# Final Report : Analysis in Machine Learning models for Sentiment Analysis

Team: Sentiment

Boyeon Park
University of Texas at Austin
Austin, USA
boyeon@utexas.edu

Moonjeog Choi
University of Texas at Austin
Austin, USA
mjchoi@utexas.edu

## I. Introduction

This project aims to apply different machine learning models on the dataset which shows users' reviews on fine foods on Amazon. Then, we'll focus on looking at the accuracies of each model by implementing vectorors, such as count vector, tf-idf, and word2vec, and adjusting the parameters. The reviews will be categorized into different scores indicating their sentimental feelings. 1 and 2 indicate negative reviews. 4 and 5 indicate positive reviews. 3 is dropped from our dataset. The models we will be focusing on include Naive Bayes, Logistic Regression, KNN, Gradient Boosting, and LSTM.

## II. Related work

Sentiment Analysis has been a popular topic for researchers for a long time. In many studies, it was shown that sentiment analysis machine learning classification models of Naive Bayes, Logistic Regression and K-nearest neighbor can be well performed with text data[1][2][3]. In a study, it is demonstrated that ensemble learning techniques such as gradient boosting can be applied as an effective tool for sentiment analysis[4]. In another study, the author pointed out that recurrent backpropagation or simple neural networks are inefficient to learn information that is largely extended over time. LSTM was proposed and they showed LSTMs work better on a large variety of problems[5]. Today, there is plenty of text data that can be used for sentiment analysis, and product review data can be a good source for text analysis in that each product review receives inspections before being posted and it has a rated score that can be used as a feature of analysis[6]. As many machine learning models were proposed for sentiment analysis, this study aims to compare those models' performance on Amazon product review dataset.

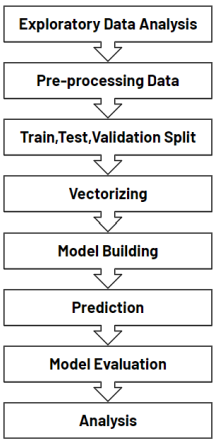
## III. Method

We experiment sentiment analysis with 5 models: Naive bayes, K-Nearest-Neighbor, Logistic regression, Gradient Boosting and LSTM. We chose an Amazon review dataset which consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plain text review.
Data includes:

- Reviews from Oct 1999 - Oct 2012
- 568,454 reviews
- 256,059 users
- 74,258 products
- 260 users with > 50 reviews
- https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

Proposed method uses the following tasks to classify sentiment analysis using machine learning techniques.

Figure 1. Proposed method




## IV. Experiment

### A. Exploratory Data Analysis

Originally, our dataset included 568,454 rows of reviews, and 10 columns, including id, productid, userid, profilename, helpfulnessnumerator, score, time, summary, text, helpfulnessdenominator. Stop words, extra white spaces were removed. Texts were all converted to lowercase. Review scores were categorized in 1, 2, 4, 5. 1

and 2 indicate negative reviews, and 4, 5 indicate positive reviews, and 3s were dropped from our database. The following shows the counts of different scores in our dataset. We have then plotted a histogram and a boxplot. From these two graphs, we could see our data is skewed. The positive scores are much more than the negative ones.

Figure 2. Histogram of Score Distribution

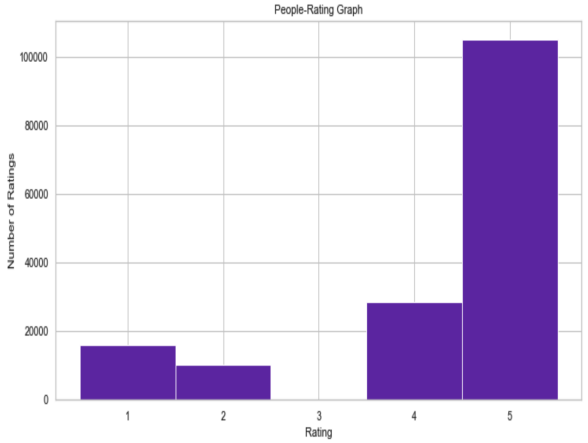
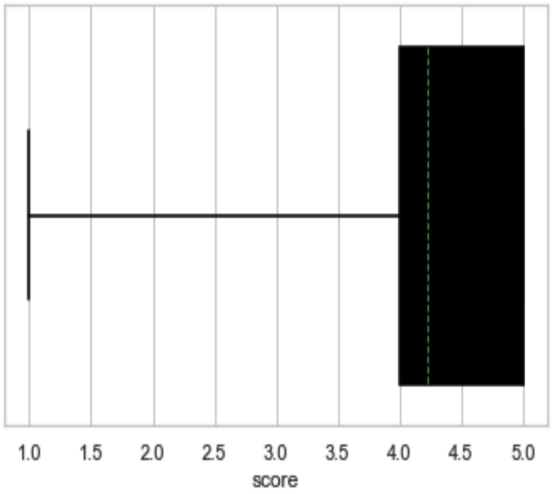



Figure 3. Boxplot of Score Distribution




B. Pre-processing and Vectorizing

At first, cleaning texts was processed by lowercasing, removing emojis, white space and stopwords. For sentiment labels, reviews with scores of 4,5 were labeled as positive and reviews with scores of 1,2 were labeled as negative. Product reviews which have more than +20 reviews per product were selected to have reliable reviews and too long reviews with more than 100 words were dropped. Using scikit learn train, test split function, we divided our data set into train, test, validation set and saved to be used for each model building. For vectorizing texts, count vector, tf-idf, and word2vec were used in machine learning models and keras tokenizer was used in LSTM.

C. Model Building and Evaluation

- Naive Bayes

Developed based on Bayes formula, Naive Bayes has assumed the value of one feature is independent of another, which helps to increase the complexity of the model compared with Bayes. Naive-bayes is widely used in text analysis, and it basically is a probabilistic classifier that aims to make classification based on probability. We have built three Naive Bayes model based sentiment classifiers, one with count vector and one with tf-idf and one with word2vec using the sklearn and gensim(word2vec) package. The performance showed in the confusion matrix that the accuracy is higher with count vector, which is 92%, while the tf-idf model is 88% and word2vec is 83%.

Table 1. NB Confusion matrix (count vector)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.74 | 0.76 | 5291 |
| 1 | 0.95 | 0.96 | 0.95 | 26731 |
| accuracy | | | 0.92 | 32022 |
| macro avg | 0.86 | 0.85 | 0.86 | 32022 |
| weighted avg | 0.92 | 0.92 | 0.92 | 32022 |

Table 2. NB Confusion matrix (tf-idf)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.32 | 0.48 | 5291 |
| 1 | 0.88 | 1.00 | 0.93 | 26731 |
| accuracy | | | 0.88 | 32022 |
| macro avg | 0.91 | 0.66 | 0.71 | 32022 |
| weighted avg | 0.89 | 0.88 | 0.86 | 32022 |

Table 3. NB Confusion matrix (word2vec)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 5291 |
| 1 | 0.83 | 1.00 | 0.91 | 26731 |
| accuracy | | | 0.83 | 32022 |
| macro avg | 0.42 | 0.50 | 0.45 | 32022 |
| weighted avg | 0.70 | 0.83 | 0.76 | 32022 |

with word2vec, for some reasons the model couldn't classify negative reviews at all but we couldn't figure out during the experiment, so it's left for our future work.

- K-Nearest Neighbor

Given a training data set, k-nearest neighbor algorithm finds K instances closest to the new input instance in the training data set. Most of these K instances belong to a certain class, and then classifies the input instance into this class. In our analysis, three KNN model based sentiment classifiers were built, one with count vector and one with tf-idf and one with word2vec. The performance showed that the accuracy is higher with word2vec which

is 92% while count vector and tf-idf models had 90% accuracy.

Table 4. KNN Confusion matrix (count vector)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.80 | 0.49 | 0.61 | 5291 |
| 1 | 0.91 | 0.98 | 0.94 | 26731 |
| accuracy | | | 0.90 | 32022 |
| macro avg | 0.85 | 0.73 | 0.77 | 32022 |
| weighted avg | 0.89 | 0.90 | 0.88 | 32022 |

Table 5. KNN Confusion matrix (tf-idf)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.85 | 0.50 | 0.63 | 5291 |
| 1 | 0.91 | 0.98 | 0.94 | 26731 |
| accuracy | | | 0.90 | 32022 |
| macro avg | 0.88 | 0.74 | 0.79 | 32022 |
| weighted avg | 0.90 | 0.90 | 0.89 | 32022 |

Table 6. KNN Confusion matrix (word2vec)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.62 | 0.72 | 5291 |
| 1 | 0.93 | 0.98 | 0.95 | 26731 |
| accuracy | | | 0.92 | 32022 |
| macro avg | 0.89 | 0.80 | 0.83 | 32022 |
| weighted avg | 0.91 | 0.92 | 0.91 | 32022 |

- Logistic regression

Logistic Regression is a ‘Statistical Learning’ technique categorized in ‘Supervised’ Machine Learning (ML) methods dedicated to ‘Classification’ tasks. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function. Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. [7] We built three Logistic regression model based sentiment classifiers, one with count vector and one with tf-idf and one with word2vec. The threshold in this project was set as 0.5 which is the default setting of sklearn package. The performance showed that Logistic regression performed really well, especially with count vector. The accuracy was 96%, and 94% and 91% for count vector, tf-idf, word2vec respectively.

Table 7. LR Confusion matrix (count vector)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 0.84 | 0.86 | 5291 |
| 1 | 0.97 | 0.98 | 0.97 | 26731 |
| accuracy | | | 0.96 | 32022 |
| macro avg | 0.93 | 0.91 | 0.92 | 32022 |
| weighted avg | 0.95 | 0.96 | 0.96 | 32022 |

Table 8. LR Confusion matrix (tf-idf)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 0.70 | 0.78 | 5291 |
| 1 | 0.94 | 0.98 | 0.96 | 26731 |
| accuracy | | | 0.94 | 32022 |
| macro avg | 0.92 | 0.84 | 0.87 | 32022 |
| weighted avg | 0.93 | 0.94 | 0.93 | 32022 |

Table 9. LR Confusion matrix (word2vec)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.61 | 0.69 | 5291 |
| 1 | 0.93 | 0.97 | 0.95 | 26731 |
| accuracy | | | 0.91 | 32022 |
| macro avg | 0.86 | 0.79 | 0.82 | 32022 |
| weighted avg | 0.90 | 0.91 | 0.90 | 32022 |

- Gradient Boosting

Gradient Boosting is an ensemble technique that applies a decision tree as a base classifier. In ensemble techniques, the weak learners combine to make a strong model. The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model by building a new model on the errors or residuals of the previous model. [4] We built three gradient boosting model based sentiment classifiers, one with count vector and one with tf-idf and one with word2vec. The performance showed that Gradient Boosting performed better with word2vec which showed 90% accuracy. Both count vector and tf-idf models performed similarly (accuracy 87% on test data set).

Table 10. GBM Confusion matrix (count vector)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.24 | 0.39 | 5291 |
| 1 | 0.87 | 0.99 | 0.93 | 26731 |
| accuracy | | | 0.87 | 32022 |
| macro avg | 0.89 | 0.62 | 0.66 | 32022 |
| weighted avg | 0.88 | 0.87 | 0.84 | 32022 |

Table 11. GBM Confusion matrix (tf-idf)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.25 | 0.39 | 5291 |
| 1 | 0.87 | 0.99 | 0.93 | 26731 |
| accuracy | | | 0.87 | 32022 |
| macro avg | 0.89 | 0.62 | 0.66 | 32022 |
| weighted avg | 0.88 | 0.87 | 0.84 | 32022 |

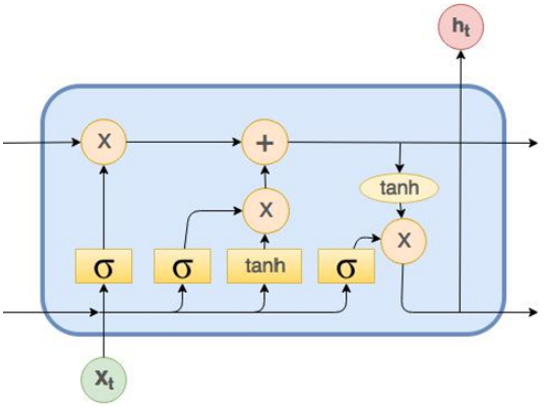
Table 12. GBM Confusion matrix (word2vec)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.86 | 0.44 | 0.58 | 5291 |
| 1 | 0.90 | 0.99 | 0.94 | 26731 |
| accuracy | | | 0.90 | 32022 |
| macro avg | 0.88 | 0.71 | 0.76 | 32022 |
| weighted avg | 0.89 | 0.90 | 0.88 | 32022 |

- LSTM

Long Short-Term Memory (LSTM) is a special type of recurrent neural network (RNN) architecture that was designed over simple RNNs for modeling temporal sequences and their long-range dependencies more accurately. The LSTM consists of units or memory blocks in the recurrent hidden layer, which contains memory cells with self-connections storing the temporal state of the network. In addition to this the network has special multiplicative units called gates to control the flow of information in the network. Each memory block in the original architecture contained an input gate and an output gate. The input gate controls the flow of input activations into the memory cell and the output gate controls the output flow of cell activations into the rest of the network. The forget gate was added to the memory block that scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cell's memory. In addition, the modern LSTM architecture might also contains peephole connections from its internal cells to the gates in the same cell which help to learn precise timing of the outputs.[8]

Figure 4. Explanation of LSTM



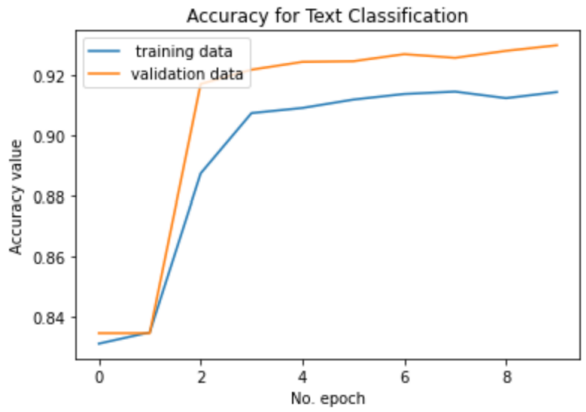

We built a LSTM based sentiment classification model using Tensorflow and keras sequential model. Before model fitting, keras tokenizer and word2vec were used for vectorization which turns each text into either a sequence of integers. First, an embedding layer was added and then, dropout layer was added to prevent overfitting. After them, LSTM layer was added. After a flatten layer, three dense layers and dropout layers between dense layers were added and the output of a probability could be obtained. For middle dense layers, 'relu' activation function was used and for the final dense layer, 'sigmoid' function was used.

Table 13. LSTM model constitution

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding (Embedding) | (None, 100, 16) | 800016 |
| dropout (Dropout) | (None, 100, 16) | 0 |
| lstm (LSTM) | (None, 100, 16) | 2112 |
| flatten (Flatten) | (None, 1600) | 0 |
| dense (Dense) | (None, 512) | 819712 |
| dropout_1 (Dropout) | (None, 512) | 0 |
| dense_1 (Dense) | (None, 8) | 4104 |
| dropout_2 (Dropout) | (None, 8) | 0 |
| dense_2 (Dense) | (None, 1) | 9 |

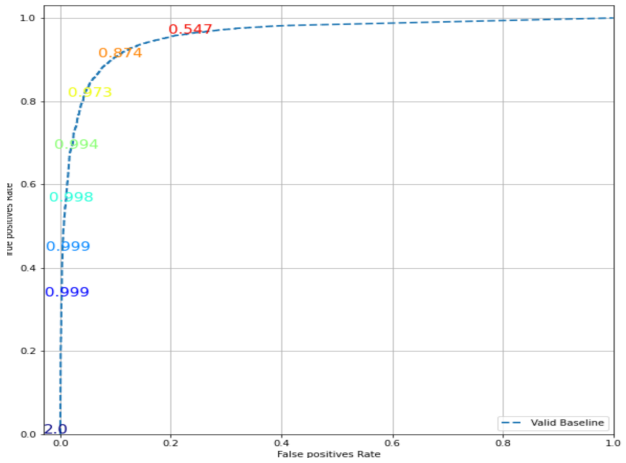
Epoch was set as 10 and we could see in each epoch, the loss decreased and the accuracy increased.

Figure 5. Accuracy changes in each epoch




To determine the threshold, we looked at ROC curve and we set prediction >= 0.89 as positive(1) prediction < 0.89 as negative(0).

Figure 6. ROC curve




The confusion matrix is shown below and the accuracy was 90% on the test data set.

Table 14. LSTM Confusion matrix(keras oov tokenizer, threshold = 0.89)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.65 | 0.90 | 0.76 | 5291 |
| 1 | 0.98 | 0.91 | 0.94 | 26731 |
| accuracy | | | 0.90 | 32022 |
| macro avg | 0.82 | 0.90 | 0.85 | 32022 |
| weighted avg | 0.92 | 0.90 | 0.91 | 32022 |

We changed the threshold to 0.5 to see the effects and found out that the accuracy got higher to 93%.

Table 15. LSTM Confusion matrix(keras oov tokenizer, threshold : 0.5)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.68 | 0.75 | 5291 |
| 1 | 0.94 | 0.97 | 0.96 | 26731 |
| accuracy | | | 0.92 | 32022 |
| macro avg | 0.89 | 0.83 | 0.85 | 32022 |
| weighted avg | 0.92 | 0.92 | 0.92 | 32022 |

And also word2vec was used for training. Compared with the model with keras tokenizer, the performance was similar but slightly lower (92%), considering it's set in the same conditions (epoch=10, threshold = 0.5).

Table 16. LSTM Confusion matrix(epoch=10, word2vec)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.61 | 0.69 | 5291 |
| 1 | 0.93 | 0.97 | 0.95 | 26731 |
| accuracy | | | 0.91 | 32022 |
| macro avg | 0.86 | 0.79 | 0.82 | 32022 |
| weighted avg | 0.90 | 0.91 | 0.90 | 32022 |

## V. Conclusion

We explored 5 models : Naive Bayes, Logistic Regression, KNN, Gradient Boosting, and LSTM for Sentiment Analysis problem with Amazon product review data. Among 5 models, Logistic Regression performed the best with the dataset in this project. In terms of vectorizing types, the results show that it's not like one is better than others but sometimes countvect performed better and sometimes word2vec performed better than others. So when selecting models and vectorizing methods we found that it's good to try various combinations of them and see what performs better.

Table 17. Model performance comparison

| classifier | countvect | tf-idf | word2vec |
|---|---|---|---|
| Naïve Bayes | **92%** | 88% | 83% |
| K-Nearest-Neighbor | 90% | 90% | **92%** |
| **Logistic Regression** | **96%** | 94% | 91% |
| Gradient Boosting | 87% | 87% | **90%** |

| classifier | keras | word2vec |
|---|---|---|
| LSTM | **92%** | 91% |

We also found that when determining the threshold of probability to classify, not only the ROC curve but also data structure can be considered. The data we used in this project was skewed data which had far more positive reviews than negative reviews, so when the threshold for a more true positive rate was set, it had higher accuracy.

Finally, the results showed that not always complex and the latest model performs better than the others, classic models can perform well like logistic regression. We could learn that model selection is not a simple work that should include considering training and computing time, data structure and feature, method, hyperparameter, overfitting, deciding evaluation matrix, etc comprehensively through this project.

REFERENCES

[1] Jagdale RS, Shirsat VS, Deshmukh SN. Sentiment analysis on product reviews using machine learning techniques. InCognitive Informatics and Soft Computing 2019 (pp. 639-647). Springer, Singapore.

[2] Prabhat A, Khullar V. Sentiment classification on big data using Naïve Bayes and logistic regression. In2017 International Conference on Computer Communication and Informatics (ICCCI) 2017 Jan 5 (pp. 1-5). IEEE.

[3] Daeli NO, Adiwijaya A. Sentiment analysis on movie reviews using Information gain and K-nearest neighbor. Journal of Data Science and Its Applications. 2020 May 11;3(1):1-7.

[4] Gradient Boosting Algorithm: A Complete Guide for Beginners. Analytics Vidhya. Anshul Saini. 2021 Sep. Available from:https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners

[5] Schmidhuber J, Hochreiter S. Long short-term memory. Neural Comput. 1997 Nov 15;9(8):1735-80.

[6] Fang X, Zhan J. Sentiment analysis using product review data. Journal of Big Data. 2015 Dec;2(1):1-4.

[7] Pang B, Lee L, Vaithyanathan S. Thumbs up? Sentiment classification using machine learning techniques. arXiv preprint cs/0205070. 2002 May 28.

[8] Pal S, Ghosh S, Nag A. Sentiment analysis in the light of LSTM recurrent neural networks. International Journal of Synthetic Emotions (IJSE). 2018 Jan 1;9(1):33-9