Text Analytics – A Comparative Analysis of Brand Reviews

Jaime Rubio Diaz

40425150

Contents

1. Introduction and Background

The objective of this report is to outline the analysis conducted on customer reviews of two brands, using a dataset of 5,722 entries. Given the significant number of missing entries in the "emotions" column, a semi-supervised learning (SSL) approach is adopted to accurately label these missing observations, drawing on both the existing labels and the customer reviews. This method enables an analysis of emotional responses across both brands. The insights gained from this analysis provide the brands with a better understanding of their performance.

Numerous studies focus on the sentiment analysis of customer reviews, due to the valuable insights they offer companies. Singla, et. al., 2017 using a data set of 400,000 reviews, carried out a supervised learning analysis to develop a classifier that categorizes reviews into two sentiment types: positive and negative. These approaches are similar to those initially employed in this project, utilizing the labelled data.

Regarding semi-supervised learning, in the context of detecting fake opinions in hotel customer reviews, Ligthart et al., (2021) found that self-training algorithms based on Naïve Bayes, maintained the performance of a supervised model with only a small portion of the data labelled. Conversely, Widmann and Verberne (2017), explored graph-based algorithms like label propagation, using the 20-Newsgroup corpus dataset, highlighting the high interpretability of these algorithms in a semi-supervised context, particularly for identifying key phrases in documents or understanding their relationships to labels.

2. Methodology

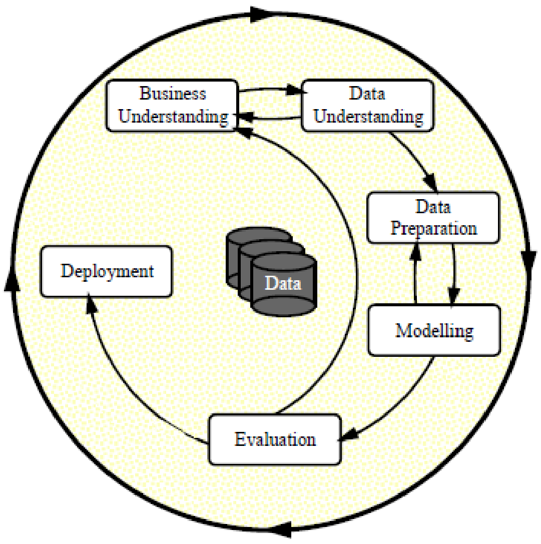


Figure 1. CRISP-DM
(Source: Wirth and Hipp, 2000)

Business Understanding

The nature of the business is unknown, but the presence of reviews in over 20 languages and mentions of delivery issues in those reviews suggest that both brands operate online. Reputation is the most important thing for online stores, as customers frequently view the reviews of others before making a purchase, therefore analysing and addressing issues highlighted in these reviews can lead to more positive reviews and attract more customers (Halim et al., 2022).

Data Understanding

Table 1. Data description

Variables		
ID_	Unique number for each customer	Categorical
brand_name_	Name of the brand	Categorical
country_	Customer country	Categorical
star_rating_	Rating from 1 to 5	Numerical (Ordinal)
emotions_	Emotion of the customer review	Categorical
text_reviews_	Text review of the customers	text

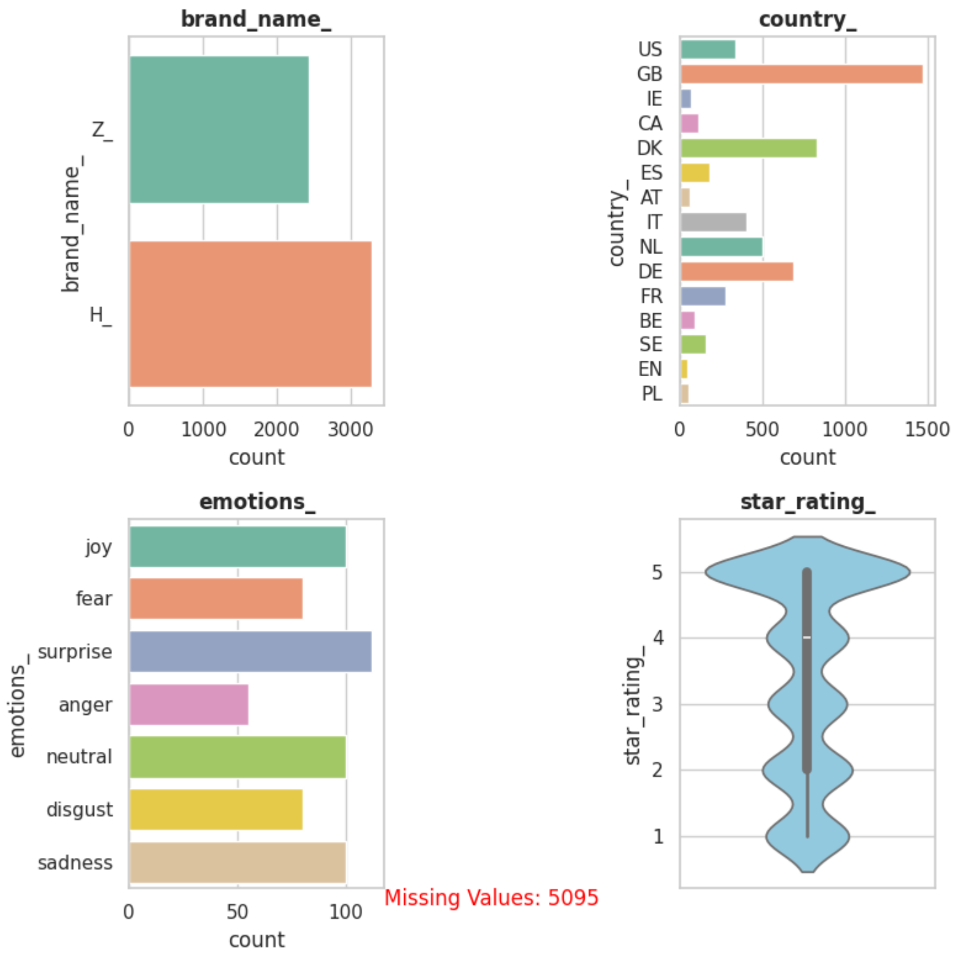


Figure 2. First data visualizations

(** Given the extensive variety of countries represented in the dataset, only the top 15 are shown in the graph.)

The dataset stores reviews for 2 brands, namely "Z_" and "H_". Notably, the dataset reveals that the predominant rating by customers is 5 stars. In terms, of emotions "surprise" is the most commonly reported. However, it is important to highlight that 5095 reviews do not have specified emotional responses. Furthermore, the data indicates that Great Britain (GB) is the most frequently mentioned country. Given that all the data is categorical, except for the star rating variable which is ordinal, frequency-related graphs such as bar charts or pie charts are suitable for the categorical data. For the distribution of the categories and the ordinal data, violin plots can be used. Violin plots are particularly useful compared to boxplots because they not only display the median and interquartile ranges but also provide a detailed visualization of the entire data distribution, including potential multimodality (Hintze & Nelson, 1998).

Data Preparation

The variables "brand_name_", "country_", "star_rating_", and "emotions_" do not require pre-processing. However, "text_reviews_" variable needs to be pre-processed. As shown in Figure 3, the reviews are in various languages, but they need to be translated into English for accurate prediction and analysis of the unlabelled emotions. For this project, Deepl API and Baidu API are used for tranlation. The Deepl API allows for the translation of up to 500,000 text characters for free, hence Baidu API is used for the remaining translations. After translation, a new data set is generated where all the reviews are in English.

Secondly, to manipulate text in Python it is necessary to clean it. Cleaning the text can enhance the performance of the algorithm by removing noise and reducing the complexity of the text. The following graph illustrates the steps taken in the data cleaning process.

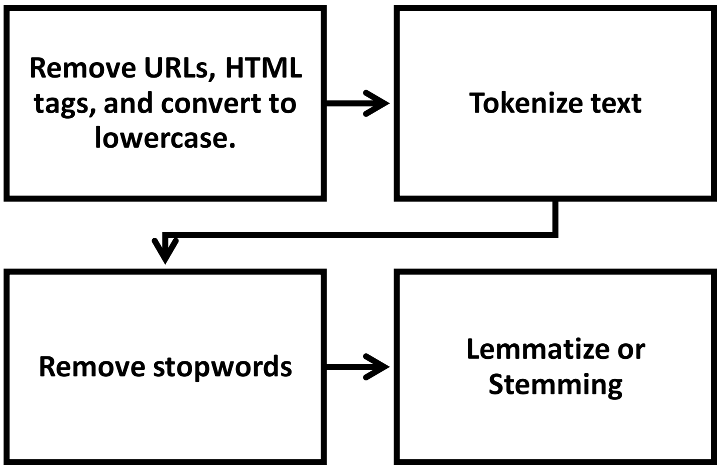


Figure 3. Cleaning text process

The process includes the removal of URLs, HTML tags and the conversion to lowercase, as well as the removal of stopwords, which are elements or words that do not contribute meaningfully to the document, after this the text is tokenized, converting it into individual words (Vijayarani at. al., 2015). Lemmatization and stemming are both text normalization techniques aimed at word reduction. Stemming simplifies words by removing endings through basic suffix elimination, often resulting in non-words; while lemmatization is a more time-complex process that considers the part of speech of a word and uses a lexical resource to ensure that the word reduction corresponds to a valid lemma (Chai, 2022).

Once the text is cleaned, it needs to be converted into a numerical format that machine learning algorithms can process. Two of the most common techniques are Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). BoW focuses on the presence or absence of words in a document, while TF-IDF takes into account not only the presence, but also the relative importance of the words within the sentence in relation to the document (Pimpalkar and Retna, 2020).

$$TF - IDF = \frac{(Times\ term\ appears\ in\ the\ sentence)}{(Total\ terms\ in\ the\ sentence)} * log \frac{(Total\ sentences)}{(Total\ sentences\ with\ the\ term)}$$

(Source: Pimpalkar and Retna, 2020)

Although using all text features can result in good performance, feature selection often enhances classifier performance (Bird et al., 2009). When using "tfidvectorizer", a parameter can be set to select the number of features for each review, choosing the most frequent words for that review. Additionally, "ngram_range" parameter can be customized during vectorization, determining the range of word combinations considered. Both parameters are evaluated with a Logistic Regression model.

Modelling

In a semi-supervised learning context, an initial supervised phase is conducted to leverage labelled data and to stablish a baseline mode. For text classification with a target variable of five levels, Gurcan (2018) concluded that Multinomial Naïve Bayes algorithm, with an accuracy of 90%, was the most effective for the task. Furthermore, Joshi and Abdelfattah (2021) conducted research to predict medical conditions associated with drug effects based on online pharmaceutical reviews using six different supervised machine learning classifiers, finding that the Linear Support Vector Classifier performed the best. Based on these findings both algorithms will be tested with our data, in combination with other classifiers.

Table 2. Supervised learning models

Supervised Learning Classifiers	
Random Forest	Random Forest expands upon the decision tree model by generating a multitude of trees in a randomized way, enhancing accuracy while preserving interpretability (Rigatti, 2017). Despite the robustness, irrelevant features can distract the model.
K-Nearest Neighbours (KNN)	The K-Nearest Neighbours classifier assigns a category to an observation by analysing the categories of the nearest examples within the dataset, it calculates proximity using distance metrics like Euclidean or Manhattan distance to identify similarities (Zhang, 2016). The performance of KNN decreases with high-dimensional data.
Support Vector Machine	This algorithm employs a nonlinear mapping to transform the data to form a hyperplane in a transformed space, where it performs classification using kernel functions (Hearst et al., 2002). SVM requires careful tuning of hyperparameters, and the selection of an appropriate kernel function.
Stochastic Gradient Descent (SGD)	Stochastic Gradient Descent (SGD) is an optimization technique, particularly effective for handling large-scale and sparse data sets, that can employ various loss functions to train models efficiently (*scikit, no* date).
Multinomial Naïve Bayes	MNB is a variant of the Naïve Bayes algorithm, often used in text classification, which is particularly useful for feature vectors where the attributes represent the frequencies by a multinomial distribution (McCallum and Nigam, 1998).

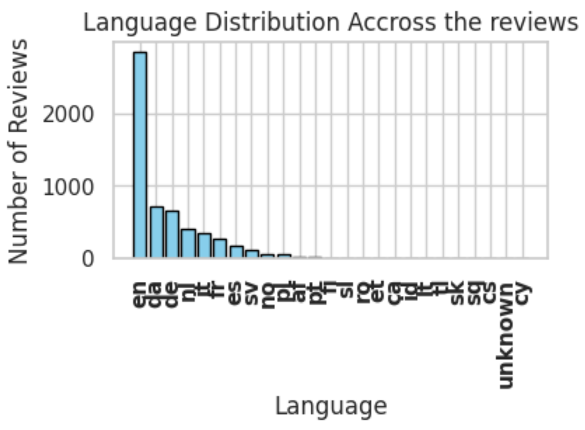
Once supervised approach is performed, a semi-supervised approach is employed for labelling the unlabelled data. Semi-supervised learning, a machine learning approach that combines labelled data with a large amount of unlabelled data, is particularly useful as labelling can be time-consuming and expensive (Zhu, 2005). Van Engelen and Hoos (2020), categorize semi-supervised techniques into inductive methods, which use all the data to create a model that can predict labels for new data, and transductive methods, that are focused on the propagation of the labels based on direct connections. Common algorithms in the first category include wrapper methods such as Self-training, while in the second category Graph-based methods. Graph-based methods propagate labels by using a graph $G = (V, E)$, where V represents pairs of labelled and unlabelled data, and E a matrix indicating similarity between V pairs, while Self-training, generates pseudo-labels to expand the training data set with labels that enhance the performance, resulting in a model similar to supervised approach (Duarte and Berton, 2023).

After semi-supervised, the best-performing model is selected, and it is used to predict the unlabelled data. These predictions are incorporated into the original dataset, resulting in a dataset without missing values, allowing for an analysis of the brand reviews.

For the analysis, considering the abundance of text features, a logistic regression model can be constructed to examine the importance of coefficients associated with specific words (Müller and Guido, 2016 pag. 426-431). The coefficients from the model are then plotted to identify the most influential words for predicting each emotion. This approach provides an overview of key linguistic indicators, revealing the most impactful words in customer reviews associated with each emotion.

3. Text Data Pre-Processing

As detailed above, the first step was to translate the text reviews into English. Figure 3 illustrates the distribution of languages in the original reviews, and Figure 4 shows the distribution after translation. Although other languages still appear in Figure 4, they are present in a negligible proportion. This may be because the API failed to recognize these languages, and therefore, did not translate them, or because the "langdetect" library incorrectly identified them due to the presence of special characters in those observations. Regardless, this is not a concern due to the small amount they represent and the overall improvement over the original reviews. Now, 5,583 reviews are in English, compared to only 2,831 in the original dataset.



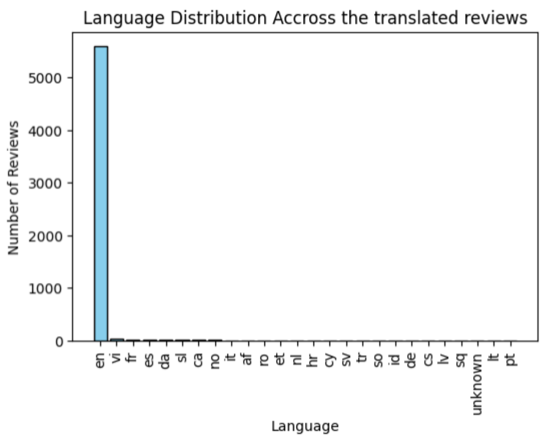


Figure 4. Reviews language distribution
Figure 5. Reviews language distribution after the translation

The methodology section detailed the steps followed for cleaning the text. Here, the results after each step are shown for the first review:

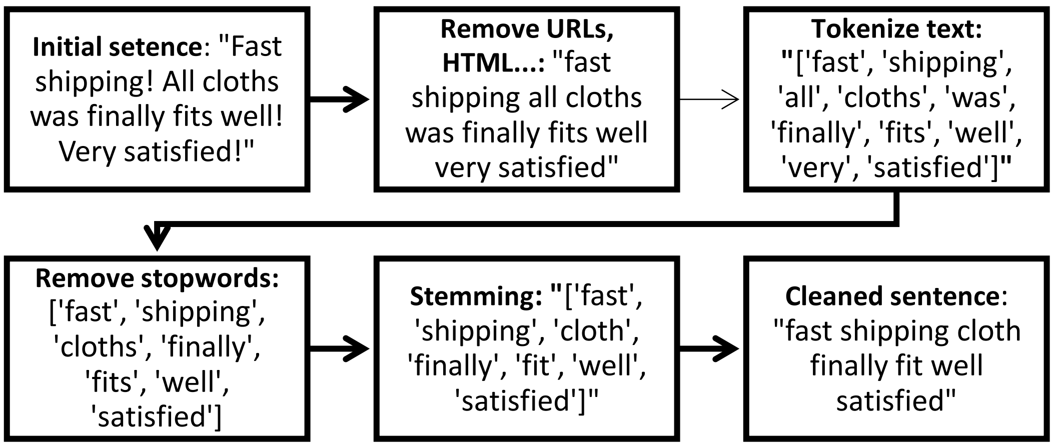
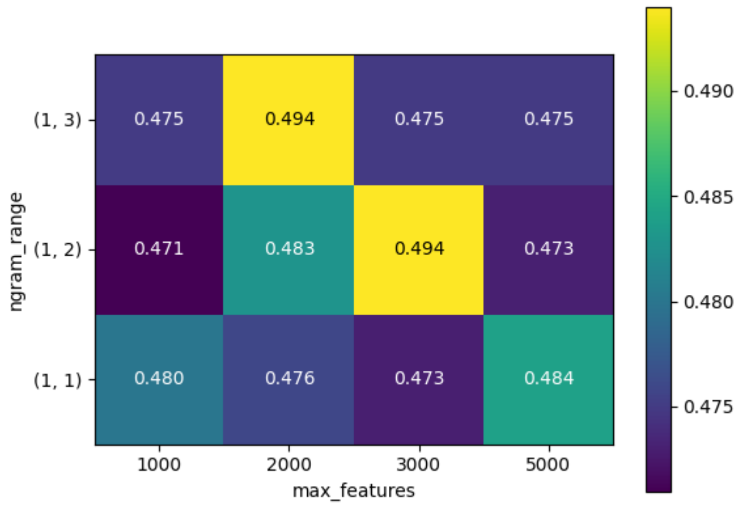

Initial setence: "Fast shipping! All cloths was finally fits well! Very satisfied!"
Remove URLs, HTML...: "fast shipping all cloths was finally fits well very satisfied"
Tokenize text: "['fast', 'shipping', 'all', 'cloths', 'was', 'finally', 'fits', 'well', 'very', 'satisfied']"
Remove stopwords: ['fast', 'shipping', 'cloths', 'finally', 'fits', 'well', 'satisfied']
Stemming: "['fast', 'shipping', 'cloth', 'finally', 'fit', 'well', 'satisfied']"
Cleaned sentence: "fast shipping cloth finally fit well satisfied"

Figure 6. Results obtained from test pre-processing

From the initial review text, "Fast shipping! All cloths was finally fits well! Very satisfied!", the text cleaning and normalization process transforms it into "fast shipping cloth finally fit well satisfied". In this example, stemming was chosen to normalize the text. This decision was based on experimentation within a logistic regression model, where stemming led to better performance.

After cleaning, the text is vectorized using the TfidfVectorizer. Different values for the ngram_range and max_features parameters are tested within a Logistic Regression model to find the most effective combination for handling the text data.



TfidfVectorizer selected parameters	
n_features	3000
ngram	(1,2)

Table 3. "TfidfVectorizer" parameters selection

4. Text Analytic

4.a. Supervised & Semi-Supervised Machine Learning

Initially, the data set is split into labelled data and unlabelled data. For the supervised learning models the labelled data is split into train data set (80%) and test data set (20%). Once split, the supervised models are constructed and their performance, in terms of accuracy, is assessed after hyperparameter tuning.

Table 4. Supervised learning models

Classifier	Best Hyperparameters	Accuracy
Random Forest	{'clf__max_depth': 20, 'clf__min_samples_split': 10, 'clf__n_estimators': 200}	0.60
K-Nearest Neighbours	{'clf__algorithm': 'auto', 'clf__n_neighbors': 7, 'clf__weights': 'distance'}	0.36
Support Vector Machine	{'clf__C': 1, 'clf__kernel': 'linear'}	0.46
SDG	**{'clf__alpha': 0.001, 'clf__loss': log, 'clf__penalty': 'l1'}**	**0.61**
Multinomial Naïve Bayes	{'clf__alpha': 0.01}	0.47

The highest accuracy among supervised classifiers is shown by Stochastic Gradient Descent (SDG) with the loss function set to "log".

Table 5. Hyperparameter explanation for the best performance model

Source: scikit (no date)

Hyperparameter explanation for SDG	
clf_loss	It is the loss function used by the model. 'log' = logistic regression
penalty	Regularization applied. 'l1' = commonly referred as lasso.
alpha	Used to compute learning rate and multiples the regularization.

Table 6. Potential reasons for performance and/or limitations supervised learning.

Classifier	Potential Reasons for Performance and/or Limitations
Random Forest	It demonstrates strong performance, probably stemming from its capability to effectively manage non-linear and categorical relationships.
K-Nearest Neighbours	The distance metrics for KNN performs worse in high-dimensional spaces, which is the case of text-data (Kouiroukidis and Evangelidis, 2011).
Support Vector Machine	Since it requires careful hyperparameter tuning, the small size of the dataset may have influenced the improper selection of hyperparameters.
SDG	SGD is robust to a large number of features in text data, the use of logistic regression (loss=log) and L1 penalty might have helped in text feature selection.
Multinomial Naïve Bayes	The use of td-idf instead of feature counts may have affected the performance of the algorithm (*scikit, no date).*

For the semi-supervised learning phase, train and test datasets need to be built. The test dataset used for the semi-supervised remains the same as that used in the supervised phase to ensure that the performance of the semi-supervised models is evaluated against labelled data. The train dataset is constructed by combining the training data set used in the supervised models and all unlabelled data.

This results in a test dataset consisting only of labelled data and a train dataset that includes both labelled and unlabelled data.

Following dataset preparation, emotions are mapped and converted to a numerical format in both the train and test datasets. With the data ready, the semi-supervised models are then constructed, employing both “Self-Training” and “Label Spreading”. For the Self-Training approach, the parameters selected are those which provided the best performance in the previous phase.

Table 7. Semi-supervised learning models

Label Spreading	**Accuracy**
Label Spreading	0.32

SelfTraining Classifier	**Accuracy**
Random Forest	0.62
Support Vector Machine	0.55
SDG	**0.63**
Multinomial Naïve Bayes	0.43

Self-training classifier shows better performance than Label Spreading, and just as in the supervised learning phase, the Stochastic Gradient Descent (SGD) classifier demonstrates the best performance among the Self-learning models.

Table 8. Potential reasons for performance and/or limitations SSL

SSL method	**Potential Reasons for Performance and/or Limitations**
Label Spreading	Label Spreading assumes that similar data points are likely to share labels, which may not accurately reflect the actual relationships in this dataset.
SelfTraining	The main limitation could be that the model may propagate errors if the initial model predictions are inaccurate, leading to amplified errors in subsequent iterations. Duarte and Berton, (2023) discusses some strategies to select more confidently labelled instances.

Utilizing the Self-training SGD classifier, the unlabelled data is effectively labelled, expanding the dataset's utility for further analysis. Following this, in line with the methodology outlined earlier, two logistic regression models are constructed, one for each brand, using the text reviews and emotions columns, obtaining an accuracy of 0.69 for Brand Z_ and 0.75 for Brand H_ (results of the models will be detailed in the “Results and Discussion” section).

4.b. Visual analytics

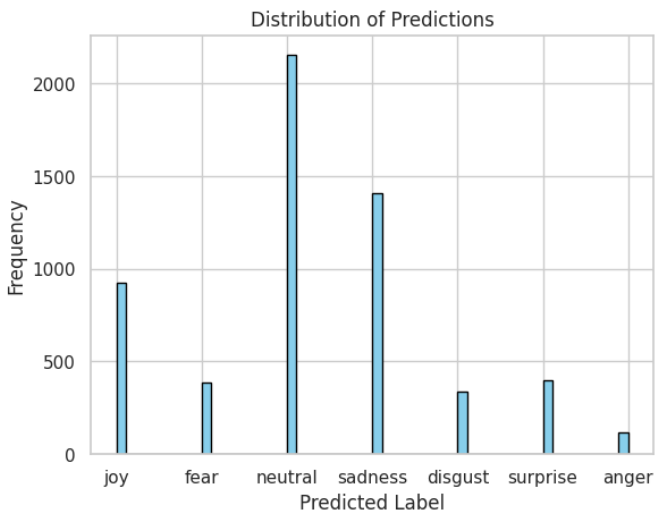


Figure 7. Predicted labels after filling missing values with predictions

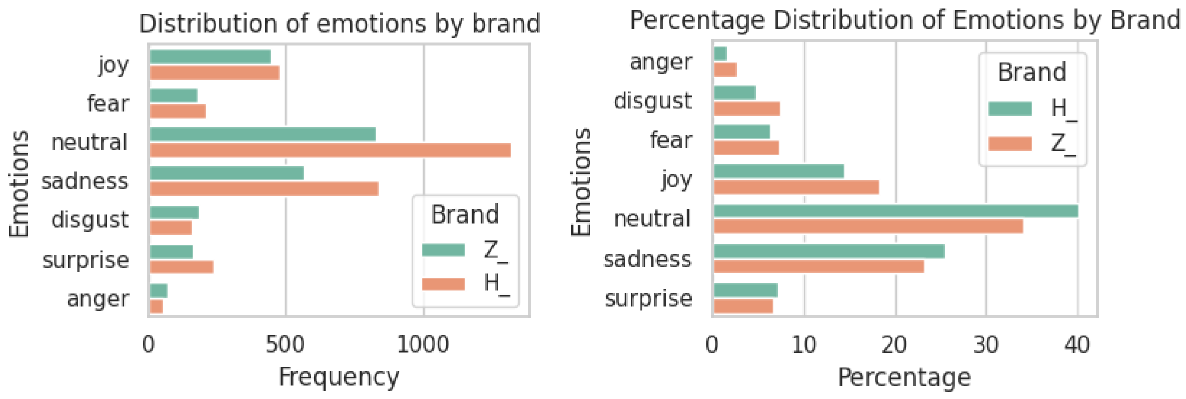


Figure 8. Emotions distributions for each brand

Bar charts illustrates a comparative analysis of the emotional responses for the two brands, based on frequency and percentage. For both brands, "neutral" is the predominant emotion, which might suggest either a general sense of impartiality among customers or potentially poor model performance in predicting nuanced emotions. Following neutral, "sadness" is the second most frequent emotion, comprising 25.55% of the reviews for Brand H_ and 23.25% for Brand Z_.

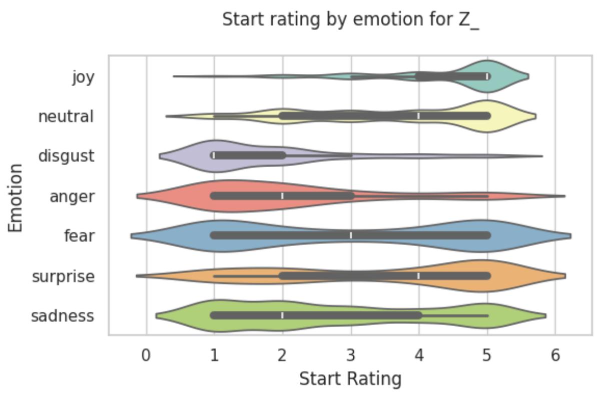


*Figure 9. Start rating by emotion for Z_*

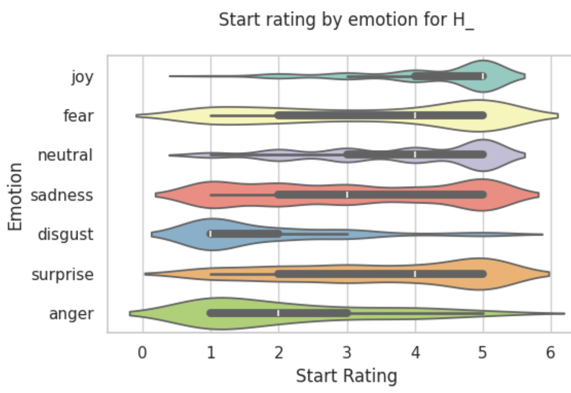


*Figure 10. Start rating by emotion for H_*

Overall, most of the reviews for both brands have a 5-star rating. Predictions suggest that higher star ratings are commonly associated with emotions such as joy, neutral, and surprise. This indicates that reviews labelled with surprise are generally positive. Conversely, the presence of negative emotions like fear or sadness, even in reviews with a 5-star rating, suggests potential areas for improvement in the model's accuracy.

4. Results and Discussion

For the supervised classifiers k-fold cross validation with k= 10 is performed and the following are the comparison of accuracies:

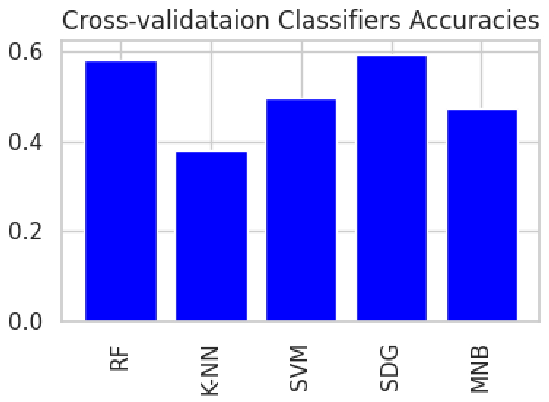


Figure 11. Accuracy comparison supervised models

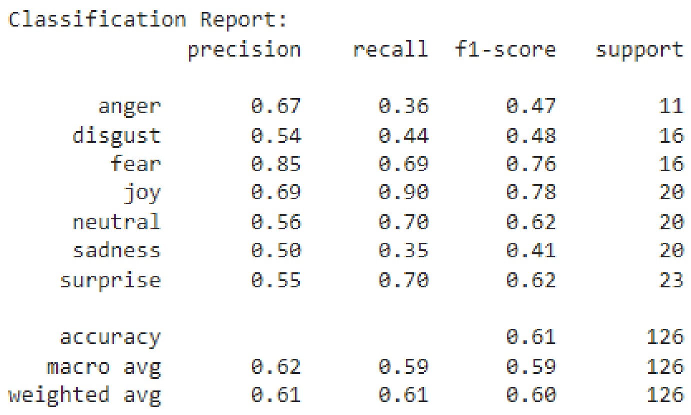
SDG has the best performance in term of accuracy so it is further explained. To explain it some metrics will be consider.

Table 9. Appropriated classification metrics

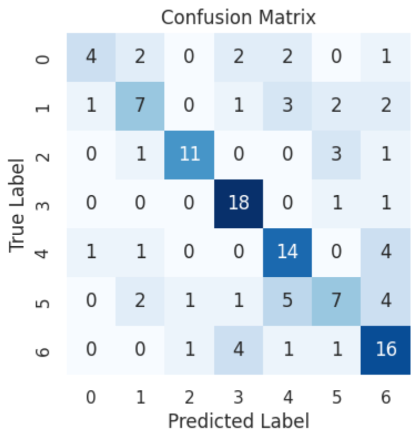
Appropriate metrics	
Confusion matrix	It compares true label with predicted labels.
Precision	It refers to the percentage of correctly identified positive results out of all results classified as positive.
Recall	It measures the proportion of positive patterns that are correctly identified and classified as such.

F1-score	It is the average between precision and recall.

(Hossin and Sulaiman, 2015)



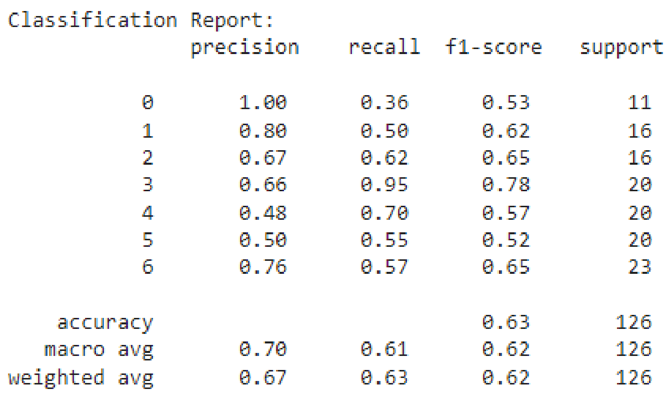
```
Classification Report:
              precision    recall  f1-score   support

       anger       0.67      0.36      0.47        11
     disgust       0.54      0.44      0.48        16
        fear       0.85      0.69      0.76        16
         joy       0.69      0.90      0.78        20
     neutral       0.56      0.70      0.62        20
     sadness       0.50      0.35      0.41        20
    surprise       0.55      0.70      0.62        23

    accuracy                           0.61       126
   macro avg       0.62      0.59      0.59       126
weighted avg       0.61      0.61      0.60       126
```



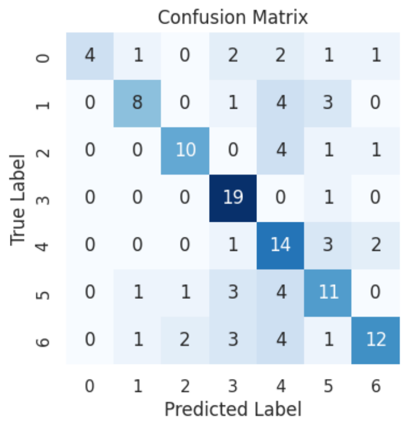
True \ Predicted	0	1	2	3	4	5	6
0	4	2	0	2	2	0	1
1	1	7	0	1	3	2	2
2	0	1	11	0	0	3	1
3	0	0	0	18	0	1	1
4	1	1	0	0	14	0	4
5	0	2	1	1	5	7	4
6	0	0	1	4	1	1	16

Figure 12. SL SDG classification report and confusion matrix

Overall, the model achieves an accuracy of 0.61. It performs best in identifying 'joy', with a precision of 0.69 and a recall of 0.90. This suggests that while the model is quite effective at detecting instances of 'joy' when they occur, it sometimes misclassifies other emotions as 'joy'. Similarly, it detects 'fear' effectively. Conversely, the model struggles with 'sadness', achieving a precision of 0.50 and a recall of 0.35.



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.36      0.53        11
           1       0.80      0.50      0.62        16
           2       0.67      0.62      0.65        16
           3       0.66      0.95      0.78        20
           4       0.48      0.70      0.57        20
           5       0.50      0.55      0.52        20
           6       0.76      0.57      0.65        23

    accuracy                           0.63       126
   macro avg       0.70      0.61      0.62       126
weighted avg       0.67      0.63      0.62       126
```



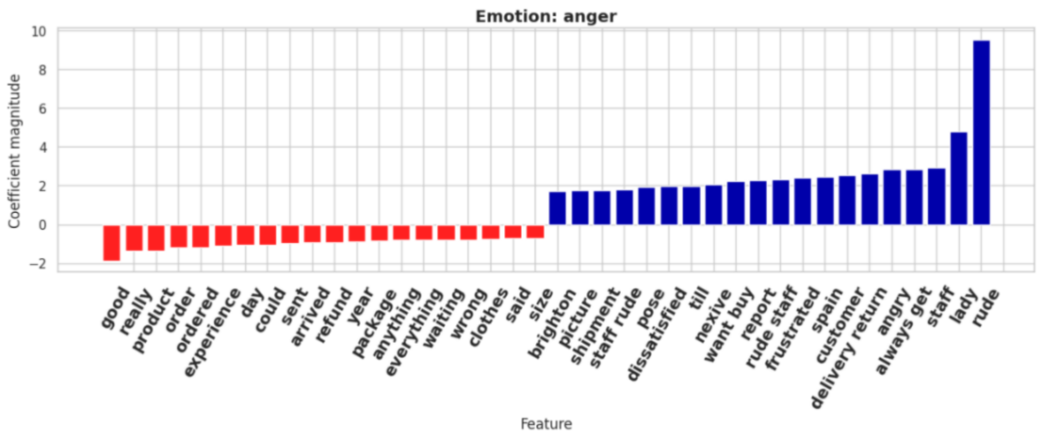
True \ Predicted	0	1	2	3	4	5	6
0	4	1	0	2	2	1	1
1	0	8	0	1	4	3	0
2	0	0	10	0	4	1	1
3	0	0	0	19	0	1	0
4	0	0	0	1	14	3	2
5	0	1	1	3	4	11	0
6	0	1	2	3	4	1	12

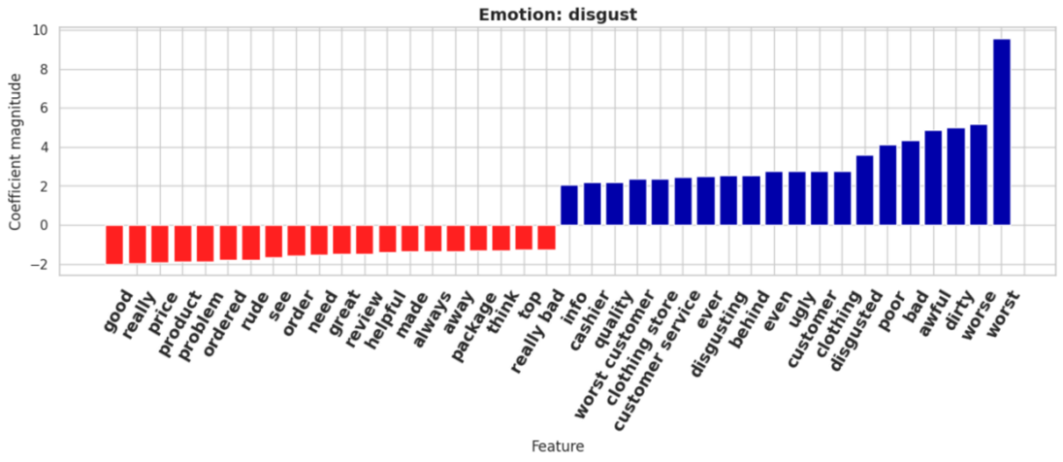
Figure 13. SSL SDG classification report and confusion matrix

Same that for supervised learning, class 3 (joy) demonstrates the highest performance, with a precision of 0.66 and a recall of 0.95, suggesting that the model is particularly effective in identifying this class. In contrast, Class 0 (anger) demonstrates perfect precision at 1.00 but a low recall of 0.36. This indicates that while all predictions made for Class 0 are correct, a significant number of actual Class 0 instances are not predicted by the model. This is confirmed by the confusion matrix. The overall accuracy of the model is 0.63, indicating a moderate performance. However, considering that there are 7 classes, the random chance of correctly predicting a class is approximately 0.14 (1/7). Moreover, the SSL with a low portion of labelled data enhances the performance of the supervised.

In terms of the logistic regression models, the most recent version of scikit-learn uses the 'lbfgs' solver by default for multiclass classification problems (scikit, no date). 'lbfgs' uses a reference class for comparison with the rest classes, where positive coefficients increase the probability of a given class relative to this reference class. Therefore, in the analysis, it is particularly important to focus on the positive coefficients. These coefficients highlight words that are significant in predicting a particular emotion.

-. Brand Z_:

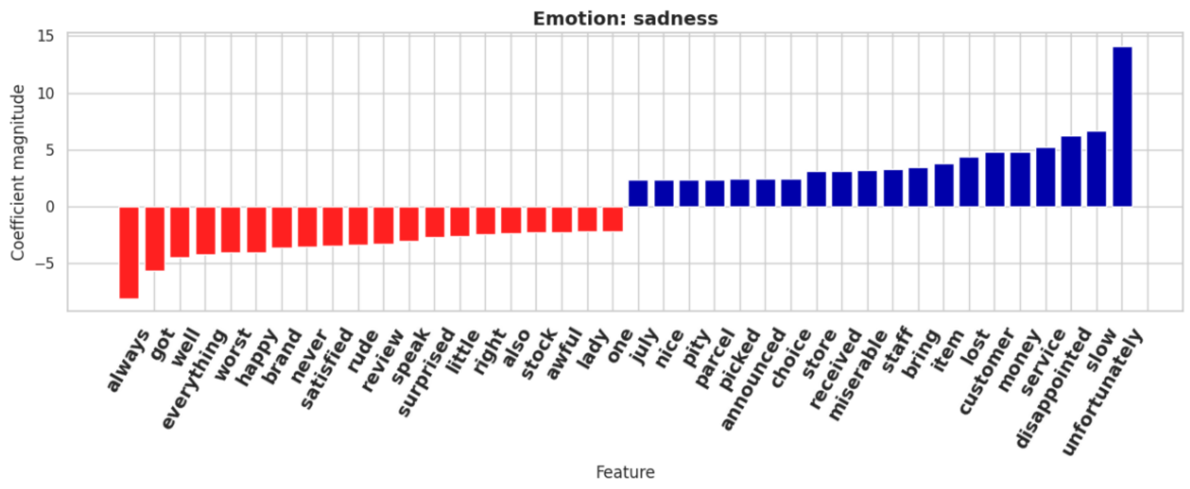


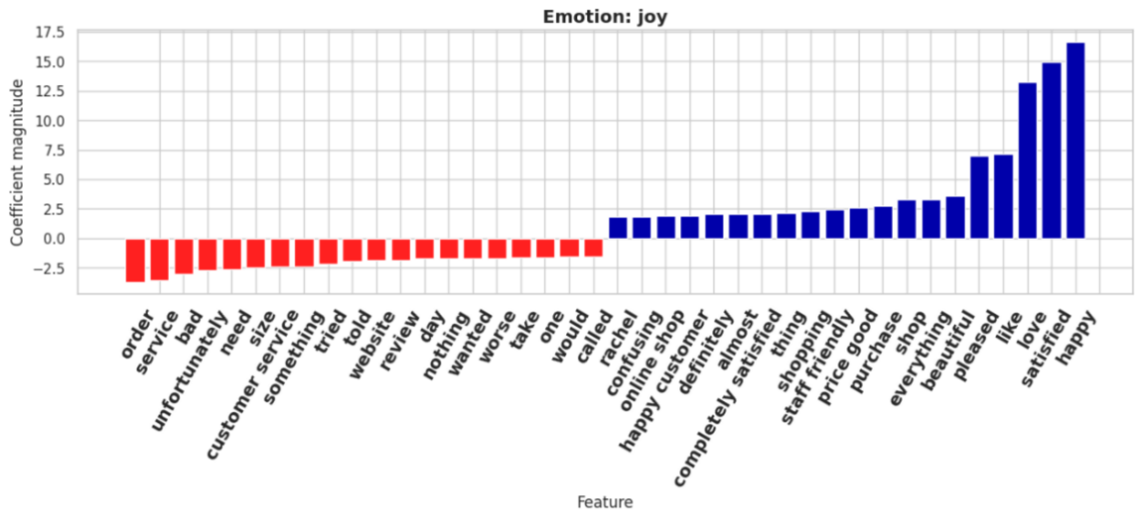


**Only these two levels are shown in this section. To increase report quality rest of the graph are shown in the Appendix.

Regarding, Brand Z_, words like “staff” and “rude staff” are strongly associated with the predictions of anger, disgust and sadness. Additionally, terms such as "worst customer" and "customer service" frequently appear in disgust, sadness and fear. Moreover, expressions related to returns are highly significant for fear and anger emotions. In the case of sadness, which is the second most predicted emotion, there are words such as “slow” or “slow delivery”.

-. Brand H_:





For Brand H_, in the context of negative emotions: for anger, words such as "rude," "angry," and "lazy" appear, disgust is influenced by harsh descriptors like "dirty" or "poor" probably indicating dissatisfaction with aspects related to product quality and other terms such as “worst customer”, “service even” or "staff”. Sadness, similar as brand Z_, is marked by words such as by "service", “staff” and "slow," illustrating scenarios where services or deliveries fell below customer expectations. In the context of positive emotions, for joy emotion “price good” accounts as a good review.

5. Conclusion and Recommendations

Based on the analysis some actionable insights and recommendation can be given to the brands. Firstly, for brand Z_ it is evident that there is substantial room for improvement in the areas of customer service, delivery processes, and return mechanisms. Secondly, Brand Z_ needs to consider an improvement in customer service, delivery processes and product quality, while the currently products’ price seems to be valued by the customers.

Customer service can be improved by the development of training programs for customer service teams to improve interaction quality and response times. For delivery process, logistics can be analysed to detect how to reduce delivery times. Moreover, it is recommended to simplify the return process by making it easier. These are general recommendations, if more information needs to be extracted it can be created a way to ask customers about these processes to obtain more inputs from them.

For future work, it is suggested to explore other semi-supervised techniques that could potentially provide better insights and performance. For instance, advanced neural network models may offer better performance. Using a convolutional neural network with a ladder network in the context of semi-supervised learning (SSL) and image classification, Liu et al. (2017) achieved an accuracy of 97.6% when labelling the unlabelled data.

6. References

1) *1.5. stochastic gradient descent* (no date) *scikit*. Available at: https://scikit-learn.org/stable/modules/sgd.html (Accessed: 04 April 2024).
2) Bird, S., Klein, E. and Loper, E. (2009) Natural language processing with python. Bejing: O'Reilly.
3) Chai, C.P. (2022) 'Comparison of text preprocessing methods', Natural Language Engineering, 29(3), pp. 509–553. doi:10.1017/s1351324922000213. Available at: https://www.cambridge.org/core/journals/natural-language-engineering/article/comparison-of-text-preprocessing-methods/43A20821D65F1C0C4366B126FC794AE3 (Accessed: 04 April 2024)
4) Duarte, J.M. and Berton, L. (2023) 'A review of semi-supervised learning for text classification', Artificial Intelligence Review, 56(9), pp. 9401–9469. doi:10.1007/s10462-023-10393-8. Available at: https://link.springer.com/article/10.1007/s10462-023-10393-8 (Accessed: 20 April 2024)
5) Gurcan, F. (2018) 'Multi-class classification of Turkish texts with machine learning algorithms', 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) [Preprint]. doi:10.1109/ismsit.2018.8567307. Available at: https://ieeexplore.ieee.org/abstract/document/8567307 (Accessed: 10 April 2024)
6) Halim, E., Wardhana, G. and Sasongko, A.H. (2022) 'Online customer reviwes as a marketing tool to generate customer purchase intention in Ecommerce', 2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom) [Preprint]. doi:10.1109/cyberneticscom55287.2022.9865598. Available at: https://ieeexplore.ieee.org/abstract/document/9865598 Accessed: 04 April 2024
7) Hearst, M.A. et al. (2002) 'Support Vector Machines', IEEE Xplore , 13(4), pp. 18–28. doi:10.1142/9789812776655_0002. Available at: https://ieeexplore.ieee.org/document/708428/citations#citations Accessed: 10 April 2024
8) Hintze, J.L. and Nelson, R.D. (1998) 'Violin plots: A box plot-density trace synergism', The American Statistician, 52(2), p. 181. doi:10.2307/2685478. Available at: https://www.tandfonline.com/doi/abs/10.1080/00031305.1998.10480559 Accessed: 20 April 2024.
9) Hossin, M. and Sulaiman, M.N. (2015) 'A review on evaluation metrics for data classification evaluations', International Journal of Data Mining & Knowledge Management Process, 5(2), pp. 1–10. doi:10.5121/ijdkp.2015.5201. Available at: https://www.researchgate.net/publication/275224157_A_Review_on_Evaluation_Metrics_for_Data_Classification_Evaluations . (Accessed: 10 April 2024)
10) Joshi, S. and Abdelfattah, E. (2021) 'Multi-class text classification using machine learning models for online drug reviews', 2021 IEEE World AI IoT Congress (AIIoT) [Preprint]. doi:10.1109/aiiot52608.2021.9454250. Available at: https://ieeexplore.ieee.org/abstract/document/9454250 (Accessed: 20 April 2024)
11) Kouiroukidis, N. and Evangelidis, G. (2011) 'The effects of dimensionality curse in high dimensional knn search', *2011 15th Panhellenic Conference on Informatics* [Online]. doi:10.1109/pci.2011.45. Available at:

https://ieeexplore.ieee.org/abstract/document/6065061?casa_token=ChWutua4R64AAAAA:2aYB4x61I8a1Y5wqxv5-ME7nJ_xWsd9SToeYrIB8cgzVBMkeTCQTNaVbAdvsNRGvhG1opdfC Accessed: 20 April 2024

12) Ligthart, A., Catal, C. and Tekinerdogan, B. (2021) 'Analyzing the effectiveness of semi-supervised learning approaches for opinion spam classification', Applied Soft Computing, 101, p. 107023. doi:10.1016/j.asoc.2020.107023. Available at: https://www.sciencedirect.com/science/article/pii/S1568494620309625 Accessed: 05 April 2024

13) Liu, B. et al. (2017) 'A semi-supervised Convolutional Neural Network for Hyperspectral Image Classification', Remote Sensing Letters, 8(9), pp. 839–848. doi:10.1080/2150704x.2017.1331053. Available at: https://www.tandfonline.com/doi/full/10.1080/2150704X.2017.1331053 Accessed: 29 April 2024

14) McCallum, A. and Nigam, K., 1998. A Comparison of Event Models for Naive Bayes Text Classification. AAAI-98 Workshop on Learning for Text Categorization. Available at: http://yangli-feasibility.com/home/classes/lfd2022fall/media/aaaiws98.pdf (Accessed: 14 April 2024)

15) Müller, A.C. and Guido, S. (2016) Introduction to machine learning with python. Nanjing: O'REILLY.

16) Pimpalkar, A.P. and Retna Raj, R.J. (2020) 'Influence of Pre-processing Strategies on the Performance of ML Classifiers Exploiting TF-IDF and BOW Features', ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, 9(2), pp. 49–68. doi:10.14201/adcaij.2023121. Available at: https://www.torrossa.com/en/resources/an/5010980# (Accessed: 07 April 2024.)

17) Rigatti, S.J. (2017) 'Random Forest', Journal of Insurance Medicine, 47(1), pp. 31–39. doi:10.17849/insm-47-01-31-39.1. Available at: https://meridian.allenpress.com/jim/article/47/1/31/131479/Random-Forest (Accessed: 10 April 2024)

18) scikit (no date) Sklearn.linear_model.logisticregression. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (Accessed: 28 April 2024).

19) scikit (no date) Sklearn.linear_model.Sgdclassifier. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html (Accessed: 29 April 2024).

20) Singla, Z., Randhawa, S. and Jain, S. (2017) 'Sentiment analysis of customer product reviews using Machine Learning', 2017 International Conference on Intelligent Computing and Control (I2C2) [online]. doi:10.1109/i2c2.2017.8321910. Available at: https://ieeexplore.ieee.org/abstract/document/8321910 (Accessed: 11 April 2024).

21) *Sklearn.naive_bayes.multinomialnb* (no date) *scikit*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html (Accessed: 24 April 2024).

22) Van Engelen, J.E. and Hoos, H.H., 2020. A survey on semi-supervised learning. Springer, 109(1), pp.373-440. https://doi.org/10.1007/s10994-019-05855-6 Available at: https://link.springer.com/article/10.1007/s10994-019-05855-6 (Accessed 20 April 2024)

23) Vijayarani, S., Ilamathi, J. and Nithya. (2015). Preprocessing techniques for text mining - an overview. International Journal of Computer Science & Communication Networks, 5(1), 7-16. ISSN: 2249-5789. Available at: https://www.researchgate.net/profile/Vijayarani-Mohan/publication/339529230_Preprocessing_Techniques_for_Text_Mining_-_An_Overview/links/5e57a0f7299bf1bdb83e7505/Preprocessing-Techniques-for-Text-Mining-An-Overview.pdf (Accessed: 11 April 2024)
24) Widmann, N. and Verberne, S. (2017) 'Graph-based semi-supervised learning for text classification', Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval [Preprint]. doi:10.1145/3121050.3121055. Available at: https://dl.acm.org/doi/abs/10.1145/3121050.3121055 (Accessed: 04 April 2024)
25) Wirth, R. and Hipp, J. (2000) *CRISP-DM: Towards a Standard Process Model for Data Mining* [Online]. Available at: https://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf (Accessed: 10 April 2024).
26) Zhang, Z. (2016) 'Introduction to machine learning: K-Nearest Neighbors', Annals of Translational Medicine, 4(11), pp. 218–218. doi:10.21037/atm.2016.03.37. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4916348/ (Accessed: 10 April 2024)
27) Zhu, X. (2005) 'Semi-supervised Learning Literature Survey', Computer Sciences Department. Available at: https://minds.wisconsin.edu/handle/1793/60444 (Accessed: 04 April 2024).

7. Python Code

```
# Load some neccesary libraries
import pandas as pd
import pandas as pd
import numpy as np
import re
import seaborn as sns
import matplotlib.pyplot as plt

import nltk
nltk.download('punkt')
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
from nltk.tokenize import word_tokenize

from nltk.stem import PorterStemmer

from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.metrics import f1_score, confusion_matrix, accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import FunctionTransformer
from sklearn.semi_supervised import SelfTrainingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC


# Conect with drive
from google.colab import drive
drive.mount('/content/drive')

# load original file
path = '/content/drive/MyDrive/A_II_Emotion_Data_Student_Copy_Final.xlsx'
data1 = pd.read_excel(path)
```

```python
data1.info()

# %% [markdown]
# Data preprocessing

# library to detect languages
pip install langdetect


# %%
#remove special characteres from the reviews
def clean_text1(text):
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'<[^>]+>', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text

# apply previous function
data1['text_reviews_1'] = data1['text_reviews_'].apply(clean_text1)

# %%
#detecte languages in the reviews
from langdetect import detect, LangDetectException

def detectar_idioma(texto):
    try:
        return detect(texto)
    except LangDetectException:
        return "unknown"

data1["language"] = data1['text_reviews_1'].apply(detectar_idioma)
df_no_ingles = data1[data1['language'] != 'en']


# total number of english reviews
total_en = (data1["language"] == "en").sum()

# print
total_en
```

```
# %%
#plot distribution of languages
language_counts = data1['language'].value_counts()

plt.figure(figsize=(4, 2))
plt.bar(language_counts.index, language_counts.values, color='skyblue', edgecolor='black')
plt.xlabel('Language')
plt.ylabel('Number of Reviews')
plt.title('Language Distribution Accross the reviews')
plt.xticks(rotation=90, fontweight='bold')
plt.show()

# %%
#delete from the data set with no english reviews those reviews from GB to reduce the size of it
df_no_ingles1 = df_no_ingles[df_no_ingles['country_'] != 'GB']

# %%
#calculate the lenght of the reviews
df_no_ingles1['longitud'] = df_no_ingles1['text_reviews_1'].str.len()

total_caracteres = df_no_ingles1['longitud'].sum()

# %%
total_caracteres

# %%
# API deepl only allows 500000 for free, so the data set is divide and Baidu is also used
df_no_ingles1['length'] = df_no_ingles1['text_reviews_1'].str.len()
df_no_ingles1['cumulative_length'] = df_no_ingles1['length'].cumsum()

# Filter the dataset
df_filtered = df_no_ingles1[df_no_ingles1['cumulative_length'] <= 500000]
df_filtered2 = df_no_ingles1[df_no_ingles1['cumulative_length'] > 500000]

#Translate the others 400000 characteres
import requests
from hashlib import md5
import random
import json
appid = '234509250023333'
```

```python
appkey = 'BRdzUuBl3BBAMC9es1d8'

def make_md5(s, encoding='utf-8'):
    return md5(s.encode(encoding)).hexdigest()

def baidu_api(query, from_lang='auto', to_lang='en'):
    endpoint = 'http://api.fanyi.baidu.com'
    path = '/api/trans/vip/translate'
    url = f"{endpoint}{path}"
    salt = random.randint(32768, 65536)
    sign = make_md5(appid + query + str(salt) + appkey)
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}
    payload = {
        'appid': appid,
        'q': query,
        'from': from_lang,
        'to': to_lang,
        'salt': salt,
        'sign': sign
    }

    try:
        r = requests.post(url, params=payload, headers=headers, timeout=10)
        result = r.json()
        if 'error_code' in result:
            print(f"Error {result['error_code']}: {result.get('error_msg', 'No error message')}")
            return query
        return result["trans_result"][0]['dst']
    except Exception as e:
        print(f"Error during translation: {e}")
        return query

def translate_text(text):
    return baidu_api(text, from_lang='auto', to_lang='en')

df_filtered2['translated_text'] = df_filtered2['text_reviews_1'].apply(translate_text)


# %%
#Translate the first 500000 characteres
```

```
import deepl

translator = deepl.Translator("ee87c8a8-fe26-4956-a1d6-bbe41f8590f1:fx")

def translate_text(text):
    try:
        result = translator.translate_text(text, target_lang="EN-GB")
        return result.text
    except Exception as e:
        print(f"Error al traducir el texto: {e}")
        return text

df_filtered['translated_text'] = df_filtered['text_reviews_1'].apply(translate_text)


# %%
#adding the Deepl translated reviews to the original data set
result_df = pd.merge(data, df_filtered[['ID_', 'translated_text']], on='ID_', how='left')

result_df['text_reviews_'] = result_df['translated_text'].combine_first(result_df['text_reviews_'])

result_df.drop(['translated_text','text_reviews_1', 'language'], axis=1, inplace=True)

# %%
#adding the Baidu translated reviews to the previous data set
merged_df = pd.merge(result_df, df_filtered2[['ID_', 'translated_text']], on='ID_', how='left')

merged_df['text_reviews_'] = merged_df['translated_text'].combine_first(merged_df['text_reviews_'])

final_df = merged_df.drop('translated_text', axis=1)

# %%
final_df.drop('Cleaned_reviews', axis=1, inplace=True)

# this data set contains same information as the original provided but with all the reviews in English
final_df.to_csv('/content/drive/MyDrive/final_reviews.csv', index=False)

# %% [markdown]
# New file with all the reviews in English
# load
path = '/content/drive/MyDrive/final_reviews.csv'
data = pd.read_csv(path)


data.info()

#remove 2 missing values originated when the reviews were translated
data = data.dropna(subset=['text_reviews_'])


# %%
#plot language distribution accross reviews
from langdetect import detect, LangDetectException

def detectar_idioma(texto):
    try:
        return detect(texto)
    except LangDetectException:
        return "unknown"

to_plot = data.copy()

to_plot["language"] = to_plot['text_reviews_'].apply(detectar_idioma)

language_counts = to_plot['language'].value_counts()

plt.figure(figsize=(6, 4))
plt.bar(language_counts.index, language_counts.values, color='skyblue', edgecolor='black')
plt.xlabel('Language')
plt.ylabel('Number of Reviews')
plt.title('Language Distribution Accross the translated reviews')
plt.xticks(rotation=90)
plt.show()

## Text Data Pre-Processing


# function to do text pre-processing
def clean_text(text):
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'<[^>]+>', '', text)  # Remove HTML tags
    text = text.lower()  # Lowercase text
    text = re.sub(r'\b\w{1,2}\b', '', text)  # Remove words with 1 or 2 letters
    text = re.sub(r'[^a-z\s]', '', text)  # Keep text with letters and spaces

    # Tokenize
    tokens = word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatize 0.56 accuracy
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Stemming 0.54 accuracy
    #stemmer = PorterStemmer()
    #tokens = [stemmer.stem(word) for word in tokens]

    return ' '.join(tokens)

# new column is created with the clean text
data['Cleaned_reviews'] = data['text_reviews_'].apply(clean_text)

## Initial descriptive analysis


sns.set(style="whitegrid")
plt.figure(figsize=(8, 8))

top_countries = data['country_'].value_counts().head(15).index
# Filtering the data to include only the top 15 countries
filtered_data = data[data['country_'].isin(top_countries)]
missing_emotions = data['emotions_'].isnull().sum()

# Adjusting the categorical columns to use the filtered data for country_
categorical_columns = ['brand_name_', 'country_', 'emotions_']
numerical_columns = ['star_rating_']
```

```
#4 subplots
rows, cols = 2, 2

#loop to iterate in the variables
for index, column in enumerate(categorical_columns):
    plt.subplot(rows, cols, index + 1)
    #take country from the data set where there are only top 15 countries
    if column == 'country_':
        sns.countplot(y=column, data=filtered_data, palette="Set2")
    #take the rest of the varaibles from the original dataset
    else:
        count_plot = sns.countplot(y=column, data=data, palette="Set2")
    plt.title(f'{column}', fontweight='bold')
    plt.tight_layout()

    #add the emotions to the missing values in the bottom left-hand corner
    if column == 'emotions_':
        plt.text(count_plot.get_xlim()[1], count_plot.get_ylim()[0], f'Missing Values: {missing_emotions}',
                 horizontalalignment='left', verticalalignment='top',
                 fontsize=12, color='red')

#plot
plt.subplot(rows, cols, len(categorical_columns) + 1)
sns.violinplot(y='star_rating_', data=data, color="skyblue")
plt.title('star_rating_', fontweight='bold')
plt.tight_layout()

plt.show()

## Machine Learning

# Unlabeled data

# data frame is created with missing observations of emotions (Unlabeled data) and Cleaned reviwes is included
in the data frame.
unlabeled_data = data[pd.isna(data['emotions_'])][['Cleaned_reviews']]
unlabeled_data['emotions_'] = -1

# Labeled Data
# %%
# data frame is created with known observations of emotions (Labeled data) and Cleaned reviwes is included in
the data frame.
labeled_data = data[data['emotions_'].notna()][['Cleaned_reviews', 'emotions_']]

# Create X and y
X = data['Cleaned_reviews']
y = data['emotions_']

# known emotions
y_labeled = labeled_data['emotions_']
# missing emotions
y_unlabeled = unlabeled_data['emotions_']
# cleaned reviews for known emotions
X_labeled = labeled_data['Cleaned_reviews']
# cleaned reviews for missing emotions
X_unlabeled = unlabeled_data['Cleaned_reviews']

# Function for calssification report
def eval_and_print_metrics(clf, X_train, y_train, X_test, y_test):
    print("Number of training samples:", len(X_train))
    print("Unlabeled samples in training set:", sum(1 for x in y_train if x == -1)) #if x == 'NaN'
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    print(
        "Micro-averaged F1 score on test set: %0.3f"
        % f1_score(y_test, y_pred, average="micro")
    )
    print("\nConfusion Matrix:\n", confusion_matrix(y_test,y_pred))
    print("\nClassification Report:\n", classification_report(y_test, y_pred,zero_division=1))
    print("\n\n")


# Split the labeled data
X_train, X_test, y_train, y_test = train_test_split(X_labeled, y_labeled, test_size=0.2, stratify=y_labeled,
random_state=42)


from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV


# set TfidfVectorizer parameter using a logistic regression

pip install mglearn

import mglearn

pipe1 = make_pipeline(TfidfVectorizer(), LogisticRegression(max_iter=10000))
#Source: (Bird, S., Klein, E. and Loper, E. (2009) Natural language processing with python. Bejing: O'Reilly.)

#set parameter of logistic reression (C), based on previous calculation it is known that we required a high value
for C
param_grid1 = {
   "logisticregression__C": [10, 100],
   "tfidfvectorizer__ngram_range": [(1, 1), (1, 2), (1, 3)],
   "tfidfvectorizer__max_features": [1000, 2000, 3000, 5000]
}

grid1 = GridSearchCV(pipe1, param_grid1, cv=5)
grid1.fit(X_train, y_train)

scores = grid1.cv_results_['mean_test_score'].reshape(len(param_grid1["logisticregression__C"]),
                                                      len(param_grid1["tfidfvectorizer__ngram_range"]),
                                                      len(param_grid1["tfidfvectorizer__max_features"]))
scores_mean = scores.mean(axis=0)

heatmap = mglearn.tools.heatmap(scores_mean, xlabel="max_features", ylabel="ngram_range", cmap="viridis",
fmt="%.3f",
                               xticklabels=param_grid1["tfidfvectorizer__max_features"],
                               yticklabels=param_grid1["tfidfvectorizer__ngram_range"])

plt.colorbar(heatmap)
plt.show()

# best ngram (1,2)
# max_features 3000
#
```

```
## Supervised Machine learning** model for labelated data


from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB

vectorizer_params = {
    'ngram_range': (1, 2),
    'max_features': 3000
}

#hyper parameter tuning

param_grid = {
    "Random Forest": {
        "clf__n_estimators": [50, 100, 200],
        "clf__max_depth": [None, 10, 20, 30],
        "clf__min_samples_split": [2, 5, 10]
    },
    "K-Nearest Neighbors": {
        "clf__n_neighbors": [3, 5, 6, 7],
        "clf__weights": ["uniform", "distance"],
        "clf__algorithm": ["auto", "ball_tree"]
    },
    "Support Vector Machine": {
        "clf__C": [0.1, 1, 10],
        "clf__kernel": ["linear", "rbf", "poly"]
    },
    "SDG": {
        "clf__alpha": [1e-5, 1e-4, 1e-3],
        "clf__penalty": ["l2", "l1"],
        "clf__loss": ["hinge", "log"]
    },
    "Naive Bayes": {
        "clf__alpha": [0.01, 0.1, 1.0]
    }
}
```

```
#classifiers
supervised_classifiers = [
    ("Random Forest", RandomForestClassifier(), param_grid["Random Forest"]),
    ("K-Nearest Neighbors", KNeighborsClassifier(), param_grid["K-Nearest Neighbors"]),
    ("Support Vector Machine", SVC(), param_grid["Support Vector Machine"]),
    ("SDG", SGDClassifier(), param_grid["SDG"]),
    ("Naive Bayes", MultinomialNB(), param_grid["Naive Bayes"])
]



classifier_names = []
accuracy_scores = []

#lop through the classifiers and parameters, saving best accuracues and names in the lists created above
for name, clf, params in supervised_classifiers:
    pipeline = Pipeline([
        ("vect", TfidfVectorizer(**vectorizer_params)),
        ("clf", clf),
    ])
    grid_search = GridSearchCV(pipeline, param_grid=params, n_jobs=-1, cv=10)
    grid_search.fit(X_labeled, y_labeled)
    best_clf = grid_search.best_estimator_
    best_params = grid_search.best_params_
    best_score = grid_search.best_score_

    # Print best hyperparameters
    print(f"Best Hyperparameters for {name}: {best_params}")

    # Evaluate and print metrics
    eval_and_print_metrics(best_clf, X_train, y_train, X_test, y_test)

    # Store the name and the accuracy for plotting
    classifier_names.append(name)
    accuracy_scores.append(best_score)


# change classifers names for graph clarity
classifier_names = ["RF", "K-NN", "SVM", "SDG", "MNB"]
```

```
# Plotting the accuracies
plt.figure(figsize=(4, 3))
plt.bar(classifier_names, accuracy_scores, color='blue')
plt.title('Cross-validataion Classifiers Accuracies')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()


# build model with the best model using the best hyper parameters
best_params_sdg = { 'alpha': 0.001,'loss': 'log',
    'penalty': 'l1'}

pipeline_sdg = Pipeline([
    ("vect", TfidfVectorizer(**vectorizer_params)),
    ("clf", SGDClassifier(**best_params_sdg))
])

pipeline_sdg.fit(X_train, y_train)

y_pred1 = pipeline_sdg.predict(X_test)

#cm
cm = confusion_matrix(y_test, y_pred1)
plt.figure(figsize=(4,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

##Semi-supervised calssifiers

#mantain x tests
test_indices = X_test.index

# Exclude test data from X_labeled and y_labeled based on the identified indices
X_labeled_filtered = X_labeled.drop(index=test_indices, errors='ignore')
y_labeled_filtered = y_labeled.drop(index=test_indices, errors='ignore')
```

```python
# Concatenate the filtered labeled data with the unlabeled data
# lop que se esta haciendo es extraer de X labeled e y labeled el test data test (me quedo con el train)
# se junta train labeled con toda la unlabeled por lo tanto tengo: X = x train label + all unlabeled. Y = y train labeled + all unlabeled
# x_test y y_test es todo labeled data para comprobar accuracy de las predicciones
X=X_combined = pd.concat([X_labeled_filtered, X_unlabeled])
y=y_combined = pd.concat([y_labeled_filtered, y_unlabeled])

# Define the mapping for labels
label_mapping = {
    'anger': 0,
    'disgust': 1,
    'fear': 2,
    'joy': 3,
    'neutral': 4,
    'sadness': 5,
    'surprise': 6, -1:-1}

# Apply the mapping to labels
y  = [label_mapping[label] for label in y]
print(y)
y_test  = [label_mapping[label] for label in y_test]
print(y_test)

#selflearning
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.semi_supervised import SelfTrainingClassifier

vectorizer_params = {
    'ngram_range': (1, 2),
    'max_features': 3000
}

random_state_seed = 40425150

# Defining classfiers with the best hyperparameter of supervised
```

```
non_graph_based_classifiers = [
    ("SelfTraining Logistic Regression", SelfTrainingClassifier(LogisticRegression(max_iter=1000),
criterion='threshold', threshold=0.9)),
    ("SelfTraining K-Nearest Neighbors",
SelfTrainingClassifier(base_estimator=KNeighborsClassifier(n_neighbors=7, algorithm='auto',
weights='distance'))),
    ("SelfTraining Naive Bayes", SelfTrainingClassifier(MultinomialNB())),
    ("SelfTraining SVC linear", SelfTrainingClassifier(SVC(probability=True, C=10, gamma='scale', kernel='linear'),
criterion='threshold', threshold=0.9)),
    ("SelfTraining SDG", SelfTrainingClassifier(SGDClassifier(alpha=0.001, loss='log', penalty='l1'), verbose =
True)),
    ("Random Forest", SelfTrainingClassifier(RandomForestClassifier(max_depth=20, min_samples_split=10,
n_estimators=200)))
]

#lop and results of self-learning classifiers
for name, clf in non_graph_based_classifiers:
    print(f"Evaluating {name}")
    st_pipeline = Pipeline([
        ("vect", TfidfVectorizer(**vectorizer_params)),
        ("clf", clf),
    ])
    eval_and_print_metrics(st_pipeline, X, y, X_test, y_test)


# Label spreading
from sklearn.semi_supervised import LabelSpreading

vectorizer_params = {
    'ngram_range': (1, 2),
    'max_features': 3000
}

ls_pipeline = Pipeline(
    [
        ("vect", TfidfVectorizer(**vectorizer_params)),
        ("toarray", FunctionTransformer(lambda x: x.toarray())),
        ("clf", LabelSpreading()),
    ]
)
```

```
ls_pipeline.fit(X, y)

eval_and_print_metrics(ls_pipeline, X, y, X_test, y_test)

# Best performance is SDG


# Report of Self learning SDG
st_svc_pipeline = Pipeline([
    ("vect", TfidfVectorizer(**vectorizer_params)),
    ("SelfTraining SGD", SelfTrainingClassifier(SGDClassifier(alpha=0.001, loss='log', penalty='l1'), verbose =
True))])

st_svc_pipeline.fit(X, y)

y_pred2 = st_svc_pipeline.predict(X_test)

print("Classification Report:")
print(classification_report(y_test, y_pred2))

cm2 = confusion_matrix(y_test, y_pred2)
plt.figure(figsize=(4,4))
sns.heatmap(cm2, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()


# Predicting with SDG the unlabeled data

predictions = st_svc_pipeline.predict(X_unlabeled)

plt.hist(predictions, bins='auto', color='skyblue', edgecolor='black')
plt.title('Distribution of Predictions')
plt.xlabel('Predicted Label')
plt.ylabel('Frequency')
plt.show()

# inverse mapping
```

```
data1 = data.copy()

inverse_label_mapping = {v: k for k, v in label_mapping.items()}

text_predictions = np.array([inverse_label_mapping[pred] for pred in predictions])

missing_emotion_indices = data1[data1['emotions_'].isna()].index

assert len(predictions) == len(missing_emotion_indices)

data1.loc[missing_emotion_indices, 'emotions_'] = text_predictions



# plot distribution of emotions after predictions
plt.hist(data1['emotions_'], bins='auto', color='skyblue', edgecolor='black')
figsize=(7, 7)
plt.title('Distribution of Predictions')
plt.xlabel('Predicted Label')
plt.ylabel('Frequency')
plt.xticks(fontweight = "bold")
plt.show()


# Analysis for each BRAND

sns.set(style="whitegrid")

# Emotions for each brand
plt.figure(figsize=(4, 3))
sns.countplot(y='emotions_', hue='brand_name_', data=data1, palette='Set2')
plt.title('Distribution of emotions by brand')
plt.xlabel('Frequency')
plt.ylabel('Emotions')
plt.legend(title='Brand')
plt.tight_layout()
plt.show()

# percentages
total_emotions_per_brand = data1.groupby('brand_name_').size().reset_index(name='total_per_brand')
```

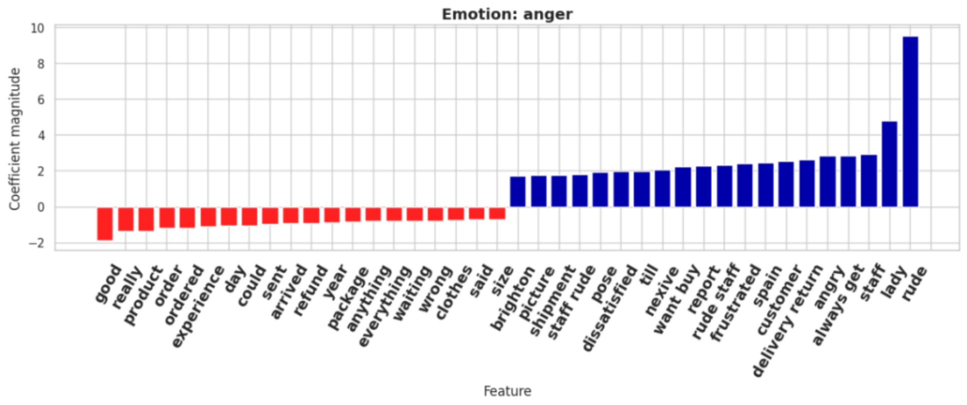
```
# Count each emotion per brand
emotion_counts = data1.groupby(['brand_name_', 'emotions_']).size().reset_index(name='counts')

# Merge the total counts into emotion counts to calculate percentages
emotion_counts = emotion_counts.merge(total_emotions_per_brand, on='brand_name_')

# Calculate percentage
emotion_counts['percentage'] = 100 * emotion_counts['counts'] / emotion_counts['total_per_brand']


sns.set(style="whitegrid")

# Create a bar plot to visualize percentages
plt.figure(figsize=(4, 3))
sns.barplot(x='percentage', y='emotions_', hue='brand_name_', data=emotion_counts, palette='Set2')
plt.title('Percentage Distribution of Emotions by Brand')
plt.xlabel('Percentage')
plt.ylabel('Emotions')
plt.legend(title='Brand')
plt.tight_layout()
plt.show()



# to identify the 2 brands
brands = data1['brand_name_'].unique()

#lop to iterate based on the 2 brands availables  to plot star rating for each brand for each emotion
for brand in brands:
   # Filter the dataset for the current brand
  brand_data = data1[data1['brand_name_'] == brand]
  plt.figure(figsize=(6, 4))
  sns.violinplot(x='star_rating_', y='emotions_', data=brand_data, palette='Set3')
  plt.title(f'Start rating by emotion for {brand}\n')
  plt.xlabel('Start Rating')
  plt.ylabel('Emotion')
  plt.tight_layout()
  plt.show()
```

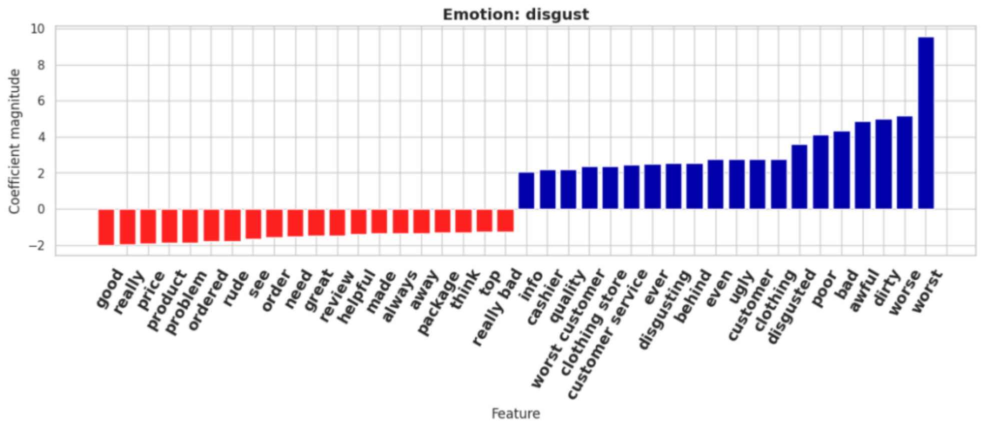
```
# LOGISTIC REGRESSION TO ANALYSE COEFFICIENTS

#Idea and part of the code is taking from: (Müller, A.C. and Guido, S. (2016) Introduction to machine learning with
python. Nanjing: O'REILLY)

def visualize_coefficients_multiclass_modified(coef_matrix, feature_names, class_labels, n_top_features=20):
    n_classes = coef_matrix.shape[0]
    for i in range(n_classes):
        plt.figure(figsize=(8, 4))
        coef = coef_matrix[i, :]
        mglearn.tools.visualize_coefficients(coef, feature_names, n_top_features=n_top_features)
        plt.title(f"Emotion: {class_labels[i]}", fontsize=14, fontweight='bold')
        plt.xticks(fontsize=14, fontweight='bold')
        plt.show()

# Loop through each brand to train and evaluate a model

for brand in brands:
    # Filter the dataset for the current brand
    brand_data = data1[data1['brand_name_'] == brand]


    # Prepare the data
    X_train2, X_test2, y_train2, y_test2 = train_test_split(
        brand_data['Cleaned_reviews'], brand_data['emotions_'], test_size=0.2, random_state=42
    )

    # Create a pipeline with TF-IDF and Logistic Regression
    model = make_pipeline(TfidfVectorizer(**vectorizer_params), LogisticRegression(max_iter=10000, C=10))

    # Train the model
    model.fit(X_train2, y_train2)

    # Evaluate the model
    predictions = model.predict(X_test2)
    print(f"Classification Report for Brand: {brand}\n")
    print(classification_report(y_test2, predictions))

    # Visualization of coefficients
    vect = model.named_steps['tfidfvectorizer']
```

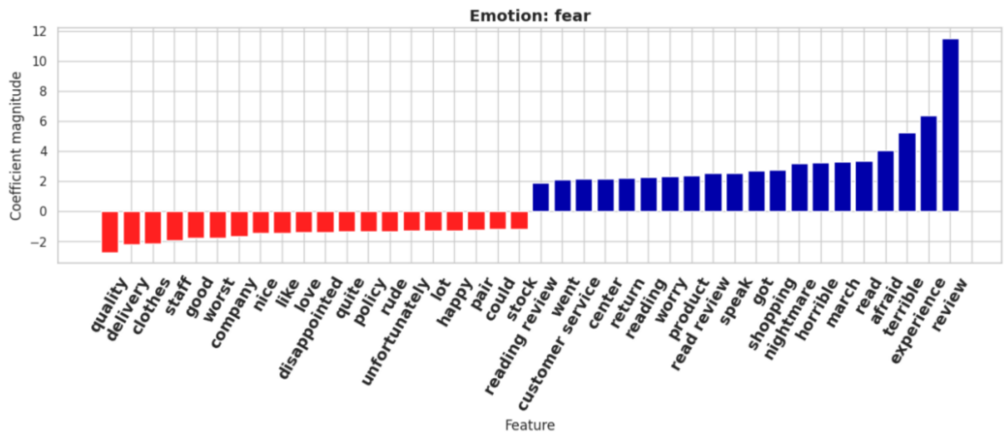
```
features_names = np.array(vect.get_feature_names_out())
coef = model.named_steps['logisticregression'].coef_
class_labels = model.named_steps['logisticregression'].classes_
visualize_coefficients_multiclass_modified(coef, features_names, class_labels, n_top_features=20)
```

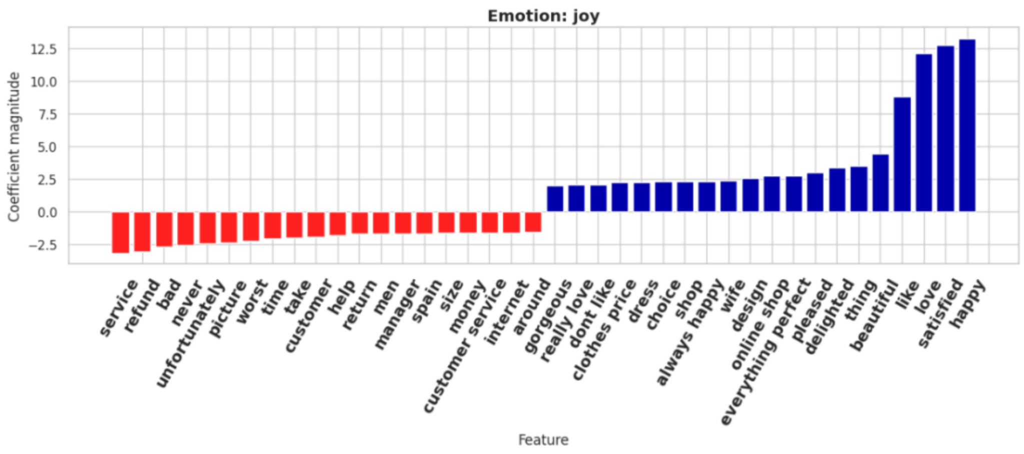
8. Appendix

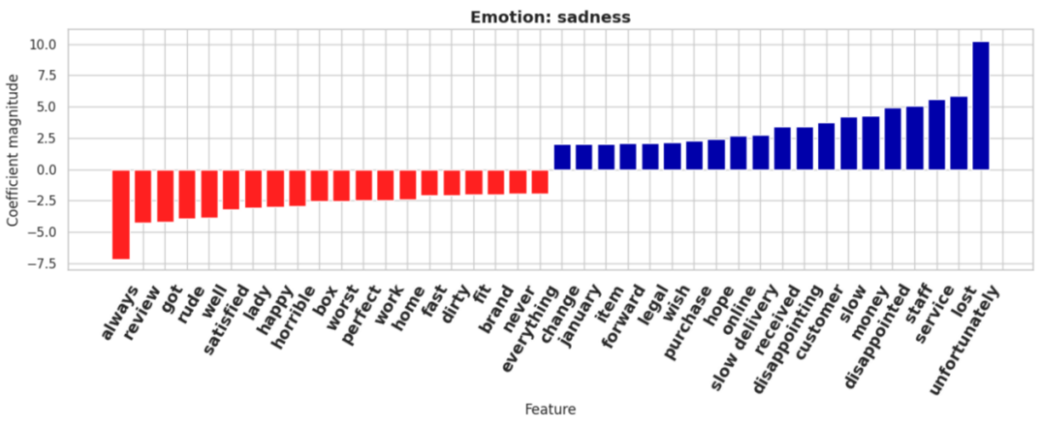
-. Brand Z_:

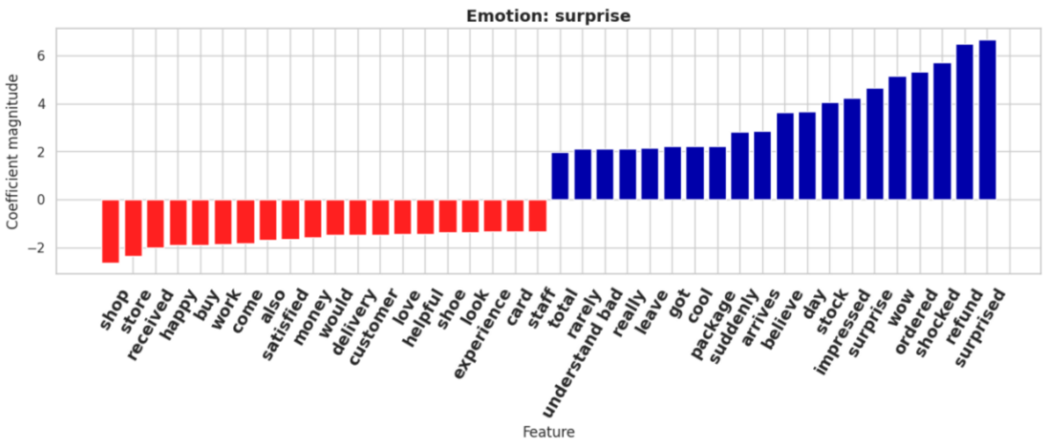


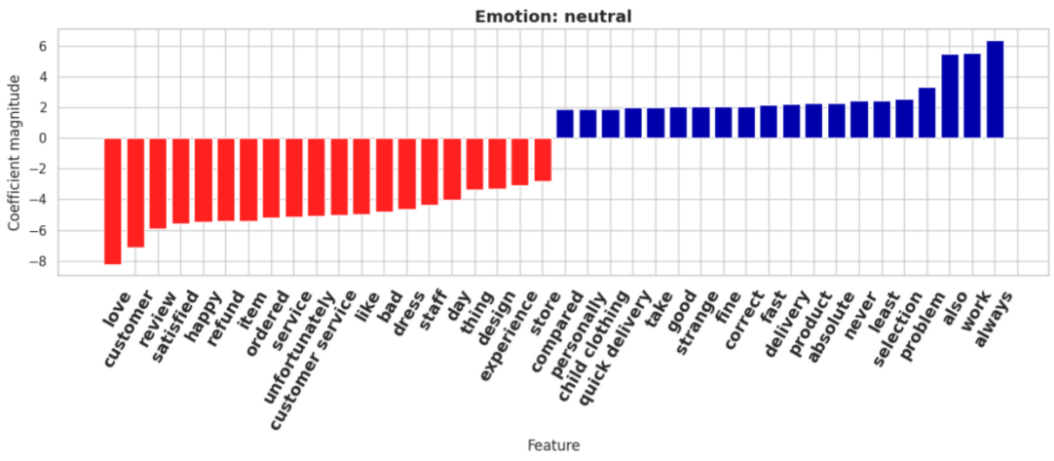



Emotion: fear
Coefficient magnitude
Feature
quality
delivery
clothes
staff
good
worst
company
nice
like
love
disappointed
quite
policy
rude
unfortunately
lot
happy
pair
could
stock
reading review
went
customer service
center
return
reading
worry
product
read review
speak
got
shopping
nightmare
horrible
march
read
afraid
terrible
experience
review

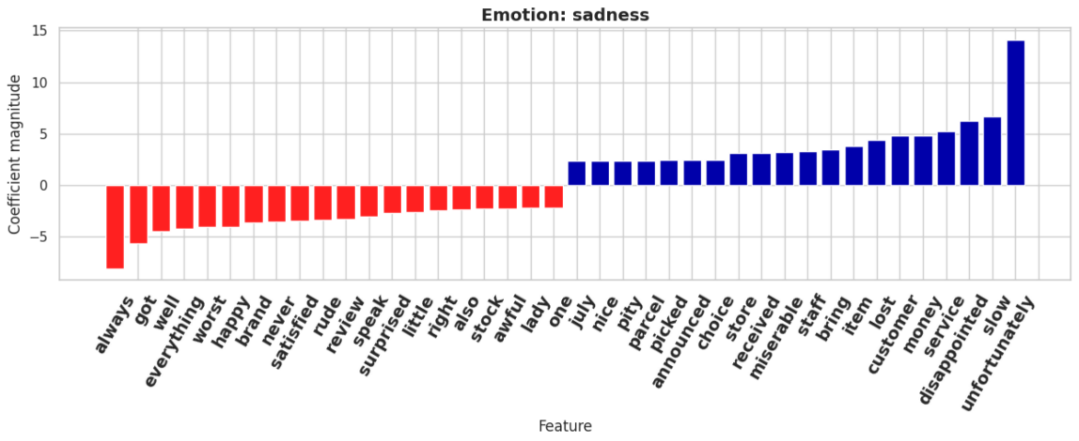

Emotion: joy
Coefficient magnitude
Feature
service
refund
bad
never
unfortunately
picture
worst
time
take
customer
help
return
men
manager
spain
size
money
customer service
internet
around
gorgeous
really love
dont like
clothes price
dress
choice
shop
always happy
wife
design
online shop
everything perfect
pleased
delighted
thing
beautiful
like
love
satisfied
happy

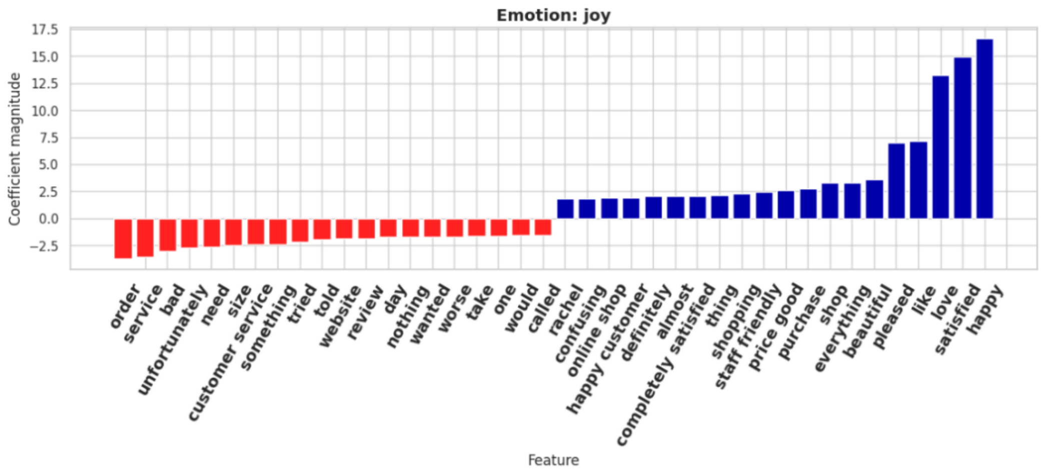

Emotion: sadness
Coefficient magnitude
Feature
always
review
got
rude
well
satisfied
lady
happy
horrible
box
worst
perfect
work
home
fast
dirty
fit
brand
never
everything
change
january
item
forward
legal
wish
purchase
hope
online
slow delivery
received
disappointing
customer
slow
money
disappointed
staff
service
lost
unfortunately

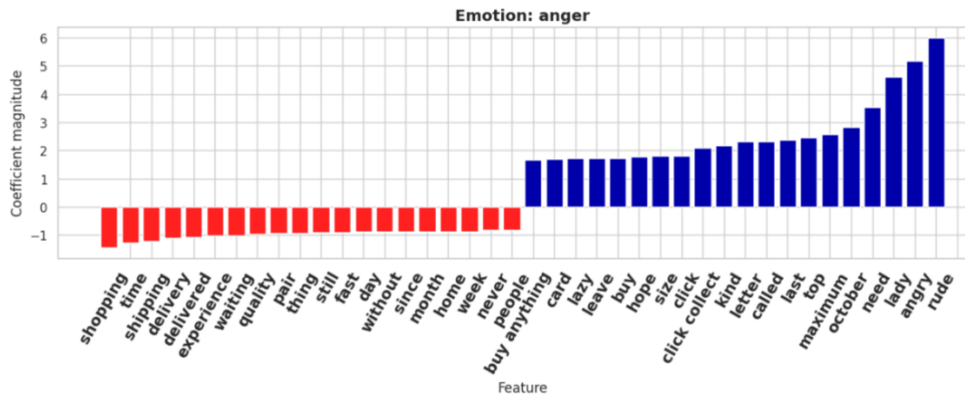

Emotion: surprise
Coefficient magnitude
Feature
shop
store
received
happy
buy
work
come
also
satisfied
money
would
delivery
customer
love
helpful
shoe
look
experience
card
staff
total
rarely
understand bad
really
leave
got
cool
package
suddenly
arrives
believe
day
stock
impressed
surprise
wow
ordered
shocked
refund
surprised

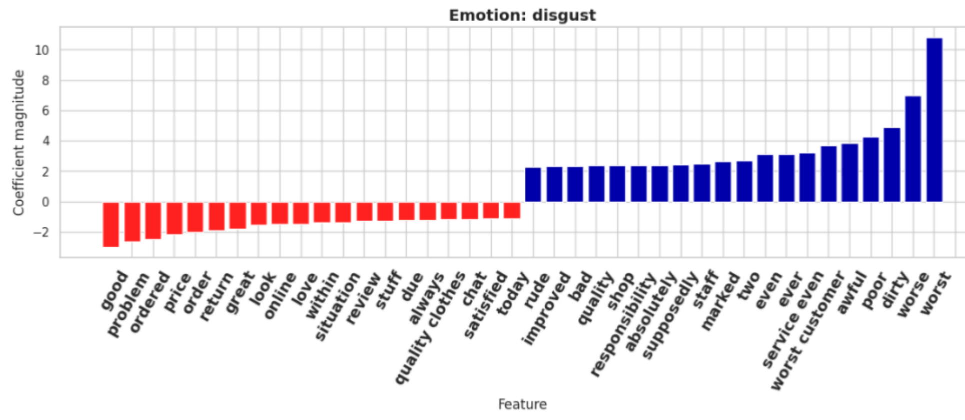

Emotion: neutral
Coefficient magnitude
Feature
love
customer
review
satisfied
happy
refund
item
ordered
service
unfortunately
customer service
like
bad
dress
staff
day
thing
design
experience
store
compared
personally
child clothing
quick delivery
take
good
strange
fine
correct
fast
delivery
product
absolute
never
least
selection
problem
also
work
always

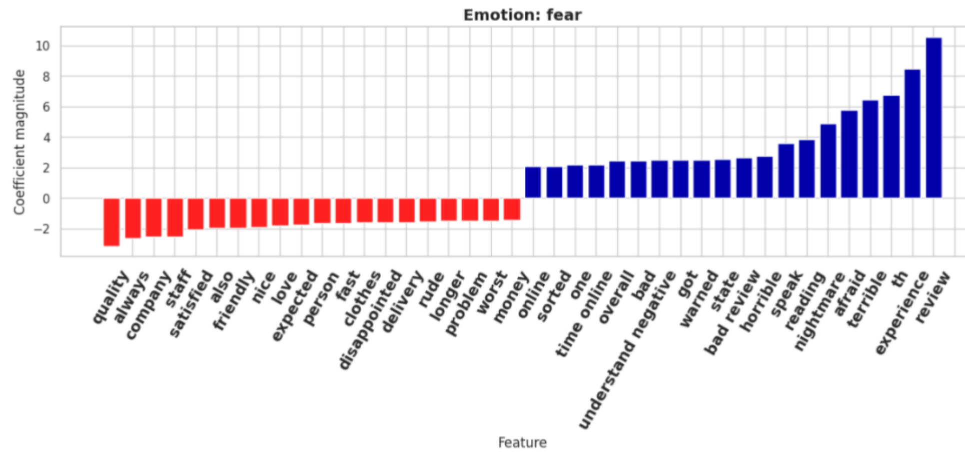
-. **Brand H_:**

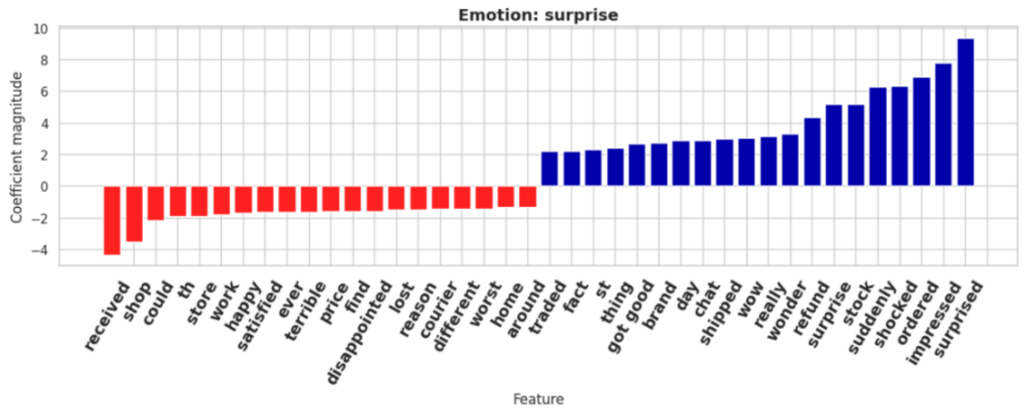

Emotion: sadness
Coefficient magnitude
Feature
always
got
well
everything
worst
happy
brand
never
satisfied
rude
review
speak
surprised
little
right
also
stock
awful
lady
one
july
nice
pity
parcel
picked
announced
choice
store
received
miserable
staff
bring
item
lost
customer
money
service
disappointed
slow
unfortunately

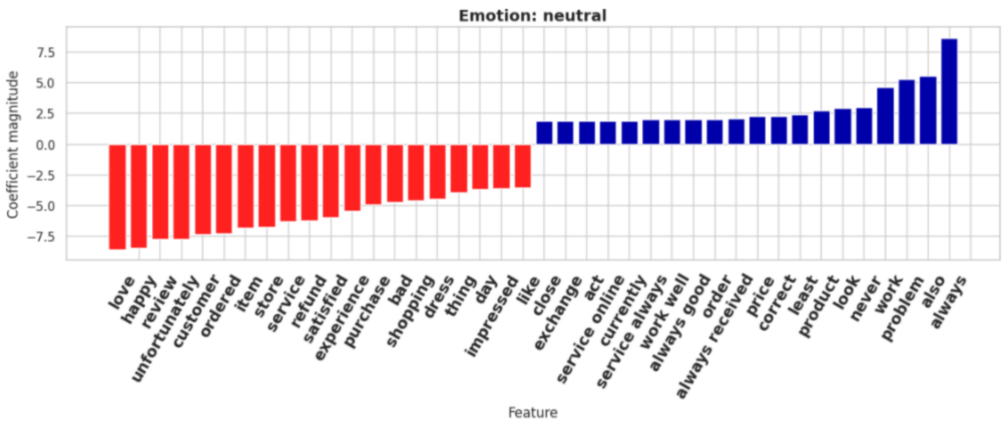

Emotion: joy
Coefficient magnitude
Feature
order
service
bad
unfortunately
need
size
customer service
something
tried
told
website
review
day
nothing
wanted
worse
take
one
would
called
rachel
confusing
online shop
happy customer
definitely
almost
completely satisfied
thing
shopping
staff friendly
price good
purchase
shop
everything
beautiful
pleased
like
love
satisfied
happy


Emotion: anger
Coefficient magnitude
shopping
time
shipping
delivery
delivered
experience
waiting
quality
pair
thing
still
fast
day
without
since
month
home
week
never
people
buy anything
card
lazy
leave
buy
hope
size
click
click collect
kind
letter
called
last
top
maximum
october
need
lady
angry
rude
Feature


Emotion: disgust
Coefficient magnitude
good
problem
ordered
price
order
return
great
look
online
love
within
situation
review
stuff
due
always
quality clothes
chat
satisfied
today
rude
improved
bad
quality
shop
responsibility
absolutely
supposedly
staff
marked
two
even
ever
service even
worst customer
awful
poor
dirty
worse
worst
Feature


Emotion: fear
Coefficient magnitude
quality
always
company
staff
satisfied
also
friendly
nice
love
expected
person
fast
clothes
disappointed
delivery
rude
longer
problem
worst
money
online
sorted
one
time online
overall
bad
understand negative
got
warned
state
bad review
horrible
speak
reading
nightmare
afraid
terrible
th
experience
review
Feature


Emotion: surprise
Coefficient magnitude
received
shop
could
th
store
work
happy
satisfied
ever
terrible
price
find
disappointed
lost
reason
courier
different
worst
home
around
traded
fact
st
thing
got good
brand
day
chat
shipped
wow
really
wonder
refund
surprise
stock
suddenly
shocked
ordered
impressed
surprised
Feature



Emotion: neutral
Coefficient magnitude
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
love
happy
review
unfortunately
customer
ordered
item
store
service
refund
satisfied
experience
purchase
bad
shopping
dress
thing
day
impressed
like
close
exchange
act
service online
currently
service always
work well
always good
order
always received
price
correct
least
product
look
never
work
problem
also
always
Feature