Report

1. Introduction

The main method that I use at last is the SVM. I uses the one that implements in sklearn library and I do some hyperparameter tuning to make model get a better result.

2. Feature selection and encoding

2.1 Features in original dataset

I directly use the four columns: Helpfulness, Time, Summary and Text, as features in my final model. Below is the reasons for using them.

2.21 Helpfulness

Helpfulness, a good review should have a higher Helpfulness score which calculate by the ratio of HelpfulnessNumerator to HelpfulnessDenominator.

2.22 Time

Time, time is an important factor, people are more likely to give same score for the certain movie in a certain time period and people themselves may have a behavioral patterns in certain time. I also translate the timestamp format in Time column into 'Year', 'Month', and 'DayOfWeek', which I also use for training.

2.23 Summary and Text

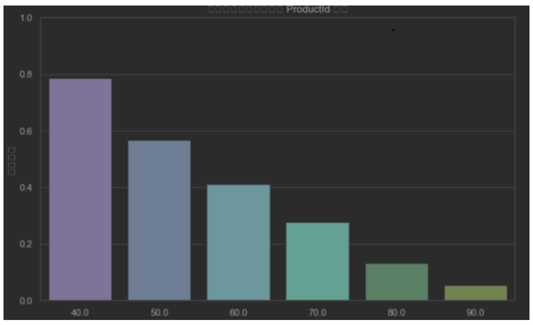
Summary and Text column are treated in the same way. I add Summary_length, Text_length, and using TF-IDF vectorization on the original data. Using Summary_length and Text_length is based on a simple idea: People are more likely to write a longer review for a movie that make them satisfied(5.0) or disappointed(1.0). Besides, people may use certain words that only in reviews of a movie that make them satisfied(5.0) or disappointed(1.0). So, using TF-IDF, which can emphasize words that are frequent in a specific document but infrequent across the entire corpus, may capture more meaningful features

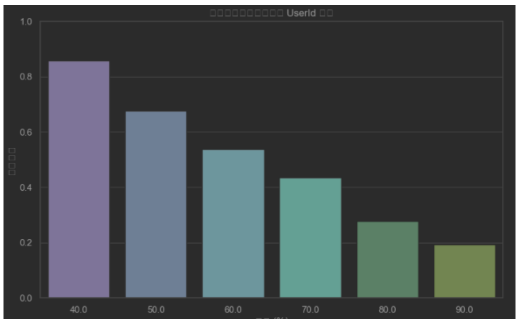
2.2 Statistical Features Based on UserId and ProductId

After I noticed there is a unbalanced distribution on the scores in the dataset, the number of review with 5.0 scores is bigger than the sum of all other scores, I think for a certain movie or person, there may be a grading tendency.

2.3.1 Proof for the idea

I cleaned the dataset and counted the number of each score type for each unique productID. Then, I calculated the proportion of the "mode" score for each ProductID and determined if a mode exists based on set thresholds (e.g., 0.4, 0.5, 0.6). I then analyzed the proportion of ProdectID with a mode under different thresholds. I did the same analysis for UserID as well, and obtained the following images.





The left one is the graph for ProdectID and the right one is for UserID, showing there are absolutely some grading tendencies for a certain movie or a person.

2.3.2 Using the tendencies

As we just allow to use method learned from the class, so I just try some simple method that gathering statistical features based on UserId and ProductId.

For UserID, I add four features:

1: User_avg_score: This feature calculates the average score for each UserId, reflecting the overall rating tendency of the user.

2: User_std_score: This feature calculates the standard deviation of scores for each UserId, measuring the dispersion of the user's ratings.

3: UserId_mode_score: This feature identifies the most frequently given score (mode) for each UserId, indicating the primary rating tendency of the user.

4: UserId_mode_proportion: This feature calculates the proportion of the mode score within all ratings of the user, representing the consistency of the user's ratings on that particular score.

The same thing is done on ProductId to get Product_avg_score, Product_std_score, Product_mode_score and Product_avg_ proportion.

By introducing these statistical features based on UserId and ProductId, the model gains a deeper understanding of the interactions between users and products, thereby enhancing the accuracy and robustness of the predictions.

3. Model Selection and Optimization

3.1 Linear Support Vector Classifier (LinearSVC)

The Linear Support Vector Classifier is a linear version of the Support Vector Machine (SVM) classification algorithm, widely used for both binary and multiclass classification tasks. LinearSVC aims to find the optimal hyperplane that maximizes the margin between different classes, thereby achieving classification.

3.2 Hyperparameter Tuning

There are four main parameter that can change in sklearn's LinearSVC: regularization parameter C, loss function loss, dual and class weight. In the experiment, I fixed loss = 'squared_hinge', dual = False, class weight = balanced, only trying to find the best C

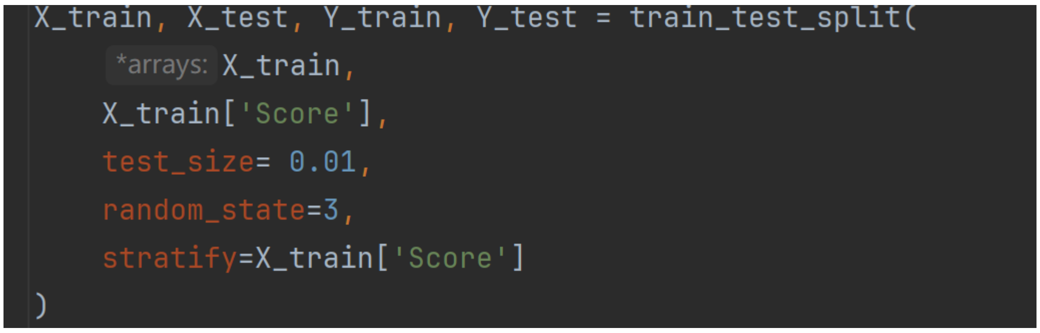
I employed GridSearchCV for hyperparameter tuning. Given the large size of the dataset, we sampled a subset from the training data to expedite the tuning process and conserve computational resources.

3.3 Model Training and Evaluation

After identifying the optimal hyperparameters through grid search, we proceeded to train the final LinearSVC model on the entire training dataset. Subsequently, we evaluated the model's performance on the test set, assessing metrics such as accuracy, classification report, and visualizing the confusion matrix to gain insights into the model's predictive capabilities.

4. Result

The best result I getting from Kaggle is using blow random seed and local train-test radio.



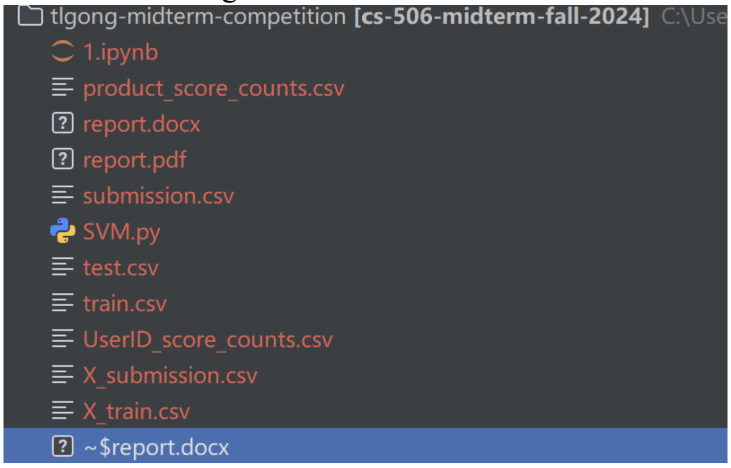
```
X_train, X_test, Y_train, Y_test = train_test_split(
    *arrays: X_train,
    X_train['Score'],
    test_size= 0.01,
    random_state=3,
    stratify=X_train['Score']
)
```

5. Appendix

Brief describe description about the files in the fold:

UserID_score_counts.csv: used in section 2.3.1 to draw graph
product_score_counts.csv: used in section 2.3.1 to draw graph
SVM.py: Main code for the experiment.
submission.csv: final result.
graph.ipynb: used to generate picture in section 2.3.1.

Because the dataset is to large I didn't put in github, when running code there should have folder arrangement like:



```
tlgong-midterm-competition [cs-506-midterm-fall-2024] C:\Use
    1.ipynb
    product_score_counts.csv
    report.docx
    report.pdf
    submission.csv
    SVM.py
    test.csv
    train.csv
    UserID_score_counts.csv
    X_submission.csv
    X_train.csv
    ~$report.docx
```

The predict answer may have a little bit difference (around 0.0005 accuracy difference on val-dataset).