**Arlens Zeqollari**
**UCSD MAS DSE220**
**June 11, 2019**

# DSE220 - Final Project Report

---

***Introduction:***

*This report summarizes the preprocessing, feature generation, challenges, and results of building a model to predict the usefulness of Amazon reviews for a large number of products. The analysis was conducted using Python with several machine learning and natural language processing libraries/resources. The model was submitted on Kaggle under the username **arlens_z**.*

---

***Preprocessing:***

*The data within this report was loaded from json files containing train and test data. The following preprocessing steps were implemented to support a more accurate model.*

*The train dataset was initially filtered to only rows containing 1 or more **outOf** because I did not want information from the **outOf** rows equal to 0 to impact the model. These values would be guaranteed to have 0 helpful votes so ignoring this information did not negatively impact the ability to correctly predict the helpfulness rate of these rows. I saw considerable improvement in when performing validation dataset evaluations and when submitting for test data evaluation. Other filters were evaluated (such as <= 10 for **outOf**, or <= 150, 200, 250 for **outOf**, but these did not improve my testing accuracy. I studied the distributions of each feature in my training dataset and tried to filter certain data to imrove generalizability of my model but ultimately ended up with a simple requirement of >= 1 for **outOf**. Such a low threshold worked well and did not remove too much data/information as the majority of data points had lower than 10 reviews. Higher thresholds reduced the amount of information available to my decision tree regressor and generally resulted in worse performance.*

*Other preprocessing steps involved simply exposing the **outOf** and **nHelpful** values from the raw data, as well as creating the **itemRate** variable during loading of the dataset, although this variable did not seem to improve the performance of my model, likely due to large errors on items that had only one or two previous results (causing extreme predictions of 100% or 0% helpful, as opposed to the more conservative average global helpfulness rate, which was likely to produce fewer errors with large magnitudes. For certain variables with appropriate distributions, I conducted transformations using log, but this proved relatively unimportant with regards to model accuracy. I also filled N/A values of columns like **price** with the mean of a particular **categoryID** in order to support still using these observations in my model without severely impacting the sensitivity to find important factors for other variables. Lastly, I generated **categoryID** columns by breaking out the value of the **categoryID** into separate columns using **pd.get_dummies()**.*

*Scaling was not performed as the **AdaBoostRegressor** Decision Tree model I utilized for my final submission do not perform differently with scaling.*

---

***Feature Generation:***

*My strategy in designing features for this dataset varied from both explorative to intentional. I suspected that a few features were likely to result in a significant improvement in the ability to predict helpfulness of reviews, but for others, I generated explorative features with a lack of a coherent hypothesis as to their importance. The full list of features I generated were shown below:*

| Feature Name | Hypothesis |
|---|---|
| **total_item_reviews** | Items with more reviews may produce a different distribution of highly helpful reviews (i.e. reviews with relatively lower outOf values may have been pushed down further in rankings, perhaps onto hidden pages). |
| **perc_helpful** | N/A. This was used as the label variable and multiplied by outOf in the test dataset to produce the final predictions. |
| **first_item_review** | N/A. This was used to produce review_lateness feature. |
| **ave_item_helpful** | This was evaluated to understand whether the average review helpfulness of specific items was an improvement over the global average when a user average helpfulness was not found for a particular item, i.e. the test dataset included a user not found in the train dataset. |
| **review_len** | Longer reviews may produce more information than shorter reviews, and this information may be generally deemed helpful more often than not. |
| **word_count** | Longer reviews may produce more information than shorter reviews, and this information may be generally deemed helpful more often than not. |
| **ave_word_len** | Reviews with longer average word length may imply a better command of the English language, and therefore a better ability to convey information about a particular product. |
| **ave_user_helpfulness** | The average user helpfulness may indicate that a new review by the same user is likely to be written in a helpful manner as well. |
| **user_experience** | The user's experience in writing reviews may suggest that the user has been encouraged to write additional reviews from positively received past reviews. In addition, this user may have an idea of how well-received reviews are written and what information they might provide. |
| **colon_density** | The colon density was observed during browsing of reviews in the train dataset as several of the highly rated reviews showed sections clearly defined, such as "Fit: …" or "The Good: …" These make it easy for shoppers to quickly get to the aspects of the review they are most interested in. |
| **numerical_density** | The amount of information regarding size, dimension, weight, etc. may play a role in the information density of a review, and therefore helpfulness. |
| **exclamation_density** | Exclamations generally imply emotional writing, and serve as a crude measure of subjectivity. Reviews like these may potentially be received negatively as partial or perhaps positively as entertaining. |
| **all_caps_density** | Reviews containing words in all caps may indicate emotion, and therefore may indicate partiality or entertaining writing style (similar to exclamation density). |
| **hyphen_density** | Reviews with hyphens may have better defined structure and breakdown of product aspects, such as "The good - …" or "The bad - …" |
| **sum_word_count** | No initial hypothesis. |
| **sum_exclamation_density** | Exclamations generally imply emotional writing, and serve as a crude measure of subjectivity. Reviews like these may potentially be received negatively as partial or perhaps positively as entertaining. |

| | |
|---|---|
| **contains_update** | Reviews with “update” may indicate that the reviewer returned to the review to improve their evaluation of the product. This requires effort and may indicate the reviewer’s willingness to provide a more helpful review. This was observed several times in the top 20 rated reviews. |
| **review_lateness** | Reviews that were generated a long time after the first review for a particular item may be too lowly ranked in a sorted list of helpful reviews in order to generate positive votes. |
| **log_rating** | The log of rating may be a better distribution for modeling purposes, as this was heavily skewed prior. |
| **lexicon_count** | The number of unique words employed by a reviewer may indicate the user’s ability to convey information and thus lead to a better review. |
| **readability**[1] | The readability score may be an important factor in the willingness of folks to actively read and understand a particular review. |
| **polarity** | The polarity may influence whether someone votes positively or negatively, as some votes may be influenced by favoritism of products or “wanting to hear what you want to hear” mentality. |
| **subjectivity** | The subjectivity of a review may indicate feelings about a review versus pure information about a review. This may influence the response of readers, who perhaps may be looking for one more than the other. |
| **char_count** | Similar to length of review, but more accurate as white space is not counted. |
| **sentence_count** | Similar to word count, length of review, and character count, but provides another level of detail for these relatively correlated measures. |
| **adj_density** | Reviews with adjectives may be more descriptive about product features/qualities and may elicit a more positive vote of the review. |
| **noun_density** | Reviews with more noun density may indicate more information about and may elicit a more positive vote of the review. |

*I also generated additional features using the more detailed **categories** column provided by the dataset. I accomplished this by creating one_hot_encoding using **MultiLabelBinarizer** and selecting the categories that had 100 or more reviews to create columns for. This value was chosen such that the test data would also contain at least one item for each of these categories, which allowed the same encoding function to be used to produce equivalent columns. The produced columns included things like “Accessories,” “Wallets,” and “Women.”*

---

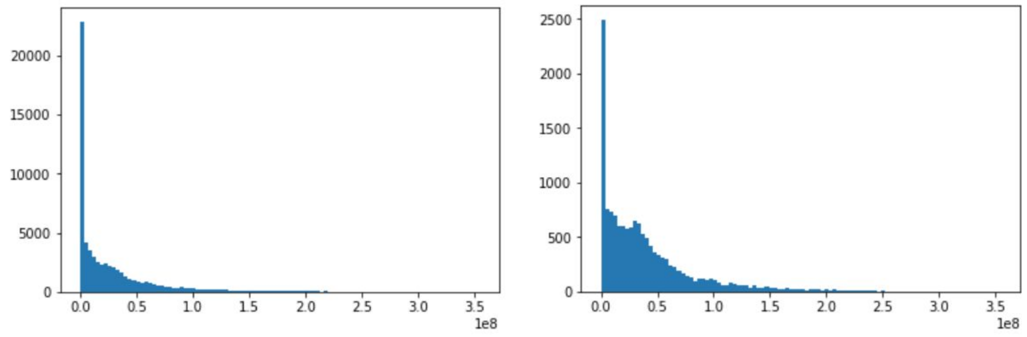
### *Challenges*

*These features provided a wealth of information about reviews, but determining whether these features were impactful towards influencing the review helpfulness was a challenging exercise. In order to better understand how these features may be used to produce a stronger model, my primary strategy included the following:*

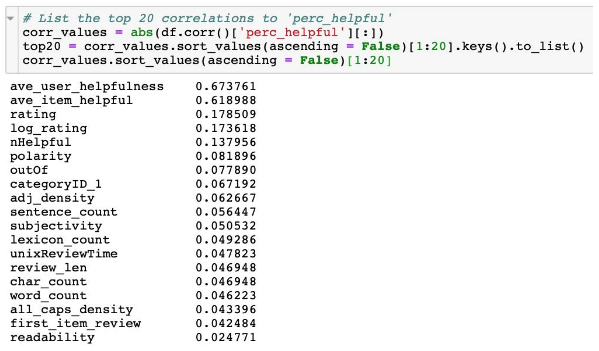
1. *Correlation Analysis*
2. *Distribution Analysis*
3. *Ablation Experimentation*

4. *Evaluating Validation Dataset*

*For example, I analyzed the distributions of the train and test dataset to ensure that my features were producing relatively equivalent distributions. Any significant disturbances to the distribution may indicate that the training data is not completely ideal for prediction or may require additional modification to features. For one example, I reviewed distributions of* ***review_lateness*** *and discovered that the calculations I was using produced negative review lateness for the test dataset because the first date of review for a particular item was computed by creating an array of the review times for the train dataset and subtracting. When items from the test dataset had earlier reviews, this produced negative values. I corrected this by evaluating the formula on the test dataset separately, and correcting for negative values with additional defined functions to produce equivalently distributed datasets:*



*This was performed for all variables iteratively to ensure that features were evaluated as equivalently as possible for both train and test datasets. In addition, correlations were used to attempt to find important features for modeling (see example below):*

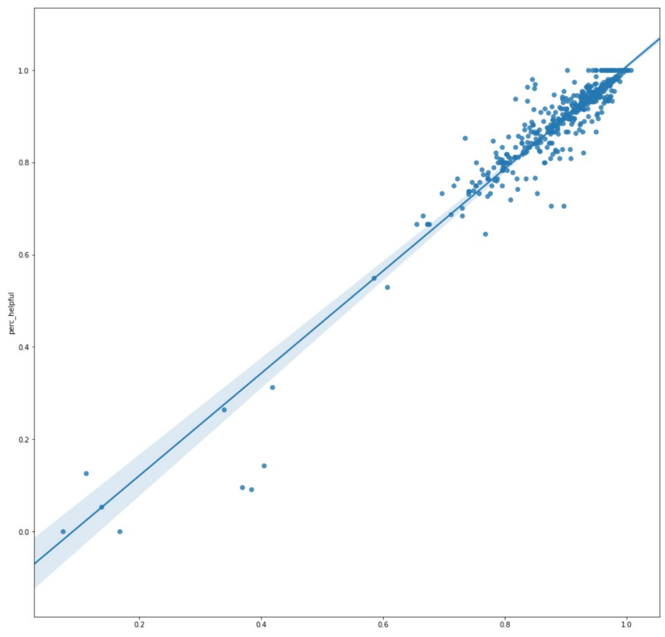


```
# List the top 20 correlations to 'perc_helpful'
corr_values = abs(df.corr()['perc_helpful'][:])
top20 = corr_values.sort_values(ascending = False)[1:20].keys().to_list()
corr_values.sort_values(ascending = False)[1:20]
```

```
ave_user_helpfulness    0.673761
ave_item_helpful        0.618988
rating                  0.178509
log_rating              0.173618
nHelpful                0.137956
polarity                0.081896
outOf                   0.077890
categoryID_1            0.067192
adj_density             0.062667
sentence_count          0.056447
subjectivity            0.050532
lexicon_count           0.049286
unixReviewTime          0.047823
review_len              0.046948
char_count              0.046948
word_count              0.046223
all_caps_density        0.043396
first_item_review       0.042484
readability             0.024771
```

*A significant challenge observed here was that the most correlated features were not always helpful with regards to model performance. Further, certain variables inherently had more information about the result of the train dataset embedded within them than they would for the test dataset (for example: ave_user_helpfulness) and had somewhat misleading performance.*

*I generated importance plots from xgb.plot_importance() in the xgboost experimentation to further understand what splits were used with a high probability in my training dataset. Ablation experimentation was also used to determine what impact each variable had separately, although this required significant processing time and sometimes would produce erratic results depending on random states of the models used. Other significant challenges included inherent distributions of the features, misleading train and val performance when performing validation hyper-parameter tuning, and processing time for data cleaning and feature generation, which required up to 30 minutes for the modeling dataframes to be built (which frustratingly had to be repeated many, many times). As some of the features had inherent advantages of predictability with respect to the training dataset, they did not produce as much benefit to the test dataset when used as modeling variables. This greatly impacted my final decisions on which variables to choose.*



*Actual vs. Predicted Helpfulness during hyperparameter tuning using validation set*

*Additionally, printing logic for the predicted_helpful.txt file was very challenging, as instances where only a few variables were modified due to my logic resulted in greatly varying test results. This caused me to abandon more advanced logic to ensure outOf < predicted and to ensure that predicted values were not negative, although I am positive an appropriate implementation could improve results once the root cause of the unexpected behavior could be found. Additional challenges included vectorization of the review text. I attempted vectorization using tf-idf but the results did not outperform the results obtained from alternative features. I believe term frequency is somewhat removed from the intent of the project, and therefore a text vectorization using a different approach may be more meaningful (such as Word2Vec or more creative word embeddings).*

---

***Results:***

*My best model produced a mean absolute error (MAE) of 0.16954. This passed several of the grading milestones but not all. My model utilized an adaboosted decision tree regression with the following hyperparameters:*

*Max_depth = 100*
*N_estimators = 500*

*My model was trained from the combination of the validation and training data after performing hyperparameter tuning. While the validation error rate was low (MAE as low as <0.10), this performance did not generalize well for the test dataset on Kaggle. It's possible that overfitting greatly impacted my test performance. The final model was trained using only a subset of features that included **char_count**, **outOf**, **price**, **word_count**, **unixReviewTime**, **categoryID_1**, **categoryID_2**, **categoryID_3**, **categoryID_4**, **lexicon_count**, **readability**, **log_rating**, **polarity**, **subjectivity**, and **sentence_count**.*

---

***Conclusion:***

*This report demonstrates that the methods utilized in this report and associated Python notebook produce a relatively successful model for evaluating helpfulness of Amazon reviews. In particular, the use of widely available packages such as TextBlob for Natural Language Processing and breakdown of categories resulted in significant improvement over more crude features such as word counts and character counts. This suggests that the polarity, subjectivity, and perhaps even writing ability of the reviewer greatly impacts the usefulness of a review. Other variables showed impacts as well, such as review time and price, which indicates that the behavior of review evaluators is affected by the item characteristics and may evolve over time in a product lifecycle or even as the marketplace has matured.*

*Opportunities for improvement include more advanced review embeddings and further model breakdowns. I suspect that producing a model for each category or price point may produce much more accurate results, due to the distributions observed in the train dataset. In practice, it may even be best to define a separate model for user characteristics of the review evaluator or shopper. A model designed to provide the most helpful reviews to a particular shopper of Amazon may be more helpful than simply calculating the usefulness of a particular review using the characteristics provided in this dataset alone, as different demographics likely have differing criteria for helpfulness of a review. Another feature I'd like to explore in the future includes misspelling identification. I think this type of analysis may affect someone's opinion of a review and/or prevent them from actively reading a review. I was unable to explore this type of analysis, as many of the top reviews still had several spelling errors present.*

---

***Appendix:***

*Formula for the readability feature, using the Flesch-Kincaid readability test:*

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$

Source: https://en.wikipedia.org/wiki/Flesch–Kincaid_readability_tests