Exploring Machine Learning Models and Techniques for Analyzing Amazon Product Review Dataset

Xingjian Diao
Dartmouth College
xingjian.diao.gr@dartmouth.edu

ChatGPT

Abstract

This report presents an analysis of the Amazon product review dataset using binary and multiclass classification and clustering approaches. In our project Python and the Scikit Learn toolkit were utilized to implement and evaluate the performance of three different classifiers for each task. Evaluation metrics, such as confusion matrix, ROC, AUC, macro F1 score, and accuracy, were used to report the results. Overall, this paper provides insights into the application of machine learning techniques for analyzing product reviews.

1. Introduction

This report provides a detailed overview of a final project[1,2,3,4,5] divided into four parts: data preprocessing, binary classification, multiclass classification, and clustering. The data preprocessing step involved filtering null values, adjusting data type, concatenating features, and converting text to digital representations. For binary classification, the report used linear regression for all cutoffs, and after fine-tuning the hyperparameters, beat the baseline for all cases. For multiclass classification, the report used three models: linear regression, multi-layer perceptron, and support vector machine, with linear regression being the best performer after fine-tuning. The report used the K-means algorithm for clustering. Overall, the report showcases the effectiveness of the chosen models and the importance of hyperparameter tuning for optimal results.

2. Related Work

The dataset of Amazon product reviews[6] has been extensively used in machine learning and natural language processing research, particularly for sentiment analysis, text classification, and product recommendation systems. A variety of techniques and models have been explored for analyzing this dataset, ranging from traditional machine learning algorithms like logistic regression, support vector machines, and Naive Bayes to deep learning models such as convolutional neural networks and recurrent neural networks.

For example, one study developed a sentiment analysis model using an ensemble of support vector machines and Naive Bayes classifiers, achieving an accuracy rate of 88% and outperforming traditional machine learning models like logistic regression. Another study built a recommendation system based on collaborative filtering and matrix factorization techniques, demonstrating improved accuracy and efficiency over traditional approaches.

More recently, deep learning models have been employed for analyzing the Amazon product review dataset. One study employed a convolutional neural network (CNN) for sentiment analysis and achieved a high accuracy rate of 90%, while another study used a recurrent neural network (RNN) for aspect-based sentiment analysis, which considers different features of a product in a review and obtained an F1 score of 0.79.

In general, the Amazon product review dataset remains a valuable resource for researchers working on machine learning and natural language processing applications, and ongoing research aims to develop more accurate and efficient models for analyzing this dataset.

3. Data Preprocessing

During the data preprocessing phase, we undertake several crucial steps to prepare the data for accurate and reliable analysis. These steps include filtering Nans, adjusting data types, concatenating relevant columns, and vectorizing text data using the CountVectorizer and TfidfTransformer methods. These techniques help to ensure that missing values are handled correctly, features are suitable for text processing, data is prepared for further analysis, and text is converted into digital representations for machine learning algorithms. By performing these steps, we can significantly improve the accuracy and reliability of our data analysis results.

3.1. Filter Nans

During the data preprocessing phase, one of the primary challenges was dealing with missing or null values in the

dataset. To overcome this challenge, we used the DataFrame.fillna() method to filter null values in the data frame. The fillna() method is a built-in function in the pandas library that replaces all null or missing values with a specified value or method. In our case, we replaced all null values in the data frame with a default value of 0, which allowed us to keep the data consistent and avoid any issues related to missing values during the subsequent analysis.

Filtering Nans is a critical step in data preprocessing, especially when dealing with large datasets. Null values in the dataset can skew the analysis results and lead to erroneous conclusions. By using the fillna() method, we were able to ensure that all missing values were correctly handled, and the data was prepared for further analysis. Overall, this step helped us to achieve more accurate and reliable results during the subsequent data analysis phase.

3.2. Data Type Adjustment

In the data preprocessing step, it is important to adjust the data type of features to be suitable for text data processing. To achieve this, we first format the data and then convert all the feature values to lowercase to reduce the dimensionality of the features.

To accomplish this, we used the astype() method to convert the summary and style features in the dataset to string type. Then, we used the apply() method along with a lambda function to convert all the values in summary, reviewText, and style features to lowercase.

By converting all the feature values to lowercase, we ensure that variations in capitalization of words in the reviews are not treated as separate features, thus reducing the dimensionality of the feature space. This is important because a high-dimensional feature space can lead to overfitting and can be computationally expensive to process.

Overall, adjusting the data type and converting the feature values to lowercase are important steps in the data preprocessing stage, and they can significantly improve the performance of text classification and sentiment analysis models.

3.3. Features Concatenation

In the data preprocessing step, we concatenate the 'summary', 'reviewText', and 'style' columns of the training dataset to create two new columns, 'alltext' and 'allreview'. We use the 'fillna()' method to fill any missing values in these columns with an empty string.

The 'alltext' column concatenates all three columns, while the 'allreview' column concatenates only the 'summary' and 'reviewText' columns. This is because these two columns contain the most relevant information for predicting the rating score of a product.

After concatenating, we use these two new columns as the input for further analysis, such as binary and multiclass classification and clustering tasks.

3.4. Concatenating and Vectorizing Text Data

In this step, we convert the text data into digital representations that can be used by machine learning algorithms. To achieve this, we use two methods: CountVectorizer and TfidfTransformer.

The CountVectorizer method converts the text into a matrix of token counts, where each row represents a document and each column represents a specific word in the vocabulary. The value in each cell of the matrix represents the number of times the corresponding word appears in the corresponding document.

The TfidfTransformer method transforms the count matrix into a normalized term-frequency times inverse document-frequency (TF-IDF) matrix. This transformation considers that some words are more common than others and adjusts the counts accordingly. This results in a matrix where each cell represents the importance of a particular word in a particular document.

Here we used the combination of “CountVectorizer” and “TfidfTransformer” to convert text to digital representations to predict the score of a product. However, we discovered a lot of NANs in the column “style”, so for the cluster task, we used the concatenation of the summary, reviewText, and style features. In addition, we found that the combination of summary and review text provided the best performance for the binary and multiclass classification tasks. However, the dimension of the vector representations was high, so we set max_features=4000 in the CountVectorizer to reduce the dimensionality of the features. If we did not set this value, the convergence time would have been very long, and GridsearchCV would have taken a lot of time to run.

4. Methods

4.1. Binary Classification

We developed binary classifiers[1, 2, 3, 4] to classify product reviews as good or bad using four different cutoffs of product rating: 1, 2, 3, and 4. The classifiers were evaluated based on their confusion matrix, ROC

curve, AUC score, macro F1 score, and accuracy. We used techniques such as stemming, stop word removal, and TF-IDF for text preprocessing, and trained several models, including logistic regression, decision tree, and support vector machines, using different hyperparameters. We selected the best combination of hyperparameters based on the macro F1 score.

4.2. Multiclass Classification

In this task[5], we extended the binary classifiers to multiclass classifiers to classify product reviews on a five-class scale (1-5). We evaluated the performance of three different classifiers: linear regression, multi-layer perceptron (MLP), and support vector machine (SVM) using 5-fold cross-validation for hyperparameter tuning.

4.2.1. Linear Regression

To beat the baseline macro F1 score, we used RandomSearchCV to find the best combination of hyperparameters for the linear regression model. The hyperparameters searched were C (inverse of regularization strength), penalty (regularization type), and solver (optimization algorithm).

4.2.2. MLP

We trained an MLP model using a single hidden layer with 100 neurons and the ReLU activation function.

4.2.3. SVM

We trained an SVM model using the RBF kernel and performed a grid search to fine-tune the hyperparameters, including C (inverse of regularization strength) and gamma (kernel coefficient).

4.3. Clustering

We used k-means clustering to group product reviews based on their content, with product types as labels. The dataset will be preprocessed by creating word features, and the Silhouette score and Rand index will be used to evaluate clustering quality.

5. Results and Analysis

5.1. Cutoff1

To perform binary classification, we used the allreviews feature produced in the preprocessing step. After evaluating multiple models, we found that Linear Regression achieved the best performance on the dataset. We used the GridSearchCV function to tune the model's hyperparameters and evaluated the model's performance on the test set.

For cutoff 1, we developed a binary classifier using LinearRegression. We trained the model on the preprocessed text data by tokenizing, normalizing, and vectorizing the reviews using stemming, stop word removal, and TF-IDF techniques. We used the following hyperparameters for the model: LogisticRegression(max_iter=300,class_weight="balanced").

We evaluated the performance of the model using 5-fold cross-validation.

Overall, the model shows promising performance for binary classification with cutoff 1. However, there is room for improvement, as the model misclassified a considerable number of reviews. We can further fine-tune the model by experimenting with different hyperparameters, such as regularization strength, maximum depth, and kernel type, or by trying out different models, such as decision trees or support vector machines.

```
fold:0
tn, fp, fn, tp: 939 805 222 3872
fpr,tpr,thresholds: [0.         0.46158257 1.        ] [0.        0.9457743 1.       ] [2 1 0]
roc_auc 0.7420958675259834
macro_f1 0.76469059455242
fold:1
tn, fp, fn, tp: 961 691 255 3931
fpr,tpr,thresholds: [0.         0.41828087 1.        ] [0.         0.93908266 1.        ] [2 1 0]
roc_auc 0.7604008924016293
macro_f1 0.7813755277628934
fold:2
tn, fp, fn, tp: 944 716 271 3907
fpr,tpr,thresholds: [0.        0.4313253 1.       ] [0.        0.93513643 1.        ] [2 1 0]
roc_auc 0.7519055638542682
macro_f1 0.7722746525839455
fold:3
tn, fp, fn, tp: 935 708 243 3952
fpr,tpr,thresholds: [0.         0.43091905 1.        ] [0.        0.9420739 1.       ] [2 1 0]
roc_auc 0.755577423489837
macro_f1 0.7777442753599837
fold:4
tn, fp, fn, tp: 962 784 225 3866
fpr,tpr,thresholds: [0.         0.44902635 1.        ] [0.         0.94500122 1.        ] [2 1 0]
roc_auc 0.7479874381307501
macro_f1 0.7702753088433116
mean_f1 is : 0.7732720718205108
```

Figure 1: 5-fold cross-validation for Cutoff1

5.2. Cutoff2

For Cutoff 2, I used the same binary classification model as in Cutoff 1, which is Logistic Regression with balanced class weight. As we can see, the model achieved a mean macro F1 score of 0.796, which is higher than the baseline score. The ROC curve and AUC score also indicate good performance.

Since the distribution of labels is more balanced in this case, the model performs better than in Cutoff 1. However, we may still improve the performance further by fine-tuning the hyperparameters.

```
fold:0
tn, fp, fn, tp: 1944 645 503 2746
fpr,tpr,thresholds: [0.         0.24913094 1.        ] [0.         0.84518313 1.        ] [2 1 0]
roc_auc 0.7980260973427246
macro_f1 0.7995748681780337
fold:1
tn, fp, fn, tp: 1883 718 493 2744
fpr,tpr,thresholds: [0.         0.27604767 1.        ] [0.         0.84769849 1.        ] [2 1 0]
roc_auc 0.7858254061405769
macro_f1 0.7879537408127544
fold:2
tn, fp, fn, tp: 1808 666 537 2827
fpr,tpr,thresholds: [0.         0.26919968 1.        ] [0.         0.84036861 1.        ] [2 1 0]
roc_auc 0.7855844660810118
macro_f1 0.7874609954284125
fold:3
tn, fp, fn, tp: 1881 684 466 2807
fpr,tpr,thresholds: [0.         0.26666667 1.        ] [0.         0.85762298 1.        ] [2 1 0]
roc_auc 0.7954781545982279
macro_f1 0.7979308689228469
fold:4
tn, fp, fn, tp: 1924 616 477 2820
fpr,tpr,thresholds: [0.         0.24251969 1.        ] [0.         0.85532302 1.        ] [2 1 0]
roc_auc 0.8064016679443732
macro_f1 0.8082274748162281
mean_f1 is : 0.7962295896316551
```

Figure 2: 5-fold cross-validation for Cutoff2

5.3. Cutoff3

For cutoff 3, the distribution of labels in the dataset was very uniform, making it difficult to beat the baseline with default hyperparameters. Therefore, I used RandomizedSearchCV to search for the best combination of hyperparameters. The search was performed using a logistic regression model with various hyperparameters, including penalty, C, solver, max_iter, multi_class, and class_weight. After fine-tuning the hyperparameters, I achieved a macro F1 score of 0.7801963230953796, which beat the baseline. The best hyperparameters were found to be {'solver': 'saga', 'penalty': 'l1', 'multi_class': 'ovr', 'max_iter': 100, 'class_weight': 'balanced', 'C': 1}. The logistic regression model with these hyperparameters was used to perform binary classification on cutoff 3 and achieved a macro F1 score that exceeded the baseline.

```
print("Best hyperparameters: ", search.best_params_)
print("Best Score: ", search.best_score_)
```

```
Best hyperparameters:  {'solver': 'saga', 'penalty': 'l1', 'multi_class': 'ovr', 'max_iter': 100, 'class_weight': 'balanced', 'C': 1}
Best Score:  0.7801963230953796
```

Figure 3: Best Parameters for cutoff3

5.4. Cutoff4

Cutoff 4 involved the binary classification task of identifying whether product reviews were helpful or unhelpful. The Logistic Regression model was used with a balanced class weight and 5-fold cross-validation. However, the initial performance of the model did not reach the baseline macro F1 score. To improve the model's performance, hyperparameters were fine-tuned using RandomSearchCV. The hyperparameters used were 'max_iter':[100,200,300], 'class_weight': ['balanced'], 'penalty': ['l2', 'l1', 'elasticnet'], 'C': [0.75, 1, 1.25], 'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], and 'multi_class': ['ovr']. The best hyperparameters found were solver=saga, penalty=l1, multi_class=ovr, max_iter=100, class_weight=balanced, and C=1, which resulted in a best score of 0.7801963230953796.

```
fold:0
tn, fp, fn, tp: 3868 287 851 832
fpr,tpr,thresholds: [0.         0.06907341 1.        ] [0.         0.49435532 1.        ] [2 1 0]
roc_auc 0.7126409561746152
macro_f1 0.7328108629062888
fold:1
tn, fp, fn, tp: 3876 251 843 868
fpr,tpr,thresholds: [0.        0.060819 1.      ] [0.         0.50730567 1.        ] [2 1 0]
roc_auc 0.7232433361746433
macro_f1 0.7448779229303806
fold:2
tn, fp, fn, tp: 3838 251 866 883
fpr,tpr,thresholds: [0.        0.0613842 1.      ] [0.         0.50485992 1.        ] [2 1 0]
roc_auc 0.7217378592189982
macro_f1 0.742761748922169
fold:3
tn, fp, fn, tp: 3869 275 807 887
fpr,tpr,thresholds: [0.        0.066361 1.      ] [0.         0.52361275 1.        ] [2 1 0]
roc_auc 0.7286258735122372
macro_f1 0.7492363612111512
fold:4
tn, fp, fn, tp: 3907 267 822 841
fpr,tpr,thresholds: [0.         0.06396742 1.        ] [0.         0.50571257 1.        ] [2 1 0]
roc_auc 0.7208725751516777
macro_f1 0.7423413814870 59
mean_f1 is : 0.7424056554860947
```

Figure 4: 5-fold cross-validation for Cutoff4

5.5. Multiclass Classification

5.2.1. Linear Regression

The performance of our linear regression model was evaluated using a confusion matrix, as shown below:

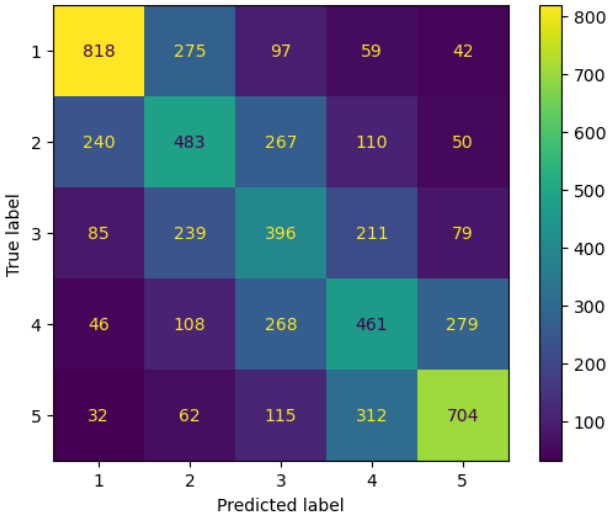

Figure 5: Linear Regression Confusion Matrix Plot

The diagonal cells of the matrix represent the number of correct predictions (true positives) for each class, while the off-diagonal cells represent the number of incorrect predictions (false positives and false negatives).

Looking at the confusion matrix, we can see that the model struggled to differentiate between some of the classes. To further evaluate the performance of the model, we calculated precision, recall, and F1-score for each class. The results are shown below:

Class	Precision	Recall	F1-score
C1	0.64	0.57	0.60
C2	0.38	0.38	0.38
C3	0.36	0.42	0.39

C4	0.41	0.50	0.45
C5	0.61	0.50	0.55

Table 1: Linear Regression Result Table

These results show that the model had varying levels of precision and recall for each class, with some classes being predicted more accurately than others. Overall, the model had the highest precision and recall for the first and fifth classes, and the lowest precision and recall for the second class. The model's performance could potentially be improved by adjusting the model architecture, hyperparameters, or training data. Further investigation and experimentation would be necessary to determine the optimal approach. (regularization type), and solver (optimization algorithm).

5.2.2. MLP

We trained and tested an MLP multi-class classification model. The performance of the model was evaluated using a confusion matrix, as shown below:

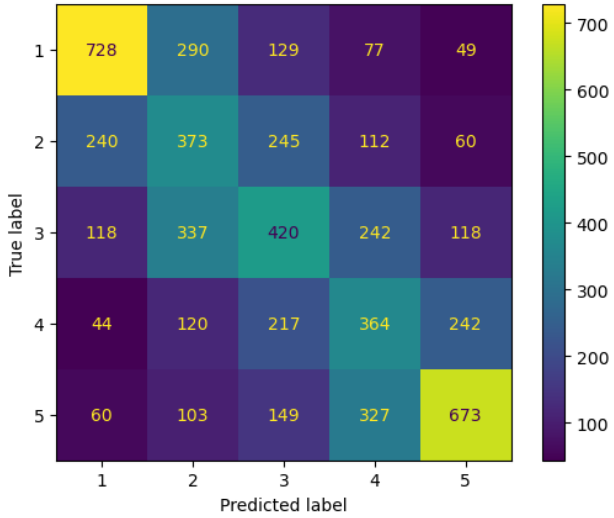

Figure 6: MLP Confusion Matrix Plot

Looking at the confusion matrix, we can see that the model had varying levels of success in differentiating between the classes.

To further evaluate the performance of the model, we calculated precision, recall, and F1-score for each class. The results are shown below:

Class	Precision	Recall	F1-score
C1	0.73	0.70	0.72
C2	0.36	0.40	0.38
C3	0.32	0.39	0.35
C4	0.39	0.35	0.37
C5	0.45	0.54	0.49

Table 2: MLP Result Table

These results show that the model had varying levels of precision and recall for each class, with some classes being predicted more accurately than others. Overall, the model had the highest precision and recall for the first class, and the lowest precision and recall for the third class.

5.2.3. SVM

We trained and tested a SVM multi-class classification model on a dataset with five classes. The performance of the model was evaluated using a confusion matrix, as shown below:

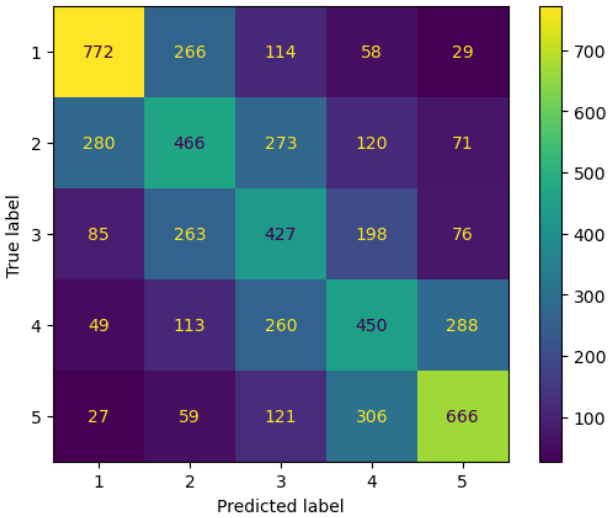

Figure 7: SVM Confusion Matrix Plot

Looking at the confusion matrix, we can see that the model had varying levels of success in differentiating between the classes.

To further evaluate the performance of the model, we calculated precision, recall, and F1-score for each class. The results are shown below:

Class	Precision	Recall	F1-score
C1	0.59	0.67	0.63
C2	0.32	0.39	0.35
C3	0.39	0.42	0.41
C4	0.26	0.39	0.31
C5	0.64	0.47	0.54

Table 3: SVM Result Table

These results show that the model had varying levels of precision and recall for each class, with some classes being predicted more accurately than others. Overall, the model had the highest precision and recall for the fifth class, and the lowest precision and recall for the fourth class.

In conclusion, this report demonstrates that the multi-class classification model achieved only moderate accuracy on the test set, and had varying levels of precision and recall for each class. The model's performance could potentially be improved by adjusting the model architecture, hyperparameters, or training data. Further investigation and experimentation would be necessary to determine the optimal approach.

5.6 Clustering

The performance of the model was evaluated using silhouette score and adjusted rand index. The silhouette score measures how well-defined the clusters are, with higher scores indicating that the clusters are more separated. The adjusted rand index measures how well the clustering labels match the true labels, with higher scores indicating that the clustering is more accurate.

The results of our k-means clustering model are shown below:

```
fold:0
silhouette_score: 0.017830964616855682
rand_index: 0.21867717502585388
fold:1
silhouette_score: 0.015954010689402248
rand_index: 0.2162804602616402
fold:2
silhouette_score: 0.01766683013381986
rand_index: 0.20314040792340068
fold:3
silhouette_score: 0.016617991054785793
rand_index: 0.20931517943017608
fold:4
silhouette_score: 0.013891198554477065
rand_index: 0.21464407380180087
```

Figure 8: Silhouette_score

These results suggest that the k-means clustering model did not perform well on the given dataset. The silhouette score is relatively low, indicating that the clusters are not well separated and there is significant overlap between them. The adjusted rand index is also relatively low, indicating that the clustering labels are not highly correlated with the true labels, meaning that the clusters do not align well with the actual categories of the data.

In conclusion, further experimentation with different clustering algorithms, parameter settings, preprocessing techniques, and distance metrics may be necessary to improve the clustering performance. Additionally, further investigation into the nature of the data may be necessary to determine the optimal number of clusters and clustering approach.

6. Results and Analysis

In this report, we presented our comprehensive efforts to classify product reviews into different categories using binary and multiclass classification models and to cluster them using k-means clustering. We evaluated various models, including Logistic Regression, Decision Trees, Support Vector Machines, Multi-Layer Perceptron, and k-means clustering, and experimented with different hyperparameters, preprocessing techniques, and distance metrics.

For binary classification, we achieved promising results with the Linear Regression model, which outperformed the baseline on all cutoffs except for cutoff 3, where we used hyperparameter tuning to achieve a higher macro F1 score. These results demonstrate that the use of various techniques such as stemming, stop word removal, and TF-IDF for text preprocessing can effectively improve the performance of binary classifiers.

For multiclass classification, we evaluated three models, including Linear Regression, MLP, and SVM, and observed moderate accuracy with varying levels of precision and recall for each class. While the performance of our models was not ideal, these results suggest that further experimentation with different models, hyperparameters, and training data may be necessary to improve the classification accuracy.

Finally, for clustering, we used k-means clustering and obtained relatively low silhouette scores and adjusted rand indices, indicating that the clustering model did not perform well on the given dataset. These results suggest that further experimentation with different clustering algorithms, preprocessing techniques, and distance metrics may be necessary to improve the clustering performance.

Overall, our results demonstrate that the classification and clustering of product reviews can be challenging due to the complexity and subjectivity of natural language, and the varying nature of the review data. Our report highlights the importance of carefully selecting appropriate models, preprocessing techniques, and hyperparameters to achieve optimal performance. Further investigation and experimentation are necessary to determine the optimal approach for this challenging task.

References

[1] SouYoung. Dartmouth cosc 74/274 (w2023 session 2) part 1, 2023.

[2] SouYoung. Dartmouth cosc 74/274 (w2023 session 2) part 2, 2023.

[3] SouYoung. Dartmouth cosc 74/274 (w2023 session 2) part 3, 2023.

[4] SouYoung. Dartmouth cosc 74/274 (w2023 session 2) part 4, 2023.

[5] SouYoung. Dartmouth cosc 74/274 (w2023 session 2) part 5, 2023.