Exam Solution

Question1: (4 Marks)

We can pre-process this two data sets using Python as following steps:

Step1: loading data:

Firstly, we need to load our data from provided data resources.

Both dataset1 and dataset2 can be regarded as two csv files, specially when we load Dataset1, we can use method pandas.read_csv (“csv_file.csv”, na_values = [‘n/a’, ‘-’]) to convert non-standard missing data to NaN.

Step2: dealing with unnecessary data:

For dataset1:

It can be noticed that there is one row that is meaningless, we need to drop this row with all NaN by using method dataset1.dropna(), and we can set attribute axis=0 which means we remove rows (instead of columns) .

For dataset2:

We need to replace value ‘-’ in dataset2 with ‘morty’, since ‘morty’ occurs at most times in the datasets.

Step3: merging datasets:

In order to have complete dataset, we can merge dataset1 and dataset2 by using common attribute Device_ID, and the method we choose to use is pandas.merge().

Step4: formatting data:

After processing data using the first three steps, we need to format our data correctly.

What we aim to do is to use those data to train some model, so we have to convert our data to the useful type. Encoding Device_ID, location and operator to distinct numeric integer for each instance, which means each device, location and operator have an unique identifier to indicate specific instance.

(e.g. {‘B1834’ : 1, B9872 : 2, N2543 : 3, ……})

And also, we need to convert ‘Quality Tested Date/Time’ column data type into correct data type by applying a function which we define previously to change data into date() type and time() type.

Question2: (4 Marks)

A)

I tend to use Hierarchal Clustering approach (agglomerative clustering algorithm) to solve this problem.

For the choosing criteria about linkage:

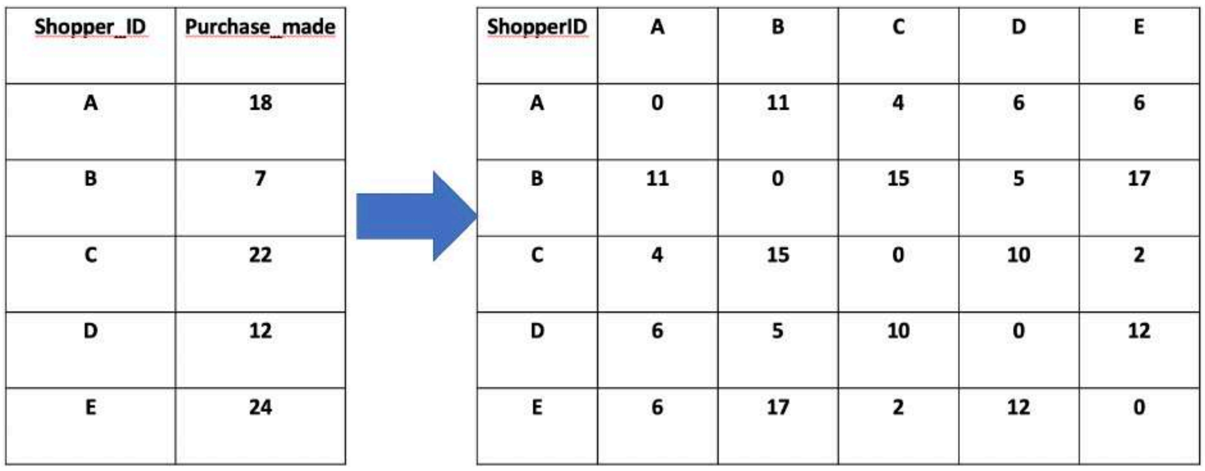
I choose single-linkage as the criteria of measuring distance between two clusters.

The reason I use Clustering is that in machine learning, unsupervised learning good be used in discovering groups or outliers and detecting patterns.

B)

Calculation steps:

Step1: compute the proximity matrix;



Shopper_ID	Purchase_made
A	18
B	7
C	22
D	12
E	24

ShopperID	A	B	C	D	E
A	0	11	4	6	6
B	11	0	15	5	17
C	4	15	0	10	2
D	6	5	10	0	12
E	6	17	2	12	0

Step2: let each data observation be a cluster;

Step3: repeat: merge the two closest clusters and update the proximity matrix;

Step4: until only a single cluster remains;

Iteration process:

Iteration 1: Group: 1: A;   2: B;   3: C;   4: D;   5: E;

Iteration 2: Group: 1: A;   2: B;   3: D;   4: (C, E);

Iteration 3: Group: 1: (A, C, E);   2: B;   3: D;

Iteration 4: Group: 1: (A, C, E);   2: (B, D);

Iteration 5: Group: 1: (A, B, C, D, E).

C)

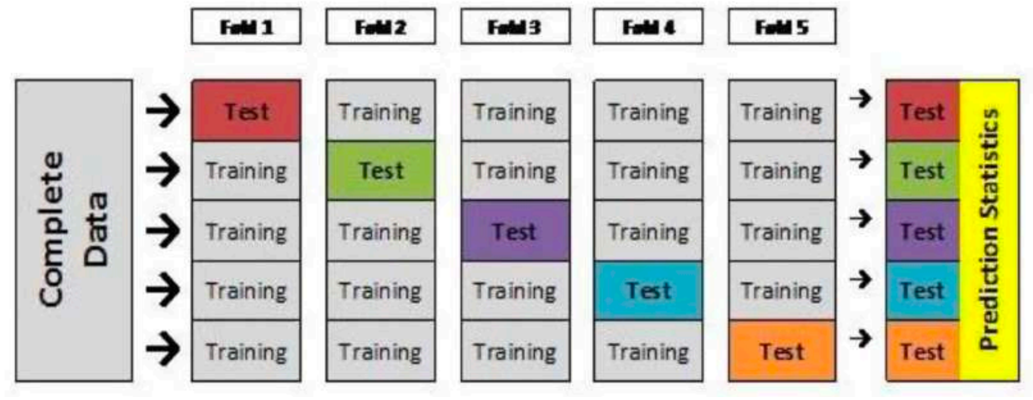
To decide the number of groups eventually, I use Dendrogram which is a tree-like diagram that records sequences of merges, whenever two classes are merged, we draw joining them together in the dendrogram and the height of the join will be the single-linkaged distance between these points.

Finally, we can draw a horizontal line to set the threshold and the number of vertical lines that are being intersected is the number of clusters.

Eventually, we determine two groups (one group (purchase made) >= 15, (purchase made) < 15 for another).

Question3: (2 Marks)

Example figure:



We use 10-Folds Cross-Validation: so N1 = 10, which means we need compute 10 times error.

We have 100 training samples, so in order to compute each error, we need to build a model with data of size 90, and test the model on the data of size 10.

We use 1/10 of original data size as our test data size.

N1 = 10; N2 = 90; N3 = 10;

Question4: (2 Marks)

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

F1-score= 2*Precision*Recall/(Precision+Recall)

FP: the classifier predicts it negative, but it is actually positive!

FN: the classifier predicts it positive, but it is actually negative!

Precision = 8 / (8 + 2) = 0.8

Recall = 8 / (8 + 12) = 0.4

F1 = 2*0.8*0.4 / (0.8 + 0.4) = 0.53

Question5: (3 Marks)

A)

Token-based method is a good choice, users enter their login credentials, Server verifies the credentials are correct and returns a token. This token is stored client-side (local storage). Subsequent requests to the server include this token.

With this method, the password is not sent around and server authenticates users by decoding token rather than using password, which can make sure to minimize the attack window if the credentials are leaked.

B)

If this organization wants to track the usage of their API and do some rate limiting, API seems to be the best choice.

API keys are useful in this regard:

Require API for every request to the protected endpoint.

We can return 429 (too many requests HTTP response code) when requests are coming too quickly in order to do some rate limiting.

Question6: (2 Marks)

If HTTP respond successfully:

Respond content below:

201 Created

Location: /orders?1

Content-Type: application/xml

<drink>latte</drink>

<something is about payment link information>

</order>

Else if HTTP respond process get something wrong:

4xx Bad request (e.g, 400 missing required field)

5xx Processing error (e.g, 503 missing required field)

<error messages>

Question7: (4 Marks)

A)

It is a classification problem, we have not enough knowledge about which attribute is more important. Compare with other models, K-Nearest Neighour might be the best choice for us.

KNN classifier is a non-parametric and instance-based learning algorithm. By looking at our original data, feature dimension is simple (avoiding high dimensional data) and our data has no mis-labeled data since it is a statistics about affected virus.

Based on those traits, KNN seems to be a good choice for this problem, because KNN is sensitive to mis-labeled data and suffers from "curse of dimensionality".

B)

Feature extraction:

For table_1, GPS location information can be splited into different directional data, we can change GPS location to specific location data in term of different direction: [N, S, E, W], which more 4 columns need to be added. The similar transformation can be utilized in column "timestamp";

For table_2, we do no need too much information about "Name" and "Contact number", because these features seem nothing to do with our prediction. We can drop them and do a regularization if necessary.

Optional transform: we could make a regularization if necessary.

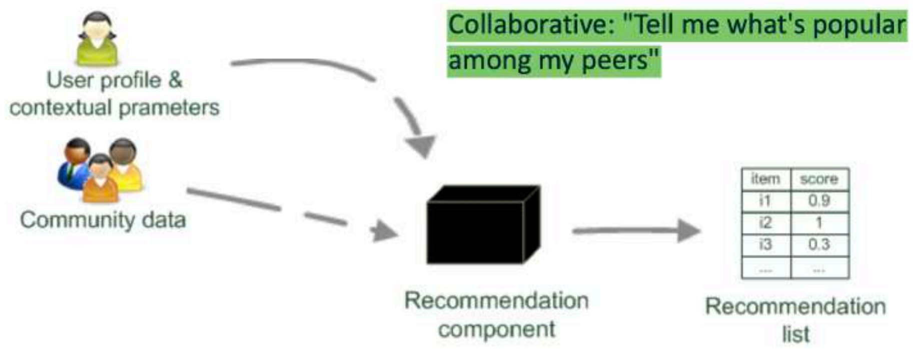
Question8: (4 Marks)

A)

R could be a hybrid system that combines User-based collaborative filtering Content-based recommender system.

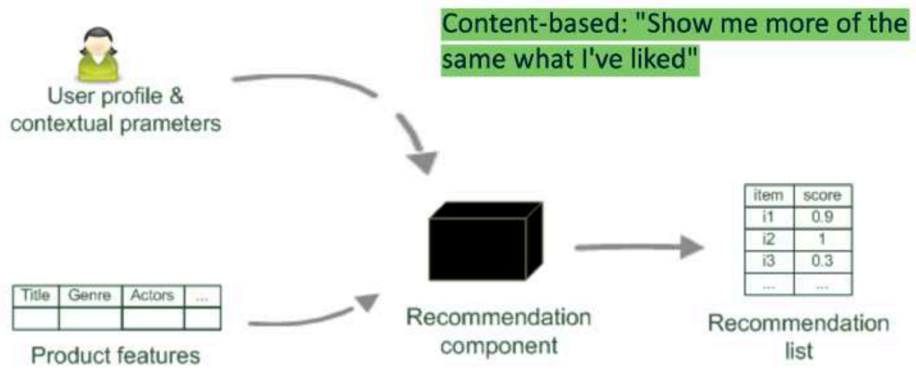
Firstly, User-based collaborative filtering is about using the “wisdom of the crowd” to recommend items. As we can see in the data records, there are more users prefer “Comedy” and “Action” than those who rating other types of moives, which implicitly gives us an information that “Comedy” movies are more likely to gain popularity among peers of user U_1 and these information are supposed to be regard as community data for user 1. Athough the table does not show too much information about users, the total number of rating reveals that some similar tastes among user U_1’s peers.

Here below is the fundamental concept about Collaborative:



Apart from User-based collaborative filtering, Content-based recommendation is being used as well. The information we get is some data about avaliable movies such as genre, release decade and director. By the way, some sort of user profile information describing what the user likes is also avaliable. And our goal is to use those data to measure which movie is the most suitable to user U_1 based on the keyitems overlap with some weights (note that: attribute “genre” seems to be more important when it comes to prediction, so “genre” should account for a greater weight).

Here below is the fundamental concept about Content-based recommendation:


Content-based: "Show me more of the same what I've liked"
User profile & contextual prameters
Title | Genre | Actors | ...
Product features
Recommendation component
item | score
i1 | 0.9
i2 | 1
i3 | 0.3
Recommendation list

B)

If we tend to generate a suitable recommended movie for user_2, it is hard for our recommender system R as our current recommendation does not contain enough information about user_2. No matter content-based recommendations or collaborative plans we try to apply, they both need user profile or contextual parameters, which is actually what we lack currently.

If we want to do recommendation for user_2, our system need additional information about user_2 like which movies the user_2 has rated, or which types of movies (or other information about movies) user_2 prefer?

When we get above mentioned information, we can make an good recommendation for user_2.