Assignment 1 : Report

Rajeshwari Sah: PID - A59004038

Abstract

In this project, I have implemented Latent Factor model for recipe rating prediction task and XgBoost classifier for cooking prediction task. My username on Kaggle is **Rajeshwari Sah**. For task 1 of cooking prediction, my raking on private leaderboard is **34** with a score of 0.72670. For task 3: rating prediction, my ranking on private leaderboard is **27** with a score of 0.79278.

1 Task 1: Cooking prediction

1.1 Implementation choices

In the assignment, I have used a Xgboost classifier to predict the cooking activity. There are many features that were considered for training. Primarily, popularity, user's jaccard similarity, user's cosine similarity, recipe's jaccard similarity, recipe's cosine similarity and average rating of the recipes. For the cold start problem I used the global average rating for recipe. I have also addressed the negative sampling for each of the entry in the dataset by randomly choosing one negative sample for each of the user item entry in the provided data sample. I have achieved an accuracy of **72.67%** on the overall dataset. Feature importance plot of different features of the models are given below:

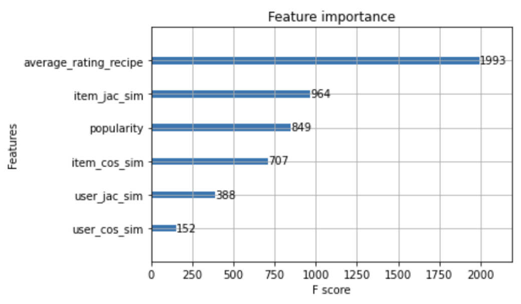

Figure 1: Feature importance Task 1

As we can see in Figure 1, features like average rating of recipes, item's jaccard similarity and popularity of the recipes remained some of the important features.

1.2 Performance on Validation set

33% of the entire dataset was considered for validation. Similar to training data, negative samples were also present in validation dataset. I achieved an accuracy of **77.17%** on validation dataset. The performance on the validation set can be observed in figure 3 and 2. Figure 2 shows the log loss graph and Figure 3 depicts the classification error plot. Figure 4 plots the confusion matrix obtained on the validation set.

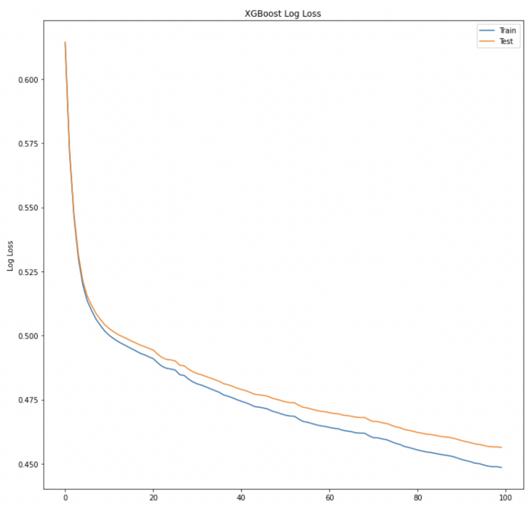

Figure 2: Log loss: Task 1

1.3 Other design choices explored

I initially started with the baseline code and experimented with the popularity based approach. In the baseline code stub the threshold for classification is set to 50% of the total cooked recipes. Based on the feature importance plot, I tried to experiment with just popularity as my feature. Using the baseline approach with a slight modification on threshold i.e

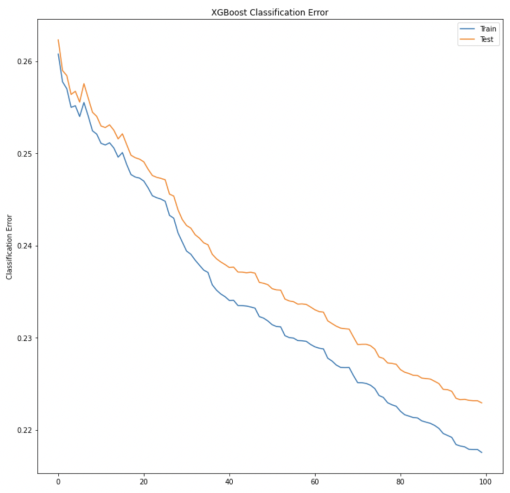

Figure 3: Classification error plot: Task 1

69% of total cooked recipes I was able to achieve an accuracy of 70.29% on the leader board.

I also experimented with other features on the Xgboost model. Features considered for this case was popular recipe, user count, recipe count, similarity, combination of popular recipe and similarities, combination of user count and similarities. This model did not perform very well, giving me an accuracy of 68% overall.

I also experimented with other algorithms like Random Forrest, Neural nets, decision trees, linear regression, FM and BPR. All of these model achieved an accuracy of less than 68% on the leaderboard which was less my best performing model.

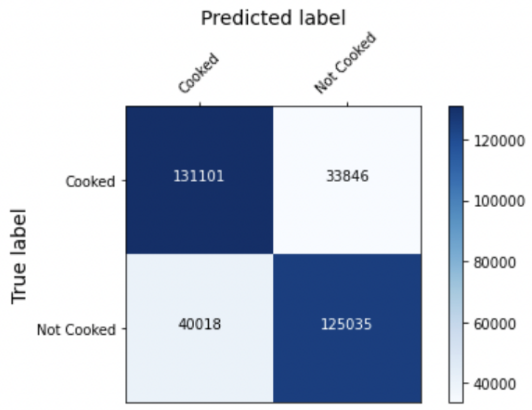

Figure 4: Confusion matrix: Task 1

1.4 Task 1 Kaggle Performance: Cooking prediction

For cooking prediction task, I am ranked 34 on the private leader-board with a score of 0.72670.

2 Task 3: Rating prediction

2.1 Implementation choices

For the rating prediction task, I have used complete latent factor model with low dimensional user and item terms. Latent factor models factorize user ratings of recipes into user and item feature vectors. By taking this approach, the models aim to find low-rank feature matrices to describe the data. This is accomplished by a penalized least squares loss function which is minimized using the Stochastic Gradient Descent searching technique. The dimension chosen was as low as 2. Overall I achieved a Mean Squared error of 0.79278 on the private leader board.

2.2 Performance analysis

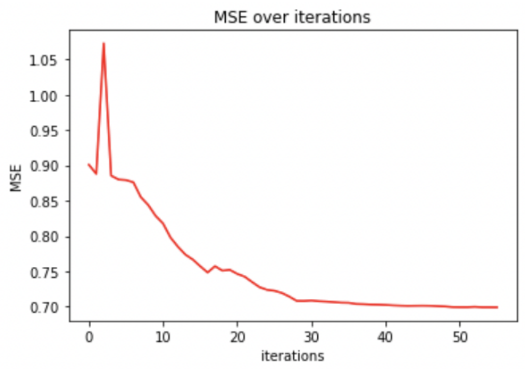

Figure 5: MSE over iterations: Task 3

Figure 5 plot shows the MSE over different iterations. As we can see in the graph, MSE for the first iterations is .90. With subsequent iterations the model is able to learn the interactions better reducing the MSE over the next few iterations. Beyond 50th iteration we can see that there is stagnation in the reduction of MSE. Using this information, I have run the optimizer for 60 iterations only with a very small lambda of 0.000022.

2.3 Other design choices explored

For rating prediction task, I explored matrix factorization methods like SVD, SVD++. Upon grid search on hyperparameters of SVD model, the best performance that I could get was an MSE of 0.82242. One point to note that, with even a small change in parameters there was more than 1% change in the MSE sometimes, which indicated the possibility of overfitting of the model. I made use of both the surprise and tensorflow libraries for this implementation.

I also experimented with the baseline code stub and latent factor models with just biases. As expected, bias only LFM did not perform well when compared to complete LFM achieving an MSE slightly above the baseline.

2.4 Task 3 Kaggle Performance: Rating prediction

For rating prediction task, I am ranked 27 on the private leader-board with a score of 0.79278.