MGS 3701 / MKT 3950: Data Mining

Chapter2: Overview of the Data Mining Process

Chungil Chae

温州肯恩大学
WENZHOU-KEAN UNIVERSITY

2022/01/01 (updated: 2022-02-13)

Chapter Objectives

- Overview of steps involved in data mining
  - from a clear goal definition to model deployment
- Issues related to data collection, cleaning, and preprocessing
- The notion of data partitioning
- The steps of model building

1.Introduction

The core of this course focuses on what has come to be called predictive analytics, the tasks of classification and prediction as well as pattern discovery, which have become key elements of a “business analytics” function in most large firms.

- Not covered in this course but important extent are two simpler database methods that are sometimes considered to be data mining techniques:
  - (1) OLAP (online analytical processing)
  - (2) SQL (structured query language).
- OLAP and SQL searches on databases are
  - descriptive in nature
  - based on business rules set by the user

Although SQL queries are often used to obtain the data in data mining, they do not involve statistical modeling or automated algorithmic methods.

2.Core Ideas in Data Mining

- **Classification**
  - to examine data where the classification is unknown or will occur in the future
- **Prediction**
  - to predict the value of a numerical variable (e.g., amount of purchase) rather than a class (e.g., purchaser or nonpurchaser).
- **Association Rules & Recommends**
  - to find such general associations patterns between items in large databases
  - “what goes with what.”
- **Predictive Analytics**
  - Classification, prediction, and to some extent, association rules and collaborative filtering constitute the analytical methods employed in predictive analytics.
- **Data & Dimension Reduction**
  - This process of consolidating a large number of records (or cases) into a smaller set is termed **data reduction**.
  - Reducing the number of variables is typically called **dimension reduction**
- **Data Exploration & Visualization**
  - Exploration is aimed at understanding the global landscape of the data, and detecting unusual values
  - Exploration by creating charts and dashboards is called Data Visualization or Visual Analytics.

Supervised and Unsupervised Learning

- Supervised learning
  - Goal: Predict a single “target” or “outcome” variable
  - Training data, where target value is known
  - Score to data where value is not known
  - Methods: Classification and Prediction
- Unsupervised learning
  - Goal: Segment data into meaningful segments; detect patterns
  - There is no target (outcome) variable to predict or classify
  - Methods: Association rules, collaborative filters, data reduction & exploration, visualization

Supervised: Classification

- Goal: Predict categorical target (outcome) variable
- Examples: Purchase/no purchase, fraud/no fraud, creditworthy/not creditworthy...
- Each row is a case (customer, tax return, applicant)
- Each column is a variable
- Target variable is often binary (yes/no)

Supervised: Prediction

- Goal: Predict numerical target (outcome) variable
- Examples: sales, revenue, performance
- Each row is a case (customer, tax return, applicant)
- Each column is a variable
- Taken together, classification and prediction constitute “predictive analytics”

Unsupervised: Association Rules

- Goal: Produce rules that define “what goes with what” in transactions
- Example: “If X was purchased, Y was also purchased”
- Rows are transactions
- Used in recommender systems – “Our records
- show you bought X, you may also like Y” Also called “affinity analysis”

Unsupervised: Collaborative Filtering

- Goal: Recommend products to purchase
- Based on products that customer rates, selects, views, or purchases
- Recommend products that “customers like you” purchase (user-based)
- Or, recommend products that share a “product purchaser profile” with your purchases (item- based)

Unsupervised: Data Reduction

- Distillation of complex/large data into simpler/smaller data - Reducing the number of variables/columns (e.g., principal components)
- Reducing the number of records/rows (e.g., clustering)

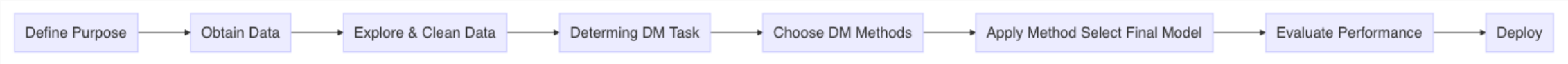
Unsupervised: Data Visualization

- Graphs and plots of data
- Histograms, boxplots, bar charts, scatterplots
- Especially useful to examine relationships between pairs of variables

3.The Steps in Data Mining

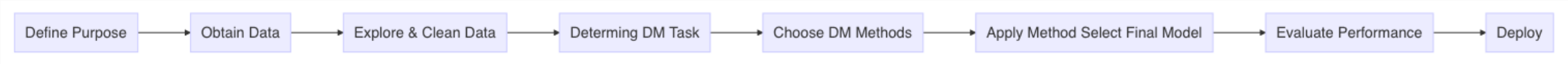
The general steps of data mining

- **Define Purpose**
  - Develop an understanding of the purpose of the data mining project.
  - How will the stakeholder use the results?
  - Who will be affected by the results?
  - Will the analysis be a one-shot effort or an ongoing procedure?
- **Obtian Data**
  - Obtain the dataset to be used in the analysis.
  - Sampling from a large database for an analysis.
- **Explore & Clean Data**
  - Explore, clean, and preprocess the data.
  - This step involves verifying that the data are in reasonable condition.
  - How should missing data be handled?
  - Are the values in a reasonable range, given what you would expect for each variable?
  - Are there obvious outliers?
  - The data are reviewed graphically



> *Reduce the data dimension, if necessary.* Dimension reduction can involve operations such as eliminating unneeded variables, transforming variables (e.g., turning "money spent" into "spent > $100" vs. "spent ≤ $100"), and creating new variables (e.g., a variable that records whether at least one of several products was purchased). Make sure that you know what each variable means and whether it is sensible to include it in the model.

- **Determining DM Task**
  - Determine the data mining task. (classification, prediction, clustering, etc.).
  - This involves translating the general question or problem of Step 1 into a more specific data mining question.
- **Apply Method & Select Final Model**
  - Choose the data mining techniques to be used. (regression, neural nets, hierarchical clustering, etc.).
- **Evaluate Performance**
  - Use algorithms to perform the task.
  - This is typically an iterative process trying multiple variants, and often using multiple variants of the same algorithm (choosing different variables or settings within the algorithm).



> *Interpret the results of the algorithms.* This involves making a choice as to the best algorithm to deploy, and where possible, testing the final choice on the test data to get an idea as to how well it will perform. (Recall that each algorithm may also be tested on the validation data for tuning purposes; in this way, the validation data become a part of the fitting process and are likely to underestimate the error in the deployment of the model that is finally chosen.)

- **Deploy**
  - Deploy the model.
  - This step involves integrating the model into operational systems and running it on real records to produce decisions or actions.

SEMMA & CRISP-DM

SEMMA: Developed by SAS

- Sample
  - Take a sample from the dataset; partition into training, validation, and test datasets
- Explore
  - Examine the dataset statistically and graphically
- Modify
  - Transform the variables and impute missing values
- Model
  - Fit predictive models
- Assess
  - Compare models using a validation dataset

CRISP-DM: Implemented in IBM SPSS Modeler

- Business Understanding
- Data Understanding
- Data Preparation
- Modeling
- Evaluation
- Deployment

Organization of Datasets

- Variables are in columns and records are in rows.
- Dataset
  - West Roxbury, Boston, in 2014. 14
  - Variables are recorded for over 5000 homes.
  - Each row represents a home—the first home's assessed value

Predicting Home Values in the West Roxbury Neighborhood

- Realtors now list houses and their prices on the web, and estimates of house and condominium prices have become widely available

What Happended?

- Zillow (www.zillow.com) is the most popular online real estate infor mation site in the United States
  - in 2014 they purchased their major rival, Trulia. By 2015, Zillow had become the dominant platform for checking house prices and, as such, the dominant online advertising venue for realtors.
  - What used to be a comfortable 6% commission structure for realtors, affording them a handsome surplus (and an oversupply of realtors)
  - Commission structure for realtors was being rapidly eroded by an increasing need to pay for advertising on Zillow.
  - Zillow gets much of the data for its “Zestimates” of home values directly from publicly available city housing data, used to estimate property values for tax assessment.

Motivation

- A competitor seeking to get into the market would likely take the same approach. So might realtors seeking to develop an alternative to Zillow.
  - A simple approach would be a naive, model-less method—just use the assessed values as determined by the city.
  - Those values, however, do not necessarily include all properties, and they might not include changes warranted by remodeling, additions, etc.
  - Moreover, the assessment methods used by cities may not be transparent or always reflect true market values.
  - However, the city property data can be used as a starting point to build a model, to which additional data (such as that collected by large realtors) can be added later.

Action

- Let's look at how Boston property assessment data, available from the city of Boston, might be used to predict home values.
  - The data in WestRoxbury.csv includes information on single family owner-occupied homes in West Roxbury, a neighborhood in southwest Boston, MA, in 2014.
  - The data include values for various predictor variables, and for an outcome—assessed home value ("total value").
  - This dataset has 14 variables

Loading and Looking at the Data in R

- Codebook

Variables	Description
TOTAL.VALUE	Total assessed value for property, in thousands of USD
TAX	Tax bill amount based on total assessed value multiplied by the tax rate, in USD
LOT.SQFT	Total lot size of parcel in square feet
YR.BUILT	Year the property was built
GROSS.AREA	Gross floor area
LIVING.AREA	Total living area for residential properties (ft)
FLOORS	Number of floors
ROOMS	Total number of rooms
BEDROOMS	Total number of bedrooms

	Variables	Description
10	FULL.BATH	Total number of full baths
11	HALF.BATH	Total number of half baths
12	KITCHEN	Total number of kitchens
13	FIREPLACE	Total number of fireplaces
14	REMODEL	When the house was remodeled (Recent/Old/None)

```
dim(data)
```

```
## [1] 5802   14
```

```
head(data)
```

```
##   TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 1       344.2 4330     9965     1880       2436        1352      2     6
## 2       412.6 5190     6590     1945       3108        1976      2    10
## 3       330.1 4152     7500     1890       2294        1371      2     8
## 4       498.6 6272    13773     1957       5032        2608      1     9
## 5       331.5 4170     5000     1910       2370        1438      2     7
## 6       337.4 4244     5142     1950       2124        1060      1     6
##   BEDROOMS FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODEL
## 1        3         1         1       1         0    None
## 2        4         2         1       1         0  Recent
## 3        4         1         1       1         0    None
## 4        5         1         1       1         1    None
## 5        3         2         0       1         0    None
## 6        3         1         0       1         1     Old
```

```r
data[1:10, 1]  # show the first 10 rows of the first column only
```

```
##  [1] 344.2 412.6 330.1 498.6 331.5 337.4 359.4 320.4 333.5 409.4
```

```r
data[1:10, ]  # show the first 10 rows of each of the columns
```

```
##    TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 1        344.2 4330     9965     1880       2436        1352      2     6
## 2        412.6 5190     6590     1945       3108        1976      2    10
## 3        330.1 4152     7500     1890       2294        1371      2     8
## 4        498.6 6272    13773     1957       5032        2608      1     9
## 5        331.5 4170     5000     1910       2370        1438      2     7
## 6        337.4 4244     5142     1950       2124        1060      1     6
## 7        359.4 4521     5000     1954       3220        1916      2     7
## 8        320.4 4030    10000     1950       2208        1200      1     6
## 9        333.5 4195     6835     1958       2582        1092      1     5
## 10       409.4 5150     5093     1900       4818        2992      2     8
##    BEDROOMS FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODEL
## 1         3         1         1       1         0    None
## 2         4         2         1       1         0  Recent
## 3         4         1         1       1         0    None
## 4         5         1         1       1         1    None
```

```r
data[5, 1:10]  # show the fifth row of the first 10 columns
```

```
##   TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 5       331.5 4170     5000     1910       2370        1438      2     7
##   BEDROOMS FULL.BATH
## 5        3         2
```

```r
data[5, c(1:2, 4, 8:10)]  # show the fifth row of some columns
```

```
##   TOTAL.VALUE  TAX YR.BUILT ROOMS BEDROOMS FULL.BATH
## 5       331.5 4170     1910     7        3         2
```


```r
data[, 1]  # show the whole first column
```

```
##    [1]  344.200  412.600  330.100  498.600  331.500  337.400  359.400  320.400
##    [9]  333.500  409.400  313.000  344.500  315.500  575.000  326.200  298.200
##   [17]  313.100  344.900  330.700  348.000  317.500  330.800  357.800  414.700
##   [25]  318.800  346.200  245.100  317.400  247.300  320.800  328.800  293.600
##   [33]  280.100  342.500  337.200  336.500  315.600  296.500  239.700  318.400
##   [41]  431.500  349.800  344.000  358.300  346.200  490.700  371.900  325.300
##   [49]  340.500  432.500  334.700  314.300  405.500  326.000  285.600  345.900
##   [57]  317.300  277.800  306.800  265.700  356.800  299.900  287.800  296.000
##   [65]  300.600  566.300  286.000  351.200  365.100  330.000  354.100  335.800
##   [73]  345.000  356.400  379.000  352.100  437.300  404.200  283.400  326.400
##   [81]  309.200  332.900  260.400  339.300  307.900  346.300  479.100  335.600
##   [89]  261.500  355.100  356.800  287.300  331.900  387.100  307.600  360.900
##   [97]  466.100  340.000  303.900  303.500  301.400  339.900  281.300  530.100
##  [105]  310.400  344.600  525.100  285.300  302.300  243.300  334.900  327.500
##  [113]  586.600  347.300  267.600  274.200  404.600  339.900  420.800  283.500
##  [121]  620.300  346.300  295.070  297.600  310.700  442.600  277.500  310.600
##  [129]  327.100  307.500  243.300  352.800  258.600  295.500  490.700  330.300
##  [137]  290.800  350.700  301.100  278.700  298.200  249.000  287.500  434.300
##  [145]  508.800  338.800  363.500  335.300  353.500  291.500  325.800  368.200
##  [153]  361.700  286.300  315.700  282.400  337.500  320.100  272.900  243.200
```

```
data$TOTAL.VALUE  # a different way to show the whole first column
```

```
##    [1]  344.200  412.600  330.100  498.600  331.500  337.400  359.400  320.400
##    [9]  333.500  409.400  313.000  344.500  315.500  575.000  326.200  298.200
##   [17]  313.100  344.900  330.700  348.000  317.500  330.800  357.800  414.700
##   [25]  318.800  346.200  245.100  317.400  247.300  320.800  328.800  293.600
##   [33]  280.100  342.500  337.200  336.500  315.600  296.500  239.700  318.400
##   [41]  431.500  349.800  344.000  358.300  346.200  490.700  371.900  325.300
##   [49]  340.500  432.500  334.700  314.300  405.500  326.000  285.600  345.900
##   [57]  317.300  277.800  306.800  265.700  356.800  299.900  287.800  296.000
##   [65]  300.600  566.300  286.000  351.200  365.100  330.000  354.100  335.800
##   [73]  345.000  356.400  379.000  352.100  437.300  404.200  283.400  326.400
##   [81]  309.200  332.900  260.400  339.300  307.900  346.300  479.100  335.600
##   [89]  261.500  355.100  356.800  287.300  331.900  387.100  307.600  360.900
##   [97]  466.100  340.000  303.900  303.500  301.400  339.900  281.300  530.100
##  [105]  310.400  344.600  525.100  285.300  302.300  243.300  334.900  327.500
##  [113]  586.600  347.300  267.600  274.200  404.600  339.900  420.800  283.500
##  [121]  620.300  346.300  295.070  297.600  310.700  442.600  277.500  310.600
##  [129]  327.100  307.500  243.300  352.800  258.600  295.500  490.700  330.300
##  [137]  290.800  350.700  301.100  278.700  298.200  249.000  287.500  434.300
##  [145]  508.800  338.800  363.500  335.300  353.500  291.500  325.800  368.200
##  [153]  361.700  286.300  315.700  282.400  337.500  320.100  272.900  243.200
```

```r
data$TOTAL.VALUE[1:10]  # show the first 10 rows of the first column
```

```
##  [1] 344.2 412.6 330.1 498.6 331.5 337.4 359.4 320.4 333.5 409.4
```

```r
length(data$TOTAL.VALUE)  # find the length of the first column
```

```
## [1] 5802
```

```r
mean(data$TOTAL.VALUE)  # find the mean of the first column
```

```
## [1] 392.6857
```


```
summary(data)  # find summary statistics for each column
```

```
##   TOTAL.VALUE          TAX          LOT.SQFT       YR.BUILT      GROSS.AREA
##  Min.   : 105.0   Min.   : 1320   Min.   :  997   Min.   :   0   Min.   : 821
##  1st Qu.: 325.1   1st Qu.: 4090   1st Qu.: 4772   1st Qu.:1920   1st Qu.:2347
##  Median : 375.9   Median : 4728   Median : 5683   Median :1935   Median :2700
##  Mean   : 392.7   Mean   : 4939   Mean   : 6278   Mean   :1937   Mean   :2925
##  3rd Qu.: 438.8   3rd Qu.: 5520   3rd Qu.: 7022   3rd Qu.:1955   3rd Qu.:3239
##  Max.   :1217.8   Max.   :15319   Max.   :46411   Max.   :2011   Max.   :8154
##   LIVING.AREA       FLOORS          ROOMS          BEDROOMS      FULL.BATH
##  Min.   : 504   Min.   :1.000   Min.   : 3.000   Min.   :1.00   Min.   :1.000
##  1st Qu.:1308   1st Qu.:1.000   1st Qu.: 6.000   1st Qu.:3.00   1st Qu.:1.000
##  Median :1548   Median :2.000   Median : 7.000   Median :3.00   Median :1.000
##  Mean   :1657   Mean   :1.684   Mean   : 6.995   Mean   :3.23   Mean   :1.297
##  3rd Qu.:1874   3rd Qu.:2.000   3rd Qu.: 8.000   3rd Qu.:4.00   3rd Qu.:2.000
##  Max.   :5289   Max.   :3.000   Max.   :14.000   Max.   :9.00   Max.   :5.000
##    HALF.BATH         KITCHEN        FIREPLACE        REMODEL
##  Min.   :0.0000   Min.   :1.000   Min.   :0.0000   Length:5802
##  1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:0.0000   Class :character
##  Median :1.0000   Median :1.000   Median :1.0000   Mode  :character
##  Mean   :0.6139   Mean   :1.015   Mean   :0.7399
##  3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000
```

Sampling from a Database

- Typically, data mining on less than the complete database.
- Data mining algorithms will have varying limitations on the numbers of records and variables,
- limitations that may be specific to computing power and capacity as well as software limitations. Even within those limits, many algorithms will execute faster with smaller samples.
- Accurate models can often be built with as few as several thousand records.

```
# random sample of 5 observations
s <- sample(row.names(data), 5)
data[s,]
```

```
##      TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 51         334.7 4210     4584     1907       3288        1694    2.0     7
## 720        287.6 3618     4216     1923       2704        1270    1.5     6
## 730        236.9 2980     2990     1915       1516         972    2.0     5
## 5340       380.8 4790     6381     1954       2653        1416    2.0     8
## 2712       362.4 4558     3900     1915       3864        2018    2.0    10
##      BEDROOMS FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODEL
## 51          3         1         1       1         0    None
```

Oversampling Rare Events in Classification Tasks

> If the event we are interested in classifying is rare, sampling a random subset of records may yield so few events (e.g., purchases) that we have little information on them. We would end up with lots of data on nonpurchasers and non-fraudulent transactions but little on which to base a model that distinguishes purchasers from nonpurchasers or fraudulent from non-fraudulent.

In such cases,

- Sampling procedure to overweight the rare class (purchasers or frauds) relative to the majority class (nonpurchasers, non-frauds)
- Assuring an adequate number of responder or “success” cases to train the model is just part of the picture.

The costs of mis-classification.

- Whenever the response rate is extremely low, we are likely to attach more importance to identifying a responder than to identifying a non-responder.
- If the costs of failing to locate responders are comparable to the costs of misidentifying responders as non-responders, our models would usually achieve highest overall accuracy if they identified everyone as a non-responder
- In such a case, the misclassification rate is very low—equal to the rate of responders—but the model is of no value.

More generally, we want to train our model with the asymmetric costs in mind so that the algorithm will catch the more valuable responders, probably at the cost of “catching” and misclassifying more non-responders as responders than would be the case if we assume equal costs.

Preprocessing and Cleaning the Data

Types of Variables

- Variables can be
  - Numerical
  - Text (character/string),
  - Continuous (able to assume any real numerical value, usually in a given range),
  - Integer (taking only integer values),
  - Categorical (assuming one of a limited number of values)
  - Date

Categorical Variables

- Categorical variables can be either
  - coded as numerical (1, 2, 3)
  - text (payments current, payments not current, bankrupt).
- Categorical variables can be
  - unordered (called nominal variables) with categories
    - e.g. North America, Europe, and Asia;
  - they can be ordered (called ordinal variables) with categories
    - e.g. high value, low value, and nil value.

- Continuous variables can be handled by most data mining routines with the exception of the naive Bayes classifier, which deals exclusively with categorical predictor variables.
- The machine learning roots of data mining grew out of problems with categorical outcomes; the roots of statistics lie in the analysis of continuous variables.
- Sometimes, it is desirable to convert continuous variables to categorical variables.
    - This is done most typically in the case of outcome variables, where the numerical variable is mapped to a decision
        - credit scores above a certain threshold mean “grant credit,”
        - a medical test result above a certain threshold means “start treatment”

```r
names(data)  # print a list of variables to the screen.
```

```
##  [1] "TOTAL.VALUE" "TAX"         "LOT.SQFT"    "YR.BUILT"    "GROSS.AREA"
##  [6] "LIVING.AREA" "FLOORS"      "ROOMS"       "BEDROOMS"    "FULL.BATH"
## [11] "HALF.BATH"   "KITCHEN"     "FIREPLACE"   "REMODEL"
```

```r
t(t(names(data)))  # print the list in a useful column format
```

```
##       [,1]
##  [1,] "TOTAL.VALUE"
##  [2,] "TAX"
##  [3,] "LOT.SQFT"
##  [4,] "YR.BUILT"
##  [5,] "GROSS.AREA"
##  [6,] "LIVING.AREA"
##  [7,] "FLOORS"
##  [8,] "ROOMS"
##  [9,] "BEDROOMS"
## [10,] "FULL.BATH"
## [11,] "HALF.BATH"
## [12,] "KITCHEN"
## [13,] "FIREPLACE"
```

Reviewing Variables in R

```
colnames(data)[1] <- c("TOTAL.VALUE")  # change the first column's name
class(data$REMODEL) # REMODEL is a factor variable
```

```
## [1] "character"
```

```
class(data[ ,14]) # Same.
```

```
## [1] "character"
```

```
levels(data[, 14])  # It can take one of three levels
```

```
## NULL
```

```
class(data$BEDROOMS)  # BEDROOMS is an integer variable
```

```
## [1] "integer"
```

```
class(data[, 1])  # Total_Value is a numeric variable
```

Handling Categorical Variables

> Categorical variables can also be handled by most data mining routines, but often require special handling.

- If the categorical variable is ordered (age group, degree of creditworthiness, etc.), we can sometimes code the categories numerically (1, 2, 3, ...) and treat the variable as if it were a continuous variable.
- The smaller the number of categories, and the less they represent equal increments of value, the more problematic this approach becomes, but it often works well enough.
- Nominal categorical variables, however, often cannot be used as is.
- In many cases, they must be decomposed into a series of binary variables, called dummy variables.
  - For example, a single categorical variable that can have possible values of “student,” “unemployed,” “employed,” or “retired” would be split into four separate dummy variables:

Creating Binary Dummies

```
# use model.matrix() to convert all categorical variabl
# a set of dummy variables. We must then turn the resul
# a data frame for further work.
xtotal <- model.matrix(~ 0 + BEDROOMS + REMODEL, data =

#xtotal$BEDROOMS[1:5]  # will not work because xtotal i
xtotal <- as.data.frame(xtotal)
t(t(names(xtotal)))  # check the names of the dummy var
```

```
##      [,1]
## [1,] "BEDROOMS"
## [2,] "REMODELNone"
## [3,] "REMODELOld"
## [4,] "REMODELRecent"
```

```
head(xtotal)
```

```
##   BEDROOMS REMODELNone REMODELOld REMODELRecent
## 1        3           1          0             0
## 2        4           0          0             1
## 3        4           1          0             0
## 4        5           1          0             0
## 5        3           1          0             0
## 6        3           0          1             0
```

```r
xtotal <- xtotal[, -4]  # drop one of the dummy variables.
# In this case, drop REMODELRecent.
head(xtotal)
```

```
##   BEDROOMS REMODELNone REMODELOld
## 1        3           1          0
## 2        4           0          0
## 3        4           1          0
## 4        5           1          0
## 5        3           1          0
## 6        3           0          1
```

Variable Selection

Other things being equal, parsimony, or compactness, is a desirable feature in a model.

the more variables we include and the more complex the model, the greater the number of records we will need.

Models based on many variables are often less robust. More data quality and availability issues, and require more data cleaning and pre-processing.

So then..."*How many varibles and how much data?*"

How Many Variables and How Much Data?

Statisticians give us procedures to learn with some precision how many records we would need to achieve a given degree of reliability with a given dataset and a given model.

These are called “power calculations”

- Power calculations and are intended to assure that an average population effect will be estimated with sufficient precision from a sample.

Data miners’ needs are usually different, because the focus is not on identifying an average effect but rather on predicting individual records.

This purpose typically requires larger samples than those used for statistical inference.

- A good rule of thumb is to have **10 records for every predictor variable.**
- Another rule used for classification procedures, is to have **at least 6 × m × p records** (Delmaster and Hancock, 2001, p. 68), where m is the number of outcome classes and p is the number of variables.

In general

Compactness or parsimony is a desirable feature in a data mining model.

- We often end up with many more after creating new variables.
- Data visualization and dimension reduction methods help reduce the number of variables.

Pay close attention to the variables

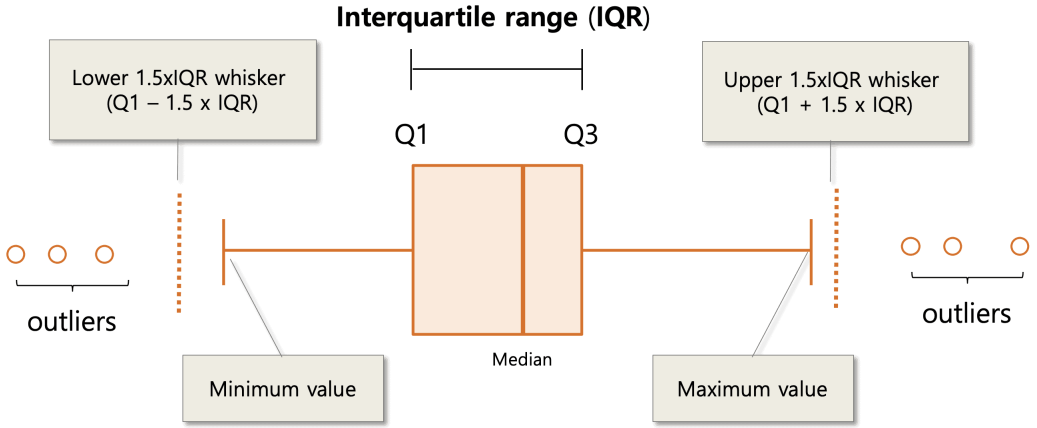
- Someone with domain knowledge should be consulted
- knowledge of what the variables represent is typically critical for building a good model and avoiding errors.

Examples

- Building a prediction model for the total purchase amount spent by customers
- Building a classification model for loan default at the time a customer applies for a loan.

Outliers

- The more data we are dealing with, the greater the chance of encountering erroneous values resulting from
  - measurement error
  - data-entry error
- If the erroneous value is in the same range as the rest of the data, it may be harmless.
- If it is well outside the range of the rest of the data (a misplaced decimal, for example), it may have a substantial effect on some of the data mining procedures we plan to use.



Values that lie far away from the bulk of the data are called outliers.

- The term **far away** is deliberately left vague; It is an arbitrary decision.
- Analysts use rules of thumb
  - **“anything over three standard deviations away from the mean is an outlier”**
- but no statistical rule can tell us whether such an outlier is the result of an error.

In this statistical sense

An outlier is not necessarily an invalid data point, it is just a distant one.

The purpose of identifying outliers

Usually to call attention to values that need further review. (the case of a misplaced decimal)

Judgments best made by someone with domain knowledge

- Statistical procedures can do little beyond identifying the record as something that needs review.
  - If manual review is feasible, some outliers may be identified and corrected.
  - If the number of records with outliers is very small, they might be treated as missing data.

How do we inspect for outliers?

- One technique is to
    1. sort the records by the first column (e.g., using the R function order(),
    2. then review the data for very large or very small values in that column.
    3. Then repeat for each successive column.
- Another option is to examine the minimum and maximum values of each column using R's min() and max() functions.
- Clustering techniques: Considers each record as a unit.; identifying clusters of one or a few records that are distant from others.

Missing Values

- Typically, some records will contain missing values.
    - Missing values is small -> omit.
    - A large number of variables -> Problem
        - 30 variables
        - 5% of the values are missing (spread randomly)
        - Almost 80% of the records would have to be omitted from the analysis.
- Replace the missing value with an imputed value
    - if among 30 variables
    - Household income is missing for a particular record
        - Substitute the mean household income
        - Add any information about how household income affects the outcome variable.
- It merely allows us to proceed with the analysis and not lose the information contained in this record for the other 29 variables.

```r
# To illustrate missing data procedures, we first convert a few entries for
# bedrooms to NA's. Then we impute these missing values using the median of the
# remaining values.
rows.to.missing <- sample(row.names(data), 10)
data[rows.to.missing,]$BEDROOMS <- NA
summary(data$BEDROOMS)  # Now we have 10 NA's and the median of the
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.00    3.00    3.00    3.23    4.00    9.00      10
```

```r
# remaining values is 3.
# replace the missing values using the median of the remaining values.
# use median() with na.rm = TRUE to ignore missing values when computing the median.
data[rows.to.missing,]$BEDROOMS <- median(data$BEDROOMS, na.rm = TRUE)
summary(data$BEDROOMS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   3.000   3.229   4.000   9.000
```

Normalizing (Standardizing) and Rescaling Data

Normalizing

Some algorithms require that the data be normalized before the algorithm can be implemented effectively.

- To normalize a variable,
  - **Standardizing**
    - subtract the mean from each value and then divide by the standard deviation.
    - In R, function scale() performs this operation.
  - **Z-score**
    - Number of standard deviations away from the mean

Normalizing is one way to bring all variables to the same scale.

Rescaling

- Rescaling each variable to a [0,1] scale.
  - Subtracting the minimum value and then dividing by the range.
  - Subtracting the minimum shifts the variable origin to zero.
  - Dividing by the range shrinks or expands the data to the range [0,1].
  - In R, rescaling can be done using function rescale() in the scales package.

Normalizing and Scaling Case

The case of clustering.

- Clustering typically involves calculating a distance measure that reflects **how far** each record is **from a cluster center** or **from other records**.
- With multiple variables, different units will be used:
  - days, dollars, counts, and so on.
  - If the dollars are in the thousands and everything else is in the tens, the dollar variable will come to dominate the distance measure.
  - Changing units from, say, days to hours or months, could alter the outcome completely.

5.Predictive Power and Overfitting

> In supervised learning, a key question presents itself: How well will our prediction or classification model perform when we apply it to new data?

- Comparing the performance of various models
- Choose the one shows the best when it is implemented in practice.
- A key concept is to make sure that our chosen model generalizes beyond the dataset.
- To assure generalization
  - Data partitioning
  - Try to avoid overfitting.

Overfitting

The more variables we include in a model, the greater the risk of overfitting the particular data used for modeling.

- A basic purpose of building a model is to represent relationships among variables
- This representation will do a good job of predicting future outcome values

> We want the model to do a good job of describing the data we have, but we are more interested in its performance with future data.

What is overfitting?

Overfitting is a concept in data science, which occurs when a statistical model fits exactly against its training data. When this happens, the algorithm unfortunately cannot perform accurately against unseen data, defeating its purpose.

Overfitting Causes and Consequence

Causes:

- Too many predictors
- A model with too many parameters
- Trying many different models

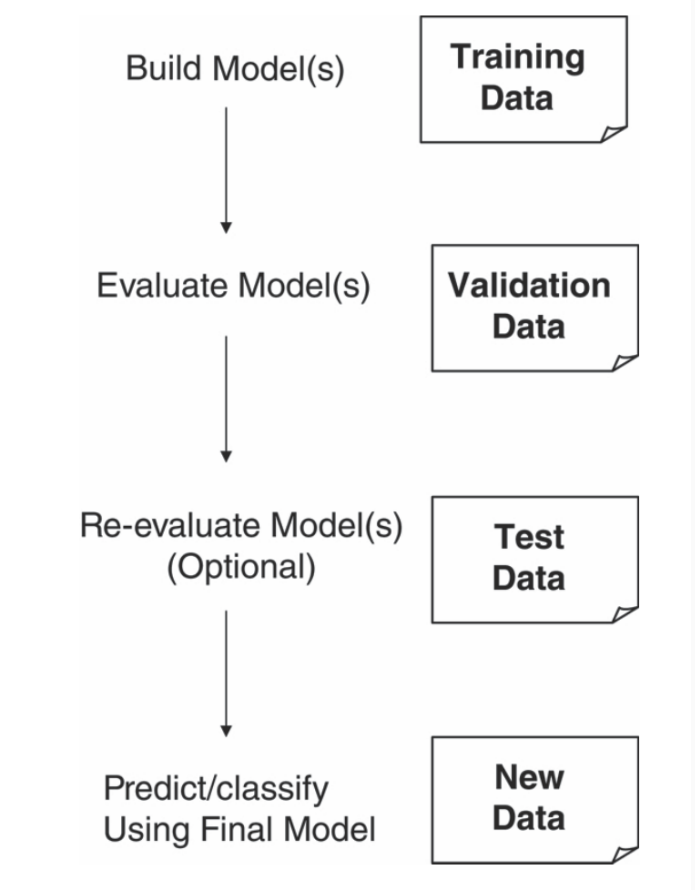
Consequence

Deployed model will not work as well as expected with completely new data

Creation and Use of Data Partitions

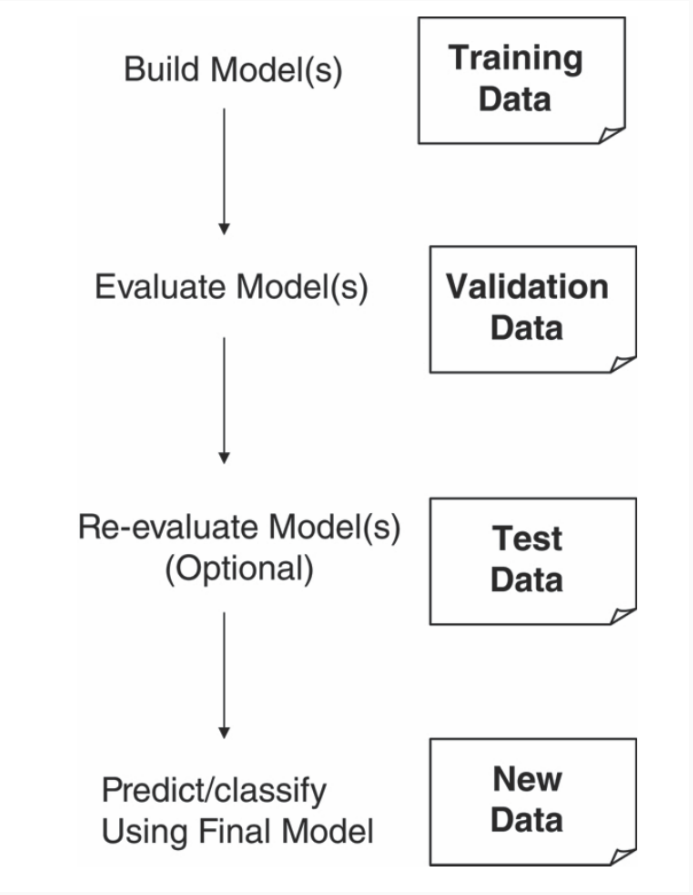
Partitioning the Data

- Problem: How well will our model perform with new data?
- Solution: Separate data into two parts
  - **Training** partition to develop the model
  - **Validation** partition to implement the model and evaluate its performance on "new" data



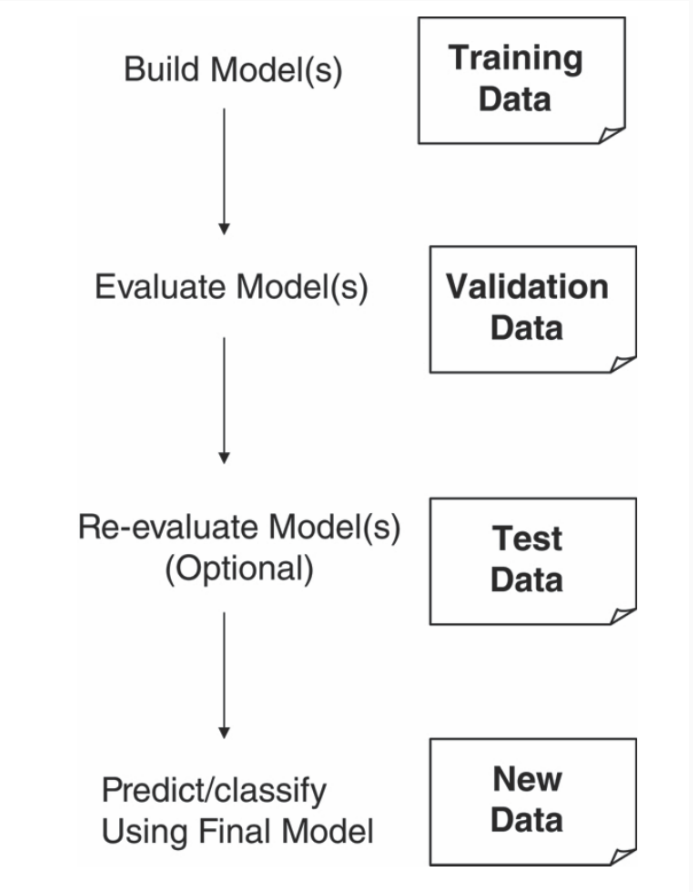
Test Partition

- When a model is developed on training data, it can overfit the training data (hence need to assess on validation)
- Assessing multiple models on same validation data can overfit validation data
- Some methods use the validation data to choose a parameter. This too can lead to overfitting the validation data
- Solution:final selected model is applied to a test partition to give



Cross Validation

- Repeated partitioning = cross-validation (“cv”)
- k-fold cross validation, e.g. k=5
  - For each fold, set aside 1/5 of data as validation
  - Use full remainder as training
  - The validation folds are non-overlapping




```
set.seed(1)
## Case1
## partitioning into training (60%) and validation (40%)
# validation
train.rows <- sample(rownames(data), dim(data)[1]*0.6)
head(train.rows, 3)
```

```
## [1] "1017" "4775" "2177"
```

```
# collect all the columns with training row ID into training set:
train.data <- data[train.rows, ]
head(train.rows, 3)
```

```
## [1] "1017" "4775" "2177"
```

```r
# assign row IDs that are not already in the training set, into validation
valid.rows <- setdiff(rownames(data), train.rows)
head(valid.rows,3)
```

```
## [1] "2" "5" "6"
```

```r
valid.data <- data[valid.rows, ]
head(valid.data,3)
```

```
##   TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 2       412.6 5190     6590     1945       3108        1976      2    10
## 5       331.5 4170     5000     1910       2370        1438      2     7
## 6       337.4 4244     5142     1950       2124        1060      1     6
##   BEDROOMS FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODEL
## 2        4         2         1       1         0  Recent
## 5        3         2         0       1         0    None
## 6        3         1         0       1         1     Old
```

```
## Case2
## partitioning into training (50%), validation (30%), test (20%)
# randomly sample 50% of the row IDs for training
train.rows <- sample(rownames(data), dim(data)[1]*0.5)
head(train.rows, 3)
```

```
## [1] "1886" "3515" "460"
```

```
# sample 30% of the row IDs into the validation set, drawing only from records
# not already in the training set
# use setdiff() to find records not already in the training set
valid.rows <- sample(setdiff(rownames(data), train.rows),
                     dim(data)[1]*0.3)
head(valid.rows,3)
```

```
## [1] "5642" "2766" "3676"
```

```
# assign the remaining 20% row IDs serve as test
test.rows <- setdiff(rownames(data), union(train.rows, valid.rows))
head(test.rows, 3)
```

```
## [1] "1"  "4"  "10"
```

```r
# create the 3 data frames by collecting all columns from the appropriate rows
train.data <- data[train.rows, ]
head(train.data, 3)
```

```
##      TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 1886       356.7 4487     5121     1935       2044        1218    2.0     6
## 3515       333.3 4192     3760     1915       2877        1295    1.0     7
## 460        298.6 3756     6935     1965       2628        1469    1.5     6
##      BEDROOMS FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODEL
## 1886        3         1         1       1         0  Recent
## 3515        4         1         1       1         1    None
## 460         2         2         0       1         0    None
```

```r
valid.data <- data[valid.rows, ]
head(valid.data, 3)
```

```
##      TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 5642       318.0 4000     4026     1940       2700        1461    2.0     6
## 2766       498.7 6273    11905     1910       3030        1817    2.0     7
## 3676       331.8 4174     4166     1940       2676        1497    1.5     7
##      BEDROOMS FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODEL
## 5642        3         1         1       1         1    None
```

Modeling Process

We now describe in detail the various model stages using the West Roxbury home values example.

- **1. Determine the purpose.**
  - Let's assume that the purpose of our data mining project is to predict the value of homes in West Roxbury for new records.
- **2. Obtain the data.**
  - We will use the 2014 West Roxbury housing data.
- **3. Explore, clean, and preprocess the data.**
  - Data dictionary
  - These descriptions are available on the "description"
  - Variable names are cryptic and their descriptions may be unclear or missing.
- **4. Reduce the data dimension.**
  - it has only 13 variables
  - If we had many more variables, at this stage we might want to apply a variable reduction technique
- **5. Determine the data mining task.**
  - The specific task is to predict the value of TOTAL VALUE using the predictor variables.

- **6. Partition the data (for supervised tasks).**
  - The training partition is used to build the model,
  - The validation partition is used to see how well the model does when applied to new data.
  - We need to specify the percent of the data used in each partition.
- **7. Choose the technique.**
  - In this case, it is multiple linear regression.
  - Having divided the data into training and validation partitions, we can build a multiple linear regression model with the training data.
  - We want to predict the value of a house in West Roxbury on the basis of all the other predictors (except TAX).
- **8. Use the algorithm to perform the task.**
  - we use the lm() function to predict house value with the training data,
  - use the same model to predict values for the validation data.
  - Note that the predicted values are often called the fitted values, since they are for the records to which the model was fit.

- Continued from "**8. Use the algorithm to perform the task.** "
  - Prediction error can be aggregated in several ways. Five common measures
    - Error (actual-predicted)
    - ME (mean error)
    - RMSE (root-mean-squared error = square root of average squared error)
    - MAE (mean absolute error)
    - MPE (mean percentage error)
    - MAPE (mean absolute percentage error)
- **9. Interpret the results.**
  - try other prediction algorithms (e.g., regression trees) and see how they do error-wise.
  - try different “settings” on the various models
  - After choosing the best model, we use that model to predict the output variable in fresh data. (the model with the lowest error on the validation data while also recognizing that “simpler is better”)
- **10. Deploy the model.**
  - Applied to new data to predict TOTAL VALUE for homes where this value is unknown.
  - Predicting the output value for new records is called scoring.
  - For predictive tasks, scoring produces predicted numerical values.
  - For classification tasks, scoring produces classes and/or propensities.

fitting a regression model to training data (West Roxbury)

```
reg <- lm(TOTAL.VALUE ~ ., data = data, subset = train.rows)
tr.res <- data.frame(train.data$TOTAL.VALUE, reg$fitted.values, reg$residuals)
head(tr.res)
```

```
##      train.data.TOTAL.VALUE reg.fitted.values reg.residuals
## 1886                  356.7          356.7174  -0.017447269
## 3515                  333.3          333.2672   0.032764265
## 460                   298.6          298.6112  -0.011193854
## 855                   265.3          265.3050  -0.005003036
## 4094                  575.1          575.0798   0.020211093
## 3581                  348.0          347.9750   0.025021258
```

code for applying the regression model to predict validation set (West Roxbury)

```
pred <- predict(reg, newdata = valid.data)
vl.res <- data.frame(valid.data$TOTAL.VALUE, pred, residuals =valid.data$TOTAL.VALUE - pred)
head(vl.res)
```

```
##      valid.data.TOTAL.VALUE     pred    residuals
## 5642                  318.0 318.0047 -0.004675490
## 2766                  498.7 498.6882  0.011822683
## 3676                  331.8 331.8371 -0.037075586
## 2054                  371.9 371.8998  0.000225092
## 2217                  436.2 436.2079 -0.007860638
## 3117                  280.2 280.1660  0.034011570
```

code for computing model evaluation metrics

```
library(forecast)
# compute accuracy on training set
accuracy(reg$fitted.values, train.data$TOTAL.VALUE)
```

```
##                       ME       RMSE        MAE          MPE        MAPE
## Test set -5.339452e-16 0.02273249 0.01969236 2.69018e-06 0.005303597
```

```
# compute accuracy on prediction set
pred <- predict(reg, newdata = valid.data)
accuracy(pred, valid.data$TOTAL.VALUE)
```

```
##                       ME       RMSE        MAE           MPE        MAPE
## Test set -0.000165257 0.02300795 0.01984774 -9.541149e-05 0.005331648
```

Data frame with three record to be scored

```
new.data # you need to create new data
```

```
##      TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS BEDROOMS
## 100 3818     4200     1960       2670        1710    2.0    10        4
## 101 3791     6444     1940       2886        1474    1.5     6        3
## 102 4275     5035     1925       3264        1523    1.0     6        2
##     FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODEL
## 100         1         1       1         1    None
## 101         1         1       1         1    None
## 102         1         0       1         0  Recent
```

```
pred <- predict(reg, newdata = new.data)
pred
```

```
##      100      101      102
## 303.5376 301.3916 339.8632
```

7.Using R for Data Mining on a Local Machine

- The data mining process require a huge number of records.
- The use of a sample of 20,000 is likely to yield as accurate
- The number of records required in each partition (training, validation, and test) can be accommodated within the memory limit allowed in R (memory.limit()).
- When we apply Big Data analytics in R, it might be useful
  - to remove unused objects (function rm())
  - to call the garbage collection (function gc()) afterwards.

8.Automating Data Mining Solutions

In most supervised data mining applications,

- the goal is not a static, one-time analysis of a particular dataset.
- Rather, we want to develop a model that can be used on an ongoing basis to predict or classify new records.
- Our initial analysis will be in prototype mode, while we explore and define the problem and test different models.
- We will follow all the steps outlined earlier in this chapter.

At the end of that process,

- we will typically want our chosen model to be deployed in automated fashion.
- In practice, this is done by building the chosen algorithm into the computational setting in which the rest of the process lies.

The different components of the system communicate with one another via Application Programming Interfaces (APIs)

- that establish locally valid rules for transmitting data and associated communications. An
- API for a data mining algorithm would establish the required elements for a predictive algorithm to work—the exact predictor variables, their order, data formats, etc.

It would also establish the requirements for communicating the results of the algorithm. Algorithms to be used in an automated data pipeline will need to be compliant with the rules of the APIs where they operate.

Finally, once the computational environment is set and functioning,

- the data miner's work is not done.
- The environment in which a model operates is typically dynamic
- Predictive models often have a short shelf life—one leading consultant finds they rarely continue to function effectively for more than a year.
- So, even in a fully deployed state, models must be periodically checked and re-evaluated.
- Once performance flags, it is time to return to prototype mode and see if a new model can be developed.

Summary

- Data Mining consists of supervised methods (Classification & Prediction) and unsupervised methods (Association Rules, Data Reduction, Data Exploration & Visualization)
- Before algorithms can be applied, data must be explored and pre-processed
- To evaluate performance and to avoid overfitting, data partitioning is used
- Models are fit to the training partition and assessed on the validation and test partitions
- Data mining methods are usually applied to a sample from a large database, and then the best model is used to score the entire database

Chapter Video

References

Shmueli, G., P. C. Bruce, P. Gedeck, and N. R. Patel (2019). Data mining for business analytics: concepts, techniques and applications in Python. John Wiley & Sons.

Shmueli, G., P. C. Bruce, I. Yahav, N. R. Patel, and K. C. Lichtendahl Jr (2017). Data mining for business analytics: concepts, techniques, and applications in R. John Wiley & Sons.

Assignment