Data Mining I

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. Support Vector Machines
8. Parameter Tuning
4. Model Evaluation

4.1 Metrics for Performance Evaluation
   • How to measure the performance of a model?

4.2 Methods for Performance Evaluation
   • How to obtain reliable estimates?
3.1 Metrics for Performance Evaluation

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

- Confusion Matrix:

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=No</td>
<td>Class=No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class=No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>FP</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class=No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Accuracy and Error Rate

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>TP</td>
</tr>
<tr>
<td>Class=No</td>
<td>FP</td>
</tr>
</tbody>
</table>

- 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 $\frac{9990}{10000} = 99.9\%$
  - 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></th>
<th>Classified Positive</th>
<th>Classified Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Positive</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>Actual Negative</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

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.

$$p = \frac{TP}{TP + FP} \quad r = \frac{TP}{TP + FN}$$
Precision and Recall Example

This confusion matrix gives us

- precision $p = 100\%$ and
- recall $r = 1\%$

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

We want measure that combines precision and recall.
F$_1$-Measure

- It is hard to compare two classifiers using two measures.
- 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.
- For F$_1$-value to be large, both $p$ and $r$ must be large.
F$_1$-Measure
### Alternative for Unbalanced Data: Cost 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>C(Yes</td>
</tr>
<tr>
<td>Class=No</td>
<td>C(Yes</td>
</tr>
</tbody>
</table>

\[ C(i|j) : \text{Cost of misclassifying class} \ j \ \text{example as class} \ i \]
## Computing Cost of Classification

### Cost Matrix

| ACTUAL CLASS | PREDICTED CLASS | \(C(i|j)\) | + | - |
|--------------|----------------|------------|---|---|
| +            | -1             | 100        |
| -            | 1              | 0          |

### Example:
- Credit card fraud.
- It is expensive to miss fraud transactions.

### Models

#### Model \(M_1\)

<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 \(M_2\)

<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
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)

• Drawing a ROC Curve
  – Sort classifications of positive class according to confidence scores
  – Evaluate
    • 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**

- **Best possible result:**
  - all correct predictions have higher confidence than all incorrect ones

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

- **Comparing algorithms:**
  - Curve A above curve B means algorithm A better than algorithm B

- **Frequently used quality criterion**
  - Area under ROC curve
3.2 Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?
- Performance of a model may depend on other factors besides the learning algorithm:
  - Size of training and test sets (it often expensive to get labeled data)
  - Class distribution (balanced, screwed)
  - Cost of misclassification (your goal)
- Methods for estimating the performance
  - Holdout
  - Random Subsampling
  - Cross Validation
Learning Curve

- Learning curve shows how accuracy changes with varying sample size.
- Conclusion: Use as much data as possible for training.
Holdout Method

- The **holdout method** reserves a certain amount for testing and uses the remainder for training.
- Usually: one third for testing, the rest for training
- Applied when lots of sample data is available.
- For "unbalanced" datasets, samples might not be representative
  - Few or none instances of the minority class/classes
- **Stratified sample: Balance the data**
  - Make sure that each class is represented with approximately equal proportions in both subsets by increasing the number of examples of the minority class.
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 (possibly with stratification)
  - The error rates on the different iterations are averaged to yield an overall error rate

- Still not optimal as the different test sets overlap.
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 value for \( k \) : 10

- Why ten? Extensive experiments have shown that this is the good choice to get an accurate estimate

- Often the subsets are stratified before the cross-validation is performed
X-Validation in RapidMiner

The diagram shows a process flow for evaluating a model using k-NN (k-Nearest Neighbors) for both training and testing. The process includes:

- **Training**:
  - k-NN node
  - Node for training data (tra)
  - Node for model (mod)
  - Node for test data (tes)
  - Node for performance evaluation (perf)

- **Testing**:
  - Node for applying the model (lab)
  - Node for model data (mod)
  - Node for performance evaluation (per)

The validation process is configured with:

- **Validation (X-Validation)**
  - Leave one out: unchecked
  - Number of validations: 10
  - Sampling type: Stratified sampling
Evaluation Summary

- Use Split-Validation sets for LARGE data sets
- Use Cross-Validation for SMALL data sets
- Balance “un-balanced” data by increasing the number of positive examples in the training set.
- Avoid Overfitting.
5. Rule-Based Classification

- Classify records by using a collection of “if…then…” rules
- Rule: \((\text{Condition}) \rightarrow y\)
  - where
    - \(\text{Condition}\) is a conjunction of attribute tests
    - \(y\) is the class label
- LHS: rule antecedent or condition
- RHS: rule consequent
- Examples of classification rules:
  - \((\text{Blood Type}=\text{Warm}) \land (\text{Lay Eggs}=\text{Yes}) \rightarrow \text{Birds}\)
  - \((\text{Taxable Income} < 50\text{K}) \land (\text{Refund}=\text{Yes}) \rightarrow \text{Cheat} = \text{No}\)
Example of a Rule-based Classifier

R1: (Give Birth = no) \(\land\) (Can Fly = yes) \(\rightarrow\) Birds  
R2: (Give Birth = no) \(\land\) (Live in Water = yes) \(\rightarrow\) Fishes  
R3: (Give Birth = yes) \(\land\) (Blood Type = warm) \(\rightarrow\) Mammals  
R4: (Give Birth = no) \(\land\) (Can Fly = no) \(\rightarrow\) Reptiles  
R5: (Live in Water = sometimes) \(\rightarrow\) 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>warm</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>fishes</td>
</tr>
<tr>
<td>turtle</td>
<td>cold</td>
<td>yes</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>no</td>
<td>yes</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 (LHS) of the rule.

R1: (Give Birth = no) \( \land \) (Can Fly = yes) \( \rightarrow \) Birds
R2: (Give Birth = no) \( \land \) (Live in Water = yes) \( \rightarrow \) Fishes
R3: (Give Birth = yes) \( \land \) (Blood Type = warm) \( \rightarrow \) Mammals
R4: (Give Birth = no) \( \land \) (Can Fly = no) \( \rightarrow \) Reptiles
R5: (Live in Water = sometimes) \( \rightarrow \) 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 hawk => Bird
The rule R3 covers grizzly bear => Mammal
Rule Coverage and Accuracy

- **Coverage of a rule:**
  - Fraction of records that satisfy the antecedent \((LHS)\) of a rule

- **Accuracy of a rule:**
  - Fraction of records that satisfy both the antecedent \((LHS)\) and consequent \((RHS)\) of a rule

\((\text{Status} = \text{Single}) \rightarrow \text{No}\)

**Coverage = 40\%, \ Accuracy = 50\%**

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

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>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
Solutions

- **Not Exhaustive Rule Set**
  - Some unseen records are not covered by the rules
  - Add default rule: () \(\rightarrow Y\)

- **Not Mutually Exclusive Rule Set**
  - An unseen record might be covered by multiple rules
  - Option 1: Ordered Rules
    - Order rules (e.g. prefer rules with high accuracy)
    - Classify record according to the highest-ranked rule
  - Option 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 rank 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 fired, it is assigned to the default class

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>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 each path in the tree.
- The generated rules are mutually exclusive and exhaustive.

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 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, \( r: A \rightarrow y, \)
   - consider an alternative rule \( r': A' \rightarrow y \)
     where \( A' \) is obtained by removing one of the conjuncts in \( A \)
   - Compare the pessimistic error rate for \( r \) against all \( r \)'s
   - Prune if one of the \( r \)'s has lower pessimistic error rate
   - 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 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 Step (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: $\{\} \Rightarrow \text{class}$
- Add conjuncts that maximize FOIL’s information gain measure.
Rule Growing within the Ripper Algorithm

- Add conjunct that maximizes FOIL’s information gain measure:
  - R0: {} => class (initial rule)
  - R1: {A} => class (rule after adding conjunct)
  - Gain(R0, R1) = t \log \left( \frac{p1}{p1+n1} \right) - \log \left( \frac{p0}{p0 + n0} \right) 
  - where
    t: number of positive instances covered by both R0 and R1
    p0: number of positive instances covered by R0
    n0: number of negative instances covered by R0
    p1: number of positive instances covered by R1
    n1: number of negative instances covered by R1
- Stop when rule no longer covers negative examples.
Rule Pruning

- Prune rule in order to
  - Avoid overfitting
  - Decrease its generalization error
  - Similar to post-pruning of decision trees

- Remove one of the conjuncts in the rule

- Compare error rate on validation set before and after pruning

- 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
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

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)
- Are well suited to handle imbalanced data sets
  - as they learn rules for the minority class first
- Can classify new instances rapidly
- Performance comparable to decision trees