**Preliminary analysis:**

After I downloaded all the files, I started to think what’s the first step of data analysis. To me, I started my work by understanding the training dataset. First I used pd.read\_csv to read the csv file and store it in data frames. Then I used several functions: df.head() which returned the first 5 rows of the data frame; df.dtypes which returned the type of each columns; df.describe() which returned the descriptive statistic information about the data frame including count, mean, standard deviation, maximum, and minimum values. An interesting thing is that, after studying the description of the df.describe() function in the pandas API reference, I found that the describe() function return the statistic information excluding nan values which is really useful and technical. From the information printed above, I found out that for the Summary and Text column, it’s Object type with a paragraph of words describing people’s perspective on the product. So, I think if I can transfer the sentences into numerical values which can be used as features in our machine learning models. As I don’t know any API’s doing that, so I searched on google and I found a tool called TextBlob which is a python library for processing textual data, diving into common natural language processing tasks. I planed to use its sentimental analysis function. As we know, the sentiment of a sentence is evaluated by the sentimental words which are positive or negative. TextBlob will read in a text message as input and divide it into individual words, and assign to each word an individual score based on the words sentiment, and the final sentimental score is calculated by adding up each individual scores with adding coefficient to each score. And the range for the final score of a text is -1.0 to 1.0 which -1.0 is negative and 1.0 is positive. Then I used pd.pivot\_table to see the relation between ProductId, UserId, and Score. As in Chinese, we call it perspectivity table which wanted to show the strength of its clearness while evaluating data. It gathered the scores with same ProductId and UserId and calculated the average score. Then I used df.sort\_values(by=[‘Score],ascending=false) which gave me the order of ProductId and UserId from high average score to low average score. And I found the common point for high score is their sentimental score are close to 1.0. So I thought if I can use the average score of the same ProductId and UserId to hep my model make predictions. But as I mentioned at the beginning, they were Object Type and some are Letters, so I think if I can convert it into numbers. With this thought, I started googling again, I found there’s an API called category\_encoders. It is a set of scikit learn style transformers for encoding categorical variables into numeric variables with different techniques. The technique I chose to use is Target Encoder which replace the features with blending posterior probability of the target given particular categorical value and the prior probability of the target over the training set.

**Feature extraction, model testing, techniques tried and decision making:**

After having a basic understanding of the data and some APIs which might be useful, I started my steps in adding features to the data. In the feature\_extraction.py, I first added two new columns called sentimenral1 and sentimental2 which stores the sentimental score calculated by Textblob for the original data column Summary and Text. While in this process I met a problem that some summaries and text are blank with is “nan” values, so I set a conditional logic criteria which only calculate the score for Summary and Text that are not blank and remain the blank Summary and Text with blank score. I chose to drop the nan values columns later when before I do training in models, not to drop the nan value columns as first I consider it’s only Summary or Text that are nan but for other columns are not, so it might be useful when I operate other columns. Another reason is that validation and submission dataset is also adding features in this file, if I drop the nan value columns, these files will contain nothing as the score column is blank which we need to predict.

Then I started to train models. I first used df.dropna(axis=0,how=’any’) to drop nan values, axis=0 which means dropping the rows and how=’any’ means dropping the rows if any of the column is nan. I split the training dataset into trainset and validation set. From sklearn I imported my models. First I tried KNN as it’s already provided in the file, I set the model to KNN and used model.fit to train the model with my training data and used model.predict to predict validation data and compared my model prediction with the actual values with the scoring rubric of RMSE, receiving a score of 1.26 which was a good starting try. Further, I tried GBDT which is an improvement of decision tree by bosting gradients. Although the RMSE for validation dataset is 1.37, I passed this model as for decision tree each branch is like a yes, no question instead of numerical features, so I consider the GBDT model does not fit our requirement. Then, I tried Kmeans as it was introduced during lecture. I used the elbow method to plot the graph of relationship between SSE and number of clusters. Based on the graph, when number of clusters=3 it’s at the elbow point which means that before this point number of clusters increase, SSE decrease dramatically and after this point, number of clusters increase, SSE decrease slowly. For my kmeans method, I set k=3 and calculated the score of RMSE for checking the validation of my model, but this time I received a score of 11.16 which means this model dose not fit this dataset. With my incomprehension, I found documents about machine learning which divided it into two parts, supervised learning and unsupervised learning. Supervised learning which requires training sets and testing set for models to fix and improve and use the features and trained model to make predictions. Most famous models are KNN, and regression. And unsupervised learning is the training data does not contain features and exact output, so we need to cluster the data due to their similarities, and Kmeans is a famous model. So, as our data contains training set and testing set and we want to use the features of data to make predictions, I found out that I should focus on supervised learning models. As mentioned above regression is a kind of supervised model, so I made my work on linear regression model. Linear regression fits a linear model with coefficients, like for our model we have many features: sentimental1, sentimental2, and linear regression will calculate a coefficient for each feature and add together to get a final score. Y=a1x1+a2x2+a3x3+……anxn+constant. Similar to KNN, I imported LinearRegression from sklearn.linear\_model and set the model to LinearRegression() and use my training data to fit the model. Then I use model.predict to predict the validation data and compare with the actual values receiving a RMSE of 1.03 which is a huge improvement reaching 1.0.

I think linear regression is a good model which fits this data set. So I started to think what other improvements I can do to reduce RMSE. I looked at my column of sentimental 1 and sentimental2 I found out that in the same row some are really different, for example, in a row sentimental1 is near 0.8 and sentimental2 is below 0, as this the summary and text on the same product it should not differ so much. So, I think if I can make a new column as the average of sentimenal1 and sentimenal2 for each row which can neutralize the big different scores. In the feature extraction, I added the column named sentimental3 to be the average of sentimental1 and sentimental2. After adding the feature sentimental3 into the model, my RMSE score on validation data headed towards 1.02 and Kaggle score headed towards 0.9998 which is my first time under 1.0. Then, I looked back at the preliminary analysis it made, I found that I haven’t use the productid and userid. As I mentioned before in the preliminary analysis part, I planed to use target encoding to encode productid and userid into numerical values. I imported category\_ecorders and set the encoder to Target encoder and fit the model with my training set and transformed the columns of userid, productid to new columns product and user which stores the average score for the same productid and userid. Then I tried to train my linear regression model with the new train data which added the product and user feature, however, the RMSE on Kaggle increase to 1.06. After some time going over again my dataset, I found that maybe the unit of the numerical values are different. For the columns of productid and userid it’s between 0.0 to 5.0 but for columns like sentimental1, sentimental2, and sentimental3 its range is between -1.0 to 1.0. So I think if I can scale these columns to a new one with fits for every column. In the feature\_extraction.py, I imported preprocessing from sklearn which provides the standard scaler function and the transformer API helping to center and scale each feature by computing the relevant statistics on the samples in the training set. After standardizing the dataframe, I trained my model again, however, to my surprise, it only reduced the score to 1.03 which is not better than the score when I did not used the product and user column. This is one thing I might investigate in the future that theoretically should improve my score. Then before the final submission I made my last try which I changed my model to logistic regression which measures the relationship between categorical dependent variable and independent variable by estimating probabilities using logistic function, which is a cumulative distribution function. I used the data with 9 columns to train the model and received a RMSE score of 0.73 for validation data, however, on Kaggle I only received 1.05 which meant that my model might be overfitting which is my model fits exactly with the training data and validation data, as it learns to much features and details which hurt the prediction of the model. As time pass and I don’t have further time to optimize other models, I choose to use linear regression as my final model. As we know, sklearn models has a lot of parameters, I tried to manually change the parameters hopping to optimize my model. As there’re a lot of parameters and testing time is huge, I started to think if there’s a function that can help me do that. I found an API called GridSearchCV in sklearn.model\_selection. This API reads in the model you want to use and the parameters you want to test and then fit with your training set and will return the best fit parameters for your model based on your training set. With the help of GridSearchCV, I found the best fit parameter for my linear regression model and it helped to reduce the score for 0.01.

**Future work and deficiency:**

There’re still some thoughts that I haven’t achieved and some problems I would like to find out in the future. As mentioned above while I was training the logistic regression model, it seems to be overfitting as I added to much feature to the training set. I would like to test if regularization, dropout features, and data augmentation can fix my overfitting problem. Also, I would like to try TFIDF on the summary and text column instead of sentimental analysis, as I think the sentimental result might be largely influenced by some strong perspective words. Also, a few predictions are out of range larger than 5.0 and smaller than 1.0, I tried to fit these values to 5.0 and 1.0 as well as mean values of the data and the former performed better. But this only occur with linear regression model but not with logistic regression model, and I wanted to figure out why in the future.