Midterm Report by Haoyu Zheng

1. Main Summary and Workflow:

For this task which asks us to predict the star rating of the amazon user movie views, I tried two different approaches at first, but then decided to choose with another one. After running the source code which is posted on github, I found the most important information are both texts, the content of user reviews. So I decided to use TF-IDF to vectorize all the text files. Since by that, We can treat all text files as vectors which allows us to better fit the model. And I also encountered some problems when I tried to vectorize both the test set and the train set at the same time, which will be further discussed in the next session. After that, I found two models which may suit this task when researching online. First one is KNN, which we already tried to implement in homework. However, when I first tried to implement this, I found it's hard to decrease the running time, so I found another model which is SVM, Support Vector Machine. This also works with TF-IDF when processing text like reviews. Eventually, after trying with multiple parameters and size of datasets, I was able to get results from both models which are very close.Details will be further discussed in the following sections. Besides, there are also some tricky problems I have encountered, which will be discussed later.

2. Feature Extraction:

For this section, I will mainly talk about the details of my choice of feature extraction. There are mainly **Id, ProductId, UserId, Helpfulness, Score, Time, Summary** and **Text** features in the dataset. After a brief exploration, I found that the most important information is stored in Summary and Text columns, which are both texts. I choose to ignore the **Helpfulness** and **Time** column, because, for Time, it is hard to convert to the same type of information which we will be dealing with **Summary** and **Text**, also, the time period don't seem to affect reviewers decision compared to the other two columns. For **Summary** and **Text**, they both directly indicate the reviews impression and feelings, which can be treated as the most important data fields. At the beginning, I wasn't sure about whether I should also consider **ProductId, UserId.** After

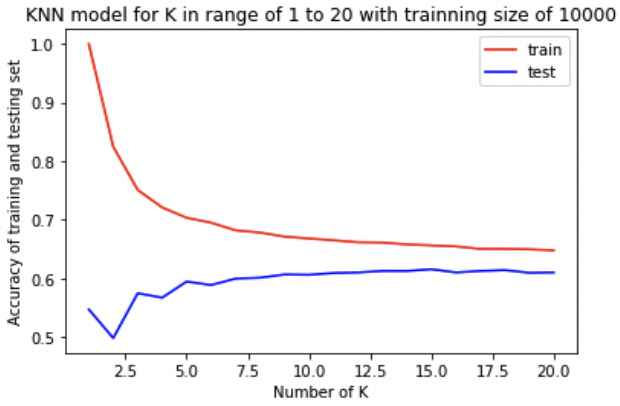
exploring, I found that many reviewers seem to have high bias, for example, they only review the things they like etc. So I decided to use TF-IDF on **ProductId, UserId, Summary** and **Text.** Before vectorization, I concatenated all four columns to a new column named "**Context**". I then Performed TFIDFvectorizer on the training and testing dataset.

3. Model Selection and Parameter Tuning:

During this midterm, I tried two different models to train my vectorized data which is described above.

A. KNN (k-Nearest Neighbors):

This is the classification model we discussed in class. It will take the vectorized data and classify them according to its nearest k neighbors. During the midterm, I first tried the full training set of approximately 1,300,000 samples. The model and TF-IDF took too long to run, so I have to decrease the size of the training set. I used the same method as we used in homework to find the proper K for the model. See below.



I found that it is safe to use K = 10 or 11 because the model accuracies tend to converge and won't cause extra computing time during the training since the K is not a large number. Finally, I tried with a training set of 500,000 samples, which provides me an accuracy close to 60%.

B. LinearSVC (Linear Support Vector Classifier) or SVM:

Support Vector Machines are a type of supervised machine learning algorithm that provides analysis of data for classification and regression analysis. And at first, I followed

https://www.kaggle.com/code/hsrobo/tf-idf-svm-baseline/script on how to properly use the model. I first tested the model based on 10,000 samples. Then I discovered that the SVC is running too slow for larger datasets. After going through the documentation and some tutorials online, I found that SVC is not good for large datasets, unlike the one I use for this midterm which is LinearSVC. See below two pictures.

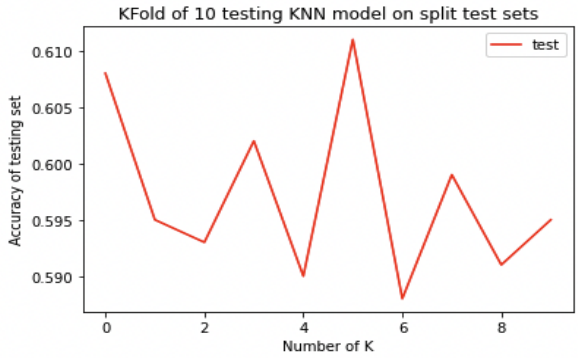
```
clf = SVC(probability=True, kernel='rbf')
```

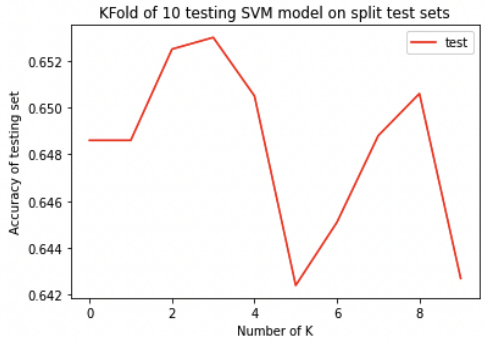
```
clf = svm.LinearSVC(loss='hinge')
```

These two methods are in the same library but LinearSVC can better handle our datasets, because it only uses linear loss functions rather than SVC which tries all, even with some non-linear ones. Then I choose to implement using LinearSVC.

4. Local Validation:

I used the same simple method we used in class, KFold, to test and evaluate the performance of my model. See below graph. KNN on the right and SVM on the left.





The result on Kaggle is very close to my local validation using KFold.