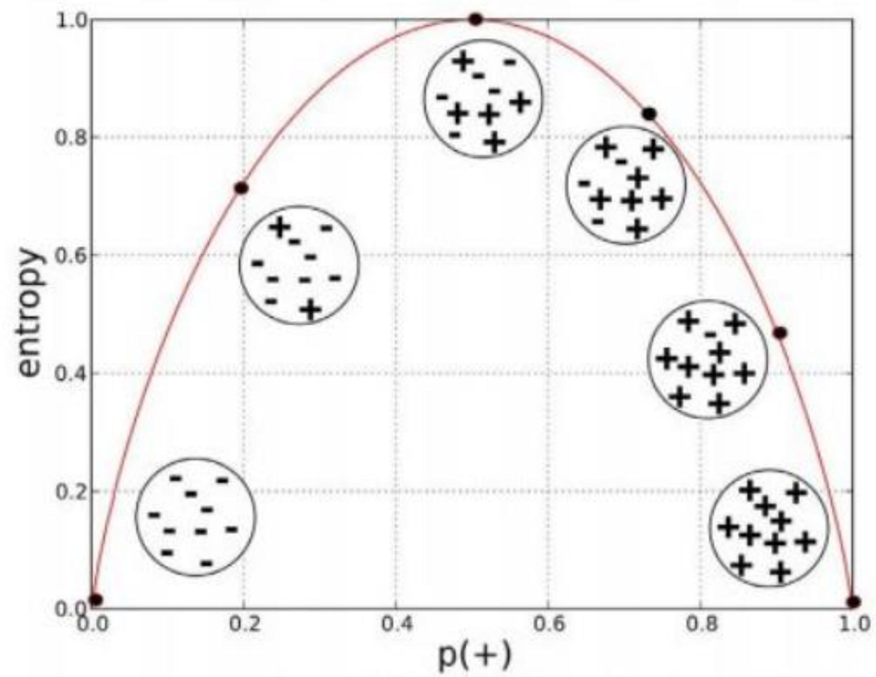


Choose Attributes to Split on (Entropy)

- ❑ Intuition: pick the attribute that **maximizes the purity** of the resulting split
- ❑ We measure "purity" through **Entropy:**

$$Entropy = -\sum_{k=1}^{m} p_k log_2(p_k)$$

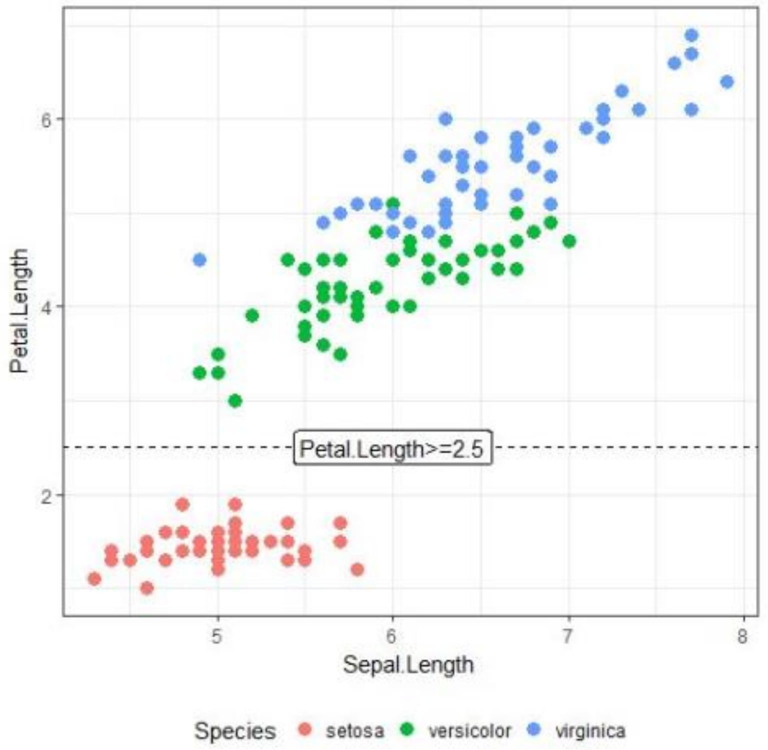
- where m = number of classes, p_k is the prob. of the class in a given partition
- Between $[0, log_2(m)]$
- 0, most pure, all records belong to the same class

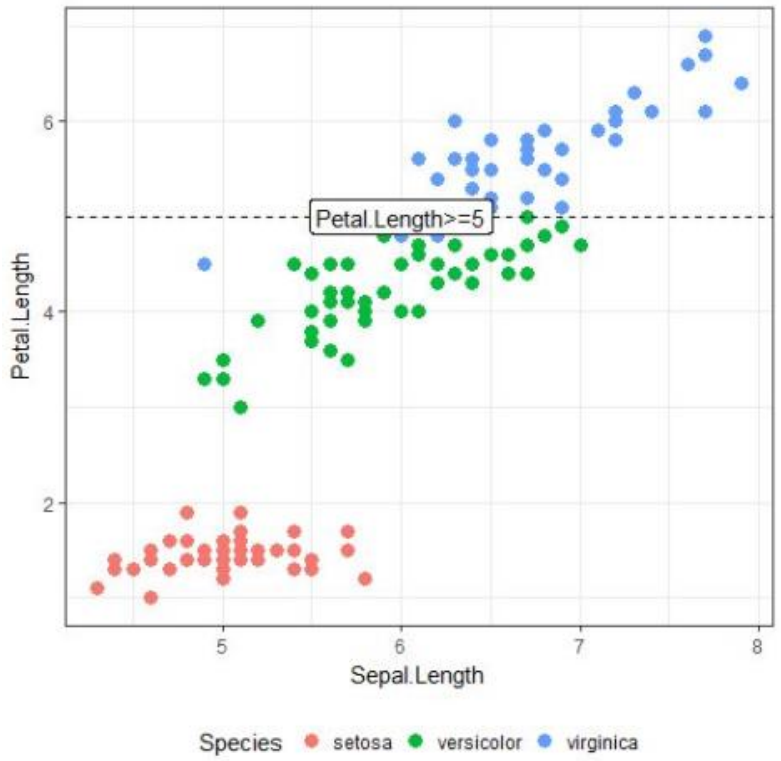


Source: "Data Science for Business" by Provost & Fawcett, 2013

- ❑ E.g., if the data contains 70% of Class A and 30% of Class B then:
  - ○ Entropy = - [0.7*log2(0.7) + 0.3*log2(0.3)] ~ 0.88
- ❑ Higher the entropy, more impure the data.
  - ○ Highest (2 classes) = - [0.5*log2(0.5) + 0.5*log2(0.5)] = 1

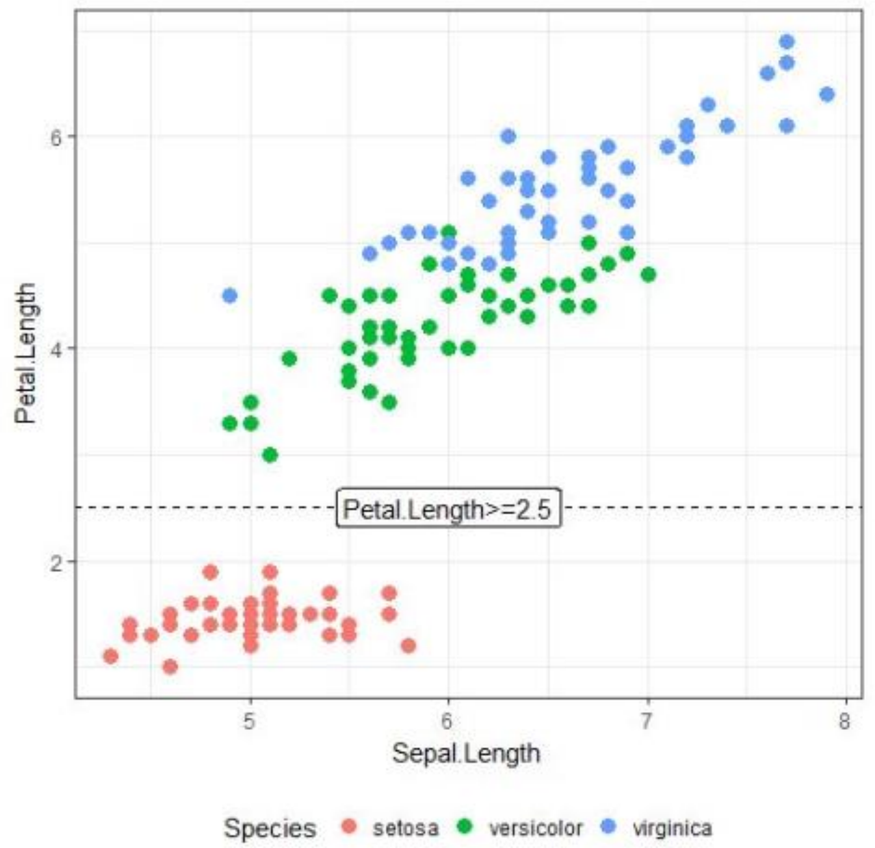
Choose Attributes to Split on





- ❑ **Information Gain** :Compare shift from original entropy. **(Higher the better)**
  - ○ Information Gain = Entropy(Parent) – Weighted Avg (Entropy(Children))

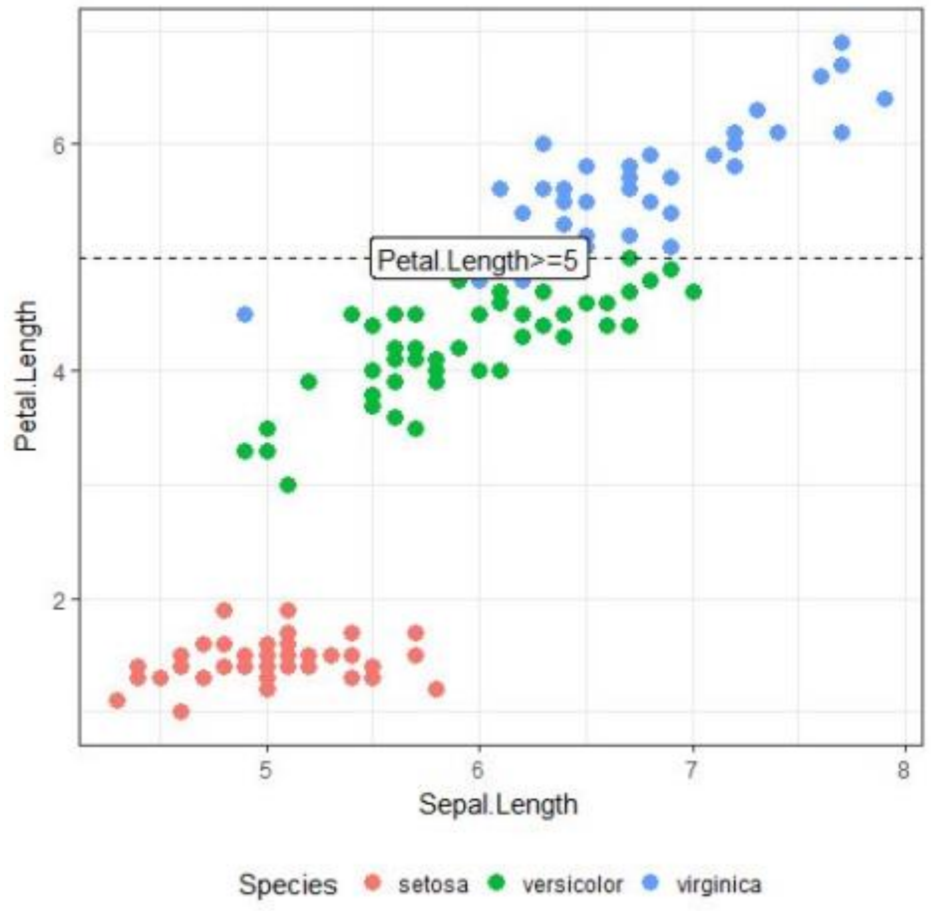
Choose Attributes to Split on



- ❑ Calculate the entropies of the two partitions
  - ○ **Petal.Length <2.5:** $-1*\log_2(1) = 0$
  - ○ **Petal.Length >=2.5:** $-[0.5*\log_2(0.5)+0.5*\log_2(0.5)] = 1$
  - ○ **Weighted Entropy:** $0 *(50/150) + 1 *(100/150) =$ **0.67**
- ❑ Calculate the entropy of the parent (in this case, all the data points together, before initial split)
  - ○ $-[0.33*\log_2(0.33) + 0.33*\log_2(0.33) + 0.33*\log_2(0.33)] =$ **1.58**
- ❑ **Information Gain: 1.58 – 0.67 = 0.91**

The data: 150 points in total, 50 red, 50 green, 50 blue

Choose Attributes to Split on



Above the line, you have 43 points (41 blue and 2 green)

Below the line, you have 107 points (50 red, 9 blue and 48 green)

- ❑ Calculate the entropies of the two partitions
  - ○ **Petal.Length <5:**

    $-[0.47*\log_2(0.47) + 0.45*\log_2(0.45) + 0.08*\log_2(0.08) = 1.31$
  - ○ **Petal.Length >=5:**

    $-[0.95*\log_2(0.95) + 0.05*\log_2(0.05)] = 0.28$
  - ○ **Weighted Entropy:**

    $1.31 * (107/150) + 0.28 * (43/150) = 1.01$
- ❑ The entropy of the parent

  $-[0.33*\log_2(0.33) + 0.33*\log_2(0.33) + 0.33*\log_2(0.33)] =$ **1.58**
- ❑ Information Gain: 1.58 – 1.01 = **0.57**

Splitting on Petal.Length at 2.5 is better as the Information Gain is higher

Decision Trees: Procedure Recap

- ❑ Final tree is constructed as follows:
  - Start with the attribute-split for **the root node**: this is going to be the attribute-split that leads to the **largest information gain**, compared to all the others
  - From there, the algorithm proceeds looking at the information gain of subsequent splits on different attributes-values, conditional on the previous one
  - Note:
    - ✓ **The same attribute can be used again**, as long as the partitions/splits are not overlapping
    - ✓ <u>Not all the attributes</u> available in the dataset need to be used in the tree

Classification Probabilities and Cutoffs

- ❑ Many classifiers (including k-NN, Decision Trees) not only produce a class prediction, but they also produce a **probability** that the data point belongs to that class
  - k-NN: % of a class among k-neighbors
  - Decision tree: % of a class in a leaf node
- ❑ Predictions can then generated based on **a cutoff** value:
  - o Default cutoff is 0.5 (50%) (majority rule):
    - ✓ If probability of belonging to class "A" is >50%, then classify as belonging to class "A"
    - ✓ Otherwise, classify as class "B"
  - o <u>Different cutoffs can be set and can lead to different classification results</u>

Evaluating Model Performance

- ❑ In classification, performance is evaluated using a **confusion matrix**

		Actual Class	
		Yes (Pos.)	**No (Neg.)**
Predicted Class	**Yes (Pos.)**	True Positive	False positive (Type I error)
	No (Neg.)	False Negative (Type II error)	True Negative

- ❑ For each cell, e.g., True Positive (TP), **the number of cases** for which we predict Yes, and the actual class is Yes

- ❑ In classification, a **Confusion matrix** is created to calculate:
  - ○ **Accuracy (error rate):** overall performance
    - ✓ Not good in unbalanced dataset
    - ✓ E.g., 95 "Yes, Loan", 5 "No Loan"
  - ○ **Precision:** class specific
  - ○ **Recall:** class specific
  - ○ **F-measure:** class specific

Intuition Behind Each Measure

- ❑ **Accuracy:** Among all predictions, how many did the model got right?
  Accuracy = (TP + TN) / (TP + TN + FP + FN)
- ❑ **Precision:** For each class, how many did the model get right?
  - Precision_pos. = positive predictive value (**PPV**) = TP / (TP + FP)
  - Precision_neg. = negative predictive value (**NPV**) = TN / (TN + FN)
- ❑ **Recall:** How many per each class were recovered?
  - Recall_pos. = **Sensitivity** = TP / (TP + FN)
  - Recall_neg. = **Specificity** = TN / (TN + FP)
- ❑ **F-measure:** the harmonic mean of precision and recall
  - F1_score = 2* ((recall * precision) / (recall + precision))
  - It is a more reliable measure than accuracy when your data is unbalanced
  - The higher the F1 score, the better the classification model

		Actual Class	
		Yes (Pos.)	No (Neg.)
Predicted Class	Yes (Pos.)	True Positive	False positive (Type I error)
	No (Neg.)	False Negative (Type II error)	True Negative

TP: True Positive

TN: True Negative

FP: False Positive

FN: False Negative

Example

- ❑ Accuracy: 0.7 =70%
- ❑ Recall (Sensitivity) for class **Non-Fraud**: =0.75 =75%
- ❑ Recall (Specificity) for class **Fraud**: =0.5 =50%
- ❑ Precision (PPV) for class **Non-Fraud**: =0.86 =86%
- ❑ Precision (NPV) for class **Fraud**: =0.33 =33%

		Actual Class	
		Non-Fraud	**Fraud**
Predicted Class	**Non-Fraud**	60	10
	Fraud	20	10

- F1_Score for **Non-Fraud**: 2* ((0.75*0.86) / (0.75 + 0.86)) =0.80
- F1_Score for **Fraud**: 2 *((0.5*0.33)/(0.5 + 0.33)) =0.40
- In this example, the overall accuracy of the model is about 70%. Nevertheless, if we look at the F1 scores, we see that for the class **Fraud**, F1 score is about 40%.
- Since this dataset is unbalanced (there are way more cases of **Non-Fraud** than **Fraud**), F1 scores is a more appropriate measure.

Differing Misclassification Costs

- ❑ In many real-world applications, different types of mistakes in classification are not equally important
  - E.g., missing a fraud transaction is very costly, classify a non-fraud as fraud is not as detrimental
- ❑ Instead of maximizing accuracy, sometimes we want to **minimize misclassification cost**
- ❑ One way to minimize misclassification cost is to **adjust the cutoff value**
  - E.g., if misclassifying "Fraud" as "Non-Fraud" is far more costly than the other way around, we want to **reduce cutoff** for classifying a transaction as Fraud.

Cost Matrix		Actual	
		+	-
Predicted	+	TP (P1)	FN (C1)
	-	FP (C2)	TN (P2)

Avg Misclassification cost =(C2*FP + C1* FN)/All

Questions?