Phase-2 Analysis Report

Sentiment Analysis on Amazon *Office products* dataset

Organization: Centennial College

Team:

|  |  |
| --- | --- |
| Name | Student Id |
| Aiju Kinoshita | 301137120 |
| Austin Joyal | 301200483 |
| Parth Patel | 301207843 |
| Uma Pavan Kumar Chukkapalli | 301211308 |
| Vinay Mannem | 301211306 |

Date: April 16th, 2023

Course: COMP262-002

**Table of Contents**

[**About dataset**. 3](#_Toc132744062)

[List of the main finding of the dataset. 3](#_Toc132744063)

[Conclusions 5](#_Toc132744064)

[**Text cleaning and pre-processing.**  5](#_Toc132744065)

[Text representation. 6](#_Toc132744066)

[Chose the appropriate columns for your sentiment analyzer. 6](#_Toc132744067)

[**Modeling**  6](#_Toc132744068)

[Lexicon-based approach 6](#_Toc132744069)

[Machine Learning Approach 8](#_Toc132744070)

[**Comparisons** 13](#_Toc132744071)

[Conclusion 15](#_Toc132744072)

[**Review of “Recommender systems based on user reviews: the state of the art”** 16](#_Toc132744073)

[Idea for enhancing the rate values in Amazon Office Products dataset. 16](#_Toc132744074)

[Implementation of <chosen method> 16](#_Toc132744075)

[**References** 17](#_Toc132744076)

[**Appendix 1:** 19](#_Toc132744077)

[Project Plan 19](#_Toc132744078)

[**Appendix 2:** 20](#_Toc132744079)

[Meeting Register: 20](#_Toc132744080)

## **About dataset**.

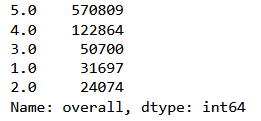
The Dataset is an updated version of the Amazon review dataset released in 2014. In total, the dataset contains 800357 rows × 12 columns. The columns are:

1. reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B
2. asin - ID of the product, e.g. 0000013714
3. reviewerName - name of the reviewer
4. vote - helpful votes of the review
5. style - a dictionary of the product metadata, e.g., "Format" is "Hardcover"
6. reviewText - text of the review
7. overall - rating of the product
8. summary - summary of the review
9. unixReviewTime - time of the review (unix time)
10. reviewTime - time of the review (raw)
11. image - images that users post after they have received the product
12. *Verified- verified user or not eg: true/false*

All columns are object type except *overall*, *unixReviewTime* which are int64 and *verified* which is bool type.

### List of the main finding of the dataset.

* The dataset is not balanced. We found that positive rated products outnumber the negative rated products making it an imbalanced dataset.

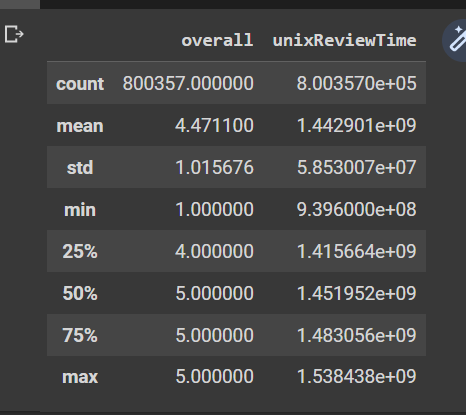


* Dataset have 27965 products but very few products have the greater number of reviews whereas others have significantly a smaller number of reviews

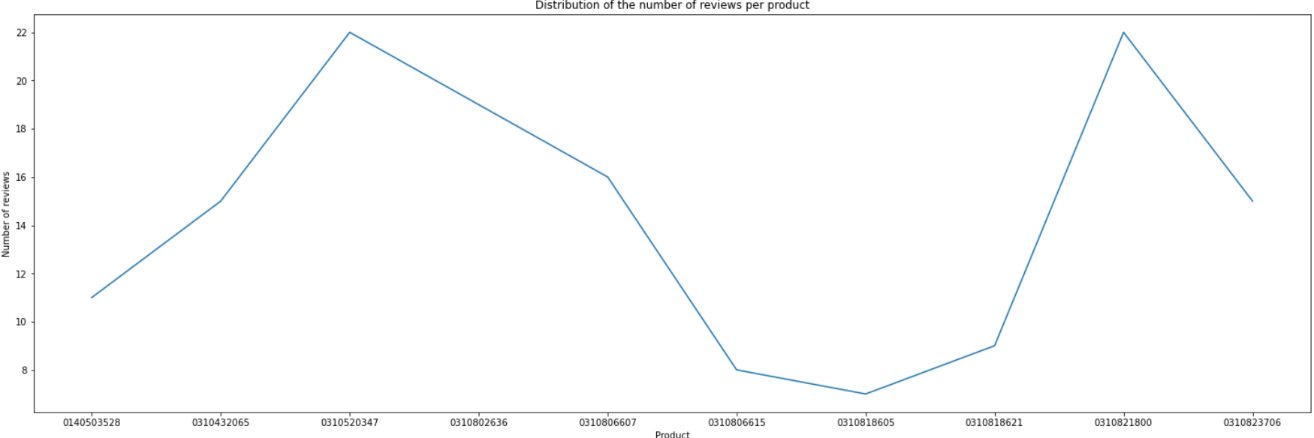




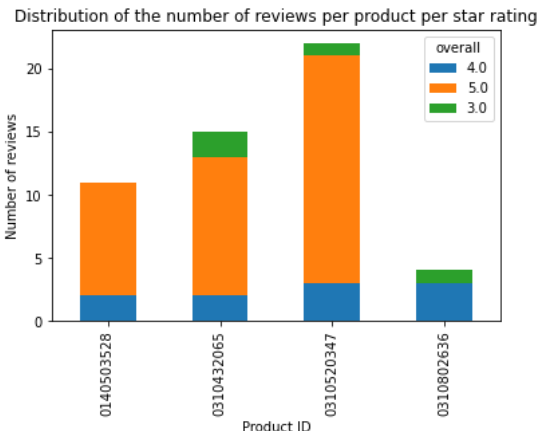
We see that average of review rating is around 4.47 which might imply that dataset contains more positive reviews:



Distribution of reviews per product:



Distribution of ratings of each product:



### Conclusions

Overall, the dataset has plenty data with helpful features to accomplish our sentimental analysis tasks and to evaluate or train our models, though we still need to do following processes:

1. Drop empty reviews
2. Text cleaning
3. Balance the dataset

Since our tasks are sentimental analysis and not product specific tasks, we would focus on balancing the number of ratings per classes rather than the balancing the number of reviews per product.

## **Text cleaning and pre-processing.**

On the text cleaning and pre-processing parts, we did the following on the data to make the data suitable for the model.

Note that balancing the dataset would hold after dropped null values.

Following actions are applied on *reviewText* column:

1. Drop null values.
2. Balance the dataset with remaining items
3. Lowercasing the reviews
4. Remove punctuation.
5. Remove digits using regular expressions.
6. Remove stop words from ‘English’ stop words library.
7. Tokenize the text using sentence tokenizer first followed by word tokenizer.
8. Lemmatization using wordnet from nltk library.

Following actions are applied on *overall* column:

1. Create a new label for sentimental analysis from the *overall* column.

After every processing, stratified sample the records for 500-1000 with new label.

### Text representation.

TF-IDF vectorizer was the best vector representation compared to ‘Bag of Words’ since the TF\_IDF model contains information on the more important words and the less important ones as well whereas BOW contains the count of each word in the vocabulary.

### Chose the appropriate columns for your sentiment analyzer.

We identified that the sentiment analyzer requires *reviewText* column and the *label* column (the new column of sentiments based on the *overall* column).

## **Modeling**

### Lexicon-based approach

We worked on the three models:

* VADR

VADR (Valence Aware Dictionary for Sentiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. It is available in the NLTK package and can be applied directly to unlabeled text data.

Usage:

VADR\_analyzer = SentimentIntensityAnalyzer()

vs = VADR\_analyzer.polarity\_scores(text)

The output vs is a dictionary of values, for example: {'neg': 0.0, 'neu': 0.857, 'pos': 0.143, 'compound': 0.3612}. The dictionary gives the score in each of the following categories:

Negative, neutral, positive, compound (computed by normalizing the scores)

* TextBlob

Like VADER, TextBlob is another Lexicon based model that can be used for sentiment analysis. It provides a simple API for common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. We are doing sentiment analysis with TextBlob.

Usage:

res = TextBlob("This movie is amazingly directed")

print(res.sentiment.polarity)

This will return the polarity values for the text. If the polarity is greater than 0, the text is positive. If the polarity is less than 0, the text is negative. Otherwise, a neutral text.

* SentiWordNet

SentiWordNet assigns three scores to each word: positivity, negativity, and neutrality, ranging from 0 to 1.0, where 0 represents the lowest intensity and 1.0 represents the highest intensity. The scores are based on the semantic relations of the words in WordNet, such as their synsets (groups of words with similar meanings) and their part-of-speech tags. However, it should be noted that SentiWordNet is not perfect and may produce inaccurate or inconsistent scores in some cases, especially for words with multiple senses or ambiguous meanings.

#### Evaluating

VADR

 Calendar

Description automatically generated

We have collected whether the analyzer output is Positive, Negative, or Neutral by accessing its polarity scores, and if the polarity score of negative is greater than the positive, then we assigned the review as Negative review. If the polarity score of positive is greater than the negative, then we assigned the review as Positive review. For all others, we assigned the review as Neutral review. From the classification report, the model failed to detect Negative reviews which would lead to the accuracy score the lowest among the Lexicon models.

TextBlob

 Calendar

Description automatically generated

We have collected whether the analyzer output is Positive, Negative, or Neutral by accessing its predicted sentiment polarity value. If the polarity value is equal to 0, then the sentiment of the review is Neutral. If the polarity value is less than 0, then the sentiment of the review is Negative. If the polarity value is greater than 0, then the sentiment of the review is Positive. From the classification report, Positive and Negative sentiments are well extracted from the review, though the Neutral sentiments were hard for the model to extract.

SENTIWORDNET

While fitting the pre-processed text to the model, since some review text only had stop word in its corpus, SENTIWORDNET received empty string which SENTIWORDNET cannot handle, so exception handling has been applied. We found two reviews have become empty string.

 Calendar

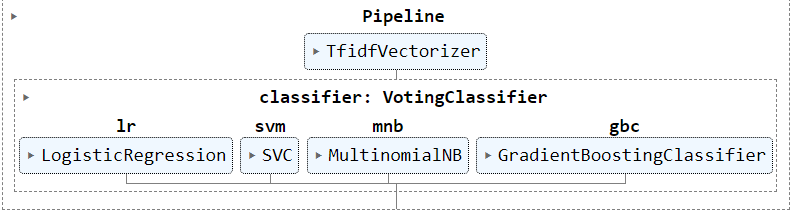
Description automatically generated

We have collected whether the analyzer output is Positive, Negative, or Neutral by accessing its sentiment score. If the sentiment score is larger than 0, then the sentiment of the review is marked as Positive. If the sentiment score is smaller than 0, then the sentiment of the review is marked as Negative. For other reviews which is if the sentiment score is 0, then we mark the review as Neutral. Additionally, for empty corpus, we assume that the review is Neutral. From the classification report, the model showed similar characteristics with TextBlob, though has better prediction on Neutral reviews.

Overall, every Lexicon based model could make better predictions on Negative and Positive reviews than Neutral reviews. Since Neutral reviews contain both Positive and Negative words in same corpus, and the distribution of each word in a corpus varies, it is understandable that predicting Neutral statement may be difficult task.

### Machine Learning Approach

Since we are required to build several supervised machine learning models to evaluate, we have built voting classifier with Logistic Regression Classifier, Random Forest Classifier, Support Vector Machine Classifier, Decision Tree Classifier, Extra Trees Classifier, Multinomial Naïve Bayes Classifier, and Gradient Boosting Classifier.



Logistic Regression is a supervised machine learning classification algorithm that predicts the probability of an event occurring based on input features. The strength of the algorithm is its simplicity and interpretability. It can be used for a wide range of classification tasks and can provide insights into the relationships between the input features and the target variable. Additionally, Logistic Regression can be easily extended to handle multi-class classification problems and can handle missing values in the input data. However, Logistic Regression may not perform well when the data is not linearly separable, and other more complex models may be needed for such cases.

For Logistic Regression modeling, we have used “balanced” class weight which the algorithm automatically adjusts the weights of the classes based on the number of instances in each class and set max iteration to 1400.

Random Forest Classifier is an ensemble learning algorithm that combines multiple decision trees to make more accurate predictions. The strength of the algorithm is its high accuracy, ability to handle large datasets with high dimensionality, and robustness to noisy data. It is also relatively fast to train and can handle missing values in the input data. Additionally, Random Forest Classifier provides a measure of feature importance, which can be used for feature selection and to gain insights into the relationship between the input features and the target variable. However, Random Forest Classifier may not perform well on very small datasets and may not provide as much interpretability as some other models.

For Random Forest Classifier modeling, we have used “entropy” criterion which let the algorithm split the data based on the information gain, set maximum depth to 22, minimum number of samples to split into 4, and 250 estimators.

Support Vector Machine (SVM) Classifier is a powerful supervised machine learning classification algorithm that works by finding the optimal hyperplane that separates different classes in the data. The strength of the algorithm is its ability of handling both linear and non-linear classification tasks, high accuracy, and ability to work effectively with high-dimensional data. Moreover, SVM Classifier is robust to overfitting, and could handle non-linearly separable data by using kernel functions. However, SVM Classifier may not perform well on too large datasets or datasets with many noisy features. It can also be sensitive to the choice of hyperparameters, which would require some tuning on it.

For SVM Classifier modeling, we have set probability parameter to True so that the algorithm would estimate class probabilities using Platt scaling or other similar method and keep the other parameters as default.

Decision Tree Classifier is a simple yet powerful supervised machine learning algorithm for classification tasks that works by recursively splitting the data into subsets based on the values of input features. The strength of the algorithm is its simplicity, interpretability, and ability to handle both numerical and categorical data. Additionally, Decision Tree Classifier can handle non-linearly separable data and can be used for feature selection. However, Decision Tree Classifier can suffer from overfitting on noisy data or small datasets. It can also be sensitive to the choice of hyperparameters, which shares same issue with SVM.

For Decision Tree Classifier modeling, we used “entropy” criterion, maximum depth of 22, and balanced class weight.

Extra Trees Classifier is another ensemble learning algorithm that combines multiple decision trees to make more accurate predictions. The strength of the algorithm is its high accuracy, ability to handle large datasets with high dimensionality, and robustness to noisy data. It is also relatively fast to train and can handle missing values in the input data. Additionally, Extra Trees Classifier provides a measure of feature importance, which can be used for feature selection and to gain insights into the relationship between the input features and the target variable. However, Extra Trees Classifier may not perform well on very small datasets and may not provide as much interpretability as some other models.

For Extra Tree Classifier modeling, we used “entropy” criterion, maximum depth of 22, and balanced class weights.

Multinomial Naïve Bayes is a probabilistic supervised machine learning algorithm used for classification tasks that works by calculating the probabilities of different classes based on the frequencies of the input features. The strength of the algorithm is its simplicity, speed, and ability to handle discrete data like word counts. Additionally, it works well with text data and requires relatively few training samples. Multinomial Naive Bayes is also robust to noise and can handle irrelevant or redundant features in the data. However, Multinomial Naive Bayes makes the strong assumption of independence between the input features, which may not hold in practice. It can also suffer from the "zero-frequency" problem when working with rare features that do not appear in the training data.

For Multinomial Naïve Bayes modeling, we used default parameters.

Gradient Boosting Classifier is also an ensemble learning algorithm that uses a series of weak learners, typically decision trees, to make more accurate predictions. The strength of the algorithm is its high accuracy, ability to handle complex datasets with high dimensionality, and robustness to noise and outliers. It is also relatively fast to train and can handle missing values in the input data. Additionally, Gradient Boosting Classifier provides a measure of feature importance, which can be used for feature selection and to gain insights into the relationship between the input features and the target variable. However, Gradient Boosting Classifier may not perform well on very small datasets and may require tuning of the hyperparameters to achieve optimal performance as like Decision Tree Classifier.

For Gradient Boosting Classifier modeling, we used 250 estimators, a learning rate of 1.0, and maximum depth of 22.

Voting Classifier is an ensemble learning algorithm that combines the predictions of multiple classifiers to make a final prediction. The classifier has its strength on ability to improve the accuracy and robustness of predictions by combining the intensities of different models. It is simple to implement, and it also can help to reduce overfitting by combining the predictions of different models that have been trained on different subsets of the data. However, the performance of Voting Classifier depends on the performance of the individual classifiers and their diversity. It may also be sensitive to imbalanced datasets where one class has a much higher frequency than the other classes.

For Voting Classifier modeling, we composed all above estimators, and set to soft voting since it practically obtains more accurate prediction than hard voting.

Pipeline the TF-IDF vectorizer and the voting classifier.

#### Training

Split the dataset into training (70%) and testing (30%) dataset.

Features are string of lemmatized corpus and labels are sentiment value labels. Note that lemmatized corpus should be converted into string to pass TF-IDF in the pipeline.

Train the Voting Classifier with training dataset.

#### Testing

Retrieve the prediction with trained model using testing dataset and compare with true prediction. Calendar

Description automatically generatedBackground pattern

Description automatically generated

The first scores, confusion matrix, and classification report are analysis of our model. Model score have calculated with training dataset, and Test score have calculated with testing dataset. According to those two scores, it shows that the model have well learned about the training dataset but, it have difficulties on testing dataset. The following matrix and classification report have hold upon testing dataset. From the report, the model have higher accuracy on prediction Negative and Positive reviews, but poor with Neutral reveiws. The second heat map is constructed with prior confusion matrix to visually see the model’s strengths and weaknesses. From the confusion matrix heatmap, the trained voting classifier has confident prediction on True-Positive prediction of Positive and Negative sentiments, but not for Neutral sentiment. Also note that it the False-Positive prediction on Neutral review as Positive sentiment have occurred many times. Since the False-Positive prediction on Negative review as Positive and False-Positive prediction on Positive review as Negative have recorded few occurrences, The model has good performance on differentiating Positive reviews and Negative reviews.

We have also evaluated each classifier in the voting classifier.

Calendar

Description automatically generatedCalendar

Description automatically generatedCalendar

Description automatically generatedCalendar

Description automatically generated

Calendar

Description automatically generatedCalendar

Description automatically generated

Calendar

Description automatically generated

For every model, they all have similar characteristics: predicting better on Positive and Negative reviews but not for Neutral reviews. Among every model, Logistic Regression Classifier model and Multinomial Naïve Bayes Classifier model have performed best, while Extra Trees Classifier model perform the worst.

Overall, the model has trained enough with the training dataset which reached itself to overfitting situation. The model is good at predicting whether it is Positive or Negative from each review, but it mixed up for Neutral reviews, which do make sense since their review may contain both negative and positive words to make our model confusing.

## **Comparisons**

For comparison, we have also conducted analysis with original text since we noticed it does makes difference, and we could compare the accuracy with original text and accuracy with preprocessed text as well.

From comparing the lexicon models, we found that VADR worked with good accuracy with non-preprocessed data while TextBlob did well with preprocessed data compared to the original text data. SENTIWORDNET have scored similar accuracy with original data and preprocessed data.

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy (with original text)** | **Accuracy (with preprocessed text)** |
| VADR | 47.72% | 34.65% |
| TextBlob | 32.73% | 42.20% |
| SENTIWORDNET | 43.52% | 42.32% |

From comparison with lexicon-based models and a machine learning Voting Classifier model, we have better prediction on our machine learning model than the Lexicon-based model. In terms of accuracy with preprocessed text, we made 9% to 16% increase, and in term of accuracy with original text, we have made 5% to 20% increase on its accuracy. In terms of f1-score with preprocessed text, we made 0.1 to 0.55 increase on its Negative f1-score, -0.13 to 0.13 increase on its Neutral f1-score, and 0.07 to 0.47 increase on its Positive f1-score. In terms of f1-score with original text, we have made 0.06 to 0.57 increase on its Negative f1-score, -0.09 to 0.19 increase on its Neutral f1-score, and -0.01 to 0.55 increase on its Positive f1-score.

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy (with original)** | **Accuracy (with preprocessed)** |
| VADR | 47.72% | 34.65% |
| TextBlob | 32.73% | 42.20% |
| SENTIWORDNET | 43.52% | 42.32% |
|  |  |  |
| Voting Classifier | 52.99% | 51.39% |

|  |  |  |
| --- | --- | --- |
| **Model** | **F1-score (with original)**  **Negative : Neutral : Positive** | **F1-score (with preprocessed)**  **Negative : Neutral : Positive** |
| VADR | 0.52 : 0.21 : 0.58 | 0.00 : 0.50 : 0.14 |
| TextBlob | 0.02 : 0.49 : 0.02 | 0.42 : 0.14 : 0.54 |
| SENTIWORDNET | 0.53 : 0.14 : 0.50 | 0.45 : 0.24 : 0.50 |
|  |  |  |
| Voting Classifier | 0.59 : 0.40 : 0.57 | 0.55 : 0.37 : 0.61 |

For each supervised machine learning model including Voting Classifier, Multinomial Naïve Bayes Classifier scored the best accuracy which is 54.98% followed by Logistic Regression which the difference of accuracy is 0.4% with preprocessed dataset. With the original dataset, which the preprocessing tasks would be held by TF-IDF built-in preprocessor, Voting Classifier scored the best accuracy which is 52.99%, and Extra Trees Classifier comes to the next with difference of 1.2%.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | | **Accuracy (with original)** | **Accuracy (with preprocessed)** |
| Voting Classifier | | 52.99% | 51.39% |
| Logistic Regression | 50.60% | 54.58% |
| Random Forest Classifier | 51.00% | 51.00% |
| Support Vector Machine Clf. | 51.39% | 51.79% |
| Decision Tree Classifier | 48.61% | 43.82% |
| Extra Trees Classifier | 51.79% | 47.01% |
| Multinomial Naive Bayes | 50.60% | 54.98% |
| Gradient Boosting Classifier | 49.80% | 49.40% |

Voting Classifier (preprocessed text dataset) Voting Classifier (original text dataset)

Background pattern

Description automatically generated Background pattern

Description automatically generated

From the above confusion matrix heatmap, both classifiers have better ability to differentiate Positive or Negative sentiments from the review than detecting the review’s sentiment is Neutral or not. The classifier trained with TF-IDF built-in preprocessor preprocessed text has better True-Positive prediction on Negative and Positive, though, it also has more False-positive prediction on both sentiments. As of Neutral sentiment review, the model trained with TF-IDF built-in preprocessor preprocessed text has better True-Positive and True-Negative predictions than the classifier trained with text dataset preprocessed by us.

### Conclusion

Based on the observation and comparison of Lexicon-based model and Machin Learning approach model, it can be concluded that the Voting Classifier Pipelined model trained with data processed by TF-IDF built-in preprocessor performs better in term of accuracy and f1-scores for our sentimental analysis task. This model showed an accuracy of 52.99% with original text and 51.39% accuracy with preprocessed text, and both are higher than any Lexicon-based models. Moreover, confusion matrix heatmap showed the Voting Classifier Pipelined model trained with original text dataset have less False-Positive prediction on Neutral reviews and more True-Positive prediction on other two sentiment reviews. Therefore, it is recommended to use the Voting Classifier pipelined with TF-IDF vectorizer which also do the preprocessing tasks to the text dataset trained with un-processed text dataset.

## **Review of “Recommender systems based on user reviews: the state of the art”**

### Idea for enhancing the rate values in Amazon Office Products dataset.

User Rating Enhancement using Opinion Mining:

Our aim is to use a Natural Language Processing (NLP) technique to preprocess the review text, extract opinion words, and determine the sentiment orientation of each opinion word. It also calculates the overall sentiment strength of each review and maps it to a corresponding rating on a 5-point scale. The ratings are then added to the dataframe for further analysis.

It is achieved by using Python and various libraries such as Pandas, NLTK, and WordNetLemmatizer. The NLTK library is used for preprocessing the text by tokenizing and lemmatizing the words, removing stop words, and extracting adjectives and verbs as opinion words. The sentiment orientation of each opinion word is determined by comparing their frequency in positive and negative reviews. The overall sentiment strength of each review is calculated by counting the number of positive and negative opinion words and mapping it to a corresponding rating on a 5-point scale. Finally, the ratings are added to the dataframe for further analysis.

The goal is to successfully enhance the user rating based on the opinions in the review text. The output is a dataframe containing the original review text, sentiment label (positive or negative), and the calculated rating on a 5-point scale. The ratings range from 1 to 5, with 1 being the lowest and 5 being the highest. The ratings are based on the overall sentiment strength of each review, which is determined by the frequency of positive and negative opinion words in the review text.

The code developed for enhancing user rating based on the opinions in the review text is an effective way to analyze and quantify the sentiment of the users. The use of NLP techniques and libraries such as NLTK and WordNetLemmatizer makes the process easier and faster. The ratings obtained through this method can be used for various purposes such as product improvement, marketing strategy, and customer satisfaction analysis.

*Implementation of user and product profile building*:

It can be used to build profiles of users and products based on their reviews and sentiments. These profiles can be used for various applications, such as recommendation systems.

The code uses various machine learning techniques to accomplish this. First, it reads in a dataset of Amazon product reviews that has been preprocessed and saved as a CSV file. Then it reads in a separate labeled dataset of sentiment analysis and splits it into training and testing sets. After that, it trains a Support Vector classifier on the training set to perform sentiment analysis on the reviews. The classifier's performance is evaluated on the testing set, and then the classifier is used to predict the sentiment label for each review in the product review dataset. Using Non-negative Matrix Factorization (NMF) performed the topic modeling on the reviews. This helps to identify the main topics that are being discussed in the reviews.

Using the sentiment labels and dominant topics identified by NMF, the code builds user and product profiles. The user profiles are based on the reviews and sentiments of each user, and the product profiles are based on the reviews and sentiments of each product. Finally, using cosine similarity to calculate the similarity between users and products. This similarity can be used to make recommendations to users based on the preferences of similar users.

Overall, this second part of state of art is to build user and product profiles and make recommendations based on those profiles. It is a powerful tool for anyone who wants to analyze customer reviews and improve their recommendation systems.

## **References**

Baccianella, S., Esuli, A., & Sebastiani, F. (n.d.). SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining.

Chen, L., Chen, G., & Wang, F. (2015, January 22). Recommender systems based on user reviews:.

Hutto, C. (2022, April 1). *VADER-Sentiment-Analysis*. Retrieved from GitHub: https://github.com/cjhutto/vaderSentiment

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (n.d.). *sklearn.ensemble.ExtraTreesClassifier*. Retrieved from Scikit-learn: Machine Learning: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (n.d.). *sklearn.ensemble.GradientBoostingClassifier¶*. Retrieved from Scikit-learn: Machine Learning: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (n.d.). *sklearn.ensemble.RandomForestClassifier*. Retrieved from Scikit-learn: Machine Learning: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (n.d.). *sklearn.ensemble.VotingClassifier¶*. Retrieved from Scikit-learn: Machine Learning: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (n.d.). *sklearn.linear\_model.LogisticRegression¶*. Retrieved from Scikit-learn: Machine Learning: https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (n.d.). *sklearn.naive\_bayes.MultinomialNB¶*. Retrieved from Scikit-learn: Machine Learning: https://scikit-learn.org/stable/modules/generated/sklearn.naive\_bayes.MultinomialNB.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (n.d.). *sklearn.svm.SVC*. Retrieved from Scikit-learn: Machine Learning: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (n.d.). *sklearn.tree.DecisionTreeClassifier*. Retrieved from Scikit-learn: Machine Learning: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

TextBlob. (n.d.). *TextBlob: Simplified Text Processing*. Retrieved from TextBlob: https://textblob.readthedocs.io/en/dev/

## **Appendix 1:**

### **Project Plan**

## **Appendix 2:**

### Meeting Register:

Meeting was scheduled every Wednesday in the following weeks:

|  |  |  |
| --- | --- | --- |
| **Date** | **Members attended** | **Subjects discussed** |
| Week 3 | All | Data acquisition methods |
| Week 4 | All | Data exploration methods and findings |
| Week 5 | All | Further data exploration, preprocessing |
| Week 6 | All | Preprocessing and modeling |
| Week 7 | All | Preprocessing, modeling, and comparing |
| Week 9 | All | Performance enhancing by preprocessing the data vigorously |
| Week 10 | All | State of Art research |
| Week 11 | All | State of Art research |
| Week 12 | All | State of Art research, State of Art models implementation |
| Week 13 | All | State of Art models implementation, Report and PPT |