Natural Language Processing (NLP) & Recommender System

Project Report - Sentiment Analysis Model

Group 5:

Manipal Sidhu

Mahpara Rafia Radmy

Ronald Saenz Huerta

Kanishka Dhir

Prepared for:

Professor Mayy Habayeb

Table of contents

1. Phase 1 4

1.1. Data Exploration 4

1.1.1. Dataset Info 4

1.1.2. Descriptive Statistics 5

1.1.3. Correlation Analysis 5

1.1.4. Distribution Of Ratings and Reviews 6

1.1.5. Conclusion 7

1.2. Dataset Pre-processing 7

1.2.1. Basic Dataset Pre-processing 8

1.2.2. Dataset Pre-processing for TextBlob model 10

1.2.3. Dataset Pre-processing for VADER model 12

1.3. Models 12

1.3.1. TextBlob 12

1.3.2. Valence Aware Dictionary and Sentiment Reasoner (VADER) 15

1.4. Testing results summary 17

1.4.1. TextBlob 17

1.4.2. Vader 17

1.4.3. Model Comparison 18

1.5. Final Conclusion 19

2. Phase 2 20

2.1. Machine Learning Approach 20

2.2. Data Pre-processing: 21

2.2.1. Basic Pre-processing: 21

2.2.2. Data Distribution: 21

2.2.3. TextBlob: 21

2.2.4. VADER: 22

2.2.5. Machine Learning: 22

2.3. Machine Learning Models Declaration 27

2.3.1. Logistic Regression 27

2.3.2. Support Vector Machine (SVM) 27
2.3.3. Naive Bayes 27
2.3.4. Gradient Boosting 27
2.4. Machine Learning Models Results 28
2.4.1. Logistic Regression: 28
2.4.2. Support Vector Machine (SVM): 28
2.4.3. Naive Bayes: 29
2.4.4. Gradient Boosting: 29
2.4.5. Model Results 29
2.5. Machine Learning Models vs Lexicon Models 31
2.6. Research Paper- Recommender systems based on user reviews: the state of the art 32
2.6.1. Pseudo Code: 32
2.6.2. Diagram: 33
2.7. Final Conclusion 36
3. References 37
4. Appendix 1: Project plan 38

1. Phase 1

1.1. Data Exploration

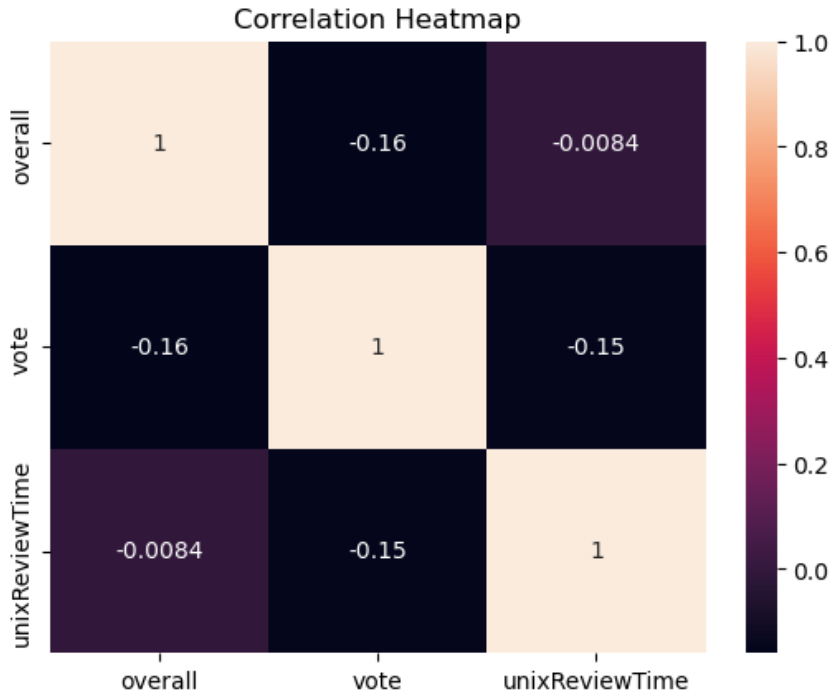
1.1.1. Dataset Info

The dataset under consideration comprises customer reviews for gift cards. It encapsulates a variety of information pertaining to reviews, reviewers, and products. It encompasses 2,972 reviews for 148 unique gift card products and provides a comprehensive insight into customer sentiments and opinions. Each review was contributed by one of the 458 distinct reviewers. On average, products have received a high rating of approximately 4.89 with 2,838 reviews being verified and each review fetching around 5.16 votes. The dataset is organized across 12 different columns, each serving a specific purpose. The “overall” column reflects the numerical rating given to a product, while “verified” indicates whether a review is authenticated. The dataset also records textual details with columns like “reviewerID” capturing the unique identifier of a reviewer, and “asin” pinpointing the specific gift card product. Other descriptors such as “reviewerName”, “reviewText”, “summary”, “image”, “style”, and “reviewTime” further enrich the dataset by offering nuanced insights into the reviewer’s identity, sentiments, and the timing of the review. Lastly, the “unixReviewTime” column provides a timestamp for each review, and the “vote” column quantifies the popularity or impact of a review through the number of votes it has garnered.

1.1.2. Descriptive Statistics

The descriptive statistics shed light on the dataset's attributes. With 2,972 reviews captured, the average rating sits at a commendable 4.88 on a 5-point scale. Reviews span across various timestamps, with "unixReviewTime" falling between 1.33e+09 and 1.53e+09. Notably, out of all reviews, 208 have been actively engaged with, averaging 5.16 votes each.

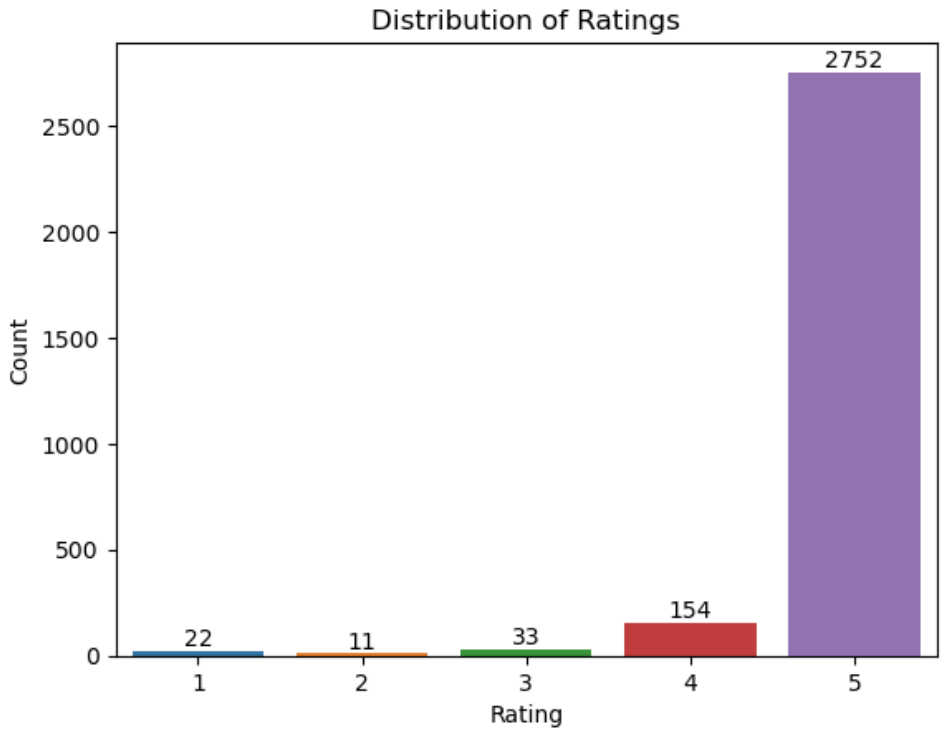
1.1.3. Correlation Analysis



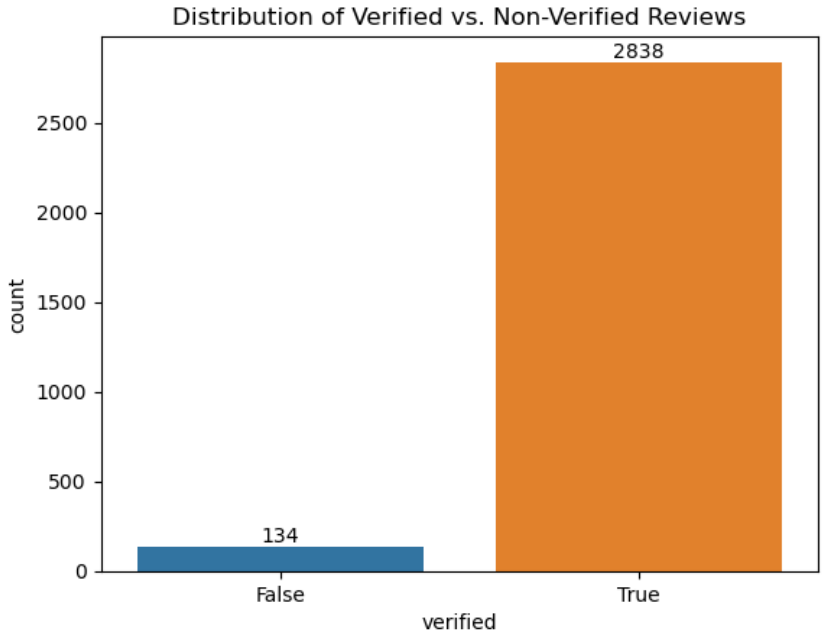
The heatmap shows the correlation between three columns (overall, vote, unixReviewTime). Specifically, there's a mild negative association between 'overall' ratings and 'vote', denoted by a -0.16 correlation. Furthermore, the connection between 'overall' ratings and 'unixReviewTime' is very slight almost negligible at -0.0084. Lastly, 'vote' and 'unixReviewTime'

also exhibit a negative relationship with a correlation of -0.15, hinting that as the review time progresses, there might be a slight decrease in the votes.

1.1.4. Distribution Of Ratings and Reviews



The graph shows how people rated from 1 to 5. Most people, about 93% (2,752 out of 2,972), gave a 5-star rating, showing they were very happy. Only about 0.74% (22 out of 2,972) gave a 1-star rating, 0.37% (11 out of 2,972) gave a 2-star, and 1.11% (33 out of 2,972) gave a 3-star rating. A slightly larger group, about 5.18% (154 out of 2,972), gave a 4-star rating. So, the majority had a great experience. The average rating or score in the "overall" column is approximately 4.89.



The graph displays the distribution of verified versus non-verified reviews. About 95.5% (2,838 out of 2,972) of the reviews are verified, showing a high level of authenticity or validation in the review process. The remaining 4.5% (134 out of 2,972) of the reviews are not verified.

1.1.5. Conclusion

The data exploration reveals a predominantly positive sentiment from the reviewers. The combination of a high average rating with the substantial proportion of verified reviews suggests that the feedback is both positive and credible. The product, service, or content under review seems to be well-received by its users.

1.2. Dataset Pre-processing

The data pre-processing is a crucial step in NLP projects to clean, transform, and organize the data with the goal to prepare for further analysis. The dataset is related to reviews from

Amazon Gift Cards which were written by users with their personal opinions. However, the dataset could contain noise in the form of special characters, blank spaces, digits, emoticons, emojis, URLs, and other irrelevant elements, which can impact the performance of NLP, Machine Learning, or AI models. Therefore, it is necessary to conduct comprehensive dataset preprocessing to prepare the data for modeling.

1.2.1. Basic Dataset Pre-processing

Firstly, it is necessary to make a basic Dataset Pre-processing to clean, transform, and organize the data. This step is crucial for modeling because it is necessary to decide what columns/features should be dropped, modified, merged, or added.

For this project, there are some steps to perform the dataset before modeling:

1. Removed all non-verified records

   For this step, it was necessary to check the original dataset which contains a column called “verified” which is related to the authenticity of the user review. It could help to determine if the user is a real person or a robot. In the data exploration, the distribution of Verified and Non-Verified Reviews shows 134 reviews that are not verified.

2. Dropped unwanted columns

   To perform the sentiment analysis processing, it is necessary to drop unnecessary columns that do not contribute anything. After to analyze the dataset,

there are some columns that should be dropped such as 'verified', 'reviewerID', 'asin', 'reviewerName', 'reviewTime', 'style', 'unixReviewTime', 'vote', and 'image'.

3. Dropped duplicates

    To maintain data integrity, it is necessary to eliminate duplicate records. There are 892 duplicated records that should be eliminated from the dataset.

4. Labeled the data based on the value of “rating of the product”

    After to check the original dataset, it contains a column called “overall” which is associated to the rating of the product. It is a form to categorize each review using the rating. Consequently, it was necessary to assign a sentiment analysis, using the following logic: ratings 4 and 5 as “Positive”, rating 3 as “Neutral”, and ratings 1, and 2 as “Negative”

5. Merge both columns “reviewText” and “summary” into a new column called “text”

    After to check the original dataset, it contains two columns called “reviewText” and “summary” that have text related to the user review. The consolidation of both columns in a new column called “text” help to simplified subsequent text analysis processes, fostering a more streamlined analytical workflow.

1.2.2. Dataset Pre-processing for TextBlob model

For TextBlob model, it was necessary to make some dataset pre-processing to perform the dataset.

For this project, there are some steps to perform the dataset before modeling:

1. Exclusion of Special Characters and Digits

   This step is necessary to streamline the data, ensuring that the model focused only in the linguistic that could carry sentiment. Numbers or special characters could be irrelevant and cause some noise for modeling.

2. Trimming trailing whitespaces

   To avoid some noise into the data, it is necessary to trim trailing whitespaces. This step is necessary to maintain consistency.

3. Punctuation removal

   To avoid some noise into the data, it is necessary to remove punctuation. The punctuation could sometimes distort the meaning of the sentence, and this step fosters a cleaner analysis. For that reason, this step is crucial to improve the accuracy of the sentiment analysis.

4. URL elimination

Web links do not contribute to sentiment analysis, for that reason it is necessary to remove from the text with the goal to improve our data for modeling.

5. Removal of Stop Words

In sentiment analysis, some words such as “a”, “the”, and “is” have limited semantic meaning and could be disregarder. For that reason, is necessary to eliminate common and non-informative words know such as “stop words”.

6. Expanding contractions

To perform modeling some expansions of the contractions should be used to enable the understanding of the complete word. With expanding contractions, it could ensure that words like “can’t” were transformed into “cannot”.

7. Tokenization

To make it more amenable for analysis, it was necessary to use tokenization which is crucial to breaks down text into its basic elements.

8. Rejoining Tokens

Post tokenization, the individual token should be reassembled into coherent text.

After to make data pre-processing, the text is ready to sentiment analysis. Lemmatization and lowercase are not necessary to apply because TextBlob was embedded those steps.

1.2.3. Dataset Pre-processing for VADER model

For VADER model, it is not necessary to do data pre-processing because it is specifically designed for sentiment analysis for social media posts. It is well-suited for processing short and informal textual data. It can effectively analyze the sentiment intensity and polarity of the text. VADER provides a proper handling of sentences with:

- Typical negations
- Use of contractions as negations
- Use of punctuation as signal of increment of the sentiment intensity
- Use of word-shape (ALL CAPS) as signal of emphasis
- Use of degree modifiers as alteration of the sentiment intensity
- Use of the slang words
- Use of emoticons and emojis
- Use of the initialisms and acronyms.

For that reason, the pre-processing was not required.

1.3. Models

1.3.1. TextBlob

TextBlob is a Python library for basic natural language processing (NLP) tasks. It offers a simple API for common tasks like tokenization, part-of-speech tagging, noun phrase extraction,

and sentiment analysis. TextBlob Built on the NLTK and another package called Pattern, TextBlob provides an easy-to-use interface in NLP.

1.3.1.1. Assumptions/Heuristics/algorithms used

The TextBlob by default uses the Pattern library for its sentiment analysis. Pattern's sentiment analysis is built on large dataset annotated with polarity and subjectivity. Sentiment of a text is calculated based on the words it contains and their respective subjectivity and polarity.

Alternate to pattern we can use Naive Bayes classifier trained using NLTK on a movie review corpus. This method uses the probabilities of observing specific words given their sentiment labels.

We assume that polarity of text in range of –0.2 to +0.2 will have sentiment neutral whereas polarity of text greater than +0.2 will have positive and less than –0.2 will have negative sentiment.

1.3.1.2. How it works

TextBlob is a Python library for processing textual data, and it provides a simple API for diving into common natural language processing (NLP) tasks.

TextBlob first Tokenize and preprocesses data (removing stop words, lowercasing etc.), and then TextBlob can tag each token with its corresponding Part-Of-Speech like noun, verb, etc. We can also call this as a feature extraction in which we convert the tokenized words into features that model can understand.

The sentiment property of the api/library returns polarity and subjectivity.

Polarity ranges from –1.0 to +1.0 while subjectivity ranges from 0 to +1.0. Polarity measures the emotion. Where +1.0 refers to positive and –1.0 refers to negative. While subjectivity refers to opinion or views which needs to analyze in given context where 0 is very objective and +1.0 is very subjective. A subjective instance may or may not carry any emotion.

1.3.1.3. External Datasets

TextBlob relies on external datasets and resources for various functionalities like:

- **Sentiment Analysis->** TextBlob uses the '**Pattern'** library for sentiment analysis which comes with a built-in sentiment lexicon. This lexicon is used to determine polarity (positivity/negativity) and subjectivity of a text. It is not exactly a dataset but rather a collection of words and their associated sentiment scores.
- **POS Tagging and Noun Phrase Extraction->** For these functionalities, TextBlob leverages corpora and trained models from the Natural Language Toolkit (NLTK). Specifically, it typically uses the Penn Treebank dataset for POS tagging.
- **Tokenization->** TextBlob uses NLTK's tokenization methods, while not directly relying on a specific dataset but have been informed and refined by numerous corpora.
- **Translation and Language Detection->** TextBlob offloads these tasks to the Google Translate API. This is not directly about an external dataset, but it is worth noting since the translation capability relies on an external service.

1.3.2. Valence Aware Dictionary and Sentiment Reasoner (VADER)

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon-based sentiment analysis tool that uses a pre-built dictionary of words and their associated sentiment scores to determine the sentiment of a given piece of text. Its lexicon is specifically tuned to handle informal language and features such as slang, emoticons, and capitalization commonly used in social media text, news articles, blogs etc.

1.3.2.1. Assumptions/Heuristics/algorithms used

We have Assume that valance score of a text ranging from –0.2 to +0.2 will be count as a neutral sentiment. Valance score Greater than +0.2 and less than –0.2, we will count as a positive and negative sentiment respectively.

1.3.2.2. How it works

VADER relies on a predefined dictionary (or lexicon) that maps words and other numerous lexical features common to sentiment expression in microblogs.

These features include:

- A full list of Western-style emoticons (ex - :D and :P)
- Sentiment-related acronyms (ex- LOL and ROFL)
- Commonly used slang with sentiment value (ex- Nah and meh)

The valence scores in VADER's lexicon range from -4 (most negative) to +4 (most positive). The sentiment score of text is calculated as a sum of intensity score of words in the text. Words that are neutral or have no clear sentiment typically have a score close to 0.

To calculate the composite sentiment score for an entire piece of text, VADER doesn't just sum up the valence scores of individual words. Instead, it incorporates various heuristics and rules, including:

- Adjusting scores for booster words (e.g., "very" or "extremely") that can amplify the sentiment of a neighboring word.
- Handling negations that can reverse the sentiment (e.g., "not good" is negative despite the word "good" being positive).
- Accounting for the effects of punctuation, capitalization, and other linguistic cues.

After processing the text and applying these rules, VADER produces a compound sentiment score that ranges from -1 (most negative) to +1 (most positive). This score offers a holistic view of the text's overall sentiment. In addition to the compound score, VADER also provides individual scores for the positive, neutral, and negative sentiments present in the text.

1.3.2.3. External Datasets

VADER has its own dataset called vader_lexicon.txt which is validated by multiple independent human judges. VADER incorporates a "gold-standard" sentiment lexicon that is especially attuned to microblog-like contexts.

Other than it, VADER also uses external datasets like

- nytEditorialSnippets_GroundTruth.txt
- nytEditorialSnippets_anonDataRatings.txt
- movieReviewSnippets_GroundTruth.txt
- movieReviewSnippets_anonDataRatings.txt
- amazonReviewSnippets_GroundTruth.txt
- amazonReviewSnippets_anonDataRatings.txt
- tweets_GroundTruth.txt,tweets_anonDataRatings.txt

1.4. Testing results summary

The summary analysis based on the testing outcomes of the two approaches—Vader and Textblob that we used to develop our model is presented below.

1.4.1. TextBlob

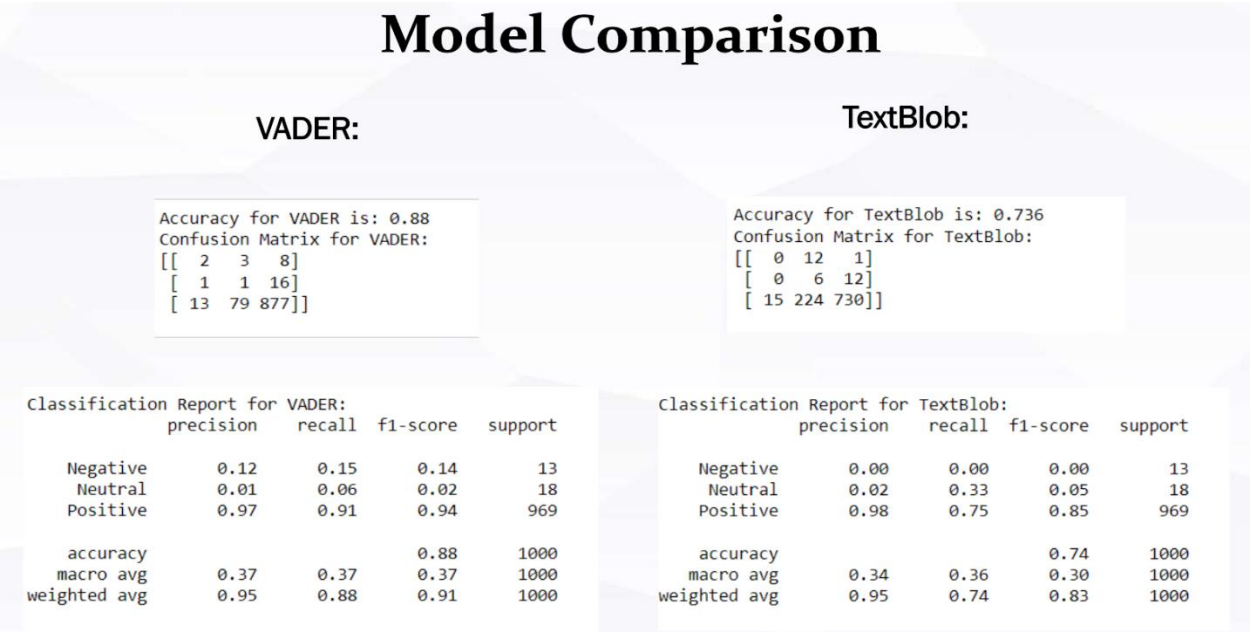
The TextBlob Model had an accuracy of about 73%, with a precision of 98% for positive sentiment, 2% for neutral sentiment and 0% for negative sentiment. With a weighted average of 95%.

1.4.2. Vader

The VADER model had an accuracy of about 88%, with precision of 97% for positive sentiment, 1% for neutral sentiment and 12% for negative sentiment. With a weighted average of 95%

1.4.3. Model Comparison

Overall, the Vader model performed the best out of two models, with the highest accuracy and f1-score for positive sentiment. The Textblob model does not perform as well as the Vader model, but it still has decent accuracy. One thing we should note is that both models have identical weighted avg score for precision of about 95%.



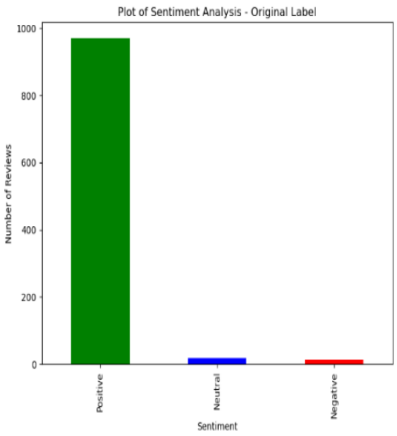
In the Original Label Rating, a vast majority of reviews, 969 to be precise and categorized as positive with a minimal 18 as neutral and 13 as negative. The TextBlob Lexicon Model has identified 743 reviews as positive, 242 as neutral and 15 as negative. Comparatively, the VADER Lexicon Model labels 901 reviews as positive, 83 as neutral and 16 as negative. Both the TextBlob and VADER models depict a higher number of neutral reviews compared to the Original Label.

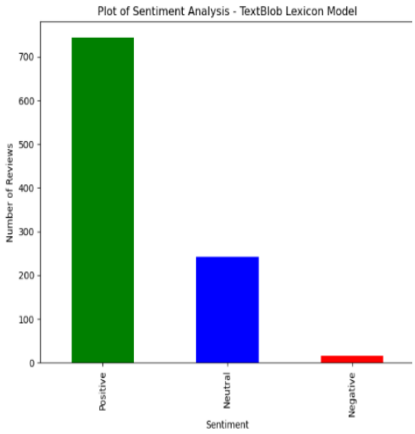
Sentiment Analysis

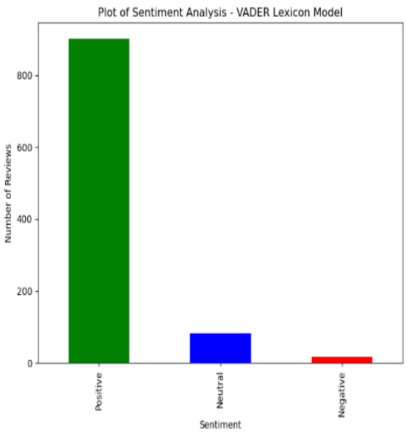
Original Label Rating:
Positive: 969 reviews
Neutral: 18 reviews
Negative : 13 reviews

TextBlob Label Rating:
Positive: 743 reviews
Neutral: 242 reviews
Negative : 15 reviews

VADER Label Rating:
Positive: 901 reviews
Neutral: 83 reviews
Negative : 16 reviews







1.5. Final Conclusion

The data exploration reveals a predominantly positive sentiment from the reviewers. The combination of a high average rating with the substantial proportion of verified reviews suggests that the feedback is both positive and credible. The product, service, or content under review seems to be well-received by its users.

In data preprocessing we include steps regarding handling missing data, removing duplicates, removing stop-words, combining review text and summary etc. for textblob, whereas we only use clean text which is combination of review text and summary for VADER as it is. We dropped columns which are not useful for sentiment analysis like Style, vote, image, and reviewerName, etc.

Then we perform sentiment analysis using textblob and VADER by categorizing reviews into positive, negative, and neutral categories based on their ratings.

When comparing the sentiment analysis capabilities of VADER and TextBlob, VADER clearly emerges as the superior model. It consistently outperforms TextBlob across key metrics, particularly in accuracy and in identifying positive sentiments. While both models have their merits, for tasks demanding higher precision and recall, VADER appears to be the more reliable choice.

2. Phase 2

2.1. Machine Learning Approach

In phase #2 of our project, we focused on implementing and evaluating machine learning models for sentiment analysis on gift card reviews. Firstly, we prepared the remaining dataset by cleaning the text, labeling based on ratings. Then, we split it into 200 rows for validation set i.e. comparing with lexicon approach, and the remaining will be used for training and testing sets. Then, we split it into training and testing sets, using 70% for training and 30% for testing with stratified sampling. Our approach involved preprocessing the text data which included removing HTML tags, converting text to lowercase, removing special characters and stopwords. We then trained and evaluated several machine learning models, such as Logistic Regression, SVM, Naive Bayes and Gradient Boosting, using metrics like accuracy, precision, recall and F1 score to select the best model for predicting sentiment in the test set.

2.2. Data Pre-processing:

2.2.1. Basic Pre-processing:

We have implemented the same data pre-processing steps as in phase #1 (Refer to section 2.1)

2.2.2. Data Distribution:

2.2.2.1. Lexicon Approach Sampling:

After Basic pre-processing, out of 1,946, we selected a subset of 200 reviews for comparison with Lexicon-based Sentiment Analysis.

2.2.2.2. ML Model Preparation

Reserved remaining 1,746 reviews for Machine Learning models which has been further split into training set 70% of 1,746 reviews, focusing on model training and optimization and testing set of remaining 30% of the data, used to evaluate model performance.

There are the Final Dataset Shapes:

- Training Data (ML) = Reviews (X): 1,222 samples & Overall Rating (y): 1,222 samples
- Testing Data (ML) = Reviews (X): 524 samples & Overall Rating (y): 524 samples

2.2.3. TextBlob:

For TextBlob, we have kept the same preprocessing steps consistent with phase #1 (Refer to section 2.2). However, we have now retained negative stop words which we didn't include in phase #1.

2.2.4. VADER:

In VADER, we have kept almost the same preprocessing steps as phase #1 (Refer to Section 1.2.3). However, in phase #2 we made an additional adjustment by removing URLs. This step was necessary because VADER does not automatically handle URL removal as part of its built-in preprocessing.

2.2.5. Machine Learning:

In the context of preparing text data for sentiment analysis using a machine learning model, it is crucial to process the dataset thoroughly to enhance model performance and accuracy. Different from the TextBlob & VADER model approach, this ML model requires a series of specific pre-processing steps to refine the dataset.

There are the pre-processing steps used:

- URL Removal

URLs often contain no sentiment value and can introduce noise into the data. By eliminating web links, the focus is directed towards the textual content that holds potential sentiment.

- Special Character and Digit Exclusion

Special characters and digits are typically irrelevant for natural language processing tasks and can detract from the model's ability to learn from the text content.

- Whitespace Trimming

Trailing whitespaces can create inconsistencies in the data. Removing these ensures a cleaner, more uniform dataset & helping the model to process text more effectively.

- Lowercasing

Converting all text to lowercase helps maintain consistency in the data. This step ensures that the model does not differentiate between words based on their case.

- Punctuation Removal

Punctuations can sometimes distort the meaning of sentences or add unnecessary complexity. Removing them fosters a more straightforward analysis of the text.

- Stop Word Removal with Custom Adjustments

Standard stop words (like "the", "is", and "in") are usually removed as they typically do not contribute to sentiment. However, this model retains certain negative contractions and negations (like "don't", "isn't", "no") which are crucial for sentiment analysis.

- Contraction Expansion

Expanding contractions (e.g., transforming "can't" into "cannot") ensures that the model comprehends the full meaning of words, which is vital for accurate sentiment analysis.

- Tokenization

Breaking down the text into individual words or tokens is essential for analysis. It helps the model to understand and process each word separately.

- Lemmatization

This step involves converting words to their base or root form. Unlike stemming, lemmatization takes into consideration the context of words, thus enhancing the quality of the processed data.

- Rejoining Tokens

After processing, lemmatized tokens are reassembled into coherent text. This step is crucial for ensuring that the text remains understandable and retains its original meaning, albeit in a processed form.

- Feature Extraction

We then apply Term Frequency-Inverse Document Frequency (TF-IDF) for feature extraction, converting text data into a numerical format suitable for machine learning models. The reason to use TF-IDF is because TF-IDF considers the frequency of words in a document against their occurrence across the entire corpus. This approach highlights words that are frequent in a document but not common across all documents, thereby emphasizing the unique content of each document. It also helps to reduce the dimensionality of the feature space and due to its effectiveness in capturing document content while filtering out noise, it is widely accepted.

- Balancing the Dataset

Considering the imbalance in our dataset, we employ oversampling techniques to increase the representation of minority classes. This approach helps in improving the model's performance and generalizability. Before oversample the labels:

- Initial Class Distribution

Initially, our dataset exhibited a severe imbalance. The distribution of the classes was as follows:

- Positive: 1188 samples
- Neutral: 19 samples
- Negative: 15 samples

This disproportion heavily skewed towards the ‘Positive’ class which could potentially lead to a model biased towards predicting this class while underperforming on ‘Neutral’ and ‘Negative’ classes.

- Application of Random Over Sampling

To rectify this imbalance, we employed the RandomOverSampler technique. This method works by randomly duplicating instances from the minority classes until all classes have an equal number of instances. After applying RandomOverSampler, the class distribution was balanced:

- Positive: 1188 samples
- Neutral: 1188 samples
- Negative: 1188 samples

This balanced distribution ensures that each class has equal representation in our dataset & allowing for a more equitable training process for our machine learning models.

Figures 6.1 and 6.2 in the report visually represent the class distributions before and after the application of RandomOverSampler, respectively.

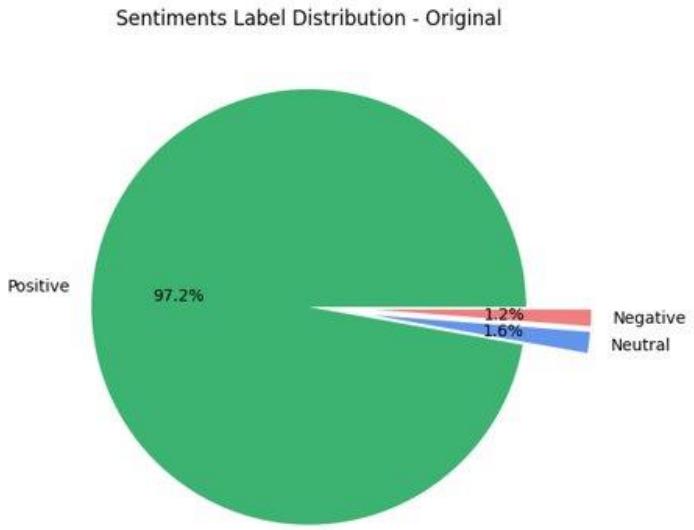


Figure 6.1 - Sentiments Label Distribution before Up-Sampling

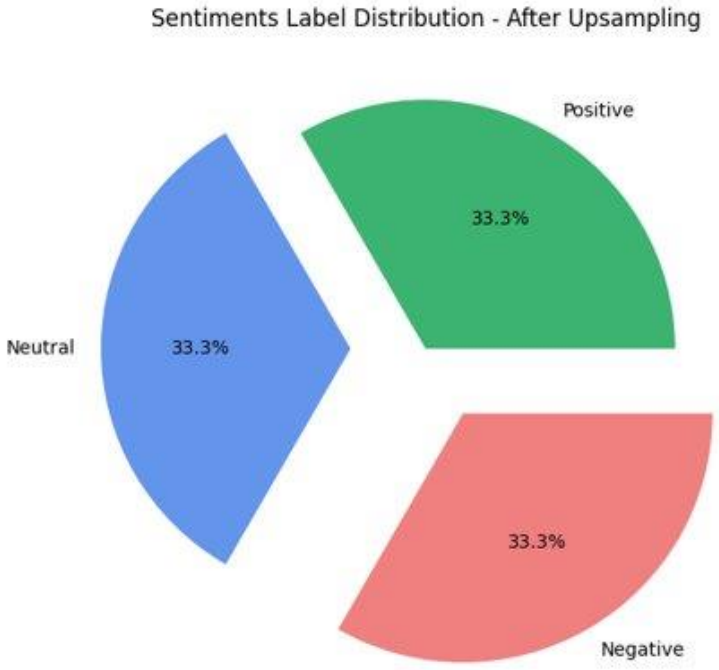


Figure 6.2 - Sentiments Label Distribution Up-Sampling

2.3. Machine Learning Models Declaration

2.3.1. Logistic Regression

The model used is LogisticRegression, which is particularly designed to predict the probability of a binary outcome. It has been set up with a key parameter **class_weight="balanced"**, which is used to handle imbalanced datasets by adjusting weights inversely proportional to class frequencies. The **max_iter** parameter is set to 500, defining the maximum number of iterations taken for the solvers to converge.

2.3.2. Support Vector Machine (SVM)

The SVM model is implemented using the SVC class, a popular choice for classification tasks. Similar to Logistic Regression, it has **class_weight="balanced"** to address class imbalance. It has a higher **max_iter** value of 10,000, allowing more iterations for the model to find the optimal margin between classes.

2.3.3. Naive Bayes

This model employs the MultinomialNB algorithm, which is effective for classification with discrete features like word counts for text classification. It uses the default parameters which are generally a good starting point for baseline modeling.

2.3.4. Gradient Boosting

The Gradient Boosting model utilizes the GradientBoostingClassifier, known for combining multiple weak learning models to create a strong predictive model. It is configured with default parameters which often yield a competitive model without extensive parameter tuning.

2.4. Machine Learning Models Results

We trained four different types of machine learning models to understand people's feelings from their gift card reviews. These models were Logistic Regression, Support Vector Machine (SVM), Naive Bayes, and Gradient Boosting. We have used 70% of our data to train and 30% on testing these models.

2.4.1. Logistic Regression:

The model did well in the training, with an accuracy of about 99.86%. When we tested it, it scored 98.28% in accuracy. This model also showed the best overall performance with a high testing accuracy of 98.28% and an F1 Score of 98.26%. Its precision and recall are also impressive at 98.30% and 98.28%, respectively. These balanced scores across different classes suggest that Logistic Regression performed consistently well and accurately identifying various sentiments in the reviews.

2.4.2. Support Vector Machine (SVM):

The SVM model was close to Logistic Regression in performance, achieving an accuracy of 99.97% in training, 97.71% in testing and an F1 Score of 97.13%. While its precision was slightly lower at 97.04%, the recall was on par at 97.71%. This indicates a slightly better identification of the Positive class, showing its strength in recognizing more clear-cut sentiments.

2.4.3. Naive Bayes:

Naive Bayes had a lower accuracy of 98.99% in training, 94.66% in testing and an F1 Score of 95.58%, compared to the first two models. This model showed a higher precision of 96.79% but a lower recall of 94.66%. It excelled more with Positive reviews, indicating a tendency to better identify clear positive sentiments but struggled a bit with Neutral reviews, especially in recognizing them accurately.

2.4.4. Gradient Boosting:

Gradient Boosting had an accuracy of 100% in training (which is great, but we need to make sure it's not overfitting the data), 97.33% in testing and an F1 Score of 97.16%, which is lower than Logistic Regression and SVM. The precision and recall were 97.14% and 97.33%, respectively. Like Naive Bayes, this model was more effective in identifying Positive sentiments, but its performance was not as balanced across different classes, with a slight struggle in accurately categorizing Neutral sentiments.

2.4.5. Model Results

This table shows the metrics (accuracy, precision, recall, F1 score, and confusion matrix) of the 4 models using the testing data.

Model	Accuracy	Precision	Recall	F1	Confusion Matrix
Logistic Regression	0.9828	0.9830	0.9828	0.9826	[[5 0 1] [1 4 3] [2 2 506]]
Support Vector Machine (SVM)	0.9771	0.9704	0.9771	0.9713	[[1 0 5] [1 2 5] [0 1 509]]
Naive Bayes	0.9466	0.9679	0.9466	0.9558	[[3 0 3] [1 4 3] [8 13 489]]
Gradient Boosting	0.9733	0.9714	0.9733	0.9716	[[4 0 2] [0 2 6] [4 2 504]]

After analyzing the results for the Machine Learning model, Logistic Regression and SVM are giving better accuracy than others so, we chose Logistic Regression and Support Vector Machine (SVM) models for their strong track record with binary classification tasks like ours. Logistic Regression is advantageous for its ability to handle imbalanced datasets and provide probabilistic outputs, which is crucial for understanding the confidence behind each sentiment prediction. SVM is favored for its robustness in high-dimensional spaces and its capability to handle complex patterns, a common feature of text data. While both models performed well with the TF-IDF representation of our dataset, we recognize that results might vary with different data representations or further model tuning. The choice of these models is a strategic step to

leverage their individual strengths, potentially creating a more accurate and reliable sentiment analysis framework.

2.5. Machine Learning Models vs Lexicon Models

The comparative analysis of the Lexicon Models (TextBlob and Vader) versus the Machine Learning Models (Logistic Regression & Support Vector Machine) reveals insightful distinctions in performance metrics for sentiment analysis. The Lexicon Models, particularly TextBlob, demonstrate lower accuracy (0.685 for TextBlob and 0.885 for Vader) compared to the Machine Learning Models, which exhibit notably higher accuracy (0.990 for Logistic Regression and 0.985 for SVM). This suggests that while Lexicon Models are useful for quick, rule-based sentiment analysis but they may lack the nuanced understanding and adaptability of Machine Learning Models.

Model Type	Model	Accuracy	Precision	Recall	F1	Confusion Matrix
Lexicon Model	TextBlob	0.685	0.9567	0.685	0.7950	[[2 0 0] [0 0 3] [5 55 135]]
Lexicon Model	Vader	0.885	0.9588	0.885	0.9204	[[0 2 0] [0 0 3] [2 16 177]]
Machine Learning Model	Logistic Regression	0.990	0.9901	0.990	0.9887	[[1 0 1] [0 2 1] [0 0 195]]
Machine Learning Model	Support Vector Machine (SVM)	0.985	0.9852	0.985	0.9817	[[1 0 1] [0 1 2] [0 0 195]]

2.6. Research Paper- Recommender systems based on user reviews: the state of the art

Upon meticulous analysis of the research document - “Recommender systems based on user reviews: the state of the art”, we explored various methodologies aimed at enhancing ratings. Section 4.2: Rating profile – inferring ratings from reviews - of the paper mentions that “In this section, we survey approaches that aim to infer a user’s overall preference for a product based on the opinions s/he expresses in the review, which can act as a *virtual rating* (also called an *inferred rating*, *opinion rating*, or *text-based rating*) for a CF system.” (Para 1, page 110). In the subsequent section 4.3.3, it is reiterated that “As mentioned in Sect. 4.2, “virtual ratings are the overall opinions inferred from reviews.” (Para 1, page 116). The study also highlights “*opinion post-filtering*, in which ratings and overall opinions are first used independently to train two prediction models, and then a linear combination of the two models is realized to obtain the final rating prediction” (Para 2, page 116). Notably, during the evaluation of three proposed variations on the Amazon dataset the opinion post-filtering method exhibited superior predictive accuracy, particularly in terms of the Root Mean Squared Error (RMSE) (Para 3, pg 116)

In alignment with this research, we proceeded with the post-filtering method, leveraging our two most performant machine learning models—SVM and Logistic Regression—for sentiment analysis. Here, SVM incorporates the inferred ratings while Logistic Regression considers the actual rating values.

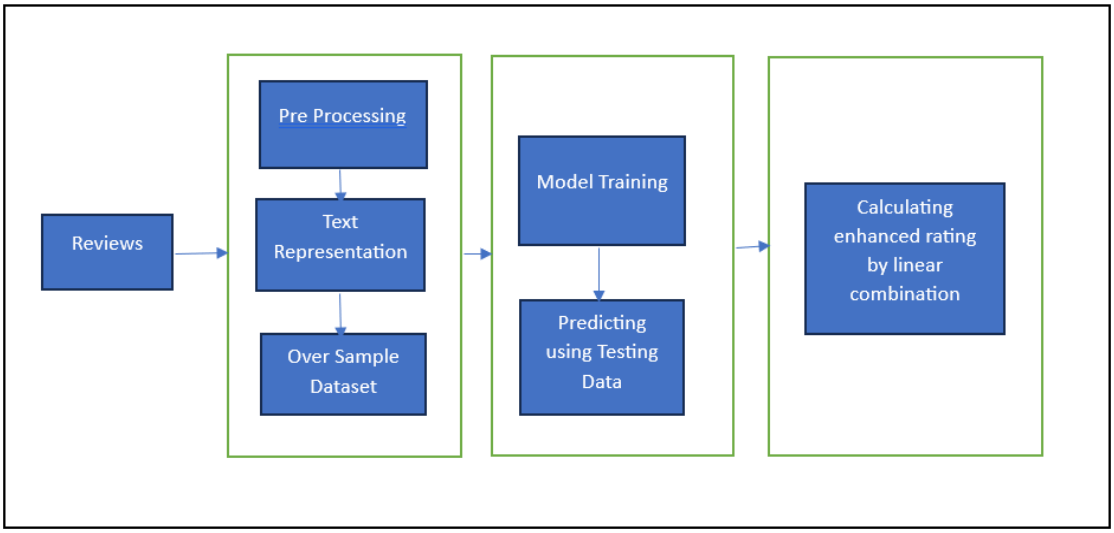
2.6.1. Pseudo Code:

```
# Train SVM and Logistic Regression models

svm_model = SVM()
```

```
svm_model.fit(X_train_resampled, y_train_resampled)

logistic_regression_model = LogisticRegression()

logistic_regression_model.fit(X_train_resampled,
y_train_resampled)

# Calculate inferred ratings

inferred_rating_svm = svm_model.predict(X_test_tfidf)

inferred_rating_lr                                    =
logistic_regression_model.predict(X_test_tfidf)

# Calculate enhanced ratings using linear combination

enhanced_rating  =  (svm_weight  *  inferred_rating_svm)  +
(lr_weight * inferred_rating_lr)

# Evaluate performance with MSE on testing data

mse_sample_df   =   mean_squared_error(sample_df['overall'],
enhanced_rating_sample)
```

2.6.2. Diagram:

This is the diagram that explain the pseudocode and the implementation of the state-of-the-art model to enhance the rating in our dataset.



The method began by utilizing pre-processed data from the 'text_df' dataset, considering the overall rating values as X and Y. The dataset was split into 70% training and 30% testing segments, comprising 1222 and 524 entries, respectively. Text representations in the form of TF-IDF count frequency vectors were created to facilitate model training. To counter class imbalance, we employed RandomOverSampler, elevating the minority class values for ratings 4 through 1 to match the count of rating 5 (1112 instances).

Before upsampling, the rating distribution was as follows:

{5: 1112, 4: 77, 3: 19, 1: 8, 2: 6}

Post-upsampling, the balanced distribution stood at:

{5: 1112, 3: 1112, 4: 1112, 1: 1112, 2: 1112}

Subsequently, we trained SVM and Logistic Regression models using the upsampled overall ratings, achieving an impressive training accuracy of 0.99 for both. The calculation of enhanced ratings involved a weighted formula:

*enhanced_rating = (svm_weight * inferred_rating_svm) + (gb_weight * inferred_rating_gb)*

Through rigorous experimentation with different weight combinations, we found that a weight distribution of 0.4 for SVM and 0.6 for Logistic Regression in the linear combination yielded the best performance. This selection was based on optimizing the mean square error on the testing data, showcasing the effectiveness of a 0.4-0.6 weight allocation for these models. Upon evaluation with 500 reviews from the testing data, the resultant mean squared error (MSE) of 0.2 demonstrated robust predictive performance.

Extending this analysis to predict enhanced ratings for the 200 reviews segregated in the 'sample_df' dataset, we achieved an impressive MSE of 0.15, showcasing the robustness of our model. Notably, as a post-processing step, we rounded off the enhanced ratings to the nearest integer value to maintain ratings within the standardized range of 1 to 5. A comprehensive comparison between the enhanced ratings and the actual ratings is depicted below:

	overall	enhanced_rating	label	enhaced_label	text
2303	5	5	Positive	Positive	Always a Great gift for any occasion Five Stars
2957	4	4	Positive	Positive	Christmas Gift Four Stars
686	5	5	Positive	Positive	I have used these for several years and it is a very classy way to give a gift card! Way to go Amazon! I have used these for several years and it is ...
1501	5	5	Positive	Positive	Nice to have for any gift that might come up like birthdays for grandchildren as a extra. Five Stars
690	3	4	Neutral	Positive	It's okay. Works fine. Some scuffs. Works fine. Some scuffs
2273	5	5	Positive	Positive	This card made a great Christmas gift for my sister-in-law, who is very hard to choose for. A great last-minute gift
1160	5	5	Positive	Positive	Used this as a thank you for a young lady who did some errands for me. She was really surprised and very grateful. Used this as a thank you for a young lady ...
2609	5	5	Positive	Positive	Love amazon lighting deal gift cards! These were small and super cute. Perfect for gifting! Awesome buy!
1656	5	5	Positive	Positive	Not a Starbucks fan but this made for a nice little extra gift for Christmas for the nieces who love it. Not a Starbucks fan but this made for a nice little extra gift for Christmas for the nieces who ...
2813	5	5	Positive	Positive	family will love it Five Stars
2822	5	5	Positive	Positive	This was a great idea for a gift card for those who celebrate Christmas or the "Holidays" Cute Gift Card
99	4	5	Positive	Positive	It was a good gift with a fun card to send it in. I enjoyed sending it to a special child in our family. Fun Card
1746	5	5	Positive	Positive	GRANDDAUGHTER LOVES GOING THERE WITH HER GIRLFRIENDS. Five Stars
569	5	5	Positive	Positive	Attractive gift tin. Reusable.
435	5	5	Positive	Positive	Worked as it should Worked as it should
2232	5	5	Positive	Positive	This was my first food gift card.....This is what started my desire to continue purchasing them. After doing the many focus and surveys online they pay you in a Amazon Gift Code..How easy it is to find the many food locations many people enjoy..I now have started getting them,saving them to give out for ,any occasion . very fast delivery HAPPY GIFT CARDS
2331	5	5	Positive	Positive	FAMILY LOVES EATING OUT. Five Stars
2824	5	5	Positive	Positive	This was a great addition to our gifts this year. Loved the tag look to add to presents. Five Stars
1738	5	5	Positive	Positive	What can you say it's a gift card - Shipping was fast. Buy Me
2791	5	5	Positive	Positive	Thought they were kind of grown up for my grandsons until I opened them. Really cute pop up! Great gifts

2.7. Final Conclusion

The comprehensive analysis and practical implementation of advanced recommender systems demonstrates a significant stride in the domain of sentiment analysis and rating prediction. It was inspired by the pivotal insights from the paper “Recommender systems based on user reviews: the state of the art,” notably advances the methodology of inferring and enhancing user ratings from textual reviews.

By integrating the strengths of both Support Vector Machine (SVM) and Logistic Regression models, we successfully harnessed the nuanced sentiment expressed in user reviews. This integration led to the development of a robust system capable of accurately predicting user preferences and opinions, which are crucial for reliable recommendation systems. The novel approach of using a weighted linear combination of inferred ratings from these models proved to be highly effective, as evidenced by the low mean squared error (MSE) scores obtained during testing.

In conclusion, this underscores the transformative power of machine learning in understanding and predicting user sentiments. The significant reduction in MSE and the high accuracy of our enhanced ratings affirm the efficacy of our approach, paving the way for more sophisticated and user-centric recommender systems in the future.

3. References

- Project, U. C. R. (n.d.). Amazon Review Data (2018). https://nijianmo.github.io/amazon/index.html
- Tutorial: Quickstart - TextBlob 0.16.0 documentation (n.d.). https://textblob.readthedocs.io/en/dev/quickstart.html
- Cjhutto. (n.d.). CJHUTTO/Vadersentiment: Vader sentiment analysis. GitHub. https://github.com/cjhutto/vaderSentiment
- Vader sentiment analysis: A complete guide, algo trading and more. Quantitative Finance & Algo Trading Blog by QuantInsti. https://blog.quantinsti.com/vader-sentiment/
- Todi, M. (2019, September 23). Sentiment analysis using the vader library. Medium. https://medium.com/analytics-vidhya/sentiment-analysis-using-the-vader-library-a91a888e4afd
- Chen, L., Chen, G., & Wang, F. (2015). Recommender systems based on user reviews: the state of the art. User Modeling and User-Adapted Interaction, 25(2), 99–154. https://doi.org/10.1007/s11257-015-9155-5

4. Appendix 1: Project plan

The purpose of this project is to create a sentiment analysis model that can classify customers textual reviews on amazon data as positive, negative, or neutral sentiment. Sentiment analysis is an important part of NLP that enables machines to understand human emotions conveyed in text. It aids businesses in gauging customer feedback, helps in automating responses to user queries, social media monitoring etc.

The goal of this project is to present an in-depth overview of the essential steps involved in creating a sentiment analysis model utilizing the Lexicon method.

- Data exploration is done in the first section. a vital first step in comprehending the information provided. It helps us make sense of the data provided and decides what needs to be done for future tasks.
- The second step is Data preprocessing. This step is necessary so that data is clean, consistent, and ready for further processing.
- Model training and assessment constitute the last step, whereby two lexicon methods (Vader, TextBlob, and Sentiwordnet models) out of three should be chosen.
- Phase 1 ends with the selection of the two best models, which will then be compared to each other based on accuracy and other performance measures.
- In the second phase, we will explore machine learning algorithms for sentiment analysis, including Logistic Regression, SVM, Naïve Bayes, and Gradient Boosting. We will train two chosen models and document their training results.

- The models will be rigorously tested using a set portion of the data. We will record metrics such as accuracy, precision, recall, and F1 score. A unique experiment will be designed to compare the Lexicon-based models with the machine learning models to ensure a fair and comprehensive comparison.
- The project will culminate in a presentation, a detailed project report, and the submission of all documented code, including references to any external datasets used. Our final deliverables aim to provide a holistic view of the process and outcomes of developing a state-of-the-art sentiment analysis model.