# COMP3430/COMP8430 – Data Wrangling – 2019

# Lab 5: Classification for Record Linkage Week 7

## Overview and Objectives

In today's lab we continue with our record linkage system that we started in labs 3 and 4, this time looking at the classification step as discussed in lectures 17 and 18 (week 7). Make sure you view these lectures before coming into the lab session.

There are many different classification models used for record linkage, ranging from simple exact classification, through to complex machine learning techniques that incorporate relationships and other features. Essentially all of them are binary classification models, i.e. given a pair of records (or a weight vector of similarities for a pair of records) they aim to classify a given pair of records as either a match or a non-match.

As with lab 4, we provide you with a skeleton Python module `classification.py` (available in week 5 in the archive `comp3430_comp8430_reclink-lab-3-6.zip`) which contains a simple classification function, and ask you to implement some more classification functions yourself.

## Lab Questions

As a group, discuss the classification outcomes for the following two vectors of similarities (assumed to have been calculated by the comparison functions as we discussed in the previous lab):

- Comparing record pair *(r1,r2)* resulted in: [0.5, 0.9, 0.2, 0.8, 1.0, 0.0, 0.7]
- Comparing record pair *(r3,r4)* resulted in: [0.8, 0.7, 1.0, 0.9, 0.6, 0.7, 0.9]

Apply the following two classification methods (as discussed below):

- **Threshold based classification**, using the average similarity value across all attributes in a given weight (similarity) vector (calculate the average of all similarities and check if this average is at least the given classification threshold for the record pair to be classified a match).

  Set the thresholds for the average similarities to (a) 0.5 and (b) 0.7.
- **Minimum threshold based classification**, which requires all the similarity values in a given weight vector to be at least the provided threshold in order to classify a record pair as a match.

  Set the thresholds for the minimum similarities to (c) 0.5 and (d) 0.7.

What are the classification outcomes of (a) to (d)?

Moving on to the Python programs, begin by revising what we did in labs 3 and 4 and how the overall record linkage program is setup. Then open the module `classification.py` in a text editor and have a look at the function `exactClassify()` which we have provided, to see what the inputs and outputs for this function are. You can test this on some of the example data and see how it works. Once you understand how the classification function works and your are comfortable with what needs to be done, then implement the following three classification functions yourself (otherwise ask your tutor to explain how the `exactClassify()` function works):

1. **Threshold based classification** as discussed above.
2. **Minimum threshold based classification** as discussed above.
3. **Weighted average based classification**, which is similar to first function above, but you have to provide a vector of numerical weights (assumed to be non-negative) for the attributes in the similarity vector, and use a weighted average instead of the simple average. Think about how to normalise a weighted average such that it again will be in the 0 to 1 range.

As usual, once you have finished your implementation, please experiment by testing the different classification techniques on the different data sets. **Note that the focus of this lab is not on the implementation only, but also about your understanding of how different classification techniques for record linkage work. For students doing COMP3430, this will be important for the upcoming final project in the Data Wrangling course.**

Some questions to explore are:

1. How many matches do you find with each classification technique for the same threshold value?
2. Do different thresholds have an effect on the number of matches?
3. How do different weights for the weighted average function influence the number of matches?
4. Are some of the results clearly wrong?

If you finish implementing the techniques above, the function `supervisedMLClassify`, implements a decision tree classifier (as covered in a data mining course), so please experiment with it and explore how it works.

You may also wish to modify the weights for the weighted average based classifier above, so that rather than manually assigning weights to the different attributes, you can make it related to the distribution of values. One possible weighting scheme is to calculate the weights for an attribute as $log_2|\mathbf{U}|$ where $|\mathbf{U}|$ is the number of unique values in the attribute. For example, for gender $|\mathbf{U}| = 2$ (resulting in a weight of 1.0), for month $|\mathbf{U}| = 12$ (resulting in a weight of 3.58), and so on.

If you still have time then please feel free to look at any other binary classification technique that you are interested in. The Python library `scikit-learn` (sklearn) has implementations of many different machine learning classifiers and (in theory) any of them can be applied to record linkage. Please ask us if you would like some suggestions or advice to implement them.