Machine Learning

Lecture BigData Analytics

Julian M. Kunkel

julian.kunkel@googlemail.com

University of Hamburg / German Climate Computing Center (DKRZ)

2017-11-24

Disclaimer: Big Data software is constantly updated, code samples may be outdated.

Outline

Data Mining (Knowledge Discovery) [1,35]

Definition

- **Data mining**: process of discovering patterns in large data sets
  - (Semi-)Automatic analysis of large data to identify interesting patterns
  - Using artificial intelligence, machine learning, statistics and databases

Tasks / Problems for data mining

- **Classification**: predict the category of samples
- **Regression**: find a function to model numeric data with the least error
- **Anomaly detection**: identify unusual data (relevant or error)
- **Association rule learning**: identify relationships between variables
- **Clustering**: discover and classify similar data into structures and groups
- **Summarization**: find a compact representation of the data

Terminology for Input Data [1, 40]

- **Sample**: instances (subset) of the unit of observation
- **Feature**: measurable property of a phenomenon (explanatory variable)
  - The set of features is usually written as vector (f1, ..., fn)
- **Label/response**: outcome/property of interest for analysis/prediction
  - Dependent variable
  - Discrete in classification, continuous in regression

Forms of features/labels

- Numeric: a (potentially discrete) number characterizes the property
  - e.g., age of people
- Categorical/nominal: a set of classes
  - e.g., eye color
  - Dichotomous (binary) variable: contains only two classes (Male: Yes/No)
- Ordinal: an ordered set of classes
  - e.g., babies, teens, adults, elderly

Example Data

Imagine we have data about alumni from the university

Field of study	Gender	Age	Succ. exams	Fail. exams	Avg. grade*	Graduate	Dur. studies
CS	M	24	21	1	2.0	Yes	10
CS	M	22	5	2	1.7	Enrolled	2
Physics	F	23	20	1	1.3	Enrolled	6
Physics	M	25	8	10	3.0	No	10

- Categorical: field of study, gender, graduate, (favourite colour)
- Numeric: age, successful/failed exams, duration of studies
- Numeric: average grade; Ordinal: very good, good, average, failed

Our goal defines the machine learning problem

- Predict if a student will graduate $\Rightarrow$ classification
  - Prescriptive analysis: we may want to support these students better
- Predict the duration (in semesters) for the study $\Rightarrow$ regression
- Clustering to see if there are interesting classes of students
  - We could label these, e.g., the prodigies, the lazy, ...
  - Probably not too helpful for the listed features

Terminology for Learning [40]

- **Online learning**: update the model constantly while it is applied
- **Offline (batch) learning**: learn from data (training phase), then apply
- **Supervised learning**: feature and label are provided in the training
- **Unsupervised learning**: no labels provided, relevant structures must be identified by the algorithms, i.e., descriptive task of pattern discovery
- **Reinforcement learning**: algorithm tries to perform a goal while interacting with the environment
  - Humans use reinforcement, (semi)-supervised and unsupervised learning

Overview of Machine Learning Algorithms (Excerpt)

Classification

- k-Nearest neighbor
- Naive bayes
- Decision trees
- Classification rule learners

Regression/Numeric prediction

- Linear regression
- Regression trees
- Model trees

Regression & classification

- Neuronal networks
- Support vector machines

Pattern detection

- Association rules
- k-means clustering
- density-based clustering
- model-based clustering

Meta-learning algorithms

- Bagging
- Boosting
- Random forests

Machine Learning in Practice [1]

Process / Phases

1. Data collection: combining data into a single source
2. Data exploration and preparation: inspection and data cleanup
3. Model training: depending on machine learning task choose algorithm
4. Model evaluaton: check accuracy of the model
5. Model improvement: if necessary try to improve accuracy by utilizing advanced methods or providing additional input

Cross Industry Standard Process for Data Mining [39]

CRISP-DM is a commonly used methodology from data mining experts

Phases

- **Business understanding**: business objectives, requirements, constraints; converting the problem to a data mining problem
- **Data understanding**: collecting initial data, exploration, assessing data quality, identify interesting subsets
- **Data preparation**: creation of derived data from the raw data (data munging)
- **Modeling**: modeling techniques are selected and applied to create models, assess model quality/validation
- **Evaluation** (wrt business): check business requirements, review construction of the model(s), decide use
- **Deployment**: applying the model for knowledge extraction; creating a report, implementing repeatable data mining process

Source: Kenneth Jensen [38]

Normalization of Data [1, p. 72]

- Several algorithms require that numeric variables are normalized
  - The numbers of the feature vector are treaded identically
  - Example: in the features (age, income, is_male), income is $>>$ age
- Treatment: scale features similar, e.g., all values between 0 and 1

Min-Max normalization

- $X_{new} = \frac{X - min(X)}{max(X) - min(X)}$

Z-Score standardization

- $X_{new} = \frac{X - mean(X)}{StdDev(X)}$
- Especially useful for normal distributed data

Dummy Coding [1]

- Problem: distance is not defined for categorical data
  - Regression does not make sense for categorical data
- Dummy coding transforms N classes into N-1 dummy (proxy) variables
  - 0 indicates instance is of given class
  - 1 indicates use other class
  - The last class is the reference class
- Dummy coding works well for features
  - Independent prediction of several “feature” classes must be resolved, i.e., more than one class is predicted as 1

Example

- Color: Red, blue, green
- Dummy variables: color_red, color_blue, color_green
- Color green could be omitted and be the reference

Treating Missing Data [32, 33, 1, p.300]

- Problem: a feature is not available for an example

Alternatives

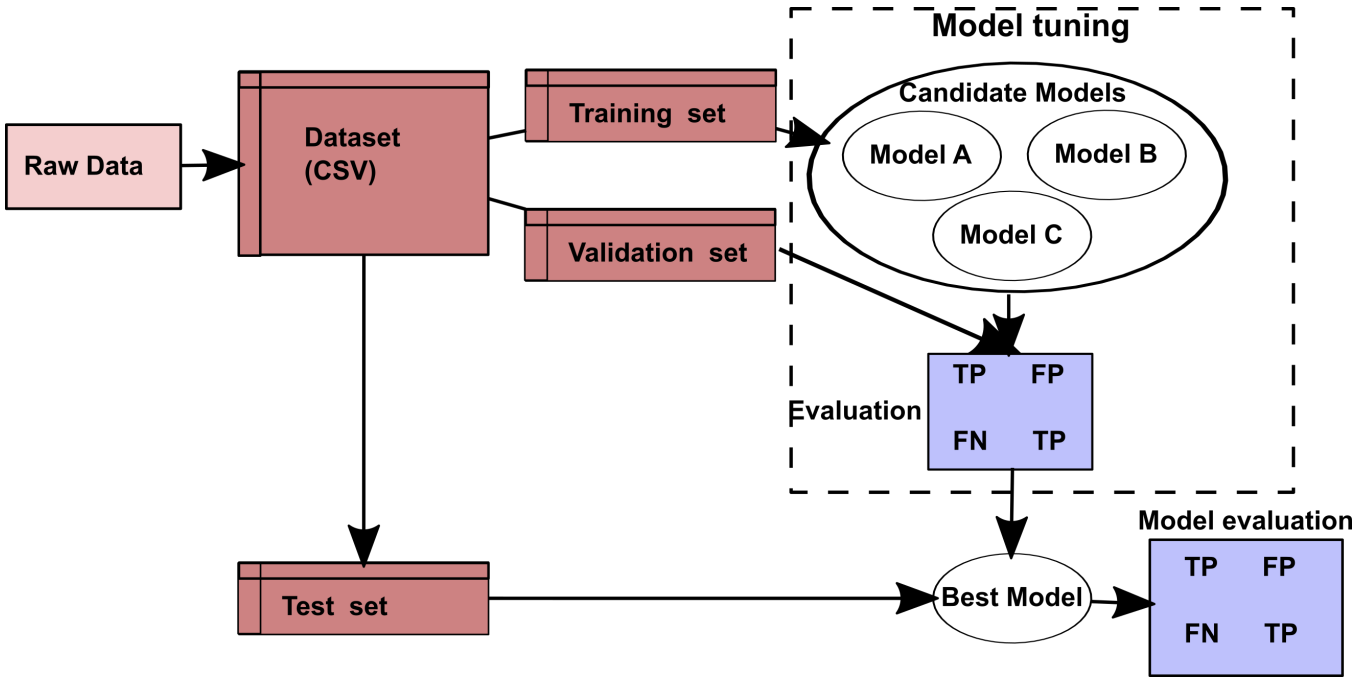
- Deletion: remove examples with missing (N/A) data
  - Problem: we may have many features of which many examples miss one
- Imputation: replace N/A with substitution values
  - Hot-deck imputation: replace value with a random value from similiar entity
  - Last observation carried forward: simply use last observed value
  - Replace with median, mean (of similar entities)
  - Interpolation (or Kriging)
  - Apply a regression model
  - Statistic regression: replace with mean + random variance
- Replacing too many instances may complicate analysis/exploration

Strategy for Learning [40]

- Goal: Learn properties of the population from a sample
- Data quality is usually suboptimal
  - Erroneous samples (random noise, ambivalent data)
  - **Overfitting**: a model describes noise in the sample instead of population properties
  - **Underfitting**: a model ignores small but important patterns
  - **Robust** algorithms reduce the chance of fitting noise
- How accurate is a specific model on the **population**?
  - Should we train a model on our data and check its accuracy on the same?
    - As the model is trained on the data, it should be able to be accurate
    - A lookup table might reproduce the data perfectly but is not useful
  - Resubstitution error: training/testing with the same data
    - Shows how well the model can fit
    - A bad fit can be an indicator for ambivalent/erroneous data
    - A bad fit can also show that the method is not appropriate for the data
    - Personally, I always do check model quality first on the resubstitution error

Systematic Testing with the Holdout Method

- Split data into **training** (50%), **test** (25%) and **validation** (25%) set
  - Training set: build/train model from this data sample
  - Validation set: check model quality and refine the models
  - Test set: check final model accuracy on this set (expected accuracy)
- Once the best model is identified, train it on complete data set



Holdout method. The figure is based on [1, p.337].

Supplementary Strategies

Problems

- Sometimes we have not sufficient training samples
- Suboptimal selection of training samples may cause problems
  - Classification: some classes may have only a few training samples

k-fold cross validation

- Prevents cases in which we partition data suboptimally
- See next slide

Leave-one-out cross validation

- Builds model with all elements except one
- Compute model accuracy on the last (test) element
- Repeat the process for each element

k-fold cross validation

1. Split data into k sets
2. For all permutations: train from k-1 sets, validate with remaining set
3. Compute average error metrics

Example with the iris data set

```
1  library(cvTools)
2  set.seed(123) # initialize random seed generator
3
4  data(iris)
5  # create 10 folds
6  f = cvFolds(nrow(iris), K=10, R=1, type="random")
7
8  # retrieve all sets
9  for (set in 1:10){
10   validation = iris[ f$subsets[f$which == set] ,] # 135 elements
11   training = iris[ f$subsets[f$which != set], ] # 15 elements
12
13   # TODO Now build your model with training data and validate it
14   # TODO Build error metrics for this repeat
15 }
16
17 # Output aggregated error metrics for all repeats
18
19 # Some packages perform the k-cross validation for you
```

Creating only one training set

```
1 # create two classes, train and validation set
2 mask = sample(2, nrow(iris), repl=T, prob=c(0.9,0.1))
3 validation = iris[mask==1, ]
4 training = iris[mask==2, ]
```

Stratified sampling [11]

- Stratification: dividing the population into homogeneous subgroups before sampling
  - e.g., for clinical trials: people (not) having a disease and smokers, 4 groups
  - Draw the same number of random samples from each group
- If we have the data already:
  - Split the observed samples into classes and distribute these instances across traing/test/validation set
  - Alternatively: Draw the same number of elements from each class

Example problem: class imbalance problem [1, p. 312]

- Consider we have test A for a disease
- We know that 990 people are healthy and 10 people have the disease
- Assume the test always reports “healthy”
  - Is this a good test? It is correct in 99% of cases!

$\Rightarrow$ A careful assessment of model performance is needed

Evaluating Model Performance

- Idea: compare true value with predicted “value” on the training data
- Algorithms return the predicted class/numeric value
  - Classification returns the class (e.g., color, healthy: yes/no)
  - Regression the numeric value
- Algorithms may return a probability of the prediction
  - Likelihood that the value was correct on the training/test set
  - Sometimes the choice is tight, i.e., 49% class A vs. 51% class B
  - We may skip such results and say we cannot determine the class!
- There are different metrics to assess the quality of the model
  - Metrics depend on the problem: classification vs. regression

Assessing Correctness of Classification Models

Confusion matrix

- Visualizes the performance of the classification
- Shows count in observation (row) and prediction class (column)

	Class A	Class B	Class C
Class A	**AA**	AB	AC
Class B	BA	**BB**	BC
Class C	CA	CB	**CC**

- Often one class is of interest (e.g., class A)
  - True positive (TP): observation is true, predicted as true (AA)
  - False positive (FP): observation is false, prediction is true (BA, CA)
  - True negative (TN): observation is false, predicted as false (BB, CC)
  - False negative (FN): observation is true, prediction is false (else)
- There are useful metrics defined on these values
  - Accuracy, error rate, sensitivity, specifity, precision, recall
  - Kappa statistic: correctness vs. random correctness
  - F-measure (F-score): weights precision and recall equally

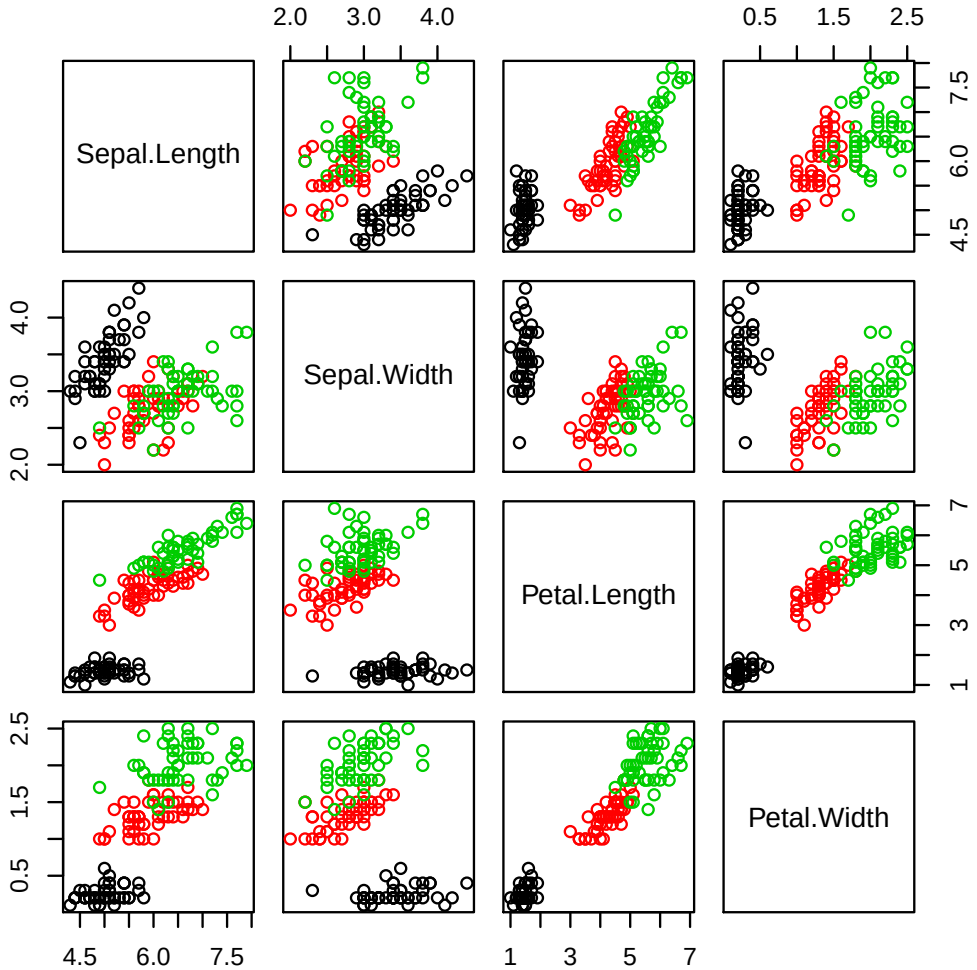
Evaluating Model Performance for Numerical Data

- **Residual**: difference of observation[1] and estimated (predicted) value
  - Residual (error): $e = o - e$
  - In our test/validation set we have n samples for which we compute residuals
- **Mean absolute error**: $MAE = \frac{1}{n} \sum_{i=1}^{n} |e_i|$
- **Mean square error**: $MSE = sqrt(\frac{1}{n} \sum_{i=1}^{n} (o_i - e_i)^2)$
- **Mean absolute percentage error**: $MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{o_i - e_i}{o_i} \right|$
- We may compute correlation of observation and estimation

[1] Also called actual value, but I prefer observation since we do not know if it is the true value.

Classification: Supervised Learning

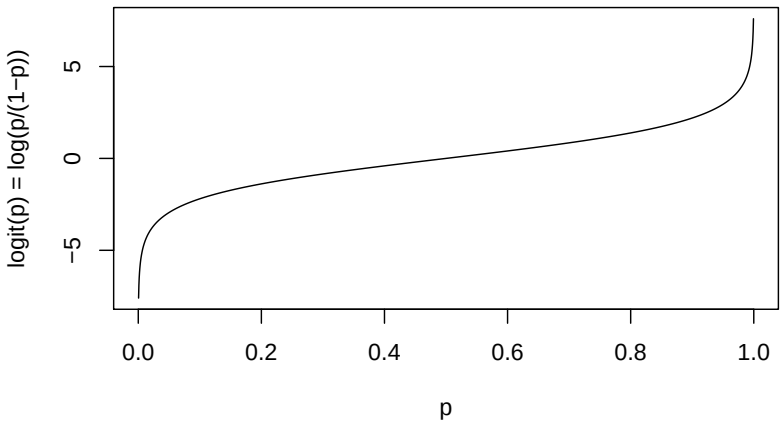
- Goal: Identify/predict the class of previously unknown instances



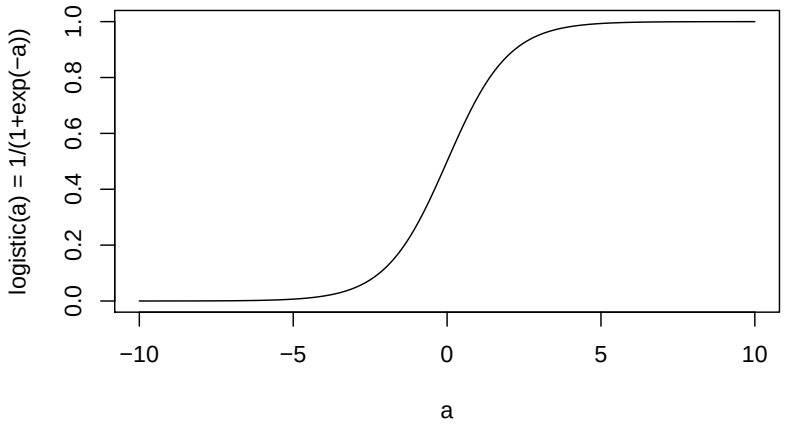
Each class (flower type) is visualized in its own color

Generalized Linear Model (GLM) [34]

- LM expects numeric data and normal distribution of error values
- GLM is a linear model that map the response via a link function
  - e.g., improve accuracy for binary result variable by computing a probability



Plot of logit(p)



$\text{logit}^{-}1(\alpha) = \text{logistic}(\alpha)$(for binary)

Example in R

```
1 d$female = (d$gender == "female") # convert into dichotomous var
2 d$grade = factor(d$grade) # convert variable into a categorical var
3
4 m = glm(formula = graduate ~ female + age + grade + exams_succ + exams_fail,
      ↪ family=binominal(link="probit"), data=d)
```

k-Nearest Neighbor (k-NN) [1]

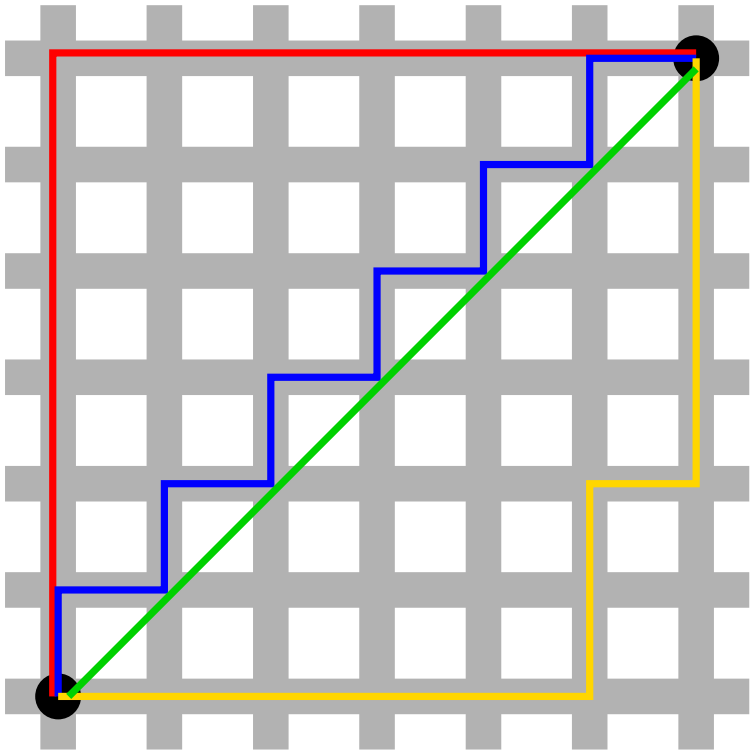
- Prediction: compute distance of new sample to k nearest samples
  - Majority of neighbors vote for new class
- Strengths:
  - Simple and effective supervised learning algorithm
  - No assumption about data distribution
  - Fast training
- Weaknesses:
  - Does not create a model
  - Parameter k needs to be set
  - Slow classification
  - Normalization (min/max) required, nominal features and missing data?

Example in R

```
1 library(kknn)
2 m = kknn(Species ~ Sepal.Width + Petal.Length + Petal.Width + Sepal.Length, train=training, test=validation, k=3)
3
4 # Create a confusion matrix
5 table(validation$Species, m$fit)
6 #             setosa versicolor virginica
7 #  setosa          3          0         0
8 #  versicolor      0          7         0
9 #  virginica       0          1         4
```

Supporting Topic: Distance Metrics

- Consider two vectors $v = (v_1, ..., v_n)$ and $w = (w_1, ..., w_n)$
- Euclidean distance: $d(v, w) = \sqrt{(v_1 - w_1)^2 + ... + (v_n - w_n)^2}$
- Manhattan distance: $d(v, w) = |(v_1 - w_1)| + ... + |(v_n - w_n)|$



Red: Manhattan distance. Green: diagonal, euclidean distance. Blue, yellow: equivalent Manhattan distances [12]

Naive Bayes [1]

- Idea: predict class based on probabilities of occurrence in the training
  - e.g., email containing *medication, viagra, shop* is likely Spam
- Based on Bayesian methods
  - $P(A)$: Probability outcome A is observed = count A / count all observations
  - Assume independence: $P(A \cap B) = P(B|A) \cdot P(A) = P(A|B) \cdot P(B)$
  - $P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A) \cdot P(A)}{P(B)}$ (Probability of A under condition B)
- Naive assumptions: independence and equal importance of features
- Classification: $P(C_L|F_1, ..., F_n) = \frac{1}{Z} p(C_L) \prod_{i=1}^{n} p(F_i|C_L)$
- Strengths:
  - Simple, fast and effective
  - Works well with noisy and missing data
  - Small number of training samples required
  - Probability for a prediction can be obtained (confidence)
- Weaknesses:
  - Assumes that all features are equally important
  - Suboptimal for datasets with many numeric features
  - Probabilities are less reliable than predicted classes

Example: Spam Filter [1]

- Goal: Classify an email as Ham or Spam based on text
- $w_i = 1$, if a word occurs in message i, 0 otherwise
- Summarize w_i based on the labels and create tables

Probability table calculated from the training set for each word

	Medication		
Frequency	Yes	No	Total
Spam	4	16	20
Ham	1	79	80
Total	5	95	100

	Shop		
Frequency	Yes	No	Total
Spam	3	17	20
Ham	20	60	80
Total	23	77	100

Classification of new e-mails

- $P(Spam|Medication, Shop) = 4/20 \cdot 3/20 \cdot 20/100 = 0.006$
- $P(Ham|Medication, Shop) = 1/80 \cdot 20/80 \cdot 80/100 = 0.0025$

$\Rightarrow P(Spam) = 0.006/(0.006 + 0.0025) = 70.6\%$

Data Pre-Processing [1]

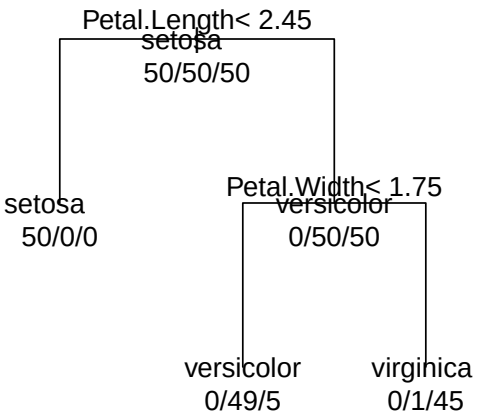
- Problem: predict 0 if a feature is missing in a class level while training
  - e.g., in our Spam classifier a word is never seen in a spam email
  - Observation in practice leads to multiplication with zero
  - Solution: missing data is treated with the Laplace estimator:
    Add 1 to the count of each class-feature to ensure it occurs
  - Other values work too, but ensure that probability for class sum to 1
  - Alternative: Ignore attribute from calculation

	Medication		
Likelihood	Yes	No	Total
Spam	5/22	17/22	22
Ham	2/82	80/82	82
Total	7	97	104

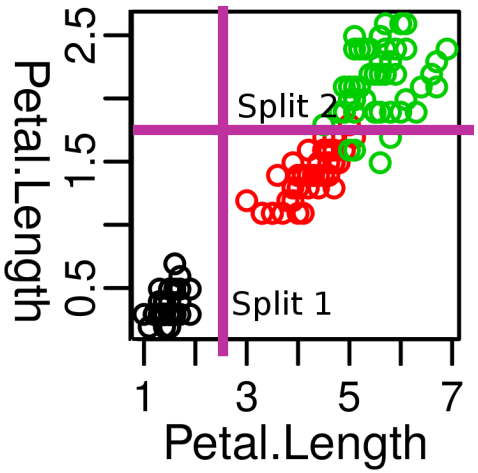
- Predicting numeric features with Naive Bayes
  - Create interval classes (bins) for numeric data
    - e.g., Class 1 are all instances between 0 and 10
    - Selection of cut points should be inspired by data distribution
    - Quantiles are (trivial but) potential cut points
  - Alternative: use a probability density function
    - Training: estimate parameters based on distribution of a class
    - Prediction for x: multiply by PDF(x) (instead of probability of a class)

Decision Trees

- Tree data structures, a node indicates an attribute and threshold
  - Follow left edge if value is below threshold otherwise right
  - Leafs are decisions
  - Can separate data horizontally and vertically
- Classification trees (for classes) and regression trees for continuous vars
- Various algorithms to construct a tree
  - CART: Pick the attribute to maximize information gain of the split
- Knowledge (decision rules) can be extracted from the tree
- Tree pruning: Recursively remove unlikely leafs (reduces overfitting)



(a) Tree for iris data



(b) Split of the values

Decision Trees with R

- Rpart package supports regression (method="anova")
- Classification (with two classes method="poisson" else "class")
- Control object defines requirements for splitting (e.g., observations per leaf, cost complexity (cp) factor)

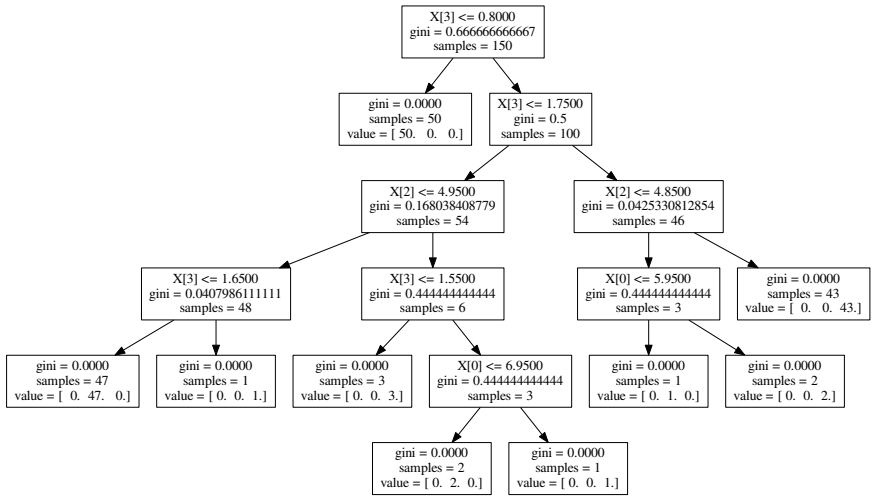
```
1  library(rpart)
2  data(iris) # The iris data (from the slide before9
3  # Create a classification tree based on all inputs
4  m = rpart(Species ~ Sepal.Width + Petal.Length + Petal.Width + Sepal.Length, data=iris, method="class",
5    control = rpart.control(minsplit=5, cp = 0.05)) # require a minimum number of 5 observations
6  summary(m) # print details of the tree
7
8  plot(m, compress=T, uniform=T, margin=0.7) # plot the tree
9  text(m, use.n=T, all=T) # add text to the tree, plot all nodes not only leafs
10 m = prune(m, cp=0.05) # prune the tree, won't change anything here
11
12 p = predict(m, iris[150,], type="class") # predict class of data in the data frame, here one instance virginica
13 p = predict(m, iris[150,], type="prob") # predict probabilities
14 #    setosa versicolor virginica
15 # 150      0 0.02173913 0.9782609
16
17 # Confusion matrix, training and test data is identical to show ambivalence of the model
18 table(iris$Species, predict(m, iris, type="class"))
19 #             setosa versicolor virginica
20 #  setosa         50          0         0
21 #  versicolor      0         49         1
22 #  virginica       0          5        45
23 table(iris$Species == predict(m, iris, type="class")) / nrow(iris) # show fraction of predictions
24 # FALSE  TRUE
25 # 0.04  0.96
```

Machine Learning with Python

- Recommended package: `scikit-learn` [2]
- Provides classification, regression, clustering, dimensionality reduction
- Supports via model selection and preprocessing

Example: Decision tree

```
1  from sklearn.datasets import load_iris
2  from sklearn import tree
3  iris = load_iris()
4  m = tree.DecisionTreeClassifier()
5  m = m.fit(iris.data, iris.target)
6
7  # export the tree for graphviz
8  with open("iris.dot", 'w') as f:
9    tree.export_graphviz(m, out_file=f)
10
11 # To plot run: dot -Tpdf iris.dot
```



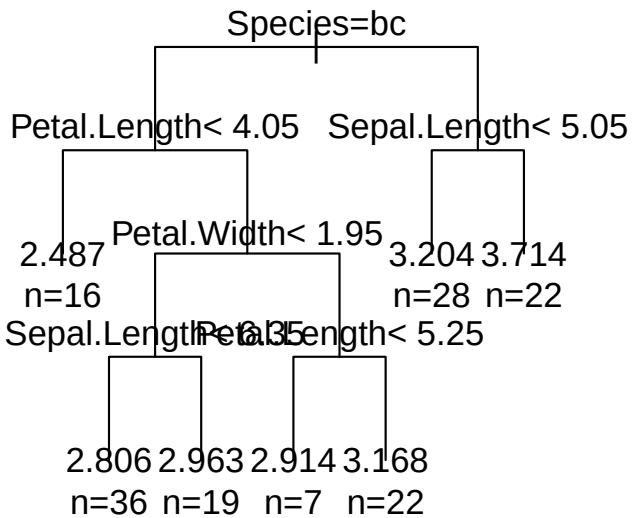
Sklearn decision tree

[2] http://scikit-learn.org/stable/

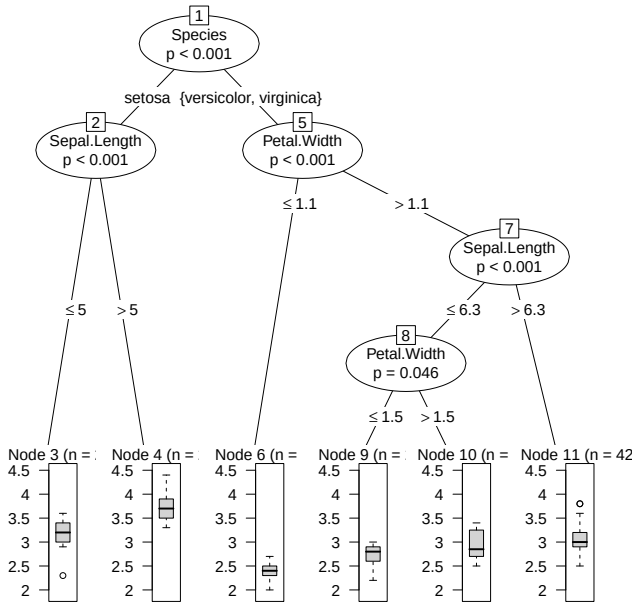
Regression Trees

- Regression trees predict numeric values
  - They usually optimize mean-squared error
- Party package uses statistical stopping rules (no pruning needed)

```
1 # Create a regression tree for Sepal.Width which optimizes mean-squared error
2 m = rpart( Sepal.Width ~ Species + Petal.Length + Petal.Width + Sepal.Length, data=iris, method="anova")
3 plot(m, compress=T, uniform=T, margin=0.7) # plot the tree
4 text(m, use.n=T) # add text to the tree
5
6 library(party) # package for recursive partitioning using nonparametric regression
7 m = ctree( Sepal.Width ~ Species + Petal.Length + Petal.Width + Sepal.Length, data=iris)
```



Regression tree for Sepal.Width



Regression tree with party

Model Trees [1, p. 202ff, 214ff]

- Problem: CART trees can predict only one class/value per leaf
  - This is suboptimal for regression trees as the accuracy depends on the leafs
- Model trees add a linear regression model on the leaf node
  - Especially unused attributes can be used to predict the numerical value
- M5-prime (M5P) algorithm is state of the art

Example for the Iris data and starting to compare model quality

```
library(RWeka)
m5 = M5P( Sepal.Width ~ Species + Petal.Length +
       ↪ Petal.Width + Sepal.Length, data=iris)
p5 = predict(m5, iris)
#M5 pruned model tree:
#(using smoothed linear models)
#Species=setosa <= 0.5 : LM1 (100/55.118%)
#Species=setosa >  0.5 : LM2 (50/57.858%)
#
#LM num: 1
#Sepal.Width =
# -0.2457 * Species=virginica,setosa
# + 0.4834 * Petal.Width
# + 0.1839 * Sepal.Length
# + 1.0373
#LM num: 2
#Sepal.Width =
# 0.0896 * Species=virginica,setosa
# - 0.0987 * Petal.Width
# + 0.6784 * Sepal.Length
# - 0.0485
# Number of Rules : 2
```

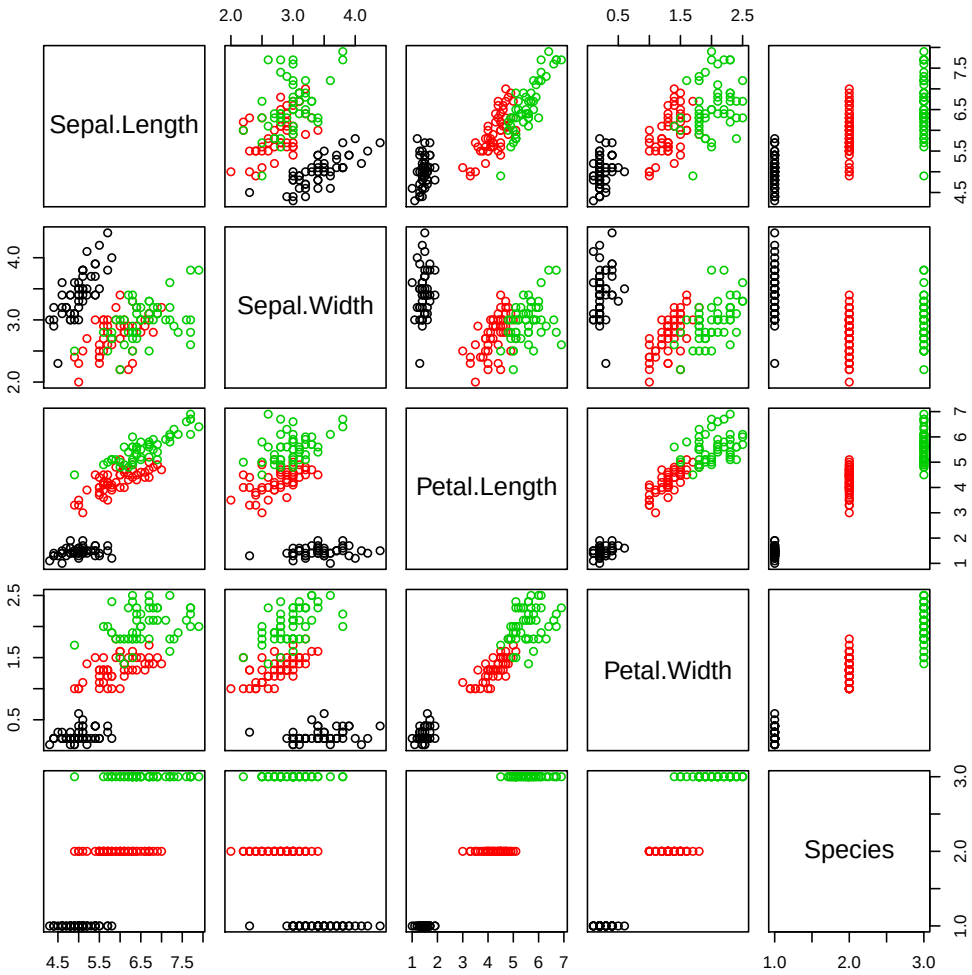
```
# Compare the error of CART (rpart), party and M5P
MAEi = function (p){
  return (mean(abs(iris$Sepal.Width - p)))
}
print(sprintf("rpart: %f party: %f m5p: %f", MAEi(pm),
       ↪ MAEi(pp), MAEi(p5)))
#MAE rpart: 0.203207 party: 0.206693 m5p: 0.189899

MSEi = function (p){
  return (sqrt(sum((iris$Sepal.Width - p)^2)/nrow(iris)))
}
print(sprintf("MSE rpart: %f party: %f m5p: %f", MSEi(pm),
       ↪ MSEi(pp), MSEi(p5)))
#MSE rpart: 0.260416 party: 0.265323 m5p: 0.246495

MAPEi = function (p){
  return (100*sum(abs((iris$Sepal.Width -
         ↪ p)/iris$Sepal.Width))/nrow(iris))
}
print(sprintf("MAPE rpart: %.1f%% party: %.1f%% m5p:
       ↪ %.1f%%", MAPEi(pm), MAPEi(pp), MAPEi(p5)))
#MAPE rpart: 6.8% party: 6.9% m5p: 6.4%
```

Clustering

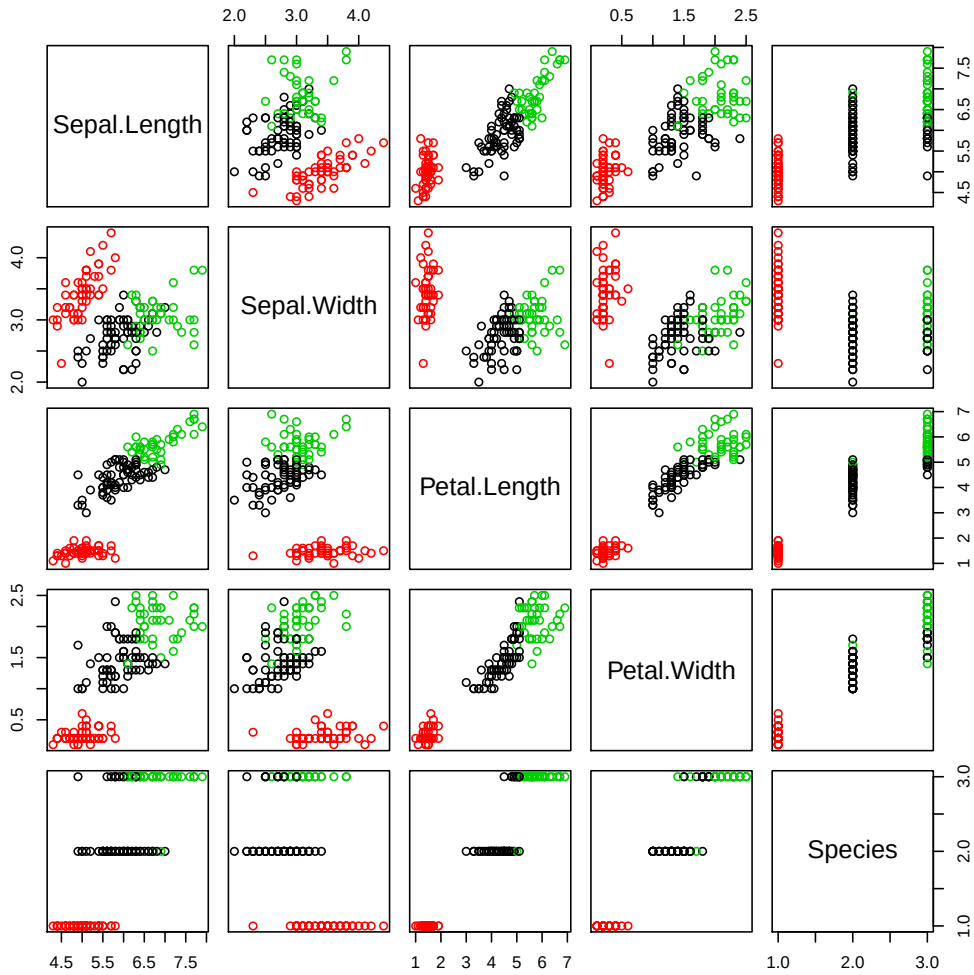
- Partition data into “similar” observations
- Allows prediction of a class for new observations
- Unsupervised learning strategy
- Clustering based on distance metrics to a center (usually euclidean)
  - Can identify regular (convex) shapes
  - k-means: k-clusters, start with a random center, iterative refinement
- Hierarchical clustering: distance based methods
  - Usually based on distance matrix (N^2 elements)
  - Agglomerative (start with individual points) or divisive
- Density based clustering uses proximity to cluster members
  - Can identify any shape
  - DBSCAN: requires the density parameter (eps)
  - OPTICS: nonparametric
- Model-based: automatic selection of the model and clusters
- Normalization of variable ranges is mandatory
  - One dimension with values in 0 to 1 is always dominated by one of 10 to 100

K-means Clustering

```
1 p = kmeans(iris[,1:4], centers=3) # cluster in 4 dimensions, exclude species
2 plot(iris, col=p$cluster)
```



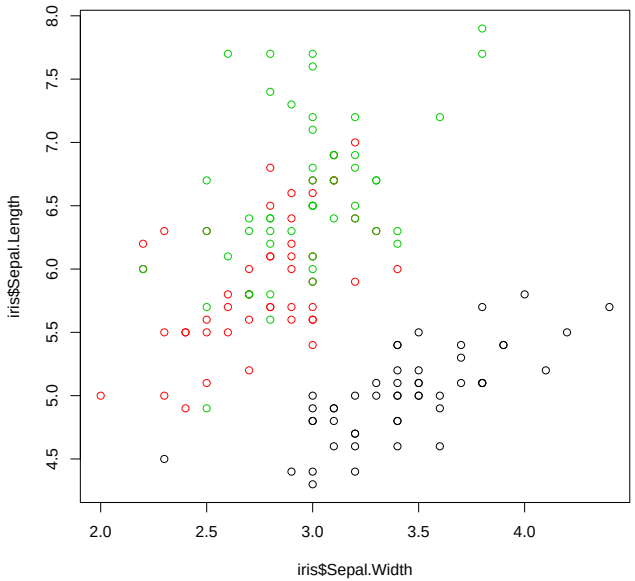
Real species



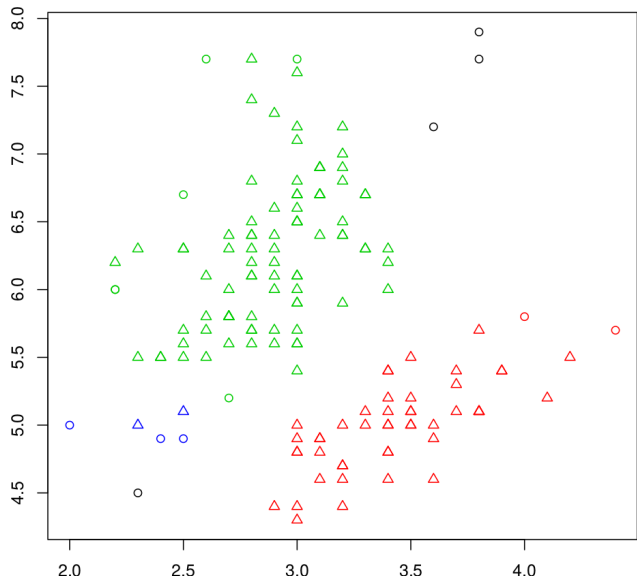
Kmeans in 4D

Density-Based Clustering

```
1  library(fpc) # for dbscan
2  # For illustration purpose, we cluster the 4D feature set only using two variables
3
4  # 2D plot coloring the species
5  plot(iris$Sepal.Width, iris$Sepal.Length, col=iris$Species)
6
7  # Create a 2D matrix as input for dbscan
8  d = cbind(iris$Sepal.Width, iris$Sepal.Length)
9
10 # try to identify classes, showplot illustrates the process
11 p = dbscan(d, eps=0.35, showplot=1)
```



Real species (classes)

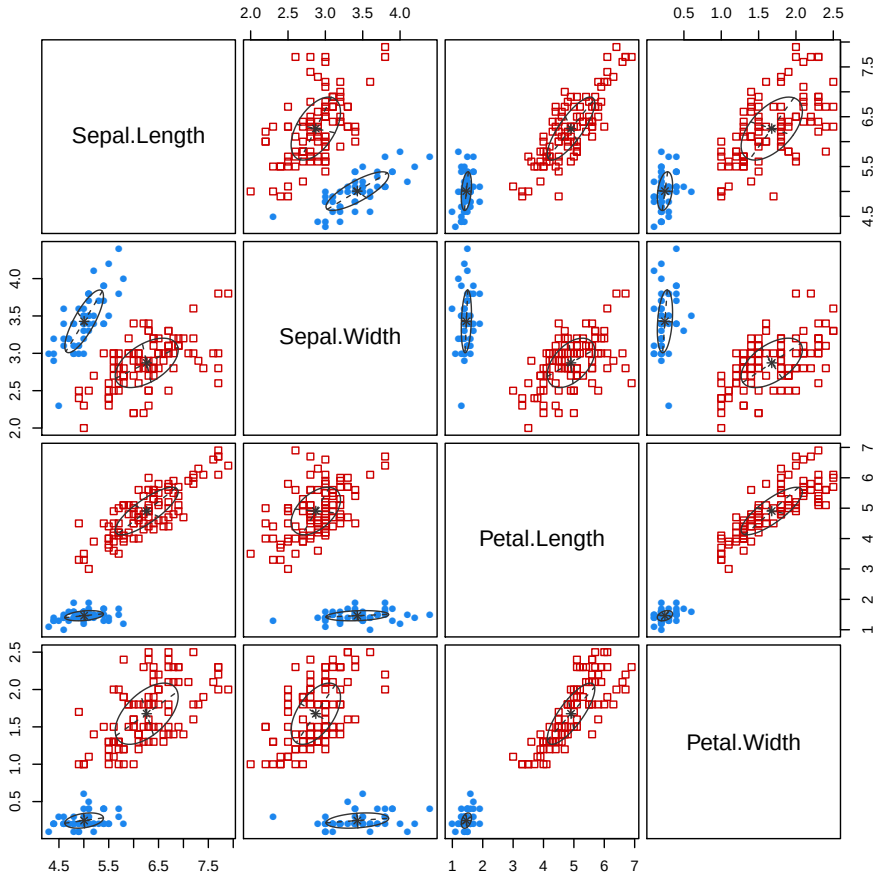


Output of dbscan

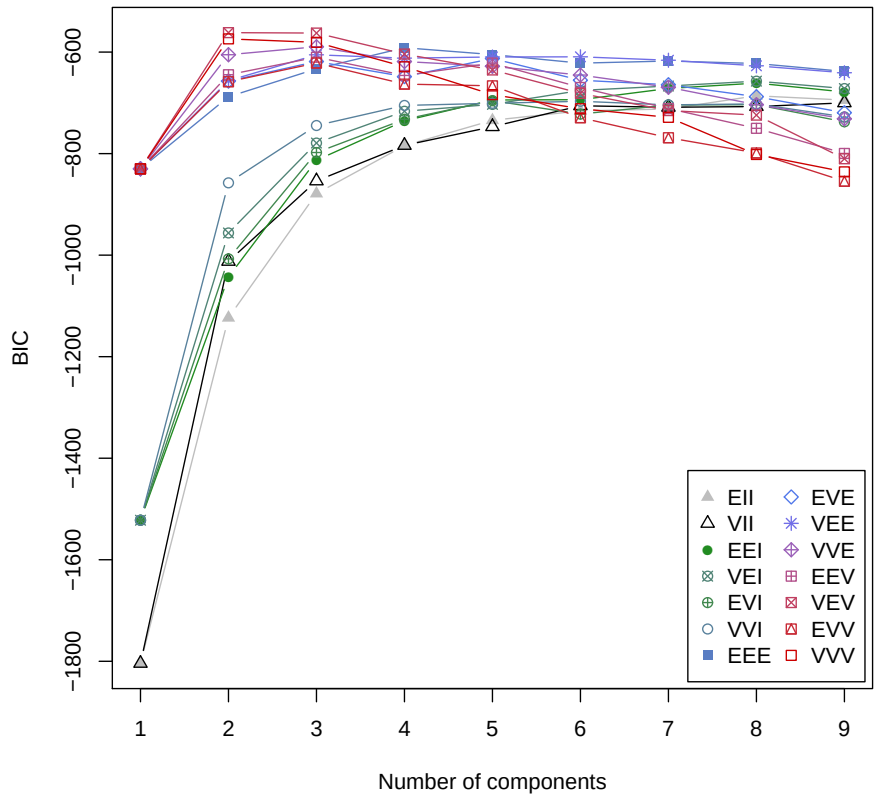
Model Based Clustering

- Automatic selection of model and cluster number
- Uses bayesian information criterion (BIC) and expectation-maximization

```
1 library(mclust)
2 m = Mclust(iris[,1:4]) # chooses a ellipsoidal model
```



Model-based classification

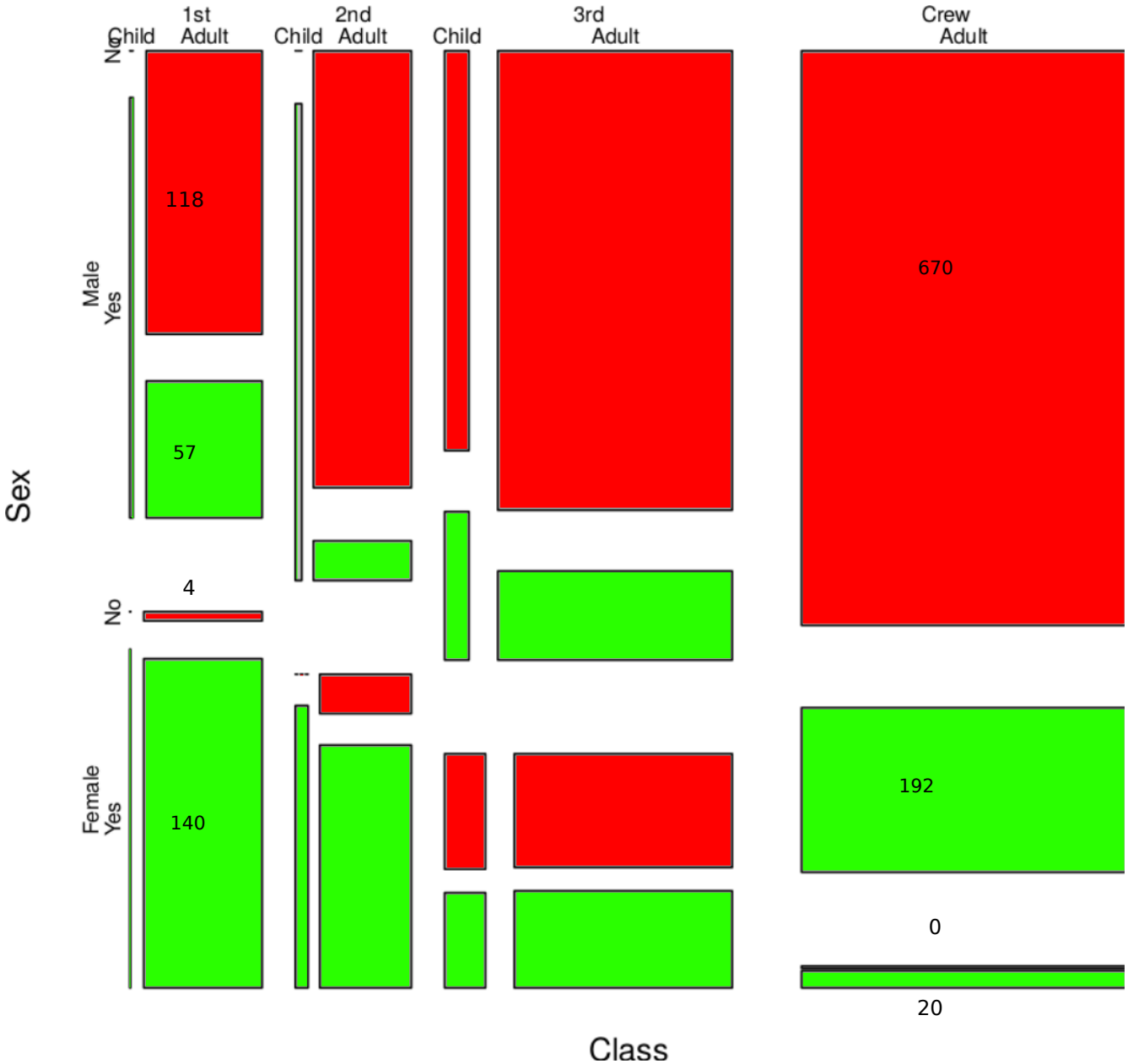


BIC

Association Rule Mining [44]

- Discover interesting relations in correlated facts and extract rules
- Identify frequent item sets “likes HR, likes BigData“
- Example association rule: “likes HR, likes BigData $\Rightarrow$ likes NTHR”
- Data are individual transactions, e.g., purchases, with items
  - Items $I = i_1, ..., i_n$
  - Transactions $T = t_1, ..., t_n$
  - Each t_i is a subset of I, e.g., items bought together in a market basket
- Several algorithms exist, e.g., APRIORI, RELIM
- Relevance of rules is defined by support and confidence
  - Assume $X \Rightarrow Y$ be an association rule, X, Y are item-sets
  - support(X): number of transactions which contains item-set X
  - confidence($X \Rightarrow Y$) = $support(X \cup Y)/support(X)$: fraction of transactions which contain X and Y. Indicates if the rule is good

Titanic Dataset Shows the Classes of Survivors



Association Analysis with Python Using Pyming[3]

```
1  from pymining import itemmining, assocrules
2  import csv
3  with open('titanic2.csv', 'r') as csvfile:
4    reader = csv.reader(csvfile)
5    data = [ r for r in reader ]
6
7  # apply relim algorithm
8  r = itemmining.get_relim_input(data)
9  # find frequent items (more than 1000 instances)
10 itemsets = itemmining.relim(r, min_support=1000)
11 # {frozenset(['No']): 1490, frozenset(['Male', 'Adult', 'No']): 1329, frozenset(['Adult', 'No']): 1438, frozenset(['Adult']):
       ↪ 2092, frozenset(['Male', 'Adult']): 1667, frozenset(['Male', 'No']): 1364, frozenset(['Male']): 1731}
12
13 # mine the association rules
14 r = itemmining.get_relim_input(data)
15 itemsets = itemmining.relim(r, min_support=1)
16 rules = assocrules.mine_assoc_rules(itemsets, min_support=2, min_confidence=0.7)
17 # ((['Adult', 'No']), (['Male']), 1329, 0.9242002781641169), ((['No']), (['Male', 'Adult']), 1329, 0.8919463087248322), ...
18 # identify only survival-relevant rules with two or one items/attributes
19 relevant = [ (p, "Yes" in c,supp,conf) for p, c, supp, conf in rules if (c == frozenset(['No']) or c == frozenset(['Yes']))
       ↪ and len(p) <= 2]
20 relevant.sort(key=lambda x : x[1]) # sort based on the survival
21 for p, c, supp, conf in relevant:
22   print(("%d,%.2f: %s <= %s" % (supp, conf, c, p)).replace("frozenset",""))
23 #1329,0.80: False <= (['Male', 'Adult'])
24 #476,0.76: False <= (['Adult', '3rd'])
25 #154,0.86: False <= (['Male', '2nd'])
26 #422,0.83: False <= (['Male', '3rd'])
27 #344,0.73: True <= (['Female'])
28 #316,0.74: True <= (['Adult', 'Female'])
29 #6,1.00: True <= (['1st', 'Child'])
30 #24,1.00: True <= (['2nd', 'Child'])
```

[3] https://github.com/bartdag/pymining

Meta-Learning [1, p.359ff]

- Idea: combine multiple (weak) models to improve model performance
- **Ensemble**: team of models used by a meta-model
- Approach: take data subset to train each model and combine prediction
  - Allocation function: defines which training data each model receives
  - Combination function: resolves disagreement
  - Stacking: learn the combination function (as a ML model)
- Advantage of meta-learning:
  - Generalizability: prevents overfitting of training data
  - Performance: small models are faster to train, parallel training is possible
  - Nuanced understanding: subtle patterns are better covered than in a global model

Bagging [1, p. 362]

- Bagging == bootstrap aggregation
- Approach:
  1. Generate many training datasets by sampling the training data
  2. Train a model for each training dataset (using the same learning algorithm)
  3. Combine predictions
     - Voting (for classification problems)
     - Averaging (for prediction problems)
- Particularly useful on unstable learners
  - Unstable learners: training algorithms depending heavily on the input data
- Random forest
  - Build many decision trees ($\Rightarrow$ forest)
  - Increase variety by choosing a small set of features for each tree randomly

Boosting [1, p.366ff]

- Differences to bagging
  - Generate complementary learners
  - Weight vote of learner based on past performance
- Adaptive boosting: learn the difficult-to-classify examples
  - First classifier: train on all data
  - Subsequent rounds: remove correct predictions (with a high probability)
  - Stop after desired accuracy is reached or classifier does not improve
  - Weight vote based on accuracy on the training data on which it was built
  - In R: AdaBoost.M1 algorithm / C5.0 algorithm
- Alternative approach: predictor for predicting the error

Summary

- Data preprocessing normalizes data / treats missing data
- Machine learning problems:
  - Classification, regression, clustering, association rule mining
- Holdout method: strategy to assess model quality
- Evaluation of model performance is crucial
  - Categorical data: confusion matrix + metrics
  - Numerical data: residual, MAE, MSE, MAPE
- Classification: K-NN, Naive Bayes, trees
- Regression: linear models, regression trees
- Clustering: k-means, density-based
- Association rule mining: market basket analysis with APRIORI

Bibliography

1 Book: Machine Learning with R. Brett Lantz. 2015. PACKT publishing.

10 Wikipedia

11 https://en.wikipedia.org/wiki/Stratified_sampling

12 https://en.wiktionary.org/wiki/Manhattan_distanc

32 https://en.wikipedia.org/wiki/Imputation_(statistics)

33 https://en.wikipedia.org/wiki/Missing_data

34 https://en.wikipedia.org/wiki/Generalized_linear_model

35 https://en.wikipedia.org/wiki/Data_mining

38 https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining

39 ftp://ftp.software.ibm.com/software/analytics/spss/support/Modeler/Documentation/14/UserManual/CRISP-DM.pdf

40 https://en.wikipedia.org/wiki/Machine_learning

41 http://www.rdatamining.com/docs/introduction-to-data-mining-with-r

42 CRAN Task View: Machine Learning & Statistical Learning https://cran.r-project.org/web/views/MachineLearning.html

43 http://www.rdatamining.com/examples/text-mining

44 https://en.wikipedia.org/wiki/Association_rule_learning