# Announcements:

- Upload homework to the class website on moodle http://moodle.extn.washington.edu/course/view.php?id=6277
- Capstone project is mandatory
- Submit at least 4 of the 6 homeworks

  *If you are unable to attend class, please watch the recorded lecture;*
  *Homework can be up to one week late and I will still review/count it;*
  *Groups are welcome to work together on assignments;*
  *Effort is more important than accuracy;*

- Use class moodle site to ask questions, get assistance, share ideas…

# Data Science

## Deriving Knowledge from Data at Scale

Winson Taam

Oct 12th, 2015



# Lecture Outline

| | |
|---|---|
| • Class Discussions | 20 minutes |
| • Machine Learning Primer | 60 minutes |
| • Break | 15 minutes |
| • Forecasting, 1/2 | 60 minutes |
| • Closing discussion | 15 minutes |

*Laying a foundation for Prediction and Forecasting…*

# WEKA http://www.cs.waikato.ac.nz/ml/weka/, *install by next class…*


Machine Learning Group at the University of Waikato

Project **Software** Book Publications People Related

## Weka 3: Data Mining Software in Java

Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

Found only on the islands of New Zealand, the Weka is a flightless bird with an inquisitive nature. The name is pronounced like **this**, and the bird sounds like **this**.

Weka is open source software issued under the **GNU General Public License**.

Yes, it is possible to apply Weka to **big data**!

**Data Mining with Weka** is a 5 week MOOC, which was held first in late 2013. Check out the **MOOC site** for video lectures and details on how to enrol into this course and a new, advanced Weka course.

Getting started

- **Requirements**
- **Download**
- **Documentation**
- **FAQ**
- **Getting Help**

Further information

- **Citing Weka**
- **Datasets**
- **Related Projects**
- **Miscellaneous Code**
- **Other Literature**

Developers

- **Development**
- **History**
- **Subversion**
- **Contributors**

# Lecture Outline

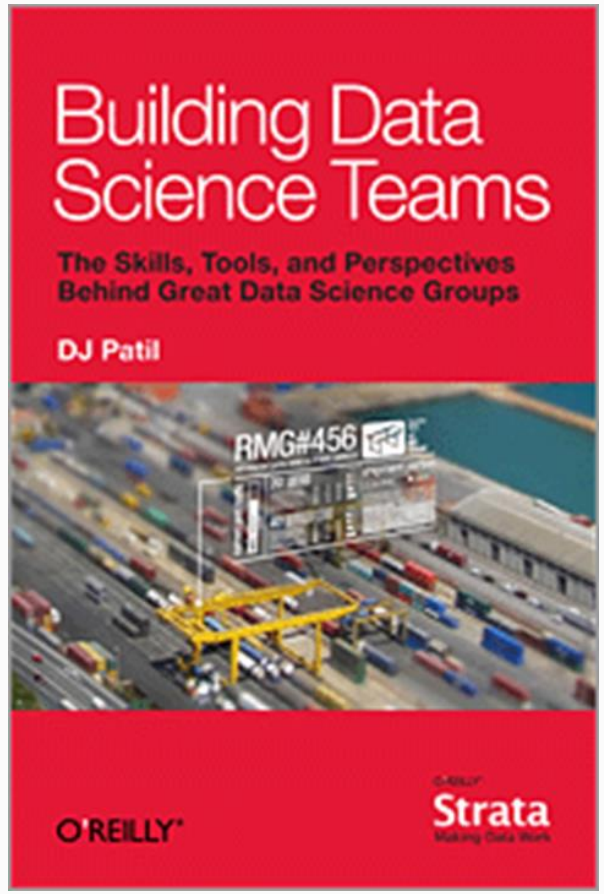
| | |
|---|---|
| • Class Discussions | 20 minutes |
| • Machine Learning Primer | 60 minutes |
| • Break | 15 minutes |
| • Forecasting, 1/2 | 60 minutes |
| • Closing discussion | 15 minutes |

# What was the objective of this?

Get you to think about data science teams

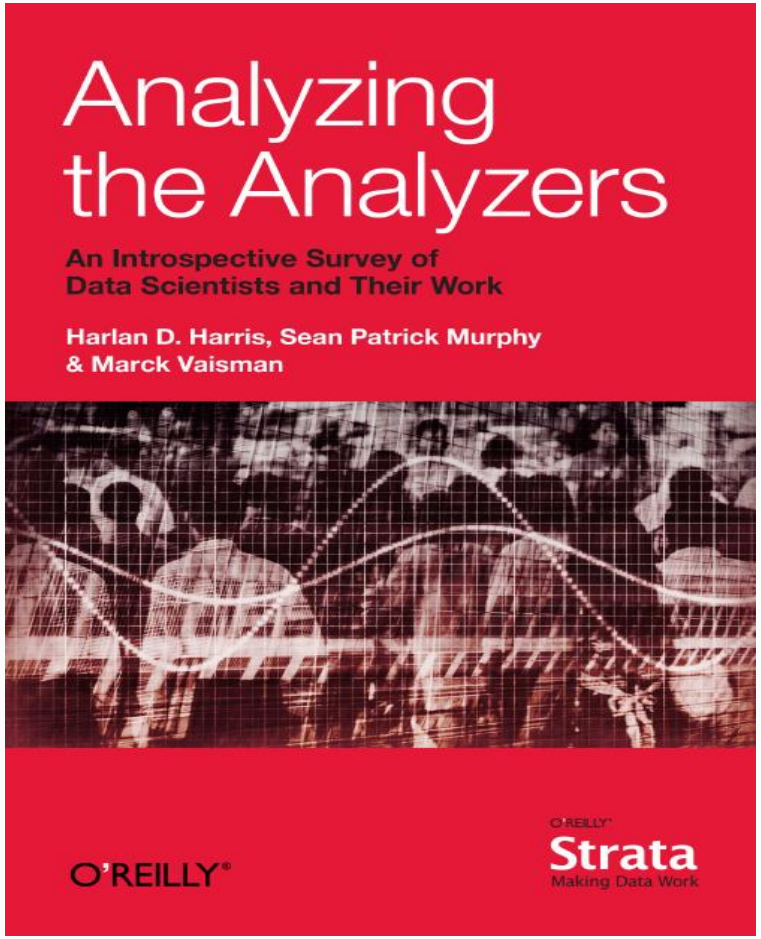
- No one person can be the perfect data scientist, *so we need teams*
- Who would you want to work with?

Your key takeaways from this article?



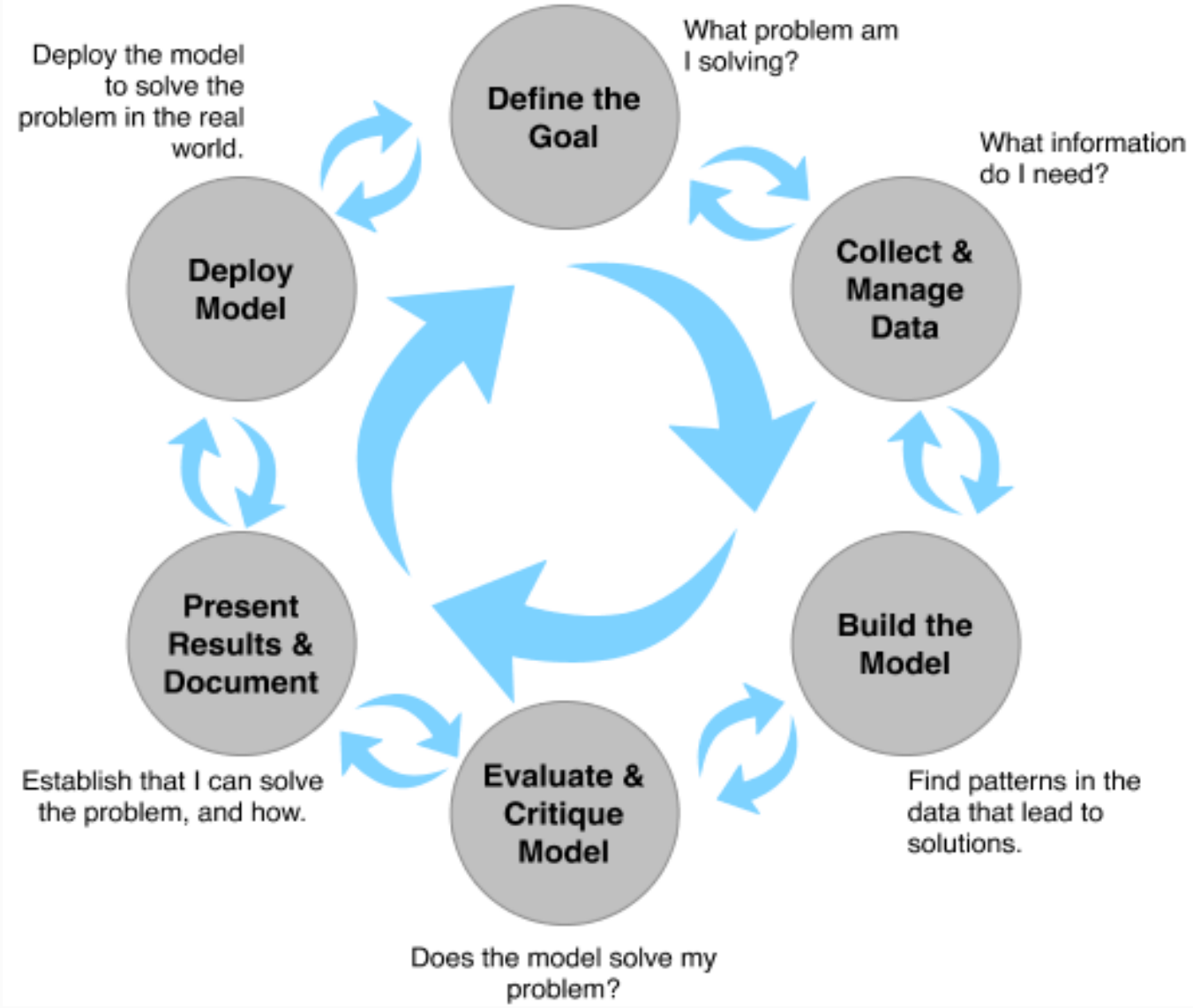
# Discussion on Assigned Reading…

## Week One




# The Data Science Workflow

*Loops within loops…*



# The Data Science Workflow
## Define the Goal

The first task in a data science project is to define a measurable & quantifiable goal. At this stage, *learn all that you can about the context of your project*.

- Why do the project sponsors want the project in the first place? What do they lack, and what do they need?
- What are they doing to solve the problem now, and why isn't that good enough?
- What resources will you need: what kind of data, how much staff, will you have domain experts to collaborate with, what are the computational resources?
- How do the project sponsors plan to deploy your results? What are the constraints that have to be met for successful deployment?

Not We want to get better at finding bad loans.
But We want to reduce our rate of loan charge-offs by at least 10%, using a model that predicts which loan applicants are likely to default.

*A concrete goal begets concrete **stopping conditions** and concrete **acceptance criteria**. The less specific the goal, the likelier that the project will go unbounded, because no result will be "good enough." If you don't know what you want to achieve, you don't know when to stop trying – or even what to try. When the project eventually terminates – because either time or resources run out – **no one will be happy with the outcome…***

# The Data Science Workflow
## Data Collection and Management

This step encompasses identifying the data you need, exploring it, and conditioning it to be suitable for analysis. This stage is often the most time-consuming step in the process. It's also one of the most important.

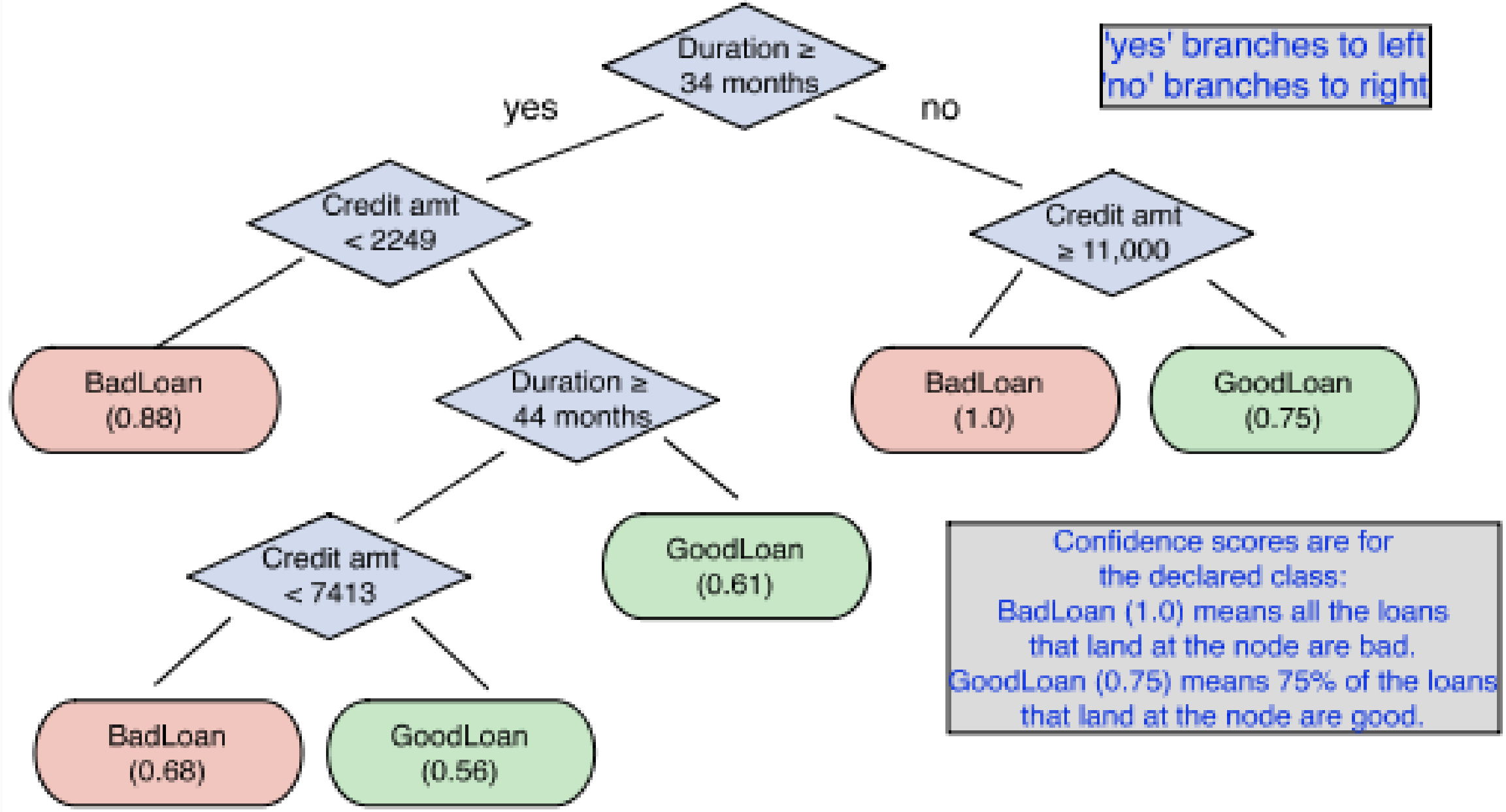
- What data is available to me?
- Will it help me solve the problem?
- Is it enough?
- Is it of good enough quality?

Rule First piece of data is *very informative*, data set utility is roughly logarithmic in size.
Prefer Direct measurements, but if they are not available identify proxy variables.

# The Data Science Workflow
## Modeling

We get to statistics and machine learning during the modeling stage. Here is where you try to extract useful insights from the data. Many modeling procedures make specific assumptions about data distribution and relationships, *there will be back-and-forth between the modeling and data cleaning stages as you try to find the best way to represent and model the data*. The most common data science modeling tasks are:

- Classification: deciding if something belongs to one category or another.
- Scoring: predicting or estimating a numeric value such as a price or probability.
- Ranking: learning to order items by preferences.
- Clustering: grouping items into most similar groups.
- Finding Relations: finding correlations or potential causes of effects seen in the data.
- Root cause analysis: scientific planned experiment to uncover causal effects
- Characterization: very general plotting and report generation from data.


Duration ≥ 34 months
yes
no
'yes' branches to left
'no' branches to right
Credit amt < 2249
Credit amt ≥ 11,000
BadLoan (0.88)
Duration ≥ 44 months
BadLoan (1.0)
GoodLoan (0.75)
Credit amt < 7413
GoodLoan (0.61)
BadLoan (0.68)
GoodLoan (0.56)
Confidence scores are for the declared class:
BadLoan (1.0) means all the loans that land at the node are bad.
GoodLoan (0.75) means 75% of the loans that land at the node are good.

# The Data Science Workflow
## Model Evaluation and Critique

Once you have a model, you need to determine if it meets your goals.

- Is it accurate enough for your needs?
- Does it generalize well?
- Does it perform better than "the obvious guess"? Better than whatever estimate you currently use?
- Do the results of the model (coefficients, clusters, rules) make sense in the context of the problem domain?

If you've answered "no" to any of the above questions, it's time to loop back to the modeling step — or decide that the data doesn't support the goal you are trying to achieve.

# The Data Science Workflow
## Model Deployment and Maintenance

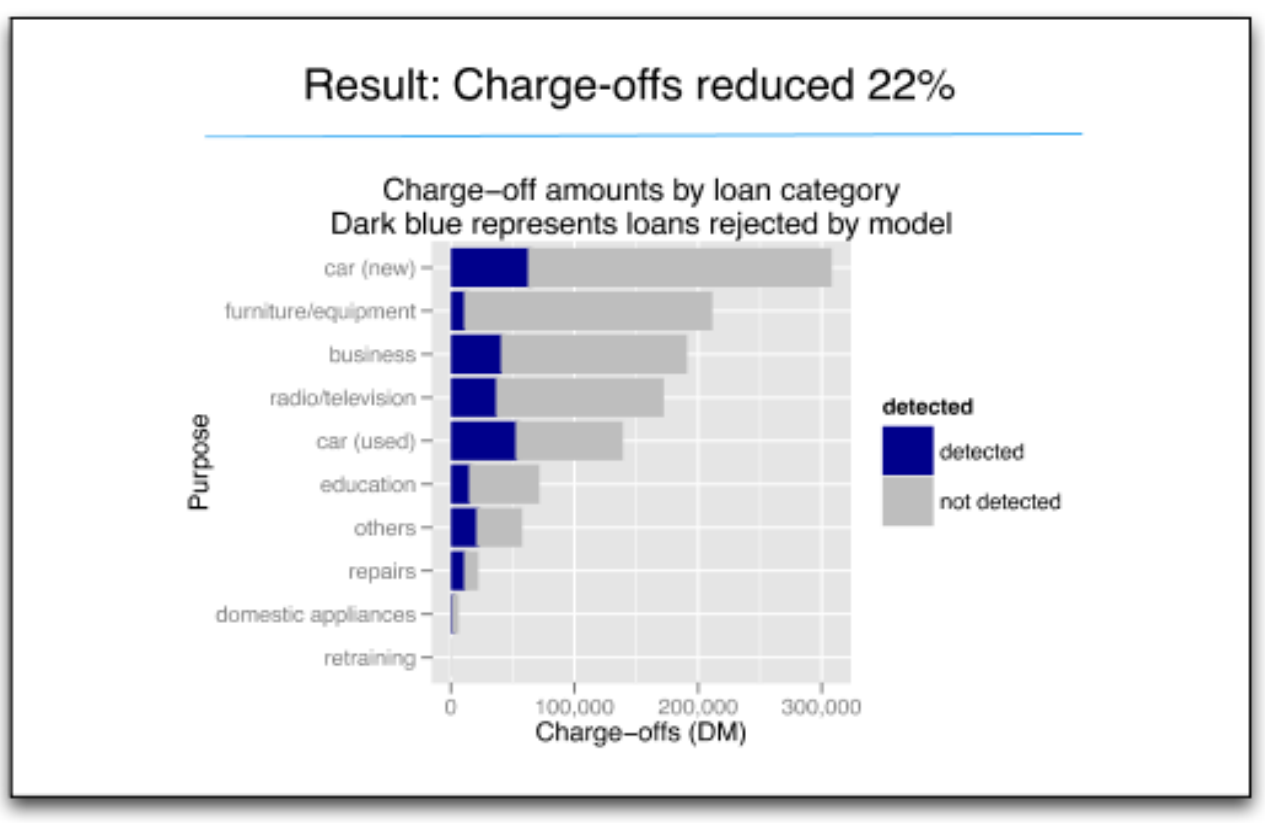
Finally, the model is put into operation.

In many organizations this means the data scientist no longer has primary responsibility for the day-to-day operation of the model. However, you still should *ensure that the model will run smoothly* and will not make disastrous unsupervised decisions. You also want to make sure that the *model can be updated* as its environment changes. And in many situations, the model will initially be *deployed in a small pilot program*.

The test might bring out issues that you didn't anticipate, and you will have to adjust the model appropriately.

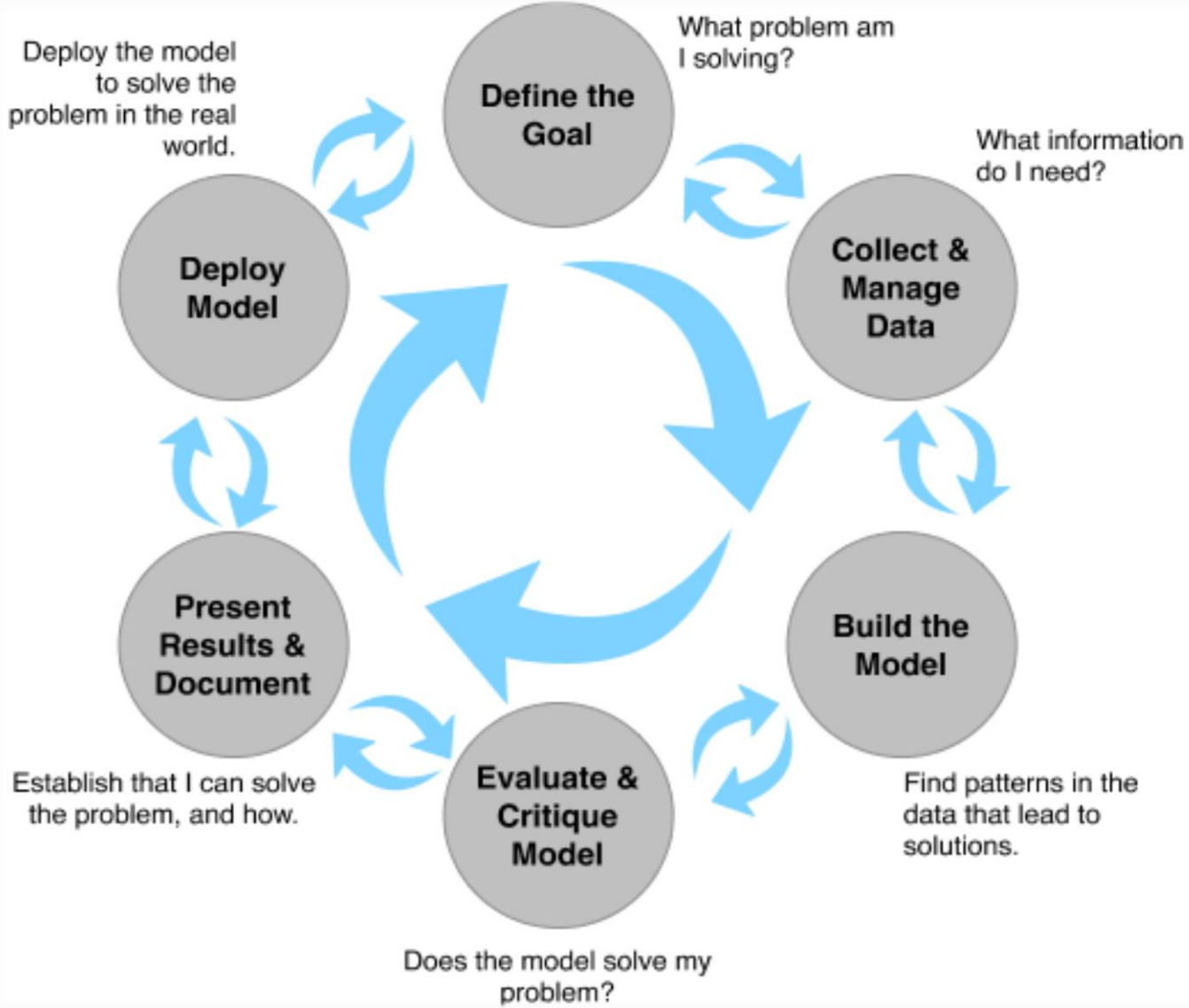
# The Data Science Workflow
## Presentation and Documentation

Once you have a model that meets your success criteria, you will present your results to your project sponsor and other stakeholders. You must also document the model for those in the organization who are responsible for using, running, and maintaining the model once it has been deployed. *Model interpretability may be an issue*…



# The Data Science Workflow

*Loops within loops…*



# Lecture Outline

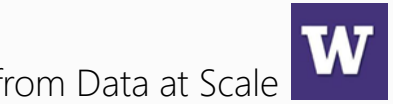
| | |
|---|---|
| • Class Discussions | 30 minutes |
| • Machine Learning Primer | 60 minutes |
| • Break | 15 minutes |
| • Forecasting | 60 minutes |
| • Closing discussion | 15 minutes |

# What is Machine Learning?

"The goal of machine learning is to build computer systems that can adapt and learn from their experience."

– Tom Dietterich





# An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- A decision is needed: whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- Problem: to predict high-risk patients and discriminate them from low-risk patients.

# Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- Problem: to decide whether an application should be approved, or to classify applications into two categories, approved and not approved.

# Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have "experiences".
- A computer system learns from data, which represent some "past experiences" of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.

# The data and the goal

- Data: A set of data records (also called examples, instances or cases) described by
  - *k* attributes: $A_1$, $A_2$, ... $A_k$.
  - a class: Each example is labelled with a pre-defined class.

- Goal: To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

# An example: data (loan application)

***Approved or not***

| ID | Age | Has_Job | Own_House | Credit_Rating | Class |
|---|---|---|---|---|---|
| 1 | young | false | false | fair | **No** |
| 2 | young | false | false | good | **No** |
| 3 | young | true | false | good | **Yes** |
| 4 | young | true | true | fair | **Yes** |
| 5 | young | false | false | fair | **No** |
| 6 | middle | false | false | fair | **No** |
| 7 | middle | false | false | good | **No** |
| 8 | middle | true | true | good | **Yes** |
| 9 | middle | false | true | excellent | **Yes** |
| 10 | middle | false | true | excellent | **Yes** |
| 11 | old | false | true | excellent | **Yes** |
| 12 | old | false | true | good | **Yes** |
| 13 | old | true | false | good | **Yes** |
| 14 | old | true | false | excellent | **Yes** |
| 15 | old | false | false | fair | **No** |

# An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

| Age | Has_Job | Own_house | Credit-Rating | Class |
|---|---|---|---|---|
| young | false | false | good | ? |

# Supervised vs. unsupervised Learning

- Supervised learning: classification is seen as supervised learning from examples.
  - Supervision: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a "teacher" gives the classes (supervision).
  - Test data are classified into these classes too.

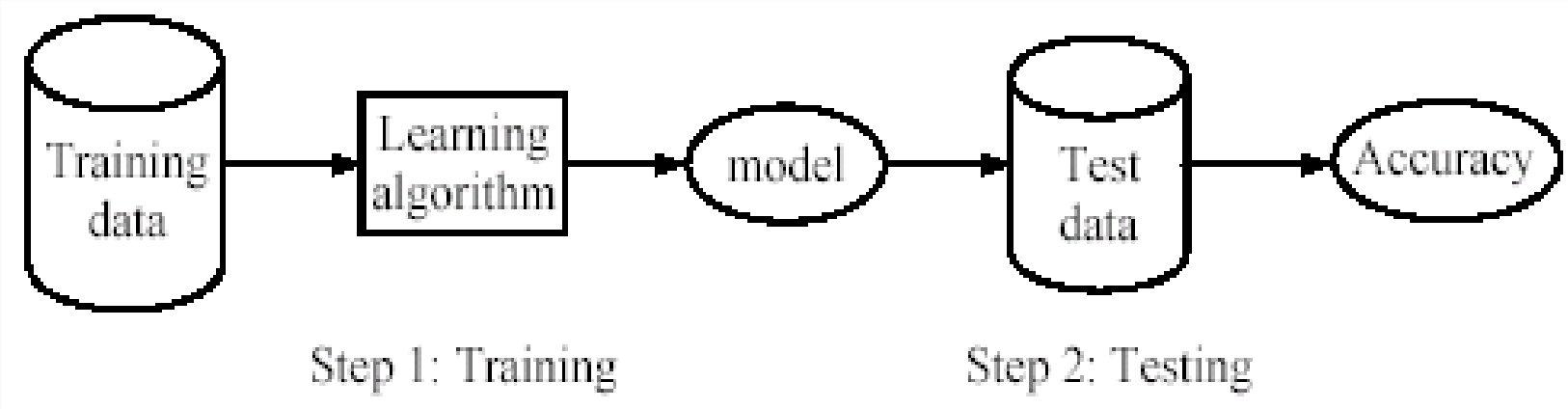
- Unsupervised learning (clustering)
  - Class labels of the data are unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

# Supervised learning process: two steps

Learning (training): Learn a model using the training data

Testing: Test the model using unseen test data to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



# What do we mean by learning?

- Given
  - a data set *D*,
  - a task *T*, and
  - a performance measure *M*,

  a computer system is said to **learn** from *D* to perform the task *T* if after learning the system's performance on *T* improves as measured by *M*.
- In other words, the learned model helps the system to perform *T* better as compared to no learning.

# An example

- Data: Loan application data
- Task: Predict whether a loan should be approved or not.
- Performance measure: accuracy.

No learning: classify all future applications (test data) to the majority class (i.e., Yes):

Accuracy = 9/15 = 60%.

- We can do better than 60% with learning.

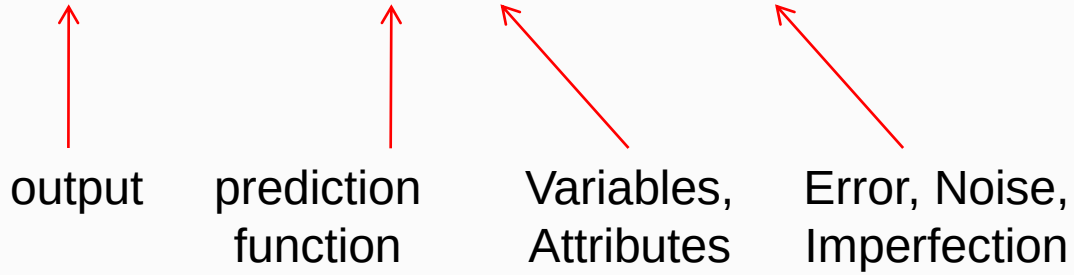
# Fundamental assumption of learning

Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

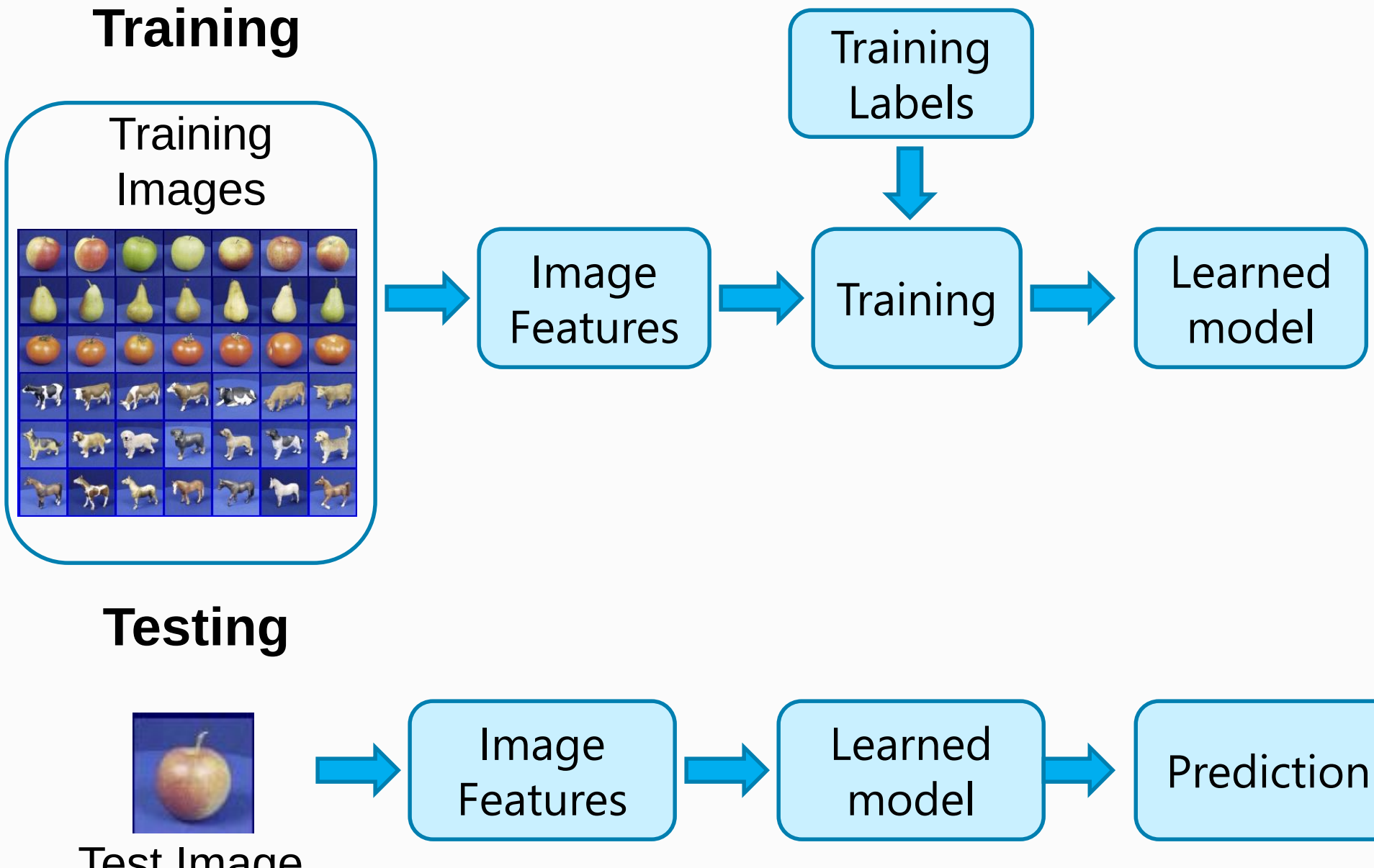
# The machine learning framework

$$y = f(\mathbf{x}) + e$$

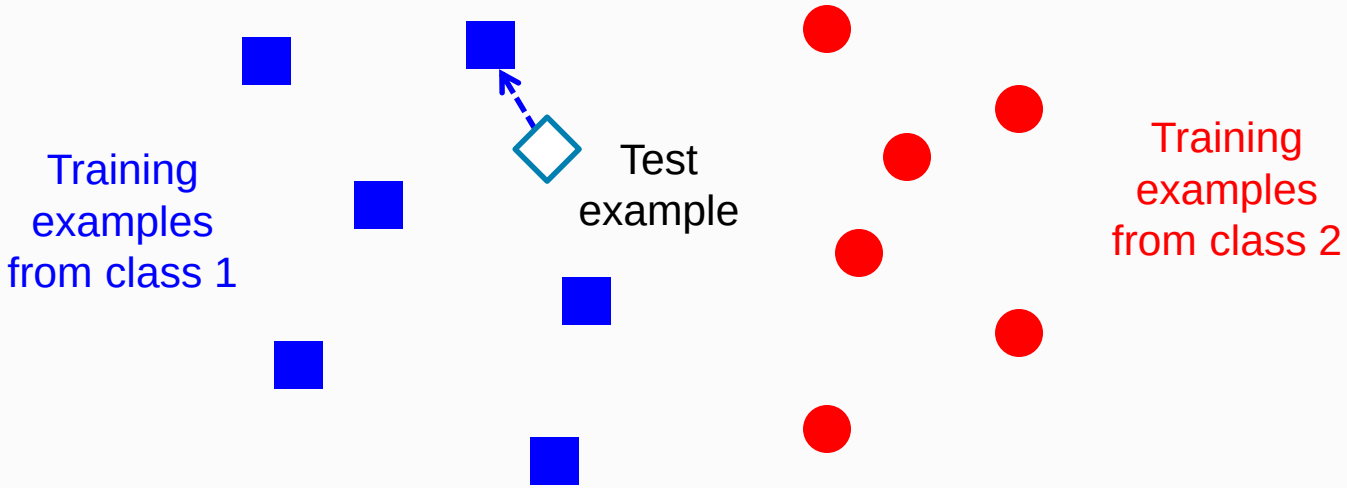



- **Training**: given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function $f$ by minimizing the prediction error on the training set
- **Testing**: apply $f$ to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

# Steps


Training
Training Images
Training Labels
Image Features
Training
Learned model
Testing
Test Image
Image Features
Learned model
Prediction

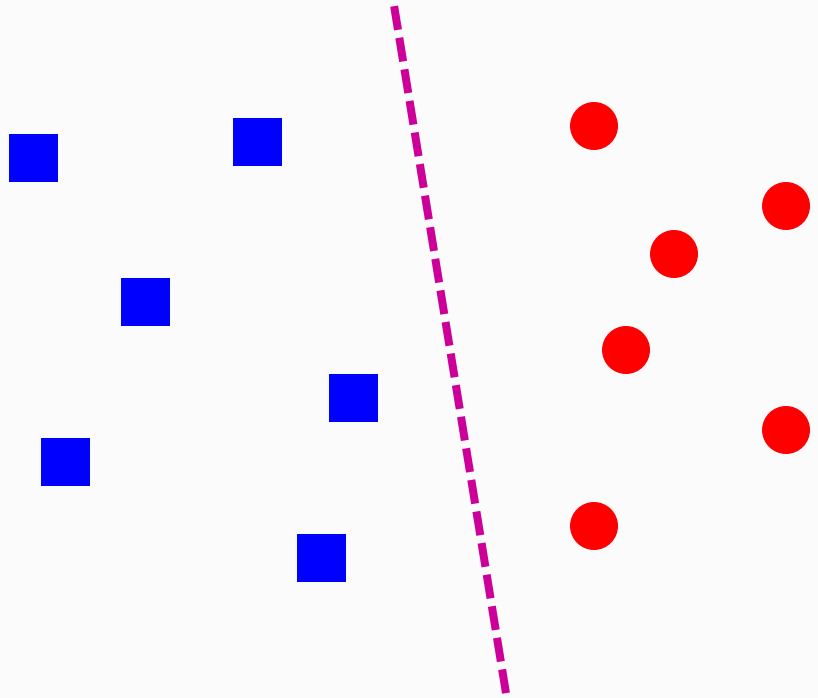
# Classifiers: Nearest neighbor




f(**x**) = label of the training example nearest to **x**

- All we need is a distance function for our inputs
- No training required!

# Classifiers: Linear



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

# Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- Etc.

*Which is the best one?*

# Generalization

Training set (labels known)

Test set (labels unknown)

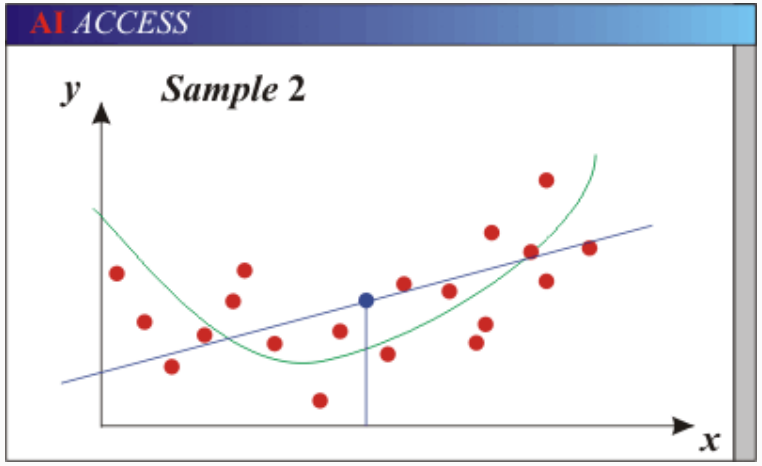
How well does a learned model generalize from the data it was trained on to a new test set?

# Generalization

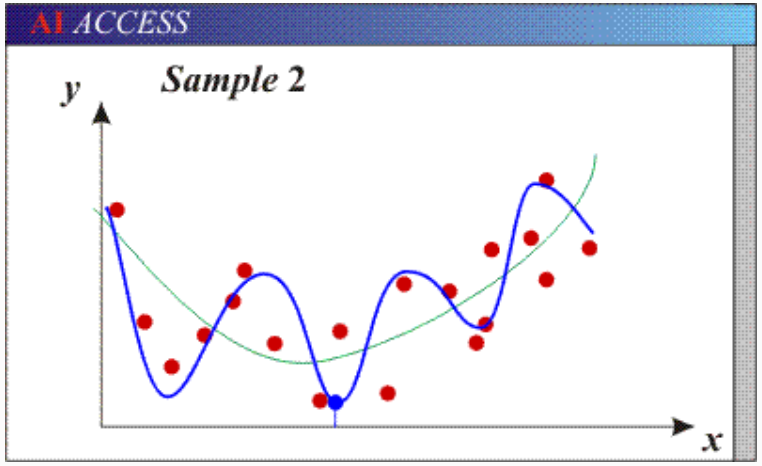
- Components of generalization error
  - **Bias**: how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance**: how much models estimated from different training sets differ from each other
- **Underfitting**: model is too "simple" to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting**: model is too "complex" and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

# No Free Lunch Theorem

# Bias-Variance Trade-off




- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).




- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

# Bias-Variance Trade-off

$$E(MSE) = noise^2 + bias^2 + variance$$

Unavoidable error

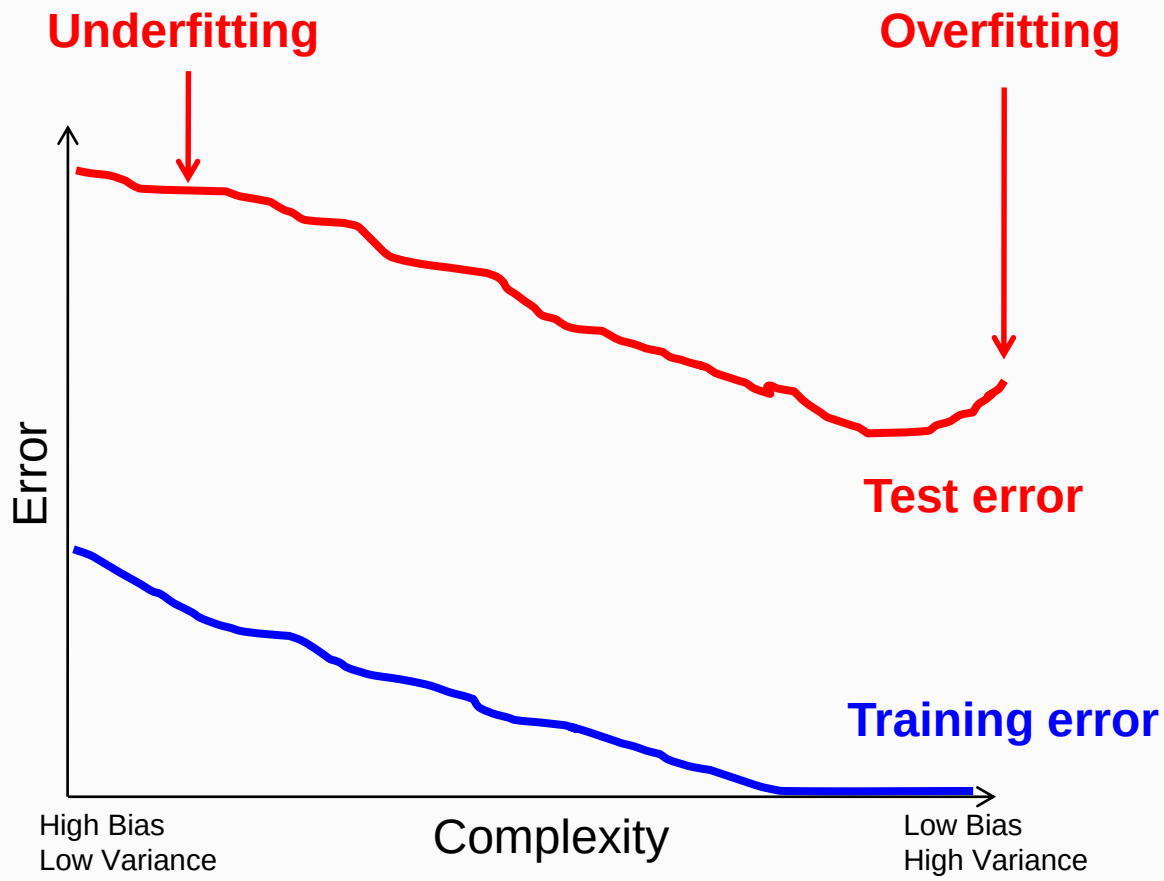
Error due to incorrect assumptions

Error due to variance of training samples

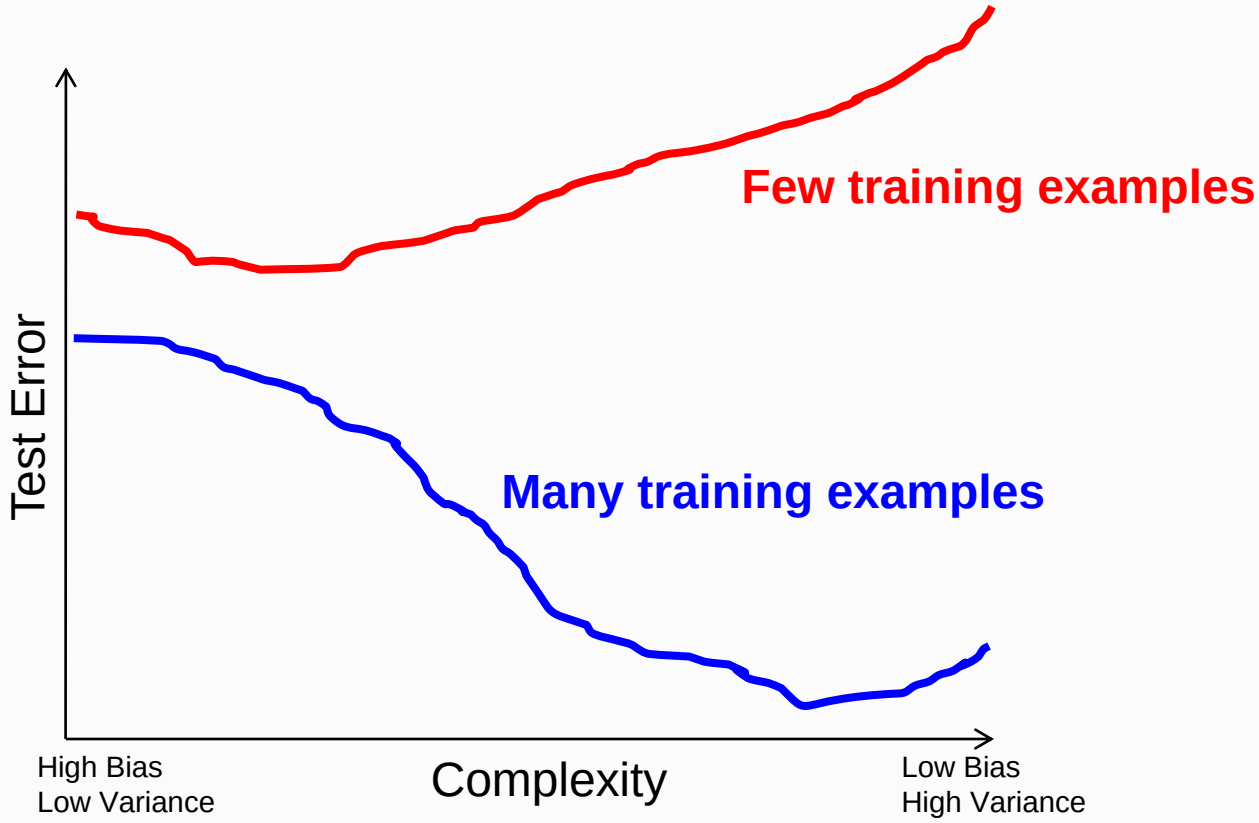
See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):

- http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

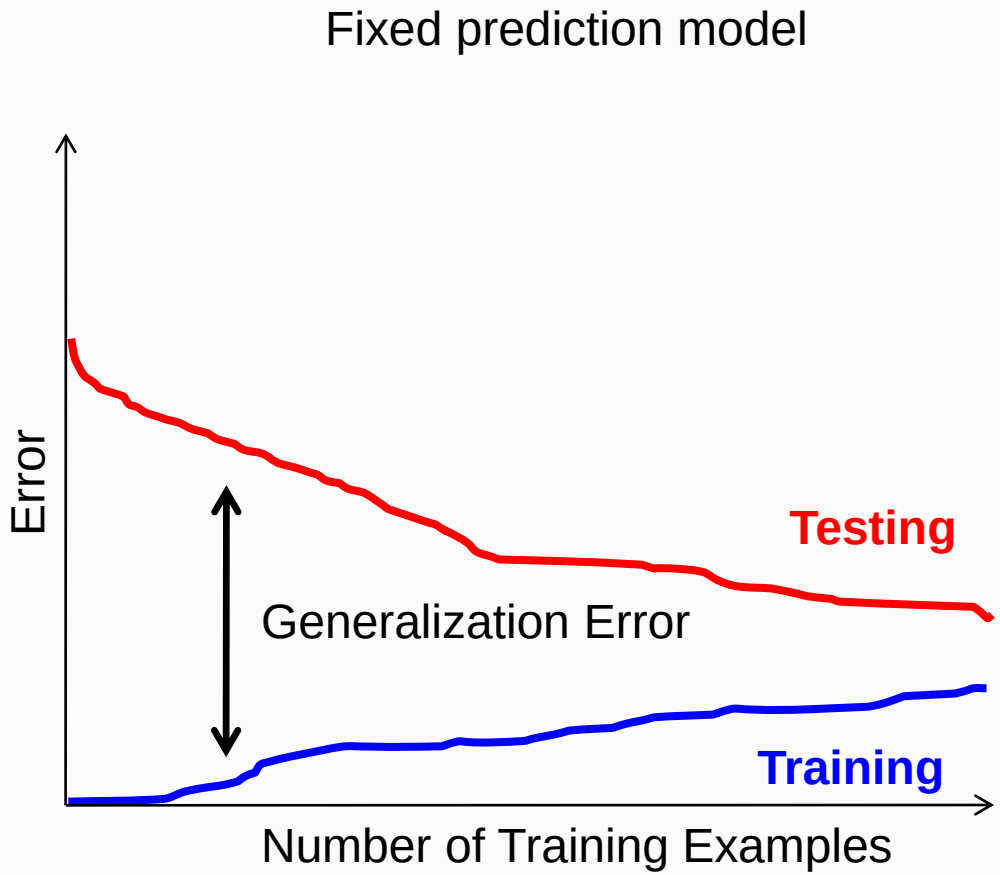
# Bias-variance tradeoff



# Bias-variance tradeoff



# Effect of Training Size



# Remember…

- No classifier is inherently better than any other: you need to make assumptions to generalize

- Three kinds of error
  - Inherent: unavoidable
  - Bias: due to over-simplifications
  - Variance: due to inability to perfectly estimate parameters from limited data

# How to reduce variance?

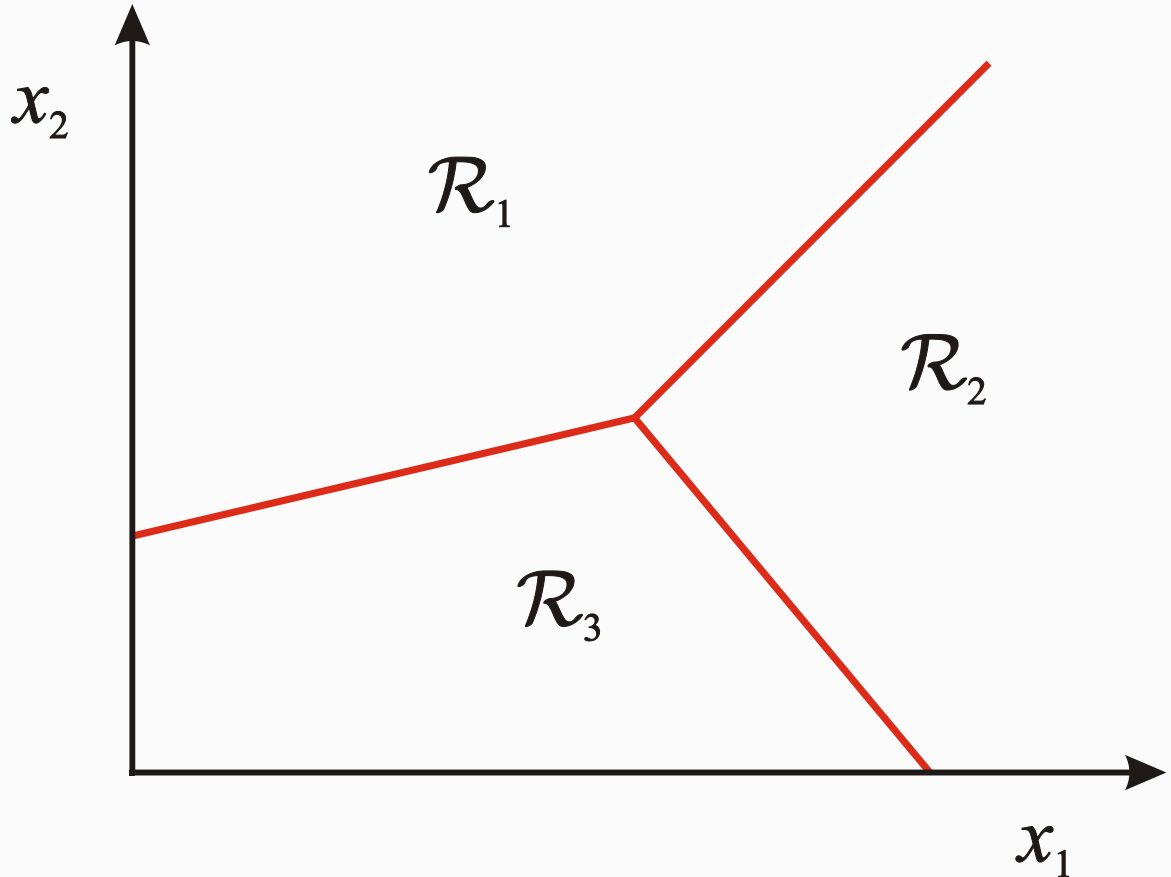
- Choose a simpler classifier
- Regularize the parameters
- Get more training data

# Very brief tour of some classifiers

- K-nearest neighbor
- SVM
- Boosted Decision Trees
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Etc.

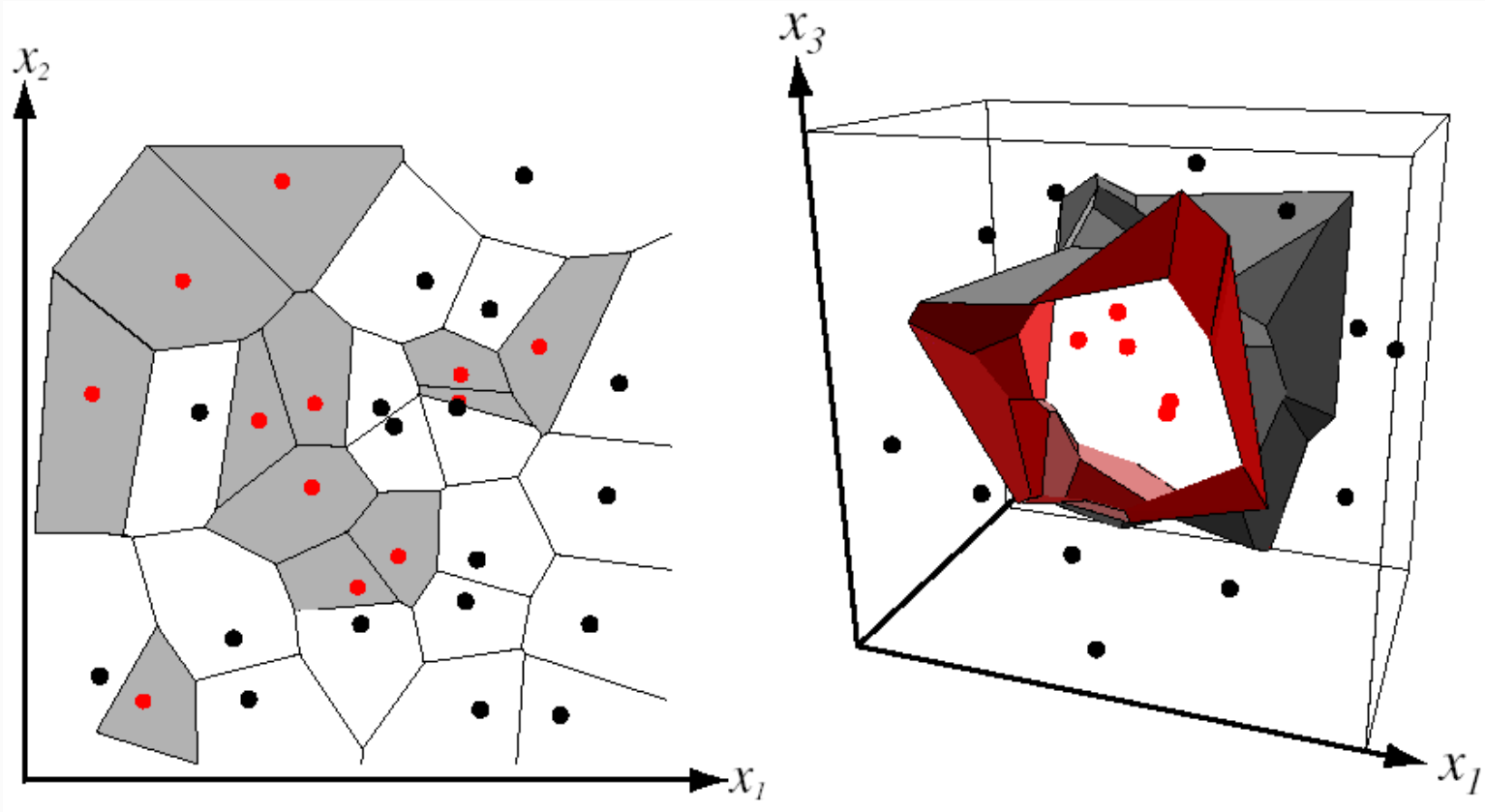
# Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



# Nearest Neighbor Classifier

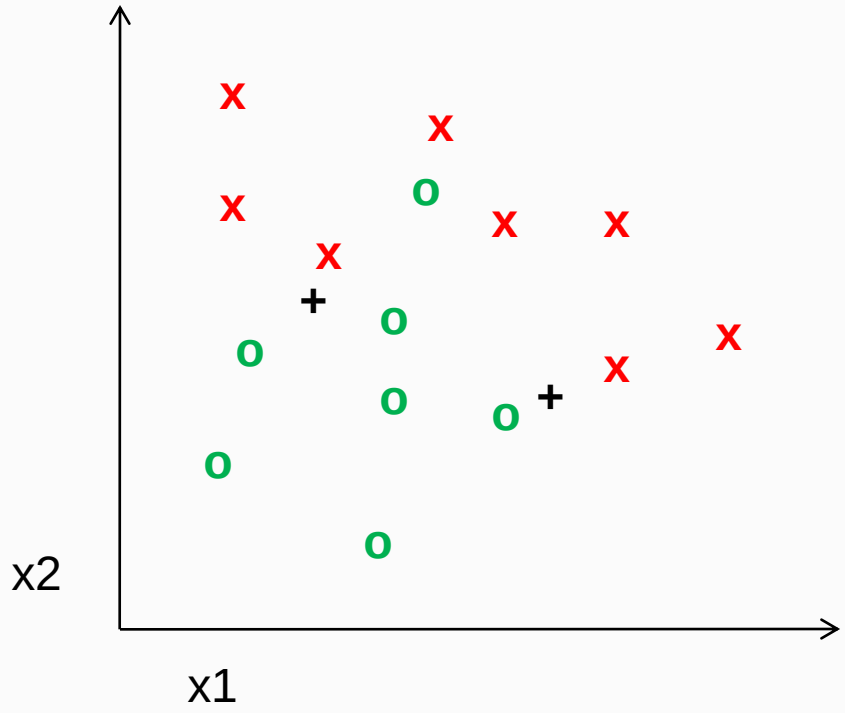
- Assign label of nearest training data point to each test data point



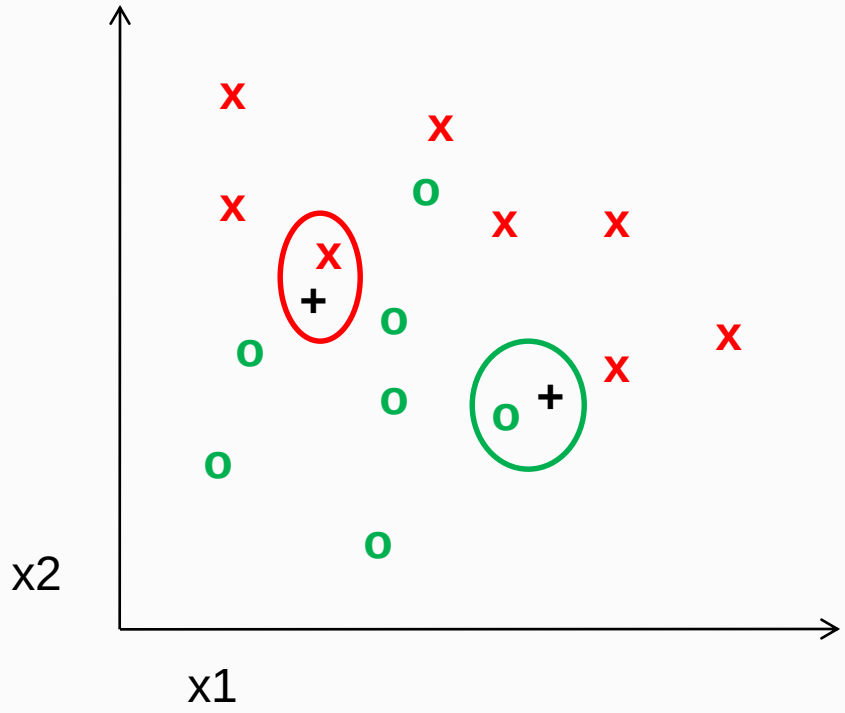

from Duda *et al.*

Voronoi partitioning of feature space
for two-category 2D and 3D data

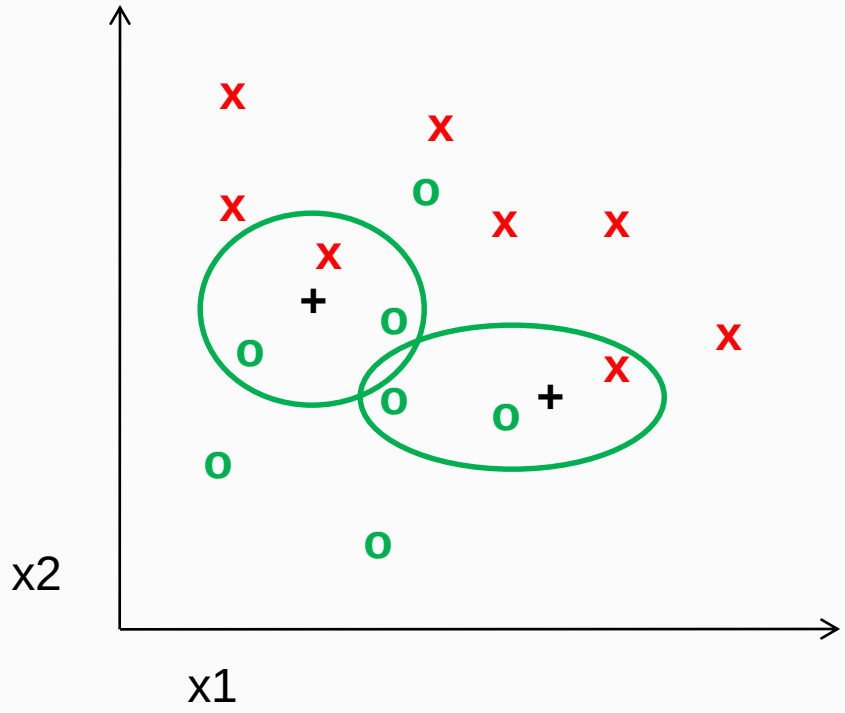
# K-nearest neighbor



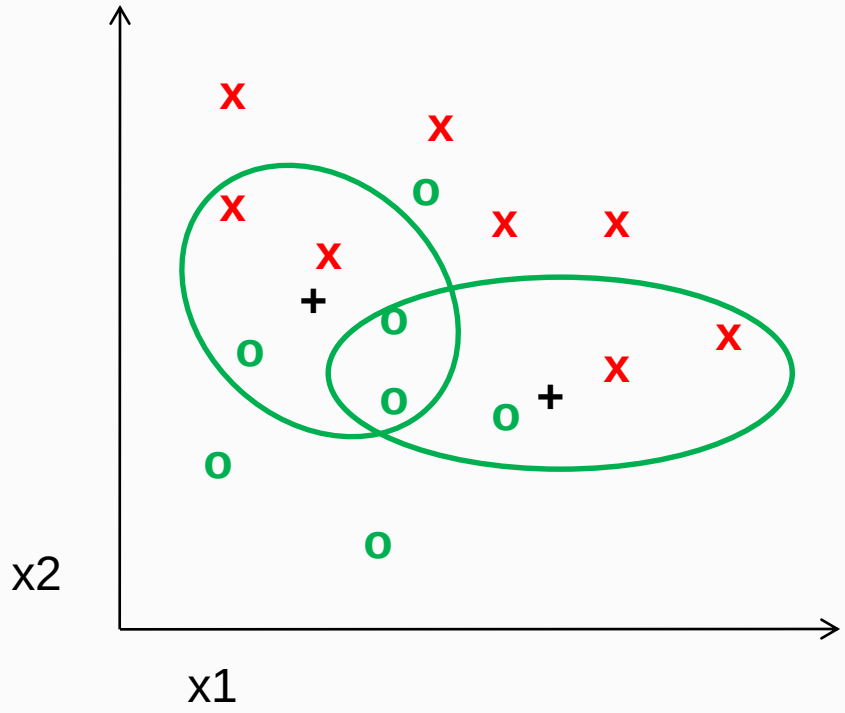
# 1-nearest neighbor



# 3-nearest neighbor



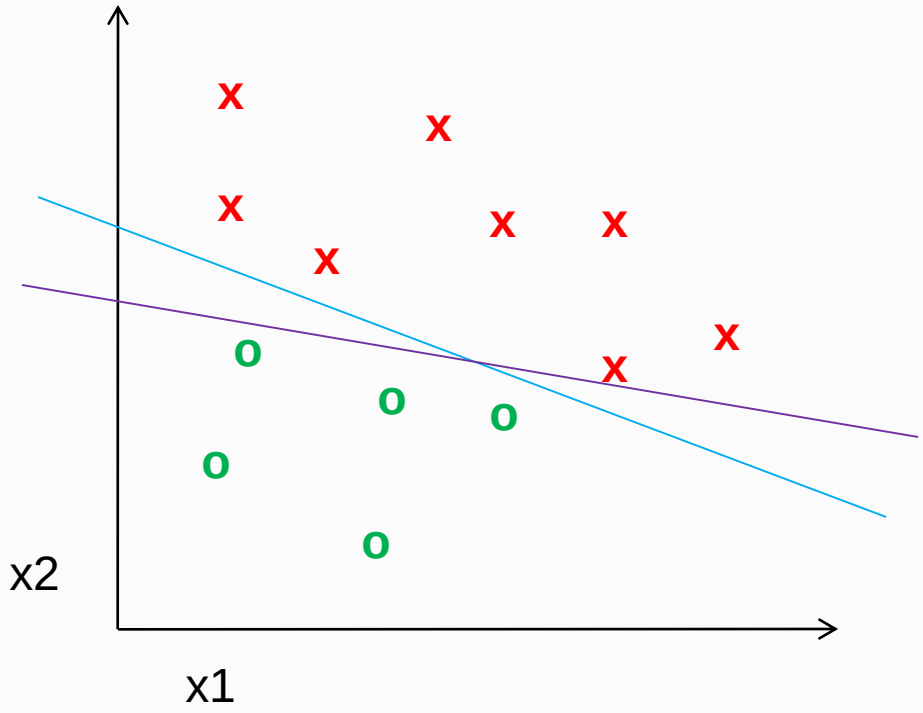
# 5-nearest neighbor



# Using K-NN

- Simple, a good one to try first
- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error

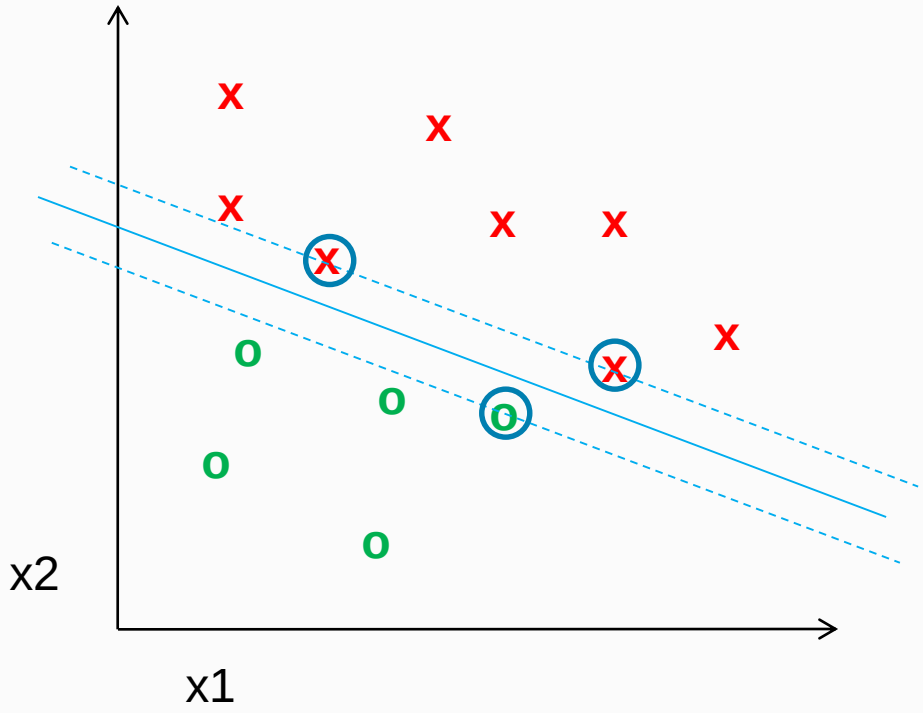
# Classifiers: Linear SVM




- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

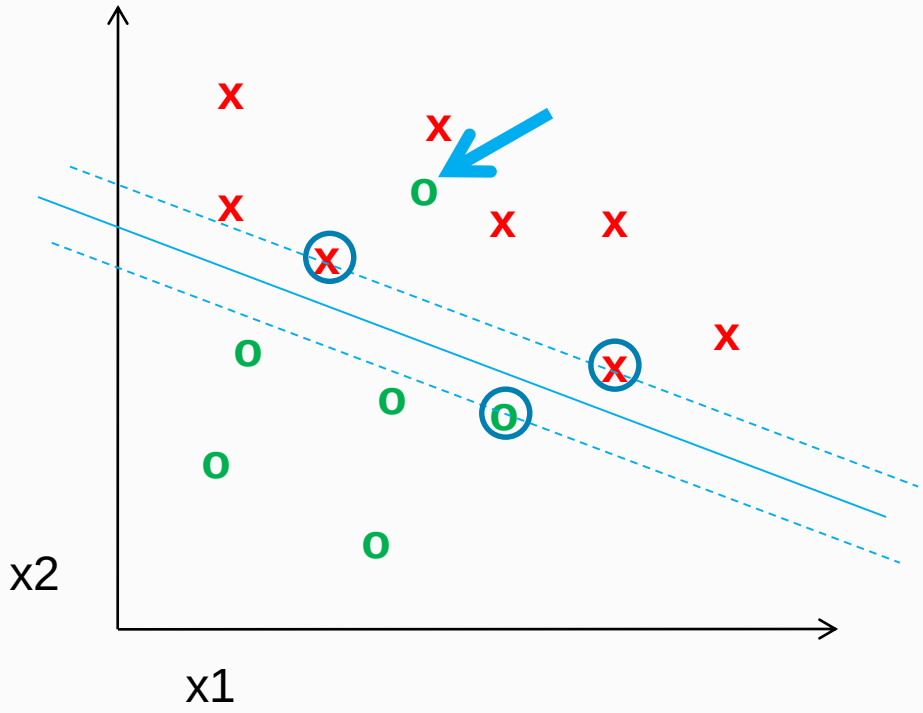
# Classifiers: Linear SVM




- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

# Classifiers: Linear SVM

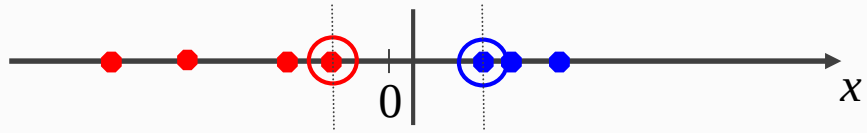



- Find a *linear function* to separate the classes:

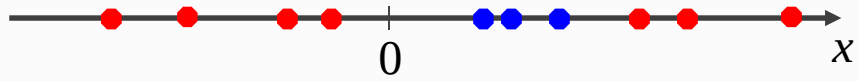
$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

# Nonlinear SVMs

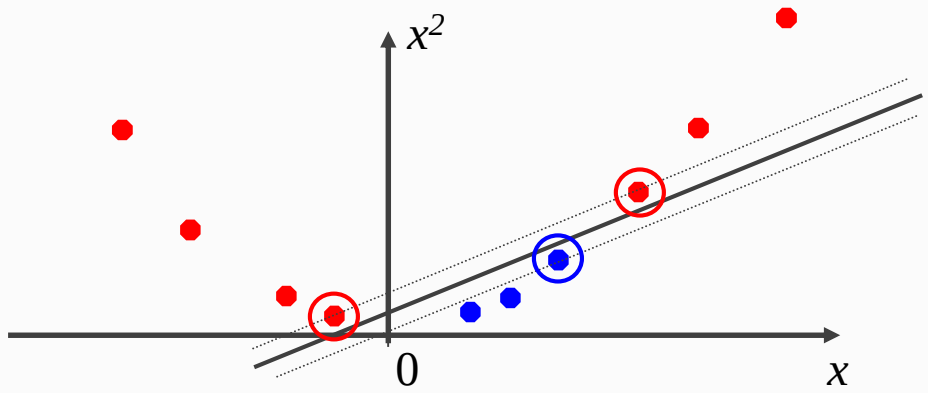
- Datasets that are linearly separable work out great:



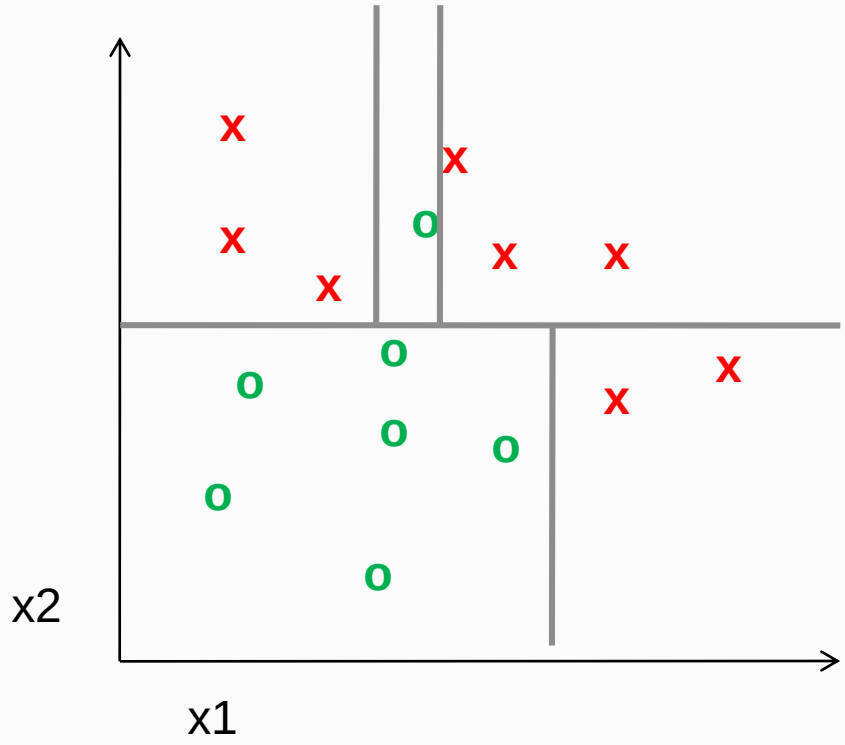

- But what if the dataset is just too hard?




- We can map it to a higher-dimensional space:



# Classifiers: Decision Trees

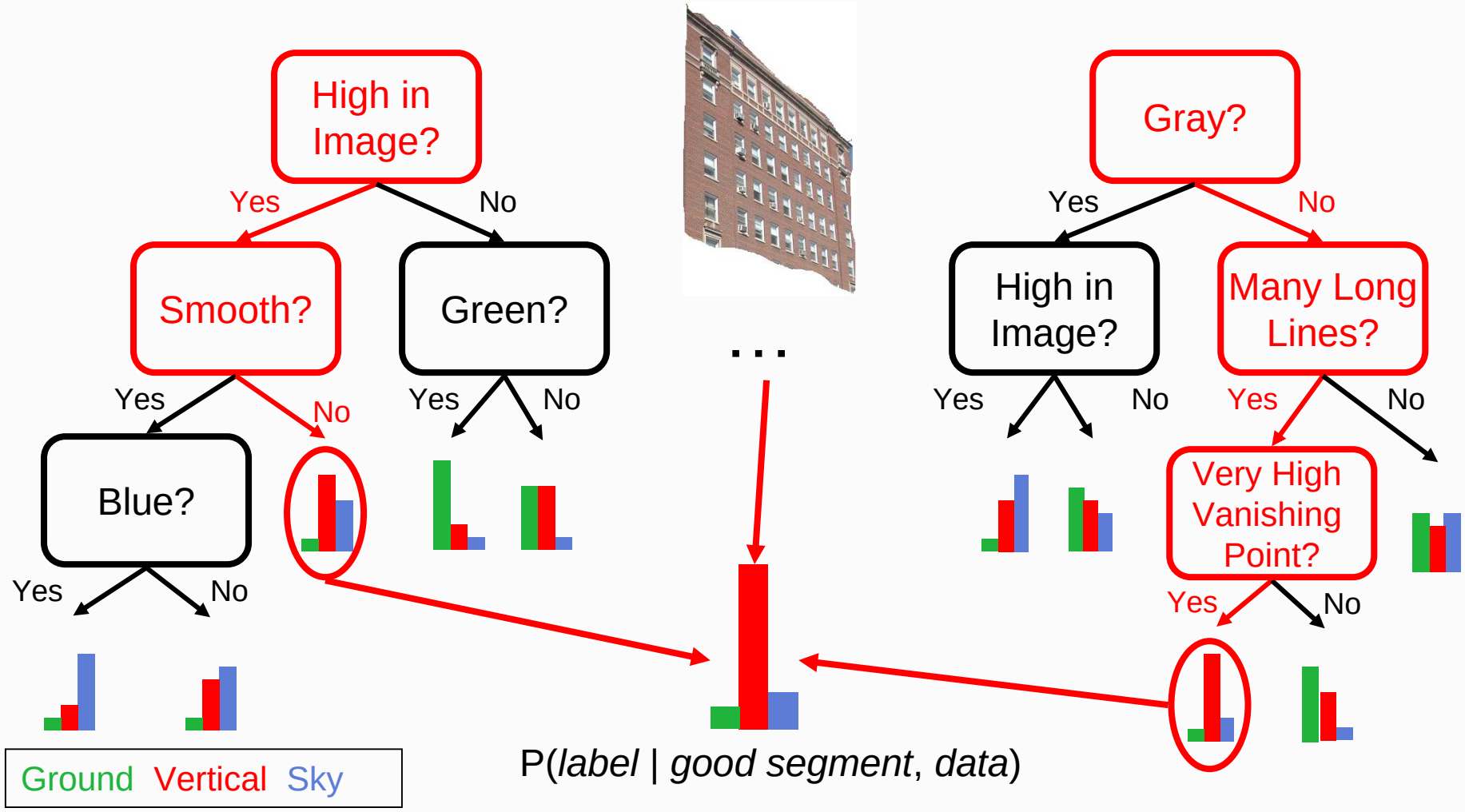


# Ensemble Methods: Boosting

**Discrete AdaBoost(Freund & Schapire 1996*b*)**

1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights $w_i$ on the training data.
   (b) Compute $\mathrm{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\mathrm{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

# Boosted Decision Trees



# Model Validation

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (50%)
  - Validation set (25%)
  - Test (publication) set (25%)

- Resample when there is few data
- Resample from the minority when there is data skew

# Evaluating Models

- Infinite data is best, but…
- N (N=10) Fold cross validation
  - Create N folds or subsets from the training data (approximately equally distributed with approximately the same number of instances).
  - Build N models, each with a different set of N-1 folds, and evaluate each model on the remaining fold
  - Error estimate is average error over all N models

# What to remember about classifiers

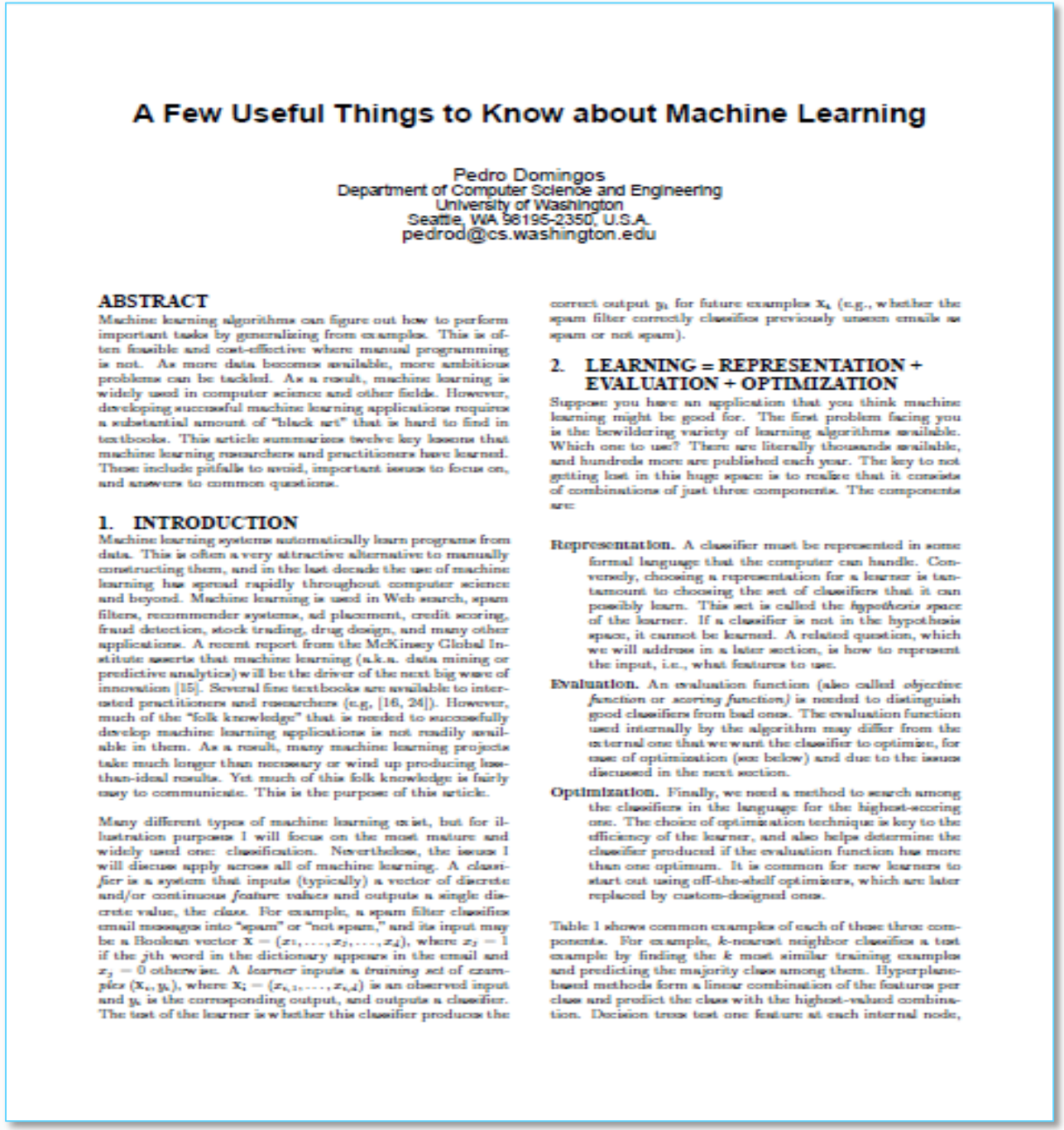
- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

# Some Machine Learning References

- General
  - Tom Mitchell, *Machine Learning*, McGraw Hill, 1997
  - Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995
  - Hastie, Tibshirani, Friedman, "Elements of Statistical Learning: Data Mining, Inference, Prediction" (2001, 2008)
- Adaboost
  - Friedman, Hastie, and Tibshirani, "Additive logistic regression: a statistical view of boosting", Annals of Statistics, 2000
- SVMs
  - http://www.support-vector.net/icml-tutorial.pdf

# Out of Class Reading

## Week Two



**A Few Useful Things to Know about Machine Learning**

Pedro Domingos
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
pedrod@cs.washington.edu

**ABSTRACT**

Machine learning algorithms can figure out how to perform important tasks by generalizing from examples. This is often feasible and cost-effective where manual programming is not. As more data becomes available, more ambitious problems can be tackled. As a result, machine learning is widely used in computer science and other fields. However, developing successful machine learning applications requires a substantial amount of "black art" that is hard to find in textbooks. This article summarizes twelve key lessons that machine learning researchers and practitioners have learned. These include pitfalls to avoid, important issues to focus on, and answers to common questions.

**1. INTRODUCTION**

Machine learning systems automatically learn programs from data. This is often a very attractive alternative to manually constructing them, and in the last decade the use of machine learning has spread rapidly throughout computer science and beyond. Machine learning is used in Web search, spam filters, recommender systems, ad placement, credit scoring, fraud detection, stock trading, drug design, and many other applications. A recent report from the McKinsey Global Institute asserts that machine learning (a.k.a. data mining or predictive analytics) will be the driver of the next big wave of innovation [15]. Several fine textbooks are available to interested practitioners and researchers (e.g., [16, 24]). However, much of the "folk knowledge" that is needed to successfully develop machine learning applications is not readily available in them. As a result, many machine learning projects take much longer than necessary or wind up producing less-than-ideal results. Yet much of this folk knowledge is fairly easy to communicate. This is the purpose of this article.

Many different types of machine learning exist, but for illustration purposes I will focus on the most mature and widely used one: classification. Nevertheless, the issues I will discuss apply across all of machine learning. A *classifier* is a system that inputs (typically) a vector of discrete and/or continuous *feature* values and outputs a single discrete value, the *class*. For example, a spam filter classifies email messages into "spam" or "not spam," and its input may be a Boolean vector $\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_d)$, where $x_j = 1$ if the $j$th word in the dictionary appears in the email and $x_j = 0$ otherwise. A *learner* inputs a *training set* of *examples* $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})$ is an observed input and $y_i$ is the corresponding output, and outputs a classifier. The test of the learner is whether this classifier produces the correct output $y_t$ for future examples $\mathbf{x}_t$ (e.g., whether the spam filter correctly classifies previously unseen emails as spam or not spam).

**2. LEARNING = REPRESENTATION + EVALUATION + OPTIMIZATION**

Suppose you have an application that you think machine learning might be good for. The first problem facing you is the bewildering variety of learning algorithms available. Which one to use? There are literally thousands available, and hundreds more are published each year. The key to not getting lost in this huge space is to realize that it consists of combinations of just three components. The components are:

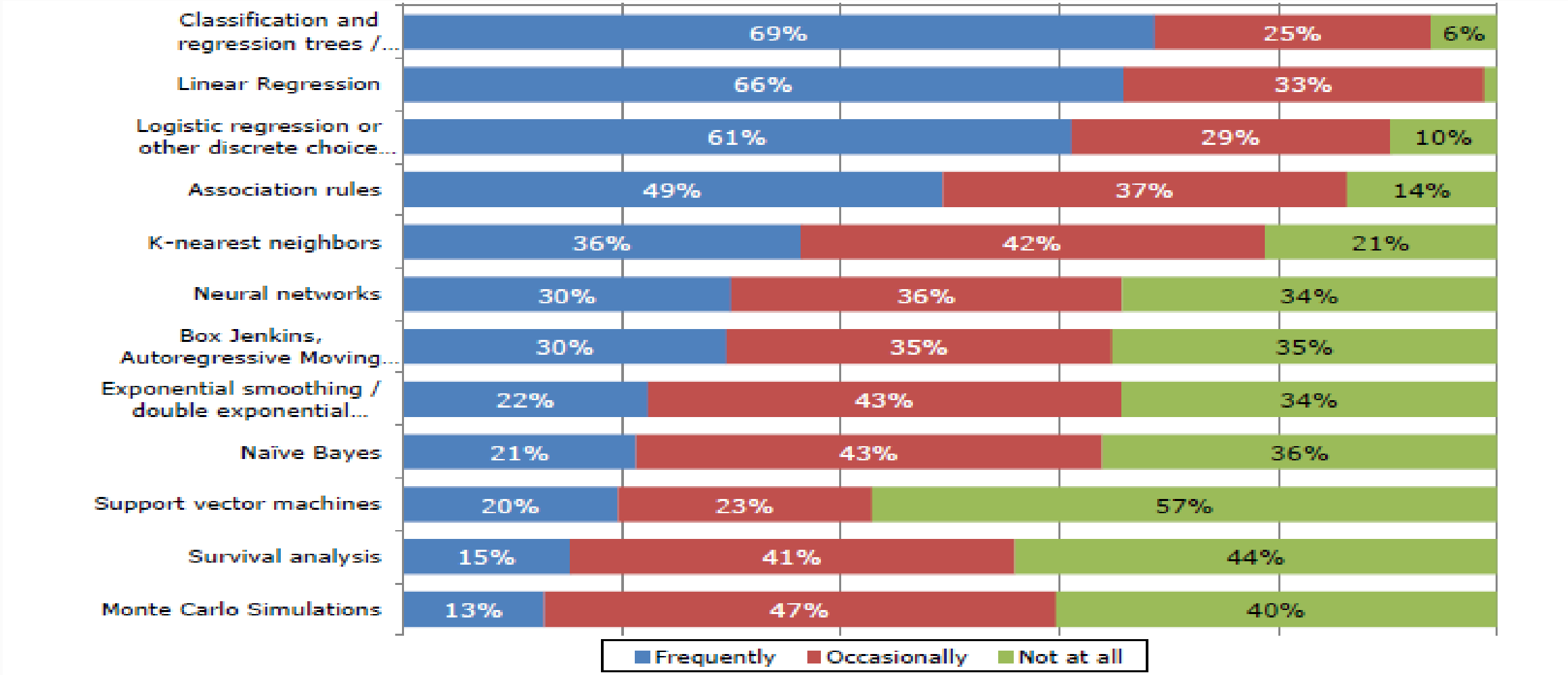
**Representation.** A classifier must be represented in some formal language that the computer can handle. Conversely, choosing a representation for a learner is tantamount to choosing the set of classifiers that it can possibly learn. This set is called the *hypothesis space* of the learner. If a classifier is not in the hypothesis space, it cannot be learned. A related question, which we will address in a later section, is how to represent the input, i.e., what features to use.

**Evaluation.** An evaluation function (also called objective *function* or *scoring function*) is needed to distinguish good classifiers from bad ones. The evaluation function used internally by the algorithm may differ from the external one that we want the classifier to optimize, for ease of optimization (see below) and due to the issues discussed in the next section.

**Optimization.** Finally, we need a method to search among the classifiers in the language for the highest-scoring one. The choice of optimization technique is key to the efficiency of the learner, and also helps determine the classifier produced if the evaluation function has more than one optimum. It is common for new learners to start out using off-the-shelf optimizers, which are later replaced by custom-designed ones.

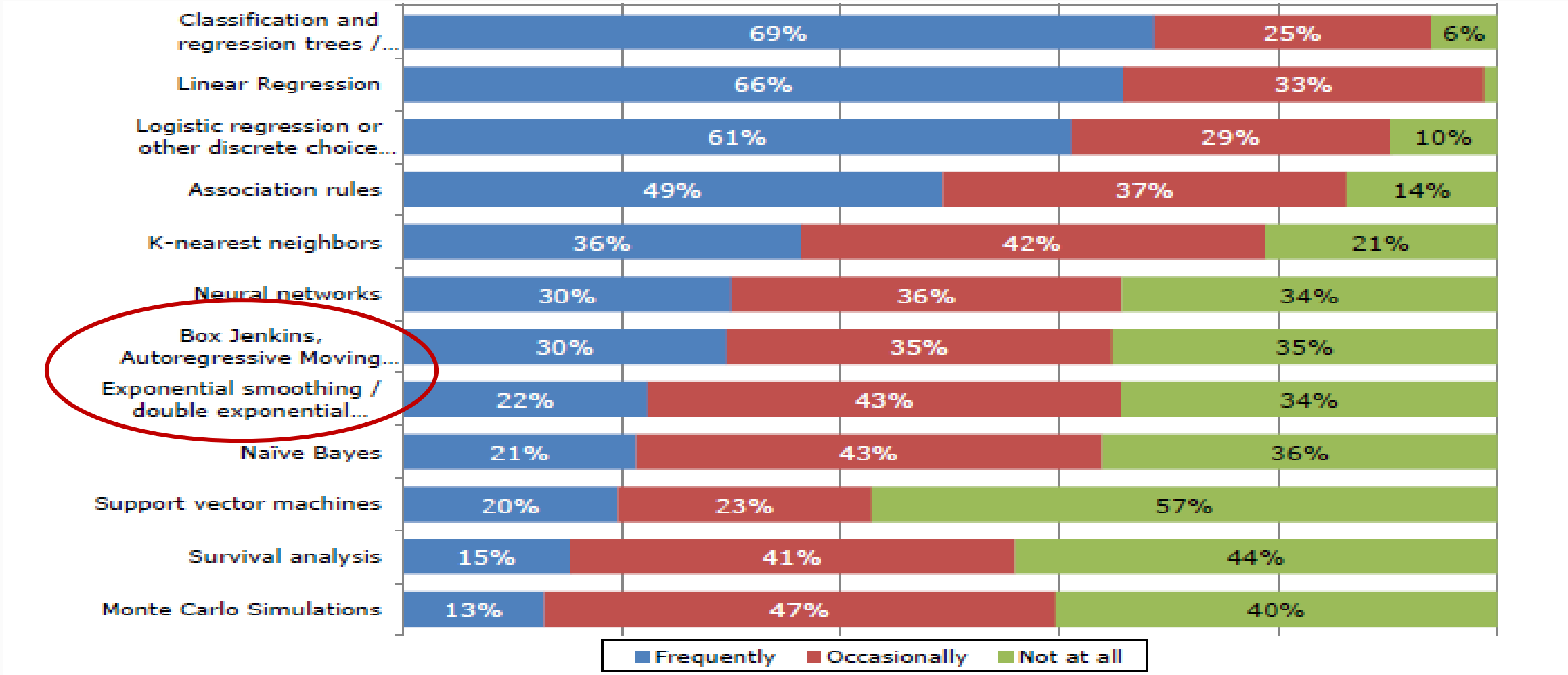
Table 1 shows common examples of each of these three components. For example, $k$-nearest neighbor classifies a test example by finding the $k$ most similar training examples and predicting the majority class among them. Hyperplane-based methods form a linear combination of the features per class and predict the class with the highest-valued combination. Decision trees test one feature at each internal node,

# Organizations Employ a Variety of Predictive Analytics Algorithms




*Classification and regression trees / decision trees and Linear Regression are the most popular predictive analytics techniques used.*

# Organizations Employ a Variety of Predictive Analytics Algorithms




*Classification and regression trees / decision trees and Linear Regression are the most popular predictive analytics techniques used.*

# Lecture Outline

| | |
|---|---|
| • Class Discussions | 20 minutes |
| • Machine Learning Primer | 60 minutes |
| • Break | 10 minutes |
| • Forecasting, 1/2 | 60 minutes |
| • Closing discussion | 15 minutes |

# Lecture Outline

| | |
|---|---|
| • Class Discussions | 30 minutes |
| • Machine Learning Primer | 60 minutes |
| • Break | 15 minutes |
| • Forecasting, 1/2 | 60 minutes |
| • Closing discussion | 15 minutes |

# Forecasting

- Predict the next number in the pattern:

a) 3.7, 3.7, 3.7, 3.7, 3.7, ?

b) 2.5, 4.5, 6.5, 8.5, 10.5, ?

c) 5.0, 7.5, 6.0, 4.5, 7.0, 9.5, 8.0, 6.5, ?

# Forecasting

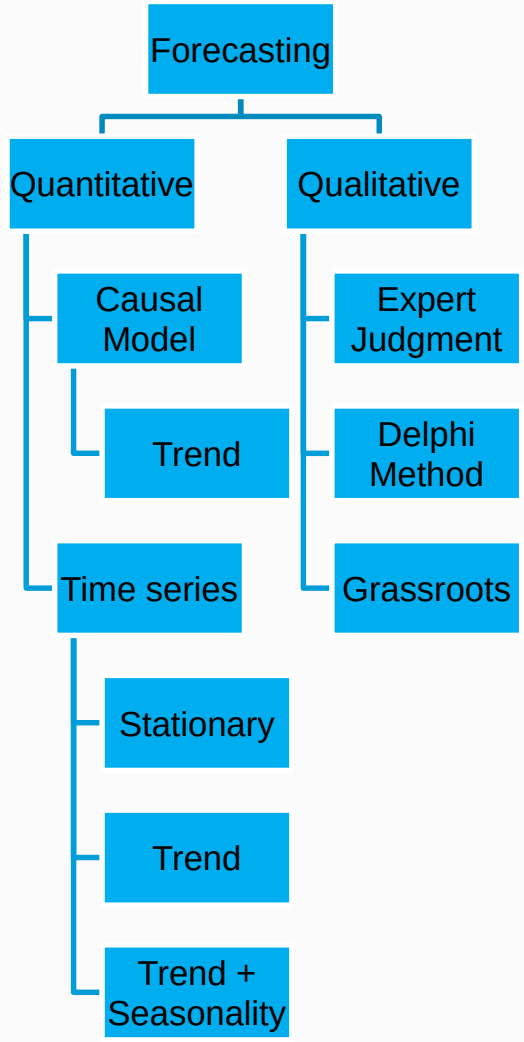
- Predict the next number in the pattern:

a) 3.7, 3.7, 3.7, 3.7, 3.7, 3.7

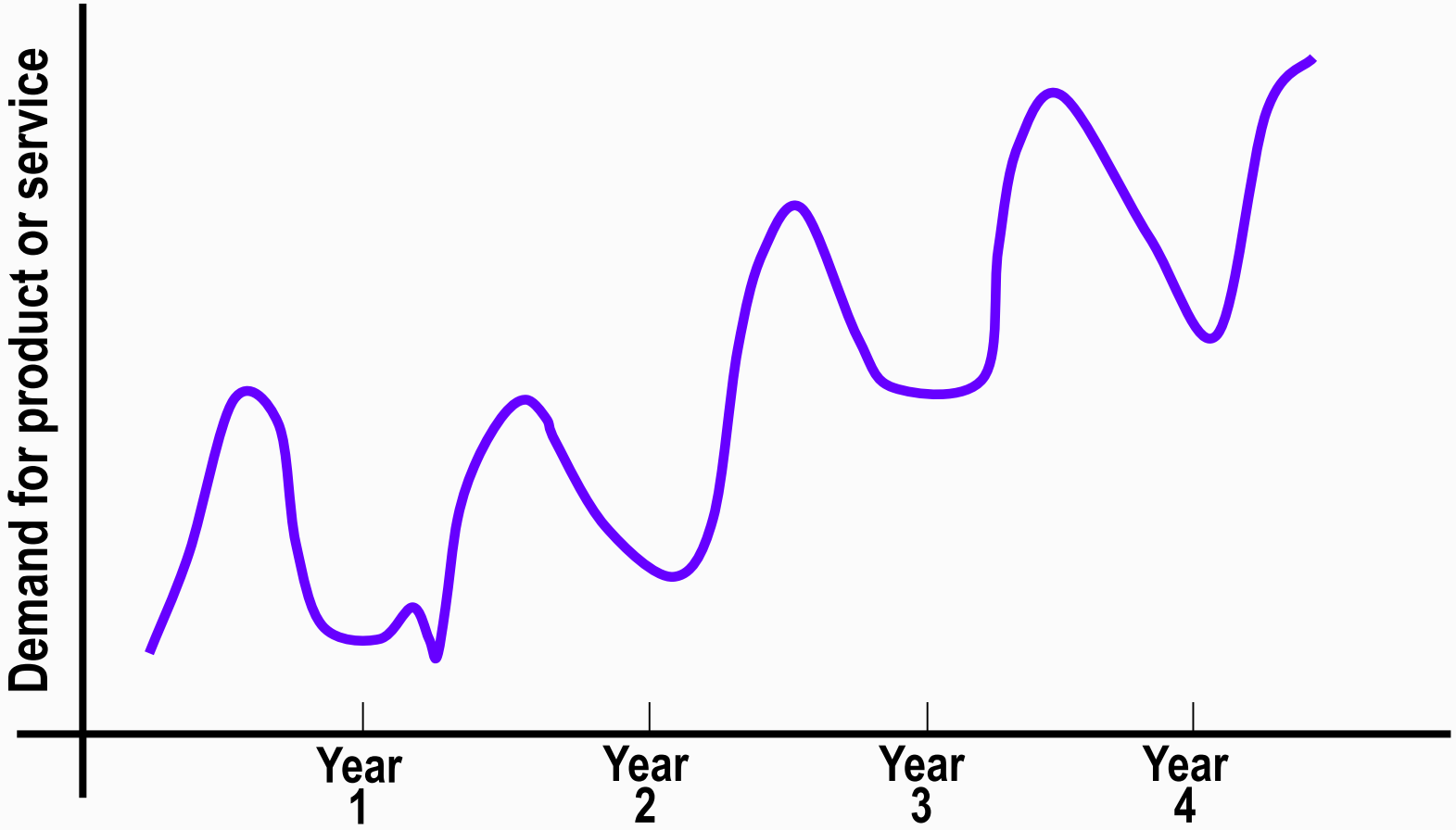
b) 2.5, 4.5, 6.5, 8.5, 10.5, 12.5

c) 5.0, 7.5, 6.0, 4.5, 7.0, 9.5, 8.0, 6.5, 9.0

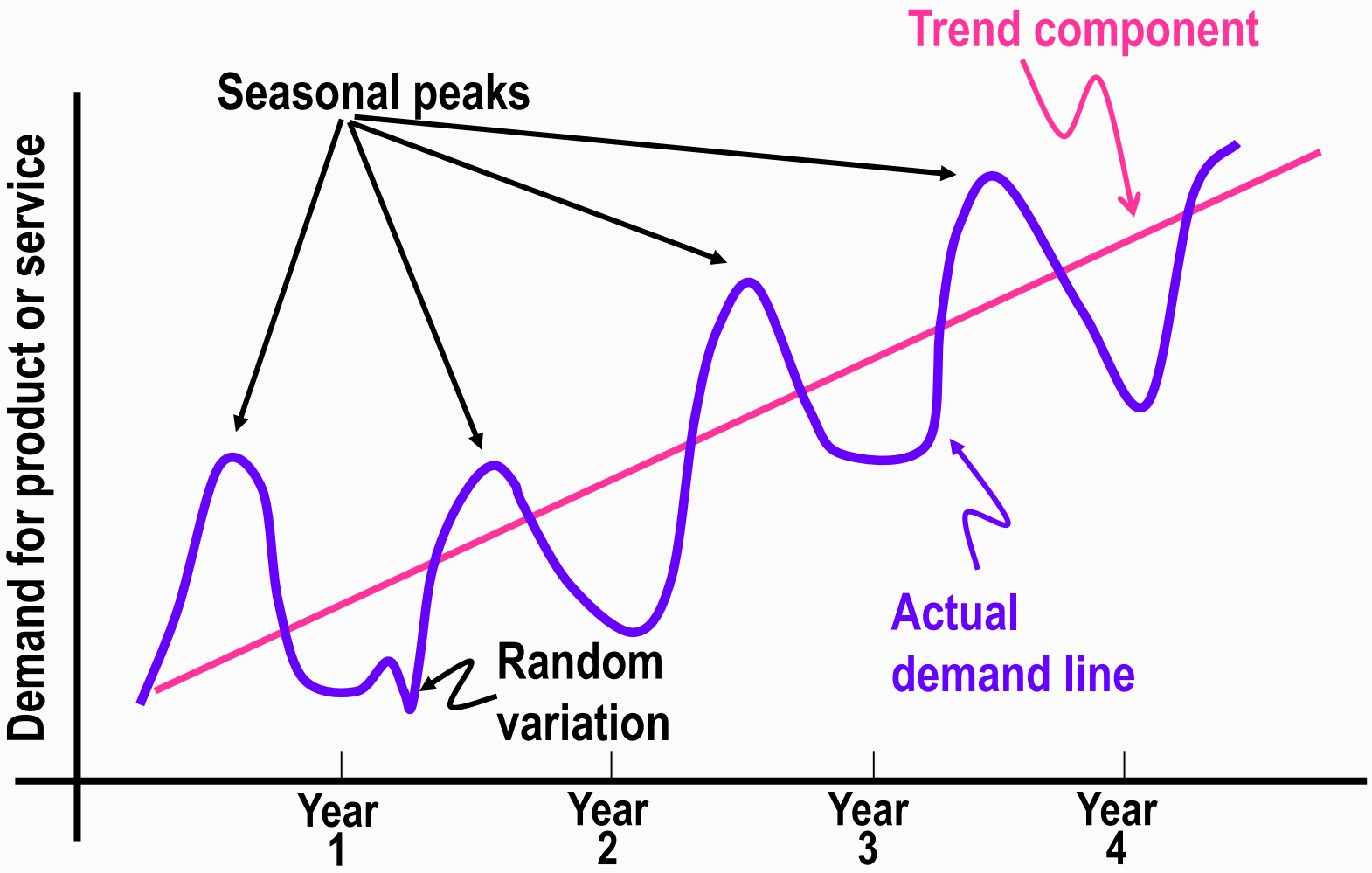
# Forecasting



# Product Demand over Time



# Product Demand over Time



# Cyclical Variation – Sample Chart

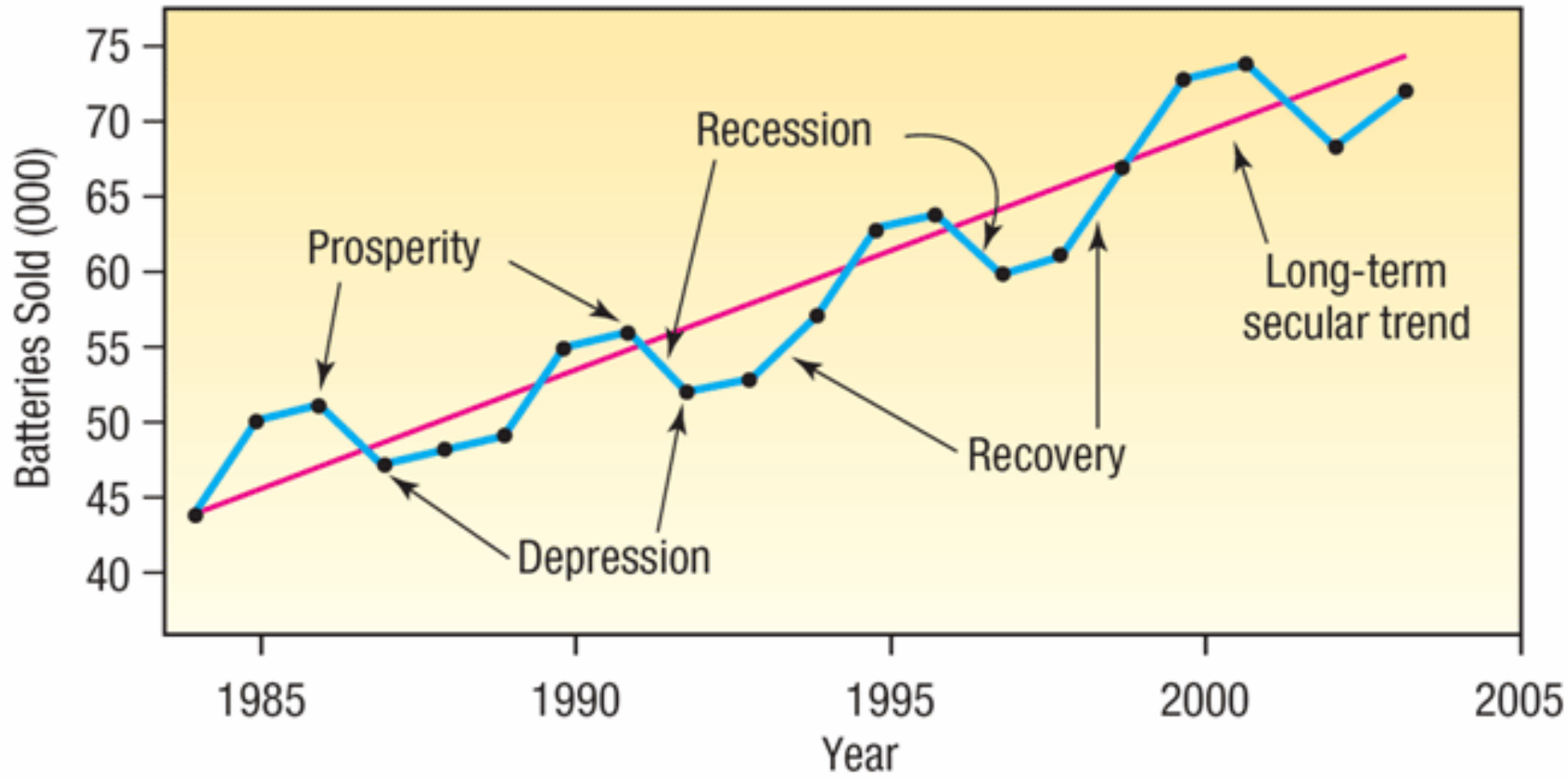



**CHART 16–1** Batteries Sold by National Battery Retailers, Inc., from 1984 to 2004

# Seasonal Variation – Sample Chart

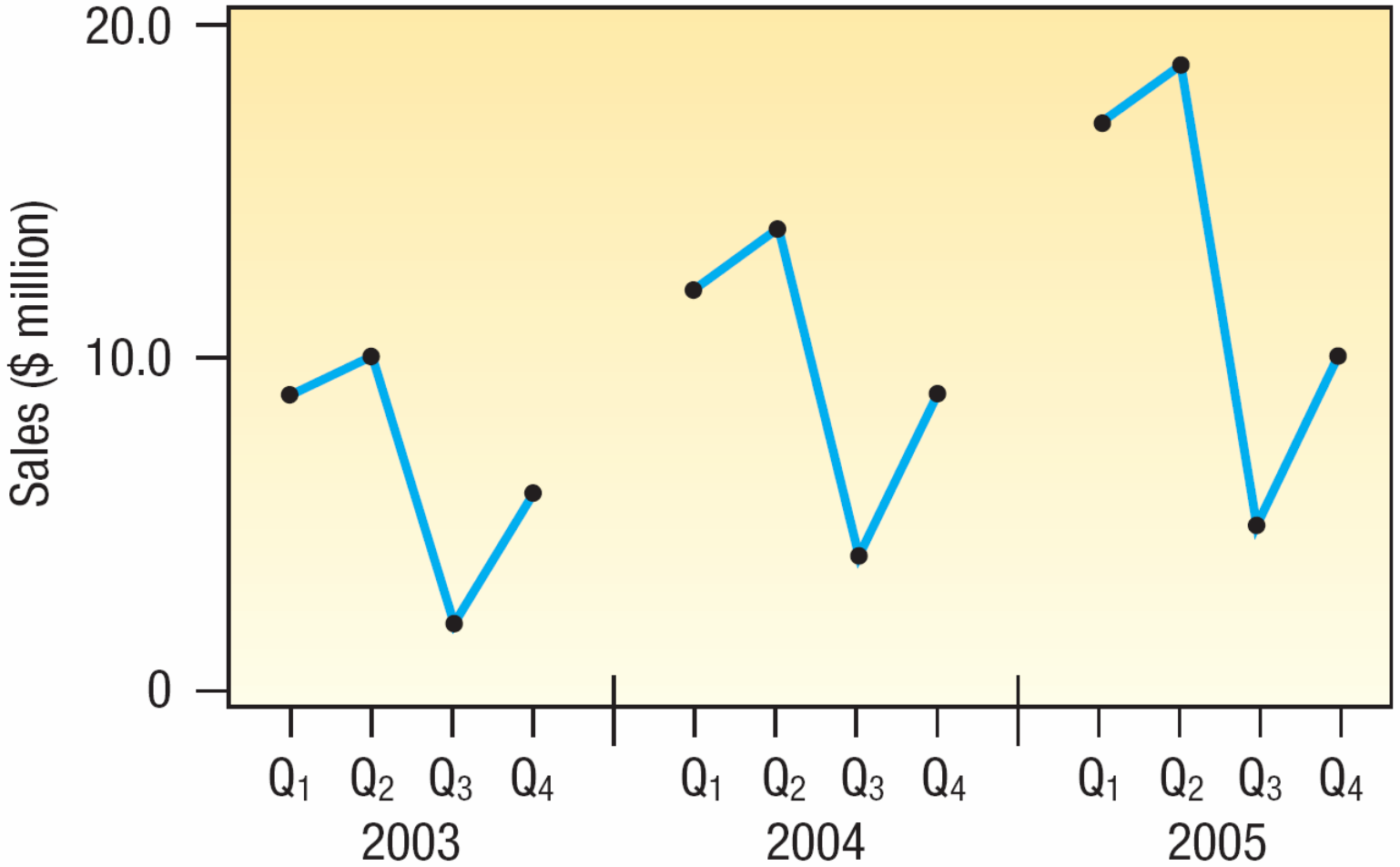



**CHART 16–2** Sales of Baseball and Softball Equipment, Hercher Sporting Goods, 2003–2005 by Quarter

# Outline

- What is forecasting?
- Types of forecasts
- Time-Series forecasting
  - Naïve
  - Moving Average
  - Exponential Smoothing
  - Regression
- Good forecasts

# Stationary data forecasting

Naïve

- *I sold 10 units yesterday, so I think I will sell 10 units today.*

n-period moving average

- *For the past n days, I sold 12 units on average. Therefore, I think I will sell 12 units today.*

Exponential smoothing

- *I predicted to sell 10 units at the beginning of yesterday; At the end of yesterday, I found out I sold in fact 8 units. So, I will adjust the forecast of 10 (yesterday's forecast) by adding adjusted error (α * error). This will compensate over (under) forecast of yesterday.*

# Naïve Model

- The simplest time series forecasting model
- Idea: "what happened last time (last year, last month, yesterday) will happen again this time"
- Naïve Model:
  - Algebraic: $F_t = Y_{t-1}$
    - $Y_{t-1}$ : actual value in period t-1
    - $F_t$ : forecast for period t
  - Spreadsheet: B3: = A2; Copy down

# 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
- Assumes average is a good estimator of future

$$F_{t+1} = \frac{A_t + A_{t-1} + A_{t-2} + \ldots + A_{t-n+1}}{n}$$

$F_{t+1}$ = Forecast for the upcoming period, t+1
$n$ = Number of periods to be averaged
$A_t$ = Actual occurrence in period t

# Simple Moving Average

$$F_{t+1} = \frac{A_t + A_{t-1} + A_{t-2} + \ldots + A_{t-n+1}}{n}$$

You're manager in Amazon's electronics department. You want to forecast ipod sales for months 4-6 using a 3-period moving average.

| Month | Sales (000) |
|---|---|
| 1 | 4 |
| 2 | 6 |
| 3 | 5 |
| 4 | ? |
| 5 | ? |
| 6 | ? |

# Simple Moving Average

$$F_{t+1} = \frac{A_t + A_{t-1} + A_{t-2} + \ldots + A_{t-n+1}}{n}$$

You're manager in Amazon's electronics department. You want to forecast ipod sales for months 4-6 using a 3-period moving average.

| Month | Sales (000) | Moving Average (n=3) |
|---|---|---|
| 1 | 4 | NA |
| 2 | 6 | NA |
| 3 | 5 | NA |
| 4 | ? | (4+6+5)/3=5 |
| 5 | ? | |
| 6 | ? | |

# Simple Moving Average

## What if ipod sales were actually 3 in month 4

| Month | Sales (000) | Moving Average (n=3) |
|---|---|---|
| 1 | 4 | NA |
| 2 | 6 | NA |
| 3 | 5 | NA |
| 4 | 3 | 5 |
| 5 | ? | |
| 6 | ? | |

# Simple Moving Average

## Forecast for Month 5?

| Month | Sales (000) | Moving Average (n=3) |
|---|---|---|
| 1 | 4 | NA |
| 2 | 6 | NA |
| 3 | 5 | NA |
| 4 | 3 | 5 |
| 5 | ? | (6+5+3)/3=4.667 |
| 6 | ? | |

# Simple Moving Average

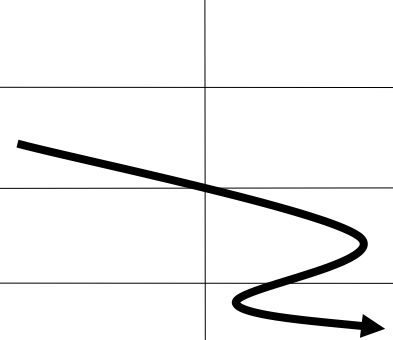
## Actual Demand for Month 5 = 7

| Month | Sales (000) | Moving Average (n=3) |
|---|---|---|
| 1 | 4 | NA |
| 2 | 6 | NA |
| 3 | 5 | NA |
| 4 | 3 | 5 |
| 5 | ?7 | 4.667 |
| 6 | ? | |

# Simple Moving Average

## Forecast for Month 6?

| Month | Sales (000) | Moving Average (n=3) |
|---|---|---|
| 1 | 4 | NA |
| 2 | 6 | NA |
| 3 | 5 | NA |
| 4 | 3 | 5 |
| 5 | 7 | 4.667 |
| 6 | ? | (5+3+7)/3=5 |



# Moving Average Method - Example

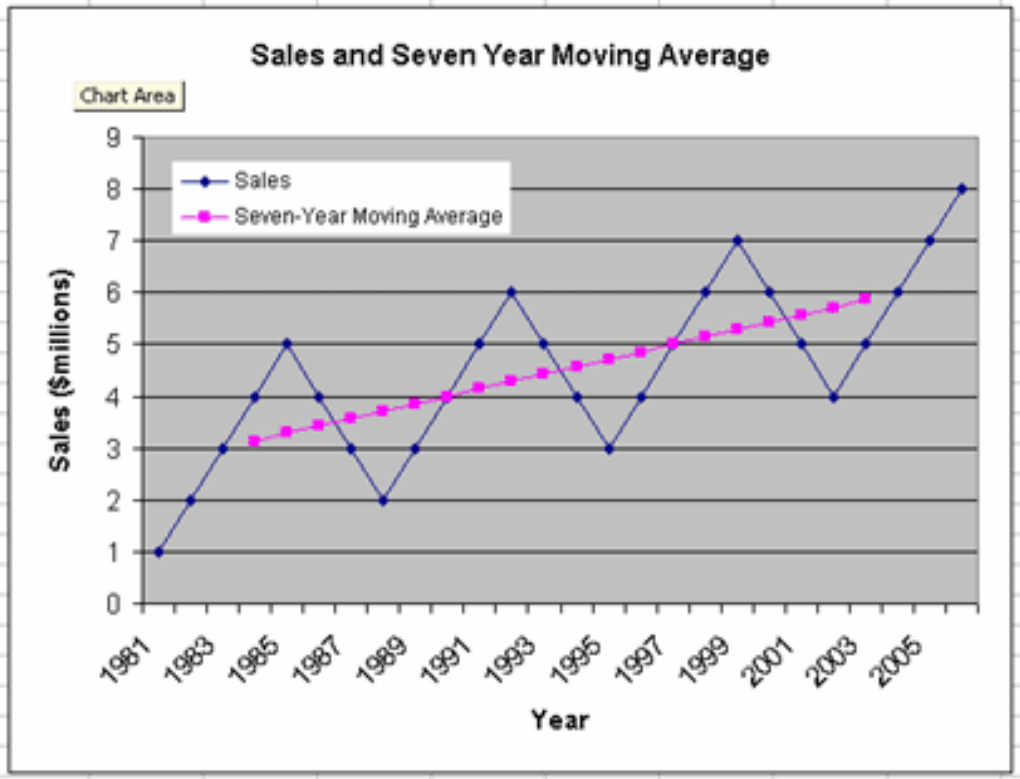
Microsoft Excel - Ch16 - Forecasting

File Edit View Insert Format Tools Data Window Help

Arial 10

C5 =AVERAGE(B2:B8)

| | A | B | C |
|---|---|---|---|
| 1 | Year | Sales ($M) | MA7 |
| 2 | 1981 | 1 | |
| 3 | 1982 | 2 | |
| 4 | 1983 | 3 | |
| 5 | 1984 | 4 | 3.143 |
| 6 | 1985 | 5 | 3.286 |
| 7 | 1986 | 4 | 3.429 |
| 8 | 1987 | 3 | 3.571 |
| 9 | 1988 | 2 | 3.714 |
| 10 | 1989 | 3 | 3.857 |
| 11 | 1990 | 4 | 4.000 |
| 12 | 1991 | 5 | 4.143 |
| 13 | 1992 | 6 | 4.286 |
| 14 | 1993 | 5 | 4.429 |
| 15 | 1994 | 4 | 4.571 |
| 16 | 1995 | 3 | 4.714 |
| 17 | 1996 | 4 | 4.857 |
| 18 | 1997 | 5 | 5.000 |
| 19 | 1998 | 6 | 5.143 |
| 20 | 1999 | 7 | 5.286 |
| 21 | 2000 | 6 | 5.429 |
| 22 | 2001 | 5 | 5.571 |
| 23 | 2002 | 4 | 5.714 |
| 24 | 2003 | 5 | 5.857 |
| 25 | 2004 | 6 | |
| 26 | 2005 | 7 | |
| 27 | 2006 | 8 | |



# Three-year and Five-Year Moving Averages

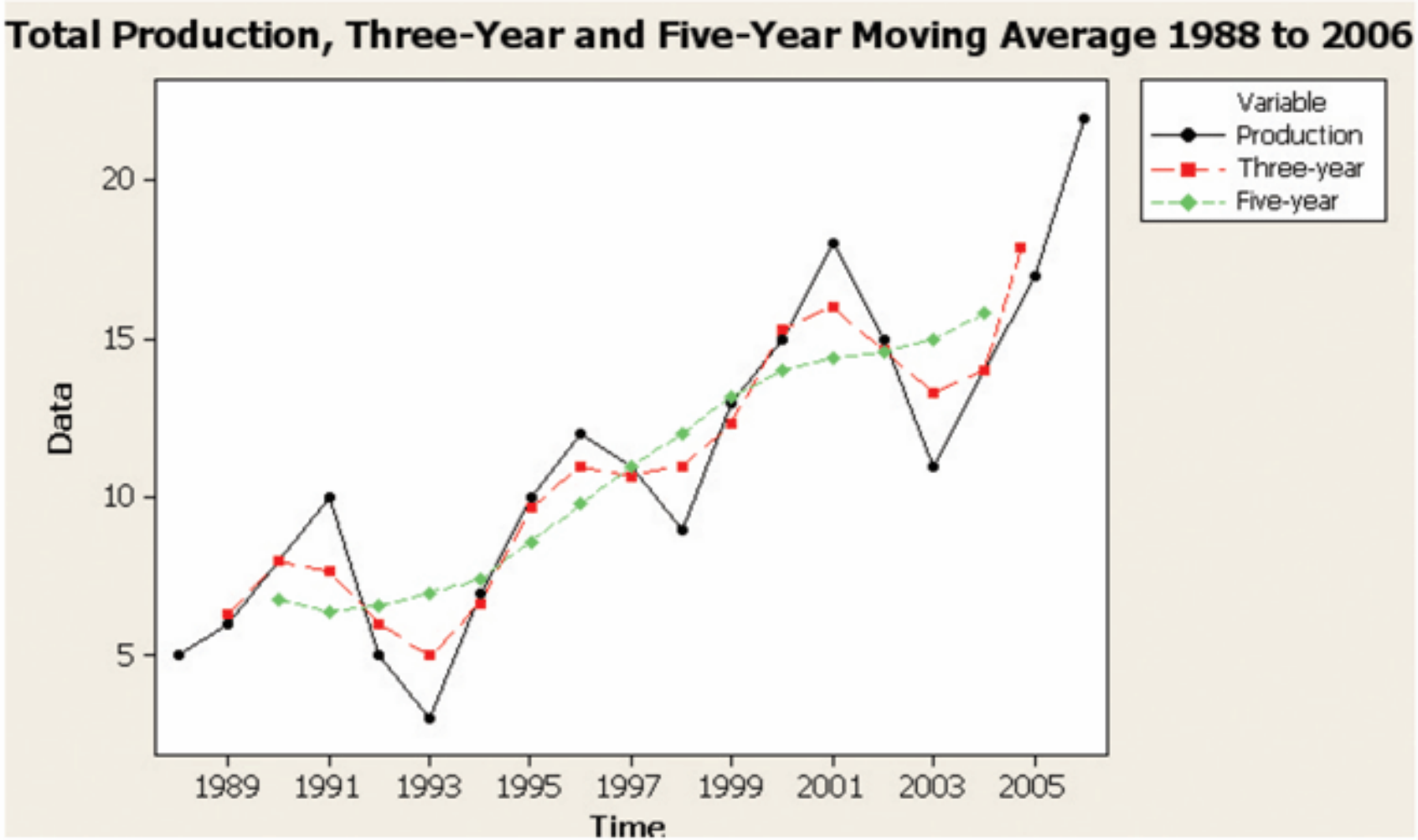



**CHART 16–4** A Three-Year and Five-Year Moving Average 1988 to 2006

# Weighted Moving Average

- A simple moving average assigns the same weight to each observation in averaging
- Weighted moving average assigns different weights to each observation
- Most recent observation receives the most weight, and the weight decreases for older data values
- In either case, the sum of the weights = 1

# Weighted Moving Average: 3/6, 2/6, 1/6

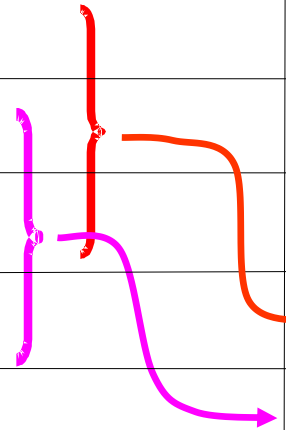
$$F_{t+1} = w_1 A_t + w_2 A_{t-1} + w_3 A_{t-2} + \ldots + w_n A_{t-n+1}$$

| Month | Sales (000) | Weighted Moving Average |
|---|---|---|
| 1 | 4 | NA |
| 2 | 6 | NA |
| 3 | 5 | NA |
| 4 | ? | 31/6 = 5.167 |
| 5 | ? | |
| 6 | ? | |

# Weighted Moving Average: 3/6, 2/6, 1/6

$$F_{t+1} = w_1 A_t + w_2 A_{t-1} + w_3 A_{t-2} + \ldots + w_n A_{t-n+1}$$

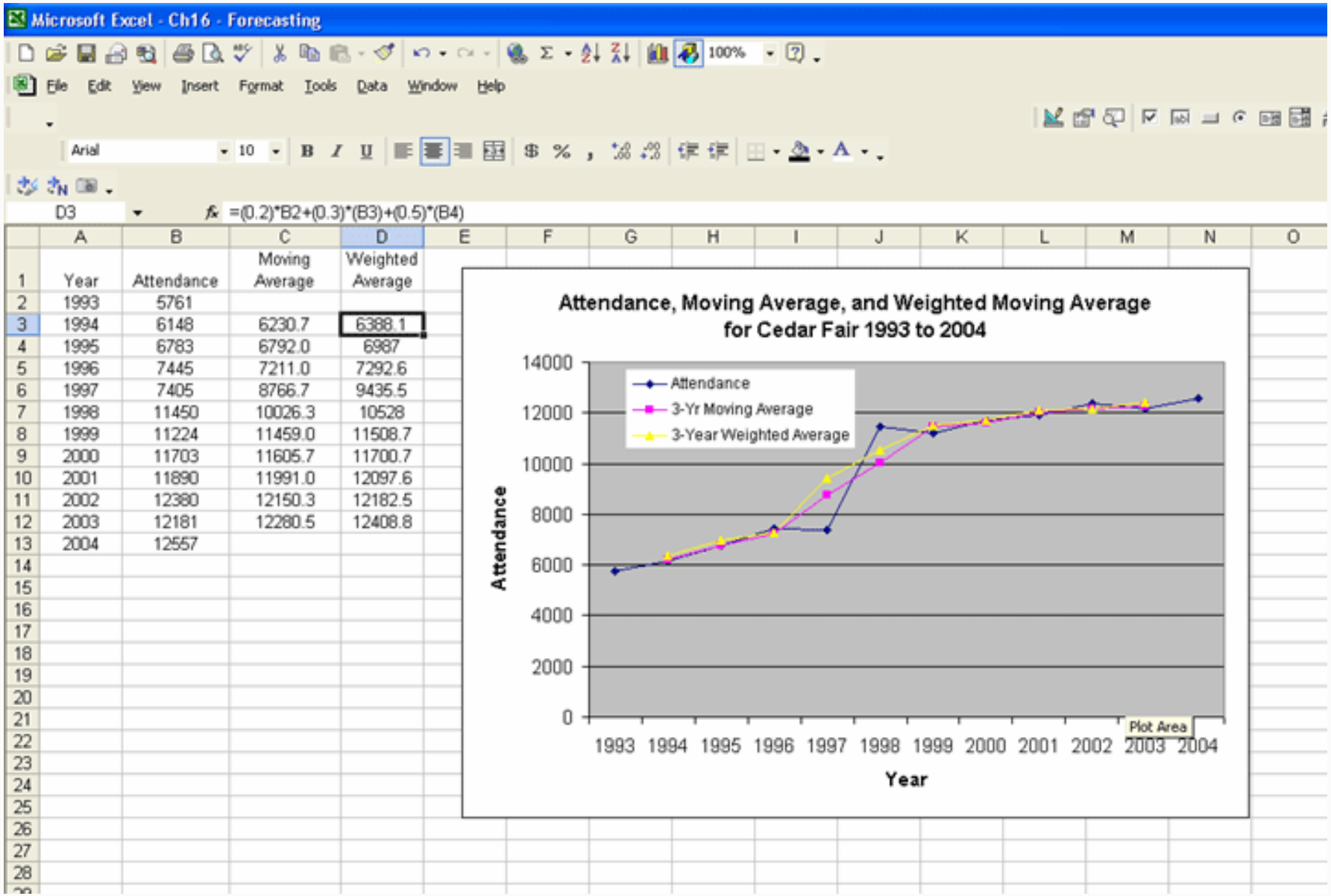
| Month | Sales (000) | Weighted Moving Average |
|---|---|---|
| 1 | 4 | NA |
| 2 | 6 | NA |
| 3 | 5 | NA |
| 4 | **3** | 31/6 = 5.167 |
| 5 | **7** | 25/6 = 4.167 |
| 6 | | 32/6 = 5.333 |



# Weighted Moving Average - Example

| Year | Attendance (000) | Weighted Moving Average | Found by |
|---|---|---|---|
| 1993 | 5,761 | | |
| 1994 | 6,148 | 6,388 | .2(5,761) + .3(6,148) + .5(6,783) |
| 1995 | 6,783 | 6,987 | .2(6,148) + .3(6,783) + .5(7,445) |
| 1996 | 7,445 | 7,293 | .2(6,783) + .3(7,445) + .5(7,405) |
| 1997 | 7,405 | 9,436 | .2(7,445) + .3(7,405) + .5(11,450) |
| 1998 | 11,450 | 10,528 | .2(7,405) + .3(11,450) + .5(11,224) |
| 1999 | 11,224 | 11,509 | .2(11,450) + .3(11,224) + .5(11,703) |
| 2000 | 11,703 | 11,701 | .2(11,224) + .3(11,703) + .5(11,890) |
| 2001 | 11,890 | 12,098 | .2(11,703) + .3(11,890) + .5(12,380) |
| 2002 | 12,380 | 12,183 | .2(11,890) + .3(12,380) + .5(12,181) |
| 2003 | 12,181 | 12,409 | .2(12,380) + .3(12,181) + .5(12,557) |
| 2004 | 12,557 | | |

# Weighed Moving Average – An Example




| | Year | Attendance | Moving Average | Weighted Average |
|---|---|---|---|---|
| 2 | 1993 | 5761 | | |
| 3 | 1994 | 6148 | 6230.7 | 6388.1 |
| 4 | 1995 | 6783 | 6792.0 | 6987 |
| 5 | 1996 | 7445 | 7211.0 | 7292.6 |
| 6 | 1997 | 7405 | 8766.7 | 9435.5 |
| 7 | 1998 | 11450 | 10026.3 | 10528 |
| 8 | 1999 | 11224 | 11459.0 | 11508.7 |
| 9 | 2000 | 11703 | 11605.7 | 11700.7 |
| 10 | 2001 | 11890 | 11991.0 | 12097.6 |
| 11 | 2002 | 12380 | 12150.3 | 12182.5 |
| 12 | 2003 | 12181 | 12280.5 | 12408.8 |
| 13 | 2004 | 12557 | | |

# Exponential Smoothing

- Concept is *simple*!
  - Make a forecast, *any* forecast
  - Compare it to the actual
  - Next forecast is
    - *Previous* forecast plus an adjustment
    - Adjustment is fraction of previous forecast error
  - Essentially
    - Not really forecast as a function of *time*
    - Instead, forecast as a function of *previous actual and forecasted value*

# Exponential Smoothing

- Assumes the most recent observations have the highest predictive value
  - gives more weight to recent time periods

$$F_{t+1} = F_t + \alpha(A_t - F_t)$$

$e_t$

$F_{t+1}$ = Forecast value for time $t+1$

$A_t$ = Actual value at time $t$

$\alpha$ = Smoothing constant

**Need initial forecast $F_t$ to start.**

# Exponential Smoothing – Example 1

$$F_{t+1} = F_t + \alpha(A_t - F_t)$$

| i | Ai |
|---|---|
| Week | Demand |
| 1 | 820 |
| 2 | 775 |
| 3 | 680 |
| 4 | 655 |
| 5 | 750 |
| 6 | 802 |
| 7 | 798 |
| 8 | 689 |
| 9 | 775 |
| 10 | |

**Given the weekly demand data what are the exponential smoothing forecasts for periods 2-10 using $\alpha$=0.10?**

**Assume $F_1=D_1$**

# Exponential Smoothing – Example 1

$$F_{t+1} = F_t + \alpha(A_t - F_t)$$

| i | Ai | Fi |
|---|---|---|
| Week | Demand | $\alpha = 0.1$ |
| 1 | 820 | 820.00 |
| 2 | 775 | |
| 3 | 680 | |
| 4 | 655 | |
| 5 | 750 | |
| 6 | 802 | |
| 7 | 798 | |
| 8 | 689 | |
| 9 | 775 | |
| 10 | | |

$F_2 = F_1 + \alpha(A_1 - F_1) = 820 + .1(820 - 820)$

**$= 820$**

# Exponential Smoothing – Example 1

$$F_{t+1} = F_t + \alpha(A_t - F_t)$$

| i | Ai | Fi |
|---|---|---|
| Week | Demand | $\alpha = 0.1$ |
| 1 | 820 | 820.00 |
| 2 | 775 | 820.00 |
| 3 | 680 | |
| 4 | 655 | |
| 5 | 750 | |
| 6 | 802 | |
| 7 | 798 | |
| 8 | 689 | |
| 9 | 775 | |
| 10 | | |

$F_3 = F_2 + \alpha(A_2 - F_2) = 820 + .1(775 - 820)$

**$= 815.5$**

# Exponential Smoothing – Example 1

$$F_{t+1} = F_t + \alpha(A_t - F_t)$$

| i | Ai | Fi |
|---|---|---|
| Week | Demand | $\alpha = 0.1$ |
| 1 | 820 | 820.00 |
| 2 | 775 | 820.00 |
| 3 | 680 | 815.50 |
| 4 | 655 | |
| 5 | 750 | |
| 6 | 802 | |
| 7 | 798 | |
| 8 | 689 | |
| 9 | 775 | |
| 10 | | |

**This process continues through week 10**

# Exponential Smoothing – Example 1

$$F_{t+1} = F_t + \alpha(A_t - F_t)$$

| i | Ai | Fi | |
|---|---|---|---|
| Week | Demand | $\alpha$ = *0.1* | $\alpha$ = *0.6* |
| 1 | 820 | 820.00 | 820.00 |
| 2 | 775 | 820.00 | 820.00 |
| 3 | 680 | 815.50 | 793.00 |
| 4 | 655 | 801.95 | 725.20 |
| 5 | 750 | 787.26 | 683.08 |
| 6 | 802 | 783.53 | 723.23 |
| 7 | 798 | 785.38 | 770.49 |
| 8 | 689 | 786.64 | 787.00 |
| 9 | 775 | 776.88 | 728.20 |
| 10 | | 776.69 | 756.28 |

**What if the $\alpha$ constant equals 0.6**

# Exponential Smoothing

- How to choose $\alpha$
  - depends on the emphasis you want to place on the most recent data
- Increasing $\alpha$ makes forecast more sensitive to recent data

# Forecast Effects of Smoothing Constant $\alpha$

$$F_{t+1} = F_t + \alpha (A_t - F_t)$$

or

$$F_{t+1} = \underbrace{\alpha}_{w_1} A_t + \underbrace{\alpha(1-\alpha)}_{w_2} A_{t-1} + \underbrace{\alpha(1-\alpha)^2}_{w_3} A_{t-2} + \ldots$$

| $\alpha$= | Weights | | |
|---|---|---|---|
| | Prior Period | 2 periods ago | 3 periods ago |
| | $\alpha$ | $\alpha(1-\alpha)$ | $\alpha(1-\alpha)^2$ |
| $\alpha$= 0.10 | 10% | 9% | 8.1% |
| $\alpha$= 0.90 | 90% | 9% | 0.9% |

# Simple Exponential Smoothing

## Properties of Simple Exponential Smoothing

- Widely used and successful model
- Requires very little data
- Larger $\alpha$, more responsive forecast; Smaller $\alpha$, smoother forecast

  "best" $\alpha$ can be found by a solver package
- Suitable for relatively stable time series

# Time Series Components

## Trend

- persistent upward or downward pattern in a time series

## Seasonal

- Variation dependent on the time of year
- Each year shows same pattern

## Cyclical

- up & down movement repeating over long time frame
- Each year does not show same pattern

## Noise or random fluctuations

- follow no specific pattern
- short duration and non-repeating

# Linear Trend Plot

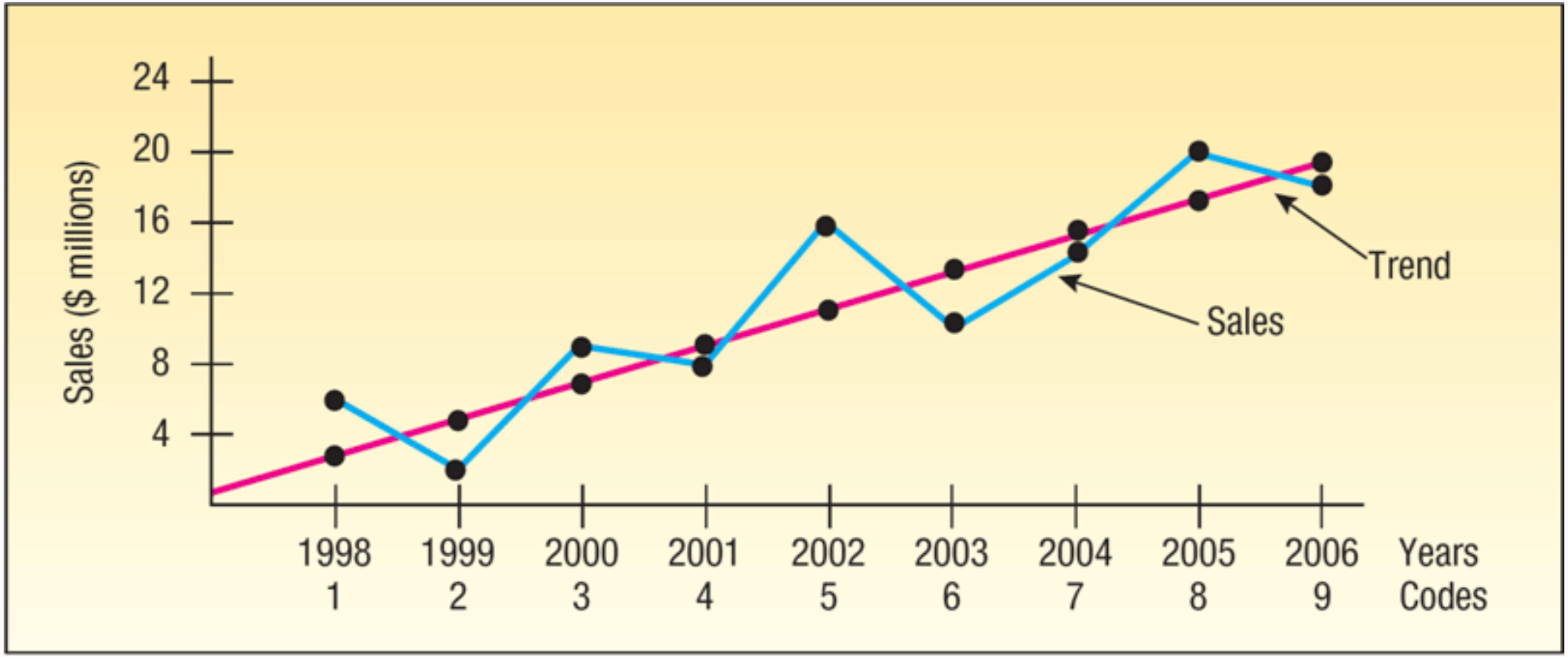
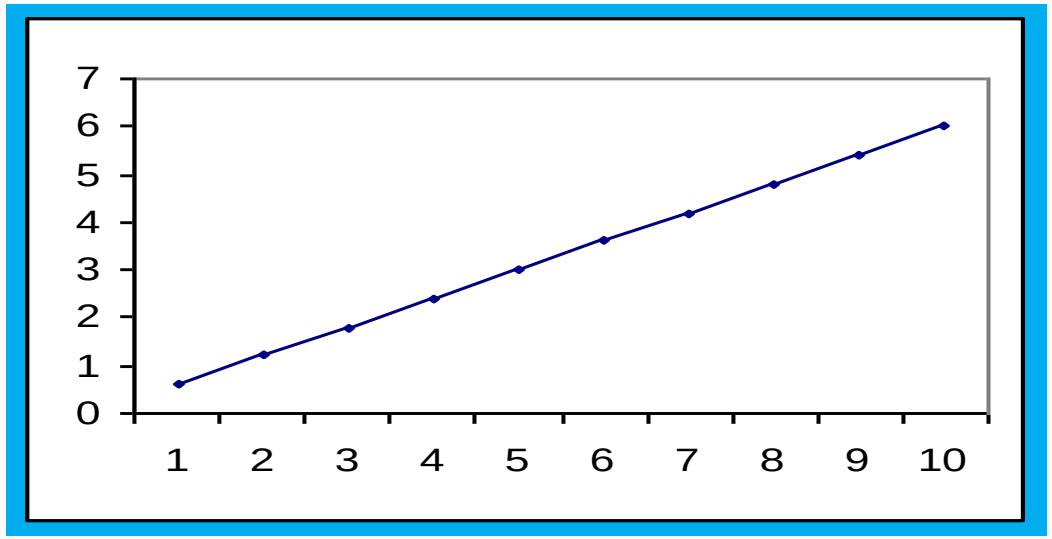



**CHART 16–5** A Straight Line Fitted to Sales Data

# Trend Model

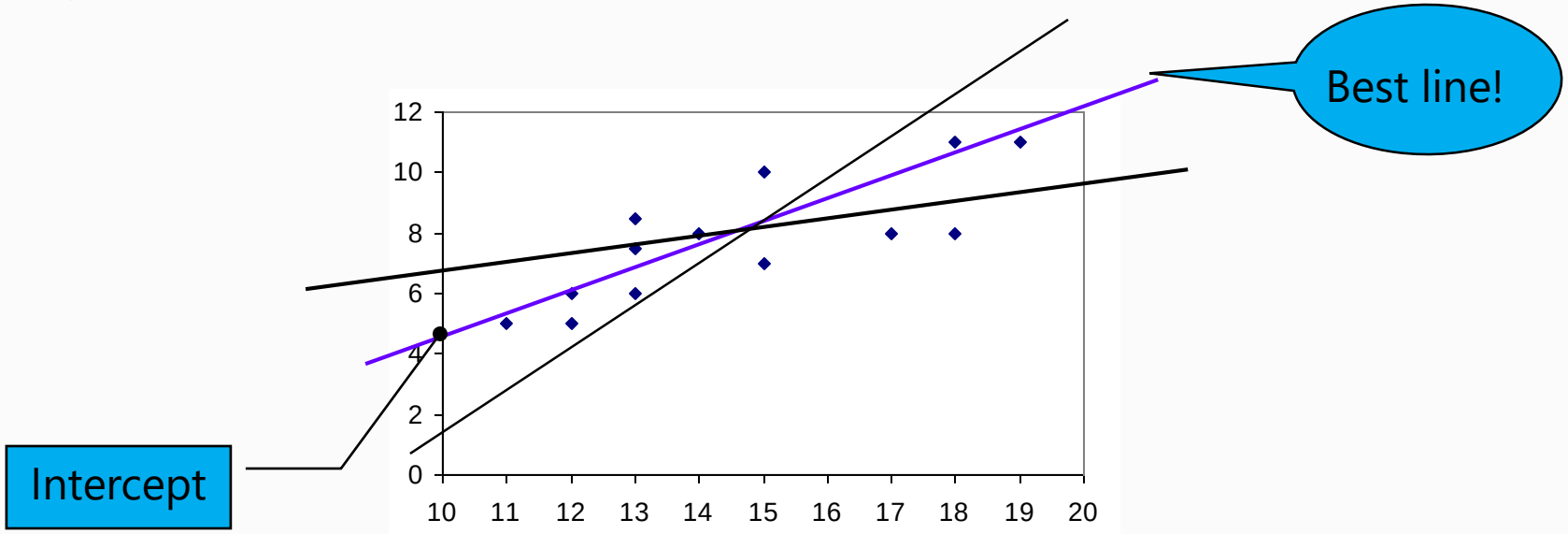
- Curve fitting method used for time series data (also called time series regression model)
- Useful when the time series has a clear trend
- Can not capture seasonal patterns
- Linear Trend Model: $Y_t = a + bt$
  - $t$ is time index for each period, $t$ = 1, 2, 3,…



# Pattern-based forecasting - Trend

Regression – Recall Independent Variable X, which is now time variable – e.g., days, months, quarters, years etc.

- *Find a straight line that fits the data best.*

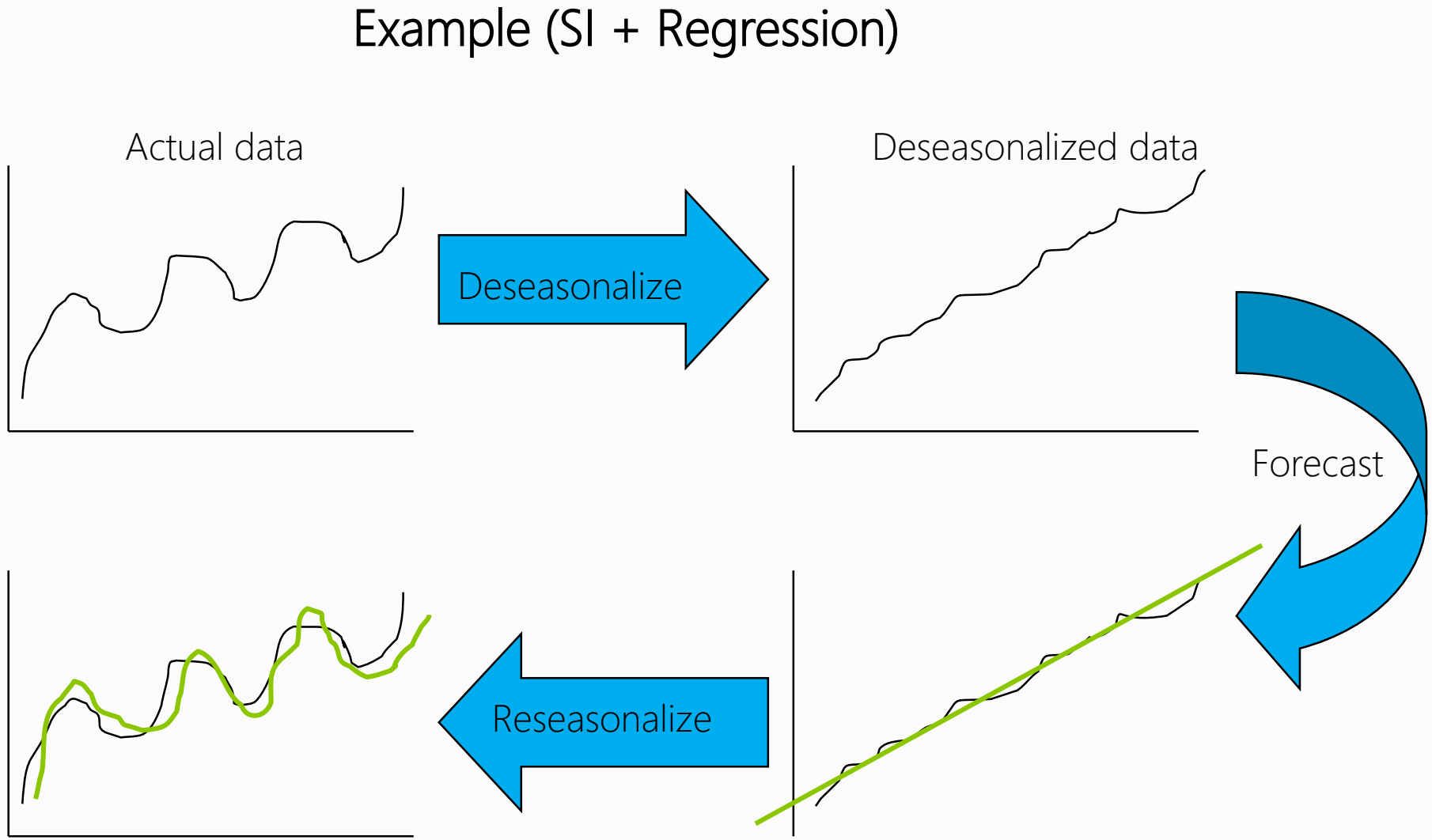



- *y = Intercept + slope * x (e.g. $y = b_0 + b_1x$)*
- *Slope = change in y / change in x*

# Pattern-based forecasting – Seasonal

- Once data turn out to be seasonal, *deseasonalize* the data.
  - The methods we have learned (Heuristic methods and Regression) is not suitable for data that has pronounced fluctuations.
- Make forecast based on the deseasonalized data
- *Reseasonalize* the forecast
  - Good forecast should mimic reality. Therefore, it is needed to give seasonality back.

# Pattern-based forecasting – Seasonal



# Linear Trend – Using the Least Squares Method

- Use the least squares method in Simple Linear Regression to find the best linear relationship between 2 variables
- Code time (*t*) and use it as the independent variable
- E.g. let *t* be 1 for the first year, 2 for the second, and so on (if data are annual)

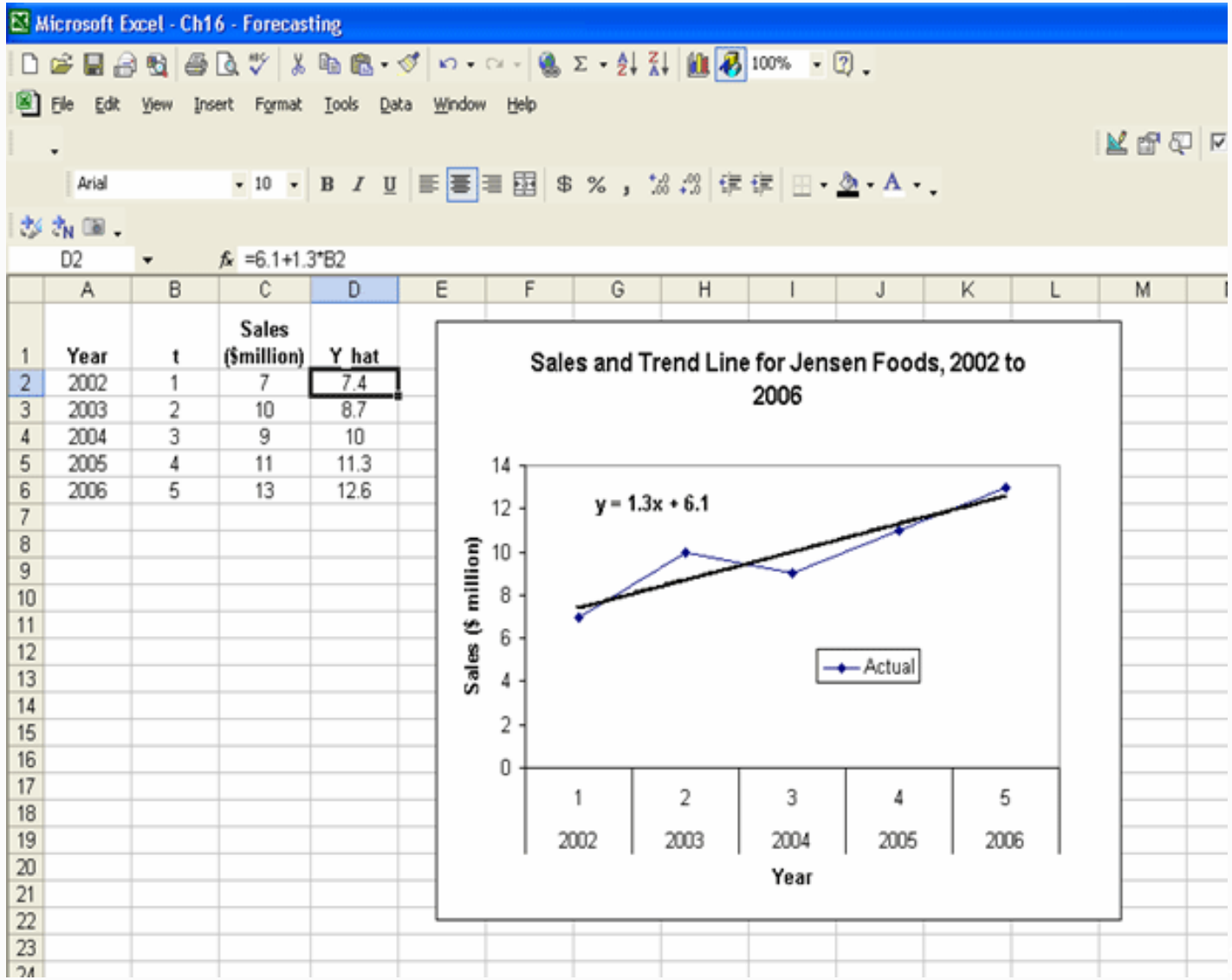
# Linear Trend – Using the Least Squares Method: An Example

The sales of Jensen Foods, a small grocery chain located in southwest Texas, since 2002 are:

| Year | Sales ($ mil.) |
|---|---|
| 2002 | 7 |
| 2003 | 10 |
| 2004 | 9 |
| 2005 | 11 |
| 2006 | 13 |

| Year | *t* | Sales ($ mil.) |
|---|---|---|
| 2002 | 1 | 7 |
| 2003 | 2 | 10 |
| 2004 | 3 | 9 |
| 2005 | 4 | 11 |
| 2006 | 5 | 13 |

# Linear Trend – Using the Least Squares Method
## Example Using Excel




| | A | B | C | D |
|---|---|---|---|---|
| 1 | Year | t | Sales ($million) | Y_hat |
| 2 | 2002 | 1 | 7 | 7.4 |
| 3 | 2003 | 2 | 10 | 8.7 |
| 4 | 2004 | 3 | 9 | 10 |
| 5 | 2005 | 4 | 11 | 11.3 |
| 6 | 2006 | 5 | 13 | 12.6 |

# Nonlinear Trends

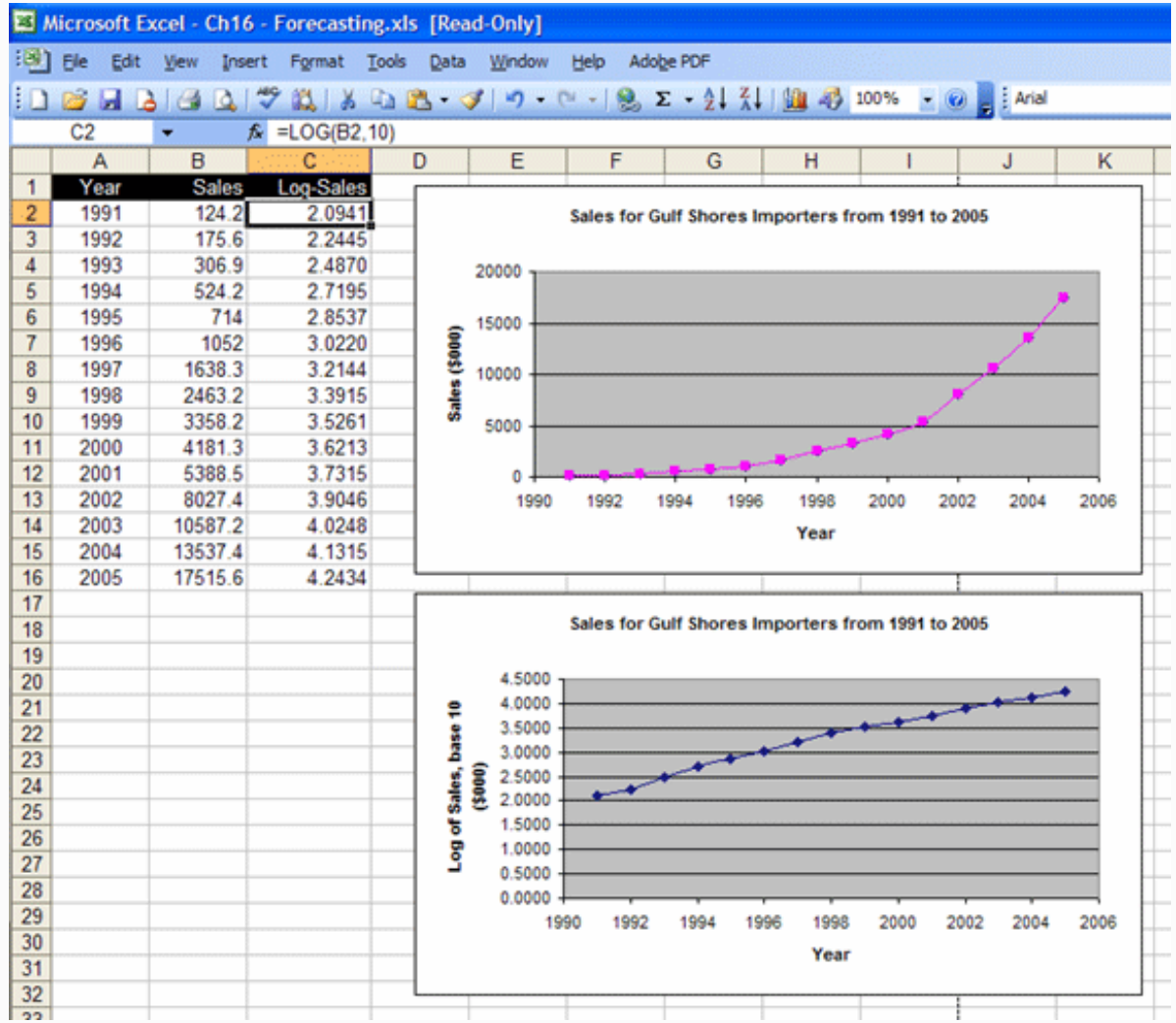
- A linear trend equation is used when the data are increasing (or decreasing) by equal amounts
- A nonlinear trend equation is used when the data are increasing (or decreasing) by increasing amounts over time
- When data increase (or decrease) by equal *percents or proportions* plot will show curvilinear pattern

# Log Trend Equation – Gulf Shores Importers Example

- Top graph is plot of the original data
- Bottom graph is the log base 10 of the original data which now is linear

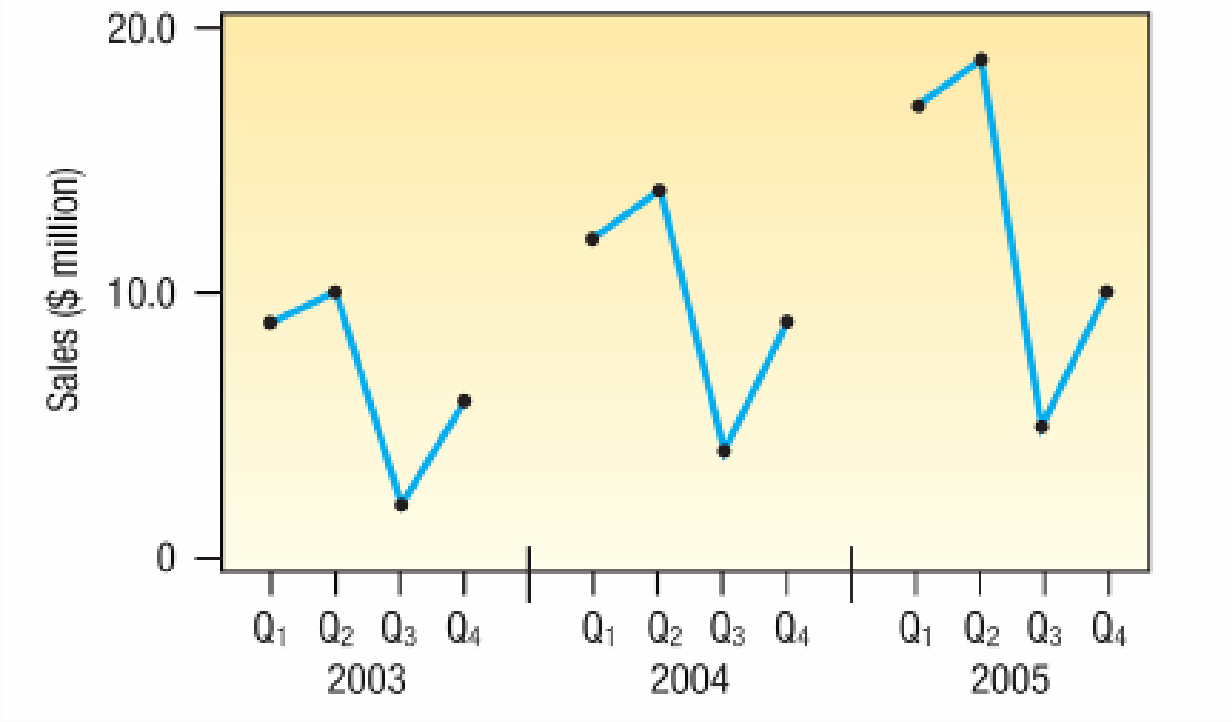
  (Excel function:

  =log(x) or log(x,10)
- Using Data Analysis in Excel, generate the linear equation



# 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



# Pattern-based forecasting – Seasonal

- Deseasonalization
  - *Deseasonalized data = Actual / SI*
- Reseasonalization
  - *Reseasonalized forecast*

    *= deseasonalized forecast * SI*

# Seasonal Index

What's an *index*?

- Ratio
- SI = ratio between actual and average demand

Suppose

- SI for quarter demand is 1.20
  - What's that mean?
  - Use it to forecast demand for next fall

So, *where* did the 1.20 come from?!

# Calculating Seasonal Indices

- Quick and dirty method of calculating SI
  - For each *year*, calculate average demand
  - Divide each demand by its yearly average
    - This creates a ratio and hence a *raw index*
    - For each *quarter*, there will be as many raw indices as there are years

  - Average the raw indices for each of the quarters
  - The result will be *four* values, one SI per quarter

# Seasonal Index – An Example

The table below shows the quarterly sales for Toys International for the years 2001 through 2006. The sales are reported in millions of dollars. Determine a quarterly seasonal index using the ratio-to-moving-average method.

| Year | Winter | Spring | Summer | Fall |
|---|---|---|---|---|
| 2001 | 6.7 | 4.6 | 10.0 | 12.7 |
| 2002 | 6.5 | 4.6 | 9.8 | 13.6 |
| 2003 | 6.9 | 5.0 | 10.4 | 14.1 |
| 2004 | 7.0 | 5.5 | 10.8 | 15.0 |
| 2005 | 7.1 | 5.7 | 11.1 | 14.5 |
| 2006 | 8.0 | 6.2 | 11.4 | 14.9 |

- Step (1) – Organize time series data in column form
- Step (2) Compute the 4-quarter moving totals
- Step (3) Compute the 4-quarter moving averages
- Step (4) Compute the centered moving averages by getting the average of two 4-quarter moving averages
- Step (5) Compute ratio by dividing actual sales by the centered moving averages

| Year | Quarter | (1) Sales ($ millions) | (2) Four-Quarter Total | (3) Four-Quarter Moving Average | (4) Centered Moving Average | (5) Specific Seasonal |
|---|---|---|---|---|---|---|
| 2001 | Winter | 6.7 | | | | |
| | Spring | 4.6 | | | | |
| | | | 34.0 | 8.500 | | |
| | Summer | 10.0 | | | 8.475 | 1.180 |
| | | | 33.8 | 8.450 | | |
| | Fall | 12.7 | | | 8.450 | 1.503 |
| | | | 33.8 | 8.450 | | |
| 2002 | Winter | 6.5 | | | 8.425 | 0.772 |
| | | | 33.6 | 8.400 | | |
| | Spring | 4.6 | | | 8.513 | 0.540 |
| | | | 34.5 | 8.625 | | |
| | Summer | 9.8 | | | 8.675 | 1.130 |
| | | | 34.9 | 8.725 | | |
| | Fall | 13.6 | | | 8.775 | 1.550 |
| | | | 35.3 | 8.825 | | |
| 2003 | Winter | 6.9 | | | 8.900 | 0.775 |
| | | | 35.9 | 8.975 | | |
| | Spring | 5.0 | | | 9.038 | 0.553 |
| | | | 36.4 | 9.100 | | |
| | Summer | 10.4 | | | 9.113 | 1.141 |
| | | | 36.5 | 9.125 | | |
| | Fall | 14.1 | | | 9.188 | 1.535 |
| | | | 37.0 | 9.250 | | |
| 2004 | Winter | 7.0 | | | 9.300 | 0.753 |
| | | | 37.4 | 9.350 | | |
| | Spring | 5.5 | | | 9.463 | 0.581 |
| | | | 38.3 | 9.575 | | |
| | Summer | 10.8 | | | 9.588 | 1.126 |
| | | | 38.4 | 9.600 | | |
| | Fall | 15.0 | | | 9.625 | 1.558 |
| | | | 38.6 | 9.650 | | |
| 2005 | Winter | 7.1 | | | 9.688 | 0.733 |
| | | | 38.9 | 9.725 | | |
| | Spring | 5.7 | | | 9.663 | 0.590 |
| | | | 38.4 | 9.600 | | |
| | Summer | 11.1 | | | 9.713 | 1.143 |
| | | | 39.3 | 9.825 | | |
| | Fall | 14.5 | | | 9.888 | 1.466 |
| | | | 39.8 | 9.950 | | |
| 2006 | Winter | 8.0 | | | 9.888 | 0.801 |
| | | | 40.1 | 10.025 | | |
| | Spring | 6.2 | | | 10.075 | 0.615 |
| | | | 40.5 | 10.125 | | |
| | Summer | 11.4 | | | | |
| | Fall | 14.9 | | | | |

# Seasonal Index – An Example

| Year | Winter | Spring | Summer | Fall | |
|---|---|---|---|---|---|
| 2001 | | | 1.180 | 1.503 | |
| 2002 | 0.772 | 0.540 | 1.130 | 1.550 | |
| 2003 | 0.775 | 0.553 | 1.141 | 1.535 | |
| 2004 | 0.753 | 0.581 | 1.126 | 1.558 | |
| 2005 | 0.733 | 0.590 | 1.143 | 1.466 | |
| 2006 | 0.801 | 0.615 | | | |
| Total | 3.834 | 2.879 | 5.720 | 7.612 | |
| Mean | 0.767 | 0.576 | 1.144 | 1.522 | 4.009 |
| Adjusted | 0.765 | 0.575 | 1.141 | 1.519 | 4.000 |
| Index | 76.5 | 57.5 | 114.1 | 151.9 | |

**CORRECTION FACTOR FOR ADJUSTING QUARTERLY MEANS**

$$\text{Correction factor} = \frac{4.00}{\text{Total of four means}}$$

$$\text{Correction factor} = \frac{4.00}{4.009} = 0.997755$$

# Actual versus Deseasonalized Sales for Toys International

Deseasonalized Sales = Sales / Seasonal Index

| Year | Quarter | (1) Sales | (2) Seasonal Index | (3) Deseasonalized Sales |
|---|---|---|---|---|
| 2001 | Winter | 6.7 | 0.765 | 8.76 |
| | Spring | 4.6 | 0.575 | 8.00 |
| | Summer | 10.0 | 1.141 | 8.76 |
| | Fall | 12.7 | 1.519 | 8.36 |
| 2002 | Winter | 6.5 | 0.765 | 8.50 |
| | Spring | 4.6 | 0.575 | 8.00 |
| | Summer | 9.8 | 1.141 | 8.59 |
| | Fall | 13.6 | 1.519 | 8.95 |
| 2003 | Winter | 6.9 | 0.765 | 9.02 |
| | Spring | 5.0 | 0.575 | 8.70 |
| | Summer | 10.4 | 1.141 | 9.11 |
| | Fall | 14.1 | 1.519 | 9.28 |
| 2004 | Winter | 7.0 | 0.765 | 9.15 |
| | Spring | 5.5 | 0.575 | 9.57 |
| | Summer | 10.8 | 1.141 | 9.47 |
| | Fall | 15.0 | 1.519 | 9.87 |
| 2005 | Winter | 7.1 | 0.765 | 9.28 |
| | Spring | 5.7 | 0.575 | 9.91 |
| | Summer | 11.1 | 1.141 | 9.73 |
| | Fall | 14.5 | 1.519 | 9.55 |
| 2006 | Winter | 8.0 | 0.765 | 10.46 |
| | Spring | 6.2 | 0.575 | 10.79 |
| | Summer | 11.4 | 1.141 | 9.99 |
| | Fall | 14.9 | 1.519 | 9.81 |

# Classical Decomposition

- Start by calculating seasonal indices
- Then, *deseasonalize* the demand
  - Divide actual demand values by their SI values
    $y' = y / SI$
  - Results in transformed data (new time series)
  - Seasonal effect removed
- Forecast
  - Regression if deseasonalized data is trendy
  - Heuristics methods if deseasonalized data is stationary
- Reseasonalize with SI

*…but what is a good forecast?*

# A Good Forecast

Has a small error

- Error = Demand - Forecast

# Measures of Forecast Error

a. MAD = Mean Absolute Deviation

$$MAD = \frac{\sum_{t=1}^{n} |A_t - F_t|}{n}$$

($|A_t - F_t|$ is $e_t$)

b. **MSE = Mean Squared Error**

$$MSE = \frac{\sum_{t=1}^{n} (A_t - F_t)^2}{n}$$

c. RMSE = Root Mean Squared Error

$$RMSE = \sqrt{MSE}$$

- Ideal values =0 (i.e., no forecasting error)

# MAD Example

$$MAD = \frac{\sum_{t=1}^{n} |A_t - F_t|}{n} = \frac{40}{4} = 10$$

What is the MAD value given the forecast values in the table below?

| Month | Sales ($A_t$) | Forecast ($F_t$) | $\lvert A_t - F_t \rvert$ |
|---|---|---|---|
| 1 | 220 | n/a | |
| 2 | 250 | 255 | 5 |
| 3 | 210 | 205 | 5 |
| 4 | 300 | 320 | 20 |
| 5 | 325 | 315 | 10 |

$$\sum_{t=1}^{n} |A_t - F_t| = 40$$

# MSE/RMSE Example

$$MSE = \frac{\sum_{t=1}^{n}(A_t - F_t)^2}{n} = \frac{550}{4} = 137.5$$

**What is the MSE value?**

$$RMSE = \sqrt{137.5} = 11.73$$

| Month | Sales ($A_t$) | Forecast ($F_t$) | $\lvert A_t - F_t \rvert$ | $(A_t - F_t)^2$ |
|---|---|---|---|---|
| 1 | 220 | n/a | | |
| 2 | 250 | 255 | 5 | 25 |
| 3 | 210 | 205 | 5 | 25 |
| 4 | 300 | 320 | 20 | 400 |
| 5 | 325 | 315 | 10 | 100 |

$$\sum_{t=1}^{n}(A_t - F_t)^2 = 550$$

# Forecast Bias

- How can we tell if a forecast has a positive or negative bias?

- TS = Tracking Signal
  - Good tracking signal has low values

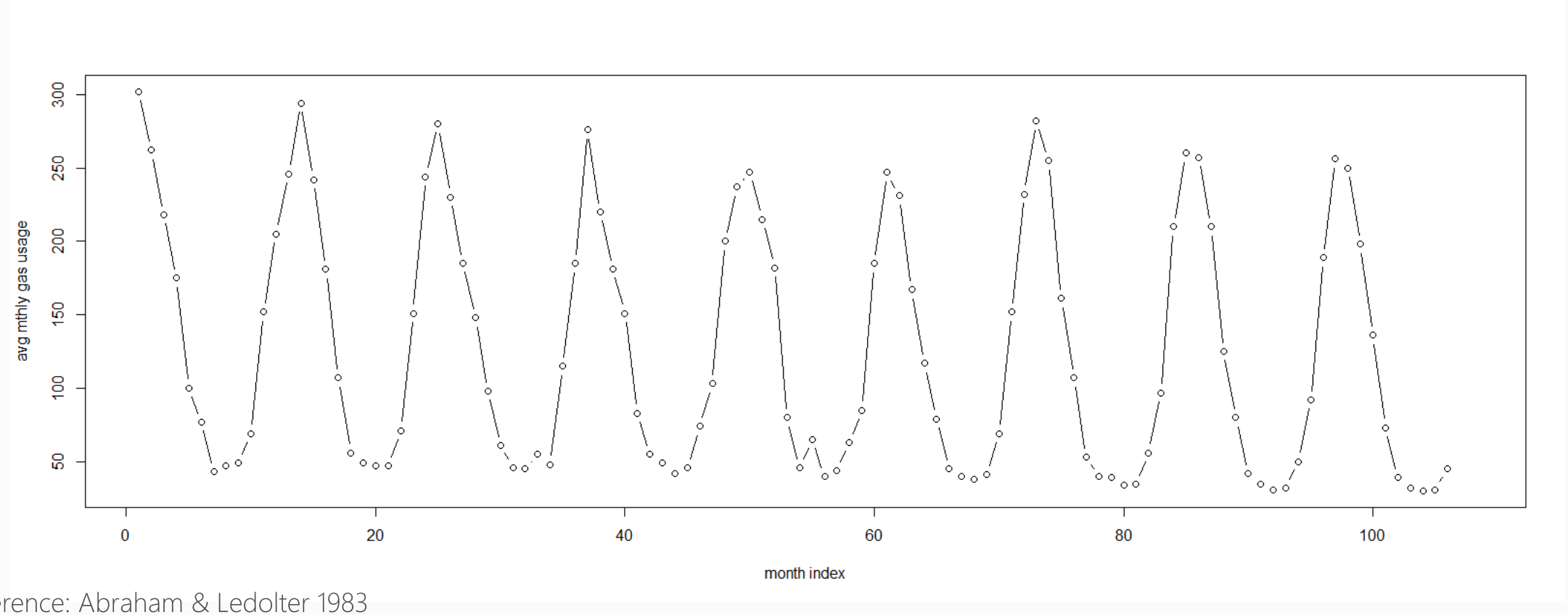
$$\text{TS} = \frac{\text{RSFE}}{\text{MAD}} = \frac{\sum_t (actual_t - forecast_t)}{\text{MAD}}$$

# Can you…

- describe general forecasting process?
- compare and contrast trend, seasonality and cyclicality?
- describe the forecasting method when data is stationary?
- describe the forecasting method when data shows trend?
- describe the forecasting method when data shows seasonality?

# In class exercise

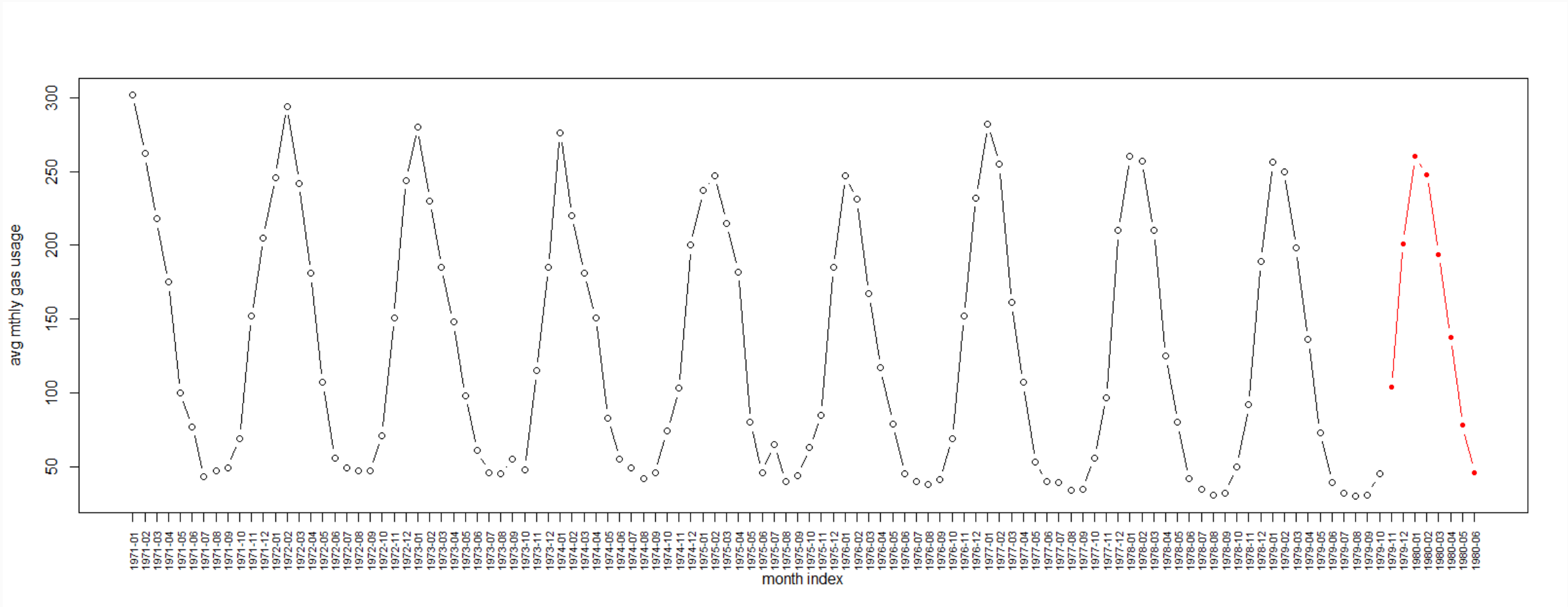
- An utility company in the Midwest region wished to forecast residential average monthly gas usage. It had monthly data from Jan 1971 to Oct 1979. In this exercise , forecast usage for Nov 1979 to Jun 1980.




Reference: Abraham & Ledolter 1983

# In class exercise (2)

- Using ARIMA with seasonal component, we arrived with a good model. "ARIMA(1,0,0)(0,1,1)12"



# In class exercise (3)

```
## Oct 12, 2015 W.Taam
#

## input data from PC
dd <- read.csv('C:/Users/whoever/wherever/monthly_avg_gas_usage_jan71_oct79.txt',
               header=F, sep='', strip.white=T )

## sanity check on the input
dd

## string out the data in a vector
dat <- c( t( as.matrix( dd ) ) )

## drop missing during read
dat <- dat[-c(107,108)]

#visualize the raw data as it is
plot(dat, type='b', ylab='avg mthly gas usage', xlab='month index')


## underfitting: fit a straight line regression to the gas usage
fit0 <- lm( dat~ c(1:106) )

## naive approach: find monthly usage by averaging over years of the same month
fit1 <- apply( dd, 2, mean, na.rm=T )


## find the ACF and PACF of the raw data to get a sense of the correlation over time
acf(dat[c(1:106)])
pacf(dat[c(1:106)])
  ## correlation shows sign of periodicity of 6
```

```
## ARIMA model with seasonality
## (1,0,0)(1,0,0)6
fit <- arima( dat, order=c(1,0,0),seasonal=list(order=c(1,0,0),period=6))
acf(fit$residuals[c(1:106)])
pacf(fit$residuals[c(1:106)]) ## residuals exhibit autocorrelations at periodic lags

## (1,0,0)(0,1,0)6
fit <- arima( dat, order=c(1,0,0),seasonal=list(order=c(0,1,0),period=6))
acf(fit$residuals[c(1:106)])
pacf(fit$residuals[c(1:106)]) ## residuals exhibit autocorrelations at periodic lags

## (0,0,1)(0,1,0)12
fit <- arima( dat, order=c(0,0,1),seasonal=list(order=c(0,1,0),period=12))
acf(fit$residuals[c(1:106)])
pacf(fit$residuals[c(1:106)]) ## residuals exhibit autocorrelations at periodic lags

## (0,0,1)(0,1,1)12
fit <- arima( dat, order=c(0,0,1),seasonal=list(order=c(0,1,1),period=12))
acf(fit$residuals[c(1:106)])
pacf(fit$residuals[c(1:106)])   ## residuals exhibit nearly zero autocorrelations at high lags

## (0,0,1)(1,1,0)12
fit <- arima( dat, order=c(0,0,1),seasonal=list(order=c(1,1,0),period=12))
acf(fit$residuals[c(1:106)])
pacf(fit$residuals[c(1:106)])   ## residuals exhibit nearly zero autocorrelations at high lags

## (1,0,0)(1,1,0)12
fit <- arima( dat, order=c(1,0,0),seasonal=list(order=c(1,1,0),period=12))
acf(fit$residuals[c(1:106)])
pacf(fit$residuals[c(1:106)])   ## residuals exhibit nearly zero autocorrelations at high lags

## (1,0,0)(0,1,1)12
fit <- arima( dat, order=c(1,0,0),seasonal=list(order=c(0,1,1),period=12))
acf(fit$residuals[c(1:106)])
pacf(fit$residuals[c(1:106)])   ## residuals exhibit nearly zero autocorrelations at high lags
```

# In class exercise (4)

```
## solely rank the models by Akaike information criteria AIC
AIC(
  arima( dat, order=c(1,0,0),seasonal=list(order=c(1,0,0),period=6)),
  arima( dat, order=c(1,0,0),seasonal=list(order=c(0,1,0),period=6)),
  arima( dat, order=c(0,0,1),seasonal=list(order=c(0,1,0),period=12)),
  arima( dat, order=c(0,0,1),seasonal=list(order=c(0,1,1),period=12)),
  arima( dat, order=c(0,0,1),seasonal=list(order=c(1,1,0),period=12)),
  arima( dat, order=c(1,0,0),seasonal=list(order=c(1,1,0),period=12)),
  arima( dat, order=c(1,0,0),seasonal=list(order=c(0,1,1),period=12))
)

## make month labels
tmp <- expand.grid( c(1971:1980), c('01','02','03','04','05','06','07','08','09','10','11','12'))
yyyymm <- sort( paste(tmp[,1],tmp[,2],sep='-') )

## show forecast from final model with the forecast
plot(dat, type='b', ylab='avg mthly gas usage', xlab='month', xlim=c(1,114), xaxt='n' )
lines( c(107:114), unlist(predict(fit,8)[1]), col='red', type='b', pch=20)
axis(1, at=c(1:114), labels=yyyymm[c(1:114)], las=2, cex.axis=0.7)
```

# Lecture 3 Outline, Oct 19th

| | |
|---|---|
| • Class Discussions | 15 minutes |
| • Forecasting, continued (2/2) | 45 minutes |
| • Weka Tutorial | 30 minutes |
| • Break | 10 minutes |
| • Decision Trees and Random Forests | 60 minutes |
| • Hands on, decision tree in Weka | 15 minutes |

# Data Science

## Deriving Knowledge from Data at Scale

*That's all for tonight….*

Oct 12th , 2015