Data Mining ITU 2017 Spring - Individual Assignment

Richard Banyi, *Student, ITU*

***Index Terms*—Data-mining, preprocessing, KNN classification, K-Means Clustering, Apriori - frequent pattern mining.**

I. Data Preprocessing

THe first part of the preprocessing was to getting know the dataset and explore if the data quality satisfy the requirements of the intented use. Firstly I have loaded the dataset and examined the dimensionality and the accuracy, completeness, consistency of the data. The input dataset consisted of 67 rows x 45 colums. After the dataset was identified I have created several methods for data cleaning for filling in missing values, filtering out numbers with regex, convert to float data types, replacing inconsistencies.

A. Dimensionality Reduction

After the data was cleaned, I have picked the features that I have decided to used for implementing KNN for classification, KMeans for Clustering and Apriori frequent pattern mining. The subset of features I have used are: *age, shoe_size, height, language, gender.* Most of the features like *age, shoe_size, height* are discreate features, I have used descriptive statistics to see the distribution of these features. The other 2 features were nominal *gender* and *language*

TABLE I
Descriptive Statistic

	age	shoe_size	height
count	67	67	67
mean	40.701493	41.537313	175.298507
std	118.907784	5.915640	25.702856
min	22.000000	2.000000	34.000000
25%	24.000000	40.750000	172.000000
50%	25.000000	42.500000	180.000000
75%	28.000000	44.250000	186.500000
max	999.000000	49.000000	205.000000

B. Data Transformation

Through the use of descriptive statistics, it was clear that the discreate features need to be normalized. Therefore I have used Z-score normalization to ensure that all the features are rescaled and contribute equally, which was crucial in measuring Euclidean distance.

Alternaly I have also used Min-Max scalling, in this case the data is scaled to a fixed range bettween 0-1.

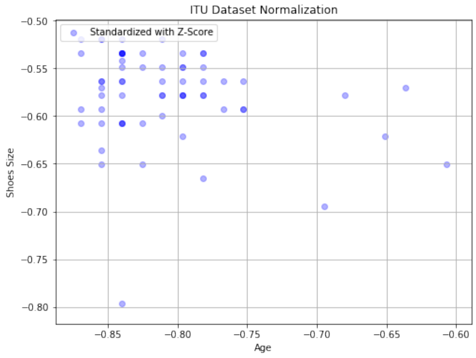

Fig. 1. Z-score Normalization

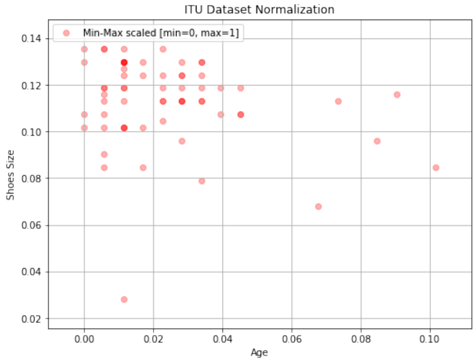

Fig. 2. Min-Max Scalling

In order to implement apriori - frequent pattern mining I had to split the data so that every programming language will be atomic, e.g. in each column is one value.

```
[[r, java],
 [java, css, html],
 [javascript, java],
 [java], ... ]
```

II. Clustering

I have chosen K-means as the clustering algorithm. My goal was to find out the clusters on discreate features *age, height and shoe size*. The algorithm first select random k objects (vectors) from the dataset as initial clusters centers. Than for each object (vector) from the dataset the Euclidean distance is computed between that object and the cluster centroids and it's assigned to the cluster which is the most similiar (shortest distance). Afterwards each cluster centers are recomputed and all the objects are then reassigned using

P. Gonzlez de Prado Salas is with the Department of Advanced Programming, IT University of Copenhagen, Denmark
S. Risi Head of Data Mining Cource, IT University of Copenhagen.
Manuscript received May 26, 2017

the updated cluster centers. And the iteration continues until the centroids stabilize. The algorithm calculate how much the centroids moved in each iteration and compare from the previous.

A. Results

The graph below shows the data points and the cluster center points marked as X in 3 dimensional space. In order to choose the appropriate K I have measured the cluster quality by the method called *within-cluster variaton*, e.g. sum of squared error. I've computed the summed squared distances for each cluster, which is the inertia and took the average of all the clusters. As the chart below shows, as I have increased K the inertia went down - the lower intertia is the data points are closer to the centre points, therefore more clusters means data points get represented better. In conclusion, the best value for K was the point where by increased K it didn't reduce the inertia much more, therefore *k=5*.

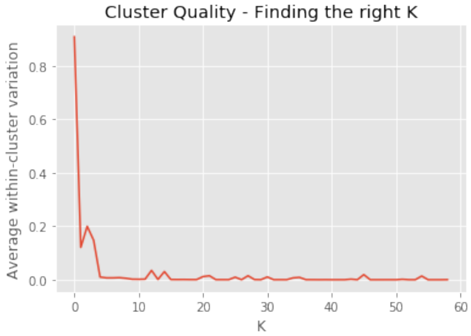

Fig. 3. Average within-cluster variation

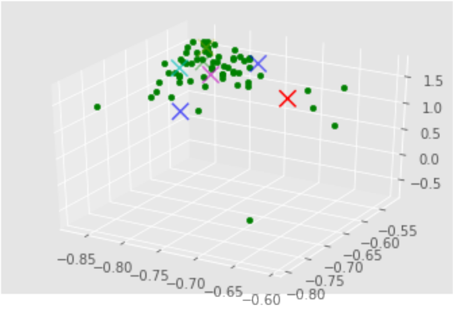

Fig. 4. Cluster Centroids

III. Classification

For the supervised learning I have choosed to implement K-Nearest Neighbours. Again I have used the same dataset as for the the clustering, therefore the question I have tried to find answer is *What is the person's gender based on their age, heigh and shoe size?*

A. Results

I have used a standart ratio of spliting the data (2/3 for training and 1/3 for testing). I have gained accuracy mostly around 80% - 90% correct predicted class for the gender. I suppose this high accuracy is the result that the dataset is unbalanced. There is total of 55 males and 9 females classifiers. The line chart below shows how the accuracy differ when I have increased *k* for the following spliting: Train: 43 and Test: 20.

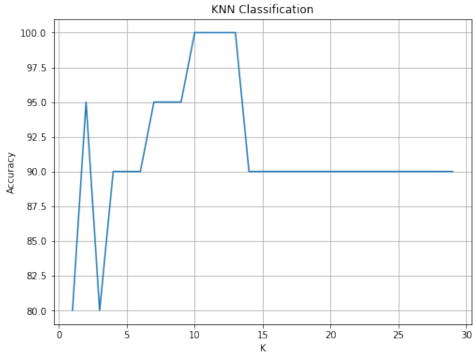

Fig. 5. Classification

Also the accuracy greatly varied accross different ratio of training and testing. The chart below shows different accuracy obtained for different split of data (train and test) and *k = 5.*

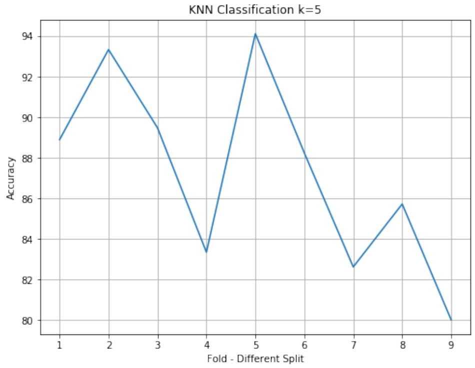

Fig. 6. KNN Classificatio with different split

IV. Apriori

Lastly, for the frequent pattern mining I have used Apriori Algorithm in order to found out which programming languages are most likely group together within ITU students. *What itemset of programming languages do itu students know?*.

A. Results

The implementation consisted of 2 parts, first I have found the all the frequent itemsets which meet the minimum support level *min_sup=0.2*.

```
[[({'c#'}),
  ({'c'}),
  ({'python'}),
  ({'javascript'}),
  ({'java'}),
  ({'f#'}),
  ({'c++'})],
 [({'c', 'java'}),
  ({'c#', 'java'}),
  ({'c++', 'java'}),
  ({'c#', 'python'}),
  ({'f#', 'java'}),
  ({'c#', 'c++'}),
  ({'java', 'javascript'}),
  ({'java', 'python'}),
  ({'c#', 'javascript'})],
 [({'c#', 'java', 'javascript'}),
  ({'c#', 'c++', 'java'}),
  ({'c#', 'java', 'python'})]
```

Then I have generated strong assocation rules from that frequent itemset which satisfied the minimum support and minimum confidence *min_conf=0.7*.

```
[(({'c'}), ({'java'}), 1.0),
 (({'c#'}), ({'java'}), 0.9655172413793103),
 (({'c++'}), ({'java'}), 0.95),
 (({'f#'}), ({'java'}), 1.0),
 (({'c++'}), ({'c#'}), 0.9000000000000001),
 (({'javascript'}), ({'java'}), 0.95),
 (({'python'}), ({'java'}), 0.9047619047619047),
 (({'c++'}), ({'c#', 'java'}), 0.8500000000000001)]

```

The rules above show up in at least 20% off all the transactions. Therefore 90% of students who knows python also knows java programming language.