How many stars will I give?
Predicting ratings of Amazon reviews

Tonia Chu

Under the mentorship: Srdjan Santic

For the course: Data Science Career Track (Springboard)

Content

- Introduction
- Deeper dive into the data set
- Pre-processing
- Feature Engineering
- Modeling and Machine learning
- Results and Discussion
- Future Work

Introduction

- Many consumers are effectively influenced by online reviews when making their purchase decisions.
- The star-rating, i.e. stars from 1 to 5 on Amazon, gives a quick overview of the product quality.
- **The purpose of this project is to develop models that are able to predict the user rating from the text review.**

DEEPER DIVE INTO THE DATA SET

- We get dataset from Amazon product data, which contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 ~ July 2014.
- In this project, we use 5-core dataset of Clothing and Shoes, which is subset of the data in which all users and items have at least 5 reviews.

Sample review

"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",

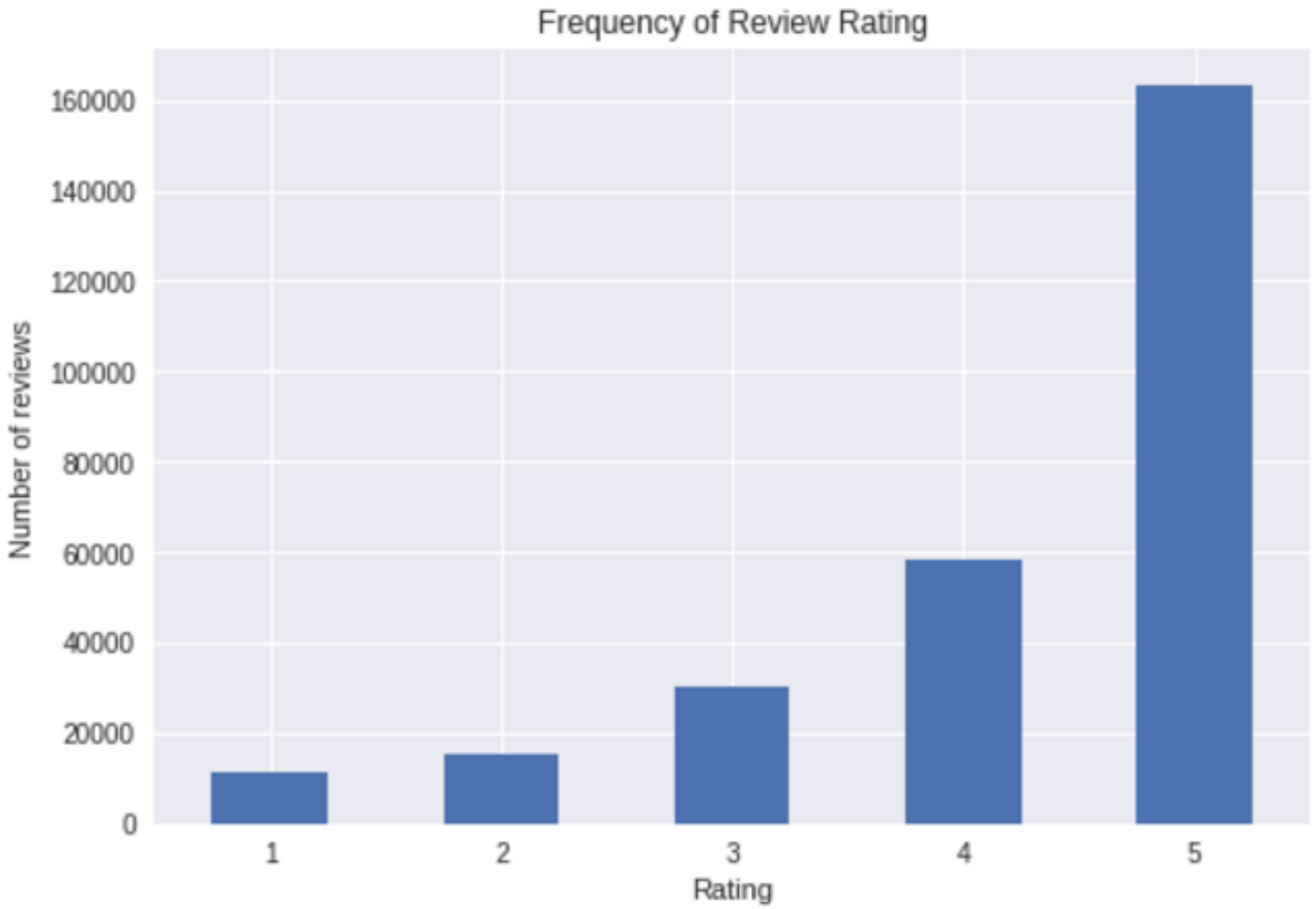
"unixReviewTime": 1252800000,

"reviewTime": "09 13, 2009"

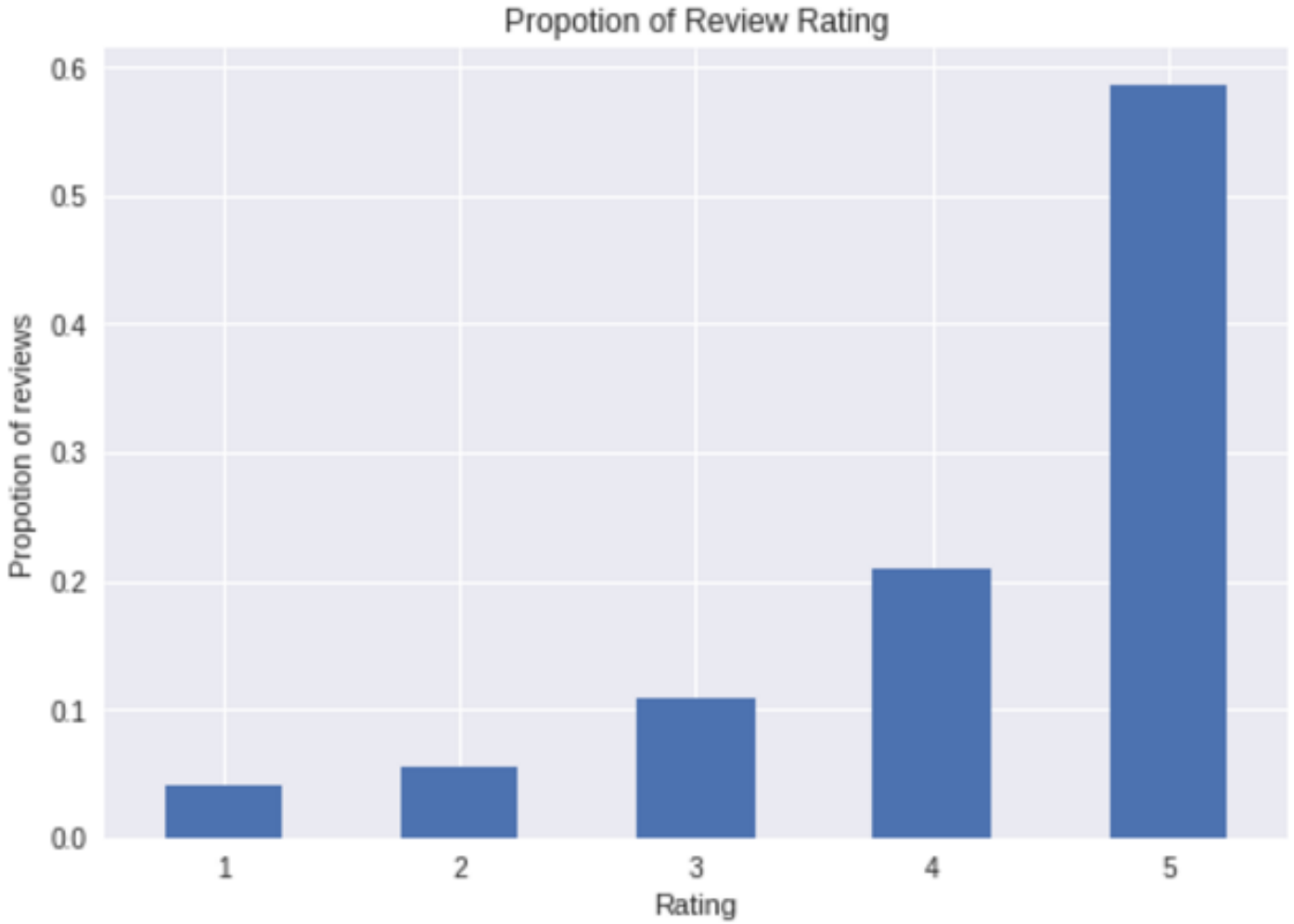
Summary of the dataset

- Number of reviews: 278677
- Number of unique reviewers: 39387
- Prop of unique reviewers: 0.141
- Number of unique products: 23033
- Prop of unique products: 0.083
- Average rating score: 4.245

Distribution of rating score



Distribution of rating proportion



Subset data

- We select the first 30,000 reviews as our sub dataset.
- Distribution of the ratings is as following:

Rating	
1	1070
2	1541
3	2842
4	6042
5	18505

Pre-processing —— Text Normalization

- Cleaning Text
- Tokenizing Text
- Removing Special Characters
- Expanding Contractions
- Case Conversions
- Removing Stopwords
- Correcting Words
- Stemming
- Lemmatization

1. Expanding Contractions

- Contractions are shortened version of words or syllables. They exist in either written or spoken forms. Shortened versions of existing words are created by removing specific letters and sounds. In case of English contractions, they are often created by removing one of the vowels from the word.
- By nature, contractions do pose a problem for NLP and text analytics because, to start with, we have a special apostrophe character in the word. Ideally, we can have a proper mapping for contractions and their corresponding expansions and then use it to expand all the contractions in our text.

2. Removing Special Characters

- One important task in text normalization involves removing unnecessary and special characters. These may be special symbols or even punctuation that occurs in sentences.
- This step is often performed before or after tokenization. The main reason for doing so is because often punctuation or special characters do not have much significance when we analyze the text and utilize it for extracting features or information based on NLP and ML.

3. Tokenizing Text

- *Tokenization* can be defined as the process of breaking down or splitting textual data into smaller meaningful components called tokens.
- *Sentence tokenization* is the process of splitting a text corpus into sentences that act as the first level of tokens which the corpus is comprised of. This is also known as sentence segmentation , because we try to segment the text into meaningful sentences.
- *Word tokenization* is the process of splitting or segmenting sentences into their constituent words. A sentence is a collection of words, and with tokenization we essentially split a sentence into a list of words that can be used to reconstruct the sentence.

4. Removing Stopwords

- Stopwords are words that have little or no significance. They are usually removed from text during processing so as to retain words having maximum significance and context.
- Stopwords are usually words that end up occurring the most if you aggregated any corpus of text based on singular tokens and checked their frequencies. Words like a, the , me , and so on are stopwords.

5. Correcting Words

- Correcting Repeating Characters
- Correcting Spellings

6. Lemmatization

- The process of lemmatization is to remove word affixes to get to a base form of the word.
- The base form is also known as the root word, or the lemma, will always be present in the dictionary.

Feature Engineering —— Feature Extraction

- *Feature engineering* is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive.
- In ML terminology, features are unique, measurable attributes or properties for each observation or data point in a dataset. Features are usually numeric in nature and can be absolute numeric values or categorical features that can be encoded as binary features for each category in the list using a process called one-hot encoding.
- The process of extracting and selecting features is both art and science, and this process is called *feature extraction* or *feature engineering*.

Feature-extraction techniques

- Bag of Words model
- TF-IDF model
- Averaged Word Vectors
- TF-IDF Weighted Averaged Word Vectors

1. Bag of Words Model

- The Bag of Words model is perhaps one of the simplest yet most powerful techniques to extract features from text documents.

- The essence of this model is to convert text documents into vectors such that each document is converted into a vector that represents the frequency of all the distinct words that are present in the document vector space for that specific document.

2. TF-IDF Model

- TF-IDF stands for *Term Frequency-Inverse Document Frequency*, a combination of two metrics: term frequency and inverse document frequency.

- Mathematically, TF-IDF is the product of two metrics and can be represented as tfidf = tf * idf, where *term frequency*(tf) and *inverse-document frequency*(idf) represent the two metrics.

3. Averaged Word Vectors

- In this technique, we will use an average weighted word vectorization scheme, where for each text document we will extract all the tokens of the text document, and for each token in the document we will capture the subsequent word vector if present in the vocabulary.
- We will sum up all the word vectors and divide the result by the total number of words matched in the vocabulary to get a final resulting averaged word vector representation for the text document.

3. Averaged Word Vectors

$$AWV(D) = \frac{\sum_{1}^{n} wv(w)}{n}$$

- AVW(D) is the averaged word vector representation for document D, containing words w1, w2, ..., wn,
- wv(w) is the word vector representation for the word w.

4. TF-IDF Weighted Averaged Word Vectors

- Now we use a new and novel technique of weighing each matched word vector with the word TF-TDF score and summing up all the word vectors for a document and dividing it by the sum of all the TF-IDF weights of the matched words in the document.
- This would basically give us a TF-IDF weighted averaged word vector for each document.

4. TF-IDF Weighted Averaged Word Vectors

$$TWA(D) = \frac{\sum_{1}^{n} wv(w) \times tfidf(w)}{n}$$

- TWA(D) is the TF-IDF weighted averaged word vector representation for document D, containing words w1, w2, ..., wn,
- wv(w) is the word vector representation,
- tfidf(w) is the TF-IDF weight for the word w.

Modeling and Machine learning

- Classification Models
- Evaluating Classification Models
- Confusion Matrix of Models
- Hyperparameter Tuning

Classification Models

- Logistic Regression
- Multinomial Naive Bayes
- Linear Support Vector Classification
- SGD Classifier
- Random Forest Classifier

Evaluating Classification Models

30K data	Ratings	Bag of words features	tfidf features	Averaged word vector features	tfidf weighted averaged word vector features
Logistic Regression	1	0.25	0.15	0.07	0.03
	2	0.2	0.08	0.02	0.02
	3	0.23	0.17	0.07	0.05
	4	0.26	0.23	0.18	0.16
	5	0.91	0.95	0.96	0.97
	avg	0.654	0.658	0.641	0.636
Multinomial Naive Bayes	1	0.3	0		
	2	0.18	0		
	3	0.27	0.01		
	4	0.32	0.03		
	5	0.86	0.99		
	avg	0.637	0.621		
Linear Support Vector Classification	1	0.31	0.29	0.02	0.01
	1	0.24	0.19	0.01	0
	3	0.23	0.23	0.03	0.02
	4	0.25	0.26	0.17	0.16
	5	0.87	0.91	0.97	0.98
	avg	0.63	0.655	0.639	0.636
SGD Classifier	1	0.3	0.25	0.1	0.06
	2	0.16	0.12	0.06	0
	3	0.14	0.12	0	0.12
	4	0.36	0.14	0.4	0.15
	5	0.84	0.97	0.86	0.9
	avg	0.624	0.655	0.621	0.6
Random Forest Classifier	1	0.09	0.07	0.1	0.09
	2	0.05	0.05	0.07	0.08
	3	0.1	0.11	0.15	0.14
	4	0.2	0.18	0.26	0.24
	5	0.91	0.93	0.85	0.84
	avg	0.616	0.625	0.6	0.588

Confusion Matrix of Models

- A *confusion matrix* is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.
- Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class.

Confusion Matrix of Models

- Logistic Regression using Bag of words features

0	1	2	3	4	5
1	54	42	19	20	79
2	20	62	60	47	119
3	21	45	129	155	219
4	8	16	93	318	773
5	9	11	77	245	3359

- Logistic Regression with Tfidf features

0	1	2	3	4	5
1	32	28	27	21	106
2	12	26	59	47	164
3	3	19	99	161	287
4	2	6	55	272	873
5	0	4	21	157	3519

Confusion Matrix of Models

- Multinomial Naive Bayes using Bag of words features

0	1	2	3	4	5
1	65	48	32	23	46
2	30	55	83	55	85
3	14	53	151	172	179
4	9	30	128	383	658
5	23	32	89	389	3168

- Linear Support Vector Classification using Tfidf features

0	1	2	3	4	5
1	62	35	24	23	70
2	28	60	50	58	112
3	19	42	129	168	211
4	12	18	93	314	771
5	7	14	61	257	3362

Confusion Matrix of Models

- SGD Classifier using Tfidf features

0	1	2	3	4	5
1	54	22	10	10	118
2	20	36	34	41	177
3	12	28	69	118	342
4	9	14	35	164	986
5	3	7	16	68	3607

- Random Forest Classifier using Averaged word vector features

0	1	2	3	4	5
1	21	24	34	48	87
2	22	23	58	80	125
3	20	27	86	171	265
4	19	36	84	313	756
5	22	56	94	399	3130

Hyperparameter Tuning

- *Hyperparameters* are parameters whose values are set prior to the commencement of the learning process. By contrast, the values of other parameters are derived via training. Given these hyperparameters, the training algorithm learns the parameters from the data.

- *Hyperparameter tuning* is the problem of choosing a set of optimal hyperparameters for a learning algorithm. The same kind of machine learning model could require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters, and have to be tuned so that the model can best solve the machine learning problem.

Hyperparameter tuning result

No.	Models	Accuracy before tuning	Accuracy after tuning	Improved accuracy
1	Logistic Regression using Bag of words features	0.654	0.658	0.004
2	Logistic Regression with Tfidf features	0.658	0.662	0.004
3	Multinomial Naive Bayes using Bag of words features	0.637	0.636	-0.001
4	Linear Support Vector Classification using Tfidf features	0.655	0.662	0.007
5	SGD Classifier using Tfidf features	0.655	0.657	0.002
6	Random Forest Classifier using Averaged word vector features	0.600	0.637	0.037

Results and Discussion

We can improve the models by following attempts:

- Train the models with bigger dataset and more data.
- Set ngram_range to (1,2) and (1,3) to extract better features.
- Use Summary texts as well as review texts to predict the rating scores.
- Create the hyperparameter grid with more parameters and bigger search space.

Future Work

- Use full datasets of different categories, such as Books, Electronics or Movies and TV.
- Develop models to predict ratings with review texts and summary.
- Evaluate models with data from other categories or other websites beside Amazon.
- Use advanced modern NLP tech in text preprocessing.

Thank You!