University of Toronto

MIE1624 Data Science and Analytics

Recommender System

April 5th, 2021

Group 18

Kaggle Team ID: Group18

Bill Zhao: 1002286796

Derick Liu: 1002196394

Lingjun Zhao: 1007601285

Lv He: 1002648637

Nanxi Liu : 1002943283

Yongqi Zhen: 1003029616

Ziquan Wang: 1003043887

Introduction

The goal of this project is to build a recommender system by predicting the overall scores in Amazon Music Reviews. There are 150,000 reviews in the training data, which contains the review texts, summary texts, ID of the reviewer, ID of the product, category of the product, review time, time of the review since 1970. The target is the overall score. After feature engineering and model implementation on the training data, we applied these procedures on the test data, which contains 20,000 reviews, and tried to lower the MSE.

In this project, the TF-IDF method is used for NLP feature selection. Various algorithms are employed to build the prediction models, such as linear regression, logistic regression, neural network and Naive Bayes.

Part1 : Data Cleaning & Feature Selection

1.1 Cleaning Step

The data cleaning process follows the steps below:

We first drop the less important columns: “image”, “unixReviewTime”, “reviewHash” and drop any row or column that contains missing value (NaN) in the training data. Then we create new columns “reviewer_avg_array” and “item_avg_array”: average overall score group by “reviewerID” and “itemID”. Next, we encode the “price” column to be numeric and start to modify the “reviewText” and “summary columns”. For this procedure, we first remove special characters, numbers, punctuations, convert all characters in the text to lowercase and remove all stop words. After cleaning those two columns, we create a new column “text”, which combines the columns “reviewText” and “summary”. Lastly, we create dummy variables for the “category” column by one-hot.

The feature selection process follows the steps below:

We first prepare the data for modelling with the TF-IDF features using TfidfVectorizer( ) from sklearn.feature_extraction.text, set the max_features =2000. Then we construct the training data matrix containing features, while only the TF-IDF vectorizer feature is used in our optimal model. At last, we scale features to have a mean of 0 and a standard deviation of 1using StandardScaler from sklearn.preprocessing, which standardizes the numeric columns and then mitigates the influence of very large/small data.

1.2 Problems in Cleaning Process

The first problem is encoding the “price” column. In the beginning, we tried to split the prices into 10 buckets and every bucket has a range of 10% . However, we realized that the prices are not uniformly distributed, so directly using the numerical price value is a better option for the model implementation part.

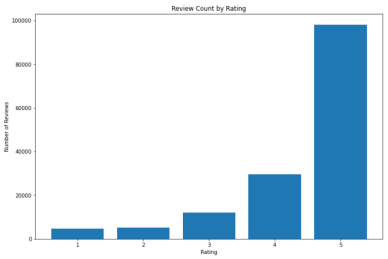
The second problem we faced was how to handle the “reviewerID” and “itemID” columns. It was impractical to encode these two columns by one-hot, since it would bring a very large matrix. To extract useful information from these two columns, we decided to compute the average overall scores group by “reviewerID” and “itemID”.

The third problem is selecting the scaling method. We tried both standardization by StandardScaler and normalization by MinMaxScaler. After model implementation, we found

standardization gave better prediction results.

1.3 Exploratory Analysis & Visualization

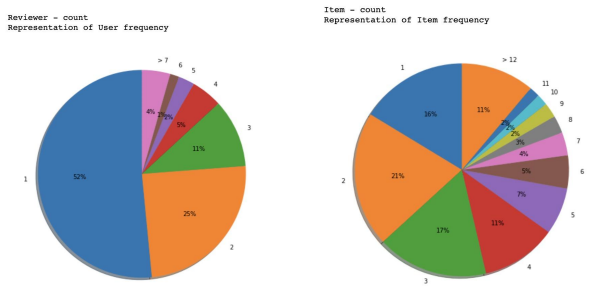
1. Distribution of overall scores:



Most of the overall scores are 5, and only a few overall scores are 1 & 2.

Therefore, the training data set is unbalanced.

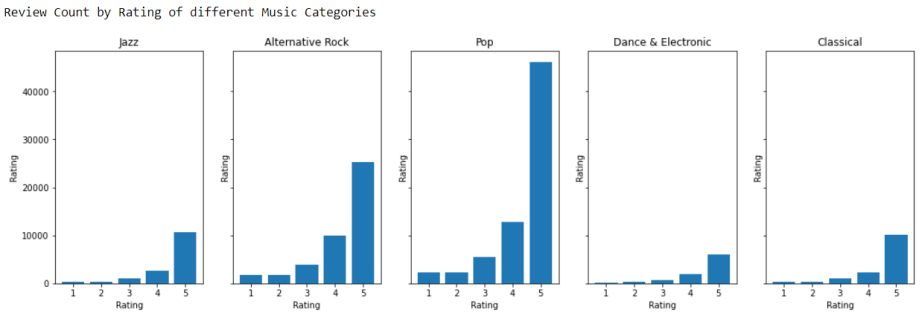
2. Proportion of repetitions in "ReviewerID" (L) and "itemID" (R):



Both "reviewerID" and "itemID" have repetitive records. More than 50% of the reviewers only reviewed once, which means "reviewerID" might be a less useful feature. "ItemID" will play a more important role compared to "reviewerID".

3. Distribution of overall scores in each category

There are obviously different numbers of reviews in each category, while all categories have the similar trends/breakdowns in overall scores. Having "category" as a feature may not impact the overall scores prediction results.



Part 2 : Implementation of Models

2.1 Optimal model

Choosing a suitable model is the first challenge we initially faced. We first applied both logistic regression model and Naive-Bayes model on the training dataset, and the training MSE is surprised to be found around 0.7. However, there is a huge disparity between the training MSE and the testing MSE that the testing set shows a result of 1.2 after we applied the optimal model on the testing dataset. Consequently, we exchanged the original models and chose a linear regression model temporarily. The MSE for the training set is relatively low, which is around 0.58. And the MSE after we applied into Kaggle is around 0.58,

meeting the strong baseline. In detail, the summary and reviewText columns are combined into one tf-idf vector. The principle behind the linear regression is to predict the continuous value of the discrete target. The variables we picked for the linear regression model are ‘TF-IDF features of the "combined" column’.

2.2 Optimization process

We performed hyperparameter tuning with cross validation for our logistic regression and Naive Bayes models. For logistic regression, the optimal C value was found to be 1. For our linear regression model, we found out that we were able to obtain a lower MSE value if we do not round the predictions.

2.3 Problems Faced

During the model implementation stage, there are several main problems that we had to deal with.

The first crucial issue was that the test MSE values of all our models when submitting to Kaggle seems to be much higher than the MSE we obtained from cross-validation. In order to bring down our test MSE value, we tried numerous countermeasures including adding and removing features as well as modifying the existing features of the dataset. Moreover, the hyperparameter tuning is also performed to obtain a lower training MSE value. Ultimately, we were able to reduce our test MSE value substantially by combining the summary and reviewText columns into one tf-idf vector and using that in our models. Moreover, using decimal numbers instead of integers on the predictions was another approach that reduced MSE according to the MSE formula. Accordingly, we applied the linear regression model with all the conditions we generated above, and the testing mse indicated an outcome of 0.58549, which met all the baselines.

In contrast, other models did not exhibit very good performance.

With respect to the neural networks model, there is one essential issue in that it took a very long time to converge. This shortcoming can be fixed by removing some of the excess intermediate layers in our neural networks model, as well as increasing the batch size from 32 to 256 in order to get faster convergence on our model. Despite the fact that the neural network model predicts on large volumes efficiently, it is still relatively difficult to find the optimal architecture as it takes much time to find the best model.

In terms of the Naive Bayes model, it unsuccessfully reduced MSE. The main reason lies is that the model is quite unstable, a small change would fluctuate the model’s whole structure. Hence there is a large disparity between the training MSE and testing MSE.

Additionally, overfitting is the vital issue we faced for the logistic regression model. Although the MSE in our training set was relatively low, around 0.6, the MSE outcome on the Kaggle was around 1.1. It should be acknowledged that overfitting was attributed to those numerous inaccurate features we chose in the data cleaning process, in other words, the model catched too many random inputs to predict efficiently. To reduce overfitting problems, we performed cross validation on the linear regression model. Due to the reason that overfitting problems incline to occur in the nonparametric and nonlinear models, the linear regression model avoided the overfitting problem perfectly.

Part 3 : Conclusion and Future Implementation

3.1 Conclusion

To sum up, the linear regression model's prediction MSE passed the strong baseline and got 0.58 in the prediction set.

Our proposed model worked by combining the summary and reviewText columns into one tf-idf vector and using decimal numbers instead of integers on the predictions, while others failed mostly because they cannot reduce MSE properly.

One crucial problem is about choosing the "max_features" in TF-IDF vectorizer. There are more than 30,000 words in the corpus after we vectorized the text columns, while a matrix with such many columns would be very computationally expensive. We tuned the hyperparameter "max_features" in the range from 250 to 10,000, and ultimately we set the max_features=2000, as too many features would result in very slow computation speed, and too few features would degrade our model's performance and increase the MSE.

Compared to the three alternatives we tested (logistic regression, multinomial Naïve-Bayes, and neural network), our optimal linear regression model had the lowest MSE value on both the training and test set.

The results of our model implementation and testing signifies that regression algorithms (such as the linear regression we used) will often tend to have lower MSE values than classification algorithms since regression algorithms are able to predict float values.

3.2 Business Implementation

In this modern world we are overloaded with data and this data provides us with useful information. But it's not possible for the user to extract the information which interests them from these data. Recommendation systems are developed to help the user to find out information about the product more effectively, and it can be applied to various products so that the customers can get better user experience and the firm can gain more profits by using such digital business form.

In this case, recommendation systems are the main part of Amazon's spotlight on the retail value of AI plans. The reason lies is that the recommendation system can personalize customers' preference , not only for their music system but also for their online shopping store. In other words, people can be introduced to their favorite types of songs and items. According to the Mckinsey & Company report, 35% of Amazon's revenue is stimulated from the recommendation engine.

What's more, many digital platforms such as Netflix and Spotify use this recommendation system to provide movies and music for their customers. It is mentionable that the user's experience increases significantly, and these platforms' revenue boosted simultaneously.

In the future, more firms can utilize the systems to target customers' favorite and to launch customers' favorite products to finish the precise marketing procedure.