COMP40370 Practical 2 – Sample Solution

DATA PREPROCESSING

Prof. Tahar KECHADI

Academic year 2021-2022

Refer with the sample iPython Notebook: run.ipynb

Question 1: Advanced Data Exploration

A module coordinator has just completed the module assessments, and s/he would like to perform a quick analysis on the students results in various components of the module. The main objective is to see if there is any correlation between the assessment components. The students' results are given in the file "*Students_Results.csv*". Using Python script, answer the following questions:

1. Find the minimum, maximum, mean and standard deviation for each Homework column and the exam column.

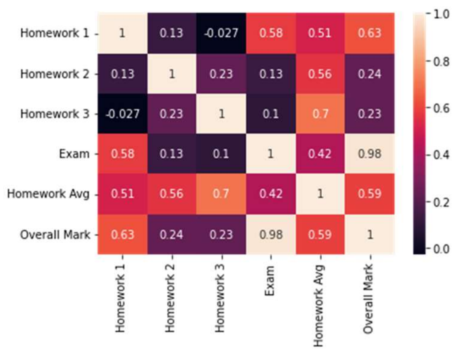
   Those number are taken by ignoring the missing values (i.e. before filling the missing values by zeros).

	Homework 1	Homework 2	Homework 3	Exam
Minimum	31	0	5	22
Maximum	90	98	100	98
Mean	55.6	89.8	47.7	65.2
Standard Deviation	17.88	15.44	21.59	15.14

2. Add an additional named as 'Homework Avg' for the average homework mark for each student. Assume that the weighting of the homework average is 25% and that of the examination is 75%, add an additional column named 'Overall Mark' for the overall folded mark.
   We assume that all absentees are given 0 marks. Then we fill the missing values with zeros and find 'Homework Avg' and 'Overall Mark' using python column operations.
      Homework Avg = (Homework 1+Homework 2+ Homework 3) / 3.0
      Overall Mark = 0.25 x Homework Avg + 0.75 x Exam

3. Construct a correlation matrix of homework and exam variables. What can you conclude from the matrix?

   You can use pandas corr() and seaborn heatmap() functions to visualise the correlation matrix with a colour gradient. It is better to remove the Student ID column from the data-frame to improve the clarity of the matrix. Generally, Correlation Coefficient (CC) above 0.4 is considered strong correlation, between 0.2 and 0.4 as moderate and below 0.2 as weak.



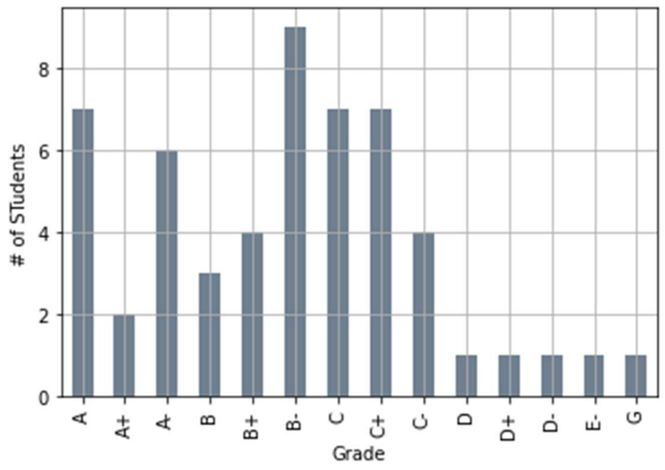
   According to the correlation matrix, Exam and Homework Avg has a CC of 0.42. It is a good observation from two independent data attributes. It shows that good or weak students have performed equally in both homeworks and exam. It also shows that markings of both modes of assessments are consistent. Poor correlations between Homework 1, 2 and 3 are not a good observation, which we expect good students perform well in all the homeworks. Since homeworks are given equal weights, three homeworks should have some consistency. There should be a moderate correlation between them. There is a 0.98 correlation between Exam and Overall Mark since we weight Exam with a 75% weight. CC between Homework Avg (which we weight only 25%) and Overall Mark of 0.59 is a satisfactory behaviour.

4. Discuss various ways of treating the missing values in the dataset.

   We should not fill missing values with zeros all the time. Missing values can be treated logically based on certain justifications from the reality. For example, in this case, if a student is absent or not submitted without a valid reason, he/she will be given 0 marks. In some reasonable cases (e.g. student is in sick, loss of a close relative), student might be given average of submitted homeworks (e. g. average of 2 will be considered for the third). Sometimes we can fill missing values based on statistical methods. e.g. find the nearest neighbour (logical but not geographical proximity in this case) based on the not-null attributes (Homework 1,2,3 and Exam) and get its nearest neighbours filled value to fill the missing value. Develop a fitted regression line using Homework 1,2,3 and Exam using the not-null data and use that line to fill the missing values. (In temporal and spatial data we can use interpolation techniques.)

5. Use UCD grading system to convert the final mark into a grade (column named 'Grade'). Produce a histogram for the grades.

UCD grading system can be found in https://www.ucd.ie/registry/t4media/Standard%20Conversion%20Grade%20Scale.pdf. You can simplify it for the program. Important thing is how you write an efficient Python program to get those grades. You can use pandas apply() function with a lambda or user-defined function. Sometimes you can assume an absentee for the exam is given 'ABS' grading by ignoring homeworks. In that case zero is different to a null. Histogram for the grades is shown below.



6. Save the newly generated dataset to "*./output/question1_out.csv*".

   Check the output file.

Question 2: Data Transformation

The *file "Sensor_Data.csv*" contains data obtained from a sensory system. Some of the attributes in the file need to be normalised, but you don't want to lose the original values.

1. Generate a new attribute called "*Original Input3*" which is a copy of the attribute "*Input3"*.

   Refer the run.ipynb program and the question2_out.csv file.

2. Do the same with the attribute "*Input12"* and copy it into Original "*Input12"*.

   Refer the run.ipynb program and the question2_out.csv file.

3. Normalise the attribute "*Input3*" using the z-score transformation method.

   To normalise the values using z-score you can use sklearn StandardScaler() function or do it from the scratch using the z-score definition. Check whether the mean of the new column is 0 and standard deviation is 1.

4. Normalise the attribute "*Input12*" in the range 0.0 and 1.0.

   To normalise the values between 0 and 1 use sklearn MinMaxScaler() function or do it from the scratch. Check min is 0 and max is 1 after the transformation.

5. 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.

   Refer the run.ipynb program. Since this has a large number of columns using a similar pattern for header names, you are encouraged to use a for-loop to create a column header label and access it from the data-frame rather than writing a line for each column operation.

6. Save the newly generated dataset to "*./output/question2_out.csv*".

   Check the output file.

Question 3: Data Reduction and Discretisation

The files "*DNA_Data.csv*" contains biological data arranged into multiple columns. We need to compress the information contained in the data.

1. Reduce the number of attributes using Principal Component Analysis (PCA), making sure at least 95% of all the variance is explained.
   Use sklearn PCA() function with PCA(0.95) format to get 22 PCs explaining 95% variance. Instead, you can use PCA without a specified variance. Then use explained_variance_ratio_ returned array to find the number of PCs with cumulative sum of variance less than or equal to 95%. You can't specify the number of PCs which returns 95% explained variance directly.

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*".
   Refer the program. To generate the new discretised columns, use a for-loop to generate the labels instead of writing 22 lines of codes. cut() function can be used to bin with equal lengths. You can use a label for each bin e.g. 'ELB1', 'ELB2', …..'ELB10'.

3. Discretise PCA-generated attribute subset into 10 bins, using bins of equal frequency (they should all 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_freq*".
   Refer the program. To generate again the new discretised columns, use a for-loop to generate the label instead of writing 22 lines of codes. qcut() function can be used to bin with equal number of data points (quantiles). You can use a label for each bin e.g. 'EFB1', 'EFB2', …..'EFB10'.

4. Save the generated dataset to "*./output/question3_out.csv*".

   Check the output file.