Data Science

Deriving Knowledge from Data at Scale

Data Exploration, Transformation and Feature Selection

Wee Hyong Tok
3 May 2017

Data Characterization

1. Unique values
2. Most frequent values
3. Highest and lowest values
4. Location and dispersion – gini, statistical test for dispersion
5. Quartiles

Data Cleaning

1. Missing values
2. Outliers
3. Coding

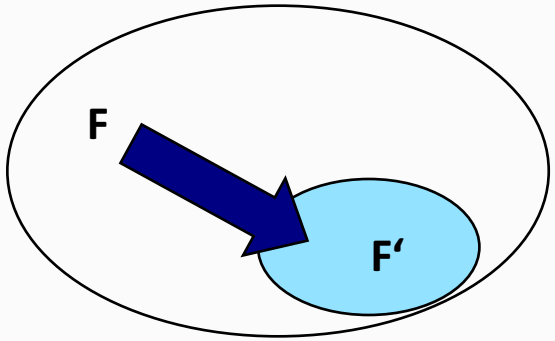
Feature Selection

Simple Definition

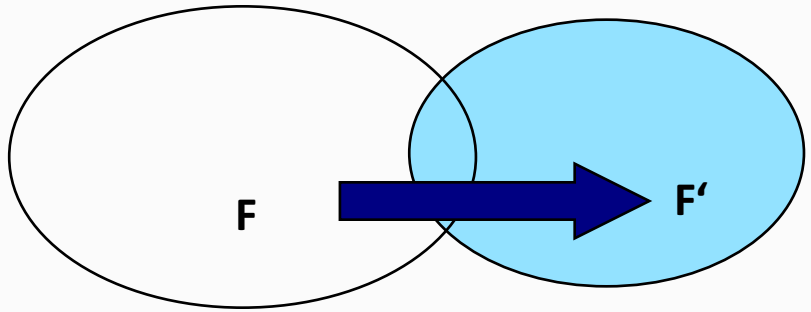
$$F=\{f_1,\ldots,f_i,\ldots,f_n\}$$

Given a set of features ,the **feature selection problem** is to find a subset $F' \subseteq F$ that "*maximizes the learners ability to classify patterns*". **Feature extraction** creates new features F → F'…

$$\{f_1,\ldots,f_i,\ldots,f_n\} \xrightarrow{f.selection} \{f_{i_1},\ldots,f_{i_j},\ldots,f_{i_m}\}$$



- **Feature extraction** creates new features F → F'…

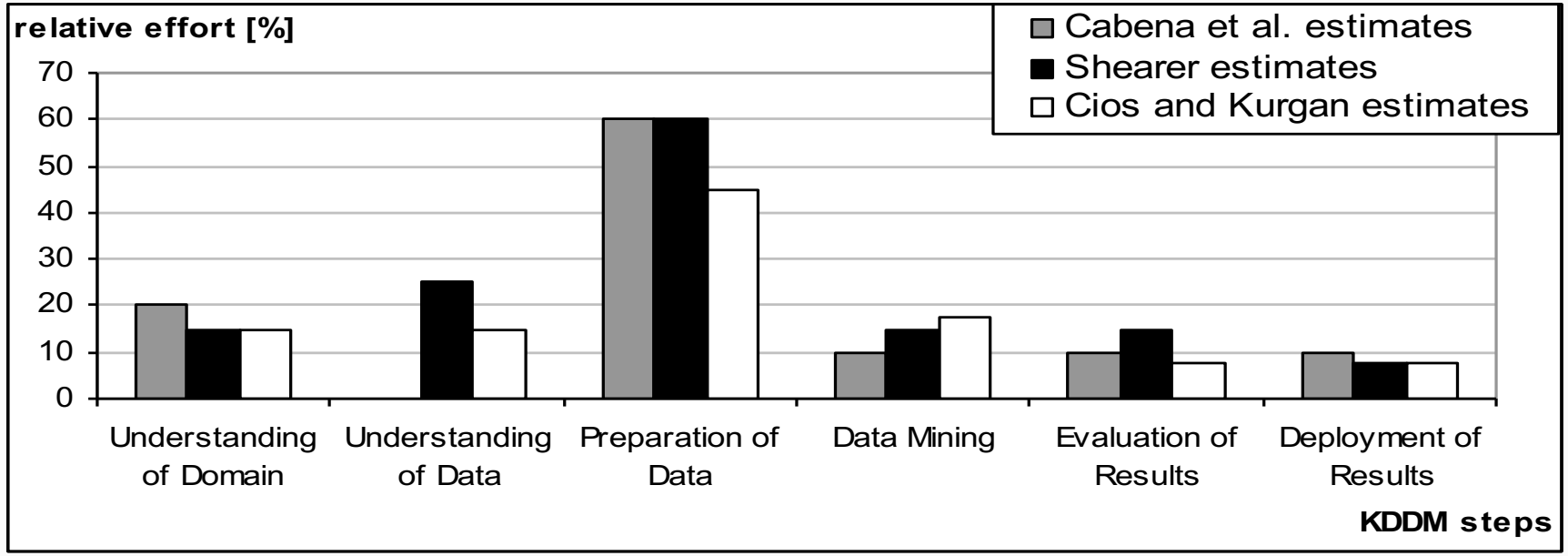


$$\{f_1,\ldots,f_i,\ldots,f_n\} \xrightarrow{f.extraction} \{g_1(f_1,\ldots,f_n),\ldots,g_j(f_1,\ldots,f_n),\ldots,g_m(f_1,\ldots,f_n)\}$$

Time Spent on KDD Components

An important aspect of the KDD contest is relative time to complete each step

- It enables precise scheduling
- Estimates by researchers and practitioners are shown below
  - Specific estimated values depend on existing knowledge about the domain, skill level of the humans, complexity of the problem, etc.
  - Data preparation step is by far the most time consuming step



Out of Class Reading, highly recommended

A Study on the Importance of and Time Spent on Different Modeling Steps

M. Arthur Munson
Sandia National Laboratories
Livermore, CA 94551, USA
mamunso@sandia.gov

ABSTRACT

Applying data mining and machine learning algorithms requires many steps to prepare data and to make use of modeling results. This study investigates two questions: (1) how time consuming are the pre- and post-processing steps? (2) how much research energy is spent on these steps? To answer these questions I surveyed practitioners about their experiences in applying modeling techniques and categorized data mining and machine learning research papers from 2009 according to the modeling step(s) they addressed. Survey results show that model building consumes only 14% of the time spent on a typical project; the remaining time is spent on pre- and post-processing steps. Both survey responses and the categorization of research papers show that data mining and machine learning researchers spend the majority of their energy on algorithms for constructing models and significantly less energy on other steps. These findings collectively suggest that there are research opportunities to simplify the steps that precede and follow model building.

1. INTRODUCTION

In this paper I investigate how time consuming the various modeling steps are for practitioners and how much research effort is focused on each step. To answer these questions I surveyed practitioners about their experiences in applying data mining (machine learning) techniques and manually categorized the 2009 proceedings from two of the top conferences in the area (ICML 2009[1] and KDD 2009[2]) based on the step(s) they addressed.

There are two main findings in this study. First, in the typical project only 14% of the time is spent building the model. The rest of the time is spent preparing to do model learning and verifying the results after model construction. In contrast, the data mining and machine learning research communities spend the majority of their energy on how to learn a model from data, and moderate energy or less on other modeling steps. This finding is supported both by survey responses and by the distribution of papers at ICML 2009 and KDD 2009. In addition to documenting the current state of practice and research, these findings suggest that there are research opportunities to simplify the steps before and after model building. Such improvements would greatly benefit practitioners and should facilitate further adoption

Out of Class Reading, highly recommended

Journal of Machine Learning Research 3 (2003) 1157-1182 Submitted 11/02; Published 3/03

An Introduction to Variable and Feature Selection

Isabelle Guyon ISABELLE@CLOPINET.COM
Clopinet
955 Creston Road
Berkeley, CA 94708-1501, USA

André Elisseeff ANDRE@TUEBINGEN.MPG.DE
Empirical Inference for Machine Learning and Perception Department
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany

Editor: Leslie Pack Kaelbling

Abstract

Variable and feature selection have become the focus of much research in areas of application for which datasets with tens or hundreds of thousands of variables are available. These areas include text processing of internet documents, gene expression array analysis, and combinatorial chemistry. The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of

1. **Do you have domain knowledge?** If yes, construct a better set of "ad hoc" features.
2. **Are your features commensurate?** If no, consider normalizing them.
3. **Do you suspect interdependence of features?** If yes, expand your feature set by constructing conjunctive features or products of features, as much as your computer resources allow you (see example in Section 4.4).
4. **Do you need to prune the input variables** (e.g. for cost, speed or understanding reasons)? If no, construct disjunctive features or weighted sums of features (e.g. by clustering or matrix factorization, see Section 5).
5. **Do you need to assess features individually** (e.g. to understand their influence on the system or because their number is so large that you need to first filter)? If yes, use a variable ranking method (Sect 2 and Sect 7.2); else, do it anyway to get baseline results.
6. **Do you need a predictor?** If no, stop.
7. **Do you suspect your data is "dirty"** (has a few meaningless input patterns and/or noisy outputs or wrong class labels)? If yes, detect outlier examples using the top ranking variables obtained in step 5 as representation; check and/or discard them.
8. **Do you know what to try first?** If no, use a linear predictor. Following the ranking of step 5, construct a sequence of predictors of same nature using increasing subsets of features. Can you match or improve performance with a smaller subset? If yes, try a non-linear predictor with that subset.
9. **Do you have new ideas, time, computational resources, and enough examples?** If yes, compare several feature selection methods, including your new idea, correlation coefficients, backward selection and embedded methods (Sect 4). Use linear and non-linear predictors. Select the best model (Section 6).
10. **Do you want a stable solution** (to improve performance and/or understanding)? If yes, subsample your data and redo your analysis for several "bootstraps" (Section 7.1).

Motivation: Real world examples

Example (1)

- Astro-particle physics dataset
  - Binary classification
  - 4 features; Training set: 3000; Test set: 4000
  - Direct use of SVMs gives 67% accuracy on Test set
- Using proper data transformation gives 96% accuracy on test set
- Proper tuning of hyper-parameters gives 97% accuracy on test set

Lesson: Correct data transformation is important!

Motivation: Real world examples

Example (2): KDD Cup 2001

- Detect binding of proteins to Thrombin
  - Binary classification (imbalanced: only 2% positive)
  - 139,351 features; Training set: 1951; Test set: 636
- Metric: Mean of accuracies on positive and negative classes
- SVM teams used all features. They all got <60% mean accuracy
- Winner selected a subset of 200 features, trained a decision tree and got 68.44% mean accuracy
- Post-competition, one SVM team trained an SVM with a similar feature selection to get 83% mean accuracy.

Lesson: A model that uses lots of features can turn out to be very sub-optimal, however well it is designed!

Motivation: Real world examples

Example (3)

- Splice dataset (Detect intron-exon boundary)
  - Binary classification
  - 60 features; Training set: 1000; Test set: 2175
  - Direct use of SVMs gives 13% error rate on Test set

- Using proper feature selection gives 5% error rate on test set

Lesson: Feature selection can be crucial even when the number of features is small!

Motivation: Real world examples

Example (4)

- German credit dataset (Predict a customer's credit worthiness)
  - Binary classification
  - 21 features; Training set: 500; Test set: 500

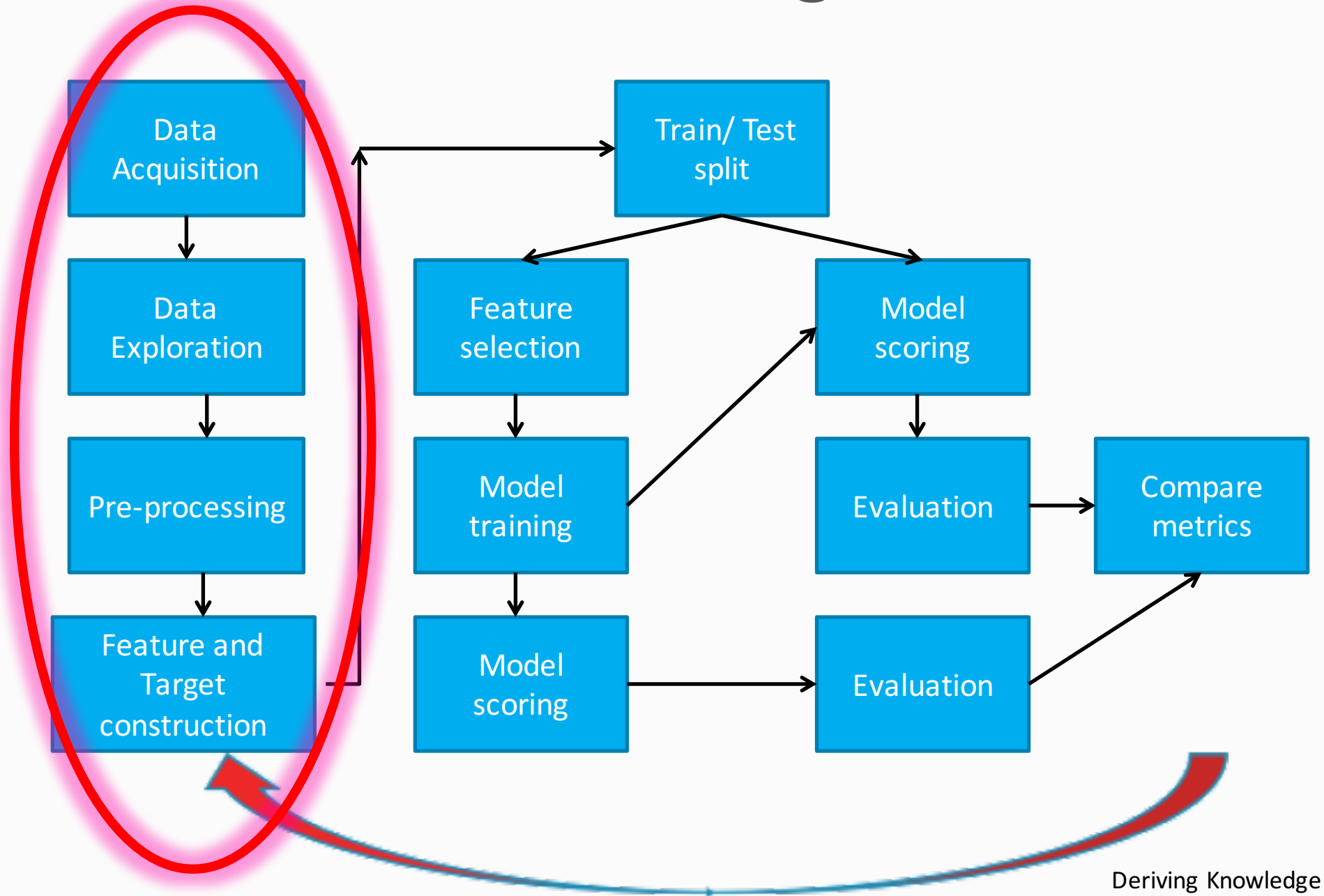
- Decision tree design in *Knime* gives 32.1% error rate on Test set

- Decision tree design in *tLC* gives 24% error rate on Test set

Lesson: Variations of the same ML method can give vastly different performances!

Data Wrangling

Steps in Predictive Modeling Process



Data Acquisition

Depends on the domain

- Data can be
  - Sampled
    - Uniformly (e.g. one out of every X data points is recorded)
    - Stratified (e.g. all outages and a sample of normal operations is recorded)
    - Subject to an earlier predictive model
      - E.g. credit card defaults are subject to prior model of credit scores
  - Missing attribute values
  - Missing records (e.g. some sensors have broken down)
  - Misleading (e.g. sensors saturated and show the same value)
  - Biased (e.g. sensors in a portion of field may be ineffective)
  - Often dependent on the problem itself
    - E.g. in drug effectiveness studies, experimental drugs are only administered to critically ill patients

Data Exploration

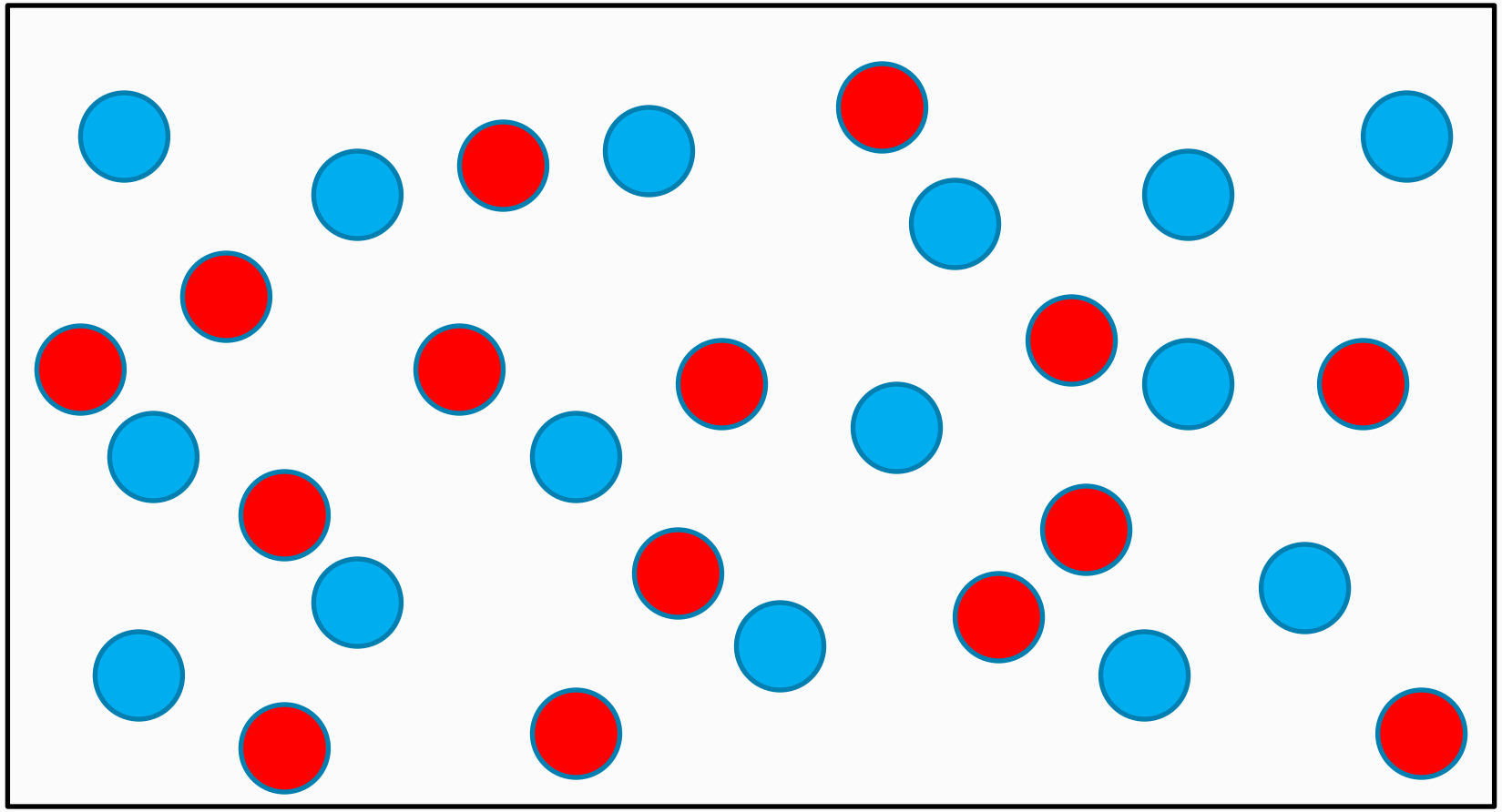
Critical step in the modeling process

- Important to understand the
  - Quality and Quantity of data
  - Statistical properties of variables
  - Relationships between variables
  - Missing values
  - Biases in the data acquisition

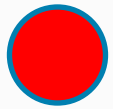
Modeling Dataset

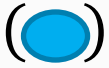
- Split the dataset into a training and testing subset.
- Splits can be
  - Random
    - Each data record is randomly assigned to either train or test
  - Stratified
    - Assign so that each split (train/test) has the same relative proportions of each pair of $(\boldsymbol{x}, \boldsymbol{y})$ E.g.,
      - 67% of records with $(x = 1)$ and $(\boldsymbol{y} = 0)$ go to train
      - 33% of records with $(x = 1)$ and $(\boldsymbol{y} = 1)$ each go to test
  - Temporal datasets can be split either:
    - Horizontal
      - Random over time
    - Vertical
      - Split at a point in time

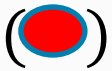
Modeling Dataset



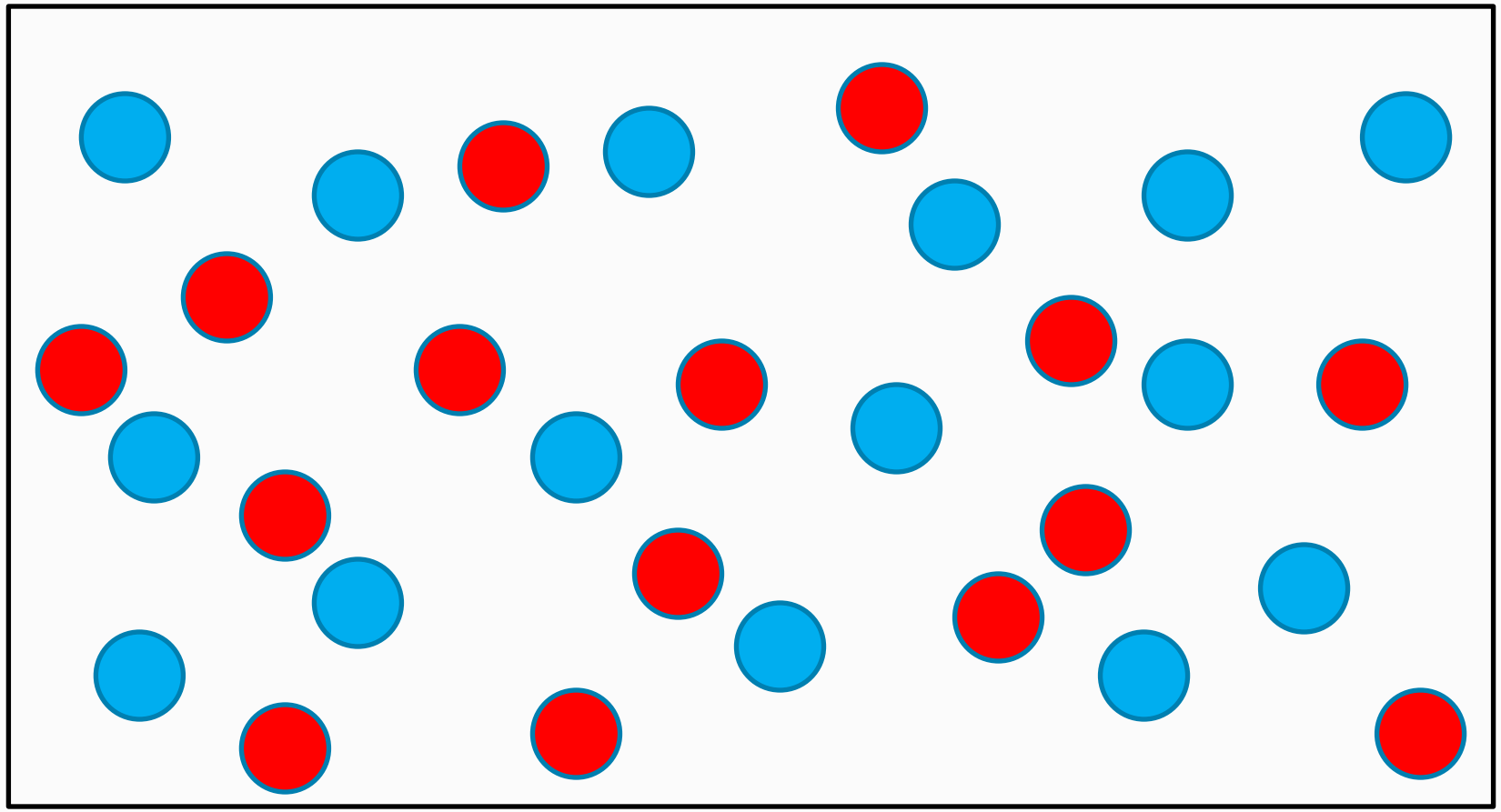
15 

15 

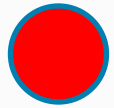
$P($  $) = 0.5$

$P($  $) = 0.5$

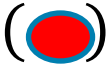
Train/Test Split: Random



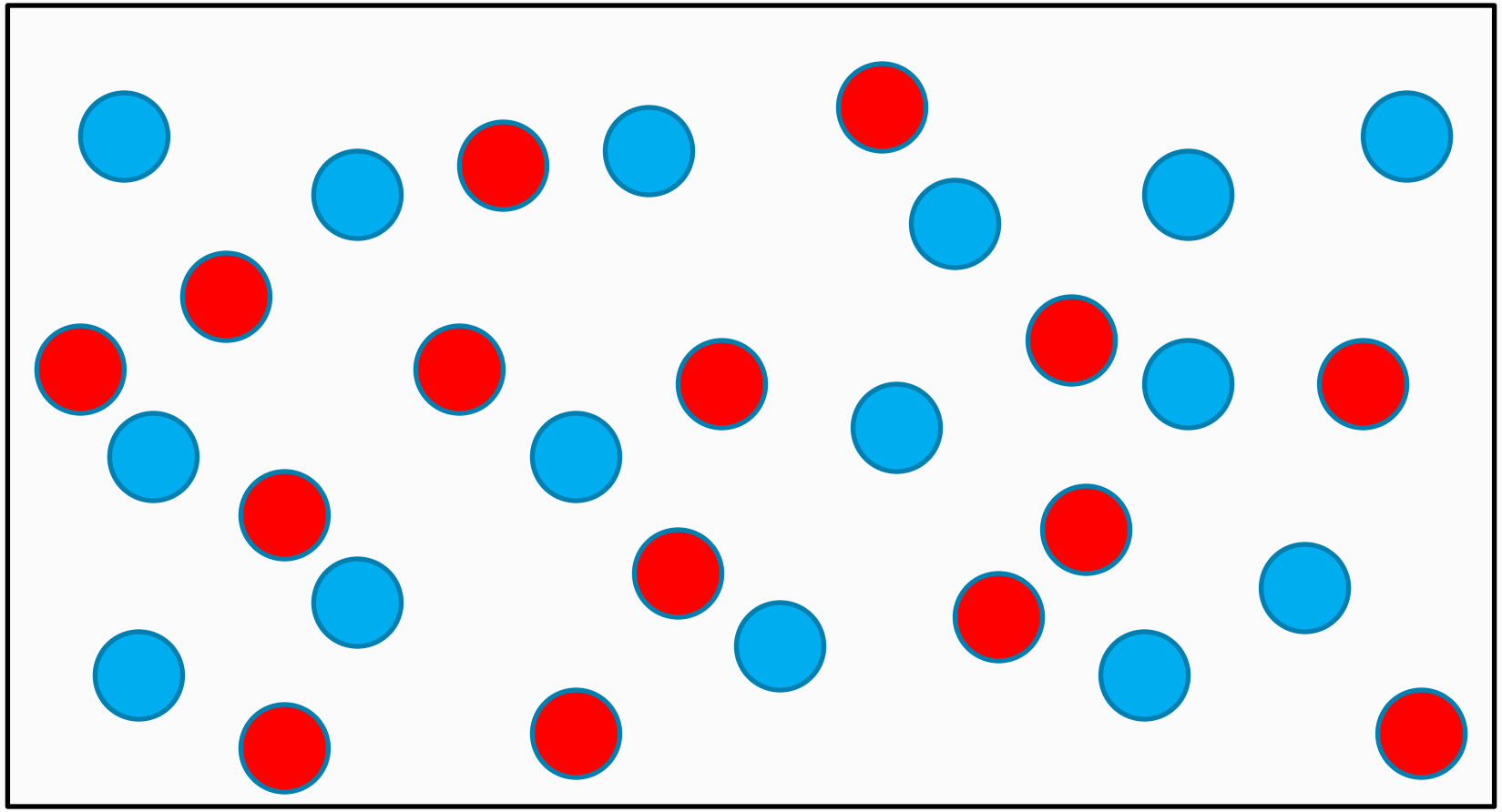
7 

13 

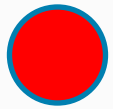
$P($  $) = 0.35$

$P($  $) = 0.65$

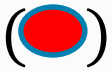
Train/Test Split: Stratified



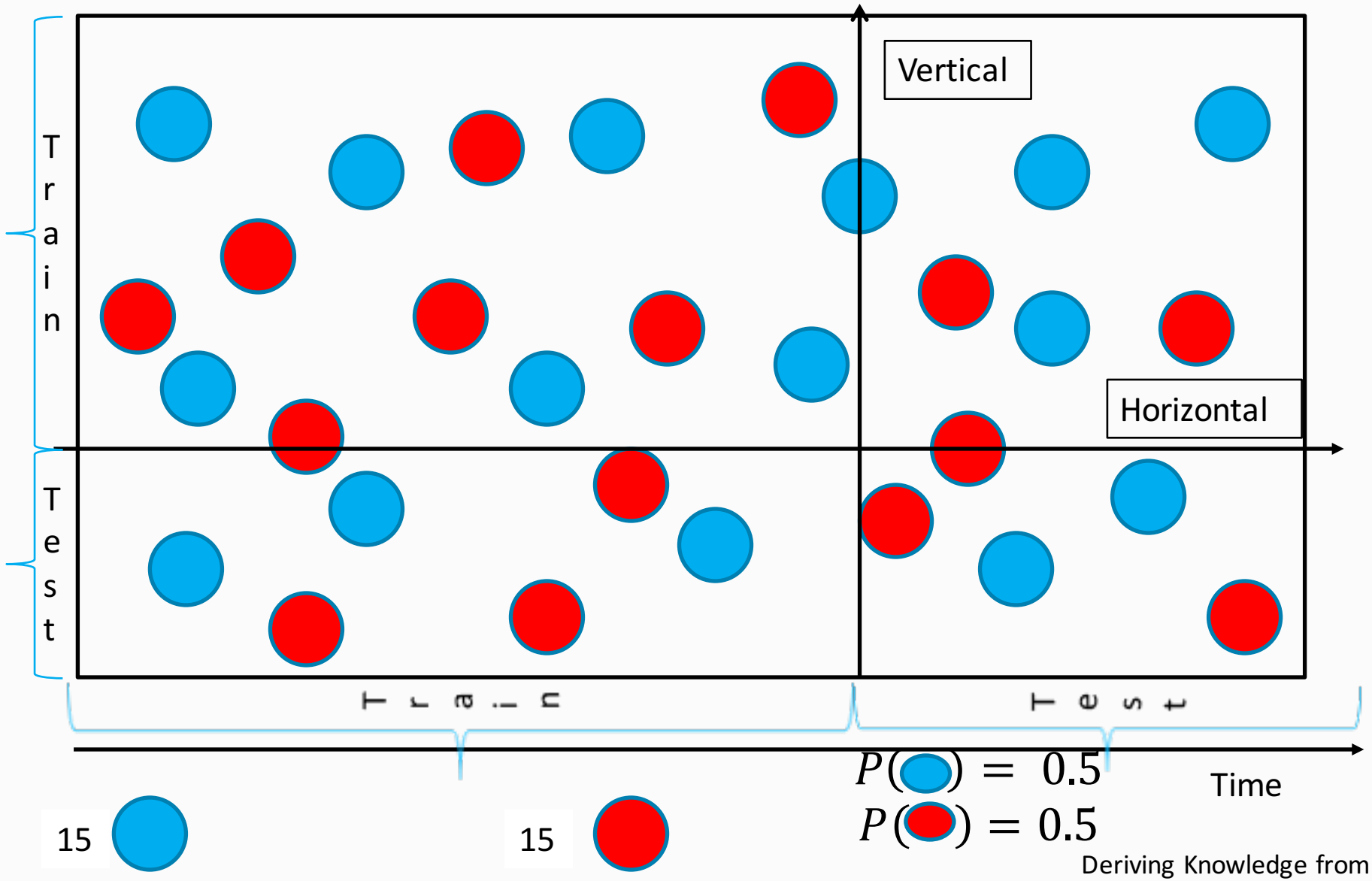
10 

10 

$P($  $) = 0.5$

$P($  $) = 0.5$

Train/Test Split: Temporal Splits



Data Cleaning

1. Missing values
2. Outliers
3. Coding
4. Constraints

Data Cleaning

Missing values – *UCI machine learning repository, 31 of 68 data sets reported to have missing values. "Missing" can mean many things…*

MAR: "Missing at Random":

– usually best case

– usually not true

Non-randomly missing

Presumed normal, so not measured

Causally missing

– attribute value is missing because of other attribute values (or because of the outcome value!)

Dealing With Missing Data

Throw away cases with missing values

- in some data sets, most cases get thrown away
- if not missing at random, throwing away cases can bias sample towards certain kinds of cases

Treat “missing” as a new attribute value

- what value should we use to code for missing with continuous or ordinal attributes?
- if missing causally related to what is being predicted?

Impute (fill-in) missing values

- once filled in, data set is easy to use
- if missing values poorly predicted, may hurt performance of subsequent uses of data set

Missing Values: Imputing

Fill-in with mean, median, or most common value

Predict missing values using machine learning

Expectation Minimization (EM):

- Build model of data values (ignore missing values)
- Use model to estimate missing values
- Build new model of data values (including estimated values from previous step)
- Use new model to re-estimate missing values
- Re-estimate model
- Repeat until convergence

Missing Values: Potential Problems

Imputed values may be inappropriate:

- in medical databases, if missing values not imputed separately for male and female patients, may end up with male patients with 1.3 prior pregnancies, and female patients with low sperm counts
- many of these situations will not be so humorous/obvious!

If some attributes are difficult to predict, filled-in values may be random (or worse)

Some of the best performing machine learning methods are impractical to use for filling in missing values (neural nets)

Beware of coding - reliably detect missing cases can be difficult

Data Cleaning

Outliers – *may indicate 'bad data' or it may represent something scientifically interesting in the data…*

Simple working definition: an outlier is an element of a data sequence S that is inconsistent with expectations, based on the majority of other elements of S.

Sources of outliers

- Insurance company sees niche of sports car enthusiasts, married boomers with kids and second family car. Low risk, lower rate to attract. Simple case where outlier carries meaning for modeling…

Data Cleaning

Outliers – *may indicate 'bad data' or it may represent something scientifically interesting in the data…*

Outliers can distort the regression results.

When an outlier is included in the analysis, it can pull the regression line towards itself.

This can result in a solution that is more accurate for the outlier, but less accurate for all the other cases in the data set.

Data Cleaning

Outliers – *may indicate 'bad data' or it may represent something scientifically interesting in the data…*

Identify outliers

- Question origin, domain knowledge invaluable
- Dispersion – *"spread" of a data set, departure from central tendency, use a box plot…*

Deal with outliers

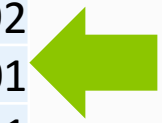
- Winsorize – Set all outliers to a specified percentile of the data. Not equivalent to trimming, which simply excludes data. In a Winsorized estimator, extreme values are instead replaced by certain percentiles (the trimmed minimum and maximum). Same as clipping in signal processing.

{92,19,**101**,58,**1053**,91,26,78,10,13,**-40**,**101**,86,85,15,89,89,28,-5,41}

Winsorize

-40
-5
10
13
15
19
26
28
41
58
78
85
86
89
89
91
92
101
101
1053

10th percentile



90th percentile

{92,19,**101**,58,**1053**,91,26,78,10,13,**-40**,**101**,86,85,15,89,89,28,**-5**,41}

Mean =101.5

{92,19,**101**,58,**101**,91,26,78,10,13,**-5**,**101**,86,85,15,89,89,28,**-5**,41}

Mean = 55.65

Data Cleaning

Outliers – *may indicate 'bad data' or it may represent something scientifically interesting in the data…*

Identify outliers

- Question origin, domain knowledge invaluable
- Dispersion – *"spread" of a data set, departure from central tendency, use a box plot…*

Deal with outliers

- Include – Robust statistics, a convenient way to summarize results when they include a small proportion of outliers. A hot topic for research, see NIPS 2010 Workshop, Robust Statistical learning (robustml).

- Data transformation can eliminate the extreme tendency of the outlier e.g. transforming to Log scale converts extreme values to acceptable range

Data Cleaning

Coding – shape and enrich…

Numerical data

- Binning – a mapping to discrete categories;
- Recenter – shift by c where max, min, avg and median shift, the range and standard deviation **will not** shift;
- Rescale – multiply everything by d, all measures change;
- Standard ND – recenter, make mean 0, divide all previous values by SD

Character data

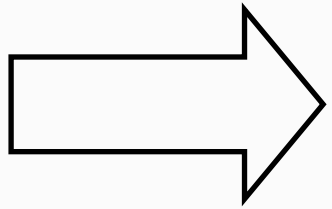
- Lower case
- Spellcheck
- Data extraction (e.g. regular expressions)

Categorical variables

- Non-numeric variables with a finite number of levels
  - E.g. "red", "blue", "green"
- Some ML algorithms can only handle numeric variables
- Solution: 1-to-N coding

1-to-N Coding

feature
red
blue
green
red
red
green
blue



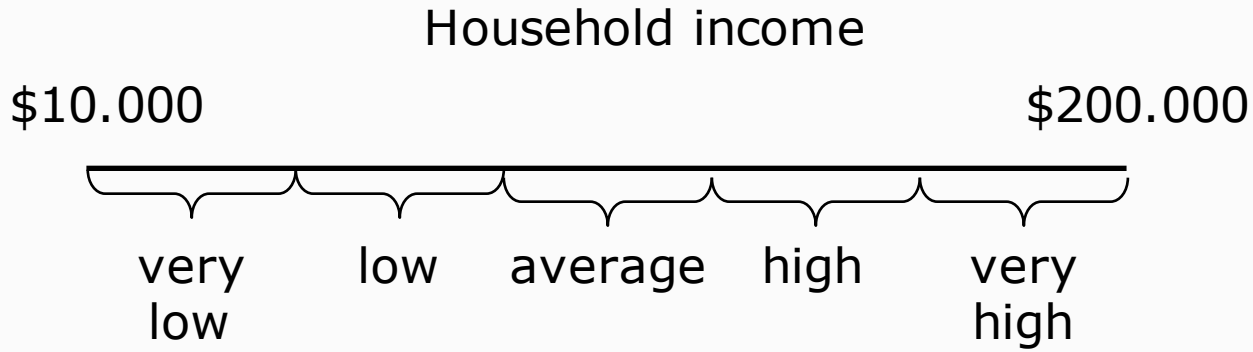
red	blue	green
1	0	0
0	1	0
0	0	1
1	0	0
1	0	0
0	0	1
0	1	0

Scaling of Continuous Variables

- Many ML algorithms rely on measuring the distance between 2 samples
- There should be no difference if a length variable is measured in cm, inch, or km
- To remove the unit of measure (e.g. kg, mph, …) each variable dimension is normalized:
  - subtract mean
  - divide by standard deviation

Output variables

- If ML requires categorical output (continuous output = regression)
- ML methods can be applied by binning continuous output (loss of prediction accuracy)



Feature Selection

- Process of selecting a subset of features that are good predictors of the target

- Useful for
  - Controlling complexity of model
  - Speed up model learning without reducing accuracy
  - Improve generalization capability

Feature Selection, **3 types of methods**

Filter Methods, select a subset of features before training a model, e.g.

- Correlation with target,
- Mutual Information between feature and target

- *Simple to implement, and have reasonable performance*

Wrapper Methods, search combination of feature space by training and evaluating model using a subset of features, e.g.

- Forward, backward, step-wise feature selection,
- Genetic algorithms.

- *Computationally expensive and prone to over-fitting*

Embedded Methods, feature subset is chosen as part of model training, e.g.

- LASSO (L-1) regression, Regularized **decision trees, random forests**

- *Typically robust to over-fitting, but has hyper parameters that will need to be fit using a validation data*

Feature Selection and Engineering

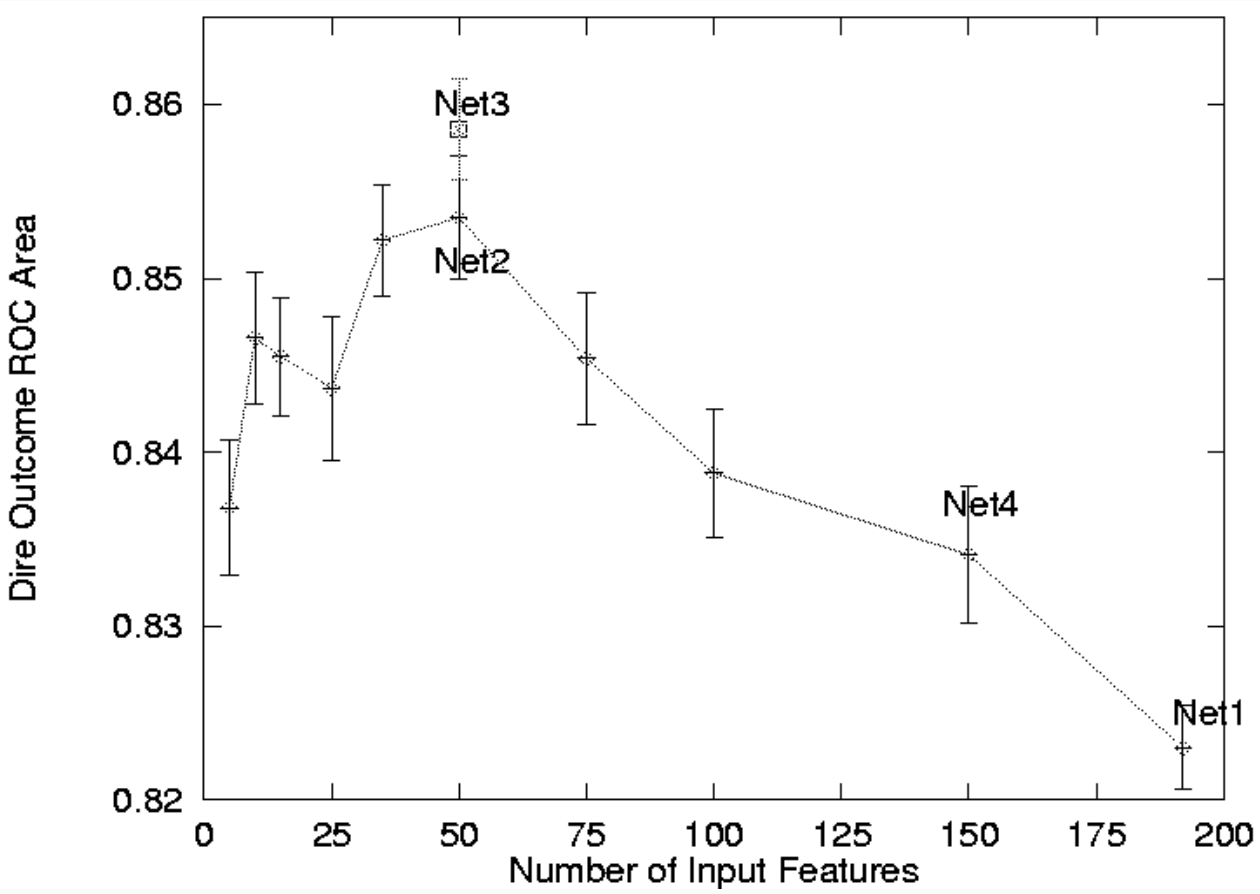
Data Integration

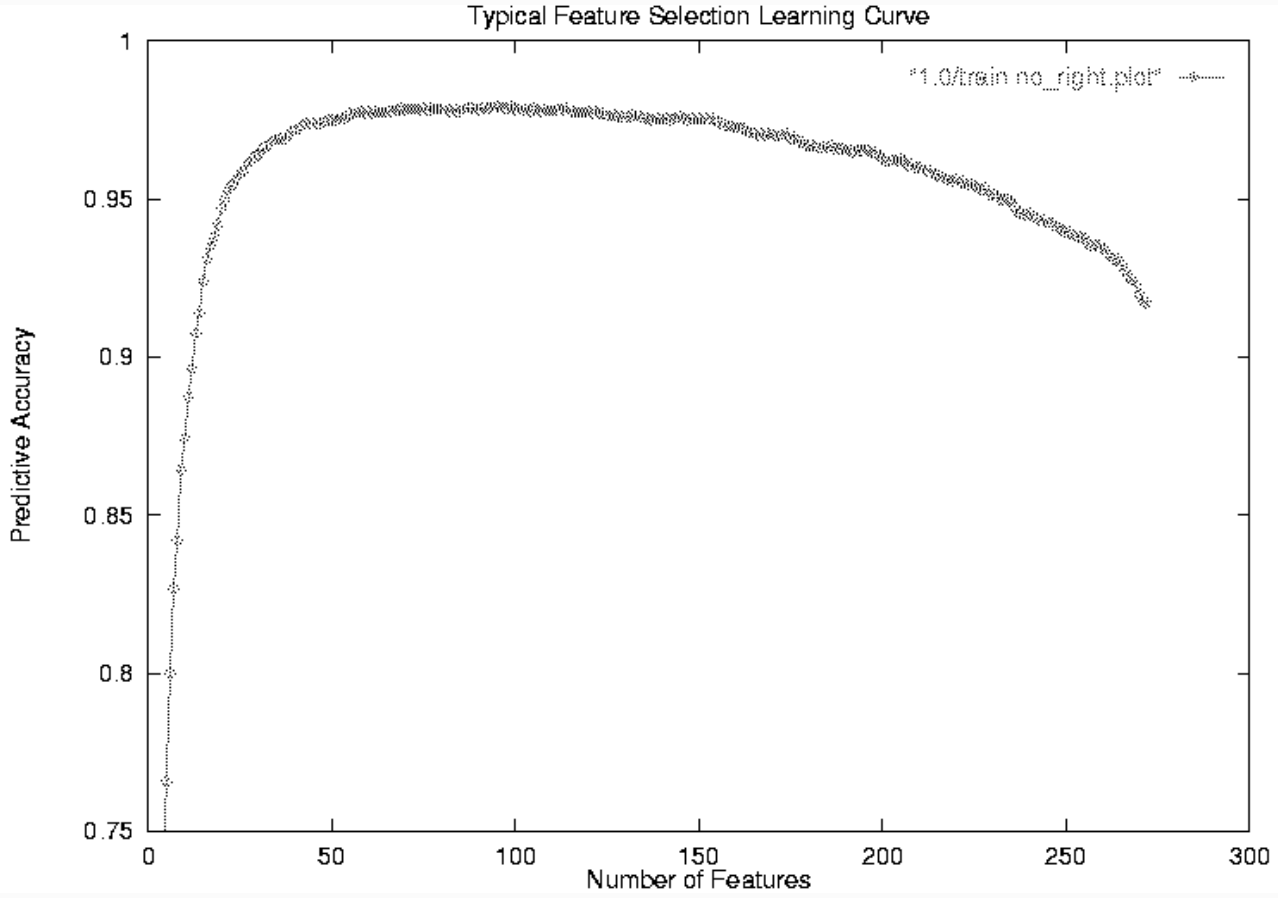
Most learning methods implicitly do feature selection:

- **Decision Trees**: use info gain or gain ratio to decide what attributes to use as tests. Many features don't get used.
- **Neural nets**: backpropagation learns strong connections to some inputs, and near-zero connections to other inputs.
- **kNN (any similarity based learning)**: weights in Weighted Euclidean Distance determine how important each feature is. Weights near zero mean feature is not used.
- **SVMs**: maximum margin hyperplane may focus on important features, ignore irrelevant features.

So why do we need feature **selection**?

Feature Selection and Engineering





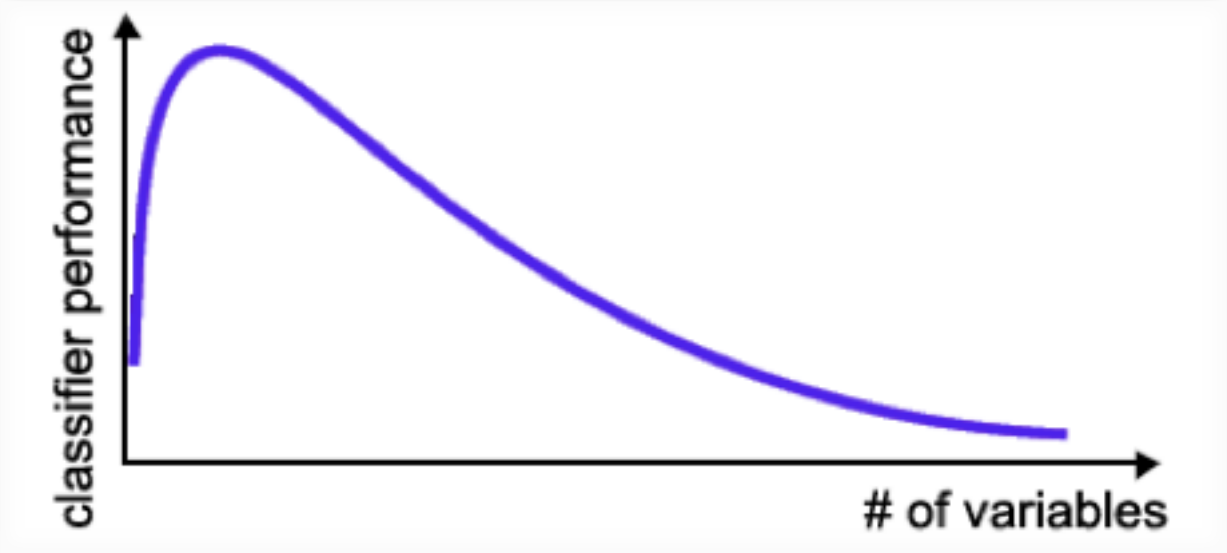
Too many redundant, irrelevant features confuse learner;
Faster training, better performance, simpler models…

Feature Selection and Engineering

Curse of Dimensionality

- The required number of samples (to achieve the same accuracy) grows exponentially with the number of variables!
- In practice: number of training examples is fixed!

  the classifier's performance will degrade for a large number of features!



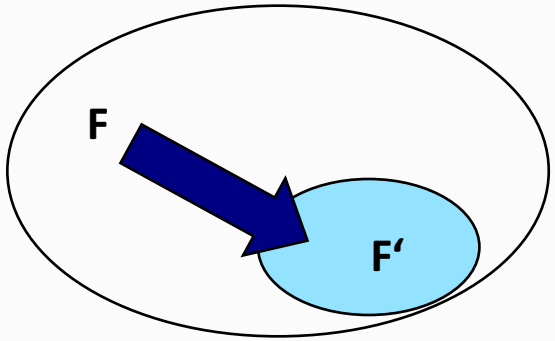
In many cases the information lost by discarding variables is made up for by a more accurate mapping/sampling in the lower-dimensional space !

Feature Selection

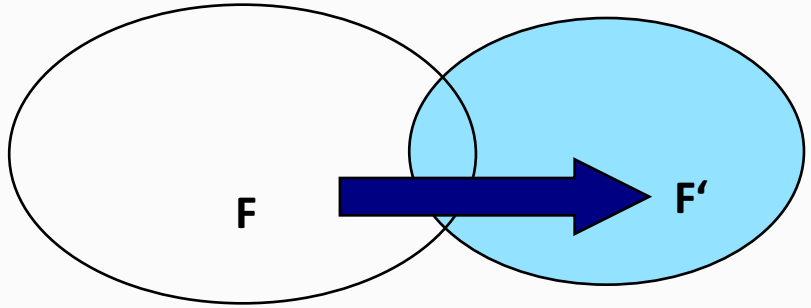
Simple Definition

Given a set of features $F = \{f_1, \ldots, f_i, \ldots, f_n\}$ the **feature selection problem** is to find a subset $F' \subseteq F$ that "*maximizes the learners ability to classify patterns*". **Feature extraction** creates new features F → F'…

$$\{f_1, \ldots, f_i, \ldots, f_n\} \xrightarrow{f.selection} \{f_{i_1}, \ldots, f_{i_j}, \ldots, f_{i_m}\}$$



Feature extraction creates new features F → F'…



$$\{f_1, \ldots, f_i, \ldots, f_n\} \xrightarrow{f.extraction} \{g_1(f_1, \ldots, f_n), \ldots, g_j(f_1, \ldots, f_n), \ldots, g_m(f_1, \ldots, f_n)\}$$

Feature Selection and Engineering

Optimality?

In theory the goal is to find an optimal set of features, one that maximizes the scoring function…

In real world applications this is usually not possible

- For most problems it is computationally intractable to search the whole space of possible feature subsets
- One usually has to settle for approximations of the optimal subset
- Most of the research in this area is devoted to finding efficient search-heuristics

Dealing with Nominal Attributes

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	78	false	yes
rain	70	96	false	yes
rain	68	80	false	yes
rain	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rain	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rain	71	80	true	no

Attributes:

- Outlook (overcast, rain, sunny)
- Temperature real
- Humidity real
- Windy (true, false)
- Play (yes, no)

Standard Spreadsheet Format

OutLook	OutLook	OutLook	Temp	Humidity	Windy	Windy	Play	Play
overcast	rain	sunny			TRUE	FALSE	yes	no
0	0	1	85	85	0	1	1	0
0	0	1	80	90	1	0	0	1
1	0	0	83	78	0	1	1	0
0	1	0	70	96	0	1	1	0
0	1	0	68	80	0	1	1	0
0	1	0	65	70	1	0	0	1
1	0	0	64	65	1	0	1	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

Normalization

- Min-max normalization: linear transformation from v to v’
  - v’ = (v – min)/((max - min)*(newmax - newmin)) + new min
  - Ex: transform $30000 between [10000..45000] into [0..1]
    ==> (30 – 10)/(35(1)) + 0 = 0.5714
- z-score normalization: normalization of v into v’ based on attribute value mean and standard deviation
  - v’ = (v-Mean)/StandardDeviation
- Normalization by decimal scaling
  - moves the decimal point of v by j positions such that j is the minimum number of positions moved so that absolute maximum value falls in [0..1].
  - $v' = v / 10^{j}$
  - Ex: if v ranges between -56 and 9976, j=4 ==> v’ ranges between -0.0056 and 0.9976

Discretization/Binning

Less features, more discrimination ability

- Discretization is used to reduce the number of values for a given continuous attribute
  - usually done by dividing the range of the attribute into intervals
  - interval labels are then used to replace actual data values
- Discretization can also be used to generate concept hierarchies
  - reduce the data by collecting and replacing low level concepts (e.g., numeric values for “age”) by higher level concepts (e.g., “young”, “middle aged”, “old”)

Discretization Methods

- Equal-width (distance) partitioning
  - Divides the range into *N* intervals of equal size: uniform grid
  - The most straightforward, but outliers may dominate presentation
  - Skewed data is not handled well

- Equal-depth (frequency) partitioning
  - Divides the range into *N* intervals, each containing approximately same number of samples

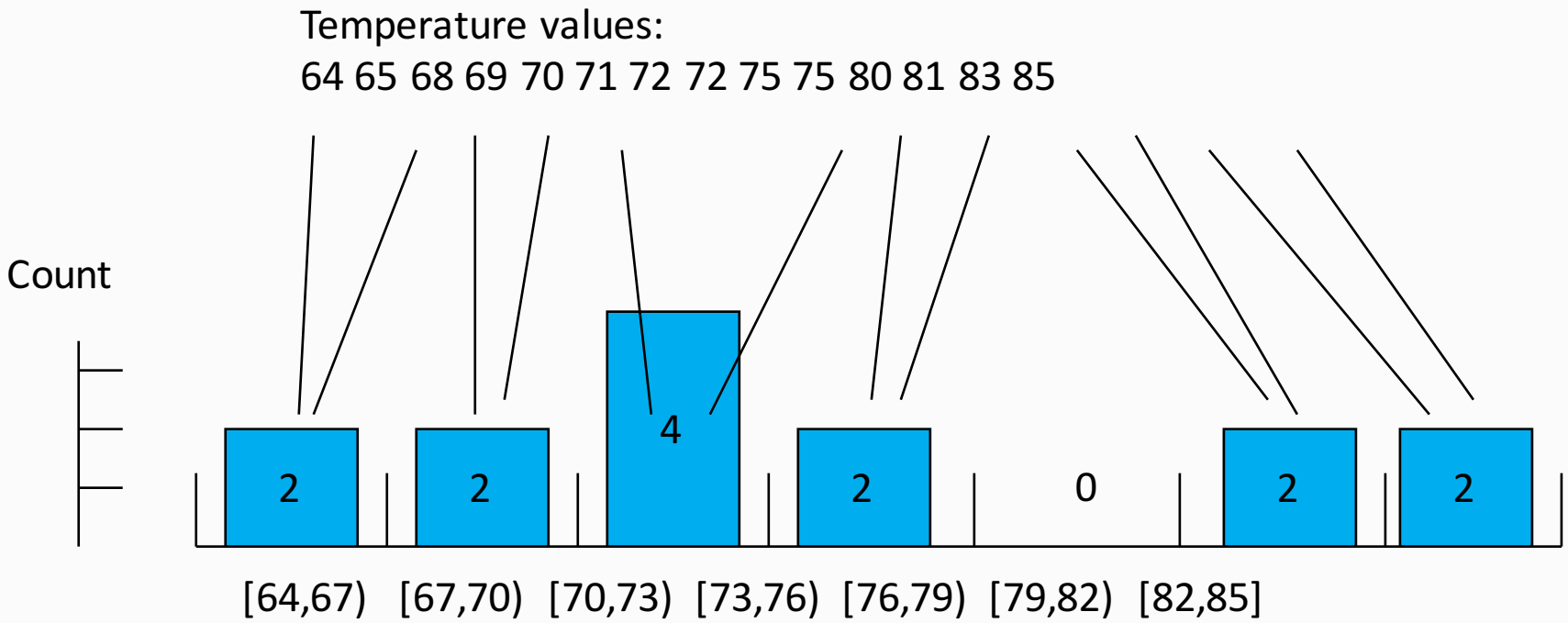
- Class label based partitioning
  - χ^2, CAIM, CAIR Discretization
  - Maximum Entropy Discretization

Equal width partitioning

1. Find the minimum and maximum values for the continuous feature/attribute F_i

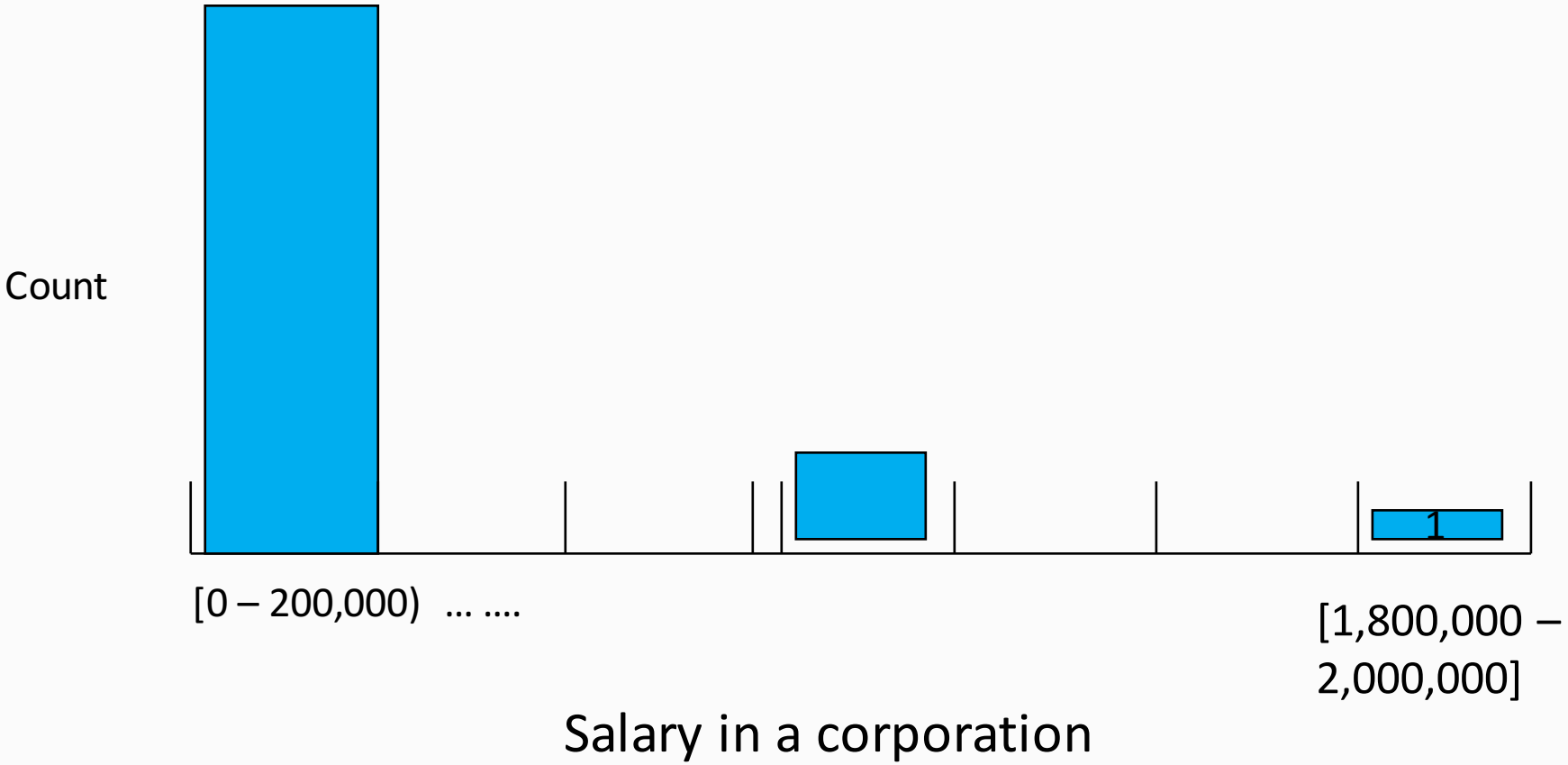
2. Divide the range of the attribute F_i into the user-specified, n_{Fi} ,equal-width discrete intervals

Equal-Width Partitioning



Equal Width, bins Low <= value < High

Equal-Width partitioning can produce clumping



Equal Height partitioning

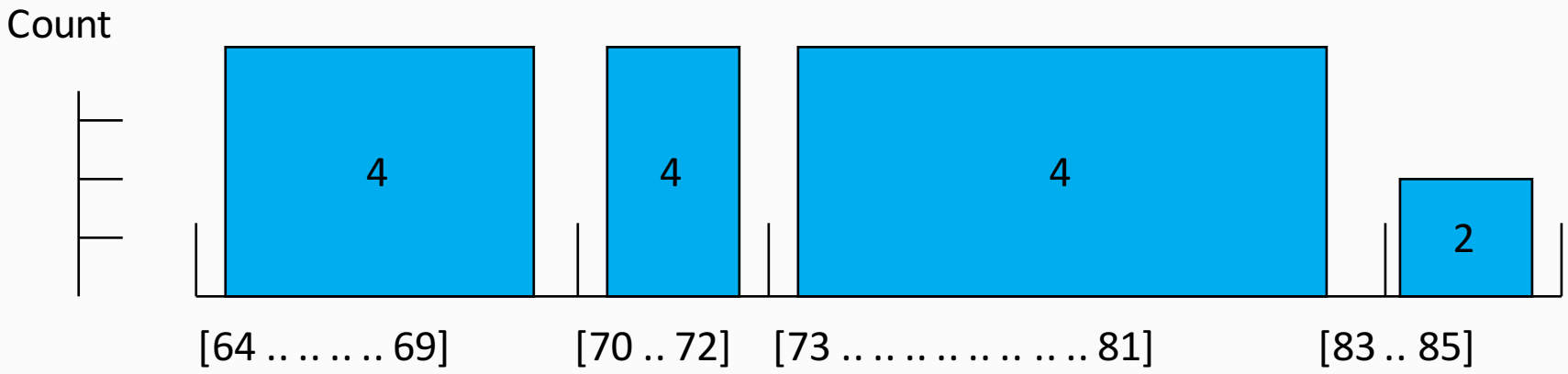
1. Sort values of the discretized feature F_i in ascending order

2. Find the number of all possible values for feature F_i

3. Divide the values of feature F_i into the user-specified n_{Fi} number of intervals, where each interval contains the same number of sorted sequential values

Equal-Height partitioning

Temperature values:
64 65 68 69 70 71 72 72 75 75 80 81 83 85



Equal Height = 4, except for the last bin

Equal-height partitioning: advantages

- Generally preferred because avoids clumping
- In practice, “almost-equal” height binning is used which avoids clumping and gives more intuitive breakpoints
- Additional considerations:
  - don’t split frequent values across bins
  - create separate bins for special values (e.g. 0)
  - readable breakpoints (e.g. round breakpoints)

Derived Variables

- Better to have a fair modeling method and good variables, than to have the best modeling method and poor variables
- Insurance Example: People are eligible for pension withdrawal at age 59 ½. Create it as a separate Boolean variable!
- Advanced methods exist for automatically examining variable combinations, but they can be computationally very expensive!

Special Transformations

Domain expertise, play a hunch in terms of feature discrimination

Example: *Date/Time* attribute

- Time of a day
- Day of the week
- Day of the month
- Month of the year
- Day of the year
- Quarter of the year
- A holiday or not

Which ones to use depends on the prediction problem being solved

- Ex: For prediction of traffic on a freeway, Time of day, Day of the week, A holiday or not etc. will be useful

Data Science

Deriving Knowledge from Data at Scale

That's all for tonight….