MDMI Assignment

Data Mining, IT-University of Copenhagen
Niels Søholm - nmar@itu.dk - 2014-03-26

Frequent Pattern: Programming language patterns

My goal is to find patterns in student favorite choice of programming language. I'm going to do this by running **Apriori** on preprocessed version of the *"favorite programming language"* attribute.

I'm expecting similar languages to occur together often. I also have a feeling that languages commonly taught at ITU (java and C#) might be frequent. We may see that certain people have a tendency to favor web development centric languages such as PHP and JavaScript. Or certain groups may favor languages commonly associated with statistics such as Matlab and Python.

The programming language column contains lists of programming languages separated by commas or spaces. My approach to deal with this was to try and split the lists into single languages. I counted the frequency of each of the resulting languages and ultimately set a threshold of at least 2 occurrences. This is simply to filter out invalid output and focus on the values that may actual give interesting patterns. I'm likely to miss a few languages, but if they don't occur more than once, then they won't produce any frequent patterns anyway. I would characterize this as a form of **attribute construction**. I produce a number of tuples containing the new values, in essence i'm introducing a number of new attributes for each student indicating whether or not they have one of the now finite nominal attributes.

The result is represented by the table below. These are the strong patterns (support threshold: 10%) including their association rules and lift:

Pattern	**Support**	**Lift**
'java', 'python'	0.210	0.806
'c#', 'java'	0.419	0.983

'c#', 'java', 'php'	0.226	1.391
'c#', 'java', 'python'	0.129	0.382

C# and Java has a very high support, which was also expected by my hypothesis, however the lift is a bit disappointing. This is likely due to the fact that C# itself has an extremely high probability. There are simply too many C# favoring students to backup the pattern despite the high support.

The only pattern with a significant lift is 'c#', 'java', 'php'.

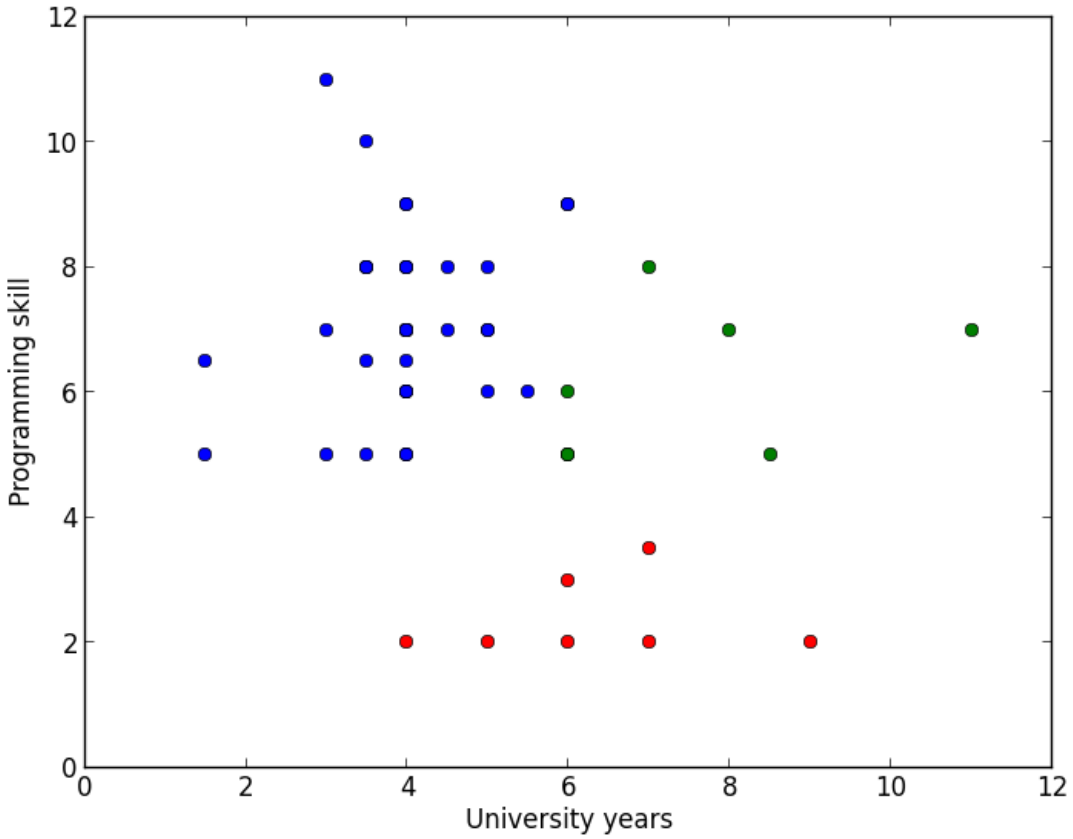
Clustering: University years and programming skill

My goal is to cluster students based on university years and programming skill.

I'm going to do this by running **K-means** on cleaned versions of the 2 columns “uni_yrs” and “prog_skill”. Cleaning involves parsing input strings to floating point values. As well as **outlier removal** based on the ranges defined in the original questions from the questionnaire.

My hypothesis is that there ought to be some correlation between university year and ones ability to programming. At least for the majority of students. The hypothesis stems from the expectations that students have been studying computer science related topics throughout most of their years. Since this is an elective master's course we may see clusters of people who diverge from this hypothesis, perhaps because of a different background or higher level of modesty about their own abilities.

I have experimented with k-values of 2, 3, 4. I have verified using 2D plotting. 3 seems to be a good number of clusters:



The 3 clusters might be individually described as:

- (blue) A very strong cluster of students in the early to mid of their studies who feel very confident in their own programming skills.
- (red) A cluster in the mid to late end of their studies who have very modest programming skills
- (green) A few students in their late studies with mid to high programming skills

My hypothesis about a correlation between programming skill and university years does not seem to hold. In fact, if we are to believe the students' self-evaluation, there seem to be a strong cluster of student (blue) who diverge directly from this hypothesis.

Classification: Operating system

My goal is to construct a classifier capable of determining the favorite operating system of a

student based on their favorite SQL server and programming skill.

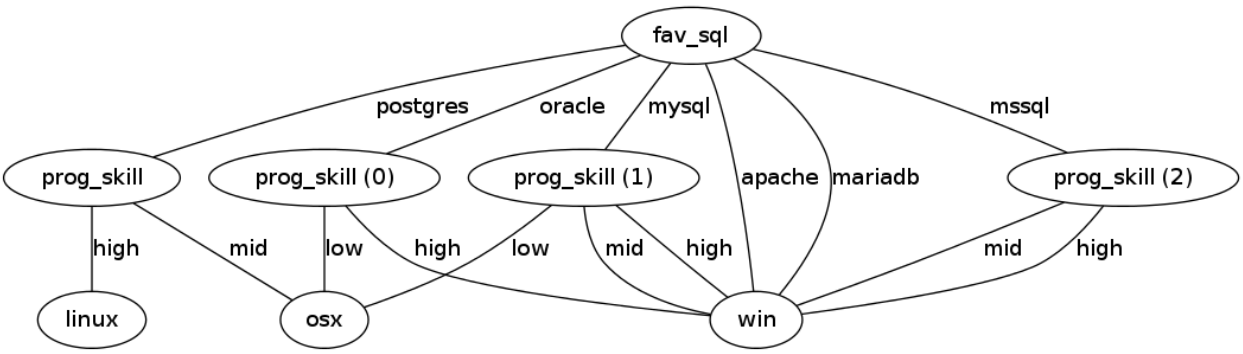
I'm going to do this by constructing a **decision tree using ID3** algorithm. I will use cleaned versions of the 3 columns "prog_skill", "OS" and "FavSQLServ".

The operating system data calls for a bit of data transformation. Many abbreviations and versions of the different OSs occur in the dataset. One could argue that for our purposes it is more interesting to consider OSs on a conceptually higher-level. The compromise has been to do **concept hierarchy generation** based on explicit grouping. The groups are "Windows", "OSX" and "Linux-based". I accomplish the grouping by introducing a number of regular expressions (e.g. "linux|debian|ubuntu") for each group and then grouping the students based on which expressions their input matches.

The favorite SQL servers attribute is cleaned in a similar fashion.

I decided to do some preprocessing of the programming skill data. The data is used as numeric data in the clustering section above, however in this section nominal data is preferable. Otherwise we will get a very broad tree where programming skill may split into 10 different options. This would likely increase risk of overfitting and make the tree less comprehensible. My solution is to introduce **range-based concept hierarchies**. So i have 3 different nominal options: low (1-3), mid (4-7), high (8-10)

The resulting decision tree looks like this:



I used cross-validation to verify the accuracy. There is very little data (I ended up with 40 valid records). So i used 4 runs with 30 training records and 10 test records in each run

Confusion matrix of a single test run

classes	**win**	**osx**	**linux**	**recognition (%)**
win	6	0	0	100.00%
osx	1	2	0	66.67%
linux	1	0	0	0.00%
total	8	2	0	75.00%

The average accuracy for each class respectively:

- Windows: **95%**
- OS X: **55 %**
- Linux: **0%**

As the data indicates, the classifier is pretty bad at classifying anything but Windows. Since most of the students use Windows this is not such a great feat. It does however manage to classify OS X users with more than 50% accuracy.

Conclusion

There are a few patterns in MDMI student language preferences. However we only found a single 3-tuple pattern with significant lift. Some vague clusters can be found when considering programming skill vs. university years. The amount of data seems to be too little to classify operating system, this may also be because of the high amount of Windows users attending the class.

For some attributes the noise is great. When the data is nominal string input it is hard to make a qualified interpretation of noisy or invalid entries. My decision to, in most cases, discard entries with invalid data may have been a bad one. On the other hand, there is likely too little data to make a proper substitute estimate. In general, all conclusions seem to suffer from lack of qualified data.