CS506 Midterm - Report

Nikhil Ramchandani

The following report aims to show how I tackled the midterm. It will discuss my first thoughts, methodology, and evaluation.

Initial Thoughts

- I recognized the large amount of data available. I thought this could be quite useful in terms of perfecting and training a model
- Took a look at the columns available and started to think of what features I could extract
- Ran starter to code to see the expected workflow - saw how the base columns performed
- Started to think about different data science techniques I can use (especially ones learned in the semester)

Feature Extraction and Analysis

Time Features

I first converted the Unix time given to a usable format and noted individual features such as the day of the week, month, year, and day. These were the features I was directly able to extract from the timestamp given. However, I didn't have too much confidence in the importance of such features and I was able to verify this by running the KNN starter code and didn't see much of a difference. I wanted to see if time could be useful, so I plotted out the reviews with date on the x-axis and rating on the y-axis. I started to notice a correlation but I felt like I needed another way to verify the importance of features. I decided to use a random forest classifier model to pick out important features. This became the main way, I judged features as I could get a set quantitative value to compare one another.

```
# Train a Random Forest model
rf_model = RandomForestClassifier(n_estimators=100,max_depth=20, random_state=0)
rf_model.fit(X_train_select, Y_train)

# Get feature importances
importances = rf_model.feature_importances_

# Create a DataFrame for visualization
feature_importance_df = pd.DataFrame({
    'Feature': features,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

# Plot feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Feature Importance from Random Forest')
plt.show()
```

With this, I was able to determine that the recency of a review was going to be a useful feature. It was the first feature I found that took over all the base features.

Text / Summary Features

I decided to explore text and summary analysis since they form a significant part of user reviews and were previously not utilized, as the models relied on numeric inputs.

I first decided to look at the structure of the text/summary such as length, character count, sentence length, and more. I saw minor improvements in the accuracy of the model but decided to pause testing and decided to extract as many features as I could. I started to pick out the noun, verb, and adjective counts as well as punctuation count. Additionally, I extracted upper and lower case counts.

Additionally, I explored alternative methods and libraries for text analysis. After conducting some research, I chose to apply the TF-IDF vectorizer to each text and summary and then calculated the cosine similarity between these vectors to measure the similarity between the two qualitative inputs. Moreover, in the past, I have worked with NLP tools for stock trading analysis and thought it would be interesting to bring some concepts over like sentiment analysis. I started with the Textblob library but was suggested to switch to Vader from AI. Apparently, Vader was better for the task I was dealing with as it was optimized for short social media posts like reviews. I decided to test this out and plotted these two sentiment analysis techniques as well as all the other features mentioned above. I did this by looking into the importance of each feature in the random forest classifier. I was able to notice that the sentiment analysis from Vader was by far the most effective feature I currently had. The rest of the text features were relatively unimportant.

At this point, I noticed how the random forest model was achieving much better accuracy than the KNN model. The KNN model was not increasing past 0.5 accuracy. Therefore I decided to stop using the KNN model.

Feature Extractions Using Average Scores

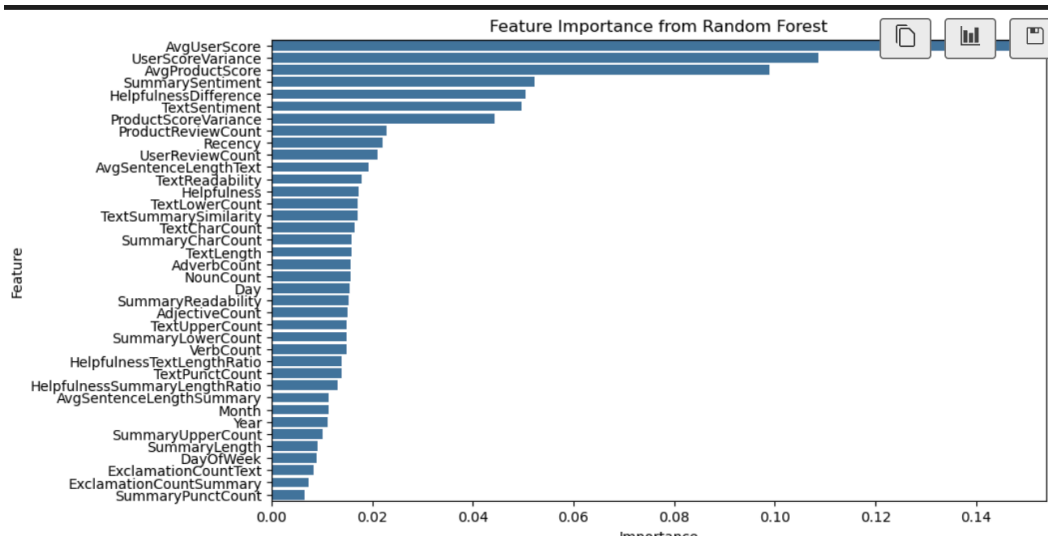
In class, the idea of using the average product score and average user score was brought up. I was quite confident that these two features being able to have a big impact on my model. However, I was afraid of leaking data. Therefore, I made sure to calculate both of these only using the first 75% of the training data (not including the rows being submitted). I then assigned such values to corresponding User and Product ID’s. If there was a new User or Product in the testing/submission set I just assigned the global score average. This is something I thought would be a good placeholder and I could back to it to test other values such as zero or other values. Once I had these features, I saw a huge improvement and was able to break through 60% accuracy in my model.

Further Feature Extraction

I then decided to try out other stuff that could be useful such as the user and product rating variance. Additionally, I decided to count the number of reviews on a specific product and the number of reviews made by a user. For these features, I used the same method to prevent data leakage. Null values were covered by the corresponding averages or zeros.

Lastly, I started to play around with helpfulness and found the best I could do was use the helpfulness difference as a feature. Below is a chart to illustrate the importance of the different

features mentioned above.



Model Fitting

I also ran TF-IDF (max_features = 1000) with the above features, however, after running this for 24 hours I still had no output. With the deadline approaching, I decided to revert back to just using the extracted features and to stick with the random forest classifier which had a quick runtime. I was getting a local accuracy of around 65%. While this was much higher than the previous score, I thought it would be a good idea to try using the top ten to fifteen most prominent features. With this, I reached a local accuracy of 67%. However, on submission, I saw a drop to 62%. I am still not sure why there was such a high discrepancy between the two accuracy values. It could be due to data leakage, however, I can not pinpoint where this takes place as I took measures to prevent this. I kept playing around with the model and tried removing and adding the various features but stuck with using the most prominent ones as it yielded the best results locally and remotely.

Evaluation

I thought this midterm was quite interesting with the competitive aspect, however, I feel like I approached it the wrong way as I faced periods where I ran models for more than 24 hours at a time. This was not the ideal case, especially in a time-sensitive project. However, I was able to recover from this by simplifying my model.

In the future, I would like to experiment with different models as I only really go to test KNN and random forest classifiers. I also learned that using all the data available is not necessary which is something that I should have practiced from the beginning. This would have given me the time to experiment with and hyper-tune models.

Moreover, I thought of the idea of sampling data with respect to an even spread across the ratings as the data was highly skewed. It would have been interesting to see how this could affect accuracy.

Lastly, I tried to play around with glove embeddings, specifically the dataset trained on tweets. While I couldn't implement this in time, I think it would be a great next step.