Data Mining and Predictive Modeling of Amazon Customer Reviews

COSC 757.101 – Data Mining: Final Report

W. Grant Hatcher, Kevin McNamara
Department of Computer and Information Sciences, Department of Marketing
Towson University, Maryland, USA 21252, USA
Emails: whatch2@students.towson.edu, kmcnam3@students.towson.edu

***Abstract*—In this project, we explore available Amazon customer review data to analyze patterns across various attributes and develop predictive tools. Considering a dataset of over 130 million reviews acquired from Amazon's AWS service, we have the potential to consider the impacts of positive and negative reviews on product ratings and customer sentiment. Moreover, applying shallow learning techniques, such as K-Nearest Neighbor and Decision Tree algorithms, we develop high-accuracy predictive capabilities to be leveraged for consumer targeting, increased customer satisfaction, and potentially the generation of additional revenue. Taking the number of helpful votes as a measure of the efficacy of a review on a user's decision to purchase and their satisfaction with that decision, we can predict the helpfulness of a review with an accuracy of over 80% using only categorical data. In addition, through the implementation of sentiment analysis mechanisms, we increase this accuracy significantly with the addition of textual data.**

***Index Terms*—Data mining, Machine learning, Exploratory Data Analysis**

I. Introduction

Since Amazon's launch in 1994 [6], it has become the 8th highest company in the United States in terms of revenue earnings [4]. With an outsize market share in the retail industry of $355.9 billion [5], Amazon is the leader in overall retail in the US, larger than the eight largest brick-and-mortar stores combined. The tech giant naturally then possesses a wealth of data regarding customer purchases, preferences, reviews, and more. Moreover, as part of their platform, their customers have left hundreds of millions, if not billions, of reviews on items purchased. Through Amazon Web Services (AWS), Amazon has made available over 130 million of these reviews in TSV file format, organized by product category for use in testing and analysis in their cloud services. Using data mining techniques, we can harness this data to research patterns in customer sentiment and buying habits.

In our initial analysis, we note that much of the customer review data demonstrates uniformity across the different product segments. As our primary target of interest, we are attempting to apply machine learning algorithms to predict Star Rating and Helpful Votes for individual product purchase reviews. Indeed, the distribution of star ratings is very uniform across categories, the vast majority of which are the maximum of 5 stars. This presents potential challenges in finding a highly-accurate algorithm to predict the star rating, especially for the under-represented classes. In addition, the helpful vote fields have some regularity as well, with a significant minority having an outsize portion of votes.

II. Approach

A. Data Description

The dataset we are evaluating is the Amazon Customer Reviews Dataset [1] available from Amazon Web Services (AWS) cloud. The dataset consists of more than 130 million reviews of Amazon products, separated into 43 subcategories (apparel, books, grocery, jewelry, luggage, etc.), each with over a million reviews. The dataset includes the attributes *customer_id*, *helpful_votes*, *marketplace*, *product_category*, *product_id*, *product_parent*, *product_title*, *review_body*, *review_date*, *review_headline*, *review_id*, *star_rating*, *total_votes*, *verified_purchase*, and *vine*. More specifically, the field *marketplace* is always "US" for United States, and the field *product_category* is uniform within each subcategory file (apparel, books, etc.). In addition, *helpful_votes*, *total_votes*, and *product_parent* are integer values, and *review_date* is a date of format YYYY-MM-DD. The attributes *product_title*, *review_body*, and *review_headline* are all variable-length strings. Finally, *product_id* and *review_id* are alphanumeric strings of capital letters and numbers of varying length.

B. Accessing the Dataset

As the first part of the process, we had to access the data from AWS. This proved challenging, because data is segmented into 46 separate files sorted by product category and were each quite large in size. Together they reached approximately 80 GB in storage size. Moreover, with over 130 million reviews, we needed a powerful machine with enough memory and processing power to be able to extract, load in, sort and manipulate such big data into memory. This was accomplished by utilizing a Dell PowerEdge R910 server with Intel Xeon E7530 processor and 64GB memory running and Ubuntu Linux 16.04.5 LTS virtual machine on top of VMWare ESXi 6.0. This was the best machine at our disposal, however, as we will see, this still presented some limitations. Each data file was thus downloaded via URL link provided

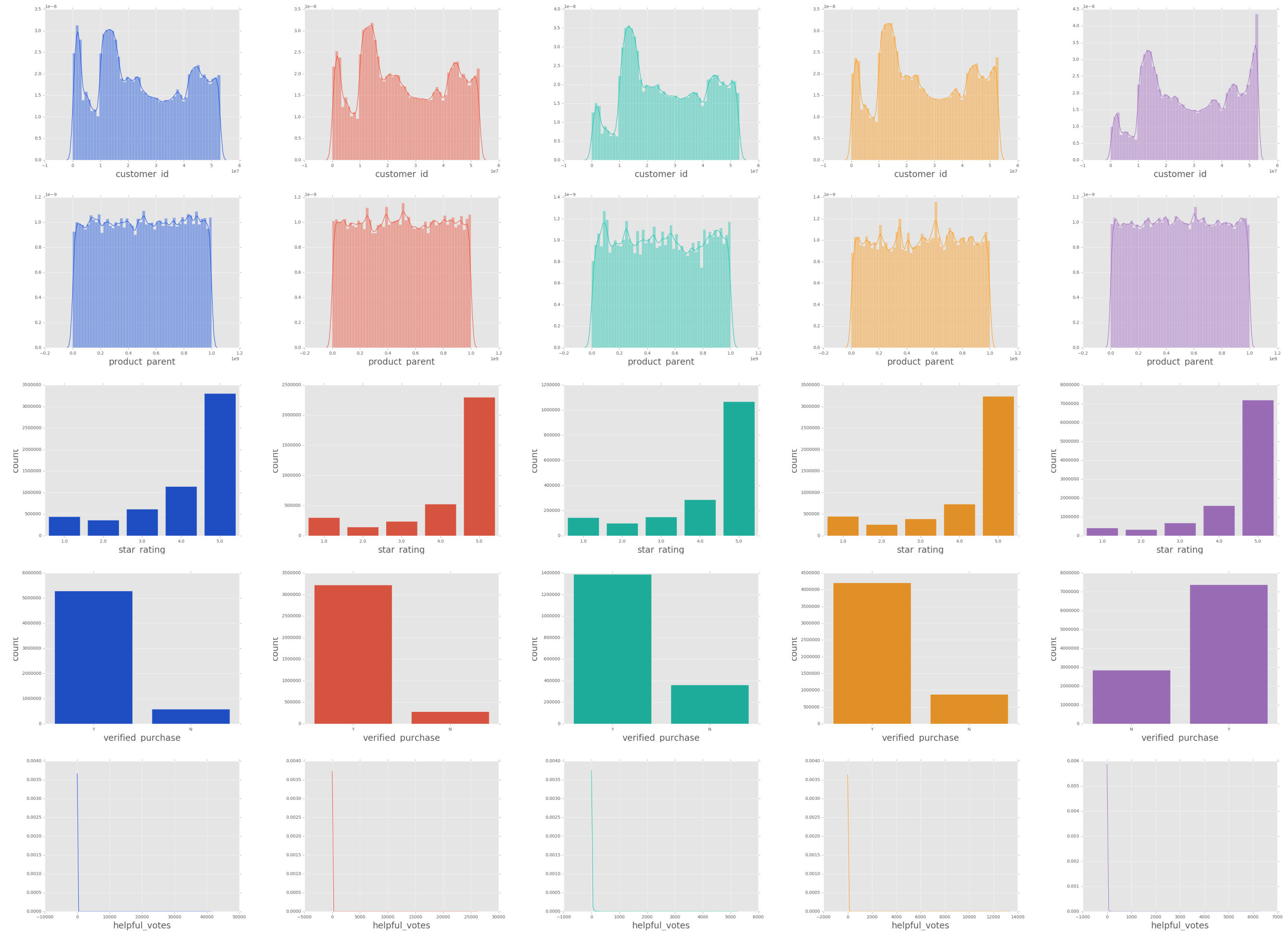


Fig. 1. Histograms and Density Estimation plots of product categories Apparel, Automotive, Baby, Beauty, and Books (left to right). Attributes are *customer_id*, *product_parent*, *star_rating*, *verified_purchase*, and *helpful_votes* (top to bottom).

in the "index.txt document", and was provided in .tsv or tab-separated file format, presumably because text data may itself have commas present, potentially breaking import processes into programs. The ability to read in these data files presented a challenge, as most traditional text editors, word processors, and spreadsheet software, especially on Windows PCs, could not open even one file or would run impossibly slowly when attempting any edits.

C. Data Preprocessing

Preparing the data for analysis, we again ran into difficulties. In such a large dataset the amount of missing data and anomalies is quite substantial. What's more, with individual files on the order of several gigabytes, it is impossible to individually search through the rows of data instances. Thus, normalizing this data and preparing it for input into learning models was certainly a challenge. In this phase, it became evident that analyzing the entire dataset (broken up into 43 product category sets in 46 files) would be nearly impossible without careful consideration of programming operations. Simple processes would consume significant amounts of time, such as loading any individual file into Python [2] as a DataFrame (a Python matrix implementation), which would take tens of minutes, and thus any later operations that fail would incur the same cost over again. Additionally, some operations we simply impossible, on one subset (individual files) or the entire dataset, such as dropping large columns from a DataFrame. As an example, trying to loop even once through several million data items is not feasible.

Despite these limitations to assessing the dataset, the Python DataFrame offered many efficient options that could be performed on the entire matrix at once. This includes removing NaN (Not a Number) and missing characters, rows with too many columns, and values that do not match the column, especially for categorical variables. Additionally, the Python .loc or locate function allows for efficient location of data subsets with little overhead, and in this experiment we are using Python 2.7 [2]. In addition, to be able to apply machine learning algorithms to our datasets, we converted non-numeric categorical strings to binned numeric values. For instance, the attributes *verified_purchase* and *vine* were converted to binary values of 0 and 1 from strings “N” and “Y” representing no

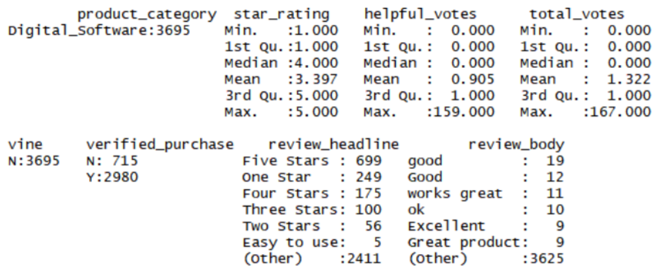
TABLE I. Data descriptions averaged over all seven product subcategories.

AVERAGE	**customer_id**	**product_parent**	**star_rating**	**helpful_votes**	**total_votes**
Count	28371981	28371981	28371981	28371981	28371981
Average	4053140	4053140	4053140	4053140	4053140
Mean	27550919	507045022	4	2	2
Std. Dev.	15383160	288812824	1	19	20
Min	10043	31422	1	0	0
25% Quartile	14223243	256026516	4	0	0
50% Quartile (Median)	26304429	518488891	5	0	0
75% Quartile	41963264	760629282	5	1	1
Max	53096535	999983463	5	14679	14952

and yes. Despite the advantages of Python, we still frequently ran into issues in trying to drop columns, as well as train and test learning models, both of which exceeded memory and ended the program.

D. Exploratory Data Analysis

Moving forward into our EDA phase, we limited our data evaluation to a subset of the entire dataset. Thus, we have analyzed and cross-referenced 7 different product category segments. These segments are: Apparel, Automotive, Baby, Beauty, Books (part 1 of 3), Camera, and Digital Software. Grant has successfully loaded and preprocessed the first three segments (Apparel, Automotive, Baby, Beauty, and Books) into his server using PowerShell. Kevin has successfully loaded the 6th and 7th segments (Camera and Digital Software, respectively) into R Studio. Initial EDA details for the Camera and Digital Software segments are shown in Figure 2 and Figure 3, respectively.



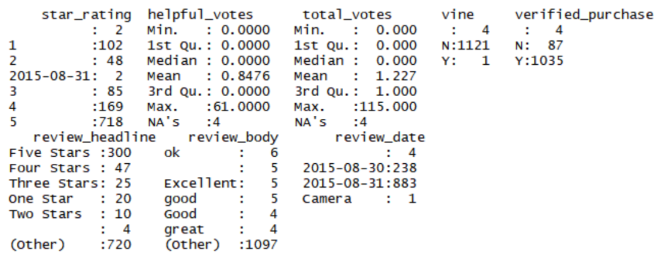
```
       product_category  star_rating     helpful_votes      total_votes
 Digital_Software:3695   Min.   :1.000   Min.   :  0.000   Min.   :  0.000
                         1st Qu.:1.000   1st Qu.:  0.000   1st Qu.:  0.000
                         Median :4.000   Median :  0.000   Median :  0.000
                         Mean   :3.397   Mean   :  0.905   Mean   :  1.322
                         3rd Qu.:5.000   3rd Qu.:  1.000   3rd Qu.:  1.000
                         Max.   :5.000   Max.   :159.000   Max.   :167.000

 vine      verified_purchase    review_headline          review_body
 N:3695    N: 715            Five Stars : 699   good           :  19
           Y:2980            One Star   : 249   Good           :  12
                             Four Stars : 175   works great    :  11
                             Three Stars: 100   ok             :  10
                             Two Stars  :  56   Excellent      :   9
                             Easy to use:   5   Great product:   9
                             (Other)    :2411   (Other)        :3625
```

Fig. 2. Data description of Camera category. Includes mean, median, mode, and quantiles of numeric data columns (*star_rating*, *helpful_votes*, and *total_votes*), as well as counts for binary attributes (*vine* and *verified_purchase*).

Looking at the histograms in Figure 1, we can see near identical distributions of customer id numbers connected to reviews. In addition, we can see that the product parents are also quite uniformly distributed. Moreover, looking at the start ratings of reviews, we see a significant number of five star reviews, much higher than any other rating, and generally exceeding the volume of all other star ratings combined. Traditionally, we see that nearly all reviews are verified purchases, and that the number of helpful votes are generally quite low, with only very few products getting many helpful review votes.

Considering all seven product categories, we can see in Table I the total number of samples considered to be over 28



```
     star_rating  helpful_votes      total_votes      vine     verified_purchase
           :  2   Min.   : 0.0000   Min.   :  0.000    :   4    :   4
 1         :102   1st Qu.: 0.0000   1st Qu.:  0.000   N:1121   N:  87
 2         : 48   Median : 0.0000   Median :  0.000   Y:   1   Y:1035
 2015-08-31:  2   Mean   : 0.8476   Mean   :  1.227
 3         : 85   3rd Qu.: 0.0000   3rd Qu.:  1.000
 4         :169   Max.   :61.0000   Max.   :115.000
 5         :718   NA's   :4         NA's   :4
   review_headline    review_body          review_date
 Five Stars :300    ok         :   6              :  4
 Four Stars : 47               :   5    2015-08-30:238
 Three Stars: 25    Excellent:   5    2015-08-31:883
 One Star   : 20    good       :   5    Camera    :  1
 Two Stars  : 10    Good       :   4
            :  4    great      :   4
 (Other)    :720    (Other)    :1097
```

Fig. 3. Data description of Digital Software category. Includes mean, median, mode, and quantiles of numeric data columns (*star_rating*, *helpful_votes*, and *total_votes*), as well as counts for binary attributes (*vine* and *verified_purchase*).

million, with the average per category being only 4 million. In addition, we see that the vast majority of products have 0 total or helpful votes, with 1 vote being the value of the 75th percentiles. In contrast, we see that the vast majority of reviews have a star rating of 5, with the 25th percentile being a star rating of 4. Note that the attribute *star_rating* represents a categorical integer of 1 through 5, while *helpful_votes* and *total_votes* are unbounded integers.

E. Shallow Learning

Once our EDA phase concluded we moved onto applying some shallow learning techniques to fit our data. Using the *helpful_votes*, *star_rating*, *total_votes*, *verified_purchase*, and *vine* attributes each as target class values, we trained and tested K-Nearest Neighbor, Decision Tree (with Gini index and Entropy), Random Forest, and Naive Bayes models on the data, resulting in five tests for each algorithm. Here, our goal was to find the best performing algorithm to apply to our data. We tested all five of these models using all the remaining attributes (including those not mentioned as classes) as training data. For each of the attributes, we averaged out the accuracy metrics for each model and compared the results against the average for all of the attributes. Also note that the string attributes could not be used in this classification step, as none of the algorithms can take a string as a variable.

The learning algorithms were all implemented through use of the sciKi-learn [3] library. These algorithms include arguments for the variation and tuning of hyperparameters, such as equations for activation and decision making. In particular, K-Nearest Neighbor (KNN) classification utilizes a function of

TABLE II. Shallow learning results on product category Apparel and Averaged across all category subsets (Apparel, Automotive, Baby, etc.).

ACCURACY	Apparel					AVERAGE				
	helpful_votes	star_rating	total_votes	verified_purchase	vine	helpful_votes	star_rating	total_votes	verified_purchase	vine
KNN-3	0.660	0.416	0.588	0.868	1.000	0.587	0.477	0.504	0.836	0.996
DT-gini	0.850	0.392	0.804	0.829	1.000	**0.808**	0.455	0.745	0.817	0.997
DT-entropy	**0.850**	0.392	**0.805**	0.831	**1.000**	0.807	0.455	**0.745**	0.819	**0.997**
RF-10	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*
NB-gaussian	0.754	**0.562**	0.701	**0.899**	1.000	0.699	**0.616**	0.656	**0.871**	0.997

PRECISION	Apparel					AVERAGE				
	helpful_votes	star_rating	total_votes	verified_purchase	vine	helpful_votes	star_rating	total_votes	verified_purchase	vine
KNN-3	0.600	0.380	0.530	**0.830**	**1.000**	0.533	0.433	0.458	0.803	**0.997**
DT-gini	**0.850**	**0.400**	**0.810**	**0.830**	**1.000**	0.805	**0.463**	**0.748**	**0.820**	**0.997**
DT-entropy	**0.850**	**0.400**	**0.810**	**0.830**	**1.000**	**0.808**	**0.463**	**0.748**	**0.820**	**0.997**
RF-10	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*
NB-gaussian	0.570	0.320	0.490	0.810	**1.000**	0.490	0.380	0.430	0.760	**0.997**

RECALL	Apparel					AVERAGE				
	helpful_votes	star_rating	total_votes	verified_purchase	vine	helpful_votes	star_rating	total_votes	verified_purchase	vine
KNN-3	0.660	0.420	0.590	0.870	**1.000**	0.590	0.478	0.505	0.837	**0.997**
DT-gini	**0.850**	0.390	**0.800**	0.830	**1.000**	**0.810**	0.455	**0.745**	0.820	**0.997**
DT-entropy	**0.850**	0.390	**0.800**	0.830	**1.000**	0.808	0.455	**0.745**	0.820	**0.997**
RF-10	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*
NB-gaussian	0.750	**0.560**	0.700	**0.900**	**1.000**	0.698	**0.615**	0.657	**0.870**	**0.997**

F1-SCORE	Apparel					AVERAGE				
	helpful_votes	star_rating	total_votes	verified_purchase	vine	helpful_votes	star_rating	total_votes	verified_purchase	vine
KNN-3	0.630	**0.400**	0.560	0.840	**1.000**	0.558	0.455	0.478	0.813	**0.997**
DT-gini	**0.850**	**0.400**	0.800	0.830	**1.000**	**0.808**	0.460	0.745	**0.820**	**0.997**
DT-entropy	**0.850**	**0.400**	**0.810**	0.830	**1.000**	**0.808**	0.460	**0.748**	**0.820**	**0.997**
RF-10	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*
NB-gaussian	0.650	**0.400**	0.580	**0.850**	**1.000**	0.578	**0.468**	0.520	0.810	**0.997**

euclidean distance to determine the separation of the nearest neighbors of data items, classifying them into groups. Decision Tree (DT) is a quite effective classification mechanism that is also very efficient. In this examination, we tested both Gini index and Entropy functions as the splitting criterion to compare. The Random Forest (RF) algorithm creates a set of n randomized decision trees and picks the best. Here we have set n to 10. Finally, Naive Bayes (NB) is a statistical learning mechanism that operates based on probability distribution of the training data.

F. Modeling and Metrics

As the long-term goal of our project, we are interested in prediction the helpfulness of a review based on natural language analysis. This requires natural language model building and sentiment analysis of the *review_headline*, *review_body*, and *product_title* attributes before integration with our more basic predictive models for *helpful_votes* and *star_rating*. To accomplish this, we vectorized the string values from the noted attributes into one-hot or binary (present/absent) encoding, taking the highest-frequency words to reduce the overall size of the data matrix. It was necessary to reduce the number of words from which to apply learning algorithms, as the diversity of several million reviews produces near unfathomable individual words, making the width of the input matrix nearly equivalent to the length. Compared to our original dataset of only fifteen attributes, this is a significant departure and surely exceeds memory in application. In our particular experiments, we also note that we have only applied sentiment analysis through string segmentation, vectorization, and one-hot encoding for a single text attribute, namely *review_headline*. This is to reduce the overall dictionary size, as well as to reduce the calculations and demonstrate a proof of concept.

As metrics for our experiment, we use the statistical measures of Accuracy, Precision, Recall, and F1-Score. Because we are using supervised learning to train our models, the ground truth values for each of our class values are known, and when we give our trained model unseen testing data, we can compare the model result with the ground truth. In this way, we derive four types of results, those are: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). More specifically, a prediction will be labeled TP if the correct positive class was assigned correctly as positive. Likewise a prediction is labeled as FP if the predicted class was the positive class, but the ground truth was the negative class. So if the positive class is 1, then predicting 1 when the answer was 0 is an FP. It should be noted that this kind of statistical measurement is particularly well-suited to computing systems, as the number of variables become too large to calculate by hand for any person, especially in multivariate systems.

In fact, in multivariate systems, we look at each category in a class alone as the positive class. So for values 0, 1, and 2, we extract TP, FP, TN, FN, Precision, Recall and F1-Score values three times, setting 0, 1, and 2 each as the positive class and the other two as the negative. So, if 2 is positive, then 1 and 0 are negative, and FP includes both 1 and 0 ground truth values predicted as 2. Thus, we derive 3 Precision, Recall, and F1-Score values and take the average of each. Also note that the overall Accuracy remains the same.

TABLE III. Shallow learning results on product category Baby and class value *helpful_votes* without (left) and with (right) Sentiment analysis using the top 500 words.

ACCURACY	Baby					Baby + Sentiment (500 Words)				
	helpful_votes	star_rating	total_votes	verified_purchase	vine	helpful_votes	star_rating	total_votes	verified_purchase	vine
KNN-3	0.5856	0.4701	0.5044	0.7485	0.9914	0.5856	0.4701	0.5044	0.7485	0.9914
DT-gini	**0.8053**	0.4602	**0.7360**	0.7680	0.9928	0.9997	**1.0000**	**0.9997**	**1.0000**	**1.0000**
DT-entropy	0.8043	0.4609	0.7350	0.7698	**0.9931**	**0.9997**	**1.0000**	0.9997	**1.0000**	**1.0000**
RF-10	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*
NB-gaussian	0.6999	**0.6114**	0.6359	**0.7947**	0.9931	infeas.	infeas.	infeas.	infeas.	infeas.

PRECISION	Baby					Baby + Sentiment (500 Words)				
	helpful_votes	star_rating	total_votes	verified_purchase	vine	helpful_votes	star_rating	total_votes	verified_purchase	vine
KNN-3	0.5300	0.4300	0.4600	0.7200	**0.9900**	0.5300	0.4300	0.4600	0.7200	0.9900
DT-gini	**0.8000**	**0.4700**	**0.7400**	**0.7700**	**0.9900**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
DT-entropy	**0.8000**	**0.4700**	**0.7400**	**0.7700**	**0.9900**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
RF-10	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*
NB-gaussian	0.4900	0.3700	0.4000	0.6300	**0.9900**	infeas.	infeas.	infeas.	infeas.	infeas.

RECALL	Baby					Baby + Sentiment (500 Words)				
	helpful_votes	star_rating	total_votes	verified_purchase	vine	helpful_votes	star_rating	total_votes	verified_purchase	vine
KNN-3	0.5900	0.4700	0.5000	0.7500	**0.9900**	0.5900	0.4700	0.5000	0.7500	0.9900
DT-gini	**0.8100**	0.4600	**0.7400**	0.7700	**0.9900**	**1.000**	**1.000**	**1.0000**	**1.0000**	**1.0000**
DT-entropy	0.8000	0.4600	**0.7400**	0.7700	**0.9900**	**1.000**	**1.000**	**1.0000**	**1.0000**	**1.0000**
RF-10	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*
NB-gaussian	0.7000	**0.6100**	0.6400	**0.7900**	**0.9900**	infeas.	infeas.	infeas.	infeas.	infeas.

F1-SCORE	Baby					Baby + Sentiment (500 Words)				
	helpful_votes	star_rating	total_votes	verified_purchase	vine	helpful_votes	star_rating	total_votes	verified_purchase	vine
KNN-3	0.5600	0.4500	0.4800	0.7300	**0.9900**	0.5600	0.4500	0.4800	0.7300	0.9900
DT-gini	**0.8000**	**0.4600**	**0.7400**	**0.7700**	**0.9900**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
DT-entropy	**0.8000**	**0.4600**	**0.7400**	**0.7700**	**0.9900**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
RF-10	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*	*infeas.*
NB-gaussian	0.5800	**0.4600**	0.4900	0.7000	**0.9900**	infeas.	infeas.	infeas.	infeas.	infeas.

Giving the definitions of TP, FP, TN and FN, we now present Accuracy, Precision, Recall, and F1-Score as:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}. \tag{1}$$

$$\text{Precision} = \frac{TP}{FP + TP}. \tag{2}$$

$$\text{Recall} = \frac{TP}{FN + TP}. \tag{3}$$

$$\text{F1-Score} = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}}, \tag{4}$$

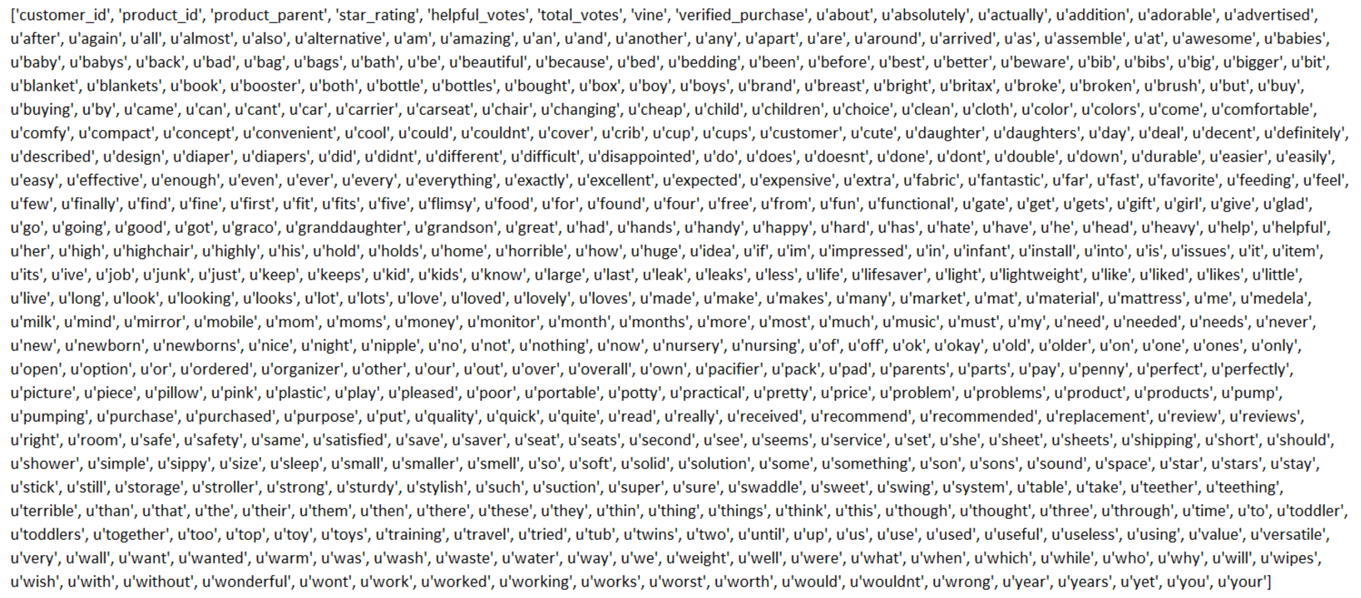
Notice that F1-Score is defined as the harmonic average of precision and recall.

III. Results

Table II shows the results of the shallow learning for all five class values targeted, and for all algorithms attempted These are labeled as K-Nearest Neighbors (KNN-3) with the number of neighbors set at 3, Decision Tree with Gini index (DT-gini) and with Entropy (DT-entropy) as splitting mechanisms, Random Forest (RF-10) with 10 nodes, and Naive Bayes (NB-gaussian) using the Gaussian mechanism.

Notice that the Random Forest algorithm was immediately infeasible on this dataset. In Table II we have provided the results from the Apparel subset (left) as well as the Average of all data subsets combined (right) for comparison. At first glance, one can see that the *star_rating* attribute is quite difficult to predict when compared to the others. In the Apparel group, the highest accuracy rating across models was just 0.562. Compared to the *helpful_votes* attribute, with a high accuracy rating of 0.850, this number does indeed look quite low. Delving further into these results we can see a relative normalcy in accuracy, precision, recall, and f1-score results across product subsets (i.e. Apparel). Additionally, we can surmise that the Decision Trees with Entropy model resulted in the highest accuracy in most metrics for the *helpful_votes*, *total_votes*, *verified_purchase*, and *vine* attributes (all but one attribute). That lonesome attribute, *star_rating*, was more likely to be predicted correctly by the Gaussian Naive Bayes model than Decision Trees with Entropy.

What's more, looking at the *helpful_votes*, and *total_votes* classes, we see that the non-Decision Tree algorithms performed significantly worse (often by 10% or more). Considering the very high accuracies of *verified_purchase* and *vine*, we note several factors. First, considering the *vine* attribute, this is described in the documentation as a promotion in which the now-defunct application Vine (a short-video capturing social media application for mobile devices) was used to add video attribution of the review. We can see from the data that this promotion was almost never taken advantage of, and such a high number of instances have a "N" or "not present" value that simple prediction of "N" is correct 99.99% of the time. Second, considering the *verified_purchase* attribute, referring back to Figure 1, we can see that this is a binary category as well, with values "Y" and "N". Looking at the figure, we note that, in general, there are significantly more "Y" values than "N" values, and that the approximate split is 85% "Y" values. Looking back at our evaluation results in Table II, we see strong accuracy scores of ranging from 82% to 90%. We can thus surmise that our data does not have enough attributes to



```
['customer_id', 'product_id', 'product_parent', 'star_rating', 'helpful_votes', 'total_votes', 'vine', 'verified_purchase', u'about', u'absolutely', u'actually', u'addition', u'adorable', u'advertised',
u'after', u'again', u'all', u'almost', u'also', u'alternative', u'am', u'amazing', u'an', u'and', u'another', u'any', u'apart', u'are', u'around', u'arrived', u'as', u'assemble', u'at', u'awesome', u'babies',
u'baby', u'babys', u'back', u'bad', u'bag', u'bags', u'bath', u'be', u'beautiful', u'because', u'bed', u'bedding', u'been', u'before', u'best', u'better', u'beware', u'bib', u'bibs', u'big', u'bigger', u'bit',
u'blanket', u'blankets', u'book', u'booster', u'both', u'bottle', u'bottles', u'bought', u'box', u'boy', u'boys', u'brand', u'breast', u'bright', u'britax', u'broke', u'broken', u'brush', u'but', u'buy',
u'buying', u'by', u'came', u'can', u'cant', u'car', u'carrier', u'carseat', u'chair', u'changing', u'cheap', u'child', u'children', u'choice', u'clean', u'cloth', u'color', u'colors', u'come', u'comfortable',
u'comfy', u'compact', u'concept', u'convenient', u'cool', u'could', u'couldnt', u'cover', u'crib', u'cup', u'cups', u'customer', u'cute', u'daughter', u'daughters', u'day', u'deal', u'decent', u'definitely',
u'described', u'design', u'diaper', u'diapers', u'did', u'didnt', u'different', u'difficult', u'disappointed', u'do', u'does', u'doesnt', u'done', u'dont', u'double', u'down', u'durable', u'easier', u'easily',
u'easy', u'effective', u'enough', u'even', u'ever', u'every', u'everything', u'exactly', u'excellent', u'expected', u'expensive', u'extra', u'fabric', u'fantastic', u'far', u'fast', u'favorite', u'feeding', u'feel',
u'few', u'finally', u'find', u'fine', u'first', u'fit', u'fits', u'five', u'flimsy', u'food', u'for', u'found', u'four', u'free', u'from', u'fun', u'functional', u'gate', u'get', u'gets', u'gift', u'girl', u'give', u'glad',
u'go', u'going', u'good', u'got', u'graco', u'granddaughter', u'grandson', u'great', u'had', u'hands', u'handy', u'happy', u'hard', u'has', u'hate', u'have', u'he', u'head', u'heavy', u'help', u'helpful',
u'her', u'high', u'highchair', u'highly', u'his', u'hold', u'holds', u'home', u'horrible', u'how', u'huge', u'idea', u'if', u'im', u'impressed', u'in', u'infant', u'install', u'into', u'is', u'issues', u'it', u'item',
u'its', u'ive', u'job', u'junk', u'just', u'keep', u'keeps', u'kid', u'kids', u'know', u'large', u'last', u'leak', u'leaks', u'less', u'life', u'lifesaver', u'light', u'lightweight', u'like', u'liked', u'likes', u'little',
u'live', u'long', u'look', u'looking', u'looks', u'lot', u'lots', u'love', u'loved', u'lovely', u'loves', u'made', u'make', u'makes', u'many', u'market', u'mat', u'material', u'mattress', u'me', u'medela',
u'milk', u'mind', u'mirror', u'mobile', u'mom', u'moms', u'money', u'monitor', u'month', u'months', u'more', u'most', u'much', u'music', u'must', u'my', u'need', u'needed', u'needs', u'never',
u'new', u'newborn', u'newborns', u'nice', u'night', u'nipple', u'no', u'not', u'nothing', u'now', u'nursery', u'nursing', u'of', u'off', u'ok', u'okay', u'old', u'older', u'on', u'one', u'ones', u'only',
u'open', u'option', u'or', u'ordered', u'organizer', u'other', u'our', u'out', u'over', u'overall', u'own', u'pacifier', u'pack', u'pad', u'parents', u'parts', u'pay', u'penny', u'perfect', u'perfectly',
u'picture', u'piece', u'pillow', u'pink', u'plastic', u'play', u'pleased', u'poor', u'portable', u'potty', u'practical', u'pretty', u'price', u'problem', u'problems', u'product', u'products', u'pump',
u'pumping', u'purchase', u'purchased', u'purpose', u'put', u'quality', u'quick', u'quite', u'read', u'really', u'received', u'recommend', u'recommended', u'replacement', u'review', u'reviews',
u'right', u'room', u'safe', u'safety', u'same', u'satisfied', u'save', u'saver', u'seat', u'seats', u'second', u'see', u'seems', u'service', u'set', u'she', u'sheet', u'sheets', u'shipping', u'short', u'should',
u'shower', u'simple', u'sippy', u'size', u'sleep', u'small', u'smaller', u'smell', u'so', u'soft', u'solid', u'solution', u'some', u'something', u'son', u'sons', u'sound', u'space', u'star', u'stars', u'stay',
u'stick', u'still', u'storage', u'stroller', u'strong', u'sturdy', u'stylish', u'such', u'suction', u'super', u'sure', u'swaddle', u'sweet', u'swing', u'system', u'table', u'take', u'teether', u'teething',
u'terrible', u'than', u'that', u'the', u'their', u'them', u'then', u'there', u'these', u'they', u'thin', u'thing', u'things', u'think', u'this', u'though', u'thought', u'three', u'through', u'time', u'to', u'toddler',
u'toddlers', u'together', u'too', u'top', u'toy', u'toys', u'training', u'travel', u'tried', u'tub', u'twins', u'two', u'until', u'up', u'us', u'use', u'used', u'useful', u'useless', u'using', u'value', u'versatile',
u'very', u'wall', u'want', u'wanted', u'warm', u'was', u'wash', u'waste', u'water', u'way', u'we', u'weight', u'well', u'were', u'what', u'when', u'which', u'while', u'who', u'why', u'will', u'wipes',
u'wish', u'with', u'without', u'wonderful', u'wont', u'work', u'worked', u'working', u'works', u'worst', u'worth', u'would', u'wouldnt', u'wrong', u'year', u'years', u'yet', u'you', u'your']
```

Fig. 4. Attributes of the Baby + Sentiment (500) learning matrix. The top 500 words were selected as one-hot (0 or 1) attributes.

better identify this class, as having so many instances provide a more than adequate sample population.

In the second part of our experiment, we have cleaned and segmented the textual data of the *review_headline* attribute, encoding individual words into a library attribute list, and encoding binary values of 1 (present) and 0 (not present) for each word in a particular data instance. The result is a massively larger matrix mostly populated with these binary values. For this experiment, we have attempted to apply all of the prior algorithms used, and have varied the maximum number of words in the library to limit the matrix size. Our first observation is that the Naive Bayes algorithm no longer can support execution on our system, exceeding memory when attempted. To further reduce the computational overhead, we have considered the smallest of our product category subsets, Baby, with only 1.75 million instances. Maintaining a training ration of 70%, this yields a test set of only 5.25 thousand.

In Table III we can see the results of adding sentiment analysis to our existing classification mechanisms. As we can see from the table, the results are a quite astounding. Note that the top 500 words were used in the sentiment section of the attributes, and can be found in Figure 4. The results for the K-Nearest Neighbors algorithm are exactly the same, with or without the additional data. In contrast, the decision tree algorithm approaches 100% accuracy *helpful_votes* and *total_votes*, and actually achieves it in *star_rating*, *verified_purchase*, and *vine*. Given these results there are several potential explanations. The first is that something wrong has occurred, and this is certainly the immediate reaction. However, review of the code and the multiple variations of the classes and algorithms lead us to believe this is not the case. Instead, it may be possible that there are several quite strong indicators within the text of the *review_headline* that highly influence the result. This can clearly be explained for star rating, as many review headlines simply state "Five Stars", and the numbers "five","four", "three", "two", and "one" all appear in the list of attributes. This is more difficult to clearly surmise in the case of verified purchase and vine. Moreover, we note that *helpful_votes* does not reach 100% accuracy, and it is not clear what its relation is to *star_rating* or the other attributes. It seems highly likely that some combinations of the words in the attribute list highly influence the perceived helpfulness of the review.

IV. Literature Review

In this section we review some relevant works to our current research project. Specifically, Diaz and Ng [8] provided an overview of relevant works on making predictions of helpful reviews. They stress the importance of context in understand the reviews. Also, they mention a lack of uniformity among approaches for predicting helpfulness which hindered their ability to compare methods. That being said, the authors specifically mention a few advance models such as probabilistic matrix factorization and HMM-LDA as well as neural networks as exciting prospects for predicting customer reviews.

In addition, Martin [7] in her 2017 unpublished masters thesis explored review text analysis in predicting review ratings. She cites differing user standards as a major hindrance to this method along with anecdotal information and differing vocabulary that users may use. Martin looked at two different Amazon datasets from distinct categories and first used binary classification to predict a "high" or "low" rating. In addition, the author attempted to find a more exact prediction using multi-class classification and logistic regression. Also, she trained and tested Naive Bayes, SVM, and Random Forest classifiers. Martin found SVM and Naive Bayes to be the most successful classifiers but noted that the binary classification also performed quite well for the other product category. Her conclusions were mixed due to differing results across product categories.

Finally, Park [9] analyzed aspects of product reviews across five categories and looked at their relevance to review helpfulness. The author then used four mining methods to find the best predictor for each product type. Park found that product differences mean algorithms need to be different across product categories. The author also concluded that the vector regression method was the most accurate predictor for each of the five categories.

V. Conclusion

In this work, we have considered the application of shallow machine learning algorithms to Amazon product review data. As we have observed previously, the Decision Tree algorithm is particularly powerful, though it generally is not the ideal candidate for all situations. In considering first the categorical data alone, we have achieved well over 80% in accuracy and F1-score in predicting the number of helpful votes for a particular review. With the addition of text-based attributes cleaned and extracted from the review headlines, we can predict the helpful votes of a review with near-exact accuracy, precision, recall, and F1. While these results need to be corroborated, this is an impressive result.

References

[1] Amazon customer reviews dataset. AWS https://registry.opendata.aws/amazon-reviews/, 2018.

[2] Python. https://www.python.org/, 2018.

[3] scikit-learn. https://scikit-learn.org/stable/, 2018.

[4] The top 10. Fortune 500 http://fortune.com/fortune500/, 2018.

[5] J. Desjardins. The extraordinary size of amazon in one chart. Visual Capitalist https://www.visualcapitalist.com/extraordinary-size-amazon-one-chart/, 2016.

[6] A. Hartmans. 15 fascinating facts you probably didn't know about amazon. Business Insider https://www.businessinsider.com/jeff-bezos-amazon-history-facts-2017-4, dec 2018.

[7] M. Martin. Predicting ratings of amazon reviews - techniques for imbalanced datasets. Master's thesis, Université de Liège, Liège, Belgique, 2017.

[8] G. Ocampo Diaz and V. Ng. Modeling and prediction of product review helpfulness: A survey. In *Proceedings of 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–708, 2018.

[9] Y.-J. Park. Predicting the Helpfulness of Online Customer Reviews across Different Product Types. *Sustainability*, 10(6):1–20, May 2018.