Sentiment Analysis of Amazon Product Reviews

Francisco J. Carrillo
franjcf@outlook.com

Abstract

The application of data science to sentiment analysis has become essential in the development of successful online products, be it in the areas of marketing (Google), entertainment (YouTube), retail (Amazon), and communication (Microsoft). Data science has allowed these sectors to monitor and influence consumer behaviour, effectively changing the way that companies interact with their consumers. Direct contact is no longer strictly necessary, it is sufficient to analyze comments, web searches, messages, or product reviews to obtain the consumers' reaction to a new product or a change in services. In this report, we present an analysis of five different classifiers on a data set comprising of 3000 online reviews labeled as either ”positive” and ”negative”. We compare and contrast the classifiers' ability to correctly predict a review label based on a ”bag of words” representation and by taking into account the length of said reviews. Furthermore we study the effects of feature selection (number of words sampled) on classifier performance. We conclude that the Logistic Regression classifier works best when compared to its counterparts, as it requires the least amount of features while obtaining the best performance in 4 out of 6 metrics. Finally, we conclude that review length is not a good predictor of sentiment.

1 Introduction

Thanks to the massive amount of data and computing power available today, it is no longer strictly necessary for large business to interact directly with their customers. Companies can now use natural language processing and statistical analysis to quickly scan through online reviews, social media comments, or online searches in order to identify consumer sentiment towards any given action. One of the simplest and quickest sentiment analyses involves the identification of positive reviews and negative reviews, a step that can be used as a stepping stone for answering much more complex questions (i.e. what is the main reason people like, or don't like, a product/action?)

In this report we focus on comparing and contrasting 5 different classifiers in order to identify which methods excel in this sort of data classification and why. We will also study the effects that feature selection (number of words in our ”bag of words” representation) on our classifiers, as well as study the predictive capability of review length on the sentiment of a specific review .

2 Related Work

2.1 Natural Language Processing

There are several classification methods that we can choose from when attempting to do natural language processing. Today it is common to use Neural Networks [5], Support Vector Machines [10], and Deep Forest[11] algorithms to analyze data. However, many, if not all, of them are dependent on feature selection. One of the most straightforward (yet naive) ways to do feature selection is to separate language strings into their corresponding ”bag of words” representation, where we only look into how many times a word is mentioned, irrespective of the context, the location in the text, or the writing style (this is effectively a trade-off between simplicity and context). These word counts, in conjunction with sentiment labels, can then be used to train the corresponding models. More complex feature selection algorithms include Latent Semantic Indexing [4] and ”Word To Vector” [6] , however we will not use or discuss them here for simplicity.

2.2 Data Processing

The data used in this study consists of 3000 customer reviews labeled into two classes: where label ”1” refers to a positive review and label ”0” refers to a negative review. For the purposes of this study, the data set was separated into a training set of 2400 samples and a testing set of 600 samples. Furthermore, the following sets where extracted from each subset: 1) A matrix containing the number of times any given word was present in each review, with the exception of any word that did not appear a minimum of 5 times in the whole set and stop words (i.e. a bag of words representation), 2) A vector with the aforementioned vocabulary, 3) A vector containing the length of each review, and 4) a vector containing the binary labels of the reviews.

All pre-processing was done with either the provided code or specifically coded for this application (please refer to the attached code). Training data pre-processing resulted in a vocabulary of 541 words (or features). This number was later reduced to 400, 300, 200, 100, 50, and 20 through a feature reduction algorithm (more an that later).

2.3 Classification Models

Our analysis involved the following 5 classification methods. These were specifically chosen to evaluate the performance of five widely different approaches, from geometric-based models to logistic regression. All models were obtained from the SciKitLearn Python libraries[7], and all parameters are standard unless otherwise specified.

- Naive Bayes Classifier
- Decision Tree Classifier
- Logistic Regression with L2 penalty
- K-Nearest-Neighbors (KNN) Algorithm with K=10
- Support Vector Machine (SVM) with Linear Kernel

2.4 Evaluation Criteria

We evaluated and compared the performance of the classification algorithms based on the following metrics: Accuracy, Precision, Recall, F1-score, Specificity, and the Area under the Curve (AUC) of the the Receiver Operating Characteristic (ROC). Accuracy is a useful estimate of the overall performance of our classifiers, but it does not take into account how good each classifier is at identifying true positives (Precision, Recall, and F1-score) or at which rate at which it identifies true negatives (Specificity). The AUC term consolidates the Recall and Specificity values into a single value in order to create a relatively robust performance metric [1] [2]. For all these metrics, the higher the value, the better. For brevity, we refer the reader to the mentioned references for the complete definition of each one of these metrics.

3 Methods

3.1 Spotlight Classifier: Naive Bayes Classifier

In contrast with Frequentist models, Bayesian statistical models are based on the idea that, for a given experiment, each possible outcome is associated with an inherent/implicit probability. In Bayesian thinking, this initial belief (the prior) needs to be updated with new observations/data in order to come up with an updated belief on the probability that a certain outcome will occur. This is summarized and formalized by Bayes Theorem [3]:

$$p(A|B) = p(A) * p(B|A)/P(B)$$

Where $p(A)$ is the prior, $p(A|B)$ is the posterior, and $p(B|A)$ is the likelihood. The ”Naive” label in our choice of classifier stems from the assumption that each outcome is conditionally independent of any other outcome, as shown by the following equation[3, 8]:

$$p(A \cap B|C) = p(A|C) * p(B|C)$$

Where A and B are shown to be in conditionally independent of C. Therefore, we can say that the probability of observing the set of features $x_1 ... x_n$ given a set of parameters $'\theta'$ and labels ’z’ is:

$$p(x_1 ... x_n|z, \theta) = \prod_i^n p(x_i|z, \theta)$$

Of course, naiveness a very strong assumption, particularly when the goal is to model the relationship between a ”bag of words”, the English language, and sentiment. This assumption completely neglects any context, word clustering effects (once a word is mentioned, the probability that it is mentioned again increases), grammar, and language structure (where adjectives are normally followed by a nouns, nouns by verbs). Nevertheless, naive models have been shown to be surprisingly robust if coupled with feature selection and enough data [3].

If we combine the previous equation with Bayes Theorem, we can then see that the resulting posterior probability can be easily obtained without having to obtain a large amount of conditional probabilities[8].

$$p(z|x, \theta) \propto p(z|\theta)p(x|z, \theta)$$

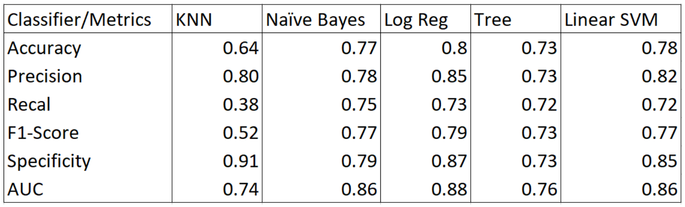
In this case, since the data is composed of discrete word counts, we will assume that we can use the Multinomial distribution to fit the likelihood data distribution $"p(x_i|z, \theta")$. In general, given that there are different types of data distributions in different problems, the appropriate models and parameters in Naive Bayes models can be fit to the data by calculating the Maximum Likelihood Estimate for each class’ distribution. To obtain a fully Bayesian model it is also necessary to choose and fit the prior distribution $"p(z|\theta")$. In this case, since our two sentiment classes are evenly distributed we will assume the prior follows the standard binomial distribution.

Having identified and fitted the appropriate distributions, the model can now calculate the probability $"p(z|x, \theta")$ that a particular review is positive or negative as a function of its features. This results in a simple linear classifier that identifies an implicit feature cutoff (i.e. a decision boundary). This is then used as our classification tool once we input new unclassified reviews into the model[3].

4 Results and Discussion

4.1 Initial Classifier Comparison

After training all of the aforementioned classifiers on the ”bag of words” representation of the training data, we evaluated their performance against the labeled testing data. Overall all the classifiers where able to sort the data significantly better than a random sorter. Nevertheless, it is clear that the worst overall classifier (although the most interesting) was the KNN; being the worst at identifying positives from negatives (low Recal), while being somewhat average at making sure positives are indeed true positives (Precision), and the best at making sure a labeled negative is a true negative (high Specificity). This type of behaviour could be explained by showing that negative indicator



Classifier/Metrics	KNN	Naïve Bayes	Log Reg	Tree	Linear SVM
Accuracy	0.64	0.77	0.8	0.73	0.78
Precision	0.80	0.78	0.85	0.73	0.82
Recal	0.38	0.75	0.73	0.72	0.72
F1-Score	0.52	0.77	0.79	0.73	0.77
Specificity	0.91	0.79	0.87	0.73	0.85
AUC	0.74	0.86	0.88	0.76	0.86

Figure 1: Classifier Performance

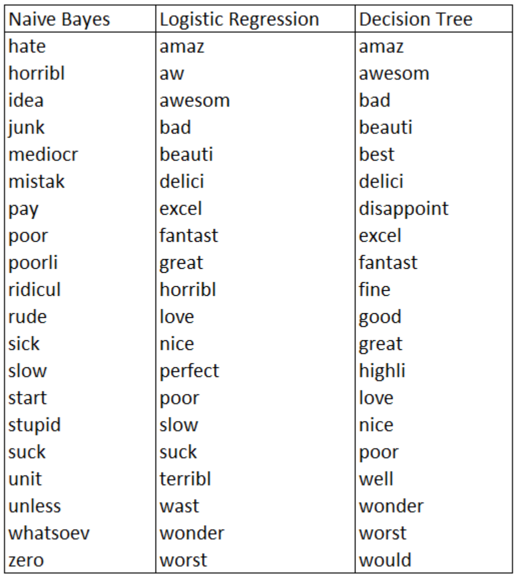
features are more prone to clustering than positive indicator features in the feature space. However, a preliminary exploratory graph did not show such a thing (see appendix). Having said that, the use of this classifier can only be justified in cases in which we need to make absolutely sure that a labeled negative is a true negative. However, this would be of little use if it can't identify positives from negatives at a satisfactory rate.

The performance of all other classifiers was somewhat even across their own individual metrics (the probability of mislabeling true positives and negatives were about equal for a given classifier). Therefore, for the rest of the report we will only rank the classifiers based on their accuracy and AUC . The resulting rank, from best to worst, was: 1) Logistic Regression, 2a) Linear SVM, 2b) Naive Bayes, 3) Decision Tree, and 4) KNN.

Furthermore, we also evaluated the classifiers' ability to predict sentiment based on the length of each individual review as the only feature. The results (see appendix) show that review length is not a good indicator of sentiment, as none of the classifiers were able to have an accuracy of more than 0.53 (the classifiers became essentially random). We then attempted to use the ”bag of words” representation in conjunction with review length. In a classic example of over-fitting, these last results actually got worse when compared to the ones shown in Figure 1. Please refer to appendix for a list of results and the ROC curves used to calculate the AUC metric in Figure 1.

4.2 Comparison with Feature Selection

We then proceeded to rank the predictive capability of each word within our ”bag of words” through a selection algorithm called Recursive Feature Elimination (RFE) [9]. This algorithm assigns a weight to each word by recursively considering smaller and smaller subsets of the features for a given classifier. The least important features are filtered out until it reaches a user-desired number of features. As stated before, we tested our algorithm performance for 20, 50, 100, 200, 300, 400, and 541 features. For simplicity, we chose only 3 classifiers out of the 5 to do this particular feature reduction test. Figure 2 shows the top 20 features for each test algorithm.



Naive Bayes	Logistic Regression	Decision Tree
hate	amaz	amaz
horribl	aw	awesom
idea	awesom	bad
junk	bad	beauti
mediocr	beauti	best
mistak	delici	delici
pay	excel	disappoint
poor	fantast	excel
poorli	great	fantast
ridicul	horribl	fine
rude	love	good
sick	nice	great
slow	perfect	highli
start	poor	love
stupid	slow	nice
suck	suck	poor
unit	terribl	well
unless	wast	wonder
whatsoev	wonder	worst
zero	worst	would

Figure 2: Most Predictive Features for Different Classifiers

As expected, each classifier had a different set of words that it considers the most important. Interestingly, the Naive Bayes classifier ranked words with a negative connotation as the most important, while the Logistic Regression and Decision Tree classifiers preferred of a mixture of both positive and negative words. However, all classifiers agreed that words such as ”poor” and ”worst” are strong indicators of a negative sentiment, while ”amazing” and ”awesome” were good indicators of a positive sentiment.

We then studied the effect that the feature number has on our classifier's Accuracy and AUC (the two metrics we chose in our previous analysis).

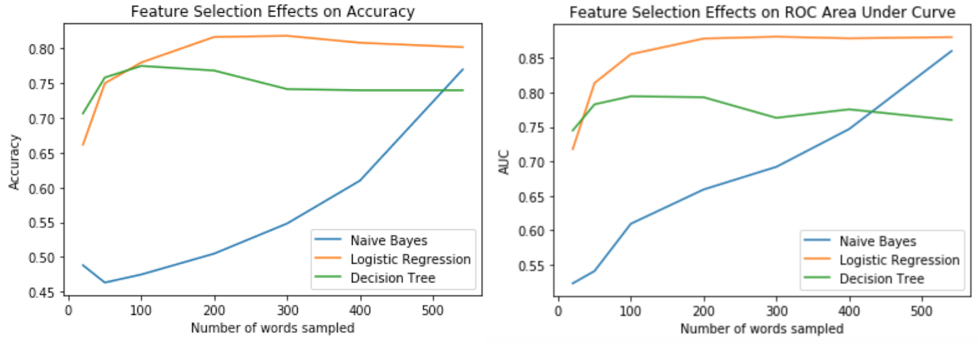


Figure 3: The Effect of Number of Features (Words) on the Accuracy and AUC of Different Classifiers

We can reach the following conclusions from Figure 3: 1) The Naive Bayes classifier requires a large number of features to even approach the accuracy and AUC of its competitors. This shows that the Navies Bayes classifier is a lot more sensible to feature selection that the other two. This may have to do with the fact that the RFE study of the Naive Bayes seems to initially favor ”negative” words; once the classifier starts including ”positive” words into the vocabulary its performance drastically increases. 2) Both the the Decision Tree and the Logistic Regression classifiers achieve their best performance with a reduced number of features (around 200), after which the models become slightly over-fitted and performance suffers. This shows that, having identified the correct features at the decision limit, the Decision Tree and the Logistic Regression classifiers have very little use of the additional features (in agreement with their modeling characteristics).

5 Conclusions

In this report we analyzed the accuracy, precision, recall, F1-scores, specificity, and AUC of 5 different classifiers when used to predict the sentiment of online reviews. Although all of them were able to sort the reviews better than a random classifier, there was definitely a clear variance between their performances. Interestingly, the worst overall classifier was the one with best specificity (KNN). Overall, we concluded that the best classifier was the Logistic Regression algorithm as it showed a higher accuracy, precision, F1- score, and AUC than any of its counterparts.

We also investigated the predictive capability of an additional feature: review length. Our analysis clearly showed that this feature is not indicative of sentiment.

Finally we studied the effect of the number of features on these metrics, concluding that that more is not always better (unless you are a Naive Bayes classifier). Once again, the Logistic Regression algorithm stood out, showing that it requires the least number of features to achieve the best performance.

There are many future directions that we can take in the further development of this study. First, it will be interesting to attempt to train the models with a larger data set in order to find the number of words/features at which the performance of the Naive Bayes classifier reaches the one of Logistic Regression (if at all). Second, it would be very insightful to expand our feature selection out of the ”bag of words” representation and into the aforementioned ”Word to Vector” or Latent Semantic Indexing algorithms. Third, we could expand the reach of our sentiment analysis by including new classification labels such as ”neutral”, ”very positive”, and ”very negative”.

References

[1] J. Brownlee. How and When to Use ROC Curves and Precision-Recall Curves for Classification in Python, 2018.

[2] Barbara Engelhardt. Precept #3.

[3] Barbara Engelhardt. Probabilistic classification models, 2019.

[4] George W Furnas, Thomas K Landauer, Scott Deerwester, Richard Harshman, and Susan T Dumais. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 2004.

[5] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A Structured Self-attentive Sentence Embedding. 2017.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. 2013.

[7] Pedregosa Fabian, Vincent Michel, Grisel OLIVIER, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Jake Vanderplas, David Cournapeau, Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Bertrand Thirion, Olivier Grisel, Vincent Dubourg, Alexandre Passos, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[8] S J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. [[Prentice Hall]], 2009.

[9] Scikit-learn. RFE Feature Selection.

[10] Sida Wang and C Manning. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 94305(1):90–94, 2015.

[11] Zhi-Hua Zhou and Ji Feng. Deep Forest. *arXiv*, 2017.

6 Appendix

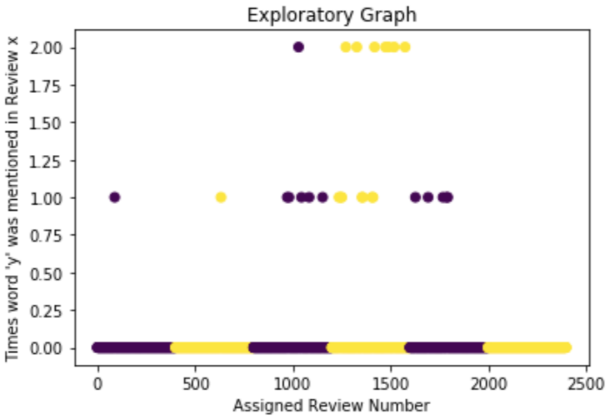


Figure 4: Exploratory Graph

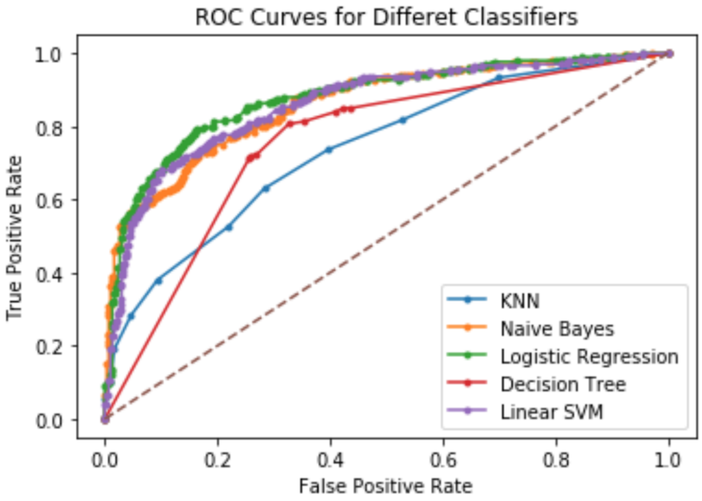
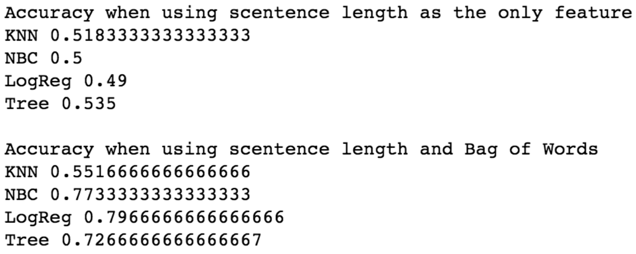


Figure 5: ROC curve for the 5 classifiers



```
Accuracy when using scentence length as the only feature
KNN 0.5183333333333333
NBC 0.5
LogReg 0.49
Tree 0.535

Accuracy when using scentence length and Bag of Words
KNN 0.5516666666666666
NBC 0.7733333333333333
LogReg 0.7966666666666666
Tree 0.7266666666666667
```

Figure 6: Review Length Accuracy Results