Amazon Reviews

DotPy

Table of contents

01
Introduction

- Business case
- Problem definition
- Objectives

Business case

1. Current challenges

- Missed Insights
- Uncertain Recommendations
- Underutilized Feedback for Improvement

Business case

2. Value of the Solution

- Improve Customer Experience And Satisfaction
- Enhance Product Quality
- Optimize Resources
- Drive Sales and Marketing

Return on Investment (ROI): Automating review analysis enables businesses to make quicker, data-driven decisions, enhancing revenue, brand reputation, and customer retention.

Business case

3. Target Audience

- Product Development Teams
- Customer Service Teams
- Marketing Teams

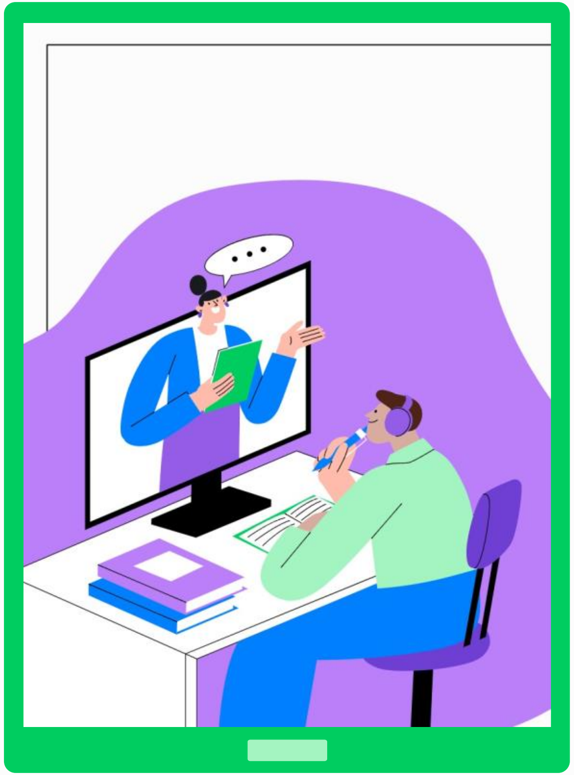
Problem Defintion

In the fast-paced world of e-commerce, platforms like Amazon rely on customer reviews to assess product quality and make key business decisions. However, with millions of reviews being posted daily, businesses face several challenges

- **Sentiment Analysis:**
- **Extracting Meaningful Insights**:
- **Predicting Recommendations:**
- **Leveraging Feedback for Improvement**

Objectives

- Improve product recommendations
- Understand common sentiment
- Enhance Product Reviews
- Detect Review Anomalies and Fake Reviews
- Predict Customer Recommendations
- Analyze Trends Over Time



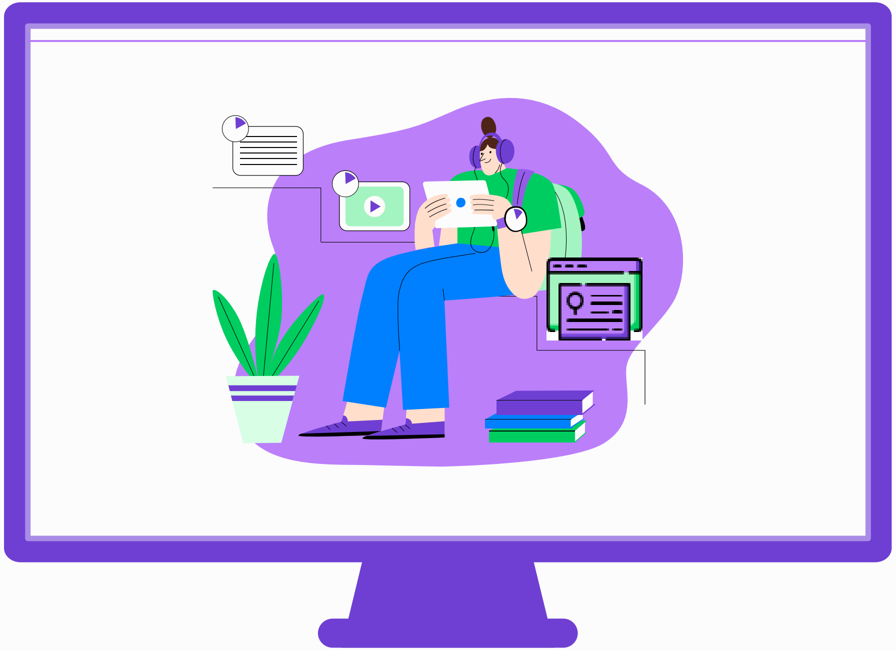
Dataset overview

```python
data.info()
```

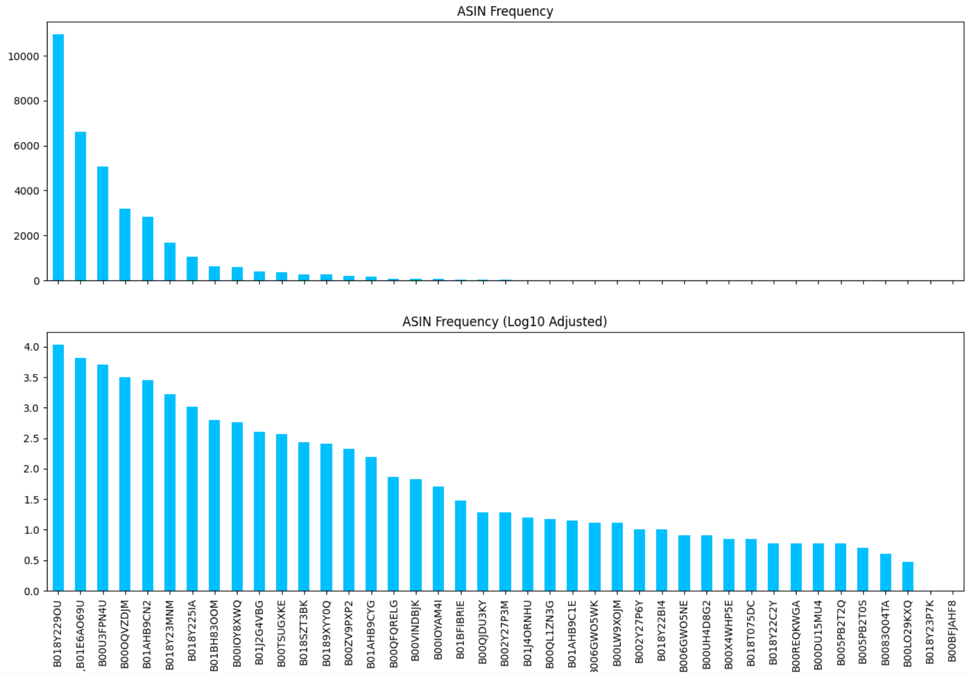
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34660 entries, 0 to 34659
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    34660 non-null  object 
 1   name                  27900 non-null  object 
 2   asins                 34658 non-null  object 
 3   brand                 34660 non-null  object 
 4   categories            34660 non-null  object 
 5   keys                  34660 non-null  object 
 6   manufacturer          34660 non-null  object 
 7   reviews.date          34621 non-null  object 
 8   reviews.dateAdded     24039 non-null  object 
 9   reviews.dateSeen      34660 non-null  object 
 10  reviews.didPurchase   1 non-null      object 
 11  reviews.doRecommend   34066 non-null  object 
 12  reviews.id            1 non-null      float64
 13  reviews.numHelpful    34131 non-null  float64
 14  reviews.rating        34627 non-null  float64
 15  reviews.sourceURLs    34660 non-null  object 
 16  reviews.text          34659 non-null  object 
 17  reviews.title         34654 non-null  object 
 18  reviews.userCity      0 non-null      float64
 19  reviews.userProvince  0 non-null      float64
 20  reviews.username      34653 non-null  object 
dtypes: float64(5), object(16)
memory usage: 5.6+ MB
```

summary of a DataFrame’s structure and information

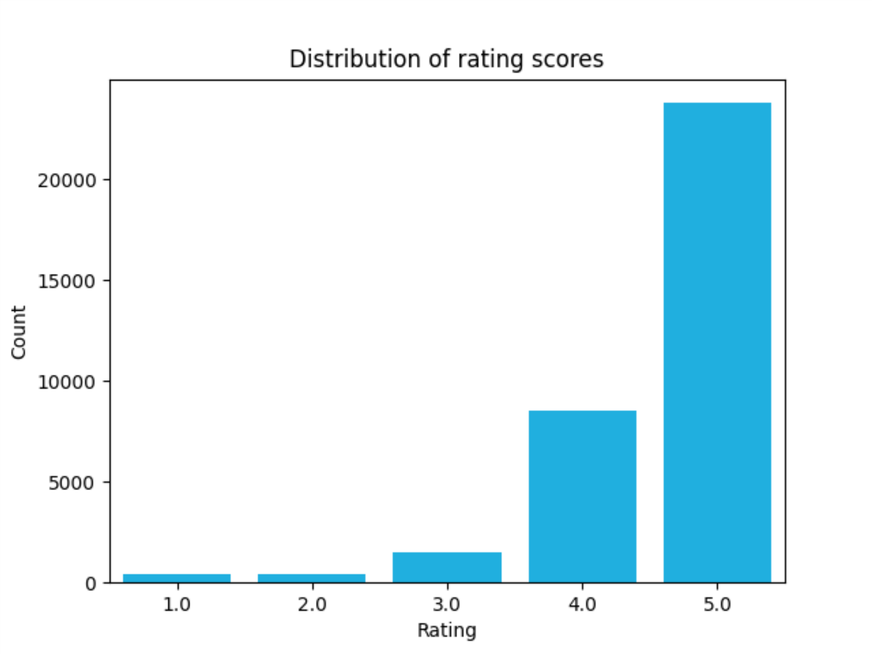
Analysis &Insights



The best products

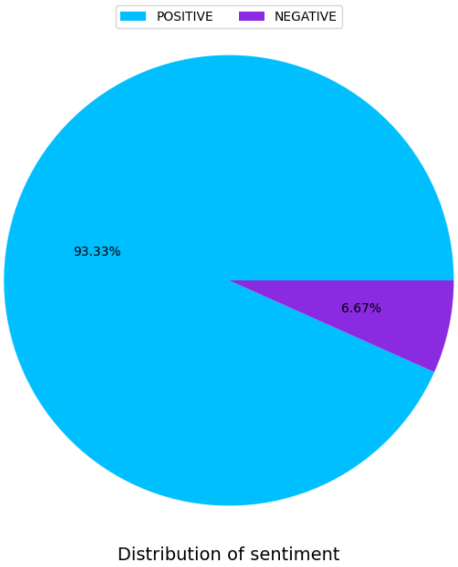


Rating of reviews

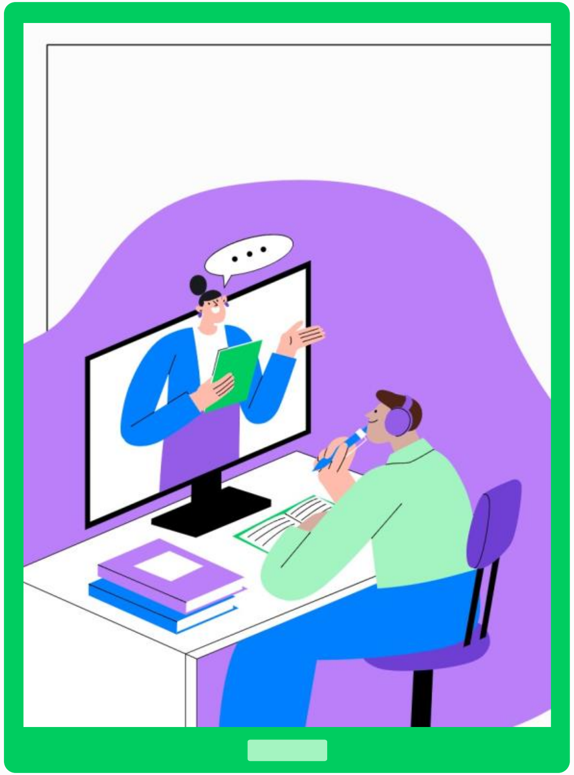


Distribution of sentiment

- Positive sentiment= 93.33%
- Negative sentiment=6.47%



Most used words in all reviews



Methodology

Feature Selection

We selected the most significant features that we are going to train our model with which are:

- Review Title
- Review Text
- Review Recommendation
- Output: How many stars

Data Preprocessing

- There were a lot of null values in the recommendation column which needed to be considered.

- We decided to fill in the null values and the empty indices with realistic values and drop the rows that have a lot of unknown data

- This way we took care of all the null and empty values in the rating and recommendation features

```
[46] data_4.isnull().sum()
```

	0
reviews.text	1
reviews.title	6
reviews.rating	33
reviews.doRecommend	594

dtype: int64

Data Preprocessing

```
47] rows_to_drop = []

    for index, row in data_4.iterrows():
        if pd.isnull(row['reviews.doRecommend']):
            if row['reviews.rating'] > 3:
                data_4.at[index, 'reviews.doRecommend'] = True
            elif row['reviews.rating'] < 3:
                data_4.at[index, 'reviews.doRecommend'] = False
            elif row['reviews.rating'] == 3:
                rows_to_drop.append(index)
        if pd.isnull(row['reviews.doRecommend']) and pd.isnull(row['reviews.rating']):
            rows_to_drop.append(index)

    data_4.drop(rows_to_drop, inplace=True)
```

NLP

Steps:

1) Remove any non-alphabetical character
2) Convert text to lowercase
3) Tokenize the text (split it to separate words)
4) Remove stop words
5) Stem the words
6) Pass the clean words to the count vectorizer to create a matrix of the unique words
7) Pass the data from the count vectorizer to the model

NLP

```
[28] from nltk.stem import PorterStemmer
     from nltk.corpus import stopwords
     from nltk.tokenize import word_tokenize
     import re

     ps = PorterStemmer()
     stop_words = set(stopwords.words("english"))

     reviews = []
     recommendations = []
     for i in range(len(data_4)):
         title = data_4['reviews.title'].iloc[i]
         text = data_4['reviews.text'].iloc[i]
         recommend = data_4['reviews.doRecommend'].iloc[i]

         if not isinstance(title, str):
             title = ""
         if not isinstance(text, str):
             text = ""

         combined_text = f"{title} {text}"

         keep_alphabet_only = re.sub('[^a-zA-Z]', ' ', combined_text)  # Remove non-alphabetical characters
         lowrecase_text = keep_alphabet_only.lower()  # Convert to lowercase
         tokenized_text = word_tokenize(lowrecase_text)  # Tokenize the text
         remove_stopwords = [ps.stem(word) for word in tokenized_text if word not in stop_words]  # Remove stopwords and stem words
         cleaned_text = ' '.join(remove_stopwords)  # Join tokens back into a single string

         reviews.append(cleaned_text)
         recommendations.append(int(recommend))
     reviews
```

Training data on XGBoost

```
[32] x_train , x_test , y_train , y_test = train_test_split(x,y,test_size = 0.15, random_state = 2)

[33] model = XGBClassifier(learning_rate = 0.2 , n_estimators = 200)

[34] model.fit(x_train,y_train)
     model_accuracy = model.score(x_test , y_test)
     print(f"Test Accuracy: {model_accuracy * 100:.2f}%")

     Test Accuracy: 73.80%
```

Training data on Naive

```
train_accuracy_naive= naive.score(x_train, y_train)
print(f"Train_Accuracy: {train_accuracy_naive * 100:.2f}%")
```

```
Train_Accuracy: 76.05%
```

```
[ ] test_accuracy_naive= naive.score(x_test, y_test)

print(f"Train_Accuracy: {test_accuracy_naive * 100:.2f}%")
```

```
Train_Accuracy: 70.94%
```

Testing

Testing model output

```python
title = input("Enter your review title: ")
text = input("Enter your review: ")
recommend = input("Do you recommend this product? (yes/no): ").strip().lower()

recommend = 1 if recommend == "yes" else 0

combined_input = f"{title} {text}"
l1 = []
clean = re.sub('[^a-zA-Z]',' ',combined_input)
clean = clean.lower()
clean = word_tokenize(clean)
clean = [ps.stem(word) for word in clean if word not in stop_words ]
clean = ' '.join(clean)
new_review_text = cv.transform([clean])
new_review_recommend = np.array([recommend]).reshape(1, -1)
new_review = hstack([new_review_text, new_review_recommend])
res = model.predict(new_review)

print(f"Predicted Review Rating: {int(res[0] + 1)}")
```

```
Enter your review title: Perfect
Enter your review: Very good product, easy to use and I really liked it
Do you recommend this product? (yes/no): yes
Predicted Review Rating: 5
```

Different examples

```
Enter your review title: Problem with screen
Enter your review: The tablet's quality is high and have a very good performance, but the screen was too small it feels hard to read on it
Do you recommend this product? (yes/no): yes
Predicted Review Rating: 4
```

```
Enter your review title: An average tablet
Enter your review: Got it for a cheap price but it has a very bad camera and dim screen resolution. Could be good for other people but not me
Do you recommend this product? (yes/no): yes
Predicted Review Rating: 3
```

```
Enter your review title: Slow processing
Enter your review: The tablet's camera and resolution is fine but some times it gets too slow and laggy especially when browsing the web
Do you recommend this product? (yes/no): no
Predicted Review Rating: 2
```

```
Enter your review title: Disappointing
Enter your review: Very bad product, very hard to use, disappointing and also very slow
Do you recommend this product? (yes/no): no
Predicted Review Rating: 1
```

Our team

Mohamed Emad

Ahmed Okasha

Thanks!

Do you have any questions?