

Adv. Methods

Theoretical methods / Clustering

Dr. Feras Al-Obeidat

What Kind of Problem do I Need to Solve?

How do I Solve it?

The Problem to Solve	The Category of Techniques	Example
I want to group items by similarity. I want to find structure (commonalities) in the data	Clustering	K-means clustering
I want to discover relationships between actions or items	Association Rules	Apriori
I want to determine the relationship between the outcome and the input variables	Regression	Linear Regression Logistic Regression
I want to assign (known) labels to objects	Classification	Naïve Bayes Decision Trees, K-NN

Clustering

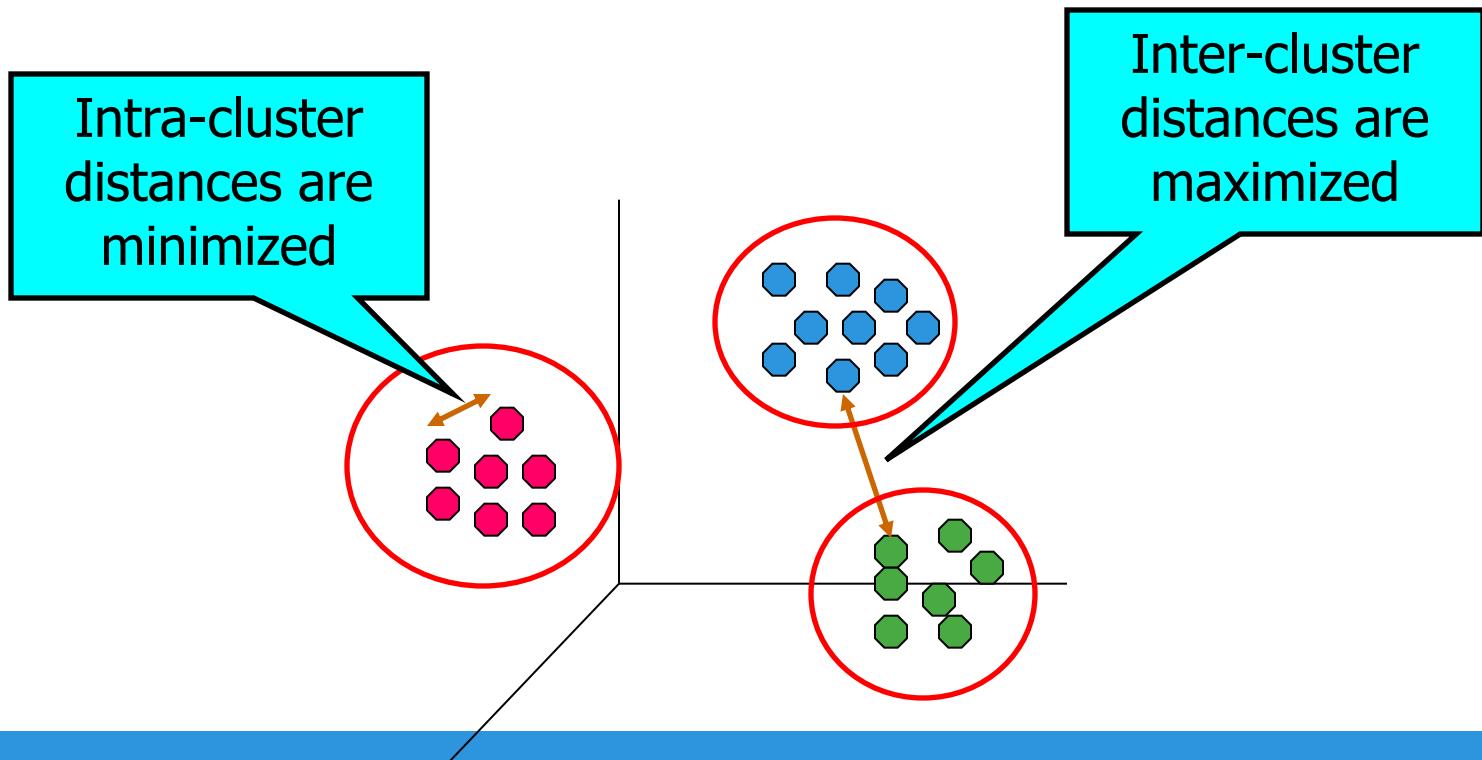
How do I **group** these documents by **topic**?

How do I **group** my customers by **purchase patterns**?

- Sort items into groups by similarity:
 - ▶ Items in a cluster are more similar to each other than they are to items in other clusters.
 - ▶ Need to detail the properties that characterize “similarity”
- Not a **predictive** method; finds similarities, relationships
- Our Example: K-means Clustering

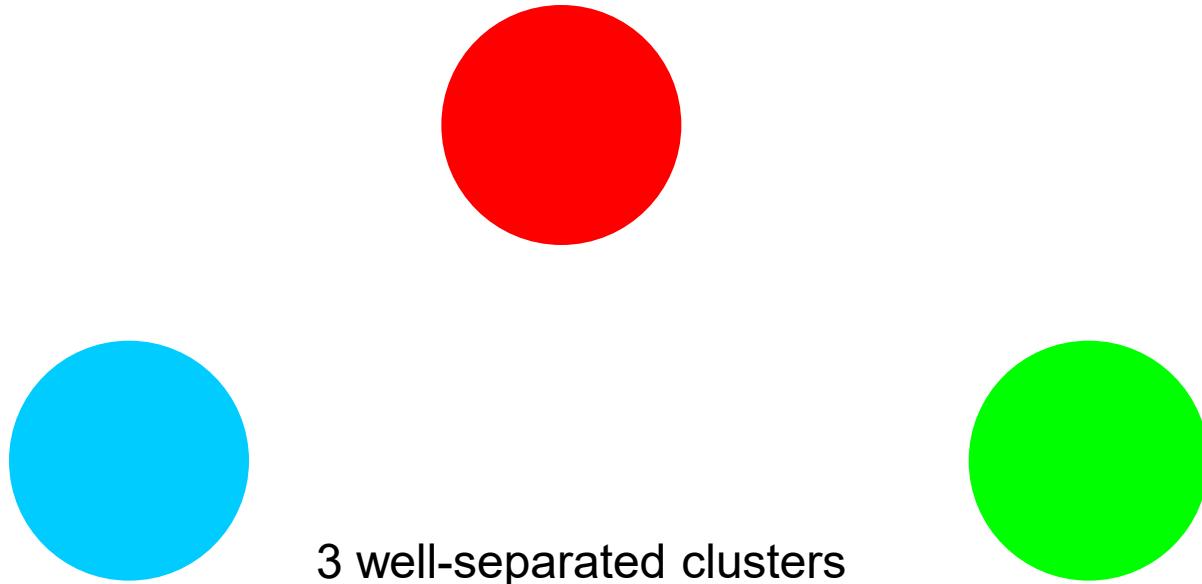
What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



Types of Clusters: Well-Separated

- Well-Separated Clusters:
 - ▶ A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



3 well-separated clusters

Types of Clusters: Center-Based

- Center-based
 - ▶ A cluster is a set of objects such that an object in a **cluster is closer (more similar) to the “center” of a cluster**, than to the center of any other cluster
 - ▶ The center of a cluster is often a **centroid**, the average of all the points in the cluster, or a **medoid**, the most “representative” point of a cluster

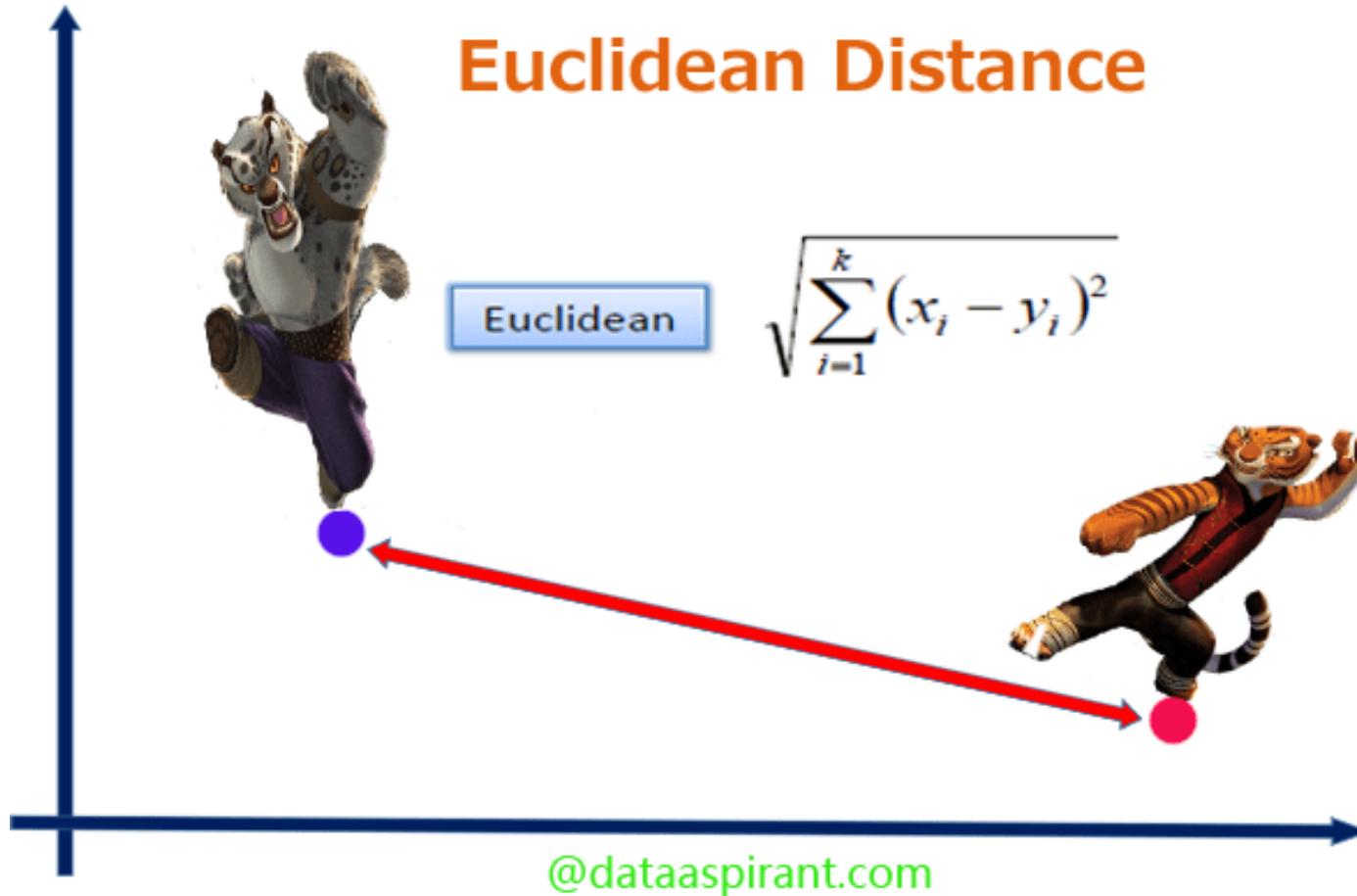


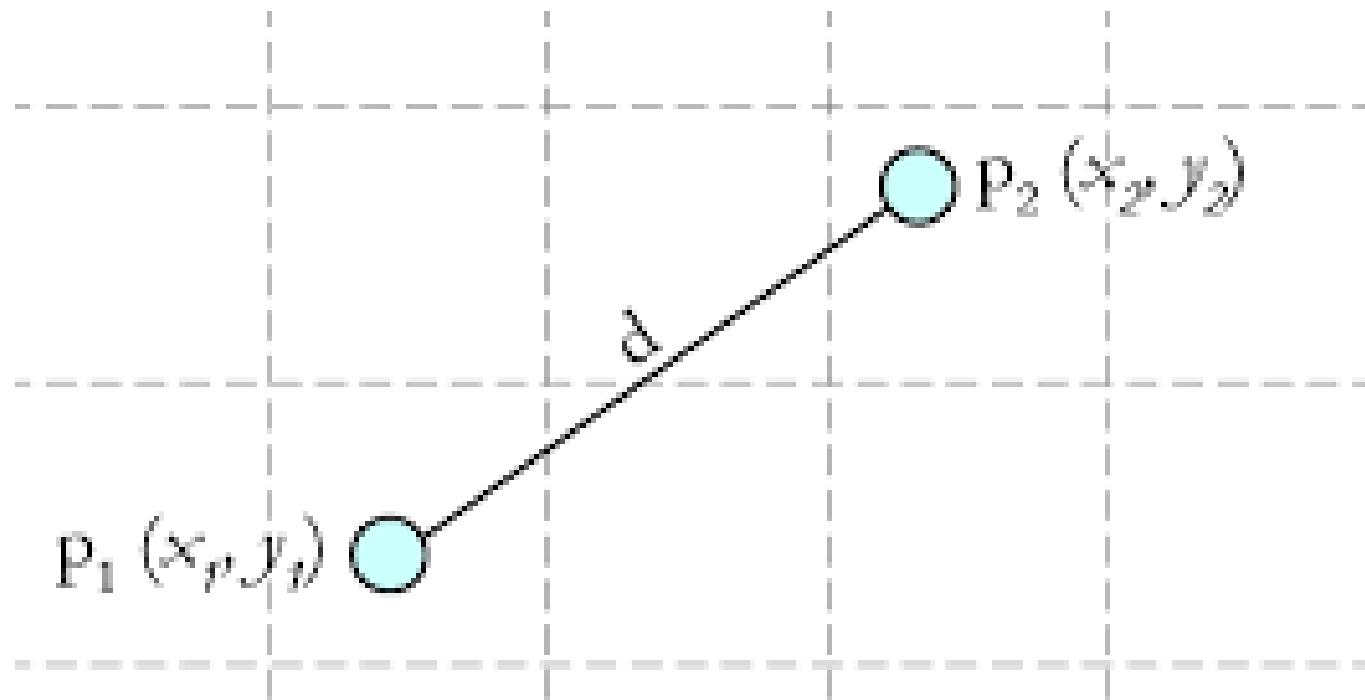
4 center-based clusters

K-Means Clustering - What is it?

- Used for clustering numerical data, usually a set of measurements about objects of interest.
- **Input:** numerical. There must be a distance metric defined over the variable space.
 - ▶ Euclidian distance
- **Output:** The centers of each discovered cluster, and the assignment of each input to a cluster.
 - ▶ Centroid

Euclidean Distance

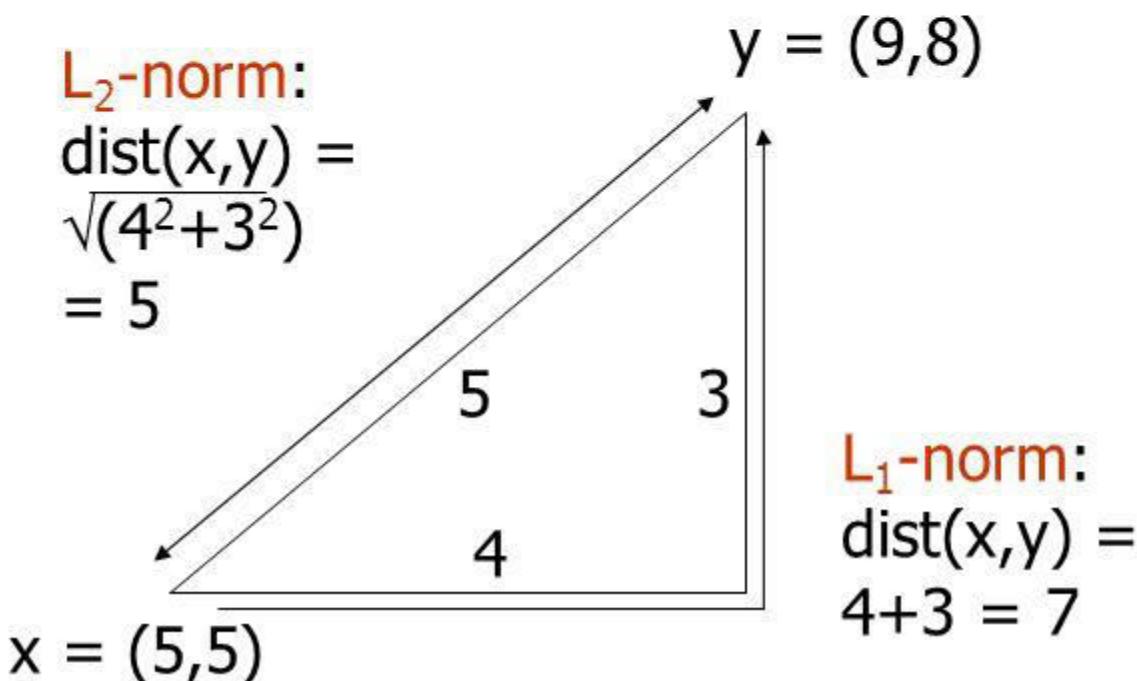




$$\text{Euclidean distance (d)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Some Euclidean Distances

- L_2 norm : $d(x,y) = \text{square root of the sum of the squares of the differences between } x \text{ and } y \text{ in each dimension.}$
 - The most common notion of “distance.”
- L_1 norm : sum of the differences in each dimension.
 - *Manhattan distance* = distance if you had to travel along coordinates only.



K-means Clustering

Characteristics

- Partitional clustering approach
- Each cluster is associated with a **centroid** (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K , must be specified
- The basic algorithm is very simple

Algorithm:

- 1: Select K points as the initial centroids.
 - 2: **repeat**
 - 3: Form K clusters by assigning all points to the closest centroid.
 - 4: Recompute the centroid of each cluster.
 - 5: **until** The centroids don't change
-

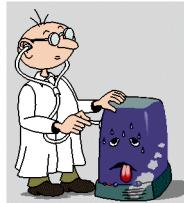
K-means Clustering – Details

- Initial centroids are often chosen randomly.
 - ▶ Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by **Euclidean distance**, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
 - ▶ Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is $O(n * K * I * d)$
 - ▶ n = number of points, K = number of clusters,
 I = number of iterations, d = number of attributes

Use Cases

- Often an exploratory technique:
 - ▶ Discover structure in the data
 - ▶ Summarize the properties of each cluster
- Sometimes a pre-step to classification:
 - ▶ "Discovering the classes"
- Examples
 - ▶ The height, weight and average lifespan of animals
 - ▶ Household income, yearly purchase amount in dollars, number of household members of customer households
 - ▶ Patient record with measures of BMI, HBA1C, HDL

Diagnostics – Evaluating the Model



- Do the clusters look separated in at least some of the plots when you do pair-wise plots of the clusters?
 - ▶ Pair-wise plots can be used when there are not many variables
- Do you have any clusters with few data points?
 - ▶ Try decreasing the value of K
- Are there splits on variables that you would expect, but don't see?
 - ▶ Try increasing the value K
- Do any of the centroids seem too close to each other?
 - ▶ Try decreasing the value of K

K-Means Clustering - Reasons to Choose (+) and Cautions (-)



Reasons to Choose (+)	Cautions (-)
Easy to implement	Doesn't handle categorical variables
Easy to assign new data to existing clusters Which is the nearest cluster center?	Sensitive to initialization (first guess)
Concise output Coordinates the K cluster centers	K (the number of clusters) must be known or decided first Wrong guess: possibly poor results

K-Means case study

- This section shows k-means clustering of iris data.
- First, we remove species from the data to cluster. After that, we apply function **kmeans()** to iris2, and store the clustering result in kmeans.result.

The cluster number is set to **3** in the code below.

```
iris2 <- iris  
iris2  
iris2$Species <- NULL  
iris2  
kmeans.result <- kmeans(iris2, 3)
```

K-Means case study, comparison with real class

- The clustering result is then compared with the class label (Species) to check whether similar objects are grouped together.

```
table(iris$Species, kmeans.result$cluster)
```

	1	2	3
setosa	0	50	0
versicolor	2	0	48
virginica	36	0	14

- The above result shows that cluster \setosa" can be easily separated from the other clusters, and
- that clusters \versicolor" and \virginica" are to a small degree overlapped with each other.

K-Means case study, Plotting the results

- Next, the clusters and their centers are plotted
- Note that there are four dimensions in the data and that only the first two dimensions are used to draw the plot below.
- Some black points close to the green center (asterisk) are actually closer to the black center in the four dimensional space.
- We also need to be aware that the results of k-means clustering may vary from run to run, due to random selection of initial cluster centers.
- `>plot(iris2[c("Sepal.Length", "Sepal.Width")], col = kmeans.result$cluster)`
- # plot cluster centers
- `> points(kmeans.result$centers[,c("Sepal.Length", "Sepal.Width")], col = 1:3, pch = 8, cex=2)`

```
points(kmeans.result$centers[,c("Sepal.Length", "Sepal.Width")], col = 1:3, pch = 8,  
cex=2)
```

Pch, cex for shapes

- pch=0,square

pch=1,circle

pch=2,triangle point up

pch=3,plus

pch=4,cross

pch=5,diamond

pch=6,triangle point down

pch=7,square cross

pch=8,star

pch=9,diamond plus

pch=10,circle plus

pch=11,triangles up and down

pch=12,square plus

pch=13,circle

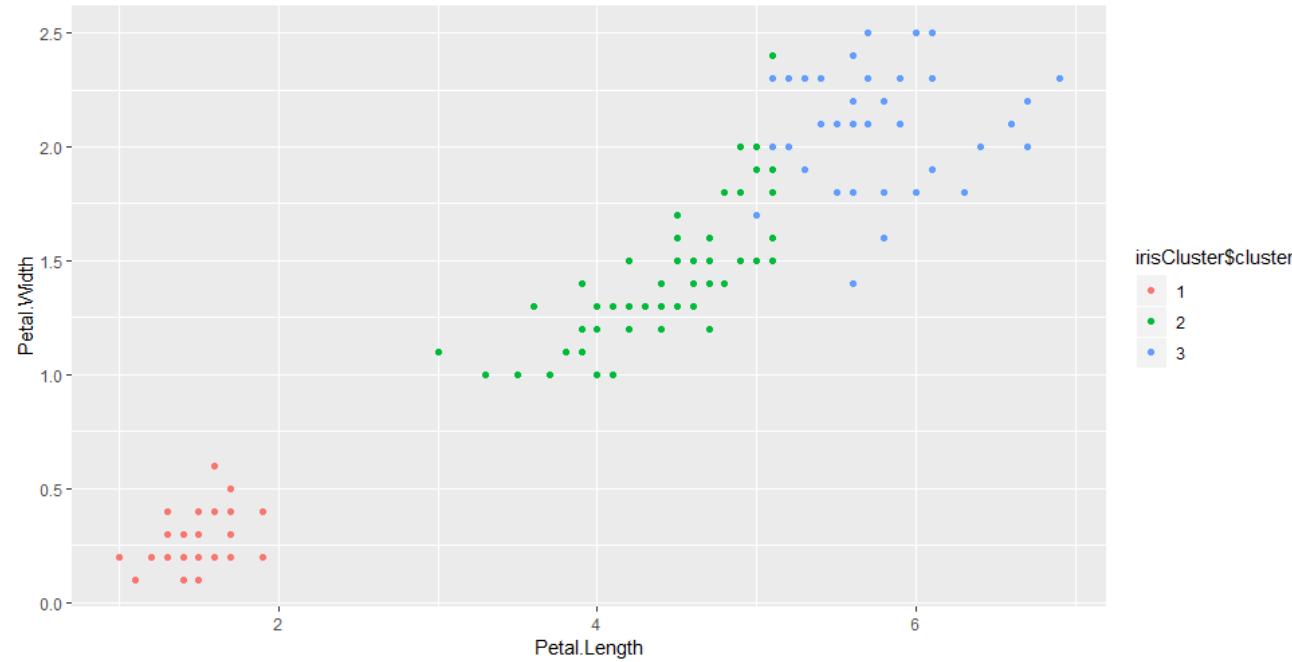
-

cex controls the symbol **size** in the plot,
default is cex=1,

col controls the **color** of the symbol border,
default is col="black".

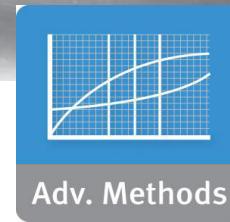
Show result in 3 dim

```
library(ggplot2)
ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
irisCluster <- kmeans(iris[, 1:4], 3)
irisCluster
table(iris$Species, irisCluster$cluster)
irisCluster$cluster <- as.factor(irisCluster$cluster)
ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()
```



geom_point()

The point geom is used to create scatterplots. The scatterplot is most useful for displaying the relationship between two continuous variables. It can be used to compare one continuous and one categorical variable, or two categorical variables



6– Advanced Analytics - Classification



Introduction



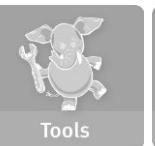
Analytics Lifecycle



Basic Methods



Adv. Methods



Tools



Lab

Data classification

Classification (supervised)

- **Decision Trees**
- Naïve Bayesian classifier
- Time Series Analysis
- Text Analysis

Classifiers

Where in the shelf should I place this item?

Is this email spam?

Is this politician Democrat/Republican/Green?

- Classification: assign labels to objects.
- Usually supervised: training set of pre-classified examples.
- Our examples:
 - ▶ Naïve Bayesian
 - ▶ **Decision Trees**
 - ▶ (and Logistic Regression)



Decision Tree Classifier - What is it?

- Used for classification:
 - ▶ Returns probability scores of class membership
 - ▶ Assigns label based on highest scoring class
- Input variables can be continuous or discrete
- Output:
 - ▶ A tree that describes the decision flow.
 - ▶ Leaf nodes return either a probability score, or simply a classification.
 - ▶ Trees can be converted to a set of "decision rules"
 - ▶ "IF income < \$50,000 AND mortgage_amt > \$100K THEN default=T with 75% probability"

Decision Tree Classifier - Use Cases

- When a series of questions (yes/no) are answered to arrive at a classification
 - ▶ Biological species classification
 - ▶ Checklist of symptoms during a doctor's evaluation of a patient
 - ▶ Financial decisions such as loan approval
 - ▶ Fraud detection

Step 1: Pick the Most “Informative” Attribute

- Entropy-based methods are one common way

$$H = - \sum_c p(c) \log_2 p(c)$$

- $H = 0$ if $p(c) = 0$ or 1 for any class
 - ▶ So for binary classification, $H=0$ is a "pure" node
- H is maximum when all classes are equally probable
 - ▶ For binary classification, $H=1$ when classes are 50/50

Step 1: Pick the Most "Informative" Attribute (Continued) Information Gain

$$\text{InfoGain}_{attr} = H - H_{attr}$$

- The information that you gain, by knowing the value of an attribute
- So the "most informative" attribute is the attribute with the highest InfoGain
- Information gain is the amount of information gained by knowing the value of the attribute
- **Information gain=**
(Entropy of distribution before the split)–(entropy of distribution after it)
- Information gain is the amount of information that's gained by knowing the value of the attribute, which is the entropy of the distribution before the split minus the entropy of the distribution after it. The largest information gain is equivalent to the smallest entropy.

Weather data

outlook	temperature	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

Steps to calculate the highest information gain on a data set

Step 1: Calculate Entropy for whole dataset:

Entropy of the whole data set

14 records, 9 are “yes” 5 are “no”

$$-\left(\frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14}\right) \approx 0.94$$

Step 2: Calculate Entropy for each attribute:

Attribute1 - outlook

The outlook attribute contains 3 distinct values:

- overcast: 4 records, 4 are “yes”
$$-\left(\frac{4}{4} \log_2 \frac{4}{4}\right) = 0$$
- rainy: 5 records, 3 are “yes”
$$-\left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5}\right) \approx 0.97$$
- sunny: 5 records, 2 are “yes”
$$-\left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5}\right) \approx 0.97$$

Expected new entropy:

$$-\left(\frac{4}{14} \times 0 + \frac{5}{14} \times 0.97 + \frac{5}{14} \times 0.97\right) \approx 0.69$$

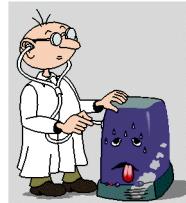
Step 3: Calculate Information gain for each attribute:

Attribute1 - outlook

Gain

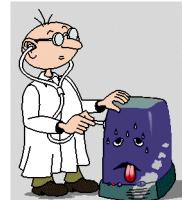
Attribute	Information Gain
outlook	$0.94 - 0.69 =$ 0.25
temperature	$0.94 - 0.91 =$ 0.03
humidity	$0.94 - 0.72 =$ 0.22
windy	$0.94 - 0.87 =$ 0.07

Diagnostics



- Hold-out data
- Confusion Matrix
- FPR/FNR, Precision/Recall
- Do the splits (or the "rules") make sense?
 - ▶ What does the domain expert say?
- How deep is the tree?
 - ▶ Too many layers are prone to over-fit
- Do you get nodes with very few members?
 - ▶ Over-fit

Diagnostics: Confusion Matrix



true positives (TP)
671 correctly
classified as good

false positives (FP)
38 incorrectly
classified as good

		Prediction		
		good	bad	
Actual Class	good	671	29	700
	bad	38	262	300
		709	291	1000

false negatives (FN)
29 incorrectly
classified as bad

Overall success rate (or accuracy):

$$(TP + TN) / (TP+TN+FP+FN) = (671+262)/1000 \approx 0.93$$

TPR: $TP / (TP + FN) = 671 / (671+29) = 671/700 \approx 0.96$

FPR: $FP / (FP + TN) = 38 / (38 + 262) = 38/300 \approx 0.13$

FNR: $FN / (TP + FN) = 29 / (671 + 29) = 29/700 \approx 0.04$

Precision: $TP / (TP + FP) = 671/709 \approx 0.95$

Recall (or TPR): $TP / (TP + FN) \approx 0.96$

true negatives
(TN)
262 correctly
classified as
bad

Decision Tree Classifier - Reasons to Choose (+) & Cautions (-)



Reasons to Choose (+)	Cautions (-)
Takes any input type (numeric, categorical) In principle, can handle categorical variables with many distinct values (ZIP code)	Tree structure is sensitive to small changes in the training data
Computationally efficient to build	In practice, decision rules can be fairly complex
Many algorithms can return a measure of variable importance	
In principle, decision rules are easy to understand	

Which Classifier Should I Try?



Typical Questions	Recommended Method
Do I want class probabilities, rather than just class labels?	Logistic regression Decision Tree
Do I want insight into how the variables affect the model?	Logistic regression Decision Tree
Is the problem high-dimensional?	Naïve Bayes
Do I suspect some of the inputs are correlated?	Decision Tree Logistic Regression
Do I suspect some of the inputs are irrelevant?	Decision Tree Naïve Bayes
Are there categorical variables with a large number of levels?	Naïve Bayes Decision Tree
Are there mixed variable types?	Decision Tree Logistic Regression
Is there non-linear data or discontinuities in the inputs that will affect the outputs?	Decision Tree

R: Decision Trees with Package party

- This section shows how to build a decision tree for the iris data with function **ctree()** in package **party**.
- We will use **iris** data for classification
- Sepal.Width, Petal.Length and Petal.Width are used to predict the **Species** of flowers.
- In the package, function **ctree()** builds a decision tree, and **predict()** makes prediction for new data.

R: Decision Trees: ctree()

- We load package **party**, build a decision tree, and check the prediction result.
- Function **ctree()**
- Below we use default settings to build a decision tree. In the code below, **myFormula** species that **Species** is the target variable and all other variables are independent variables.

```
# install.packages("party")
library(party)
myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
iris_ctree <- ctree(myFormula, data=trainData)
# check the prediction
table(predict(iris_ctree), trainData$Species)
```

R: Predict testing data

```
# predict on test data  
testPred <- predict(iris_ctree, newdata = testData)  
table(testPred, testData$Species)
```

Results:

- testPred setosa versicolor virginica

setosa	10	0	0
versicolor	0	12	2
virginica	0	0	14

R: Decision Trees: ctree(), Results on training data

```
# finding most important factors
```

```
library('FSelector')
```

```
res <- gain.ratio(Species~.,iris)
```

```
res
```



Adv. Methods

Chapter 7: Regression

Dr. Feras Al-Obeidat

Regression

- Regression focuses on the relationship between an **outcome** and its **input** variables.
 - ▶ Provides an estimate of the outcome based on the input values.
 - ▶ Models how changes in the input variables affect the outcome.
- The outcome can be **continuous** or **discrete**.
- Possible use cases:
 - ▶ Estimate the lifetime value (LTV) of a customer and understand what influences LTV.
 - ▶ Estimate the probability that a loan will default and understand what leads to default.
- **approaches: linear regression and logistic regression**

Linear Regression

- Used to estimate a continuous value as a linear function of other variables
 - ▶ **Income** as a function of years of education, age, and gender
 - ▶ **House sales price** as function of area, number of bedrooms/bathrooms, and lot size
- **Outcome** variable is continuous.
- **Input** variables can be continuous or discrete.
- **Model Output:**
 - ▶ A set of estimated **coefficients** that indicate the relative impact of each input variable on the outcome
 - ▶ A linear expression for estimating the outcome as a function of input variables

Linear Regression Model

Linear regression is to predict **response** with a linear function of **predictors** as follows:

$$y = c_0 + c_1x_1 + c_2x_2 + \cdots + c_kx_k,$$

where x_1, x_2, \dots, x_k are predictors and y is the response to predict.

First Example

Lets use R to Plot these values:

```
# Regression
```

```
Java1 <- c(95, 85, 80, 70, 60)
```

```
Java2 <- c(85, 95, 70, 65, 70)
```

```
plot(Java2, Java1)
```

First Example, building the model

we use `lm()` to build a linear regression model in R

fit <- lm(Java2 ~ Java1)

fit

First Example building the model output

Call: lm(formula = Java2 ~ Java1)

Coefficients:

Intercept: 26.7808 (This is c0)

Java1: 0.6438 (This is c1)

According to the equation:

$$Y = c0 + c1 * x1$$

$$\text{Java2} = 26.7808 + 0.6438 * \text{Java1}$$

First Example, use R for prediction

Predict what score a student can get in Java2 if she got 80 in Java1

We can do it in R:

```
(java2_ <- fit$coefficients[[1]] + fit$coefficients[[2]]*80)
```

Is it **78.28** ?

Second Example (more predictors, code in R) (1)

1. read marks

```
Java1 <- c(95, 85, 80, 70, 60, 80, 75, 90, 88) # 9 students
```

```
Java2 <- c(85, 95, 70, 65, 70, 78, 72, 88, 91)
```

```
swEng <- c(82, 89, 70, 72, 75, 77, 75, 89, 89)
```

#2. plot

```
library(scatterplot3d)
```

```
scatterplot3d(Java1, # x axis
```

```
        Java2, # y axis
```

```
        swEng, # z axis
```

```
        main="3-D Scatterplot Example 1")
```

Continue ...

Second Example (more predictors, code in R) (2)

```
# 3. predict the mark in sofware engineering based on the marks in  
java1 and java2
```

```
fit2 <- lm(swEng ~ Java1 + Java2)
```

```
fit2
```

```
# 4. visualize the model
```

```
library(visreg)
```

```
visreg(fit2)
```

Regression example

- Linear regression is demonstrated below with function lm() on the Australian CPI (**Consumer Price Index**) data, which are quarterly CPIs from 2008 to 2010
- At first, the data is created and plotted.
- In the code below, an x-axis is added manually with function axis(), where las=3 makes text vertical.

A **consumer price index (CPI)** measures changes in the **price** level of a market basket of **consumer** goods and services purchased by households.

Initialize variables

```
# each year 4 quarters  
# rep() function replicates values  
year <- rep(2008:2010, each=4)  
quarter <- rep(1:4, 3) # 3 years / 4 Q  
  
cpi <- c(162.2, 164.6, 166.5, 166.0, # 2008, q1,q2,q3,q4  
      + 166.2, 167.0, 168.6, 169.5, # 2009, q1,q2,q3,q4  
      + 171.0, 172.1, 173.3, 174.0) # 2010, q1,q2,q3,q4
```

Plot data

```
plot(cpi, xaxt="n", ylab="CPI", xlab="")
```

To further explain the output, you can add:

```
# draw x-axis
```

```
axis(1, labels=paste(year,quarter,sep="Q"), at=1:12, las=3)
```

See next slide for the output

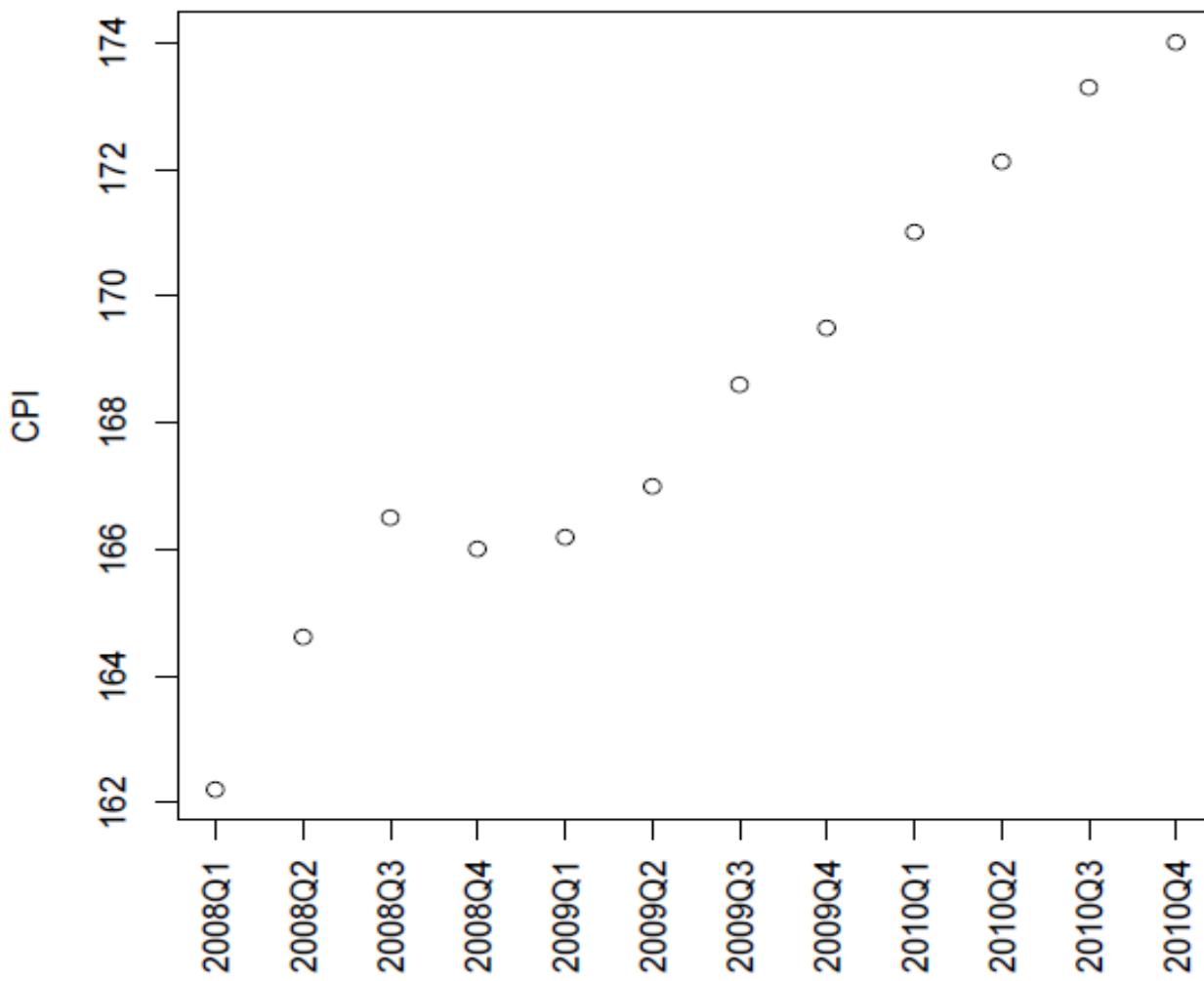
las

labels are parallel (=0) or
perpendicular(=2) to axis

Axes function

option	description
side	an integer indicating the side of the graph to draw the axis (1=bottom, 2=left, 3=top, 4=right)
at	a numeric vector indicating where tic marks should be drawn
labels	a character vector of labels to be placed at the tickmarks (if NULL, the <i>at</i> values will be used)
pos	the coordinate at which the axis line is to be drawn. (i.e., the value on the other axis where it crosses)
lty	line type
col	the line and tick mark color
las	labels are parallel (=0) or perpendicular(=2) to axis
tck	length of tick mark as fraction of plotting region (negative number is outside graph, positive number is inside, 0 suppresses ticks, 1 creates gridlines) default is -0.01

Graphic representation



Check correlation CPI with year and quarter

- We then check the correlation between CPI and the other variables, year and quarter.

```
cor(year,cpi)
```

```
[1] 0.9096316
```

```
cor(quarter,cpi)
```

```
[1] 0.3738028
```

Building the Linear Regression Model

- Then a linear regression model is built with function **lm()** on the above data:
 - ▶ Predictors: `year` and `quarter`
 - ▶ Response: CPI

```
fit <- lm(cpi ~ year + quarter)
```

```
fit
```

The model

- With the generated linear model in the previous slide, CPI is calculated as:

$$\text{cpi} = c0 + c1 * \text{year} + c2 * \text{quarter}$$

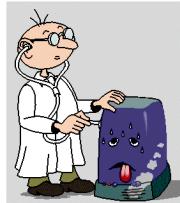
- where $c0$, $c1$ and $c2$ are coefficients from model fit. Therefore, the CPIs in 2011 can be get as follows.

```
(cpi2011 <- fit$coefficients[[1]] + fit$coefficients[[2]]*2011 +  
+ fit$coefficients[[3]]*(1:4))
```

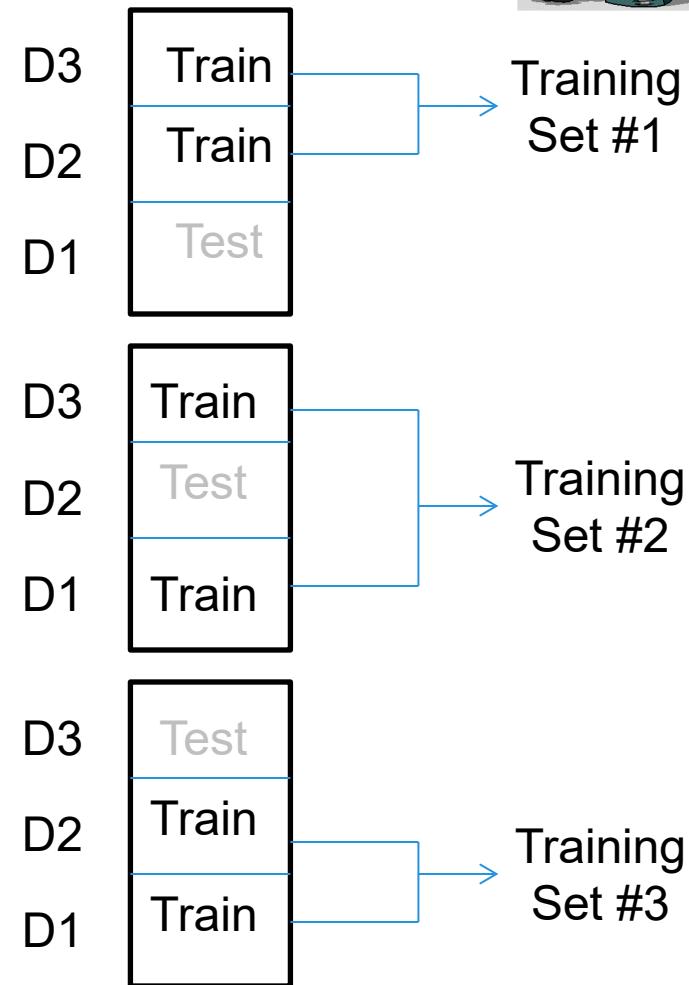
Predict new inputs

```
data2011 <- data.frame(year=2011, quarter=1:4)
cpi2011 <- predict(fit, newdata=data2011)
style <- c(rep(1,12), rep(2,4))
plot(c(cpi, cpi2011), xaxt="n", ylab="CPI", xlab="", pch=style,
col=style)
axis(1, at=1:16, las=3, labels=c(paste(year,quarter,sep="Q"),
"2011Q1", "2011Q2", "2011Q3", "2011Q4"))
```

Diagnostics – Using Hold-out Data



- Hold-out data
 - ▶ Training and testing datasets
 - ▶ Does the model predict well on data it hasn't seen?
- N-fold cross validation
 - ▶ Partition the data into N groups.
 - ▶ Holding out each group,
 - ▶ Fit the model
 - ▶ Calculate the residuals on the group
 - ▶ Estimated prediction error is the average over all the residuals.



Linear Regression - Reasons to Choose (+) and Cautions (-)



Reasons to Choose (+)	Cautions (-)
Concise representation (the coefficients)	Does not handle missing values well
Explanatory value Relative impact of each variable on the outcome	Does not work well with categorical attributes with a lot of distinct values For example, ZIP code
Easy to score data	

Random Forest

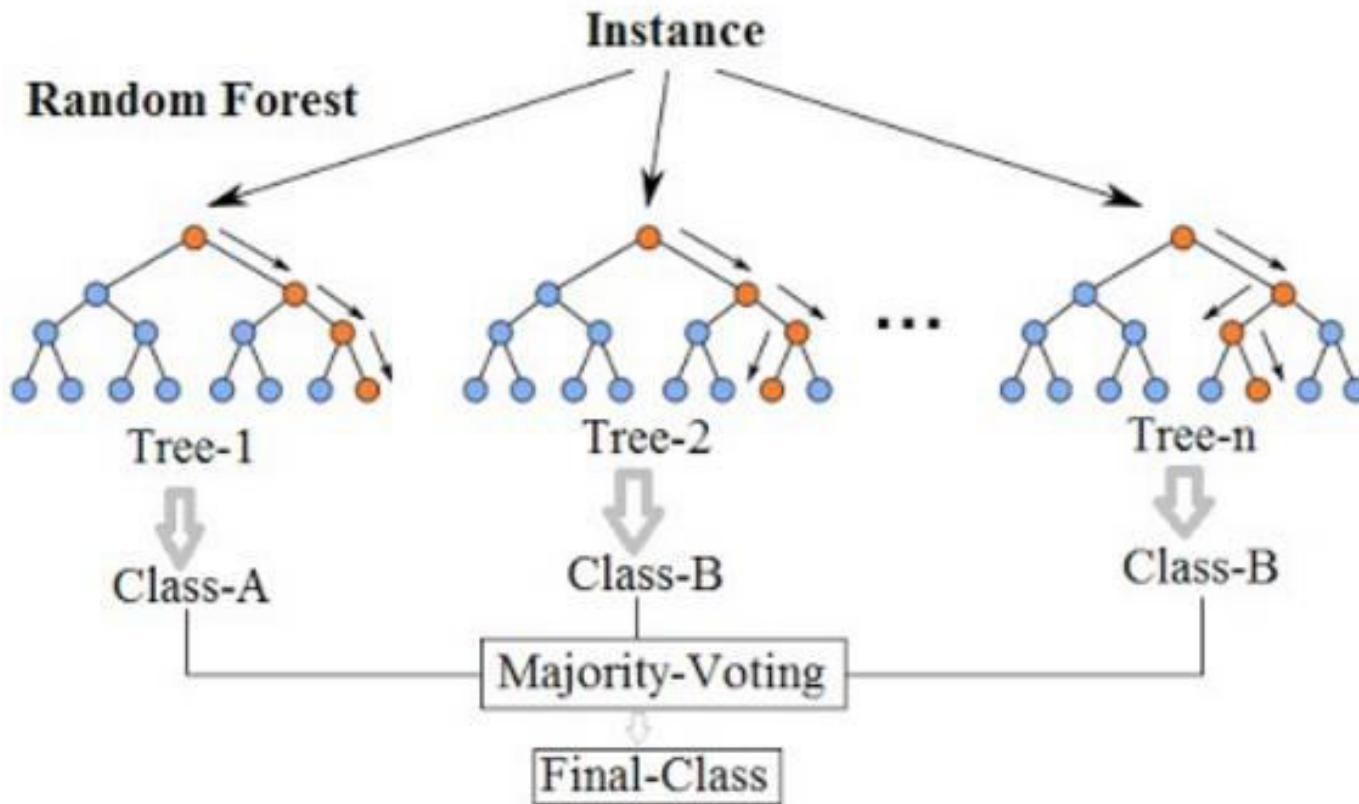
- **Random forests (RF)** or **random decision forests** are an **ensemble learning** method for classification, regression and other tasks that operate by **constructing a multitude of decision trees at training time**.
- RF is a popular machine learning algorithm (ML) that belongs to the **supervised learning** technique.
- RF can be used for both **Classification** and **Regression** problems in ML. It is based on the concept of **ensemble learning**, which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model*.

Random Forest

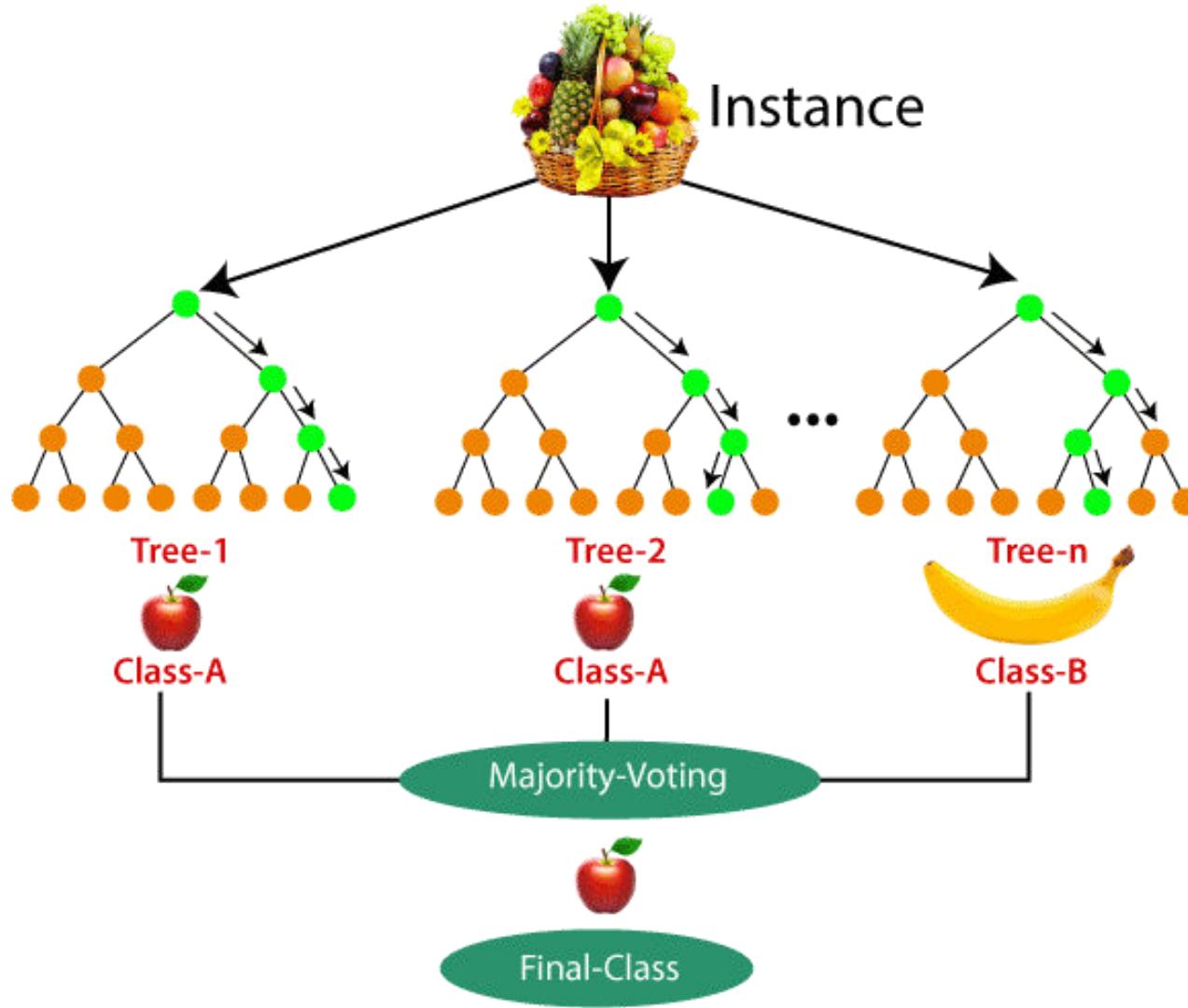
- The **output** is the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees
- Random forests generally **outperform** decision trees
- Forests are like the pulling together of decision tree algorithm efforts.
- Taking the teamwork of many trees thus **improving the performance** of a single random tree. Though not quite similar, forests give the effects of a K-fold cross validation.
- **Random forests** can be used to **rank the importance of variables** in a regression or classification problem in a natural way.

Random forest

Random Forest Simplified



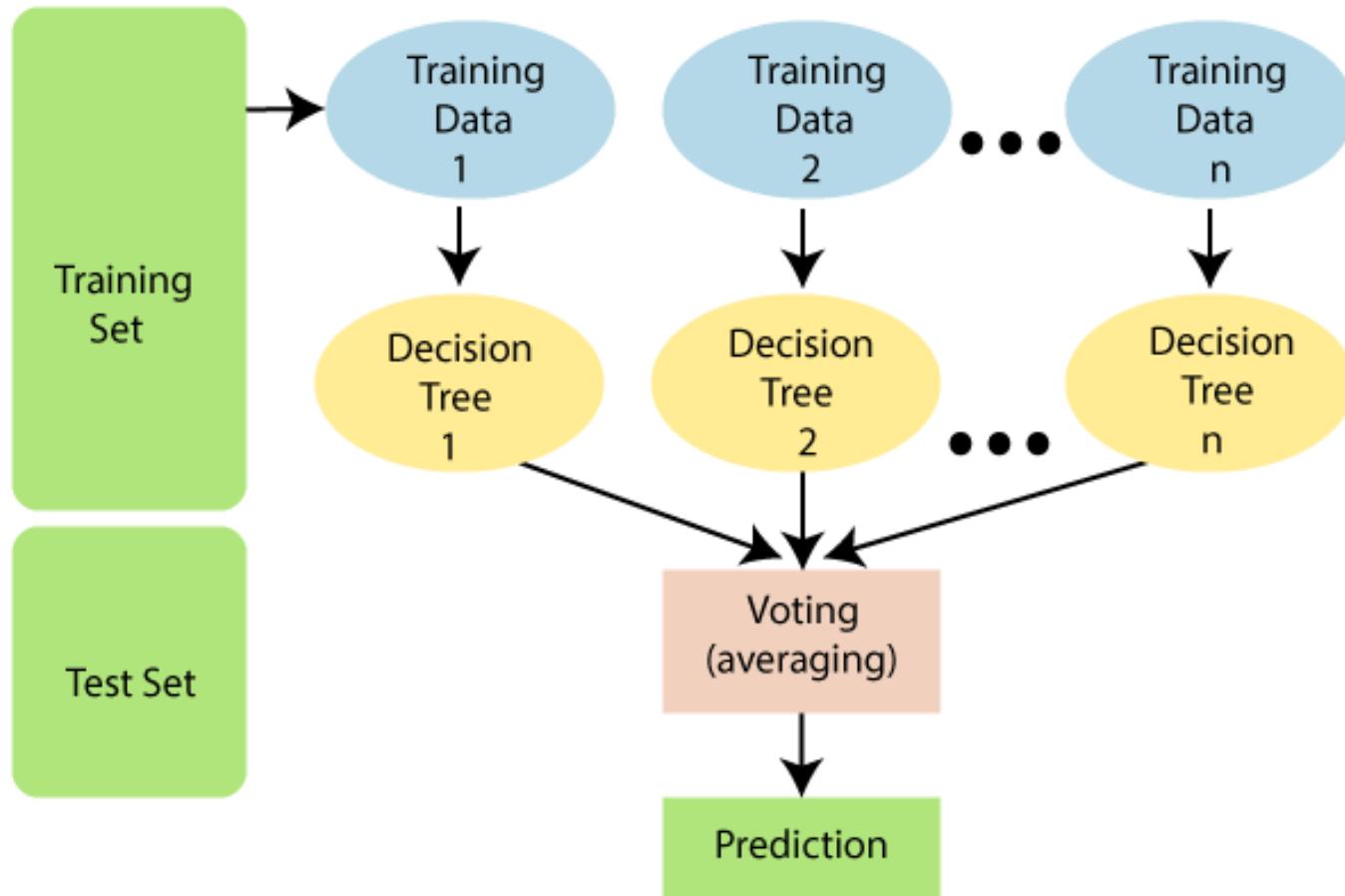
Random forest examples



RF: How it works

- **Step-1:** Select random K data points from the training set.
- **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- **Step-3:** Choose the number N for decision trees that you want to build.
- **Step-4:** Repeat Step 1 & 2.
- **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Random Forest Methodology



Why Random Forest

- Good **performance**
- It takes less training time as compared to other algorithms.
- It predicts output with **high accuracy**, even for the large dataset it runs efficiently.
- It can also **maintain accuracy** when a large proportion of data is missing.

Application of RF in R

We will use for this example: processed.cleveland.data

```
library(randomForest)
require(caTools)
#upload the file
setwd("C:\\\\Users\\\\z10095\\\\Desktop\\\\courseFiles\\\\Spring2021\\\\477\\\\")
# https://archive.ics.uci.edu/ml/machine-learning-databases/heart-
disease/
# read the file
data <- read.csv("processed.cleveland.data",header=FALSE)
head(data)
```

Data description

#cp:chest pain type

trestbps: resting blood pressure (in mm Hg on admission to the hospital)

chol: serum cholestorol in mg/dl

fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

#19 restecg: resting electrocardiographic results

-- Value 0: normal

-- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

-- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

#32 thalach: maximum heart rate achieved

exang: exercise induced angina (1 = yes; 0 = no)

Data description

#oldpeak = ST depression induced by exercise relative to rest

#41 slope: the slope of the peak exercise ST segment

#-- Value 1: upsloping

#-- Value 2: flat

#-- Value 3: downsloping

44 ca: number of major vessels (0-3) colored by flourosopy

thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

num: diagnosis of heart disease (angiographic disease status)

#-- Value 0: < 50% diameter narrowing

#-- Value 1: > 50% diameter narrowing

Exploration

```
dim(data)
```

```
names(data) <- c("age", "sex", "cp", "trestbps", "choi", "fbs",  
"restecg", "thalach", "exang", "oldpeak", "slope", "ca", "thal",  
"num")
```

```
head(data)
```

```
summary(data)
```

data preprocessing

#presence of heart disease (values 1,2,3,4) from absence of heart
#disease (**value 0**).

#replace all labels greater than 1 by 1. meaning (1,2,3 and 4 = 1)

data preprocessing:

#1. replace all values >1 by 1 for num

```
data$num[data$num > 1] <- 1
```

check the output

```
head(data)
```

```
summary(data)
```

check data type

```
sapply(data, class)
```

#2. data transformation

```
data <- transform( data,  
  age=as.integer(age),  
  sex=as.factor(sex),  
  cp=as.factor(cp),  
  trestbps=as.integer(trestbps),  
  choi=as.integer(choi),  
  fbs=as.factor(fbs),  
  restecg=as.factor(restecg),  
  thalach=as.integer(thalach),  
  exang=as.factor(exang),  
  oldpeak=as.numeric(oldpeak),  
  slope=as.factor(slope),  
  ca=as.factor(ca),  
  thai=as.factor(thai),  
  num=as.factor(num)  
)
```

3. replace all ? by NA

```
# Check data again after transformation  
sapply(data, class)  
summary(data)  
# Check for missing data or strange symbols  
# 3. replace all ? by NA, to be processed later by R  
data[ data == "?" ] <- NA  
colSums(is.na(data))
```

replace all missing values for thai with normal value

#4. replace all missing values for **thai** with normal value
data\$thai[which(is.na(data\$thai))] <- as.factor("3.0")

#5. drop the rows where **ca** is missing
data <- data[!(data\$ca %in% c(NA)),]
colSums(is.na(data))
summary(data)

#6. Get rid of symbol "?"

#6. Get rid of "?"

```
data$ca <- factor(data$ca)  
data$thai <- factor(data$thai)  
summary(data)
```

Modelling

```
# separate data into training and testing  
sample = sample.split(data$num, SplitRatio = .75)  
train = subset(data, sample == TRUE)  
test = subset(data, sample == FALSE)  
dim(train)  
dim(test)
```

```
#show testing data ,column 14 is the class label  
test[-14] # testing data without class label  
test[,14] #class label
```

apply random forest

```
# apply random forest
rf <- randomForest(
  num ~.,
  data=train
)
rf
Call: randomForest(formula = num ~ ., data = train)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 3
OOB estimate of error rate: 19.11%
Confusion matrix:
      0   1 class.error
0 101 20  0.1652893
1 23 81  0.2211538
```

Predecture

```
# Prediction excluding the class label
```

```
pred = predict(rf, newdata=test[-14])
```

```
# Compare results and accuracy
```

```
cm = table(test[,14], pred)
```

```
cm
```

```
➤ cm
```

```
➤ pred
```

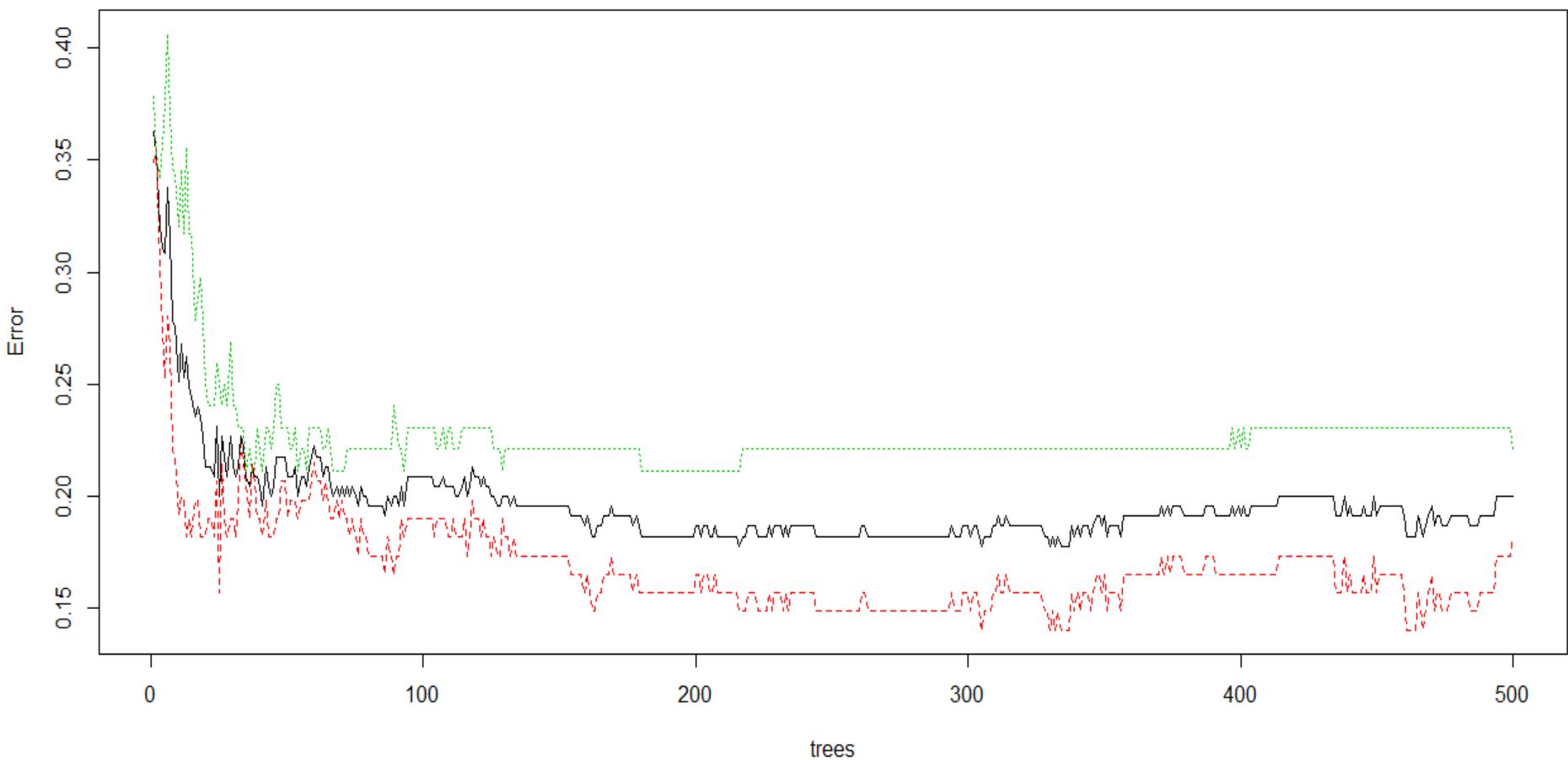
	0	1
0	32	8

1	6	28
---	---	----

Plot the RF

```
# plot the tree  
plot(rf)
```

rf



- Ref
- <https://www.javatpoint.com/machine-learning-random-forest-algorithm>
- https://en.wikipedia.org/wiki/Random_forest



Module 4 – Advanced Analytics – Machine Learning Theory and Methods- Association Rules

Feras Al-Obeidat

Machine Learning

- Machine learning is a subset of artificial intelligence (AI)
- Here, the goal, according to Arthur Samuel,¹ is to give “computers the ability to learn without being explicitly programmed”
- Tom Mitchell puts it more formally: “A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P** if its performance at tasks in **T**, as measured by **P**, improves with experience **E**.”
- Machine learning explores the use of algorithms that can learn from the data and use that knowledge to make predictions on data they have not seen before
- Data-driven predictions or decisions through building a model from sample inputs.

Machine Learning Applications Examples

- Self-driving Google car (now rebranded as WAYMO).

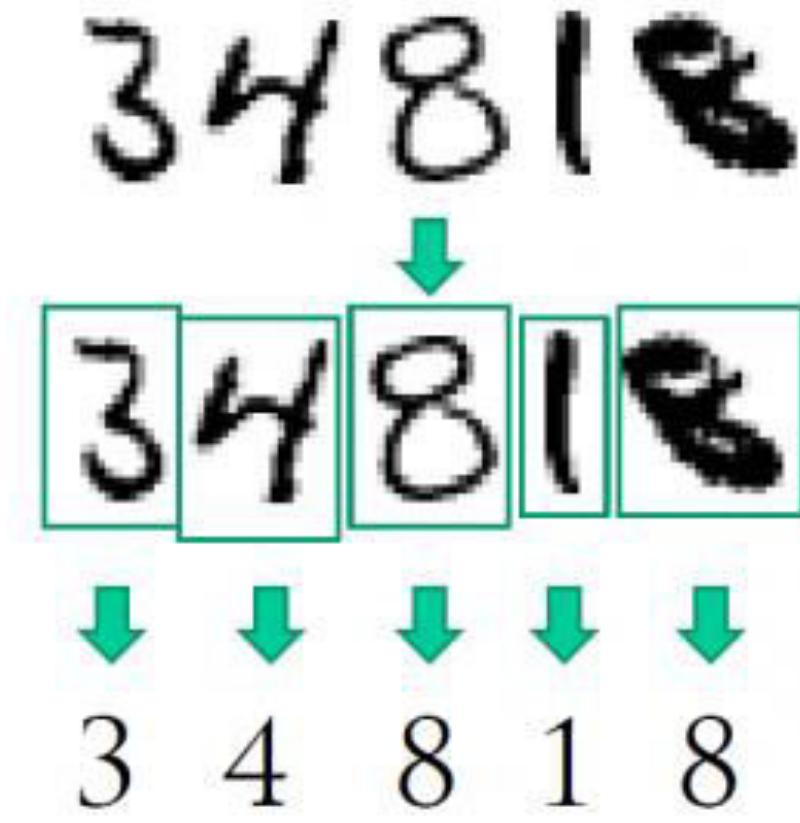


The car is taking a real view of the road to recognize objects and patterns such as sky, road signs, and moving vehicles in a different lane

The self-driving car needs not only to carry out such object recognition, but also to make decisions about navigation. the car needs to know the rules of driving, have the ability to do object and **pattern recognition**, and apply these to making decisions in real time. In addition, it needs to keep improving. That is where machine learning comes into play.

Machine Learning Applications Examples

- optical character recognition (OCR).



Humans are good with recognizing hand-written characters, but computers are not.

what we need is a basic set of rules that tells the computer what "A," "a," "5," etc., look like, and then have it make a decision based on pattern recognition.

The way this happens is by showing several versions of a character to the computer so it learns that character, just like a child will do through repetitions, and then have it go through the recognition process

Machine Learning Applications other Examples

- Facebook uses machine learning to personalize each member's news feed.
- Most financial institutions use machine learning algorithms to detect fraud.
- Intelligence agencies use machine learning to sift through mounds of information to look for credible threats of terrorism.

Machine Learning features

- In machine learning, a target is called a **label**.
- A variable in statistics is called a **feature** in machine learning.
- Machine learning algorithms are organized into a taxonomy, based on the desired out-come of the algorithm. Common algorithm types include:
 - ▶ a. **Supervised learning**. When we **know** the **labels** on the training examples we are using to learn.
 - ▶ b. **Unsupervised learning**. When we **do not know** the **labels** (or even the number of labels or classes) from the training examples we are using for learning.
 - ▶ c. **Reinforcement learning**. When we want to provide feedback to the system based on how it performs with training examples. Robotics is a well-known example.

Association Rules

Which of my products tend to be purchased together?

What do other people like this person tend to like/buy/watch?

- Discover "interesting" relationships among variables in a large database
 - ▶ Rules of the form "If X is observed, then Y is also observed"
 - ▶ The definition of "interesting" varies with the algorithm used for discovery
- **Not** a predictive method; finds similarities, relationships

Apriori Algorithm - What is it?

Support

- Earliest of the association rule algorithms
- Frequent itemset: a set of items L that appears together "often enough":
 - ▶ Formally: meets a **minimum support** criterion
 - ▶ **Support**: the % of transactions that contain L
- Apriori Property: Any subset of a frequent itemset is also frequent
 - ▶ It has at least the support of its superset

Apriori Algorithm (Continued)

Confidence

- Iteratively grow the frequent itemsets from size 1 to size K (or until we run out of support).
 - ▶ **Apriori** property tells us how to **prune the search space**
- Frequent itemsets are used to find rules $X \rightarrow Y$ with a minimum **confidence**:
 - ▶ Confidence: The % of transactions that contain X, which also contain Y
- Output: The set of all rules $X \rightarrow Y$ with minimum support and confidence

Lift

$$\text{Lift}(X \rightarrow Y) = \frac{\text{Support}(X \wedge Y)}{\text{Support}(X) * \text{Support}(Y)}$$

Example on Association Rules, example

Support



- Transaction1: {Apple, Juice, Rice, Chicken}
- Transaction2: {Apple, Juice, Rice}
- Transaction3: {Apple, Juice}
- Transaction4: {Apple, Grapes}
- Transaction5: {Milk, Juice, Rice, Chicken}
- Transaction6: {Milk, Juice, Rice}
- Transaction7: {Milk, Juice}
- Transaction8: {Milk, Grapes}

$$\text{Support(Apple)} = 4/8$$

Confidence

How likely item Juice is purchased when item Apple is purchased, expressed as {Apple -> Juice}. This is measured by the proportion of transactions with item Apple, in which Juice also appears. In Table 1, the confidence of {apple -> Juice} is 3 out of 4, or **75%**.

Confidence {Apple -> Juice} = Support {Apple, Juice}/ Support {Apple}

$$(3/8)/ (4/8)$$

$$(3/4)$$

Lift

Measure 3: Lift. This example shows how likely Juice is purchased when apple is purchased, while controlling for how popular Juice is.

In Table 1, the lift of {apple -> Juice} is **1**, which implies no strong association between items. A lift value greater than 1 means that item Juice is *likely* to be bought if apple is bought, while a value less than 1 means that Juice is *unlikely* to be bought if apple is bought.

$$\text{Lift } \{\text{Apple} \rightarrow \text{Juice}\} = \text{Support } \{\text{Apple, Juice}\} / (\text{Support } \{\text{Apple}\} * \text{Support } \{\text{Juice}\})$$

$$(3/8) / ((4/8)*(6/8))$$

$$(3/4 * (6/8)) = 1$$

Computing Confidence and Lift

Suppose we have **1000** credit records:

	free_housing	home_owner	renter	total
credit_bad	44	186	70	300
credit_good	64	527	109	700
	108	713	179	

713 home_owners, 527 have good credit.

home_owner -> credit_good has confidence $527/713 = 74\%$

700 with good credit, 527 of them are home_owners

credit_good -> home_owner has confidence $527/700 = 75\%$

The lift of these two rules is

$$0.527 / (0.700 * 0.713) = 1.055$$

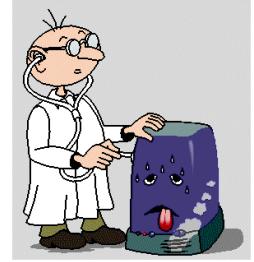
Finally: Find Confidence Rules

Rule	Set	Cnt	Set	Cnt	Confidence
IF credit_good THEN job_skilled	credit_good	700	credit_good AND job_skilled	544	544/700=77%
IF credit_good THEN home_owner	credit_good	700	credit_good AND home_owner	527	527/700=75%
IF job_skilled THEN credit_good	job_skilled	631	job_skilled AND credit_good	544	544/631=86%
IF home_owner THEN credit_good	home_owner	710	home_owner AND credit_good	527	527/710=74%

If we want confidence > 80%:

IF job_skilled THEN credit_good

Diagnostics



- Do the rules make sense?
 - ▶ What does the domain expert say?
- Make a "test set" from hold-out data:
 - ▶ Enter some market baskets with a few items missing (selected at random). Can the rules determine the missing items?
 - ▶ Remember, some of the test data may not cause a rule to fire.
- Evaluate the rules by lift or leverage.
 - ▶ Some associations may be coincidental (or obvious).

Apriori - Reasons to Choose (+) and Cautions (-)



Reasons to Choose (+)	Cautions (-)
Easy to implement	Requires many database scans
Uses a clever observation to prune the search space <ul style="list-style-type: none">•Apriori property	Exponential time complexity
Easy to parallelize	Can mistakenly find spurious (or coincidental) relationships <ul style="list-style-type: none">•Addressed with Lift and Leverage measures

Example on Weather Data to play Soccer

- We want to play soccer, but what to consider :
- (weather ?) Good or not:
 - ▶ Outlook
 - ▶ Temperature
 - ▶ Humidity
 - ▶ Windy ?



Example on Weather Data

Outlook	Temperature	Humidity	Windy	Play Soccer
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

Lab7: Association Rules

- First of all, you will need to download the **weather** data from Black Bord
- Import the data from your computer to **R**
- Set your working directory to the right place where your data is located
- # R code

```
setwd("~/data")
```

```
weather <- read.csv("weather.csv")
```

```
weather
```

Building the model R: apriori

```
# install the “arules” Library  
library(arules)  
  
# find association rules with default settings  
rules.all <- apriori(weather)  
  
inspect(rules.all)
```

Building the model R: apriori

```
inspect(rules.all) # outputs
```

lhs	rhs	<i>support</i>	<i>confidence</i>	<i>lift</i>
[1] {Outlook=Overcast}	=> {Play.Soccer=Yes}	0.2857143	1.0000000	1.555556

If Outlook is Overcast then Play Soccer is yes
with 100% confidence

rules with rhs containing "Play or not" only

```
# rules with rhs containing "Play or not" only  
rulesPlay = apriori(weather, parameter = list(minlen=2,  
supp=0.005, conf=0.8), appearance = list(  
  rhs=c(" Play.Soccer =No", " Play.Soccer =Yes"),  
  default="lhs"))  
  
inspect(rulesPlay)
```

Improve rules quality and appearance (1)

```
quality(rulesPlay) <- round(quality(rulesPlay), digits=3)
rulesPlay.sorted <- sort(rulesPlay, by="lift")
inspect(rulesPlay.sorted)
```

Visualizing Association Rules

Visualizing Association Rules

```
library(arulesViz)  
plot(rulesPlay.pruned )  
plot(rulesPlay.pruned , method="graph")
```

```

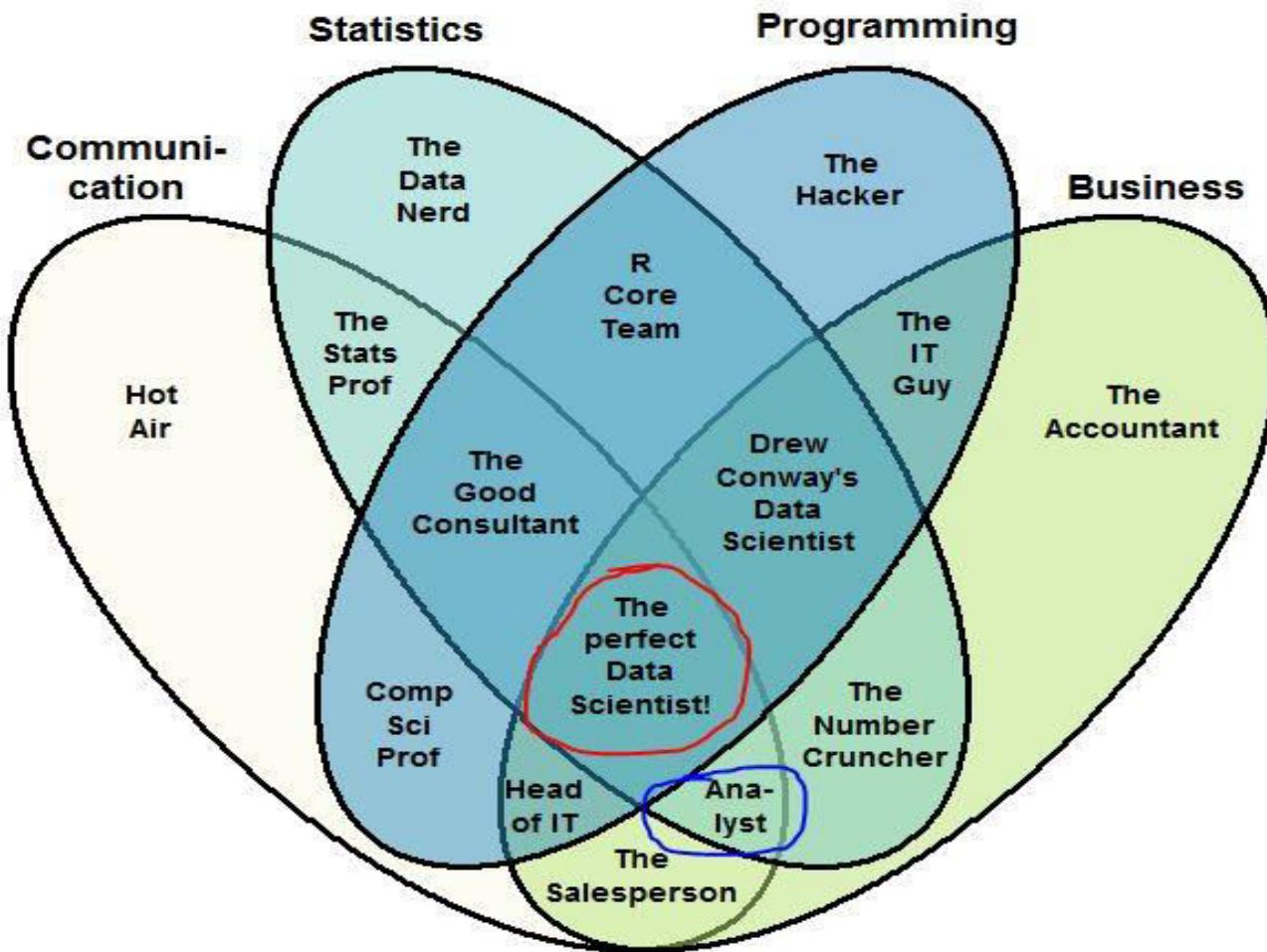
setwd("C:\\\\Users\\\\z10095\\\\Desktop\\\\data\\\\")
weather1 <- read.csv("weather.csv")
weather1
# install the "arules" Library
# install.packages("arules")
library(arules)
# find association rules with default settings
rules.all <- apriori(weather1)
inspect(rules.all)
# rules with rhs containing "Play or not" only
rulesPlay = apriori(weather1, parameter = list(minlen=2, supp=0.005, conf=0.8), appearance = list(
  rhs=c("Play.golf=No", "Play.golf=Yes"), default="lhs"))
inspect(rulesPlay)
### Quality
## for better comparison we sort the rules by confidence and add Bayado's improvement
quality(rulesPlay) <- round(quality(rulesPlay), digits=3)
rulesPlaySorted <- sort(rulesPlay, by = "lift")
inspect(rulesPlaySorted)
is.redundant(rulesPlaySorted)

## redundant rules
inspect(rulesPlaySorted[is.redundant(rulesPlaySorted)])
## non-redundant rules
inspect(rulesPlaySorted[!is.redundant(rulesPlaySorted)])
rulesPlay.pruned = rulesPlaySorted[!is.redundant(rulesPlaySorted)]
inspect(rulesPlay.pruned)

# Visualizing Association Rules
#install.packages("arulesViz")
library(arulesViz)
plot(rulesPlay.pruned )
plot(rulesPlay.pruned , method="graph")

```

The Data Scientist Venn Diagram



LOGISTIC REGRESSION

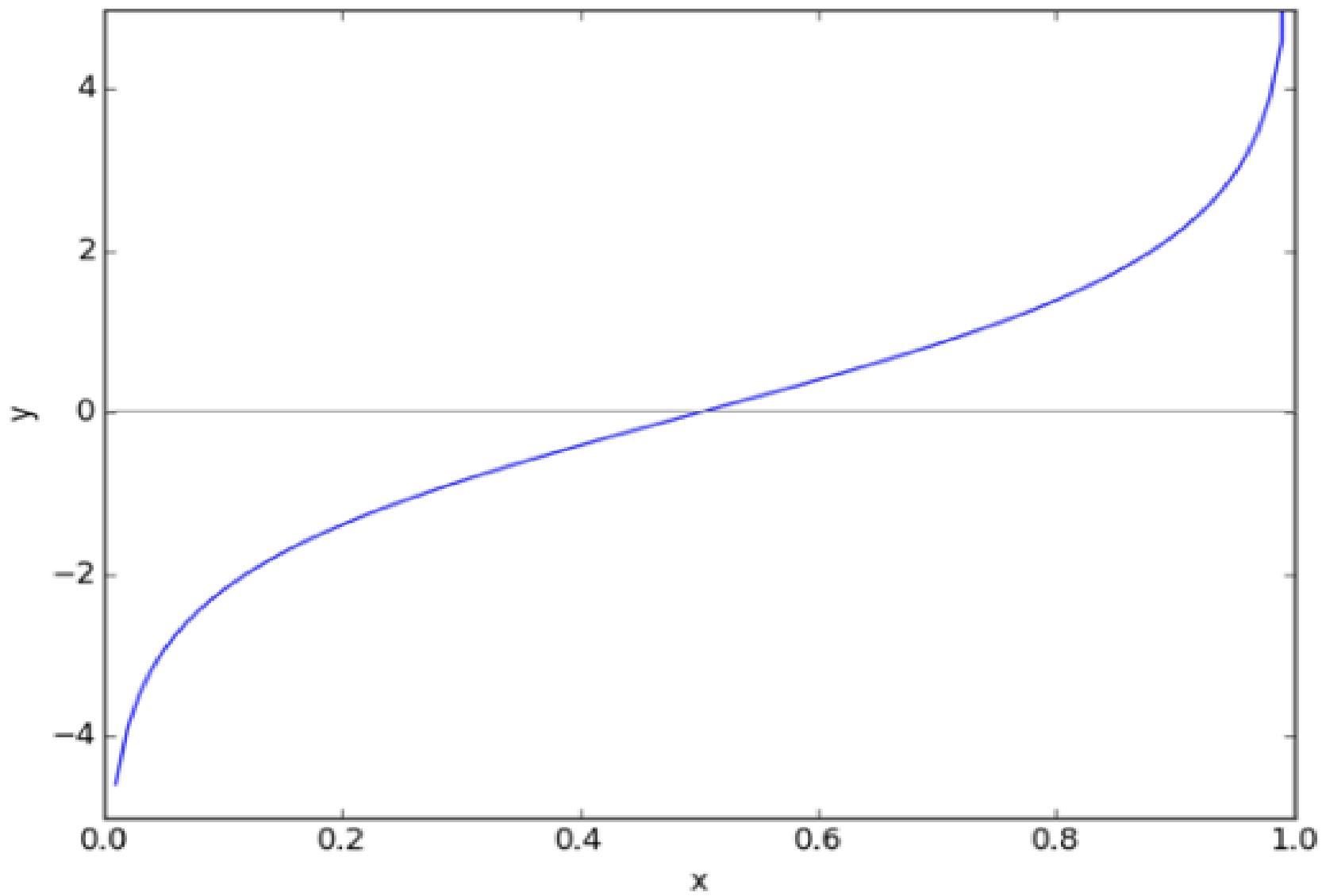
What is Logistic Regression?

- Logistic Regression (LR) is a classification algorithm.
- LR is used to predict a binary outcome:
(1 / 0, Yes / No, True / False) given a set of independent variables.

LR:

- A special case of linear regression when the **outcome** variable is **categorical**.
- LR predicts the probability of occurrence of an event by fitting data to a **logit function**.

logit function



Derivation of Logistic Regression Equation

- Logistic Regression is part of a larger class of algorithms known as Generalized Linear Model (**glm**).
- Provide a means of using linear regression to the problems which were **not directly suited** for application of linear regression. In fact, they proposed a class of different models (linear regression, ANOVA, Poisson Regression etc) which **included logistic regression as a special case**.
- The fundamental equation of generalized linear model is:

$$g(E(y)) = \alpha + \beta x_1 + \gamma x_2$$

Here, $g()$ is the link function, $E(y)$ is the **expectation** of target variable and $\alpha + \beta x_1 + \gamma x_2$ is the linear predictor (α, β, γ to be predicted). The role of link function is to 'link' the expectation of y to linear predictor.

logistic regression

- In logistic regression, the concern is about the probability of **outcome** dependent variable (**success** or **failure**). As described above, $g()$ is the link function.
- This function is established using two things:
 - ▶ Probability of Success(p) and
 - ▶ Probability of Failure($1-p$).
 - ▶ p should meet following criteria:
 - ▶▶ It must always be positive (since $p \geq 0$)
 - ▶▶ It must always be less than equals to 1 (since $p \leq 1$)

Application Example in R

- Predict survival on the Titanic

- Data description:
 - ▶ *Survival* - Survival (0 = No; 1 = Yes). Not included in test.csv file.
 - ▶ *Pclass* - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
 - ▶ *Name* - Name
 - ▶ *Sex* - Sex
 - ▶ *Age* - Age
 - ▶ *Sibsp* - Number of Siblings/Spouses Aboard
 - ▶ *Parch* - Number of Parents/Children Aboard
 - ▶ *Ticket* - Ticket Number
 - ▶ *Fare* - Passenger Fare
 - ▶ *Cabin* - Cabin
 - ▶ *Embarked* - Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

Application Example in R

- Upload and read the data

```
# upload the dataset  
setwd("C:\\\\Users\\\\z10095\\\\Desktop\\\\courseFiles\\\\Spring2021\\\\477\\\\")  
  
# read dataset  
titanic.data <-read.csv("OriginalTitanic.csv", header = TRUE, sep = ",")  
# check dataset  
titanic.data  
View(titanic.data)
```

Application Example in R

- preprocessing
- check missing values

```
#Before building the model, we need to check for missing values  
#using sapply() function  
  
sapply(titanic.data,function(x) sum(is.na(x)))
```

```
#find how many unique values there are for each variable using  
#sapply() function  
  
sapply(titanic.data,function(x)length(unique(x)))
```

Application Example in R

- estimate the missing using visualization

```
#Another way to estimate the missing values is to use a visualization  
#package "Amelia" which has  
#a plotting function missmap() that serves the purpose. It will plot  
#your dataset and highlight missing values:
```

```
library(Amelia)  
missmap(titanic.data, main="Missingvaluesvsobserved")  
#the Age column has multiple missing values. Then should clean up  
#the missing values before proceeding further.
```

Application Example in R

- replace missing values

```
#Replace missing values with the mean age value:
```

```
titanic.data$Age[is.na(titanic.data$Age)] <- mean(titanic.data$Age,na.rm=T)
```

```
# select subset of data
```

```
titanic.data <-subset(titanic.data, select=c(2,3,5,6,7,8,10,12))
```

Application Example in R

- Check categorical factor

#For the categorical variables in the dataset, using the read.table() or read.csv() by default will encode

#the categorical variables as factors. A **factor** is how R deals with **categorical** variables.

#We can check the encoding using the is.factor() function, which should return "true" for all the categorical variables:

```
is.factor(titanic.data$Sex)
```

Application Example in R

- separate data into training and testing

```
#separate the dataset into training and test sets.
```

```
#the first 800 instances for training and
```

```
#the remaining 91 as test instances. You can opt for different separation  
strategies:
```

```
train <-titanic.data[1:800,]
```

```
test <-titanic.data[801:891,]
```

Application Example in R

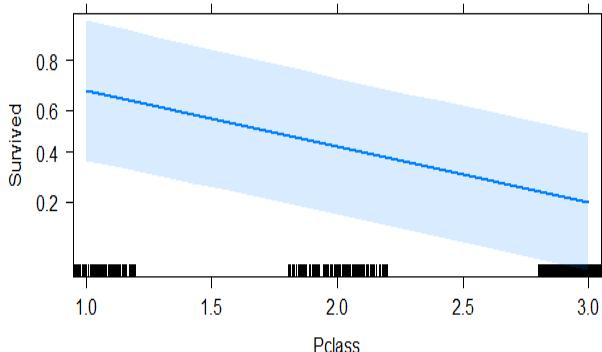
- Building the model

#building the model. We are going to use parameter family=bi-nomial (two classes or labels) in the glm() function:

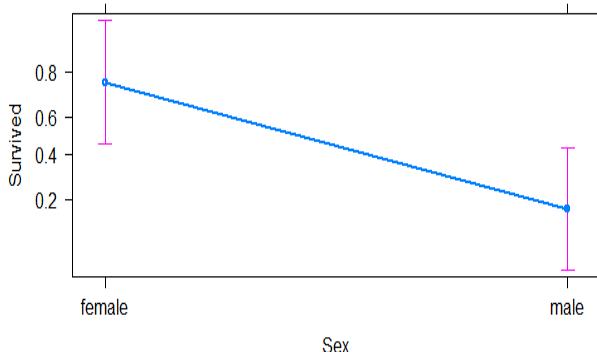
```
model <- glm(Survived ~.,family=binomial(link='logit'), data=train)
summary(model)
library(effects)
plot(allEffects(model))
```

plots

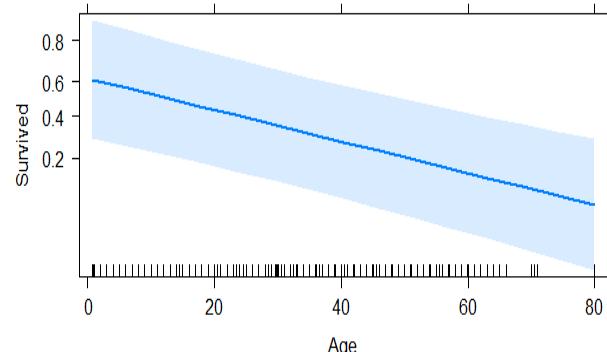
Pclass effect plot



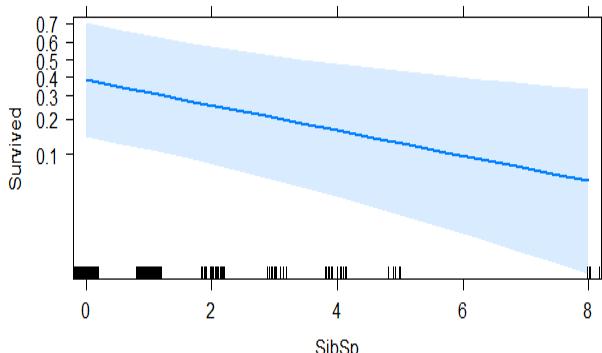
Sex effect plot



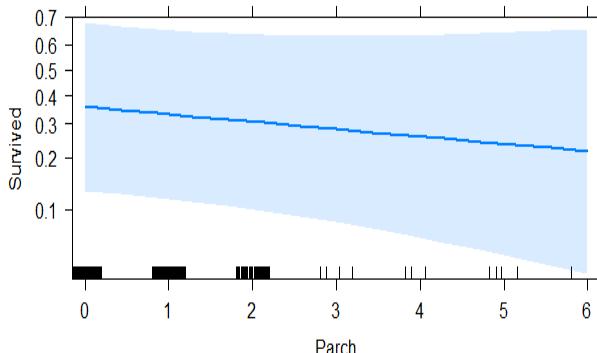
Age effect plot



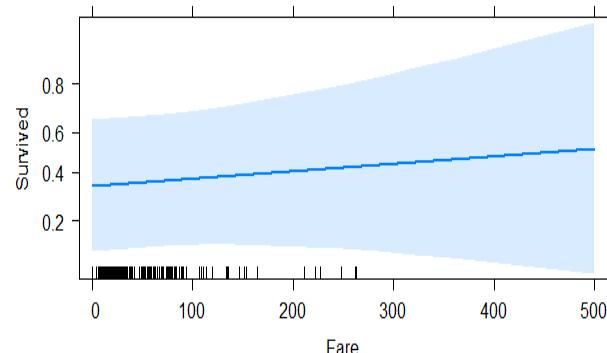
SibSp effect plot



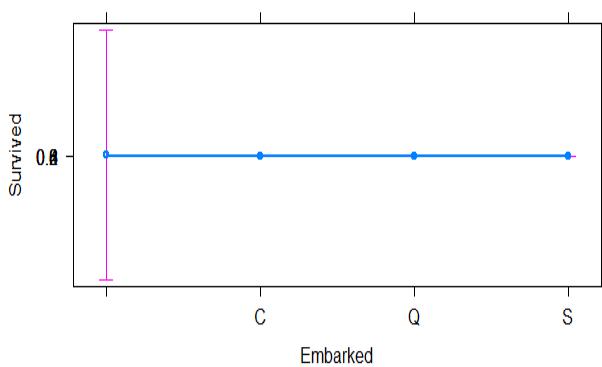
Parch effect plot



Fare effect plot



Embarked effect plot



summary(model)

```
> summary(model)

Call:
glm(formula = Survived ~ ., family = binomial(link = "logit"),
     data = train)

Deviance Residuals:
    Min      1Q  Median      3Q      Max 
-2.6059 -0.5933 -0.4250  0.6215  2.4148 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 15.947761 535.411378  0.030   0.9762    
Pclass       -1.087576  0.151088 -7.198  6.10e-13 ***  
Sexmale      -2.754348  0.212018 -12.991 < 2e-16 ***  
Age          -0.037244  0.008192 -4.547  5.45e-06 ***  
SibSp        -0.293478  0.114660 -2.560   0.0105 *    
Parch        -0.116828  0.128113 -0.912   0.3618    
Fare         0.001515  0.002352  0.644   0.5196    
EmbarkedC   -10.810682 535.411254 -0.020   0.9839    
EmbarkedQ   -10.812679 535.411320 -0.020   0.9839    
EmbarkedS   -11.126837 535.411235 -0.021   0.9834    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1066.33 on 799 degrees of freedom
Residual deviance: 708.93 on 790 degrees of freedom
AIC: 728.93

Number of Fisher Scoring iterations: 12
```

results analysis:

Results analysis:

- It is clear that **Fare** and **Embarked** are **not** statistically **significant**.
- That means no enough **confidence** that these factors contribute all that much to the overall model.
- As for the statistically **significant** variables, **Sex** has the **lowest p-value**,
- There is a strong association of the **sex** of the passenger with the probability of having **survived**.
- The male passenger is **less** likely to have survived.

As Titanic started sinking and the lifeboats were being filled with rescued passengers,

priority was given to women and children.

And thus, it makes sense that a male passenger, especially a male adult, would have had less chance of survival.

Prediction

#Now, we will see how good our model is in predicting values for test instances. By setting the parameter type='response', R will output probabilities in the form of $P(y = 1 | X)$.

#Our decision boundary will be 0.5. If $P(y = 1 | X) > 0.5$ then $y = 1$, otherwise $y = 0$.

```
fitted.results <-  
predict(model,newdata=subset(test,select=c(2,3,4,5,6,7,8)),  
type='response')
```

```
fitted.results
```

```
# binary classification
```

```
fitted.results <-ifelse(fitted.results > 0.5,1,0)
```

```
fitted.results
```

Model Accuracy

```
misClasificError <- mean(fitted.results != test$Survived)
print(paste('Accuracy',1-misClasificError))
> print(paste('Accuracy',1-misClasificError))
> [1] "Accuracy 0.846153846153846"
```

#As we can see from the above result, the accuracy of our model in predicting the labels of the test

#instances is at 0.846, which suggests that the model performed decently.

Plot the receiver operating curve (ROC)

```
#now plot the receiver operating curve (ROC) and calculate the area under  
#curve (AUC), which are typical performance measurements for a binary  
classifier.
```

#For the details of these measures, refer to Chapter 12

```
#install.packages("ROCR")
```

```
library(ROCR)
```

```
p <- predict(model, newdata=subset(test,select=c(2,3,4,5,6,7,8)), type="response")
```

```
#p
```

```
pr <- prediction(p, test$Survived)
```

```
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
```

```
plot(prf)
```

ROC and AUC

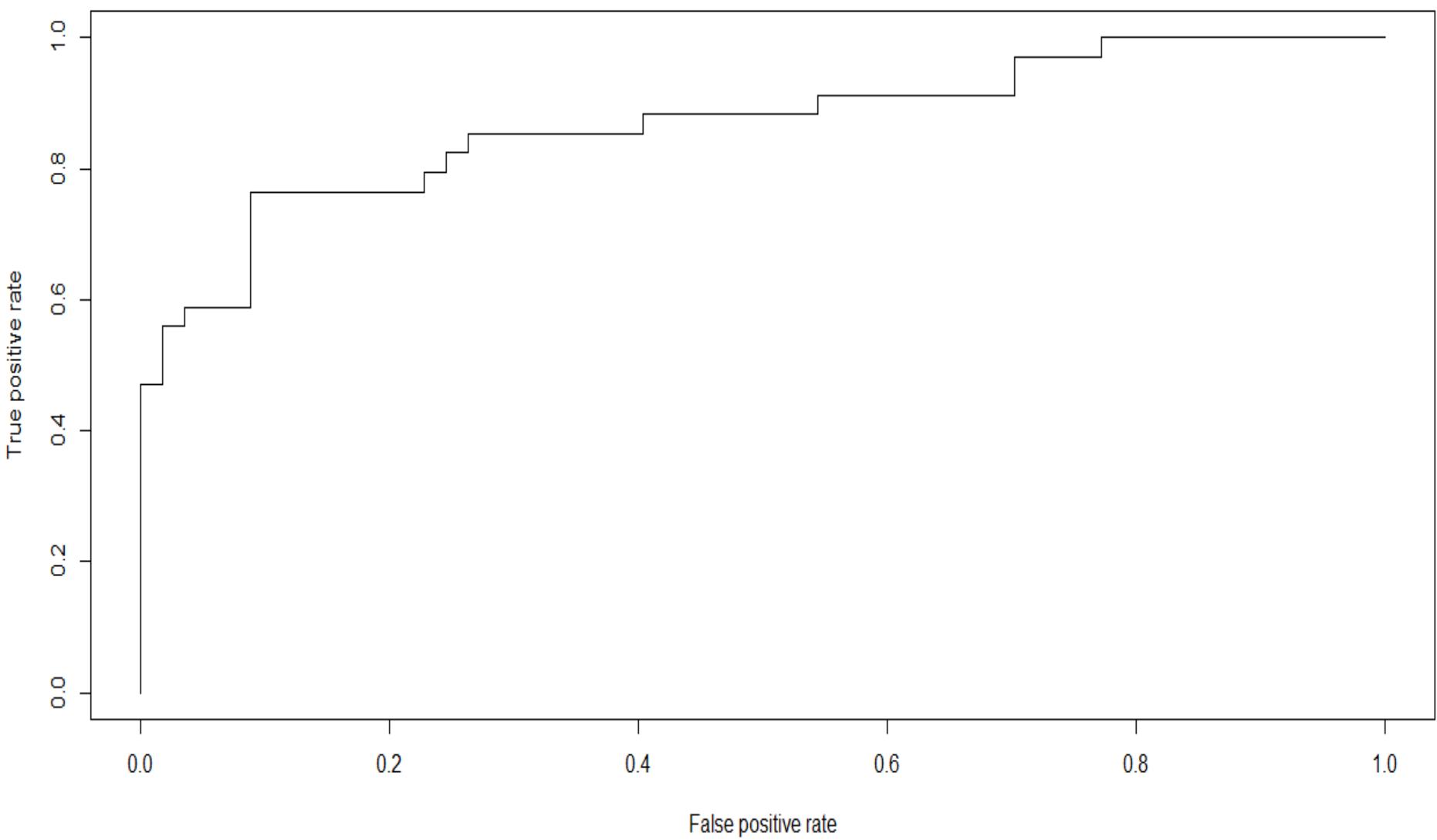
- The ROC curve is generated by plotting the **true positive rate (TPR)** against the **false positive rate (FPR)** at various threshold settings,
- **AUC** is the area under the **ROC** curve.
- TPR indicates how much of what we detected as "1" was **indeed** "1," and
- FPR indicates how much of what we detected as "1" was actually "**0**"

Typically, as one goes up, the other goes up too. Think about it - if you declare everything "1," you will have a high TPR, but you would have also wrongly labeled everything that was supposed to be "0" leading to high FPR.

- As a rule of thumb, a model with **good predictive** ability should have an AUC closer to **1** than to **0.5**.
- # from generated plot, can see that the **AUC** is quite large - covering about **87%** of the rectangle.

That is quite a good number, indicating a good and balanced classifier.

ROC Curve



calculate the area under curve (AUC)

```
auc <- performance(pr, measure = "auc")
```

```
#auc
```

```
auc <- auc@y.values[[1]]
```

```
auc
```

```
> auc [1] 0.8684211
```

from generated results can see that the area under the curve is quite large - covering about **87%** of

#the rectangle. **That is quite a good number**, indicating a good and balanced classifier.

Feature Selection

Dr. Feras Al-Obeidat

Overview

- Feature Selection FS and dimensionality reduction:
 1. Improve performance (speed, predictive power, simplicity of the model).
 2. Visualize the data for model selection.
 3. Reduce dimensionality and remove noise.
- *Feature Selection* is a process to select optimal subset of features according to a certain criterion.

Feature selection

All Features



Feature Selection



Final Features



Overview

- ▶ Other reasons for performing FS may include:
 - ▶ removing irrelevant data and noise.
 - ▶ increasing accuracy of learned models.
 - ▶ reducing the complexity of the resulting model description, improving the understanding of the data and the model.
 - ▶ Dimensionality reduction is an efficient approach to downsizing data
 - ▶ **Visualization:** projection of high-dimensional data onto 2D or 3D

Feature Selection

- ▶ Definition
 - ▶ A process that chooses an optimal subset of features according to a objective function
- ▶ Objectives
 - ▶ To reduce dimensionality and remove noise
 - ▶ To improve mining performance
 - ▶ Speed of learning
 - ▶ Predictive accuracy
 - ▶ Simplicity and comprehensibility of mined results

Application of Dimensionality Reduction

- ▶ Customer relationship management
- ▶ Text mining
- ▶ Image retrieval
- ▶ Handwritten digit recognition
- ▶ Intrusion detection

FS, how it Works..



1. Searching for the best subset of features.
2. Criteria on how to evaluating different subsets.

Perspectives: Search of a Subset of Features

► Search Directions:



► Sequential Forward Generation (SFG):

It starts with an empty set of features.

It grows until it reaches a full set of original features.

► The stopping criteria can be a threshold for the number of relevant features

Perspectives: Search of a Subset of Features

► Search Directions:

► Sequential Backward Generation (SBG):

starts with a full set of features and, iteratively, they are removed one at a time.

Here, we can identify the worst or least important feature.

- the subset is considered to be the most informative of the whole set.
- Different stopping criteria can be used.

Perspectives: Search of a Subset of Features

► Search Directions:

- Bidirectional Generation (BG): Begins the search in both directions, performing SFG and SBG concurrently.
- Random Generation (RG): It starts the search in a random direction. The choice of adding or removing a features is a random decision.

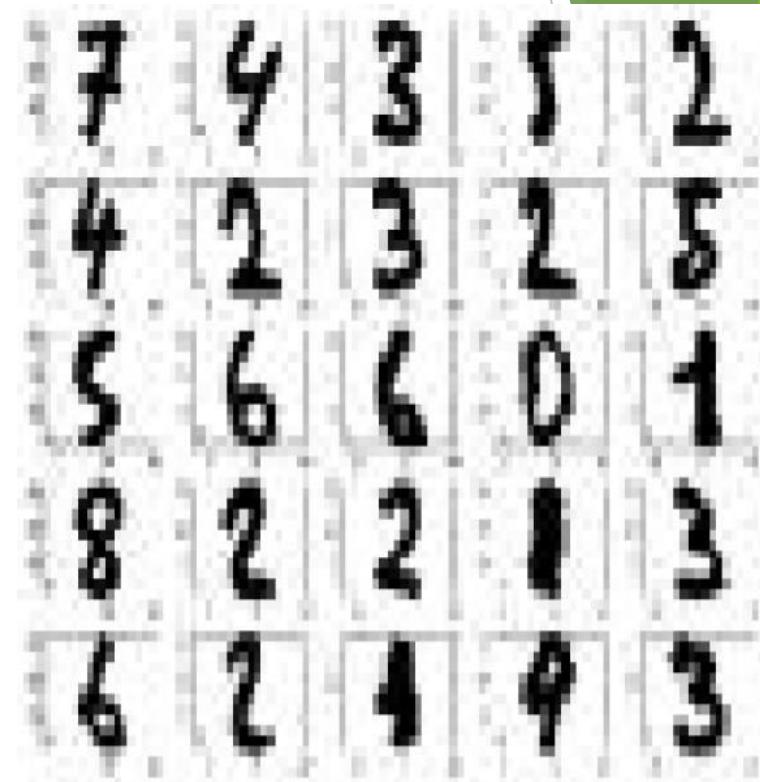
Different Aspects of Search

- ▶ Search starting points
 - ▶ Empty set
 - ▶ Full set
 - ▶ Random point
- ▶ Search directions
 - ▶ Sequential forward **selection**
 - ▶ Sequential backward **elimination**
 - ▶ Bidirectional generation
 - ▶ Random generation

Other Types of High-Dimensional Data



Face images



Handwritten digits

Perspectives: Selection Criteria

► Accuracy Measures.

► This form of evaluation relies on the classifier or learner. Among various possible subsets of features, the subset which yields the best accuracy is chosen

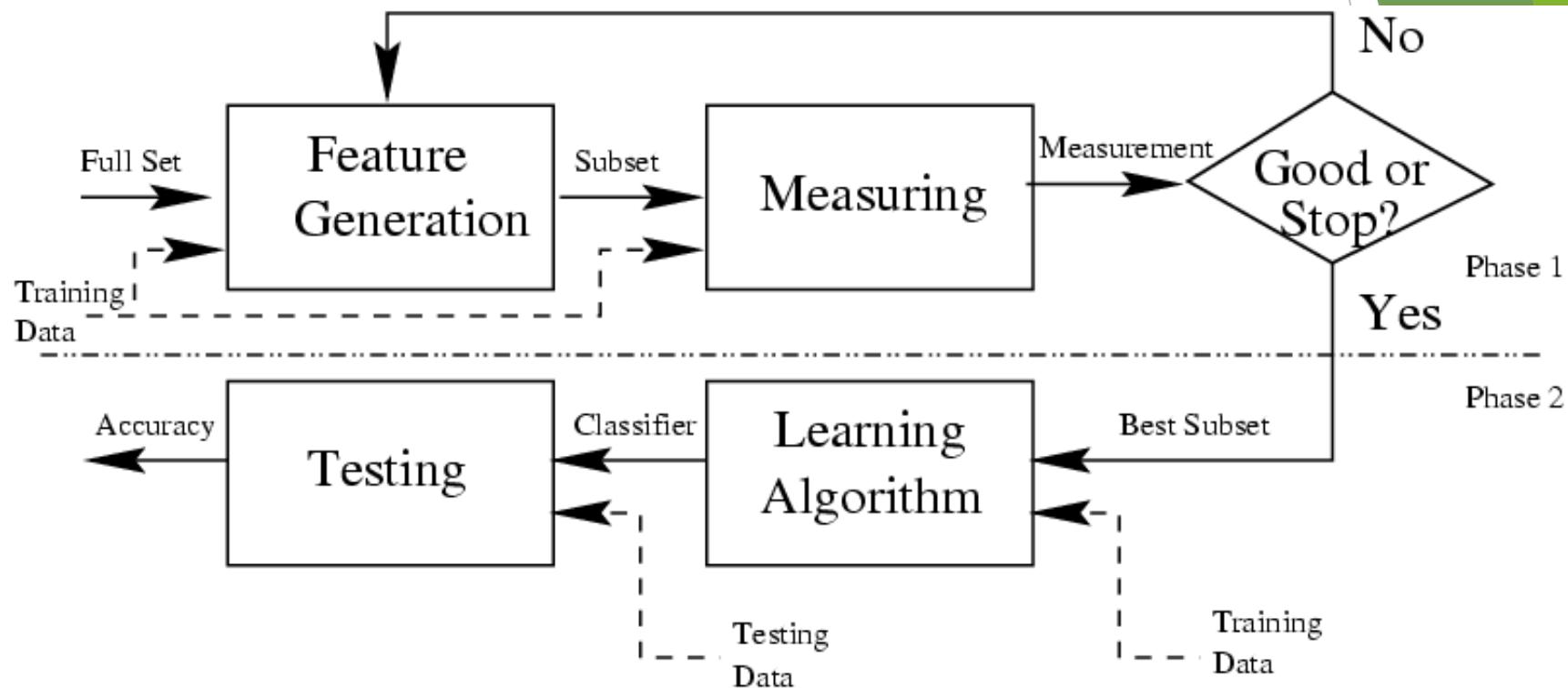


	Mathematical form
Accuracy	$\frac{tp + fp}{tp + tn + fp + fn}$
Error rate	1 – Accuracy
Chi-squared	$\frac{n(fp \times fn - tp \times tn)^2}{(tp+fp)(tp+fn)(fp+tn)(tn+fn)}$
Information gain	$e(tp + fn, fp + tn) - \frac{(tp+fp)e(tp, fp) + (tn+fn)e(fn, tn)}{tp+fp+tn+fn}$ where $e(x, y) = -\frac{x}{x+y} \log_2 \frac{x}{x+y} - \frac{y}{x+y} \log_2 \frac{y}{x+y}$
Odds ratio	$\frac{tpr}{1-tpr} / \frac{fpr}{1-fpr} = \frac{tp \times tn}{fp \times fn}$
Probability ratio	$\frac{tpr}{fpr}$

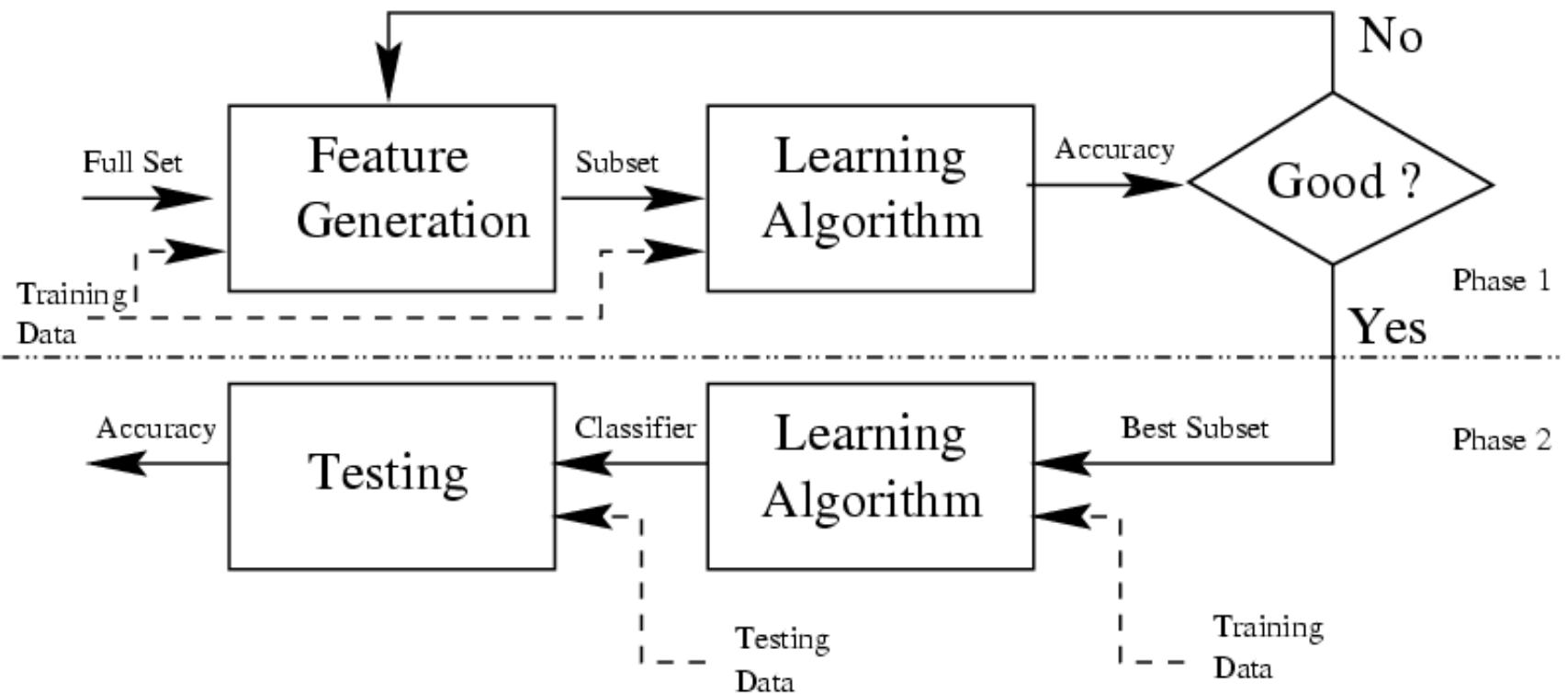
Models of Feature Selection

- ▶ Filter model
 - ▶ Separating feature selection from classifier learning
 - ▶ Relying on general characteristics and statistics of data (*correlation, distance, dependence, consistency*)
- ▶ Wrapper model
 - ▶ Relying on a predetermined classification algorithm
 - ▶ Using **predictive accuracy** as goodness measure
 - ▶ High accuracy, but **computationally expensive**

Filter Model



Wrapper Model



Representative Algorithms for Clustering

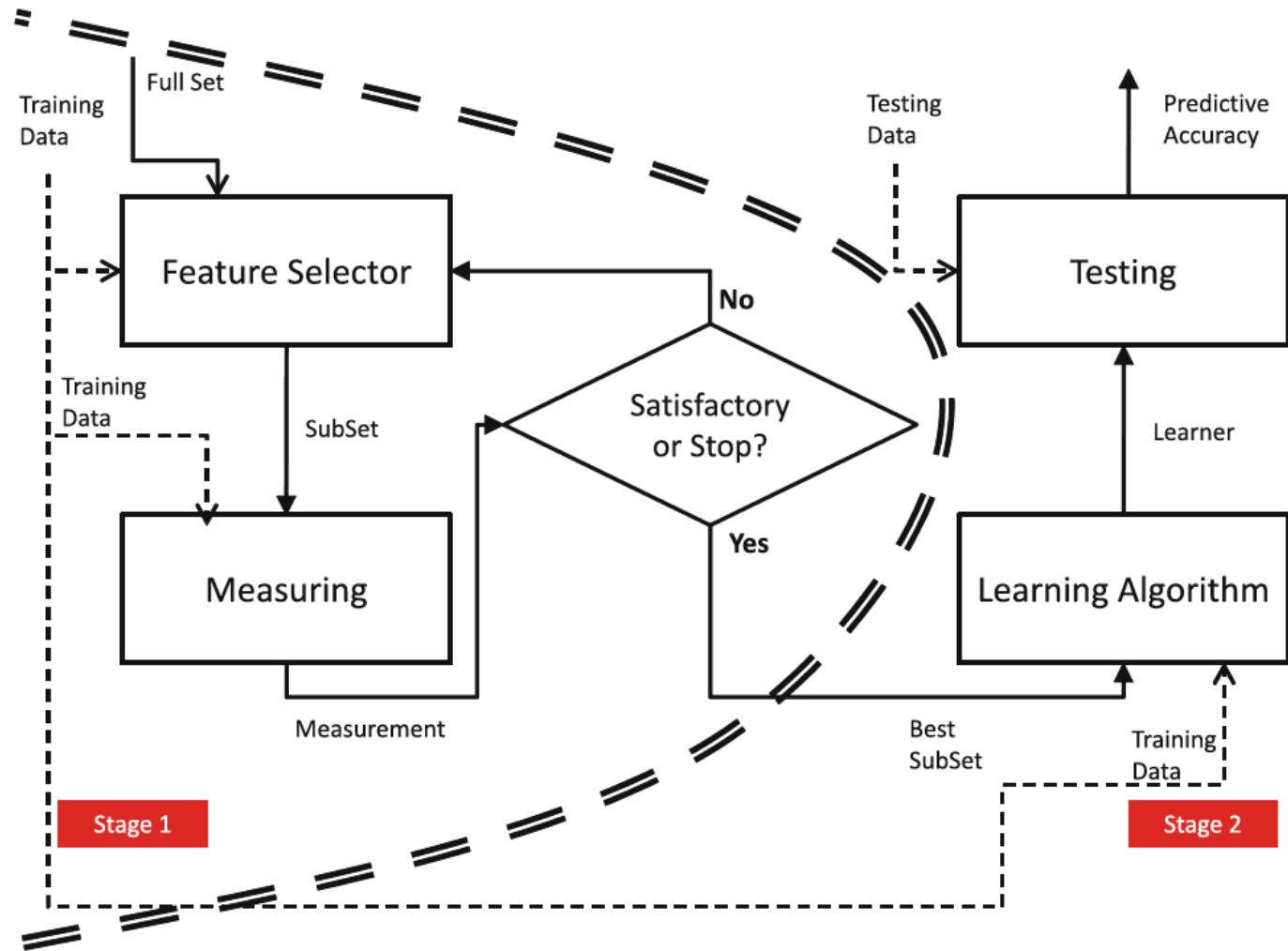
- ▶ Filter algorithms
 - ▶ Example: a filter algorithm based on entropy measure or information gain
- ▶ Wrapper algorithms
 - ▶ Example: - a wrapper algorithm based on clustering or classification accuracy
- ▶ wrapper based are advantageous for giving **better performances** since they use the target classifier the feature selection algorithm but they suffer they are **computationally expensive**.
- ▶ filter methods are **less accurate** but **faster** to compute.

Aspects: Output of Feature Selection

- ▶ Feature Ranking Techniques:
 - ▶ Output is a ranked list of features which are ordered according to evaluation measures.
 - ▶ they return the relevance of the features.
 - ▶ For performing actual FS, the simplest way is to choose the top features according the data at hand

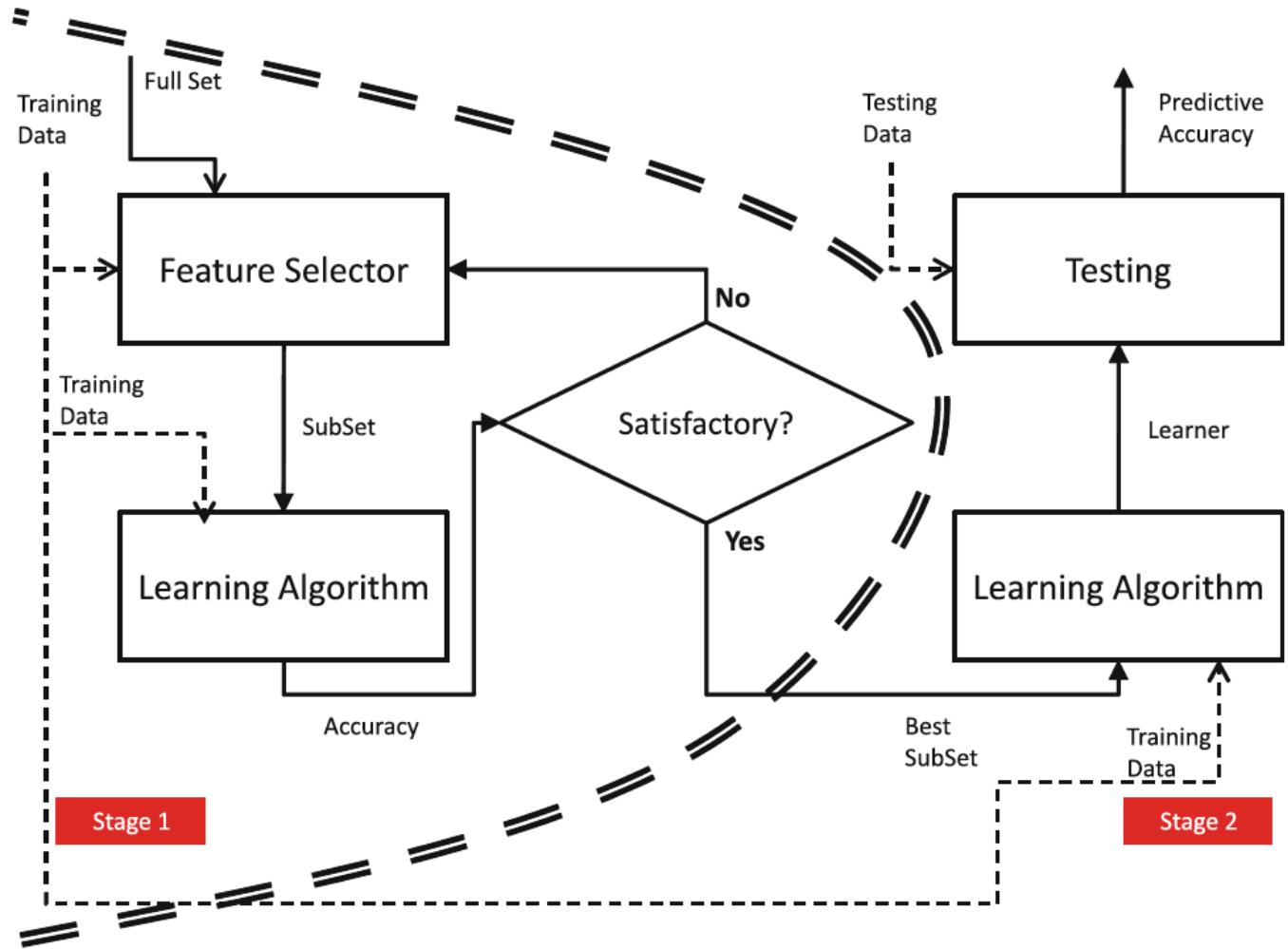
Filter Approach

► Filters:



Perspectives

► Wrappers:



Aspects: Evaluation

► Goals:

- **Inferability:** For predictive tasks, considered as an improvement of the prediction of unseen examples with respect to the direct usage of the raw training data.
- **Interpretability:** Generate more understandable structure representation that can explain the behavior of the data.
- **Data Reduction:** It is better and simpler to handle data with lower dimensions in terms of efficiency and interpretability.

Aspects: Evaluation

- ▶ Assessment measures from these goals:
 - ▶ Accuracy
 - ▶ Complexity
 - ▶ Number of Features Selected
 - ▶ Speed of the FS method

Aspects: Drawbacks of FS in some cases

- ▶ The resulted subsets of many models of FS are strongly dependent on the training set size.
- ▶ the removal of any of them will seriously effect the learning performance.
- ▶ A backward removal strategy is very slow when working with large-scale data sets.
- ▶ In some cases, the FS outcome will still be left with a relatively large number of relevant features.

Aspects: Using Decision Trees for FS

- ▶ Decision tree inducers can be considered as anytime algorithms for FS, due to the fact that they gradually improve the performance and can be stopped at any time, providing sub-optimal feature subsets.

Most Representative Methods

- ▶ Three major components to categorize combinations:
 - ▶ **Search Direction**
 - ▶ **Search Strategy**
 - ▶ **Evaluation Measure**

Most Representative Methods

Exhaustive Methods

- ▶ Cover the whole search space.
 - ▶ Best First Search
 - ▶ Best first search uses the concept of a priority queue and heuristic search. It is a search algorithm that works on a specific rule. The aim is to reach the goal from the initial state via the shortest path.
 - ▶ Beam Search
 - ▶ Beam search is an optimization of [best-first search](#) that reduces its memory requirements.

Most Representative Methods

Nondeterministic Methods

- ▶ They add or remove features to and from a subset without a sequential order.
 - ▶ **Genetic Algorithms** are the most common techniques.

Related and Advanced Topics

Leading and Recent FS Techniques

- ▶ Use of meta-heuristics:
 - ▶ Genetic algorithms.
 - ▶ Tabu search.
 - ▶ Hybridizations between genetic algorithms and local searches.

Example of feature selection in R... Wrapper approach

- ▶ In this example we will use Boruta Package
- ▶ Boruta is FS algorithm. It works as a wrapper algorithm around Random Forest.
- ▶ Random forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

How does Boruta algorithm works?

- ▶ Firstly, it adds randomness to the given data set by creating shuffled copies of all features (called shadow features).
- ▶ Then, it trains a random forest classifier on the extended data set and applies a feature importance measure.
- ▶ At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features and constantly removes features which are unimportant.
- ▶ Finally, the algorithm stops either when all features get confirmed or rejected or it reaches a specified limit of random forest runs.

Application of Boruta algorithm and Random forest in R

- ▶ Required libraries :

```
library(Boruta)
```

```
library(mlbench)
```

```
library(caret)
```

```
library(randomForest)
```

```
library(reprtree)
```

The code 1

- ▶ `set.seed(111)`
- ▶ `boruta <- Boruta(Species ~ ., data = iris, doTrace = 2,
maxRuns = 500)`
- ▶ `print(boruta)`

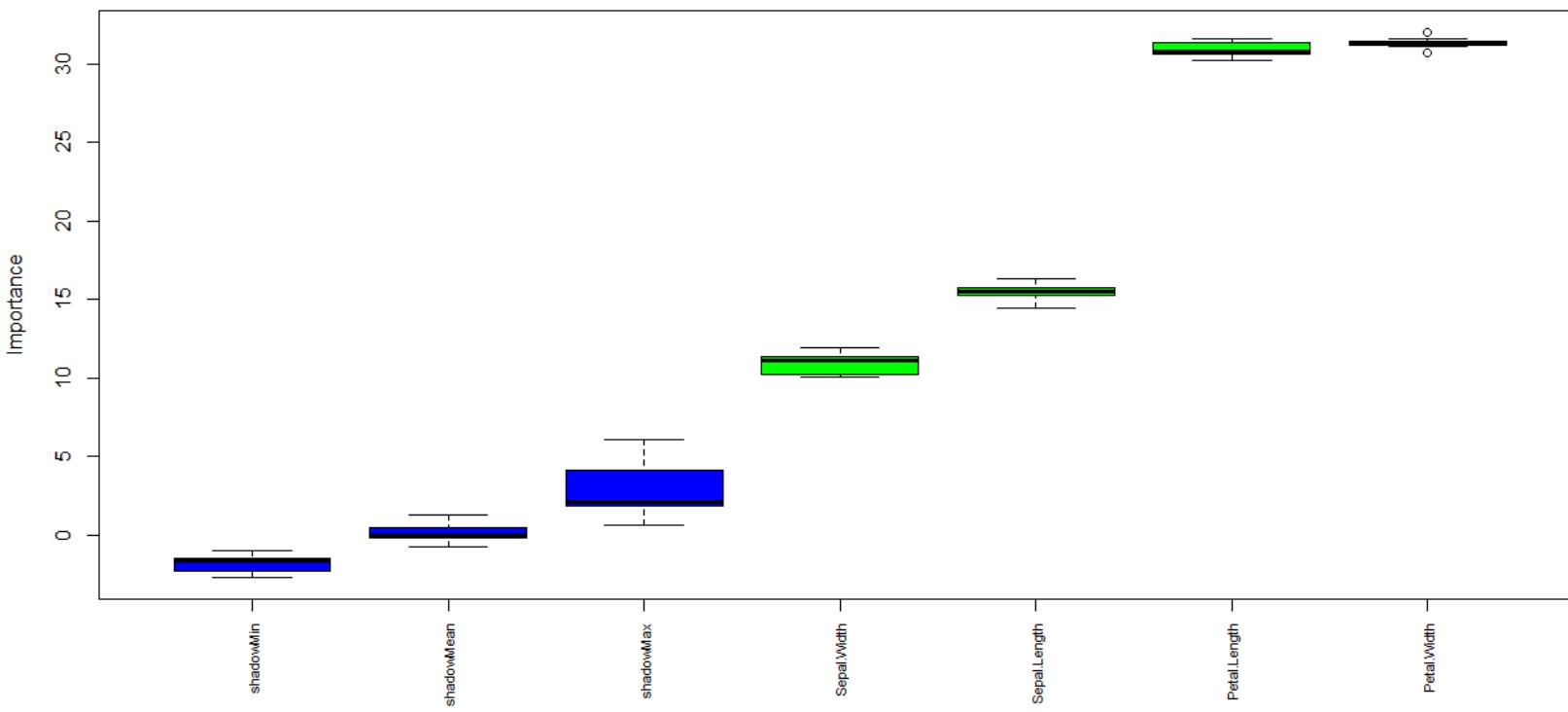
The code 2

Plot the output

```
plotImpHistory(boruta)
plot(boruta, xlab = "", xaxt = "n")
lz<-lapply(1:ncol(boruta$ImpHistory),function(i)
  boruta$ImpHistory[is.finite(boruta$ImpHistory[,i]),i])
names(lz) <- colnames(boruta$ImpHistory)
Labels <- sort(sapply(lz,median))
axis(side = 1,las=2,labels = names(Labels),
  at = 1:ncol(boruta$ImpHistory), cex.axis = 0.7)
```

Explanation of the plot

- ▶ #Blue boxplots correspond to minimal, average and maximum Z score of a shadow attribute.
- ▶ #Red, yellow and green boxplots represent Z scores of rejected, tentative and confirmed attributes



The code 3

Tentative attributes & confirmed

```
final.boruta <- TentativeRoughFix(boruta)  
print(final.boruta)
```

```
# list of confirmed attributes  
getSelectedAttributes(final.boruta, withTentative = F)
```

The code 4

Data partition and apply random forest

```
set.seed(222)

ind <- sample(2, nrow(iris), replace = T, prob = c(0.7, 0.3))

train <- iris[ind==1,]

test <- iris[ind==2,]

# Random Forest Model

#install.packages('devtools',repos='http://cran.us.r-project.org')

library(devtools)

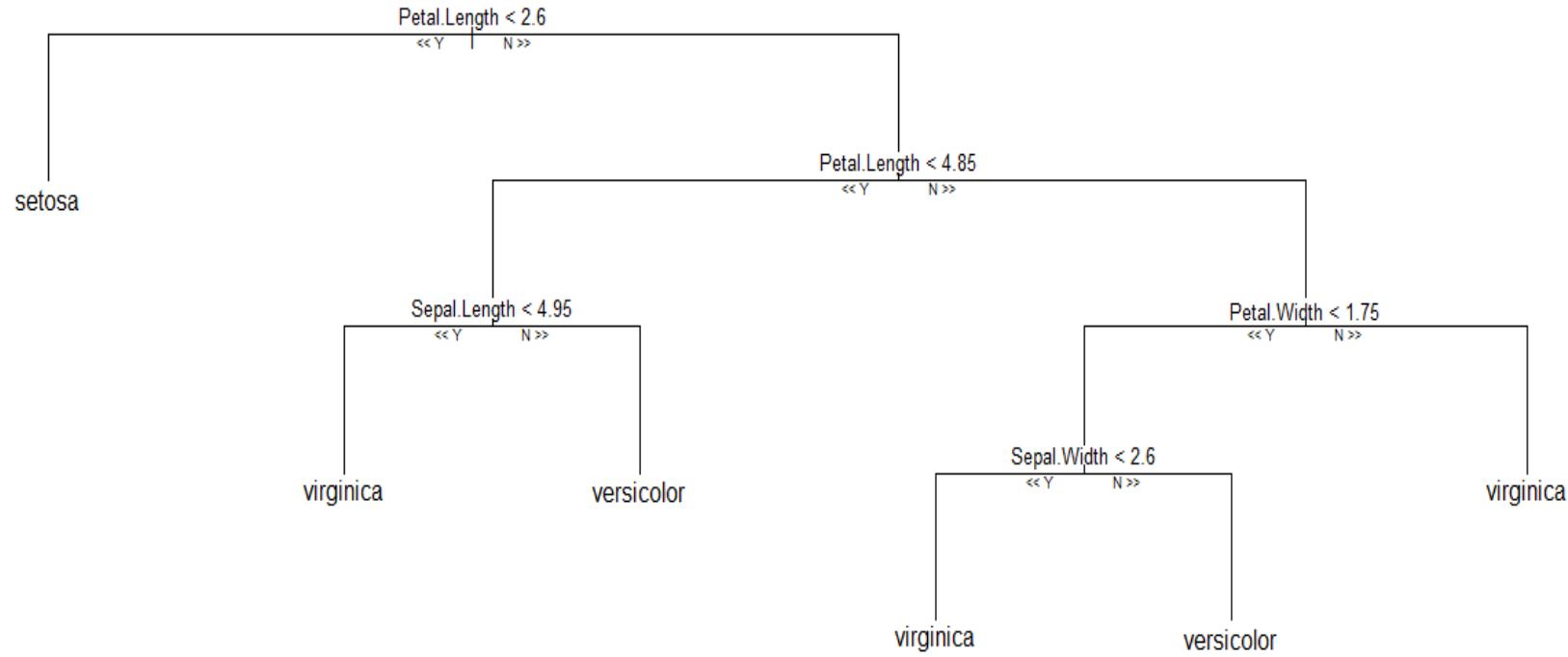
library(reprtree)

#devtools::install_github('skinner927/reprtree')

set.seed(345)

rfliris <- randomForest(Species~, data = train)

reprtree:::plot.getTree(rfliris)
```



The code 5

Prediction & Confusion Matrix

```
pliris <- predict(rfIris, test)  
confusionMatrix(pliris, test$Species)
```

Results & Accuracy ...

Confusion Matrix and Statistics

Reference

Predection setosa versicolor virginica

	setosa	versicolor	virginica
setosa	18	0	0
versicolor	0	15	3
virginica	0	0	13

Overall statistics

Accuracy: 93%

Weka

Feras Al-Obeidat

WEKA: the bird



What is WEKA?

- ▶ **Waikato Environment for Knowledge Analysis**
 - ▶ It's a data mining/machine learning tool developed by
Department of Computer Science, University of Waikato, New Zealand.
 - ▶ Weka is also a bird found only on the islands of New Zealand.

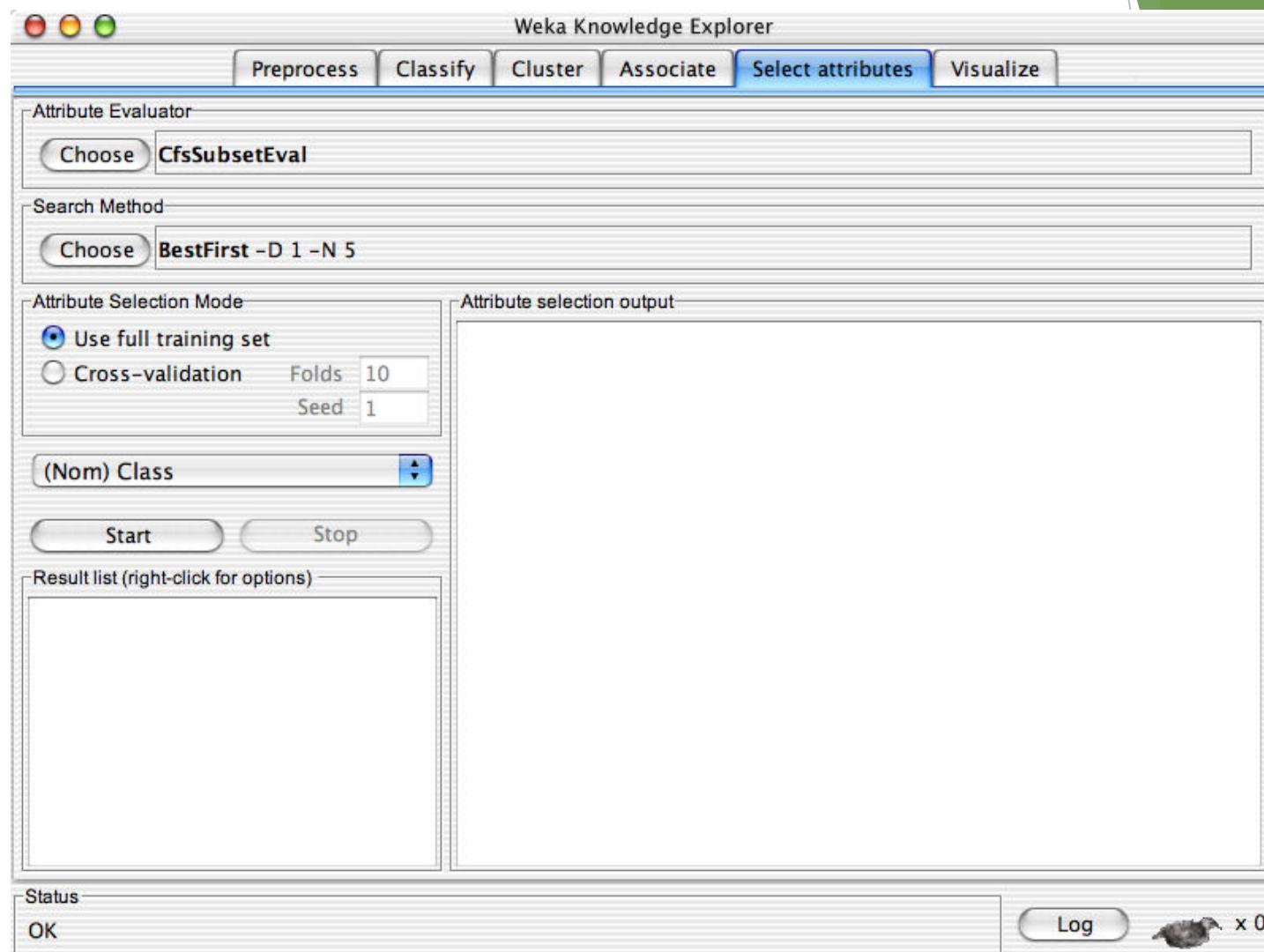


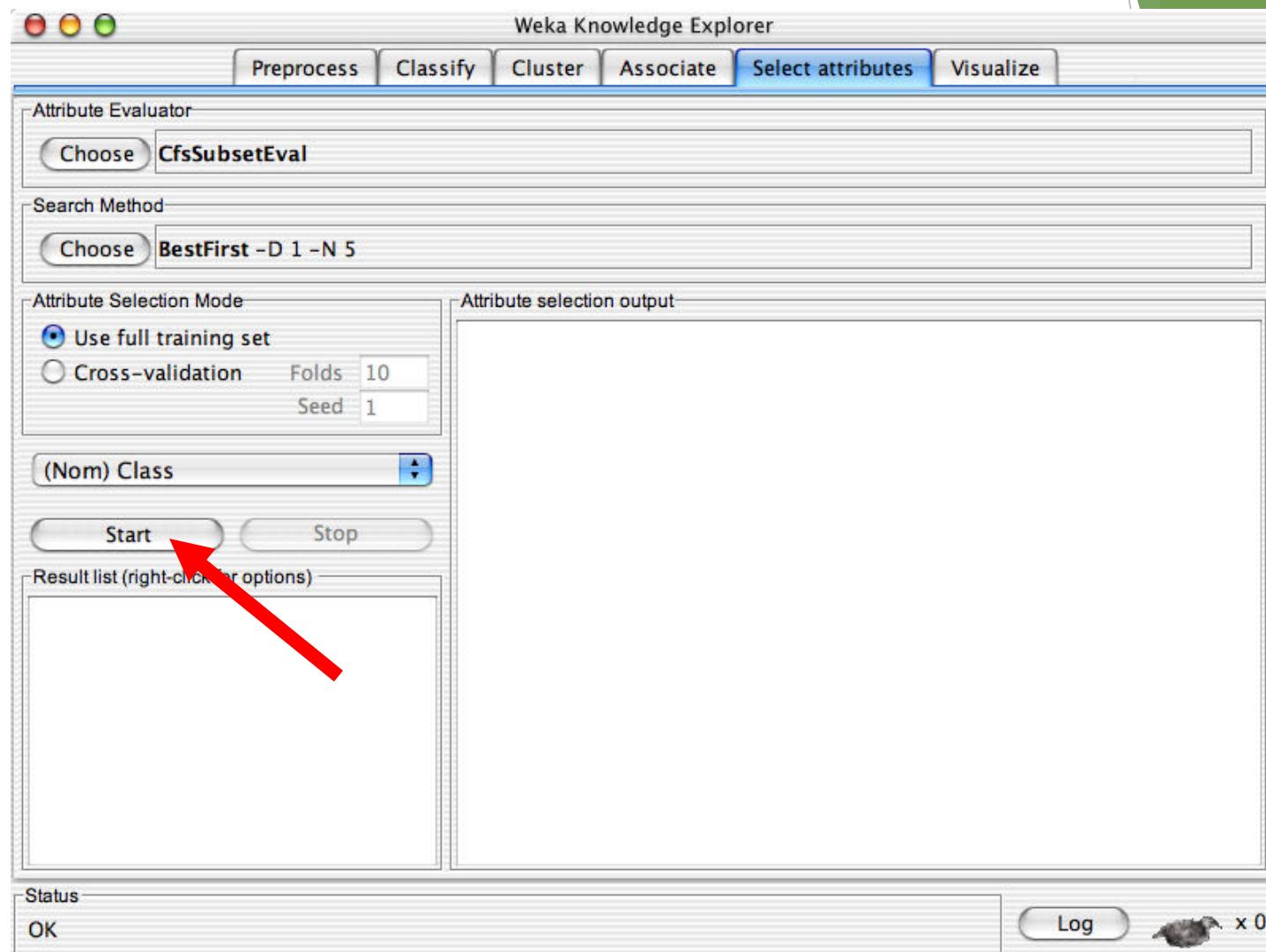
WEKA: the software

- ▶ Machine learning/data mining software written in Java (distributed under the GNU Public License)
- ▶ Used for research, education, and applications
- ▶ Complements “Data Mining” by Witten & Frank
- ▶ Main features:
 - ▶ Comprehensive set of data pre-processing tools, learning algorithms and evaluation methods
 - ▶ Graphical user interfaces (incl. data visualization)
 - ▶ Environment for comparing learning algorithms

Explorer: attribute selection

- ▶ Panel that can be used to investigate which (subsets of) attributes are the most predictive ones
- ▶ Attribute selection methods contain two parts:
 - ▶ A search method: best-first, forward selection, random, exhaustive, genetic algorithm, ranking
 - ▶ An evaluation method: correlation-based, wrapper, information gain, chi-squared, ...
- ▶ Very flexible: WEKA allows (almost) arbitrary combinations of these two





Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator
Choose **CfsSubsetEval**

Search Method
Choose **BestFirst -D 1 -N 5**

Attribute Selection Mode
 Use full training set
 Cross-validation Folds: 10
Seed: 1

(Nom) Class

Start Stop

Result list (right-click for options)
16:39:40 - BestFirst + CfsSubsetEval

Attribute selection output

```
duty-free-exports
export-administration-act-south-africa
Class
Evaluation mode: evaluate on all training data

==== Attribute Selection on all input data ===

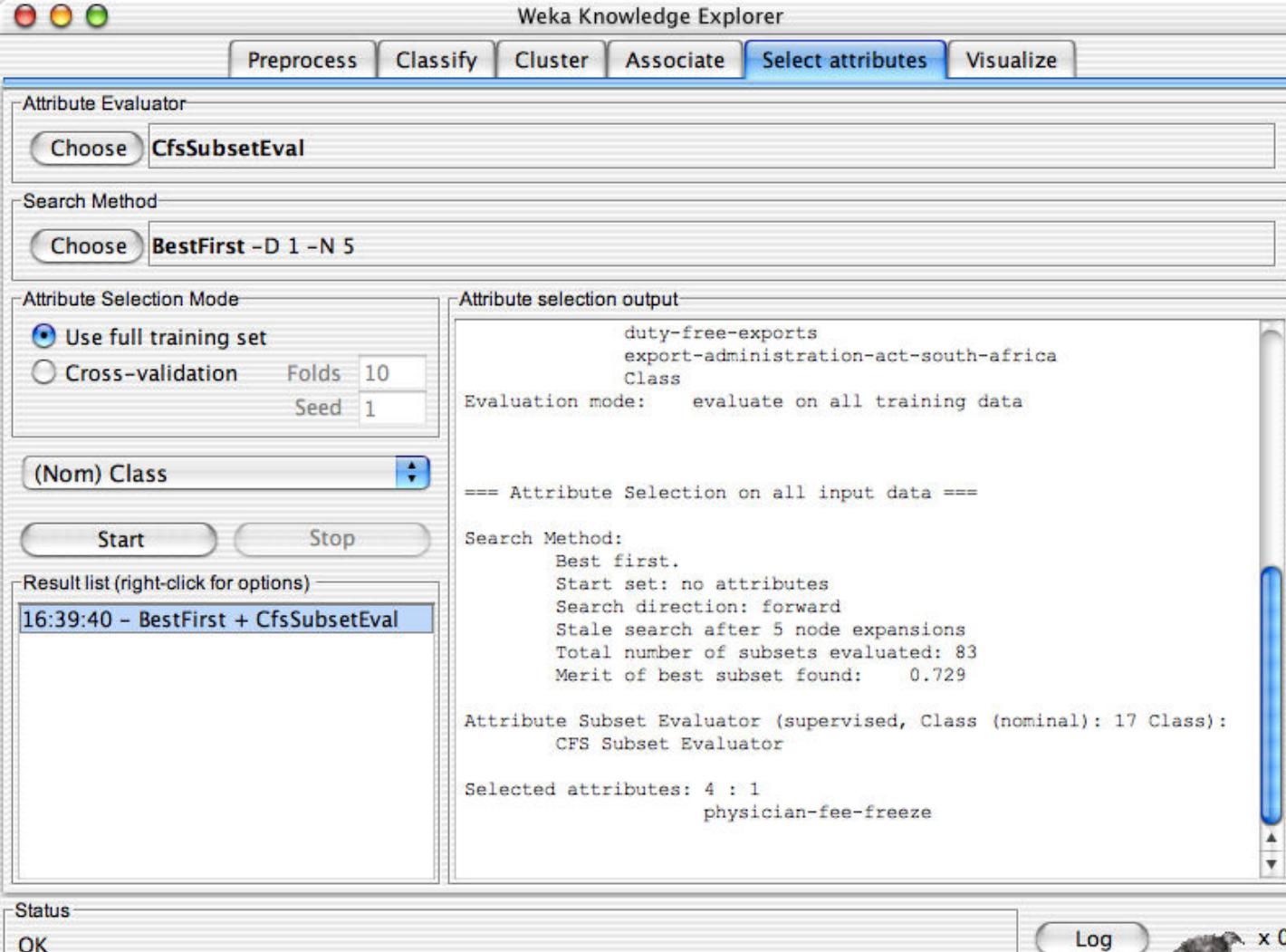
Search Method:
  Best first.
  Start set: no attributes
  Search direction: forward
  Stale search after 5 node expansions
  Total number of subsets evaluated: 83
  Merit of best subset found: 0.729

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
  CFS Subset Evaluator

Selected attributes: 4 : 1
  physician-fee-freeze
```

Status
OK

Log x 0



Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator

Choose **CfsSubsetEval**

Search Method

Choose **BestFirst -D 1 -N 5**

Attribute Selection Mode

Use full training set
 Cross-validation Folds: 10 Seed: 1

(Nom) Class

Start Stop

Result list (right-click for options)

16:39:40 - BestFirst + CfsSubsetEval

Attribute selection output

```
duty-free-exports
export-administration-act-south-africa
Class
Evaluation mode: evaluate on all training data

==== Attribute Selection on all input data ====

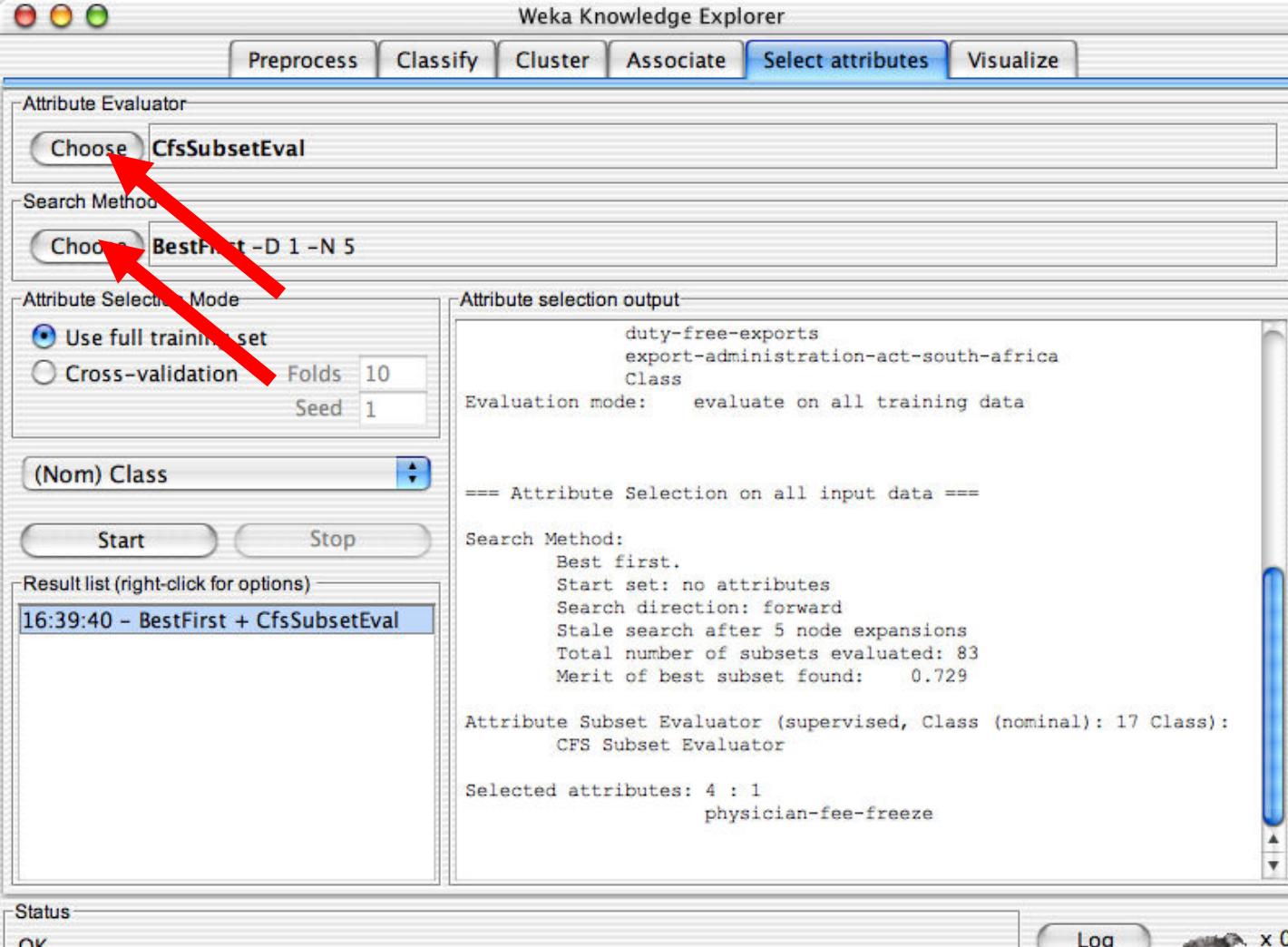
Search Method:
  Best first.
  Start set: no attributes
  Search direction: forward
  Stale search after 5 node expansions
  Total number of subsets evaluated: 83
  Merit of best subset found: 0.729

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
  CFS Subset Evaluator

Selected attributes: 4 : 1
  physician-fee-freeze
```

Status

OK Log x 0



Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator

weka
attributeSelection
CfsSubsetEval
ClassifierSubsetEval
WrapperSubsetEval
ConsistencySubsetEval
ReliefFAttributeEval
InfoGainAttributeEval
GainRatioAttributeEval
SymmetricalUncertAttributeEval
OneRAttributeEval
ChiSquaredAttributeEval
PrincipalComponents
SVMAttributeEval

Selected attributes: 4 : 1
physician-fee-freeze

duty-free-exports
export-administration-act-south-africa
Class
Evaluation mode: evaluate on all training data

Attribute Selection on all input data ===

Search Method:
Best first.
Start set: no attributes
Search direction: forward
Stale search after 5 node expansions
Total number of subsets evaluated: 83
Merit of best subset found: 0.729

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
CFS Subset Evaluator

Status OK Log x 0

This screenshot shows the Weka Knowledge Explorer interface. The 'Select attributes' tab is active. In the 'Attribute Evaluator' panel, the 'attributeSelection' class is selected, and 'CfsSubsetEval' is highlighted. The output window displays the results of the attribute selection process, including the selected attributes (duty-free-exports, export-administration-act-south-africa, Class), evaluation mode (evaluate on all training data), search method (Best first), and statistics about the search (83 subsets evaluated, Merit of best subset found: 0.729). The status bar at the bottom indicates 'OK'.

Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator

Choose **InfoGainAttributeEval**

Search Method

weka
attributeSelection
BestFirst
ForwardSelection
RaceSearch
GeneticSearch
RandomSearch
ExhaustiveSearch
Ranker
RankSearch

E308 - N - 1

Attribute selection output

```
duty-free-exports
export-administration-act-south-africa
Class
Evaluation mode: evaluate on all training data
```

Attribute Selection on all input data ===

Search Method:

```
Best first.
Start set: no attributes
Search direction: forward
Stale search after 5 node expansions
Total number of subsets evaluated: 83
Merit of best subset found: 0.729
```

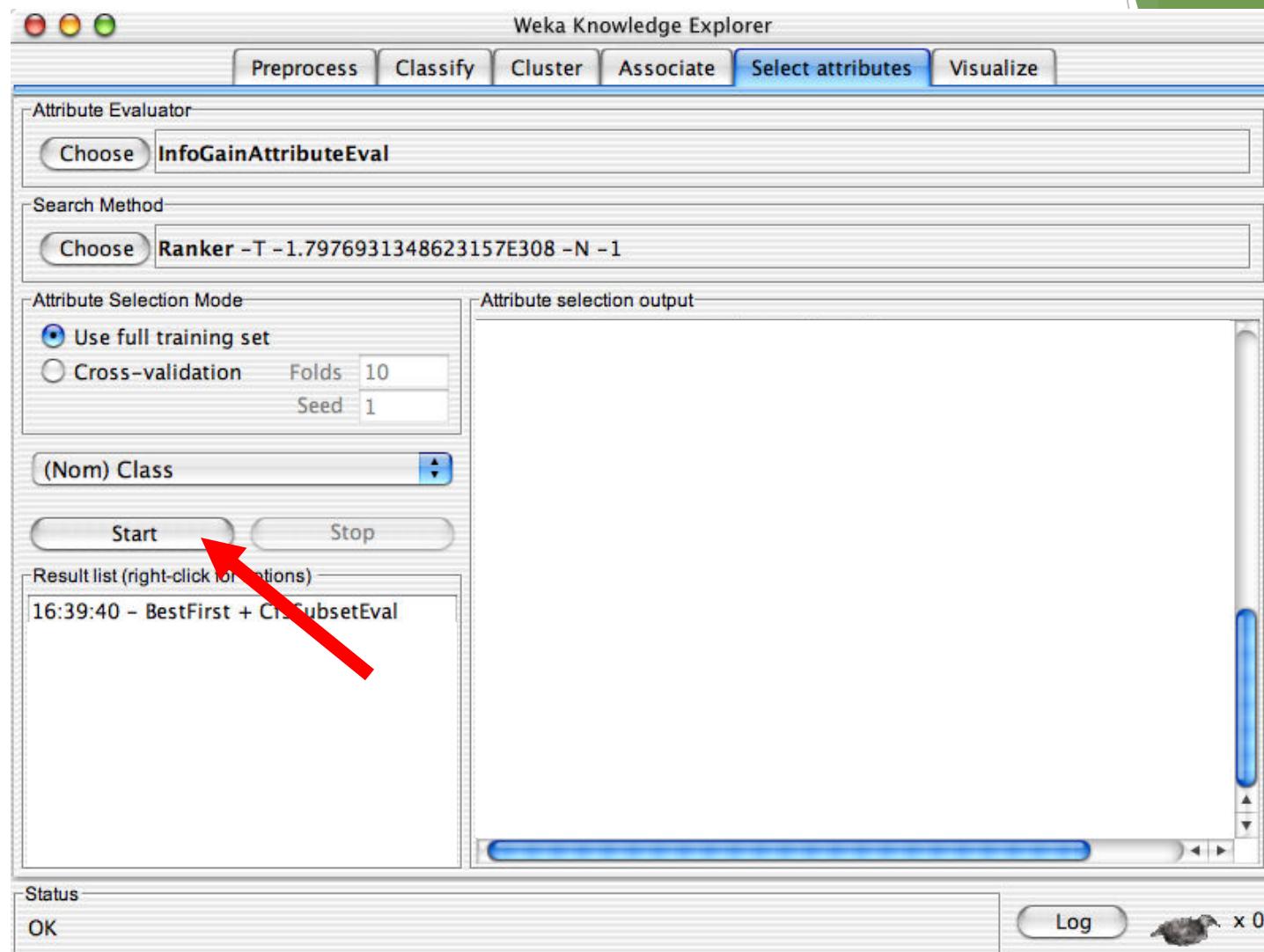
Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
CFS Subset Evaluator

Selected attributes: 4 : 1
physician-fee-freeze

Status

OK

Log x 0



Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator

Choose **InfoGainAttributeEval**

Search Method

Choose **Ranker -T -1.7976931348623157E308 -N -1**

Attribute Selection Mode

Use full training set
 Cross-validation Folds: 10 Seed: 1

(Nom) Class

Start Stop

Result list (right-click for options)

16:39:40 - BestFirst + CfsSubsetEval
16:43:05 - Ranker + InfoGainAttributeEval

Attribute selection output

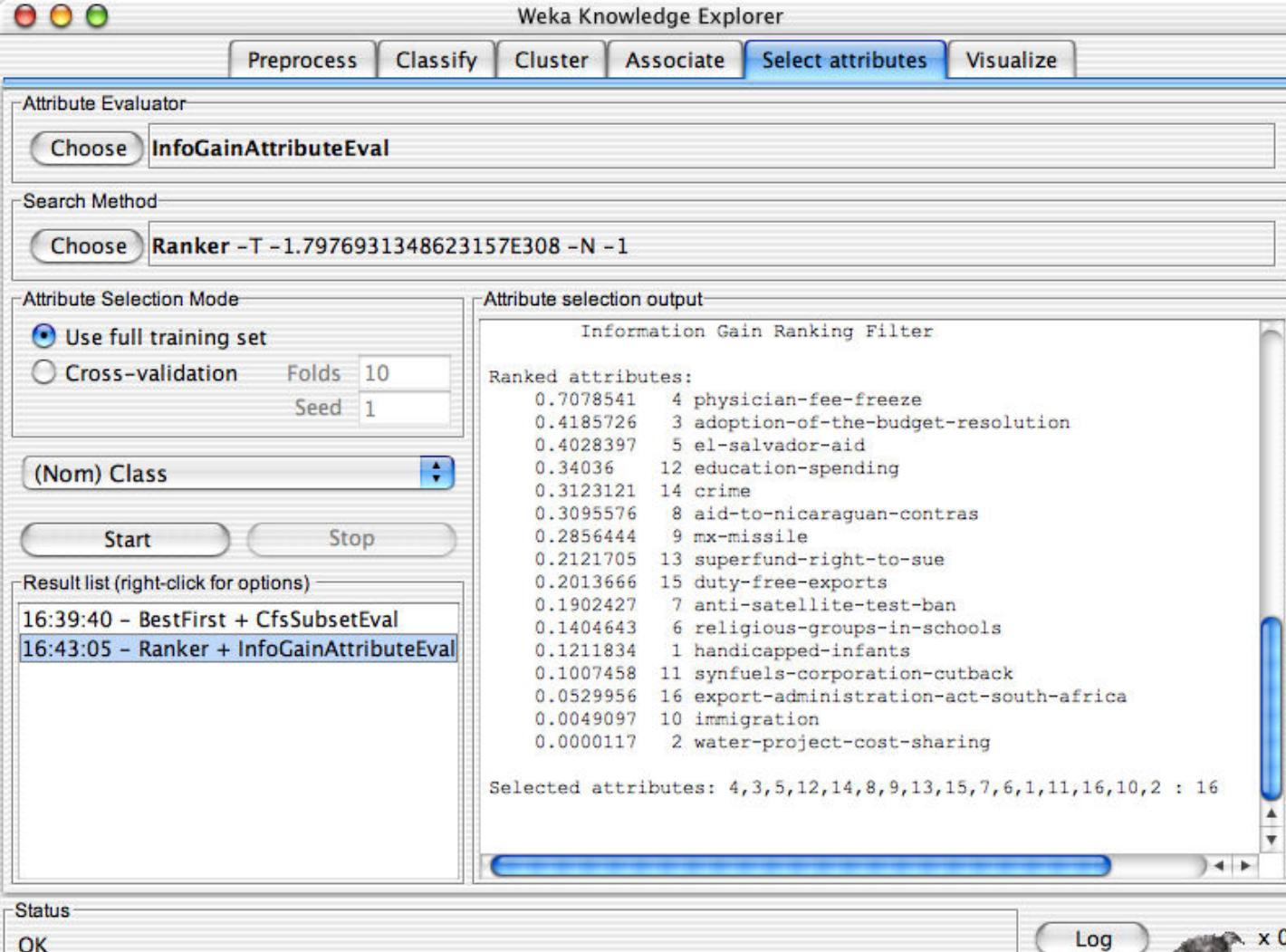
Information Gain Ranking Filter

Ranked attributes:

0.7078541	4 physician-fee-freeze
0.4185726	3 adoption-of-the-budget-resolution
0.4028397	5 el-salvador-aid
0.34036	12 education-spending
0.3123121	14 crime
0.3095576	8 aid-to-nicaraguan-contras
0.2856444	9 mx-missile
0.2121705	13 superfund-right-to-sue
0.2013666	15 duty-free-exports
0.1902427	7 anti-satellite-test-ban
0.1404643	6 religious-groups-in-schools
0.1211834	1 handicapped-infants
0.1007458	11 synfuels-corporation-cutback
0.0529956	16 export-administration-act-south-africa
0.0049097	10 immigration
0.0000117	2 water-project-cost-sharing

Selected attributes: 4,3,5,12,14,8,9,13,15,7,6,1,11,16,10,2 : 16

Status OK Log x 0



References

- ▶ <https://medium.datadriveninvestor.com/feature-selection-techniques-1a99e61da222>
- ▶ <https://www.mygreatlearning.com/blog/best-first-search-bfs/>
- ▶ https://en.wikipedia.org/wiki/Beam_search



Introduction



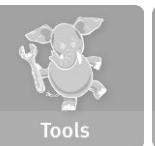
Analytics Lifecycle



Basic Methods



Adv. Methods



Tools



Lab

Module 4: Advanced Analytics – Theory and Methods

Lesson 5: Naïve Bayesian Classifiers

During this lesson the following topics are covered:

- Naïve Bayesian Classifier
- Theoretical foundations of the classifier
- Use cases
- Evaluating the effectiveness of the classifier
- The Reasons to Choose (+) and Cautions (-) with the use of the classifier

Classifiers

Where in the catalog should I place this product listing?
Is this email spam?
Is this politician Democrat/Republican/Green?

- Classification: assign labels to objects.
- Usually supervised: training set of pre-classified examples.
- Our examples:
 - ▶ Naïve Bayesian
 - ▶ Decision Trees
 - ▶ (and Logistic Regression)

Naïve Bayesian Classifier

- Determine the most probable class label for each object
 - ▶ Based on the observed object attributes
 - ▶ Naïvely assumed to be conditionally independent of each other
 - ▶ Example:
 - ▶ Based on the objects attributes {shape, color, weight}
 - ▶ A given object that is {spherical, yellow, < 60 grams},
may be classified (labeled) as a tennis ball
 - ▶ Class label probabilities are determined using Bayes' Law
- Input variables are discrete
- Output:
 - ▶ Probability score – proportional to the true probability
 - ▶ Class label – based on the highest probability score

Naïve Bayesian Classifier - Use Cases

- Preferred method for many text classification problems.
 - ▶ Try this first; if it doesn't work, try something more complicated
- Use cases
 - ▶ Spam filtering, other text classification tasks
 - ▶ Fraud detection



Technical Description - Bayes' Law

$$P(C | A) = \frac{P(A \cap C)}{P(A)} = \frac{P(A | C)P(C)}{P(A)}$$

- C is the class label:
 - ▶ $C \in \{C_1, C_2, \dots, C_n\}$
- A is the observed object attributes
 - ▶ $A = (a_1, a_2, \dots, a_m)$
- $P(C | A)$ is the probability of C given A is observed
 - ▶ Called the conditional probability



Reverend Thomas Bayes

Apply the Naïve Assumption and Remove a Constant

- For observed attributes $A = (a_1, a_2, \dots, a_m)$, we want to compute

$$P(C_i | A) = \frac{P(a_1, a_2, \dots, a_m | C_i)P(C_i)}{P(a_1, a_2, \dots, a_m)} \quad i = 1, 2, \dots, n$$

and assign the classifier, C_i , with the largest $P(C_i | A)$

- Two simplifications to the calculations
 - Apply naïve assumption - each a_j is conditionally independent of each other, then

$$P(a_1, a_2, \dots, a_m | C_i) = P(a_1 | C_i)P(a_2 | C_i) \cdots P(a_m | C_i) = \prod_{j=1}^m P(a_j | C_i)$$

- Denominator $P(a_1, a_2, \dots, a_m)$ is a constant and can be ignored

Building a Naïve Bayesian Classifier

- Applying the two simplifications

$$P(C_i | a_1, a_2, \dots, a_m) \propto \left(\prod_{j=1}^m P(a_j | C_i) \right) P(C_i) \quad i = 1, 2, \dots, n$$

- To build a Naïve Bayesian Classifier, collect the following statistics from the training data:

- ▶ $P(C_i)$ for all the class labels.
- ▶ $P(a_j | C_i)$ for all possible a_j and C_i
- ▶ Assign the classifier label, C_i , that maximizes the value of

$$\left(\prod_{j=1}^m P(a_j | C_i) \right) P(C_i) \quad i = 1, 2, \dots, n$$

Weather data set

Outlook	Temperature	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

Weather data, frequency according to class

	Outlook		Temp		Hum		Windy		Play	
	Y	N	Y	N	Y	N	Y	N	Y	N
sunny	2	3	hot	2	2	high	3	4	F	6
overc	4	0	mild	4	2	norm	6	1	T	3
rainy	3	2	cool	3	1					3
									9	5

Weather data, frequency according to class

Counts:

	Outlook		Temperature		Humidity		Windy		Play	
	Y	N	Y	N	Y	N	Y	N	Y	N
sunny	2	3	hot	2	2	high	3	4	F	6
overcast	4	0	mild	4	2	norm	6	1	T	3
rainy	3	2	cool	3	1					3

Relative frequencies: calculating probabilities 1

	Outlook		Temperature		Humidity		Windy		Play	
	Y	N	Y	N	Y	N	Y	N	Y	N
S	2/9	3/5	h	2/9	2/5	h	3/9	4/5	F	6/9
O	4/9	0/5	m	4/9	2/5	n	6/9	1/5	T	3/9
R	3/9	2/5	cl	3/9	1/5					3/9

Weather data, frequency according to class

Counts:

	Outlook		Temperature		Humidity		Windy		Play	
	Y	N	Y	N	Y	N	Y	N	Y	N
sunny	2	3	hot	2	2	high	3	4	F	6
overc	4	0	mild	4	2	norm	6	1	T	3
rainy	3	2	cool	3	1					3

Probability

	Y	N	Y	N	Y	N	Y	N	Y	N
s	2/9	3/5	h	2/9	2/5	h	3/9	4/5	F	6/9
o	4/9	0/5	m	4/9	2/5	n	6/9	1/5	T	3/9
r	3/9	2/5	cl	3/9	1/5					3/9

Weather data, frequency according to class

Counts:

	Outlook		Temperature		Humidity		Windy		Play	
	Y	N	Y	N	Y	N	Y	N	Y	N
sunny	2	3	hot	2	2	high	3	4	F	6
overc	4	0	mild	4	2	norm	6	1	T	3
rainy	3	2	cool	3	1					3

Probability

	Y	N	Y	N	Y	N	Y	N	Y	N
s	2/9	3/5	h	2/9	2/5	h	3/9	4/5	F	6/9
o	4/9	0/5	m	4/9	2/5	n	6/9	1/5	T	3/9
r	3/9	2/5	cl	3/9	1/5					3/9

We are given the following test example:

Outlook Temp. Humidity Windy **Play Y or N ?**
sunny cool high true

Weather example:

Play

$$P(O, T, H, W | \text{Play}) = P(O | \text{Play}) \cdot P(T | \text{Play}) \cdot P(H | \text{Play}) \cdot P(W | \text{Play})$$

	Outlook	Temp.	Humidity	Windy	
e ¹	sunny	cool	high	true	?

$$P(O = s | \text{play} = Y) = 2/9$$

$$P(O = s | \text{play} = N) = 3/5$$

$$P(T = c | \text{play} = Y) = 3/9$$

$$P(T = c | \text{play} = N) = 1/5$$

$$P(H = h | \text{play} = Y) = 3/9$$

$$P(O = s | \text{play} = N) = 4/5$$

$$P(W = t | \text{play} = Y) = 3/9$$

$$P(W = t | \text{play} = N) = 3/5$$

Weather example: solving our example

$$P(O, T, H, W | \text{Play}) = P(O | \text{Play}) \cdot P(T | \text{Play}) \cdot P(H | \text{Play}) \cdot P(W | \text{Play})$$

Weather example when play = Yes or No:

$$P(\text{Play} = Y | x) = P(\text{Play} = Y) \cdot [P(O = s | \text{Play} = Y) \cdot P(T = c | \text{Play} = Y) \cdot P(H = h | \text{Play} = Y) \cdot P(W = t | \text{Play} = Y)]$$

	Outlook	Temp.	Humidity	Windy	
e ¹	sunny	cool	high	true	Y/N?

$$P(\text{play} = Y) = 9/14$$

$$P(O = s | \text{play} = Y) = 2/9$$

$$P(T = c | \text{play} = Y) = 3/9$$

$$P(H = h | \text{play} = Y) = 3/9$$

$$P(W = t | \text{play} = Y) = 3/9$$

$$P(\text{play} = N) = 5/14$$

$$P(O = s | \text{play} = N) = 3/5$$

$$P(T = c | \text{play} = N) = 1/5$$

$$P(O = s | \text{play} = N) = 4/5$$

$$P(W = t | \text{play} = N) = 3/5$$

Play

Weather example when play = Yes:

$$P(Play=Y| x) = P(Play=Y) \cdot [P(O=s| Play=Y) \cdot P(T=c| Play=Y) \cdot P(H=h| Play=Y) \cdot P(W=t| Play=Y)]$$

	Outlook	Temp.	Humidity	Windy	
e ¹	sunny	cool	high	true	Y?

$$P(play = Y) = 9/14$$

$$P(O = s | play = Y) = 2/9$$

$$P(T = c | play = Y) = 3/9$$

$$P(H = h | play = Y) = 3/9$$

$$P(W = t | play = Y) = 3/9$$

$$PY = (9/14) * (2/9) * (3/9) * (3/9) * (3/9)$$

$$PY = 0.00529$$

Play

Weather example when play = No:

$$P(Play=N| x) = P(Play=N) \cdot [P(O=s| Play=N) \cdot P(T=c| Play=N) \cdot P(H=h| Play=N) \cdot P(W=t| Play=N)]$$

	Outlook	Temp.	Humidity	Windy	
e ¹	sunny	cool	high	true	N?

$$P(play = N) = 5/14$$

$$P(O = s | play = N) = 3/5$$

$$P(T = c | play = N) = 1/5$$

$$P(H = h | play = N) = 4/5$$

$$P(W = t | play = N) = 3/5$$

$$PN = (5/14) * (3/5) * (1/5) * (4/5) * (3/5)$$

$$PN = 0.02057143$$

Decision classify e as play = N

Building a Training Dataset to Predict Good or Bad Credit

- Predict the credit behavior of a credit card applicant from applicant's attributes:
 - Personal status
 - Job type
 - Housing type
 - Savings amount
- These are all categorical variables and are better suited to Naïve Bayesian Classifier than to logistic regression.

personal_status	job	housing	savings_status	credit_class
male single	skilled	own	no known savings	good
female div/dep/mar	skilled	own	<100	bad
male single	unskilled resident	own	<100	good
male single	skilled	for free	<100	good
male single	skilled	for free	<100	bad
male single	unskilled resident	for free	no known savings	good
male single	skilled	own	500<=X<1000	good
male single	high qualif/self emp/mgm	rent	<100	good
male div/sep	unskilled resident	own	>=1000	good
male mar/wid	high qualif/self emp/mgm	own	<100	bad
female div/dep/mar	skilled	rent	<100	bad
female div/dep/mar	skilled	rent	<100	bad
female div/dep/mar	skilled	own	<100	good
male single	unskilled resident	own	<100	bad
female div/dep/mar	skilled	rent	<100	good
female div/dep/mar	unskilled resident	own	100<=X<500	bad
male single	skilled	own	no known savings	good
male single	skilled	own	no known savings	good
female div/dep/mar	high qualif/self emp/mgm	for free	<100	bad
male single	skilled	own	500<=X<1000	good
male single	skilled	own	<100	good
male single	skilled	rent	500<=X<1000	good
male single	unskilled resident	rent	<100	good
male single	skilled	own	100<=X<500	good
male mar/wid	skilled	own	no known savings	good
male single	unskilled resident	own	<100	good
male mar/wid	unskilled resident	own	<100	good

Naïve Bayesian Classifiers for the Credit Example

- Class labels: {good, bad}
 - ▶ $P(\text{good}) = 0.7$
 - ▶ $P(\text{bad}) = 0.3$
- Conditional Probabilities
 - ▶ $P(\text{own}|\text{bad}) = 0.62$
 - ▶ $P(\text{own}|\text{good}) = 0.75$
 - ▶ $P(\text{rent}|\text{bad}) = 0.23$
 - ▶ $P(\text{rent}|\text{good}) = 0.14$
 - ▶ ... and so on

personal_status	job	housing	savings_status	credit_class
male single	skilled	own	no known savings	good
female div/dep/mar	skilled	own	<100	bad
male single	unskilled resident	own	<100	good
male single	skilled	for free	<100	good
male single	skilled	for free	<100	bad
male single	unskilled resident	for free	no known savings	good
male single	skilled	own	500<=X<1000	good
male single	high qualif/self emp/mgm	rent	<100	good
male div/sep	unskilled resident	own	>=1000	good
male mar/wid	high qualif/self emp/mgm	own	<100	bad
female div/dep/mar	skilled	rent	<100	bad
female div/dep/mar	skilled	rent	<100	bad
female div/dep/mar	skilled	own	<100	good
male single	unskilled resident	own	<100	bad
female div/dep/mar	skilled	rent	<100	good
female div/dep/mar	unskilled resident	own	100<=X<500	bad
male single	skilled	own	no known savings	good
male single	skilled	own	no known savings	good
female div/dep/mar	high qualif/self emp/mgm	for free	<100	bad
male single	skilled	own	500<=X<1000	good
male single	skilled	own	<100	good
male single	skilled	rent	500<=X<1000	good
male single	unskilled resident	rent	<100	good
male single	skilled	own	100<=X<500	good
male mar/wid	skilled	own	no known savings	good
male single	unskilled resident	own	<100	good
male mar/wid	unskilled resident	own	<100	good

Naïve Bayesian Classifier for a Particular Applicant

- Given applicant attributes of
 $A = \{\text{female single, owns home, self-employed, savings} > \$1000\}$
- Since $P(\text{good}|A) > P(\text{bad}|A)$, assign the applicant the label "good" credit

a_j	C_i	$P(a_j C_i)$
female single	good	0.28
female single	bad	0.36
own	good	0.75
own	bad	0.62
self emp	good	0.14
self emp	bad	0.17
savings>1K	good	0.06
savings>1K	bad	0.02

$$P(\text{good}|A) \sim (0.28 * 0.75 * 0.14 * 0.06) * 0.7 = 0.0012$$

$$P(\text{bad}|A) \sim (0.36 * 0.62 * 0.17 * 0.02) * 0.3 = 0.0002$$

Naïve Bayesian Implementation Considerations

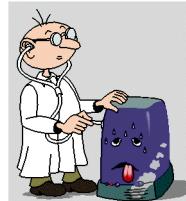
- Numerical underflow
 - ▶ Resulting from multiplying several probabilities near zero
 - ▶ Preventable by computing the logarithm of the products
- Zero probabilities due to unobserved attribute/classifier pairs
 - ▶ Resulting from rare events
 - ▶ Handled by smoothing (adjusting each probability by a small amount)
- Assign the classifier label, C_i , that maximizes the value of

$$\left(\sum_{j=1}^m \log P'(a_j | C_i) \right) + \log P(C_i)$$

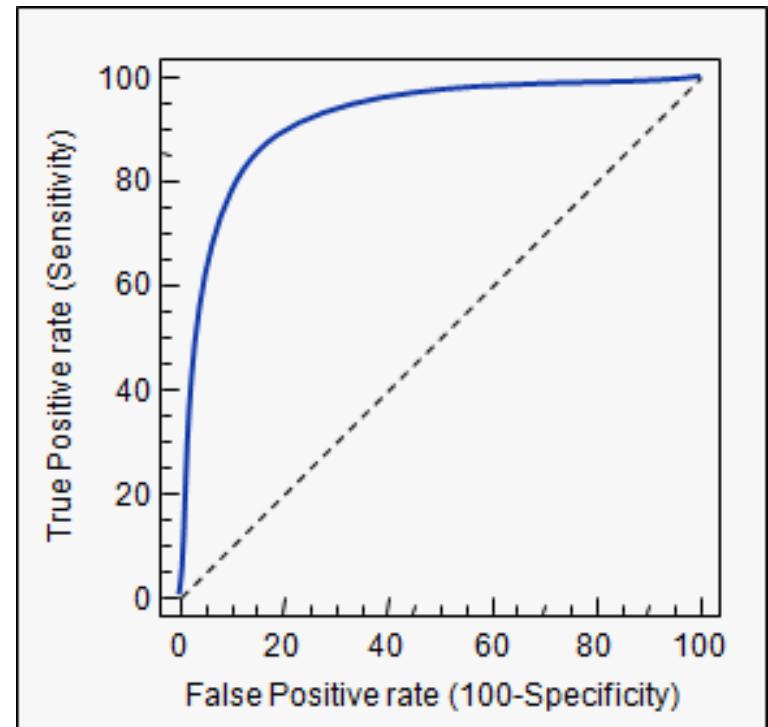
where $i = 1, 2, \dots, n$ and

P' denotes the adjusted probabilities

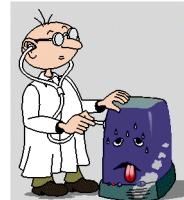
Diagnostics



- Hold-out data
 - ▶ How well does the model classify new instances?
- Cross-validation
- ROC curve/AUC



Diagnostics: Confusion Matrix



		Prediction				
		Actual Class	good	bad		
true positives (TP)	good	671	29	false negatives (FN)	700	
	bad	38	262			
false positives (FP)		709	291	true negatives (TN)		

Overall success rate (or accuracy):

$$(TP + TN) / (TP+TN+FP+FN) = (671+262)/1000 \approx 0.93$$

TPR: $TP / (TP + FN) = 671 / (671+29) = 671/700 \approx 0.96$

FPR: $FP / (FP + TN) = 38 / (38 + 262) = 38/300 \approx 0.13$

FNR: $FN / (TP + FN) = 29 / (671 + 29) = 29/700 \approx 0.04$

Precision: $TP / (TP + FP) = 671/709 \approx 0.95$

Recall (or TPR): $TP / (TP + FN) \approx 0.96$

Naïve Bayesian Classifier - Reasons to Choose (+) and Cautions (-)



Reasons to Choose (+)	Cautions (-)
Handles missing values quite well	Numeric variables have to be discrete (categorized) Intervals
Robust to irrelevant variables	Sensitive to correlated variables "Double-counting"
Easy to implement	Not good for estimating probabilities Stick to class label or yes/no
Easy to score data	
Resistant to over-fitting	
Computationally efficient Handles very high dimensional problems Handles categorical variables with a lot of levels	

Check Your Knowledge



Your Thoughts?

1. Consider the following Training Data Set:

- Apply the Naïve Bayesian Classifier to this data set and compute the probability score for $P(y = 1|X)$ for $X = (1,0,0)$

Show your work

Training Data Set

X1	X2	X3	Y
1	1	1	0
1	1	0	0
0	0	0	0
0	1	0	1
1	0	1	1
0	1	1	1

2. List some prominent use cases of the Naïve Bayesian Classifier.
3. What gives the Naïve Bayesian Classifier the advantage of being computationally inexpensive?
4. Why should we use log-likelihoods rather than pure probability values in the Naïve Bayesian Classifier?

Check Your Knowledge (Continued)



5. What is a confusion matrix and how it is used to evaluate the effectiveness of the model?
6. Consider the following data set with two input features temperature and season
 - What is the Naïve Bayesian assumption?
 - Is the Naïve Bayesian assumption satisfied for this problem?

Temperature	Season	Electricity Usage
-10 to 50 F	Winter	High
50 to 70 F	Winter	Low
70 to 85 F	Summer	Low
85 to 110 F	Summer	High



Module 4: Advanced Analytics – Theory and Methods

Lesson 5: Naïve Bayesian Classifiers - Summary

During this lesson the following topics were covered:

- Naïve Bayesian Classifier
- Theoretical foundations of the classifier
- Use cases
- Evaluating the effectiveness of the classifier
- The Reasons to Choose (+) and Cautions (-) with the use of the classifier

Lab Exercise 8: Naïve Bayesian Classifier

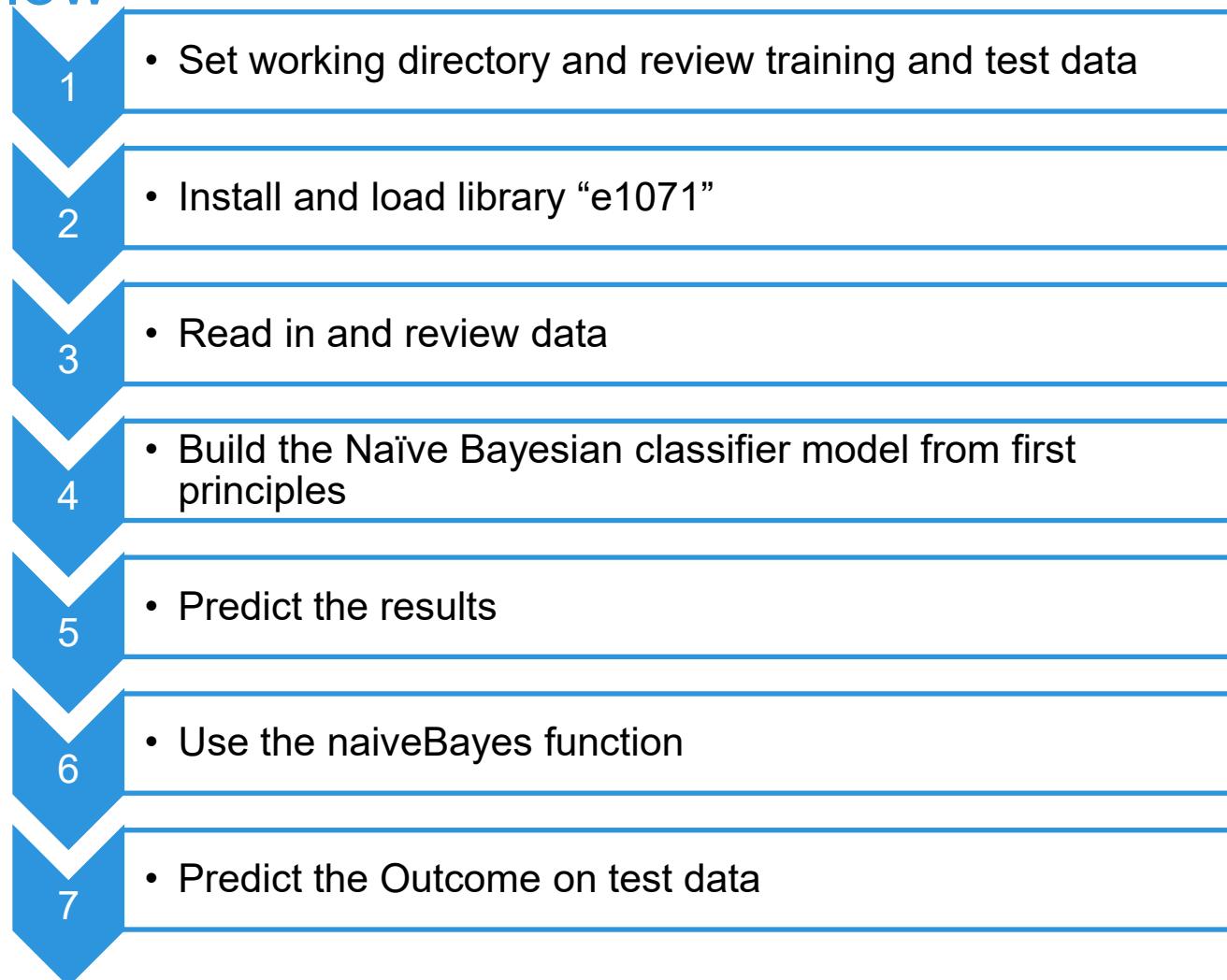


This Lab is designed to investigate and practice the Naïve Bayesian Classifier analytic technique.

After completing the tasks in this lab you should be able to:

- Use R functions for Naïve Bayesian Classification
- Apply the requirements for generating appropriate training data
- Validate the effectiveness of the Naïve Bayesian Classifier with the big data

Lab Exercise 8: Naïve Bayesian Classifier Part1 - Workflow



Naive Bayes in R

- Case Study
 - Titanic data set

Naive Bayes in R

- #Getting started with Naive Bayes
- #Install the package
- #install.packages("e1071")
- #Loading the library
- library(e1071)

Naive Bayes in R

?naiveBayes

#The documentation also contains an example implementation of Titanic dataset

Naive Bayes in R

```
# load the Titanic dataset
```

```
data("Titanic") , , Age = Child, Survived = No  
Sex
```

```
Titanic
```

	class	Male	Female
1st	0	0	
2nd	0	0	
3rd	35	17	
Crew	0	0	

```
, , Age = Adult, Survived = No  
Sex
```

	class	Male	Female
1st	118	4	
2nd	154	13	
3rd	387	89	
Crew	670	3	

Naive Bayes in R

#Save into a data frame and display it

```
Titanic_df=as.data.frame(Titanic)
```

```
Titanic_df
```

➤Titanic_df

	Class	Sex	Age	Survived	Freq
1	1st	Male	Child	No	0
2	2nd	Male	Child	No	0
3	3rd	Male	Child	No	35
4	Crew	Male	Child	No	0
5	1st	Female	Child	No	0
6	2nd	Female	Child	No	0
7	3rd	Female	Child	No	17

Naive Bayes in R

#Creating data from table

```
repeating_sequence=rep.int(seq_len(nrow(Titanic_df)), Titanic_df$Freq)
```

#This will repeat each combination equal to the frequency of each combination

repeating_sequence

Class	Sex	Age	Survived	Freq
3rd	Male	Child	No	35
3rd	Male	Child	No	35
3rd	Male	Child	No	35
3rd	Male	Child	No	35
3rd	Male	Child	No	35

...

35 times

Naive Bayes in R

```
#Create the dataset by row repetition created
```

```
Titanic_dataset=Titanic_df[repeating_sequence,]
```

```
Titanic_dataset
```

```
# no longer need the frequency, delete the attribute
```

```
Titanic_dataset$Freq=NULL
```

Remove this line

Class	Sex	Age	Survived
3rd	Male	Child	No
3rd	Male	Child	No
3rd	Male	Child	No
3rd	Male	Child	No
3rd	Male	Child	No
...			

...
35 times

Naive Bayes in R, apply the model

```
#apply the Naive Bayes model
```

```
Naive_Bayes_Model=naiveBayes(Survived ~., data=Titanic_dataset)
```

```
#Print the model outcomes
```

```
Naive_Bayes_Model
```

Naive Bayes in R, model outcomes

Naive Bayes Classifier for Discrete Predictors

Call:

```
naiveBayes.default(x = X, y = Y, laplace = laplace)
```

A-priori probabilities:

	No	Yes
	0.676965	0.323035

Conditional probabilities:

Class

	1st	2nd	3rd	Crew
No	0.08187919	0.11208054	0.35436242	0.45167785
Yes	0.28551336	0.16596343	0.25035162	0.29817159

Sex

	Male	Female
No	0.91543624	0.08456376
Yes	0.51617440	0.48382560

Age

	Child	Adult
No	0.03489933	0.96510067
Yes	0.08016878	0.91983122

Naive Bayes in R, predict the outcomes

```
#Prediction on the titanic dataset
```

```
NB_Predictions=predict(Naive_Bayes_Model,Titanic_dataset)
```

Naive Bayes in R, check accuracy/performance

#Confusion matrix to check model accuracy and performance

```
table(NB_Predictions,Titanic_dataset$Survived)
```

NB_Predictions	No	Yes
No	1364	362
Yes	126	349

Weka

Feras Al-Obeidat

WEKA: the bird



What is WEKA?

- ▶ **Waikato Environment for Knowledge Analysis**
 - ▶ It's a data mining/machine learning tool developed by Department of Computer Science, University of Waikato, New Zealand.
 - ▶ Weka is also a bird found only on the islands of New Zealand.



WEKA: the software

- Machine learning/data mining software written in Java (distributed under the GNU Public License)
- Used for research, education, and applications
- Complements “Data Mining” by Witten & Frank
- Main features:
 - ▶ Comprehensive set of data pre-processing tools, learning algorithms and evaluation methods
 - ▶ Graphical user interfaces (incl. data visualization)
 - ▶ Environment for comparing learning algorithms

Explorer: attribute selection

- Panel that can be used to investigate which (subsets of) attributes are the most predictive ones
- Attribute selection methods contain two parts:
 - ▶ A search method: best-first, forward selection, random, exhaustive, genetic algorithm, ranking
 - ▶ An evaluation method: correlation-based, wrapper, information gain, chi-squared, ...
- Very flexible: WEKA allows (almost) arbitrary combinations of these two

Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator
Choose **CfsSubsetEval**

Search Method
Choose **BestFirst -D 1 -N 5**

Attribute Selection Mode
 Use full training set
 Cross-validation Folds: 10
Seed: 1

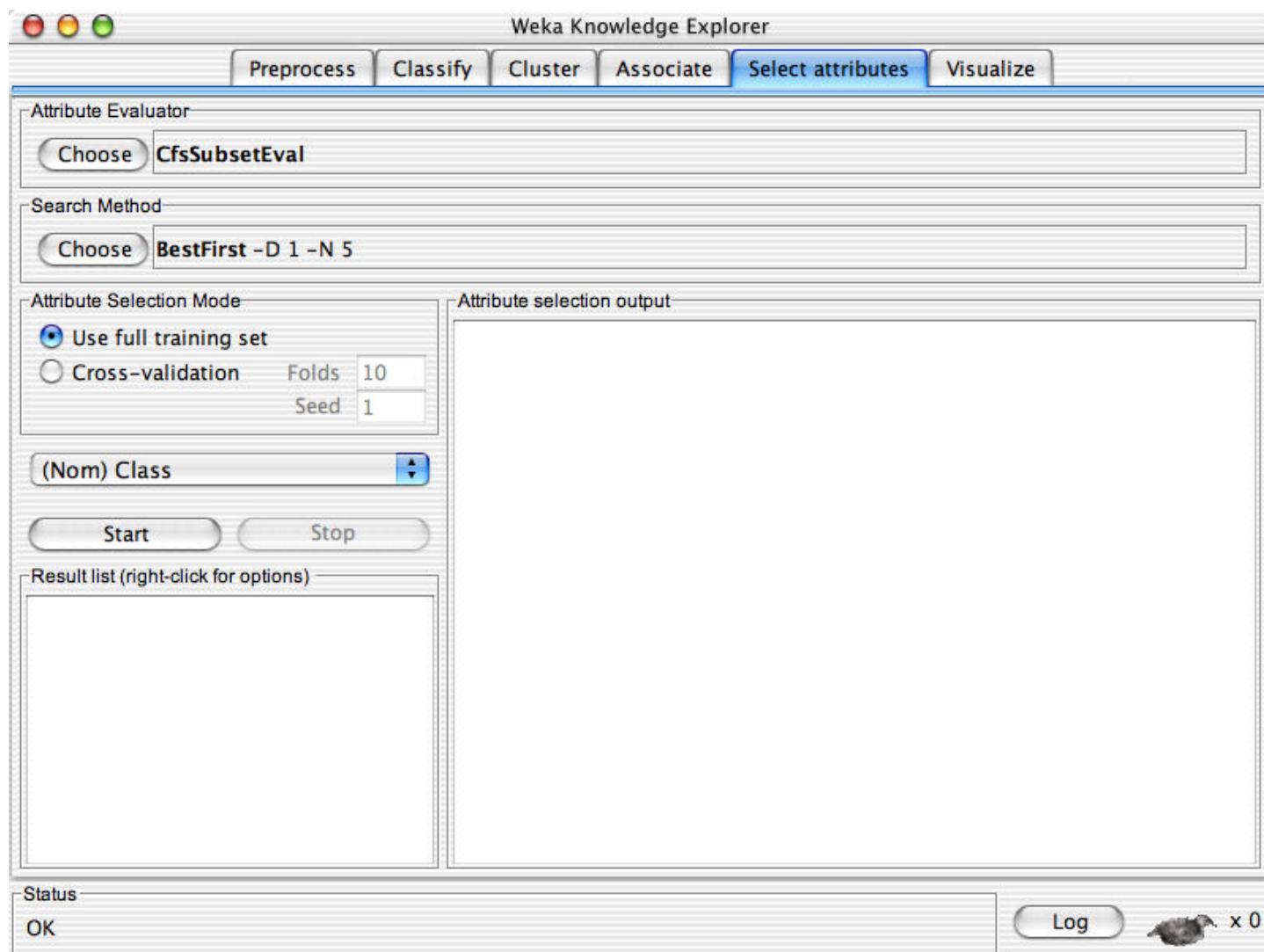
Attribute selection output

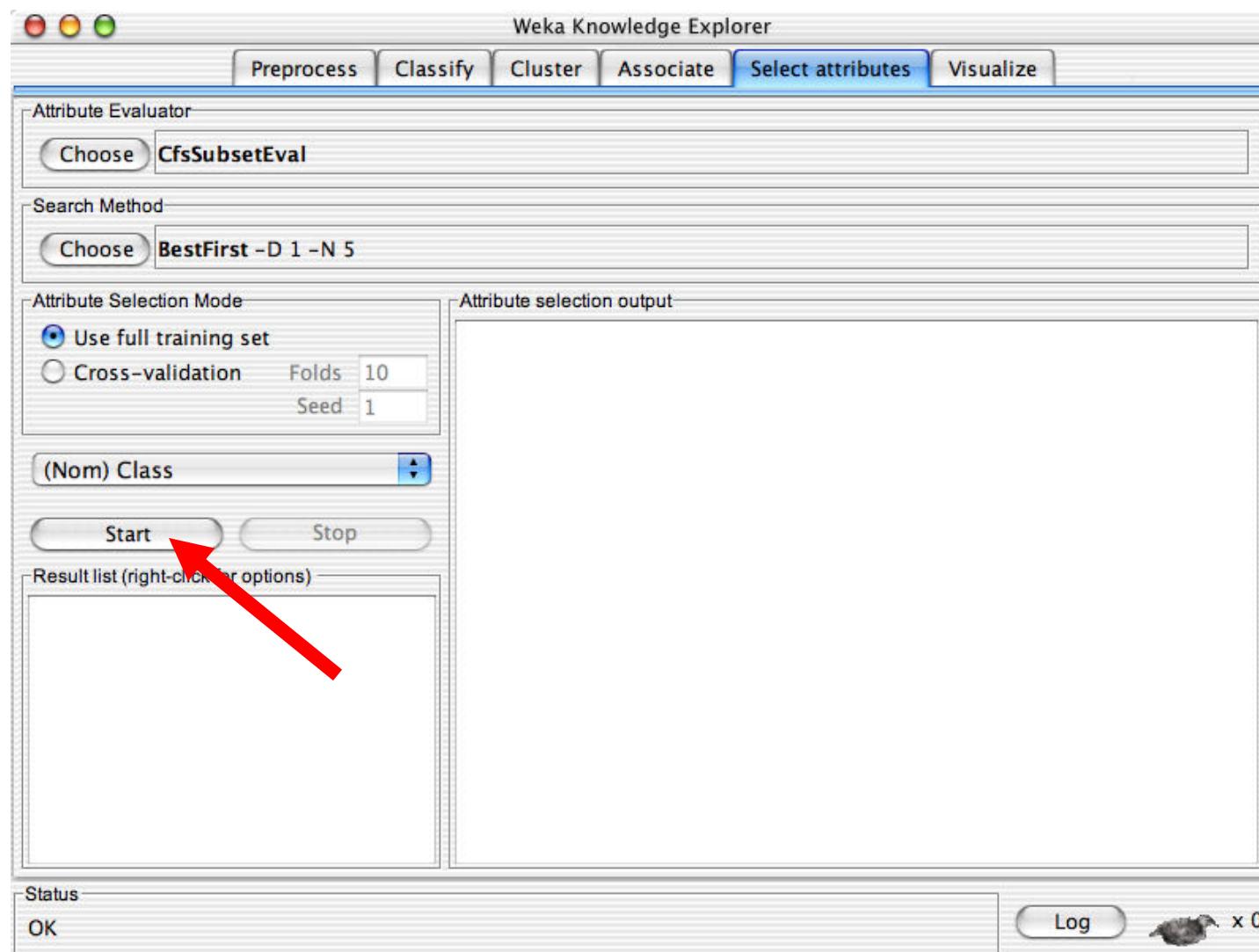
(Nom) Class

Start Stop

Result list (right-click for options)

Status
OK Log x 0





Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator
Choose **CfsSubsetEval**

Search Method
Choose **BestFirst -D 1 -N 5**

Attribute Selection Mode
 Use full training set
 Cross-validation Folds: 10
Seed: 1

(Nom) Class

Start Stop

Result list (right-click for options)
16:39:40 - BestFirst + CfsSubsetEval

Attribute selection output

```
duty-free-exports
export-administration-act-south-africa
Class
Evaluation mode: evaluate on all training data

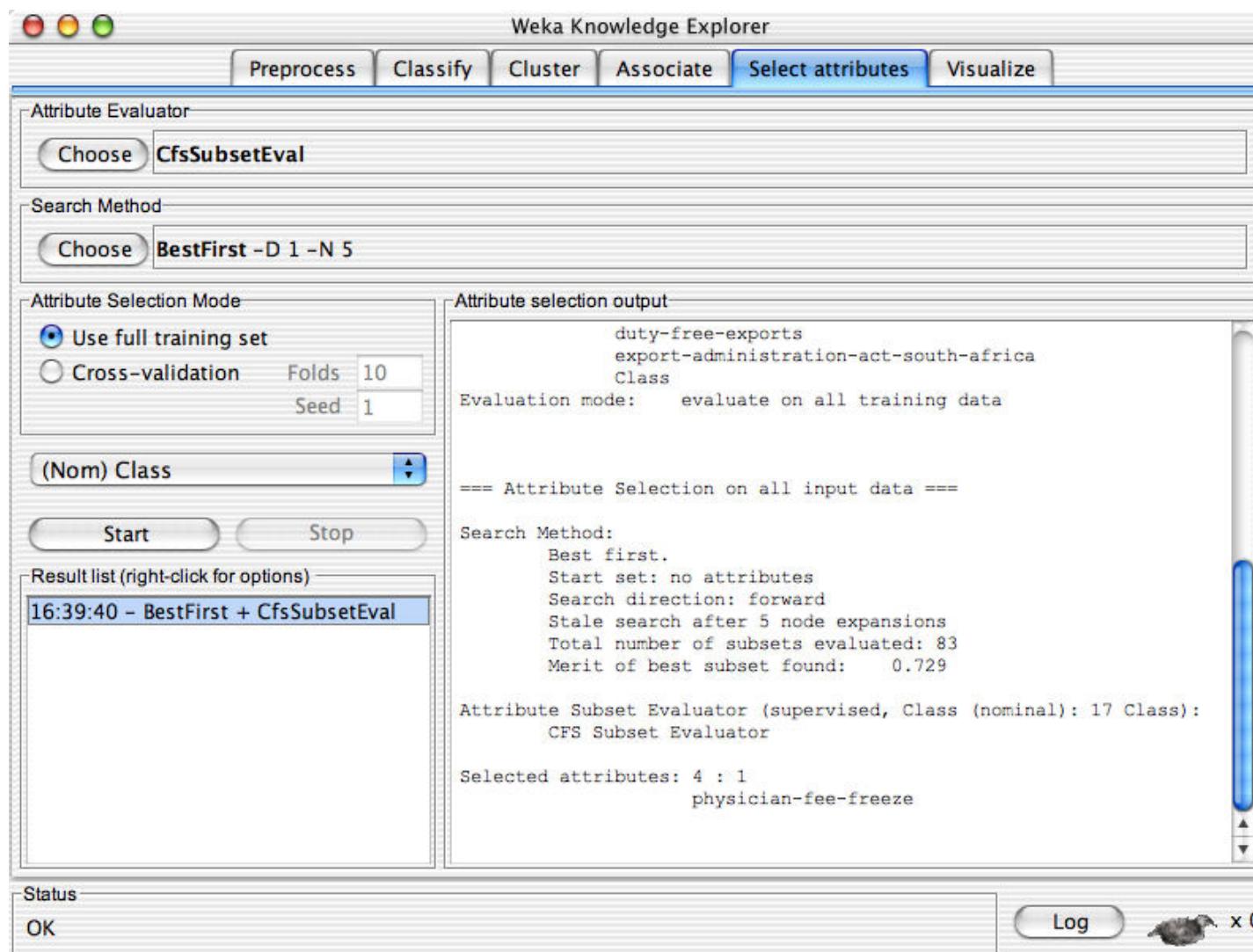
==== Attribute Selection on all input data ===

Search Method:
  Best first.
  Start set: no attributes
  Search direction: forward
  Stale search after 5 node expansions
  Total number of subsets evaluated: 83
  Merit of best subset found: 0.729

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
  CFS Subset Evaluator

Selected attributes: 4 : 1
  physician-fee-freeze
```

Status OK Log x 0



Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator

Choose **CfsSubsetEval**

Search Method

Choose **BestFirst -D 1 -N 5**

Attribute Selection Mode

Use full training set
 Cross-validation Folds: 10 Seed: 1

(Nom) Class

Start Stop

Result list (right-click for options)

16:39:40 - BestFirst + CfsSubsetEval

Attribute selection output

```
duty-free-exports
export-administration-act-south-africa
Class
Evaluation mode: evaluate on all training data

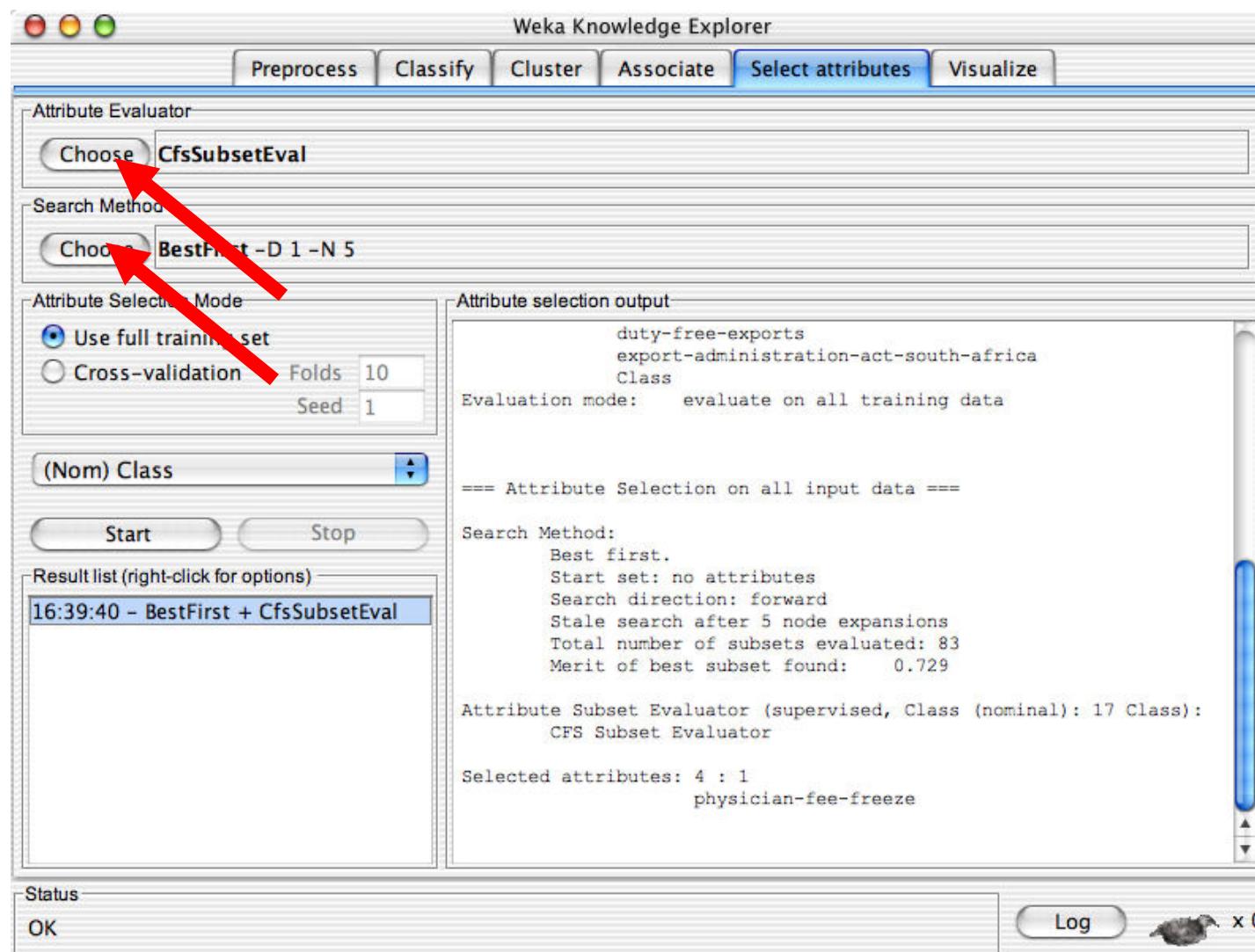
==== Attribute Selection on all input data ====
Search Method:
  Best first.
  Start set: no attributes
  Search direction: forward
  Stale search after 5 node expansions
  Total number of subsets evaluated: 83
  Merit of best subset found: 0.729

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
  CFS Subset Evaluator

Selected attributes: 4 : 1
  physician-fee-freeze
```

Status

OK Log x 0



A screenshot of the Weka Knowledge Explorer interface. The 'Select attributes' tab is active. In the 'Attribute Evaluator' section, 'CfsSubsetEval' is selected. In the 'Search Method' section, 'BestFirst -D 1 -N 5' is selected. Under 'Attribute Selection Mode', 'Use full training set' is checked. A red arrow points from the top-left towards the 'CfsSubsetEval' button. Another red arrow points from the bottom-left towards the 'BestFirst' button. On the right, the 'Attribute selection output' pane displays the results of the attribute selection process, including the selected attributes: 'duty-free-exports', 'export-administration-act-south-africa', 'Class', and 'physician-fee-freeze'. The status bar at the bottom shows 'OK' and 'Log x 0'.

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Attribute Evaluator

weka
attributeSelection
CfsSubsetEval
ClassifierSubsetEval
WrapperSubsetEval
ConsistencySubsetEval
ReliefFAttributeEval
InfoGainAttributeEval
GainRatioAttributeEval
SymmetricalUncertAttributeEval
OneRAttributeEval
ChiSquaredAttributeEval
PrincipalComponents
SVMAttributeEval

Attribute selection output:

```
duty-free-exports
export-administration-act-south-africa
Class
Evaluation mode: evaluate on all training data
```

Attribute Selection on all input data ===

Search Method:

```
Best first.
Start set: no attributes
Search direction: forward
Stale search after 5 node expansions
Total number of subsets evaluated: 83
Merit of best subset found: 0.729
```

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
CFS Subset Evaluator

Selected attributes: 4 : 1
physician-fee-freeze

Status: OK

Log x 0

Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator

Choose **InfoGainAttributeEval**

Search Method

- weka
- attributeSelection
 - BestFirst
 - ForwardSelection
 - RaceSearch
 - GeneticSearch
 - RandomSearch
 - ExhaustiveSearch
 - Ranker**
 - RankSearch

E308 - N - 1

Attribute selection output

```
duty-free-exports
export-administration-act-south-africa
Class
Evaluation mode: evaluate on all training data
```

Attribute Selection on all input data ===

Search Method:

```
Best first.
Start set: no attributes
Search direction: forward
Stale search after 5 node expansions
Total number of subsets evaluated: 83
Merit of best subset found: 0.729
```

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
CFS Subset Evaluator

Selected attributes: 4 : 1
physician-fee-freeze

Status

OK

Log x 0

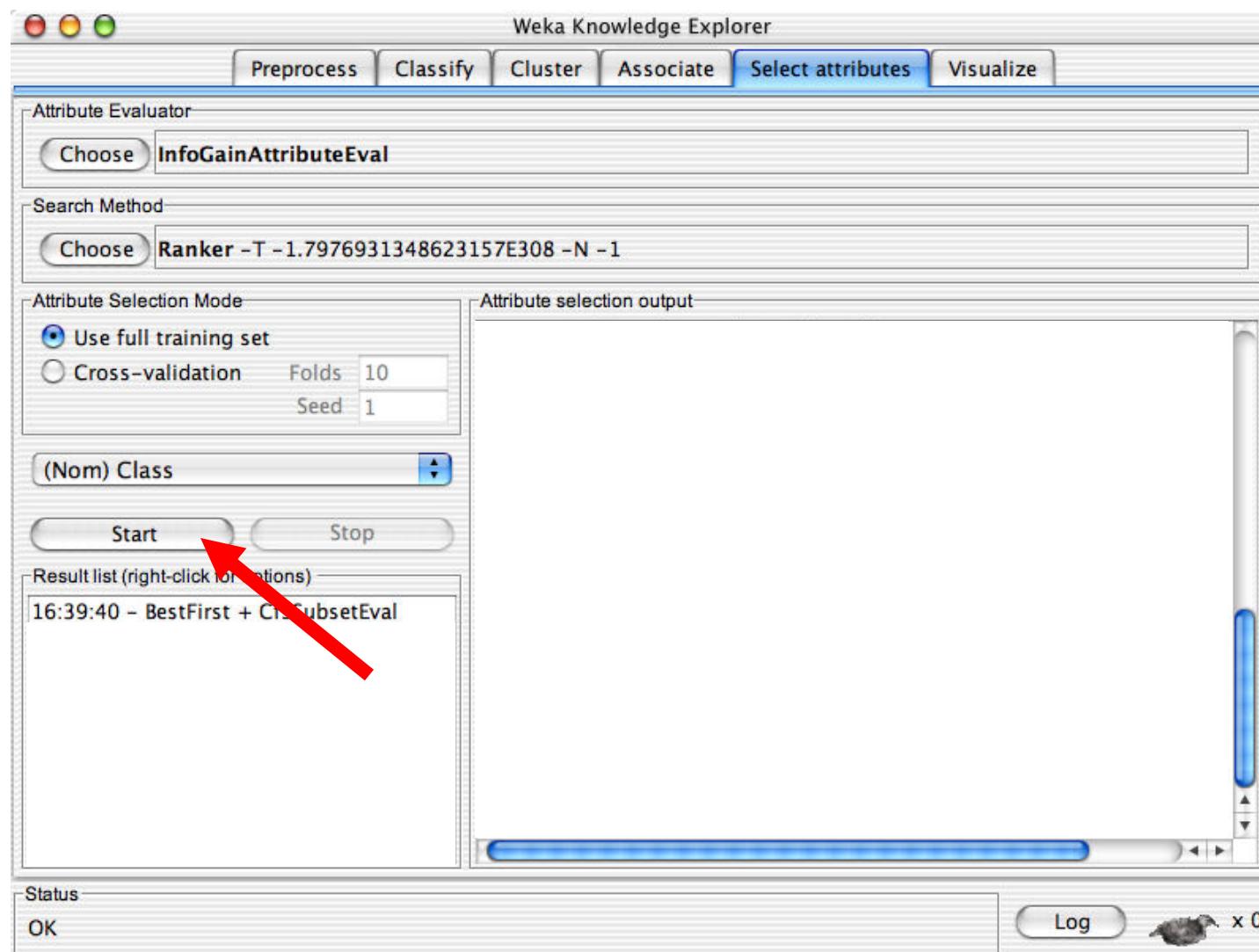
```
duty-free-exports
export-administration-act-south-africa
Class
Evaluation mode: evaluate on all training data

Attribute Selection on all input data ===

Search Method:
Best first.
Start set: no attributes
Search direction: forward
Stale search after 5 node expansions
Total number of subsets evaluated: 83
Merit of best subset found: 0.729

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
CFS Subset Evaluator

Selected attributes: 4 : 1
physician-fee-freeze
```



Weka Knowledge Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator

Choose **InfoGainAttributeEval**

Search Method

Choose **Ranker -T -1.7976931348623157E308 -N -1**

Attribute Selection Mode

Use full training set
 Cross-validation Folds: 10 Seed: 1

(Nom) Class

Start Stop

Result list (right-click for options)

16:39:40 - BestFirst + CfsSubsetEval
16:43:05 - Ranker + InfoGainAttributeEval

Attribute selection output

Information Gain Ranking Filter

Ranked attributes:

0.7078541	4 physician-fee-freeze
0.4185726	3 adoption-of-the-budget-resolution
0.4028397	5 el-salvador-aid
0.34036	12 education-spending
0.3123121	14 crime
0.3095576	8 aid-to-nicaraguan-contras
0.2856444	9 mx-missile
0.2121705	13 superfund-right-to-sue
0.2013666	15 duty-free-exports
0.1902427	7 anti-satellite-test-ban
0.1404643	6 religious-groups-in-schools
0.1211834	1 handicapped-infants
0.1007458	11 synfuels-corporation-cutback
0.0529956	16 export-administration-act-south-africa
0.0049097	10 immigration
0.0000117	2 water-project-cost-sharing

Selected attributes: 4,3,5,12,14,8,9,13,15,7,6,1,11,16,10,2 : 16

Status OK Log x 0

Explorer: data visualization

- Visualization very useful in practice: e.g. helps to determine difficulty of the learning problem
- WEKA can visualize single attributes (1-d) and pairs of attributes (2-d)
 - ▶ To do: rotating 3-d visualizations (Xgobi-style)
- Color-coded class values
- “Jitter” option to deal with nominal attributes (and to detect “hidden” data points)
- “Zoom-in” function

WEKA:: Introduction

- Created by researchers at the University of Waikato in New Zealand
- Java based
- A collection of open source ML algorithms
 - ▶ Pre-processing tools in WEKA are called “filters”
 - ▶ e.g, Discretization, normalization, resampling, attribute selection, transforming and combining attributes, ...
 - ▶ classifiers
 - ▶ clustering
 - ▶ association rule

WEKA:: Installation

- Download software from <http://www.cs.waikato.ac.nz/ml/weka/>
 - ▶ If you are interested in modifying/extending weka there is a developer version that includes the source code
- Set the weka environment variable for java
 - ▶ `setenv WEKAHOME /usr/local/weka/weka-3-0-2`
 - ▶ `setenv CLASSPATH $WEKAHOME/weka.jar:$CLASSPATH`
- Download some ML data from
<http://mlearn.ics.uci.edu/MLRepository.html>

The Explorer:

- ▶ Preprocess data
- ▶ Classification
- ▶ Clustering
- ▶ Association Rules
- ▶ Attribute Selection
- ▶ Data Visualization

WEKA:: Introduction .contd

- Routines are implemented as classes and logically arranged in packages
- Comes with an extensive GUI interface
 - ▶ Weka routines can be used stand alone via the command line
 - » Eg. `java weka.classifiers.j48.J48 -t $WEKAHOME/data/iris.arff`

WEKA:: Data format

- Uses flat text files to describe the data
- Can work with a wide variety of data files including its own “.arff” format
- Data can be imported from a file in various formats:
 - ▶ ARFF, CSV,
- Data can also be read from a URL or from an SQL database (using JDBC)

WEKA:: ARRF file format

```
@relation heart-disease-simplified
@attribute age numeric
@attribute sex { female, male}
@attribute chest_pain_type { typ_angina, asympt, non_anginal, atyp_angina}
@attribute cholesterol numeric
@attribute exercise_induced_angina { no, yes}
@attribute class { present, not_present}

@data
63,male,typ_angina,233,no,not_present
67,male,asympt,286,yes,present
67,male,asympt,229,yes,present
38,female,non_anginal,?,no,not_present
...
```

numeric attribute

nominal attribute

A more thorough description is available here
<http://www.cs.waikato.ac.nz/~ml/weka/arff.html>

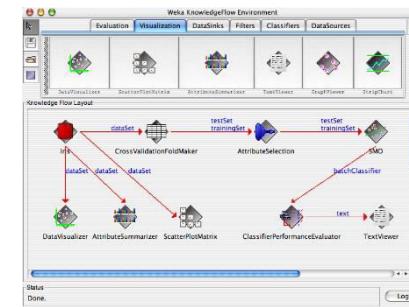
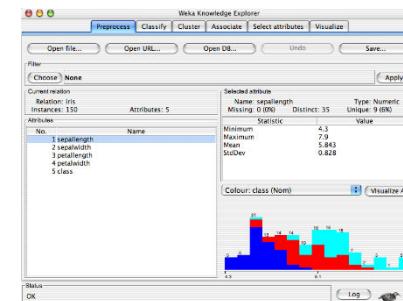
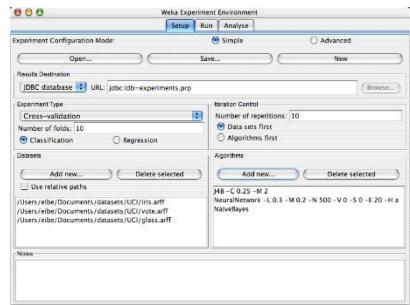
WEKA:: Explorer: Preprocessing

- Pre-processing tools in WEKA are called “filters”
- WEKA contains filters for:
 - ▶ Discretization, normalization, resampling, **attribute selection**, transforming, combining attributes, etc

```
Welcome to the WEKA SimpleCLI
Enter commands in the textfield at the bottom of
the window and press up and down arrows to move
through previous commands.

> help

Command must be one of:
java <classname> <args>
load
kill
cls
exit
help <command>
```



Weka Knowledge Explorer

[Preprocess](#)[Classify](#)[Cluster](#)[Associate](#)[Select attributes](#)[Visualize](#)[Open file...](#)[Open URL...](#)[Open DB...](#)[Undo](#)[Save...](#)

Filter

[Choose](#) **None**[Apply](#)

Current relation

Relation: None
Instances: None

Attributes: None

Selected attribute

Name: None
Missing: None Type: None
Distinct: None Unique: None

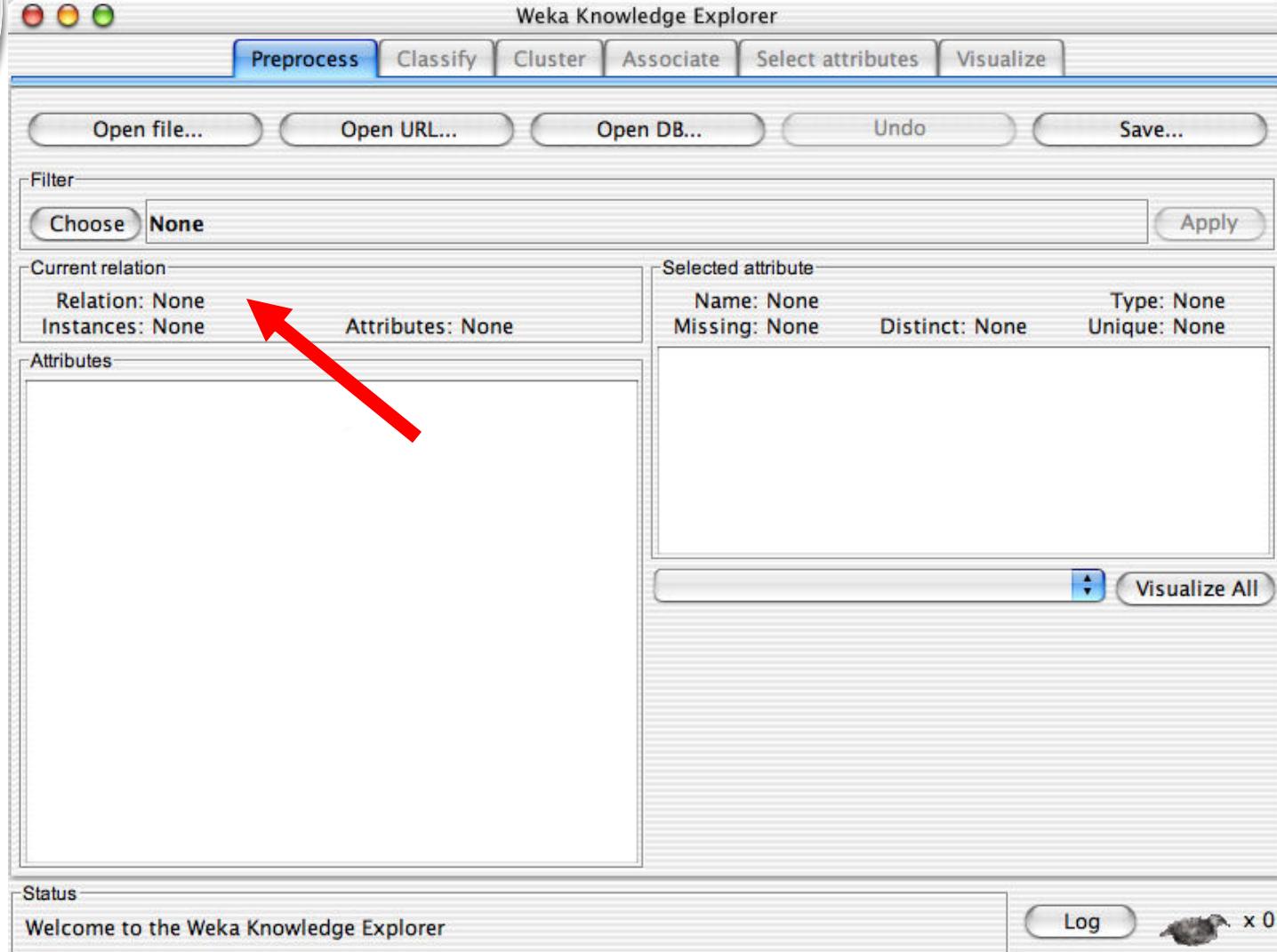
Attributes

[Visualize All](#)

Status

Welcome to the Weka Knowledge Explorer

[Log](#)



Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

Open file...

Open URL...

Open DB...

Undo

Save...

Filter

Choose None

Apply

Current relation

Relation: iris

Instances: 150

Attributes: 5

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute

Name: sepallength

Type: Numeric

Missing: 0 (0%)

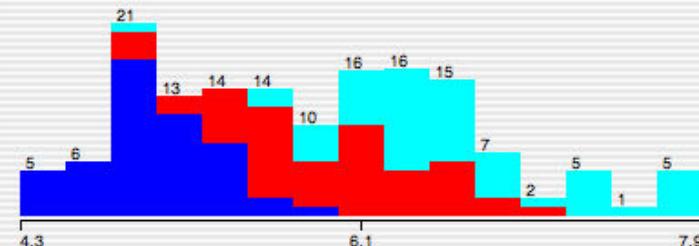
Distinct: 35

Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Colour: class (Nom)

Visualize All

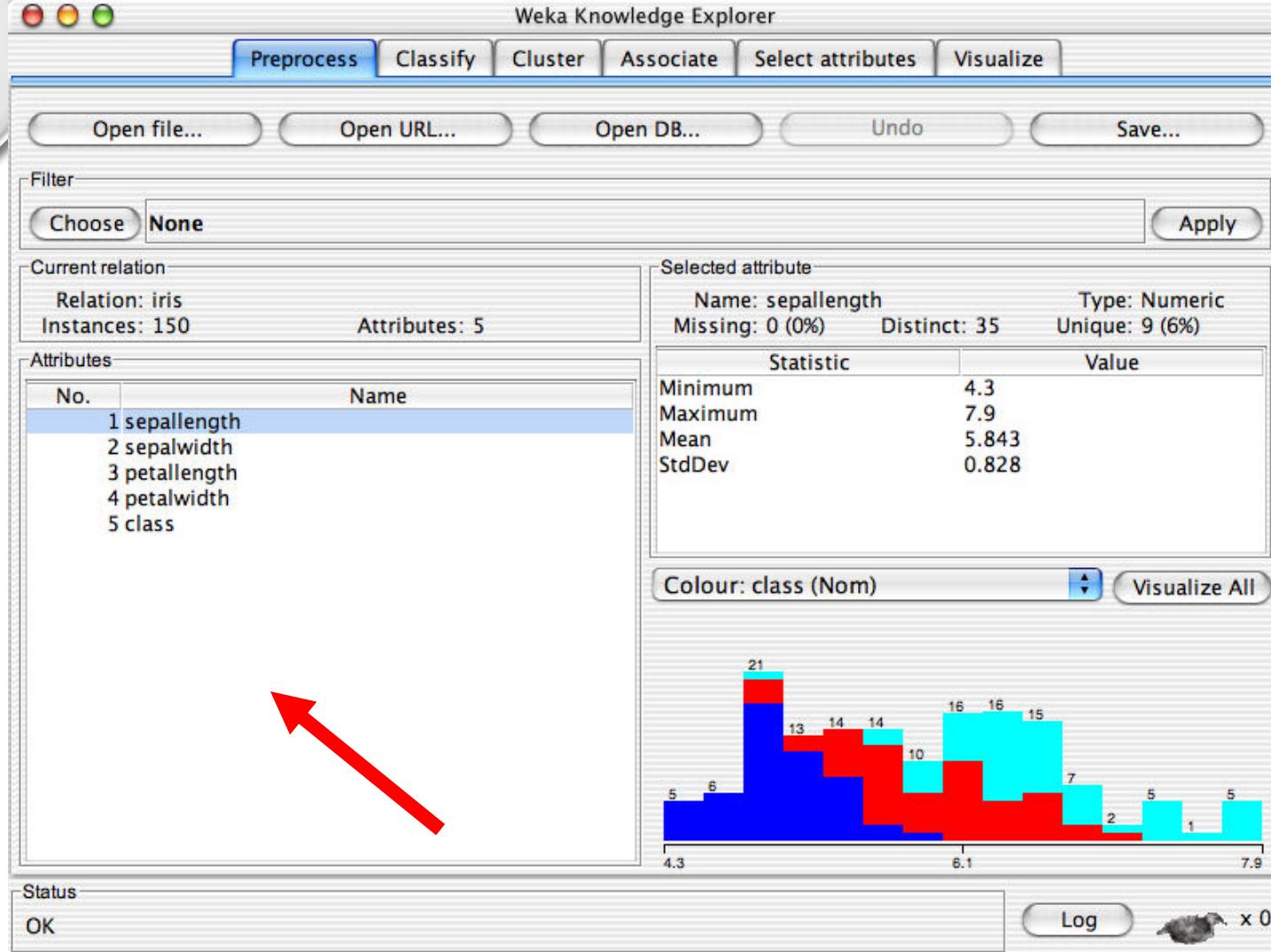


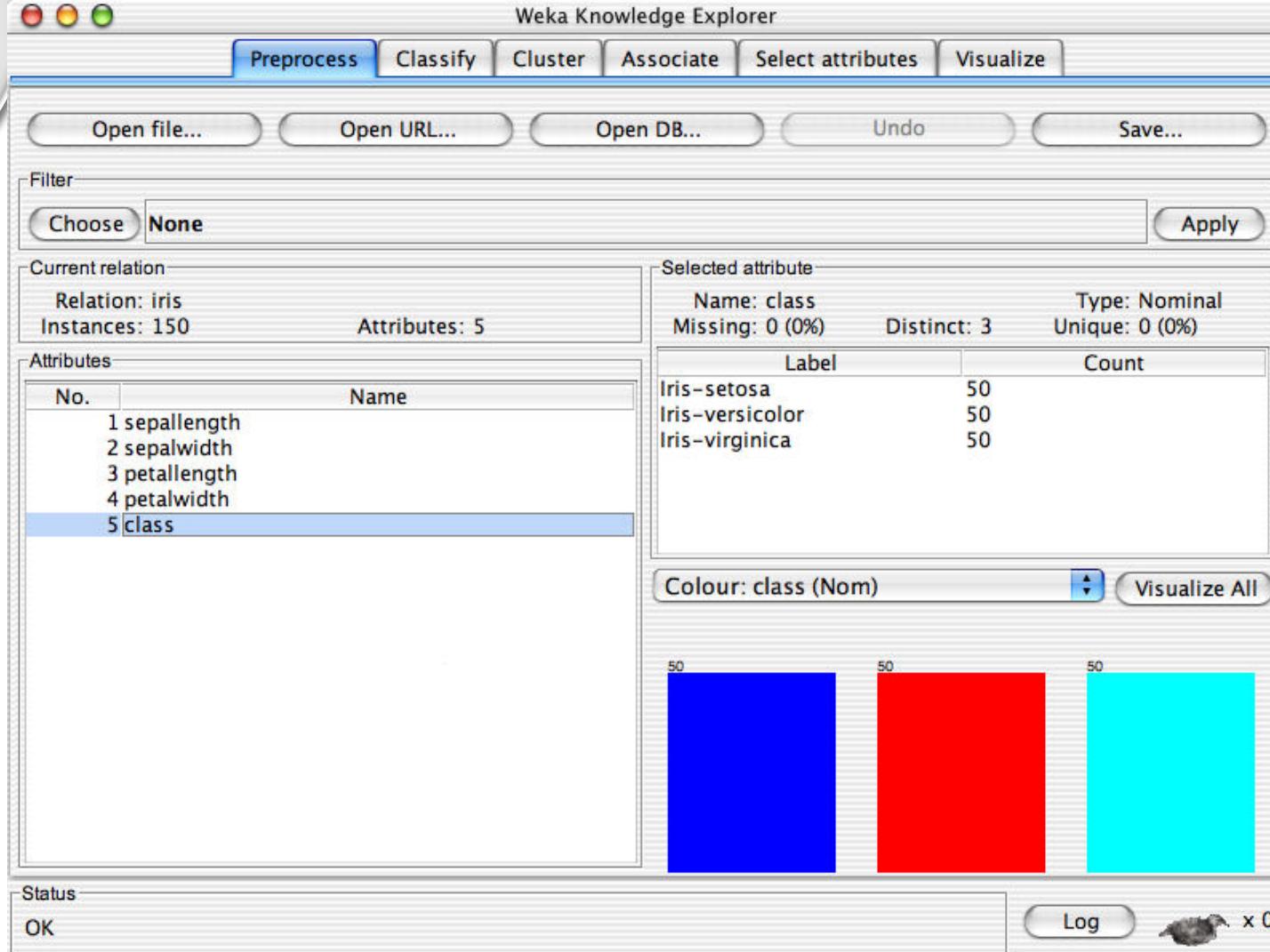
Status

OK

Log







[Open file...](#)[Open URL...](#)[Open DB...](#)[Undo](#)[Save...](#)

Filter

[Choose](#) [None](#)[Apply](#)

Current relation

Relation: iris

Instances: 150

Attributes: 5

Selected attribute

Name: class

Missing: 0 (0%)

Type: Nominal

Distinct: 3

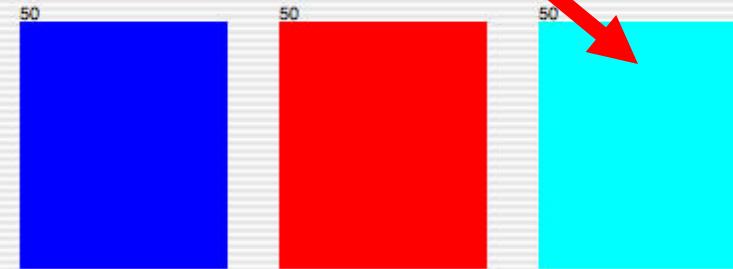
Unique: 0 (0%)

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Label	Count
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

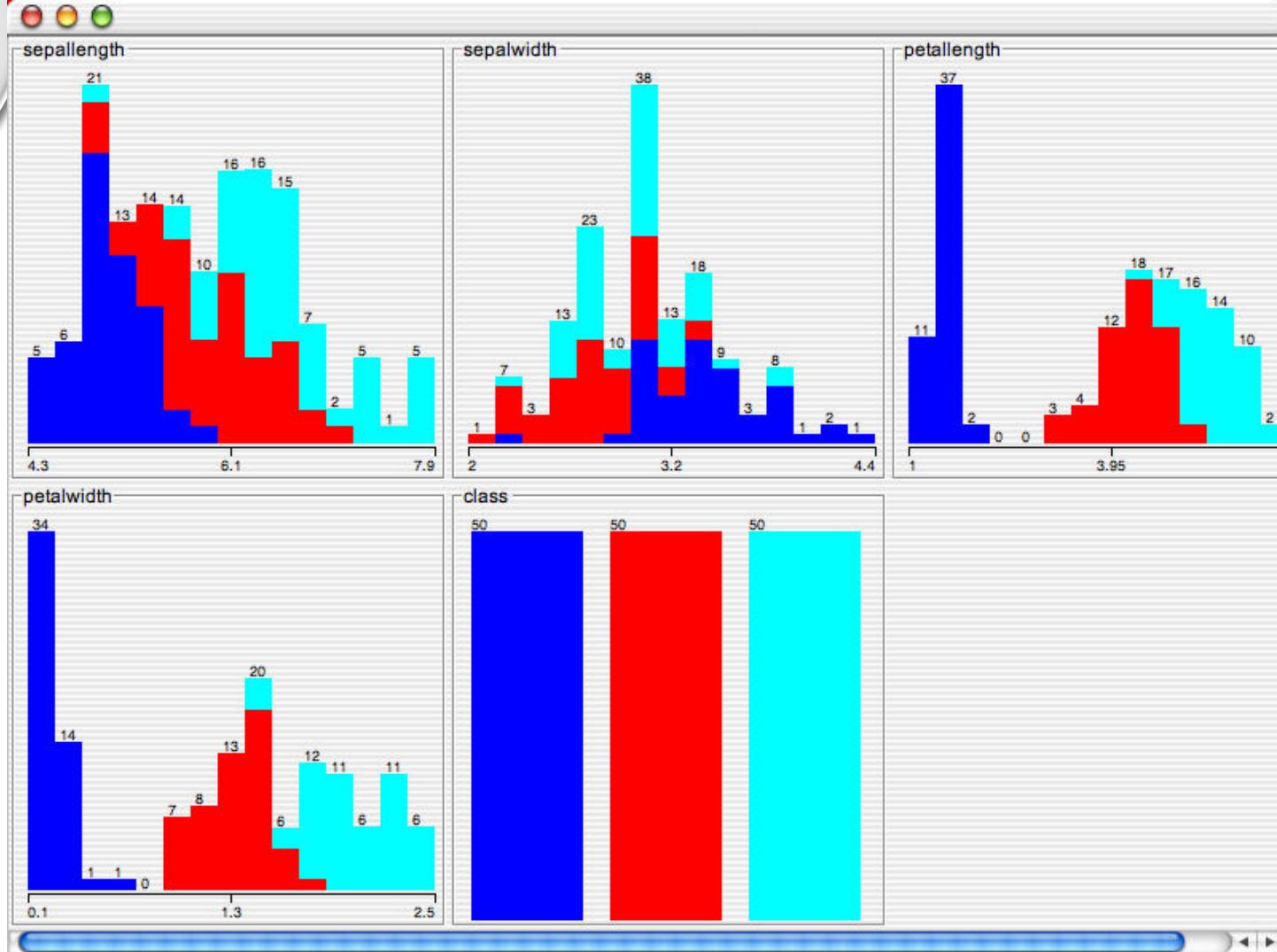
Colour: class (Nom)

[Visualize All](#)

Status

OK

[Log](#)



[Preprocess](#)[Classify](#)[Cluster](#)[Associate](#)[Select attributes](#)[Visualize](#)[Open file...](#)[Open URL...](#)[Open DB...](#)[Undo](#)[Save...](#)

Filter

Choose **None**[Apply](#)

Current relation

Relation: iris

Instances: 150

Attributes: 5

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute

Name: petallength

Missing: 0 (0%)

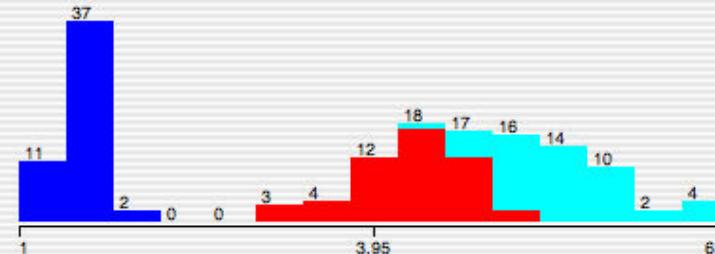
Type: Numeric

Distinct: 43

Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom)

[Visualize All](#)

Status

OK

[Log](#)

Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

Open file...

Open URL...

Open DB...

Undo

Save...

Filter

- weka
- filters
 - unsupervised
 - attribute
 - instance

Apply

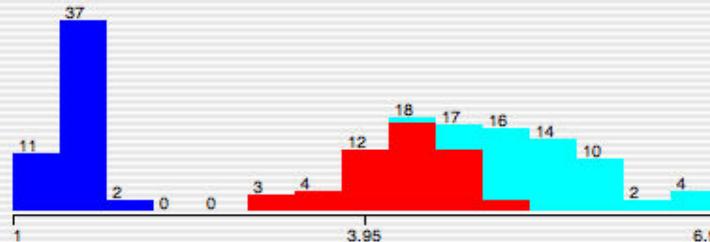
Selected attribute

Name: petallength Type: Numeric
Missing: 0 (0%) Distinct: 43 Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom)

Visualize All



Status

OK

Log



Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

Open file...

Open URL...

Open DB...

Undo

Save...

Filter

- weka
- filters
 - unsupervised
 - attribute
 - instance

Apply

Selected attribute

Name: petallength

Type: Numeric

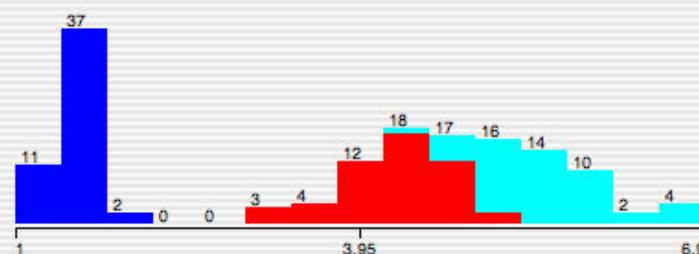
Missing: 0 (0%) Distinct: 43

Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom)

Visualize All



Status

OK

Log



x 0

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Undo Save...

Filter

weka

filters

unsupervised

attribute

- Add
- AddCluster
- AddExpression
- AddNoise
- Copy
- Discretize**
- FirstOrder
- MakeIndicator
- MergeTwoValues
- NominalToBinary
- Normalize
- NumericToBinary
- NumericTransform
- Obfuscate
- PKIDiscretize
- Remove
- RemoveType

Selected attribute

Name: petallength Type: Numeric

Missing: 0 (0%) Distinct: 43 Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom) Visualize All

A histogram showing the distribution of petallength for four classes. The x-axis ranges from 1 to 6.9. The y-axis shows the frequency of each bin. The classes are represented by different colors: blue (1), red (2), cyan (3), and green (4). The distribution is roughly bell-shaped, centered around 3.759.

Bin Range	Class	Frequency
[1.0, 1.4)	1	11
[1.4, 1.8)	1	37
[1.8, 2.2)	1	2
[2.2, 2.6)	1	0
[2.6, 3.0)	1	0
[3.0, 3.4)	2	3
[3.4, 3.8)	2	4
[3.8, 4.2)	3	12
[4.2, 4.6)	3	18
[4.6, 5.0)	3	17
[5.0, 5.4)	3	16
[5.4, 5.8)	3	14
[5.8, 6.2)	3	10
[6.2, 6.6)	4	2
[6.6, 7.0)	4	4

Status: OK Log x 0

Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

Open file...

Open URL...

Open DB...

Undo

Save...

Filter

Choose

Discretize -B 10 -R first-last

Apply

Current relation

Relation: iris

Instances: 150

Attributes: 5

Selected attribute

Name: petallength

Missing: 0 (0%)

Distinct: 43

Type: Numeric

Unique: 10 (7%)

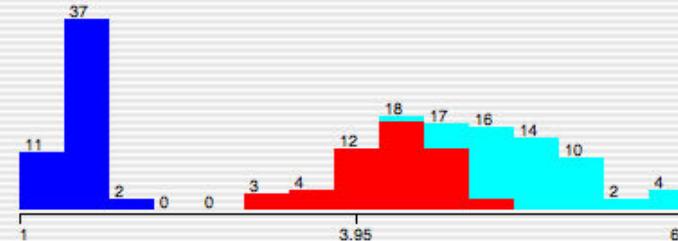
Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom)

Visualize All



Status

OK

Log



Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Undo Save...

Filter

Choose Discretize -B 10 -R first-last Apply

Current relation

Relation: iris Instances: 150 Attributes: 5

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute

Name: petallength Type: Numeric
Missing: 0 (0%) Distinct: 43 Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom) Visualize All

Frequency distribution of petallength:

Bin Range	Frequency
[1.0, 1.6)	37
[1.6, 2.2)	11
[2.2, 2.8)	2
[2.8, 3.4)	0
[3.4, 4.0)	0
[4.0, 4.6)	3
[4.6, 5.2)	4
[5.2, 5.8)	12
[5.8, 6.4)	18
[6.4, 6.9]	17
Total	150

Status

OK Log x 0

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Undo Save...

Filter

Choose Discretize -B 10 -R first-last Apply

Current relation

Relation: iris Instances: 150 Attributes: 5

Selected attribute

Name: petallength Type: Numeric
Missing: 0 (0%) Distinct: 43 Unique: 10 (7%)

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

weka.filters.unsupervised.attribute.Discretize

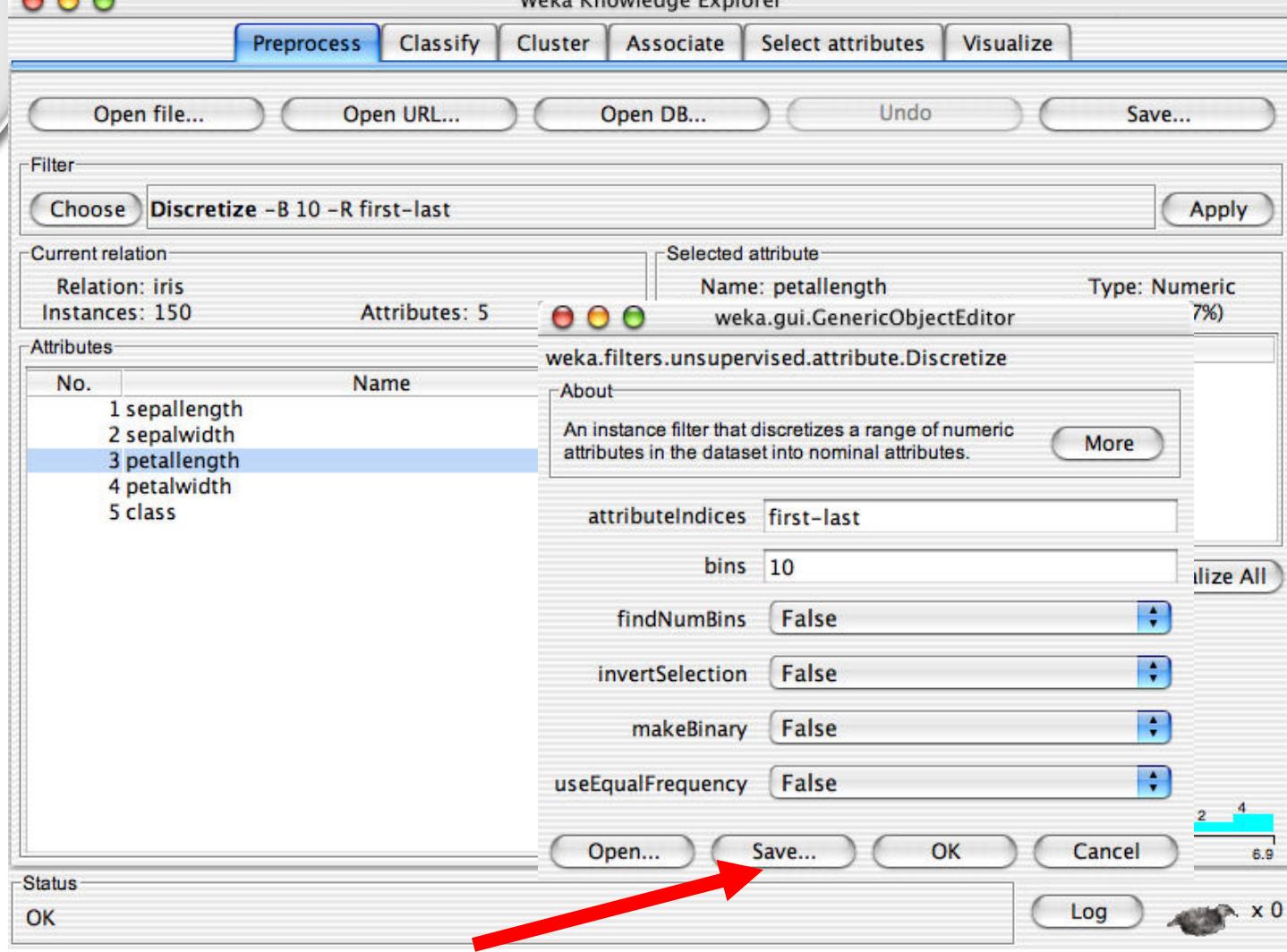
About

An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes. More

attributeIndices first-last
bins 10
findNumBins False
invertSelection False
makeBinary False
useEqualFrequency False

Status OK

Open... Save... OK Cancel



Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Undo Save...

Filter

Choose **Discretize -B 10 -R first-last**

weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.Discretize

About

An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes.

Numeric
10 (7%)

Attribute Indices: first-last

Bins: 10

Find Num Bins: False

Invert Selection: False

Make Binary: False

Use Equal Frequency: True

Open... Save... OK Cancel

Status: OK

Log x 0

1 3.95 6.9

1 3.95 6.9

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Undo Save...

Filter

Choose Discretize -B 10 -R first-last weka.gui.GenericObjectEditor Apply

weka.filters.unsupervised.attribute.Discretize

About An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes. More

Current relation Relation: iris Instances: 150 Attributes: !

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

attributeIndices first-last
bins 10
findNumBins False
invertSelection False
makeBinary False
useEqualFrequency True

Visualize All

Open... Save... OK Cancel

11 2 0 0 1 4 16 10 2 4

1 3.95 6.9

Status OK Log x 0



WEKA:: Explorer: building “classifiers”

- Classifiers in WEKA are models for predicting nominal or numeric quantities
- Implemented learning schemes include:
 - ▶ Decision trees and lists, instance-based classifiers, support vector machines, multi-layer perceptrons, logistic regression, Bayes' nets, ...
- “Meta”-classifiers include:
 - ▶ Bagging, boosting, stacking, error-correcting output codes, locally weighted learning, ...

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Undo Save...

Filter

Choose Discretize -F -B 10 -R first-last Apply

Current relation

Relation: iris Instances: 150 Attributes: 5

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute

Name: petallength Type: Numeric
Missing: 0 (0%) Distinct: 43 Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom) Visualize All

Frequency distribution of petallength:

Bin Range	Frequency
1.0 - 1.6	37
1.6 - 2.2	11
2.2 - 2.8	2
2.8 - 3.4	0
3.4 - 4.0	0
4.0 - 4.6	3
4.6 - 5.2	4
5.2 - 5.8	12
5.8 - 6.4	18
6.4 - 6.9	17
6.9 - 7.5	16
7.5 - 8.1	14
8.1 - 8.7	10
8.7 - 9.3	2
9.3 - 9.9	4

Status OK Log x 0

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Undo Save...

Filter

Choose **Discretize -F -B 10 -R first-last** Apply

Current relation

Relation: iris Instances: 150 Attributes: 5

Attributes

No.	Name
1	sepalwidth
2	petallength
3	petallength
4	petalwidth
5	class

Selected attribute

Name: petallength Type: Numeric
Missing: 0 (0%) Distinct: 43 Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom) Visualize All

Status OK Log x 0

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Undo Save...

Filter

Choose Discretize -F -B 10 -R first-last Apply

Current relation

Relation: iris-weka.filters.unsupervised.attribute.Disc...
Instances: 150 Attributes: 5

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute

Name: petallength Type: Nominal
Missing: 0 (0%) Distinct: 10 Unique: 0 (0%)

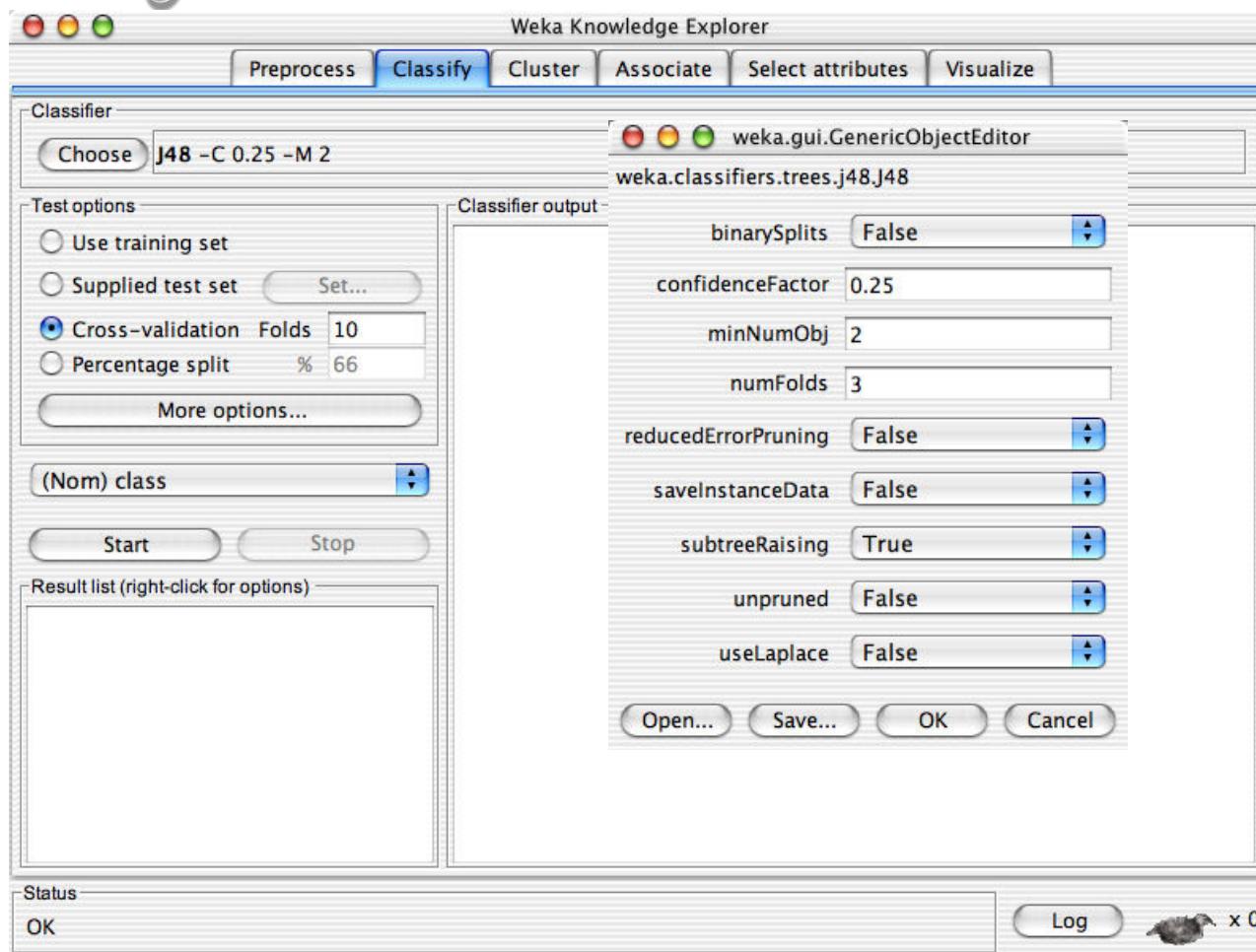
Label	Count
'(-inf-1.45]'	23
'(1.45-1.55]'	14
'(1.55-1.8]'	11
'(1.8-3.95]'	13
'(3.95-4.35]'	14
'(4.35-4.65]'	15
'(4.65-5.05]'	18

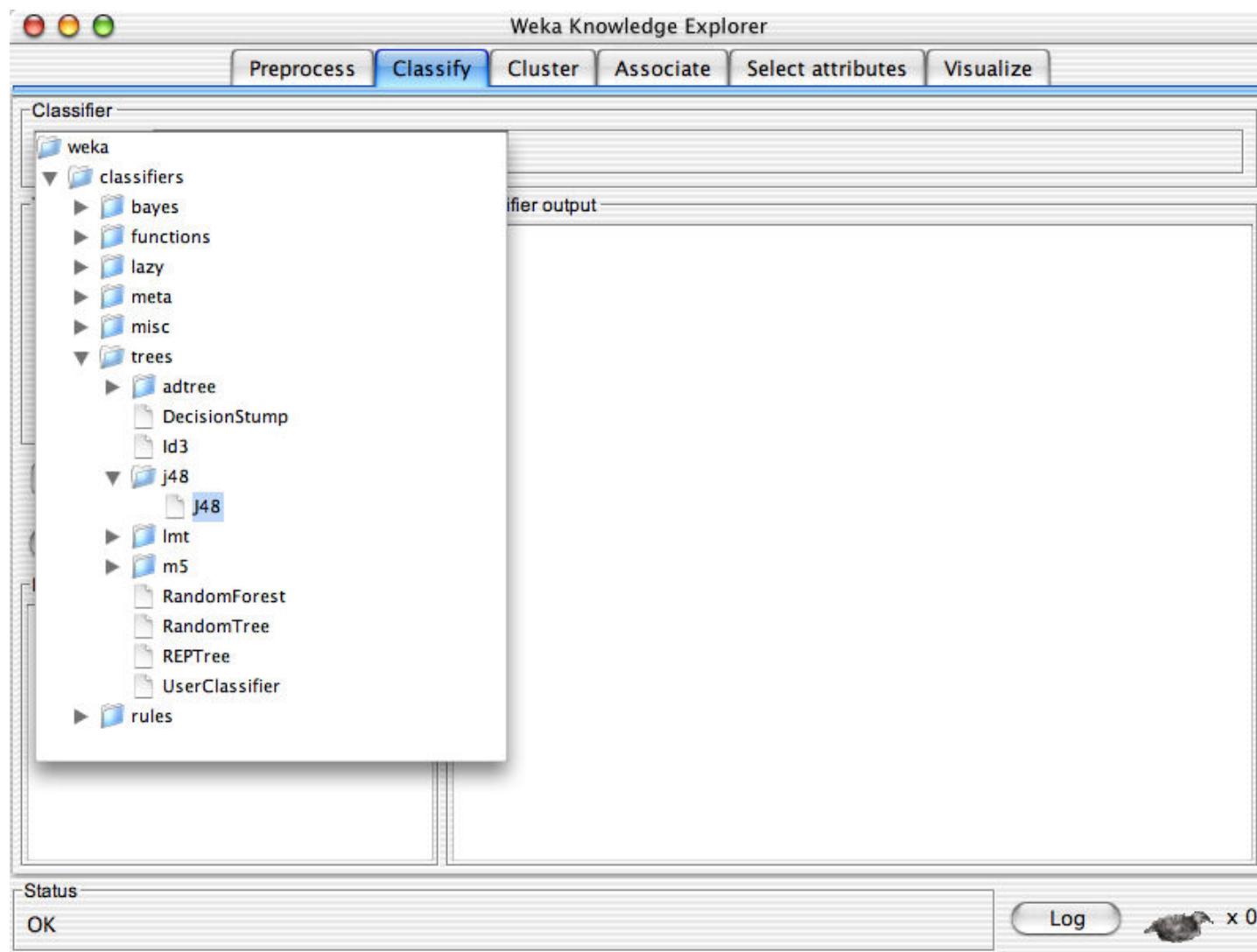
Colour: class (Nom) Visualize All

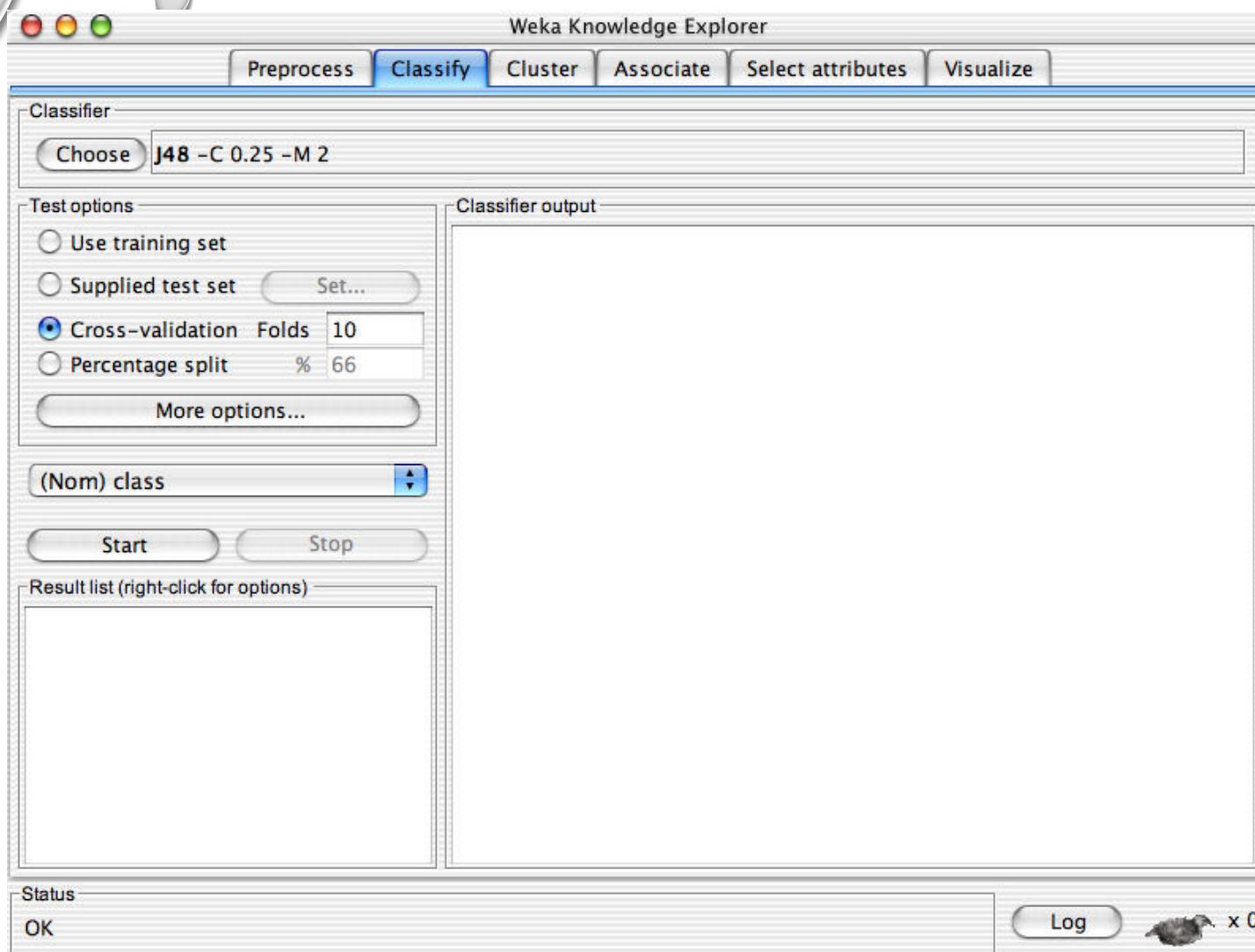
Status OK Log x 0

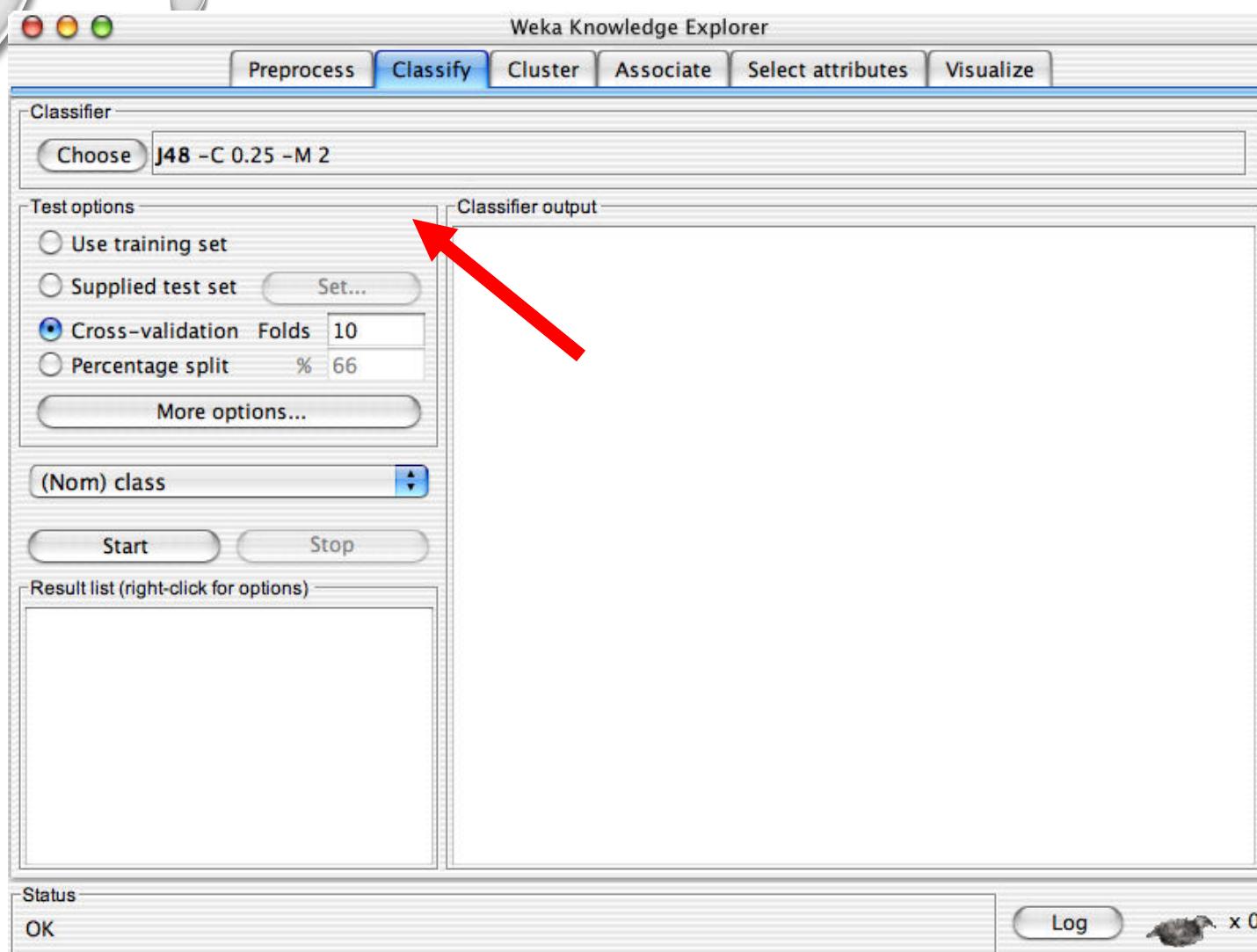
Explorer: building “classifiers”

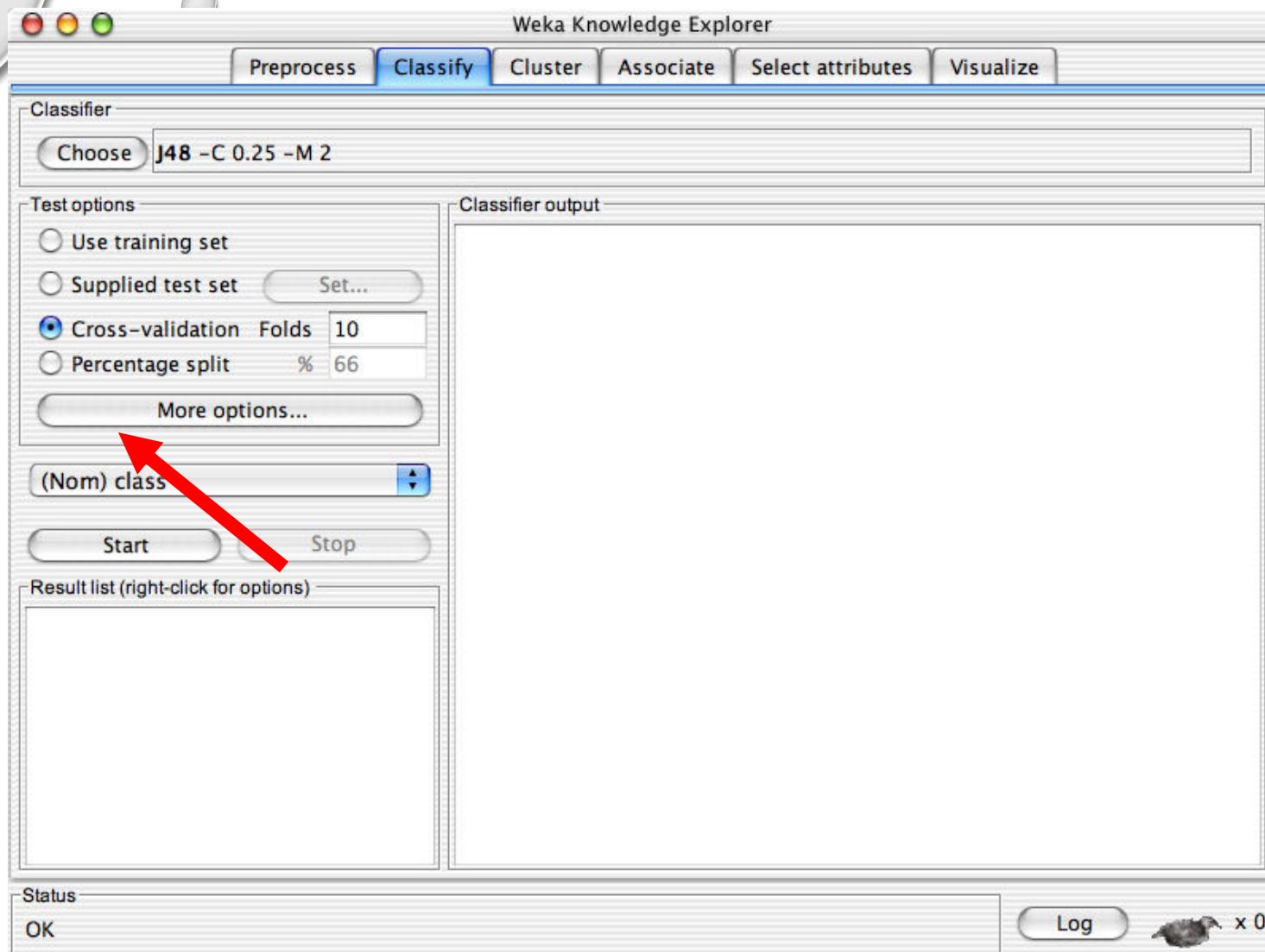
- Classifiers in WEKA are models for predicting nominal or numeric quantities
- Implemented learning schemes include:
 - ▶ **Decision trees** and lists, instance-based classifiers,
 - ▶ support vector machines,
 - ▶ multi-layer perceptrons,
 - ▶ logistic regression, Bayes' nets, ...











Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 -C 0.25 -M 2

Test options

Use training set

Supplied test set Set...

Cross-validation Folds 10

Percentage split % 66

More options...

(Nom) class

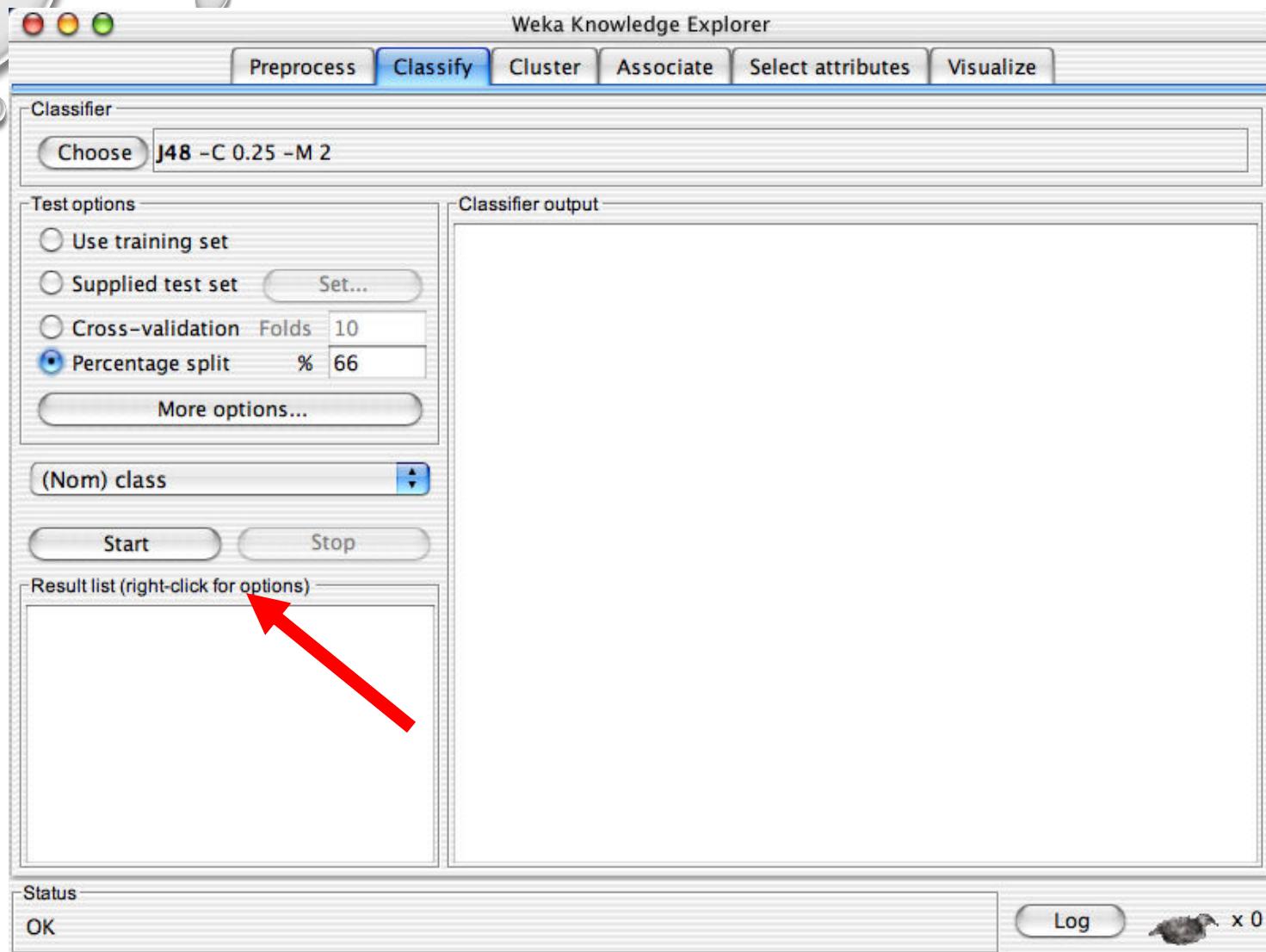
Start Stop

Result list (right-click for options)

Status

OK Log x 0

This screenshot shows the Weka Knowledge Explorer application window. The title bar reads "Weka Knowledge Explorer". Below the title bar is a menu bar with tabs: Preprocess, Classify (which is highlighted in blue), Cluster, Associate, Select attributes, and Visualize. The main area is titled "Classifier". A "Choose" button is followed by the text "J48 -C 0.25 -M 2". Below this is a section titled "Test options" containing four radio buttons: "Use training set", "Supplied test set" (with a "Set..." button), "Cross-validation" (with a "Folds" input field set to 10), and "Percentage split" (which is selected, with a "% 66" input field). There is also a "More options..." button. To the right is a large "Classifier output" pane which is currently empty. Below the "Test options" section is a dropdown menu set to "(Nom) class" with a dropdown arrow. At the bottom left is a "Status" section with the text "OK". On the far right of the status bar is a "Log" button and a small icon with the text "x 0".



Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 - C 0.25 - M 2

Test options

Use training set

Supplied test set Set...

Cross-validation Folds 10

Percentage split % 66

[More options...](#)

(Nom) class

Start Stop

Result list (right-click for options)

11:49:05 - trees.J48.J48

Classifier output

```
==== Run information ====
Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Test mode: split 66% train, remainder test

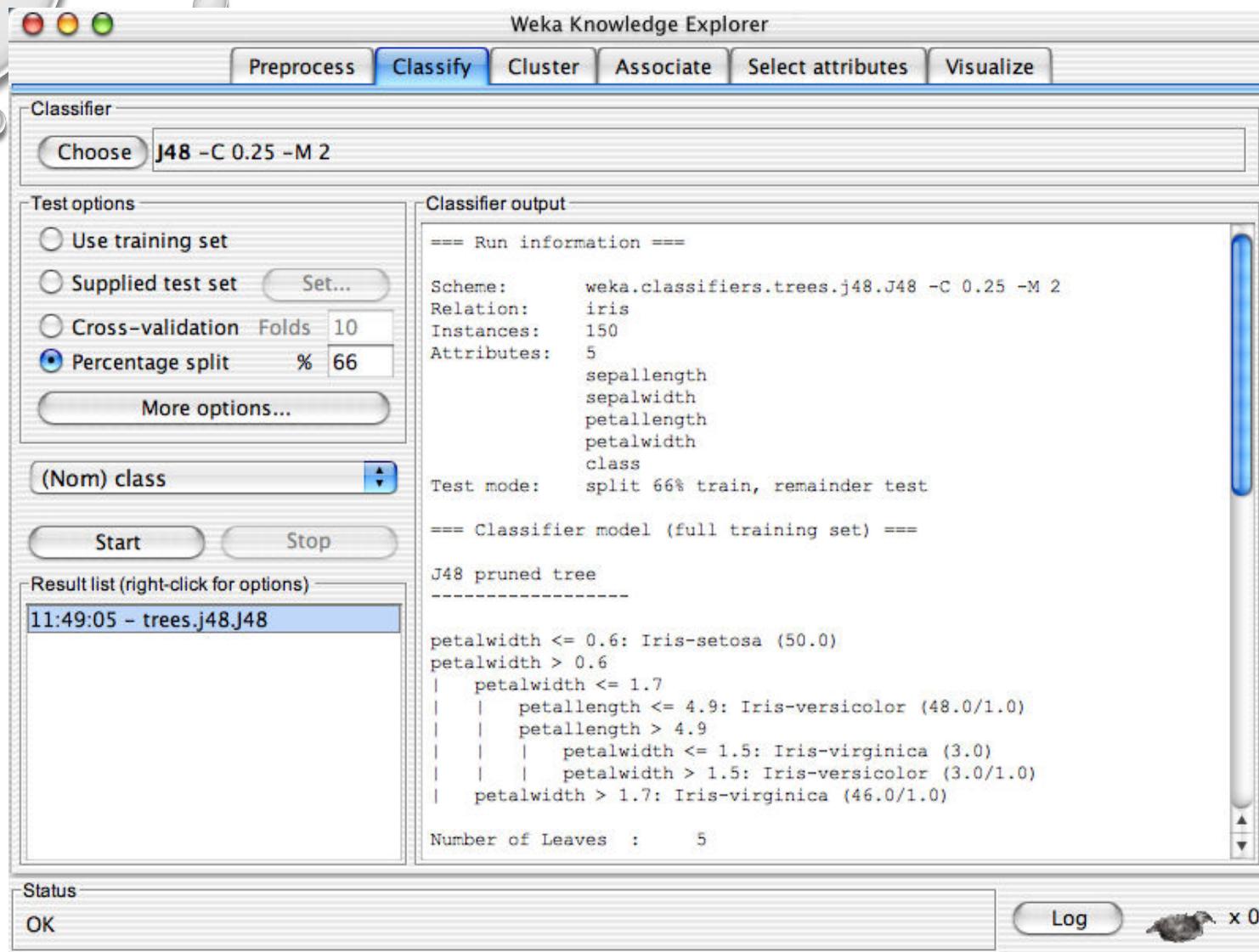
==== Classifier model (full training set) ====
J48 pruned tree
-----
petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves : 5
```

Status

OK

Log x 0



Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 - C 0.25 - M 2

Test options

Use training set
 Supplied test set Set...
 Cross-validation Folds 10
 Percentage split % 66
More options...

(Nom) class

Start Stop

Result list (right-click for options)

11:49:05 - trees.j48.J48

Classifier output

Time taken to build model: 0.24 seconds

== Evaluation on test split ==

== Summary ==

Correctly Classified Instances	49	96.0784 %
Incorrectly Classified Instances	2	3.9216 %
Kappa statistic	0.9408	
Mean absolute error	0.0396	
Root mean squared error	0.1579	
Relative absolute error	8.8979 %	
Root relative squared error	33.4091 %	
Total Number of Instances	51	

== Detailed Accuracy By Class ==

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
1	0	1	1	1	Iris-setosa
1	0.063	0.905	1	0.95	Iris-versicolor
0.882	0	1	0.882	0.938	Iris-virginica

== Confusion Matrix ==

a	b	c	<-- classified as
15	0	0	a = Iris-setosa
0	19	0	b = Iris-versicolor
0	2	15	c = Iris-virginica

Status

OK Log x 0

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 - C 0.25 - M 2

Test options

Use training set

Supplied test set Set...

Cross-validation Folds 10

Percentage split % 66

More options...

(Nom) class

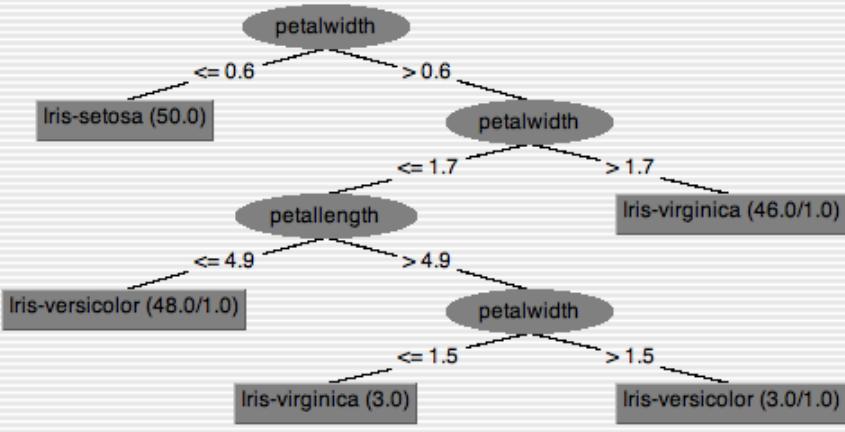
Start Stop

Result list (right-click for options)

11:49:05 - trees.j48.J48

Weka Classifier Tree Visualizer: 11:49:05 - trees.j48.J48 (iris)

Tree View



```
graph TD; Root((petalwidth)) --<= 0.6 --> Setosa[Iris-setosa (50.0)]; Root --> Virginica((Iris-virginica (46.0/1.0))); Virginica --> Versicolor1((Iris-versicolor (48.0/1.0))); Virginica --> Versicolor2((Iris-versicolor (3.0/1.0))); Versicolor1 --> Versicolor1a((Iris-versicolor (48.0/1.0))); Versicolor1 --> Virginica1((Iris-virginica (3.0))); Versicolor1a --> Versicolor1b((Iris-versicolor (3.0/1.0))); Versicolor1a --> Virginica1a((Iris-virginica (3.0)))
```

0 19 0 | b = Iris-versicolor
0 2 15 | c = Iris-virginica

Status OK Log x 0

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

weka
 └ classifiers
 └ bayes
 AODE
 BayesNetK2
 BayesNetB
 NaiveBayes
 NaiveBayesMultinomial
 NaiveBayesSimple
 NaiveBayesUpdateable
 functions
 lazy
 meta
 misc
 trees
 rules

Classifier output

```
== Evaluation on test split ===
== Summary ===

  correctly Classified Instances      50          98.0392 %
  incorrectly Classified Instances    1           1.9608 %
  kappa statistic                   0.9704
  mean absolute error              0.0239
  root mean squared error          0.1101
  relative absolute error          5.3594 %
  root relative squared error     23.2952 %
  total Number of Instances        51

== Detailed Accuracy By Class ===

      P  Rate   FP Rate   Precision   Recall   F-Measure   Class
      1       0         1         1         1
      1       0.031      0.95      1         0.974      Iris-setosa
      0.941   0         1         0.941      0.97      Iris-versicolor
                                Iris-virginica

== Confusion Matrix ===

      a   b   c   <-- classified as
  15  0   0 |  a = Iris-setosa
  0  19  0 |  b = Iris-versicolor
  0  1  16 |  c = Iris-virginica
```

Status

Problem evaluating classifier

Log x 0

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **NaiveBayes**

Test options

Use training set
 Supplied test set Set...
 Cross-validation Folds 10
 Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

11:49:05 - trees.j48.J48
14:34:28 - functions.neural.NeuralNetwork

Classifier output

```
==== Evaluation on test split ====
==== Summary ====
Correctly Classified Instances      50          98.0392 %
Incorrectly Classified Instances    1           1.9608 %
Kappa statistic                      0.9704
Mean absolute error                  0.0239
Root mean squared error              0.1101
Relative absolute error              5.3594 %
Root relative squared error        23.2952 %
Total Number of Instances           51
```

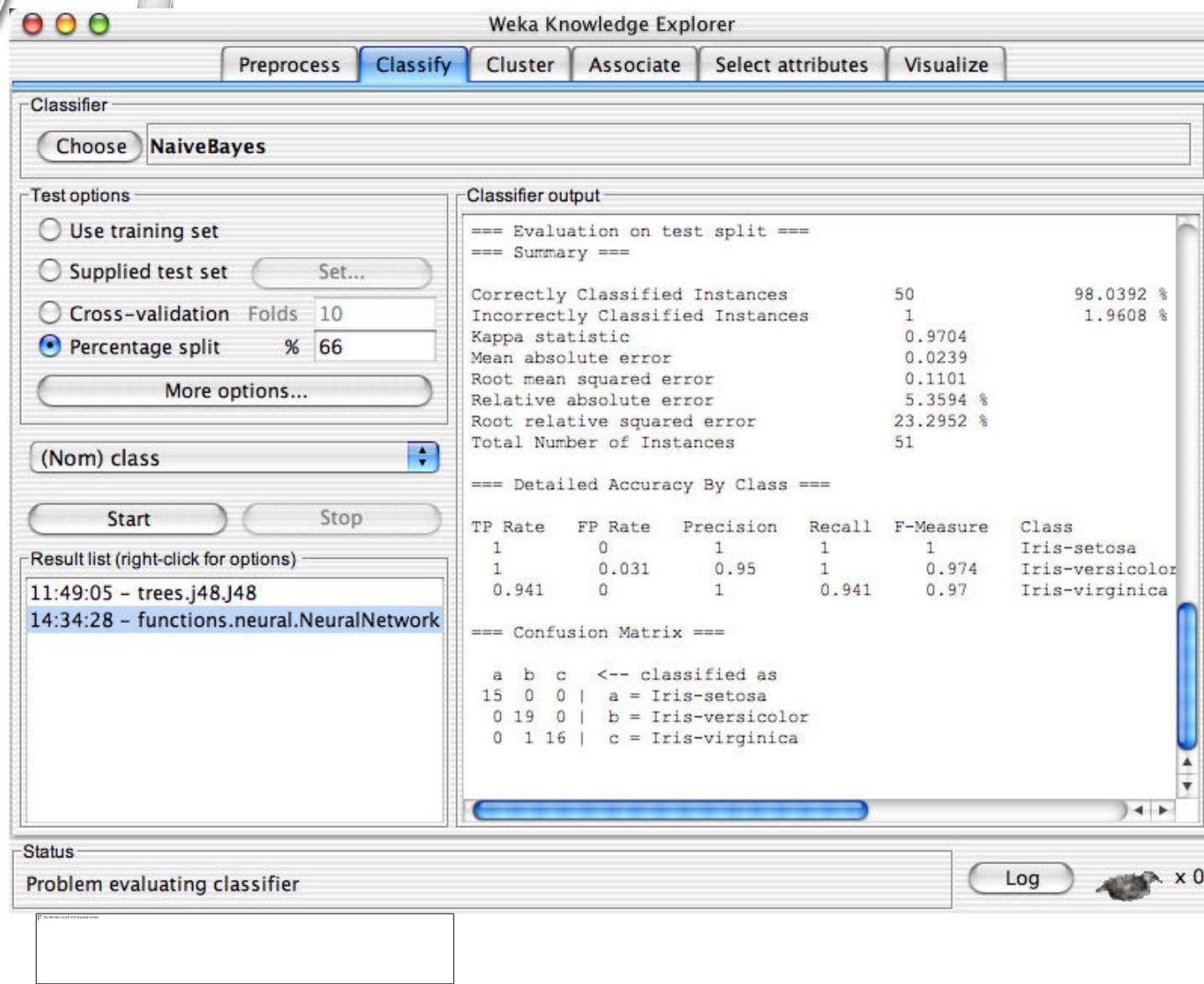
```
==== Detailed Accuracy By Class ====
TP Rate   FP Rate   Precision   Recall   F-Measure   Class
1         0         1           1         1           Iris-setosa
1         0.031     0.95        1         0.974       Iris-versicolor
0.941    0         1           0.941    0.97       Iris-virginica
```

```
==== Confusion Matrix ====
a b c  <-- classified as
15 0 0 | a = Iris-setosa
 0 19 0 | b = Iris-versicolor
 0 1 16 | c = Iris-virginica
```

Status

Problem evaluating classifier

Log x 0



Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose NaiveBayes

Test options

Use training set
 Supplied test set Set...
 Cross-validation Folds 10
 Percentage split % 66
More options...

(Nom) class

Start Stop

Result list (right-click for options)

11:49:05 - trees.j48 148
14:34:28 - functions.Neural.NeuralNetwork

Classifier output

```
==== Evaluation on test split ====
==== Summary ====
Correctly Classified Instances      50          98.0392 %
Incorrectly Classified Instances   1           1.9608 %
Kappa statistic                   0.9704
Mean absolute error               0.0239
Root mean squared error          0.1101
Relative absolute error           5.3594 %
Root relative squared error     23.2952 %
Total Number of Instances        51
```

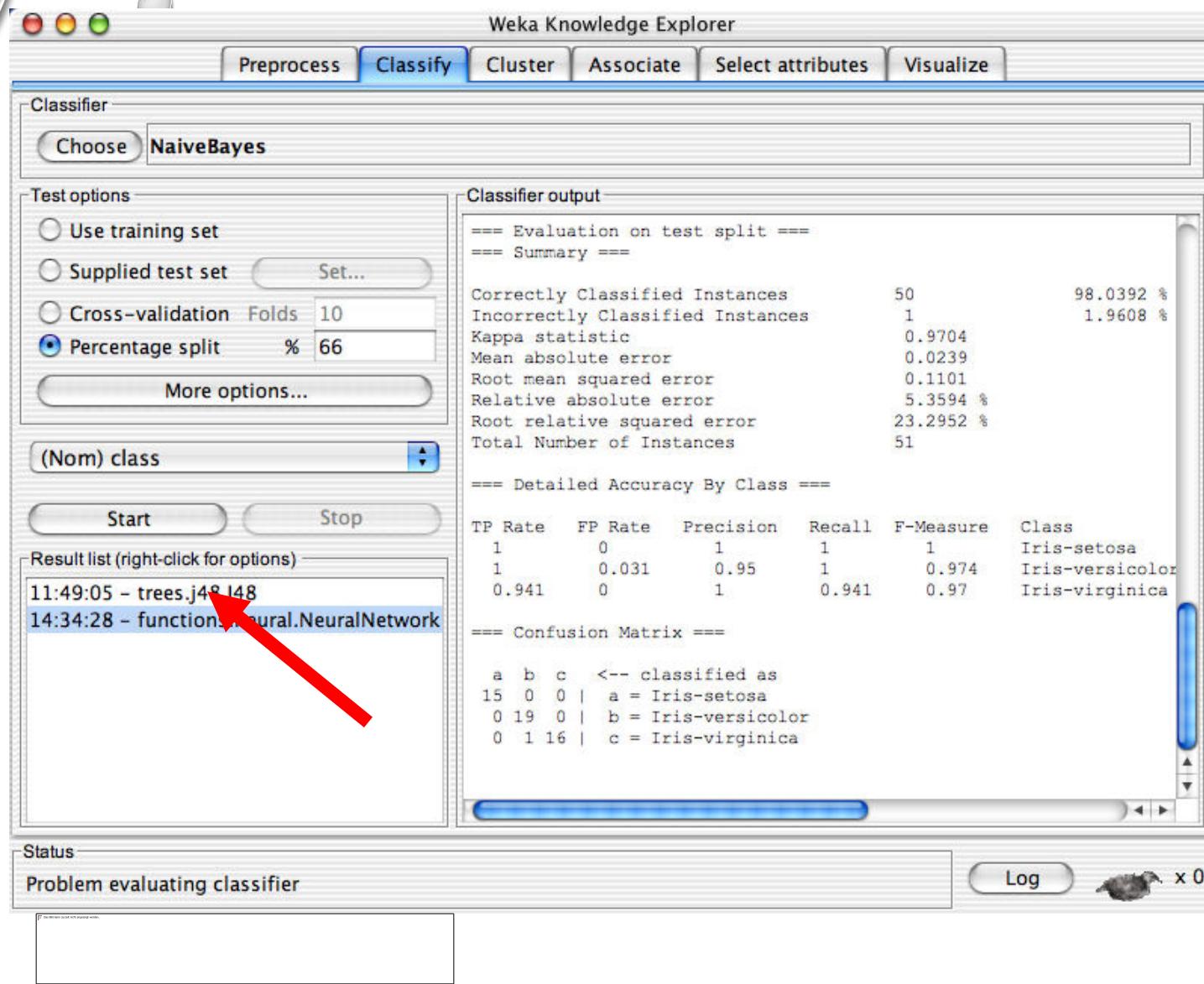
```
==== Detailed Accuracy By Class ====
TP Rate   FP Rate   Precision   Recall   F-Measure   Class
1         0         1           1         1           Iris-setosa
1         0.031     0.95       1         0.974      Iris-versicolor
0.941    0         1           0.941    0.97       Iris-virginica
```

```
==== Confusion Matrix ====
a b c  <-- classified as
15 0 0 | a = Iris-setosa
0 19 0 | b = Iris-versicolor
0 1 16 | c = Iris-virginica
```

Status

Problem evaluating classifier

Log x 0



Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose NaiveBayes

Test options

Use training set
 Supplied test set Set...
 Cross-validation Folds 10
 Percentage split % 66
More options...

(Nom) class

Start Stop

Result list (right-click for options)

11:49:05 - trees.j48.J48
14:34:28 - functions.neural.NeuralNetwork
14:48:05 - bayes.NaiveBayes

Classifier output

```
==== Evaluation on test split ====
==== Summary ====
Correctly Classified Instances           48          94.1176 %
Incorrectly Classified Instances        3           5.8824 %
Kappa statistic                         0.9113
Mean absolute error                     0.0447
Root mean squared error                 0.1722
Relative absolute error                  10.0365 %
Root relative squared error            36.4196 %
Total Number of Instances               51
```

```
==== Detailed Accuracy By Class ====
TP Rate    FP Rate    Precision    Recall    F-Measure    Class
 1         0          1           1          1          Iris-setosa
 0.947     0.063      0.9          0.947      0.923      Iris-versicolor
 0.882     0.029      0.938        0.882      0.909      Iris-virginica
```

```
==== Confusion Matrix ====
 a   b   c   <-- classified as
15   0   0 |   a = Iris-setosa
 0  18   1 |   b = Iris-versicolor
 0   2  15 |   c = Iris-virginica
```

Status OK Log x 0

Weka Knowledge Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose NaiveBayes

Test options

Use training set
 Supplied test set Set...
 Cross-validation Folds 10
 Percentage split % 66
More options...

(Nom) class

Start Stop

Result list (right-click for options)

11:49:05 - trees.j48.J48
14:34:28 - functions.neural.NeuralNetwork
14:48:05 - bayes.NaiveBayes

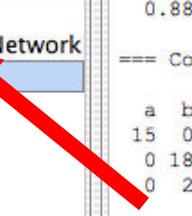
Classifier output

```
==== Evaluation on test split ====
==== Summary ====
Correctly Classified Instances          48          94.1176 %
Incorrectly Classified Instances        3           5.8824 %
Kappa statistic                         0.9113
Mean absolute error                     0.0447
Root mean squared error                 0.1722
Relative absolute error                  10.0365 %
Root relative squared error            36.4196 %
Total Number of Instances               51
```

```
==== Detailed Accuracy By Class ====
TP Rate    FP Rate    Precision    Recall    F-Measure    Class
1          0          1           1           1           Iris-setosa
0.947     0.063      0.9          0.947      0.923      Iris-versicolor
0.882     0.029      0.938       0.882      0.909      Iris-virginica
```

```
==== Confusion Matrix ====
a   b   c   <-- classified as
15  0   0   |   a = Iris-setosa
0  18  1   |   b = Iris-versicolor
0  2   15  |   c = Iris-virginica
```

Status OK Log x 0



Chapter 7

K- Nearest Neighbors

Dr. Feras Al-Obeidat,

Outlines

▶ Overview of Knowledge Discovery/Data mining/AI

- Unsupervised and supervised learning paradigm
- K-Nearest Neighborhood (K-NN)

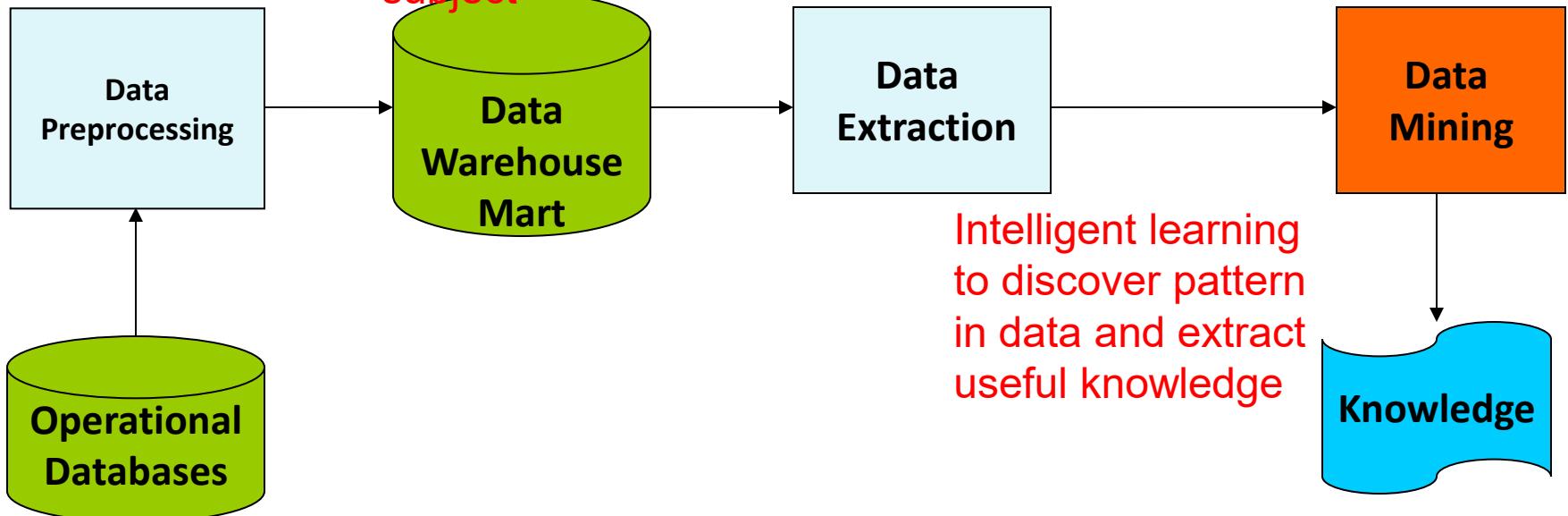
Knowledge Discovery Process

Data Mining/Machine learning algorithms are part of knowledge discovery

Date cleaning,
normalization,

Organize data
according to their
subject

Obtain necessary
and related data



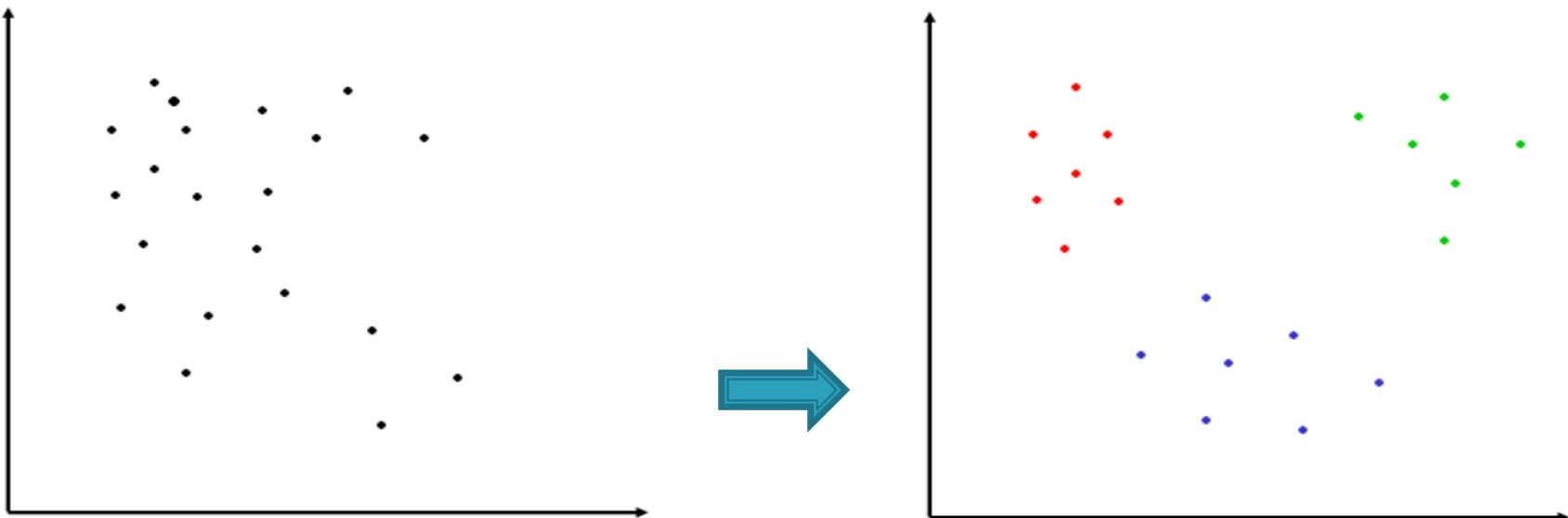
Intelligent learning
to discover pattern
in data and extract
useful knowledge

Un-Supervised Learning

Organize a collection of unlabeled data items into categories

The instances are unlabelled and the goal is to organize a collection of data items into categories,

- The items within a category are more similar to each other than they are to items in the other categories.



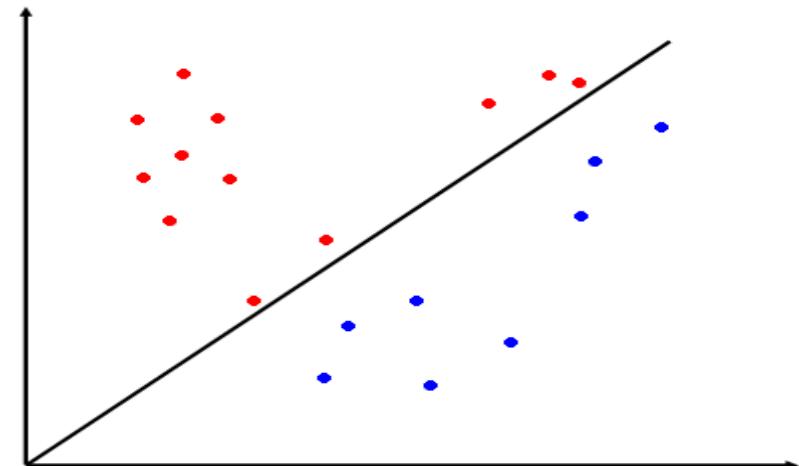
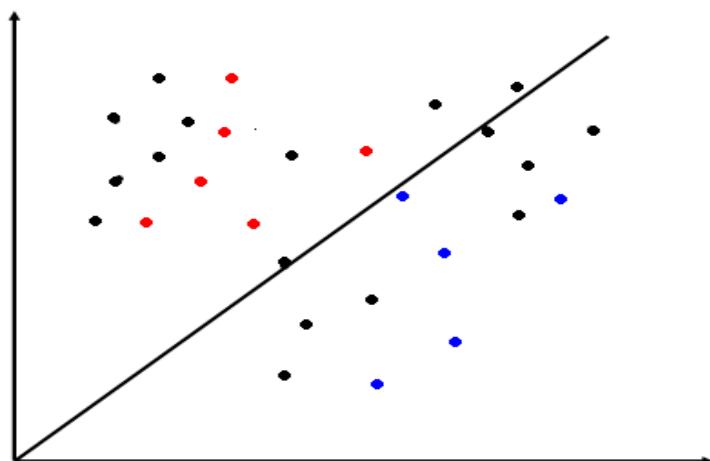
Clustering is also good approach for anomaly detection

K-Means

Supervised Learning

Predict the relationship between objects and class-labels
(Hypothesis)

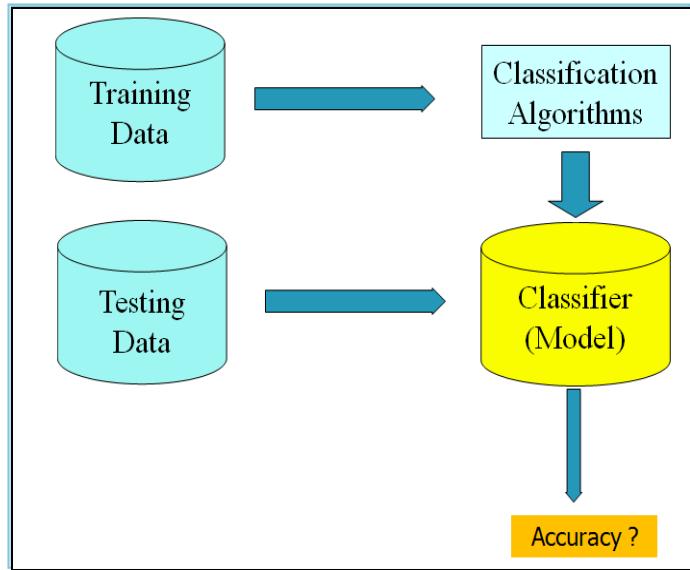
- Each object is labeled with a class.
- The target is to find the predictive relationship between objects and class-labels. (Hypothesis)



- K-NN (K- Nearest Neighbor)
- Decision Trees (Id3, C4.5)

- SVM (Support Vector Machines)
- ANN (Artificial Neural Network)
- NB (Naive Bayes)

Classification Process-Supervised



Typical applications:

- *Credit/Loan Approval.*

Use credit card information to predict customer behavior

Use credit card transactions and information on its account-holder as input variables label past transactions as : Fair or Fraud :

- *Medical Diagnosis:*

e.g., if a tumor is *cancerous* or not

inputs: age, tumor-size, breast: (left or right.), etc, ...

output: Benign or Malignant



Instance-based Learning

- ▶ K-Nearest Neighbor Algorithm
- ▶ Weighted Regression
- ▶ Case-based reasoning

K–Nearest–Neighbors Algorithm

- ▶ K nearest neighbors (KNN) is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (distance function)
- ▶ KNN has been used in statistical estimation and pattern recognition since 1970's.

K–Nearest–Neighbors Algorithm

- ▶ A case is classified by a majority voting of its neighbors, with the case being assigned to the class most common among its K nearest neighbors measured by a distance function.
- ▶ If $K=1$, then the case is simply assigned to the class of its nearest neighbor

Distance Function Measurements

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^k |x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$$

Hamming Distance

- For category variables, Hamming distance can be used.

Hamming Distance

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

K-Nearest Neighbor

- ▶ Features
 - All instances correspond to points in an n-dimensional Euclidean space
 - Classification is delayed till a new instance arrives
 - Classification done by comparing feature vectors of the different points
 - Target function may be discrete or real-valued

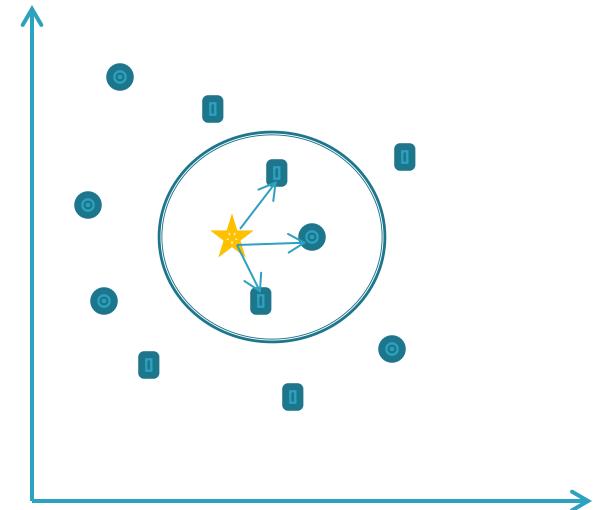
K -Nearest Neighbor (k -NN)

- Instance based learning algorithm
- Lazy learner: needs more computation time during classification
- Conceptually close to human intuition: e.g., people with similar income would live in the same neighborhood

Classification strategy:

K-NN assigns the instance to relative class group by identifying the most frequent class label.

In some case when numeric instances are involved proximity distance measures is required. E.g., Euclidean Distance



K = number of nearest neighbors for each test example

Instance-Based Learning

► Idea:

- Similar examples have similar label.
- Classify new examples like similar training examples.

Instance-Based Learning

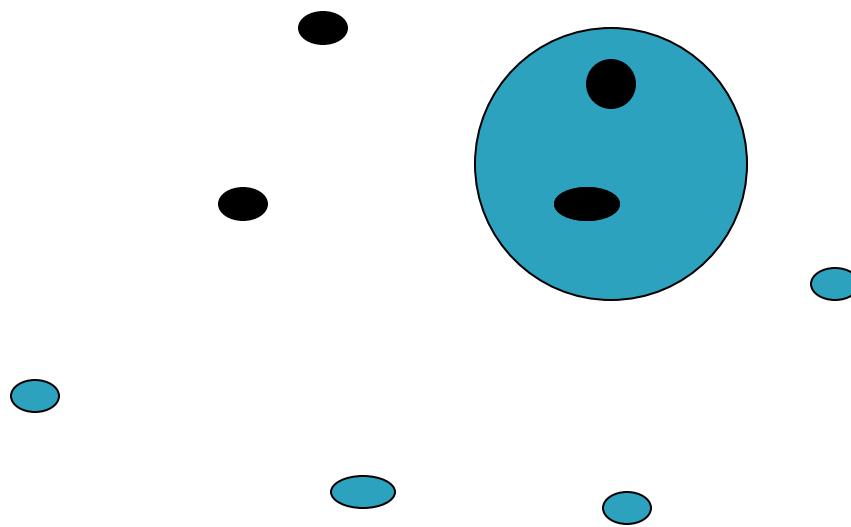
► Algorithm:

- Given some new example x for which we need to predict its class y
- Find most similar training examples
- Classify x “like” these most similar examples

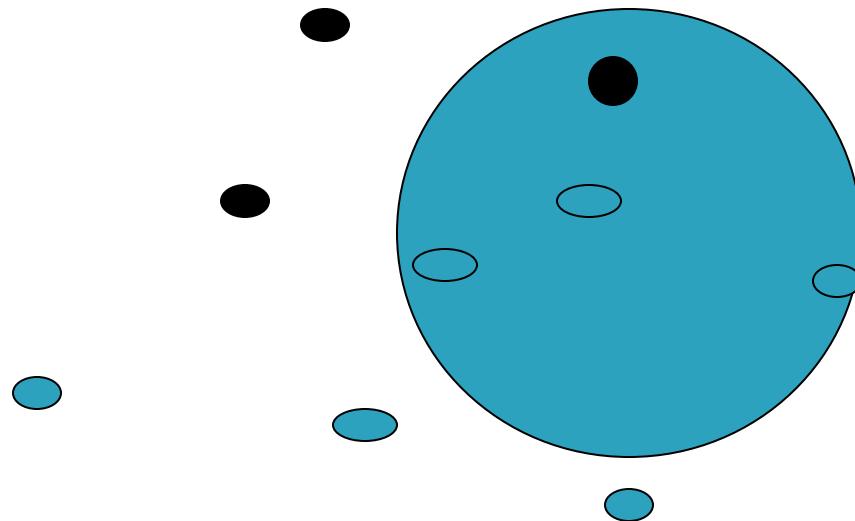
Instance-Based Learning

- ▶ Questions:
 - How to determine similarity?
 - How many similar training examples to consider?
 - How to resolve inconsistencies among the training examples?

1-Nearest Neighbor

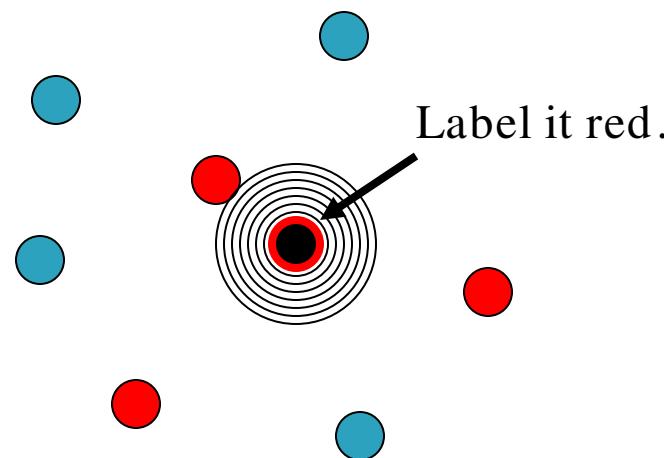


3-Nearest Neighbor



1-Nearest Neighbor

- ▶ One of the simplest of all machine learning classifiers
- ▶ Simple idea: label a new point the same as the closest known point



1-NN's Aspects as an Instance-Based Learner:

A distance metric

- Euclidean
- When different units are used for each dimension
 - normalize each dimension by standard deviation
- For discrete data, can use hamming distance
 - $D(x_1, x_2) = \text{number of features on which } x_1 \text{ and } x_2 \text{ differ}$
- Others (e.g., normal, cosine)

1-NN's Aspects as an Instance-Based Learner:

How many nearby neighbors to look at?

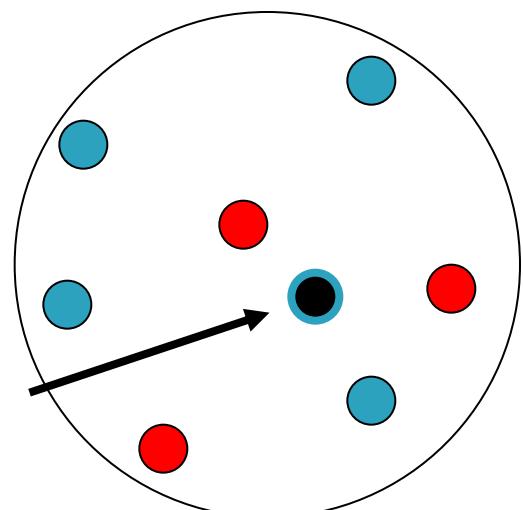
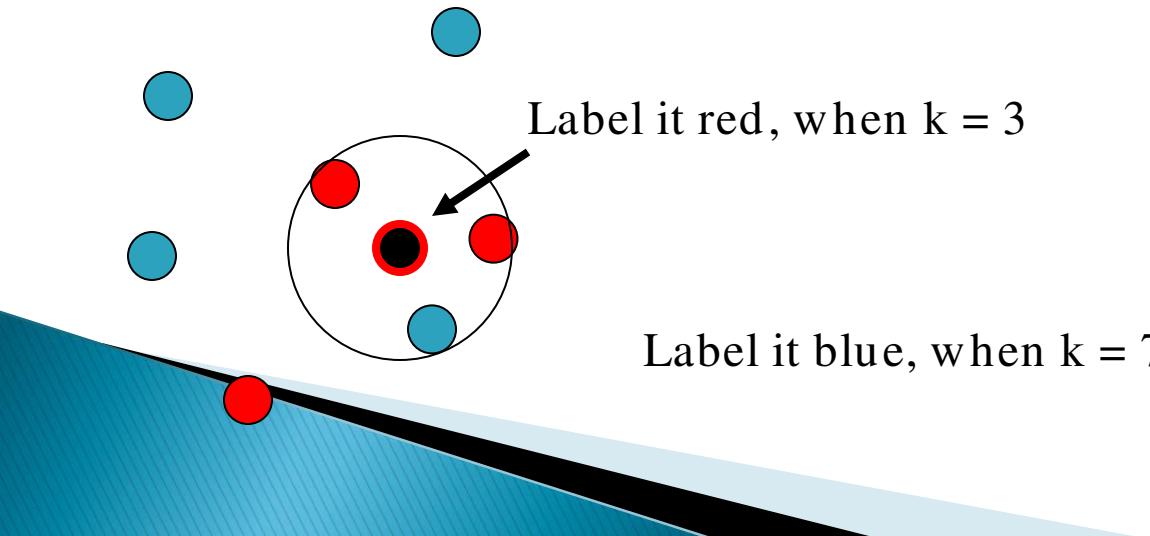
- One

How to fit with the local points?

- Just predict the same output as the nearest neighbor.

k - Nearest Neighbor

- ▶ Generalizes 1-NN to smooth away noise in the labels
- ▶ A new point is now assigned **the most frequent label of its k nearest neighbors**



KNN Example

	Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
1	great	yes	yes	normal	no	yes
2	great	no	yes	normal	no	yes
3	mediocre	yes	no	high	no	no
4	great	yes	yes	normal	yes	yes

Similarity metric: Number of matching attributes (k=2)

► New examples:

- Example 1 (great, no, **no**, normal, no) **Yes**

→ most similar: number 2 (1 mismatch, 4 match) → **yes**

→ Second most similar example: number 1 (2 mismatch, 3 match) → **yes**

KNN Example

	Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
1	great	yes	yes	normal	no	yes
2	great	no	yes	normal	no	yes
3	mediocre	yes	no	high	no	no
4	great	yes	yes	normal	yes	yes

Similarity metric: Number of matching attributes (k=2)

► New examples:

Example 2 (mediocre, yes, no, **normal**, no) Yes/No

→ Most similar: number 3 (1 mismatch, 4 match) → **no**

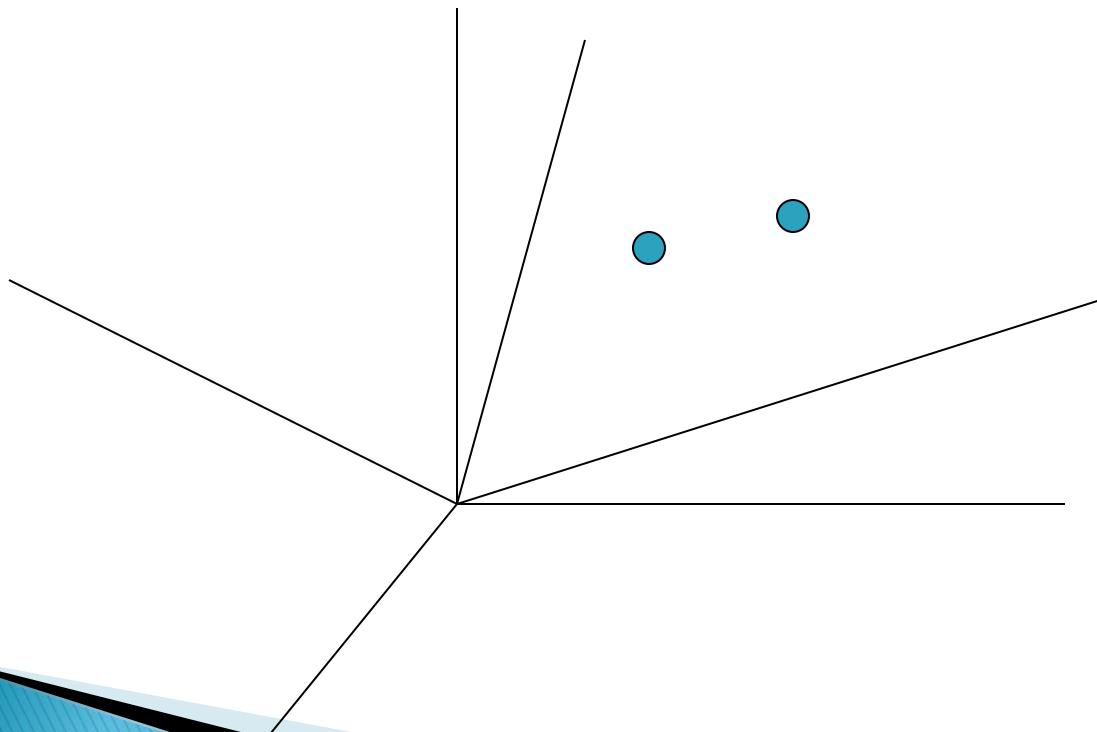
→ Second most similar example: number 1 (2 mismatch, 3 match) → **yes**

Selecting the Number of Neighbors

- ▶ Increase k:
 - Makes KNN less sensitive to noise
- ▶ Decrease k:
 - Allows capturing finer structure of space
- ▶ → Pick k not too large, but not too small
(depends on data)

Remarks

- Curse of Dimensionality



Curse-of-Dimensionality

- ▶ Prediction accuracy can quickly degrade when number of attributes grows.
 - Irrelevant attributes easily “swamp” information from relevant attributes
 - When many irrelevant attributes, similarity/distance measure becomes less reliable

Curse-of-Dimensionality

► Remedy

- Try to remove irrelevant attributes in pre-processing step
- Weight attributes differently
- Increase k (but not too much)

Advantages and Disadvantages of KNN

- ▶ Need distance/similarity measure and attributes that “match” target function.
- ▶ For large training sets,
 - Must make a pass through the entire dataset for each classification. This can be prohibitive for large data sets.
- ▶ Prediction accuracy can quickly degrade when number of attributes grows.

Advantages and Disadvantages of KNN

Simple to implement algorithm;
Requires little tuning;
Often performs quite well!
(Try it first on a new learning problem).

Using K-NN in R

Case study: Iris data set

```
#load your data  
df <- data(iris)
```

```
# look into data structure  
head(iris)  
str(iris)  
dim(iris)
```

Generate a random sample of all data

```
# Generate a random sample of all data  
# in this case 82% of the dataset.
```

```
randSelection <- sample(1:nrow(iris), 0.82 * nrow(iris))  
randSelection
```

Normalization

```
# data normalization f  
normalization <-function(x) { (x -min(x))/(max(x)-min(x)) }  
  
# Run nomalization on on coulumns which are the predictors  
  
irisNormalized <- as.data.frame(lapply(iris[,c(1:4)], normalization))  
  
summary(irisNormalized)
```

Training & Testing

```
## separate data into training and testing to  
#check model accuracy  
# get training data  
training <- irisNormalized[randSelection,]  
nrow(training)  
  
# get testing data  
testing <- irisNormalized[-randSelection,]  
nrow(testing)
```

Obtain the class label

```
# obtain the class label of train dataset because as it will  
#be used as argument in knn classifier  
targetClass <- iris[randSelection,5]  
targetClass  
summary(targetClass)  
  
# extract 5th column if test dataset to measure the  
#accuracy  
testClass <- iris[-randSelection,5]  
summary(testClass)
```

Install package class for k-nn & Build the model

```
library(class)
# building the model for classification
# run knn classifier
# here we use k = 10

classificationModel <- knn(training,testing,cl=targertClass,k=10)
classificationModel
```

confusion matrix

```
#create confusion matrix to check model #  
performance
```

```
ConfMatrix <- table(classificationModel,testClass)  
ConfMatrix
```

		testClass		
		setosa	versicolor	virginica
classificationModel	setosa	14	0	0
	versicolor	0	7	0
	virginica	0	1	6

Model Accuracy

#Calculate model accuracy

```
modelAccuracy <- function(x){sum(diag(x))/(sum(rowSums(x)))) * 100}
```

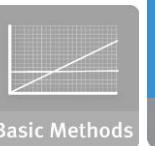
```
modelAccuracy(ConfMatrix)
```



Introduction



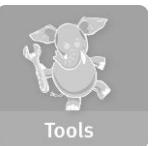
Analytics Lifecycle



Basic Methods



Adv. Methods



Tools



Lab

Module 4: Advanced Analytics – Theory and Methods

Lesson 7: Time Series Analysis

During this lesson the following topics are covered:

- Time Series Analysis and its applications in forecasting
- ARIMA Models
- Implementing the Box-Jenkins Methodology using R
- Reasons to Choose (+) and Cautions (-) with Time Series Analysis

Time Series

What is a time series?

- ▶ a collection of data recorded over a period of time (**weekly, monthly, quarterly, etc.**)
- ▶ an analysis of **history**, it can be used by management to make current decisions and plans based on long-term forecasting
- ▶ Usually assumes **past pattern** to continue into the **future**

Time Series Analysis

- Time Series: Ordered sequence of equally spaced values over time
- Time Series Analysis: Accounts for the **internal structure** of observations taken over time
 - ▶ Trend
 - ▶ Seasonality
 - ▶ Cycles
 - ▶ Random
- Goals
 - ▶ To **identify the internal structure** of the time series
 - ▶ To **forecast** future events
 - ▶ Example: Based on sales history, what will next December sales be?

Components of a Time Series

- **Secular Trend** – the smooth long term direction of a time series
- **Cyclical Variation** – the rise and fall of a time series over periods longer than one year
- **Seasonal Variation** – Patterns of change in a time series within a year which tends to repeat each year
- **Irregular Variation** – unpredictable but identifiable

Cyclical Variation – Sample Chart

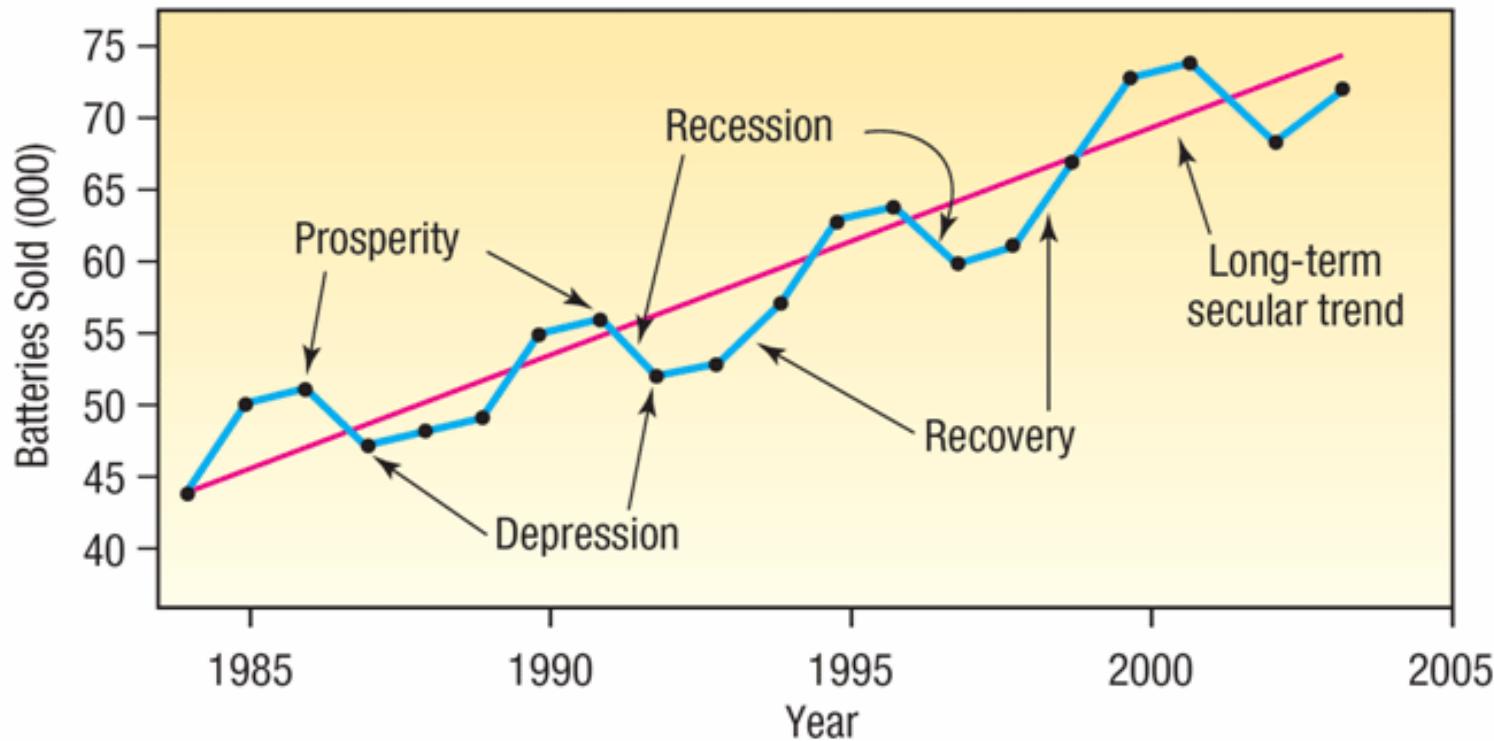


CHART 16-1 Batteries Sold by National Battery Retailers, Inc., from 1984 to 2004

Seasonal Variation – Sample Chart

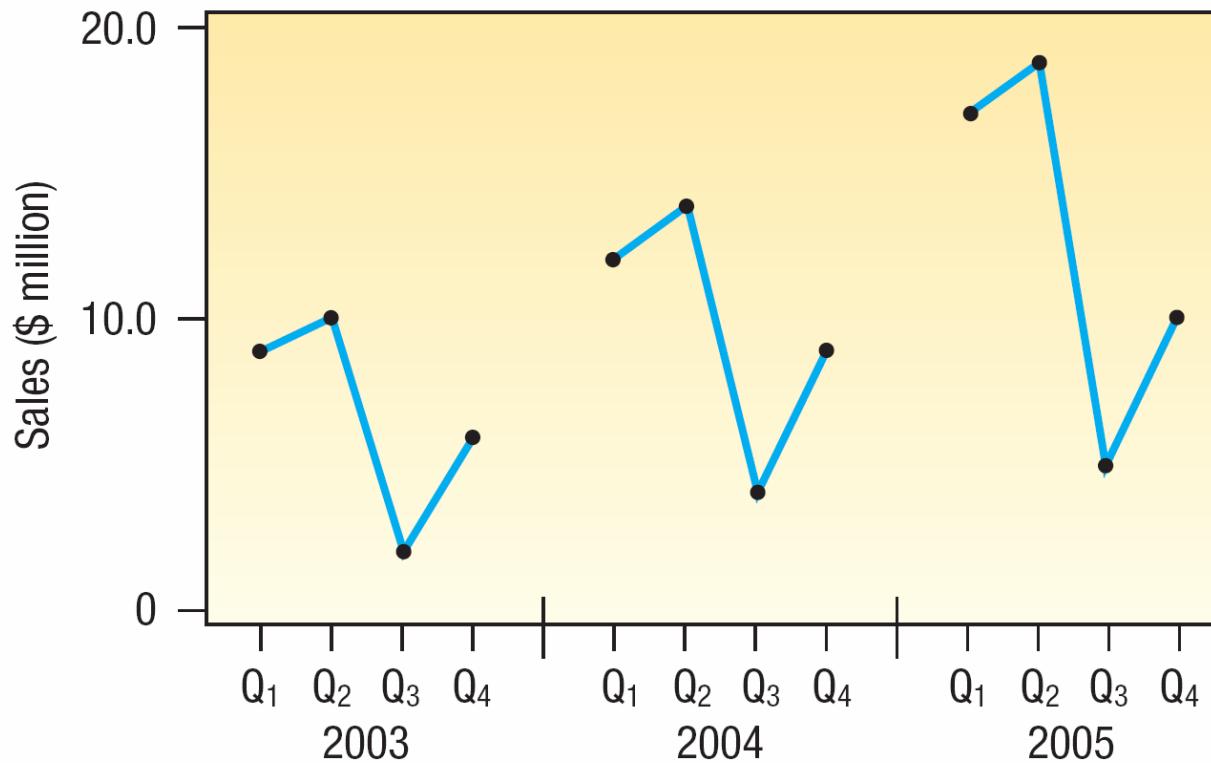
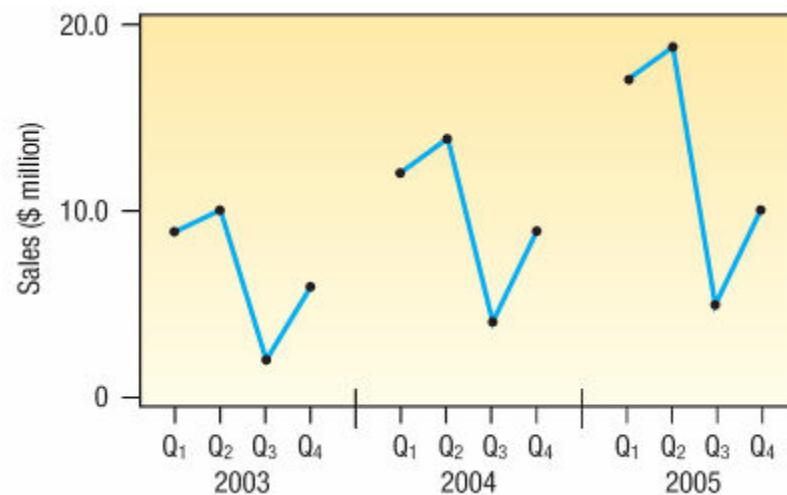


CHART 16–2 Sales of Baseball and Softball Equipment, Hercher Sporting Goods, 2003–2005 by Quarter

Seasonal Variation

- One of the **components** of a time series
- Seasonal variations are fluctuations that coincide with certain seasons and are **repeated year after year**
- Understanding seasonal fluctuations **help plan for sufficient goods and materials** on hand to meet varying seasonal demand
- Analysis of seasonal fluctuations over a period of years **help in evaluating current sales**

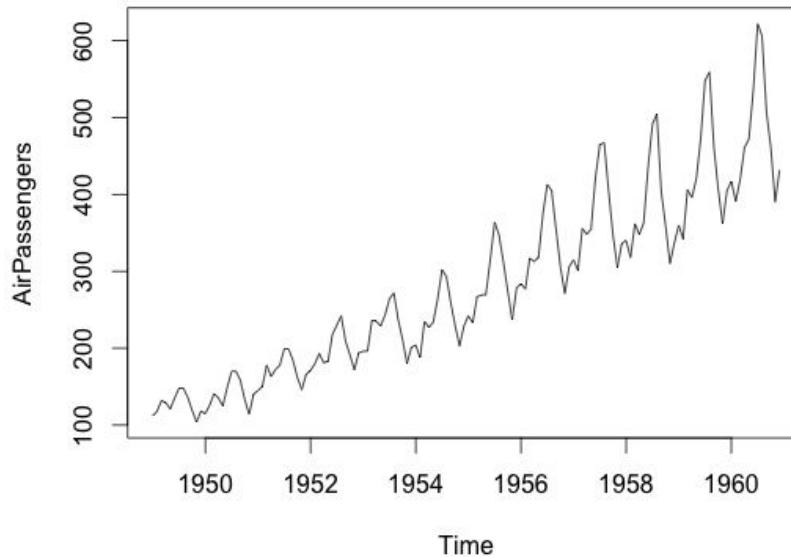


The Moving Average Method

- Useful in smoothing time series to see its **trend**
- Basic method used in **measuring seasonal fluctuation**
- Applicable when time series follows fairly linear trend that have definite rhythmic pattern

Box-Jenkins Method: What is it?

- Models historical behavior to forecast the future



- Applies **ARMA** (Autoregressive Moving Averages)
 - ▶ **Input:** Time Series
 - ▶▶ Accounting for *Trends* and *Seasonality* components
 - ▶ **Output:** Expected future value of the time series

ARIMA

- This acronym is descriptive, capturing the key aspects of the model itself:
 - ▶ **AR:** *Autoregression.* A model that uses the dependent relationship between an observation and some number of lagged observations.
 - ▶ **I:** *Integrated.* The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
 - ▶ **MA:** *Moving Average.* A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

- A standard notation is used of ARIMA(p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.
- The parameters of the ARIMA model are defined as follows:
 - ▶ **p**: The number of lag observations included in the model, also called the lag order.
 - ▶ **d**: The number of times that the raw observations are differenced, also called the degree of differencing.
 - ▶ **q**: The size of the moving average window, also called the order of moving average.

Use Cases

Forecast:

- Next month's sales
- Tomorrow's stock price
- Hourly power demand



Modeling a Time Series

- Let's model the time series as

$$Y_t = T_t + S_t + R_t, \quad t=1, \dots, n.$$

- T_t:** Trend term

- Air travel steadily **increased** over the last few years

- S_t:** The seasonal term

- Air travel **fluctuates** in a regular pattern over the course of a year

- R_t:** Random component

- To be modeled with ARMA

Stationary Sequences

- Assumes the random component is a ***stationary sequence***
 - ▶ Constant mean
 - ▶ Constant variance
 - ▶ Autocorrelation does not change over time
 - ▶▶ Constant correlation of a variable with itself at different times
- In practice, to obtain a ***stationary sequence***, the data must be:
 - ▶ De-trended
 - ▶ Seasonally adjusted

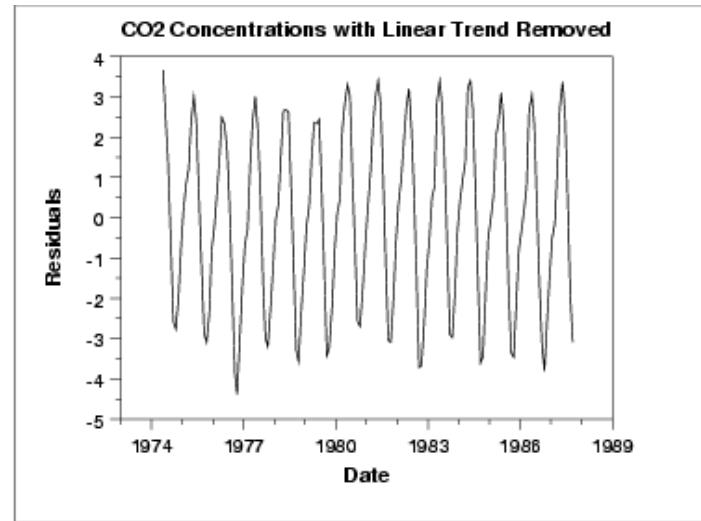
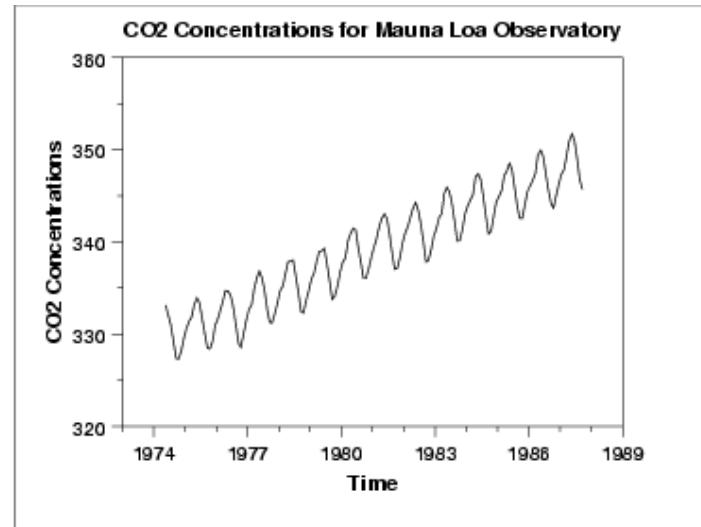
De-trending

- In this example, we see a linear **trend**, so we fit a linear model

► $T_t = m \cdot t + b$

- The **de-trended** series is then

► $Y_t^1 = Y_t - T_t$



Time Series Analysis - Reasons to Choose (+) & Cautions (-)



Reasons to Choose (+)	Cautions (-)
Minimal data collection Only have to collect the series itself Do not need to input drivers	No meaningful drivers: prediction based only on past performance No explanatory value Can't do "what-if" scenarios Can't stress test
Designed to handle the inherent autocorrelation of lagged time series	It's an "art form" to select appropriate parameters
Accounts for trends and seasonality	Only suitable for short term predictions

Time Series Analysis with R

- The function “*ts*” is used to create time series objects
 - ▶ **mydata<- ts(mydata,start=c(1999,1),frequency=12)**
- Visualize data
 - ▶ **plot(mydata)**
- De-trend using differencing
 - ▶ **diff(mydata)**
- Examine ACF and PACF
 - ▶ **acf(mydata)**: It computes and plots estimates of the autocorrelations
 - ▶ **pacf(mydata)**: It computes and plots estimates of the partial autocorrelations

Other Useful R Functions in Time Series Analysis

- **ar()**: Fit an autoregressive time series model to the data
- **arima()**: Fit an ARIMA model
- **predict()**: Makes predictions
 - ▶ “*predict*” is a generic function for predictions from the results of various model fitting functions. The function invokes particular methods which depend on the *class* of the first argument

Check Your Knowledge



Your Thoughts?

1. What is a time series and what are the key components of a time series?
2. How do we “de-trend” a time series data?
3. What makes data stationary?
4. How is seasonality removed from the data?
5. What are the modeling parameters in ARIMA?
6. How do you use ACF and PACF to determine the “stationarity” of time series data?



Module 4: Advanced Analytics – Theory and Methods

Lesson 7: Time Series Analysis - Summary

During this lesson the following topics were covered:

- Time Series Analysis and its applications in forecasting
- ARMA and ARIMA Models
- Implementing the Box-Jenkins Methodology using R
- Reasons to Choose (+) and Cautions (-) with Time Series Analysis

Lab Exercise 10: Time Series Analysis



This Lab is designed to investigate and practice Time Series Analysis with ARIMA models (Box-Jenkins-methodology).

After completing the tasks in this lab you should be able to:

- Use R functions for ARIMA models
- Apply the requirements for generating appropriate training data
- Validate the effectiveness of the ARIMA models

Lab Exercise 10: Time Series Analysis - Workflow

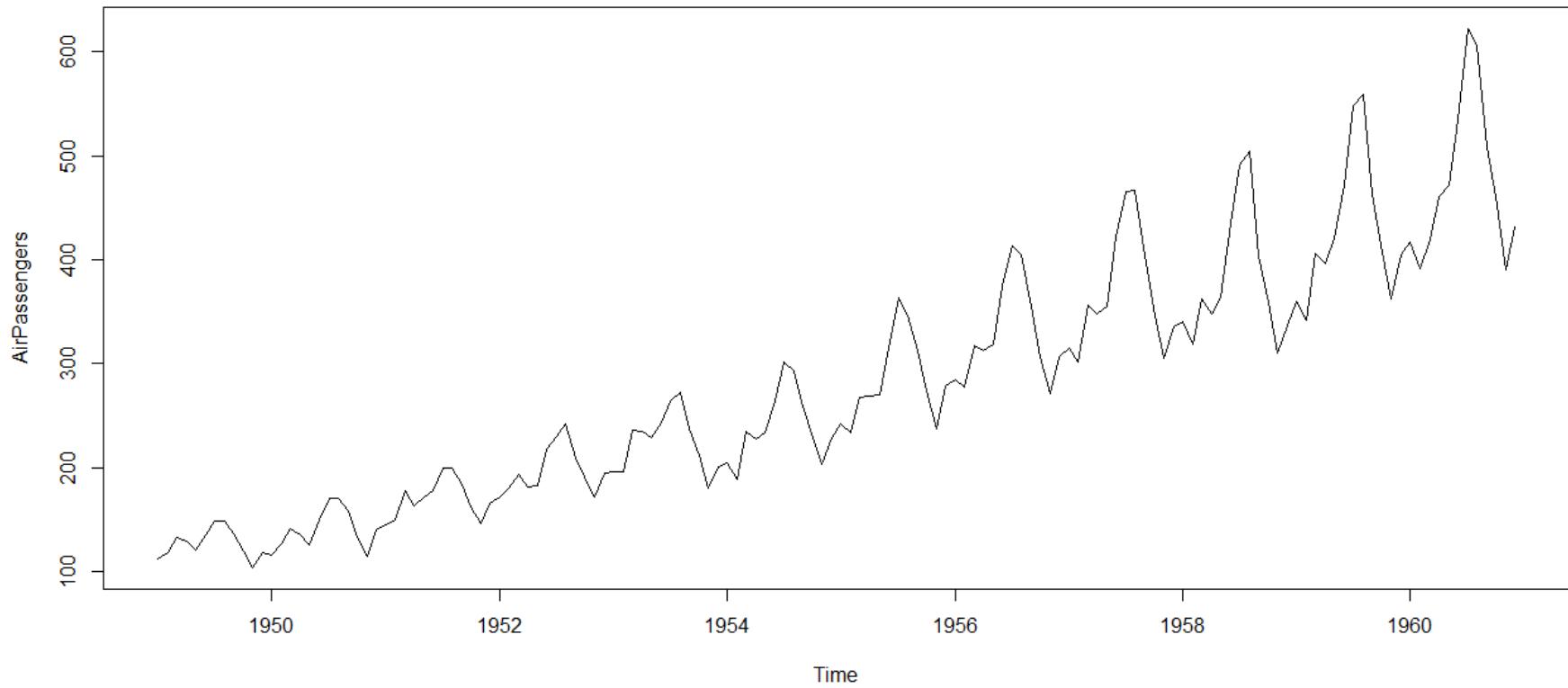
- 1 • Set the Working Directory
- 2 • Open Connection to Database
- 3 • Get Data from the Database
- 4 • Import the Table
- 5 • Review, Update, and Prepare DataFrame "msales" File for ARIMA Modeling
- 6 • Convert "sales" into Time Series Object
- 7 • Plot the Time Series
- 8 • Analyze the ACF and PACF
- 9 • Difference the Data to Make it Stationary
- 10 • Plot ACF and PACF for the Differenced Data
- 11 • Fit the ARIMA Model
- 12 • Generate Predictions
- 13 • Compare Predicted Values with Actual Values

Example

- `data(AirPassengers)`
- `# know the type of data`
- `class(AirPassengers)`
- `start(AirPassengers)`
- `end(AirPassengers) #The cycle of this time series is 12months in a year`
- `summary(AirPassengers)`

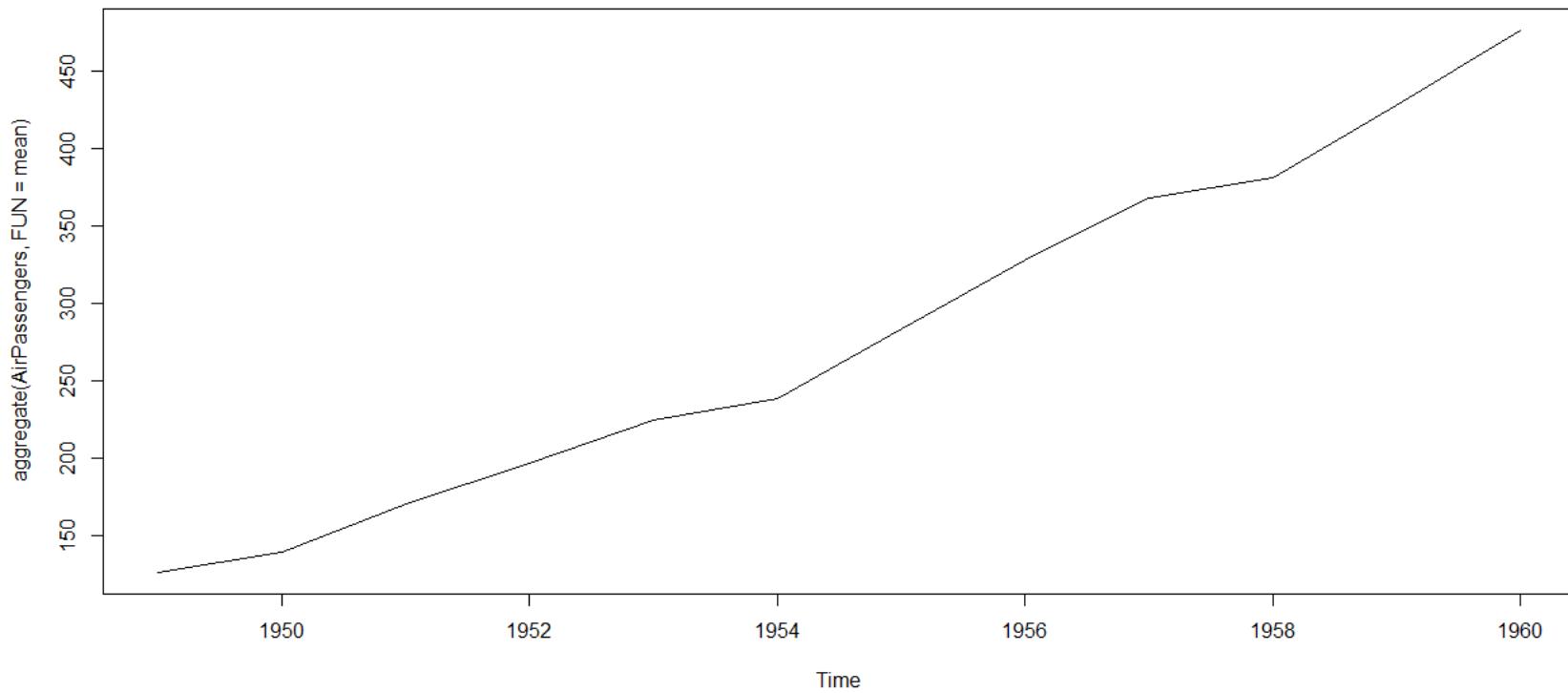
Plot time series

- `plot(AirPassengers)`



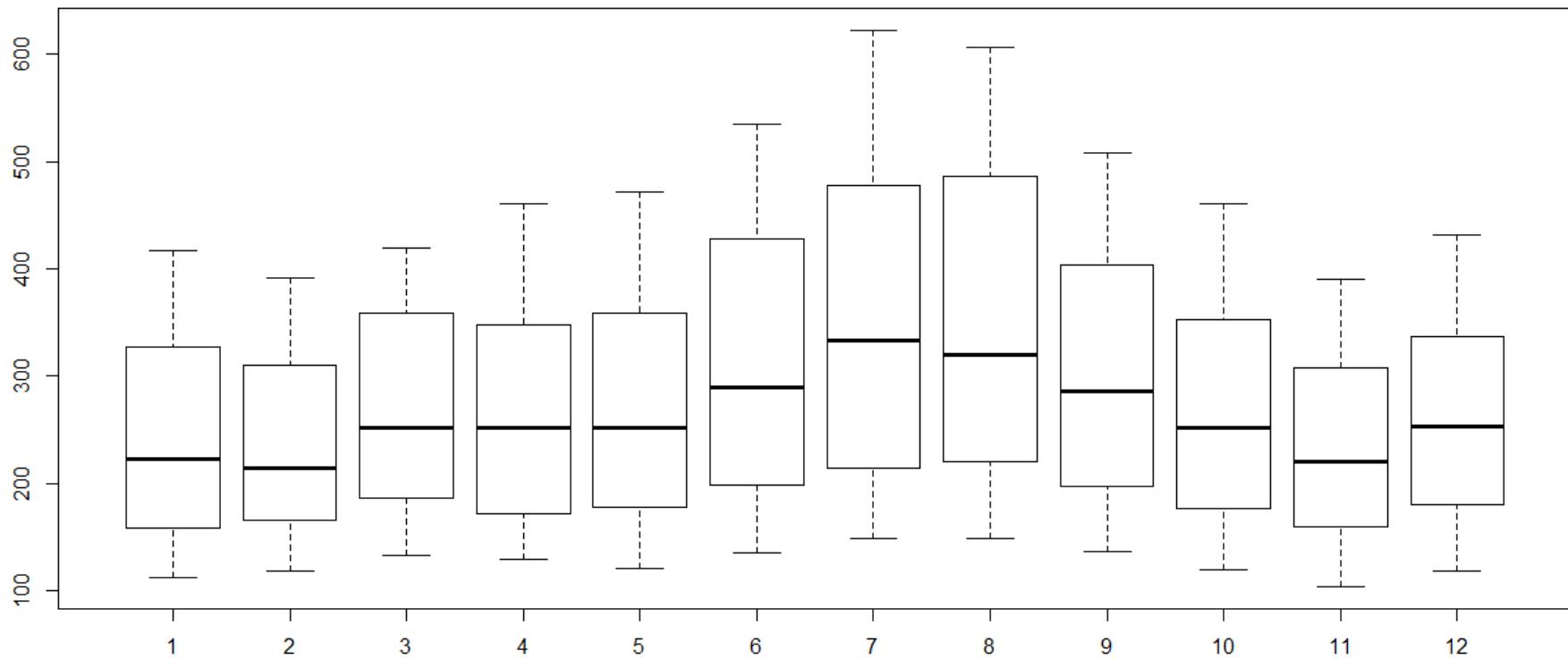
- `cycle(AirPassengers)`
- #This will print the cycle across years.

- `plot(aggregate(AirPassengers,FUN=mean))`



Box Plot

- #Box plot across months will give us a sense on seasonal effect
- `boxplot(AirPassengers~cycle(AirPassengers))`



ARIMA Modelling

- #General seasonal ARIMA models: $(0,1,1) \times (0,1,1)$

```
(fit <- arima(log(AirPassengers), c(0, 1, 1), seasonal = list(order =  
c(0, 1, 1), period = 12)))
```

Prediction Use Arima

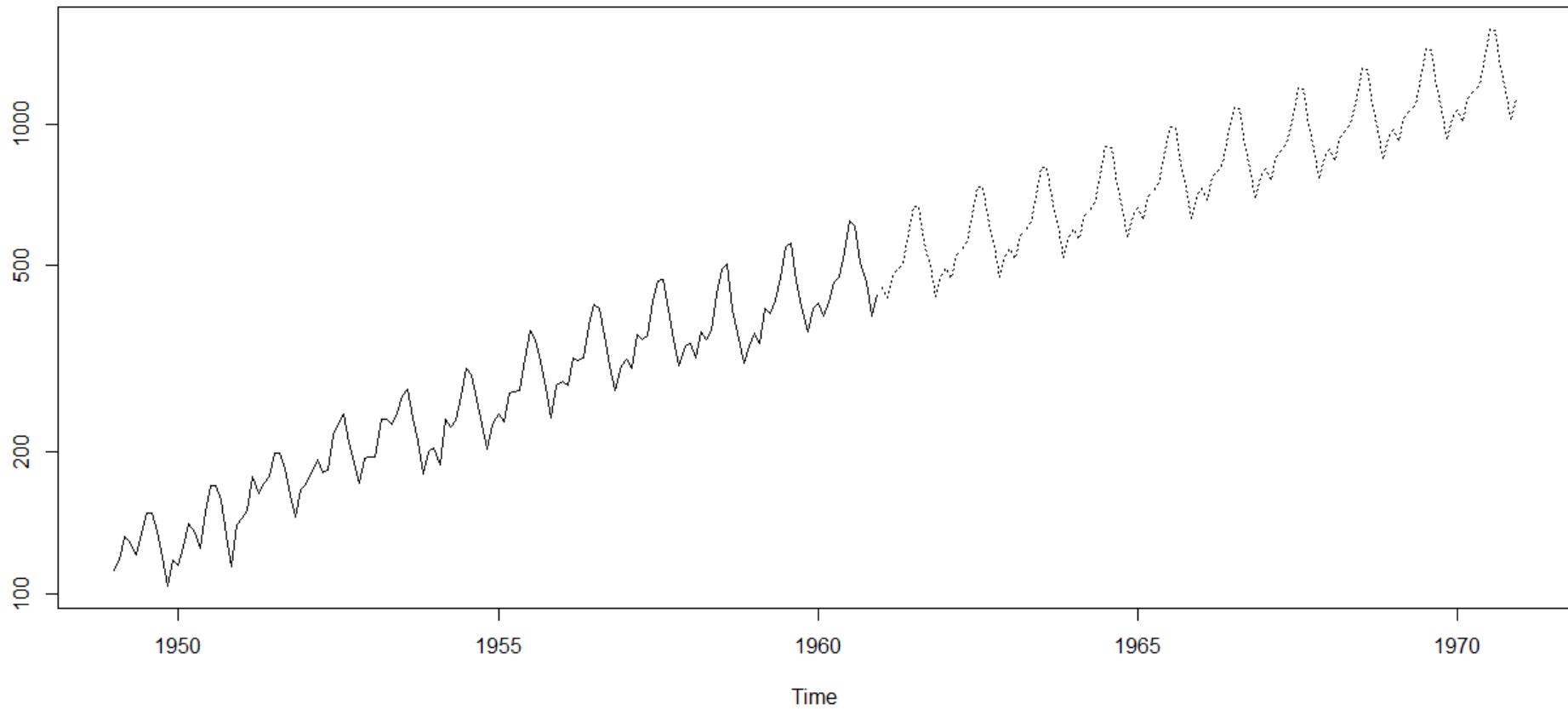
- `pred <- predict(fit, n.ahead = 10*12)`

Plot the Prediction

- #line type (lty) can be specified using either text ("blank", "solid", "dashed", "dotted", "dotdash", "longdash", "twodash") or number (0, 1, 2, 3, 4, 5, 6).
- #Note that lty = "solid" is identical to lty=1.

```
ts.plot(AirPassengers,2.718^pred$pred, log = "y", lty = c(1,3))
```

Prediction for next 10 years



Print real predicted values

- `print(2.718^pred$pred)`

```
Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      DEC  
1961 450.1371 425.4501 478.7004 492.0881 508.7261 582.9599 669.5589 666.6280 557.8234 496.8878 429.6018 476.9375
```

#explanations for the `ts.plot` arguments provided: # $2.718^{\text{pred$pred}}$: we are undoing the log from the values.

In order to do that, we need to find the log inverse of what we have got. #i.e.

```
log(forecast) = APforecast$pred #hence, #e=2.718
```

#`log = "y"` is to plot on a logarithmic scale
#`lty = c(1,3)` will set the `LineTYpe` to 1 (for solid)
for the original time series and 3 (for dotted)
for the predicted time series.

`print(2.718^pred$pred)` would give us the actual predicted values.

Other examples

- # <https://otexts.com/fpp2/lag-plots.html>
- #2.1 ts objects
- `y <- ts(round(runif(40, 50,100)), start=2012, frequency=4)`
- `y`
- `plot(y)`

Other examples

Assume you are working as data scientist in a car company and you have given the following data set, representing the sales of cars over the past years from 2015 to 2019 distributed over 4 quarters in each year:

```
numOfCarSales <- c(200, 210, 199, 222,  
+ 220, 230, 234, 223,  
+ 230, 250, 240, 244,  
+ 240, 260, 250, 277)
```

```
my_ts<-ts(numOfCarSales, start = 2015, frequency = 4)
```

```
my_ts
```

```
library(ggplot2)
```

```
library(xts)
```

```
library(fpp)
```

```
autoplot(my_ts)
```

```
autoplot(my_ts, facts = TRUE)
```

- library(ggplot2)
 - library(xts)
 - library(fpp)
-
- data(melsyd)
 - autoplot(melsyd[,"Economy.Class"]) +
 - ggttitle("Economy class passengers: Melbourne-Sydney") +
 - xlab("Year") +
 - ylab("Thousands")

a10 Monthly time series for Anti-Diabetic Drug Sales

a10 Monthly time series for Anti-Diabetic Drug Sales In Australia
From **1991 To 2008**.

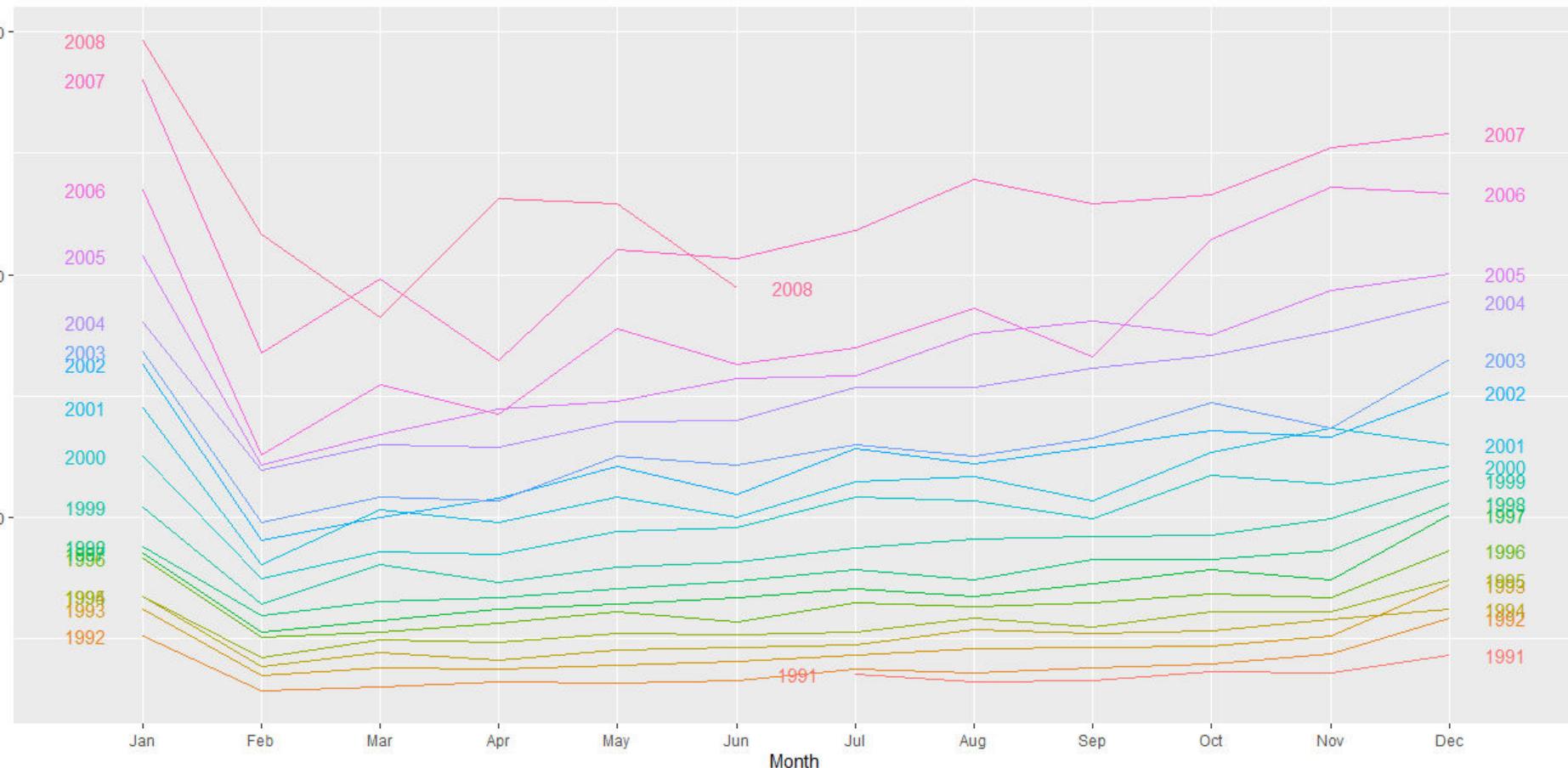
```
autoplot(a10) +  
  ggtitle("Antidiabetic drug sales") +  
  ylab("$ million") +  
  xlab("Year")
```

Seasonal plots

- ggseasonplot(a10, year.labels=TRUE, year.labels.left=TRUE) +
- ylab("\$ million") +
- ggtitle("Seasonal plot: antidiabetic drug sales")

Ggseasonplot on a10

Seasonal plot: antidiabetic drug sales



Dataset: Sales, budget and GDP dataset

- download *tute1.csv* from B.B
- Then upload dataset to R
- Explore the dataset
- Exploration in R:
 - ▶ `setwd("C:\\Users\\z10095\\Desktop\\")`
 - ▶ `tute1 <- read.csv("tute1.csv", header=TRUE)`
 - ▶ `View(tute1)`
 - ▶ `Head(tute1)`

Explore data

➤ `head(tute1)`

	<code>X</code>	<code>Sales</code>	<code>AdBudget</code>	<code>GDP</code>
1	Mar-81	1020.2	659.2	251.8
2	Jun-81	889.2	589.0	290.9
3	Sep-81	795.0	512.5	290.8
4	Dec-81	1003.9	614.1	292.4
5	Mar-82	1057.7	647.2	279.1
6	Jun-82	944.4	602.0	254.0

Modify the data

- # (The `[,-1]` removes the first column which contains the quarters as we don't need them now.)

```
mytimeseries <- ts(tute1[,-1], start=1981, frequency=4)
```

Mytimeseries

```
➤ head(mytimeseries)
```

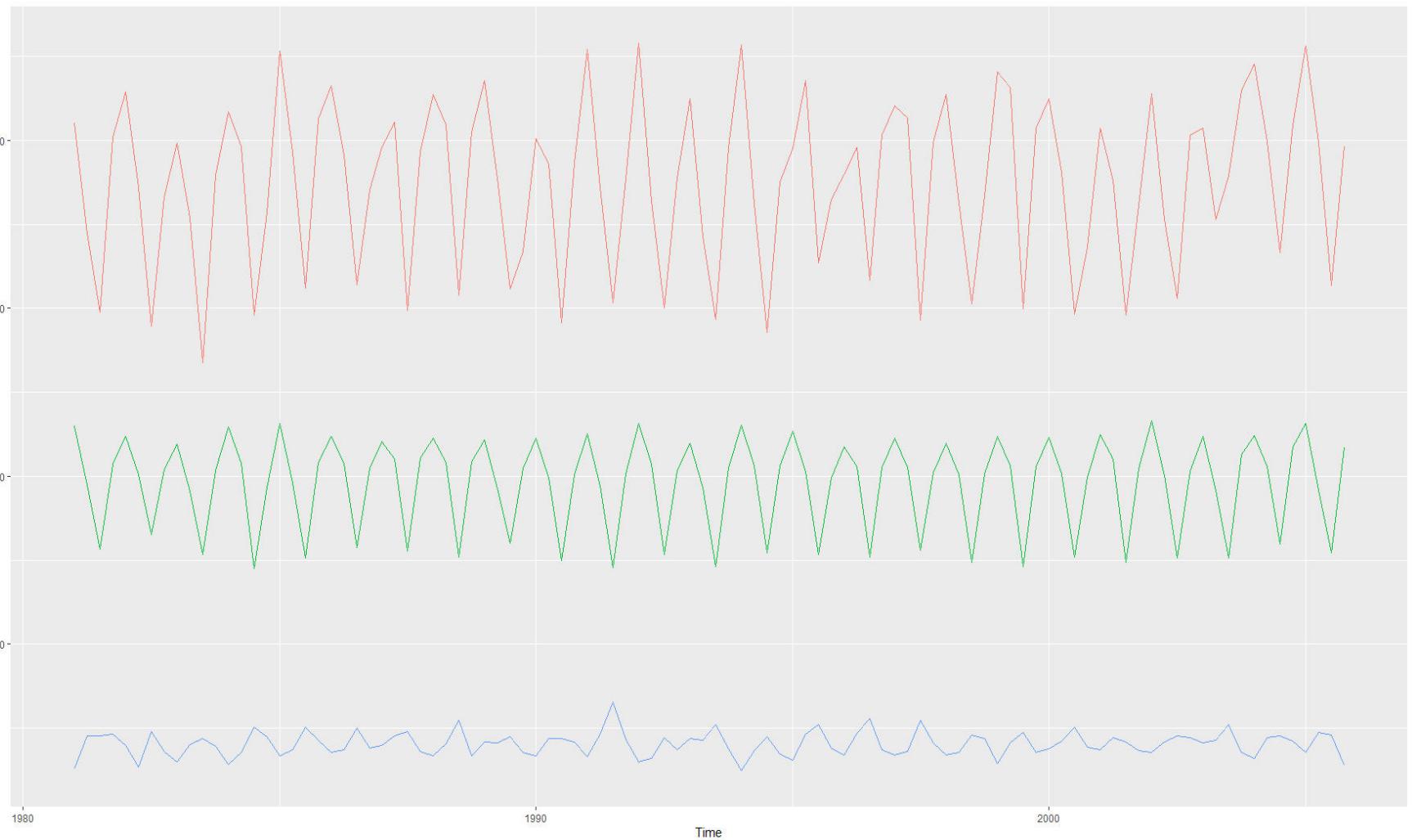
		Sales	AdBudget	GDP
1981	Q1	1020.2	659.2	251.8
1981	Q2	889.2	589.0	290.9
1981	Q3	795.0	512.5	290.8
1981	Q4	1003.9	614.1	292.4
1982	Q1	1057.7	647.2	279.1
1982	Q2	944.4	602.0	254.0

Upload the necessary libraries

- Install these packages if not already installed
 - ▶ `install.packages()`
- `library(ggplot2)`
- `library(xts)`
- `library(fpp)`

Construct time series plots

- `autoplot(mytimeseries)`

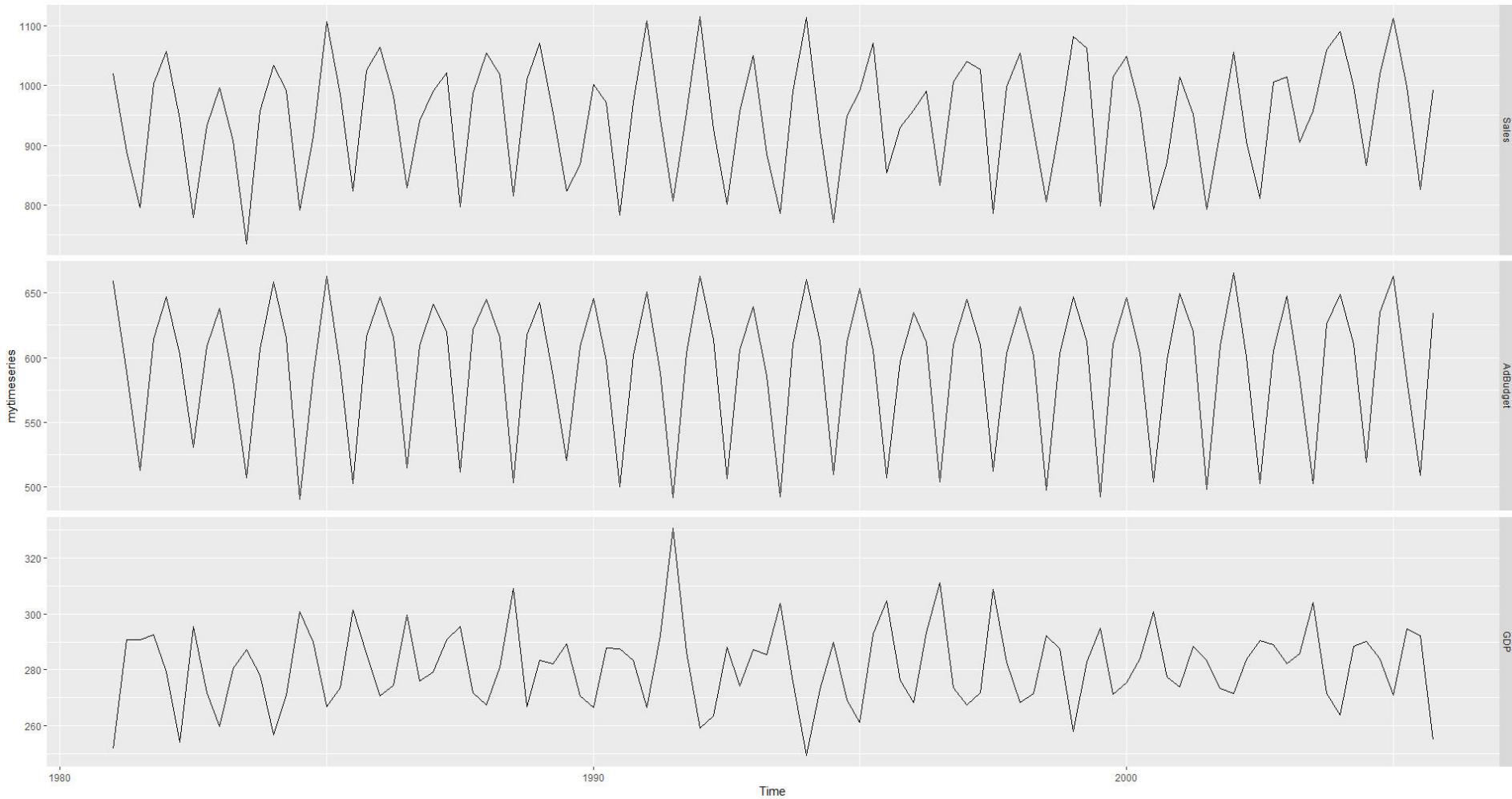


Construct time series plots of each of the three

series

#Construct time series plots of each of the three series

```
autoplots(mytimeseries, facets=TRUE)
```



Dataset: Monthly Australian retail

- `setwd("C:\\Users\\z10095\\Desktop\\")`
- Upload data file **retail.xlsx** from BB to R
- Since the file is in Excel, you need to install a new package as shown below
 - ▶ Use the following instructions:
 - ▶ `#install.packages("readxl")`
 - ▶ Since the dataset has two headers, skip the first row
 - ▶ `retaildata <- readxl::read_excel("retail.xlsx", skip=1) # skip first row`
 - ▶ Check if the data correctly uploaded to R

Explore data

View(retaildata)

KStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

rets Untitled4* data2011 ARMA time Series.R TimeSeriesOnBB.R timeSeries_good source.R TimeSeries_Excellent.R retaildata tute1

Filter

	Series ID	A3349335T	A3349627V	A3349338X	A3349398A	A3349468W	A3349336V	A3349337W	A3349397X	A3349399C	A3349874C	A3349871W	A3349790V	A33495
1	1982-04-01	303.1	41.7	63.9	408.7	65.8	91.8	53.6	211.3	94.0	32.7	126.7	178.3	
2	1982-05-01	297.8	43.1	64.0	404.9	65.8	102.6	55.4	223.8	105.7	35.6	141.3	202.8	
3	1982-06-01	298.0	40.3	62.7	401.0	62.3	105.0	48.4	215.7	95.1	32.5	127.6	176.3	
4	1982-07-01	307.9	40.9	65.6	414.4	68.2	106.0	52.1	226.3	95.3	33.5	128.8	172.6	
5	1982-08-01	299.2	42.1	62.6	403.8	66.0	96.9	54.2	217.1	82.8	29.4	112.3	169.6	
6	1982-09-01	305.4	42.0	64.4	411.8	62.3	97.5	53.6	213.4	89.4	32.2	121.6	181.4	
7	1982-10-01	318.0	46.1	66.0	430.1	66.2	99.3	58.0	223.5	83.3	31.9	115.2	173.9	
8	1982-11-01	334.4	46.5	65.3	446.2	68.9	107.8	67.2	243.9	99.3	35.0	134.3	206.6	
9	1982-12-01	389.6	53.8	77.9	521.3	90.8	155.5	146.3	392.6	142.9	51.7	194.6	346.6	
10	1983-01-01	311.4	43.8	65.1	420.3	58.0	95.1	66.6	219.7	78.5	31.4	109.8	135.3	
11	1983-02-01	327.2	39.3	62.3	428.8	63.7	105.1	59.2	228.0	72.9	29.4	102.3	144.2	
12	1983-03-01	350.9	43.4	65.7	460.0	66.0	124.1	67.3	257.5	93.3	34.2	127.5	180.5	
13	1983-04-01	323.4	43.7	61.9	429.0	58.3	112.3	57.7	228.2	111.2	39.4	150.6	199.4	
14	1983-05-01	316.6	42.3	63.7	422.6	67.8	120.5	64.9	253.2	112.5	41.4	153.9	200.5	
15	1983-06-01	325.4	40.4	64.9	430.6	64.2	115.0	58.6	237.8	103.6	37.1	140.7	175.2	
16	1983-07-01	323.1	41.6	69.5	434.2	60.8	111.7	58.8	231.3	97.4	34.1	131.5	181.4	
17	1983-08-01	338.1	42.2	67.9	448.2	64.8	117.2	64.8	246.9	96.3	34.0	130.2	179.7	
18	1983-09-01	330.6	42.5	67.5	440.6	65.1	106.9	68.7	240.7	105.6	37.2	142.9	185.0	
19	1983-10-01	351.1	45.0	66.0	467.1	66.0	114.4	64.1	248.0	97.0	37.0	132.7	184.4	

Showing 1 to 19 of 381 entries

Apply **ts()** function to the dataset also define the start date

```
# data starts in April (4): start=c(1982,4)
```

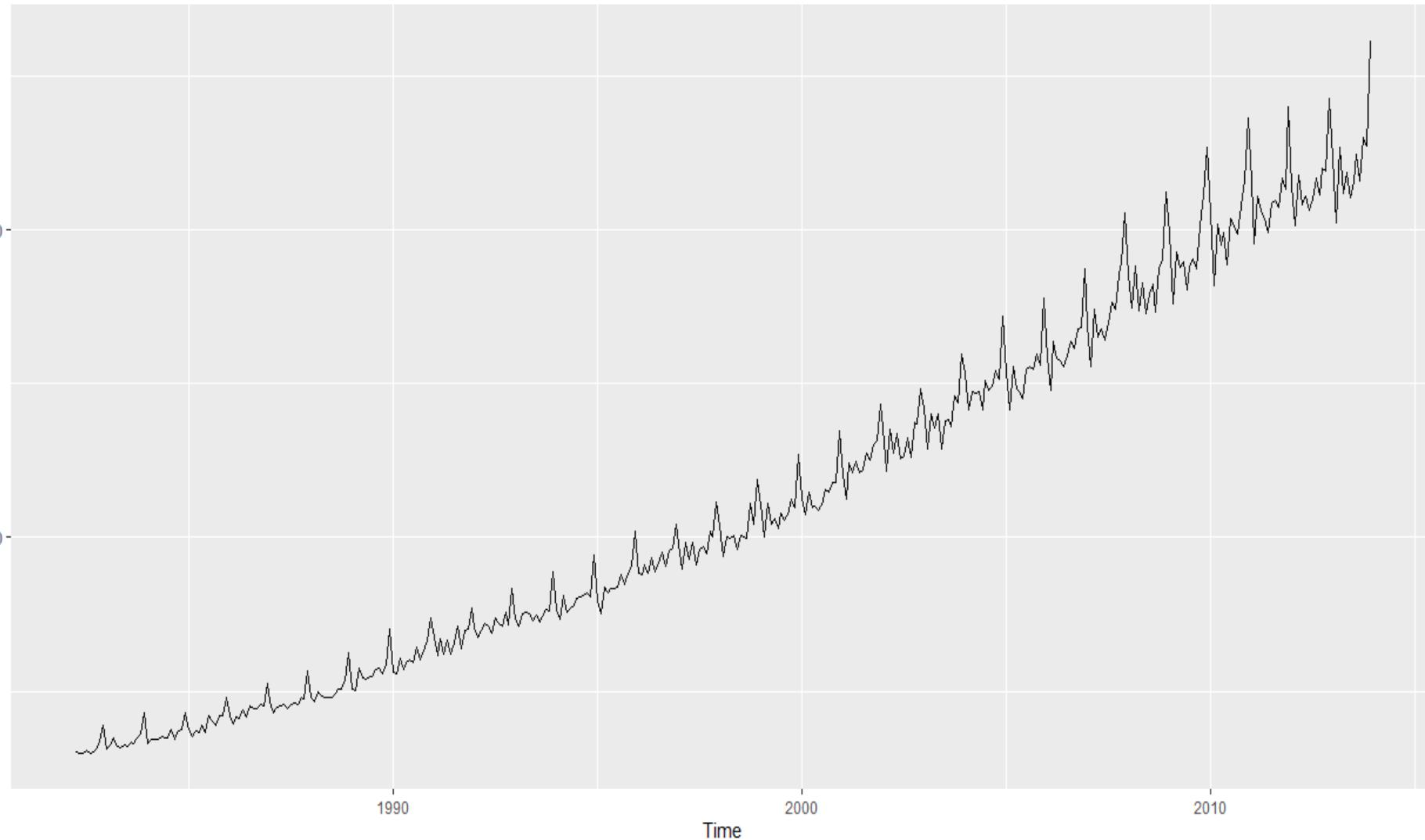
```
# here we applied on one case:
```

A3349335T: New South Wales ; Supermarket and grocery stores

```
myts <- ts(retaildata[,"A3349335T"], frequency=12, start=c(1982,4))
```

```
myts
```

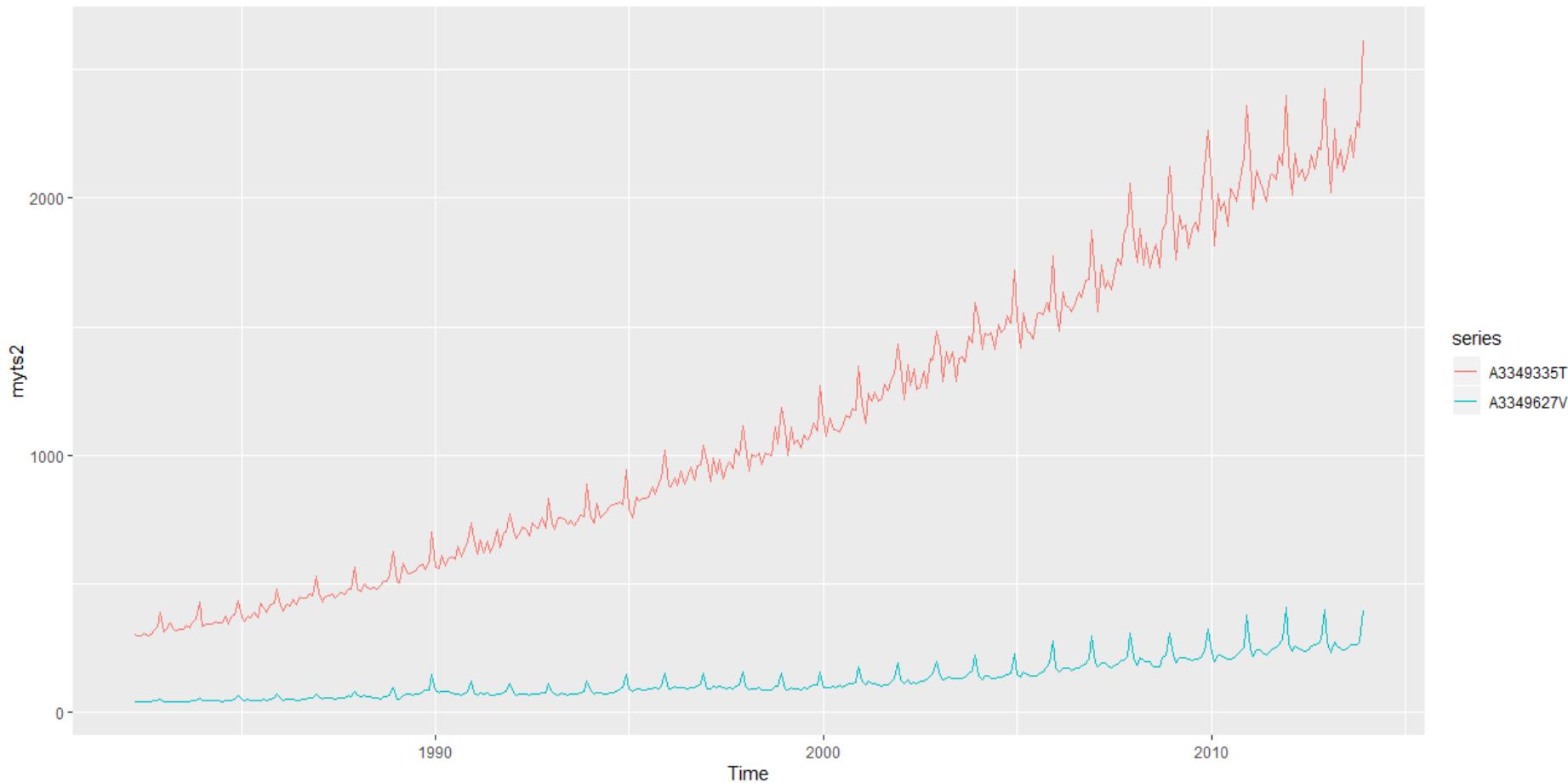
Plot the time series autplot(myts)



Plot the time for two selected series

#Construct time series plots of each of the two series

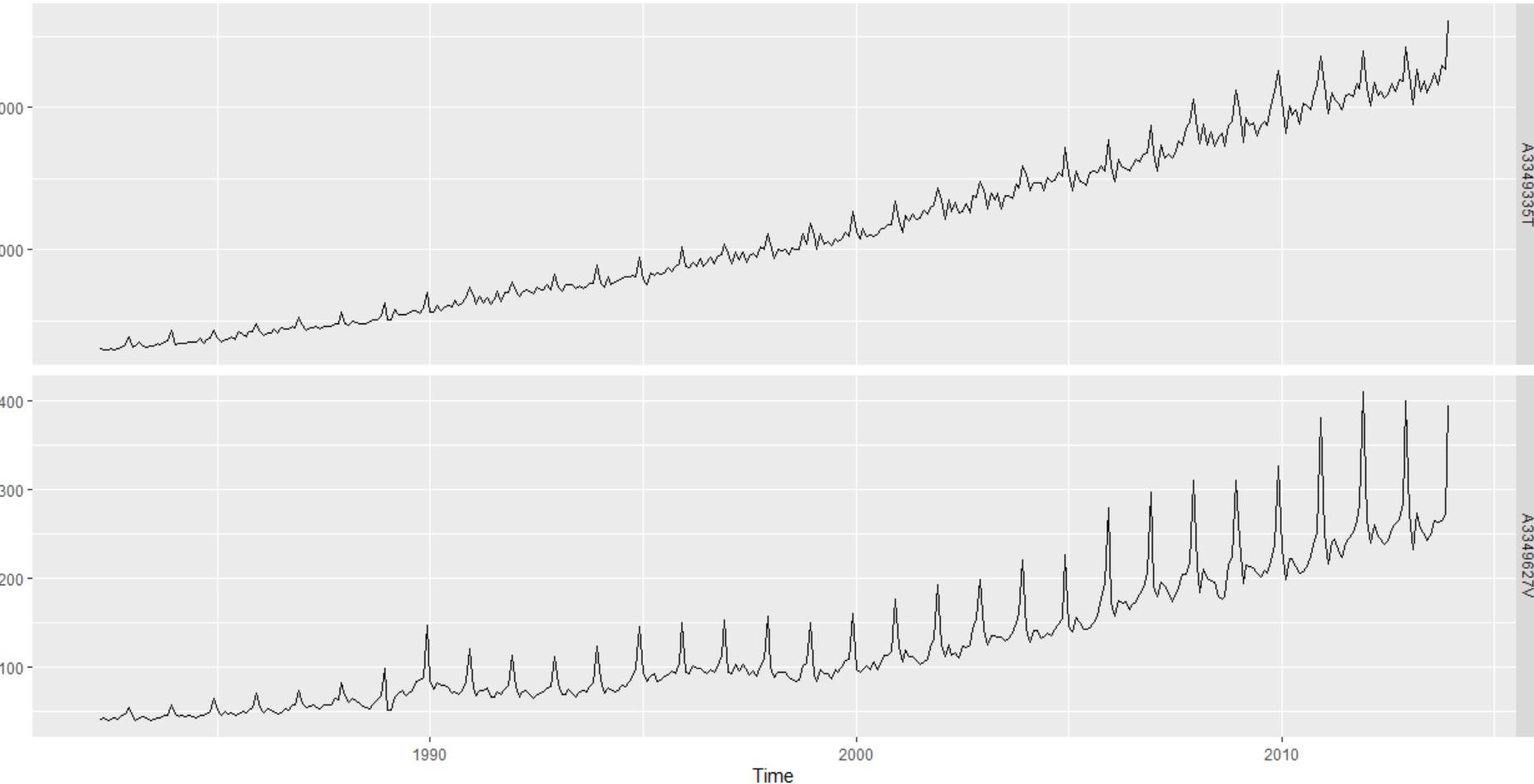
```
myts2 <- ts(retaildata[,c("A3349335T","A3349627V")], frequency=12, start=c(1982,4))  
myts2  
autoplot(myts2)
```



Plot the time for two selected series using *facets*

```
#Construct time series plots of each of the two series using facets
```

```
autoplots(myts2, facets=TRUE)
```



References

- ref: <https://otexts.com/fpp2/graphics-exercises.html>
- <https://otexts.com/fpp2/graphics-exercises.html>
- <https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html>
- <https://rpubs.com/Const4nce/223827>

CHAPTER 9

TEXT MINING AND ANALYSIS

Dr. Feras Al-Obeidat



Introduction



Analytics Lifecycle



Basic Methods



Adv. Methods



Tools



Lab

Module 4: Advanced Analytics – Theory and Methods

Lesson 8: Text Analysis

During this lesson the following topics are covered:

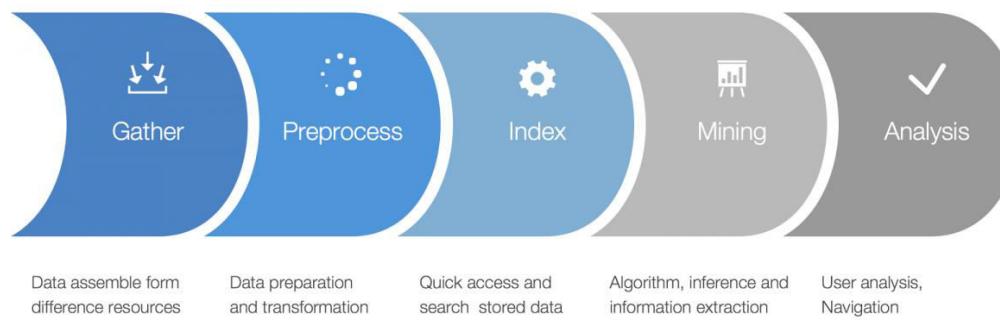
- Challenges with text analysis
- Key tasks in text analysis
- Definition of terms used in text analysis
 - Term frequency, inverse document frequency
- Representation and features of documents and corpus
- Use of regular expressions in parsing text
- Metrics used to measure the quality of search results
 - Relevance with tf-idf, precision and recall

Intro to Text Mining

- Text mining, also known as text analysis, is the process of transforming unstructured text data into meaningful and actionable information.
- Data helps companies get smart insights on people's opinions about a product or service.**
Think about all the potential ideas that you could get from analyzing emails, product reviews, social media posts, customer feedback, support tickets, etc. On the other side, there's the dilemma of how to process all this data. And that's where text mining plays a major role.

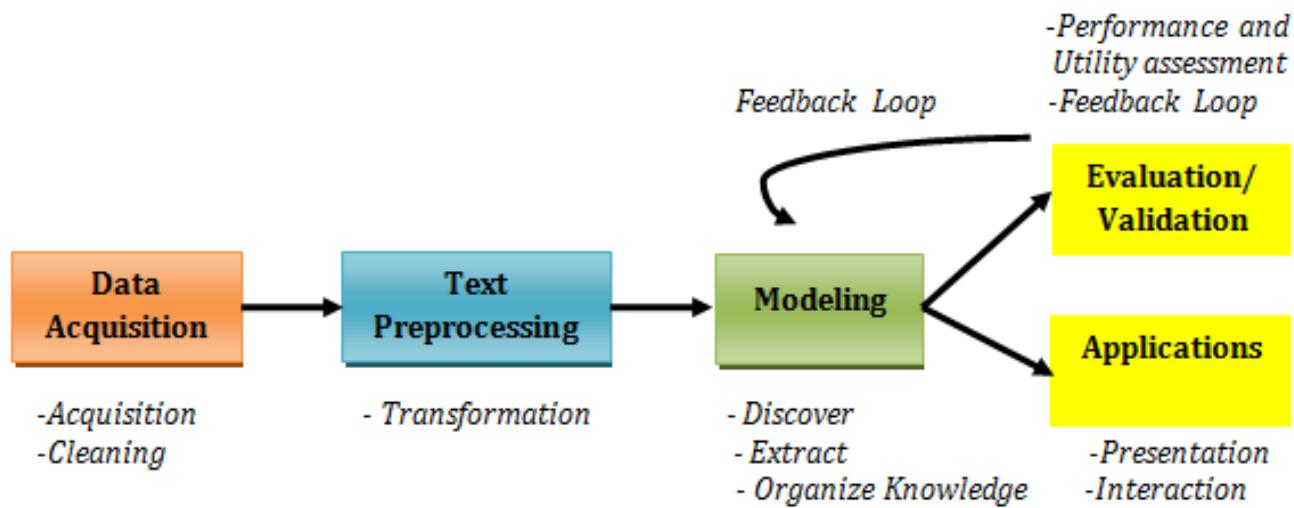
Text Mining

Text mining involves a series of activities to be performed in order to efficiently mine the information. These activities are:



Text Mining process

- **Text mining** combines notions of statistics, linguistics, and machine learning to create models that learn from training data and can predict results on new information based on their previous experience.



Text Analytics process

- **Text analytics**, on the other hand, uses results from analyses performed by text mining models, to create graphs and all kinds of data visualizations.



Basic Methods: Word Frequency

Word frequency can be used to identify the most recurrent terms or concepts in a set of data.

Finding out the most mentioned words in unstructured text can be particularly useful when analyzing customer reviews, social media conversations or customer feedback.

- For example, if the words “Expensive”, “Overpriced”, and “Overrated” frequently appear on your customer reviews, it may indicate you need to adjust your prices (or your target market!)



Basic Methods: Collocation

Collocation refers to a sequence of words that commonly appear near each other. The most common types of collocations are unigram, bigrams and trigrams

- **Bigrams** are pair of words that are likely to go together, like “**Get started**”, “**Save time**”, or “**Decision making**”.
- **Trigrams** are a combination of three words, like “**Within walking distance**” or “**Keep in touch**”.

Identifying collocations – and counting them as one single word – improves the granularity of the text, allows a *better understanding* of its semantic structure and, in the end, *leads to more accurate text mining results*.

Basic Methods: Concordance

- **Concordance** is used to recognize the particular context or instance in which a word or set of words appears. We all know that the human language can be ambiguous: the same word can be used in many different contexts. Analyzing the concordance of a word can help understand its exact meaning based on context.
- For example, here are a few sentences extracted from a set of re

Preceding context	Target	Following context
It saves time and helps teams	work	more efficiently.
Some advanced features only	work	in one language (English)
It enables us to	work	towards better conversion and retention.
We recommend this to several of the small businesses we	work	with, and they are all happy with the results.

Advanced Methods: Text Extraction

- **Text extraction** is a text analysis technique that extracts specific pieces of data from a text, like **keywords, entity names, addresses, emails**, etc. By using text extraction, companies can avoid all the hassle of sorting through their data manually to pull out key information. Some of the main tasks of text extraction:
 - ▶ Keyword Extraction
 - ▶ Name Entity Recognition
 - ▶ Feature Extraction
- Most times, it can be useful to combine text extraction with text classification in the same analysis.

Text Extraction: Keyword Extraction

- **Keyword Extraction:** keywords are the most relevant terms within a text and can be used to summarize its content. Utilizing a keyword extractor allows you to index data to be searched, summarize the content of a text or create tag clouds among other things



Text Extraction: Name Entity Recognition

- **Named Entity Recognition** allows you to identify and extract the names of companies, organizations or persons from a text.

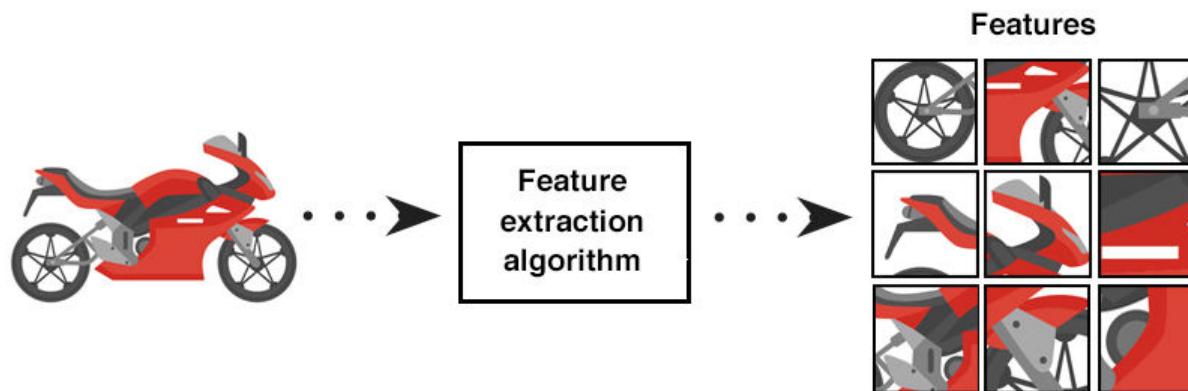
In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space - Alibaba GPE , Baidu ORG , and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space . The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the 'future AI PERSON platforms'. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL , with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE .

To further elaborate on the geographical trends, North America LOC has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG , IBM ORG , and Microsoft ORG .

“NORP” which represents “Nationalities or religious or political groups”
Geopolitical Entity (GPE).

Text Extraction: Feature Extraction

- **Feature Extraction** helps identify specific characteristics of a product or service in a set of data. For example, if you are analyzing product descriptions, you could easily extract features like “**colour**”, “**brand**”, “**model**”, etc.



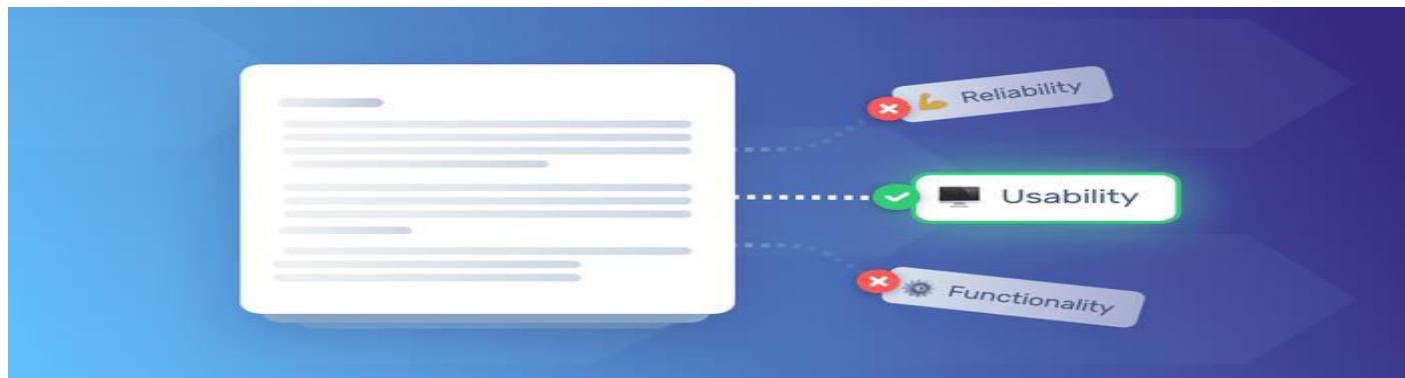
Advanced methods: Text Classification

- **Text classification** is the process of **assigning categories** (tags) to unstructured text data. This essential task of Natural Language Processing (NLP) makes it easy to organize and structure complex text, turning it into meaningful data. some of the most popular tasks of text classification are:

- ▶ **Topic Analysis**
- ▶ Language Detection
- ▶ Intent Detection
- ▶ Sentiment Analysis

Text Classification: Topic Analysis

- **Topic Analysis** (also called **topic detection**, **topic modelling**, or **topic extraction**) is a machine learning technique that organizes and understands large collections of text data, by assigning “tags” or categories according to each individual text’s topic or theme.
- For example, a support ticket saying “*My Online Order Hasn’t Arrived*” can be classified as “**Shipping Issues**”.



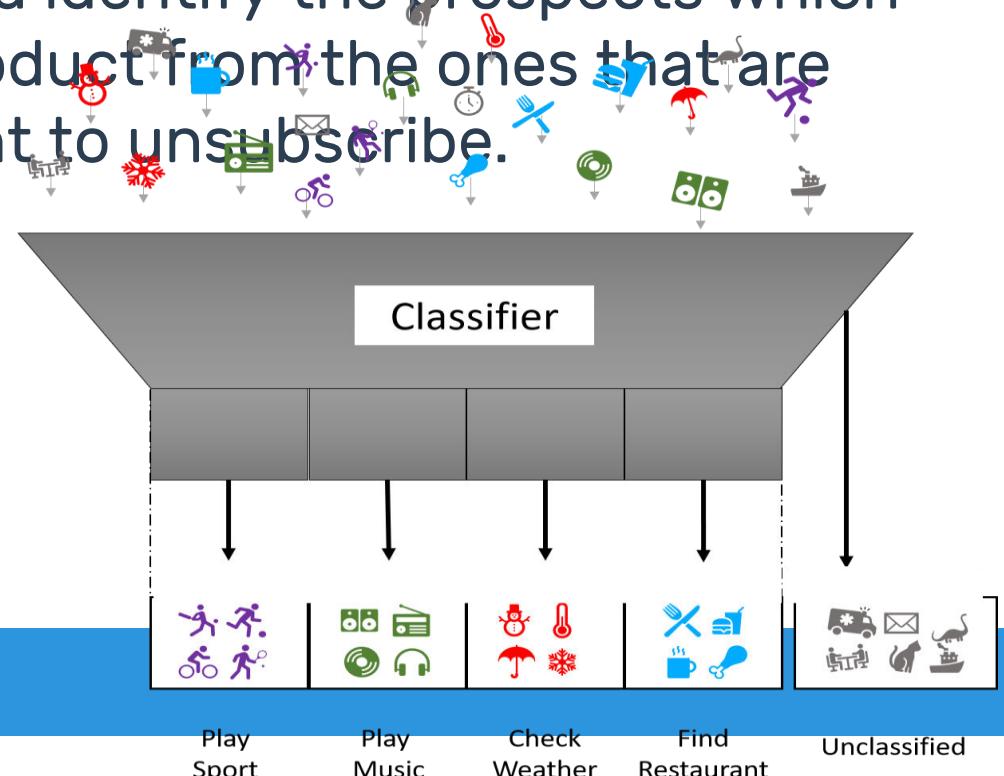
Text Classification: Language Detection

- **Language Detection** allows you to classify a text based on its language. One of its most useful applications is automatically routing support tickets to the right geographically located team. Automating this task is quite simple and helps teams save valuable time.



Text Classification: Intent Detection

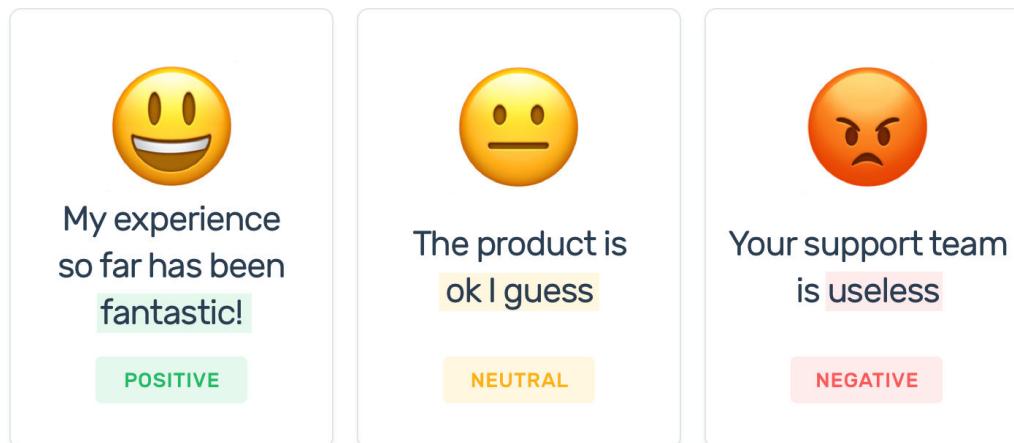
- You could use a text classifier to recognize the intentions or the purpose behind a text automatically. This can be particularly useful when *analyzing customer conversations*.
- For example, you could sift through different outbound sales email responses and identify the prospects which are interested in your product from the ones that are not, or the ones who want to unsubscribe.



Text Classification: Sentiment Analysis

- **Sentiment analysis** (also known **as opinion mining or emotion AI**) refers to the use of natural language processing to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is used for many applications, especially in business intelligence. Some examples of applications for sentiment analysis include:
 - ▶ Analyzing the social media discussion around a certain topic
 - ▶ Evaluating survey responses
 - ▶ Determining whether product reviews are positive or negative

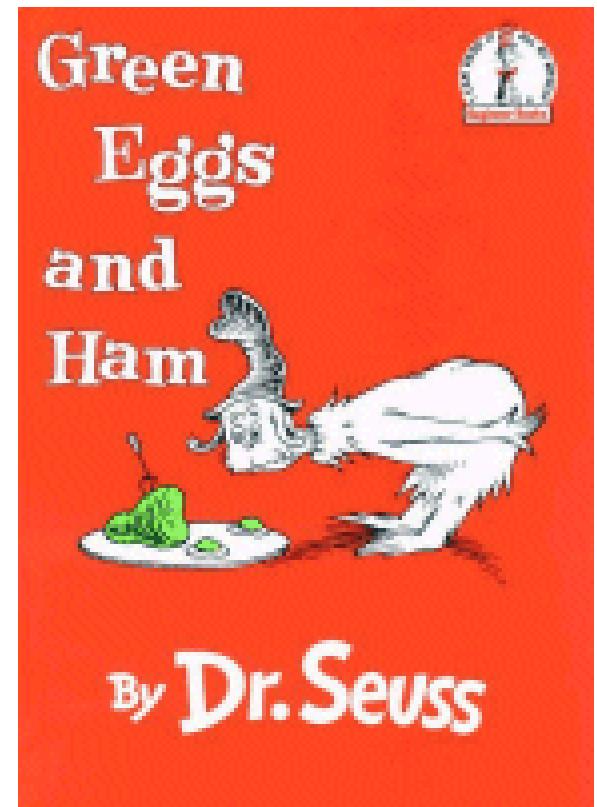
Sentiment Analysis



Text Analysis

Encompasses the processing and representation of text for analysis and learning tasks

- **High-dimensionality**
 - ▶ Every distinct term is a dimension
 - ▶ *Green Eggs and Ham*: A 50-D problem!
- **Data is Un-structured**



Text Analysis – Problem-solving Tasks

- **Parsing**

- ▶ Impose a structure on the unstructured/semi-structured text for downstream analysis

- **Search/Retrieval**

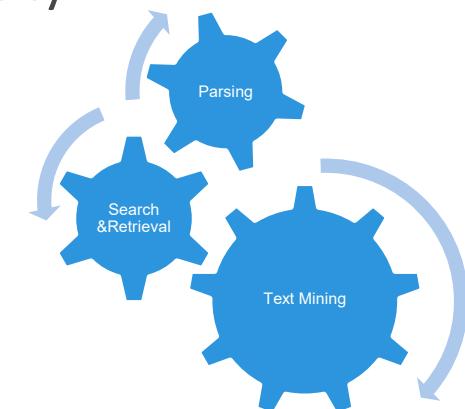
- ▶ Which documents have this **word** or phrase?
 - ▶ Which documents are about this **topic** or this entity?

- **Text-mining**

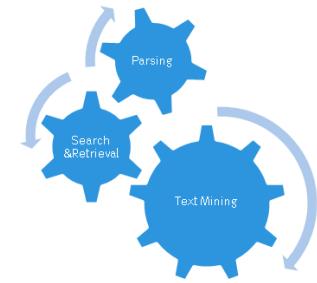
- ▶ "**Understand**" the content
 - ▶ **Clustering, classification**

- **Tasks are not an ordered list**

- ▶ Does not represent process
 - ▶ Set of tasks used appropriately depending on the problem addressed



Example: Brand Management



- **Acme** currently makes two products
 - ▶ bPhone
 - ▶ bEbook
- They have lots of competition. They want to maintain their reputation for excellent products and keep their sales high.
- What is the **buzz** on **Acme**?
 - ▶ Search for mentions of **Acme** products
 - ▶ Twitter, Facebook, Review Sites, etc.
 - ▶ What do people say?
 - ▶ Positive or negative?
 - ▶ What do people think is good or bad about the products?

Buzz Tracking: The Process



1. Monitor social networks, review sites for mentions of our products.	Parse the data feeds to get actual content. Find and filter the raw text for product names (Use Regular Expression).
2. Collect the reviews.	Extract the relevant raw text. Convert the raw text into a suitable document representation . Index into our review corpus .
3. Sort the reviews by product.	Classification (or " Topic Tagging ")
4. Are they good reviews or bad reviews? We can keep a simple count here, for trend analysis.	Classification (sentiment analysis)
5. Marketing calls up and reads selected reviews in full, for greater insight.	Search/Information Retrieval .

Parsing the Feeds

1. Monitor social networks, review sites for mentions of our products

- **Impose structure** on semi-structured data.
- We need to know where to **look for** what we are looking for.

```
<channel>
<title>All about Phones</title>
<description>My Phone Review Site</description>
<link>http://www.phones.com/link.htm</link>

<item>
<title>bPhone: The best!</title>
<description>I love LOVE my bPhone!</description>
<link>http://www.phones.com/link.htm</link>
<guid isPermaLink="false"> 1102345</guid>
<pubDate>Tue, 29 Aug 2011 09:00:00 -0400</pubDate>
</item>

</channel>
```

Regular Expressions

1. Monitor social networks, review sites for mentions of our products

- Regular Expressions (regexp) are a means for finding words, strings or particular patterns in text.
- A **match** is a Boolean response. The basic use is to ask “does this regexp match this string?”

regexp	matches	Note
b[P p]hone	bPhone, bphone	Pipe “ ” means “or”
bEbo*k	bEbk, bEbok, bEbook, bEboook ...	“*” matches 0 or more repetitions of the preceding letter
^I love	A line starting with "I love"	“^” means start of a string
Acme\$	A line ending with “Acme”	“\$” means the end of a string

Extract and Represent Text

2. Collect the reviews

Document Representation:

A structure for analysis

- **"Bag of words"**

- ▶ common representation
- ▶ A vector with one dimension for every unique term in space
 - ▶ **term-frequency (tf)**: number times a term occurs
- ▶ Good for basic search, classification

- **Reduce Dimensionality**

- ▶ **Term Space** – not ALL terms
 - ▶ no stop words: "the", "a"
 - ▶ often no pronouns
- ▶ **Stemming**
 - ▶ "phone" = "phones"

"I love LOVE my bPhone!"

Convert this to a vector in the term space:

acme	0
bebook	0
bPhone	1
fantastic	0
love	2
slow	0
terrible	0
terrific	0

Document Representation - Other Features

2. Collect the reviews

- Feature:
 - ▶ Anything about the document that is used for search or analysis.
- Title
- Keywords or tags
- Date information
- Source information
- Named entities

Representing a Corpus (Collection of Documents)

- Reverse index

2. Collect the reviews

- ▶ For every possible feature, a list of all the documents that contain that feature

- Corpus metrics

- ▶ Volume
- ▶ Corpus-wide term frequencies
- ▶ Inverse Document Frequency (**IDF**)
 - ▶ more on this later

- Challenge: a Corpus is **dynamic**

- ▶ Index, metrics must be updated continuously

Text Classification (I) - "Topic Tagging"



3. Sort the Reviews by Product

Not as straightforward as it seems

"The bPhone-5X has coverage everywhere. It's much less flaky than my old bPhone-4G."

"While I love Acme's bPhone series, I've been quite disappointed by the bEbook. The text is illegible, and it makes even my old Newton look blazingly fast."

"Topic Tagging"



3. Sort the Reviews by Product

Judicious choice of features

- ▶ Product mentioned in title?
- ▶ Tweet, or review?
- ▶ Term frequency
- ▶ Canonicalize abbreviations
 - ▶ "5X" = "bPhone-5X"

Text Classification (II) Sentiment Analysis



4. Are they good reviews or bad reviews?

- Naïve Bayes is a good first attempt
- But you need tagged training data!
 - ▶ The major bottleneck in text classification
- What to do?
 - ▶ Hand-tagging
 - ▶ Clues from review sites
 - ▶▶ thumbs-up or down, # of stars
 - ▶ Cluster documents, then label the clusters

Search and Information Retrieval



5. Marketing calls up and reads selected reviews in full, for greater insight.

- Marketing calls up documents with *queries*:
 - ▶ Collection of search terms
 - ▶ "bPhone battery life"
 - ▶ Can also be represented as "bag of words"
 - ▶ Possibly restricted by other attributes
 - ▶ within the last month
 - ▶ from this review site

Quality of Search Results



5. Marketing calls up and reads selected reviews in full, for greater insight.

- ▶ Is this document what I wanted?
- ▶ Used to rank search results
- Precision
 - ▶ What % of documents in the result are **relevant**?
- Recall
 - ▶ Of all the **relevant** documents in the corpus, what % were returned to me?

Computing Relevance (Term Frequency)



5. Marketing calls up and reads selected reviews in full, for greater insight.

- Assign each term in a document a weight for that term.
- The **weight** of a **term** t in a **document** d is a function of the number of times t appears in d .
 - ▶ The **weight** can be simply set to the number of occurrences of t in d :

$$tf(t, d) = count(t, d)$$

- ▶ The term frequency may optionally be normalized.

Inverse Document Frequency (idf)



5. Marketing calls up and reads selected reviews in full, for greater insight.

$$idf(t) = \log [N/df(t)]$$

- ▶ N : Number of documents in the corpus
- ▶ $df(t)$: Number of documents in the corpus that contain a term t
- Measures term uniqueness in corpus
 - ▶ "phone" vs. "brick"
- Indicates the **importance of the term**
 - ▶ Search (relevance)
 - ▶ Classification (discriminatory power)

TF-IDF and Modified Retrieval Algorithm



5. Marketing calls up and reads selected reviews in full, for greater insight.

term t in document d:

$$tfidf(t, d) = \mathbf{tf(t, d)} * \mathbf{idf(t)}$$

query: *brick, phone*

- Document with "brick" a few times more relevant than document with "phone" many times
- Measure of **Relevance with tf-idf**
- Call up all the documents that have any of the terms from the query, and sum up the tf-idf of each term:

$$\text{Relevance}(d) = \sum_{i \in [1, n]} tfidf(t_i, d)$$

TF-IDF and Modified Retrieval Algorithm, example

- The process to find meaning of documents using TF-IDF is very similar to Bag of words,
- Clean data / Preprocessing — Clean data (standardise data) , Normalize data(all lower case) , lemmatize data (all words to root words).
- Tokenize words with frequency
- Find TF for words
- Find IDF for words
- Vectorize vocab

example

- Let's cover an example of 3 documents -
- **Document 1** It is **going to** rain today. **1/6**
- **Document 2** Today I am not **going** outside.
- **Document 3** I am **going to** watch the season premiere. **1/8**

To find TF-IDF we need to perform the steps we laid out above, let's get to it.

- Step 1 Clean data and Tokenize

Word	Count
going	3
to	2
today	2
i	2
am	2
.	1
is	1
rain	1

Example, continue

- Step 2 Find TF for all docs

TF = (Number of repetitions of word in a document) / (# of words in a document)

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

Example, continue

- Step 3: Find IDF:

$\text{IDF} = \log[(\text{Number of documents}) / (\text{Number of documents containing the word})]$

In Excel use **LN(3/3)**

Words	IDF Value
going	$\log(3/3)$
to	$\log(3/2)$
today	$\log(3/2)$
i	$\log(3/2)$
am	$\log(3/2)$
It	$\log(3/1)$
is	$\log(3/1)$
rain	$\log(3/1)$

Example, continue

- Step 4: Build model i.e. stack all words next to each other

Words	IDF Value	Words/ Documents	Document 1	Document 2	Document 3
going	0	going	0.16	0.16	0.12
to	0.41	to	0.16	0	0.12
today	0.41	today	0.16	0.16	0
i	0.41	i	0	0.16	0.12
am	0.41	am	0	0.16	0.12
It	1.09	it	0.16	0	0
is	1.09	is	0.16	0	0
rain	1.09	rain	0.16	0	0

IDF Value and TF value of 3 documents.

Example, continue

- Step 5: Compare results and use table to ask questions

e.g., **0.41*0.12**

Words/ Documents	going	to	today	i	am	it	is	rain
Document 1	0	0.07	0.07	0	0	0.17	0.17	0.17
Document 2	0	0	0.07	0.07	0.07	0	0	0
Document 3	0	0.05	0	0.05	0.05	0	0	0

Remember, the final equation = TF-IDF = TF * IDF

Example, continue- Analysis and outcomes

- You can easily see using this table that words like ‘it’,‘is’,‘rain’ are important for document 1 but not for document 2 and document 3 which means Document 1 and 2&3 are different w.r.t talking about rain.
- You can also say that Document 1 and 2 talk about something ‘today’, and document 2 and 3 discuss something about the writer because of the word ‘I’.
- This table helps you **find similarities** and **non similarities** between documents, words and more much better than Bag Of Words.

Other Relevance Metrics



5. Marketing calls up and reads selected reviews in full, for greater insight.

- "Authoritativeness" of source
 - ▶ PageRank is an example of this
- Recency of document
- How often the document has been retrieved by other users

Effectiveness of Search and Retrieval



- Relevance metric
 - ▶ important for precision, user experience
- **Effective crawl, extraction, indexing**
 - ▶ important for recall (and precision)
 - ▶ more important, often, than retrieval algorithm
- MapReduce
 - ▶ Reverse index, corpus term frequencies, idf

Natural Language Processing

- Unstructured text mining means extracting “features”
 - ▶ Features are structured meta-data representing the document
 - ▶ Goal: “vectorize” the documents
- After vectorization, apply advanced machine learning techniques
 - ▶ **Clustering**
 - ▶ **Classification**
 - ▶▶ Decision Trees
 - ▶▶ Naïve Bayesian Classifier
 - ▶ **Scoring**
 - ▶▶ Once models have been built, use them to automatically categorize incoming documents

Example: UFOs Attack



July 15th, 2010. Raytown, Missouri

When I first noticed it, I wanted to freak out. There it was an object floating in on a direct path, It didn't move side to side or volley up and down. It moved as if though it had a mission or purpose. I was nervous, and scared, So afraid in fact that I could feel my knees buckling. I guess because I didn't know what to expect and I wanted to act non aggressive. I thought that I was either going to be taken, blasted into nothing, or...

Q: What is the witness describing?

A: An encounter with a UFO.

Q: What is the emotional state of the witness?

A: Frightened, ready to flee.

Source: <http://www.infochimps.com/datasets/60000-documented-ufo-sightings-with-text-descriptions-and-metadata>

Example: UFOs Attack

If we really are on the cusp of a major alien invasion, eyewitness testimony is the key to our survival as a species.



Strangely, the computer finds this account unreliable!

*When I **fist** noticed it, I wanted to freak out. It was a grey, saucer-like object floating in on a direct path. It **didn't** move side to side or volley up and down. It moved as if though it had a mission or purpose. I was nervous, and scared, **So afraid in fact** that I could feel my knees **buckling**. I guess because I **didn't** know what it was, and I wanted to **run** away with that I was either going to be **taken**, blasted into nothing, or...*

Machine error
object

Typo

Turn of phrase

Ambiguous meaning

“UFO” keyword missing

Source: <http://www.infochimps.com/datasets/60000-documented-ufo-sightings-with-text-descriptions-and-metadata>

Example: UFOs Attack



Investigators need to...

Search

for keywords and phrases, but your topic may be very complicated or keywords may be misspelled within the document

Manage

*document meta-data like **time, location and author**. Later retrieval may be key to identifying this meta-data early, and the document may be amenable to structure.*

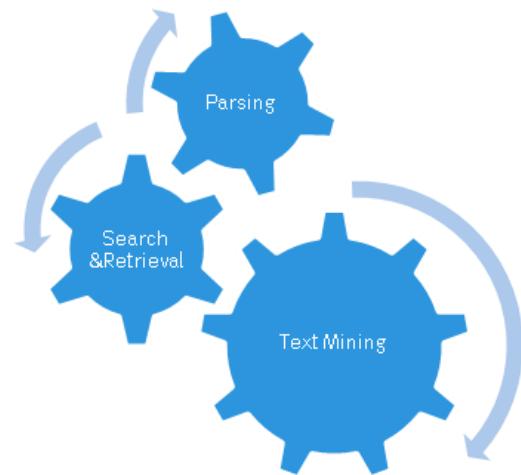
Understand

*content via **sentiment analysis, custom dictionaries, natural language processing, clustering, classification and good ol' domain expertise.***

...with computer-aided text mining

Challenges - Text Analysis

1. Finding the right structure for your unstructured data
2. Very high dimensionality
3. Thinking about your problem the right way





Introduction



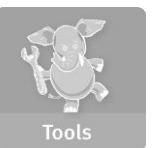
Analytics Lifecycle



Basic Methods



Adv. Methods



Tools



Lab

Module 4: Advanced Analytics – Theory and Methods

Lesson 8: Text Analysis - Summary

During this lesson the following topics were covered:

- Challenges with text analysis
- Key tasks in text analysis
- Definition of terms used in text analysis
 - Term frequency, inverse document frequency
- Representation and features of documents and corpus
- Use of regular expressions in parsing text
- Metrics used to measure the quality of search results
 - Relevance with tf-idf, precision and recall



Introduction



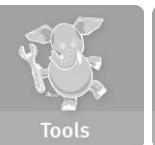
Analytics Lifecycle



Basic Methods



Adv. Methods



Tools



Lab

Module 4: Summary

Key Topics Covered in this module	Methods Covered in this module
Algorithms and technical foundations	Categorization (unsupervised) : K-means clustering Association Rules
Key Use cases	Regression Linear Logistic
Diagnostics and validation of the model	Classification (supervised) Naïve Bayesian classifier Decision Trees
Reasons to Choose (+) and Cautions (-) of the model	Time Series Analysis
Fitting, scoring and validating model in R and in-db functions	Text Analysis

Practice 1: Text Analysis : Upload file and Load Libraries

- # Load data

```
setwd("C:\\\\Users\\\\z10095\\\\Desktop\\\\")  
myFile <- 'shz.txt' # the local file path to my research prospectus
```

```
# fill = TRUE b/c rows are of unequal length  
#fill: Sometimes, we may get a file that contains the  
# unequal length of rows, and we have to add blank spaces to that missing values.  
dat <- read.table(myFile, header = FALSE, fill = TRUE)  
dat
```

Load required libraries

```
library(dplyr) # for data wrangling  
library(tidytext) # for Natural Language Processing  
library(stringr) # to deal with strings  
library(wordcloud) # to for generating word clouds  
library(knitr) # for tables, It combines many features into one package  
library(DT) # for dynamic tables  
library(tidyr)  
#Tools to help to create tidy data, where each column is a variable,  
#each row is an observation, and each cell contains a single value.
```

Format the text file reshape the .txt data frame into one column

- tidy_dat <- tidyr::gather(dat, key, word) %>% select(word)
- # 1:8 %>% sum %>% sqrt. **# Answer is 6**
- tidy_dat

```
2 Understanding
3 Born
4 Life,
5 Through
6 The
7 In
8 He
9 In
10 In
11 Eventually
12 The
13 while
14 One
```

- `1:8 %>% sum %>% sqrt`
- `## or:`
- `x=1:8`
- `x`
- `y=sum(x)`
- `Y #36`
- `sqrt(y)`
- Both gives you same answer: 6

- The gather() Function
- The second tidyr function we will look into is the gather() function. With gather() it may not be clear what exactly is going on, but in this case we actually have a lot of column names the represent what we would like to have as data values.
- data is the dataframe you are working with.
- key is the name of the key column to create.
- value is the name of the value column to create.

Count # of words

```
tidy_dat$word %>% length() #there are 2832 tokens in my document
```

```
[1] 2832
```

Count # of unique words

```
unique(tidy_dat$word) %>% length() # 695 words are unique
```

```
[1] 695
```

- **unnest_tokens**: Split a column into tokens
- **semi_join(x, y)**: Return all rows from x where there are matching values in y, keeping just columns from x. A semi join differs from an inner join because an **inner join** will return one row of x for each matching row of y, where a **semi join** will never duplicate rows of x. This is a filtering join.

Semi join and inner join

anti_join came in handy for us in a setting where we were trying to re-create an old table from the source data. We then wanted to be able to **identify the records from the original table that did not exist in our updated table**. Good example

http://zevross.com/blog/2014/08/05/using-the-r-function-anti_join-to-find-unmatched-records/

Tokenize the text and Frequency

```
tokens <- tidy_dat %>%
```

```
unnest_tokens(word, word) %>%
```

```
dplyr::count(word, sort = TRUE) %>%
```

```
ungroup()
```

```
tokens %>% head(10)
```

```
➤tokens %>% head(10)
➤# A tibble: 10 x 2
➤word n <chr> <int>
➤1 the    137
➤2 of     73
➤3 and    62
➤4 in     45
➤5 zayed   45
➤6 to     40
➤7 sheikh  39
➤8 a      32
➤9 his    25
➤10 was   22
```

Draw the Cloud

`tokens %>%`

```
  with(wordcloud(word, n, random.order = FALSE, max.words = 50,  
  colors=pal))
```

`#`

`# other choices`

```
#colors=brewer.pal(8, "Dark2")
```

```
tokens %>% with(wordcloud(word, n,  
  random.order = FALSE, max.words = 50,  
  colors=brewer.pal(8, "Dark2")))
```

```
tokens %>% with(wordcloud(word, n,  
  random.order = FALSE, max.words = 50,  
  colors="#AD1DA5"))
```

random.order

plot words in random order. If false, they will be plotted in decreasing frequency



Remove Stop Words

remove stop words

```
data("stop_words")  
tokens_clean <- tokens %>%  
  anti_join(stop_words, by = "word")
```

**identify the records from the original table
that did not exist in our updated table**

Remove numbers

remove numbers

```
nums <- tokens_clean %>% filter(str_detect(word, "^[0-9]")) %>%  
select(word) %>% unique()
```

```
tokens_clean <- tokens_clean %>%  
anti_join(nums, by = "word")
```

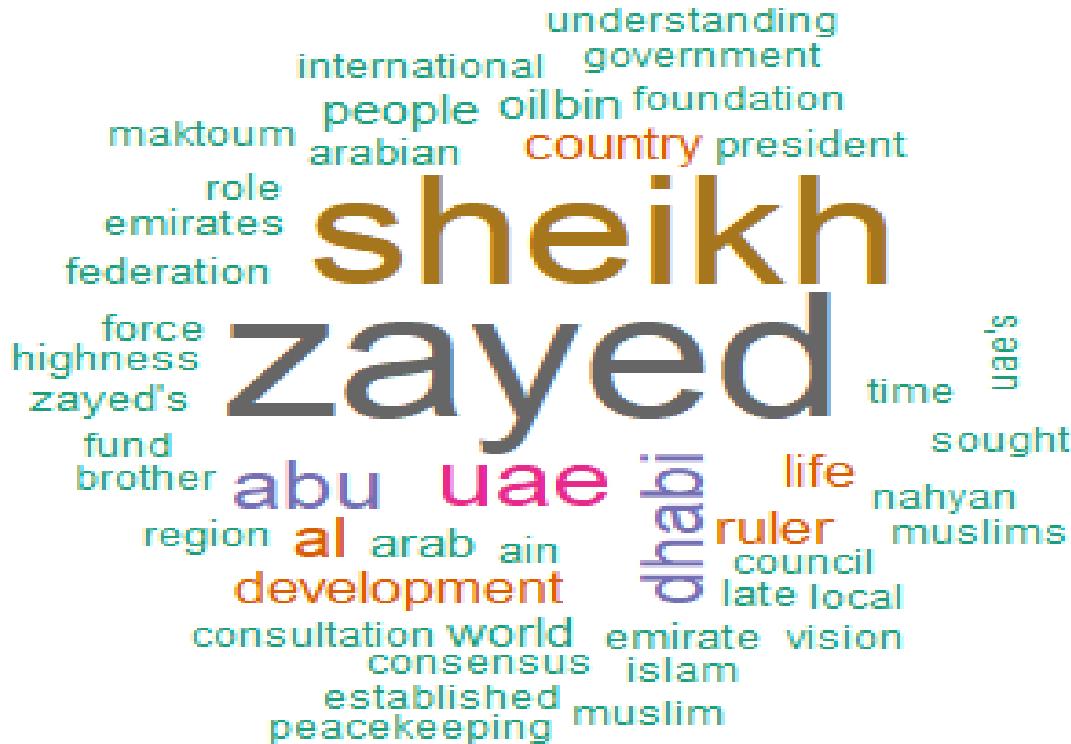
Remove other stop words

```
# remove other stop words  
uni_sw <- data.frame(word = c("i.e", "e.g."))  
  
tokens_clean <- tokens_clean %>%  
anti_join(uni_sw, by = "word")
```

Draw the Cloud again

```
# define a nice color palette  
pal <- brewer.pal(8,"Dark2")  
  
# plot the 50 most common words  
tokens_clean %>%  
  with(wordcloud(word, n, random.order = FALSE, max.words = 50,  
  colors=pal))
```

Word Cloud after cleaning



Clean data and search

```
tokens_clean %>%
```

```
  DT::datatable()
```

```
tokens_clean
```

Show 10 entries

Search:

word

n

1	zayed	45
2	sheikh	39
3	uae	17
4	abu	14
5	dhabi	12
6	al	10
7	development	8
8	ruler	7
9	country	6
10	life	6

Showing 1 to 10 of 491 entries

Previous

1

2

3

4

5

...

50

Next

Practice : Sentiment Analysis

```
# Load the library
```

```
library(RSentiment)
```

```
calculate_total_presence_sentiment(c("This is a good text", "This is a bad text",
"This is a really bad text", "This is horrible"))
```

```
> #output
```

```
> [1] "Processing sentence: this is a good text"
```

```
> [1] "Processing sentence: this is a bad text"
```

```
> [1] "Processing sentence: this is a really bad text"
```

```
> [1] "Processing sentence: this is horrible"
```

```
> [,1] [,2] [,3] [,4] [,5] [,6]
```

```
> [1,] "Sarcasm" "Negative" "Very Negative" "Neutral" "Positive" "very Positive"
```

```
> [2,] "0"      "3"        "0"        "0"      "1"      "0"
```

calculate_sentiment

```
calculate_sentiment(c("This is a good text",
                      "This is a bad text",
                      "This is a really bad text", "This is horrible"))

#output
[1] "Processing sentence: this is a good text"
[1] "Processing sentence: this is a bad text"
[1] "Processing sentence: this is a really bad text"
[1] "Processing sentence: this is horrible"

      text sentiment

1 This is a good text Positive
2 This is a bad text Negative
3 This is a really bad text Negative
4 This is horrible Negative
```

Calculating Sentiment Score

```
calculate_score(c("This is a good text",
                 "This is a bad text",
                 "This is a really bad text",
                 "This is horrible"))
```

```
[1] "Processing sentence: this is a good text"
[1] "Processing sentence: this is a bad text"
[1] "Processing sentence: this is a really bad text"
[1] "Processing sentence: this is horrible"
```

```
[1] 1 -1 -1 -1
```

Text Mining and NLP

More Examples

String Matching in R Programming

- String matching is an important aspect of any language. It is useful in finding, replacing as well as removing string(s)
 - A regular expression is a string that contains special symbols and characters to find and extract the information needed from the given data.
 - Operations on String Matching
 - ▶ Finding a String
 - ▶ **grep()** function: It returns the index at which the pattern is found in the vector.
 - ▶ *grep(pattern, string, ignore.case=FALSE)*
 - ▶ `str <- c("Man", "woman", "baby", "amman", "happy")`
 - ▶ `grep('man', str)`
- ```
> grep('man', str) [1] 2 4
```

# String Matching in R Programming

- str <- c("Man", "woman", "baby", "amman", "happy")
- grep('man', str, ignore.case = "True")
  - grep('man', str, ignore.case = "True")
  - [1] 1 2 4

**grepl() function:** It is a logical function that returns the value **True** if the specified pattern is found in the vector and **false** if it is not found.

## Syntax

*grepl(pattern, string, ignore.case=FALSE)*

To find whether any instance(s) of 'the' are present in the string.

```
str <- c("Man", "woman", "baby", "amman", "happy")
grepl('the', str)
[1] FALSE FALSE FALSE FALSE FALSE
➤ grepl('wo', str)
➤ [1] FALSE TRUE FALSE FALSE FALSE
```

# String Matching in R Programming

- **regexpr()** function: It searches for occurrences of a pattern in every element of the string.
- **Syntax:**  
***regexpr(pattern, string, ignore.case = FALSE)***

**example:** To find whether any instance(s) of 'he' is present in each string of the vector.

```
str <- c("Hello", "hello", "hi", "ahey", "aahead")
regexpr('he', str)
```

```
➤ regexpr('he', str)
➤ [1] -1 1 -1 2 3
```

**example:** To find whether any instance(s) of words starting with a vowel is present in each string of the vector.

```
str <- c("abra", "Ubria", "hunt", "quirky")
regexpr('^[aeiouAEIOU]', str)
```

```
➤ regexpr('^[aeiouAEIOU]', str)
➤ [1] 1 1 -1 -1
```

# Finding and Replacing Strings in R: **sub()** and **gsub()**.

In order to **search and replace** a particular string, we can use two functions namely, **sub()** and **gsub()**.

**sub** replaces the only first occurrence of the string to be replaced and returns the modified string.

**gsub()** replaces all occurrences of the string to be replaced and returns the modified string.

## Syntax:

*sub(pattern, replaced\_string, string)*  
*gsub(pattern, replaced\_string, string)*

**Example :** To replace the first occurrence of 'he' with 'aa'

```
str = "heutabhe"
sub('he', 'aa', str)
➤ sub('he', 'aa', str)
➤ [1] "aautabhe"
```

**Example :** To replace all occurrences of 'he' with 'aa'

```
str = "heutabhe"
gsub('he', 'aa', str)
➤ gsub('he', 'aa', str)
➤ [1] "aautabaa"
```

# Finding and Removing Strings in R: `str_remove()` and `str_remove_all()`.

`str_remove()` removes the only first occurrence of the string/pattern to be removed and returns the modified string.

`str_remove_all()` removes all occurrences of the string to be removed and returns the modified string.

## Syntax:

```
str_remove(string, pattern,
 ignore.case=False)
```

**Example** : Removing the first occurrence of **vowels** in the vector

```
library(stringr)
x <- c("apple", "pear", "banana", "orange")
str_remove(x, "[aeiou]")

➤ str_remove(x, "[aeiou]")
➤ [1] "pple" "par" "bnana" "range"
```

**Example** : Removing all occurrences of **vowels** in the vector

```
library(stringr)
x <- c("apple", "pear", "banana", "orange")
str_remove_all(x, "[aeiou]")

➤ str_remove_all(x, "[aeiou]")
➤ [1] "ppl" "pr" "bnn" "rng"
```

# More examples, text mining applications on novels

- **Sense and Sensibility** is a novel by [Jane Austen](#), published in 1811. It was published anonymously; *By A Lady* appears on the title page where the author's name might have been. It tells the story of the Dashwood sisters, Elinor (age 19) and Marianne (age 16½) as they come of age. They have an older half-brother, John, and a younger sister, Margaret (age 13).
- **Pride and Prejudice** is an 1813 romantic [novel of manners](#) written by [Jane Austen](#). The novel follows the character development of [Elizabeth Bennet](#), the dynamic [protagonist](#) of the book who learns about the repercussions of hasty judgments and comes to appreciate the difference between superficial goodness and actual goodness. Its humour lies in its honest depiction of manners, education, marriage, and money during the [Regency era](#) in [Great Britain](#).
- **Mansfield Park** is the third published novel by [Jane Austen](#), first published in 1814 by [Thomas Egerton](#). A second edition was published in 1816 by [John Murray](#), still within Austen's lifetime. The novel did not receive any public reviews until 1821.

<https://en.wikipedia.org/>

# More examples, text mining applications on novels

- *Emma*, by [Jane Austen](#), is a novel about youthful [hubris](#) and romantic misunderstandings. It is set in the fictional country village of Highbury and the surrounding estates of Hartfield, Randalls and Donwell Abbey, and involves the relationships among people from a small number of families.[\[2\]](#) The novel was first published in December 1815
- ***Northanger Abbey***: This article is about the 1817 novel. For adaptations of the novel, see [Jane Austen in popular culture § Northanger Abbey \(1817\)](#).
- ***Persuasion*** is the last novel fully completed by [Jane Austen](#). It was published at the end of 1817, six months after her death.

The story concerns Anne Elliot, a young Englishwoman of twenty-seven years, whose family moves to lower their expenses and reduce their debt by renting their home to an [Admiral](#) and his wife. The wife's brother, Navy Captain Frederick Wentworth, was engaged to Anne in 1806, but the engagement was broken when Anne was "persuaded" by her friends and family to end their relationship. Anne and Captain Wentworth, both single and unattached, meet again after a seven-year separation, setting the scene for many humorous encounters as well as a second, well-considered chance at love and marriage for Anne in her second "bloom".

<https://en.wikipedia.org/>

# Libraries used in R for text analytics

library(tidytext)

library(tidyverse)

**library(janeaustenr)**

library(stringr)

library(wordcloud)

library(reshape2)

library(textdata)

# Dictionaries used for sentiments

```
get_sentiments("afinn")
```

```
get_sentiments("bing")
```

```
get_sentiments("nrc")
```

```
data(sentiments)
```

```
#dataset structure
```

```
str(sentiments)
```

```
> get_sentiments("afinn")
> # A tibble: 2,477 x 2 word value <chr> <dbl>
> 1 abandon -2
> 2 abandoned -2
> 3 abandons -2
> 4 abducted -2
> 5 abduction -2
> 6 abductions -2
> 7 abhor -3
> 8 abhorred -3
> 9 abhorrent -3
> 10 abhors -3 # ... with 2,467 more rows
```

```
> get_sentiments("bing")
> # A tibble: 6,786 x 2 word sentiment <chr> <chr>
> 1 2-faces negative
> 2 abnormal negative
> 3 abolish negative
> 4 abominable negative
> 5 abominably negative
> 6 abominate negative
> 7 abomination negative
> 8 abort negative
> 9 aborted negative
> 10 aborts negative
```

```
> get_sentiments("nrc")
> # A tibble: 13,901 x 2 word sentiment <chr> <chr>
> 1 abacus trust
> 2 abandon fear
> 3 abandon negative
> 4 abandon sadness
> 5 abandoned anger
> 6 abandoned fear
> 7 abandoned negative
> 8 abandoned sadness
> 9 abandonment anger
> 10 abandonment fear # ... with 13,891 more rows
```

# Sample output for afinn lexicon

```
afinn_lexicon <- get_sentiments("afinn")
head(afinn_lexicon)
A tibble: 6 × 2
```

## Output

```
word score
1 abandon -2
2 abandoned -2
```

# Sample output for nrc lexicon

```
#NRC
```

```
nrc_lexicon <- get_sentiments("nrc")
head(nrc_lexicon)
```

## Output:

```
A tibble: 6 × 2
word sentiment
<chr> <chr>
1 abacus trust
2 abandon fear
```

# Sample output for bing lexicon

#BING

```
bing_lexicon <- get_sentiments("bing")
head(bing_lexicon)
```

## Output

```
A tibble: 6 × 2
word sentiment
<chr> <chr>
1 2-faced negative
2 2-faces negative
3 a+ positive
4 abnormal negative
5 abolish negative
```

## #Get the emma book and transform it into a tidy dataset

```
tidy_books <- austen_books() %>%
 filter(book == "Emma") %>%
 group_by(book) %>%
```

## Using **row\_number()** with **mutate()** will create a column of consecutive **numbers**. The **row\_number()** function is useful for creating an identification **number** (an ID variable). It is also useful for labeling each observation by a grouping variable.

##**cumsum()** function in R Language is used to calculate the **cumulative sum** of the vector passed as ## argument

# str\_detect function returns a logical **value** (i.e. FALSE or TRUE),

```
 mutate(linenumber = row_number(),
 chapter = cumsum(str_detect(text, regex("^chapter [\\divxlc]",
 ignore_case = TRUE)))) %>% ungroup() %>%
 unnest_tokens(word, text)
```

<https://www.tidytextmining.com/tidytext.html>

# nrc lexicon associated with joy

#Using the nrc lexicon, only the words that are associated to a sentiment of `joy`

```
nrc_joy <- get_sentiments("nrc") %>%
 filter(sentiment == "joy")
#Summarize the usage of `joy` words
tidy_books %>%
 semi_join(nrc_joy) %>%
 count(word, sort = T)
```

## Output

## # A tibble: 303 × 2

```
word n
<chr> <int>
1 good 359
2 young 192
3 friend 166
4 hope 143
```

# Semi join and inner join

- **semi\_join(x, y)**: Return all rows from x where there are matching values in y, keeping just columns from x. A semi join differs from an inner join because an **inner join** will return one row of x for each matching row of y, where a semi join will never duplicate rows of x. This is a filtering join.

**anti\_join** came in handy for us in a setting where we were trying to re-create an old table from the source data. We then wanted to be able to identify the records from the original table that did not exist in our updated table. Good example

[http://zevross.com/blog/2014/08/05/using-the-r-function-anti\\_join-to-find-unmatched-records/](http://zevross.com/blog/2014/08/05/using-the-r-function-anti_join-to-find-unmatched-records/)

# Application in r

Basically, the **mutate** function in R programming is used to create new variables. It is used to generate new variables from data sets.

```
tidy_books <- austen_books() %>%
 group_by(book) %>%
 mutate(linenumber = row_number(),
 chapter = cumsum(str_detect(text, regex("^chapter [\\d\\D]+")),
 ignore_case = TRUE)))) %>%
ungroup() %>%
unnest_tokens(word, text)
```

**convert the text to the tidy format using unnest\_tokens()**

**<https://www.tidytextmining.com/sentiment.html>**

# A side example, understand mutate

- `#install.packages("dplyr")`
- `library(dplyr)`
- `mtcars`
- `mtcares2 = mutate(mtcars, mtcars_new = mpg/cyl)`

The diagram shows the `mutate` function call with three annotations:

- An arrow points to the first argument `mtcars` with the label "Data Frame you are modifying".
- An arrow points to the second argument `new_var = mpg/cyl` with the label "Name of the new variable".
- An arrow points to the third argument `mpg/cyl` with the label "The modifying action you are taking (e.g., multiply by 10)".

```
mutate(data_frame, new_var = [existing_var])
```

# Understand mutate and group by

```
Creating identification number to represent 50 individual people
ID <- c(1:20)

Creating sex variable (10 males/10 females)
Sex <- rep(c("male", "female"), 10) # rep stands for replicate

Creating age variable (20-39 year olds)
Age <- c(26, 25, 39, 37, 31, 34, 34, 30, 26, 33,
 39, 28, 26, 29, 33, 22, 35, 23, 26, 36)

Creating a dependent variable called Score
Score <- c(0.010, 0.418, 0.014, 0.090, 0.061, 0.328, 0.656, 0.002, 0.639, 0.173,
 0.076, 0.152, 0.467, 0.186, 0.520, 0.493, 0.388, 0.501, 0.800, 0.482)
```

```
Creating a unified dataset that puts together all variables
tibble is a simple dataframe
data <- tibble(ID, Sex, Age, Score)
group by sex
data %>%
 group_by(Sex) %>%
 summarize(m = mean(Score), # calculates the mean
 s = sd(Score), # calculates the standard deviation
 n = n()) %>% # calculates the total number of observations
 ungroup()
##`summarise()` ungrouping output (override with `.`groups` argument)
A tibble: 2 x 4
##x m s n
##<dbl> <dbl> <dbl> <int>
##1 female 0.282 0.184 10
##2 male 0.363 0.300 10
```

```
mutate() and group_by()
data %>%
 group_by(Sex) %>%
 mutate(m = mean(Score)) %>% # calculates mean score by Sex
 ungroup()
```

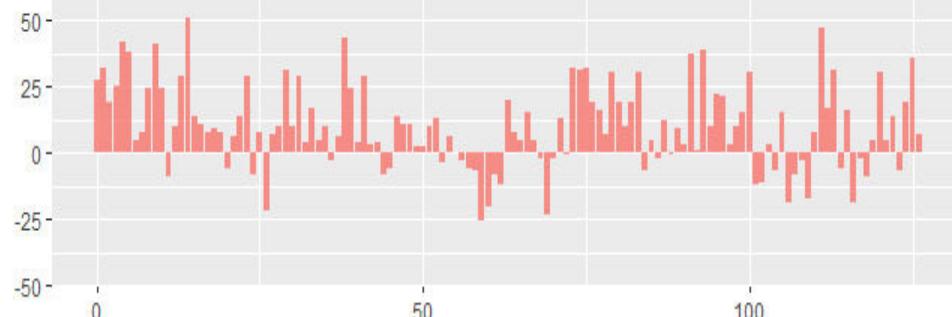
```
janeaustensentiment <- tidy_books %>%
 inner_join(get_sentiments("bing")) %>%
 count(book, index = linenumber %/%
 100, sentiment) %>%
 spread(sentiment, n, fill = 0) %>%
 mutate(sentiment = positive - negative)
```

```
head(janeaustensentiment)
Source: local data frame [6 x 5]
Groups: book, index [6]
##
book index negative positive sentiment
<fctr> <dbl> <dbl> <dbl> <dbl>
1 Sense & Sensibility 0 20 47 27
2 Sense & Sensibility 1 22 54 32
3 Sense & Sensibility 2 16 35 19
4 Sense & Sensibility 3 20 45 25
5 Sense & Sensibility 4 21 63 42
```

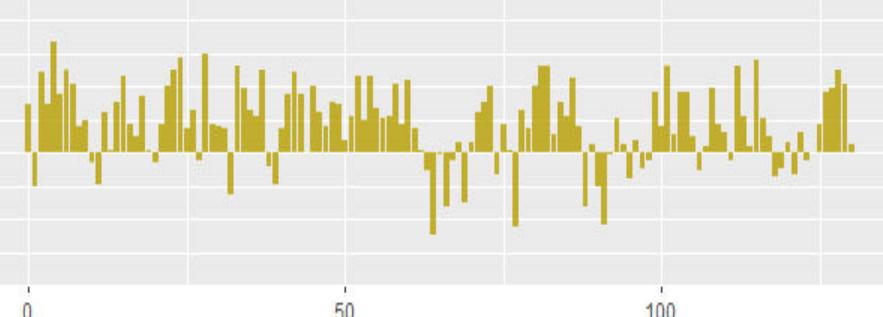
# Plot Sentiment

```
ggplot(data = janeaustensentiment, mapping = aes(x = index, y =
sentiment, fill = book)) +
 geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
 facet_wrap(facets = ~ book, ncol = 2, scales = "free_x")
```

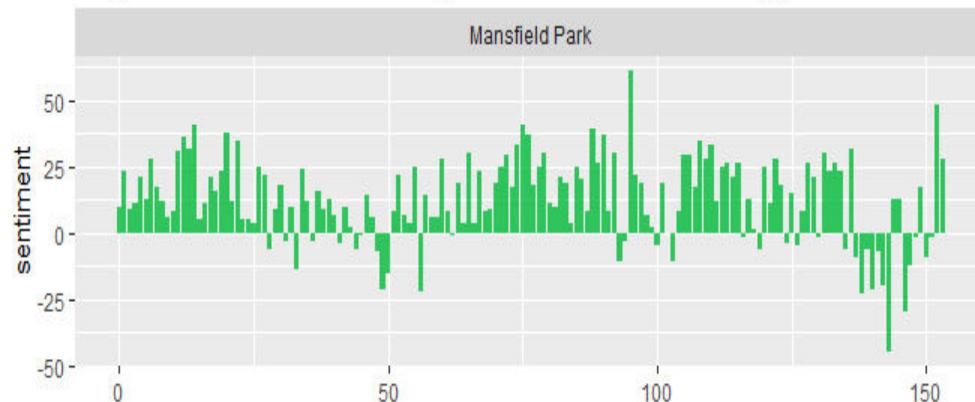
Sense & Sensibility



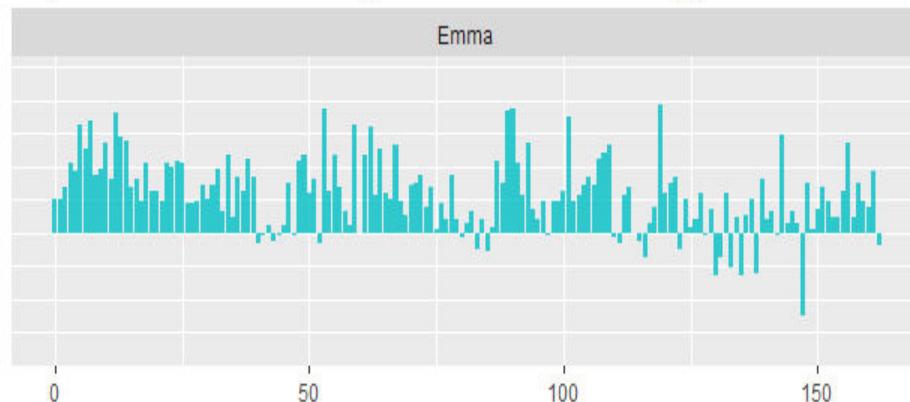
Pride & Prejudice



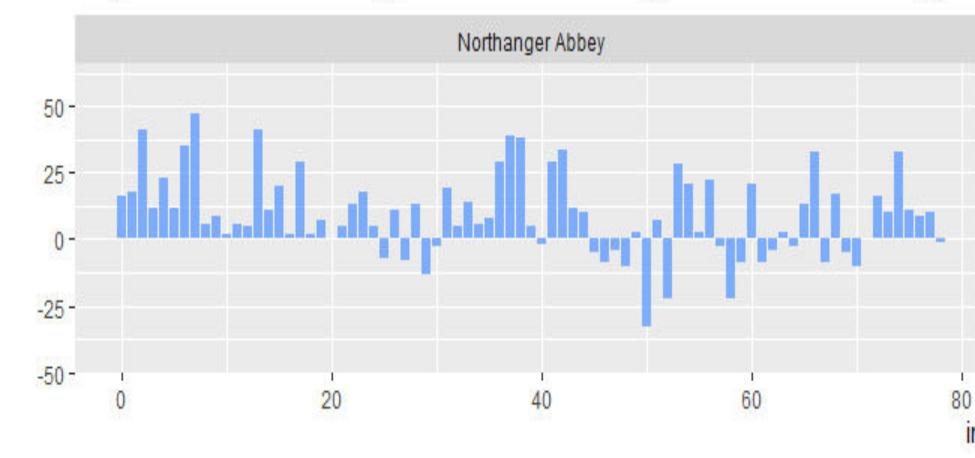
Mansfield Park



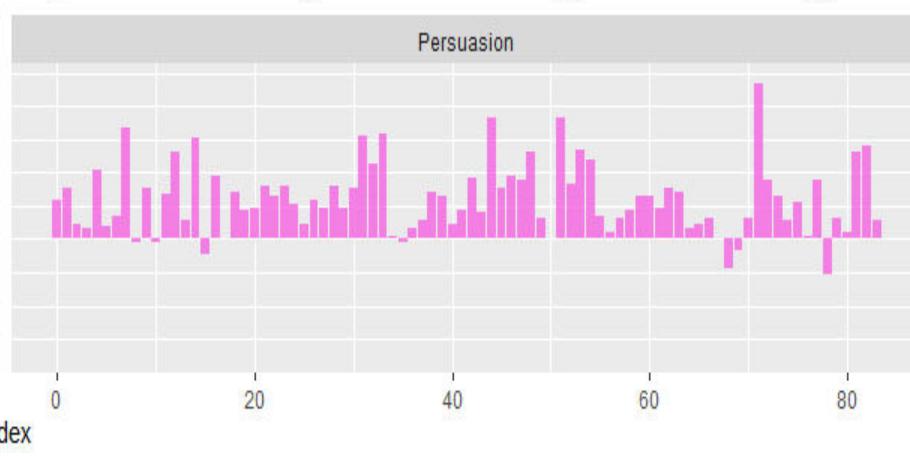
Emma



Northanger Abbey



Persuasion



# Extra References

<https://www.geeksforgeeks.org/string-matching-in-r-programming/>

<https://en.wikipedia.org/>

<https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3>

*Thank You!*

The image features the words "Thank You!" in a black, flowing cursive font. A vibrant, multi-colored brushstroke underline sweeps across the text, starting from the left and curving upwards towards the right. The colors of the brushstroke transition through a rainbow palette, including shades of blue, purple, pink, red, orange, and yellow. The background is a soft, light blue gradient.