Amazon Review Classification

Richard Robinson, Grant Kennedy

San Francisco State University

Table of Contents:

0) Preamble

Amazon uses a combination of review headlines, text, voting, and a rating system for product reviews. The utilization of the rating, a scale from 1-5, in juxtaposition with ordering of more relevant reviews allows users to make better informed decisions on purchases. The scoring is of particular interest as it is quantifying a degree of quality of the product and how it meets buyer expectations.

Many review mediums do not have a ranking system, or are confined within a two point system (either bad or good). The capability of classifying text into positive and negative, or improving the scope of the two point systems can be of great benefit for decision making.

In our work we attempt to classify textual reviews into the categorical 5 stars system as well as the binary positive or negative system. Some examples of where this type of classification may be useful include, but are not limited to Twitter and YouTube-- neither having 5 star classification.

1) Prior Studies

Before embarking on the task of classification from textual data we looked into papers done previously on this topic to refine the initial approach. From these prior works a baseline of expectations from the classification were formed and approaches of what worked well and did not work well alongside potential difficulties were taken into consideration.

Two papers referenced are Marie Martin's Predicting Ratings of Amazon Reviews - Techniques for Imbalanced Datasets (University of Liege) and Peter Brydon and Keven Groarke's Amazon Rating Prediction, UCSD .

Martin's paper is rather thorough and emphasizes that in order for results to have meaning the dataset must be rebalanced. For this means it was recommended to use undersampling. Oversampling leads to replication of data and results to be overly optimistic. Not balancing the dataset will lead to the same outcome as there is an abundance of 4 and 5 star reviews and a deficit of 2 and 3 star ones. Martin worked on the dataset using Naive Bayes, SVM, random forest, and logistic regression. No additional features were looked at besides unigrams in their work and the preprocessing of their data likely stripped some relevant information, such as capitalization,

punctuation, and stop words in headlines. These factors will be discussed in more detail in 2) System Design.

Brydon and Groarke's work aided in identifying potential interesting patterns for the data. Some components identified are how review quantity and average rating change with time. This is of interest since textual data is subjective for reviews -- the same text can have multiple different scores depending on the user. If the rating fluctuates based on time a modern model may be better created by working with datasets within more recent times and going back further may actually decrease accuracy after a cut off. They decided to use rating, helpfulness, unigrams and bigrams and also noted the imbalance in the dataset. They did not rebalance the dataset, which is an issue when working with data skewed as this is.

2) System Design

The dataset was provided by Amazon and already has scoring and categorization done for the data. This means that the data does not need to be inferred and that the ranking is known by default. This is greatly beneficial as it gives a solid and correct baseline for classification.

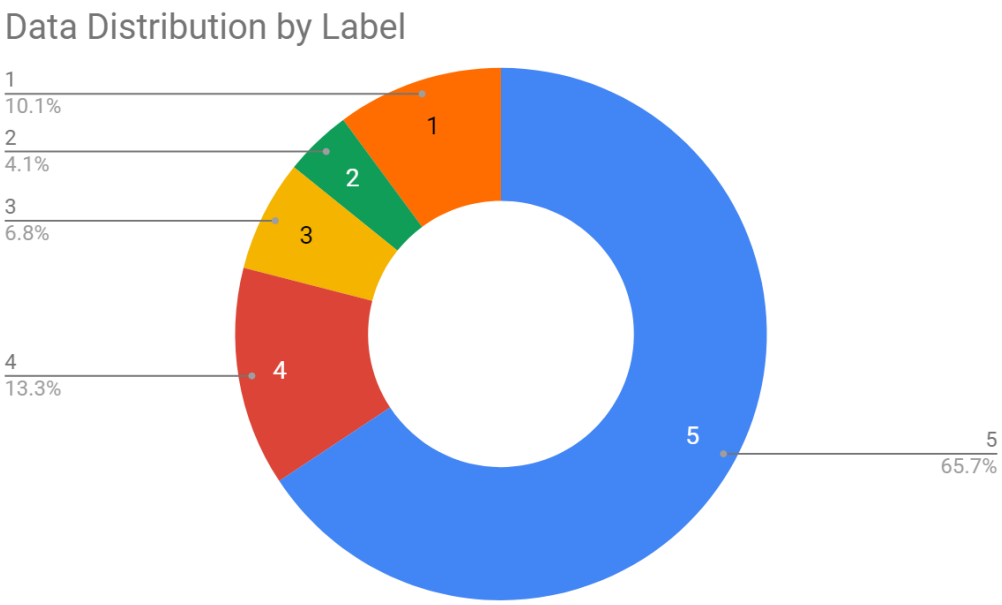
The prediction can be broken down into either binary classification or categorical classification. The binary classification is either positive or negative. For this project 3 star reviews were excluded as they can possibly be interpreted as neutral. 1 and 2 star reviews were classified as negative while 4 and 5 star reviews were considered positive. Categorical classification was done through predicting a score from 1-5; in reference to possible Amazon star rankings.

a. Data Processing

The goal of this project was to attempt to classify textual data into positive or negative as well as a 5 star system. The data was retrieved from amazon and consisted of approximately 2 million reviews after refactoring. The initial data was stored as a tab separated value file and had additional tabs, likely within the review headline or body-- causing inconsistent numbers of columns. This file was converted into a csv while dropping the rows with more columns than expected.

Reviews posted are in general exceedingly positive. This is to be expected; products in general are not designed to fail and if they do fail the

item does not tend to be up for sale very long. An example of data distribution for a set of 10,000 reviews is shown below:



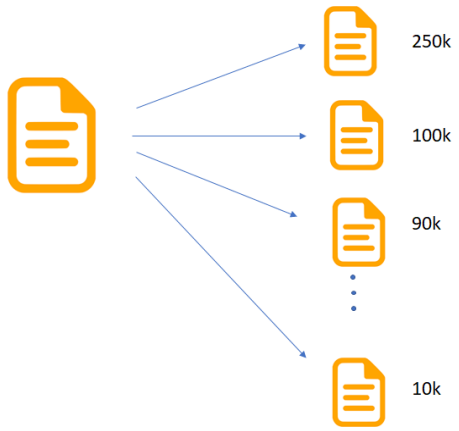
There is an abundance of 5 star reviews and a lack of 2-3 star. This is a trouble that must be taking into consideration. To account for this it was decided to use undersampling, that is to prune data from more represented classes through within the working dataset such that all classes are equally represented. This is a necessity, if the data was taken as is the model could classify all results as 5 stars and be correct approximately 65% of the time. The quality of the program would not be reflected from the results.

In order to achieve these means random undersampling was used through scikit-learn's undersampling function. A static seed was used instead of a dynamic one for replicability of results. Seed variation was analyzed and taken into consideration when reviewing our results.

Initially, a small script was assembled to sort components within the dataset and process them into their corresponding labels as separate files but the imbalanced learn method call was easy to work with (requires one file instead of 5 separate files) and ran quickly.

A file size of about two million values is not easily workable and diminishing returns are found for result quality after a threshold. The classification time for running on the full dataset is not feasible for many of

the classifiers chosen for examination. To accommodate this the file was split into 250,000, 100,000, 90,000, 80,000... 10,000 review segments.



This could then be iterated through in an increasing fashion to develop understand how runtime and quality of results are affected by the dataset size.

b. Classifier Selection

The classifiers considered for the scope of this work were SVC, nu-SVC, decision tree, random forest, naive Bayes, multi-layer perceptron (mlp), and the ensemble voting classifier. Nu-SVC and naive Bayes ultimately had the most success. On smaller datasets mlp outperformed both, but was not scalable as implemented to the larger dataset sizes.

Support vector classifiers break apart features into high dimensional space and constructs divisions in order to infer belonging to categories. The difference between nu-SVC and SVC lies within the regularization parameter; nu-SVC adds a component named nu which sets the upper bound for errors within the training set and lower bound for number of support vectors created. For instance, if nu was .1 with 1000 values the error score would be at most 100 and there would be at least 100 support vectors created.

Decision trees categorize through a set of if else rules and were found to be appropriate with boolean classification, but fell short when it came to star category classification. A random forest is a collection of decision trees.

Naive Bayes is a probabilistic classifier which assumes all components are independant and applies the chain rule and Bayes rule to categorize data.

A multi-layer perceptron consists of layers of nodes with an entry layer, an output layer, and one or more hidden layers. Weights and paths between nodes are adjusted through back propagation. When an error is detected a difference is found between expected and obtained values and adjustments are made recursively at definable adjustment rate.

The voting classifier runs a group of classifiers and then makes a decision based on a cumulation of classifications made by its contained classifiers. Weights can be assigned to give more or less pull to the classifiers within or are evenly distributed. The thought process behind using a voting classifier was that shortcomings from one classifier may be made up by strengths of another and if a consensus was made between classifiers a better result may be formed.

c. Feature Selection

For features n-grams were were a logical starting point. A count vectorizer and TFIDF vectorizer were tried with n-gram ranges of 1-2 for the body text as well as range from 1-5 for headline. It was tried with and without stopword removal. For the headline stopword removal did not benefit, but for the body it dramatically decreased runtime and slightly aided results.

Additional features looked at included emojis, exclamation marks, question marks, ellipses (...), string length, and capitalization ratio. Emojis were tried as :) may indicate contentment with a product while :(or :'(may be good indicators of dissatisfaction. String length was considered because longer explanations may come with more intense feelings towards a product, either positive or negative. Ellipses were seen in some text on dissatisfied reviews and could be an indication of sarcasm. Exclamation and question marks could potentially demonstrate excitement or confusion respectively. Capitalization ratio was chosen due to reviews utilizing it for emphasis.

Combinations of these features were tried on headline and body components alongside their n-grams independently as headline text and

body tend to have different formatting. These were referred to as body union and head union within the codebase.

On top of these features we looked at the Amazon provided helpful votes, number of votes, if a purchaser verified, and if a purchaser has vine status. Vine is an invitational only program by Amazon reserved for users who have a large number of helpful votes.

Feature vectors were appended horizontally and aggregated through the use of a column transformer.

d. Testing

The scoring metrics utilized were precision, recall, and f1. Precision is true positive / (true positive + false positive). Recall is true positive / (true positive + false negative). F1 is the harmonic mean of precision and recall. A harmonic mean is used because an arithmetic mean is not suitable for rates. Confusion matrices were plotted to get a better visualization of the classifications. Shuffled cross validation was used with values of 2- 10. Smaller amounts of iterations were done for basic running then set higher after basic adjustments were set.

Many feature combinations were tried with the classifiers present and refined to what is currently present. This was done through iterative addition of features as well as feature ablation.

When working with large datasets and a number of possible configurations testing can be difficult if run independently. In order to simplify testing a simple automation was made with user defined options.

These options include:
Classifier type: nuSVC, nb, decision tree, random forest, mlp, SVC
Categorization type: boolean, 5 star categorization
File set: set of csv file paths to run on
Plotting confusion matrix
Recording data to csv

With these configurations many user-defined tests can be done without having to manually adjust parameters between runs.

4) Results

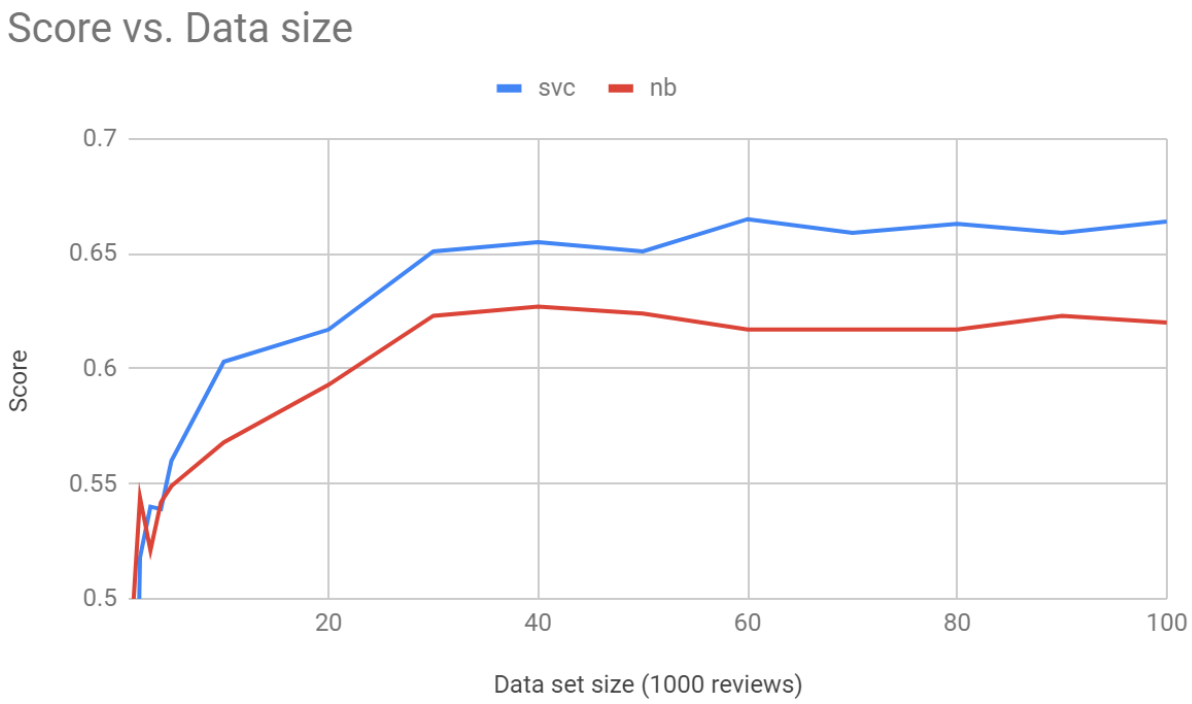
Results with dataset size 10,000 - All classifiers tried

Boolean			
	Precision	Recall	F1
svc	0.841	0.771	0.759
nu-svc	0.869	0.833	0.828
naive bayes	0.897	0.893	0.893
decision tree	0.826	0.809	0.806
random forest	0.744	0.735	0.733
mlp	0.908	0.905	0.905

Category			
	Precision	Recall	F1
svc	0.844	0.495	0.515
nu-svc	0.635	0.606	0.612
naive bayes	0.637	0.617	0.621
decision tree	0.85	0.494	0.514
random forest	0.511	0.46	0.447
mlp	0.642	0.632	0.634

Upon testing the classifiers, it was found that multi-layer perceptron performed the best in regards to both binary and categorical classification with the smallest dataset partition we have, however it is not scalable and was outdone by nu-SVC and naive Bayes on higher dataset sizes.

With higher file sizes the quality of results improved as indicated by this graph:

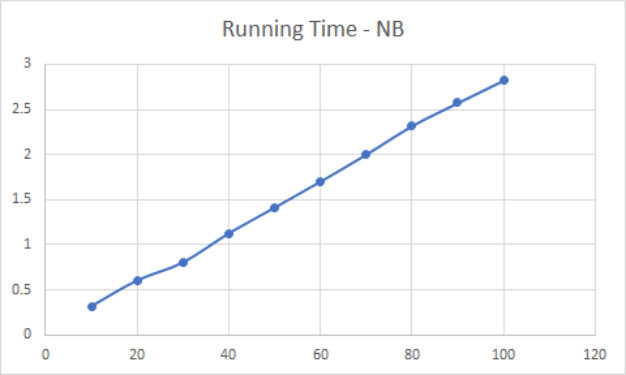


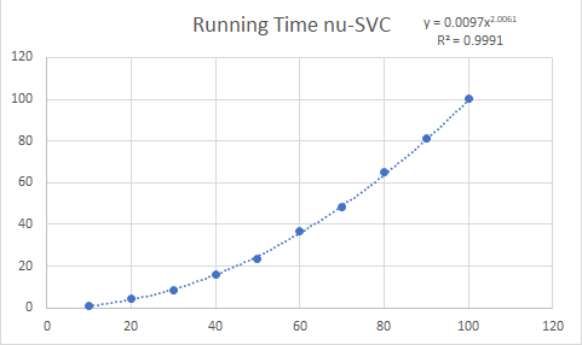
The F1 score tends to increase alongside the dataset size. The rate of increase is not linear and around 60k reviews the results see marginal improvement but continue to trend upwards.

Runtimes:
It was not possible to run mlp with higher dataset sizes with our configuration and schedules.

File Size	nb	svc	dt	rf	mlp
10k	.3	11.8	.4	.4	236.2

Naive Bayes scales linearly, which makes it very nice for program refinement and testing purposes. The other models are not linear.





Feature Discussion / Ablation:
The most useful features were found to be n-grams created through the body and headline text. A combination of these sets further improved results

Body Only

	Precision	Recall	f1
SVC	.442	.438	**.411**
NB	.442	.438	**.436**

Headline Only

	Precision	Recall	f1
SVC	.638	.596	**.601**
NB	.61	.611	**.61**

Combined

	Precision	Recall	f1
SVC	.645	.647	**.644**
NB	.657	.607	**.618**

Reasoning behind this is likely because headlines sometimes contain the rating within them such as a headline of: "Five Stars", “One Star”, etc.

Feature ablation

Model	Features Extracted	Score (f1)	Runtime (sec)	Difference after ablation
M0	All Features	0.658	50.4	
M1	M0 - verified_purchase	0.656	51.5	-0.002
M2	M1 - vine	0.656	48.6	0
M3	M2 - total_votes	0.654	48.9	-0.002
M4	M3 - helpful_votes	0.653	49.5	-0.001
M5	M4 - emojis	0.652	49.8	-0.001
M6	M5 - exclamation	0.652	49	0
M7	M6 - capitalization	0.651	45.5	-0.001
M8	M7 - body bigrams	0.649	24.7	-0.002
M9	M8 - headline bigrams	0.648	23.4	-0.001
M10	m9 - body unigrams	0.586	4.6	-0.062

The features are shown with some positive correlation with f1 score. The most important features were the headline unigrams and this was treated as the base feature in the results above.

Seed Variation:

The random seed has an impact on score, varying dramatically based on the seed value and file size. As file size increases this seed variation drops off. A snippet of the file sizes tested is shown below.

File Size	Undersample Size	F1 change	Relative Change
10,000	2,050	3.4	5.37
20,000	4,350	1.8	2.78
500,000	51,435	.5	.74

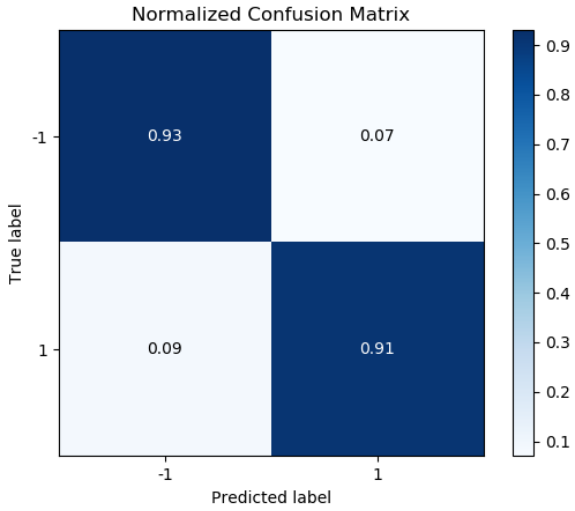
With a file size of 10,000 running naive Bayes on star category classification there is a F1 variation of 3.4% and a relative change of 5.37% relative to F1 average score for the file size. This variation drops off quickly,

but is still present at all levels of file size. Amounting to .5% variation even with a 500,000 review file size.

Best Scores:

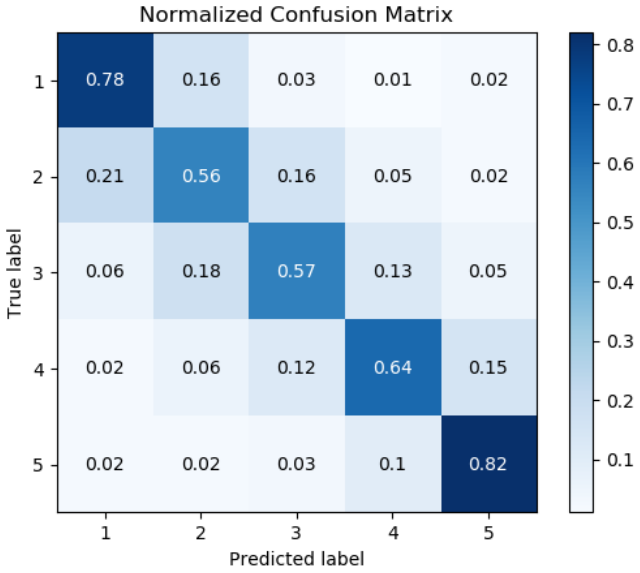
Binary classification:

	Precision	Recall	f1
nu-SVC	0.931	0.93	**0.93**
NB	0.923	0.923	**0.923**
MLP	.908	.905	**.905**



5 Star classification:

	Precision	Recall	f1
nu-SVC	0.672	0.674	**0.672**
NB	0.695	0.666	**0.668**
MLP	.642	.632	**.634**



Our best f1 score was found with nu-SVC for both binary and star categorization. Naive Bayes was just slightly behind, but with significantly reduced run time.

MLP would have likely outperformed SVC and naive Bayes, but it was not runnable on the larger datasets.

One star reviews and five star reviews were the best classified categories and this makes sense: stronger language is likely contained within these review categories, such as “amazing” or “terrible”.

The results surpass those of the paper of Marie Martin earlier cited, which had a maximum f1-score of .52 for star classification and .84 for binary classification.

These results, however, are overly optimistic as the headline or body occasionally contains the review rating inside of it, such as one star or two stars, or five stars, etc.

In order to see how well this classification performs these words would likely have to be treated as stop words.

4) Conclusions / Afterthoughts

This project was a positive one overall, and I am content with the results. Naïve Bayes and nu-SVC were found to be the best classifiers to use for the prediction of Amazon reviews.

This project was interesting from processing the data, to identifying potential classifiers, to refining features, and trying to make sense of the results.

This project worked with a large and terribly imbalanced dataset (after under sampling only around 25% of the initial reviews are used – minority class was around 5 of reviews). Handling this dataset was initially difficult as it could not be opened with most programs due to its size and had to be refactored and broken apart. The initial file also had invalid rows due to being a tsv with extra tabs in review bodies or headlines.

A large dataset causes very long run times, so analysis with run times and plots of how file size affects performance alongside seed variation had to be taken into account. A very large amount of results could be created from

this dataset and our program, but these had to be refined according to what was observed to be significant.

A difficulty was that very small improvements were found through adjustments, there were diminishing payoffs between refinement of the program in regards to time spent versus resulting scores.

The n_jobs parameter was found late into the project and would have likely sped up the process: this allows multiple cores to be used for the program.

The classification of reviews based on textual data is a difficult task as the same review text may receive multiple scores as its subjective based on the writer of the review.

Amazon also supports an update feature, which retains the initial text before the update and our program could be improved by taking this into consideration.

If the giveaway (one star, two star, etc.) inside of headlines and body of the review was found earlier I would have loved to see how the inclusion and exclusion of this affects results.

Thank you for reading and I hope you have a good rest of your day!