Project: Amazon Rating Prediction

Project Report
Post Graduate Group 10 Dubai

Introduction

Since the development of a digital marketplace, online reviews have become a crucial metric for a seller or a buyer to gauge the quality and satisfaction of a product. One of the juggernauts in this digital space is Amazon, boasting an immense data lake consisting of multiple products and reviews. This report dives into improving a standard predictive model that would derive meaningful insights from these reviews using text analysis. Sellers or buyers can use such models to gather sentiment data or product performance.

Methodology

This project employed multiple steps and approaches, including different methods for pre-processing, training, and evaluating the data. The dataset consists of Amazon reviews of other arts and crafts products and their subsequent ratings from 1 (lowest) to 5 (highest). The aim is to achieve a model that accurately predicts ratings from unseen reviews and uses the most effective approach.

Initially, multiple tokenizing, stemming, and weighting methods were used to extract relevant features from the data. These were then trained using machine learning models like logistic regression, random forest, neural networks, and Stochastic Gradient Boost. The performance of each model is assessed using appropriate evaluation metrics and cross-validation techniques.

Dataset Review

The accuracy of any model relies on the quality of the data used for its training as much as the process followed it; the data utilized for this model is a modified subset of a larger dataset that consists of reviews and metadata from Amazon spanning from May 1996 to July 2014. Data set source (http://jmcauley.ucsd.edu/data/amazon/links.htm)

The data comprises three files: the train, the test data, and the sample submission. These consist of subsequent structured columns:

Id: used as the identifier for the reviews.

Review Text: The string feedback the consumer provides about the product and its quality, highlighting their sentiment and opinions.

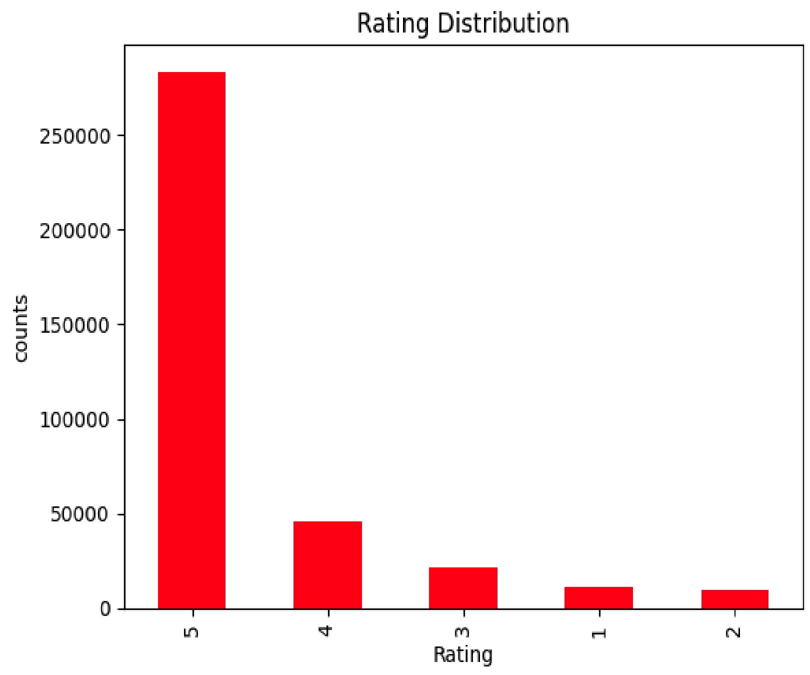
Overall: The numerical value assigned to each subsequent review ranged between 1 and 5, with 1 being the lowest and 5 the highest.

Both train and test data do not contain any null values. It is worth noting that the test data does not contain any ratings as it is meant to be unseen data. The sample file only contains ID and a dummy column for ratings. This file is used to append the predicted test values with their corresponding review IDs to be submitted on Kaggle.

Train Dataset

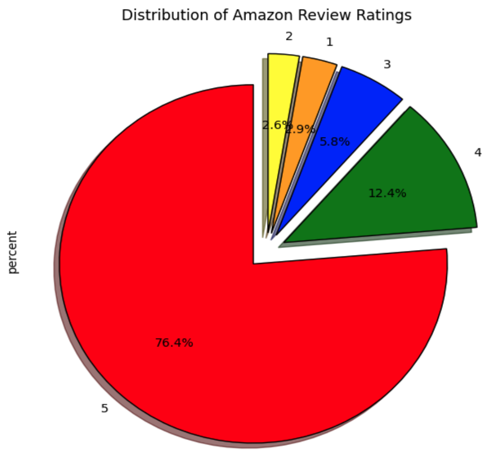
The train data set comprises 1,693,583 individual ratings, providing a robust training foundation for our model; however, the data distribution is highly Imbalanced; this imbalance can be detrimental to the outcome as traditional classification models may introduce a bias towards the majority class.

In this case, the data is mainly imbalanced towards the positive rating side, as evidenced by the distribution chart below:



The Distribution shows that Class 5 is highly overrepresented vs. Class 1 and 2, which are notably underrepresented; the implication of this imbalanced data can lead to poor generalization of the unseen data, and the model may find it difficult to classify the minority classes correctly.

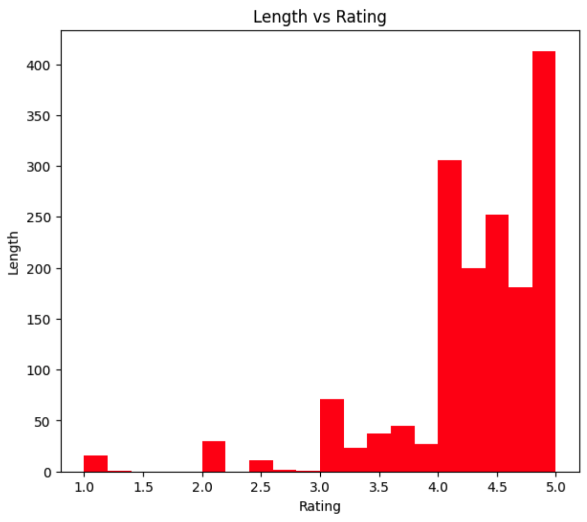
Another such illustration highlights the data imbalance in percentage terms:



We tried multiple methods during our model training and testing to reduce the effect of imbalanced data. These include class weights, SMOTE, and other techniques that will be explained further later in the models.

The dataset also exhibited variability in the lengths of the reviews; reviews ranged from short to detailed and expansive, providing an in-depth analysis of the product; the effect of this comes into play during the analysis as well as the feature extraction, longer reviews, for example, is more information-dense and contain diverse vocabulary along with deeper contextual information. However, these also increase model complexity due to the increased dimensionality. Models trained on longer reviews require additional computational power and longer training time; on the other hand, shorter reviews require less time and resources.

The histogram below depicts the Distribution of review lengths for positive ratings (4 and 5 stars) in our dataset. Notably, we observe a trend where consumers who have left positive reviews tend to provide detailed explanations about the product and their experience. This is evident from the concentration of longer review lengths in the higher rating categories, indicating a propensity among satisfied customers to explain their thoughts better.



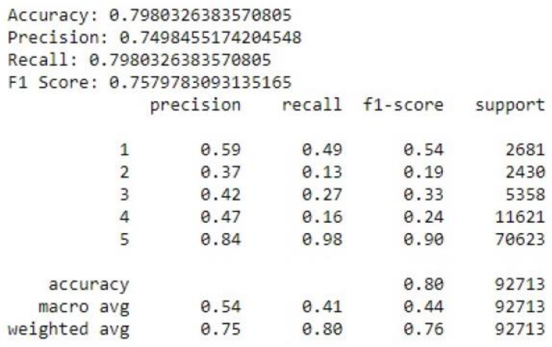
This pattern suggests that consumers who rate a product positively are more inclined to delve into specific details, share personal anecdotes, and offer comprehensive feedback. Such detailed explanations highlight the reviewer's satisfaction and provide valuable insights for potential buyers seeking in-depth information about the product.

Standard Model

The standard model was based on TF-IDF and logistic regression; the first step was to divide the training dataset into two further subsets: ‘to_train’ and ‘to_test’; the training is used to train the model, while the test data is kept for evaluating the model's performance on unseen data. However, in the standard model, the data imbalance was not taken into consideration.

The data is then transformed using the TF-IDF vectorization technique; TF-IDF is used to assign weights to each word based on its frequency in a document versus its frequency across all documents, in this case, reviews. This would capture the different ways the word has appeared across the data set.

The data is then fed into the logistic regression model, Logistic regression is chosen for its simplicity and efficiency. The model estimates the probability of the event occurring based on the given dataset, in this case the model would predict which rating class did the review belong to, based on the training data.



```
Accuracy: 0.7980326383570805
Precision: 0.7498455174204548
Recall: 0.7980326383570805
F1 Score: 0.7579783093135165
              precision    recall  f1-score   support

           1       0.59      0.49      0.54      2681
           2       0.37      0.13      0.19      2430
           3       0.42      0.27      0.33      5358
           4       0.47      0.16      0.24     11621
           5       0.84      0.98      0.90     70623

    accuracy                           0.80     92713
   macro avg       0.54      0.41      0.44     92713
weighted avg       0.75      0.80      0.76     92713
```

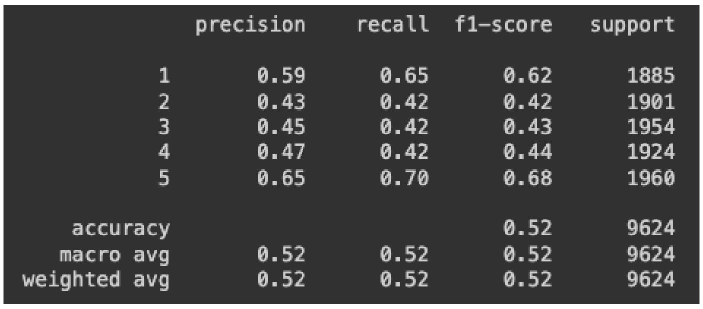
The classification report indicates the large gap between the classes from the support metric, the model overall scored 80% with high precision, recall and f1-scores for class 5 which has the largest number of reviews, the high accuracy could be due to the subsequent train split also containing data in the similar imbalanced format.

Logistic Regression with down-sampling to minority class

For this model we used the same vectorization method and classifier as the standard model, we wanted to understand the implication of introducing any basic pre-processing method along with any technique to tackle the data imbalance.

For the basic pre-processing, we started by removing HTML tags, special chars, numbers, and brackets, as well as lowering the text and removing any extra spaces. The data is then balanced using down-sampling based on the minority class values, and the count of the 2nd class is taken based on which the other classes are sampled; this process ensures that the Data is balanced, and the classes match the minority class, The data is then split into train and test data and vectorized using TF-IDF and fed into the logistic regression model.

However, down-sampling is effective only when the imbalance is not severe; in this case, due to the imbalance being large, a major chunk of the data was discarded from the other classes, which affected the overall model as informative data points were lost; this overall affected the variability of the model as well as the size of the training data, this reflected in the overall classification report:



```
              precision    recall  f1-score   support

           1       0.59      0.65      0.62      1885
           2       0.43      0.42      0.42      1901
           3       0.45      0.42      0.43      1954
           4       0.47      0.42      0.44      1924
           5       0.65      0.70      0.68      1960

    accuracy                           0.52      9624
   macro avg       0.52      0.52      0.52      9624
weighted avg       0.52      0.52      0.52      9624
```

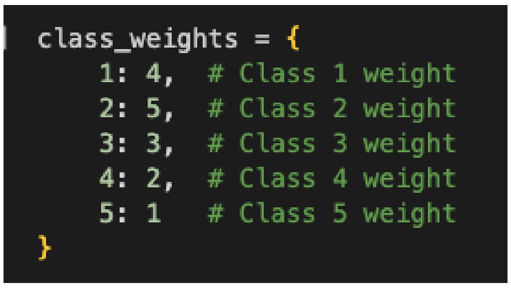
The overall accuracy is much lower as compared to the standard model, as seen in the precision, recall, and F1-score, which are low as well, indicating the lack of training data in the model and the down-sampling method in this case ineffective.

Bi-gram logistic Regression (Weighted Class)

Reflecting on the previous models result, demonstrating that under-sampling method was unsuccessful in handling the imbalance, we opted to test the weighted class method to tackle the data imbalance.

The concept of class weights refers to the process of assigning weights to each of the classes during the training of a model, this method is used to mitigate the impact of imbalanced data by ensuring that the model takes into consideration the minority class during training.

In this case, the weights assigned to the classes were as follows:

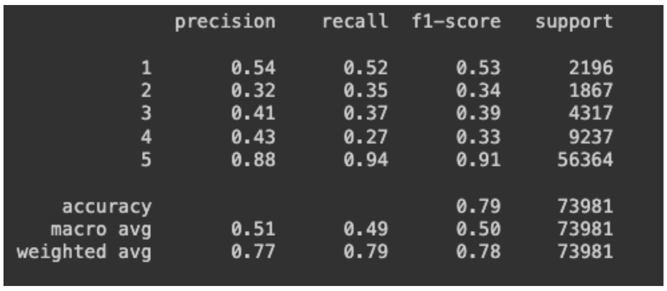


```
class_weights = {
    1: 4,   # Class 1 weight
    2: 5,   # Class 2 weight
    3: 3,   # Class 3 weight
    4: 2,   # Class 4 weight
    5: 1    # Class 5 weight
}
```

The model follows a similar pre-processing, vectorization, and classification methods, TF-IDF with logistic regression, however in this model the TF-IDF is done on bi-grams and the logistic regression is trained taking into consideration the class weights.

Bi-gram TF-IDF refers to when the text is tokenized in pairs instead of single words; bi-grams capture more contextual information compared to uni-grams, which could be beneficial in cases like these where reviews can have similar words but could be utilized in different contexts.

The result of this did show a significant improvement compared to the **under-sampling model**:



```
              precision    recall  f1-score   support

           1       0.54      0.52      0.53      2196
           2       0.32      0.35      0.34      1867
           3       0.41      0.37      0.39      4317
           4       0.43      0.27      0.33      9237
           5       0.88      0.94      0.91     56364

    accuracy                           0.79     73981
   macro avg       0.51      0.49      0.50     73981
weighted avg       0.77      0.79      0.78     73981
```

However, the accuracy remains below the standard model, in addition the precision, recall and f1-score is substantially higher for the 5th class, likely due to the data imbalance.

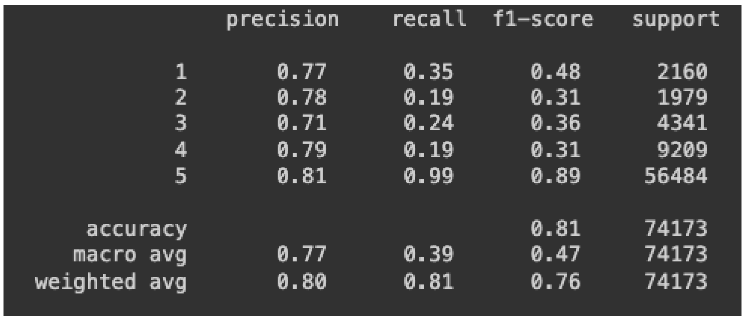
One-vs-Rest Random Forest

The text processing is similarly as per the other models and used the same vectorizer as well, however each class has a different classifier based on it.

After hyper parameter tuning using random search, the usage of one vs. rest fits one classifier per class. For each classifier, the class is fitted against all the other classes; the best parameters returned from the Random search were:

{'estimator__n_estimators': 100, 'estimator__max_depth': None}

The accuracy given based on the classification report was an improvement vs the other models:


```
              precision    recall  f1-score   support

           1       0.77      0.35      0.48      2160
           2       0.78      0.19      0.31      1979
           3       0.71      0.24      0.36      4341
           4       0.79      0.19      0.31      9209
           5       0.81      0.99      0.89     56484

    accuracy                           0.81     74173
   macro avg       0.77      0.39      0.47     74173
weighted avg       0.80      0.81      0.76     74173
```

Precision is relatively consistent across classes; recall varies significantly, with notably poor recall for classes 2 and 4. This results in a low macro-average recall of 0.39, suggesting the model struggles to identify all classes equally well. It likely performs best in the majority class (class 5) while failing to classify minority classes correctly. The F1 Score is consequently low for these classes.

Note:

An important conclusion for the bag of word representation either for unigram or bigram the computational power needed is huge to handle all these features; another experiment was conducted on tri gram with tfidf, but it gave a memory error even with decreased number of features the only way is to handle with custom generators to perform batch by batch which takes more time the sequence models and the kernel crashes in between .unigram is slightly better in terms of f1score and accuracy.

Summary of CNN Models on Text Classification:

The text processing is done are as per the other models, however the data set is cleaned by removing duplicates from the largest Class (5th Class) and weights are introduced during the model training.

The data is tokenized using the Keras library and a vocabulary is created where these tokens are mapped to a unique integer index, this is necessary for the model to represent each text as a numerical value, the data is then converted into a sequence based on the vocabulary created where each text is replaced by its corresponding integer.

This process is followed by the padding sequence, this is necessary for CNNs as all input sequences should have a uniform length, in this process zeros are added to the sequence to make sure all of them have a uniform length.

Due to having multiple CNN models, these are the essential findings and summaries of our experiment:

CNN with Embedding Layer: performed well on training data, improving accuracy with each epoch. However, this improvement was not reflected in the validation set, where accuracy dropped, indicating overfitting. The loss measures also mirrored the overfitting: while training loss declined progressively, validation loss increased, supporting the conclusion that the model’s ability to generalize to new data was damaged.

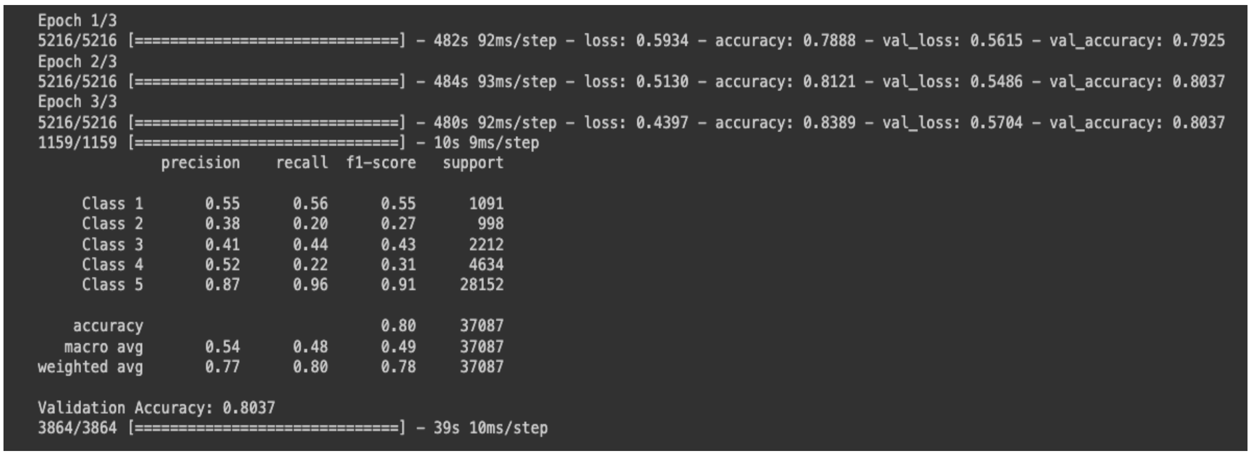
CNN with GloVe Embeddings: The incorporation of GloVe embeddings into the CNN model yielded high training and validation accuracies. However, the variability in validation accuracy suggests a challenge in further enhancing the model’s performance. Despite a stable training loss, the validation loss’s stability, combined with fluctuating accuracy, might indicate potential overfitting issues. The implementation of early stopping at the fourth epoch effectively addressed these concerns, maintaining high precision, recall, and AUC values and suggesting a competent model performance overall.

CNN with Data Augmentation: The CNN model equipped with data augmentation (synonym replacement) techniques demonstrated a tendency to overfit by epoch 4, as indicated by a divergence between training and validation performance—with training metrics remaining high and validation metrics dropping. This trend was further corroborated by a lower AUC on validation data.

Multilayer perceptron:

This model performs the text analysis similarly to the CNN model however in this model the embedding layer is followed by one or more fully dense layers as compared to CNN where the embedding layer is typically followed by a convolutional layer, pooling layer and fully dense layers.

Additionally, the MLP handles the sequential data differently as well, the CNN takes into consideration the sequential pattern of the data while the MLP considers the semantic relationship but disregards the sequential order.



```
Epoch 1/3
5216/5216 [==============================] - 482s 92ms/step - loss: 0.5934 - accuracy: 0.7888 - val_loss: 0.5615 - val_accuracy: 0.7925
Epoch 2/3
5216/5216 [==============================] - 484s 93ms/step - loss: 0.5130 - accuracy: 0.8121 - val_loss: 0.5486 - val_accuracy: 0.8037
Epoch 3/3
5216/5216 [==============================] - 480s 92ms/step - loss: 0.4397 - accuracy: 0.8389 - val_loss: 0.5704 - val_accuracy: 0.8037
1159/1159 [==============================] - 10s 9ms/step
              precision    recall  f1-score   support

     Class 1       0.55      0.56      0.55      1091
     Class 2       0.38      0.20      0.27       998
     Class 3       0.41      0.44      0.43      2212
     Class 4       0.52      0.22      0.31      4634
     Class 5       0.87      0.96      0.91     28152

    accuracy                           0.80     37087
   macro avg       0.54      0.48      0.49     37087
weighted avg       0.77      0.80      0.78     37087

Validation Accuracy: 0.8037
3864/3864 [==============================] - 39s 10ms/step
```

Initially, the model's validation loss declined, which is typically a positive indicator of learning. However, by the third epoch, the validation loss began to rise, hinting at the onset of overfitting. The model performs well in class 5 but struggles with minority classes, especially Class 2 and Class 4, which have notably low recall and F1 scores.

Analysis of Sequence Models for Text Classification:

sequence models used are Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), and Gated Recurrent Unit (GRU), Bidirectional (GRU). We focused on including embedding techniques, batch sizes, learning rates, and architectural adjustments, on model performance across key metrics: accuracy, precision, recall, F1-score, and validation loss.

LSTM with GloVe Embeddings:

An LSTM model with GloVe embeddings Utilized pre-trianed word vectors, it consists of an embedding layer initialized with GloVe embeddings, followed by one or more LSTM layers to capture sequential dependencies.

The model demonstrated a promising balance between training and validation performance, suggesting effective generalization with minimal overfitting compared to CNN and MLP models, with closely aligned training and validation metrics.

LSTM with embedding layer from Keras with batch size set to 64 and 32

A lower recall and precision for some classes but overall accuracy improved from the one with Glove and changing the architecture from the 128 neurons in LSTM layer to 64 resulted in better results and less computational power consumed for further hyperparameter tuning.

With the batch size set to 32 there's an incremental improvement across epochs, with the final validation accuracy reaching 81.63%. The smaller batch size is providing a better estimate of the gradient, potentially leading to more robust learning. But class 2 and 4 still struggling to get better results.

Adam learning rate = 0.0001

A considerable improvement in recall for class 1, The model's recall for Class 2 decreased significantly, adversely affecting the F1-score for this class.
A small learning rate might be inadequate for the model to effectively learn the features associated with Class 2 within the limited timeframe of three epochs. Conversely, we found that a smaller batch size tends to yield better results, yet the learning rate presents a nuanced challenge: as it decreases, performance deterioration occurs. This points to the need for a careful balance between the learning rate and batch size to optimize model performance across all classes.

Bidirectional LSTM (BiLSTM):

Enhancements through increased embedding dimensions led to improvements in accuracy and precision, albeit at a higher computational cost. Early signs of overfitting were observed with increased validation loss, suggesting a need for cautious optimization of training duration and monitoring of validation metrics.

Gated Recurrent Unit (GRU) and BiGRU:

These models showed continued learning and an ability to generalize well over additional training epochs. Class-specific performance exhibited nuanced improvements without immediate overfitting, although there were indications of potential overfitting risk with prolonged training that was stopped by that handling of early stopping even when the models were trained further, for the main minority classes 1 and 2 the recall has increased balancing for a better f1 score especially in GRU , as in BiGru it fluctuated a little at figuring class 4 and 2 with lower recall than GRU. As for class 5 the models are identifying in steadier pace.

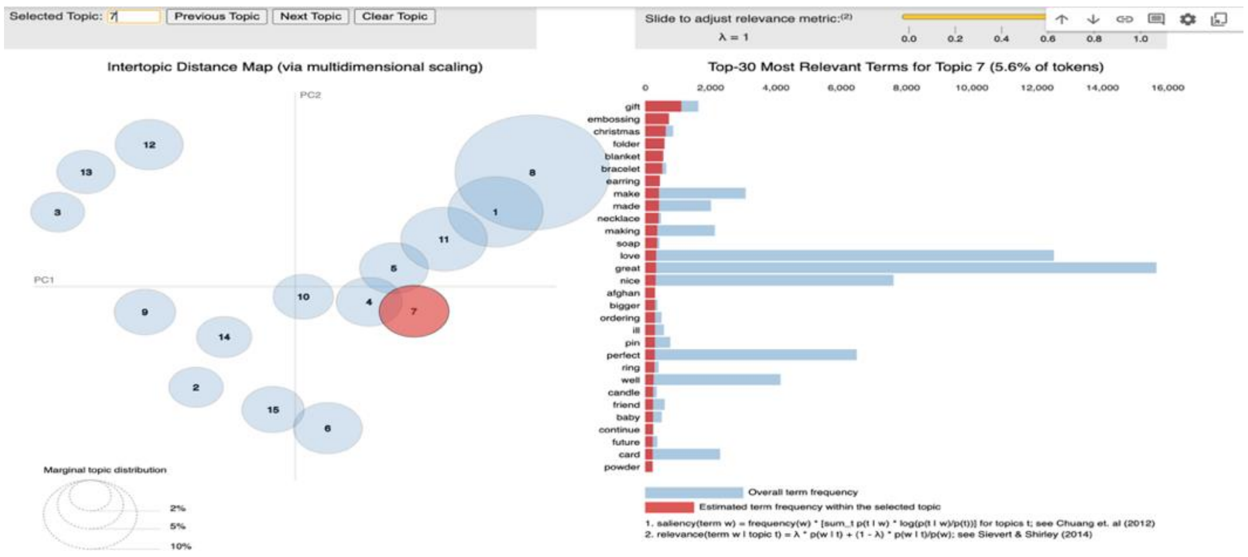
The best model Across the sequence model is GRU with 0.82068 on the 70% of the test data, aside from accuracy the metrics of recall, precision, f1 score in all classes were the best with macro f1score of 51% and weighted one of 79% with balanced scores for other metrics.

Topic modeling:

We used a special method called LDA to look closely at what's common in the best (five-star) and the worst (one-star) reviews people give. This method is good at finding hidden patterns in a lot of texts, like reviews. It compares the good and bad reviews to find out what topics come up a lot in each. This helps us understand what makes customers really happy or unhappy with something. By using LDA, we get a clearer picture of what makes a review good or bad.

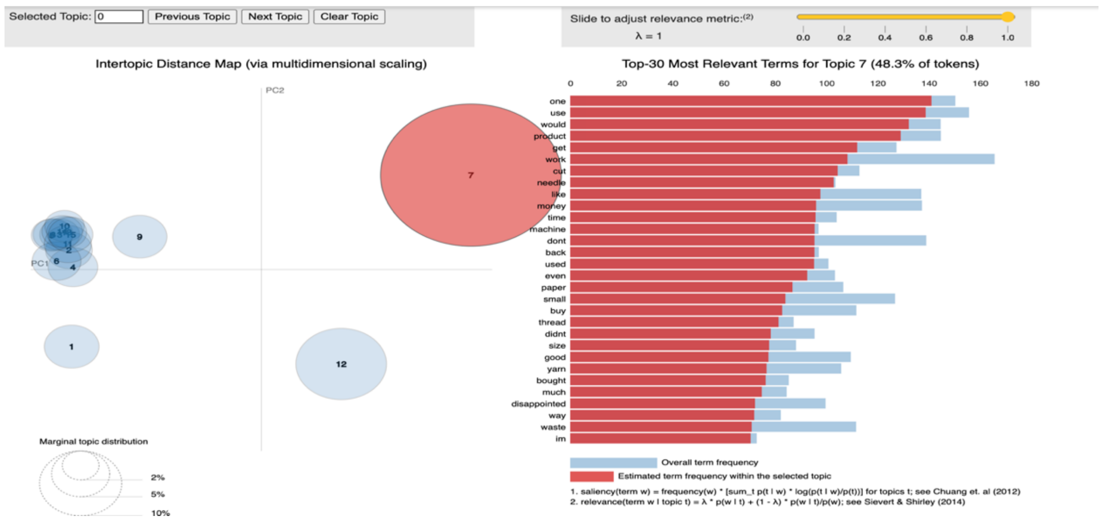
Latent Dirichlet Allocation (LDA) is a statistical model that documents as mixture of topics, where each topic is characterized by a distribution over words. This model is particularly adept at handling cases where documents encompass several distinct topics, making it a valuable tool for analyzing discrete data. The utilization of PyLDAvis for LDA output visualization facilitates a deeper understanding of the topics derived from a corpus by clearly illustrating their sizes and interrelationships.

An application of this technique to customer review analysis reveals 15 unique topics through an interactive interface, allowing for an in-depth exploration of customer feedback. If we look into Topic 7, it is identified among high-rated reviews and sheds light on the positive features highlighted by customers. Keywords such as "gift," "embossing," and "Christmas" associated with this topic suggest that products enhancing festive celebrations and creative projects are particularly appreciated. This detailed examination understanding of the attributes that drive customer satisfaction.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

On the other hand, when we look at topic 7 in low-rated reviews, particularly focusing on reveals customer dissatisfaction centered around practical issues with product use and effectiveness. Terms like "one," "use," "product," "get," and "work," alongside expressions of regret such as "disappointed," "money," and "waste," highlight the unmet expectations and perceived poor value. This detailed exploration of negative.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

In examining the range of topics extracted from customer feedback through high and low-rating filters, our analysis reveals the following insights:

For High-Rated Topics:

- Enthusiasm is evident for crafting essentials such as yarn, needles, and crochet hooks, with recurring affirmative terms like "love," "great," and "favorite."
- Materials for sewing or quilting receive praise, highlighted by words such as "useful," "quilting," and "embroidery," which reflect consumer contentment.
- A rich diversity in the tools for card-making or scrapbooking is appreciated, as suggested by frequent references to "variety," "design," and "collection."

For Low-Rated Topics:

- Dissatisfaction arises about product or service issues, discernible from terms including "problem," "return," and "issue."
- Expectation gaps are indicated by words such as "missing," "supposed," and "expected," pointing to customer letdowns.
- Comments on product quality or functionality are critical, with descriptors such as "break," "faulty," and "poor" suggesting areas needing improvement.

These findings presuppose that the language used within each topic accurately mirrors the sentiments and levels of satisfaction in the underlying reviews or textual data. This understanding helps businesses aiming to respond effectively to customer feedback, enhance product quality, and refine their marketing communications.

Key Findings:

Class Imbalance: Persistent across models, class imbalance significantly impacts the recall and F1-scores of minority classes, despite overall high accuracy and precision.

Model Generalization: Achieved through careful hyperparameter tuning and model architecture adjustments. Early stopping and other regularization techniques were pivotal in mitigating overfitting.

Hyperparameter Sensitivity: The performance of sequence models is highly sensitive to batch size, learning rate, and embedding dimensions, necessitating fine-tuning to optimize for specific task requirements.

Conclusion:

In our analysis, we found a few key challenges that affect the performance of our models.
Firstly, dealing with class imbalances proved to be a tough nut to crack. No matter what model we used, we consistently faced difficulties with minority classes, resulting in lower recall and F1 scores. We tried down-sampling to address this issue, but it ended up causing us to lose valuable data. While weighted class methods in logistic regression helped somewhat, they did not beat out the standard model.

Next, we encountered generalization problems, particularly with our CNN and MLP models, which tended to overfit. This was evident from the rise in validation loss despite lower training loss. On the flip side, our LSTM models with GloVe embeddings and BiLSTM showed better generalization with less overfitting, as indicated by similar training and validation metrics.
Regarding hyperparameter tuning, we discovered that smaller batch sizes improved LSTM performance incrementally, likely due to more accurate gradient estimation and robust learning.

However, setting a very low learning rate hurt recall, especially for underrepresented classes.

In terms of data pre-processing and vectorization, we applied basic techniques like removing HTML tags and special characters. Interestingly, removing stop words actually had a negative impact, as it stripped away relevant features and hindered our ability to differentiate context.
We also experimented with different vectorization methods and found that bi-gram TF-IDF captured more nuances but demanded more computational resources.
In assessing classification metrics, we learned that accuracy wasn't always the best indicator, especially with imbalanced data. Instead, macro and weighted average F1 scores offered a clearer picture of class-specific performance.

As for model architectures, sequence models like GRUs and BiGRUs proved effective with additional training, although there was a risk of overfitting if not monitored. Simplifying the architecture of LSTM models by reducing neurons helped lighten the computational load and improved results.
Lastly, we grappled with computational demands and efficiency. Bag-of-words representations, especially with higher n-grams, were computationally intensive and prone to memory errors. Despite their complexity, sequence models struck a good balance between performance and computational demand when finely tuned.