Classification

Exercise 3
Classification

"tree"

"tree"

"tree"

"not a tree"

"not a tree"

"not a tree"
The Classification Workflow

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Training Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>

Unseen Records

Learning algorithm

Induction

Learn Model

Apply Model

Deduction

Model
K-Nearest-Neighbour

- Calculate the distance to all other points
- Choose the nearest K neighbours
- Let them vote for a class

- Requires
  - All known records
  - Distance metric

- Often very accurate
- But also slow
Operators: Set Role

• Input Port
  • Example Set

• Output Ports
  • Changed Example Set
  • Original Example Set

• Parameters
  • Attribute Name
  • Target Role

• Classification Operators need an attribute of type ‘label’
Operators: K-NN

- **Input Port:**
  - Training data (Example Set)

- **Output Ports**
  - Classification Model
  - Training data (Example Set)

- **Parameters**
  - K
  - Weighted Vote
  - Similarity Measure
Operators: Apply Model

• Input Ports
  • Model
  • Unlabelled data (Example Set)

• Output Ports
  • Labelled data (Example Set)
  • Model

• Classification Operators do not apply the model they learn!

  • You have to apply the model to *a different* example set
Process: Classification

• Learn the model from the training data
• Apply the model to the testing data
• Check the results
Naïve Bayes Classification

• If we know the prior probability and the likelihood
  • which we can estimate from the data
• Then we can calculate the posterior probability
  • Which we use for classification

\[ P(C|A) = \frac{P(A|C)P(C)}{P(A)} \]

• Prior Probability
  • \( P(A), P(C) \) “35.7% chance of rain”, “64.3% chance of play golf”
• Likelihood, given an observation
  • \( P(A|C) \) “33% chance of rain if go play golf”
• Posterior Probability
  • \( P(C|A) \) “66.7% chance of play golf if no rain”
Operators: Naïve Bayes

• Input
  • Training data (Example Set)

• Output
  • Classification Model
  • Training data (Example Set)

• Parameters
  • Laplace Correction

• Distribution Table (in results) shows posterior probabilities
Naïve Bayes in Rapid Miner

- Probabilities can be seen as confidences in the result

<table>
<thead>
<tr>
<th>Ro...</th>
<th>Play</th>
<th>prediction(Play)</th>
<th>confidence(no)</th>
<th>confidence(yes)</th>
<th>Outlook</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>no</td>
<td>yes</td>
<td>0.333</td>
<td>0.667</td>
<td>no rain</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td>yes</td>
<td>0.333</td>
<td>0.667</td>
<td>no rain</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>yes</td>
<td>0.333</td>
<td>0.667</td>
<td>no rain</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>yes</td>
<td>0.400</td>
<td>0.600</td>
<td>rain</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>yes</td>
<td>0.400</td>
<td>0.600</td>
<td>rain</td>
</tr>
<tr>
<td>6</td>
<td>no</td>
<td>yes</td>
<td>0.400</td>
<td>0.600</td>
<td>rain</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>yes</td>
<td>0.333</td>
<td>0.667</td>
<td>no rain</td>
</tr>
<tr>
<td>8</td>
<td>no</td>
<td>yes</td>
<td>0.333</td>
<td>0.667</td>
<td>no rain</td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>yes</td>
<td>0.333</td>
<td>0.667</td>
<td>no rain</td>
</tr>
<tr>
<td>10</td>
<td>yes</td>
<td>yes</td>
<td>0.400</td>
<td>0.600</td>
<td>rain</td>
</tr>
<tr>
<td>11</td>
<td>yes</td>
<td>yes</td>
<td>0.333</td>
<td>0.667</td>
<td>no rain</td>
</tr>
<tr>
<td>12</td>
<td>yes</td>
<td>yes</td>
<td>0.333</td>
<td>0.667</td>
<td>no rain</td>
</tr>
<tr>
<td>13</td>
<td>yes</td>
<td>yes</td>
<td>0.333</td>
<td>0.667</td>
<td>no rain</td>
</tr>
<tr>
<td>14</td>
<td>no</td>
<td>yes</td>
<td>0.400</td>
<td>0.600</td>
<td>rain</td>
</tr>
</tbody>
</table>
Evaluation

• How do we know the model actually works?
  • By counting the number of errors
  • On a *different* dataset

• What’s the purpose of a model?
  • To apply it to new data where we don’t know the label

• What happened if we used the same dataset?
  • How many errors for a K-NN classifier with K=1?
  • How good would that model be on a different dataset?
For every class in our dataset, the classifier can produce one of four possible results:

<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>True Positive (TP)</td>
</tr>
<tr>
<td>Class=No</td>
<td>False Positive (FP)</td>
</tr>
</tbody>
</table>
Evaluation Measures: Accuracy

• A single measure that tells you the overall accuracy of the result

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

• “Number of correctly classified examples divided by the total number of examples.”

• Problem: Unbalanced data
  • If 99% belong to class “yes”
  • And classifier always says “yes” – 99% Accuracy
Evaluation Measures: Precision and Recall

• Measure two aspects of the result for every class

• Precision: How many of the examples that were labelled “yes” are really “yes”?
  • “the number of correctly labelled examples divided by the number of all examples that were labelled with this class”

• Recall: How many of the examples that are really “yes” were labelled “yes”?
  • “the number of correctly labelled examples divided by the number of all examples that actually belong to this class”

\[
\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}
\]
### Evaluation Measures: Precision and Recall

#### An example:

<table>
<thead>
<tr>
<th>ID</th>
<th>Prediction</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**Precision**

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

**Recall**

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

#### For class “yes”

- 1 true positive (ID 1)
- 2 false positives (ID 2 & 4)
- 1 true negative (ID 3)

\[
\text{Precision}_{\text{yes}} = \frac{1}{1 + 2} = \frac{1}{3}
\]

\[
\text{Recall}_{\text{yes}} = \frac{1}{1 + 0} = \frac{1}{1}
\]

#### For class “no”

- 1 true negative (ID 1)
- 2 false negatives (ID 2 & 4)
- 1 true positive (ID 3)

\[
\text{Precision}_{\text{no}} = \frac{1}{1 + 0} = \frac{1}{1}
\]

\[
\text{Recall}_{\text{no}} = \frac{1}{1 + 2} = \frac{1}{3}
\]
Operators: Performance (Classification)

• Input
  • Labelled Example Set

• Output
  • Performance

• Parameters
  • Performance Measures
Split-Validation / Cross-Validation

• What can you do if you only have one dataset?
  • Use one part of the data for training
  • Use *the other part* of the data for testing

• What if by accident all the easy examples are in the training set?
  • Then your model will not perform that good
  • Better to repeat the learning on different splits of the data

• X-Validation (Cross-Validation)
  • Split the dataset into X parts
  • Select one part for testing, use the rest for training
  • Repeat this until every part was used for training once
Just a reminder ...

• If you use the same data for training and evaluation...

• ... there will be no Christmas!
Operators: Split Validation / X-Validation

• Input Port
  • Training data (Example Set)

• Output Ports
  • Classification Model
  • Training data (Example Set)
  • Averageable 1 ... n

• Parameters
  • Split type
  • Split ratio
  • Sampling type
Nested Processes in Rapid Miner

- Operators can have “inner” processes that define their behaviour
- Split/X-Validation Operators have a “Training” and a “Testing” phase
  - Training: This is where you learn your model
  - Testing: This is where you evaluate
From two datasets to Split-Validation
The Mannheim RapidMiner Toolbox

• A Rapid Miner Extension with many great operators

• Developed by researchers from the Data and Web Science Group

• Contains the nearest centroid classifier