Jan Aagaard Meier. jmei@itu.dk

Data Cleaning

The first part of my data cleaning consisted of getting more data. Because face it, you can never have too much data! I did this by merging the two datasets from 2013 and 2014 students. However, the questions for the two datasets are not completely identical. Some of the row headers in the csv files are phrased slightly differently, while other are missing completely. I merged the two files (Data_Mining_Student_DataSet_Spring_2013_Fixed.csv and data_mining_2014_dataset.csv) manually, taking an optimistic approach by believing that all columns headers which seem sort of the same are actually the same question. All questions which are only in one dataset have been removed. This means that the columns Georgios_middleName and JulianHome have been removed from the 2014 version, and the columns noor_hometown, num_sequence and seq_name have been removed from the 2013 version. Furthermore, the columns asking for random numbers have been removed because I believe them to have a doubtful information value.

The following remaining features have been discretized: age (young / slightlyyoung / meh / old), favorite animal (elephant / zebra / asparagus), more mountains in Denmark (yes / no), fed up with winter (yes / no), knows neural networks (yes / no), knows SVMs (yes / no), knows SQL (yes / no), knows apriori (yes / no), favorite color (blue / green / red / orange / black / yellow / none) - based on manual observation of the most common colors in the dataset, all colors with support greater than 3 have been included.

In addition, the attributes programming skill, uni_years and english skill have been normalized using z-score normalization.

Lastly, I have preprocessed the programming languages attribute in two ways, depending on the training algorithm used. For classification using ID3, i converted the attribute into a single discrete attribute, loves_javascript, depending on whether the list of languages contained javascript or not, and for pattern mining using apriori, I converted the string of languages into a list, and did a manual discretization of the values to account for differences in capitalization, misspellings etc.

Classification

For classification i used the ID3 algorithm with information gain as my attribute selection measure. The recent years have seen a rise in the buzz around javascript, and many developers either love it

or hate it. Therefore I decided to attempt to build a classifier that can help you easily determine whether javascript will be a good topic to begin talking about whenever you meet another developer. I do this by running the ID3 algorithm with the loves_javascript attribute as my target attribute.

Initially a ran the algorithm on all the attributes on the dataset. However, it turned out, that the numerical attributes english skill and programming skill had too high information spread, meaning that they were selected as the first attributes by ID3, leading directly to a classification of true / false, based solely on the value of those attributes. A section of this tree can be seen in fig. 1, where english_skill = 56, 62 and many others lead directly to the conclusion loves_javascript = false. Because of the small size of the dataset, the tree in fig. 1 has been overfitted to the training data, and doesn't generalize very well.

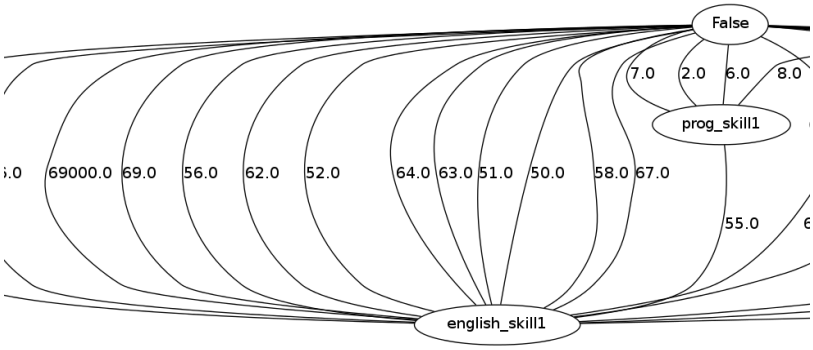

Fig. 1 - A section of the decision tree for 'loves_javascript' with numerical attributes

With the numerical attributes removed we get a far more manageable tree. An example of one such can be seen in fig. 2.

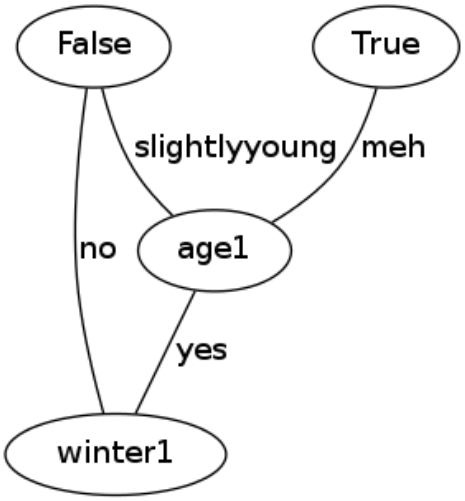

Fig. 2 - A more manageable decision tree for the class 'loves_javascript'

This decision tree has been created by randomly splitting the test data into a training and a validation set (representing ⅘ and ⅕ of the total dataset respectively). I did several runs with different splits, and used the one which had both relatively high precision and recall (seen in fig. 2). Table 1 below shows the confusion matrix for the tree. In addition I have calculated the *precision*: 0.25, and the *recall*: 0.34.

Classes	loves_javascript = yes	loves_javascript = no	Total	Recognition
loves_javascript = yes	2 TP	4 FN	6	33.34 %
loves_javascript = no	6 FP	47 TN	53	88.68 %
Total	8	51	59	83.05 %

Table 1 - Confusion matrix for the decision tree in fig. 2

Given the small dataset, two random splits can generate very different trees. Many of the trees generated yielded no true positives, and several trees yielded no positives at all. Because the data is skewed towards false, the "tree" in fig 3. yields an error rate of 0.12, while fig. 2 yield an error rate of 0.17.

Fig. 3

For the question of "*should i talk with this developer about javascript*," the short answer seems to always be "*no*".

Pattern mining

For pattern mining I have decided to mine common patterns among favorite programming languages, using each entry in the prog_langs column as a transaction. The questionaire asks for 3 favorite programming languages, but since I use apriori for pattern mining the transactions can have any length, which means I have also included answers that included 1, 2, and 4 programming languages. I have done mining on the datasets from 2013 separately, and on the two datasets merged together. Tables 2 and 3 below show the patterns and their support.

2013

C#, Java	18
Java, C++	6
C#, F#	6
Python, Java	5
C#, Python	5
C#, C++	4
C#, C	4

2014

C#, PHP, Java	7
C#, Python, Java	4

Merged

C#, PHP, Java	8
C#, Python, Java	7
C#, Java, C++	6
C#, F#, Java	5

Table 2 - Patterns generated by apriori with minimum support 4

Association rules created using the rule with highest support for each dataset:

2013

C# => Java = 18 / 25 = 72%

Java => C# = 18 / 25 = 72%

2014

C#, PHP => Java = 7 / 8 = 87.5%

C#, Java => PHP = 7 / 24 = 29.17%

PHP => C#, Java = 7 / 12 = 58.33%

Java => C#, PHP = 7 / 42 = 16.17%

C# => Java, PHP = 7 / 42 = 16.17%

Merged

C#, PHP => Java = 8 / 10 = 80%

C#, Java => PHP = 8 / 49 = 16.33%

PHP => C#, Java = 8 / 14 = 57.14%

Java => C#, PHP = 8 / 67 = 11.94%

C# => Java, PHP = 8 / 67 = 11.94%

2013

C#, Java	18

2014

C#, Java	28
Python, Java	14
Python, C#	11

Merged

C#, Java	46
Python, Java	19
C#, Python	16
C#, F#	13
C#, PHP	10

Table 3 - Patterns generated by apriori with minimum support 10

Association rules created using the rule with highest support for each dataset:

2013	*2014*	*Merged*
C# => Java = 18 / 25 = 72%	C# => Java = 28 / 42 = 66.67%	C# => Java = 46 / 67 = 68.66%
Java => C# = 18 / 25 = 72%	Java => C# = 28 / 42 = 66.67%	Java => C# = 46 /67 = 68.66%

Unfortunately I did not have time to do further evaluation of the mined association rules.

Clustering

For clustering, I have used a combination of k-means and k-modes. For numerical attributes, I have calculated the mean, and used subtraction to calculate the distance. For nominal attributes, I have calculated the mode, and calculated the distance as 0 if two rows have the same value for the attribute, and 1 otherwise.

To determine the right *k* value for the data I have used the elbow method, and plotted the sum of squared intra-cluster variance for each k value for 1 to 30. The graph is seen in fig. 4.

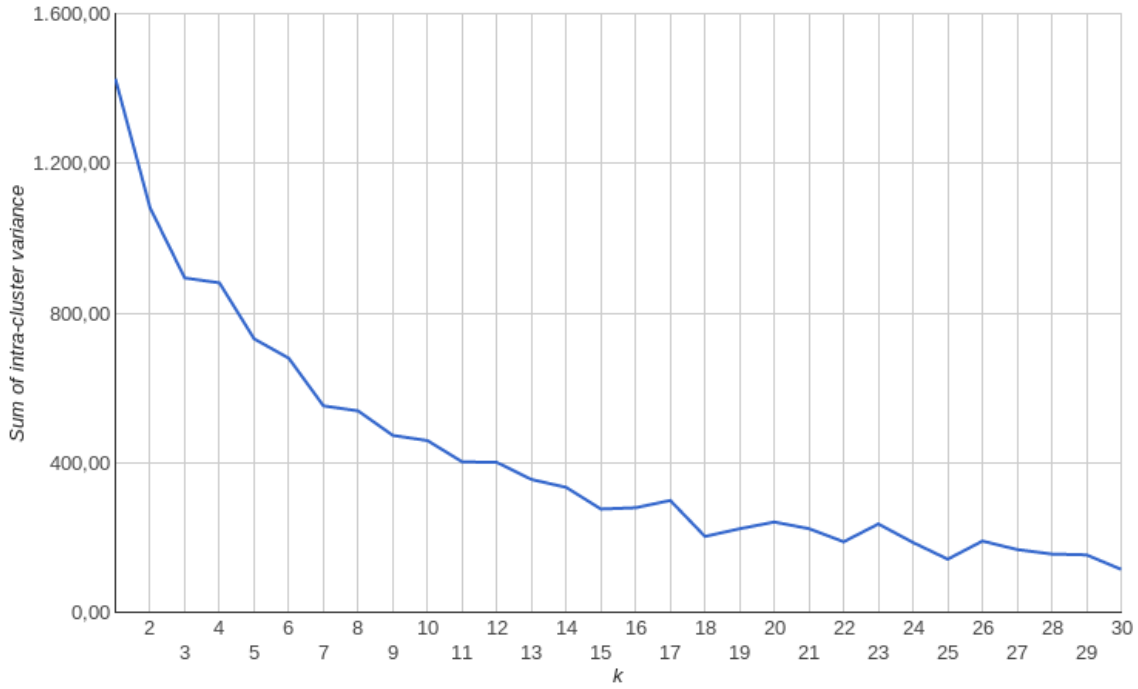

Figure 4 - The sum of squared intra-cluster variance for k-means with values 1 to 30 for the questionaire dataset

Unfortunately, no clear elbow shows on the plot above, the plot just more or less trails of. As a comparison, the elbow plot for the dataset provided for the k-means exercise (iris.csv) in fig. 5 shows a clear elbow at 3.

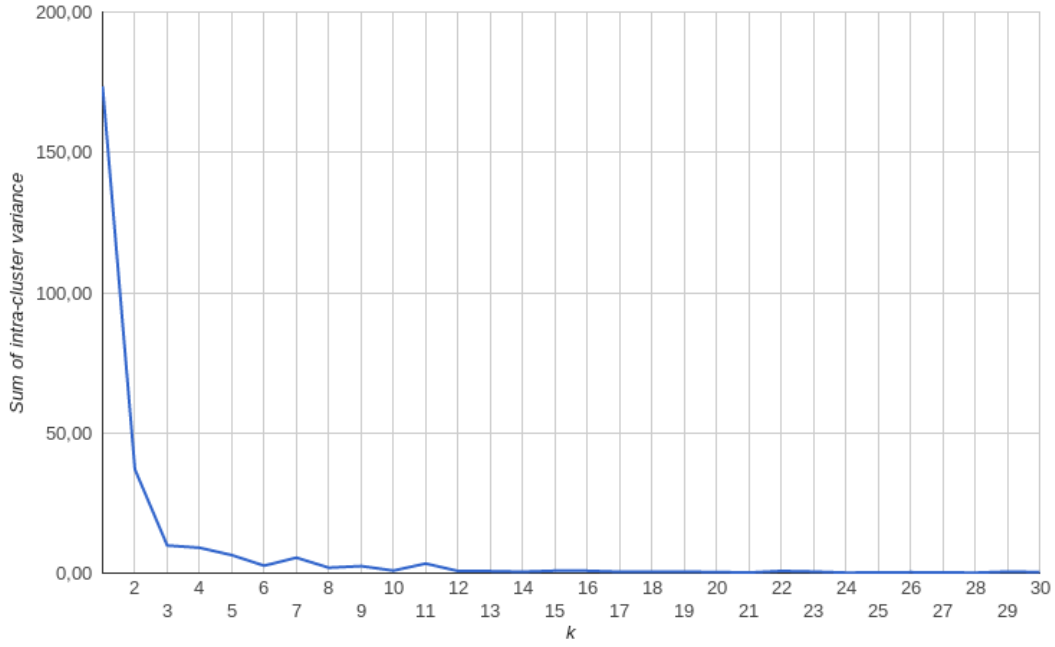

Figure 5 - The sum of squared intra-cluster variance for k-means with values 1 to 30 for the iris dataset

Since the elbow method yielded no clear result about the optimal value for k. If I had more time, I would have probably have used either cross-validation or silhouettes to determine the optimal k value, or used a completely different approach, such as self organising maps, which don't need a preset k value.