Amazon Product Rating Prediction

Avinash Jha

Under the mentorship: Mohit Khanna

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,

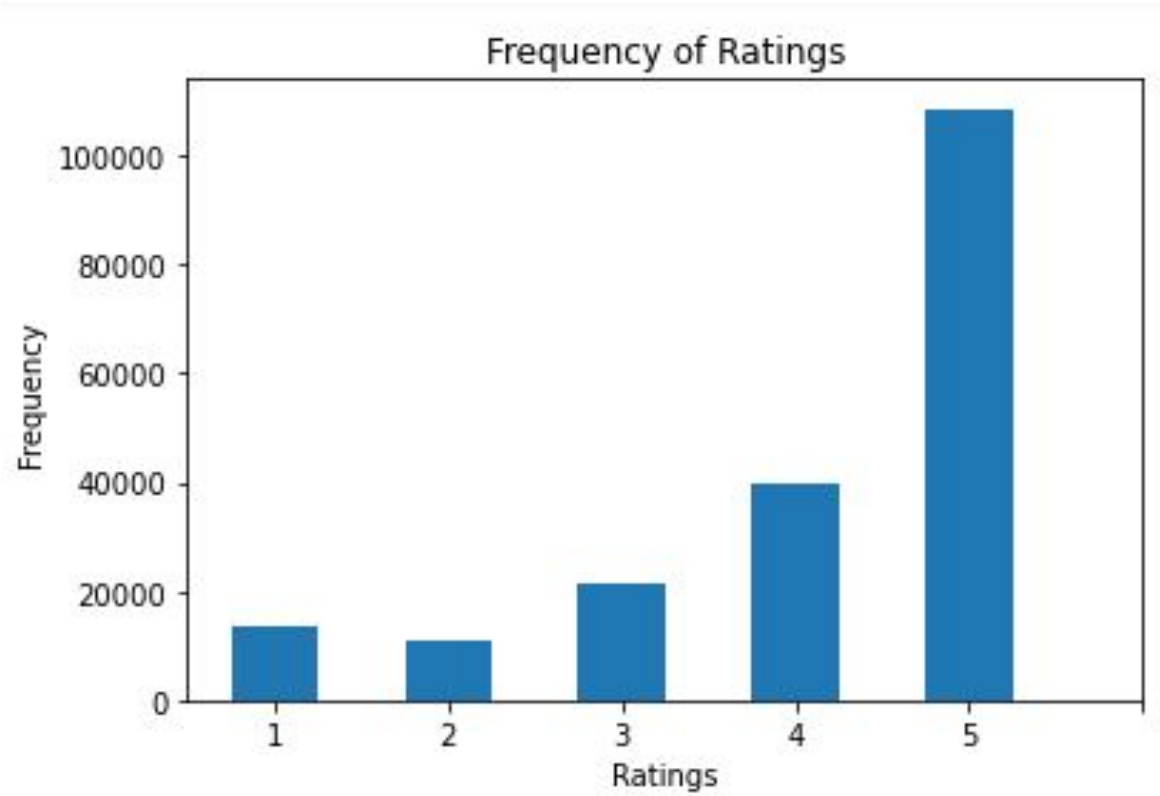
"reviewTime": "09 13, 2009"

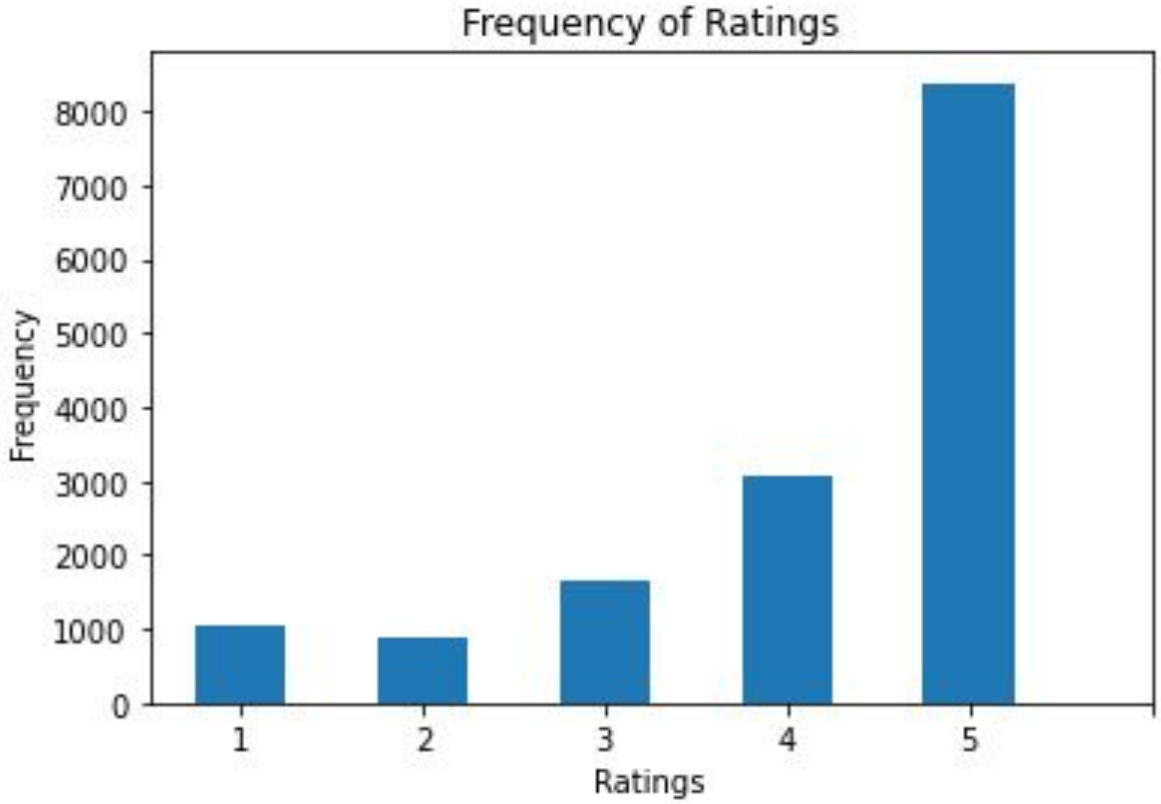
Summary of the dataset

```
##################################################
ENTIRE CELL PHONE AND ACCESSORIES DATASET

Total Number of Reviews:  194439

Total Number of Unique Users:  27879

Total Number of Unique Products:  10429

Averge Rating Score:  4.129912208970422

##################################################
```

```
##################################################
SUBSET of CELL PHONE AND ACCESSORIES DATASET

Total Number of Reviews:  15000

Total Number of Unique Users:  11436

Total Number of Unique Products:  6102

Averge Rating Score:  4.126533333333334

##################################################
```

Distribution of rating score





```
Ratings
1      13279
2      11064
3      21439
4      39993
5     108664
dtype: int64
```

```
Ratings
1     1033
2      875
3     1635
4     3075
5     8382
dtype: int64
```

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 words or combinations of words that are shortened by dropping letters or sounds and replacing them with an apostrophe. They exist in either written or spoken forms. In the 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 punctuations that occur in sentences.
- Special Characters and symbols are usually non-alphanumeric characters or even occasionally numeric characters(depending on the problem) which adds extra noise to unstructured text data and does not add much significance while analyzing text and utilizing it for feature extraction

3. Tokenizing Text

- Tokenization is the process of transforming a string or document into smaller chunks, which we call tokens. This is usually one step in the process of preparing a text for natural language processing.
- Sentence Tokenization is a process of converting text corpus into sentences which is the first level of tokens. This is also called Sentence Segmentation because we try to segment the text into meaningful sentences.
- Word Tokenization is a process of splitting sentences into words.

4. Removing Stopwords

- Stopwords are the words that has little or no significance especially when constructing meaningful features from text. They are removed from the text so that we are left with words having maximum significance.
- They are usually words that have maximum frequency if you aggregate any corpus of text based on singular tokens.
- Ex:- a, the, of and so on.

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 can be defined as a process of making a machine learning algorithm that works well by using domain knowledge to create features in the data set. It is fundamental in the application of machine learning.
- Features can be described as a unique and measurable property for each row or observation in a dataset. Machine learning algorithms usually work with numerical features. If in case they are not numerical, there are various techniques that can be used to deal with such features, like one-hot encoding, imputation, and scaling, etc.

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, A.K.A BoW is the easiest yet very effective technique to extract features from text data that can be used to train machine learning models.
- The approach is very simple, we take a text document and convert them into vectors and each vector represents the frequency of that word in that document. In simple words, a bag-of-words is a representation of text that describes the occurrence of words within a document.

2. TF-IDF Model

- TF-IDF stands for Term Frequency-Inverse Document Frequency.
- Term Frequency is nothing but what we have computed in the Bag of Words Model ie count of each word in a doc stored in the form of a vector.
- Inverse document frequency denoted by IDF is the inverse proportion of the number of total Documents and the Number of documents that have that word, on a logarithmic scale.

2. TF-IDF Model

Tf-idf formula

$$w_{i,j} = tf_{i,j} * \log(\frac{N}{df_i})$$

$w_{i,j}$ = tf-idf weight for token i in document j

$tf_{i,j}$ = number of occurences of token i in document j

df_i = number of documents that contain token i

N = total number of documents

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

Evaluating Classification Models

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

3.1 Logistic Regression with Bag of words features

0	1	2	3	4	5
1	91	29	21	15	51
2	35	25	33	20	62
3	27	30	79	66	125
4	10	17	71	172	345
5	23	20	46	172	1415

3.2 Logistic Regression with Tdidf features

0	1	2	3	4	5
1	79	2	15	14	97
2	24	2	24	34	91
3	15	4	55	72	181
4	6	3	31	156	419
5	7	1	13	95	1560

Confusion Matrix of Models

3.3 Logistic Regression with Average word vector features

0	1	2	3	4	5
1	41	0	2	5	159
2	12	1	4	17	141
3	14	0	14	34	265
4	6	0	11	56	542
5	14	0	8	36	1618

3.4 Logistic Regression with Tdidf weighted average word vector features

0	1	2	3	4	5
1	30	0	1	4	172
2	13	0	5	12	145
3	13	1	13	18	282
4	11	0	6	43	555
5	11	0	5	32	1628

Confusion Matrix of Models

3.5 Random Forest Classifier with Bag of words features

0	1	2	3	4	5
1	31	0	5	4	167
2	2	1	7	13	149
3	2	0	19	21	285
4	1	0	10	44	560
5	2	0	5	23	1646

3.6 Random Forest Classifier with Tdidf features

0	1	2	3	4	5
1	28	1	2	6	170
2	4	0	6	13	152
3	2	1	13	19	292
4	2	0	8	40	566
5	1	0	4	11	1660

Confusion Matrix of Models

3.7 Random Forest Classifier with Average word vector features

0	1	2	3	4	5
1	33	0	11	12	151
2	11	0	10	24	130
3	7	2	23	46	249
4	12	1	10	82	510
5	17	1	17	84	1557

3.8 Random Forest Classifier with Tdidf weighted average word vector features

0	1	2	3	4	5
1	28	1	8	12	158
2	8	3	7	25	132
3	15	3	19	42	248
4	9	1	11	76	518
5	12	2	23	86	1553

Hyperparameter Tuning

- Hyperparameters are parameters whose values are used to control Learning Process. Hyperparameters are set before the model begins to learn. Different models have different hyperparameters. Like the depth of a decision tree.
- Hyper-parameters are parameters that are not directly learned within estimators. In scikit-learn, they are passed as arguments to the constructor of the estimator classes.

Hyperparameter Tuning

- In machine learning, hyperparameter tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. Usually, a metric is chosen to measure the algorithm's performance on an independent data set and hyperparameters that maximize this measure are adopted. Often cross-validation is used to estimate this generalization performance.

- There are two ways of Hyperparameter tuning:-
  - GridSearchCV, exhaustively considers all parameter combinations
  - RandomSearchCV, sample a given number of candidates from a parameter space with a specified distribution

Hyperparameter tuning result

No.	Models	Accuracy before tuning	Accuracy after tuning	Improved accuracy
1	Logistic Regression using Bag of words features	0.594	0.614	0.02
2	Logistic Regression with Tfidf features	0.617	0.617	0.0
3	Random Forest Classifier using Bag of words features	0.580	0.581	0.001
4	Random Forest Classifier with Tfidf features	0.580	0.577	-0.003

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

- Full dataset or datasets of other categories can be explored.
- Summary can also be included to train the model
- Other Normalization Techniques, like stemming and Expanding Contractions can be include.
- Other Classification models can be implemented.
- Models can be evaluated using data from other websites
- Use advanced modern NLP tech in text preprocessing.

Thank You!