

# B.Sc. in Computing

## Data Mining

Lab sheet #5 = k-NN and Data Preparation



# Overview

## Objective:

- ◆ Explore data preparation techniques

## ◆ Agenda:

- ◆ Handling missing values
- ◆ Sampling a dataset
- ◆ Selecting Rows and Columns
- ◆ Binning attributes in a dataset
- ◆ Normalise attributes

# Exercises

There are a number of exercises through out the slides, and repeated again on the last slide.

Note your answers, and complete the moodle quiz when finished.

**YOU HAVE TWO WEEKS TO COMPLETE THIS  
LABSHEET**

# Using nearest neighbour

**Start** a new process, and read in the **labor-negotiations** dataset from the samples repository.

**Run** the process to view the dataset.

- How many rows does it have?
- How many rows are in each class?

**Add** an **x-validation building block** to the process, but replace the decision tree operator with a **k-NN** learner (modeling / classification and regression / lazy learners).

The default value of k in kNN is 1. Run the process with this value.

**Exercise: How accurate is the model with k=1?**

Try some other values of K to see if you can improve its accuracy, such as k=5, k=10, k=20.

**Exercise: How high can k go before accuracy starts to drop?**

**The remaining slides focus on  
data preparation techniques**

# Handling missing values

1. **Start** a new process calling it `lab5-charity`
2. Load `charityWithMissingData.csv` into your repository, making the final column, `Response To Campaign`, the class label.
3. Retrieve the dataset into your process, and view its meta data

Exercise 1: How many rows in the dataset, and what is the quality of the dataset in terms of missing values?

Exercise 2: For each column with missing data, how would you advise handling the missing data from the options below:

delete the column

delete the row

replace the missing value.

# Handling missing values

The following slides will deal with all the missing values in the dataset in a variety of ways, starting with the attribute which has the **most** missing values. . . .

This will illustrate how to:

1. select just some of the attributes
2. select just some of the rows
3. fill missing values using the average
4. fill missing values using other methods

# Deleting a column

In Rapidminer, you don't actually delete a column from the dataset, you simply ignore it using the 'select attribute' operator.

We are now going to filter out 'hasChildren' because of the amount of missing values in this column:

Add select attributes to your process and connect it to the retrieve operator. Take a look at it's parameters. Under attribute filter type select single. Click the drop down box to view the list of attributes in the dataset. Select hasChildren.

This will filter all other attributes from the dataset and just select hasChildren. We want the opposite of this, so we click on invert selection.

Run the process – hasChildren should no longer be visible in the output.



# General information on selecting attributes

Select attributes has a range of criteria under which you can select columns to include / exclude from the dataset.

Attributes can be selected by:

**Name** (single, subset, regular expression to identify a group of attribute names)

**Data Type** (value\_type, block-type (groups of attribute types))

**Quality** (remove columns with missing values)

**Based on the range of their values** (for numeric attributes)

To do: Try some of these options with the CHARITY dataset.

By default, you are selecting which attributes to INCLUDE.

Invert selection means you are specifying which attributes to EXCLUDE.

# Replacing Missing Values

The column 'Title' has about 10% of the rows will missing values. This is too much to delete the rows, and not enough to delete the column, so we will try to replace the missing values. There are a number of approaches we could use, for example:

1. Replace by the mode for that column  
OR
2. Use Gender to infer title (e.g. for males, replace with MR, and for females, replace with MRS)

# Replacing Missing Values

Taking option 1 first:

Add a **Replace Missing Values** operator to the process, and connect it to the output from **Select Attributes**.

Click on the operator to view its parameters.

For **attribute filter type**, select **Single** (we just want to work with one attribute for now).

For **attribute** select **Title**

For **default**, select **average** (which uses mode for nominal column).

Note: Under **Columns**, you can specify different replacement approaches for different columns.

Run the process.

# Replacing Missing Values

Exercise 3: Looking at the data view:

How many of the remaining rows have missing data?

What is the best way to handles these?

# Selecting / Deleting Rows

In Rapidminer, you don't actually delete rows from the dataset, you simply ignore them using the 'filter examples' operator.

We are now going to filter out the few remaining rows that have missing values:

Add filter examples to your process and connect it to the retrieve operator. Take a look at its parameters. Under condition class select no\_missing\_attributes. Run the process

Exercise 4: How many missing values are now remaining in the dataset?

# General information on **example filter**

Like Select Attributes, Example Filter has a range of criteria under which you can select rows to include / exclude from the dataset. Rows can be selected by:

**Quality** (remove rows with missing values or missing a label)

**Value** (include/remove rows based on a condition which checks values in a particular attribute, e.g. Age>40 or Title=MR, or Post Code = PE.\*)

**Data Type** (value\_type, block-type (groups of attribute types))

**Accuracy** – remove rows where the prediction is correct / incorrect.

To do: Again, try some of these options with the CHARITY dataset.

By default, you are selecting which attributes to INCLUDE. Invert selection means you are specifying which attributes to EXCLUDE.

# Alternative methods to fill missing values

An earlier slide mentioned an alternative way to fill the missing values in the Title column:

Use Gender to infer title (e.g. for males, replace with MR, and for females, replace with MRS)

This is a bit more complicated to do in RapidMiner. There is process showing how to do this is on moodle, called **lab5-charity-CalculateTitle**. Take a look at it to see if you can follow it. It splits the dataset in three part:

1. Females with missing titles – replaces title with MRS
2. Males with missing titles –replaces title with MR
3. Rows without title missing

All three groups are then merged using **APPEND**.

Note: When an operator outputs '**ori**' it means the dataset it received as input, before it carried out any changes

# Sampling the dataset to reduce the number of rows



# Sampling

Return to your own process – [lab5-charity](#).

Add an [x-validation block](#) to model the dataset set using a decision tree. Output the [dataset](#), [model](#) and [performance](#) (ave)

**Exercise 5:**

**How accurate is the decision tree?**

The dataset currently has about 2000 rows. Does it need this many rows or would it be as accurate with fewer rows? To find out, we can try sampling the dataset.

# Sampling

Place a **sample** operator before the X-Validation block.  
Set sample size to 1000.

How accurate is the decision tree now? And is it a simple or complex tree?

**Exercise 6:**

How small can the sample size go while maintaining accuracy and not overtraining the decision tree?

# Calculating new columns

# Generate Attributes

Sometimes, attributes derived from the dataset can be more predictive than the original attributes.

The Generate Attribute operator supports a number of functions which can be applied to both numeric and nominal data to generate new data.

We are going to use generate attributes to calculate the average spend per visit as follows:

Average spend per pre-campaign visit =  
 $\text{preCampaignExpenditure} / \text{preCampaignVisits}$

# Generate Attributes

Add a **Generate Attributes** operator to your process just before the X-Validation block. **Disable the X-Validation block for now.**

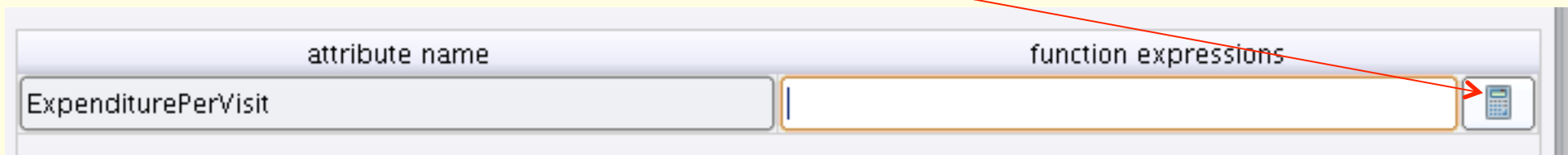
Select the operator to view its parameters.

Click on **Edit List**.

This provides you with a window to define attributes you want to generate. Click **Add Entry** (at the bottom)

The left hand side is the name of the new attribute, the right hand side is the expression to calculate the new attribute.

Call this new attribute **ExpenditurePerVisit**. Click on the calculator ICON to see the functions available to you . . .



attribute name	function expressions
ExpenditurePerVisit	

# Generate Attribute

All attributes are listed on the right hand side. Double click on an attributes to add it to the expression.

All functions available are on the left hand side. Select different options under Type to view all functions. Hover the mouse over a function to get its description.

Add the function:

**PreCampaignExpenditure/PreCampaignVisits**

Click OK (twice) to return to the process and run it

Expression: PreCampaignExpenditure/PreCampaignVisits

✓ Expression is syntactically correct.

☐ Allow Unknown?

Functions

Type: Basic Operators

Attributes

- 52MosaicGroups
- Age
- Gender
- MosaicBands
- PostCode
- PreCampaignExpenditure
- PreCampaignSpendCategory
- PreCampaignVisits**
- PreCampaignVisitsCategory

# Binning

# Binning – general information

RapidMiner has 5 operators for Binning, located under data transformations / type conversion / discretization.

**Discretize by Binning:** This performs **equiwidth** binning – converting all numeric attributes into nominal attributes. A specified number of equally sized bins is created.

**Discretise by Entropy:** bin boundaries are select with respect to the class variable, aiming to have all examples in a bin belonging to the same class.

**Discretize by Size:** the user specifies the size if each bin, and RM creates bins of equal size – (same number of **examples**). This is a form of **equi depth** binning.

**Discretize by Frequency:** the user specifies how many bins to create, and RM produces bins of equal frequency (same number of **values** in each). **Equi depth** binning

**Discretise by User Specification:** the user can specify mappings from numeric values to nominal or ordinal values.



# Binning

We are going to bin age into three categories: young, middle-aged and senior.

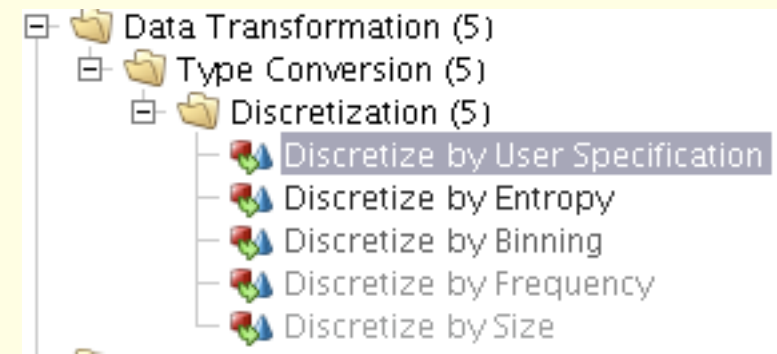
Young:  $< 30$

Middle Aged: between 30 and 55

Senior:  $> 55$

Note: in this example, we are defining the bin boundaries ourselves, so we will be using **user specified binning**.

# Binining



Enter **discretize** in the operator filter to look at the binning options.

Add **Discretize by User Specification** to the end of your process. Click on the operator to view its parameters:

As with other data preparation parameters, we specify which attributes we are working on, in this case it is a single attribute – age.

attribute filter: single

attribute: age

Scroll down to the parameter called **classes**, and click on it's edit list.

# Binning

This is where you specify the upper boundary for each bin, starting with the lowest number range (youngest ages).

Remember we are defining three bins: **young**, **middle** aged and **senior** as follows:

class names	upper limit
young	30.0
middle	55.0
senior	100.0

Enter the values above (**Add Entry** adds a new row), click OK and run the process to see the new list of values for **Age**.

# Evaluating preparation techniques

Re-enable the X-Validation block, putting it last on the operator list, and run the process again.

Exercise 7:

Were any of the additional attributes you generated useful (do they appear on the tree)?

# Evaluating preparation techniques

Data Preparation is in iterative process where you try something, and then use **X\_Validation** to see if that improves model accuracy, if not you go back and try another idea.

Decision trees work well with discretized data. Replace the discretize operator with **discretize by entropy**.

In the parameter values, select all numeric attributes:  
**attribute filter: value type**  
**value type: numeric**

Run the process again.

**Exercise 8:**

**By how much does binning by entropy improve the accuracy of the decision tree?**

# Normalise a dataset

Scale all numeric attributes to a similar range

# Normalise

Normalising numeric attributes is important for mining algorithms that use distance calculations such as k-NN and clustering algorithms.

To see how this operator works, look at the following process: [samples/02\\_preprocessing/01\\_normalisation](#)