HW2

Shujuan Chen,
Lei Chen,
Ziwei Yang,
Yibo Yan

February 2022

1 Introduction

In this assignment, our task is to do collaborative filtering on the InstEval, house voting and movie lens datasets. There is no 'NA' values in any of the dataset.

2 Model by Dataset

2.1 InstEval

Dataset Description: InstEval is University Lecture/Instructor Evaluations by Students at ETH Zurich. Detailed description of the dataset can be found here.

2.1.1 sy, dy

The initial try to this model uses two generated columns sy and dy, which are the average rating a student gave and the average rating a professor received. The base accuracy is 1.135, and the test accuracy is 0.952 (This base accuracy and test accuracy and all test accuracy below are generated from 100 different holdout sets chosen in random.)

2.1.2 sy, dy, s_lectage

The second try to this model uses side information from the column lectage, and based on that, generates the new variable s_lectage, which contains the average rating a student gives to a lecture with number of semesters back this lecture rated had taken place. The test accuracy for this model is 0.896.

2.1.3 sy, dy, s_lectage, d_studage

The third try to this model add addition side information from the column studage, and based on that, generates the new variable d_studage, which contains the average rating a lecturer received from students that have number of semesters been enrolled. The test accuracy for this model is 0.873.

2.1.4 s_lectage, d_studage

Based on how we generates s_lectage, d_studage, we believe these two columns should contain the information in sy and dy. Thus the forth try removes the sy and dy. The test accuracy turns out to be 0.874, which confirms our thoughts.

2.1.5 s_lectage, d_studage, s_dept

This fifth try we include side information from department this lecture belongs to, and thus generates a new column named s_dept, which contains the average rating a student gives to lectures belong to a specific department. The test accuracy is 0.802.

2.1.6 s_lectage, d_studage, s_dept, service

The only left side information that hasn't been used is service, so we include this column in this sixth try. The test accuracy is 0.802, which shows that this column has no effect over the rating.

2.1.7 s_lectage, d_studage, s_dept, s_dept_service'

For this seventh try, we still try to use the side information about service. Thus, we generates a new column named s_dept_service which contains the average rating a student gives to lectures belong to a specific department and whether the lecturer is same as the department. The test accuracy for this is 0.800, which slightly improve our model.

2.2 House vote

2.2.1 Dataset Description

The house vote dataset includes votes for each of the U.S. House of Representatives Congressmen on the 16 bills, which are 'party', 'handicapped-infants' , 'water-project-cost-sharing', 'adoption-of-the-budget-resolution', 'physician-fee-freeze', 'el-salvador-aid', 'religious-groups-in-schools', 'anti-satellite-test-ban', 'aid-to-nicaraguan-contras', 'mx-missile', 'immigration', 'synfuels-corporation-cutback', 'education-spending', 'superfund-right-to-sue', 'crime', 'duty-free-exports', and 'export-administration-act-south-africa'. Each row in the dataset contains the house representative's party('Democrat' or Republican') and his/her voting for each of the bills. There are two possible value for voting: 'y' and 'n.' '?' values in the dataset are treated as unknown value and were excluded from consideration.

2.2.2 Data Preprocessing

Per the prompt requested, we converted each row of the data in the original dataset to user ID, item ID, rating, side information format by splitting the one row in original dataframe into 16 rows, one row for each bill.
user ID = voter_id
item ID = bill_id
rating = bill_rating

2.2.3 A Detour

At the beginning, we treated the '?' as a third category besides 'y' and 'n'. After careful review of the prompt, we realized that '?' is unknown value and should not be in our consideration. However, it might be worthwhile to describe what we have done in classifying the 'y', 'n' and '?' categories.

The base accuracy for this dataset is around 0.5.

1. party, bill_rating and G1-16, logistic model: hvLog0 At first we try to use the unprocessed Bill voting data and voter party information. There is a slight improvement, testAcc reduced to 0.454023.

We then separate the dataset into two sub-dataset, one for each party, and build models on them.

2.a Democrats sub-data, bill_rating and G1-16: dep_hvLog0 testAcc: 0.4754098

3.b Republicans sub-data, bill_rating and G1-16: rep_hvLog0 testAcc: 0.5074627

Compare to model 1, Model 2.a & 3.a is performing not well. This is probably due to decreased sample size, and the party information is already included in the 'party' column for model 1.

We then do linear model on the same data. Model 2.b achieved 0.4871194 accuracy and model 3.b achieved a much better accuracy score, 0.4104478.

4. Linear model on voter_id bill_id bill_rating has a worse accuracy of 0.5431034. This is probably due to the addition of voter_id, which generated voter of new dummy variable. Adding party to the above model improved the accuracy to 0.5071839. However, this is not much different from baseAccu. Over-fitting is prevalent with the addition of voter_id in model.

5. Lastly, we try a linear model with bill_id bill_rating and party. This model is performing good with improved accuracy of 0.4683908.

2.2.4 voter_id, bill_id

For this try, we use voter_id and bill_id to predict bill rating. The qeLogit function automatically converts these two columns to dummy variables. However, since there are two many different voters and bills, there are two many dummy variables generated. Thus, it was extremely slow for qeLogit function to get the result. So, we don't have the test accuracy for this model.

2.2.5 bill_id, voter_yespercent

From the last try, we figure out in order to use the information in voter id and bill id, we need to extract information out and store them in less columns. Therefore, for this try, we include a new column named voter_yespercent, which contains the percent of yes voting a voter votes. The base accuracy is 0.475, and the test accuracy is 0.400 (This base accuracy and test accuracy and all test accuracy below are generated from 100 different holdout sets chosen in random.)

2.2.6 bill_id, voter_yespercent, bill_yespercent

After extracting information from voter id, for this try, we want to extract information from bill_id. Thus, we generate a new column named bill_yespercent, which contains the percent of yes votes a bill receives. The test accuracy for this method is 0.402, which is pretty much the same as the last method.

2.2.7 bill_id, voter_yespercent, party

After discussion, we believe the reason that the test accuracy doesn't improve with the newly added column bill_yespercent is bill_ already contains those information. Therefore, for this try, we remove the bill_yespercent, and adds the column party. The test accuracy is 0.412. Therefore, we decide to use only columns bill_id and voter_yespercent to predict the bill rating.

2.3 Movie Lens

2.3.1 user, item

The initial try to this model uses two columns user and item to predict movie rating. This method simply creates 2000+ dummy variables from different users and movies. The test accuracy for this method is 0.909, which means our predicted ratings are offed by 0.909 from the real ratings in average. (This test accuracy and all test accuracy below are generated from 100 different holdout sets chosen in random.)

2.3.2 userMean, itemMeam

The second try to this model uses two columns userMean and itemMean to predict movie rating. This method makes sense since a user who rates high in average tends to rate high on a new movie. Also, a movie has high rate tends to receive another high rate from a new user. The test accuracy for this method is 0.737.

2.3.3 userMean, itemMean, gendergenreMean

After first two tries, we decide to generate new variables based on side information, which are gender, age, occupation, and genre. The third try creates a new column named gendergenreMean to the data frame. The gendergenreMean contains the average rating males and females give to a specific genre of movie. After including this column as the third parameter, the test accuracy is 0.739.

2.3.4 userMean, itemMean, userMean_preGenre

From the last try, we can tell the test accuracy doesn't change after including the new column gendergenreMean. We believe the reason for that is the information contained in gendergenreMean was contained in userMean and itemMean already. Therefore, in order to lower the test accuracy, we need to include more specific information than userMean and itemMean. So for this fourth try, we add a new column named userMean_preGenre which contained the average rating a user gives to a specific genre of movie. The test accuracy for this method is 0.688.

2.3.5 userMean, itemMean, userMean_preGenre, itemMean_preKindOfUser

From the last try, we can tell we are on the right track. Thus, for the fifth try, we include a newly generated column named itemMean_preKindOfUser which contains the average rating of this movie rated by male or female within a specific age range. For the age range, we divides it to 7-15, 16-25, 26-35, 36-45, 46-55, 56-65, 66-73. Since the youngest user has age of 7 and the oldest user has age of 73. Then it turns out that the test accuracy is 0.652.

2.3.6 userMean_preGenre, itemMean_preKindOfUser

Based on how we calculate userMean_preGenre and itemMean_preKindOfUser. We believe these two colunmns should include information in userMean and itemMean. Therefore, for this sixth try, we remove userMean and itemMean. Then the test accuracy is 0.652, which confirms our thought.

2.3.7 userMean_preGenre, itemMean_preKindOfUser, occupationMean

After using the side information about genre, age and gender, there are information about occupation seems to worth trying. Thus, for the seventh try, we generate a new column named occupationMean which contains the average rating a specific occupation of user gives to movies. The test accuracy for this method is 0.650, which improve our model only slightly.