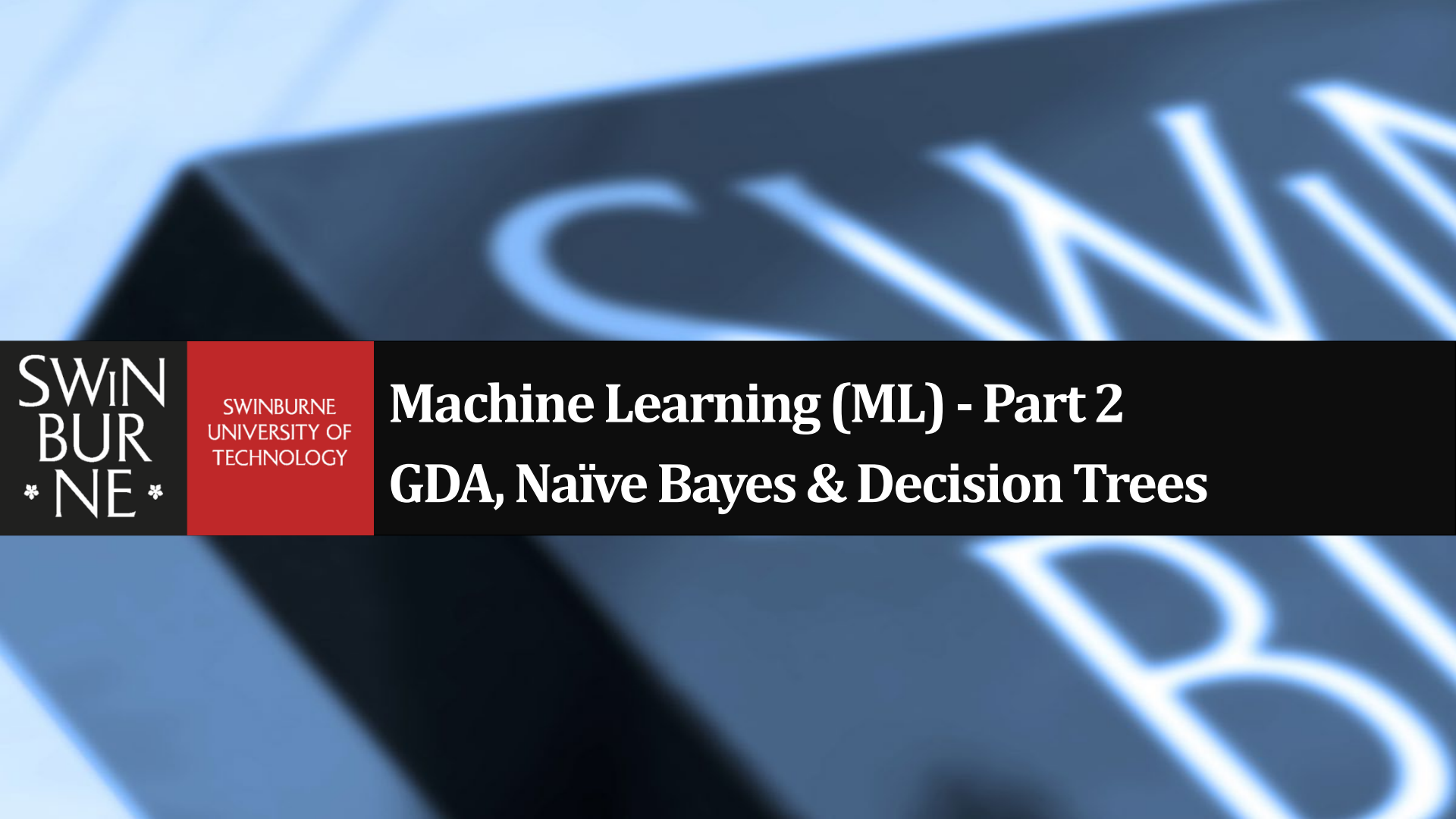


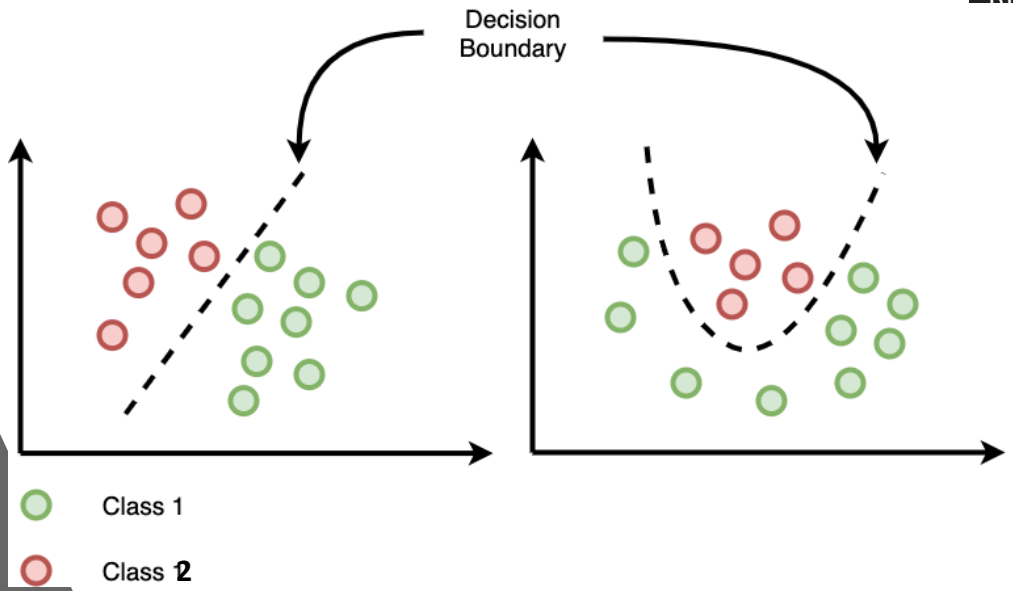
Machine Learning (ML) - Part 2

GDA, Naïve Bayes & Decision Trees

Overview

- Gaussian Discriminant Analysis (GDA)
- Naïve Bayes
- Decision Trees

Discriminative vs. Generative Learning



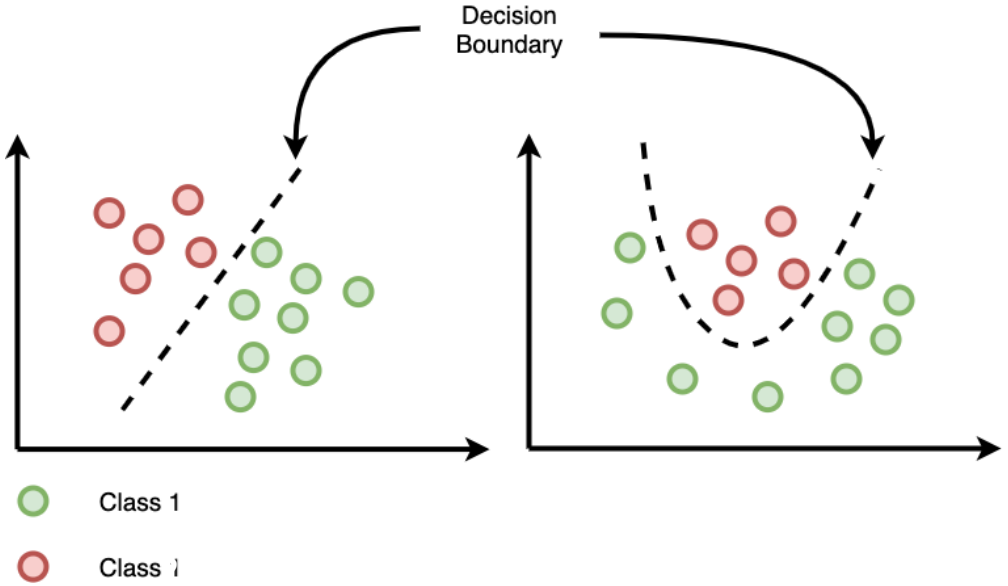
- Logistic regression can be used to identify such decision boundaries using MLE

→ **Discriminative learning**

Generative learning?

- What about we look at each class (1 & 2) separately

- Can we generate a model of the input variables x when y = 0 (**Class 1**)?
- E.g., For a Benign tumor (y=0), what would be the typical tumor_size and clump_density?
- Similarly, model of x when y = 1??



But why would we want to do that?

- Isn't the purpose of ML is to get the input data of an unseen instance and try to classify this new instance??

- This is when Bayes' rule will be used:

$$P(y = 1|x) = \frac{P(x|y = 1)P(y = 1)}{P(x)}$$

- And of course:

$$P(x) = P(x|y = 1)P(y = 1) + P(x|y = 0)P(y = 0)$$

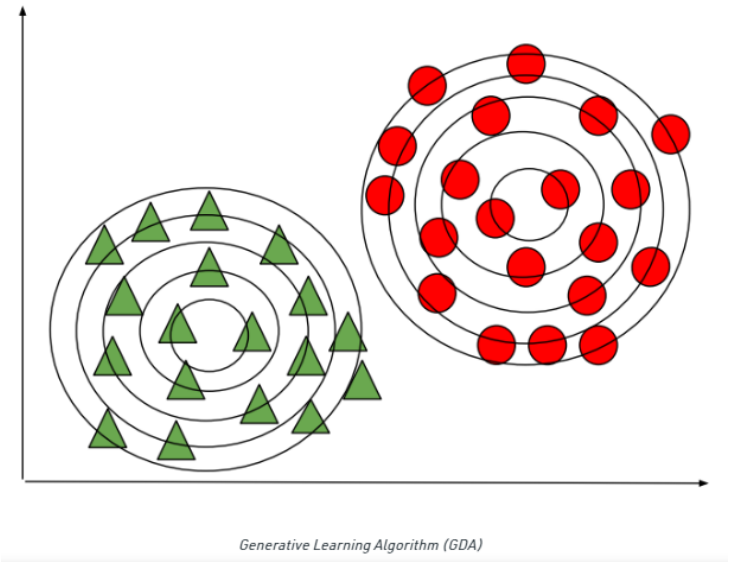
And, how?

- $P(y = 1)$ and $P(y = 0)$ are "innocent" enough: just the ratios of instances with a classification of 1 (and 0, respectively) *from the training set*!
  - You'll see later that it is another parameter that we want to train!

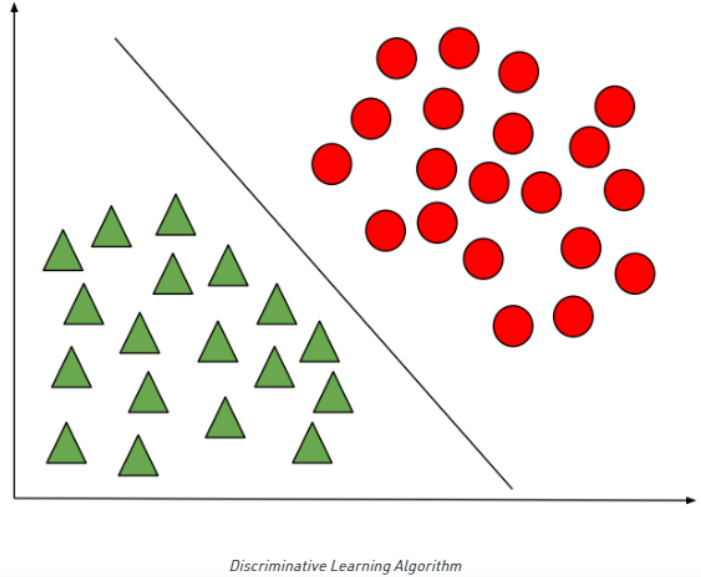
- $P(x|y = 1)$ and $P(x|y = 0)$??

- Enter **Gaussian Discriminant Analysis (GDA)**, a Generative Learning Algorithm.

Let's visualise it: (source: www.geeksforgeeks.org/)



Generative Learning Algorithm (GDA)



Discriminative Learning Algorithm

- The two sets of contours represent a generative learning model (e.g. GDA): the circle 1-contours and the triangles 0-contours.

Want some maths? (source: Andrew Ng, Stanford)

$$P(x|y=0) = \frac{1}{(2\pi)^{n/2} * |\Sigma|^{1/2}} exp(-1/2(x-\mu_0)^T \Sigma^{-1}(x-\mu_0)) \quad - \textbf{Eq 1}$$

$$P(x|y=1) = \frac{1}{(2\pi)^{n/2} * |\Sigma|^{1/2}} exp(-1/2(x-\mu_1)^T \Sigma^{-1}(x-\mu_1)) \quad - \textbf{Eq 2}$$

$$P(y) = \phi^y.(1-\phi)^{1-y} \quad - \textbf{Eq 3}$$

- Simply speaking:
  - μ_0 is the centre of the 0-contours (mean of the circle samples)
  - μ_1 is the centre of the 1-contours (mean of the triangle samples)
  - Σ (capital sigma) is the co-variance matrix of the contours
  - ϕ = P(y=1)
    - Remember we promised that we would also learn P(y=1)?

Want some maths? (source: Andrew Ng, Stanford)

- Now, given a training set $\{x^{(i)}, y^{(i)}\}_{i=1..m}$
- The joint likelihood of the parameters:

$$L(\phi, \mu_0, \mu_1, \Sigma) = \Pi_{i=1}^{m} P(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma)$$
$$= \Pi_{i=0}^{m} P(x^{(i)}|y^{(i)}).P(y^{(i)}) \quad - \mathbf{Eq} \quad \mathbf{4}$$

- According to the principle of MLE we have to choose ϕ, μ_0, μ_1, Σ to maximize function $L(\phi, \mu_0, \mu_1, \Sigma)$ above. To do so instead of maximizing the Likelihood function we can maximize the Log-Likelihood Function which is a strictly increasing function.

$$\log(L(\phi, \mu_0, \mu_1, \Sigma))$$

-

Want some maths? (source: Andrew Ng, Stanford)

- And the result is:

$$\phi = \frac{1}{n}\sum_{i=1}^{n} 1\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\sum_{i=1}^{n} 1\{y^{(i)} = 0\}x^{(i)}}{\sum_{i=1}^{n} 1\{y^{(i)} = 0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^{n} 1\{y^{(i)} = 1\}x^{(i)}}{\sum_{i=1}^{n} 1\{y^{(i)} = 1\}}$$

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T.$$

- (well, the Stanford students are actually required to prove mathematically that these equations are the results of maximizing the log likelihood function!!)

where 1{} is the indicator function 1{true} = 1 and 1{false} = 0.

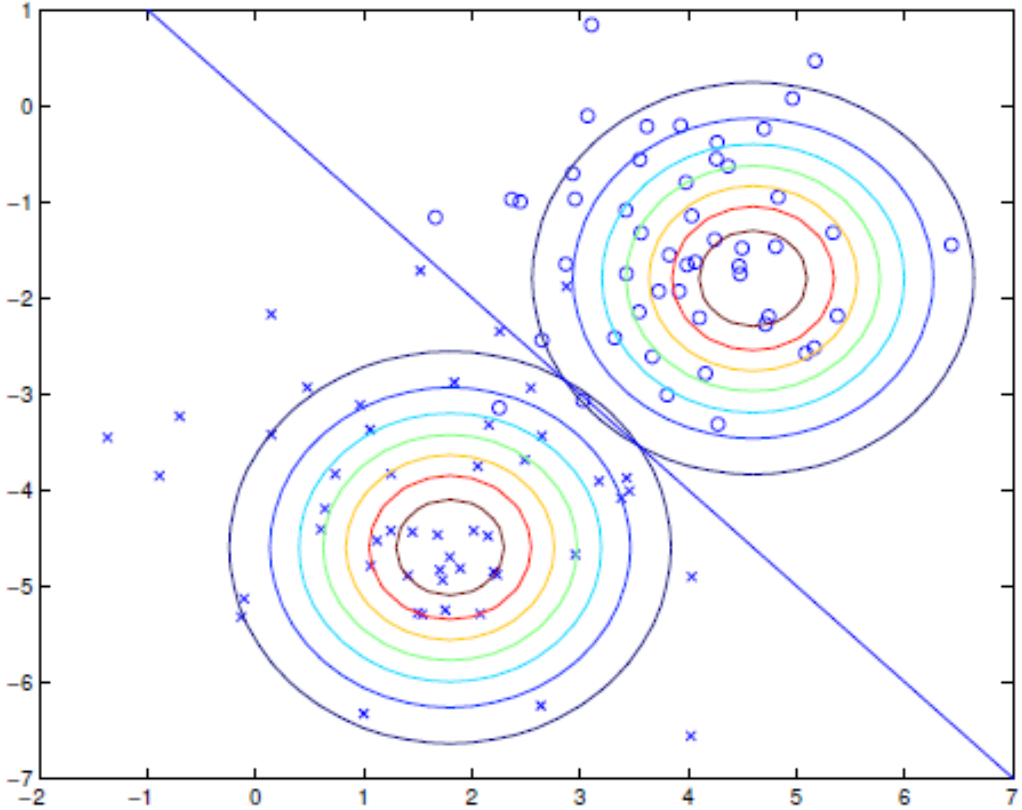
Let's go back to our original question

- Isn't the purpose of ML is to get the input data of an unseen instance and try to classify this new instance??

$$y_{new} = \underset{y}{\text{argmax}}\, P(y|x_{new}) = \underset{y}{\text{argmax}} \frac{P(x_{new}|y)P(y)}{P(x_{new})} = \underset{y}{\text{argmax}}(P(x_{new}|y)P(y))$$

and, why do we ignore P(x_{new}) in the last expression?

Let's visualise it: (source: Andrew Ng, Stanford)



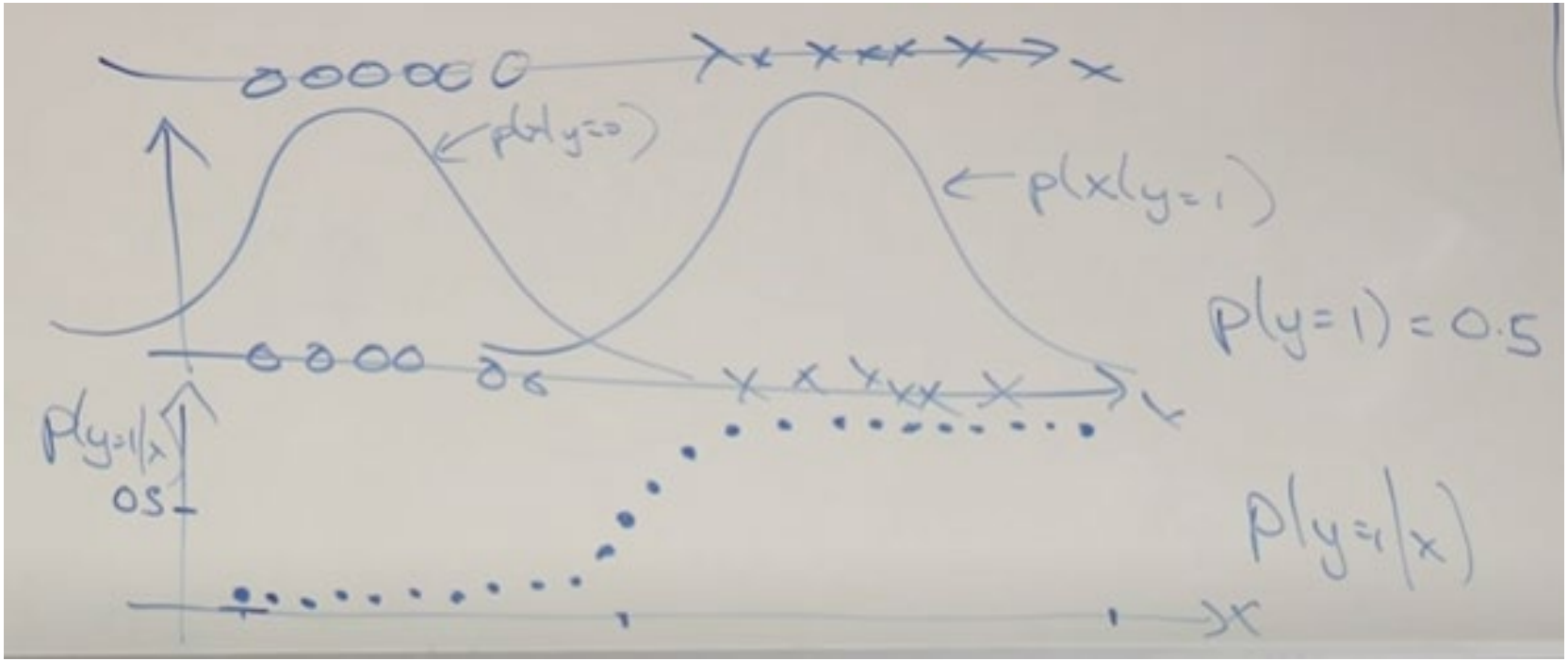
- Based on the classification derived in the preceding slide, this generative model also give you a decision boundary (the blue line) which may be slightly different from the decision boundary obtained by logistic regression.

Let's visualise it: (source: Andrew Ng, Stanford)

- For instance, the decision boundary obtained by Log Reg may look like the green line vs the blue line (GDA-decision boundary)

How do we compare GDA & Log Reg?

(source: Andrew Ng, Stanford)



How do we compare GDA & Log Reg?

(source: Andrew Ng, Stanford)

- Turns out that if you use the generative model GDA then your prediction of the target variable P(y = 1 |x) is a logistic function. To summarise:

- GDA $\begin{cases} x|\, y = 1 \sim \aleph(\mu_1, \Sigma) \\ x|\, y = 0 \sim \aleph(\mu_0, \Sigma) \\ \quad y \sim Ber(\phi) \end{cases} \Rightarrow P(y = 1\,|x) = \frac{1}{1+ e^{-gW(x)}}$

- Interestingly, the following generative model:

- $\begin{cases} x|\, y = 1 \sim Pois(\lambda_1) \\ x|\, y = 0 \sim Pois(\lambda_2) \\ \quad y \sim Ber(\phi) \end{cases} \Rightarrow P(y = 1\,|x) = \frac{1}{1+ e^{-gW(x)}}$

Naïve Bayes algorithm

- It is also a **generative** learning algorithm
- Then, why can't we just stick with GDA???
- If instead of 3, 4 input variables x_1, x_2, x_3, x_4, we actually have 10,000 input variables!!
- A very common situation in natural language processing (NLP) tasks
    - E.g., How can email service providers (such as Gmail) classify whether an email is spam vs non-spam?

- Question 1: How do we represent our input data (emails) as numerical input variables to use the ML algorithms we saw??

Naïve Bayes algorithm

- Answer: Use *feature vector*.
- Given a 10,000-word English dictionary (e.g., the top-10,000 English words seen in emails)
- The feature vector:

- $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{10,000} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 1\{\begin{bmatrix} a \\ aardvark \\ aardwolf \\ \vdots \\ zymurgy \end{bmatrix}\}$

where $1\{\}$ is again the indicator function.

Naïve Bayes algorithm

- This is when Naïve Bayes can be a useful algorithm by assuming that $x_1, x_2, x_3, \dots, x_{10,000}$ are **conditionally independent** given y
- That is,

$$P(x_1, x_2, x_3, \dots, x_{10,000} | y)$$

$$= P(x_1|y)\, P(x_2 | y, x_1)\, P(x_3|y, x_1, x_2) \dots P(x_{10,000} | y, x_1, x_2, x_3, \dots, x_{9,999})$$

$$= (\text{C.I.}) = P(x_1|y)\, P(x_2 | y)\, P(x_3|y) \dots P(x_{10,000} | y) =$$

$$= \prod_{i=1}^{10,000} P(x_i|y)$$

Naïve Bayes algorithm

- Now, let's talk about the parameters of this model:
- $\phi_{j|y=1} = P(x_j = 1|y = 1)$
- $\phi_{j|y=0} = P(x_j = 1|y = 0)$
- $\phi_y = P(y = 1)$

- Then, joint likelihood of the parameters:

$$\mathcal{L}(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) = \prod_{i=1}^{n} p(x^{(i)}, y^{(i)}).$$

Naïve Bayes algorithm

- Maximizing this with respect to ϕ_y, $\phi_{j|y=0}$ and $\phi_{j|y=1}$ gives the maximum likelihood estimates:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{n} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{n} 1\{y^{(i)} = 1\}}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{n} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{n} 1\{y^{(i)} = 0\}}$$

$$\phi_y = \frac{\sum_{i=1}^{n} 1\{y^{(i)} = 1\}}{n}$$

Decision tree

Source: Tom Mitchell, 2010

- Introduction
- Decision tree representation
- Learning algorithm

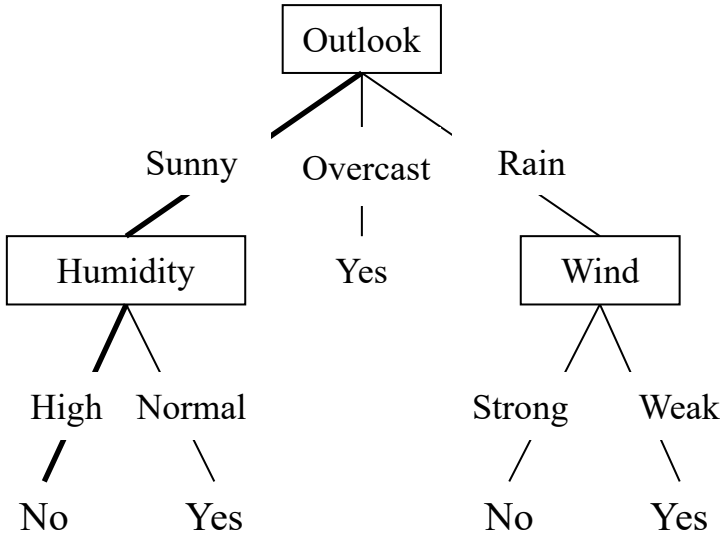
Examples of Decision Tree

- Data set

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

Examples of Decision Tree

Play tennis?



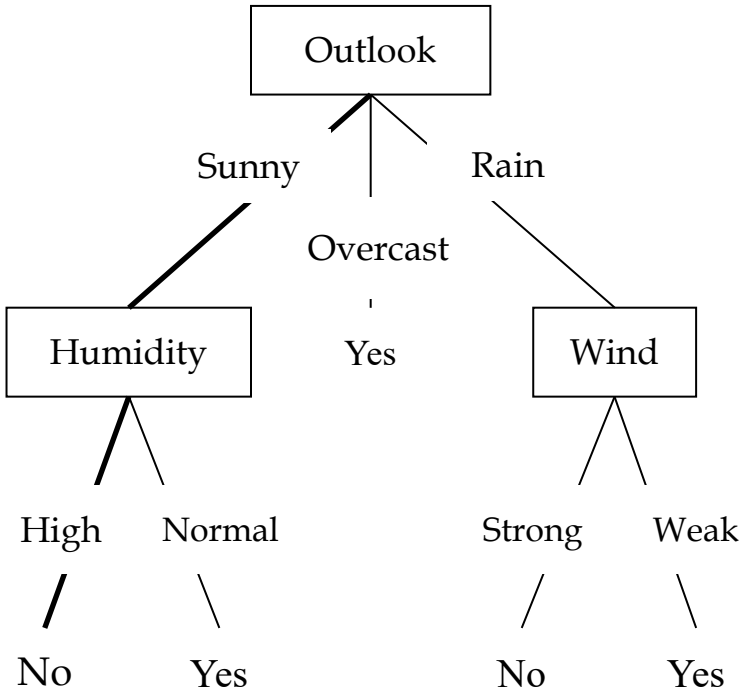
Introduction

- Decision tree learning
  - One of the most widely used and practical methods for inductive inference
  - Approximate discrete-valued target function
    - The learned function is represented by a decision tree
    - Decision tree can also be re-represented as sets of if-then rules to improve human readability
  - Robust to noisy data and capable of learning disjunctive expressions

Decision Tree Representation (1/2)

- Classification of instances
  - Decision tree classify instances by sorting them down the tree from the root to some leaf node
- Node
  - Specifies test of some attribute
- Branch
  - Corresponds to one of the possible values for this attribute

Decision Tree Representation (2/2)



- e.g.
  - Saturday morning
    (*Outlook=sunny, Temperature=Hot, Humidity=high, Wind=Strong*)➔
    (*Outlook=Sunny ∧ Humidity=High*) so *NO*
- Each path corresponds to a conjunction of attribute tests
- Decision trees represent a disjunction of conjunction of constraints on the attribute values of instances
  - *(Outlook=Sunny ∧Humidity=normal)*

    ∨(Outlook=Overcast)

    ∨(Outlook=Rain ∧Wind=Weak)

Appropriate Problems for Decision Tree Learning

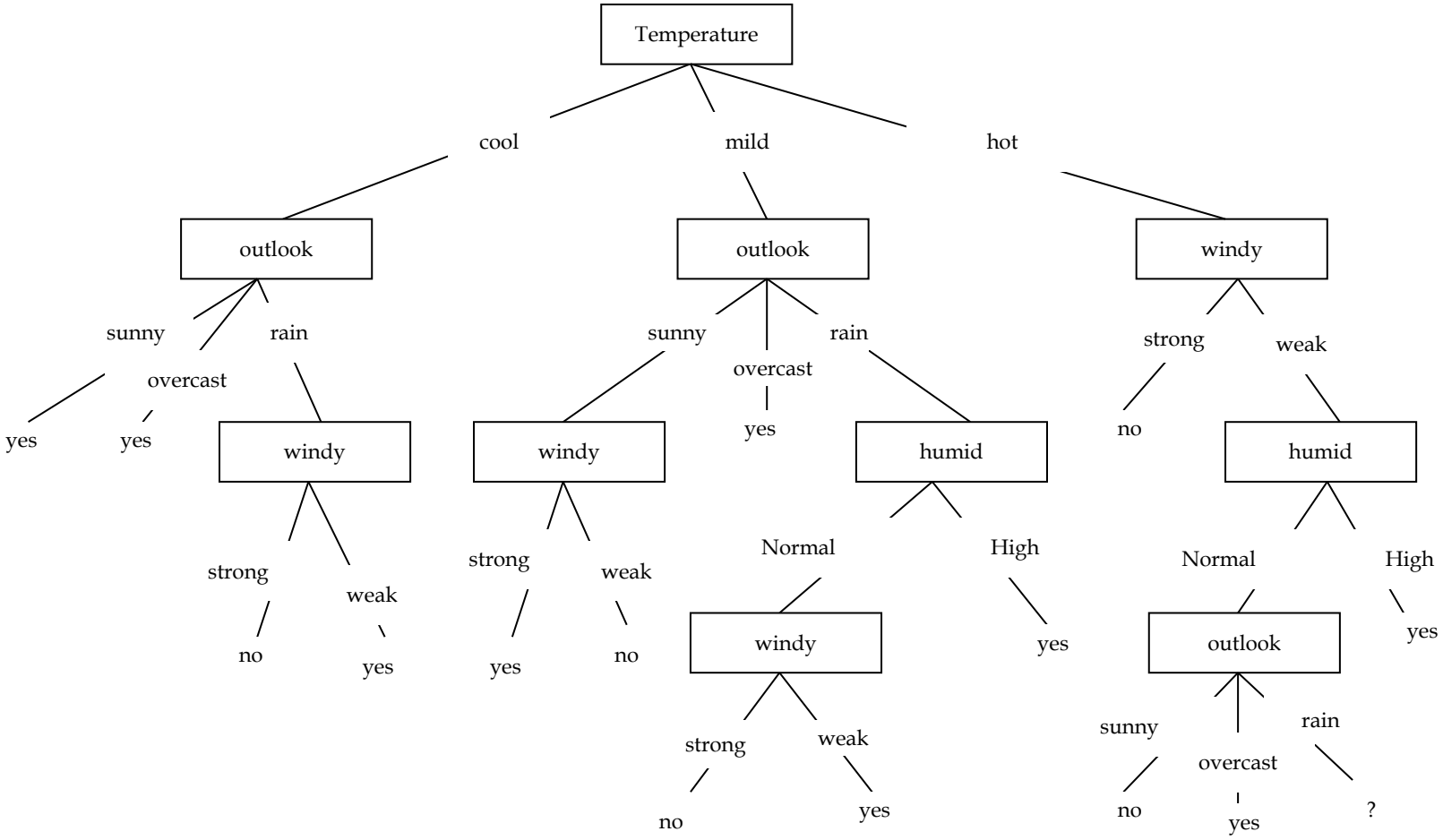
- Instances are represented by attribute-value pairs
- The target function has discrete output values
- Disjunctive descriptions may be required
- The training data may contain errors
  - Both errors in classification of the training examples and errors in the attribute values
- The training data may contain missing attribute values
- Suitable for classification

Learning Algorithm (1/5)

- Main question
  - Which attribute should be tested at the root of the (sub)tree?
    - Greedy search using some statistical measure

- Information gain
  - A quantitative measure of the worth of an attribute
  - How well a given attribute separates the training example according to their target classification
  - Information gain measures the expected reduction in entropy

Learning Algorithm (2/5)

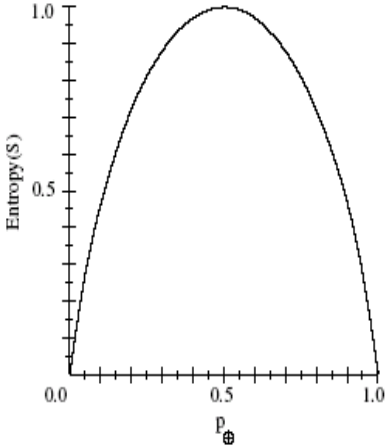


Learning Algorithm (3/5)

- Entropy
  - characterizes the (im)purity of an arbitrary of examples

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S



  - Entropy specifies the minimum # of bits of information needed to encode the classification of an arbitrary member of *S*
  - For example
    - The information required for classification of Table 3.2 $=-(9/14)\log_2(9/14)-(5/14)\log_2(5/14)=0.940$

Learning Algorithm (4/5)

- According to information theory
  - Optimal length code assigns -$\log_2 p$ bits to message having probability p
- General form of entropy

$$Entropy(s) \equiv \sum_{i=1}^{c} -P_i \log_2 P_i$$

c : Number of values.

P_i : The proportion of S belonging to class i

Learning Algorithm (5/5)

- Information gain and entropy

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- ✓ *Values (A)*: the set of all possible values for attribute *A*
- ✓ S_v : the subset of *S* for which attribute *A* has value *v*

  - First term: the entropy of the original collection
  - Second term: the expected value of the entropy after *S* is partitioned using attribute *A*

- *Gain (S ,A)*
  - The expected reduction in entropy caused by knowing the value of attribute *A*
  - The information provided about the target function value, given the value of some other attribute *A*

Learning Algorithm – ID3(1/2)

- ID3 (*Examples, Target_attribute,Attributes*)
  - Create a *Root* node for the tree
  - If all *Examples* are positive, Return the single node tree *Root*, with label= +
  - If all *Examples* are negative, Return the single node tree *Root*, with label= -
  - If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
  - Otherwise Begin

Learning Algorithm – ID3(2/2)

- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i ,of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
    - Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute, Attributes – {A}*)
- End
- Return *Root*

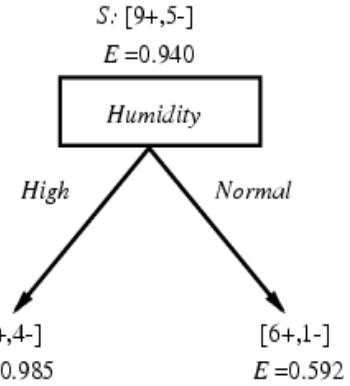
An Illustrative Example (1/2)

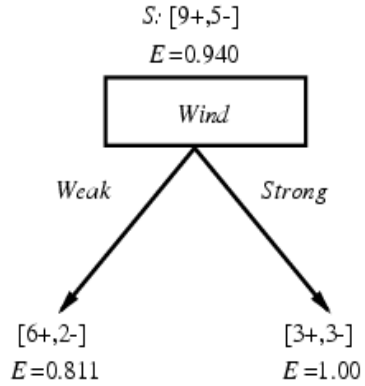
- Data set

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

An Illustrative Example (2/2)

- Selecting root
  - The information gain values for all four attributes
    - *Gain(S, Outlook)*= 0.246 ← selected as root attribute
    - *Gain(S, Humidity)*= 0.151
    - *Gain(S, Wind)*= 0.048
    - *Gain(S, Temperature)*= 0.029

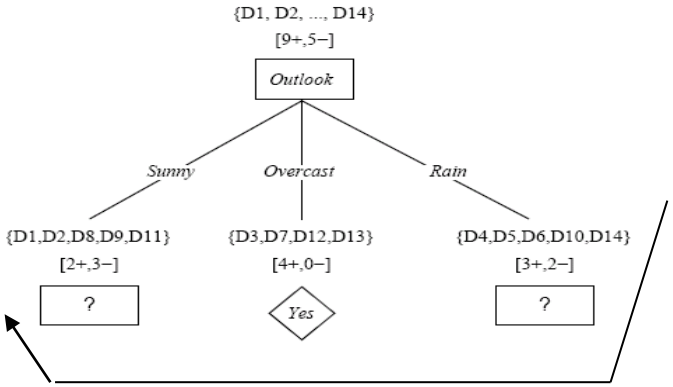




Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

- Adding a subtree



Which attribute should be tested here?

S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

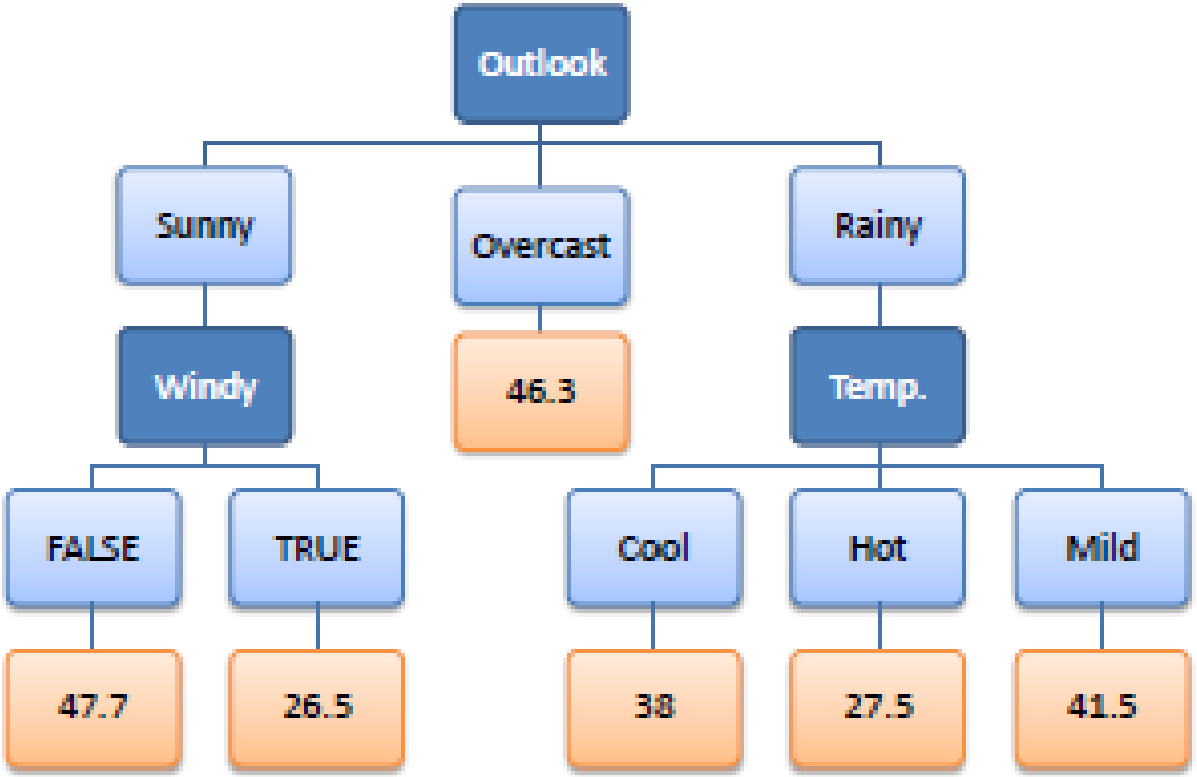
Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Decision Tree learning can be used for regression

Outlook	Temp	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

Decision Tree learning can be used for regression



Summary

- For classification problems, there are two types of ML models:
  - Discriminative models (e.g., logistic regression, nearest neighbor, etc.)
  - Generative models (e.g., GDA, Naïve Bayes)
- Generative models are known to make stronger assumptions but are also more efficient to compute
- Decision tree (DT) learning is known to be non-parametric learning algorithm
- DT learning can be used for both classification and regression leaning tasks