Sentiment Analysis Of Amazon Reviews

Luxury Beauty Dataset

Yuvraj Kapoor
kapoor.y@northeastern.edu
April 18, 2024

Abstract

Sentiment Analysis of Amazon Reviews is of high demand for customer retention, analyzing feedback and improving product quality. However, filtering the changes in the product can be challenging for the manufacturers. Moreover, these reviews vary in their structure, composition and semantics. By preprocessing textual data and employing classification models such as K-Nearest Neighbors, Decision Trees, Random Forests, and Logistic Regression, the aim is to predict customer satisfaction and sentiment orientation. The study highlights the effectiveness of machine learning in extracting meaningful insights from consumer textual reviews, contributing to better strategic decisions. The proposed approach achieves an accuracy of 90% which is very similar to the state-of-the-art techniques.

Introduction

Problem Statement

Customer reviews in general include a treasure trove of unstructured textual data that provides opinions and feedback of the user. Manual analysis would be utterly unworkable, since traditional quantitative methods are unable to capture the sentiments of human emotion and feeling. Analyzing data into an actionable insight is beneficial for both the manufacturer and the end user to make better informed decisions regarding their purchases. Moreover, since consumer expectations are notably high, understanding sentiment through product reviews is crucial for brand management and marketing strategy.

Prior Work

Previous studies have demonstrated the viability of machine learning in parsing consumer sentiment, yet there is a gap in targeted analysis for high-end

beauty products. The state-of-the-art solution consists of basic lexical approaches to machine-based approaches to hybrid approaches. Recent advancements have seen the adoption of machine learning techniques, including support vector machines (SVM), decision trees, and random forests, which have improved accuracy. While these advanced models offer improved accuracy, they also introduce challenges related to computational complexity and interpretability.

Summary of Proposed Approaches

The approach deploys natural language processing in preprocessing review data before the actual use of a number of machine learning models for classification of sentiment of the reviews into positive, neutral, or negative. The performance of these models was assessed based on accuracy, precision, and recall, with ensemble methods like Random Forest showing promising results in handling the nuanced dataset.

Methodology

In this work, the problem is formulated as a multi-class classification problem. The data parsing is taken care by the parse function reading gzip compresses files containing json entries. Each line in the gzip file is expected to be a separate JSON object, which is yielded as a Python dictionary. The getDF function is used to construct a pandas data frame by utilizing the parse method.

The data is preprocessed mainly using the nltk library. The columns such as Style, image, reviewerName, verified and unixReviewTime are dropped while reviewText, summary and votes are imputed to remove NaN values.

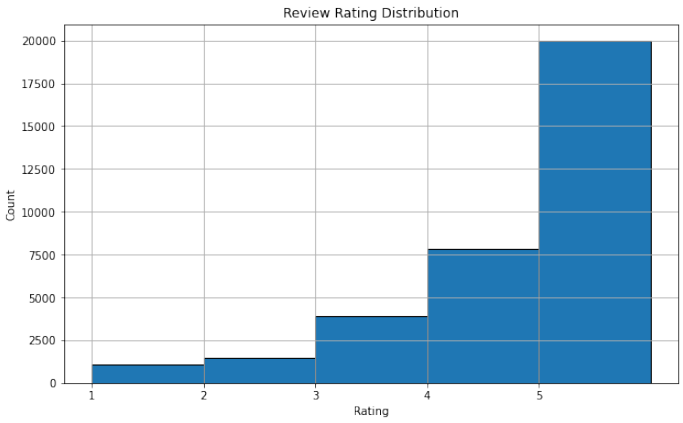
The sentiment column is created labelling 4/5 star ratings as Positive, 3 star rating as Neutral and 1/2 star ratings as Negative. The reviewText and summary

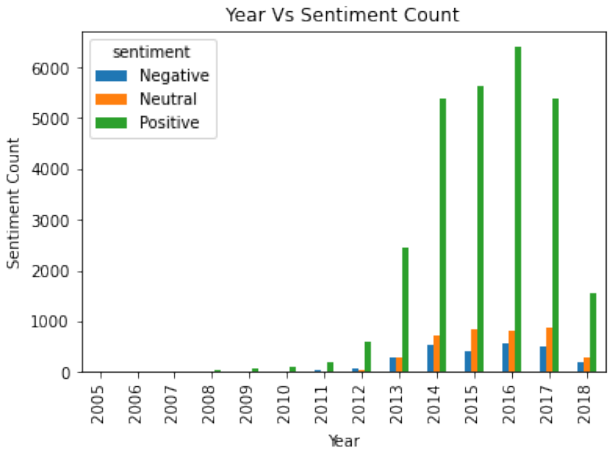
columns are concatenated to a new reviews column as both contain text for reviews.

Regular Expressions are used to clean text, lowercase it and remove all the html tags and urls. Exclamations and Question marks are kept since they are important while doing sentiment analysis. A list of custom stop-words[1] is created to account for words like not, nor, below, under, over ,etc. which are included in stopwords of preexisting libraries such as nltk that harm model's accuracy and training.

The polarity[2] score is calculated to categorize text into positive, negative, or neutral based on the sentiment expressed. The length of each review is calculated since it could correlate with the depth of the sentiment expressed. The number of words of each review are calculated for better verbosity and feature engineering practices.

Visualizations



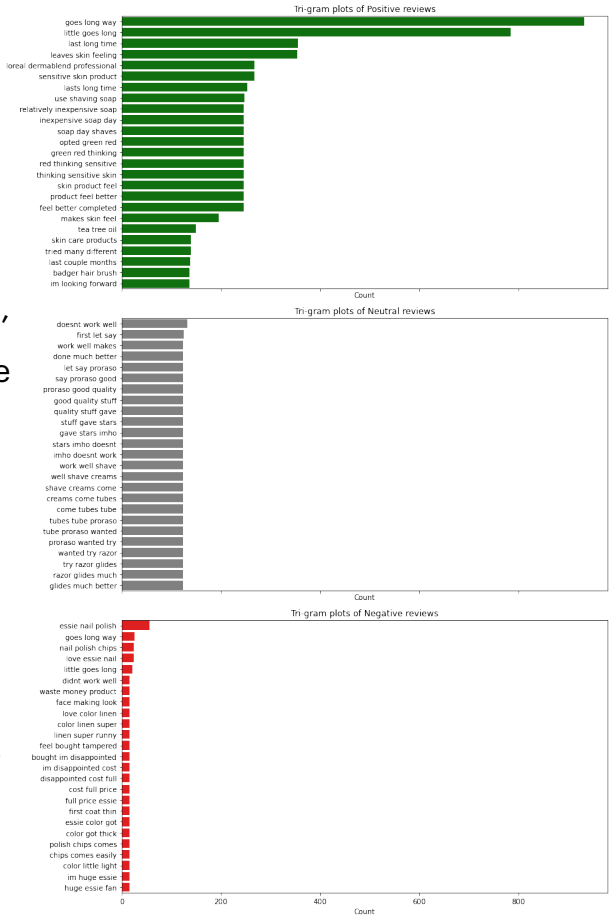


These plots shows the loss of negative and neutral reviews. Clearly, positive reviews and 5 star ratings overpower the rest of the data. The need of the

[1] Majority if these are taken from the nltk library and ChatGPT helped to suggest some more while some are removed by the n-gram analysis which is done further

[2] Polarity is a float within the range [-1.0, 1.0] where -1.0 signifies a highly negative sentiment, 0 signifies neutrality, and 1.0 signifies a highly positive sentiment.

technique of resampling is crucial to balance the nuanced dataset.



The unigram, bigram and trigram were important to account for stopwords removal and giving a better idea to streamline the TF-IDF vectorization. Here is a trigram analysis, which shows that how these words account for the sentiment expression. The unigram had similar words for all the three sentiments, evident from the dataset. The bigram had more meaningful insights segregating the position reviews from neutral and negative reviews.

The trigram analysis showcases the difference in word choice of the reviewers essential in model selection and evaluation.

Model Training

Before training the models, sentiments are encoded using the LabelEncoder dividing ***2 as Positive, 1 as Neutral and 0 as negative.*** These are important as all confusion matrixes use this encoding. Lemmatization is used for the text of the reviews and BorderlineSMOTE is used to resample the nuanced dataset as discussed in the first and second visualization above. Particularly, Borderline SMOTE helps in creating examples near the decision boundary, which can be crucial for training classifiers to make nuanced decisions in regions where class overlap makes classification difficult.

The data is divided into test-train split with the sentiment column being the response and TF-IDF is performed on the features. 20% of the dataset I used as the

testing set. TF-IDF transforms text into a numerical representation that can be used by machine learning algorithms, which otherwise cannot directly process raw text.

Finally, several classification algorithms such as Logistic Regression, Decision Tree, Random Forest, and K-Nearest Neighbors are implemented using Scikit-learn. The confusion matrix provides visual feedback on model performance by showing the percentage of correct and incorrect predictions for each class. Cross-validation is used with KNN to improve evaluation metrics. Decision Trees utilize the information gain(entropy) to train the model while Random Forests is trained with different number of trees such as 10,20 and 30. Due to time complexity, the number of trees could not be increased.

Dataset

The dataset comprised 34,278 reviews, each tagged with metadata such as reviewer ID, product ID, and review timestamps. The reviews were collected from the Amazon Luxury Beauty section. Any 5-core dataset can replace this dataset. Here's a summary of the key columns in the dataset:

overall: Rating given by the reviewer (float64).

verified: Whether the review is from a verified purchase (boolean).

reviewTime: The date of the review (object/string).

reviewerID: Identifier for the reviewer (object/string).

asin: Amazon Standard Identification Number for the product (object/string).

style: Additional attributes of the product (e.g., size) (object, partially null).

reviewerName: Name of the reviewer (object/string).

reviewText: Text of the review (object/string, occasionally null).

summary: Summary of the review (object/string, occasionally null).

unixReviewTime: UNIX timestamp of the review (int64).

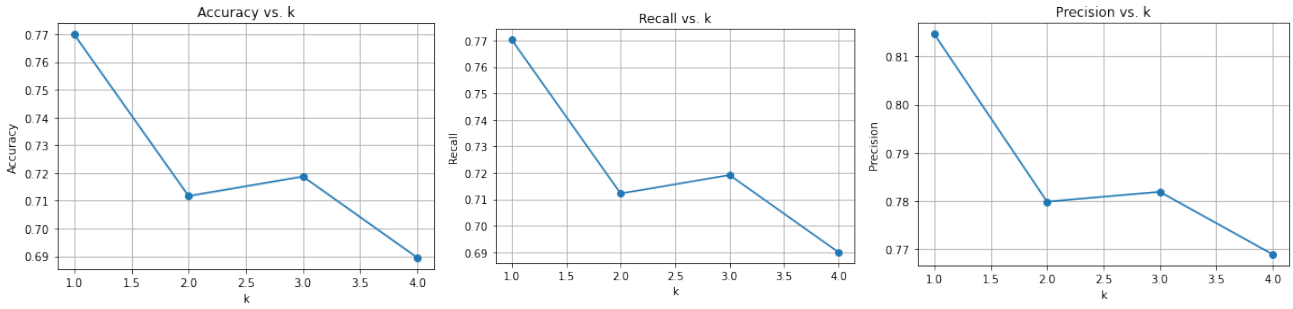
vote: Number of helpful votes (object/string, largely null).

image: Link to images included in the review (object/string, largely null).

Results

The performance of the machine learning models varied:

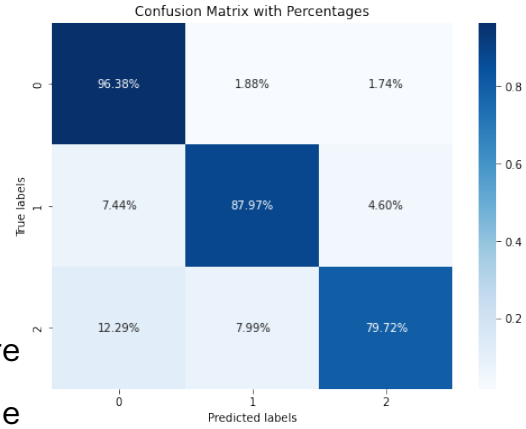
K-Nearest Neighbors: Provided baseline results with moderate accuracy. The images below summarize the results:



As we can see the optimal K even after cross-validation is 1. Since the choice of words between users is different and categorical textual data was made noise free by resampling optimal k was 1 with accuracy being 77%. The accuracy drops with increasing K since with increasing neighbors word choice varies and the model fails to generalize the problem.

Decision Trees : Showed better accuracy with little signs of overfitting.

```
:              precision    recall  f1-score   support

           0       0.83      0.96      0.89      5638
           1       0.90      0.88      0.89      5527
           2       0.93      0.80      0.86      5517

    accuracy                           0.88     16682
   macro avg       0.89      0.88      0.88     16682
weighted avg       0.89      0.88      0.88     16682
```

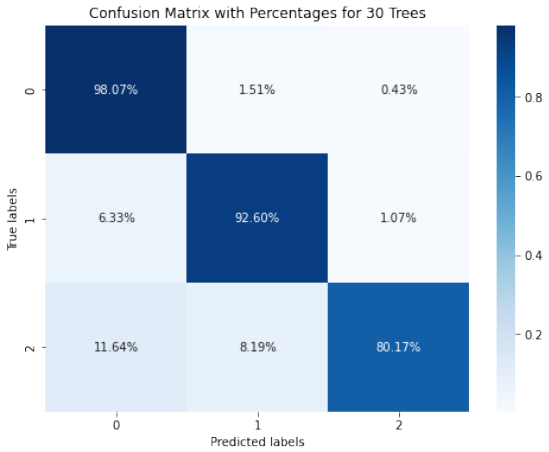


The accuracy of 88% and decent f1-score shows that DecisionTreesClassifier with the

entropy splitting criteria performs decently well. The handling of non-linear relationships, flexibility with categorical data and the robustness to outliers is the reason to choose this classifier. In the confusion matrix shown above, we can see due to robustness to the outliers the decision trees perform well on the resampled data 0(negative) and 1(neutral) making less FP, FN while giving an accuracy around 80% on the positive sentiment scores.

Random Forest: Showed improved accuracy and were particularly effective in handling the feature-rich dataset of text reviews.

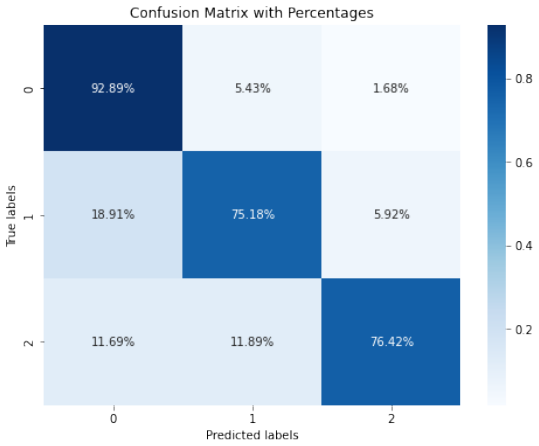
```
Classification Report:              precision    recall  f1-score   su
pport

           0       0.85      0.98      0.91      5638
           1       0.91      0.93      0.92      5527
           2       0.98      0.80      0.88      5517

    accuracy                           0.90     16682
   macro avg       0.91      0.90      0.90     16682
weighted avg       0.91      0.90      0.90     16682

--------------------------------------------------
```



The ensemble RandomForestsClassifier with 30 trees produced the best results. The accuracy of 90% with the confusion matrix behaving similarly as the Decision Trees Classifier. The 2% increase in accuracy is due to the reduction in overfitting since each tree in the forest sees only a subset of the data and features, which ensures that the trees are de-correlated and their errors cancel each other out.. With increasing the trees, we can increase accuracy better but could not be done in this work due to the memory constraints of the computer. As the number of trees increases, the probability of encountering a significant shift in the output due to a different sample or a new tree decreases. Given these facts, sometimes the performance depends on the robustness of the dataset.

Logistic Regression: Acted as a robust baseline for binary sentiment classification.

:	precision	recall	f1-score	support
0	0.76	0.93	0.83	5638
1	0.81	0.75	0.78	5527
2	0.91	0.76	0.83	5517
accuracy			0.82	16682
macro avg	0.83	0.81	0.81	16682
weighted avg	0.83	0.82	0.81	16682



This model performs moderately with 82% accuracy. The confusion matrix in terms of percentage distribution is the same with positive sentiments being around 76% but the model also classifies the neutral sentiments incorrectly while predicting the negative sentiments pretty well. Since Logistic regression models the probability of the dependent variable as a linear combination of the independent variables, it could be a limitation if the relationship between features (e.g., word frequencies) and the sentiment is complex and non-linear.

Discussion

The analysis thus leads to the assertion that ensemble methods bear the potential an increase predictive accuracy during sentiment analysis. The research highlights the need for more advanced preprocessing and suggests ways to help integrate user feedback into strategic decision-making processes.

The dataset was biased towards positive reviews more needing to add synthetic sampled examples to remove bias. The performance could vary with a perfect dataset and potentially model selection would vary.

Stopwords selection is extremely crucial leading to misinterpreted results. The stopwords I curated do a decent job and remove unnecessary words as seen in n-gram analysis. The stemming vs lemmatization is an interesting discussion but to avoid non-existing words I use lemmatization in my work.

References

He Huang, Hang Su, Seungmin Rho, and Young-Sik Jeong. 2020. Learning Semantics-Preserving Attention-Based LSTMs for Sentence Embedding. Electronics, 9(3):483. MDPI. Available online: https://www.mdpi.com/2079-9292/9/3/483

Bittlingmayer. Dataset: Amazon Reviews for Sentiment Analysis. Kaggle. Available online: https://www.kaggle.com/datasets/bittlingmayer/amazonreviews

Jianmo Ni, Jiacheng Li, and Julian McAuley. Amazon Product Data. Available online: https://nijianmo.github.io/amazon/

Imbalanced-learn developers. BorderlineSMOTE. In: imbalanced-learn documentation. Available online: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.BorderlineSMOTE.html

scikit-learn developers. LabelEncoder. In: scikit-learn documentation. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

NLTK Project. nltk.stem.wordnet. Available online: https://www.nltk.org/_modules/nltk/stem/wordnet.html

Sebastian Leier. NLTK's list of english stopwords. Gist GitHub. Available online: https://gist.github.com/sebleier/554280

Appendix

Code Structure and Dependencies

The script makes use of the following Python libraries:

pandas for data manipulation and analysis.

numpy for numerical operations.

gzip for reading gzip-compressed files.

json for parsing JSON objects.

matplotlib and seaborn for data visualization.

sklearn for machine learning model training, testing, and metrics calculation.

nltk for natural language processing tasks.

re for regular expression operations.

string for standard string operations

wordcloud and textblob for text analysis and processing.

imblearn for handling imbalanced datasets.

Function: parse(path)

Purpose: To parse a gzip compressed file that contains JSON objectsInputs:

path (str): The file path to the gzip-compressed file.

Outputs: Yields JSON objects from the file.

Function: getDF(path)

Purpose: To construct a pandas DataFrame

Inputs: path (str): The file path to the gzip-compressed file.

Outputs: DataFrame: A pandas DataFrame

Function: clean(text)

Purpose: To preprocess and clean textual data, typically by removing special characters, punctuation, and other unnecessary elements using regular expressions and string operations.

Inputs: text (str): Text data to be cleaned.

Outputs: str: The cleaned text.

Function: generate_ngrams(text, n)

Purpose: To generate n-grams from the cleaned texs.

Inputs: text (str): Text from which to generate n-grams.

n (int): The number of items in each n-gram.

Outputs: list: A list of n-grams generated from the text.

Function: plot_horizontal_chart(data)

Purpose: To create horizontal bar charts, often used to visualize the distribution of data or compare categories

Inputs: data: Data to be visualized

Outputs: None: Displays a horizontal bar chart.

Function: calculate_sentiment(text)

Purpose: To calculate the sentiment of given text.

Inputs: text (str): Text for which to calculate sentiment.

Outputs: float: Sentiment score indicating the polarity of the text.

str: Sentiment label (positive, neutral, negative) based on the score.

User Manual to Run the code

For running the demo file, it can be done on the terminal . These are the dependencies:

pip install numpy

pip install nltk

nltk.download('wordnet')

nltk.download('stopwords')

pip install scikit-learn

pip install imbalanced-learn

Use python demo.py on the terminal

For running the unittest.py file, it is designed to be run on a terminal.

These are the dependencies:

pip install pandas

Run python unittests.py

For running the Jupyter notebook please see these packages are installed:

pip install pandas numpy nltk wordcloud textblob scikit-learn matplotlib seaborn imbalanced-learn