Customer Review Helpfulness Prediction

Abstract

This report delves into the "Amazon Review Data (2018)" dataset, focusing on the Software category with over 459,000 reviews. We aim to predict review helpfulness, a crucial factor for platforms like Amazon. Our methodology involves two steps: classifying reviews as helpful or non-helpful, and predicting the number of helpfulness votes as a matrix of how helpful the review is, and could potentially used for ranking reviews.

Through data exploration analysis, we identified significant trends and developed diverse features including handcrafted, text-based (using Bag of Words, TF-IDF, and Word2Vec), and temporal features. We employed models like Logistic Regression, Ridge Classifier, Random Forest, Temporal Latent Factor, and Multi-Layer Perceptron (MLP), addressing challenges like class imbalance and model over-fitting. The models demonstrated potential for high accuracy and recall, particularly when combining handcrafted and textual features. In regression analysis, temporal features proved crucial, overshadowing the impact of text embeddings.

The study concludes that accurately predicting review helpfulness is feasible, with significant implications for enhancing e-commerce platforms' customer experience. Future research might focus on refining text feature embeddings and their influence on review helpfulness.

1. Introduction

This report will be examining the dataset Amazon Review Data (2018)[1]. This dataset is a collection of millions of reviews ranging from May 1996 to October 2018. The dataset is split into many categories; this report will only focus on the Software category as a case study.

This product category contains 459,436 reviews. Each review has:

- an overall rating
- a True or False indicating whether the reviewer is verified
- the date of the review
- the product 'ASIN' (ID number)
- the reviewer ID

and conditionally contains:

- the 'style' (a dictionary of the product metadata) in 51.01% of reviews
- a review summary in 99.99% of reviews
- review text in 99.98% of reviews
- a number of 'helpfulness' votes in 27.82% of reviews
- an image in 0.33% of reviews

We have designed a predictive task to identify which of the reviews will be voted as helpful. This can be used by a company like Amazon to boost those reviews it predicts would be helpful to users in order to bring them to the top of the list of reviews, even before they are voted on. We believe that this task is achievable if we can split it into two distinct steps: a classification step in which we first identify whether a review will get any votes. We believe this step is necessary, since most reviews (approx. 72%) in the dataset get 0 helpfulness votes. The second step of the task is, given that a review will receive at least one helpfulness vote, to predict how many helpfulness votes it will get, which is a regression task. We will explore a variety of techniques for each of these tasks and present our findings below.

2. Literature Review

2.1. Dataset Introduction and Exploration

The origin of this dataset is a paper named "Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects" by Ni et al. This paper explored techniques to generate justifications for recommendations that would be meaningful and helpful to the user's decision making when purchasing a product. It did this by annotating sections of text as 'good' or 'bad' recommendations, learning a model, and then extracting terms to try and analyze sentiment which are used to build user and item profiles. This work was able to successfully outperform unpersonalized models that it was compared to[1].

An older version of this dataset was also used in two papers. One called "Ups and Downs" by He et al. used collaborative filtering to model how fashion trends evolve over time. The paper extends the latent factor model we explored in class by adding an extra term $< \theta_u, \theta_i >$. These parameters come from features extracted from product images using a Deep CNN by left-hand multiplying them with a learned embedding matrix. This was further extended by adding a temporal visual bias term, as well as temporally evolving the previously mentioned embedding matrix[2]. The other paper by McAuley et al. explored generating recommendations based on images. It explores generating an F-dimensional feature vector using a CNN and learning a distance function such that objects that are related have lower distances than those that are not. It posits weighted nearest neighbor as a distance function, but deems it not expressive enough to model notions such as style. Instead, it explores the 'Mahalanobis distance' but employs a factorization technique to reduce the dimensionality of the learned parameters. The model achieved over 84% accuracy in the tests shown in the paper[3].

2.2. Similar Datasets

2.2.1 Amazon Datasets

The dataset which we are using is the 2018 version, in the past the 2014 and 2013 versions of the dataset have also been released. The major change between the datasets from 2014 to 2018 is that in the earlier dataset the helpfulness field used to contain an array [Helpful Votes, Total Votes] and the 2018 version only contains Helpful votes, although we believe that the Total votes could have been a very useful feature we came up with othe heuristics to estimate popularity of product and review. Another very popular amazon dataset is the Amazon Q/A dataset [4], which contains the question-answer data for every product and can be very useful to learn about every product which in turn will affect its reviews.

2.2.2 Yelp Dataset

Many other review data datasets have been used in the past. Yelp is a major one, which has been used to study review helpfulness [5], review rating prediction and fake review detection [6] extensively.

O ther datasets like IMDB reviews and Hotel Reviews have been studied extensively for sentiment analysis and other related problems.

2.3. Prior Work on Helpfulness Prediction

We are classifying the works into two buckets: Feature Selection focus, Deep Learning focus.

2.3.1 Feature Selection Focus

In their work Singh et al.[7] explore multiple handcrafted features from review text and use those for the task. Their features were based around the fact that the reviews which are easy to read are more likely to get higher helpful votes, we used a very similar feature in our work as well. In Du et al.'s[8] work they explore a much larger umbrella of features like Semantic Features, Sentiment Features, Readability Features, Structure and Syntax. We also implemented Semantic features like TF-IDF, Bag of Words and Word2Vec in great detail.

2.3.2 Deep Learning Focus

In Ahmad et al.'s work[9], they explore the usage of Deep Learning models and LLMs for this task. They have designed Supervised and Semi-supervised algorithms to train the model. The designed CNN, LSTM, BeRT based architectures to solve the problem. In Li et al.'s[10] work they explored CNN model in great detail and proposed a CNN-TRI (CNN-Text Review Interaction) where they have exploited the meta data based features like star rating along with textual features to improve the performance.

2.4. Conclusion

Overall, almost all the papers focus either on the regression task or on the classification task and we believe that our approach to solve this problem is new and robust. After going through the deep learning works we saw that the performance improvement by introducing deep learning based techniques is not substanstial given the complexity they bring so we have tried to focus mainly on Machine Learning based techniques and a very basic deep learning model like MLP.

3. Data Set Analysis

We first inspected the dataset and perform an exploratory analysis. Based on the analysis, we then selected the features and the models that will be used to perform the desired prediction task.

3.1. Data Exploration

Our data exploration process involved a comprehensive analysis of review texts and their associated metrics. The analysis provided insights of the data set from various aspects, including review lengths, ratings, helpfulness, and temporal patterns.

3.1.1 Number of Reviews per Entity

We first inspected how interactions cluster by users and items, i.e. the number of reviews per user and per item. As

shown in Figure 1, We notice a skewed distribution where a small subset of users and items have a disproportionately high number of reviews. The logarithmic transformation on the X axis highlights the heavy-tailed nature of these distributions. Yet, the user interaction are more sparse then items. The median of number of reviews per user is 1, whereas the median of item is 3.

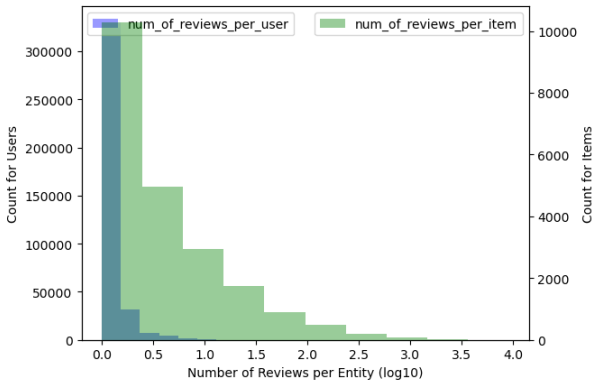


Figure 1: Distribution of Number of Review per Entity

3.1.2 Temporal and Period

The number of reviews and the number of their votes on each days (Figure 2) shows periodic spikes in the counts of both reviews and votes, potentially aligning with specific events or promotional activities. There are more reviews and more votes between day 5000 and day 6500, suggesting that Amazon's software sales are trending during this period of time. Despite these, the number of votes per review on each day shows a more interesting trend. As shown in Figure 3, older reviews tend to have more votes. Older reviews are listed and exposed to users longer so that they get more appreciation from other users. This important insight is exploited in our regression models.

In addition to temporal, we also inspected the periodic aspect of the data. As shown in Figure 4, review helpfulness is mostly equal across weekdays, but reviews from Sep. to Jan. tend to be more helpful.

3.1.3 Features over Bayes Rule

Lastly, we inspected some features' distributions when given the review's helpfulness, i.e. analyzing $P(x|y)$. As shown in Figure 5, *rating*, *verified status*, *review length*, are all shown different distributions. Specifically, helpful reviews are more likely be low rating, from non-verified purchasers, and have longer text.

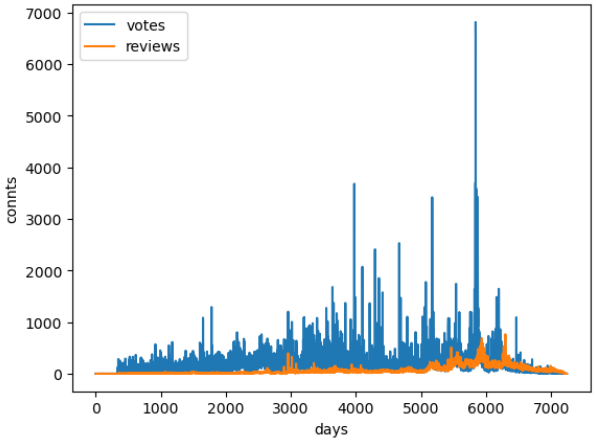


Figure 2: Number of votes and reviews on Each Day

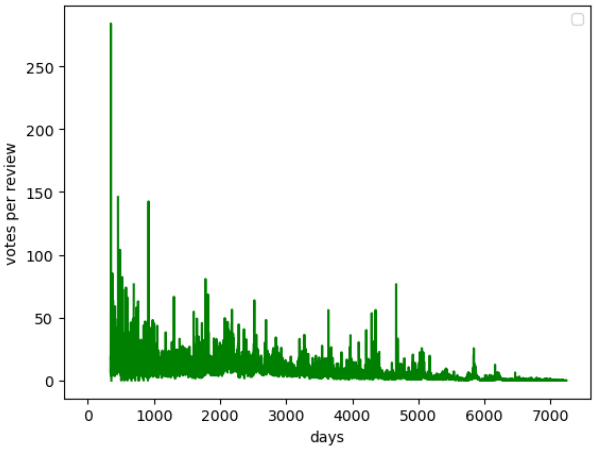


Figure 3: Number of Votes per Review on Each Day

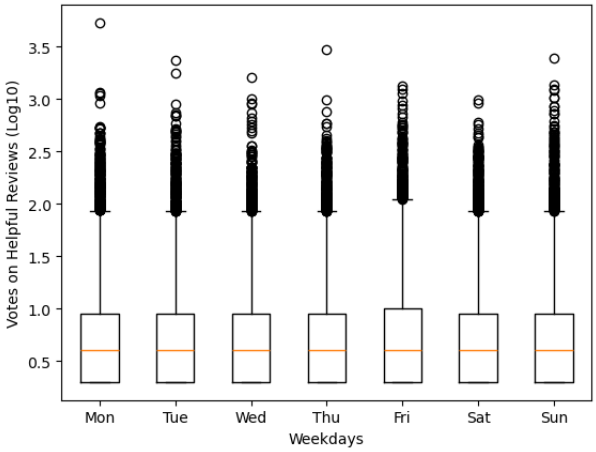
3.2. Feature Engineering

We divide the features used in our models into three parts: Handcrafted Features, Text Review Based Features and Temporal Features.

- **Handcrafted features** are based on the meta data available with the data.
- **Text review based features** are the features based on the review text provided in the dataset.
- **Temporal Features** are used to capture the dependence of data features we care about with time.

3.2.1 Handcrafted Features

These features are based on the data analysis displayed above, we used features which we found useful for distinguishing between helpful and non-helpful reviews.



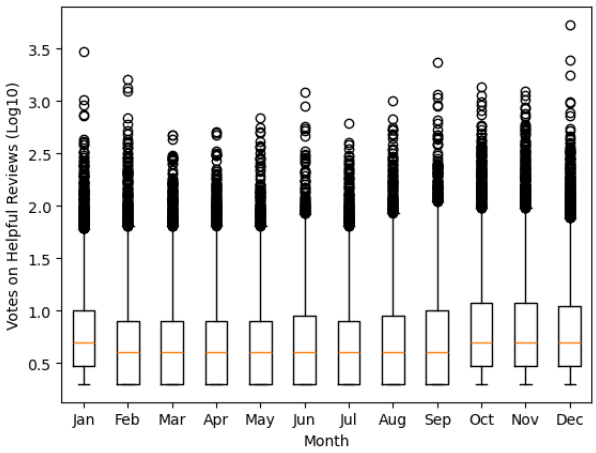


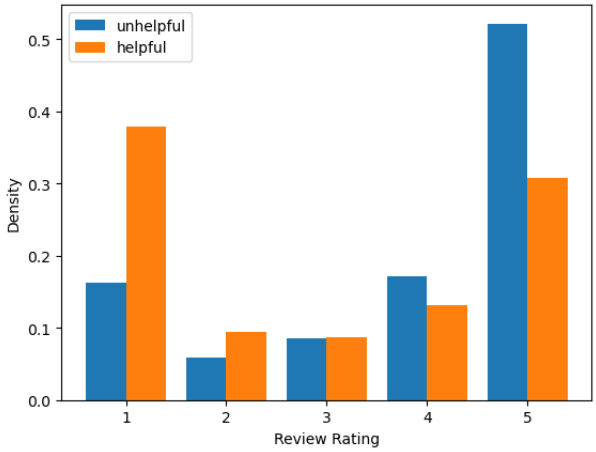
Figure 4: Helpfulness over Period

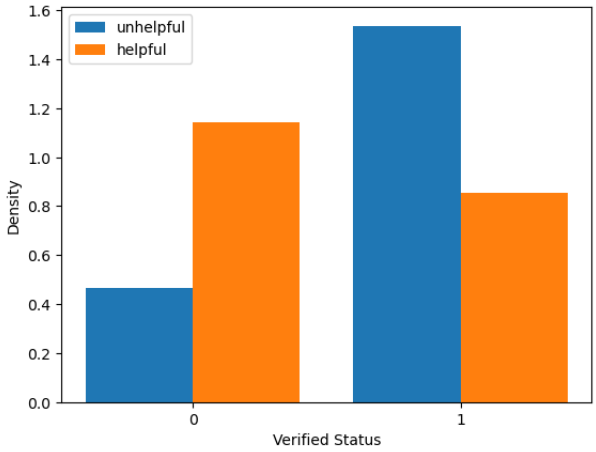
- **Verified**: We used the verified label from the data entries as a feature, we use True as 1 and False as 0.
- **Length of Review**: We used length of review in integers as a feature.
- **Readability**: We used Flesch reading ease score, with 100 being the highest score for most readable text and 0 for least readable review. This feature is useful because more readable reviews are more likely to receive helpful votes.
- **Star Rating**: We used star rating (1-5) as a feature.

3.2.2 Text Review based Features

We do some standard pre-processing like lower-casing all characters and stop word removal. After this we use the methods described below for feature extraction:

- **Bag of Words (BoW)**: We used the top-200 words from review text corpus as a vocabulary and used the standard Bag of Words representation as feature.





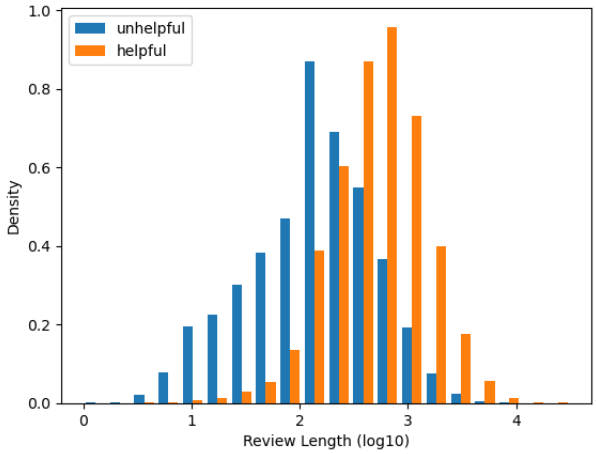


Figure 5: Feature Distributions given Helpfulness

- **TF-IDF**: We used the top-200 words from our review text corpus as a vocabulary and used the standard TF-IDF representation as feature.
- **Word2Vec**: We used a 300-dimension pre-trained Word2Vec representation of words as our starting

point. For getting the feature we just averaged the Word2Vec representation of all words in the review.

3.2.3 Temporal Features

The temporal features are based on our findings in section 3.1.2. Specifically, we defined a function to capture the votes per review trend over time.

$$\text{dev}_{a,b}(t) = a \cdot e^{-bt}$$

In which, a and b are learned parameters. Moreover, we added a periodic bias term to capture the vote difference in weekdays and month.

$$\{\beta_w | w \in \text{weekdays}\}, \{\beta_m | m \in \text{month}\}$$

4. Predictive Tasks and Modelling

To reiterate, we have picked a combination of two predictive tasks to solve the problem at hand.

1. Classification of Reviews into Helpful and Non-Helpful.

2. Predicting the number of helpful votes a review labelled 'helpful' will receive.

4.1. Classification

The Goal of this task is that given a review R, we want to classify whether this review is Helpful (label = 1) or Not-Helpful (label = 0). In a real-world setting any review which receives non-zero helpful votes is helpful and all other reviews are not-helpful. This step is necessary due to the sparsity of helpful reviews.

4.1.1 Class Imbalance

In our task we faced the issue of class-imbalance i.e the number of unhelpful reviews were much higher than number of helpful reviews. In the Dataset only 28% reviews were helpful.

This makes learning harder because without some intelligent metrics, our models would learn more about non-helpful reviews whereas for this problem we care about the helpful reviews.

4.1.2 Basic Models

- **Logistic Regression**: We experimented with logistic regression as our baseline model for the classification task. This model seems a good choice because it converges really fast and is a good starting point. We tuned the regularization parameters by maximizing accuracy on Validation set.

- **Ridge Classifier**: We tried out Ridge classifier because when using text features, the feature size is very large and strong regularization is needed to prevent over-fitting.

- **Random Forest**: We tried out random forest because it an ensemble model and for classification tasks ensemble models are known to do well.

- **MLP**: We tried out MLP model because we wanted to try out deep learning and explore the effect of non-linear activations with MLP.

4.1.3 MLP Architecture

Model Unlike the other models, we created our multi layer perceptron in PyTorch. Specifically, we created a model with a configurable number of hidden layers, and with configurable widths for each layer. We implemented early stopping (based on validation error) with patience to stop training when the model stops generalizing properly, but giving the model some extra time to escape local minima. We also implemented an adaptive learning rate to help with hyperparameter search.

Loss Function To handle the class imbalance issue inherent to the problem we made some adjustment to the loss function:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(w_j \cdot y_{ij} \cdot \log(\hat{y}_{ij})) + \lambda(\sum_{p \in \text{parameters}} \|p\|_2) \tag{1}$$

This is weighted cross-entropy with a configurable L2 regularization hyperparameter. An important design decision was the weight values. We tried multiple combinations, but the one that worked best was:

$$w_c = 1 - \frac{\sum_{i|y_i=c}^{N} 1}{\sum_{i}^{N} 1} \tag{2}$$

The goal here is to prioritize the class with less representation. Another weight function we tried but that did not provide good results was:

$$w_c = \frac{\sum_{i}^{N} 1}{\sum_{i|y_i=c}^{N} 1} \tag{3}$$

Hyperparameters The main hyperparameters that we searched were different layer configurations and different L_2 regularization terms. We searched a few configurations for each layer for up to three layers, and searched a logarithmic space of L_2 regularization values from 10^{-5} to 10^2. The accuracy and recall values reported are the maximum sum of recall and accuracy.

4.1.4 Evaluation Metrics

We used two evaluation metrics:

1. Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
2. Recall = $\frac{TP}{TP+FN}$

where, TP: True Positives, TN: True Negatives, FP: False Positives, FN: False Negatives.
The choice of metrics was based on the fact that we would like our model to predict the true labels as closely as possible and among Helpful/Non-Helpful reviews we care more about Helpful reviews (Positives) and so, a model with high accuracy and recall makes sense.

4.1.5 Results

- Results with handcrafted features

Model	Accuracy	Recall
Logistic Regression	75.9	68.2
Ridge Classifier	74.8	68.6
Random Forest	74.1	76.9
MLP	72.8	79.1

- Results with textual features

Model	Accuracy	Recall
Logistic Regression (BoW)	74.3	65.6
Ridge Classifier (BoW)	73.5	64.3
Random Forest (BoW)	71.1	65.8
MLP (BoW)	72.9	75.2
Logistic Regression (TF-IDF)	71.7	75.0
Ridge Classifier (TF-IDF)	71.9	74.5
Random Forest (TF-IDF)	70.8	65.9
MLP (TF-IDF)	71.7	76.8
Logistic Regression (Word2Vec)	67.9	81.8
Ridge Classifier (Word2Vec)	66.8	81.8
Random Forest (Word2Vec)	66.2	83.0
MLP (Word2Vec)	65.0	82.9

- Results with handcrafted + textual features

Model	Accuracy	Recall
Logistic Regression (Word2Vec)	73.8	73.9
Ridge Classifier (Word2Vec)	**73.8**	**76.6**
Random Forest (Word2Vec)	70.8	81.6
MLP (Word2Vec)	69.1	78.4
Logistic Regression (TF-IDF)	**74.7**	**74.6**
Ridge Classifier (TF-IDF)	74.6	74.7
Random Forest (TF-IDF)	72.7	77.8
MLP (TF-IDF)	69.7	77.6

4.2. Regression

4.2.1 Models

To predict how helpful a review can be, six models were trained. These models were trained only on the reviews that had a nonzero number of helpfulness votes.

- **Bias model:** $\alpha + \beta_u + \beta_i$
- **Latent factor model:** $\alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$
- **Temporal bias model:**
  $\alpha + (\beta_u + \text{dev}_u(t)) + (\beta_i + \beta_{i,weekday} + \beta_{i,month})$
- **Temporal latent factor model:**
  $\alpha + (\beta_u + \text{dev}_u(t)) + (\beta_i + \beta_{i,weekday} + \beta_{i,month}) + \gamma_u \cdot \gamma_i(t)$
- **Word2vec MLP:**
  The model first transfrom u and i into embedding, then concatenate them with dev(t), weekdays, month, verified, review length, and a 300 dim word2vec feature of the review text. Lastly, supply the concatenated vector to a three layer neural network with only one node in the output layer.
- **Tfidf MLP:**
  The structure of this model is exactly the same as word2vec MLP, except now supplying tfidf vector instead of word2vec vector.

4.2.2 Training

The training objective, i.e. the loss function, is:

$$loss = MSE + \lambda(\sum_{p \in \text{parameters}} \|p\|_2)$$

for all models. The data set is split 8:1:1 into train, validation, and test sets. Since we are able to fit all of our data in memory, no batching was used. The training loop implemented early stopping when the MSE of the validation set started to increase to prevent over-fitting. Hyper-parameters, such as λ, embedding dimensions, and hidden node size were searched with 5-fold cross-validation. Figure 6, shows the training process of the models with best hyper-parameters.

4.2.3 Results

Figure 7 shows each model's performance (MSE on test set) with respect to the model's parameter size. We noticed:

1. adding the temporal features leads to the most significant improvement;

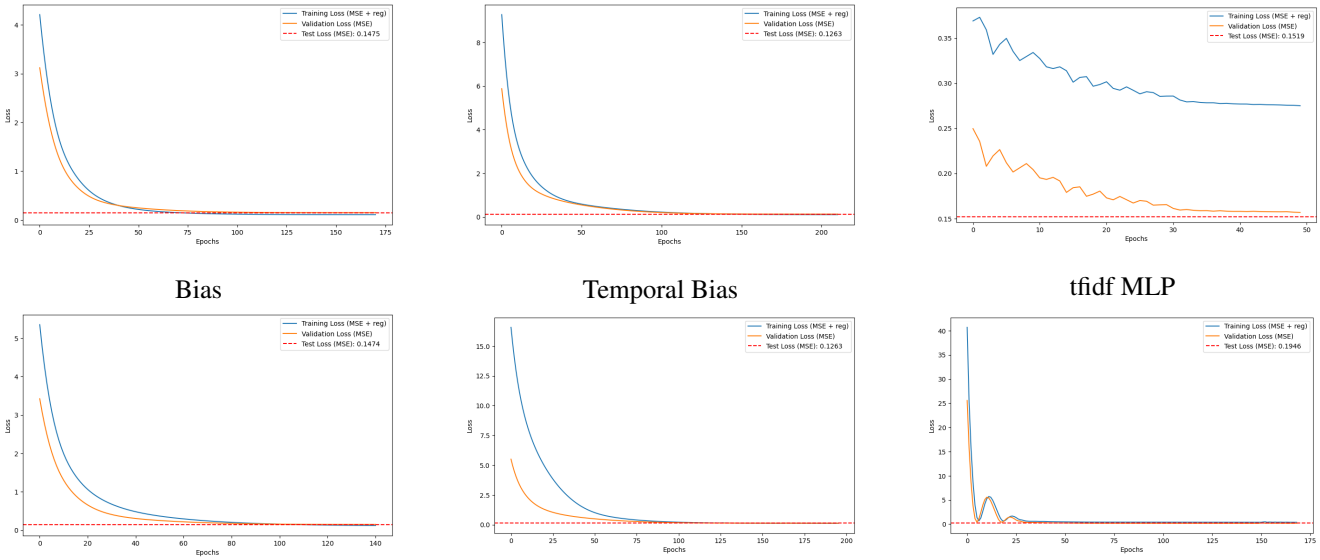


Figure 6: Training Process

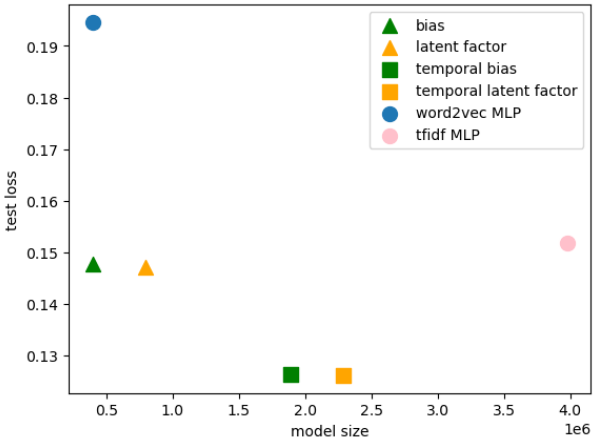


Figure 7: Regression Performance and Model Size

2. adding latent factors does improve model's performance slightly, but less significant than the temporal factors;

3. tfidf is a better text vectorization method than word2vec in our application. The averaging operation in word2vec may averaged out the feature of important words.

Overall, our best model's prediction can predict how many vote a review may has, i.e. how helpful the review is, at $MSE \sim 0.12$, which, in non-log space, is roughly a 25% error.

5. Conclusion

5.1. Handcrafted Feature Importance

We used the coefficients of the learned model to estimate the feature importance and found that verified was the most important handcrafted feature. In Figure 5, we learned that an unverified user is more likely to create a helpful review. According to Amazon, verified reviews just indicate that a user has purchased the product on Amazon, so it seems like we should expect the opposite. Rating and Review length are also quite important for predicting whether a review is helpful. These distributions are less surprising, but also very clearly diverge between classes.

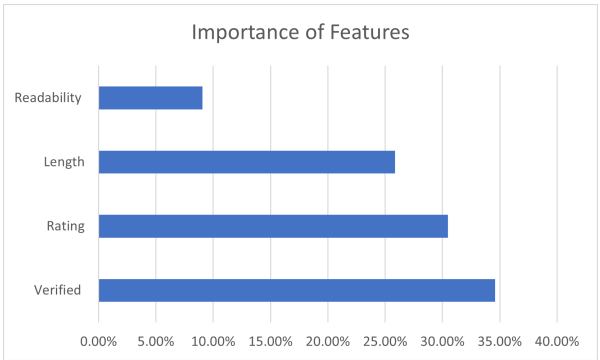


Figure 8: Importance of Various Handcrafted Features

5.2. Feature Visualization

We used Principal Components Analysis to plot the features used by our models in 2D. As visible from the plot using the features we have presented in our models in this work can identify a boundary with which we can identify most of our helpful reviews and discard a good chunk of unhelpful reviews.
The plot acts as additional evidence of the fact that using the features we have designed our classification model can predict helpfulness of a review reasonably accurately. However, it also informs us that the classes do bleed into one another to some extent. This makes us believe that our result for accuracy and recall between 70-80% is a good result based on the problem we are trying to solve and the features we are using.

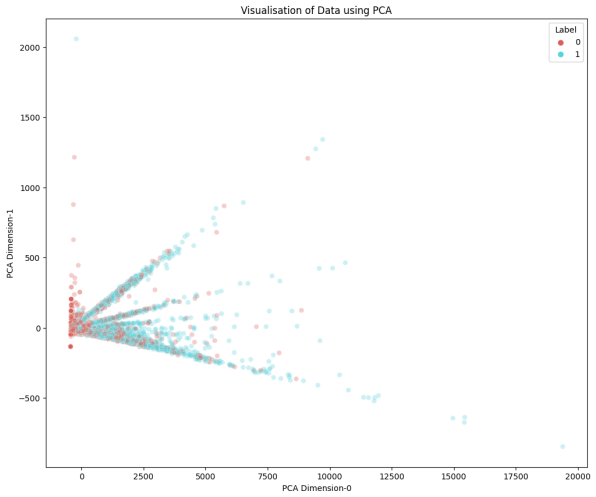


Figure 9: Plot of feature space in 2D using PCA

5.3. Summary of Results

The results above show that we can achieve over 75% accuracy and over 80% recall in the best cases. Of course, there is a balance between recall and accuracy to be achieved. It seems like the most balanced result was about 74.1% accuracy and 76.9% recall, which was achieved with just the handcrafted features by the Random Forest classifier. The actual best choice of model, however, would be determined by how it is used. It may be better to choose the MLP model that achieved 72% accuracy and 79% recall, since one may decide that it would be more useful than a slightly higher accuracy. We imagine that this recommendation system could be used to boost reviews before they receive any helpfulness votes. A company like Amazon could also use more data to solve this same problem, possibly allowing them to find a more accurate model. Since we only tested 3-layer deep neural networks, a deeper network would likely be able to outperform it. With a limited amount of data, it would be infeasible to train so many parameters at once. However, on an industrial scale, Amazon would have many times as much data as we have used for our models if they even just used all categories from the same dataset.

A sensible use of these models would be to temporarily boost the review, say for one month, after it is posted. If that review doesn't receive any helpfulness votes in that time, it would be removed from the list of boosted reviews. The boosted reviews could also be sorted by the results of our regression model. Especially since the most highly-voted reviews are the oldest ones, we believe this is important to bring users recent reviews. Products, especially on a huge platform like Amazon, may go through multiple revisions in the same listing, so the feedback from a few years ago may not fully apply to a user currently buying a product. Our best regression model can achieve approximately a 25% error on helpfulness, which we believe to be a significant result. Many reviews either have a large amount of helpfulness votes, or very few. A 25% error would allow us to preserve which of those buckets a review may fall into after we predict that it would be perceived as helpful. This result would also allow us to further filter out the approximately 25% of reviews that are misclassified as helpful, since they would likely be predicted as only slightly helpful. Our hope is also that most misclassified reviews only have a small number of helpful votes, and that if we create a cutoff for expected number of reviews we can filter out the rest of the misclassified unhelpful reviews..

Another thing to note is that, at least for classification, it doesn't seem like adding many extra dimensions in the form of text features proved to be very useful over just using our handcrafted features. Our handcrafted features did, however, include metrics of readability and review length which was directly related to the content of the text. These metrics appear to have captured a high amount of the information embedded in the text. Further work could examine tuning the length of the embedding for each type of textual feature. Including word2vec features especially reduced the performance of our model, possibly due to the averaging technique we used. This result is further confirmed by our regression results, where the text embeddings-based models performed the worst. In the case of regression, the most important additional feature over per-user and per-item bias was the temporal feature.

References

[1] Julian McAuley Jianmo Ni, Jiacheng Li. Justifying recommendations using distantly-labeled reviews and fined-grained aspects. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019. 1

[2] Julian McAuley Ruining He. Image-based recommendations on styles and substitutes. 2015. 2

[3] Julian McAuley et al. Image-based recommendations on styles and substitutes. 2015. 2

[4] Julian McAuley Mengting Wan. Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems, 2016. 2

[5] Tianxiang Zheng, Zhihao Lin, Yating Zhang, Qi Jiao, Tian Su, Hongbo Tan, Zesen Fan, Dengming Xu, and Rob Law. Revisiting review helpfulness prediction: An advanced deep learning model with multimodal input from yelp. *International Journal of Hospitality Management*, 114:103579, 2023. 2

[6] Bing Liu Arjun Mukherjee, Vivek Venkataraman and Natalie Glance. What yelp fake review filter might be doing?, 2013. 2

[7] Jyoti Prakash Singh, Seda Irani, Nripendra P. Rana, Yogesh K. Dwivedi, Sunil Saumya, and Pradeep Kumar Roy. Predicting the "helpfulness" of online consumer reviews. *Journal of Business Research*, 70:346–355, 2017. 2

[8] Michalska S Du J, Rong J, Wang H, and Zhang Y. Feature selection for helpfulness prediction of online product reviews: An empirical study. In *PLoS One*, 2019. 2

[9] Abdalraheem Alsmadi, Shadi AlZu'bi, Mahmoud Al-Ayyoub, and Yaser Jararweh. Predicting helpfulness of online reviews, 2020. 2

[10] Jaekyeong Kim Xinzhe Li, Qinglong Li. A review helpfulness modeling mechanism for online e-commerce: Multi-channel cnn end-to-end approach, 2023. 2