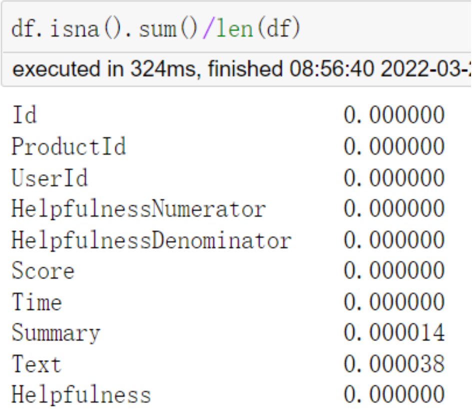
Data Analytics

Zhiyu Wang

NA Check

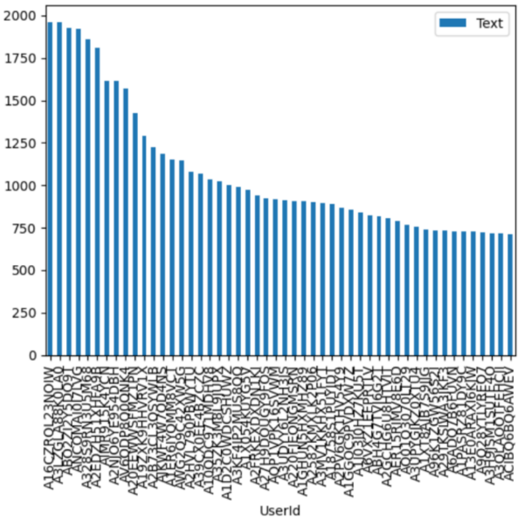
In the very first beginning, I checked the missing values in the dataset, and because it didn't have a large among of missing values in the dataset, so I just ignored the columns with missing values while training my models.

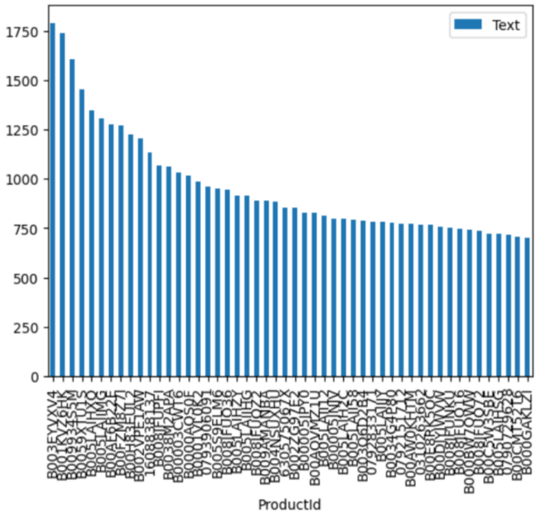


```
df.isna().sum()/len(df)
executed in 324ms, finished 08:56:40 2022-03-
Id                          0.000000
ProductId                   0.000000
UserId                      0.000000
HelpfulnessNumerator        0.000000
HelpfulnessDenominator      0.000000
Score                       0.000000
Time                        0.000000
Summary                     0.000014
Text                        0.000038
Helpfulness                 0.000000
```

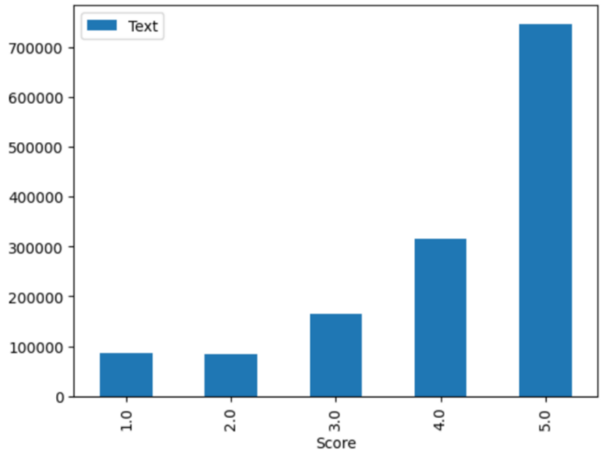
User-item Level Analysis

I checked about the number of comments from every user and number of comments from every movie and plot the distribution of top 50 on each.



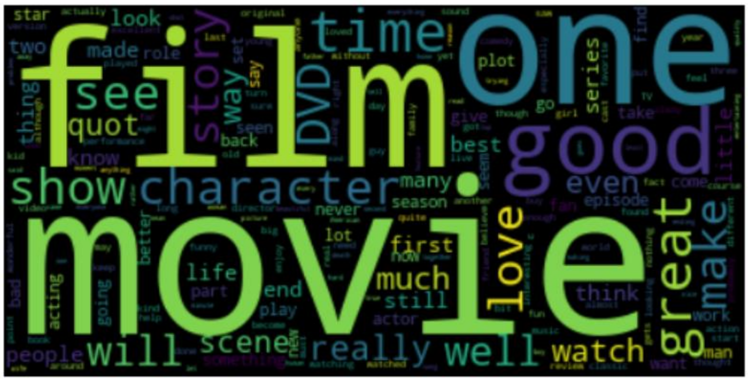


Also, I plotted the distribution of each Score and tried to figure out the relationship between the Scores and as we can see, most of the comments rated 5 and there were least than 200000 comments rated 1 or 2.



Text Level Analysis

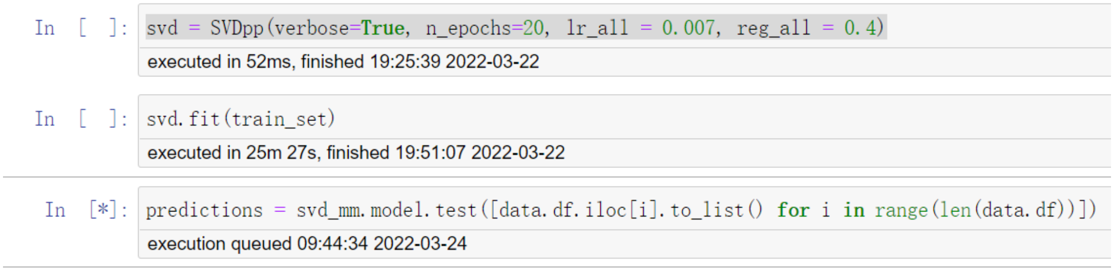
I plot the Word cloud of text of the dataset.



Feature Extraction

User-item Level Feature Extraction

In User-item level feature extraction part, instead of using collective filtering, I tried the method called Factorization Machines (https://www.jefkine.com/recsys/2017/03/27/factorization-machines/) which is a better way to model user-item relationship and generate vectors for both users and items and we used svdpp on surprise to do that.Also I used gridsearchcv to search the best parameters for the algorithm which took us a very long time to go.

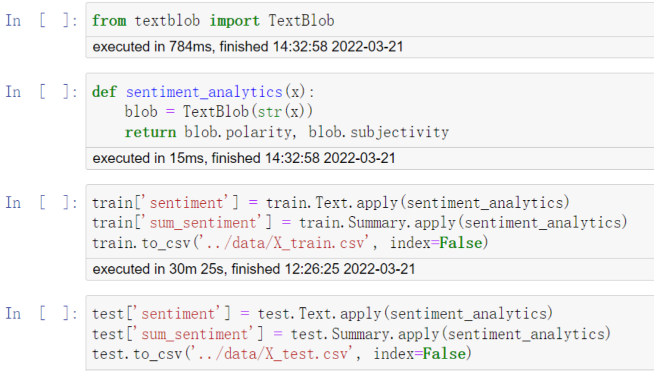


```
In [ ]: svd = SVDpp(verbose=True, n_epochs=20, lr_all = 0.007, reg_all = 0.4)
executed in 52ms, finished 19:25:39 2022-03-22

In [ ]: svd.fit(train_set)
executed in 25m 27s, finished 19:51:07 2022-03-22

In [*]: predictions = svd_mm.model.test([data.df.iloc[i].to_list() for i in range(len(data.df))])
execution queued 09:44:34 2022-03-24
```

Text Level Feature Extraction

In Text level feature extraction part, firstly I did sentiment analysis on both text and summary because I thought sentiment of a text represents the attitude of the user so I used textblob(https://towardsdatascience.com/my-absolute-go-to-for-sentiment-analysis-textblob-3ac

3a11d524) to do the analysis by applying the TextBlob function directly on the contents.

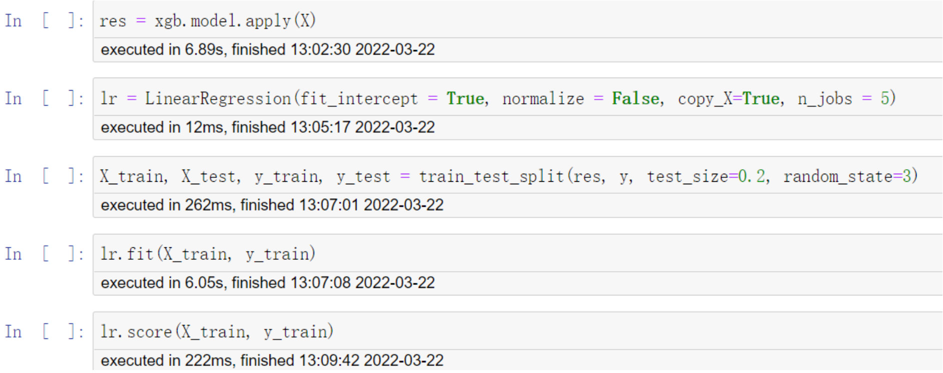


```
In [ ]: from textblob import TextBlob
executed in 784ms, finished 14:32:58 2022-03-21

In [ ]: def sentiment_analytics(x):
            blob = TextBlob(str(x))
            return blob.polarity, blob.subjectivity
executed in 15ms, finished 14:32:58 2022-03-21

In [ ]: train['sentiment'] = train.Text.apply(sentiment_analytics)
        train['sum_sentiment'] = train.Summary.apply(sentiment_analytics)
        train.to_csv('../data/X_train.csv', index=False)
executed in 30m 25s, finished 12:26:25 2022-03-21

In [ ]: test['sentiment'] = test.Text.apply(sentiment_analytics)
        test['sum_sentiment'] = test.Summary.apply(sentiment_analytics)
        test.to_csv('../data/X_test.csv', index=False)
```

What’s more, we used tf-idf on the text of each comment and generated vector for each of comment.

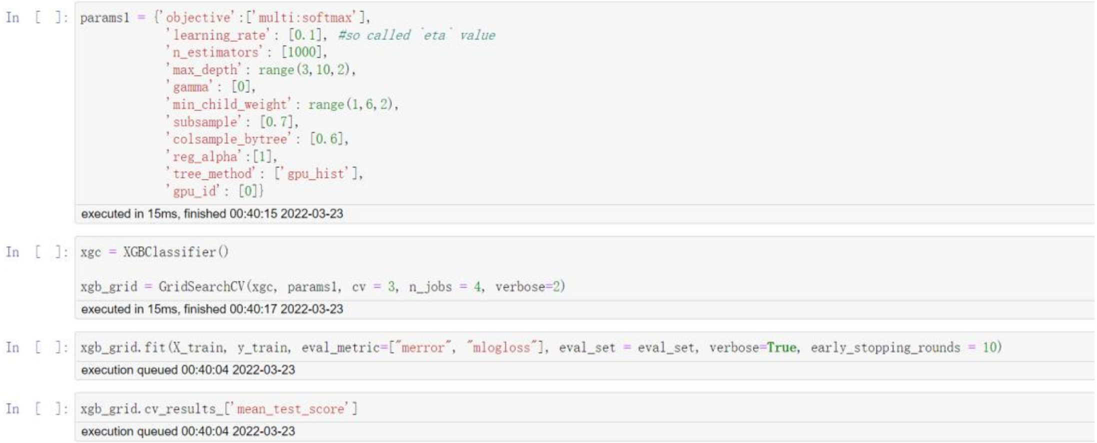
Machine Learning

After encoding our data, we used xgboost as a decoder to generate our final rate for each comment. We firstly though that it was a regression problem so we use xgboost regressor + Linea regression (inspired by xgboost+lr from Facebook predicting ctr, (https://github.com/luweikxy/machine-learning-notes/blob/master/docs/recommender-systems/industry-application/facebook/xgboost+lr/Practical-Lessons-from-Predicting-Clicks-on-Ads-at-Facebook.md)r) to generate our result but we only got a 0.46 score on our submission.



```
In [ ]: res = xgb.model.apply(X)
executed in 6.89s, finished 13:02:30 2022-03-22

In [ ]: lr = LinearRegression(fit_intercept = True, normalize = False, copy_X=True, n_jobs = 5)
executed in 12ms, finished 13:05:17 2022-03-22

In [ ]: X_train, X_test, y_train, y_test = train_test_split(res, y, test_size=0.2, random_state=3)
executed in 262ms, finished 13:07:01 2022-03-22

In [ ]: lr.fit(X_train, y_train)
executed in 6.05s, finished 13:07:08 2022-03-22

In [ ]: lr.score(X_train, y_train)
executed in 222ms, finished 13:09:42 2022-03-22
```

Then we used xgboost classifier(https://xgboost.readthedocs.io/en/stable/) on the encoded dataset with 500000 lines and also use gridsearchcv to find the best parameters for the model which gave us a 0.64101 score on Kaggle.



```
In [ ]: params1 = {'objective':['multi:softmax'],
                   'learning_rate': [0.1], #so called `eta` value
                   'n_estimators': [1000],
                   'max_depth': range(3,10,2),
                   'gamma': [0],
                   'min_child_weight': range(1,6,2),
                   'subsample': [0.7],
                   'colsample_bytree': [0.6],
                   'reg_alpha':[1],
                   'tree_method': ['gpu_hist'],
                   'gpu_id': [0]}
```

executed in 15ms, finished 00:40:15 2022-03-23

```
In [ ]: xgc = XGBClassifier()

        xgb_grid = GridSearchCV(xgc, params1, cv = 3, n_jobs = 4, verbose=2)
```

executed in 15ms, finished 00:40:17 2022-03-23

```
In [ ]: xgb_grid.fit(X_train, y_train, eval_metric=["merror", "mlogloss"], eval_set = eval_set, verbose=True, early_stopping_rounds = 10)
```

execution queued 00:40:04 2022-03-23

```
In [ ]: xgb_grid.cv_results_['mean_test_score']
```

execution queued 00:40:04 2022-03-23