Data Mining Assignment Report

Ans 1:

Steps taken:

- Our solution was to find the frequent itemsets and manipulate them to get the final answers. These were supposed to be the 3-tuple (itemID, hour, student-group).
- In the dataset, Student-ID is given (e.g. F3871). Since, the last three digits of the ID are useless to us, we made a separate column with only the first letter and the year of study. Similarly, only the hour is useful and not the complete date so we extracted the hour. This was done using “modified_creator.py”
- Also note that we have to take care of the quantity of materials bought in each transaction. Hence, using “quantity_modified_creator.py”, every transactional row has been repeated quantity number of times. Note that the quantity field in the new files is identical to the original one for it is now unnecessary.
- An interesting(and obvious) observation is that the sales in December resemble more with the sales in November and October as compared to the sales in September and August. For this reason, we examined when the change in the weather is prominent for this to be true. From the weather details of the last 3 years (taken from www.accuweather.com), we concluded that the major change in the weather is around 18th of october on an average. Ergo, the entries after 18th of october (on an average) should resemble more with the December entries. To accommodate this, we created two new (and final) file called “octPLUS18_nov_combined_quant_modified.csv” and “MINUS18quant_modified_octSales.csv”. This was done mostly by simple copy-pasting.

- The “octPLUS18_nov_combined_quant_modified.csv” was then input to the var node in IBM-SPSS to extract frequent itemsets.
- Firstly, it passes through the type node. This is because our itemID and hour are currently recognized as continuous and need to be converted to nominal so that their transactional format can be made.
- This then passes through the SetToFlag node which converts the three nominal attributes into flag format.
- All the appart from the flags of itemID, student-group and hour are filtered out and the result is passed to the apriori node. Here the constraints are that the consequent can only be from the set of itemID flags and the antecedent(s) can only be from the remaining flags. The support threshold used is 0.5% and confidence threshold used is 2.2% (Reason explained later)
- The rules from the data generated are copied to “zero_support_point1confidence.csv” (name w.r.t. To the very initial configuration). Now, “modified_zero_support_point1confidence_creator.py” is run which provides us with a dictionary inside “itemid_hour_group_dict.txt”. This dictionary entries of the form {itemID: [(hour, student-group) ...] .... }. Note that to keep the penalty to a minimum, only those rules have been taken which, in the antecedent, have both hour and student-group terms (otherwise would have to increase the prices for, say, a particular hour for all student-groups).
- As it is tough to analyze the dictionary in completely raw format, “with_newline_itemid_hr_dict.txt” is made using “new_item_id_dict_creator.py”.
- Since, we didn’t take into account the dataset before October 18, we repeated the same steps for a combined data file of all transactions before October 18. However, no new rules were found which had a good enough support and confidence and all rules were already covered. Hence, this was ignored.
- A python script called “profit_dec_creator_and_calculator.py” is used to calculate the profit obtained provided the current rules. Here, the dictionary obtained in the

previous step is copied in place of ‘maindic’ variable. The prices of all the items are increased to min(price*0.5, 10).

- The other dictionaries like prices_dic and segment_wt have been calculated using “price_dict_creator.py” (by taking into account the “newPrices.csv” file given) and manually respectively
- This script calculates the revenue change in December and the penalty using the current itemsets. Note that the support and confidence threshold were 0.5 and 2.2. This is because on these values, the change in revenue comes out to be 5.25%.
- However, these rules are very crude as they don’t take into account the penalty associated (since, they are created before profit calculation). Hence, some rules have to be manually pruned to decrease the penalty while keeping the revenue change percentage above the mentioned threshold.
- The rule pruning was done by observing that hours have a square dependence on the penalty. Hence, for each item, the total contribution in the revenue and the distribution of it w.r.t. To each hour was calculated using “revenue_contribution_calculator.py”.
- Hence, to minimize the penalty while not reducing the profit by much, the items which had the least contribution in revenue, had greater number of hours were preferred out of which the hour which had the least contribution in revenue was deleted. This process was repeated until satisfactory results were obtained.
- To evaluate the problem’s model, the ratio of the revenue change encountered v/s the penalty observed can be used. The accuracy can consequently be defined as this ratio. The higher the ratio is the better is the performance of our model and ergo, the better the accuracy.