COSC 274 Machine Learning and Statistical Data Analysis Final Report

A. Introduction

The past decade has seen the explosion of e-commerce platforms, which have rapidly transformed consumer behavior. Online reviews have emerged as a crucial source of information, enabling buyers to make informed purchasing decisions. Amazon is one such e-commerce site that has gained immense popularity due to its vast selection of goods and customer evaluations. With millions of reviews spanning diverse categories, Amazon Reviews is an extensive collection that serves as a valuable resource for shoppers worldwide. These reviews not only offer insights into the quality of the product but also provide a platform for customers to share their experiences with the product, the delivery process, and the customer service. Furthermore, customers can rate the reviews themselves, ensuring that the most helpful and reliable reviews are easily accessible to others. Overall, the impact of online reviews on e-commerce has been significant, and platforms like Amazon have played a vital role in shaping this trend.

The objective of the course project is to apply the machine learning models and concepts discussed in this course to a real-world dataset. As part of the COSC74/274 winter 2023 course final project, the Amazon Reviews dataset was analyzed to extract meaningful insights and patterns from the data, and ultimately predict user ratings from the product review data. The project involved several stages, including data preprocessing, feature extraction, model building, and data classification. The Amazon product review dataset was utilized, with a focus on binary classification, multi-class classification, and clustering methods to analyze the product reviews.

This report provides a detailed account of the entire project, including preprocessing, hyperparameter tuning, prediction analysis, and the application of machine learning algorithms. The project aimed to use various supervised and unsupervised learning algorithms to classify reviews into different categories based on their helpfulness score. Additionally, various visualization techniques were explored to gain insights into the data and provide meaningful representations of the findings.

A.1. Amazon Review Data

The datasets provided for this project are Training.csv and Test.csv. The training data contains 29189 rows, and the test data contains 4500 rows. The training dataset provides the information presented in Table 1, while the Test dataset includes the same variables, with the exception of the ”overall” data column, which needs to be predicted. In order to classify the ”overall” variable, the most useful data from this list includes ”reviewText”, ”summary”, ”verified”, ”vote”, and ”category”.

Table 1. Review-related information in the Amazon review dataset

Data	Overview
overall	Product’s rating (1-5)
verified	A boolean denoting the verification
reviewTime	Time of review
reviewerID	Amazon reviewer’s ID
asin	Product ID
reviewerName	Encoded reviewer’s name
reviewText	The Amazon review
summary	Summary of review
unixReviewTime	Unix time of review
vote	Upvotes for being helpful
image	If there is an image, link to the image
style	Style information
category	Amazon product category

It is worth noting that the ”overall” column consists of integers ranging from 1 to 5, whereas ”reviewText” and ”summary” are text data written by the reviewer. ”Verified” is a boolean that indicates whether the review has been verified by Amazon. ”Vote” denotes the integer number of other people who support the review, and ”category” includes string data with different Amazon product categories.

In addition, we’ll be utilizing Python as our primary programming language for analysis and prediction. The packages mainly employed are sklearn. It’s worth noting that neither NLTK nor network architectures are employed in this project.

B. Classification Methods

The classification section consists of four binary classifications and one multi-classification. The binary classifications are based on the ”overall” rating, with a cutoff point determining if the rating is considered zero or one. The four binary classifications have cutoffs of 1, 2, 3, and 4, and are labeled as 0 and 1. The multi-classification has five labels, corresponding to the ratings 1-5.

Each classification task consists of several steps, each of which is crucial for the success of the project. Firstly, it is necessary to pre-process the data to ensure that it is clean and ready for analysis. This may involve filling in missing values, optimizing stop words in the text data, and creating categories for specific columns. Once the data has been pre-processed, the next step is to do feature selection and then select an appropriate machine-learning model. After choosing a model, it is important to tune the hyperparameters of the respective model to ensure optimal performance.

Once the model has been trained, it is important to evaluate its performance using appropriate metrics such as f1 score, AUC score, accuracy, confusion matrix, and ROC curve. Finally, the most accurate machine learning model with its best hyperparameters can be used to predict the test data.

B.1. Preprocessing

As previously mentioned in the report, the important data used for classification are "verified", "vote", "reviewText", "summary", and "category". For preprocessing the y labels, we extract the "overall" data and then assign 0 if it is less than the cutoff and 1 if it is more than the cutoff. For preprocessing the features data, firstly the "vote" column had a few NA values which were filled with zeros. Also, categories were created based on the number of votes: "low" for less than 2 upvotes, "medium" for 3 to 10 upvotes, "good" for 11 to 50 votes, and "high" for greater than 51 votes. Secondly, no NA values were found for "category" and "verified" columns data and since it is categorical, no further preprocessing was required.

To prepare the "reviewText" and "summary" fields for classification, several preprocessing steps are necessary. First, the text data needs to be converted to lowercase to ensure consistency and remove any variations in capitalization that may cause issues during the classification process. Second, there may be unwanted content in the text data such as URLs, numbers, symbols, or other irrelevant information that does not contribute to the classification task. Therefore, any such content must be removed to ensure that the classification algorithm only focuses on relevant content. Thirdly, stop words that are commonly used in language but do not contribute to the meaning of the text need to be removed.

The stopwords list is divided into two categories. The first category includes the most commonly used English words that do not contribute to the accuracy of the classification, such as "a", "an", "the", "your", "him", etc. The second category consists of words that fall within the borderline of classification. To identify these words, the top 3000 most frequently used words from the "summary" and "reviewText" columns data are extracted separately. Then, each word is checked to find the "overall" label of the row in which the word is present, and the metric of the number of zeros divided by the sum of zeros and ones is evaluated. If the metric falls between 0.45 and 0.55, the word is considered a border zone word and added to the second category list of stop words. This process is repeated for both "summary" and "reviewText" data and for each cutoff of 1, 2, 3, and 4. By removing irrelevant content and stop words, the algorithm can focus on the most important information and make accurate predictions. Finally, the training data is split into 80% for training and 20% for testing using $sklearn.modelselection$'s train test split to extract the best hyperparameters.

B.2. Feature selection

For the feature selection, the preprocessed text data from the "summary" and "reviewText" fields are processed using the TfidfVectorizer package from $sklearn.featureextraction.text$. This process converts the data into a matrix of TF-IDF features. For the fields "verified", "category", and "vote", $sklearn.preprocessing$ OrdinalEncoder is used to convert the categories into random integer values. This will speed up the analysis and reduce memory usage.

B.3. Machine Learning models

Four different machine learning models are chosen for this specific type of classification: Logistic Regression, Multinomial Naive Bayes, Linear Support Vector Classification, and Stochastic Gradient Descent with Linear classifiers.

B.4. Hyper parameter tuning

The hyperparameters for the classification types have been fine-tuned using the same set of hyperparameters for each respective classifier. To accomplish this, a pipeline was created with preprocessing as the first step and the classification model as the second step. The pipeline was then optimized using GridSearchCV from the $sklearn.model_selection$ package, where the range of parameters and scoring were based on "f1 macro". The optimization process involved cross-validation of 5 folds. After fitting the pipeline, the best hyperparameter combination was selected and other metrics such as f1 score, AUC score, accuracy, and ROC curve were evaluated. In summary, the hyperparameter fine-tuning process was a crucial step in achieving the desired results, and the meticulous approach taken has paid off in the form of improved model performance. The range of hyperparamters used for different classifiers and for the four binary classifications and multi-classification are shown below.

Table 2. Logistic Regression

Parameter	Range
solver	liblinear
C	[0.1, 1.0, 5.0, 10.0]
max_iter	[100, 500, 1000]

Table 3. Multinomial Naive Bayes

Parameter	Range
alpha	[0.01, 0.1, 0.5, 1, 10]
fit_prior	[True,False]

Table 4. LinearSVC

Parameter	Range
C	[10, 1, 0.1, 0.01]
penalty	['l1', 'l2']

Table 5. Stochastic Gradient Descent

Parameter	Range
alpha	[0.001, 0.01, 0.1, 1]
loss	['hinge', 'perceptron']

Table 6. Vectorizer

Parameter	Range
summary_ngram_range	(1, 2)
reviewText_ngram_range	(1, 2)
summary_max_features	[10000, 32000]
reviewText_max_features	[5000, 10000]

C. Classification Results

The results for different classifiers and for different classification types are shown below.

Table 7. Results of binary classification with cutoff 1

Model	AUC score	f1 score	Accuracy
Logistic Regression	0.769389	0.789156	0.874443
MultinomialNB	0.819013	0.785497	0.84755
LinearSVC	0.766099	0.790261	0.877355
SGD classifier	0.733603	0.768355	0.872902

Figure 1. Confusion matrix and ROC curve for LinearSVC 1

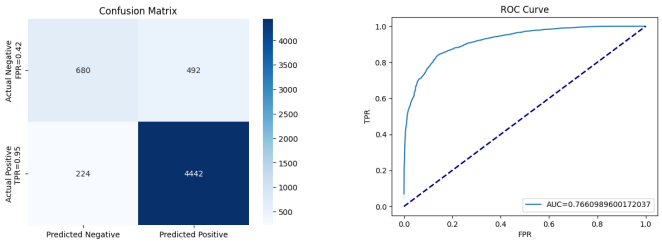


Table 8. Results of binary classification with cutoff 2

Model	AUC score	f1 score	Accuracy
Logistic Regression	0.846928	0.849768	0.856115
MultinomialNB	0.842310	0.841278	0.846180
LinearSVC	0.848613	0.850808	0.856800
SGD classifier	0.841993	0.845003	0.851662

Figure 2. Confusion matrix and ROC curve for LinearSVC 2

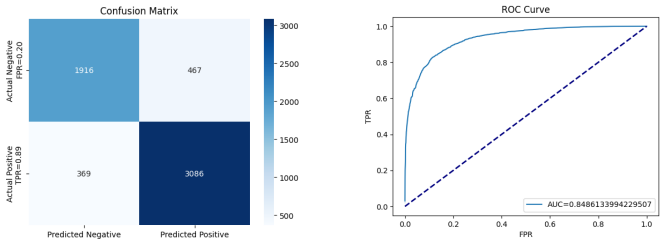


Table 9. Results of binary classification with cutoff 3

Model	AUC score	f1 score	Accuracy
Logistic Regression	0.854149	0.858265	0.866393
MultinomialNB	0.861791	0.861431	0.867592
LinearSVC	0.861962	0.863807	0.870675
SGD classifier	0.852854	0.855370	0.862967

Figure 3. Confusion matrix and ROC curve for LinearSVC 3

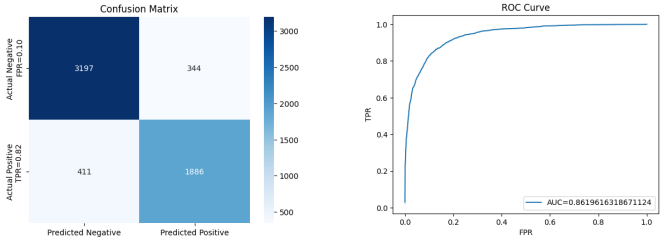
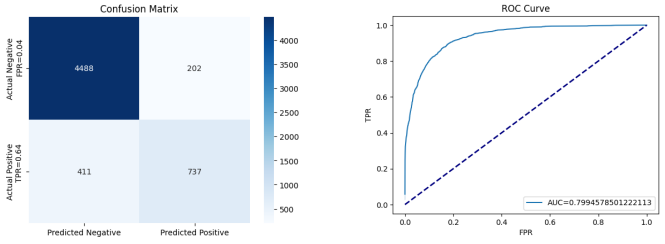


Table 10. Results of binary classification with cutoff 4

Model	AUC score	f1 score	Accuracy
Logistic Regression	0.795931	0.814364	0.889860
MultinomialNB	0.839291	0.821096	0.880781
LinearSVC	0.799458	0.821175	0.894998
SGD classifier	0.771849	0.801313	0.887633

Figure 4. Confusion matrix and ROC curve for LinearSVC 4



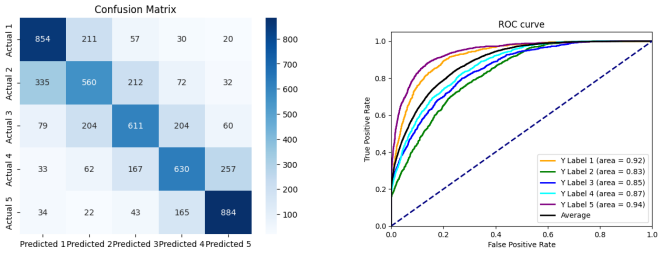
C.1. Analysis

After conducting a thorough analysis of the results, it is undoubtedly clear that the LinearSVC model has outperformed the other three models in the f1 score as shown from the results for each of the classifier and classification type.

Table 11. Results of Multi classification

Model	AUC score	f1 score	Accuracy
Logistic Regression	0.888729	0.600205	0.602604
MultinomialNB	0.887109	0.602023	0.601062
LinearSVC	0.887789	0.602928	0.606201
SGD classifier	0.879063	0.588616	0.590785

Figure 5. Confusion matrix and ROC curve for LinearSVC Multi



However, it is important to note that tuning the parameters for the LinearSVC model can consume a significant amount of time and memory, which can be a drawback for some users. While LinearSVC is the top performer in terms of accuracy, it is worth mentioning that the MultinomialNB and Logistic Regression models have also demonstrated strong performance in comparison to LinearSVC. In fact, the MultinomialNB and Logistic Regression models have the added advantage of being computationally less expensive than the LinearSVC model. Moving on to the hyperparameters that performed well for each classifier, detailed discussions are provided below.

C.1.1 LinearSVC

The optimal combination of hyperparameters for the LinearSVC model is C=0.1 and penalty='l2'. It is worth noting that C is associated with the regularization parameter, where a smaller value corresponds to a higher degree of regularization, which could fit the model well. As for the loss parameter, l2 is preferred over l1 as it sums the squares of the weights, providing better results.

C.1.2 MultinomialNB

The optimal combination of hyperparameters for the MultinomialNB model is alpha=0.5 and prior=False. The multinomial Naive Bayes classifier is typically well-suited for classification with discrete features, such as text data. In our classification data, a smoothing parameter of alpha 0.5 is the optimal value compared to the default 1.0. Additionally, due to insufficient information on prior knowledge of classification labels, it is expected that we will not select learning class prior probabilities.

C.1.3 Logistic Regression

Based on the findings, the best combination of hyperparameters for the Logistic Regression model is C=5.0, max_iter=100, and solver=liblinear. If you have small amounts of data, we recommend using the solver liblinear as it performs faster than the default solver, libfgs. The parameter C is the regularization parameter, where a higher value corresponds to a lower degree of regularization. A C value of 5.0 indicates that lower regularization is necessary to achieve improved results.

C.1.4 Stochastic Gradient Descent Classifier

After trying out various combinations, it was found that the optimal hyperparameters for the SGD classifier are alpha=0.001 and loss=hinge. The SGD classifier uses stochastic gradient descent (SGD) learning to implement regularized linear models. However, it was observed that this classifier did not perform as well as the other three models. It is worth noting that the perceptron model did not perform well for this classification dataset, and is therefore not recommended.

C.1.5 TfidfVectorizer

Finally, the TfidfVectorizer is used to create TFID features matrix from the words in the data. For all classifiers and types, on an average, the best parameter combination for max features in reviewText is 5000, while the best parameter combination for summary is 32000. the recommended ngram_range for reviewText is (1,2), and the recommended ngram_range for summary is also (1,2). Generally, more data leads to better accuracy in max features. However, it is also important to improve the data with better preprocessing techniques. It is worth noting that "reviewText" had a more diverse usage of words than "summary" and is thus less useful for final classification. For this reason, fewer features are recommended in comparison to "summary". An ngram_range of (1,2) works better than (1,1) because phrases such as "best product" need to be considered instead of just "product".

D. Classification Predictions

The optimized hyperparameters are utilized to predict labels for the test data, resulting in the following f1 scores below in the Table 12.

The results suggest that a single model may not consistently be effective when it comes to various types of classifications. After hyperparameter tuning, LinearSVC performed the best for all types of classifications, but the difference in f1 scores compared to other models was not significant. LinearSVC appears to work well when the cutoff falls somewhere in the middle, slightly compromising on

Table 12. Final f1 score results of different classification types with best estimated hyper parameters

Classification	F1 score	Model
Binary Classification 1	0.7833	MultinomialNB
Binary Classification 2	0.8386	LinearSVC
Binary Classification 3	0.8765	LinearSVC
Binary Classification 4	0.82121	Logistic Regression
Multi-Classification	0.61413	MultinomialNB

performance and memory usage. However, when the cutoff is either 1 or 4, MultinomialNB and Logistic Regression outperform LinearSVC in terms of f1 scores.

For multiclassification, it is recommended to optimize preprocessing by removing all stop words. This is because having maximum data is advisable for achieving the desired results in multiclassification. Additionally, when the data contains a lot of noise and is extensive, MultinomialNB has been shown to perform better than other models for this type of classification data. It is important to note that, while LinearSVC may not be the best model across all types of classifications, it is still a strong contender that performs well in various cases.

E. Clustering

Clustering is a powerful technique widely used in machine learning and data analysis. The goal is to group similar data points based on their features. By finding patterns or structures in the data, clustering can support further analysis and decision-making processes.

For the scope of this project, clustering needs to be performed on the provided test data, without utilizing the ”category” columns. The goal is to identify hidden patterns and structures in the data. After clustering has been performed, the resulting clusters are compared against the ”category” labels. To evaluate the effectiveness of the clustering, Silhouette, and Rand Index scores will be used, which will provide a quantitative measure of how well the clustering has performed.

Given a large amount of data, it is important to check the number of clusters for various data combinations, in order to identify the optimal number of clusters for each combination. To perform k-means clustering, four combinations of data are taken and an optimal number of clusters for each combination will be evaluated. This approach will ensure that our clustering is both accurate and effective, enabling us to make informed decisions based on the results.

Similar to classification, clustering is only performed on the columns ”reviewText”, ”summary”, ”verified”, ”vote”, and ”category”. However, it is important to include ”style” data as it significantly contributes to clustering since it is directly related to the ”category” labels. We analyze four combinations of data columns: all data, only text data, non-text data, and only style data. Preprocessing is reduced to only removing NA values as reducing data will only decrease the accuracy of the clusters.

E.1. Clustering model and parameters

Clustering is performed on the different combinations (Table 13) of test data using kmeans clustering model. The default parameters will be maintained for all clustering operations, except for the crucial parameter of the number of clusters. By varying n, the number of clusters, the robustness of the model can be tested under different conditions. In this project, we will use a comprehensive range of values for n, including 2, 3, 4, 5, 6, 7, 8, 9, 10, 100, 300, and 500. These values will be applied to various data combinations, ensuring that the model is tested under a wide range of conditions. The effectiveness of each value of n will be estimated based on the silhouette scores. This rigorous testing procedure will allow in identifying the optimal number of clusters for the given data, providing with a more accurate and nuanced understanding of the underlying patterns and relationships within the data set.

Table 13. Data combinations used for clustering

Data name	Columns
All Data	”reviewText”, ”summary”, ”verified”, ”vote”, ”category” and ”style”
Non-Text Data	”verified”, ”vote”, ”category” and ”style”
Only Text Data	”reviewText” and ”summary”
Only Style Data	”style”

F. Clustering Results

The following results were observed in the K-means clustering for different data combinations, with a focus on prioritizing Silhouette scores for varying numbers of clusters.

Table 14. Clustering results of Data combinations

Data name	Clusters	Silhouette score	Rand index
All Data	2	0.669292	0.002898
Non-Text Data	500	0.692069	0.084945
Only Text Data	2	0.675094	0.112200
Only Style Data	500	0.733584	0.161114

F.1. Analysis

The results of the models have shown that the inclusion of text data in the clustering model has a significant impact on the number of clusters needed to achieve higher Silhouette scores. It was observed that when text data is

not included in the model, a larger number of clusters are required to achieve better Silhouette scores. On the other hand, when text data is included, a smaller number of clusters can achieve higher Silhouette scores. This suggests that text data plays a crucial role in the clustering model.

Moreover, it was found that using all data, including text and style data, results in a much lower Rand index score compared to using only style data. This finding indicates that text data might be less reliable in determining the similarity between data points. However, further investigation is needed to fully understand the relationship between text data and the clustering model performance.

G. Conclusion

To summarize, this report discusses the utilization of the Amazon reviews dataset to predict the overall rating of a review. The report emphasizes the importance of data preprocessing, hyperparameter tuning, and model selection in achieving the desired results. Analyzing stop words, specifically the extraction of the second bucket mentioned in the report, significantly improved the model scores. The summarized results show that the LinearSVC model outperformed the other models for classification results in f1 scores, although the MultinomialNB and Logistic Regression models also demonstrated strong performance. Regarding clustering, Silhouette, and Rand Index scores were utilized to evaluate the effectiveness of the clustering. It is observed that the inclusion of text data in the clustering model has a significant impact on the number of clusters needed to achieve higher Silhouette scores. When text data is not included in the model, a larger number of clusters is required to achieve better Silhouette scores.