Analyzing and Categorizing Amazon Product Reviews: Machine Learning and Statistical Data Analysis

Supritha Shankar Rao
Dartmouth College
supritha.s.rao.th@dartmouth.edu

Abstract

Machine Learning and Statistical Data Analysis is performed on the Amazon Product Reviews. These are analyzed and categorized using three techniques in Machine Learning, Binary Classification, Multi-Class Classification and Clustering. The various classifiers used to obtain the results include the Logistic Regression, Decision Tree and the Random Forest Classifier. This is a COSC74/274 Machine learning course project and the main goal is to implement the concepts covered in class for a real-world problem with a real-world dataset.

1. INTRODUCTION

The world had been changing in a great pace towards digitalization and there has been a sudden boom in the e-commerce industry. Today, we encounter that this industry has seen only an upward trajectory and people shopping through these websites and buying things on the net is becoming the new norm.[1]. One such e-commerce site that has been undoubtedly the market king is Amazon, like seen the Figure 1 below, [2]

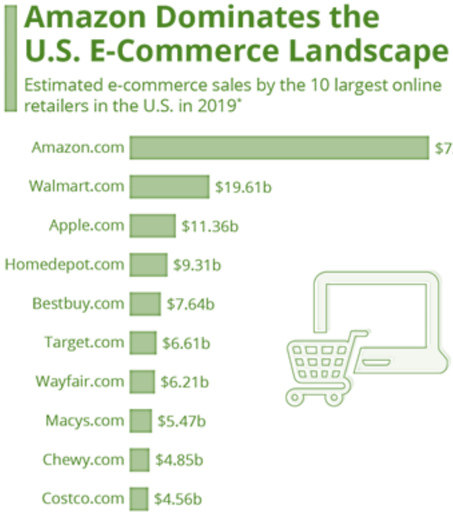


Figure 1:10 online e-commerce sales estimates

It’s not just with the selling of goods through these websites but they offer much more services and one prominent useful feature that they possess are the product reviews from the people who have purchased and used them already. There are ratings out of 5 stars, comments and feedbacks about products which gives in a lot of insightful information to the new customer in order to make the purchase and also the company to decide on what's their star product and which are the ones that is bringing their brand down.

Hence, as a part of the Machine Learning and Statistical Data Analysis final project, we will be looking at drawing insights from the Amazon Review Data set provided and use various techniques like binary classification, multi-class classification and clustering. Under these we will make use of logistic regression, decision tree classifier, random forest classifier and k-means methodologies to analyze and categorize the product reviews from Amazon.

1.1. Amazon Review Dataset

To perform our analysis two files named Training.csv and Test.csv is provided. The former file consists 29,189 rows of review-related information as shown in the Table 1 below, and the later one contains exactly all the same features as the training file, but the overall variable is withheld for predictions from the analysis for each product.

Fields	Description
overall	Product’s rating on a scale of (1-5)
verified	A Boolean variable denoting if the review has been verified by Amazon
reviewTime	Time of review
reviewerID	The unique ID of the reviewer
asin	Product ID. One product will have many different reviews
reviewerName	Encoding reviewer’s username
reviewText	The Amazon review
unixReviewTime	Unix time of review
vote	Number of people who voted this review as being helpful
image	If there is an image, link to the image
style	If there is style information then it is embedded in a dictionary here.
Category	Amazon product category
summary	brief summary of the review

Table 1: Amazon Review Dataset

2. OBJECTIVES

We will now see the various objectives and the three tasks which are, binary and multi-class classification, clustering that is performed on our amazon review dataset to drawn meaningful insights.

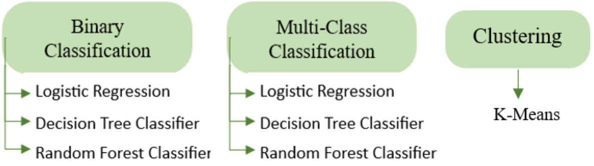


Figure 2: Clustering, Binary & Multi-Class classification

2.1. Binary Classification

The objective is the develop binary classifiers to classify the product reviews as good or bad. the cutoff of "goodness" of product rating is 1,2,3 and 4 for the four models developed. Here the cutoff is not an input to the model. For example, when cutoff=4, all samples with a rating of <=4 will have label 0, and all the samples with a rating >4 have label 1. To carry this task, 3 classifiers were used, they are:

- **Logistic Regression:**
  Here we using the liblinear solver, where the model gives out the estimates of the probability of the event, that is 0 or 1 based on the logistic function.[3]
- **Decision Tree Classifier:**
  They are non-parametric supervised learning method used for used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation. [4]
- **Random Forest Classifier:**
  It's a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy.[5]

2.2. Multi-Class Classification

The objective is to turn the binary classifier into a multiclass classifier where the target classes are 1,2,3,4 and 5. So here we classify the product rating on a five-class scale. To achieve these, we once again use the three classifiers mentioned above which are the Logistic regression but here using the saga solver, followed by decision tree and random forest classifier.

2.3. Clustering

Here the objective is to cluster the amazon product reviews in the test dataset and create word features from the data and perform clustering using the k-means. K-Means is one of the simplest and a popular unsupervised machine learning algorithms where the main objective is grouping similar data points together, in our case in the product review dataset and discover patterns.[6]

3. METHODS

In this section a list of different methodologies used are mentioned which includes the various libraries used, feature selection, data preprocessing and hyper parameter tunings.

3.1. Libraries Used

To avoid complexity in coding and make it a simple readable code, a variety of built-in python libraries are used like

- **Pandas:**
  this provides fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.[7] In the project these are used to fetch and create and manipulate the data frames from the csv files provided.
- **NumPy:**
  It is an open source Python library for working with numerical data in python, and it's at the core of the scientific Python and PyData ecosystems.[8] In the analysis we use these for constructing and working around the matrices and to perform mathematical and simple functions on array.
- **MatPlotLib:**
  It provides comprehensive library for creating static, animated, and interactive visualization.[9] Here, in this project we use this library to plot graphs, scatterplots and majorly used to plot the ROC Curves for the analysis.
- **Scikit-learn:**
  This library in python provides many unsupervised and supervised learning algorithms. It's built upon some of the technologies like NumPy, pandas, Matplotlib (mentioned above). The functionality that it provides include- Regression, classification, clustering, model selection, preprocessing. It's a great aid for creating robust machine learning programs.[10]

  In the project, these are used for running the various machine learning models, LogisticRegression, RandomForestClassifier, DecisionTreeClassifier and K-Means. Further in order to preprocess the given data and pipeline them as inputs to the models, we used TfidfVectorizer, OneHotEncoder, Column Transformer, Count Vectorizer.

3.2. Feature Selection

The objective is to turn the binary classifier into a multiclass classifier where the target classes are 1,2,3,4 and 5. So here we classify the product rating on a five-class scale. To achieve these, we once again use the three classifiers mentioned above which are the Logistic regression but here using the saga solver, followed by decision tree and random forest classifier.

Fields	Feature ✓ / Discard ✗	Reason
verified	✗	Didn't consider after few iteration of trials
reviewTime	✗	Not a significant information
reviewerID	✗	Not a significant information
asin	✗	Not a significant information
reviewerName	✗	Not a significant information
reviewText	✓	Good predictor of the overall rating
unixReviewTime	✗	Not a significant information
Vote	✓	Shows the quality of reviews
Image	✗	Didn't consider after few iteration of trials
Style	✗	Didn't consider after few iteration of trials
Category	✓	Improves the predictive power of other features that are considered
summary	✓	Good predictor feature along the reviewText

Table 2: Feature Selection

Also, when it came to clustering, the same features yielded low rand index and silhouette score and hence different features were selected. Clustering involves grouping of similar data and "style" was one such field that had different JSON values for the variety of categories. Hence, for clustering I went ahead with style as a feature and this on combination with other variables didn't show up any significant difference and hence at the end, I used only style as the feature of clustering.

3.3. Data Preprocessing

For the classification, after the selection of the features, data preprocessing was done which included-

Replacing the nan values with empty ('') characters to avoid any disturbance while processing. Followed by the conversion of the category from dtype=String to dtype=category for more efficient usage of the data. Similarly, to know the quality of reviews the 'vote' feature was convert from dtype=String to dtype=int. In order to use the two text fields which were 'summary' and 'reviewText', CountVectorizer was used to encode the word features. This helped to populate the sparse matrix to the model. OneHotEncoder was used on 'category' and 'vote' in order to populate categorial information to the model. ColumnTransformer and pipelines class was eventually used to get a sequential data to the models and into the transformers as seen in Figure 3.

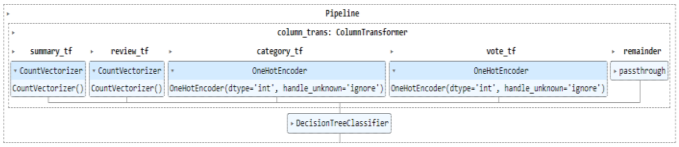


Figure 3: Pipeline for Decision Tree Classifier

In the same space for clustering, on freezing the usage of just one feature which is "style", TfidfVectorizer was used in order to encode it. The specialty of TfidfVectorizer is that it helps to impart attention to the frequency as well as the importance of the different words in the text-based field(style).

4. RESULTS AND ANALYSIS

In this section the final analysis and results from the various models and classifiers are mentioned.

4.1. Classification

For each of the binary classification and the multi-class classification we used Logistic regression, decision tree classifier and random forest classifier

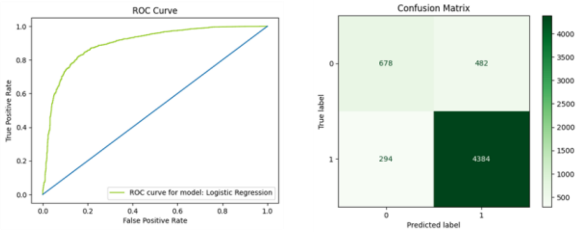
4.1.1 Logistic Regression

So, to go ahead with the logistic regression, I started off with default LogisticRegression method without any tuning and it didn't converge so upon trying more I found that the 'liblinear' solver was a much suited as coordinate descent algorithm was used which works well even with noisy data and large set of fields. Hence for binary classification I freezed on using this method. For multi-class I again began with the default, but it showed similar results and then I went ahead with 'liblinear' which also failed to perform above par and hence used the 'saga' solver that used stochastic average gradient descent algorithm. Finally, this solver used with setting the hyper parameter as "multinomial".

Then on finalizing of the solver, it was time for tuning the parameters for further optimizing the models. The hyper-parameter "C", "l1_ratio" and "max_iter" had to be carefully set. "C" setting was tricky as a value too low would make the model underfit and the same on been high would lead to overfitting. "max_iter" was used so that the solver doesn't get stuck in the local minima and helps controlling the iterations. The "l1_ratio" determines the regularization penalties applied to the model. The hyper-parameters are further tuned using "GridSearchCV" which is also from Scikit-Learn, which finds the best fit with 5 fold cross validation.

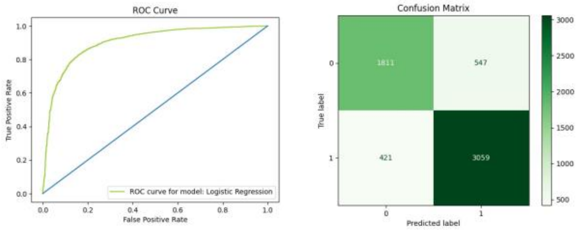
The confusion matrix and ROC Curve for the logistic regression for both binary and multi-class classification can be found in Figure 4 below,

Binary Classification cutoff=1



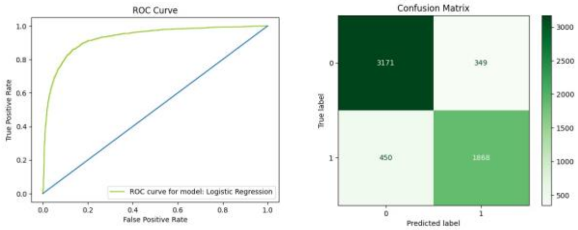
Hyper-Parameters	Macro-F1 Score(Test)	Macro-F1 Score	AUC	Accuracy
C: 0.5, max_iter: 50, penalty: ‘l2’, solver=liblinear	0.77899	0.7773	0.8975	0.8670

Binary Classification cutoff=2



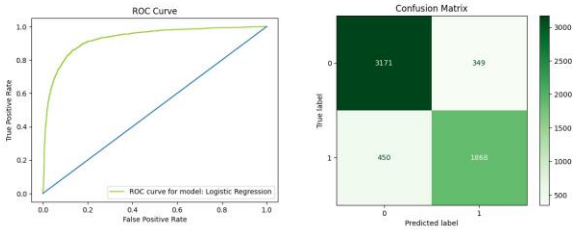
Hyper-Parameters	Macro-F1 Score(Test)	Macro-F1 Score	AUC	Accuracy
C: 0.5, max_iter: 50, penalty: ‘l2’, solver=liblinear	0.82617	0.8262	0.9036	0.8341

Binary Classification cutoff=3



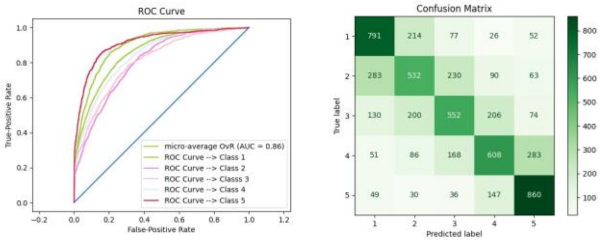
Hyper-Parameters	Macro-F1 Score(Test)	Macro-F1 Score	AUC	Accuracy
C: 0.1, max_iter: 50, penalty: ‘l2’, solver=liblinear	0.85576	0.8559	0.9276	0.8631

Binary Classification cutoff=4



Hyper-Parameters	Macro-F1 Score(Test)	Macro-F1 Score	AUC	Accuracy
C: 0.1, max_iter: 25, penalty: ‘l2’, solver=liblinear	0.81512	0.8184	0.9210	0.8946

Multi-Class Classification



Hyper-Parameters	Macro-F1 Score (Test)	Macro-F1 Score	AUC	Accuracy
C=1.2, model__penalty=l2	0.58854	0.56897	0.86	0.57262

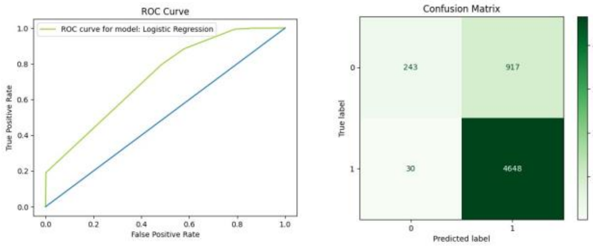
Figure 4: Logistic Regression-ROC Curve & Confusion Matrix + classification metrics.

Figure 4 has a column named “Macro-F1 Score (Test)”, this represents the Kaggle competition scores (this is the best performed model when compared to the other two.

4.1.2 Decision Tree Classifier

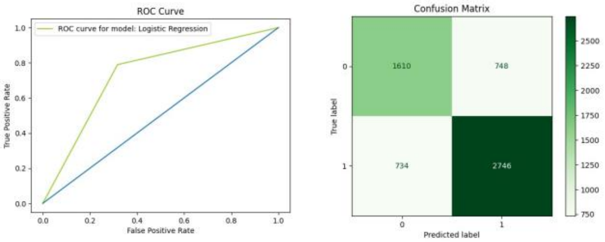
The next model that was put into try was the Decision Tree Classifier. On execution we could see a dip in the F1 score, AUC and accuracy. On further understanding how these model works, I went ahead tuning in the hyper-parameters’ “splitter” and “max_depth” to make the model perform better than the trial run. By tuning it as “splitter”, we get the advantage of controlling the splitting nodes which directly shows results on an accuracy and the decision tree been formed. The other options were the usage of “random” which would most probably lead to lower accuracy and a simpler decision tree and “best” would result in high accuracy. Also, the “max_depth” used in this case, helped find the ideal depth of the tree with best fit, while not getting overfitted or underfitted. This eventually helps in generalizing to new data and escalate its predictive power. The hyper-parameters are further tuned using “GridSearchCV” which is also from Scikit-Learn, the confusion matrix and ROC Curve for the decision tree classifier for both binary and multi-class classification can be found in Figure 5 below,

Binary Classification cutoff=1



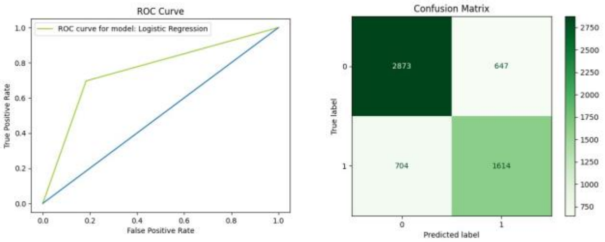
Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
max_depth:5, model__splitter=“best”	0.62334	0.7255	0.8377

Binary Classification cutoff=2



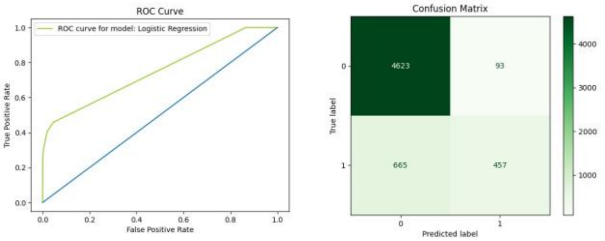
Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
max_depth:5, model__splitter=“random”	0.7361	0.7359	0.74614

Binary Classification cutoff=3



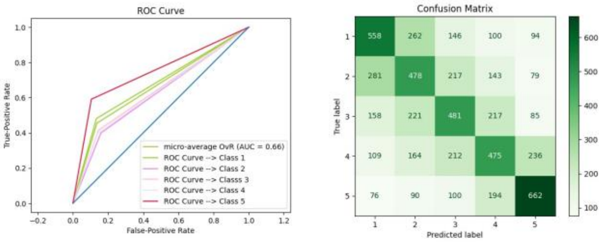
Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
max_depth:50, model__splitter=“random”	0.75729	0.756	0.76858

Binary Classification cutoff=4



Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
max_depth:5, model__splitter=“best”	0.73544	0.750	0.87016

Multi-Class Classification



Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
model__max_depth=None, model__splitter=“random”	0.4555	0.66	0.4546

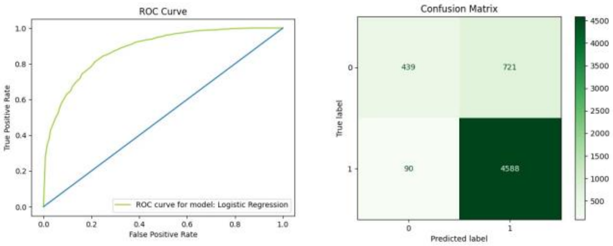
Figure 5: Decision Tree Classifier-ROC Curve & Confusion Matrix + classification metrics.

4.1.3 Random Forest Classifier

Finally, the last model implemented is the Random Forest Classifier, while this model had a very long runtime and since the it is usually the combination of decision tree, I was in the run for it to perform better than the rest. Here using random subset of the features fed to the model, a decision tree is formed. This randomness is expected to improve the scores and the accuracy.

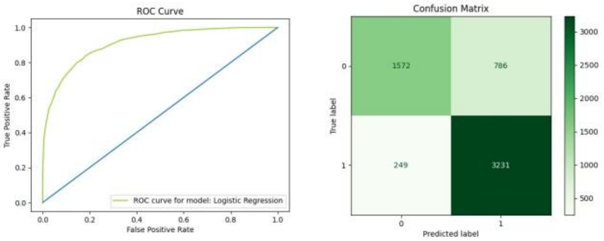
Again, “max_depth” is used for hyper-parameter tuning along with “max_features” as it controls the features list while splitting into each node. It does help in finding the right number of features that has to be used per tree which eventually leads to an improved performance of the classifier.

Binary Classification cutoff=1



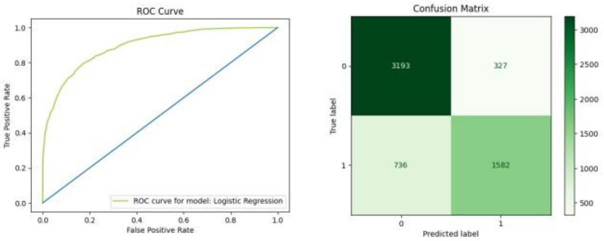
Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
model__max_depth=None, model__max_features=“0.4”	0.7193	0.880	0.8610

Binary Classification cutoff=2

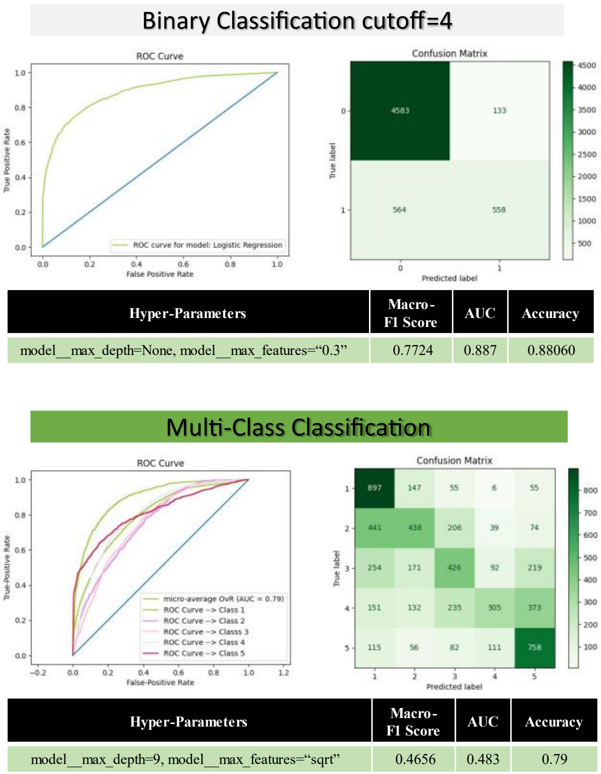


Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
model__max_depth=None, model__max_features=“sqrt”	0.8071	0.910	0.8227

Binary Classification cutoff=3



Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
model__max_depth=None, model__max_features=“0.3”	0.8029	0.895	0.817



Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
model__max_depth=None, model__max_features="0.3"	0.7724	0.887	0.88060

Hyper-Parameters	Macro-F1 Score	AUC	Accuracy
model__max_depth=9, model__max_features="sqrt"	0.4656	0.483	0.79

Figure 6: Random Forest Classifier-ROC Curve & Confusion Matrix + classification metrics.

The confusion matrix and ROC Curve for the random forest classifier for both binary and multi-class classification can be found in Figure 6 below,

4.2. Clustering

K-Means clustering is used here in order to cluster our product reviews. The product categories are the labels for the clustering. There are 6 product categories namely automotives, CDs, grocery, cell_phones, sports and toys and therefore we used 6 clusters. Like mentioned in earlier section, the field "style" proved to be enough for achieving a silhouette score, and a higher rand index with reviewText along with style. The feature "style" seemed to be pertinent to a category as usually the ones within an umbrella of category hold in almost same words in the style for products. Hence, I went ahead using only "style". Here again GridSearchCV was used to tune for securing an even better score.

Again, to increase the performance, tuning of "max_df" and "min_df" (using TfidfVectorizer) helped in controlling the inclusion of terms in the resultant vectors of text. This led to better performance as tuning of these helps in a better control of vocabulary size and content. Lastly, on tuning "max_features", the overall number of terms included in the resulting vector representation of text was limited and this led to a more perfect clustering.

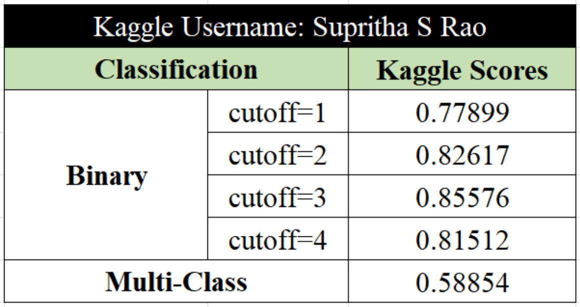
Table 3 shows the various features used, along with the hyper-parameters along the Silhouette Score and Rand Index.

Features	Hyper-parameters	Rand Index	Silhouette Score
style	max_df: 0.5, max_features: 3, min_df: 0.1, max_iter: 300	0.20005	0.99813
Style, reviewText	max_df=0.7,min_df=0.1, max_iter= 300	0.99893	0.25034

Table 3: Clustering-KMeans

5. CONCLUSION

From all the results seen for classification, we can conclude that the Logistic Regression when tuned in appropriately gives us results that overpowers the model results from either the Random Forest Classifier or Decision Tree Classifier for the provided Amazon Product Review Dataset. Hence, it's safe to tell that with proper feature selection followed up by data pre-processing and hyper-parameter tuning, desired results and be obtained. Coming to the clustering we saw how again the right set of parameters and clustering can help produce larger Silhouette score and Rand Index.



Kaggle Username: Supritha S Rao		
Classification		**Kaggle Scores**
Binary	cutoff=1	0.77899
	cutoff=2	0.82617
	cutoff=3	0.85576
	cutoff=4	0.81512
Multi-Class		0.58854

Figure 7: Kaggle

References

[1] Fokina, M. (2023). Online Shopping Statistics: Ecommerce Trends for 2023. Tidio. https://www.tidio.com/blog/online-shopping-statistics/FirstName Alpher and FirstName Fotheringham-Smythe. Frobnication revisited. *Journal of Foo*, 13(1):234–778, 2003.

[2] Richter, F. (2020, August 7). Amazon Dominates the U.S. E-Commerce Landscape. Statista Infographics. https://www.statista.com/chart/14043/top-10-online-stores-in-the-us/FirstName Alpher and FirstName Gamow. Can a computer frobnicate? In *CVPR*, pages 234–778, 2005.

[3] *What is Logistic regression? | IBM*. (n.d.). https://www.ibm.com/topics/logistic-regression

[4] *1.10. Decision Trees*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/tree.html#tree

[5] *Machine Learning Random Forest Algorithm - Javatpoint*. (n.d.). www.javatpoint.com. https://www.javatpoint.com/machine-learning-random-forest-algorithm

[6] Ecosystem, E. (2022, May 17). Understanding K-means Clustering in Machine Learning. Medium.

https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[7] pandas - Python Data Analysis Library. (n.d.). https://pandas.pydata.org/

[8] *NumPy: the absolute basics for beginners — NumPy v1.24 Manual*. (n.d.). https://numpy.org/doc/stable/user/absolute_beginners.html

[9] Matplotlib — Visualization with Python. (n.d.). https://matplotlib.org/

[10] Codecademy. (n.d.). What is Scikit-Learn? *Codecademy*. https://www.codecademy.com/article/scikit-learn