

AND S



INFORMATION

GROUP 1

at Arthur

Moe Myint Arthur
Arnav Dighe
Kyanne Lam
My Nguyen

Abstract

The purpose of this paper is to report the process of handling the ISA 414 final project, Evaluating Amazon Brands. The paper serves as a documentation of how we gathered the data, treated the data (cleaning, mining, modeling, etc.), our conclusions from the output, recommended improvements and potential future plans from the processed results. This paper will touch on the business problem of what, why, and how we performed these tasks, the target users, data wrangling, exploratory data analysis, data modeling, model evaluation, and recommendations based on final results. Methods used include data cleaning using PorterStemmer, calculating sentiment scores using Compound Vader Sentiment Score, replacing each word with its associated sentiment score from suitable cluster (positive vs negative) to compute average closeness sentiment score using Word2Vec with K Means Clustering. Results drawn from activities mentioned above serve as metrics for a comparison between a model that uses external sentiment data from Reddit vs one that does not. We find that the difference in performance between these two types of models are not significant. Among models we used, XGBoost performs better overall than Random Forest in classifying the star rating of reviews. No determined conclusions should be made towards the validity and quality of reviews from Amazon until further model and dataset improvements. For future inferential improvement, we recommended trying out other methods, like BERT or pre-trained Word2Vec models, as well as gathering more external data for sentiment analysis, or better yet, a source of external sentiment data for individual products, and to do more extensive hyperparameter tuning for current models or try other models (Extra Trees Classifier, etc.).

Introduction

In this project, we are evaluating Amazon brands and recommending the best ones based on sentiment analysis of customer reviews. Amazon reviews tell us how satisfied a customer is with the product they purchased. To verify the quality of Amazon reviews, we looked into other data sources to see if brand quality and star-rating are relevant. We want to identify brands with great customer service and highest rated products to find good brands. After that we asked ourselves, “are the reviews truly reflective of the real image of the brand?” We need to explore if there is some discrepancy or inaccuracy between ratings and the related brands. Therefore, we collected customer sentiment from Reddit to see what people are really saying about these brands and whether the opinion is different from the reviews.

We want to target customers of all ages that use Amazon to buy products along with the sellers on Amazon. The sellers can keep track of the customers’ responses to their products. Buyers that have the ability to see our model can be more certain of the brands or products that they buy from random sellers based on reviews. This would eventually lead to improved product sales through analyzing customer sentiment and help users to identify the best brands.

Essentially, the problem we are tackling is making sure that customer reviews are accurate and true representations of the brand or product that they are applicable to. This matters because sellers and buyers should be sure of the product they are buying or selling and that they are getting the product that is true to what they are looking for. Sometimes, items can be misleading online whether it be in size or just blatant misrepresentation of the product in pictures. We hope to reduce the possibility of customers buying products that are not what they appeared to be through a sentiment analysis of the customer reviews.

Gathering Data

The Amazon’s Customer Reviews data is retrieved from Julian McAuley, UCSD. This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014 from different areas. For this project, we only focus on the reviews for the Electronics category with the storage size of 5GB along with its 1.7GB metadata file. Following here is a snippet of the original JSON reviews file.

```
{  
  "reviewerID": "A2SUAM1J3GNN3B",  
  "asin": "0000013714",  
  "reviewerName": "J. McDonald",  
  "helpful": [2, 3],
```

```
"reviewText": "I bought this for my husband who plays the piano. He is having a wonderful  
time playing these old hymns. The music is at times hard to read because we think the book was  
published for singing from more than playing from. Great purchase though!",  
"overall": 5.0,  
"summary": "Heavenly Highway Hymns",  
"unixReviewTime": 1252800000,  
"reviewTime": "09 13, 2009"  
}
```

where

reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B

asin - ID of the product, e.g. 0000013714

reviewerName - name of the reviewer

helpful - helpfulness rating of the review, e.g. 2/3

reviewText - text of the review

overall - rating of the product

summary - summary of the review

unixReviewTime - time of the review (unix time)

reviewTime - time of the review (raw)

The metadata file has the following structure:

asin - ID of the product, e.g. 0000031852

title - name of the product

price - price in US dollars (at time of crawl)

imUrl - url of the product image

related - related products (also bought, also viewed, bought together, buy after viewing)

salesRank - sales rank information

brand - brand name

categories - list of categories the product belongs to

We then imported these two files into MongoDB for its flexibility and power in creating a single unified view in ways that other databases cannot [1]. It enables document embedding to describe nested structures and is easily adaptable to data variations across document generations [1]. As a result, a resilient repository is created that doesn't malfunction or require redesigning whenever something changes [1]. MongoDB's scale-out architecture can support humongous databases in terabytes and can also be deployed and run on a desktop, a massive cluster of computers in a data center, or in a public cloud for collaboration between members of our team [1].

Before we moved on to our other data source, we performed exploratory data analysis on the reviews data that we discussed above. We believe that by performing this kind of preemptive

analysis, we can identify trends or other supporting information that will help guide us further into the project.

We discovered that *star_rating* concentrated heavily right towards the positive end. We inferred that either customers tend to give overly optimistic reviews of products or this could be a sign that some of the reviews are false and padded by sellers to improve the image of their goods. Figure 1 illustrates the over-optimism of *star_rating*. In the figure, we grouped 4-5 star ratings into group 1 and the rest into group 0. Further explanation will be provided below. Observe that the amount of 4-5 star ratings is over twice as much as all other ratings combined.

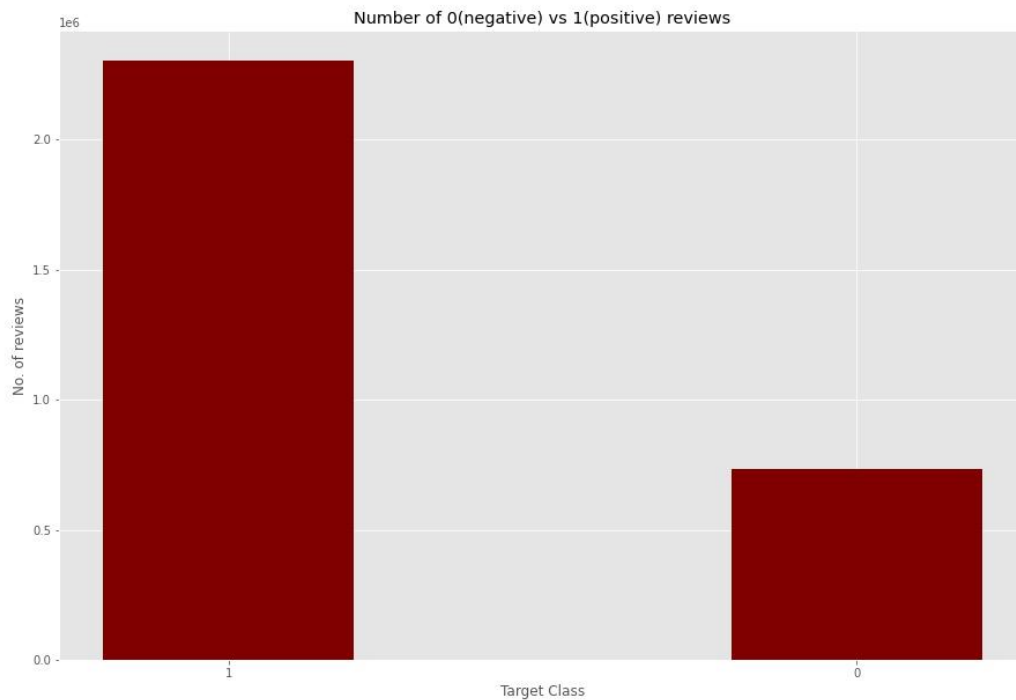


Figure 1: *star_rating* Is Heavily Optimistic

A detailed explanatory discovery analysis in html file is attached for an indepth insight into our dataset. In addition to the Amazon Reviews data, we aimed to utilize another source of data for an external perspective on the sentiment of customers about brands. We discussed the use of Twitter, because it reaches the greatest number of consumers and is most representative of the general demographics. However, there were limitations and issues with gaining access to the Twitter API for data gathering.

Instead, we utilized posts from Reddit to gain an external perspective on the sentiment of customers about brands. Our first approach was to use the Reddit API to loop through the list of

brands we extracted from the Amazon Reviews dataset and search for matching names of subreddits (specific pages for groups of Reddit posts). However, this limited our data gathering to only well-known brands that have a subreddit named after them which yielded us only over 3000 rows, not nearly enough for analysis. We had an even greater concern; whenever our data gathering code encountered a brand without a subreddit, the loop would crash. Hence, we were looking at manually gathering Reddit posts for each brand, an impractical task. Figure 3 illustrates our first iteration.

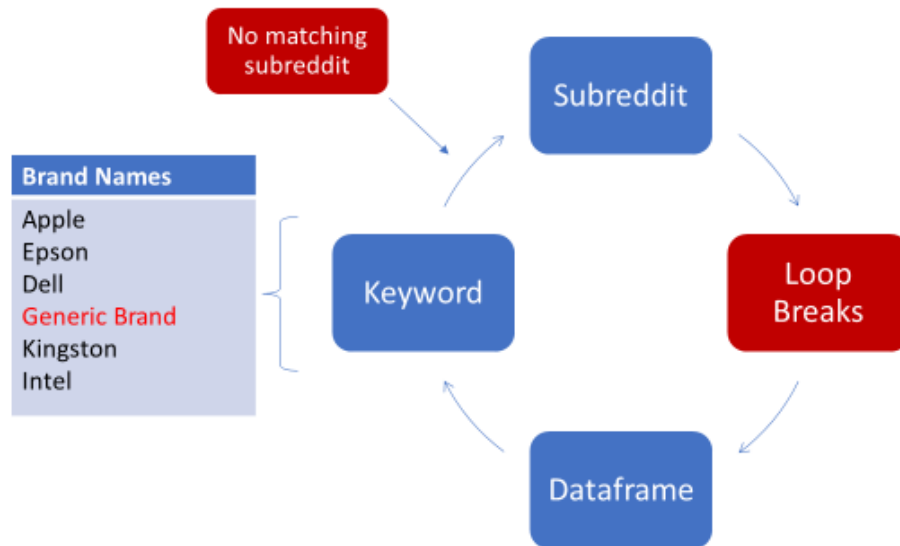


Figure 3: First iteration of Reddit Data Gathering

To circumvent this issue, we decided to discard the main Reddit API which restricted us to searching for specific subreddits. We chose PushShift API with Team 4's suggestion, a more lenient API that did not restrict us by subreddit. With instructions from Pypi (Python Package Index), we used Pushshift to search through our list of brands across the entire Reddit database, filtering for data between 2009-2015 to be consistent with the Amazon Reviews dataset. This shift not only allowed us to gather significantly more data than before (from 3,000 rows to over 19,000), but also allowed us to automate our process in a loop that was not prone to breaking. This API was also not constrained to API security or scraping limits like the Twitter API. Hence, it was the perfect data source for the scope of this project. Figure 4 is a sample of raw data gathered with the Pushshift API.

selftext	title	brand
	b-practical's User Profile - Barnes & Noble	Barnes & Noble
	My MyWishList by Barbara Carr - Barnes & N...	Barnes & Noble
	rowenacherry's Library - Barnes & Noble	Barnes & Noble
[deleted]	Had a coffee at this Barnes & Noble and di...	Barnes & Noble
[deleted]	Had a coffee at this Barnes & Noble and di...	Barnes & Noble

Figure 4: Sample Pushshift Data

Data Cleaning and Feature Engineering:

After we gathered data from both Amazon and Reddit, we took several steps to sanitize and transform to suitable formats for our analysis. As textual data is one of the main focus of our project, special characters and emoticons removal are required to be removed to achieve accurate sentiment scores.

Hence, for both the Amazon Reviews and Pushshift API datasets, we used various sub-libraries under Natural Language Toolkit for text preprocessing. Since review sentiment is our main focus, stopwords like prepositions, pronouns, conjunctions will not provide any useful information for our analysis [2]. Examples of a few stop words in English are “the”, “a”, “an”, “so”, “what”. For our project, *stopwords* and *punkt* from *nlk* are utilized since they are one of the most diverse and well-known libraries in the natural language processing area. Additionally, *PorterStemmer* is also utilized during our data cleaning process to stem words. With stemming, words are reduced to their word stems. A word stem is simply an equal to or smaller form of the word; it need not be the same root as a morphological root from a dictionary. Rule-based stemming algorithms are typical and are considered a heuristic process that lops off the ends of words. A word is looked at and passed through a sequence of conditionals that determine how to trim it down. Observe from Figure 4 that rows often did not have *selftext* (body of the post). To get a more salient analysis with missing *selftexts*, we decided to combine the *title* of each post to the *selftexts* into a column called *alltext*.

For the Amazon Reviews dataset, we split the helpful column into *helpful_yes* and *helpful_no*, separating the two numbers in the original column. We also removed dollar signs from the *price* of products. Considering that we want to classify whether we should recommend brands based on their predicted star rating, we converted the *star_rating* variable into either 0 for negative sentiment or 1 for positive for sentiment. Based on the skewness of our data, we chose to use 4-5 stars as the cutoff for assigning a 1, and assigned 0 for lower values. In cases where price data

was missing, we imputed the mean price for that row's specific brand. We also perform undersampling class 1 to be equal to class 0 in the train set to solve our imbalance issue mentioned earlier in the above EDA discussion.

	reviewText	helpful_no	helpful_yes	target	price_cleaned
	great when it works. if it loses connection co...	0	0	0	9.55
	this tablet exceeded my expectations in terms ...	3	5	1	178.99
	the replacement did do the job. however, the &...	1	1	0	17.35
	i have five internal and external hard drives ...	0	0	1	18.99
	this is exactly what i needed for a yagi anten...	2	2	1	13.75

Figure 5: Processed Amazon Reviews Data

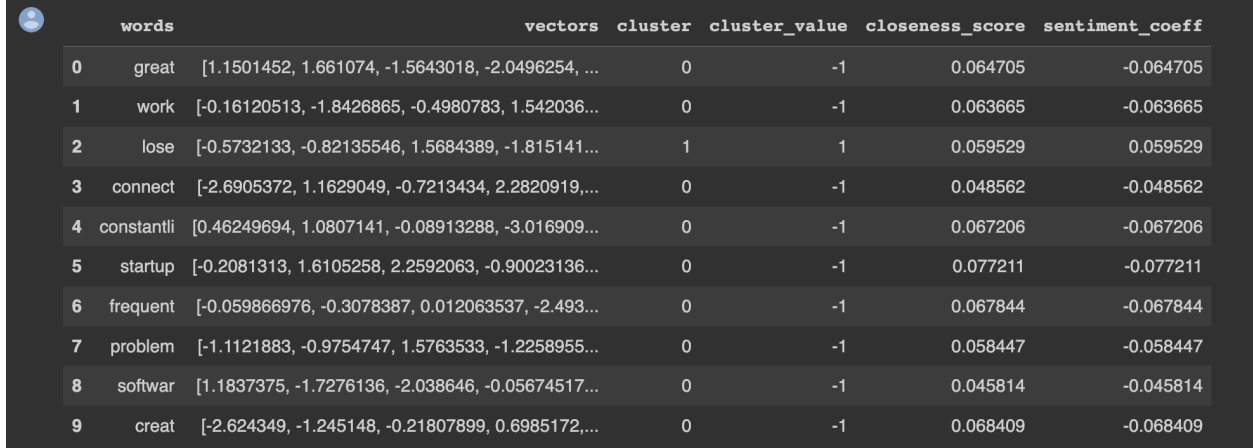
	brand	selftext_clean	title_clean	alltext_clean
0	Barnes & Noble		user profil barn amp nobl	user profil barn amp nobl
1	Barnes & Noble		mywishlist barbara carr barn amp nobl	mywishlist barbara carr barn amp nobl
2	Barnes & Noble		rowenacherri librari barn amp nobl	rowenacherri librari barn amp nobl
3	Barnes & Noble	delet coffe barn amp nobl discov charg wifi thru amp...	delet coffe barn amp nobl discov charg wifi th...	delet coffe barn amp nobl discov charg wifi th...
4	Barnes & Noble	delet coffe barn amp nobl discov charg wifi thru amp...	delet coffe barn amp nobl discov charg wifi th...	delet coffe barn amp nobl discov charg wifi th...

Figure 6: Processed Reddit Data

We then used Vader from NLTK to find sentiment scores for the Amazon review text as well as combination text from Reddit. In an article for the Journal of Big Data, Xing and Fang define sentiment as the “attitude, thought, or judgment prompted by feeling.” Vader helps us identify these emotions, such as positive or negative, by assigning scores based on what it detects. By analyzing these scores in predictive models, we can discern how people feel about certain entities (Xing & Fang, 2015).

As required by our business case, we created two datasets. One contained external sentiment scores from Reddit and one contained purely Amazon reviews. Lastly, we had a Word2Vec Model with K Means Clustering to replace each word with the associated sentiment score from the suitable cluster (positive vs negative). As one of the most popular techniques to learn word embeddings, Word2vec uses a two-layer neural network to cluster words together by a trained set of word vectors. Its input is a text corpus and its output is a set of vectors [3]. Word embedding via word2vec can make natural language computer-readable, then further implementation of mathematical operations on words can be used to detect their similarities [3]. In our sentiment analysis, we first initiated an analyzer and then iterated through each sentence in the corpus. Next we analyze the sentiment and provide the output of each sentiment score (pos, neg, neu, compound). When it comes to the K Means Clustering, we had a model with two clusters, 10,000

maximum iterations, random state being true, and the number of iterations is 50. We went through the process of scaling the data as well because that dataset was not scaled well to begin with. Therefore, before training the algorithm, we will need to scale our data down.



	words	vectors	cluster	cluster_value	closeness_score	sentiment_coeff
0	great	[1.1501452, 1.661074, -1.5643018, -2.0496254, ...	0	-1	0.064705	-0.064705
1	work	[-0.16120513, -1.8426865, -0.4980783, 1.542036...	0	-1	0.063665	-0.063665
2	lose	[-0.5732133, -0.82135546, 1.5684389, -1.815141...	1	1	0.059529	0.059529
3	connect	[-2.6905372, 1.1629049, -0.7213434, 2.2820919,...	0	-1	0.048562	-0.048562
4	constantli	[0.46249694, 1.0807141, -0.08913288, -3.016909...	0	-1	0.067206	-0.067206
5	startup	[-0.2081313, 1.6105258, 2.2592063, -0.90023136...	0	-1	0.077211	-0.077211
6	frequent	[-0.059866976, -0.3078387, 0.012063537, -2.493...	0	-1	0.067844	-0.067844
7	problem	[-1.1121883, -0.9754747, 1.5763533, -1.2258955...	0	-1	0.058447	-0.058447
8	softwar	[1.1837375, -1.7276136, -2.038646, -0.05674517...	0	-1	0.045814	-0.045814
9	creat	[-2.624349, -1.245148, -0.21807899, 0.6985172,...	0	-1	0.068409	-0.068409

Figure 7: Closeness Scores

Model Building and Evaluation:

Instead of immediately fitting our models into a single train and test split, we repeatedly trained and validated the models' performance k times within the original training set via 5-fold cross-validation, in which 4 folds are training data while the remaining fold becomes the validation data.

For our project, we choose 2 classical supervised learning algorithms: Random Forest and XGBoost to evaluate the performance of our compressed data due to their track records in high sentiment classifying accuracy. Random Forest Trees (RFT) is a supervised learning method for classification and regression that consists of a collection of decision trees (DT), in which each tree is trained on a different data sample from the training data with replacement (i.e. bootstrap sampling) [4]. The class that the majority of the trees chose is the output of the random forest for classification problems. The mean or average prediction of each individual tree decides the final output for the regression task. On the other hand, eXtreme Gradient Boosting (XGBoost) is a scalable, effective, adaptable, portable and improved version of the gradient boosting algorithm [5]. It is designed for efficacy, computational speed and model performance while being an open-source library and a part of the Distributed Machine Learning Community [5]. The software and hardware features of XGBoost are the ideal combination for enhancing current boosting methods accurately and quickly.

In the hyperparameter tuning process, for each model Random Forest Trees and XGBoost, we use grid search to optimize and find the most suitable parameters. For each model, the

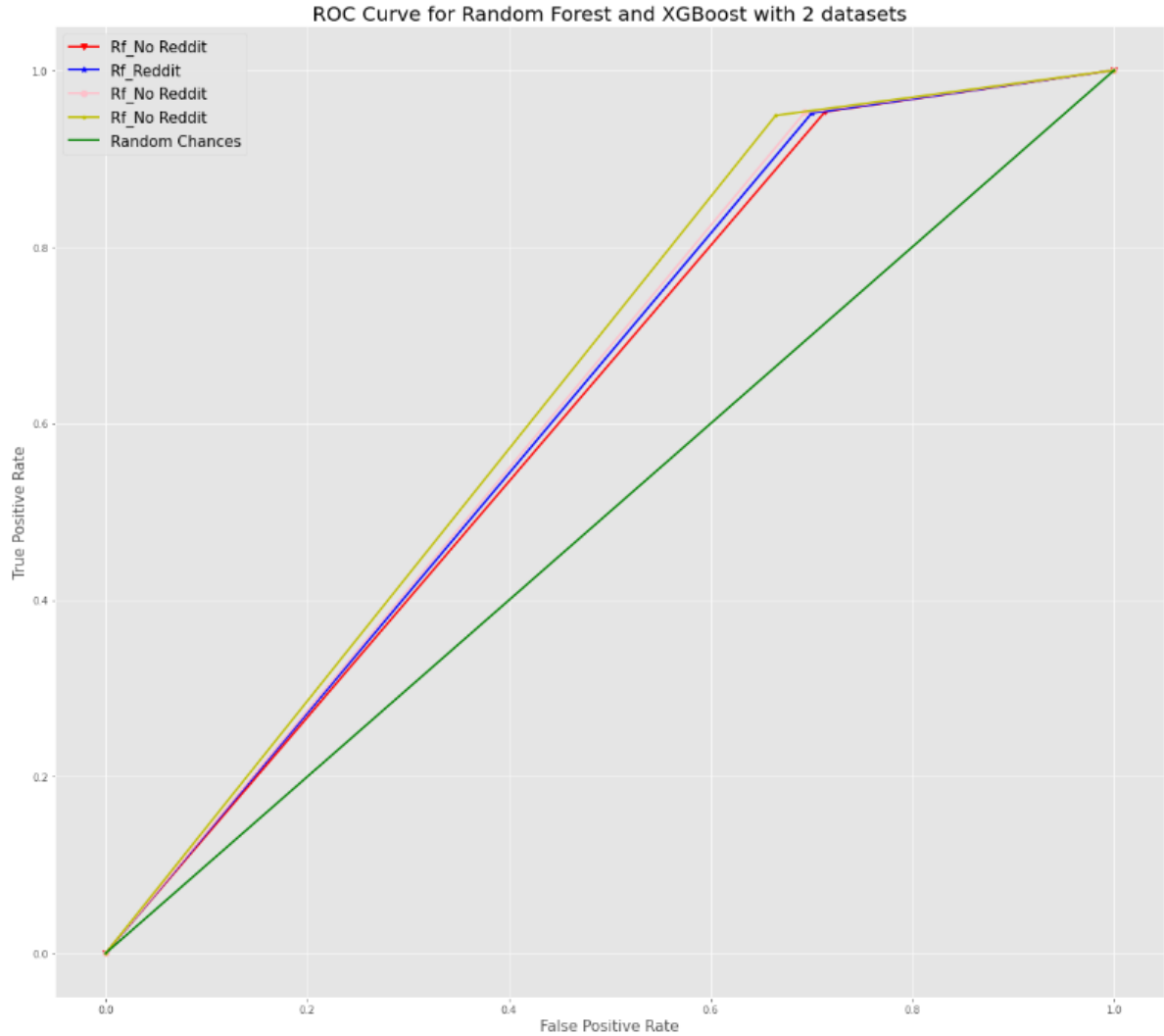
combination of parameter values that achieves the highest average accuracy during the 5-fold cross-validation will be the final model to evaluate on the test set. The hyperparameter tuning options are defined differently for each model. In Random Forest Trees, we optimally search for the maximum depth of the tree, the number of features to consider when looking for the best split, the optimal number of trees in the forest, and metric criteria. Meanwhile, in XGBoost, we search for kernel types, kernel coefficients, independent terms in the kernel function, class, and weighing options.

Machine Learning Models	Hyperparameters	Values
Random Forest	'n_estimators' 'max_depth' 'criterion' 'max_features'	[10, 30, 50, 80, 100] [3, 5, 7, 9], ['gini', 'entropy'] ['sqrt', 'log2', None]
XGBoost	'max_depth' 'n_estimators' 'learning_rate'	(3, 5, 7, 9) (10, 30, 50, 80, 100), [0.1, 0.01, 0.05]

After determining each model's best hyperparameters, we evaluate the performance of these models with following metrics: accuracy (the percentage of correctly classified sentiment score), recall (fraction of positive sentiments that were identified), precision (the percentage of true positives that were identified), and F1 (the harmonic mean between precision and recall).

Models - Dataset	Accuracy	Recall	Precision	F1 Score
Random Forest- Reddit	0.79	0.3	0.67	0.41
Random Forest - No Reddit	0.79	0.29	0.67	0.4
XGBoost - Reddit	0.8	0.34	0.68	0.45
XGBoost - No Reddit	0.79	0.31	0.68	0.42

From the results above, XGBoost performs slightly better than Random Forest for all 4 metrics so it is the recommended model to use for the scope of our projects. There is no clear distinction between the scores for models trained Reddit data and one without Reddit sentiment data. The low recall scores here indicates weak performance in classifying true positives in a correct manner out of all the actual positive values.



Plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold levels yields the ROC curve [6]. The ROC can be conceptualized as a plot of the power as a function of the decision rule's Type I Error (when the performance is calculated from just a sample of the population, it can be thought of as estimators of these quantities) [6]. With the aid of ROC analysis, it is possible to choose potentially optimal models and exclude less-than-ideal ones without first establishing the cost context or the class distribution [6]. Based on the plot above, there is no significant discrepancies between these models and all of them outperform our threshold. Aligned with our above metrics table, the yellow line representing XGBoost model on data with no Reddit sentiment score performs the best with largest area under the curve.

Conclusion

Based on our analysis, we cannot claim that there is a difference in metrics between models that included the Reddit dataset and models that did not. Overall, XGBoost performed better than Random Forest in classifying customer sentiments. No determined conclusions should be made

towards the validity and quality of reviews from Amazon until further model and dataset improvements.

Moving forward, there are several areas we can improve upon to help us make a more confident claim about the validity of Amazon reviews. For instance, the data we scraped from Reddit uses brand as a search condition, which limits us to posts about brands. The Amazon Reviews dataset provides information on each individual review and contains product ID information. If there was a more niche platform where buyers post about products using identifiable characteristics such as product names or product ID's, we can make a more relevant connection between our base Amazon Reviews dataset and the external sentiment information by matching products. There is a likelihood that this will improve the credibility of our analysis because we would not have to aggregate sentiment scores by brand on the external data.

In addition, we could also add additional datasets, either for further sentiment analysis or to add other numeric fields to our analysis. Getting data from additional sources can lower any potential bias. Recall that we subsetting the electronics category from Amazon Reviews. We can also add extra information by utilizing other categories in our analysis, though this would require greater resources.

Lastly, more classifiers might be investigated, either as modifications of the ones used here. For example, hyperparameter tuning for Random Forest with the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node and the threshold for impurity values can be a possible approach. Another potential improvement for our project can be applying different types of classifiers like Bagging Decision Tree, Extra Trees or Linear/Non-linear Support Vector Machine.

References:

[1] <https://www.mongodb.com/why-use-mongodb>

[2] Khanna, Chetna. "Text Pre-Processing: Stop Words Removal Using Different Libraries." *Medium*, Towards Data Science, 10 Feb. 2021, <https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a>.

[3] <https://en.wikipedia.org/wiki/Word2vec>

[4] Ho, Tin Kam (1995). Random Decision Forests, Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282. Archived from the original on 17 April 2016. Retrieved 5 June 2016.

[5] <https://en.wikipedia.org/wiki/XGBoost>

[6] https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Fang, Xing, and Justin Zhan. "Sentiment Analysis Using Product Review Data." *Journal of Big Data*, vol. 2, no. 1, 2015, <https://doi.org/10.1186/s40537-015-0015-2>.

McAuley, Julian. "Amazon Product Data." Amazon Review Data, <https://jmcauley.ucsd.edu/data/amazon/>.

Documentation for the Pushshift API can be accessed here: <https://pypi.org/project/pushshift.py/>