Data Mining

Classification

- Part 2 -
Outline

1. What is Classification?
2. K-Nearest-Neighbors
3. Decision Trees
4. Model Evaluation
5. Rule Learning
6. Naïve Bayes
7. Artificial Neural Networks
8. Support Vector Machines
9. Parameter Tuning
4. Model Evaluation

Central Question:

How good is a model at classifying unseen records?

4.1 Metrics for Model Evaluation

- How to measure the performance of a model?

4.2 Methods for Model Evaluation

- How to obtain reliable estimates?
4.1 Metrics for Model Evaluation

- Focus on the **predictive capability** of a model
- Rather than how much time it takes to classify records or build models.

**Confusion Matrix**

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=Yes</td>
<td><strong>True Positives</strong></td>
</tr>
<tr>
<td>Class=No</td>
<td>False Positives</td>
</tr>
</tbody>
</table>
Accuracy and Error Rate

\[
Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{Correct\ predictions}{All\ predictions}
\]

Error Rate = 1 – Accuracy

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>Class=Yes</th>
<th>Class=No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>TP 25</td>
<td>FN 4</td>
<td></td>
</tr>
<tr>
<td>Class=No</td>
<td>FP 6</td>
<td>TN 15</td>
<td></td>
</tr>
</tbody>
</table>

Acc = \frac{25 + 15}{25 + 15 + 6 + 4} = 0.80
The Class Imbalance Problem

- Sometimes, classes have very unequal frequency
  - Fraud detection: 98% transactions OK, 2% fraud
  - E-commerce: 99% surfers 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 negative examples = 9990
    Number of positive examples = 10
  - If model predicts all examples to belong to the negative class, the accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any positive example.
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>Actual Negative</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td></td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

\[
p = \frac{TP}{TP + FP} \quad r = \frac{TP}{TP + FN}
\]

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.
Precision and Recall - Visualized

How many examples that are classified positive are actually positive?

Precision = \( p = \frac{TP}{TP + FP} \)

Which fraction of all positive examples is classified correctly?

Recall = \( r = \frac{TP}{TP + FN} \)

Source: Walber
Precision and Recall – A Problematic Case

This confusion matrix gives us
- precision \( p = 100\% \)
- recall \( r = 1\% \)

because we only classified one positive example correctly and no negative examples wrongly.

Thus, we want a measure that
1. combines precision and recall and
2. is large if both values are large.
F$_1$-Measure

- F$_1$-score combines precision and recall into one measure.

\[
F_1 = \frac{2pr}{p + r}
\]

F$_1$-score is the harmonic mean of precision and recall.

\[
F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}
\]

- The harmonic mean of two numbers tends to be closer to the smaller of the two.
- Thus, for the F$_1$-score to be large, both $p$ and $r$ must be large.
F₁-Measure Graph

Low threshold: Low precision, high recall
Restrictive threshold: High precision, low recall

Optimal Threshold
Cost-Sensitive Model Evaluation

$C(i|j)$: Cost of misclassifying a class $j$ record as class $i$

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C(i</td>
<td>j)$</td>
<td>$\text{Class}=\text{Yes}$</td>
</tr>
<tr>
<td>$\text{Class}=\text{Yes}$</td>
<td>$C(\text{Yes</td>
<td>Yes})$</td>
<td>$C(\text{No</td>
</tr>
<tr>
<td>$\text{Class}=\text{No}$</td>
<td>$C(\text{Yes</td>
<td>No})$</td>
<td>$C(\text{No</td>
</tr>
</tbody>
</table>
Example: Cost-Sensitive Model Evaluation

### Cost Matrix

| ACTUAL CLASS | PREDICTED CLASS | C(i|j) | + | - |
|--------------|-----------------|-------|---|---|
| +            | -1              | 100   |   |   |
| -            | 1               | 1     |   |   |

#### Model M₁

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>150</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>60</td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

**Accuracy = 80%**

**Cost = 3910**

#### Model M₂

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>250</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>5</td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>

**Accuracy = 90%**

**Cost = 4255**

Use case:
- credit card fraud.
- it is expensive to miss fraudulent transactions.
ROC Curves

• Some classification algorithms provide confidence scores
  – how sure the algorithms is with its prediction
  – e.g., KNN (the neighbor’s vote), Naive Bayes (the probability)

• ROC curves visualize true positive rate and false positive rate in relation to the algorithm’s confidence.

• Drawing a ROC Curve
  – Sort classifications according to confidence scores
  – Scan over data
    – right prediction: draw one step up
    – wrong prediction: draw one step to the right
  – Exact method: Tan et al, Chapter 5.7.2
Interpreting ROC Curves

- The steeper, the better
  - random guessing results in the diagonal
  - so a decent classification model should result in a curve above the diagonal

- Comparing models:
  - Curve A above curve B means model A better than model B

- Frequently used quality criterion
  - Area under ROC curve
4.2 Methods for Model Evaluation

- How to obtain a reliable estimate of performance?
- Which labeled records to use for training and which for testing?

- Methods for estimating the performance measures discussed in the last section:
  1. Holdout Method
  2. Random Subsampling
  3. Cross Validation
The learning curve shows how accuracy changes with growing training set size.

Conclusion: Use as much data as possible for training.

Problem: Labeling data often takes a lot of effort.
**Holdout Method**

- The *holdout method* reserves a certain amount of the labeled data for testing and uses the remainder for training.
- Usually: One third for testing, the rest for training

![Training Set](image1) ![Test Set](image2)

- For unbalanced datasets, random samples might not be representative
  - few or no records of the minority class/classes
- *Stratified sample*: Sample each class independently, so that records of the minority class are present in each sample.
Random Subsampling

- Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training
  - The error rates on the different iterations are averaged

- Still not optimal as the different test sets may overlap
  - Some outliers might always end up in the test sets
  - Records that are important for learning (the red tree) always in test sets
Cross-Validation

- **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

- This is called **k-fold cross-validation**

- The error estimates are averaged to yield an overall error estimate

- Frequently used: \( k = 10 \) (90% training, 10% testing)
  - Why ten? Experiments have shown that this is the good choice to get an accurate estimate and still use as much data as possible for training.

- Often the subsets are generated using stratified sampling
X-Validation in RapidMiner
Evaluation Summary

- Performance Metrics
  - Use accuracy
  - If interesting class is infrequent, use precision, recall and F1

- Estimation
  - Use cross-validation
  - If dataset is large and computation takes too much time, use holdout method

- To increase model performance
  1. Balance “unbalanced” data by increasing the number of positive examples in the training set (Rapidminer: training panel inside the validation operator)
  2. Optimize the parameters of the learning algorithm
  3. Avoid overfitting
5. Rule-Based Classification

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

- Classification rule:  \( \text{Condition} \rightarrow y \)
  - \( \text{Condition} \) is a conjunction of attribute tests (rule antecedent)
  - \( y \) is the class label (rule consequent)

- Examples of classification rules:
  
  R1: (Blood Type=Warm) \( \land \) (Lay Eggs=Yes) \( \rightarrow \) Birds
  R2: (Taxable Income < 50K) \( \land \) (Refund=Yes) \( \rightarrow \) Cheat = No

- Rule-based classifier
  - Set of classification rules
Example of a Rule-based Classifier

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

<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>human</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>python</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>reptiles</td>
</tr>
<tr>
<td>salmon</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>fishes</td>
</tr>
<tr>
<td>whale</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>frog</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>amphibians</td>
</tr>
<tr>
<td>komodo</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>reptiles</td>
</tr>
<tr>
<td>bat</td>
<td>warm</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>pigeon</td>
<td>warm</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>birds</td>
</tr>
<tr>
<td>cat</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>leopard shark</td>
<td>cold</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>fishes</td>
</tr>
<tr>
<td>turtle</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>reptiles</td>
</tr>
<tr>
<td>penguin</td>
<td>warm</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>birds</td>
</tr>
<tr>
<td>porcupine</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>eel</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>fishes</td>
</tr>
<tr>
<td>salamander</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>amphibians</td>
</tr>
<tr>
<td>gila monster</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>reptiles</td>
</tr>
<tr>
<td>platypus</td>
<td>warm</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>owl</td>
<td>warm</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>birds</td>
</tr>
<tr>
<td>dolphin</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>eagle</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>birds</td>
</tr>
</tbody>
</table>
5.1 Applying a Rule-based Classifier

- A rule \( r \) covers an instance \( x \) if the attributes of the instance satisfy the condition of the rule

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

<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 hawk \( \rightarrow \) Bird
- The rule R3 covers grizzly bear \( \rightarrow \) Mammal
Rule Coverage and Accuracy

- Coverage of a rule
  - Fraction of all records that satisfy the condition of a rule.

- Accuracy of a rule
  - Fraction of covered records that satisfy the consequent of a rule.

- Example
  - R1: (Status=Single) → No
  - Coverage = 40%
  - Accuracy = 50%

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</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>
Characteristics of Rule-Based Classifiers

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

- **Exhaustive Rule Set**
  - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
  - Each record is covered by at least one rule
A Rule Set that is not Mutually Exclusive and Exhaustive

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

<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 turtle triggers both R4 and R5
- A dogfish shark triggers none of the rules
Fixes for not Mutually Exclusive and Exhaustive Rule Sets

- Not Exhaustive Rule Set
  - Problem: Some unseen records are not covered by the rules
  - Solution: Add default rule: () → Y

- Not Mutually Exclusive Rule Set
  - Problem: An unseen record might be covered by multiple rules
  - Solution 1: Ordered Rules
    - Order rules (e.g. prefer rules with high accuracy)
    - Classify record according to the highest-ranked rule
  - Solution 2: Voting
    - Let all matching rules vote and assign the majority class label
    - The votes may be weighted by rule quality (e.g. accuracy)
Ordered Rule Set

- Rules are ordered according to their priority (e.g. accuracy)
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fires, it is assigned to the default class

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>turtle</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>?</td>
</tr>
</tbody>
</table>
5.2 Learning Rule-based Classifiers

1. Direct Method
   • Extract rules directly from data
   • Example algorithm: RIPPER

2. Indirect Method
   • Extract rules from other classification models (e.g. decision trees)
   • Example: C4.5rules
5.2.1 Indirect Method: From Decision Trees To Rules

- Approach: Generate a rule for every path from the root to one of the leaf nodes in the decision tree.
- Rule set contains as much information as the tree.
- The generated rules are mutually exclusive and exhaustive.

Classification Rules

(Rrefund=Yes) ==> No
(Rrefund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No
(Rrefund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes
(Rrefund=No, Marital Status={Married}) ==> No
The Generated Rules Can Be Simplified

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

Simplified Rule: \((\text{Status} = \text{Married}) \rightarrow \text{No}\)
Indirect Method: C4.5rules

1. Extract rules from an unpruned decision tree
2. For each rule, $r: A \rightarrow y$,
   1. consider an alternative rule $r': A' \rightarrow y$
      where $A'$ is obtained by removing one of the conjuncts in $A$
   2. compare the pessimistic error rate for $r$ against all $r$'s
      • estimate pessimistic error or measure it using validation dataset
3. prune if one of the $r$'s has lower pessimistic error rate
4. repeat until we can no longer improve generalization error

– Effect of rule simplification: Rule set is no longer mutually exclusive
  – A record may trigger more than one rule
  – Solution?
    • use ordered rule set or unordered rule set and voting schemes
Indirect Method in RapidMiner
Direct Method: RIPPER

- Learns ordered rule set from training data.
- For 2-class problem
  - Choose the less frequent class as positive class, and the other as negative class
  - Learn rules for the positive class
  - Negative class will be default class
- For multi-class problem
  - Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class)
  - Learn the rule set for smallest class first, treat the rest as negative class
  - Repeat with next smallest class as positive class
Sequential Covering

RIPPER uses sequential covering to learn a rule list for each class.

1. Start from an empty rule list
2. Grow a rule that covers as many positive examples as possible
3. Remove training records covered by the rule
4. Repeat Steps 2 and 3 until stopping criterion is met
Example of Sequential Covering ...

(i) Original Data

(ii) Step 1
Example of Sequential Covering

(iii) Step 2

(iv) Step 3
Aspects of Sequential Covering

1. Rule Growing
2. Rule Pruning
3. Instance Elimination
4. Stopping Criterion
Rule Growing within the RIPPER Algorithm

- Start from an empty rule: \( \{} \rightarrow \text{class} \)
- Step by step add conjuncts so that
  1. the accuracy of the rule improves
  2. the rule still covers many examples
Rule Growing within the Ripper Algorithm

- Goal: Prefer rules with high accuracy and high support count
- Add conjunct that maximizes FOIL’s information gain measure
  - $R_0$: $\emptyset \rightarrow \text{class}$ (initial rule)
  - $R_1$: $\{A\} \rightarrow \text{class}$ (rule after adding conjunct)
- Stop when rule no longer covers negative examples

\[
\text{Gain}(R_0, R_1) = t \left[ \log \left( \frac{p_1}{p_1 + n_1} \right) - \log \left( \frac{p_0}{p_0 + n_0} \right) \right]
\]

where
- $t$: number of positive instances covered by both $R_0$ and $R_1$
- $p_0$: number of positive instances covered by $R_0$
- $n_0$: number of negative instances covered by $R_0$
- $p_1$: number of positive instances covered by $R_1$
- $n_1$: number of negative instances covered by $R_1$
Rule Pruning

- Because of the stopping criterion, the learned rule is likely to overfit the data.

- Thus, the rule is pruned afterwards using a validation dataset.
  - similar to post-pruning of decision trees
Rule Pruning

- **Goal:**
  - Decrease generalization error of the rule

- **Procedure**
  1. Remove one of the conjuncts in the rule
  2. Compare error rates on a validation dataset before and after pruning
  3. If error improves, prune the conjunct

- **Measure for pruning**

  \[ v = \frac{p - n}{p + n} \]

  - \( p \): number of positive examples covered by the rule in the validation set
  - \( n \): number of negative examples covered by the rule in the validation set
Instance Elimination

- Why do we remove positive instances?
  - Otherwise, the next rule is identical to previous rule

- Why do we remove negative instances?
  - Prevent underestimating accuracy of rule
  - Compare rules R2 and R3 in the diagram
    - 3 errors vs. 2 errors
Stopping Criterion

- When to stop adding new rules to the rule set?

- RIPPER
  - Error rate of new rule on validation set must not exceed 50% 
  - Minimum description length should not increase more than d bits
RIPPER in RapidMiner

- **criterion**: `information_gain`
- **sample ratio**: 0.9
- **pureness**: 0.9
- **minimal prune benefit**: 0.25
- **use local random seed**: unchecked
RIPPER in RapidMiner

RuleModel

if wage-inc-1st > 2.650 and statutory-holidays > 10.500 then good (0 / 19)
if wage-inc-1st ≤ 3.600 and statutory-holidays ≤ 11.500 then bad (13 / 0)
else good (0 / 6)

correct: 38 out of 38 training examples.
Advantages of Rule-Based Classifiers

- Easy to interpret for humans (eager learning)
- Performance comparable to decision trees
- Can classify unseen instances rapidly
- Are well suited to handle imbalanced data sets
  - as they learn rules for the minority class first

Chapter 4.2: Confusion Matrix and Accuracy
Chapter 5.7: Precision, Recall, F1, ROC Curves
Chapter 4.5: Holdout Method, Cross-Validation
Chapter 5.1: Rule-Based Classification