

## DATA ANALYTICS TASKS AND METHODS

■ Imdadullah Khan

# Data Analytics Tasks and Methods

---

Data Analytics is the process

- to discover patterns in data
- to find relationships in data
- to (automatically) extract knowledge from data
- to summarize data in ways that are understandable and useful

Discovering knowledge from data often requires learning

## ■ Descriptive Analytics

- Uncover patterns, correlations, trends and trajectories describing the data
- Clustering/grouping the data
- Detecting outliers (anomalies)
- Explanatory in nature
- Require post-processing to validate and explain the results

## ■ Predictive Analytics

- Predict value of an attribute based on values of other attributes
- Predicted attribute is called target/dependent/response variable
- Attributes used for prediction are called explanatory or independent variables
- Classification: nominal target attribute (class labels)
- Regression: numeric target attribute

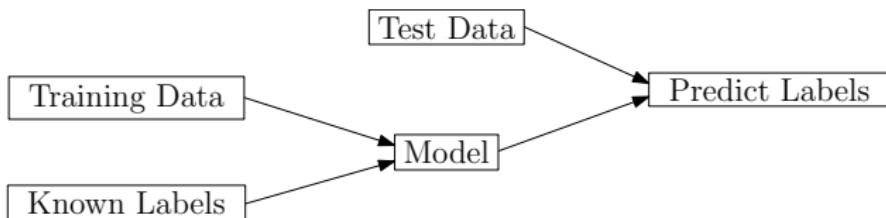
# Data Analytics Tasks

---

- **Clustering:** Partition data into meaningful groups
- **Outlier Detection:** Detect points that are unusual (unlike others)
- **Classification:** Assign (predefined) class labels to each object
- **Regression:** Find a function that models (continuous) target variable
- **Association Analysis:** Find patterns in data that describe relationships
- **Recommendation:** Predict an unknown rating based on known ratings
- **Community Detection:** Find (overlapping) communities of objects (nodes) in networks
- **Centrality and Important nodes:** Find important (or evaluate importance of) nodes in networks

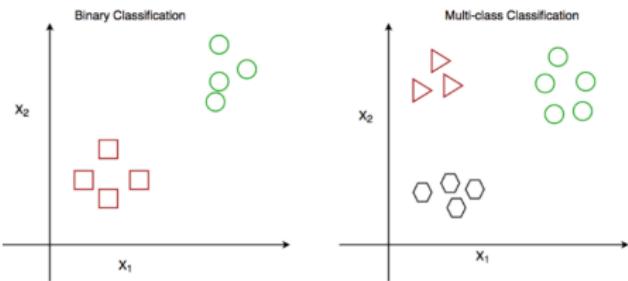
## Supervised Learning

- For some data items the correct results (values of the target variable) are given (**ground truth**)
- We want to learn a model that generalizes
- i.e. the model is able to perform accurately on new/unseen/unlabeled examples
- **Classification**, where the target is a categorical attribute
- **Regression**, where the target is a continuous attribute

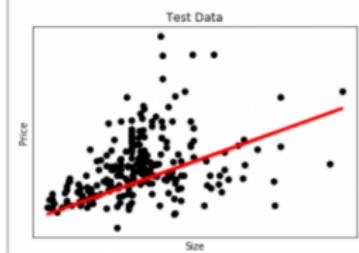


# Classification and Regression

## Classification



## Regression



## Unsupervised Learning

- No correct output is provided
- Learning and analytics is done using statistical properties of data
- Clustering
- Outlier detection
- Modeling the density of data
- Dimensionality reduction

## Classification: Definition

- Given a collection of objects
- Each object is a feature vector and has a class label

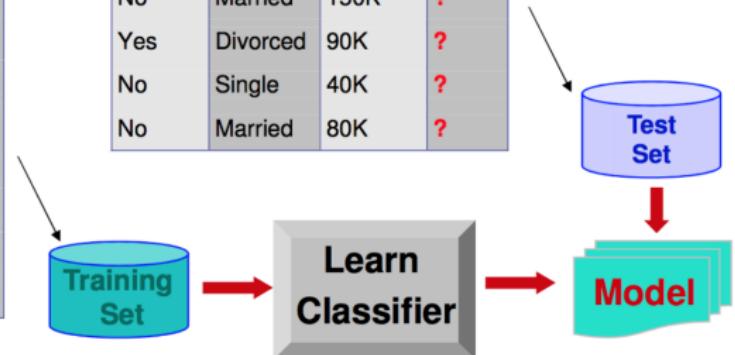
	Features: $X$					Class: $y$
	$x_1$	$x_2$	$\dots$	$\dots$	$x_m$	$y$
$o_1$						
$o_2$						
$\vdots$						
$o_n$						

- **Training Set:** Instances whose class labels are used for learning
- Find a model for the class attribute as a function of other attributes
- Goal: Model should accurately assign class labels to unlabeled instances
- **Test Set:** Feature vectors with the same attribute as training set but missing/hidden class labels

# Classification

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?



# Classification: Applications

---

## ■ Targeted Advertisement

- Enhance marketing by identifying customers who are likely to buy a product
- Use customer purchase history, demographics etc. for similar (old) products
- **buy/no buy** as class labels

## ■ Fraud Detection

- Predict fraudulent credit card transaction
- User transactions history and card holders characteristics
- **fair/fraud** as class labels

## ■ Predict Customer Attrition/Churn

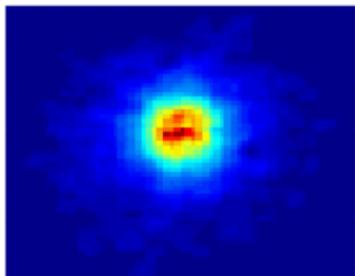
## Sky Survey Cataloging

- Classify astronomical objects as stars or galaxies
- Use telescoping survey images (from Palomar Observatory)
- 3000 images with  $23040 \times 23040$  pixels per image
- Extract features values 40 features per object
- star/galaxy as class labels

# Classification: Applications

Courtesy: <http://aps.umn.edu>

*Early*



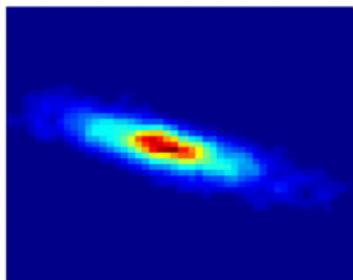
**Class:**

- Stages of Formation

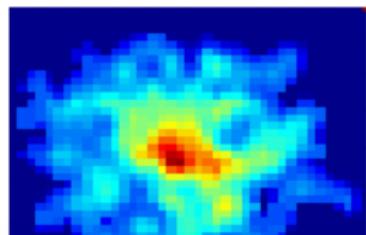
**Attributes:**

- Image features,
- Characteristics of light waves received, etc.

*Intermediate*



*Late*



**Data Size:**

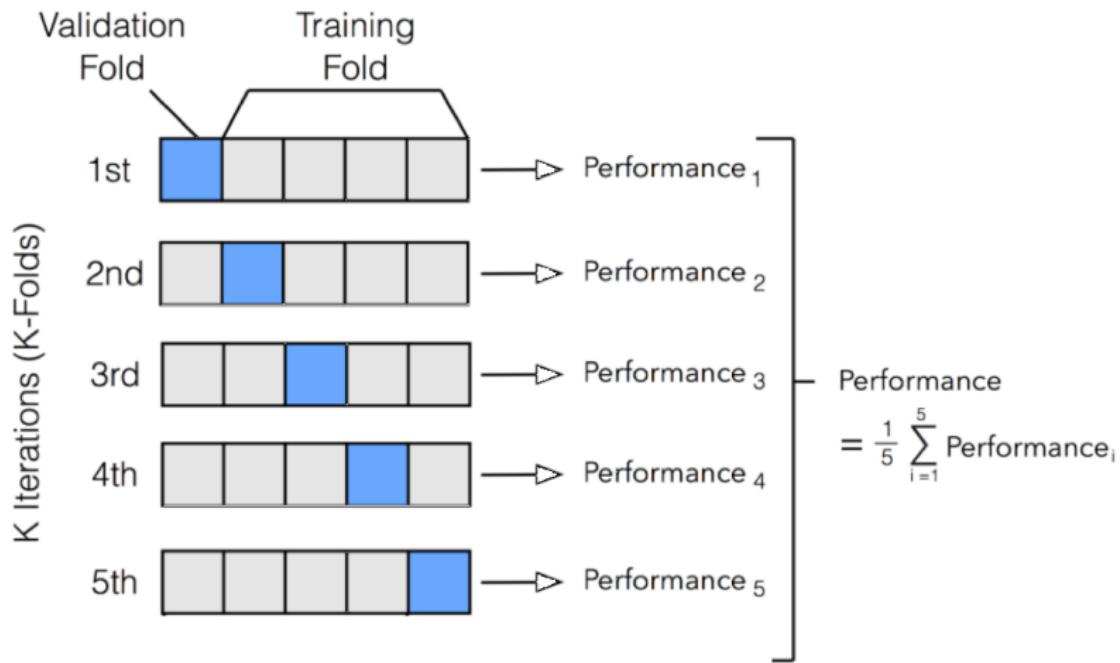
- 72 million stars, 20 million galaxies
- Object Catalog: 9 GB

# Classification

---

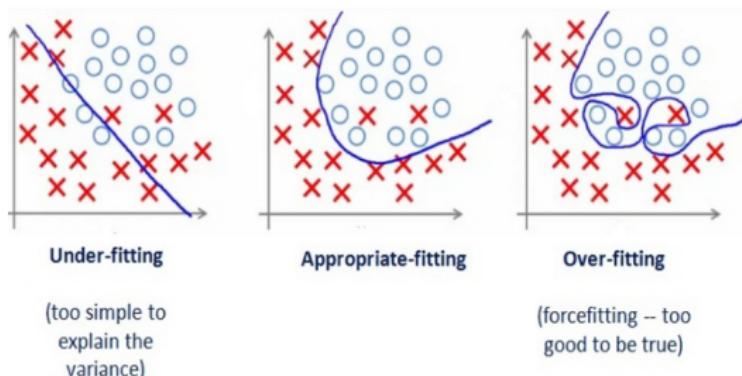
- The model is learned by finding patterns in training set
- Training set accuracy does not (necessarily) indicate generalization power of the model
- A **validation set** (a subset of training set) is used to learn parameters and tune architecture of classifier and estimate error
- For generalization of the model, it is important that validation set is representative of the input instances
- Obtained by random split
- 70 – 30, 80 – 20 random Train-Validation split (multiple samples)
- 5-fold, 10-fold cross-validation
- Since test set is never used during training, it provides an unbiased estimate of generalization error

# Classification: Cross-Validation

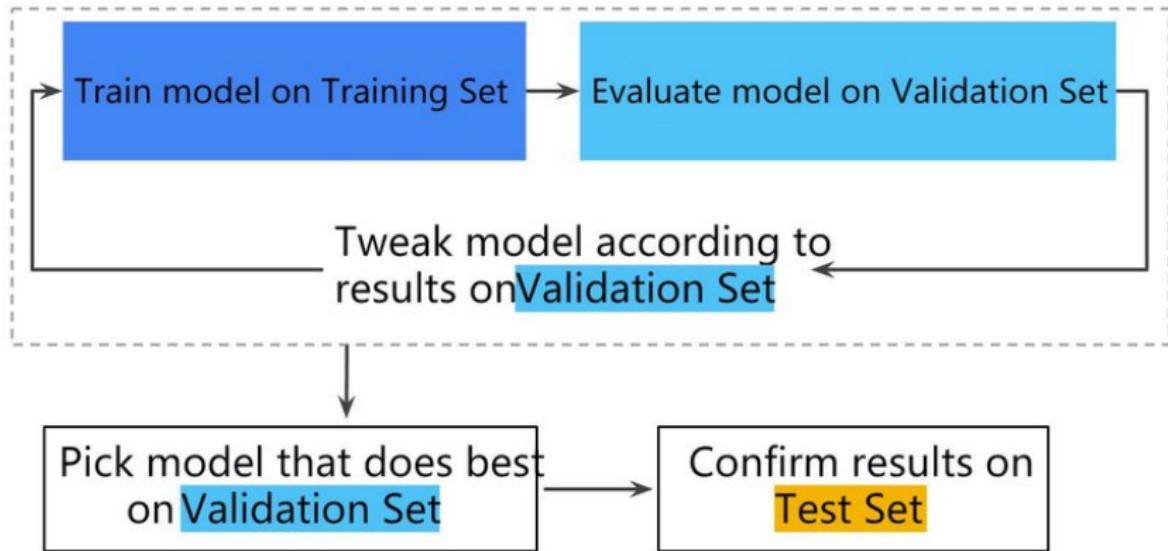


# Classification: OVERFITTING

- The model learns the data and not the underlying function
- Essentially learning by-rote
- Model has too much freedom (many parameters with wider ranges)
- When the model performs very well on training data but doesn't generalize to testing data, the phenomenon is called **overfitting**
- Cross-validation, early stopping, regularization, model comparison, Bayesian priors help avoiding overfitting



# Classification: Validation to avoid OVERFITTING



# Classification: Validation to avoid OVERFITTING



# Classification: Evaluation Metrics

Binary Classifiers (for classifying into two classes) are evaluated by tabulating the classification results in a [Confusion Matrix](#)

		Actual Classes	
		Positive	Negative
Predicted Classes	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Some summary statistics of the confusion matrix are

$$\text{ACCURACY} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{ERROR} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Accuracy and error are usually reported as percentages

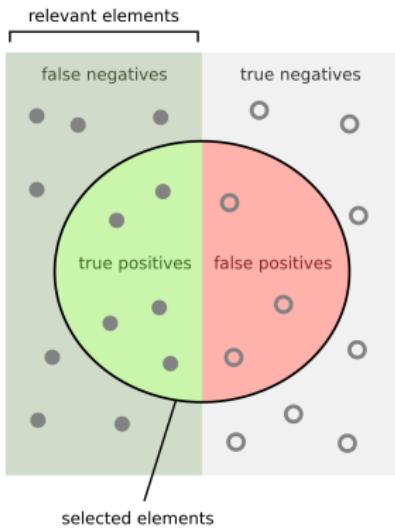
# Classification: Evaluation Metrics

		Actual Classes	
		Positive	Negative
Predicted Classes	Positive	True Positive	False Positive
	Negative	False Negative	True Negative



- With big imbalance in classes, ACCURACY and ERROR are misleading
- e.g. in a tumors dataset 99% samples are negative
- (Blindly) predicting all as negatives gives 99% accuracy
- But cancer is not detected
- Have to use cost matrix/loss function (essentially weighted accuracy)

# Classification: Evaluation Metrics



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

$$\text{PRECISION} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

sensitivity (measure of exactness)

$$\text{RECALL} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

specificity (measure of completeness)

■ *F-measure*: Maximize both

$$\blacksquare F_1 = \frac{2}{\frac{1}{\text{PRECISION}} + \frac{1}{\text{RECALL}}}$$

■ *F<sub>k</sub>* measure weighs PRECISION and RECALL differently

■ Weighted harmonic mean

# Classifier/Model

---

- A classifier utilizes training data to understand how input variables are related to the class variable
- A **model** is built, which can be used to predict labels for unseen data
- Kinds of Classifiers
  - **Lazy Classifiers**
  - **Eager Classifiers**

# Kinds of Classifiers

---

## ■ Lazy Classifiers:

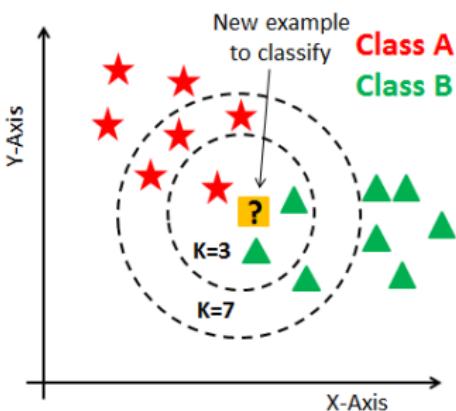
- Store the training data and wait for testing data
- For an unseen test data record (data point), assign class label based on the most related points in the training data
- Less training time, more prediction time
- Examples: *k*-Nearest Neighbor (kNN) Classifier

## ■ Eager Classifiers:

- Construct a classification model based on training data
- For a test data point, use the model to assign class label
- More training time but less prediction time
- Examples: Naive Bayes, Decision Tree

# k-Nearest Neighbor (kNN)

- k-NN is a simple method used for regression and classification
- In **k-NN regression**, for a test instance  $x$  the value of target variable  $y$  is the 'average' of values of  $y$  of  **$k$ -nearest neighbors of  $x$**  in train set
- In **k-NN classification** the class label of a test instance  $x$  is the most common class among the  **$k$  nearest neighbors of  $x$**  in the train set



- Assign the test instance (?) class A (★) or class B (▲)
- $k = 3$  nearest neighbors ( $\ell_2$  distance)  
1 ★ and 2 ▲  $\implies$  assigned label = ▲
- $k = 7$  nearest neighbors ( $\ell_2$  distance)  
4 ★ and 3 ▲  $\implies$  assigned label = ★

## k-Nearest Neighbor (kNN)

- In **k-NN classification** the class label of a test instance  $x$  is the most common class among the  **$k$  nearest neighbors of  $x$**  in the train set
- Assumes that the proximity measure captures class membership
- Thus, definition of proximity measure (defining ‘nearest’) is critical
- The parameter  $k$  is important and sensitive to local structure of data
- In **k-NN regression**, the ‘average’ can be the weighted mean (weighted by similarity), in this case generally take  $k$  so all points are included in neighborhood

$$y(x) = \frac{\sum_{x' \in D} sim(x, x') y(x')}{\sum_{x' \in D} sim(x, x')}$$

▷  $y$  is the target variable for instance  $x$

# Naive Bayes Classifier

---

- Classify  $\mathbf{x} = (x_1, \dots, x_n)$  into one of  $K$  classes  $C_1, \dots, C_K$
- Naive Bayes is a conditional probability model
  - It computes probabilities  $Pr[\text{class} = C_j | \mathbf{x}]$  for each class  $C_j$
- Assumes that
  - 1 All attributes are equally important
  - 2 Attributes are statistically independent given the class label
    - knowing value of one attribute says nothing about value of another
- Independence assumption is almost never correct (thus the word Naive) but works well in practice
- Model is the probabilities calculated from training data for each attribute with respect to class label

# Naive Bayes Classifier

Classify  $\mathbf{x} = (x_1, \dots, x_n)$  into one of  $K$  classes  $C_1, \dots, C_K$

We want to compute this. The **Posterior probability** of class  $C_j$  given the object  $\mathbf{x}$

The **Likelihood**: Probability of predictor(s) given a class  $C_j$ . Computed from frequencies of predictor(s) in class  $C_j$  in train set

**Prior**: Probability of class  $C_j$ , without considering  $\mathbf{x}$ . Estimated from frequency of labels  $C_j$  in train set

$$P(C_j | \mathbf{x}) = \frac{P(\mathbf{x} | C_j) \times P(C_j)}{P(\mathbf{x})}$$

Apply the independence assumption

$$P(\mathbf{x} | C_j) = P(x_1 | C_j) \times P(x_2 | C_j) \times \dots \times P(x_n | C_j)$$

Substitute in numerator and ignore the denominator

$$P(C_j | \mathbf{x}) = P(x_1 | C_j) \times P(x_2 | C_j) \times \dots \times P(x_n | C_j) \times [P(C_j)]$$

## Naive Bayes: Running Example

Train on records of weather conditions and whether or not game was played.  
Given weather condition (test instance) predict whether game will be played

Outlook	Temp.	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

# Naive Bayes: Running Example

$$P(\text{play} = \text{yes} | \mathbf{x}) = P(\text{outl} = *|\text{yes}) \times P(\text{temp} = *|\text{yes}) \times P(\text{humid} = *|\text{yes}) \times P(\text{Wind} = *|\text{yes}) \times [P(\text{yes})]$$

$$P(\text{play} = \text{no} | \mathbf{x}) = P(\text{outl} = *|\text{no}) \times P(\text{temp} = *|\text{no}) \times P(\text{humid} = *|\text{no}) \times P(\text{Wind} = *|\text{no}) \times [P(\text{no})]$$

Outlook	Temp.	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	false	yes
overcast	cool	normal	false	yes
sunny	mild	high	false	yes
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	false	yes
overcast	mild	high	false	yes
overcast	hot	normal	false	yes
rainy	mild	high	false	yes

Outlook	yes	no
sunny	2	3
overcast	4	0
rainy	3	2
TOTAL	9	5

# Naive Bayes: Running Example

Play		Outlook	yes	no
Outlook	yes			
sunny	2	3		
overcast	4	0		
rainy	3	2		
TOTAL	9	5		



Outlook	Temp.	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

For each attribute...

Play		Play		Play		Play		Play		Play		Play
Outlook	yes	no	Temp.	yes	no	Humid.	yes	no	Windy	yes	no	
sunny	2	3	hot	2	2	high	3	4	false	6	2	yes 9
overcast	4	0	mild	4	2	normal	6	1	true	3	3	no 5
rainy	3	2	cool	3	1							
TOTAL	9	5	TOTAL	9	5	TOTAL	9	5	TOTAL	9	5	TOTAL 14

# Naive Bayes: Running Example

	Play			Play			Play			Play			Play
Outlook	yes	no	Temp.	yes	no	Humid.	yes	no	Windy	yes	no		
sunny	2	3	hot	2	2	high	3	4	false	6	2	yes	9
overcast	4	0	mild	4	2	normal	6	1	true	3	3	no	5
rainy	3	2	cool	3	1								
TOTAL	9	5	TOTAL	9	5	TOTAL	9	5	TOTAL	9	5	TOTAL	14



Convert values to ratios

	Play			Play			Play			Play			Play
Outlook	yes	no	Temp.	yes	no	Humid.	yes	no	Windy	yes	no		
sunny	0.22	0.60	hot	0.22	0.40	high	0.33	0.80	false	0.67	0.40	yes	0.64
overcast	0.44	0.00	mild	0.44	0.40	normal	0.67	0.20	true	0.33	0.60	no	0.36
rainy	0.33	0.40	cool	0.33	0.20								

2 occurrences of Play = no, where Outlook = rainy  
5 occurrences of Play = no

# Naive Bayes: Running Example

Given weather condition  $\mathbf{x} = (\text{sunny}, \text{cool}, \text{high}, \text{true})$ , will game be played?

$$\begin{aligned}P(\text{play} = \text{yes} | \mathbf{x}) &= P(\text{sunny}|\text{yes}) \times P(\text{cool}|\text{yes}) \times P(\text{high}|\text{yes}) \times P(\text{true}|\text{yes}) \times [P(\text{yes})] \\&= 0.22 \times 0.33 \times 0.33 \times 0.33 \times [0.64] = \mathbf{0.0053}\end{aligned}$$

$$\begin{aligned}P(\text{play} = \text{no} | \mathbf{x}) &= P(\text{sunny}|\text{no}) \times P(\text{cool}|\text{no}) \times P(\text{high}|\text{no}) \times P(\text{true}|\text{no}) \times [P(\text{no})] \\&= 0.60 \times 0.20 \times 0.80 \times 0.80 \times [0.36] = \mathbf{0.0206}\end{aligned}$$

	Play		Play		Play		Play		Play			
Outlook	yes	no	Temp.	yes	no	Humid.	yes	no	Windy	yes	no	Play
sunny	0.22	0.60	hot	0.22	0.40	high	0.33	0.80	false	0.67	0.40	yes 0.64
overcast	0.44	0.00	mild	0.44	0.40	normal	0.67	0.20	true	0.33	0.60	no 0.36
rainy	0.33	0.40	cool	0.33	0.20							

## Naive Bayes: Issues

---

Some issues for Naive Bayes classifier: you are encouraged to read about

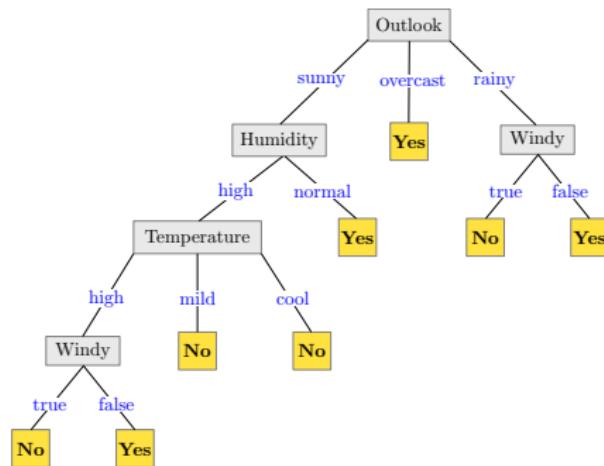
- **Zero frequency problem:** probability = 0 for an attribute in a class
- For example:  $P[Outlook = \text{sunny} | \text{yes}] = 0$
- One zero would make whole product zero
- Solution: Laplace smoothing (add-one smoothing)
- Missing value of an attribute for a test instance
- usually attribute is omitted from probability calculation
- Another issue with Naive Bayes
- **What if values of attributes are continuous?**
- Discretization solves the problem in many cases
- Can also assume a probability distribution for each continuous attribute and learn distribution parameters from training set

## Decision Tree

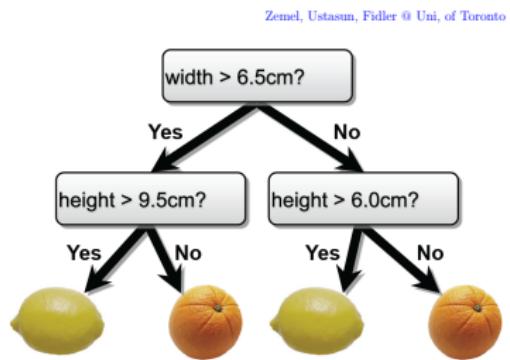
---

- Fundamentally, an **if-then** rule set for classifying objects
- Builds model in the form of a tree structure

# Decision Tree



Decision tree for binary classification of instance with nominal attributes

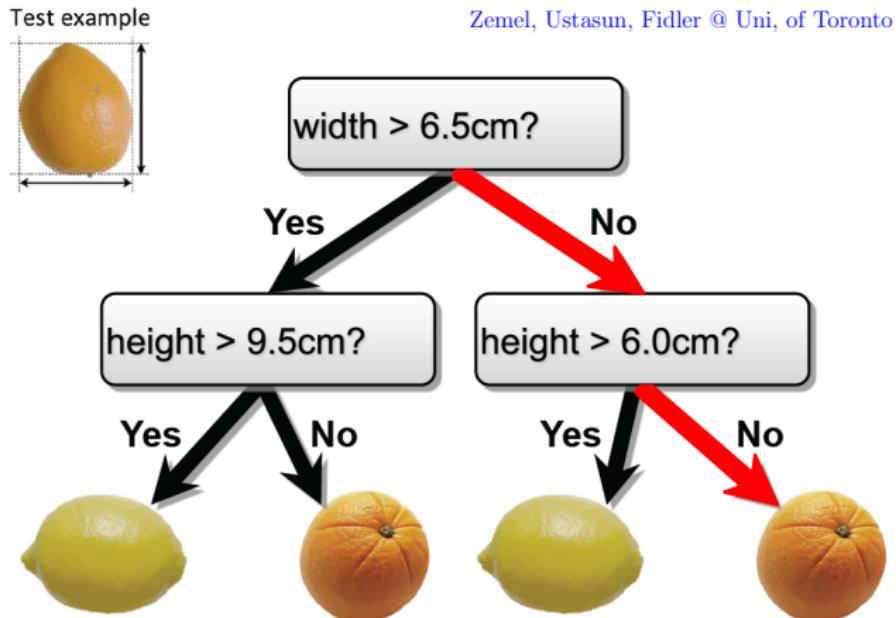


Decision tree for binary classification of instance with numeric attributes

- Each internal node tests an attribute  $x_i$
- Branches correspond to possible (subsets of) value of  $x_i$
- Each leaf node assigns a class label  $y$

# Classification using Decision Trees

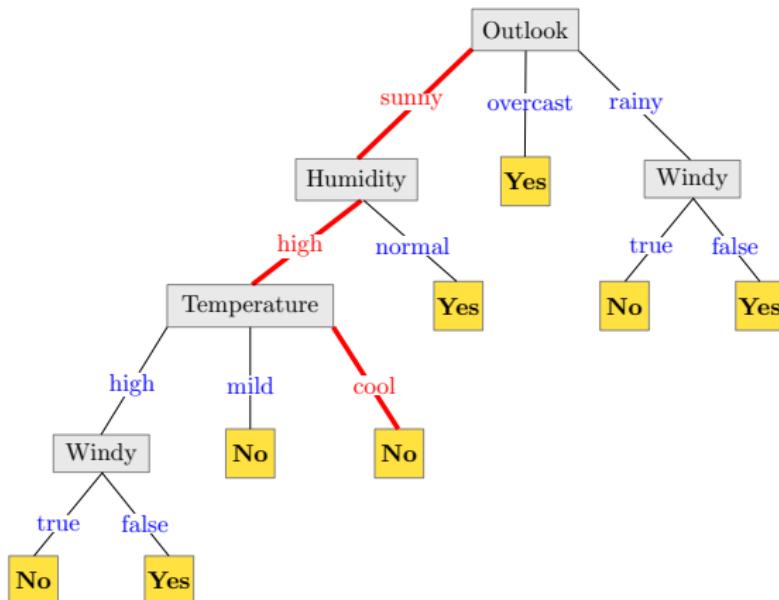
- To classify a test instance  $x$  traverse the tree from root to leaf
- Take branches at internal nodes according to results of their tests
- Predict the class label at the leaf node reached



# Classification using Decision Trees

- To classify a test instance  $x$  traverse the tree from root to leaf
- Take branches at internal nodes according to results of their tests
- Predict the class label at the leaf node reached

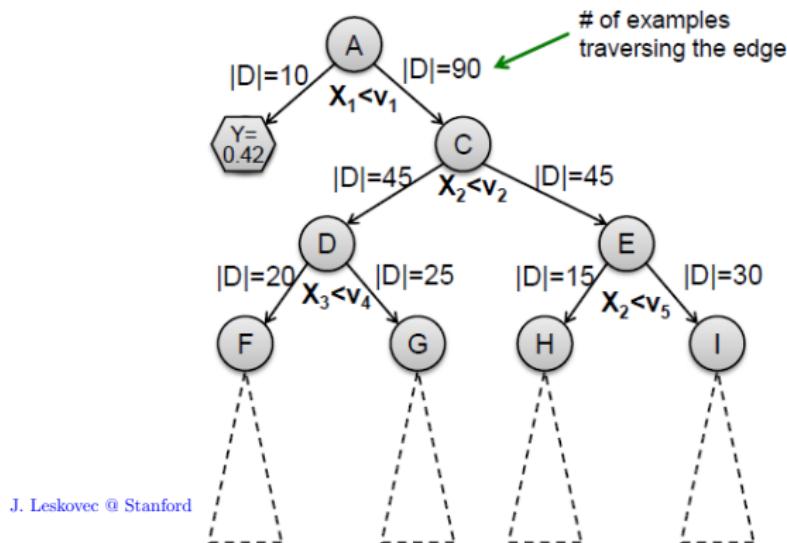
Given weather condition  $\mathbf{x} = (\text{sunny}, \text{cool}, \text{high}, \text{true})$ , will game be played?



# Building Decision Tree

Building the best decision tree is NP-HARD problem

**Training dataset  $D^*$ ,  $|D^*| = 100$  examples**

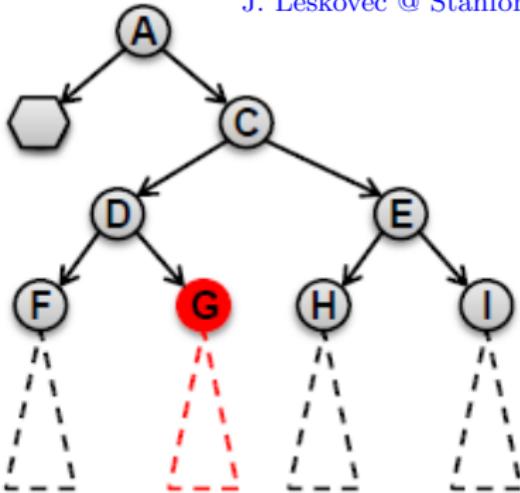


- Recursively build the tree top-down, using greedy heuristics
- Start with an empty decision tree
- Split the **current dataset** by **the best attribute** until stopping condition

# Building Decision Tree

- Suppose at some node  $G$  in the tree built so far

J. Leskovec @ Stanford



- Shall we continue building the tree?

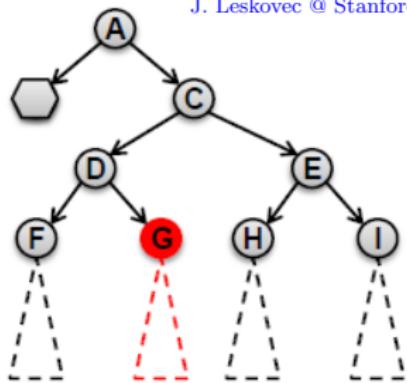
- If Yes,  $G$  is internal, which attribute to split on (test)?
- If No,  $G$  is leaf, what is the prediction rule?

# Building Decision Tree

Stop when the leaf (subtree at  $G$ )

- is pure (purity?) or
- When the size of sub-dataset at  $G$  is small e.g.  $|D_G| \leq 5$
- ...

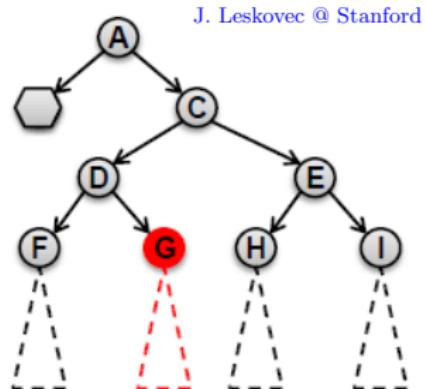
J. Leskovec @ Stanford



# Building Decision Tree

If we stop at  $G$ , then prediction at  $G$  can be

- mode of class labels in the sub-dataset  $D_G$
- For a numeric target variable prediction could be an average of target variable values in  $D_G$
- (in case target variable is numeric it is called **Regression Tree**)

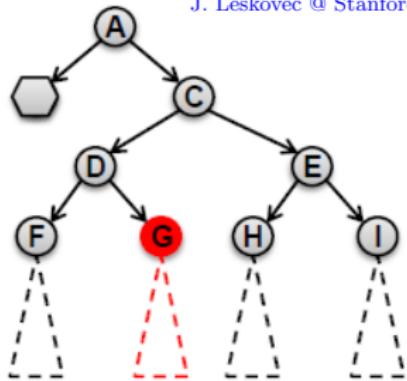


# Building Decision Tree

Top attributes are selected based on metrics such as:

- Entropy
- Information Gain
- Gini Index

J. Leskovec @ Stanford



Common algorithms for Decision Tree are ID3, C4.5, ...

# Entropy

In information theory, **entropy** quantify the average level of information content or uncertainty in a random variable

Flip a fair and a biased coin and the outcome is

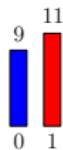
Outcome of Coin 1

1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 ... ?



Outcome of Coin 2

1 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 ... ?



In which case would we be more surprised if the next outcome is a 1?

Max surprise is for fair coin ( $p = 1/2$ ) (no reason to expect one outcome more than other), entropy is maximum = 1 bit (unit of entropy). Min entropy value is 0 bit for  $p = 0$  or  $p = 1$

These slides about information theory concepts are adapted from Grosse, Farahmand, & Carrasquilla, Uni. of Toronto

# Entropy

In general, for a random variable  $X$  taking value  $x_1, \dots, x_n$  has entropy

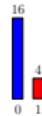
$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i)$$

- For fair coin  $p = 1/2$ ,  $H(\cdot) = -1/2 \log 1/2 - 1/2 \log 1/2 = 1$
- For one-sided coin  $p = 1$  or  $p = 0$ ,  $H(\cdot) = -1 \log 1 - 0 \log 0 = 0$
- For coin-1 in example  $H(\cdot) = -16/20 \log 16/20 - 4/20 \log 4/20 = 0.721928$
- For coin-2 in example  $H(\cdot) = -9/20 \log 9/20 - 11/20 \log 11/20 = 0.99277$

Flip a fair and a biased coin and the outcome is

Outcome of Coin 1

1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 ... ?



Outcome of Coin 2

1 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 ... ?



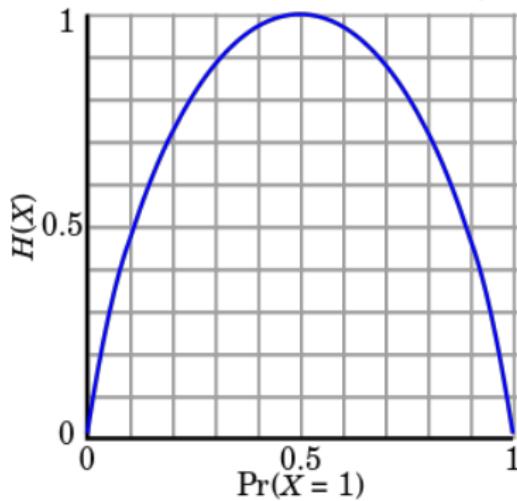
In which case would we be more surprised if the next outcome is a 1?

# Entropy

In general, for a random variable  $X$  taking value  $x_1, \dots, x_n$  has entropy

$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i)$$

source: Wikipedia



Entropy  $H(X)$  (i.e. the expected surprisal) of a coin flip,  
measured in bits, graphed versus the bias of the coin  
 $\Pr(X = 1)$ , where  $X = 1$  represents a result of heads

## Entropy of joint distribution

Entropy of the joint distribution of random variables  $X$  and  $Y$

- Example:  $X = \{\text{Raining, Not raining}\}$ ,  $Y = \{\text{Cloudy, Not cloudy}\}$

		Cloudy	Not Cloudy
Raining	24/100	1/100	
Not Raining	25/100	50/100	

$$\begin{aligned}H(X, Y) &= - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(x, y) \\&= -\frac{24}{100} \log_2 \frac{24}{100} - \frac{1}{100} \log_2 \frac{1}{100} - \frac{25}{100} \log_2 \frac{25}{100} - \frac{50}{100} \log_2 \frac{50}{100} \\&\approx 1.56 \text{ bits}\end{aligned}$$

Grosse, Farahmand, & Carrasquilla, Uni. of Toronto

# Conditional Entropy

- Example:  $X = \{\text{Raining, Not raining}\}$ ,  $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- What is the entropy of cloudiness  $Y$ , given that it is raining?

$$\begin{aligned}H(Y|X=x) &= -\sum_{y \in Y} p(y|x) \log_2 p(y|x) \\&= -\frac{24}{25} \log_2 \frac{24}{25} - \frac{1}{25} \log_2 \frac{1}{25} \\&\approx 0.24 \text{ bits}\end{aligned}$$

- We used:  $p(y|x) = \frac{p(x,y)}{p(x)}$ , and  $p(x) = \sum_y p(x,y)$  (sum in a row)

Grosse, Farahmand, & Carrasquilla, Uni. of Toronto

# Conditional Entropy

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- The expected conditional entropy:

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X=x) \\ &= - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y|x) \end{aligned}$$

# Conditional Entropy

- Example:  $X = \{\text{Raining, Not raining}\}$ ,  $Y = \{\text{Cloudy, Not cloudy}\}$

		Cloudy	Not Cloudy
Raining	24/100	1/100	
Not Raining	25/100	50/100	

- What is the entropy of cloudiness, given the knowledge of whether or not it is raining?

$$\begin{aligned}H(Y|X) &= \sum_{x \in X} p(x)H(Y|X=x) \\&= \frac{1}{4}H(\text{cloudy}| \text{is raining}) + \frac{3}{4}H(\text{cloudy}| \text{not raining}) \\&\approx 0.75 \text{ bits}\end{aligned}$$

Grosse, Farahmand, & Carrasquilla, Uni. of Toronto

# Conditional Entropy

---

- Some useful properties:

- ▶  $H$  is always non-negative
- ▶ Chain rule:  $H(X, Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$
- ▶ If  $X$  and  $Y$  independent, then  $X$  doesn't tell us anything about  $Y$ :  
 $H(Y|X) = H(Y)$
- ▶ But  $Y$  tells us everything about  $Y$ :  $H(Y|Y) = 0$
- ▶ By knowing  $X$ , we can only decrease uncertainty about  $Y$ :  
 $H(Y|X) \leq H(Y)$

Grosse, Farahmand, & Carrasquilla, Uni. of Toronto

# Information Gain

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

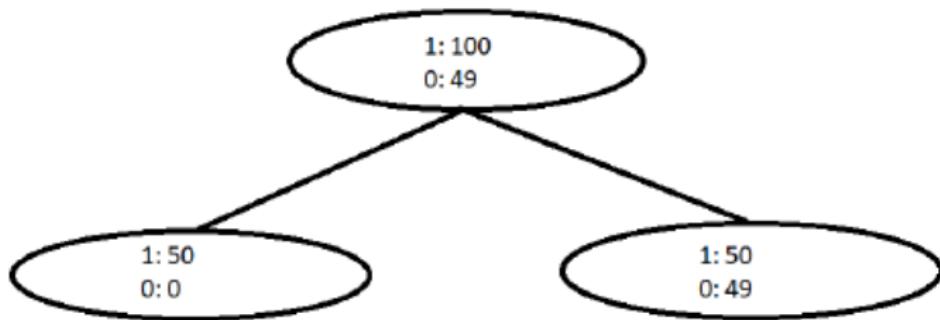
- How much information about cloudiness do we get by discovering whether it is raining?

$$\begin{aligned}IG(Y|X) &= H(Y) - H(Y|X) \\&\approx 0.25 \text{ bits}\end{aligned}$$

- This is called the information gain in  $Y$  due to  $X$ , or the mutual information of  $Y$  and  $X$
- If  $X$  is completely uninformative about  $Y$ :  $IG(Y|X) = 0$
- If  $X$  is completely informative about  $Y$ :  $IG(Y|X) = H(Y)$

# Information Gain

- Information gain measures the informativeness of a variable, which is exactly what we desire in a decision tree attribute!
- What is the information gain of this split?



- Root entropy:  $H(Y) = -\frac{49}{149} \log_2\left(\frac{49}{149}\right) - \frac{100}{149} \log_2\left(\frac{100}{149}\right) \approx 0.91$
- Leaf's entropy:  $H(Y|left) = 0$ ,  $H(Y|right) \approx 1$ .
- $IG(split) \approx 0.91 - \left(\frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 1\right) \approx 0.24 > 0$

Grosse, Farahmand, & Carrasquilla, Uni. of Toronto

## Classification: Some other Concepts

---

Some other concepts related to classification you should be familiar with

- Decision boundary
- ROC-Curve
- Multi-Class classification from binary classifier
  - - ONE-VS-ALL
    - ONE-VS-REST
- Some classification algorithms that you should read about, at least wikipedia level reading is essential in order to be able to read papers discussing them and use them in your projects
- In particular, you should also know
- Random Forest, Support Vector Machine, Neural Networks, Deep Learning