ieeta instituto de engenharia electrónica e telemática de aveiro

universidade de aveiro

Departamento de Eletrónica, Telecomunicações e Informática

Machine Learning

LECTURE 8

Naïve Bayes & Decision Tree classifiers

Petia Georgieva
(petia@ua.pt)

universidade de aveiro

Outline

1. Naïve Bayes (NB) classifier

3. Decision tree (DT) classifier

3. Performance indicators ROC & AUC

Bayesian Classifier

- Given labeled data with p classes ($C_1, C_2,\ldots,C_p$), each example has n features $x=[x_1, x_2,\ldots,x_n]$
- Each feature (x_i) is treated as a random variable.
- Compute the probability of a new example
- $x_{new}=(x_{new1}, x_{new2},\ldots,x_{newn})$ to belong to each one of the classes.
- The class with the highest probability is assigned to the new example.
- Use Bayes Theorem to compute the *aposteriory* probabilities

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$$

$$x_{new} \Rightarrow C_i : \quad P(C_i|x_{new}) > P(C_j|x_{new}), \quad \forall j \neq i$$

Bayes Theorem

- Aposteriori probability: $P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$

- Prior (class) probability: $P(C_i)$

- Likelihood (conditional probability): $P(x|C_i)$

- Total probability (p classes) : $P(x) = \sum_{i=1}^{p} P(x|C_i)P(C_i)$

The decision:

$$x \Rightarrow C_i : \quad P(x|C_i)P(C_i) > P(x|C_j)P(C_j), \quad \forall i \neq j$$

Prior (class) & Likelihood estimation

- How to estimate **prior (class) probability:**

$P(C_i)$= Nº of train examples that belong to C_i / Nº of all train examples

- **Non-parametric likelihood estimation** $P(x|C_i)$
  (estimated directly from train data)

  i) Categorical (string valued) features:

$P(x=$*'string value'*$|C_i)$ = # of train examples that belong to C_i & has this *'string value'* / # of train examples that belong to C_i

Nonparametric probability estimation -example

Tid	Refund	Marital Status	Taxable Income	Evade
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

$P(C_i)$= # of train examples that belong to C_i / # of all train examples

P(Class = No) = 7/10
P(Class = Yes) = 3/10

Feature Status has 3 string values : single, married, divorced

$P(x=$'*string value*'$|C_i)$ = # of train examples that belong to C_i & has this '*string value*' / # of train examples that belong to C_i

P(Status='Married'|Class No) = 4/7
P(Status='Married'|Class Yes) = 0/3

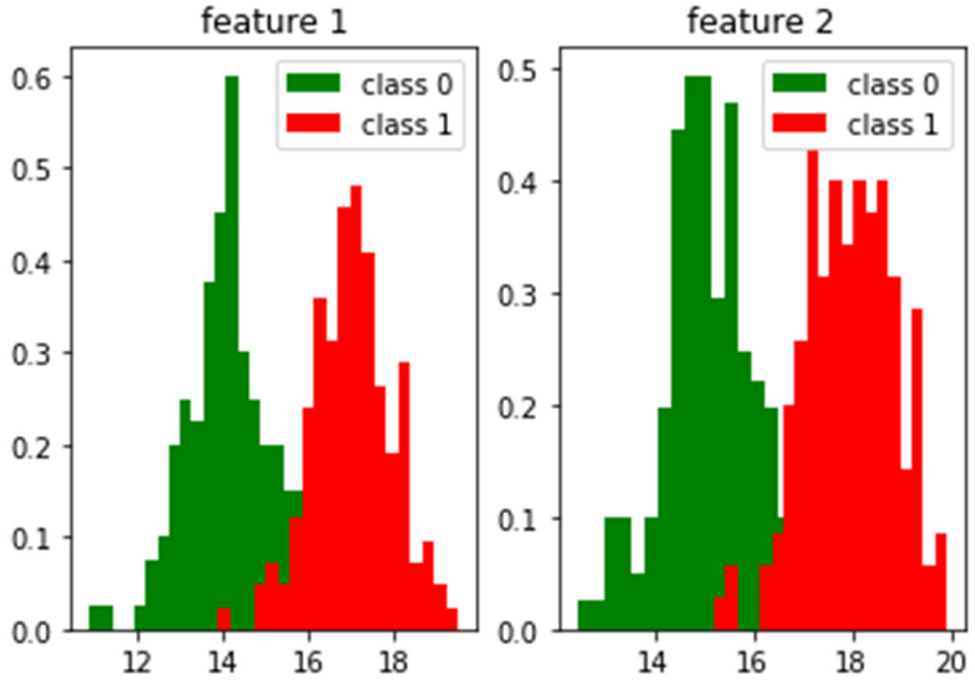
Nonparametric Likelihood estimation

- $P(x|C_i)$ (estimated directly from train data)

  ii) Continuous (real number) features:

  from the train examples that belong to C_i compute the histograms of each feature for a number of bins).

$P(x = real\ number|C_i)$ = # of train examples that belong to C_i & belong to a given bin / # of train examples that belong to C_i



Parametric Likelihood estimation

feature vector $x=[x_1, x_2,\ldots,x_n]$

- x_i - continuous feature that follows a certain distribution, for example Gaussian distribution for each class.

- From the training set compute the parameters of this distribution (sample mean and standard deviation of the feature for each class C_j)

- The likelihood that $x_i = value$ belongs to class C_j is

$$P(x_i = value \mid C_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(value - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

- The same applies to all continuous features with Gaussian distribution. Likelihood of **mutually independent features** is

$$P(x / C_j) = \prod_{i=1}^{n} P(x_i / C_j)$$

Parametric Likelihood estimation -example

Tid	Refund	Marital Status	Taxable Income	Evade
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

Determine the Gaussian distributions for each feature-class pair - (x_i, C_j)

$$P(x_i \mid C_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

For (Income, Class=No):

- sample mean = 110k
- sample variance = 2975

$$P(Income = 120k \mid No) = \frac{1}{\sqrt{2\pi 2975}} e^{-\frac{(120k - 110k)^2}{2(2975)}} = 0.0072$$

Naïve Bayes Classifier - example

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

4 Features: Give Birth, Can Fly, Live in Water, Have legs

2 Classes: M (mammals)

N (non-mammals)

$$P(x_{new} \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(x_{new} \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(x_{new} \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(x_{new} \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

x_{new}

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

$P(x_{new} \mid M)P(M) > P(x_{new} \mid N)P(N)$

=> Assign Class Mammal

Naive Bayes Classifier – Laplace correction

- If Likelihood of one of mutually independent features = 0, the entire Likelihood (product of the features likelihoods) =0.

  To prevent this, apply Laplace correction.

  Likelihood estimation:

$$\text{Original}: \quad P(x_i = \text{'string value'} \mid C_j) = \frac{N_{ij}}{N_{Cj}}$$

$$\text{Laplace correction}: P(x_i = \text{'string value'} \mid C_j) = \frac{N_{ij} + k}{N_{Cj} + k|x_i|}$$

N_{ij} = # of train examples with x_i ='string value' in class C_j

N_{Cj} = # of all train examples of class C_j

k- smoothing factor (e.g. k=1)

$|x_i|$ = # of possible string values of feature x_i

Naive Bayes (NB) Classifier -problems

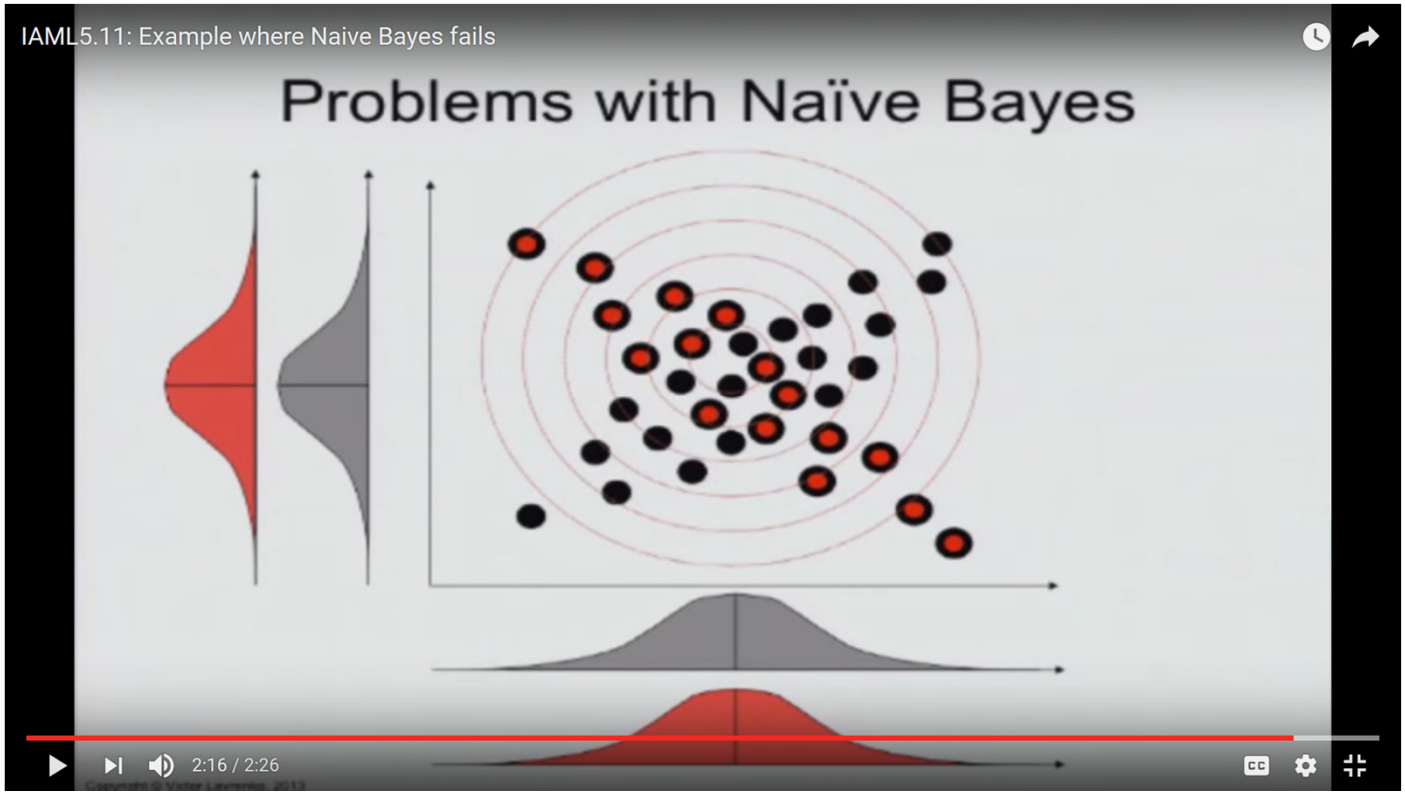
- Main assumption of NB classifier:

  the features are independent !!!

- The NB classifier will fail if the only thing that separates two classes is the co-variance of the attributes (not the mean or the variance).

- Solution: Multivariate Gaussian Distribution, compute covariance matrix, compute joint distribution.

NB will not be able to separate these classes



2 feature, 2 class problem:
The Gaussian distribution of the data for each class is the same, but for the black class we have a positive correlation between the features (if x1 increases, x2 also increases), and for the red class we have a negative correlation between the features (if x1 increases, x2 decreases).

Spam classification - Quiz

Given a set of messages classified as spam (3 messages) and not spam (5 messages).
We get a new message M =”today is secret”.
Compute the probability that M is a spam message applying the Bayes rule.
P(Spam/M) =?

Spam	Not spam (Ham)
1. Offer is secret	1. Play sport today
2. Click secret link	2. Went play sport
3. Secret sport link	3. Secret sport vents
	4. Sport is today
	5. Sport costs money

.

Create a “bag of words” and count the occurrences of words in spam and not spam messages.
P(today/Spam)=0/9 =>Apply Laplace smoothing !

word	frequency	Spam	Not Spam
offer	1	1	0
is	2	1	1
secret	4	3	1
click	1	1	0
sports	6	1	5
link	2	2	0
play	2	0	2
today	2	0	2
went	1	0	1
event	1	0	1
costs	1	0	1
money	1	0	1
12	24	9	15

Document Classification

Create a dictionary (bag of words for each type of documents)

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

Data set may be sparse.

Bayes rule - Quiz

We have two coins:
one fair P1 (heads)=0.5 and one loaded with P2 (heads)=1.

We now pick a coin at random with 0.5 chance. We flip this coin, and see "heads".

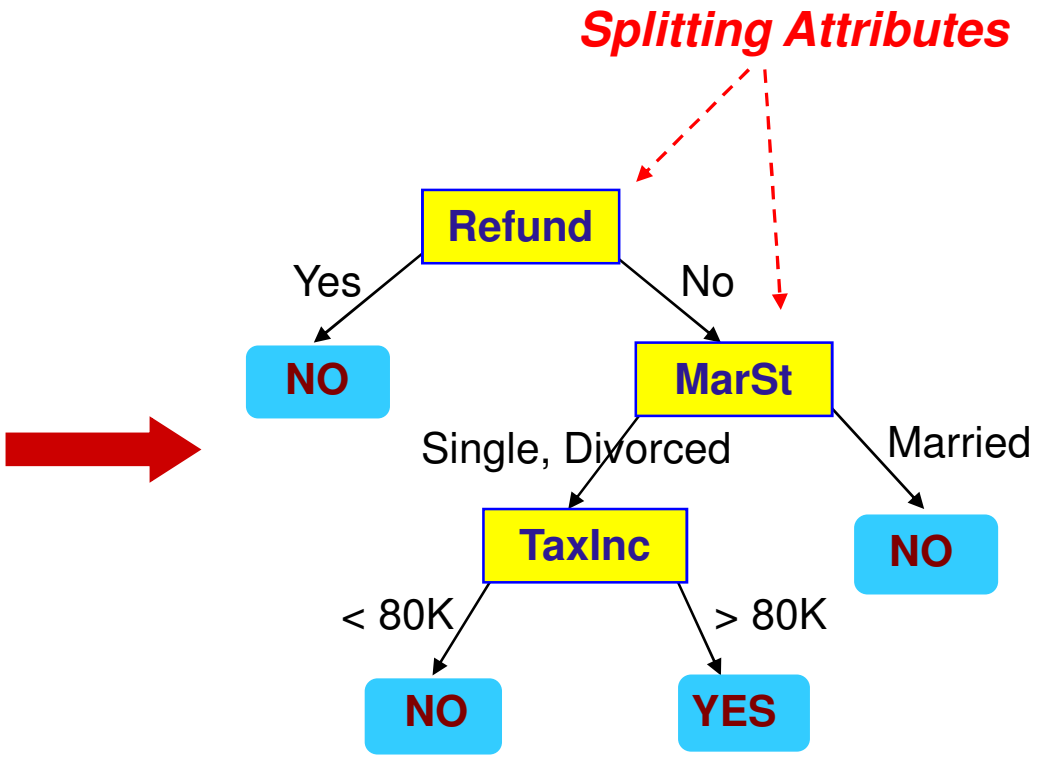
What is the probability this is the loaded coin? What is the probability this is the fair coin ?

Classes ?
Features ?
Prior (class) probabilities ?
Likelehood ?

Classification by Decision Tree – model 1

	categorical	categorical	continuous	class
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

Training Data

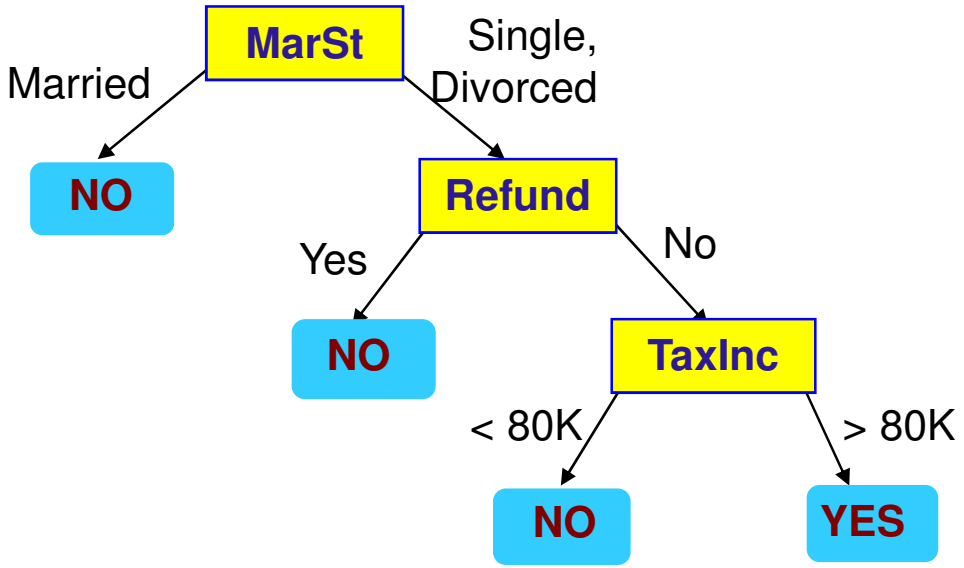


Model: Decision Tree

Classification by Decision Tree – model 2

	categorical	categorical	continuous	class
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

Training Data



There could be more than one tree that fits the same data!

Decision Tree

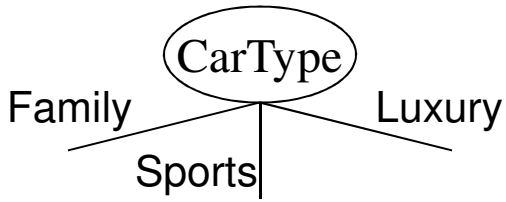
Decision Tree has 2 basic type of nodes:

- **Leaf node** - a class label, determined by majority vote of training examples reaching that leaf.

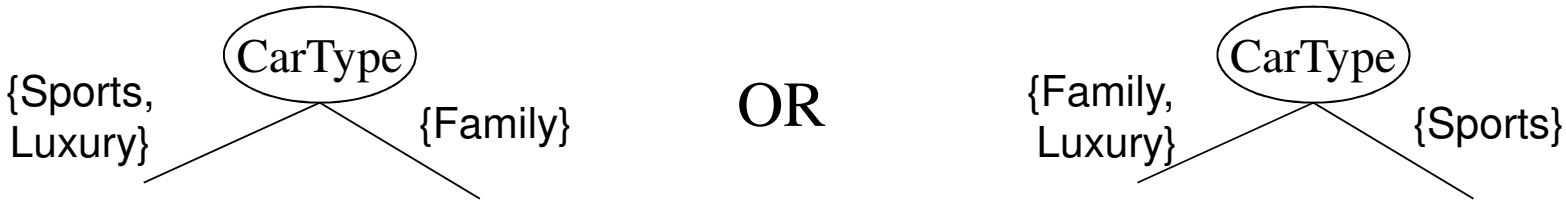
- **Node** is a question on one feature. It branches out according to the answers.
  - **root node**: the first (top) node of the tree
  - **child node**: the next node to a current node

Splitting of Categorical Features

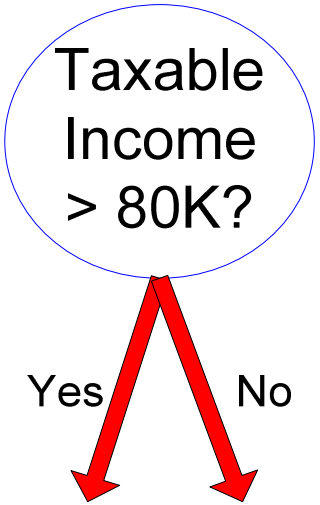
- Multi-way split: Use as many partitions as distinct values.



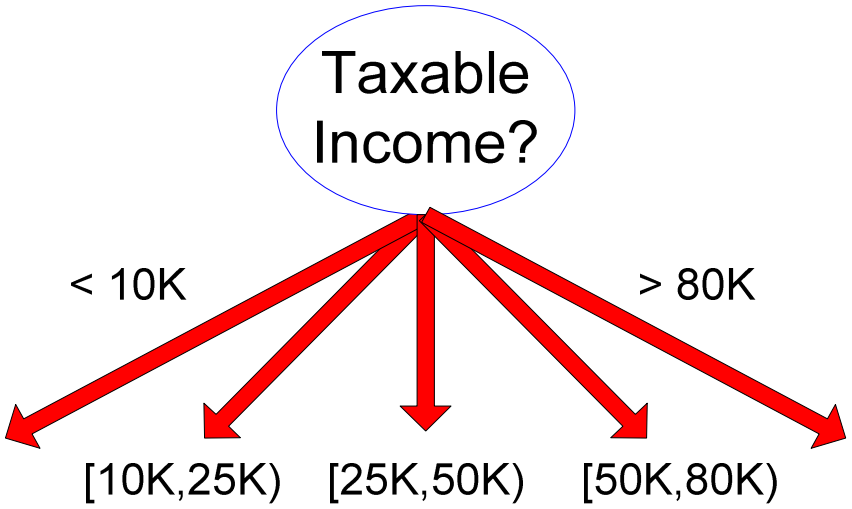
- Binary split: Divides values into two subsets.



Splitting of Continuous Features



(i) Binary split



(ii) Multi-way split

How to determine the best split ?

We want:

- To get the smallest tree
- Pure leaf nodes, i.e. all examples having (almost) the same class (ex. C0 or C1).
- Choose the feature that produces the “purest” (the most homogeneous) nodes
- We need a measure of node impurity (homogeneity).

C0: 5 C1: 5

Non-homogeneous,

High degree of impurity

C0: 9 C1: 1

Homogeneous,

Low degree of impurity

Measures of Node Impurity

- **Gini Index** at a given node:

$$GINI(node) = 1 - \sum_{Class_j} [p(Class_j \mid node)]^2$$

- **Classification error** at a given node:

$$Error(node) = 1 - \max_{Class_j} p(Class_j \mid node)$$

- **Entropy (H)** at a given node:

$$H(node) = -\sum_{Class_j} p(Class_j \mid node) \log p(Class_j \mid node)$$

p (Class_j / node): probability of *Class_j* at a given *node*

Computing GINI - example

$$GINI(node) = 1 - \sum_{Class_j} [p(Class_j \mid node)]^2$$

(*p* - probability)

C1	**0**
C2	**6**

p(C1) = 0/6 = 0 p(C2) = 6/6 = 1

Gini = 1 - p(C1)² - p(C2)² = 1 - 0 - 1 = 0

C1	**1**
C2	**5**

p(C1) = 1/6 p(C2) = 5/6

Gini = 1 – (1/6)² – (5/6)² = 0.278

C1	**2**
C2	**4**

p(C1) = 2/6 p(C2) = 4/6

Gini = 1 – (2/6)² – (4/6)² = 0.444

Computing Classification Error - example

$$Error(node) = 1 - \max_{Class_j} p(Class_j \mid node)$$

C1	**0**
C2	**6**

p(C1) = 0/6 = 0 p(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

p(C1) = 1/6 p(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C1	**2**
C2	**4**

p(C1) = 2/6 p(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Entropy

p(head)=0.5
p(tail)=0.5
H=1

p(head)=0.51
p(tail)=0.49
H=0.9997

- Entropy is a probabilistic measure of information uncertainty. In DTree we use it to measure the purity of a node.

- The node is pure if it has only examples that belongs to one class => entropy H = 0, no uncertainty (this is what we want !)

- If the node has equal number of examples of all classes => entropy reaches maximum H =1, the result is very uncertain.

Computing Entropy - example

$$H(node) = -\sum_{Class_j} p(Class_j \mid node) \log p(Class_j \mid node)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

H = – 0 log(0) – 1 log(1) = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

H = – (1/6) $\log_2$(1/6) – (5/6) $\log_2$(1/6) = 0.65

C1	**2**
C2	**4**

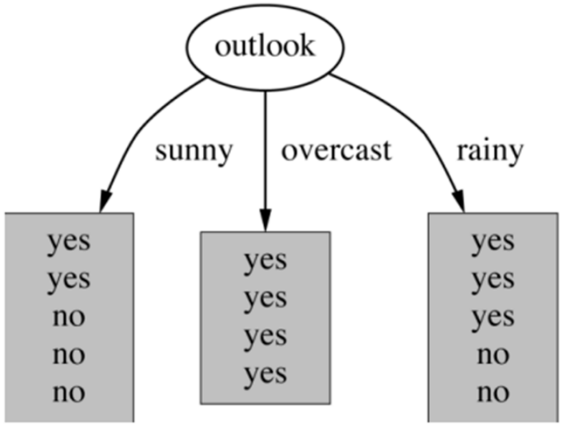
P(C1) = 2/6 P(C2) = 4/6

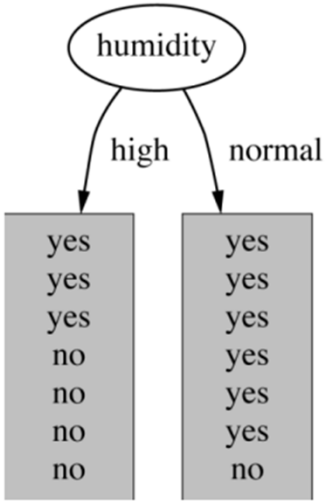
H = – (2/6) $\log_2$(2/6) – (4/6) $\log_2$(4/6) = 0.92

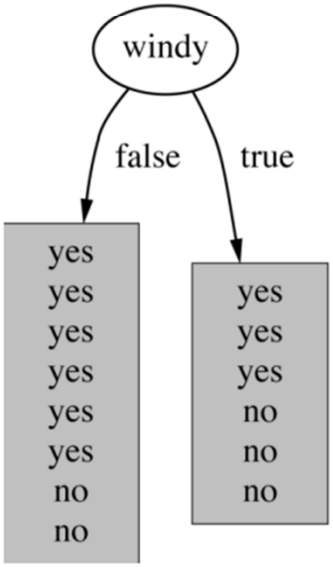
Example – Weather data (Play golf or Not)

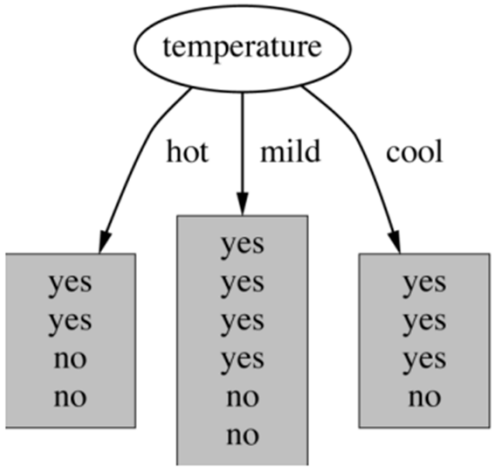
OUTLOOK	TEMP	HUMIDITY	WINDY	PLAY
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No%%

Which feature to select as root node?



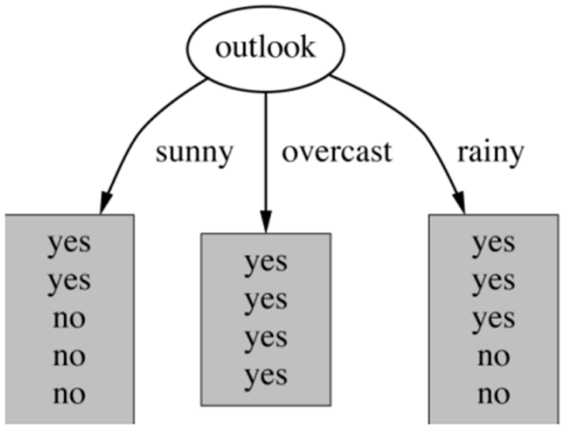






Information Gain

Information Gain = Entropy (H) before split - Entropy (H) after split at one node (one feature). How much uncertainty was reduced after splitting on a given feature.



$$H(node) = -\sum_{Class_j} p(Class_j \mid node) \log p(Class_j \mid node)$$

Feature OUTLOOK:

H(before split)=-[(9/14)*$\log_2$ (9/14)+(5/14)*$\log_2$ (5/14)] = 0.9403
(14 example: 9 ex. class Yes ; 5 ex. class No)

H(Sunny)=-[(2/5)*$\log_2$ (2/5)+(3/5)*$\log_2$ (3/5)]
(5 ex. with Outlook=Sunny: 2 ex. class Yes; 3 ex. class No)

H(Overcast)=-[4/4*$\log_2$ (4/4)]
(4 ex. with Outlook=Overcast - 4 ex. class Yes ; 0 ex. class No)

H(Rainy)=-((2/5)*$\log_2$ (2/5)+(3/5)*$\log_2$ (3/5))
(5 ex. with Outlook=Rainy - 3 ex. class Yes; 2 ex. class No)

H(after split) =(5/14)*H(Sunny)+(4/14)*H(Overcast)+(5/14)*H(Rainy) =0.6935

Information Gain=H(before split) –H (after split) = 0.9403 - 0.6935=0.2467

Building a Decision Tree - example

Feature OUTLOOK: Information Gain = 0.2467

Feature TEMPERATURE : Information Gain = 0.029

Feature HUMIDITY: Information Gain = 0.157

Feature WINDY: Information Gain = 0.048

Choose feature that maximizes the information gain (minimizes the node impurity) !

Decision: Choose OUTLOOK as the root node.

This procedure is repeated for each node.

Splitting stops when data cannot be split any further or

the class is defined by the majority of elements of a subset.

Gain Ratio

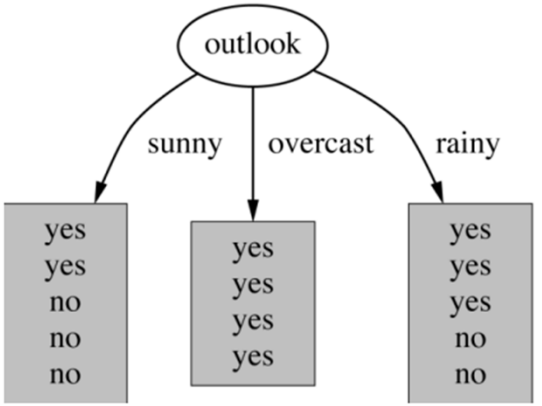
- **Gain ratio** is a modification of the splitting criteria to deal better with overfitting. Gain ratio takes number and size of branches into account when choosing the most favorable feature to split at a node.

  Gain ratio =Information_Gain / Split Entropy

For OUTLOOK:

Split Entropy=-[(5/14)*$\log_2$ (5/14)+(4/14)*$\log_2$ (4/14)+(5/14)*$\log_2$ (5/14)]

(3 branches, with 5, 4, 5 examples per branch)



Gain ratio= 0.247 / 1.577

Choose feature that maximizes the gain ratio.

Other DT Algorithms

- **ID3 (Iterative Dichotomiser 3)** algorithm. Use Information Gain to select the best attribute.
- **Modified ID3** – use Gain ratio.
- **C4.5 (**extension of ID3)- deals with numeric attributes, missing values, noisy data
- **CART (Classification And Regression Tree)** – similar approach
- Random Forest – ensemble of DT classifiers

Remark: There are many other feature selection criteria!

DT Pruning

Two strategies to prevent DT overfitting:

- **Post-pruning** - First, build full decision tree and then discard unreliable parts
- **Pre-pruning** – stop growing a branch when information gain does not change significantly
- Post-pruning preferred in practice—pre-pruning can "stop too early"

Performance Evaluation – Confusion Matrix

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Python: from sklearn.metrics import confusion_matrix

Performance metrics from Conf Matrix

True Positive Rate (TPR), Sensitivity, Recall
of all positive examples the fraction of correctly classified

$$TPR = \frac{TP}{TP + FN}$$

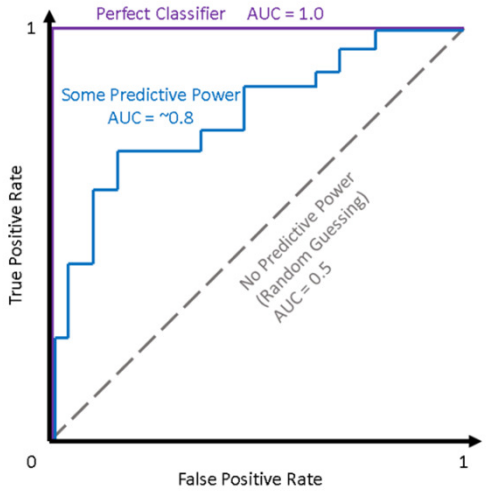
True Negative Rate (TNR), Specificity
of all negative examples the fraction of correctly classified

$$TNR = \frac{TN}{TN + FP}$$

False Positive Rate (FPR) - how often an actual negative instance will be classified as positive, i.e. “false alarm”

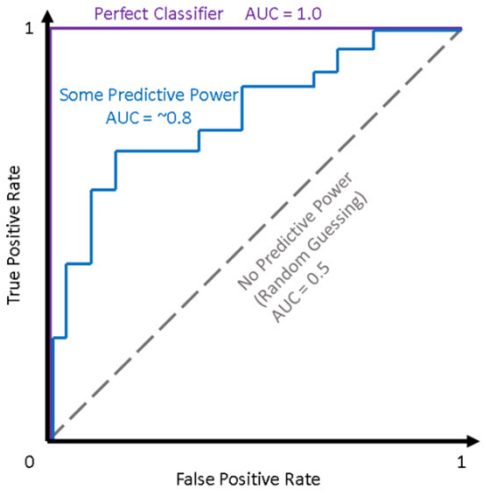
$$FPR = 1 - TNR = \frac{FP}{FP + TN}$$

Receiver Operating Characteristic (ROC) curve



ROC curve: **True Positive Rate (TPR)** against **False Positive Rate (FPR)** for a single classifier at a variety of **thresholds**.
For example, in logistic regression, if an observation is predicted to be > 0.5, it is labelled as positive.
But, we can choose any threshold between 0 and 1 (0.1, 0.3, 0.6, 0.99, etc.).
ROC curves visualize how these choices affect classifier performance.
For multi K-class problem, draw K ROC curves.
Python: from sklearn.metrics import roc_curve

Area Under the (ROC) Curve - AUC



ROC curve is useful for visualization, but it's good to have also a single metric => AUC.
The higher the AUC score, the better a classifier performs for the given task.
For a classifier with no predictive power (i.e., random guessing) => AUC = 0.5.
For a perfect classifier => AUC = 1.0.
Most classifiers fall between 0.5 and 1.0.

Python: from sklearn.metrics import auc