

Introduction to Data Mining Algorithms

Parkavi.A

Assistant Professor

Ramaiah Institute of Technology

Bangalore

Why do we need data mining?????

Why do we need data mining?

- ▶ **Data mining** is the process of analyzing large amounts of **data** in an effort to find correlations, patterns, and insights.
- ▶ **Data mining** is **Data** Discovery in other word.
- ▶ To discover relationship between two or more variables in **data**, we require **Data Mining**.
- ▶ It can improve customer service, better target marketing campaigns, identify high-risk clients, and improve production processes.
- ▶ In short, because it can help companies to make or save money.

Where do we need, data mining????

Data Mining has been used to

- ▶ Identify unexpected shopping patterns in supermarkets.
- ▶ Optimize website profitability by making appropriate offers to each visitor.
- ▶ Predict customer response rates in marketing campaigns.
- ▶ Defining new customer groups for marketing purposes.
- ▶ Predict customer defections: which customers are likely to switch to an alternative supplier in the near future.
- ▶ Distinguish between profitable and unprofitable customers.
- ▶ Improve yields in complex production processes by finding unexpected relationships between process parameters and defect rates.
- ▶ Identify suspicious (unusual) behavior, as part of a fraud detection process.

How do we do, data mining???

Using (Top 10) Data Mining Algorithms

- ▶ 1. Apriori
- ▶ 2. Naive Bayes
- ▶ 3. PageRank
- ▶ 4. kNN
- ▶ 5. C4.5
- ▶ 6. Support vector machines
- ▶ 7. AdaBoost
- ▶ 8. EM
- ▶ 9. k-means
- ▶ 10. CART

Why do we need association rule mining (Apriori Algorithm)?

- ▶ **Shopping** centers use association rules to place the items next to each other so that users buy more items.
- ▶ **Amazon**, they use association mining to recommend you the items based on the current item you are browsing/buying
- ▶ **Google auto-complete**, where after you type in a word it searches frequently associated words that user type after that particular word.

What is Apriori Algorithm?

- ▶ The **Apriori Algorithm** is an influential **algorithm** for mining frequent itemsets for boolean association rules.
- ▶ **Apriori** uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data
- ▶ Apriori algorithm is a classical algorithm in data mining. It is used for mining frequent itemsets and relevant association rules. It is devised to operate on a database containing a lot of transactions, for instance, items brought by customers in a store.

Association rules

- Association rule learning is a prominent and a well-explored method for determining relations among variables in large databases.

Let $I = \{i_1, i_2, i_3, \dots, i_n\}$ be a set of n attributes called items and $D = \{t_1, t_2, \dots, t_n\}$ be the set of transactions. It is called database. Every transaction, t_i in D has a unique transaction ID, and it consists of a subset of itemsets in I .

A rule can be defined as an implication, $X \longrightarrow Y$ where X and Y are subsets of I ($X, Y \subseteq I$), and they have no element in common, i.e., $X \cap Y = \emptyset$. X and Y are the antecedent and the consequent of the rule, respectively.

Transaction Example

Transaction ID	Onion	Potato	Burger	Milk	Beer
T1	1	1	1	0	0
T2	0	1	1	1	0
T3	0	0	0	1	1
T4	1	1	0	1	0
T5	1	1	1	0	1
T6	1	1	1	1	1

Support

Support

The support of an itemset X , $supp(X)$ is the proportion of transaction in the database in which the item X appears. It signifies the popularity of an itemset.

$$supp(X) = \frac{\text{Number of transaction in which } X \text{ appears}}{\text{Total number of transactions}}.$$

In the example above, $supp(Onion) = \frac{3}{6} = 0.5$.

If the sales of a particular product (item) above a certain proportion have a meaningful effect on profits, that proportion can be considered as the support threshold. Furthermore, we can identify itemsets that have support values beyond this threshold as significant itemsets.

Confidence

Confidence of a rule is defined as follows:

$$\text{conf}(X \rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}$$

It signifies the likelihood of item Y being purchased when item X is purchased. So, for the rule {Onion, Potato} => {Burger},

$$\text{conf}(\{Onion, Potato\} \Rightarrow \{Burger\}) = \frac{\text{supp}(\{Onion, Potato, Burger\})}{\text{supp}(\{Onion, Potato\})} = \frac{3}{6} * \frac{6}{4} = 0.75$$

$$\text{supp}(o) = 3/6$$

$$\text{Supp}(p) = 4/6$$

$$\text{Supp}(b) = 4/6$$

$$\text{supp}(o \cup p) = 4/6$$

$$\text{supp}(o \cup p \cup b) = 3/6$$

$$\text{supp}(o \cup p) = 4/6$$

Lift

The lift of a rule is defined as:

$$lift(X \longrightarrow Y) = \frac{supp(X \cup Y)}{supp(X) * supp(Y)}$$

This signifies the likelihood of the itemset Y being purchased when item X is purchased while taking into account the popularity of Y .

In our example above,

$$lift(\{Onion, Potato\} \implies \{Burger\}) = \frac{supp(\{Onion, Potato, Burger\})}{supp(\{Onion, Potato\}) * supp(Burger)} =$$
$$\frac{3}{6} * \frac{6 * 6}{4 * 4} = 1.125$$

Mining Associations with Apriori

- ▶ Mine frequent itemsets, association rules or association hyperedges using the Apriori algorithm
- ▶ **Usage :**
 - ▶ `apriori(data, parameter = NULL, appearance = NULL, control = NULL)`
- ▶ **Arguments**
 - ▶ **Data:** object of class `transactions` or any data structure which can be coerced into `transactions` (e.g., a binary matrix or `data.frame`).
 - ▶ **Parameter :** object of class `APparameter` or named list. The default behavior is to mine rules with minimum support of 0.1, minimum confidence of 0.8, maximum of 10 items (`maxlen`), and a maximal time for subset checking of 5 seconds (`maxtime`).
 - ▶ **Appearance :** object of class `APappearance` or named list. With this argument item appearance can be restricted (implements rule templates). By default all items can appear unrestricted.
 - ▶ **Control :** object of class `APcontrol` or named list. Controls the algorithmic performance of the mining algorithm (item sorting, report progress (verbose), etc.)

Details Contd..

- ▶ Calls the C implementation of the Apriori algorithm by Christian Borgelt for mining frequent itemsets, rules or hyperedges.
- ▶ Apriori only creates rules with one item in the RHS (Consequent)! The default value in APparameter for minlen is 1. This means that rules with only one item (i.e., an empty antecedent/LHS) like $\{\} \Rightarrow \{\text{beer}\}$ will be created. These rules mean that no matter what other items are involved the item in the RHS will appear with the probability given by the rule's confidence (which equals the support). If you want to avoid these rules then use the argument `parameter=list(minlen=2)`.
- ▶ If the minimum support is chosen too low for the dataset, then the algorithm will try to create an extremely large set of itemsets/rules. This will result in very long run time and eventually the process will run out of memory. To prevent this, the default maximal length of itemsets/rules is restricted to 10 items (via the parameter `element maxlen=10`) and the time for checking subsets is limited to 5 seconds (via `maxtime=5`).
- ▶ Value : Returns an object of class rules or itemsets.

Discretize()

- ▶ Convert a Continuous Variable into a Categorical Variable
 - ▶ This function implements several basic unsupervised methods to convert continuous variables into a categorical variables (factor) suitable for association rule mining.
- ▶ Usage
 - ▶ `discretize(x, method="interval", categories = 3, labels = NULL, ordered=FALSE, onlycuts=FALSE, ...)`

Arguments Contd...

- ▶ **X :** a numeric vector (continuous variable).
- ▶ **Method :** discretization method. Available are: "interval" (equal interval width), "frequency" (equal frequency), "cluster" (k-means clustering) and "fixed" (categories specifies interval boundaries).
- ▶ **Categories :** number of categories or a vector with boundaries (all values outside the boundaries will be set to NA).
- ▶ **Labels :** character vector; names for categories.
- ▶ **Ordered :** logical; return a factor with ordered levels?
- ▶ **Onlycuts :** logical; return only computed interval boundaries?

Contd...

- ▶ Value: A factor representing the categorized continuous variable or, if `onlycuts=TRUE`, a vector with the interval boundaries.

How will we do association rule mining through apriori algorithm?

- Suppose you have records of large number of transactions at a shopping center as follows:

Transactions	Items bought
T1	Item1, item2, item3
T2	Item1, item2
T3	Item2, item5
T4	Item1, item2, item5

Let's start with an example

Transaction ID	Items Bought
T1	{Mango, Onion, Nintendo, Key-chain, Eggs, Yo-yo}
T2	{Doll, Onion, Nintendo, Key-chain, Eggs, Yo-yo}
T3	{Mango, Apple, Key-chain, Eggs}
T4	{Mango, Umbrella, Corn, Key-chain, Yo-yo}
T5	{Corn, Onion, Onion, Key-chain, Ice-cream, Eggs}

Golden rule!

- ▶ Now, we follow a simple golden rule: we say an item/itemset is frequently bought if it is bought at least 60% of times. So for here it should be bought at least 3 times.
- ▶ For simplicity
M = Mango
O = Onion
And so on....

So the table becomes

Transaction ID	Items Bought
T1	{M, O, N, K, E, Y}
T2	{D, O, N, K, E, Y}
T3	{M, A, K, E}
T4	{M, U, C, K, Y}
T5	{C, O, O, K, I, E}

Step 1

- Count the number of transactions in which each item occurs, Note 'O=Onion' is bought 4 times in total, but, it occurs in just 3 transactions.

Item	No of transactions
M	3
O	3
N	2
K	5
E	4
Y	3
D	1
A	1
U	1
C	2
I	1

Step 2

- Now remember we said the item is said frequently bought if it is bought at least 3 times. So in this step we remove all the items that are bought less than 3 times from the above table and we are left with

Item	Number of transactions
M	3
O	3
K	5
E	4
Y	3

Step 2 continued...

- This is the single items that are bought frequently. Now let's say we want to find a pair of items that are bought frequently. We continue from the above table (Table in step 2)

Step 3

- We start making pairs from the first item, like MO,MK,ME,MY and then we start with the second item like OK,OE,OY. We did not do OM because we already did MO when we were making pairs with M and buying a Mango and Onion together is same as buying Onion and Mango together. After making all the pairs we get,

Item pairs
MO
MK
ME
MY
OK
OE
OY
KE
KY
EY

Step 4

- ▶ Now we count how many times each pair is bought together. For example M and O is just bought together in {M,O,N,K,E,Y}
- ▶ While M and K is bought together 3 times in {M,O,N,K,E,Y}, {M,A,K,E} AND {M,U,C, K, Y}
- ▶ After doing that for all the pairs we get

Item Pairs	Number of transactions
MO	1
MK	3
ME	2
MY	2
OK	3
OE	3
OY	2
KE	4
KY	3
EY	2

Step 5

- Golden rule to the rescue. Remove all the item pairs with number of transactions less than three and we are left with

Item Pairs	Number of transactions
MK	3
OK	3
OE	3
KE	4
KY	3

Step 5 Continued...

- ▶ These are the pairs of items frequently bought together.
- ▶ Now let's say we want to find a set of three items that are brought together.
- ▶ We use the above table (table in step 5) and make a set of 3 items.

Step 6

- ▶ To make the set of three items we need one more rule (it's termed as self-join),
- ▶ It simply means, from the Item pairs in the above table, we find two pairs with the same first Alphabet, so we get

OK and OE, this gives OKE

KE and KY, this gives KEY

- ▶ Then we find how many times O,K,E are bought together in the original table and same for K,E,Y and we get the following table

Item Set	Number of transactions
OKE	3
KEY	2

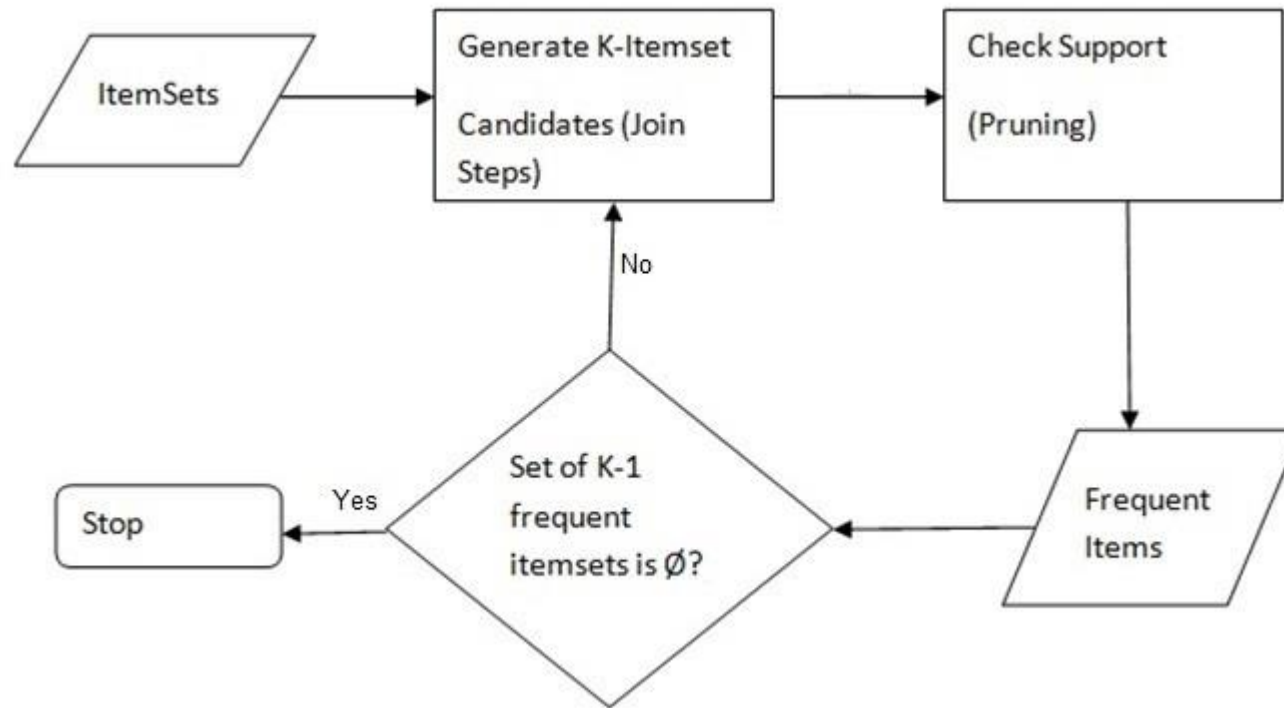
Step 6 Continued

- ▶ While we are on this, suppose you have sets of 3 items say ABC, ABD, ACD, ACE, BCD and you want to generate item sets of 4 items you look for two sets having the same first two alphabets.
- ▶ ABC and ABD \rightarrow ABCD
- ▶ ACD and ACE \rightarrow ACDE
- ▶ And so on ... In general we have to look for sets having just the last alphabet/item different.

Step 7

- ▶ So we again apply the golden rule, that is, the item set must be bought together at least 3 times which leaves us with just OKE, Since KEY are bought together just two times.
- ▶ Thus the set of three items that are bought together most frequently are O,K,E.

General Process of the Apriori algorithm



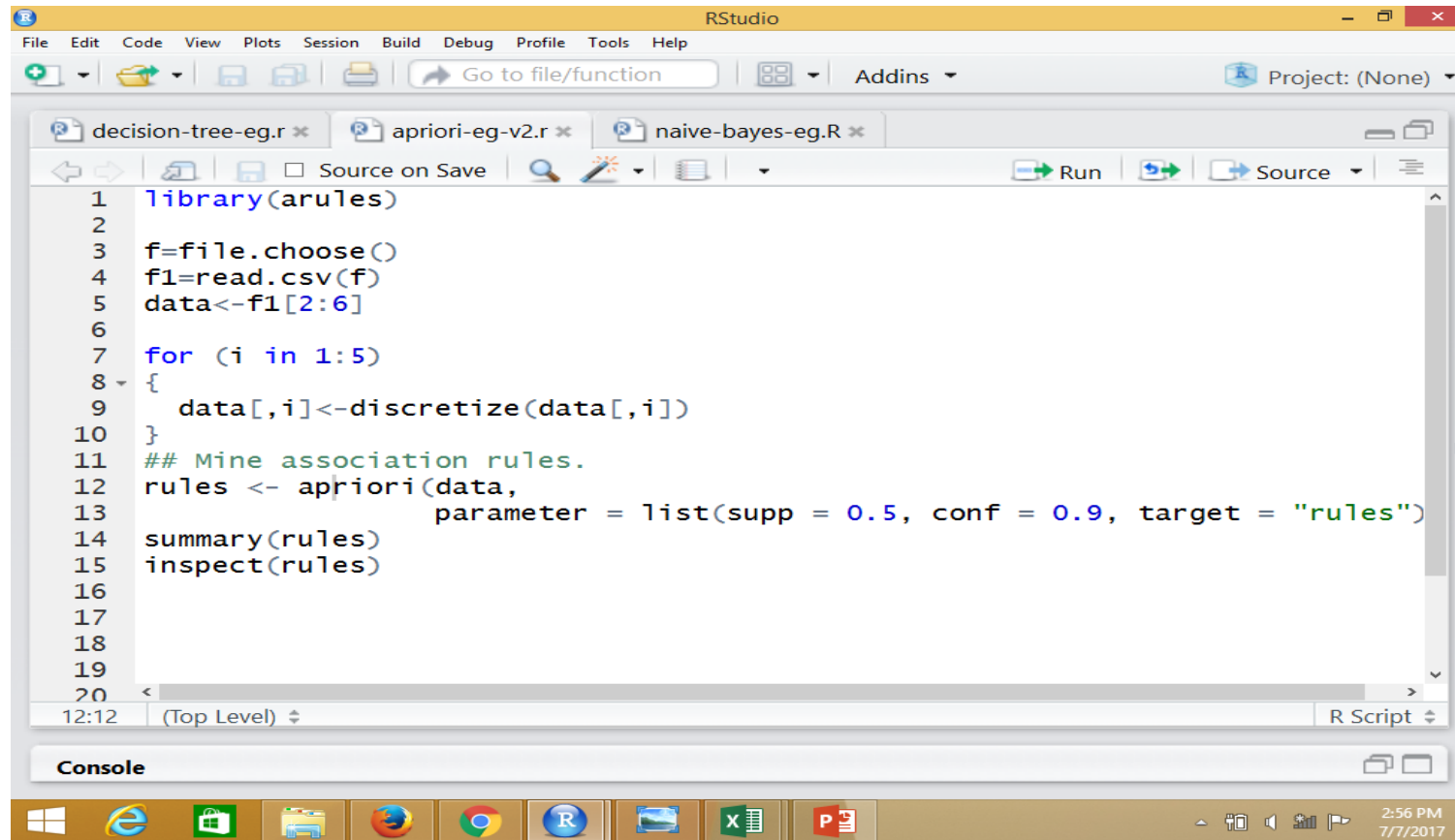
Apriori Algorithm

- ▶ The entire algorithm can be divided into two steps:
 - ▶ **Step 1:** Apply minimum support to find all the frequent sets with k items in a database.
 - ▶ **Step 2:** Use the self-join rule to find the frequent sets with $k+1$ items with the help of frequent k -itemsets. Repeat this process from $k=1$ to the point when we are unable to apply the self-join rule.

Write your own example now!!!!

How can we do Association rule mining in R using Apriori function?

► Go to Code



The screenshot shows the RStudio interface with three open files: 'decision-tree-eg.r', 'apriori-eg-v2.r', and 'naive-bayes-eg.R'. The 'apriori-eg-v2.r' file is active, displaying the following R code:

```
1 library(arules)
2
3 f=file.choose()
4 f1=read.csv(f)
5 data<-f1[2:6]
6
7 for (i in 1:5)
8 {
9   data[,i]<-discretize(data[,i])
10 }
11 ## Mine association rules.
12 rules <- apriori(data,
13                 parameter = list(supp = 0.5, conf = 0.9, target = "rules")
14 summary(rules)
15 inspect(rules)
16
17
18
19
20
```

The console at the bottom is empty. The system tray at the bottom right shows the time as 2:56 PM on 7/7/2017.

Stud data....

Now try the same code given in previous slide

transactions-stud - Excel

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW Sign in

Clipboard Font Alignment Number Styles Cells Editing

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Rollno	S1	S2	S3	S4	S5									
2	1	25	23	12	4	23									
3	2	12	23	23	22	12									
4	3	12	13	14	15	16									
5	4	13	14	15	16	17									
6	5	14	23	24	23	22									
7	6	23	22	24	25	23									
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															
24															
25															
26															

transactions-stud

READY 2:57 PM 7/7/2017

References

- ▶ http://www.albionresearch.com/data_mining/why.php
- ▶ <http://www.kdnuggets.com/2015/05/top-10-data-mining-algorithms-explained.html>
- ▶ <http://nikhylvithlani.blogspot.in/2012/03/apriori-algorithm-for-data-mining-made.html>
- ▶ <http://blog.hackerearth.com/beginners-tutorial-apriori-algorithm-data-mining-r-implementation>

Why Naïve Bayesian Algorithm?

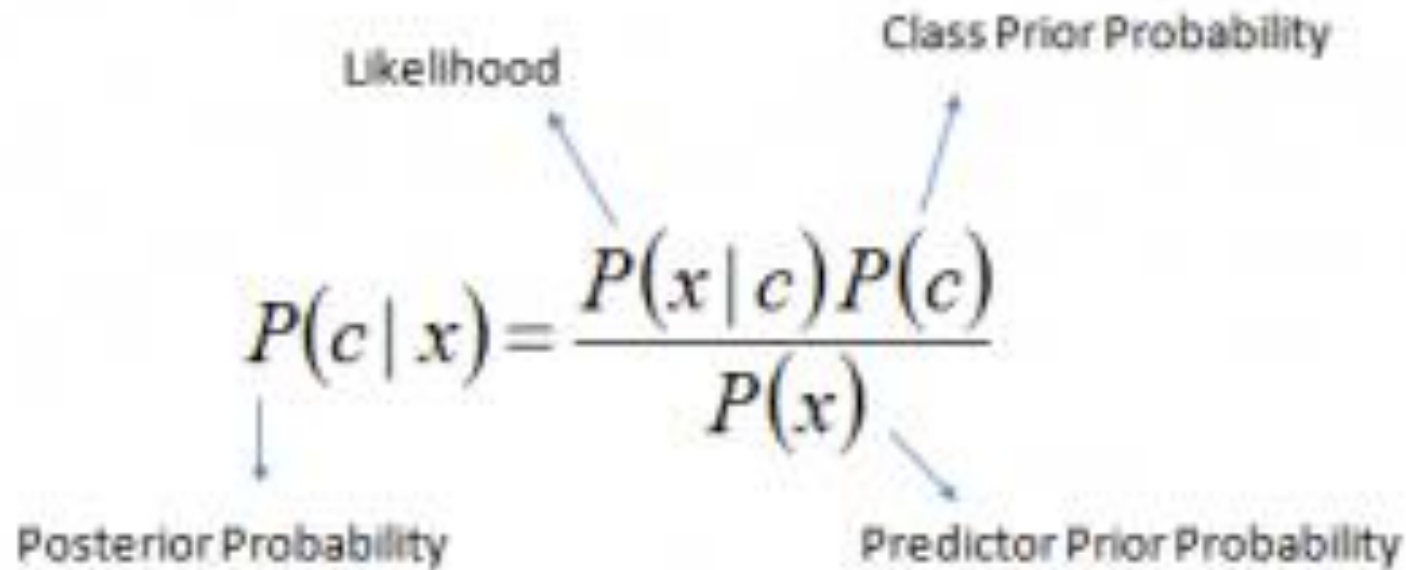
- ▶ If we are working on a **classification problem** and we have generated the set of hypothesis, created **features** and discussed the importance of variables. **Within an hour**, stakeholders want to see the first cut of the model.
- ▶ What will we do? we have hundreds of thousands of data points and quite a few variables in our training data set. In such situation, we can use '**Naïve Bayes**', which can be extremely **fast** relative to other classification algorithms. It works on Bayes theorem of probability to predict the class of unknown data set.

Naive Bayes classifier

- ▶ In simple terms, a Naive Bayes classifier assumes that the presence of a particular **feature in a class** is unrelated to the presence of any other feature. For example, a fruit may be considered to be an **apple** if it is red, round, and about 3 inches in diameter.

Bayes theorem

- Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation below:



The diagram shows the Bayes' theorem equation with four labels and arrows pointing to the corresponding parts of the formula:

- Likelihood** points to $P(x|c)$
- Class Prior Probability** points to $P(c)$
- Posterior Probability** points to $P(c|x)$
- Predictor Prior Probability** points to $P(x)$

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

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

R Code syntax: NaiveBayes

► Naive Bayes Classifier

- Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

► Usage

- `"NaiveBayes"(formula, data, ..., subset, na.action = na.pass)`
- `"NaiveBayes"(x, grouping, prior, usekernel = FALSE, fL = 0, ...)`

Arguments

- ▶ **X**
 - ▶ a numeric matrix, or a data frame of categorical and/or numeric variables.
- ▶ **Grouping**
 - ▶ class vector (a factor).
- ▶ **Formula**
 - ▶ a formula of the form `class ~ x1 + x2 +`. Interactions are not allowed.
- ▶ **Data**
 - ▶ a data frame of predictors (categorical and/or numeric).
- ▶ **Prior**
 - ▶ the prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
- ▶ **Usekernel**
 - ▶ if TRUE a kernel density estimate (density) is used for density estimation. If FALSE a normal density is estimated.

Contd..

- ▶ `fL`
 - ▶ Factor for Laplace correction, default factor is 0, i.e. no correction.
- ▶ ...
- ▶ arguments passed to `density`.
- ▶ `Subset`
 - ▶ for data given in a data frame, an index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
- ▶ `na.action`
 - ▶ a function to specify the action to be taken if NAs are found. The default action is not to count them for the computation of the probability factors. An alternative is `na.omit`, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.)

How Naive Bayes algorithm works?

Let us see with an Example

- ▶ A training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to **classify whether players will play or not based on weather condition.**
- ▶ Step 1: Convert the data set into a frequency table
- ▶ Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.
- ▶ Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Continued...

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	$\approx 4/14$	0.29
Rainy	3	2	$\approx 5/14$	0.36
Sunny	2	3	$\approx 5/14$	0.36
All	5	9		
	$\approx 5/14$	$\approx 9/14$		
	0.36	0.64		

Problem: Players will play if weather is sunny. Is this statement is correct?

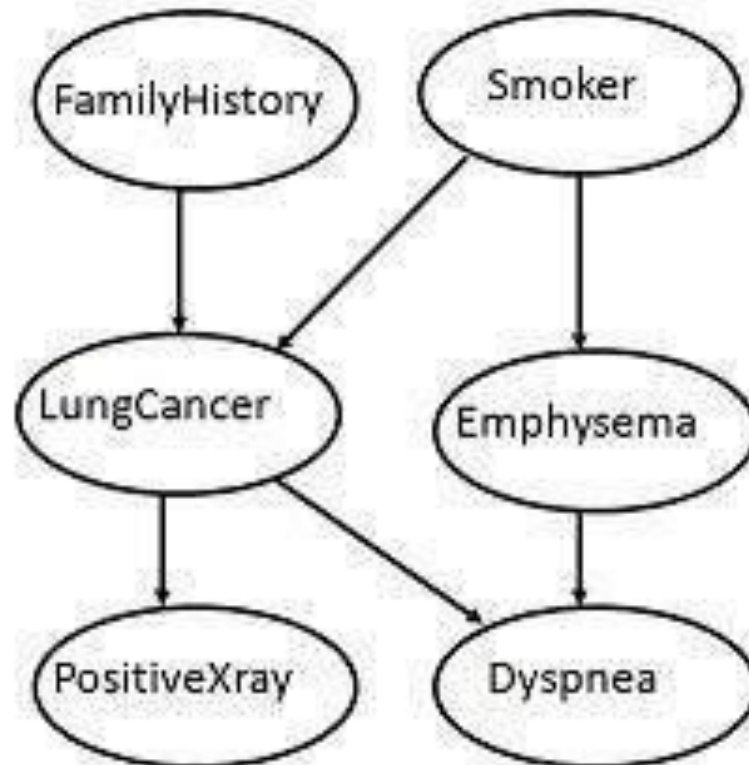
- ▶ We can solve it using above discussed method of posterior probability.
- ▶ $P(\text{Yes} \mid \text{Sunny}) = P(\text{Sunny} \mid \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$
- ▶ Here we have $P(\text{Sunny} \mid \text{Yes}) = 3/9 = 0.33$, $P(\text{Sunny}) = 5/14 = 0.36$, $P(\text{Yes}) = 9/14 = 0.64$
- ▶ Now, $P(\text{Yes} \mid \text{Sunny}) = 0.33 * 0.64 / 0.36 = 0.60$, which has higher probability.
- ▶ Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

Where can we use Naïve Bayes method?

- ▶ Real time Prediction
- ▶ Multi class Prediction
- ▶ Text classification/ Spam Filtering/ Sentiment Analysis
- ▶ Recommendation System

Another Example

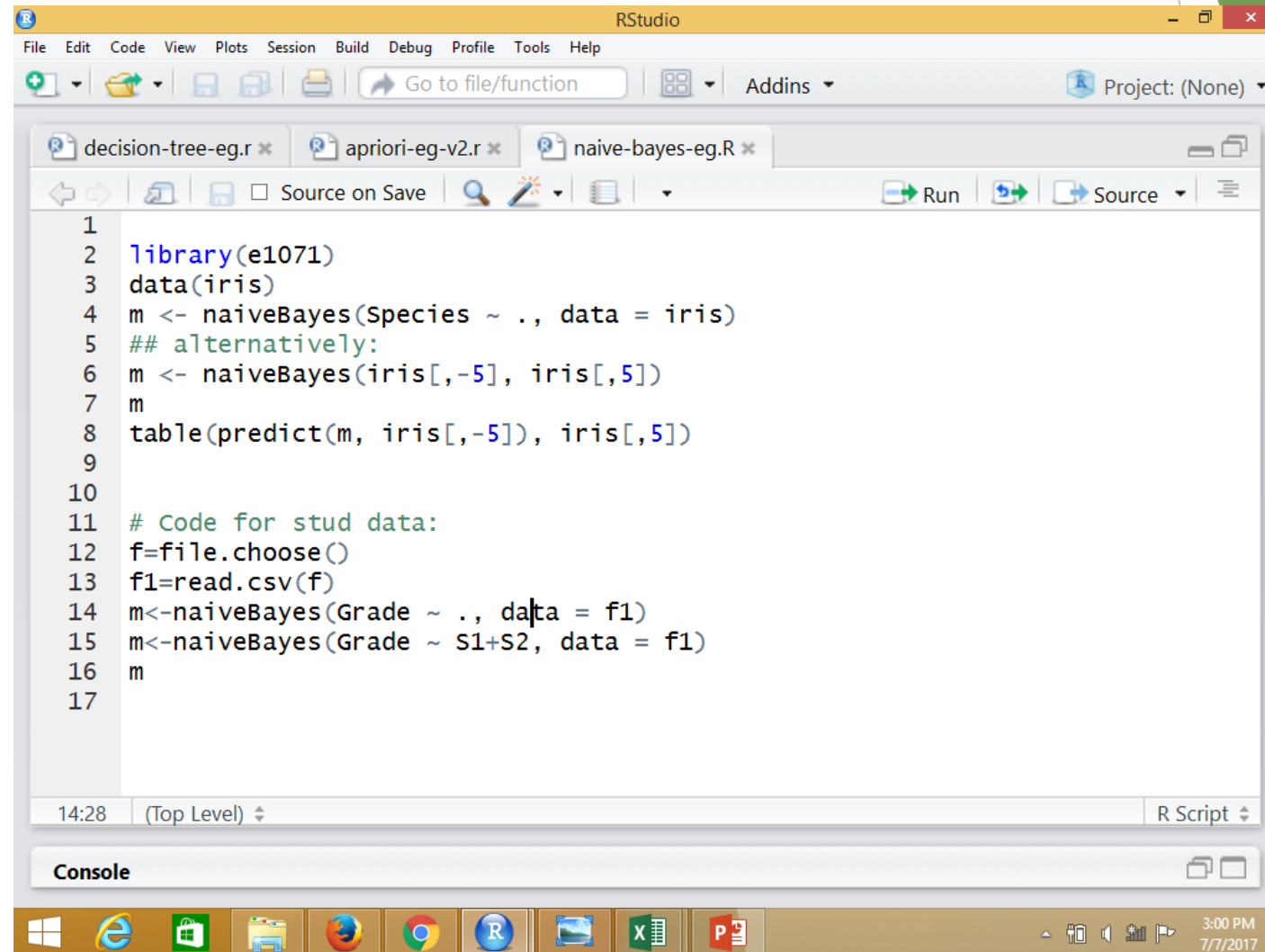
- For example, lung cancer is influenced by a person's family history of lung cancer, as well as whether or not the person is a smoker. It is worth noting that the variable Positive Xray is independent of whether the patient has a family history of lung cancer or that the patient is a smoker, given that we know the patient has lung cancer.



Write your own example now!!!!

How to Apply Naïve Bayes method in R

► Go to code

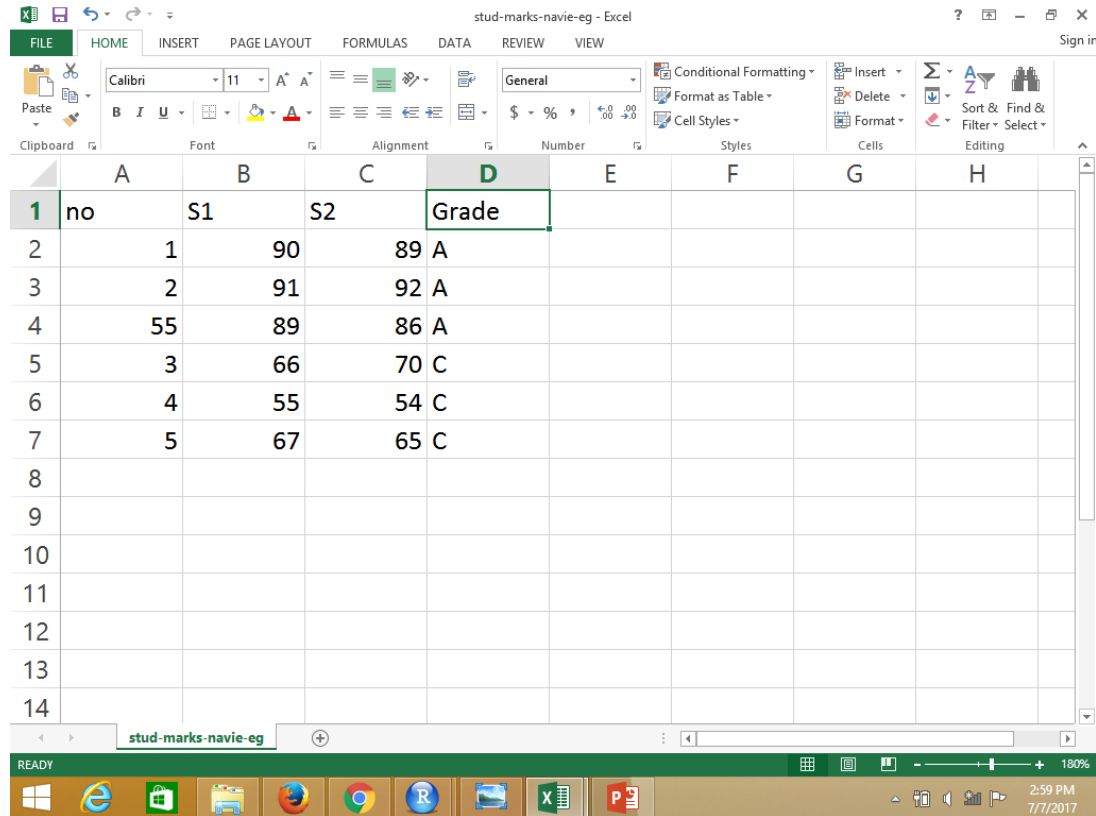


The screenshot shows the RStudio interface with a script editor containing R code for Naïve Bayes classification. The code includes loading the 'e1071' library, using the 'iris' dataset, and applying the 'naiveBayes' function. It also shows an alternative method for reading data from a file. The console at the bottom is empty.

```
1  
2 library(e1071)  
3 data(iris)  
4 m <- naiveBayes(Species ~ ., data = iris)  
5 ## alternatively:  
6 m <- naiveBayes(iris[,-5], iris[,5])  
7 m  
8 table(predict(m, iris[,-5]), iris[,5])  
9  
10  
11 # Code for stud data:  
12 f=file.choose()  
13 f1=read.csv(f)  
14 m<-naiveBayes(Grade ~ ., data = f1)  
15 m<-naiveBayes(Grade ~ S1+S2, data = f1)  
16 m  
17
```

14:28 (Top Level) R Script

Stud data



The screenshot shows a Microsoft Excel spreadsheet titled "stud-marks-navie-eg - Excel". The spreadsheet contains a table with student data. The columns are labeled A through H, and the rows are numbered 1 through 14. The data is as follows:

	A	B	C	D	E	F	G	H
1	no	S1	S2	Grade				
2	1	90	89	A				
3	2	91	92	A				
4	55	89	86	A				
5	3	66	70	C				
6	4	55	54	C				
7	5	67	65	C				
8								
9								
10								
11								
12								
13								
14								

The Excel interface includes the ribbon with tabs for FILE, HOME, INSERT, PAGE LAYOUT, FORMULAS, DATA, REVIEW, and VIEW. The HOME tab is active, showing options for Clipboard, Font, Alignment, Number, Styles, Conditional Formatting, Insert, Delete, Format, Cell Styles, and Editing. The status bar at the bottom indicates "READY" and the time "2:59 PM 7/7/2017".

References

- ▶ <https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/>
- ▶ https://www.tutorialspoint.com/data_mining/dm_bayesian_classification.htm
- ▶ <http://ugrad.stat.ubc.ca/R/library/e1071/html/naiveBayes.html>

Page ranking –Data mining technique

- ▶ As the **web is growing** rapidly, the users get easily lost in the web's rich hyper structure.
- ▶ The primary goal of the web site owner is to provide the relevant information to the users to fulfill their needs. Web mining technique is used to **categorize users** and pages by analyzing **users behavior**, the content of pages and order of URLs accessed.
- ▶ The PageRank algorithm outputs a **probability distribution** used to represent the likelihood that a person randomly clicking on links will arrive at any particular page.
- ▶ A probability is expressed as a numeric value between 0 and 1. A **0.5 probability** is commonly expressed as a "**50% chance**" of something happening. Hence, a PageRank of 0.5 means there is a 50% chance that a person clicking on a random link will be directed to the document with the 0.5 PageRank.

WEB MINING

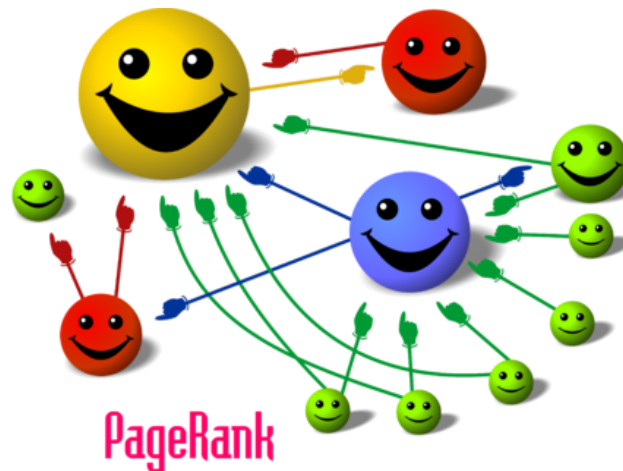
- ▶ Resource finding
- ▶ Information selection and pre-processing
- ▶ Generalization : automatically discovers general patterns at individual web sites
- ▶ Analysis : knowledge discovery process

WEB MINING CATEGORIES

- ▶ Web Content Mining
- ▶ Web Structure Mining : Interlinking
- ▶ Web Usage Mining

PageRank

- ▶ This algorithm was developed by **Brin and Page at Stanford University** which extends the idea of **citation analysis**
- ▶ The link from one page to another is considered as a **vote**. Not only the number of **votes that a page receives** is important but the importance of **pages that casts the vote** is also important



Page rank algorithm

- ▶ PageRank (PR) is an algorithm used by Google Search to **rank** websites in their search engine results. PageRank is a way of measuring the importance of website pages.

$$PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

Where:

- ▶ $PR(A)$ is the PageRank of page A,
- ▶ $PR(Ti)$ is the PageRank of pages Ti which link to page A,
- ▶ $C(Ti)$ is the number of outbound links on page Ti ;
- ▶ d is a damping factor which can be set between 0 and 1.

Continued

- ▶ PageRank™ algorithm does **not rank the whole website**, but it's determined for each page individually.
- ▶ Eg:
 - ▶ Assume a small universe of four web pages: **A**, **B**, **C** and **D**. Links from a page to itself, or multiple outbound links from one single page to another single page, are ignored.
 - ▶ Hence the initial value for each page in this example is 0.25.
 - ▶ The PageRank transferred from a given page to the targets of its outbound links upon the next iteration is divided equally among all outbound links.

$$PR(A) = PR(B) + PR(C) + PR(D).$$

- ▶ If the only links in the system were from pages **B**, **C**, and **D** to **A**, each link would transfer 0.25 PageRank to **A** upon the next iteration, for a total of 0.75.

Example continued

- Suppose instead that page **B** had a link to pages **C** and **A**, page **C** had a link to page **A**, and page **D** had links to all three pages. Thus, upon the first iteration, page **B** would transfer half of its existing value, or 0.125, to page **A** and the other half, or 0.125, to page **C**. Page **C** would transfer all of its existing value, 0.25, to the only page it links to, **A**. Since **D** had three outbound links, it would transfer one third of its existing value, or approximately 0.083, to **A**. At the completion of this iteration, page **A** will have a PageRank of approximately 0.458.

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}$$

Page rank continued

- ▶ In other words, the PageRank conferred by an outbound link is equal to the document's own PageRank score divided by the number of outbound links $L()$.

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}.$$

- ▶ In the general case, the PageRank value for any page u can be expressed as

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

- ▶ i.e. the PageRank value for a page u is dependent on the PageRank values for each page v contained in the set B_u (the set containing all pages linking to page u), divided by the number $L(v)$ of links from page v

Damping factor

- ▶ The PageRank theory holds that an **imaginary surfer** who is **randomly** clicking on links will eventually stop clicking. The probability, at any step, that the person will continue is a **damping factor d** . Various studies have tested different damping factors, but it is generally assumed that the damping factor will be set around 0.85
- ▶ The damping factor is subtracted from 1 (and in some variations of the algorithm, the result is divided by the number of documents (N) in the collection) and this term is then added to the product of the damping factor and the sum of the incoming PageRank scores

Page ranking Continued

$$PR(A) = \frac{1-d}{N} + d \left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \dots \right)$$

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Continued

- The PageRank values are the entries of the dominant right eigenvector of the modified adjacency matrix.

$$\mathbf{R} = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \vdots \\ PR(p_N) \end{bmatrix}$$

where \mathbf{R} is the solution of the equation

$$\mathbf{R} = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \vdots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} \ell(p_1, p_1) & \ell(p_1, p_2) & \cdots & \ell(p_1, p_N) \\ \ell(p_2, p_1) & \ddots & & \vdots \\ \vdots & & \ell(p_i, p_j) & \\ \ell(p_N, p_1) & \cdots & & \ell(p_N, p_N) \end{bmatrix} \mathbf{R}$$

Page ranking Code

► [Go to Code](#)

References

- ▶ <https://pdfs.semanticscholar.org/12e3/bd728c12bd8f3dfb5c381ec92c3e23337326.pdf>
- ▶ www2.cs.uh.edu/~ceick/UDM/dm_PageRank.pptx
- ▶ <https://en.wikipedia.org/wiki/PageRank>

KNN Datamining Technique

- ▶ **K Nearest Neighbor**
- ▶ KNN assumes that the data is in a *feature space*. More exactly, the data points are in a metric space. The data can be scalars or possibly even multidimensional vectors. Since the points are in feature space, they have a notion of distance – This need not necessarily be Euclidean distance although it is the one commonly used.
- ▶ **The learning process**
 - ▶ Unlike many artificial learners, ***instance-based learners*** do not abstract any information from the training data during the learning phase. Learning is merely a question of encapsulating the training data. The process of ***generalization*** is postponed until it is absolutely unavoidable, that is, at the time of *classification*.

Classification

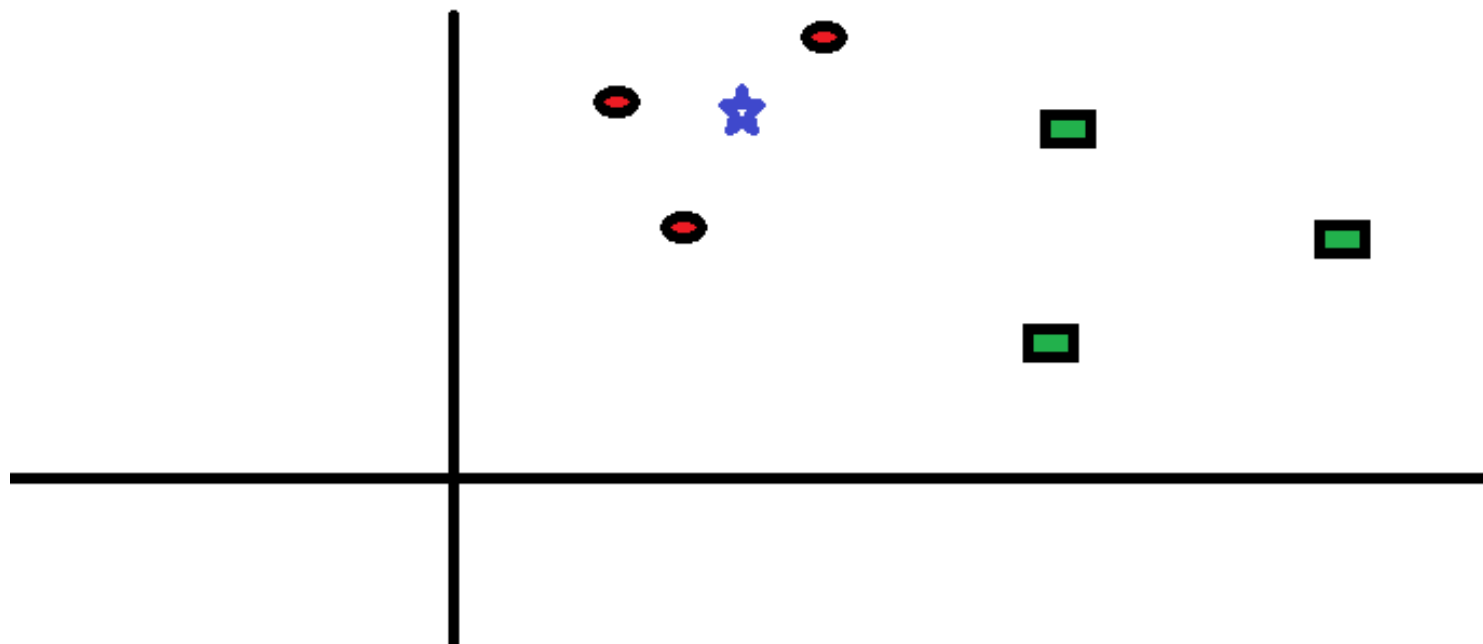
- ▶ Classification (*generalization*) using an *instance-based* classifier can be a simple matter of locating the nearest neighbour in *instance space* and labelling the unknown instance with the same class label as that of the located (known) neighbour. This approach is often referred to as a *nearest neighbour classifier*.
- ▶ More robust models can be achieved by locating k , where $k > 1$, neighbours and letting the majority vote decide the outcome of the class labelling. A higher value of k results in a smoother, less locally sensitive, function. The *nearest neighbour classifier* can be regarded as a special case of the more general *k-nearest neighbours classifier*

KNN

- *Instance-based* classifiers such as the k NN classifier operate on the premises that classification of unknown instances can be done by relating the unknown to the known according to some distance/similarity function. The intuition is that two instances far apart in the *instance space* defined by the appropriate *distance function* are less likely than two closely situated instances to belong to the same class.

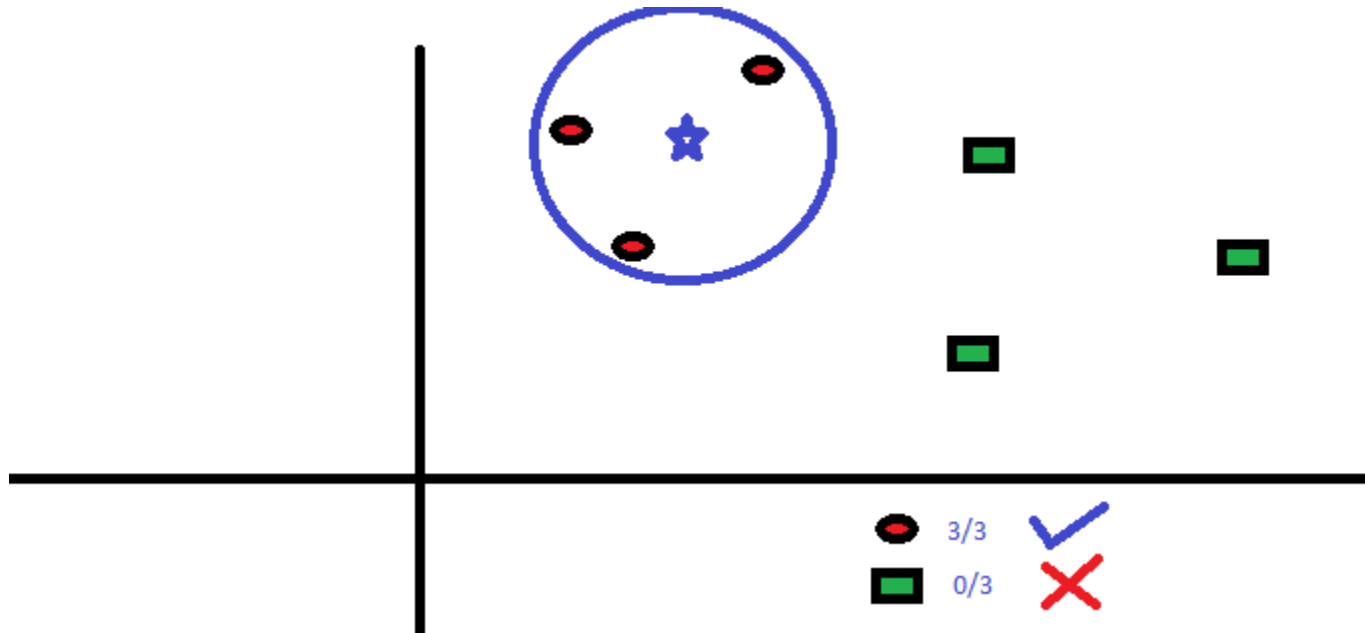
How does the KNN algorithm work?

Example : Students Grades



Example continued

- The “K” in KNN algorithm is the nearest neighbors we wish to take vote from. Let's say $K = 3$



KNN for Classification

- ▶ In this algorithm, we will given some data points for training and also a new un-labelled data for testing. Our aim is to find the class label for the new point.
- ▶ **Case 1: $k=1$** , Let x be the point to be labeled . Find the point closest to x . Let it be y . Now nearest neighbor rule asks to assign the label of y to x .
- ▶ **Case 2 : $k = K$ or k -Nearest Neighbor Rule**

Continued

- ▶ In pattern recognition, the ***k*-nearest neighbors algorithm (*k*-NN)** is a non-parametric method used for classification and regression
- ▶ In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k -NN is used for classification or regression:
 - ▶ In *k*-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor
 - ▶ In *k*-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors

Contd..

- ▶ The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.
- ▶ In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.
- ▶ For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (Hamming distance). In the context of gene expression microarray data, for example, k -NN has also been employed with correlation coefficients such as Pearson and Spearman.

Continued

- ▶ A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number.
- ▶ One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point.

Applications of KNN

- ▶ Nearest Neighbor based Content Retrieval
 - ▶ handwriting detection
 - ▶ closest gesture
- ▶ Gene Expression

Another Example :

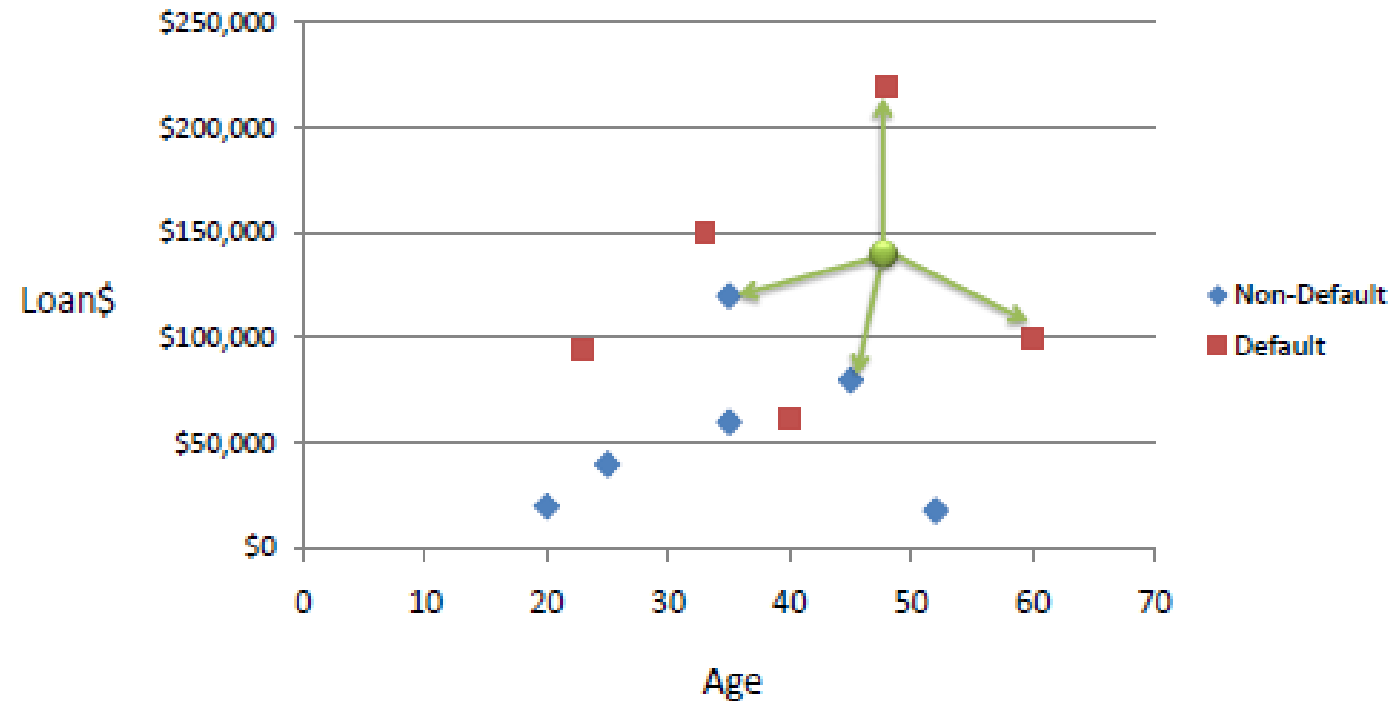
Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.

Age	Loan	Default	Distance	
25	\$40,000	N	102000	
35	\$60,000	N	82000	
45	\$80,000	N	62000	
20	\$20,000	N	122000	
35	\$120,000	N	22000	2
52	\$18,000	N	124000	
23	\$95,000	Y	47000	
40	\$62,000	Y	80000	
60	\$100,000	Y	42000	3
48	\$220,000	Y	78000	
33	\$150,000	Y	8000	1
48	\$142,000	?		

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Continued



We can now use the training set to classify an unknown case (Age=48 and Loan=\$142,000) using Euclidean distance. If **K=1** then the nearest neighbor is the last case in the training set with Default=Y.

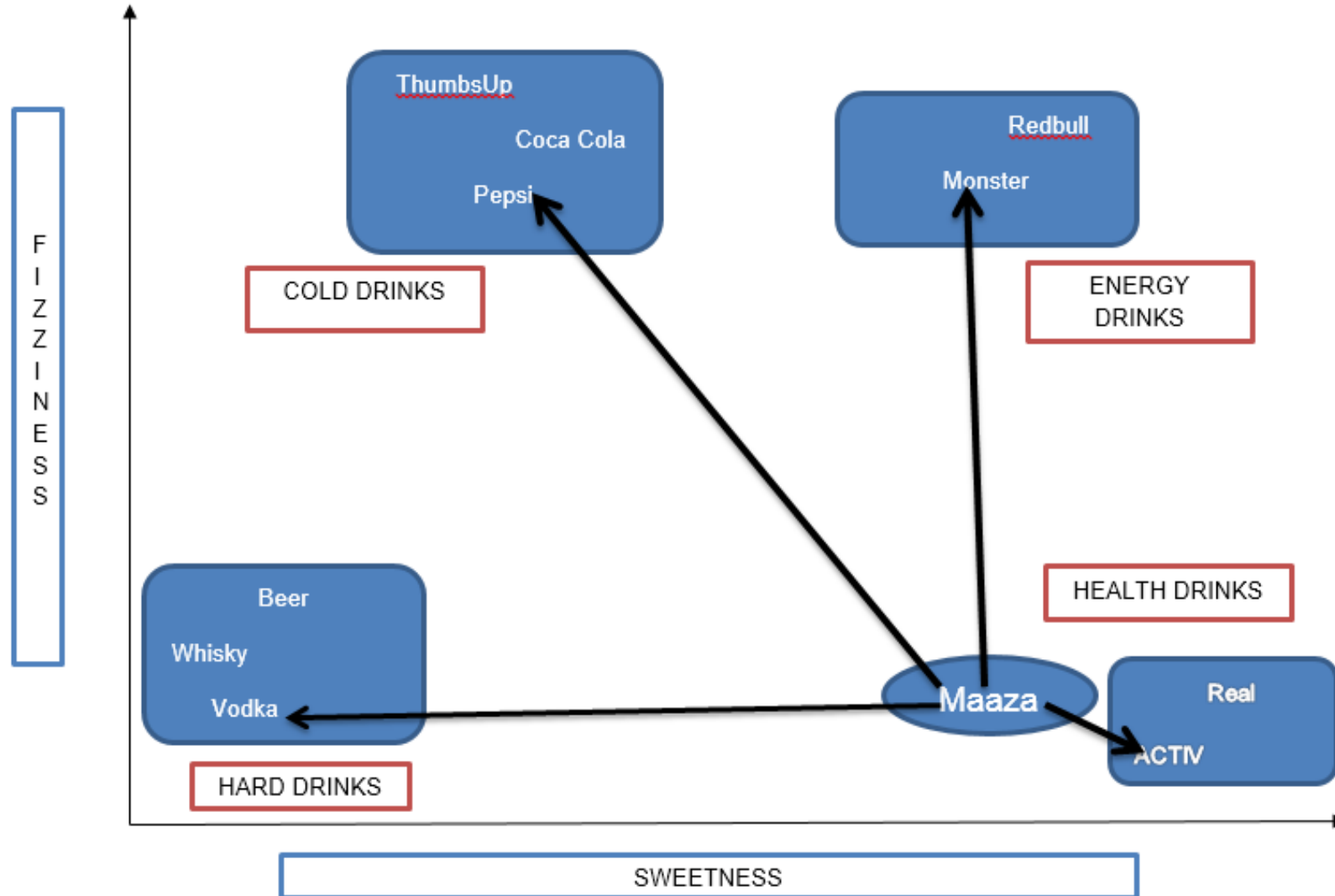
$$D = \text{Sqrt}[(48-33)^2 + (142000-150000)^2] = 8000.01 \gg \text{Default=Y}$$

Another Example

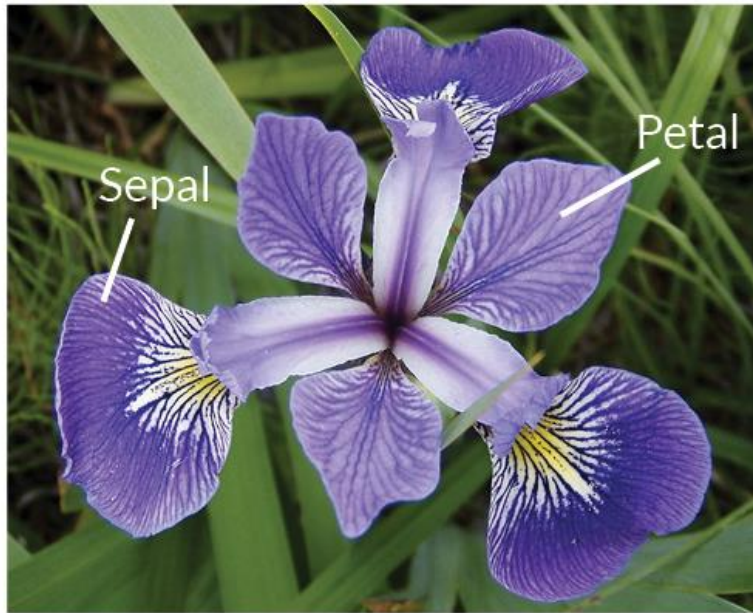
- Let's consider 10 'drinking items' which are rated on two parameters on a scale of 1 to 10. The two parameters are “sweetness” and “fizziness”.

Ingredient	Sweetness	Fizziness	Type of Drink
Monster	8	8	Energy booster
ACTIV	9	1	Health drink
Pepsi	4	8	Cold drink
Vodka	2	1	Hard drink

Continued



Example for Code



Iris Versicolor



Iris Setosa



Iris Virginica

How to do in R?

► Go to R

k-Nearest Neighbor Classification

- ▶ Nearest neighbour classification that can return class votes for all classes.
- ▶ Usage
 - ▶ `knn3(x, ...)`
 - ▶ # S3 method for formula
 - ▶ `knn3(formula, data, subset, na.action, k = 5, ...)`
 - ▶ # S3 method for data.frame
 - ▶ `knn3(x, y, k = 5, ...)`
 - ▶ # S3 method for matrix
 - ▶ `knn3(x, y, k = 5, ...)`
 - ▶ # S3 method for knn3
 - ▶ `print(x, ...)`
 - ▶ `knn3Train(train, test, cl, k = 1, l = 0, prob = TRUE, use.all = TRUE)`

Arguments

- ▶ **X**
 - ▶ a matrix of training set predictors
- ▶ **...**
 - ▶ additional parameters to pass to `knn3Train`. However, passing `prob = FALSE` will be overridden.
- ▶ **Formula**
 - ▶ a formula of the form `lhs ~ rhs` where `lhs` is the response variable and `rhs` a set of predictors.
- ▶ **Data**
 - ▶ optional data frame containing the variables in the model formula.
- ▶ **Subset**
 - ▶ optional vector specifying a subset of observations to be used.
- ▶ **na.action**
 - ▶ function which indicates what should happen when the data contain NAs.

Contd....

- ▶ K
 - ▶ number of neighbours considered.
- ▶ Y
 - ▶ a factor vector of training set classes
- ▶ Train
 - ▶ matrix or data frame of training set cases.
- ▶ Test
 - ▶ matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
- ▶ Cl
 - ▶ factor of true classifications of training set

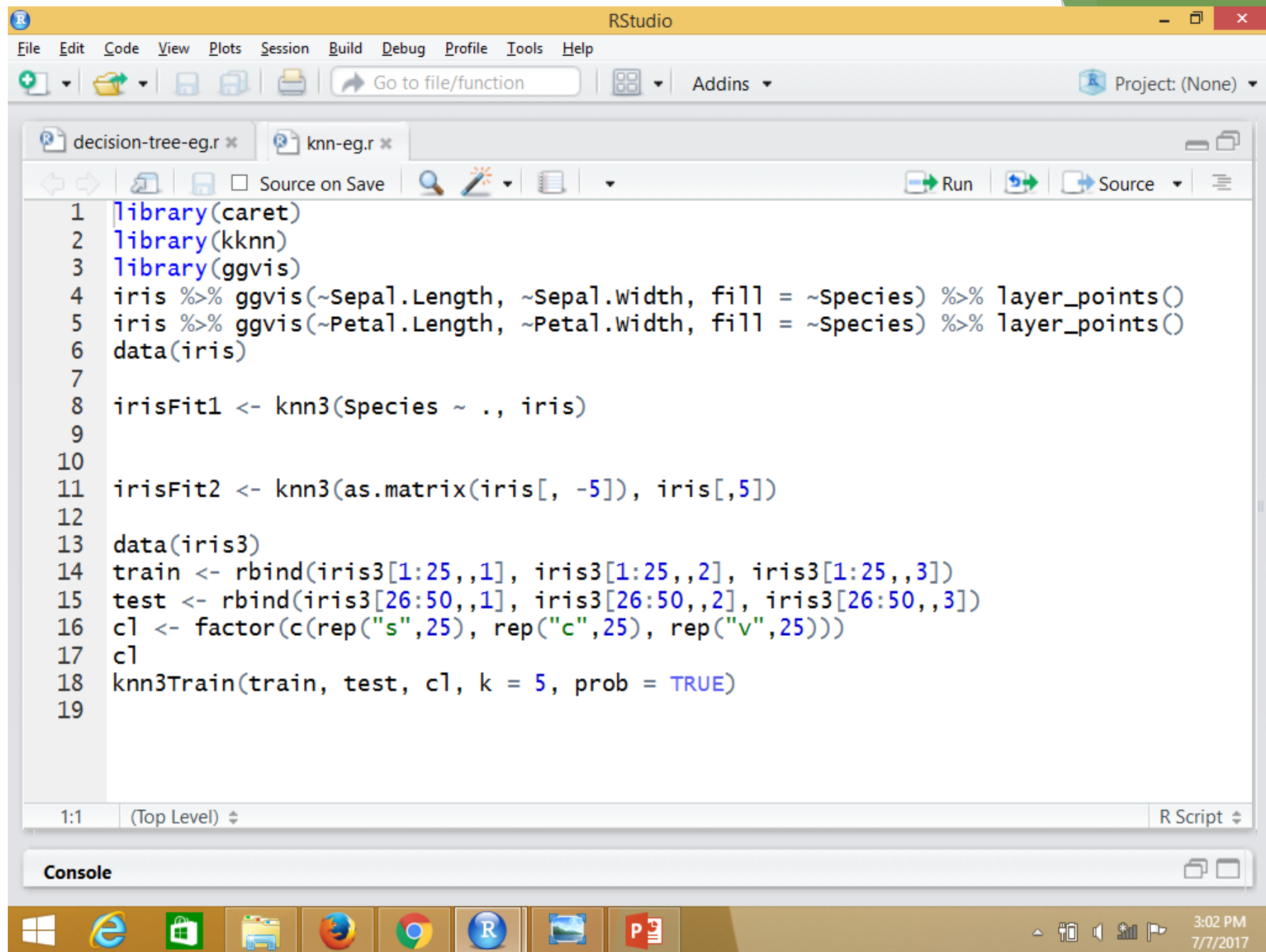
Contd..

- ▶ L
 - ▶ minimum vote for definite decision, otherwise doubt. (More precisely, less than $k-l$ dissenting votes are allowed, even if k is increased by ties.)
- ▶ Prob
 - ▶ If this is true, the proportion of the votes for each class are returned as attribute prob.
- ▶ use.all
 - ▶ controls handling of ties. If true, all distances equal to the k th largest are included. If false, a random selection of distances equal to the k th is chosen to use exactly k neighbours.

Details

- ▶ `knn3` is essentially the same code as `ipredknn` and `knn3Train` is a copy of `knn`. The underlying C code from the class package has been modified to return the vote percentages for each class (previously the percentage for the winning class was returned).

R code



```
1 library(caret)
2 library(kknn)
3 library(ggvis)
4 iris %>% ggvis(~Sepal.Length, ~Sepal.Width, fill = ~Species) %>% layer_points()
5 iris %>% ggvis(~Petal.Length, ~Petal.Width, fill = ~Species) %>% layer_points()
6 data(iris)
7
8 irisFit1 <- knn3(Species ~ ., iris)
9
10
11 irisFit2 <- knn3(as.matrix(iris[, -5]), iris[,5])
12
13 data(iris3)
14 train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
15 test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
16 cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
17 cl
18 knn3Train(train, test, cl, k = 5, prob = TRUE)
19
```


R code for Stud data

stud-marks-navie-eg - Excel

	A	B	C	D	E	F	G	H	I	J	K
1	no	S1	S2	Grade							
2	1	90	89	A							
3	2	91	92	A							
4	55	89	86	A							
5	3	66	70	C							
6	4	55	54	C							
7	5	67	65	C							

stud-marks-navie-eg

RStudio

```
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins Project: (None)

decision-tree-eg.r knn-eg-stud.r*
Source on Save Run Source

1 library(caret)
2 library(kknn)
3 library(ggvis)
4 f=file.choose()
5 f1=read.csv(f)
6 k1 <- kknn3(Grade ~ ., f1)
7
8 train <- rbind(f1[1:3,2:3],f1[4:6,2:3])
9 test <- f1[1:2,2:3]
10 c1 <- factor(c(rep("A",3), rep("C",3)))
11 c1
12 kknn3Train(train, test, c1, k = 5, prob = TRUE)
13
```

13:1 (Top Level) R Script

Console F:/MSRIT-2014-onwards/MSRIT 2017/Data-science-vocational-course-2017/

```
> c1
[1] A A A C C C
Levels: A C
> kknn3Train(train, test, c1, k = 5, prob = TRUE)
[1] "A" "A"
attr(,"prob")
      A      C
[1,] 0.6 0.4
[2,] 0.6 0.4
```

3:09 PM 7/7/2017

References

- ▶ <https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/>
- ▶ <https://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/>
- ▶ https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- ▶ http://www.fon.hum.uva.nl/praat/manual/kNN_classifiers_1_What_is_a_kNN_classifier.html
- ▶ http://www.saedsayad.com/k_nearest_neighbors.htm
- ▶ http://people.revoledu.com/kardi/tutorial/KNN/KNN_Numerical-example.html
- ▶ <https://www.rdocumentation.org/packages/caret/versions/6.0-76/topics/knn3>

C4.5 algorithm : Decision tree

- ▶ C4.5 is a computer program for inducing classification rules in the form of decision trees from a set of given instances
- ▶ Select one attribute from a set Of training instances
- ▶ Select an initial subset of the training instances
- ▶ Use the attribute and the subset of instances to build a decision tree
- ▶ Use the rest of the training instances (those not in the subset used for construction) to test the accuracy of the constructed tree
- ▶ If all instances are correctly classified – stop
- ▶ If an instances is incorrectly classified, add it to the initial subset and construct a new tree

Continued

- ▶ Iterate until
 - ▶ A tree is built that classifies all instance correctly
- OR
- ▶ A tree is built from the entire training set
 - ▶ Choose an attribute that best differentiates the instances contained in T
 - ▶ Create a tree node whose value is the chosen attribute

Simplified Algorithm

- ▶ Let T be the set of training instances
- ▶ Choose an attribute that best differentiates the instances contained in T
- ▶ Create child links from this node where each link represents a unique value for the chosen attribute
- ▶ Use the child link values to further subdivide the instances into subclasses

Example

► Credit Card Promotion Data

Attribute Name	Value Description	Numeric Values	Definition
Income Range	20-30K, 30-40K, 40-50K, 50-60K	20000, 30000, 40000, 50000	Salary range for an individual credit card holder
Magazine Promotion	Yes, No	1, 0	Did card holder participate in magazine promotion offered before?
Watch Promotion	Yes, No	1, 0	Did card holder participate in watch promotion offered before?
Life Ins Promotion	Yes, No	1, 0	Did card holder participate in life insurance promotion offered before?
Credit Card Insurance	Yes, No	1, 0	Does card holder have credit card insurance?
Sex	Male, Female	1, 0	Card holder's gender
Age	Numeric	Numeric	Card holder's age in whole years

Problem to be Solved from Data

- ▶ **A Credit Card Company is going to do a life insurance promotion** – sending the promo materials with billing statements. They have done a similar promotion in the past, with results as represented by the data set. They want to target the new promo materials to credit card holders similar to those who took advantage of the prior life insurance promotion.
- ▶ Use supervised learning with output attribute = life insurance promotion to develop a profile for credit card holders likely to accept the new promotion.

Sample of Credit Card Promotion Data

Income Range	Magazine Promo	Watch Promo	Life Ins Promo	CC Ins	Sex	Age
40-50K	Yes	No	No	No	Male	45
30-40K	Yes	Yes	Yes	No	Female	40
40-50K	No	No	No	No	Male	42
30-40K	Yes	Yes	Yes	Yes	Male	43
50-60K	Yes	No	Yes	No	Female	38
20-30K	No	No	No	No	Female	55
30-40K	Yes	No	Yes	Yes	Male	35
20-30K	No	Yes	No	No	Male	27
30-40K	Yes	No	No	No	Male	43
30-40K	Yes	Yes	Yes	No	Female	41

Problem Characteristics

- ▶ Life insurance promotion is the output attribute
- ▶ Input attributes are income range, credit card insurance, sex, and age
- ▶ Attributes related to the instance's response to other promotions is not useful for prediction because new credit card holders will not have had a chance to take advantage of these prior offers (except for credit card insurance which is always offered immediately to new card holders)
- ▶ Therefore, magazine promo and watch promo are not relevant for solving the problem at hand – disregard – do not include this data in data mining

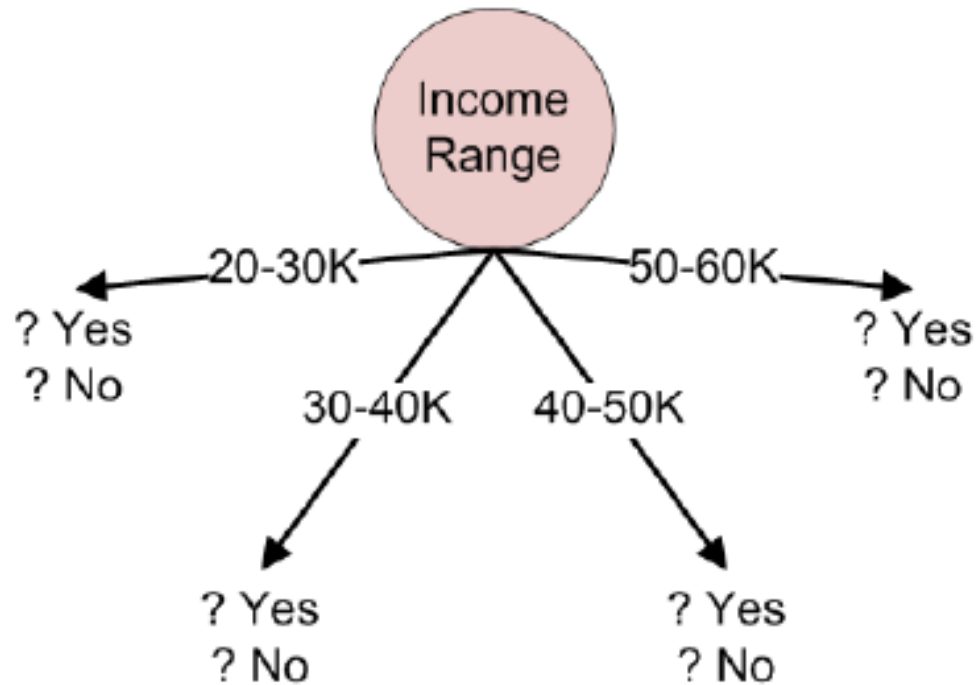
Apply the Simplified C4.5 Algorithm to the Credit Card Promotion Data

Income Range	Magazine Promo	Watch Promo	Life Ins Promo	CC Ins	Sex	Age
40-50K	Yes	No	No	No	Male	45
30-40K	Yes	Yes	Yes	No	Female	40
40-50K	No	No	No	No	Male	42
30-40K	Yes	Yes	Yes	Yes	Male	43
50-60K	Yes	No	Yes	No	Female	38
20-30K	No	No	No	No	Female	55
30-40K	Yes	No	Yes	Yes	Male	35
20-30K	No	Yes	No	No	Male	27
30-40K	Yes	No	No	No	Male	43
30-40K	Yes	Yes	Yes	No	Female	41

Step 2: Which input attribute best differentiates the instances?

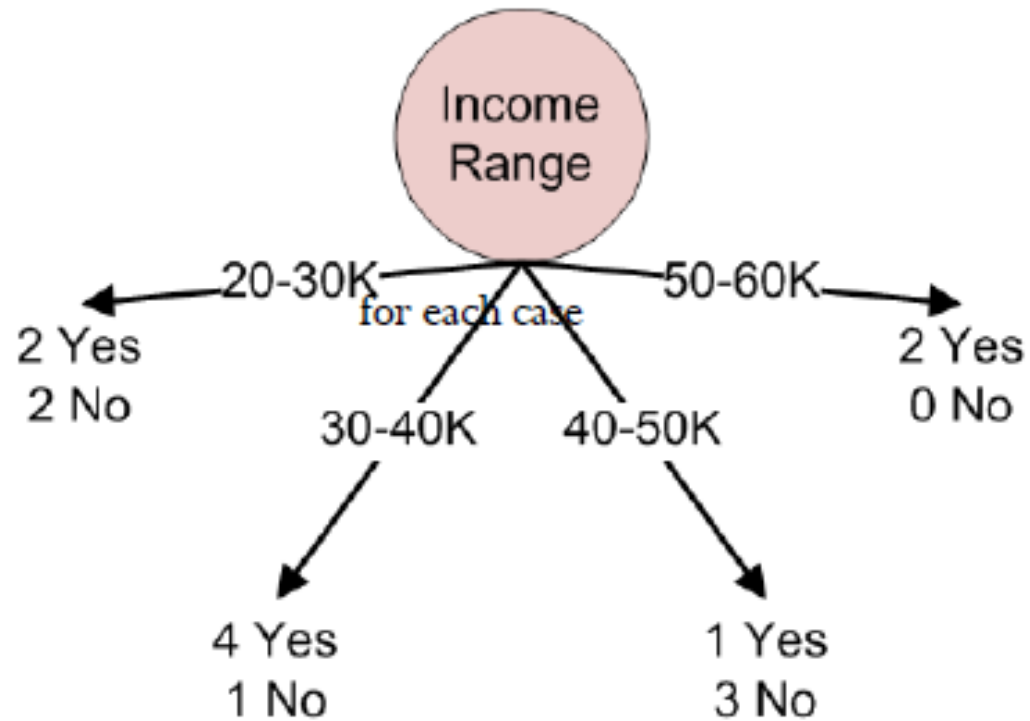
Income Range	Magazine Promo	Watch Promo	Life Ins Promo	CC Ins	Sex	Age
40-50K	Yes	No	No	No	Male	45
30-40K	Yes	Yes	Yes	No	Female	40
40-50K	No	No	No	No	Male	42
30-40K	Yes	Yes	Yes	Yes	Male	43
50-60K	Yes	No	Yes	No	Female	38
20-30K	No	No	No	No	Female	55
30-40K	Yes	No	Yes	Yes	Male	35
20-30K	No	Yes	No	No	Male	27
30-40K	Yes	No	No	No	Male	43
30-40K	Yes	Yes	Yes	No	Female	41

Apply Simplified C4.5



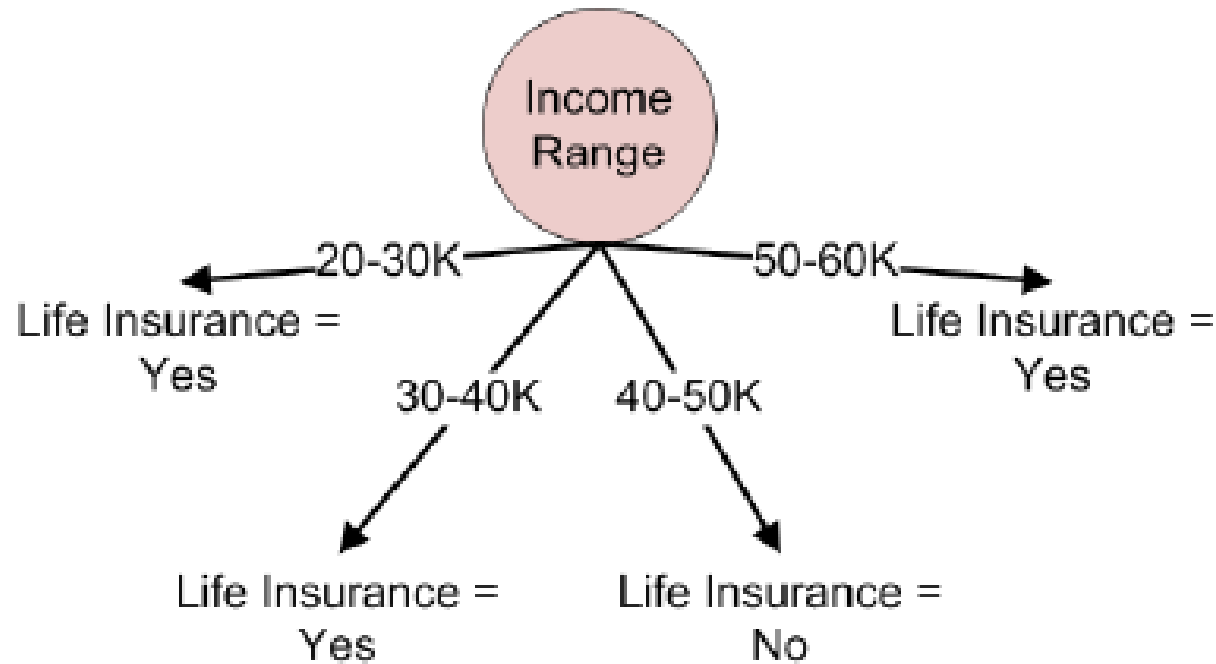
For each case (attribute value), how many instances of Life Insurance Promo = Yes and Life Insurance Promo = No?

Apply Simplified C4.5



For each branch, choose the most frequently occurring decision. If there is a tie, then choose Yes, since there are more overall Yes instances (9) than No instances (6) with respect to Life Insurance Promo

Apply Simplified C4.5



Evaluate the classification model (the tree) on the basis of accuracy. How many of the 15 training instances are classified correctly by this tree?

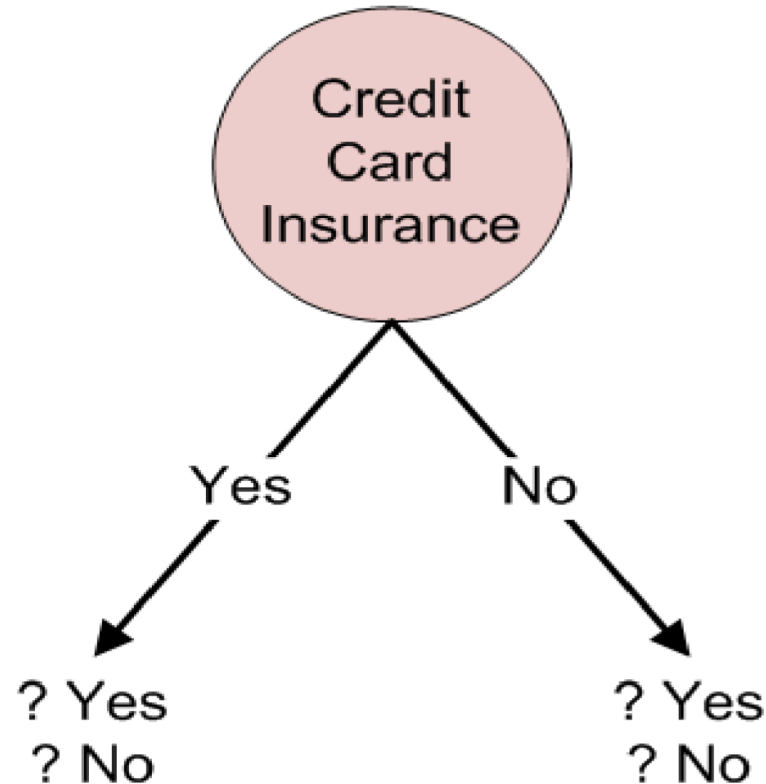
Apply Simplified C4.5

- ▶ Tree accuracy = $11/15 = 73.3\%$
- ▶ Tree cost = 4 branches for the computer program to use
- ▶ Goodness score for Income Range attribute is $11/15/4 = 0.183$
- ▶ Including Tree “cost” to assess goodness lets us compare trees

Apply Simplified C4.5

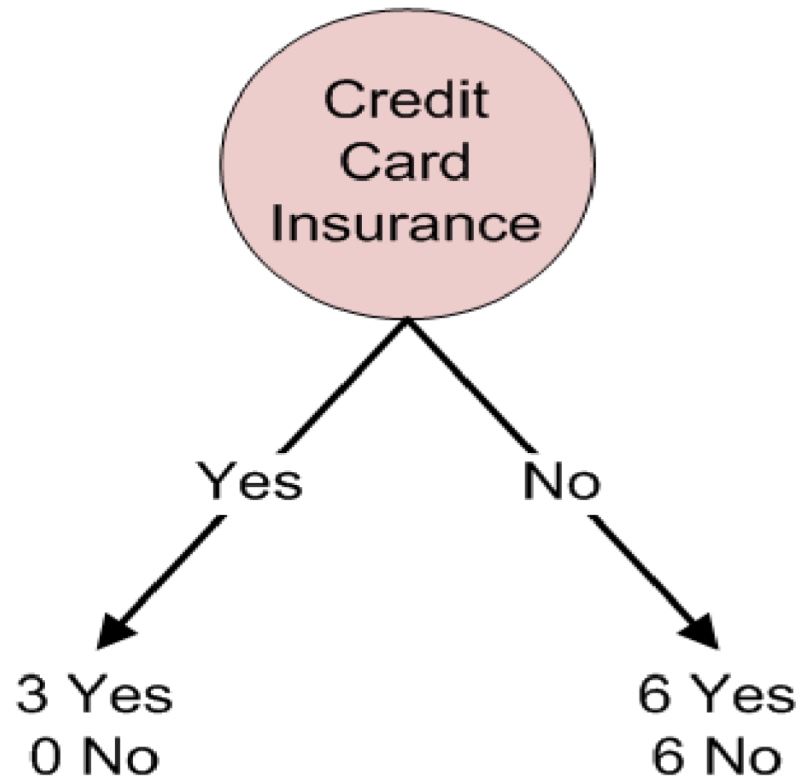
Consider a Different Top-Level Node

- For each case (attribute value), how many instances of Life Insurance Promo = Yes and Life Insurance Promo = No?



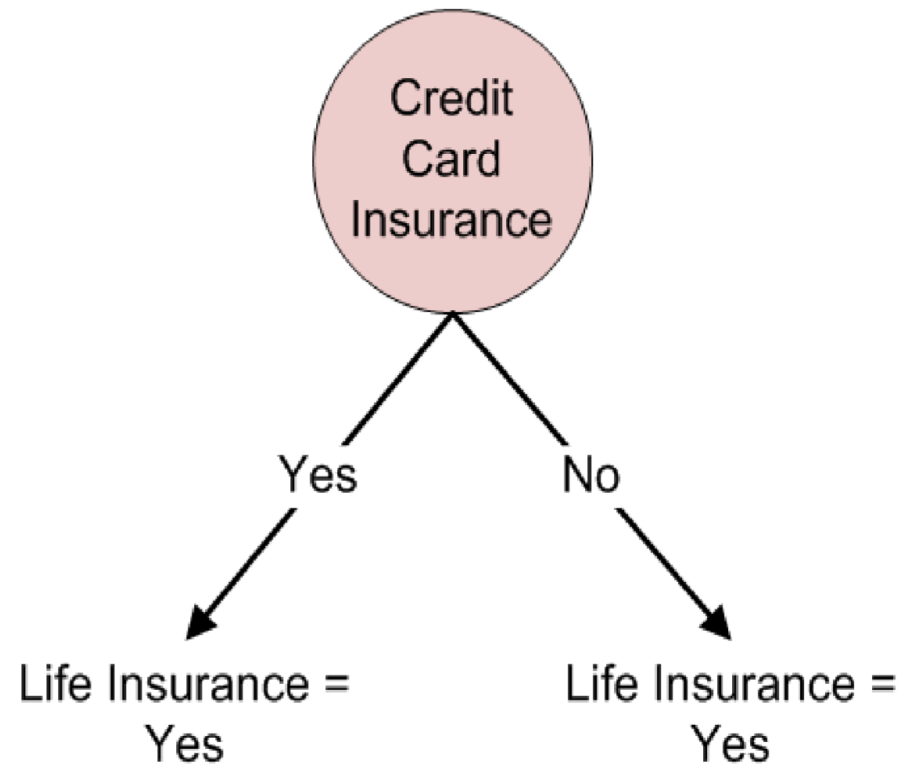
Apply Simplified C4.5

- For each branch, choose the most frequently occurring decision. If there is a tie, then choose Yes, since there are more total Yes instances (9) than No instances (6).



Contd

- Evaluate the classification model (the tree). How many of the 15 training instances are classified correctly by this tree?

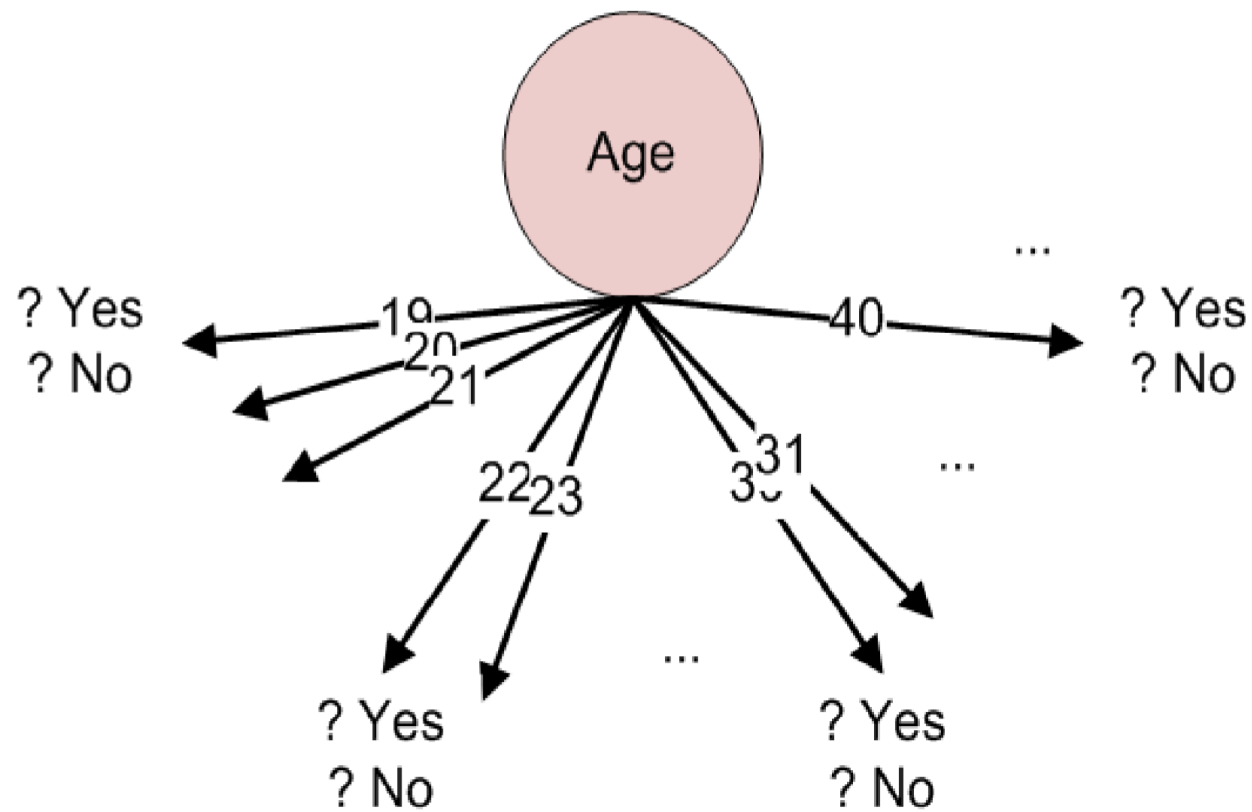


Contd

- ▶ Tree accuracy = $9/15 = 60.0\%$
- ▶ Tree cost = 2 branches for the computer program to use
- ▶ Goodness score for Income Range attribute is $9/15/2 = 0.300$
- ▶ Including Tree “cost” to assess goodness lets us compare trees

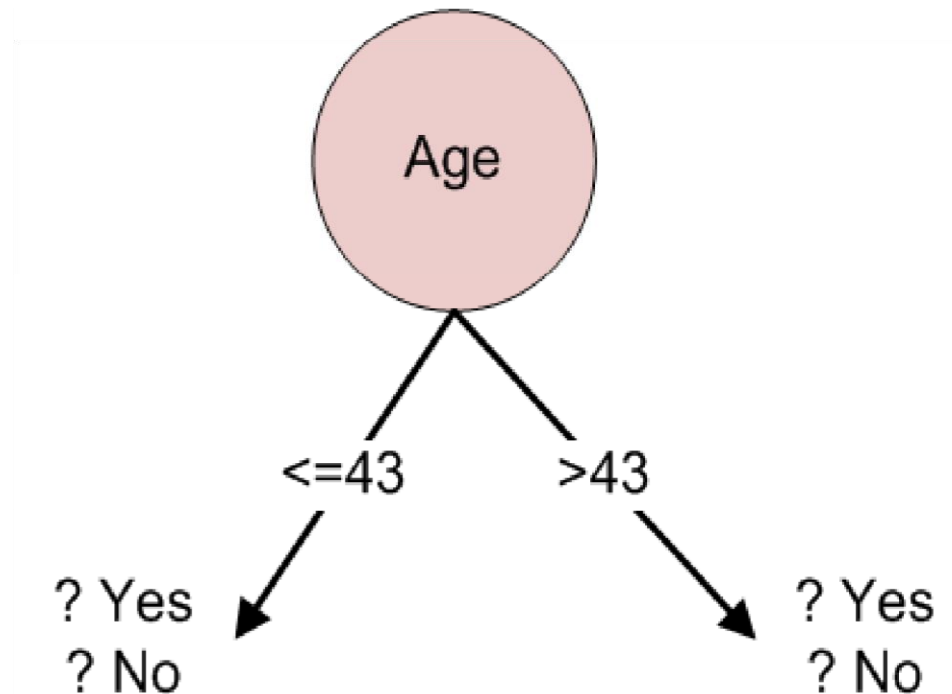
Contd

- What's problematic about this?



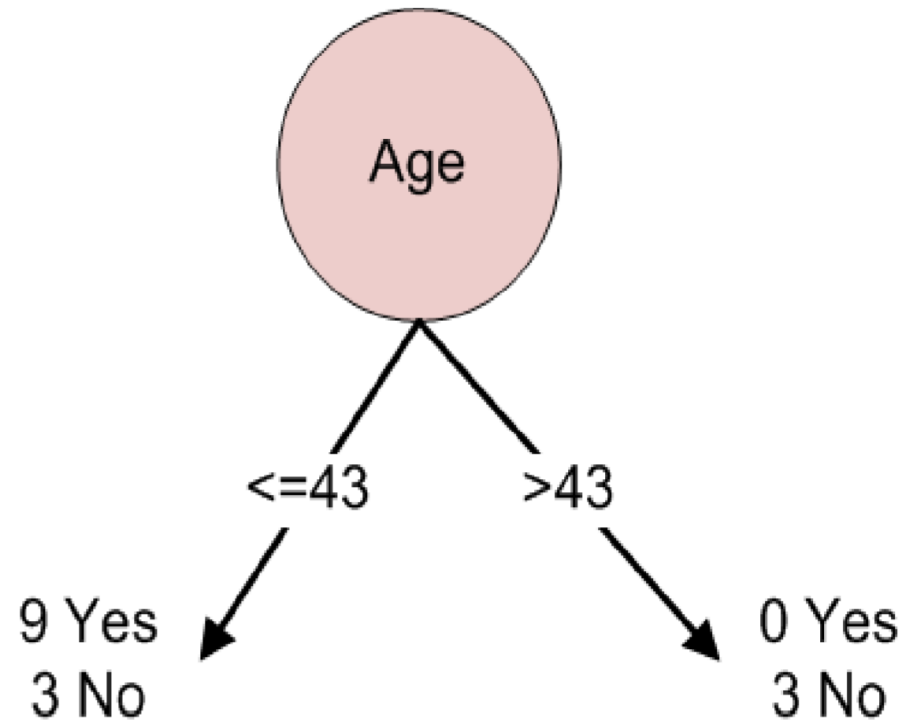
Contd

- ▶ How many instances for each case?
- ▶ A binary split requires the addition of only two branches. Why 43?



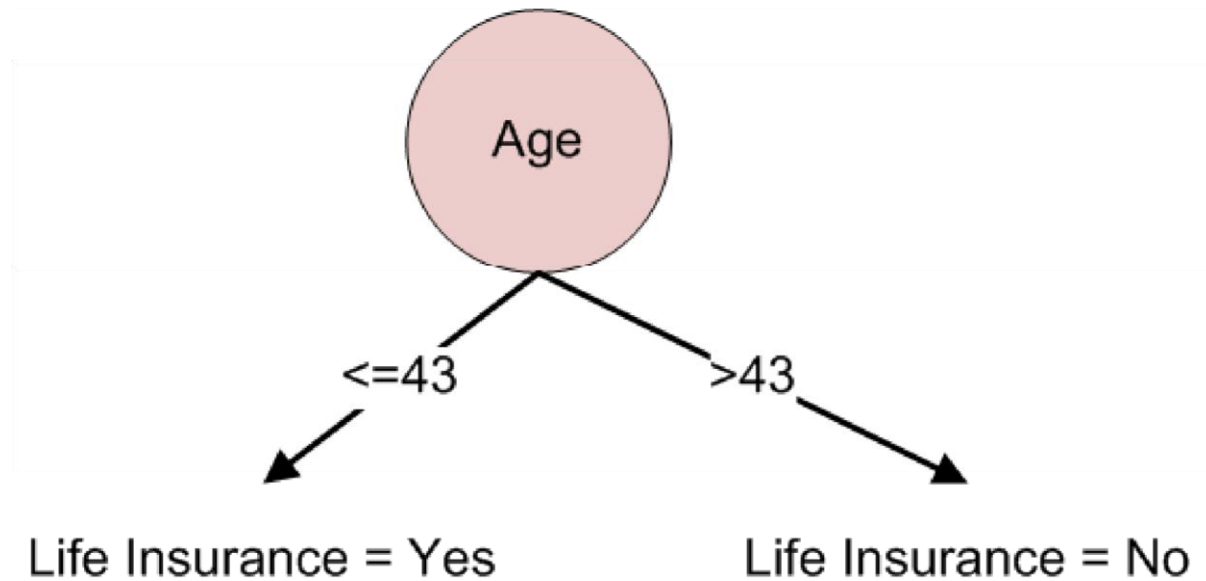
Contd

- For branch decision, If tie in branch, choose the most frequently occurring decision. there is a tie, then choose Yes, since there are more total Yes instances (9) than No instances (6).



Contd

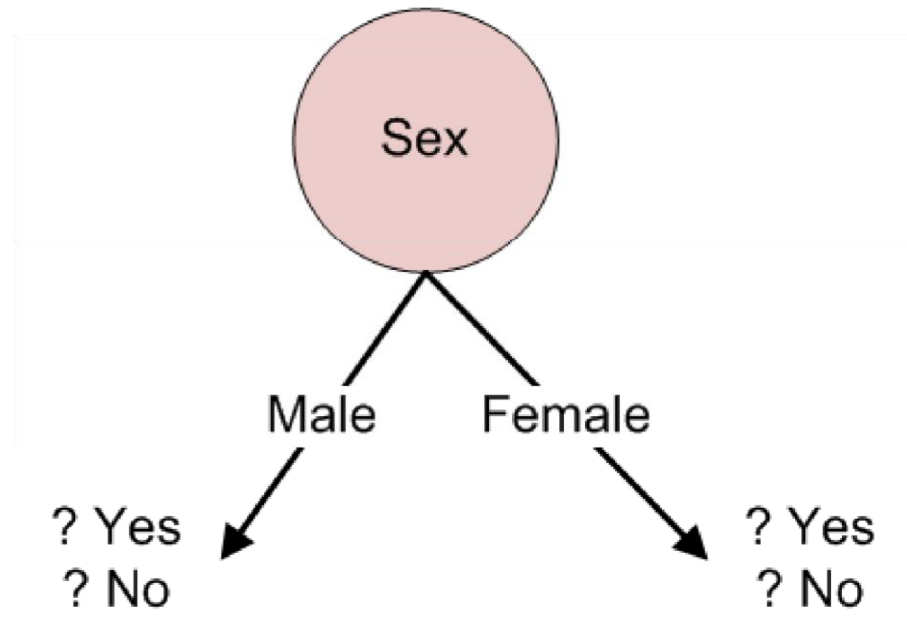
- For this data, a binary split at 43 results in the best “score”.



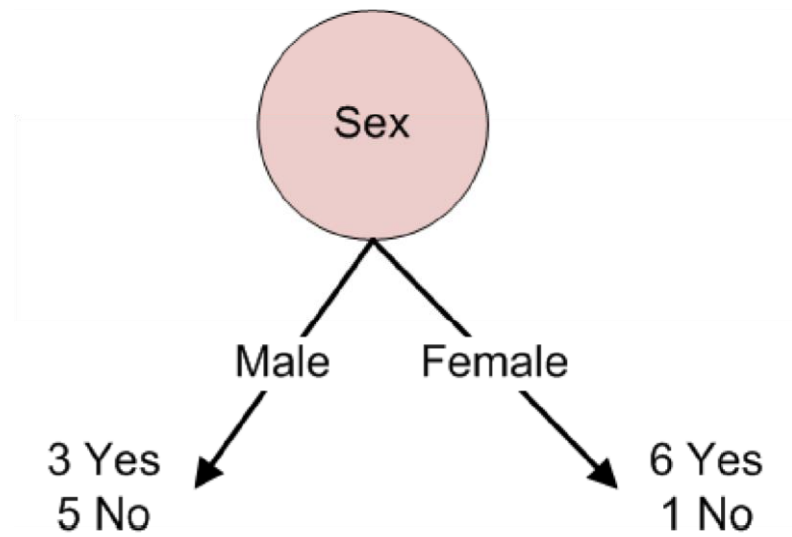
Contd

- ▶ Tree accuracy = $12/15 = 80.0\%$
- ▶ Tree cost = 2 branches for the computer program to use
- ▶ Goodness score for Income Range attribute is $12/15/2 = 0.400$
- ▶ Including Tree “cost” to assess goodness lets us compare trees

Contd

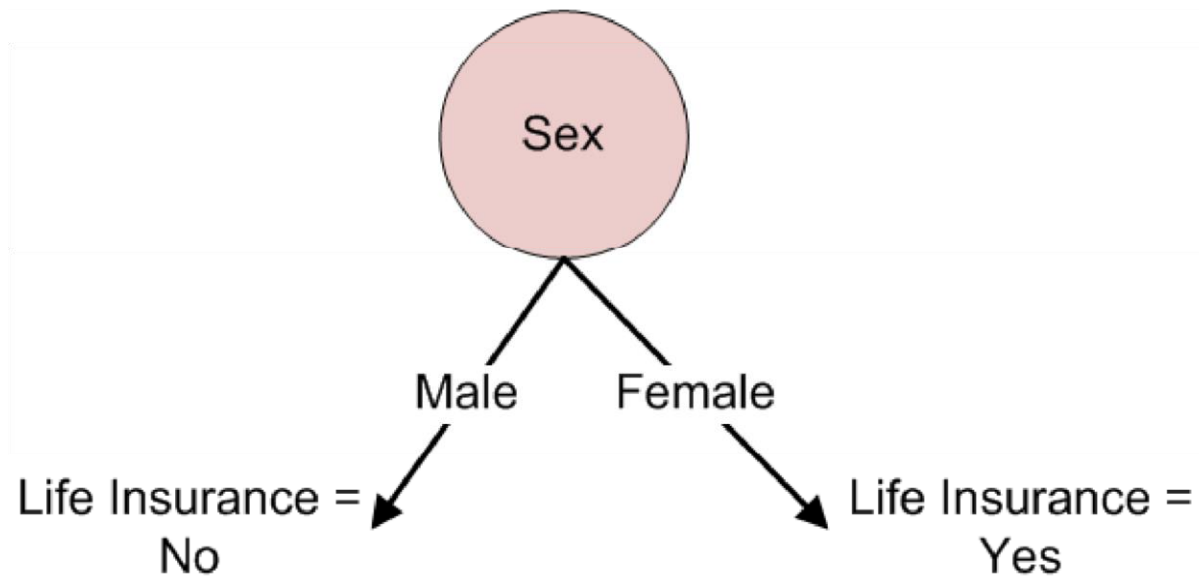


Contd



Contd

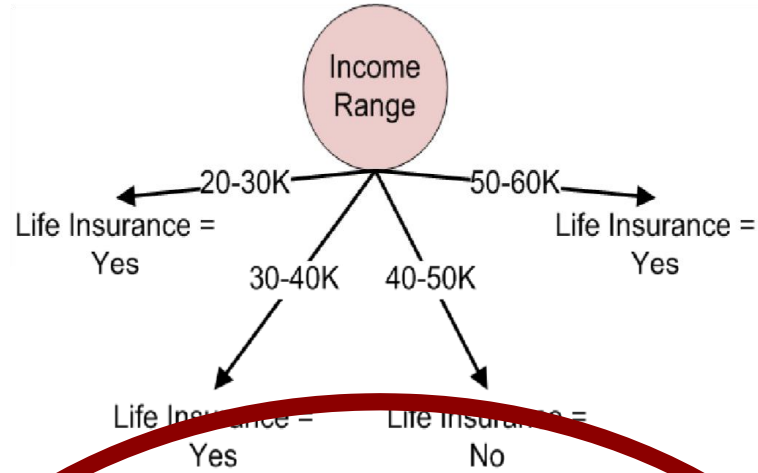
- Evaluate the classification model (the tree). How many of the 15 training instances are classified correctly by this tree?



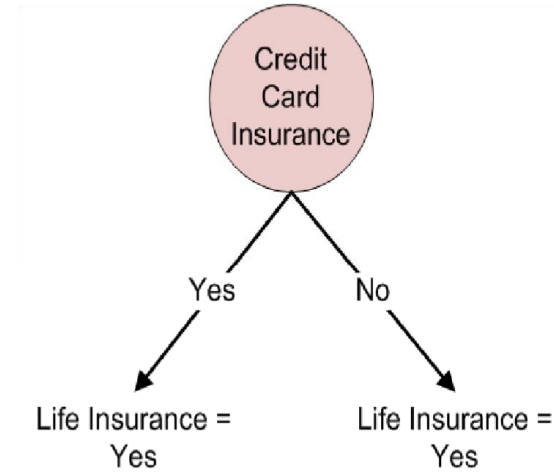
Contd...

- ▶ Tree accuracy = $11/15 = 73.3\%$
- ▶ Tree cost = 2 branches for the computer program to use
- ▶ Goodness score for Income Range attribute is $11/15/2 = 0.367$
- ▶ Including Tree “cost” to assess goodness lets us compare trees

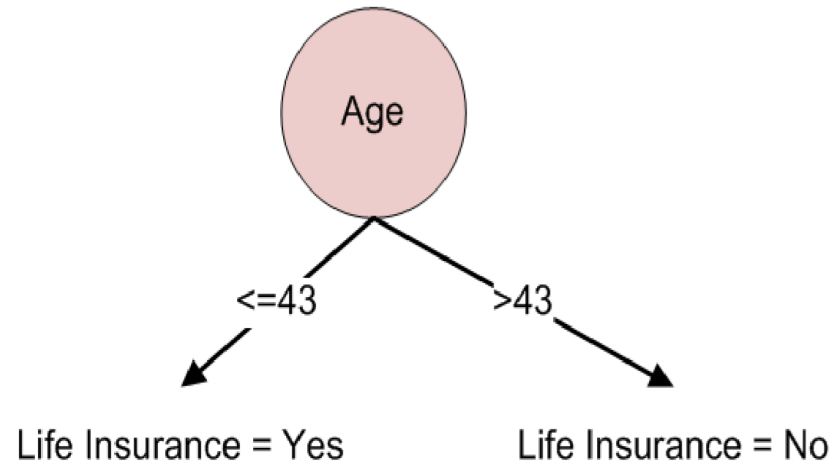
Model “goodness” = 0.183



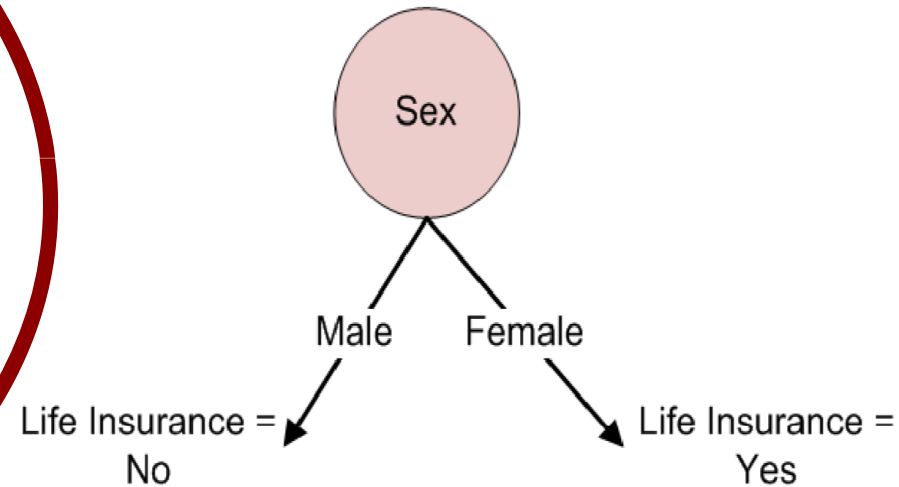
Model “goodness” = 0.30



Model “goodness” = 0.40



Model “goodness” = 0.367

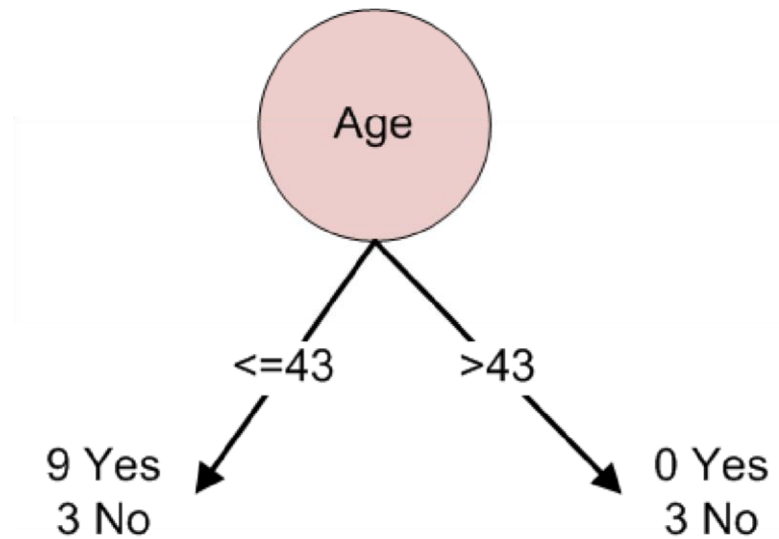


Contd

- ▶ Consider each branch and decide whether to terminate or add an attribute for further classification
- ▶ Different termination criteria make sense
- ▶ If the instances following a branch satisfy a predetermined criterion, such as a certain level of accuracy, then the branch becomes a terminal path
- ▶ No other attribute adds information

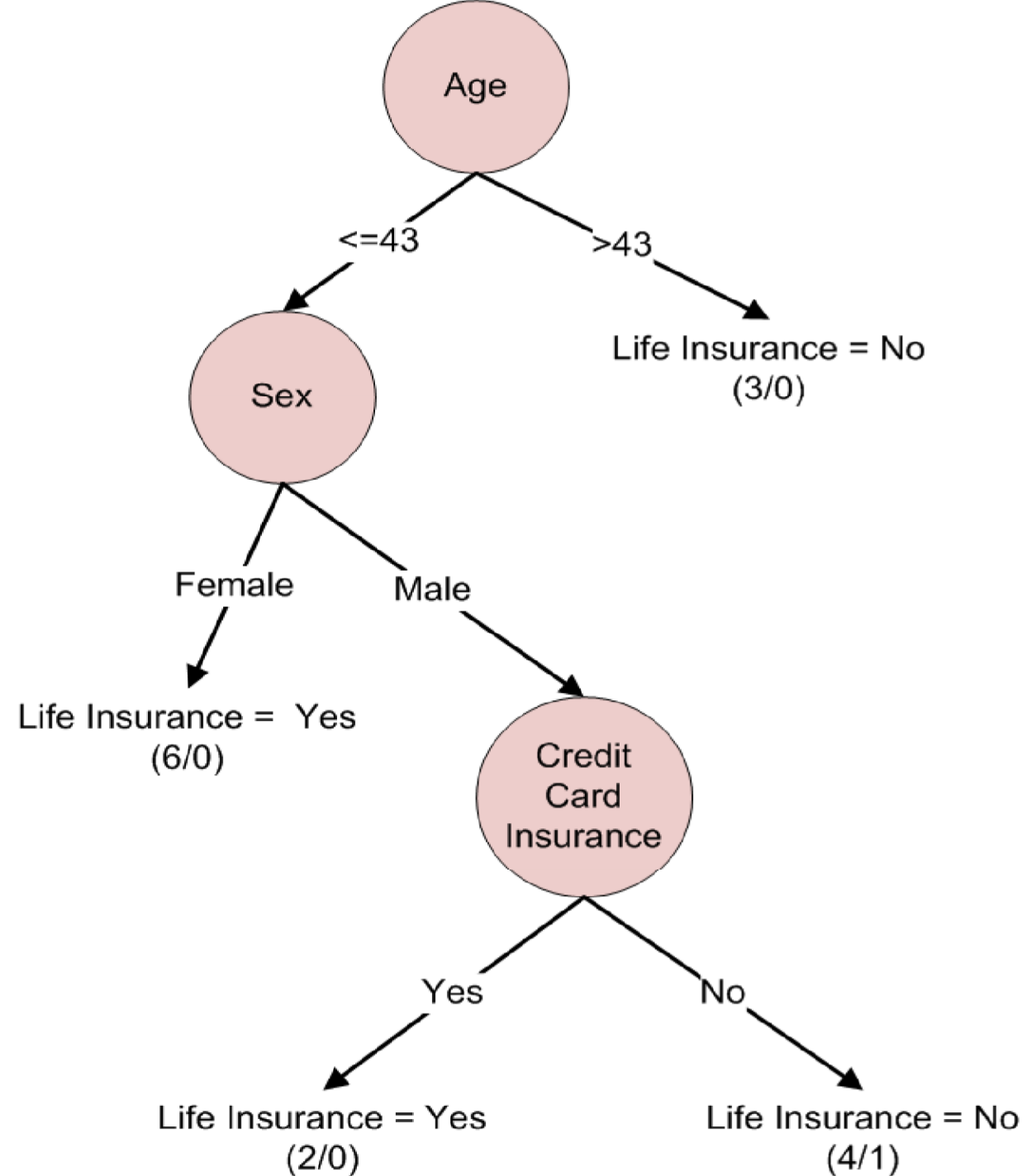
Contd

- 100% accuracy for >43 branch



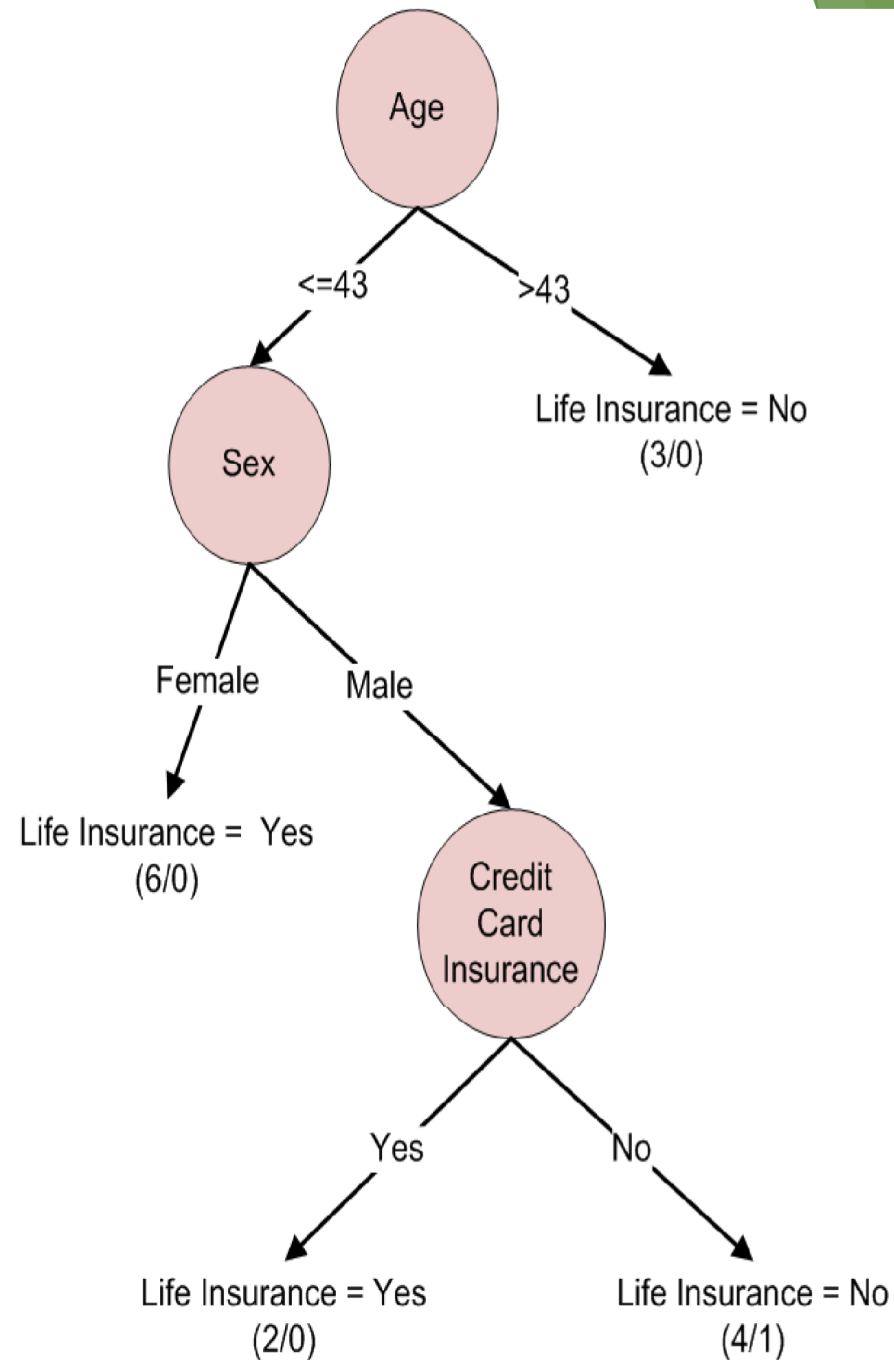
Contd

- Production rules are generated by following to each terminal branch



Contd

► If Age ≤ 43 AND Sex = Male AND CCIns = No
Then Life Insurance
Promo = No
Accuracy = 75%
Coverage = 26.7%



Contd

- Simplify the Rule

If Sex = Male AND CCIns= No

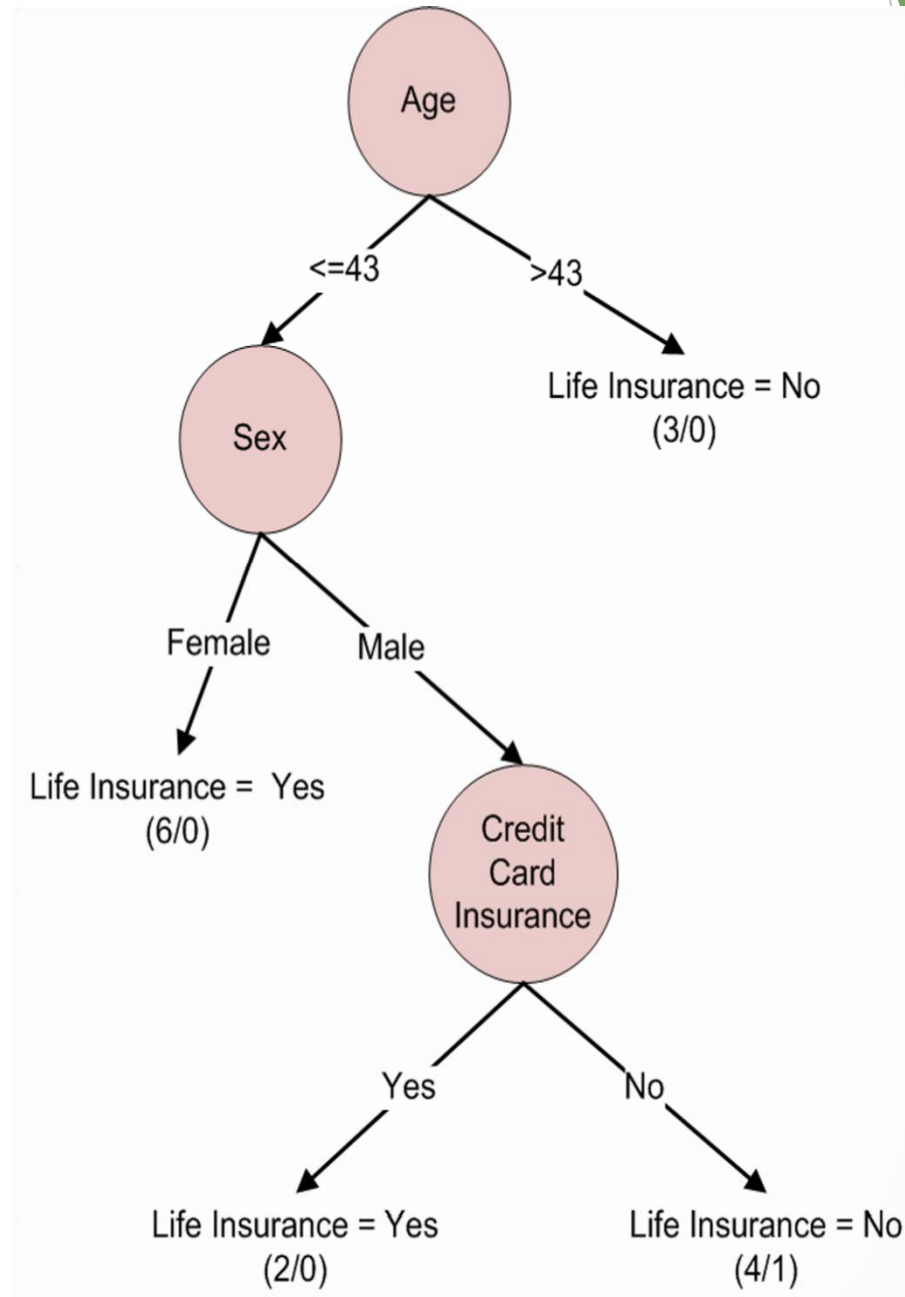
- Then Life Insurance

Promo = No

- Accuracy = 83.3%

- Coverage = 40.0%

- This rule is more general,
more accurate



Contd...

- ▶ `ctree(formula, data, weights, subset, na.action = na.pass, control = ctree_control(...), ytrafo = NULL, scores = NULL, ...)`

Arguments

- ▶ `formula`
a symbolic description of the model to be fit.
- ▶ `data`
a data frame containing the variables in the model.
- ▶ `subset`
an optional vector specifying a subset of observations to be used in the fitting process.
- ▶ `weights`
an optional vector of weights to be used in the fitting process. Only non-negative integer valued weights are allowed.
- ▶ `na.action`
a function which indicates what should happen when the data contain missing value.

Contd..

- ▶ control

a list with control parameters, see `ctree_control`.

- ▶ ytrafo

an optional named list of functions to be applied to the response variable(s) before testing their association with the explanatory variables. Note that this transformation is only performed once for the root node and does not take weights into account. Alternatively, `ytrafo` can be a function of data and weights. In this case, the transformation is computed for every node. This feature is experimental and the user interface likely to change.

- ▶ scores

an optional named list of scores to be attached to ordered factors.

- ▶ ...

arguments passed to `ctree_control`.

How to do in R?

► Go to Code

References

- ▶ http://www.uh.edu/~smiertsc/4397cis/C4.5_Decision_Tree_Algorithm.pdf









