DATA MINING – Practical 2

Weronika Wolska

17301623

Question 1:

Part 1: Generate a new attribute called Original Input3 which is a copy of the attribute Input3. Do the same with the attribute Input12 and copy it into Original Input12.

As shown in lines 27-31 in run.py file: create new column called "Original Input 3" and assign it the value of column "Input 3", create new column called "Original Input 12" and assign it the value of column "Input 12".

Part 2: Normalise the attribute Input3 using the z-score transformation method.

The method for calculating z-score is defined in lines 13 and 14, and follows the formula:

$$Z = (X – \mu) / \sigma$$

Where:

Z = z-score,

X = observed value,

μ = mean of the sample,

σ = standard deviation of the sample

The mean of Input3 column was calculated in line 39 and gave the value 4.4957….

The standard deviation of Input 3 column was calculated in line 41 and gave the value 1.1884….

The function was applied to all values in Input3 using a for loop and calling the function for each value i in the column (lines 46-47)

Part 3: Normalise the attribute Input12 in the range [0.0, 1.0].

The method for calculating the min/max transformation was defined in lines 17 and 18 and follows the formula:

$$Z = (x – \min(x)) / (\max(x)) – \min(x))$$

Where,

Z = the transformed value,

x = current value,

min(x) = minimum value in Input12, (=4.686, as calculated in line 58)

max(x) = max value in Input12 (=5.818, as calculated in line 56)

The function was applied to all values in Input12 using a for loop and calculating the function for each value i, as shown in lines 61 and 62.

Part 4: Generate a new attribute called Average Input, which is the average of all the attributes from Input1 to Input12. This average should include the normalised attributes values but not the copies that were made of these.

The average was calculated using the formula:

$$Avg = (x_1+x_2+....+x_a)/a$$

Where x_1, x_2,….,x_a are the value i from Input1, Input2,…., Input12, and a is the number of columns, i.e. 12. This was done for each of the lines in the file in lines 75-77 in run.py and the whole code for this part of the question can be found in lines 68-79 of run.py

Part 5: Save the newly generated dataset to ./output/question1 out.csv.

This was done in the same way as per Pracical 1, using the .to_csv(filepath, index=False) command, as can be found in line 85 of run.py

Question 2: (*incomplete due to lack of information/resources)

Part 1: Reduce the number of attributes using Principal Component Analysis (PCA), making sure at least 95% of all the variance is explained.

Firstly, we need to scale down all of the values in the file to be between 0.0 and 1.0. I have done this using StandardScaler from sklearn.decomposition, as can be seen in lines 96-100 of run.py

Next, I have imported PCA from the same library and made a PCA object, initialised to be: `pca = PCA(0.95)`, in order to ensure at least 95% of all variance is explained. Then we need to use the .fit() command to apply this to our dataset (line 107).

Part 2: Discretise the PCA-generated attribute subset into 10 bins, using bins of equal width. For each component X that you discretise, generate a new column in the original dataset named pcaX width. For example, the first discretised principal component will correspond to a new column called pca1 width.

This can be done using KBinsDiscretizer from the sklearn.preprocessing library. We initialise the discretiser with n_bins = 10, as seen in line 116. We then use the .fit() command on our dataset. The generating columns part is incomplete as I do not know how to write pca values into a csv file.

Part 3: Discretise PCA-generated attribute subset into 10 bins, using bins of equal frequency (they should all contain the same number of points). For each component X that you discretise, generate a new column in the original dataset named pcaX freq. For example, the first discretised principal component will correspond to a new column called pca1 width.

This question was not completed.

Part 4: Save the generated dataset

As per Practical 1 and question 1 part 5 above, I used the .to_csv(filepath, index=False) command, as can be found in line 132 of run.py