Text Review Sentiment Analysis using Classification Modelling vs Regression Modelling - Data Quality Report

CONTRIBUTORS: ADAM CLEAVER BENG, ING. LUKAS TOPINKA, M.SC.
DATASCIENTEST SUPERVISOR: MAËLYS BASSILEKIN BLANC

Introduction

This report analyses data from Amazon reviews from the Appliances Category, the data was originally collected in 2014 and most recently updated in 2018. Though the data has been parsed for NLP usage, extra Data Cleaning and preprocessing is required to enable the variety of modelling techniques that will be tested in the main report.

The features will be derived from the review text of each review in the data and the target variable is the rating from 1-5 stars. In this report, the feature text data will be checked for duplicates and NaN values, and vectorized.

Other columns will also be processed to enable further investigations and project expansions.

Method

Data Source

Source Citation:

> **Justifying recommendations using distantly-labeled reviews and fined-grained aspects**
>
> Jianmo Ni, Jiacheng Li, Julian McAuley
>
> *Empirical Methods in Natural Language Processing (EMNLP), 2019*
>
> https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

From this data, only the appliance reviews were utilized for this project. These were selected as the number of records is between 500k and 1m records. This criterion was estimated to ensure that enough data is present to produce a well-fitting model, but not so many that computational times will exceed 24 hours.

Column Heading	Type	Brief
overall	Integer	[TARGET] rating of the product
vote	object	helpful votes of the review
verified	bool	verified buyers
reviewTime	object	time of the review (raw)
reviewerID	object	"ID of the reviewer, e.g. A2SUAM1J3GNN3B"
asin	object	"ID of the product, e.g. 0000013714"
style	object	"a dictionary of the product metadata, e.g., ""Format"" is ""Hardcover"""
reviewerName	object	name of the reviewer
reviewText	object	[FEATURE] text of the review
summary	object	summary of the review
unixReviewTime	integer	time of the review (unix time)
image	object	images that users post after they have received the product

Data Quality

The source data is converted from a JSON to a DataFrame using Pandas, there are 602,777 records included.

In those 602,777 records 1,572,305 cells hold NaN values, including 324 in the initial feature column reviewText. There were also 3640 duplicated rows.

Table 1. Raw data NaN Report

Column Heading	Number of Null records
overall	0
vote	537515
verified	0
reviewTime	0
reviewerID	0
asin	0F
style	464804
reviewerName	15
reviewText	324
summary	128
unixReviewTime	0
image	593519

Fig 2. Raw data MIssing Value Heatmap

Initially, rows with no votes were dropped, however it was later decided to keep these records due to the quantity of data lost by excluding them. Any NaN values in votes were instead converted to an integer zero to indicate that the review had received zero votes.

For future expansions on this project, the columns were converted to appropriate Dtypes, reviewtime to datetime64 and unixReviewTime to int. ReviewYear, reviewMonth and reviewDay were added as additional integer columns.

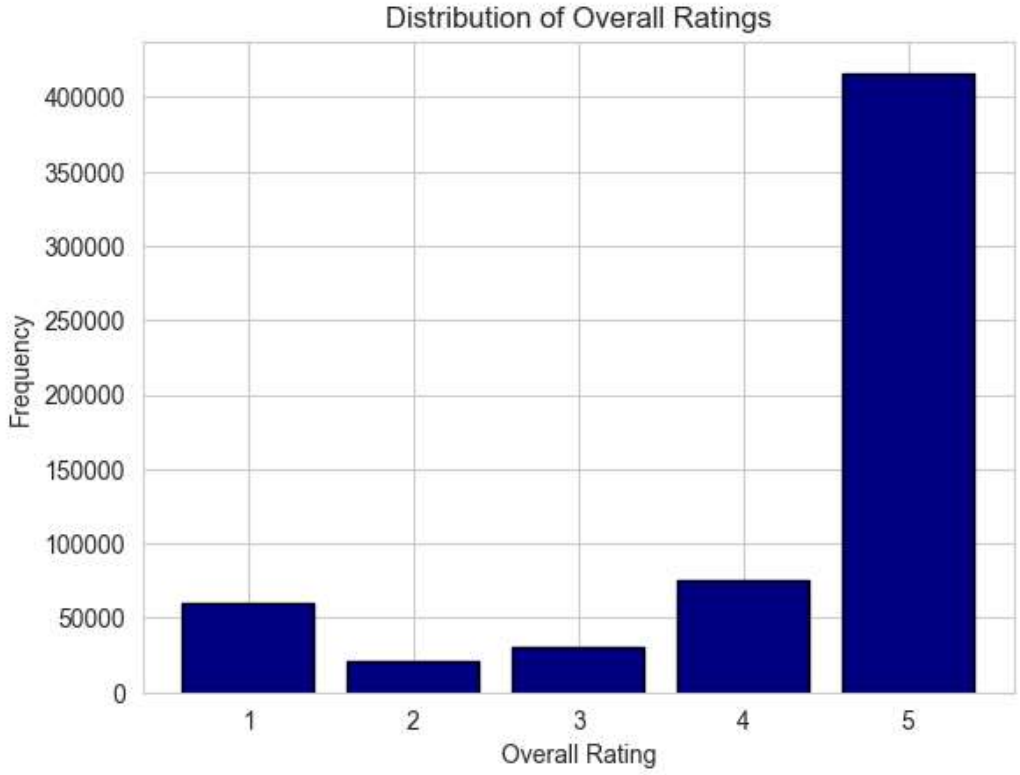
Duplicates were removed and NaN columns were replaced with empty strings. On close inspection it was deemed that the style column does not hold any relevant data for further analysis.

The target column 'overall' is imbalanced. This indicates that the data will need to be sampled to increase classification accuracy. 69% of the reviews awarded a rating of 5, which indicates that a very basic model that simply calls every review 5 stars will be 69% accurate against this data.

Table 3. Raw data overall distribution balance

Rating	Percentage of Reviews
5	69%
4	13%
3	5%
2	3%
1	10%

Fig 4. Barchart representing overall column distribution



The verified Purchasers make up over 75% of the reviews. It could become relevant to include verification status if classification results are poor, though it is unlikely to affect how efficiently our model can analyze the sentiment of the review text. Most of the reviews were recorded in 2016, with a sharp decline in 2017. The low quantity of reports in 2018 could be attributed to the collection time of the sample and perhaps indicates a partial year. Very few reviews were left between the years 2000 and 2010.

The baseline Accuracy is 69% as a simple model that allocates 5 stars to all reviews will be correct 69% of the time against the imbalanced dataset.

Fig 5. Verification column distribution

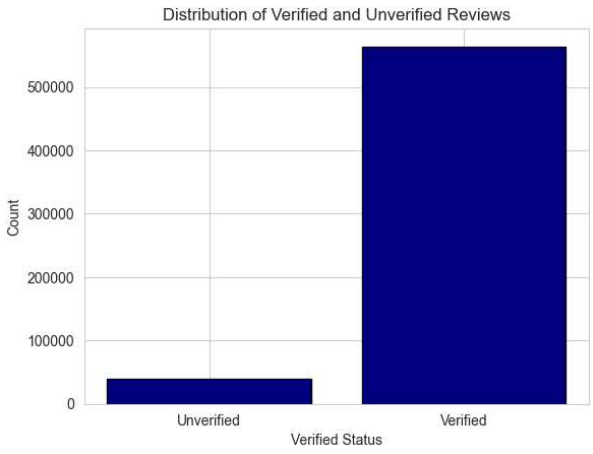
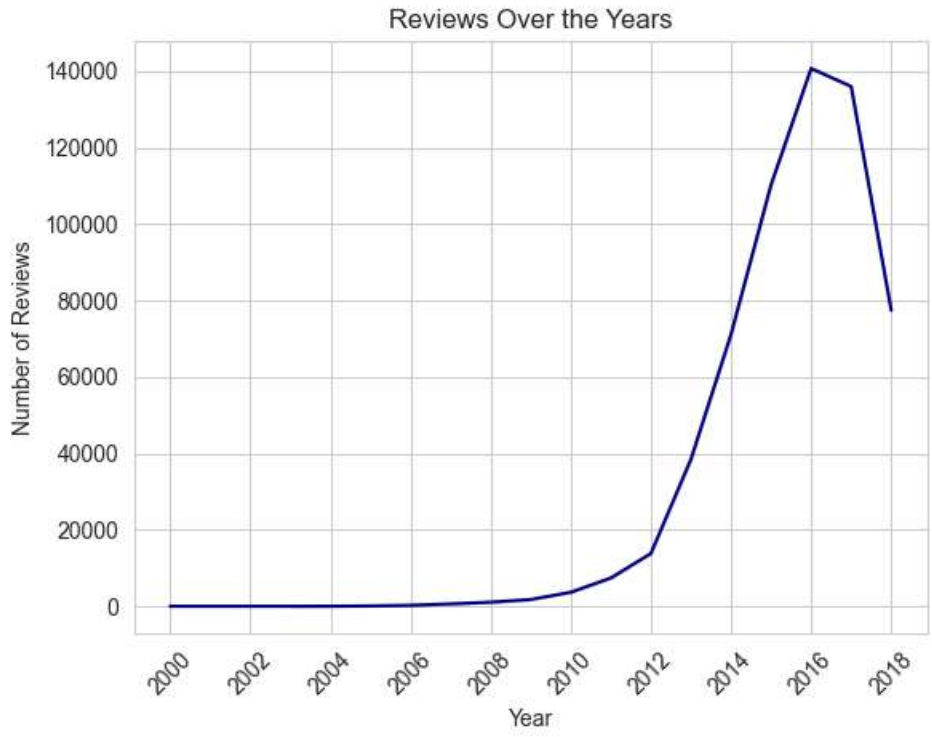
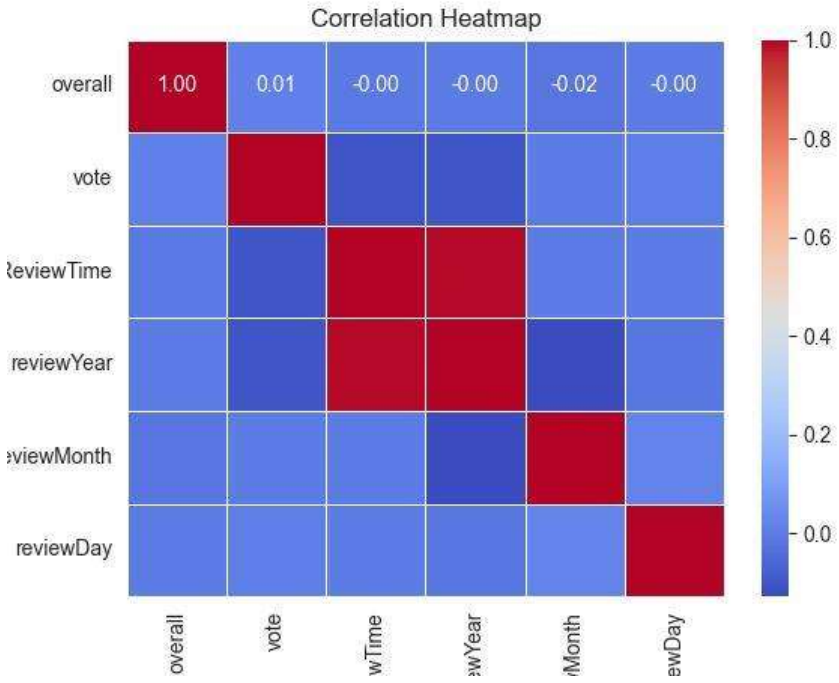


Fig 6. Reviews by year



The numerical columns other than the feature column show approximately 0 correlation to the target overall column.

Fig 7. Numerical Column Correlation Heatmap.



The text available in the reviewText and summary Columns were displayed in WordCloud diagrams. It can be observed that the WordCloud main featured words do not include any brand names, though both Clouds feature the word filter which could be considered for removal as it has no bearing on sentiment. However similarly, it could be that filters receive better reviews which could be important when defining use cases for the model at a later stage. Based on these WordClouds no word removals were made, however it is important to note that in the total dataset there are over 68,000 unique words, so this visualisation may not be optimal.

Fig 8. reviewText Column WordCloud.

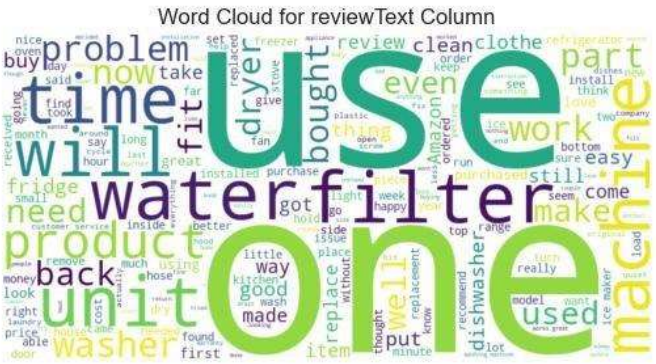


Fig 9. Summary Column WordCloud

Data Preprocessing

For further preprocessing, only the selected Target and Features columns will be used.

Table 9. Selected Target and Features

overall	reviewText
2	"Bought it in August 2017, it croaked in December 2017. RIP."
4	Working great as a replacement for my Sharp original
1	"FROM SELLER- I was able to find parts listings, but no other literature. It should be pretty self explanatory though. It is pretty plug and play.... YOU NEED TO TRY YOUR PLUG AND PLAY- AND YOU WILL DISCOVER IT IS NOT SELF A EXPLAINER THERE ARE MANY STEPS. THAT NEED TO BE DONE.IF YOU DO NOT KNOW WHAT YOU ARE DOING -- DON""T BELITTLE YOUR CUSTOMER"
5	Closest to the one I had originally used.
3	"It doesn't stay cold enough to safely store food. I've tried every trick, actually considering returning it."

To achieve the best possible results from the dataset it is essential to reduce and format the dataset into a data type and format that enables the models to generate the best possible accuracies.

The reviews include html tags for videos and images if any were included in the review. This text must be removed, as must any text including numbers, misspellings must be converted to the correct spelling and tokenised. To achieve these results, Regex, pyspellchecker and the RegExTokenizer were used to preprocess the text in each review. Each review was also compared to nltk's English stopwords and stopwords were removed.

Several further methods were implemented to ready the dataset for modelling. These included WordNetLemmatizer and the EnglishStemmer from the nltk.stem library. These models reduce the words to roots of the word in different formats.

To enable machine learning on these stemming methods, the datasets need to be converted to number vectors. This was achieved using a further two models in the CountVectorizer and TFIDF Vectorizer from sci-kit Learns text library.

It is suspected that different methods of modelling will respond to different methods of preprocessing. In recognition of this suspicion, the review text will be preprocessed using multiple methods.

Method 1 – Tokenization, lemmatizing vs Stemming, CountVectorizer vs TFIDF

In the first preprocessing method the text will be tokenized and vectorized. Models will be tested against four combinations of methods, including:

- Lemmatization, CountVectorization
- Lemmatization, TFIDF Vectorization
- Stemming, CountVectorization
- Stemming, TFIDF Vectorization

The Lemmatizer used is the WordNetLemmatizer from the ntlk library, the Stemmatizer used is the ntlk.stem.snowball EnglishStemmer, the CountVectorizer and the TFIDF Vectorizer are both from the sklearn.feature_extraction.text library, being the CountVectorizer and the TfidfVectorizer class functions.

Table 10. Lemmatized – Stemmatized word Comparison

Lemmatized Words	Count	Stemmatized Words	Count
wa	16756	work	15452
filter	13306	filter	13749
work	11596	great	11162
great	11108	fit	10080
water	9746	use	9891
fit	9740	water	9760
product	7939	product	7982
good	7668	good	7705
ice	5879	just	7247
price	5820	replac	7208
easy	5204	perfect	6976
like	5097	instal	6578
time	4927	need	6275
use	4924	price	6243
humidifier	4809	look	6009
look	4144	ice	5902
perfect	3881	like	5396
new	3808	easi	5204
make	3745	time	5155
year	3734	humidifi	4841
doe	3588	make	4419

Before any text is processed it is tokenized using the Regex Tokenizer to remove all grammar and numbers. Late in the project, it was discovered that this included items from html div tags that were included in the reviewText if the customer had included any video or image references in their review and we updated the Regex Tokenizer to exclude items between the ‘<’, ‘>’ symbols, using a lazy conversion method.

Fig 11. Lemmatized Text Wordcloud

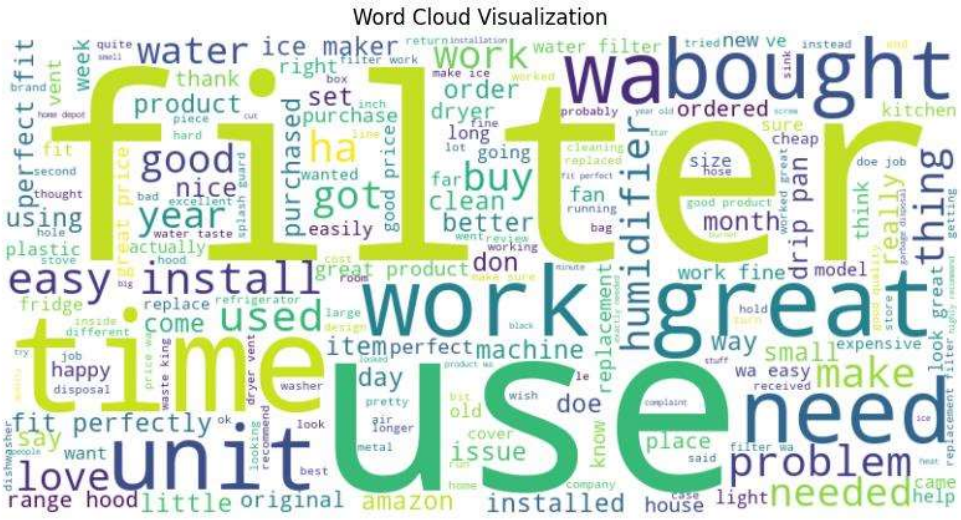
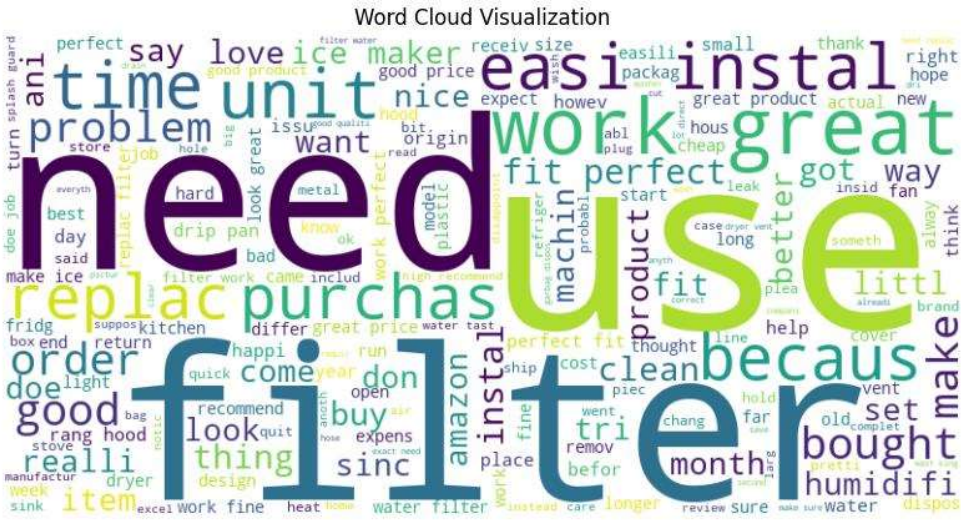


Fig12. EnglishStemmer Text Wordcloud



During Vectorization a working memory limit was regularly hit as there are 68,000 unique words in the dataset. To work around this these data were reduced to a 10% sample of the total dataset, however this introduced a significant reduction in accuracy on the models and final modelling was performed on as much of the dataset as possible within a 24-hour compute time.

The Vectorization methods produce a vector which can be represented as a dataframe by using the get_features_out() method. Each unique string is assigned a column and the count of the appearances in each review are stored in the row associated with the rating. Since all features are of the same scale, no scaling is necessary, though some models were scaled to reduce comput time or fit model prerequisites. Most often the vector data was passed as a sparse matrix, which reduces the data processed by omitting columns with no values for each record.

Table 13. Example CountVector Data

Overall Rating	sharp	self	store	used	explained	customer	safely	august	original	rip
2	0	0	0	0	0	0	0	1	0	1
4	1	0	0	0	0	0	0	0	1	0
1	0	2	0	0	1	1	0	0	0	0
5	0	0	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	1	0	0	0

Method 2 – Google Word2Vec

The Google word2vec preprocessing method was implemented on the dataset by splitting the data into small chunks and concatenating a final library. The main advantage of word2vec over the other methods is that the 68000 unique word entries are compressed into a vector with only 300 features. This means that the models have far less data to process, though noise in the reviews will be baked into the vector.

The Vector is created by taking all reviews in chunks of 50,000 records and removing all text between the div tags. The text is next processed through the text tokenizer and CountVectorized whilst removing stop words. Each batch of 50,000 reviews is stored in a new Dataframe, with the column headings set by the CountVectorizer. Once all files are stored in new dataframes the headers are extracted into a list and the unique values in the total list become the Word2Vec Vocabulary. A new Dataframe is created with the vocabulary as its headers and each dataframe of 50,000 vectorized records is processed through the Word2Vec preprocessing model. In this instance the standard Google Vector model was used, but the option remains available to create a model unique to this use case.

The dataset is now formatted as a DataFrame with 300 features and 602,000 records, the text data is ready to be modelled and is saved as a csv file for future use.

The 300-feature length vector is derived from each word in the review. Each word has a 300-feature vector representation in the model and the sentence or review is represented by the sum of those vectors. In this way the Word2Vec method should be able to detect relationships between words, as well as the individual word values, which will make the models more likely to accurately predict phrases like “not good” or “not bad”

On Sampling

It has been identified that our dataset is imbalanced, and so sampling will need to be performed to give the model a greater chance of accurately predicting review sentiment. Like the other preprocessing methods, it is believed that different preprocessing methods will react well with different modelling techniques, so four sampling techniques have been selected to Sample the data.

From the Imblearn over_sampler library the RandomOverSampler and the SMOTE Sampler are selected, and from the Imblearn under_sampler library the RandomUnderSampler and ClusterCentroids are selected. It is suspected that the ClusterCentroids Sampler will perform better than the others if the data reacts well to classification techniques and if true this would support the theory that classification is a better modelling tool for sentiment analysis than regression models.