Final Project : Amazon Product Review Using Sentiment Analysis

Group Members - Sruti Munukutla, Mahvash Maghrabi, Gmon Kuzhiyanikkal

INTRODUCTION:

Github -
https://github.com/skeerti2/Amazon-Product-Review-Recommendation-System

Working Pattern -
Mobile Electronics Dataset - Mahvash Maghrabi
Furniture Dataset - Sruti Munukutla
Books Dataset - Gmon Kuzhiyanikkal

About the Dataset

Data Source: https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt

This dataset consists of reviews of products from Amazon marketplace from 1995 until 2015. It has 130M+ customer reviews written in multiple languages.

The dataset consists of reviews for multiple categories (Kitchen appliances, Books, Mobile Electronics, Furniture, Software, Tools, Sports etc). We decided to compile data from Books, Mobile Electronics and Furniture and perform sentiment analysis on their reviews. Dataset is compiled in a Tab Separated Value format (TSV). It has 15 features and a total of 104855 (Mobile Electronics reviews) + 100000 (Books) + 791702 (Furniture) = 996557 data points.

We are trying to predict the sentiment analysis of the common day products that we buy from online. We are using three different amazon dataset for working on this. The feature vector of the dataset are as follows :

marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes	total_votes	vine	verified_purchase	review_headline	review_body	review_date

MOBILE ELECTRONICS DATASET:

The mobile electronics dataset has 15 features and 104852 rows. The first step was importing all the necessary libraries required for the implementation. The Natural Language Toolkit (NLTK) is a popular open-source library for natural language processing (NLP) in Python. It provides an easy-to-use interface for a wide range of tasks, including tokenization, stemming, lemmatization, parsing, and sentiment analysis. One of the major advantages of using NLTK is its extensive collection of corpora, which includes text data from various sources such as books, news articles, and social media platforms. This provides a rich data source for training and testing models.

While reading the dataset using the pandas dataframe there were a few errors. Which were handled using “error_bad_lines=False” which ignored two lines that were causing the error in the dataset. Ignoring two lines in the dataset would not affect the dataset accuracy highly.

TEXT PREPROCESSING

The next step was preprocessing the text. Text preprocessing is a crucial step in performing sentiment analysis, as it helps to clean and normalise the text data, making it easier to analyse. The preprocessing step involves a series of techniques that help transform raw text data into a form you can use for analysis. Some common text preprocessing techniques include tokenization, stop word removal, stemming, and lemmatization.

Tokenization

Tokenization is a text preprocessing step that involves breaking down the text into individual words or tokens. This helps in analysing text data as it helps to separate individual words from the raw text and makes it easier to analyse and understand.

Stop Words

Stop word removal is another important step in text preprocessing. It involves removing common and irrelevant words that do not convey much sentiment. Stop words are words that are very common in a language and do not carry much meaning, such as "and," "the," "of," and "it." These words can cause noise and skew the analysis if not removed.

Stemming and Lemmatization

Stemming and lemmatization are used to reduce words to their root forms. Stemming involves removing the suffixes from words, such as "ing" or "ed," to reduce them to their base form. Lemmatization involves reducing words to their base form based on their part of speech.

Removing Punctuations and Float Values

The punctuation marks in the dataset were not required as they had no contribution in showing the sentiment of the reviews hence those were removes. Also few comets had numbers and float values which were affecting the accuracy of sentiment analysis hence they were removed too.

Ground Truth Labels

The dataset had no ground truth hence the star ratings were used to create ground truth for the dataset. The approach followed was as such that the reviews that received star rating lease than or equal to 2 were classified as negative comments and the ones whose ratings were 3,4 and 5 were classified as positive reviews.

Sentiment Analysis

Sentiment analysis is a technique that is used to determine the emotional tone or sentiment expressed in the text or product review in this case . It analyzes the words and phrases used in the text to identify the sentiment whether the review is positive or negative.

Used the SentimentIntensityAnalyzer which is a class in the Natural Language Toolkit (NLTK) library in Python that is used to analyze and determine the sentiment of a piece of text in this case the review. The SentimentIntensityAnalyzer uses a lexicon-based approach to sentiment analysis, which means that it uses a predefined set of words with associated polarity scores to determine the sentiment of the text.

The results obtained from sentiment analysis were categorical values so the categorical values were converted into numerical values for training the ML models. The numerical values were stored in the column ‘sentiment_numerical’.

Now the next step was to check the accuracy of the results obtained from the sentiment analysis with the ground truth labels obtained from the star ratings. A classification report was printed out and the overall accuracy obtained was 80%. The classification report results are as follows :

	precision	recall	f1-score	support
0	0.61	0.48	0.54	24869
1	0.85	0.90	0.88	79983
accuracy			0.80	104852
macro avg	0.73	0.69	0.71	104852
weighted avg	0.79	0.80	0.80	104852

As the results obtained by sentiment analysis gave a good accuracy of 80 % when comparing with the ground truth labels, it can now be used as the output variables for our Machine Learning models. The next step was training the ML models.

BAG OF WORDS MODEL

The bag of words model is a technique that represents text data as a set of numerical features. In this model, each document or piece of text is represented as a "bag" of words, with each word in the text represented by a separate feature or dimension in the resulting vector. The value of each feature is determined by the number of times the corresponding word appears in the text. This allows analysing text data using machine learning algorithms which typically require numerical input. By representing text data as numerical features, we can train machine learning models to classify text or analyse sentiments.

LOGISTIC REGRESSION

Logistic regression is a popular statistical and machine learning technique that is used for binary classification tasks. It involves predicting a binary outcome based on a set of input features. It is a generalised linear model that models the probability of the binary outcome as a function of the input features.
In this project after splitting into training and testing sets. A bag-of-words vectorizer is created using CountVectorizer, which converts the text into a sparse matrix of word frequencies. The training data is fit and transformed using the vectorizer to create a bag-of-words representation of the reviews. The testing data is transformed using the same vectorizer. The logistic regression model is then trained using the bag-of-words representation of the training data. The model is used to make predictions on the testing data using predict. The classification report is as follows. It gets an overall accuracy of 91%.

	precision	recall	f1-score	support
0	0.79	0.74	0.76	3965
1	0.94	0.95	0.95	17006
accuracy			0.91	20971
macro avg	0.86	0.85	0.86	20971
weighted avg	0.91	0.91	0.91	20971

DECISION TREES

A decision tree is a popular supervised machine learning algorithm used for both classification and regression tasks. It is a tree-like model in which each internal node represents a test on a feature or attribute, each branch represents the outcome of the test, and each leaf node represents a class label or a numeric value that represents a prediction. A bag-of-words vectorizer was created using CountVectorizer. The model had an overall accuracy of 85 %. The results are as follows :

	precision	recall	f1-score	support
0	0.61	0.62	0.62	3965
1	0.91	0.91	0.91	17006
accuracy			0.85	20971
macro avg	0.76	0.76	0.76	20971
weighted avg	0.85	0.85	0.85	20971

RANDOM FOREST CLASSIFIER

The Random forest is a supervised Machine learning algorithm used for classification or regression using decision trees. Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. It creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees from a randomly selected subset of the training set and then it collects the votes from different decision trees so as to decide the final prediction.
The dataset is trained using the Random Forest Classifier model. It divides the data set in training and testing sets using the trin_test_split function. A bag-of-words vectorizer was created using CountVectorizer. The input features are 'review_body' and 'review_headline' and the output features are 'sentiment_numerical'. The overall accuracy obtained was 90%. The results from classification report are as follows :

	precision	recall	f1-score	support
0	0.82	0.59	0.68	3965
1	0.91	0.97	0.94	17006
accuracy			0.90	20971
macro avg	0.86	0.78	0.81	20971
weighted avg	0.89	0.90	0.89	20971

ITERATIONS

Two variations were performed on the Decision Tree Model as it had the lowest overall accuracy. The overall accuracy is considered as the evaluation metric.

The bag of words counts the occurrence of each word in a document and represents the document as a vector of word counts. Limitations is that it treats all words equally, regardless of their importance or rarity. This means that common words like "the" and "and" will have a high count and therefore a high weight in the vector representation, even though they may not be very informative about the content of the document. Tf-idf is a weighting scheme that takes into account the importance of each word in a document and in the corpus as a whole. By using tf-idf, the weight of a term increases with its frequency in the document, but is offset by its rarity in the corpus. This means that rare but informative words will have a higher weight than common but less informative words. Therefore, tf-idf is often better than the bag of words model because it provides a more informative representation of the text, taking into account the relative importance of different words.

Grid search with cross-validation is a technique used to optimize hyperparameters for machine learning algorithms. Hyperparameters are settings that are set before the training process and can have a significant impact on the performance of the model. It involves defining a grid of hyperparameter values and then training and evaluating the model for each combination of values in the grid. Cross-validation is used to assess the performance of the model on each combination of hyperparameters by splitting the training data into multiple folds and evaluating the model on each fold. Hence these variations will do better on the metric which is the overall accuracy than the baseline method.

TF-IDF

Term Frequency-Inverse Document Frequency is a numerical statistic that is used to reflect the importance of a word in a document or a collection of documents.The term frequency (TF) of a word in a document is simply the count of how many times that word appears in the document. The inverse document frequency (IDF) of a word is a measure of how rare or unique that word is across all documents in the corpus. The TF-IDF score of a word in a document is calculated as the product of its term frequency and inverse document frequency. It gives a higher weight to words that appear frequently in a document but rarely in the corpus as a whole, and a lower weight to words that appear frequently in both the document and the corpus. The results obtained are as follows. The accuracy increased from 85% to 86 %

	precision	recall	f1-score	support
0	0.62	0.62	0.62	3965
1	0.91	0.91	0.91	17006
accuracy			0.86	20971
macro avg	0.77	0.77	0.77	20971
weighted avg	0.86	0.86	0.86	20971

GRID SEARCH CV

GridSearchCV is a hyperparameter tuning technique used in machine learning to find the optimal set of hyperparameters for a given model.It works by taking a set of hyperparameters, usually defined as a dictionary, and creating all possible combinations of hyperparameters from that set. Then the model is then trained and evaluated for each combination of hyperparameters and the best combination is selected.

The hyperparameters considered for the dictionary were :

'criterion': ['gini', 'entropy']

'max_depth': [None, 5, 10, 15]

'min_samples_split': [2, 5, 10]

'min_samples_leaf': [1, 2, 4]

The accuracy increased from 85% to 87%. The best hyperparameters obtained and the classification report results are as follows :

```
Best hyperparameters: {'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10}
              precision    recall  f1-score   support

           0       0.65      0.65      0.65      3965
           1       0.92      0.92      0.92     17006

    accuracy                           0.87     20971
   macro avg       0.79      0.79      0.79     20971
weighted avg       0.87      0.87      0.87     20971
```

FURNITURE DATASET

The Amazon furniture reviews dataset consists of 792114 rows. Since processing and running ML algorithms on such a huge dataset was taking too much time, the first 20,000 rows have been considered. The 3 ML models used are:

- Logistic Regression
- Decision Tree
- Support Vector Machine

The Preprocessing for review_body column consisting of the reviews includes the following steps:

Punctuation removal & Tokenizer:

This is achieved using a regex tokenizer which matches words and removes any unwanted punctuation.

Removing stop words: Since stop words do not add to any context, they were removed during pre-processing.

Lemmatization: To reduce a word to its meaningful base form depending on the context, lemmatization was performed.

Hence, the line:

This desk is very study and it has a beautiful finish, I think that it is just a little pricey for the size.

Is reduced to:

desk study beautiful finish think little pricey size

ML Models:

For all the models, since the dataset does not provide the ground truth, I have used star ratings to calculate the ground truth. All reviews which have star rating <= 2 are assigned sentiment 0 (negative review) and all reviews with star rating > 2 are assigned sentiment 1 (positive review).

Bag of Words model is used to get the Count Vector. The count vector is an input vector which consists of the frequency/count of the word occurring for a particular review body. It does not remember the textual sequence and also does not maintain any context. It is simply the count statistics of the word. Hence for the first iteration, this model was considered. Each of the count vectors is added to the dataframe as an 'x_vector' column. The data frame is then split into test and training data. The reason for doing this is that, using a separate CountVectorizer() function for test and training data yields different lengths of input vector. Running ML Models on separate length x_vector columns in test and train data sets resulted in errors. For the Furniture dataset, test and train dataset is the same throughout for all models.

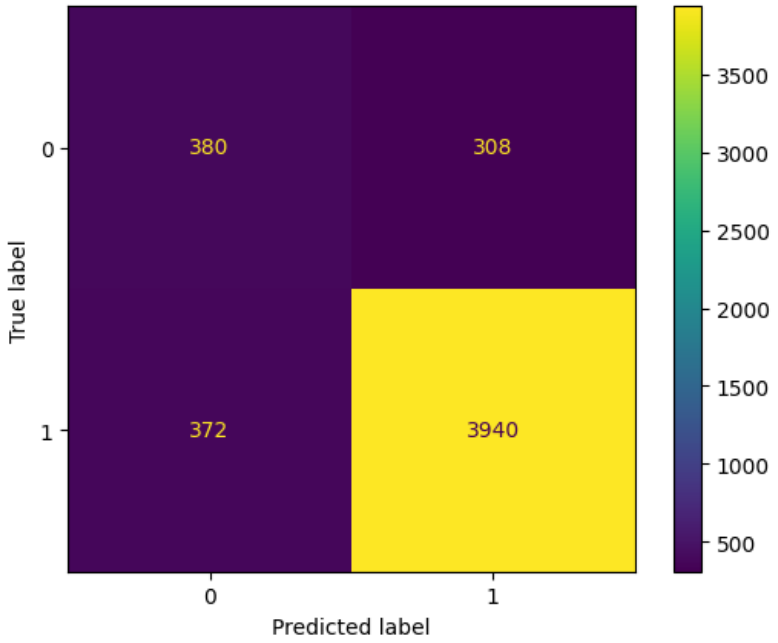
Logistic Regression

Since Logistic Regression is best used for prediction when the dataset is categorical, we decide this as one of our classical ML models. The accuracy for furniture data using Logistic regression is: 91.88%. The model had a convergence error due to max iterations limit. Increasing the max_iterations to 200 resolved the convergence error.

Decision Tree

Decision Tree is a supervised model which can also be used for classification. The model predicts the value of the target by learning simple decision rules inferred from the dataset. Since the total dataset has 20,000 rows and test dataset is 25%, it has 5,000 rows. Accuracy is 86.86%. The classification report for decision tree is as shown below:

	precision	recall	f1-score	support
0	0.51	0.55	0.53	688
1	0.93	0.91	0.92	4312
accuracy			0.86	5000
macro avg	0.72	0.73	0.72	5000
weighted avg	0.87	0.86	0.87	5000

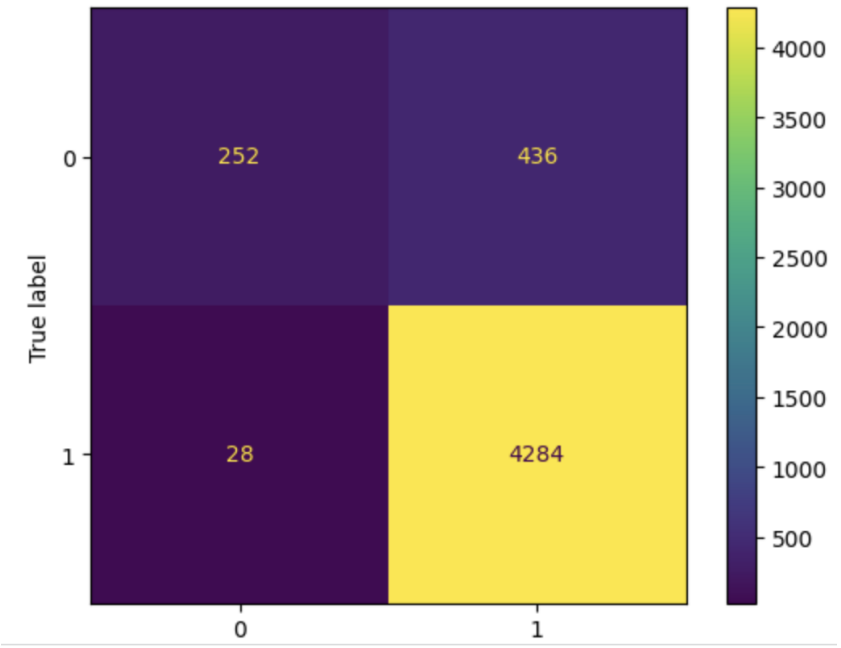


Support Vector Machine

The third model I chose is Support Vector Machine. SVM is a supervised ML model which can be used for classification/regression. SVM performs classification by finding the hyperplane that differentiates the data plotted on an n-dimensional space. The accuracy for SVM using Bag of words model is 90.72%.

The classification report and confusion matrix for SVM are shown below.

	precision	recall	f1-score	support
0	0.90	0.37	0.52	688
1	0.91	0.99	0.95	4312
accuracy			0.91	5000
macro avg	0.90	0.68	0.73	5000
weighted avg	0.91	0.91	0.89	5000



Since SVM gave the best results among the other two models, I decided to perform iterations on SVM.

Iterations

1. **TF-IDF - unigrams**

TF IDF for a word in text is obtained by multiplying 2 metrics:

Term Frequency and Inverse Document Frequency across a set of documents.

Term Frequency is the raw count/occurences of the word in the document/review.

Document frequency is the number of documents in which the word occurs. Hence, inverse document frequency is an indication of how rare/common a word is in the entire corpus. If the inverse document frequency is close to 0, it has a high occurrence across all documents and hence is more common. Multiplying both the numbers gives the TF IDF score of the word. The higher the TF IDF score, the more information a word carries about the document.

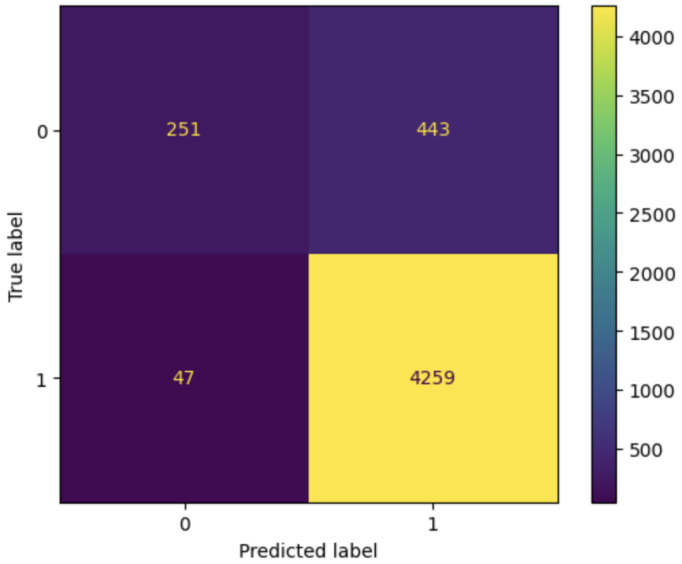
For the first variation, I calculated the TF IDF score of all unigrams.

The accuracy improved from 90.2% to 91.3%. This is because the TF IDF vector now carries more information about words important in the

document/review body than a simple count vector. This can help in better classification of sentiment.

```
print(classification_report(y_test, y_pred_svc_tfidf))
```

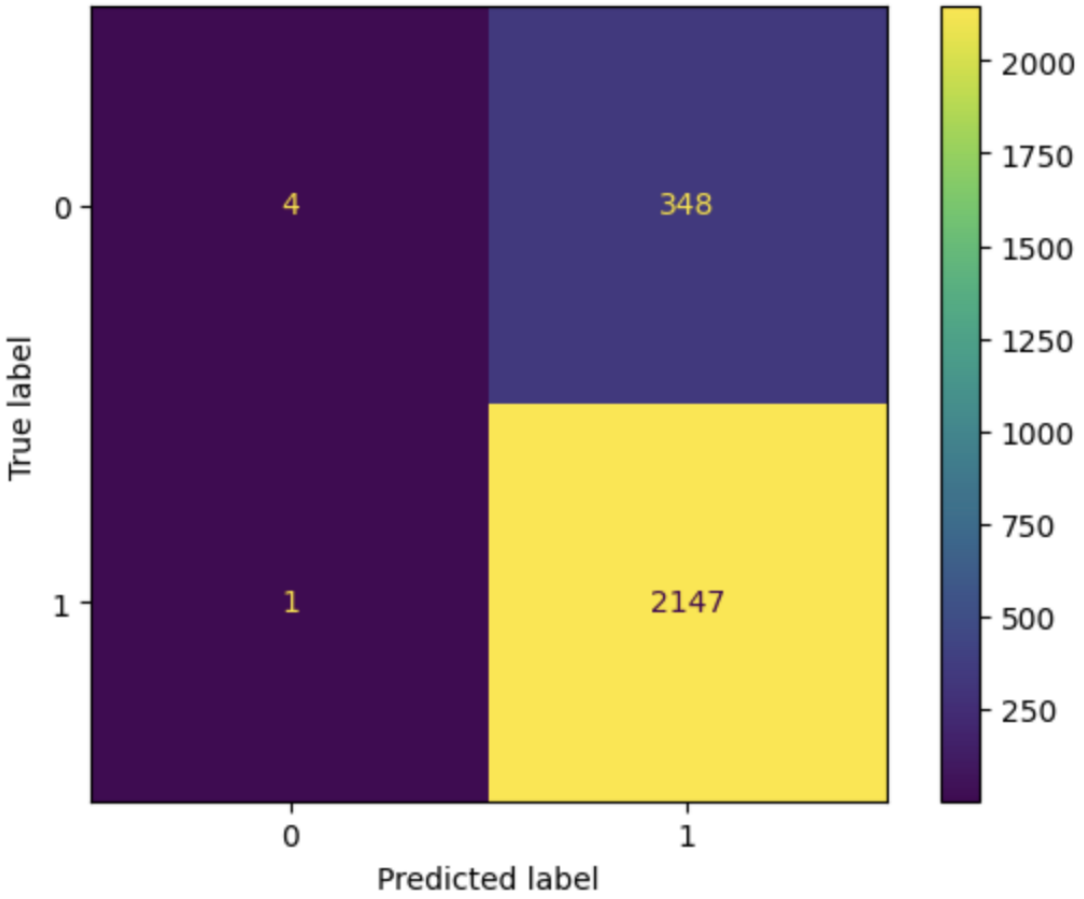
```
              precision    recall  f1-score   support

           0       0.90      0.45      0.60       727
           1       0.91      0.99      0.95      4273

    accuracy                           0.91      5000
   macro avg       0.91      0.72      0.78      5000
weighted avg       0.91      0.91      0.90      5000
```



2. TF-IDF : Bigrams

For the second variation, I used Bigrams. Since unigrams only capture what words occur in a document and are not able to capture the context in which a word appears, having context (surrounding words) would better represent the sentiment of the review. For example, “product good” vs “product not good” (review post processing) changes the sentiment of the review entirely. For this purpose, in the next iteration, I decided to investigate the performance of the model when extracting bigrams instead of unigrams. The model performance is discussed below:

	precision	recall	f1-score	support
0	0.80	0.01	0.02	352
1	0.86	1.00	0.92	2148
accuracy			0.86	2500
macro avg	0.83	0.51	0.47	2500
weighted avg	0.85	0.86	0.80	2500



BOOKS DATASET:

It has 15 features and a total 100000.To commence the implementation process, the required libraries were imported, including the Natural Language Toolkit (NLTK), a well-known open-source Python library for Natural Language Processing (NLP). NLTK offers a user-friendly interface for several NLP tasks, like tokenization, stemming, lemmatization, parsing, and sentiment analysis. The NLTK's vast corpus collection, comprising textual data from diverse sources such as books, news articles, and social media platforms, is one of its significant benefits. This offers a comprehensive dataset for model training and testing.

The 15 features of the book dataset are, given below:

marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes	total_votes	vine	verified_purchase	review_headline	review_body	review_date

The variable type of the each features are given below:

Variable types

Categorical	9
Numeric	4
Boolean	2

I have used a star_rating to create a sentiment analysis feature based on the rating that is

Positive if rating >= 4

Neutral , if rating == 3

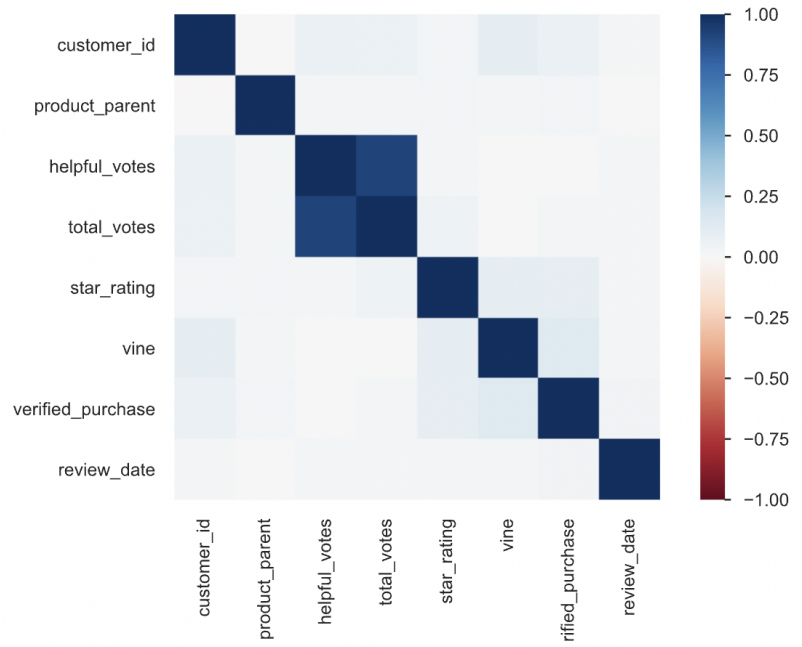
Negative if rating < 3

The pandas profiling of the rating for 20000 rows is given below:

star_rating
Categorical

Distinct	5
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	1.1 MiB

Sentiment feature is used in the model as the output and I have tried to preprocess the review body to numerical value and sample correlation of all the headers in the dataset is given below:



PREPROCESSING:

Review body in the tsv dataset is text data, I need to convert to numerical data and I have done following preprocessing on the review body data header, before training them on any model:

1. **Tokenization:** The input text is tokenized into individual words or tokens using the word_tokenize() method from the nltk.tokenize module. Tokenization splits the text into meaningful units that can be processed further.
2. **Stop words removal:** Stop words are words that are commonly used in a language but do not carry much meaning. These words are often removed from the text to focus on the more important content. In this step, the stopwords.words('english') method from the nltk.corpus module is used to obtain a set of English stop words. The set of stop words is then removed from the list of tokens using a list comprehension.
3. **Stemming:** Stemming is the process of reducing words to their root form. The PorterStemmer() from the nltk.stem module is used to perform stemming on the filtered tokens. The stemmer reduces words to their base or root form, which helps to reduce the dimensionality of the data and improve the performance of text-based models.

4. **Joining tokens:** Finally, the stemmed tokens are joined back into a string using the join() method and returned as the preprocessed text. The output is a string of cleaned and processed text that can be used for further analysis or modelling

After following step has be done on the data, I have done vectorization on the data to convert it from the Text data to numerical data and The process, I have tried to uses is:

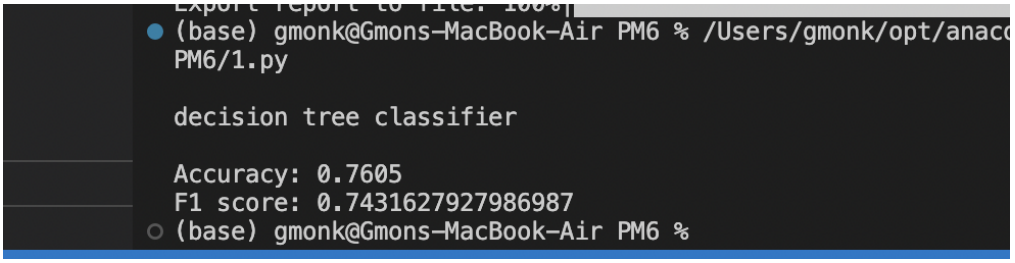
1. **CountVectorizer():** The first transformer in the pipeline is CountVectorizer(). It converts a collection of text documents into a matrix of token counts. In other words, it takes the text data and creates a matrix of how many times each word appears in each document. It does this by tokenizing the text (breaking it into individual words), converting each word to lowercase, and removing stop words.
2. **TfidfTransformer():** The second transformer in the pipeline is TfidfTransformer(). TF-IDF stands for Term Frequency-Inverse Document Frequency, which is a measure of the importance of each word in a document. This transformer takes the count matrix created by the CountVectorizer() and applies the TF-IDF formula to it. The TF-IDF formula assigns a weight to each word in each document based on how frequently it appears in the document and how rare it is across all documents. The result is a matrix where each cell represents the importance of a word in a particular document.

By combining these transformers in a pipeline, the preprocessor object can be used to preprocess text data in a single step. This preprocessed data can then be fed into machine learning models for further analysis.

Machine Learning Models:

1.DECISION TREES:

The decision tree is a frequently used algorithm for supervised machine learning that can handle both classification and regression tasks. It has a hierarchical structure resembling a tree, with internal nodes representing tests on features or attributes, branches indicating test outcomes, and leaf nodes containing class labels or numeric predictions. In this study, a bag-of-words vectorizer was produced using CountVectorizer, and the resulting model achieved an accuracy rate of 76%. The findings of the study are presented below.

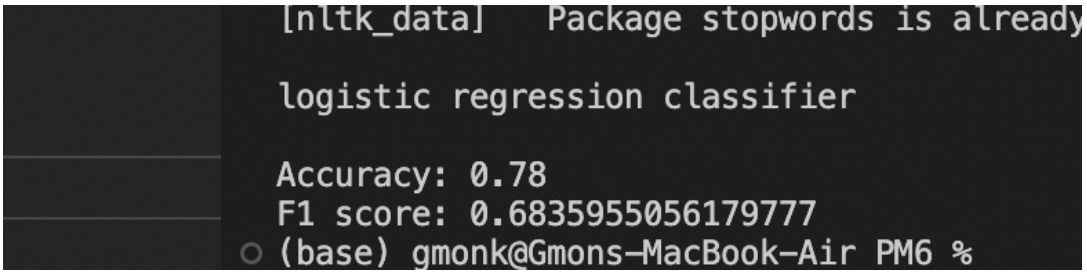


PS: python program to run this is decision_tree_book.py

2.**LOGISTIC REGRESSION**

Logistic regression is a widely used statistical and machine learning method that specialises in binary classification tasks. Its objective is to make predictions about a binary outcome based on a given set of input features. The technique employs a generalised linear model that models the probability of the binary outcome as a function of the input features.

In this particular project, the dataset was split into training and testing sets. A CountVectorizer was utilised to create a bag-of-words representation of the text data, which converts it into a sparse matrix of word frequencies. The training data was fitted and transformed using the vectorizer to generate a bag-of-words representation of the reviews. Similarly, the testing data was transformed using the same vectorizer. The logistic regression model was then trained using the bag-of-words representation of the training data. The model was used to make predictions on the testing data using the predict function. The classification report revealed an overall accuracy of 78%.

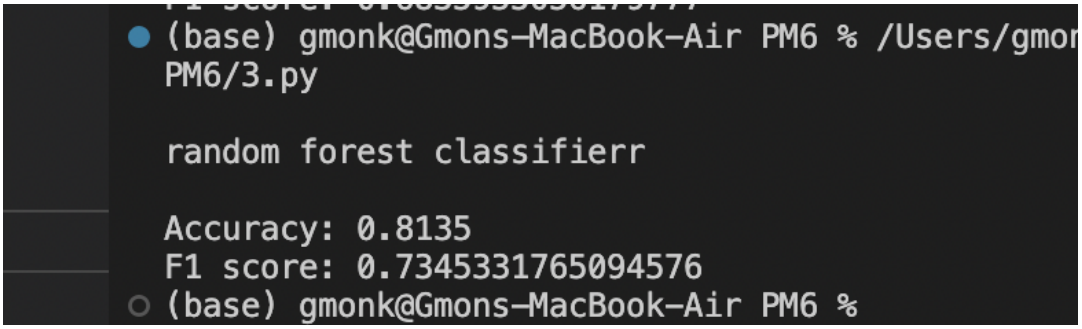


PS: python program to run this is ‘logistic_regression_book.py’

3.RANDOM FOREST CLASSIFIER:

The Random Forest is a machine learning algorithm that operates under the supervised learning paradigm, and it is applied for either classification or regression purposes by employing decision trees. Random Forest comprises a vast number of decision trees that act collectively as an ensemble. Each tree predicts a class, and

the class with the most votes becomes the final prediction of the model. The algorithm generates a collection of decision trees based on a randomly chosen subset of the training set. Essentially, it constructs a group of decision trees using a randomly selected subset of the training set and combines the votes from different trees to determine the ultimate prediction.
To train the model, the dataset was split into training and testing sets using the train_test_split function, and then the Random Forest Classifier model was applied. The overall accuracy achieved was 81%. The classification report displays the following results:

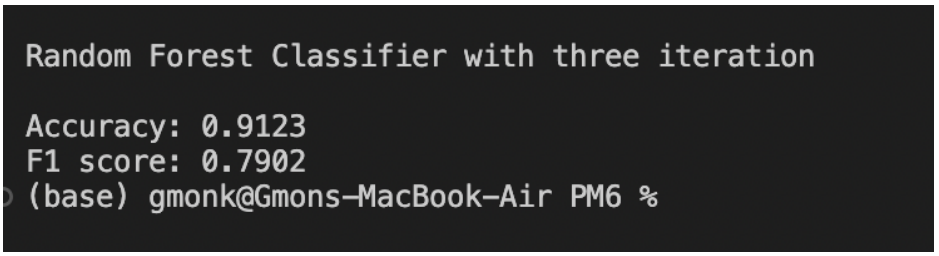


PS: python program to run this is ‘random_forest_book.py’

ITERATIONS:

I have made the following changes to the Random forest model, because it is performing better than other models. I was able to achieve an Accuracy of 91%. The changes that i have made are

1. Used the TfidfVectorizer instead of the CountVectorizer to generate TF-IDF features instead of simple count-based features.
2. Used n-grams of size 1 and 2 instead of single words as features.
3. Performed grid search over the hyperparameters of the Random Forest classifier to find the best combination of hyperparameters.



PS: Python program to run this is ‘iteration.py’

SUMMARY:

We are trying to predict the sentiment analysis of the Amazon review data.This collection of data contains evaluations of items sold on Amazon's marketplace spanning from 1995 to 2015. With over 130 million reviews written in various languages, it includes feedback for numerous categories such as Kitchen appliances, Books, Mobile Electronics, Furniture, Software, Tools, and Sports. The focus of our analysis was on reviews for Books, Mobile Electronics, and Furniture, and the dataset was presented in a Tab Separated Value format (TSV). It contains 15 attributes and a total of 996,557 data points, which consist of 100,000 reviews for Books, 104,855 reviews for Mobile Electronics, and 791,702 reviews for Furniture.

To begin with, the mobile electronics dataset was subjected to three different methods - logistic regression, decision tree, and random forest. Among these methods, logistic regression showed better performance with a 91% accuracy rate. For the decision tree model, we conducted two variations - TF-IDF and Grid search CV - resulting in an overall increase in accuracy from 85% to 87%.

Moving on to the book dataset, we utilised logistic regression, decision tree, and random forest methods. Random forest outperformed the other methods, achieving an accuracy rate of 81%. Further, we implemented an iterative process using the same random forest model, incorporating changes to the vectorization technique and adding a cv search. Additionally, we used n-grams of size 1 and 2 instead of individual words as features. These modifications resulted in a significant improvement in the model's accuracy, increasing it from 81% to an impressive 91.23%.

Regarding the furniture dataset, we employed three different methods - logistic regression, decision tree, and support vector machine. Logistic regression performed the best with an accuracy rate of around 91.88%, followed by the support vector machine. As part of our iterative approach, we first applied TF-IDF unigram, which resulted in an accuracy improvement from 90.2% to 91.3% for the support vector machine. For the second variation, we utilised the TF-IDF diagram technique.

Although we initially planned to implement a recommendation system using our model, time constraints compelled us to focus on developing a sentiment analysis model for user review datasets. Nonetheless, we are proud of the work we have done and the knowledge we have gained in the process. As a future extension, we plan to implement a product recommendation system based on the ML models implemented for sentiment analysis.

REFERENCES:

1. https://www.datacamp.com/tutorial/text-analytics-beginners-nltk
2. https://www.techtarget.com/searchbusinessanalytics/definition/opinion-mining-sentiment-mining#:~:text=Sentiment%20analysis%2C%20also%20referred%20to,a%20product%2C%20service%20or%20idea.
3. https://www.kaggle.com/datasets/bittlingmayer/amazonreviews
4. https://www.mygreatlearning.com/blog/bag-of-words/
5. https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a

ACKNOWLEDGEMENT

I, Mahvash Maghrabi, and my group partners Sruti Munukutla, Gmon Kuzhiyanikkal would like to express our gratitude and appreciation to all those who gave us the opportunity to complete this project and the report. We extend our special thanks to Professor Bruce Maxwell who provided us the opportunity to work on this project and always gave us stimulating suggestions and encouragement during the lectures. We would also like to express our gratitude to our Teaching Assistants, Srishti Hedge and Yiming Ge, who were always available to resolve our doubts and queries. Lastly, we would like to thank each other as we all worked together and supported each other throughout the project.