LABORATORY OF DATA SCIENCE

Reminds on Data Mining
BI Architecture

Figure 1. Typical business intelligence architecture.

- **Data sources**
  - External Data Sources
  - Operational Databases
- **Data movement, streaming engines**
  - Extract Transform Load (ETL)
  - Complex Event Processing Engine
- **Data warehouse servers**
  - Relational DBMS
  - MapReduce engine
- **Mid-tier servers**
  - OLAP Server
  - Enterprise search engine
- **Front-end applications**
  - Search
  - Spreadsheet
  - Dashboard
  - Ad hoc query

- 6 lessons – data access
- 4 lessons – data quality & ETL
- 1 lessons – analytic SQL
- 5 lessons – OLAP and reporting
- 4 lessons – data mining

Lab of Data Science
Data Mining Techniques

- Classification/Regression
- Association Rule Discovery
- Clustering
- Sequential Pattern Discovery
- Deviation Detection
- Text Mining
- Web Mining
- Social Network Analysis
- ...

Lab of Data Science
Tools for data mining

- From **DBMS**
  - SQL Server Analysis Services
  - Oracle Data Miner
  - IBM DB2 Intelligent Miner (discontinued)

- From **Statistical analysis**
  - IBM Modeler (formerly SPSS Clementine)
  - SAS Miner

- From **Machine-Learning**
  - Knime
  - Weka

- An updated list
Standards

- XML representation of data mining models
  - Predictive Modelling Markup Language: PMML

- API for accessing data mining services
  - Microsoft OLE DB for DM
  - Java JDM

- SQL Extensions for data mining
  - Standard SQL/MM Part 6 Data Mining
  - Oracle, DB2 & SQL Server have non-standard extensions
    - SSAS DMX query language and Data Mining queries
Weka

- Suite for machine learning / data mining
- Developed in Java
  - Distributed with a GNU GPL licence
  - Since 2006 it is part of the BI Pentaho suite
- References
  - “Data Mining” by Witten & Frank, 3rd ed., 2011
  - On line docs
    http://www.cs.waikato.ac.nz/ml/weka/index.html
- Features / limits:
  - A complete set of tools for pre-processing, classification, clustering, association rules, visualization
  - Extensible (documented APIs)
  - Not very efficient / scalable (data are maintained in main memory)
Weka versions

- May 2015, Weka 3.7.12 (developer version)
- Patch distributed by the teacher
  - To be copied in the Weka installation directory
  - It includes setting for:
    - Larger memory occupation (Java default is 80Mb)
    - Data types for SQL Server RDBMS
    - Driver JDBC
- Weka Light
  - Minimal version 3.7.12, patch already included
Weka interfaces

- GUI chooser and console with errors/warnings
Weka interfaces: Explorer

**Explorer**: GUI with distinct panels for preprocessing, classification, clustering, ...
Weka interfaces: KnowledgeFlow

KnowledgeFlow: GUI with data flow
Simple CLI (Call Level Interface):
command line interface
Experimenter: automation of large experiments by varying datasets, algorithms, parameters, ..
Details

- Weka manual
  - Installation directory, or at the weka website

Lab of Data Science
Filters

- **Conversions**
  - MakeIndicator, NominalToBinary, NumericToBinary, NumericToNominal

- **Selections**
  - RemovePercentage, RemoveRange, RemoveWithValue, SubSetByExpression

- **Sampling**
  - Resample, SpreadSubSample, StratifiedRemoveFolds

- **Transformation**
  - Add, AddExpression, AddNoise, AddValues

- **Normalization**
  - Center, Normalize, Standardize

- **Discretization**
  - Discretize

- **Cleaning**
  - NumericCleaner, Remove, RemoveByType, RemoveUseless

- **Missing Values**
  - ReplaceMissingValues
Reminds on classification
Who are my best customers?

- ... given their age and frequency of visit!
- Good customers = top buyers, buy more than X, ...
... described with a decision tree!
A set of examples (or instances or cases) which described a concept or event (class) given predictive attributes (or features)

- Attributes can be either continuous or discrete (maybe discretized)
- The class is discrete

<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>Don't Play</td>
</tr>
<tr>
<td>sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>Don't Play</td>
</tr>
<tr>
<td>overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>Don't Play</td>
</tr>
<tr>
<td>overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>sunny</td>
<td>72</td>
<td>95</td>
<td>false</td>
<td>Don't Play</td>
</tr>
<tr>
<td>sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>Don't Play</td>
</tr>
</tbody>
</table>
A function \( f(sample) = class \), called a **classification model**, that describes/predict the class value given the feature values of a sample obtained by generalizing the samples of the training set.

- **Usage of a classification model:**
  - **descriptively**
    - Which customers have abandoned?
  - **predictively**
    - Over a **score set** of samples with unknown class value
    - Which customers will respond to this offer?
How to evaluate a class model?

- **Holdout method**
  - Split the available data into two sets
  - Training set is used to build the model
  - Test set is used to evaluate the interestingness of the model
    - Typically, training is 2/3 of data and test is 1/3

### Quality measure

<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>Don't Play</td>
</tr>
<tr>
<td>sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>Don't Play</td>
</tr>
<tr>
<td>overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>Don't Play</td>
</tr>
<tr>
<td>overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>sunny</td>
<td>72</td>
<td>95</td>
<td>false</td>
<td>Don't Play</td>
</tr>
<tr>
<td>sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>Play</td>
</tr>
</tbody>
</table>

### Model extraction

<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>Don't Play</td>
</tr>
</tbody>
</table>
How good is a classification model?

- **Stratified holdout**
  - Available data is divided by stratified sampling wrt class distribution

- **(Stratified) n-fold cross-validation**
  - Available data divided into n parts of equal size
  - For \( i=1..n \), the \( i \)-th part is used as test set and the rest as training set for building a classifier
  - The average quality measure of the n classifiers is statistically more significative than the holdout method
  - The FINAL classifier is the one training from all the available data
    - Cross-validation is useful when data is scarce or attribute distributions are skewed

Lab of Data Science
Quality measures: accuracy

- **Accuracy**: percentage of cases in the test set that is correctly predicted by the model
  - E.g., accuracy of 80% means that in 8 cases out of 10 in the test set the predicted class is the same of the actual class

- **Misclassification % = (100 – accuracy)**

- **Lower bound on accuracy: majority classifier**
  - A trivial classifier for which f(case) = majority class value
  - Its accuracy is the percentage of the majority class
  - E.g., two classes: fraud 2% legal 98%
  - Its hard to beat the 98% accuracy

Lab of Data Science
Quality measures: confusion matrix

Correctly Classified Instances  42
Incorrectly Classified Instances  15
Total Number of Instances  57

== Confusion Matrix ==

\[
\begin{array}{cc}
14 & 6 \\
9 & 28
\end{array}
\]

Predicted class: a = bad
Actual class: b = good

Lab of Data Science
Quality measures: precision

**Precision**: accuracy of predicting “C”

- # Cases predicted Class=C and with real Class=C
- # Cases predicted Class=C

76% of times predictions >50K are correct
Recall: coverage of predicting “C”

# Cases predicted Class=C and with real Class=C
# Cases with real Class=C

59.4% of real class >50K are found by predictions
Measures: lift chart

- **Classifier:** \( f(\text{sample, class}) = \text{confidence} \)
  - and then \( f(\text{sample}) = \text{argmax}_{\text{class}} f(\text{sample, class}) \)
  - E.g., \( f(\text{sample, play}) = 0.3 \quad f(\text{sample, don’t play}) = 0.7 \)

- **Samples in the test set can be ranked according to a fixed class**
  - Rank customers on the basis of the classifier confidence **they will respond** to an offer

- **Lift chart**
  - **X-axis:** ranked sample of the test set
  - **Y-axis:** percentage of the total cases in the test set with the actual class value included in the ranked sample of the test set (i.e., recall)
  - Plots: performance of a classifier vs random ranking
  - Useful when resources (e.g., budget) are limited
Contacting only 50% of customers will reach 80% of those who respond. Lift = 80/50 = 1.6
Lift Chart - variants

- Lift(X) = recall(X)
  - Estimation of random classifier lift
  - Previous example, Lift(50%) = 80%

- LiftRatio(X) = recall(X) / X
  - Ratio of lift over random order
  - Previous example, LiftRatio(50%) = 80% / 50% = 1.6

- Profit chart
  - Given a cost/benefit model, the Y axis represent the total cost/gain when contacting X and not contacting TestSet\X
The unbalancing problem

- For unbalanced class values, it is difficult to obtain a good model
  - Fraud = 2%  Normal = 98%
    - The majority classifier is accurate at 98% but it is not useful

- Oversampling and Undersampling
  - Select a training set with a more balanced distribution of class values A and B
    - 60-70% for class A and 30-40% for class B
    - By increasing the number of cases with class B (oversampling) or by reducing those with class A (undersampling)
  - The training algorithm has more chances of distinguishing characteristics of A VS B
    - The test set MUST have the original distribution of values

- Cost Sensitive Classifier, Ensembles (bagging, boosting, stacking)
  - Weights errors, build several classifiers and average their predictions
Rule based classification
Rule-Based Classifier

- Classify records by using a collection of “if…then…” rules

- Rule: \((\text{Condition}) \rightarrow y\)
  - where
    - \text{Condition} is a conjunctions of attributes
    - \(y\) is the class label
  - \(LHS\): rule antecedent or condition
  - \(RHS\): rule consequent

- Examples of classification rules:
  - \((\text{Blood Type}=\text{Warm}) \land (\text{Lay Eggs}=\text{Yes}) \rightarrow \text{Birds}\)
  - \((\text{Taxable Income} < 50\text{K}) \land (\text{Refund}=\text{Yes}) \rightarrow \text{Evade}=\text{No}\)
Rule-based Classifier (Example)

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → Amphibians
A rule \( r \) covers an instance \( x \) if the attributes of the instance satisfy the condition of the rule.

- \( R1: (\text{Give Birth} = \text{no}) \land (\text{Can Fly} = \text{yes}) \rightarrow \text{Birds} \)
- \( R2: (\text{Give Birth} = \text{no}) \land (\text{Live in Water} = \text{yes}) \rightarrow \text{Fishes} \)
- \( R3: (\text{Give Birth} = \text{yes}) \land (\text{Blood Type} = \text{warm}) \rightarrow \text{Mammals} \)
- \( R4: (\text{Give Birth} = \text{no}) \land (\text{Can Fly} = \text{no}) \rightarrow \text{Reptiles} \)
- \( R5: (\text{Live in Water} = \text{sometimes}) \rightarrow \text{Amphibians} \)

<table>
<thead>
<tr>
<th>Name</th>
<th>Blood Type</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>hawk</td>
<td>warm</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>grizzly bear</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>?</td>
</tr>
</tbody>
</table>

The rule \( R1 \) covers a hawk \( \Rightarrow \) Bird
The rule \( R3 \) covers the grizzly bear \( \Rightarrow \) Mammal
Rule Coverage and Accuracy

- **Coverage of a rule:**
  - Fraction of records that satisfy the antecedent of a rule

- **Accuracy of a rule:**
  - Fraction of records that satisfy both the antecedent and consequent of a rule

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

(Status=Single) → No

Coverage = 40%, Accuracy = 50%
How does Rule-based Classifier Work?

R1: (Give Birth = no) \land (Can Fly = yes) \rightarrow \text{Birds}

R2: (Give Birth = no) \land (Live in Water = yes) \rightarrow \text{Fishes}

R3: (Give Birth = yes) \land (Blood Type = warm) \rightarrow \text{Mammals}

R4: (Give Birth = no) \land (Can Fly = no) \rightarrow \text{Reptiles}

R5: (Live in Water = sometimes) \rightarrow \text{Amphibians}

<table>
<thead>
<tr>
<th>Name</th>
<th>Blood Type</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemur</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>turtle</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>?</td>
</tr>
<tr>
<td>dogfish shark</td>
<td>cold</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>?</td>
</tr>
</tbody>
</table>

A lemur triggers rule R3, so it is classified as a mammal

A turtle triggers both R4 and R5

A dogfish shark triggers none of the rules
Characteristics of Rule-Based Classifier

- **Mutually exclusive rules**
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every record is covered by **at most** one rule

- **Exhaustive rules**
  - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
  - Each record is covered by **at least** one rule
From Decision Trees To Rules

Classification Rules

(Refund=Yes) ==> No
(Refund=No, Marital Status={Single, Divorced}, Taxable Income<80K) ==> No
(Refund=No, Marital Status={Single, Divorced}, Taxable Income>80K) ==> Yes
(Refund=No, Marital Status={Married}) ==> No

Rules are mutually exclusive and exhaustive
Rule set contains as much information as the tree
Rules Can Be Simplified

Initial Rule: \((\text{Refund} = \text{No}) \land (\text{Status} = \text{Married}) \rightarrow \text{No}\)

Simplified Rule: \((\text{Status} = \text{Married}) \rightarrow \text{No}\)
Effect of Rule Simplification

- Rules are no longer mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
    - Unordered rule set – use voting schemes

- Rules are no longer exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class
Building Classification Rules

- **Direct Method:**
  - Extract rules directly from data
  - e.g.: RIPPER, CN2, Holte’s 1R

- **Indirect Method:**
  - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
  - e.g.: C4.5rules