Data Mining I

Classification Workflow with Rapidminer
Outline

1. Data Import
2. Preprocessing
3. Classification
4. Evaluation
Data Import

- import your data into Rapidminer Repository
  - Everything in one place
  - Valuable meta-data for further processing

- Use the import wizard, if available
Preprocessing

- Look at your data
  - What is the target attribute?
    - Is the target attribute already a label?
  - What is the distribution of labeled examples by class?
    - Is my classifier capable of handling imbalanced data?
  - What other attributes are available?
    - Is my classifier able to handle these types of attribute?
  - What are the ranges of the attributes?
    - Is my classifier good in handling various ranges?
  - What attributes correlate?
    - Is my classifier able to handle strongly correlating attributes?

- See Exercise 1 for more information.
Set Roles & Normalization

• Set roles for attributes

• Normalize attribute values
Discretize

- Numerical attributes can be divided into bins using discretization
- By Size (equally sized data ranges per bin)

- By Frequency (equally sized number of examples per bin)
Balancing

- Sampling (with balancing)
- Multiplication of data
  - Filter under-represented class examples
  - Append them to original example set
Classification

- Input: data set with labels
- Output: classification modell

Known Classifiers:
- K-NN
- Naive Bayes
- Decision Tree (Hunts & ID3)
- Rule Induction & Tree to Rules
- Support Vector Machine (libSVM)
- Neural Networks
Evaluation

- Evaluate on dedicated test data set

- Evaluate on one data set using
  - Split validation
  - X-Validation
Split-/Cross-Validation

- Split-validation is a *holdout method*, which reserves a certain amount for testing and uses the remainder for training.
  - First step: split data at a ratio in test and training set
  - Second step: learn a model on the training set and evaluate the model on the test set

- *Cross-validation* avoids overlapping test sets
  - First step: data is split into $k$ subsets of equal size
  - Second step: each subset in turn is used for testing and the remainder for training

Important: Never ever use the same example set for training & testing!
### Accuracy and Error Rate

- Most widely-used metrics:

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

**Error Rate** = 1 – **Accuracy**
Limitation of Accuracy: Unbalanced Data

- Sometimes, classes have very unequal frequency
  - Fraud detection: 98% transactions OK, 2% fraud
  - eCommerce: 99% don’t buy, 1% buy
  - Intruder detection: 99.99% of the users are no intruders
  - Security: >99.99% of Americans are not terrorists

- The class of interest is commonly called the positive class, and the rest negative classes.

- Consider a 2-class problem
  - Number of Class 0 examples = 9990, Number of Class 1 examples = 10
  - If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example
Precision and Recall

Alternative: Use measures from information retrieval which are biased towards the positive class.

<table>
<thead>
<tr>
<th>Actual Positive</th>
<th>Classified Positive</th>
<th>Classified Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td>FN</td>
<td>TN</td>
</tr>
<tr>
<td>FP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
p = \frac{TP}{TP + FP}.
\]

\[
r = \frac{TP}{TP + FN}.
\]

Precision \( p \) is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.

Recall \( r \) is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.
Performance

- Standard Measures
  - Accuracy
  - Precision
  - Recall

- Task Specific
  - Misclassification Costs
Questions?