
From Raw Data to Informed Decisions: Analyzing Amazon Book Reviews

Alberti A. Ligari D. Andreoli A.¹

¹ *Data Science and Big data Analytics course, University of Pavia, Department of Computer Engineering (Data Science), Pavia, Italy*

Github page: <https://github.com/DavideLigari01/data-science-project>

Date: September 18, 2023

Abstract —prova

Keywords —Big Data • Hadoop • Spark • ML • MongoDB • Data Analysis • Data Visualization • Python

CONTENTS

1 Introduction

2 Discovery

3 Data Preparation

4 Local Hypotheses Testing

5 Spark Hypotheses Testing

6 Helpfulness Prediction

7 Conclusions

8 Python Script

1. INTRODUCTION

In the age of digital commerce, customer reviews profoundly impact product perception and purchase decisions. Amazon, with its extensive repository of book reviews spanning nearly two decades, holds a wealth of valuable insights, sentiments, and trends. This project aims to create a scalable solution for uncovering patterns, sentiment trends, and correlations within the realm of book reviews, utilizing advanced tools and technologies like Hadoop, Spark, MongoDB, and Python libraries such as Pandas and Scikit-learn. In this report, we provide a detailed exploration of our project, covering stages from initial data discovery and preparation to feature extraction, model building, and evaluation.

2. DISCOVERY

Team

1

Tools

1

Framing

3. DATA PREPARATION

1

Data Collection

Hypothesis Generation

2

Data Cleaning

4

Data Aggregation

4

The MapReduce job performs an inner join operation between the "Data table" and the "Rating table" based on the book title, resulting in a single file containing the joined records from both tables.

4

Mapper

4

The Mapper script processes input data line by line, converting each line into a key-value structure. The key represents the book title, and the value contains the remaining line content. To distinguish between records from the 'Data table' and 'Rating table' and ensure the correct processing order in the Reducer phase, the Mapper appends a special character ('-' for 'Data table' and 'www' for 'Rating table') as the second key element. This ensures that 'Data table' records are processed before 'Rating table' records during subsequent MapReduce phases.

Reducer

The Reducer script processes intermediate output records generated by the Mapper, aiming to join 'Data' and 'Rating' records for the same title. The Reducer reads records sequentially, storing 'Data' and 'Rating' information separately. When both 'Data' and 'Rating' records for the same

title are available, the Reducer performs the join operation by combining the data from these records.

MongoDB loading

4. LOCAL HYPOTHESES TESTING

Hypothesis 1

Reviews with longer text have higher helpfulness ratings.

Metric: Correlation coefficient (e.g., Pearson's correlation) between review length and helpfulness score.

Missing Values:

- 'review/text': remove the entire sample
- 'review/helpfulness': remove the entire sample

Data Transformation:

- 'review/text': Count the number of words in each review removing punctuation and stopwords.
- 'review/helpfulness': $helpfulness = \frac{x}{y} \sqrt{y}$

Description and Results The data cleaning and 'review/helpfulness' transformation process was performed using the 'pymongo' library to exploit the MongoDB efficiency. Specifically, we defined a pipeline to perform the needed operations. As regards the 'review/text' transformation, we used the 'nltk' library to tokenize the text, remove punctuation, stopwords and eventually count the number of words.

The correlation coefficient between the two variables is 0.3313 with a p-value < 0.05 , indicating a statistically significant correlation.

A graphical confirmation is provided by Figure 1. Indeed there is a positive correlation until around 400 words, after which the boxplot stabilizes. Thereby, we decided to analyze the correlation within the review length groups. As a results (Table 4) we got that the correlation is positive and statistically significant for the reviews with length between 0 and 400 words, while it becomes negative and statistically significant for the reviews with length greater than 750 words. As regards the reviews in the middle (i.e. between 400 and 750 words), the correlation is negligible.

Conclusion: The hypothesis is confirmed, but the correlation is not very strong and changes depending on the review length.

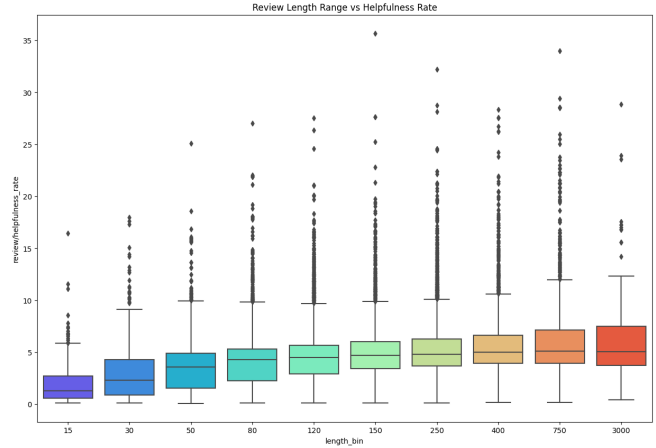


Fig. 1: Correlation between review length and helpfulness score for different review length groups

Table 1: Correlation Coefficients and P-values for Different Groups

Group Number	Correlation Coefficient	P-value
400	0.2216	0.0000
750	-0.0188	0.2585
3000	-0.1418	0.0065

Hypothesis 2

Hypothesis 3

Reviews with higher book rating have higher helpfulness ratings.

Metric: Correlation coefficient (e.g., Pearson's correlation) between review rating and helpfulness score.

Missing Values:

- 'review/score': remove the entire sample
- 'review/helpfulness': remove the entire sample

Data Transformation:

- 'review/score': no operations required
- 'review/helpfulness': $helpfulness = \frac{x}{y} \sqrt{y}$

Description and Results As in the previous hypothesis, we decided to deal with missing values and data transformation directly with a MongoDB query. With the data ready to be processed, we decided to perform a prior investigation on the distribution of votes on the 4 rating categories. As shown by Figure 2, there is a **positive bias** as people are more prone to vote a positive reviews rather than a negative one. Specifically, the large majority of votes for rating 5, were composed by a number of total votes just equal to 1. This introduces a bias in our results, since according to the formula used to compute the helpfulness score, such a small number of total votes would lead to a small helpfulness score. To avoid this we decided to retain only the reviews with a number of total votes greater than 20.

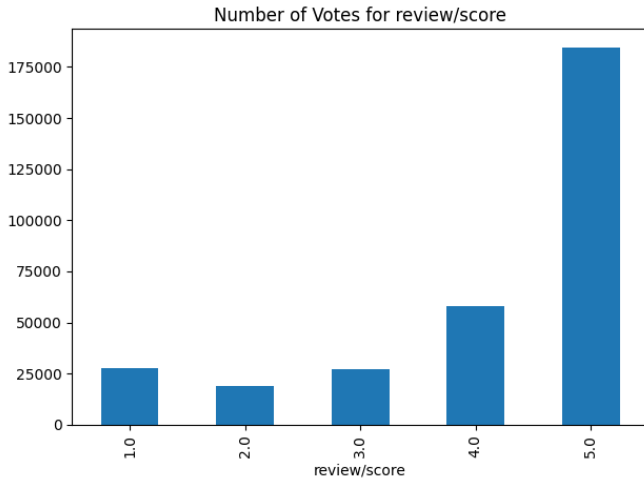


Fig. 2: Distribution of votes on the 4 rating categories

The Spearman correlation coefficient between the two variables is equal to 0.5247, with a p-value of 0.0.

Conclusion: The hypothesis is confirmed as there is a positive and statistically significant correlation between the two variables. This result is confirmed by the boxplot in Figure 3.

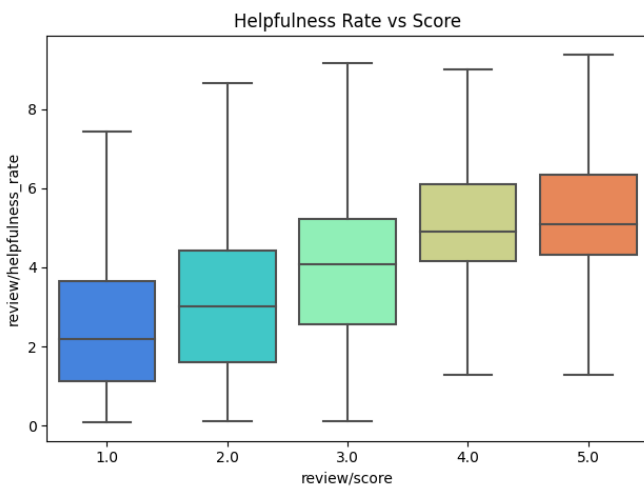


Fig. 3: Boxplot of the correlation between review rating and helpfulness score

Hypothesis 4

Hypothesis 4 explores the impact of individual users' unique personalities, personal preferences, and the potential for anonymous users to overrate books on rating scores. We tested this hypothesis by considering the rating score as the primary metric. The research team and any records with missing values were excluded from the analysis. The hypotheses under examination were as follows:

H0 (Null Hypothesis): The rating score is not influenced by the user's profileName. All rating scores are drawn from the same distribution, implying equal means and variances for each user's rating scores.

H1 (Alternative Hypothesis): The rating score is affected by the user, suggesting that each user's rating scores follow a distinct distribution.

For the sake of consistency, users with fewer than 20 reviews were excluded from the analysis, as a limited number of reviews cannot reliably estimate statistical measures.

The statistical test employed was ANOVA, which assesses differences in means between user groups. The results yielded an F-statistic of 1.5374 and a corresponding P-value of 0.0670. These results indicate that although there may be some variance in rating scores among different users, the evidence to reject the null hypothesis (H0) and conclude that user personalities significantly impact rating scores is not robust.

This conclusion is further supported by the accompanying boxplot (Figure 4), which illustrates variations in the distribution of rating scores across users.

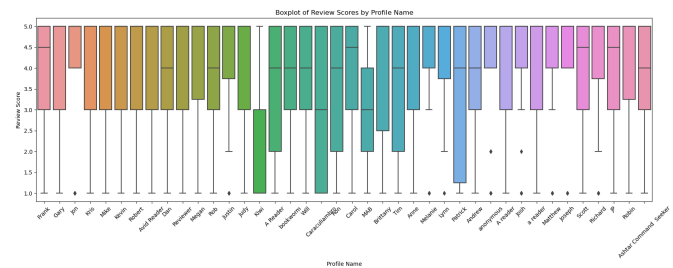


Fig. 4: Distribution of rating scores across users

Hypothesis 5

Hypothesis 5 examines the influence of book categories on review scores. To test this hypothesis, we considered the rating score as the metric and removed missing values. Two competing hypotheses were established:

H0 (Null Hypothesis): Rating scores are not related to the book categories, as all rating scores are drawn from the same distribution.

H1 (Alternative Hypothesis): Rating scores are affected by the book category, indicating that the rating scores of each category follow different distributions.

As in the previous hypothesis, categories with fewer than 20 reviews were omitted for consistency. An ANOVA (Analysis of Variance) test was conducted to assess the validity of these hypotheses. The results of the test revealed an F-statistic of 0.177 and a P-value of 0.999. A low F-statistic value and a P-value close to 1 suggest that there is not much variation between the means of different categories. Therefore, we could not reject the null hypothesis (H0) and concluded that book categories do not significantly impact rating scores. This result was further supported by the accompanying boxplot (Figure 5), which showed that the distribution of rating scores was similar across categories.

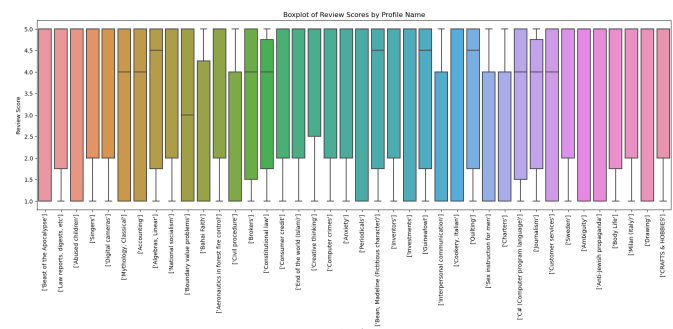


Fig. 5: Distribution of rating scores across categories

Hypothesis 6

- The larger the number of books published for a category, the higher the review score.
- The larger the number of books published by publishers, the higher the review score.

Metric: Correlation Coefficient.

Missing Values:

- 'publisher': remove the entire sample
- 'review/score': remove the entire sample
- 'categories': remove the entire sample

Data Transformation:

- 'categories': GroupBy categories.
- 'publisher': GroupBy publisher.
- 'review/score': Compute the average review/score for each publisher and category.

Description and Results All the data cleaning and transformation steps were performed using MongoDB with the aggregation pipeline, to have a more efficient and faster computation.

Specifically the data cleaning steps were performed using the *\$match* operator, while the data transformation steps were performed using the *\$group* operator with *\$avg* operation. Finally a *\$project* operator was used to select the fields of interest. To reduce bias, we removed the categories having less than 50 books and the publishers having less than 20 books. The results are shown in Table 2.

Conclusion: The hypotheses are falsified since the metrics shows no correlation between the two variables in both cases.

Table 2: Correlation Values and P-values for Categories and Publishers

Variable	Correlation Value	P-value
Category	-0.0806	0.558
Publisher	-0.0673	0.151

Curiosity We performed two complex MongoDB queries (reported as example in section 'Code') to answer to two questions:

- Which are the best publishers? (i.e. capable of getting avg scores above 4.5 in lots of categories)
- In which categories are the best publishers focused?

The results are reported in Figure 6 and Figure 7.

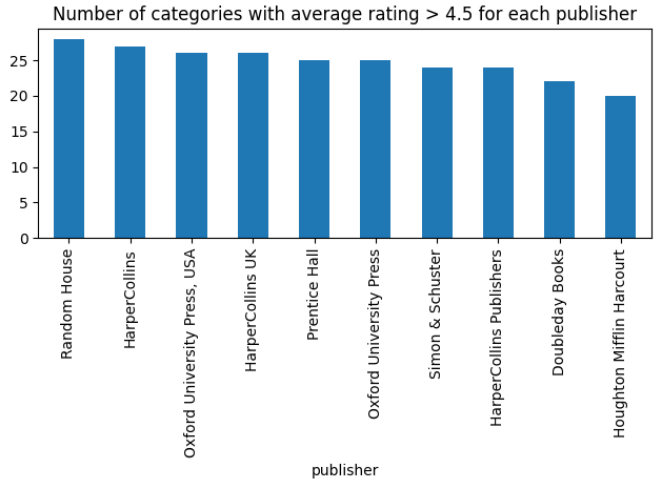


Fig. 6: Which are the best publishers?

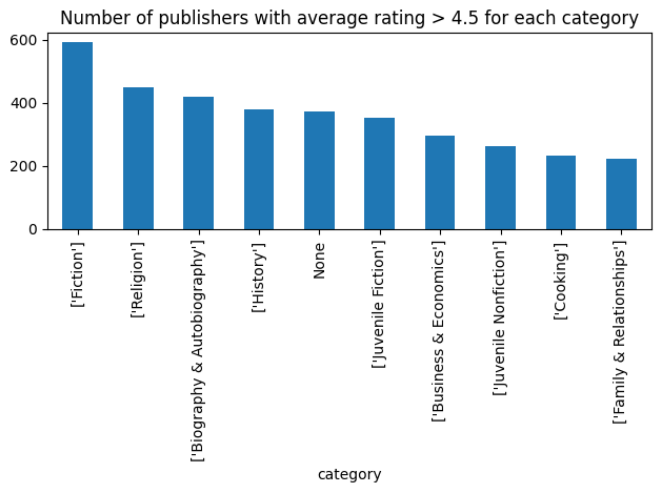


Fig. 7: In which categories are the best publishers focused?

5. SPARK HYPOTHESES TESTING

6. HELPFULNESS PREDICTION

7. CONCLUSIONS

8. PYTHON SCRIPT

This is the script to answer the question: In which categories are the best publishers focused?

```

1 # Deal with missing values
2 pipeline_missing = {'$match': {
3   'review/score': {'$exists': True, '$ne': 0},
4   'publisher': {'$exists': True, '$ne': None},
5   'categories': {'$exists': True},
6 }}
7 }
8
9 # Compute average rating for each tuple category,
  publisher
10 pipeline_average_rating = {'$group': {
11   '_id': {
12     'category': '$categories',
13     'publisher': '$publisher',
14   },
15   'avg_score': {'$avg': '$review/score'},
16   'count': {'$sum': 1}

```

```

17 }
18 }
19
20 # Show average rating for category for each
    publisher
21 pipeline_publisher = {'$group': {
22     '_id': '$_id.category',
23     'avg_score/publisher': {
24         '$push': {
25             'publisher': '$_id.publisher',
26             'avg_score': '$avg_score',
27             'count': '$count'
28         }
29     }
30 }
31 }
32
33 # Unwind the list of categories
34 pipeline_unwind = {'$unwind': '$avg_score/
    publisher'}
35
36 # Remove categories or publisher with less than '
    threshold' reviews
37 threshold = 0
38 pipeline_remove = {'$match': {
39     'avg_score/publisher.count': {'$gte':
        threshold}
40 }
41 }
42
43 # Count the number of categories with average
    rating > 4.5
44 pipeline_counts = {'$project': {
45     'category': '$_id',
46     '_id': 0,
47     'publisher': '$avg_score/publisher.publisher',
48     'count': {
49         '$sum': {
50             '$cond': {
51
52                 'if': {'$gt': ['$avg_score/
                    publisher.avg_score', 4.5]},
53                 'then': 1,
54                 'else': 0
55             }
56         }
57     }
58 }
59 }
60
61 # Sum the results for each publisher. If Total >
    10, then the hypothesis is False
62 pipeline_sum = {'$group': {
63     '_id': '$category',
64     'total': {'$sum': '$count'}
65 }
66 }
67
68 pipeline_sort = {'$sort': {
69     'total': -1
70 }
71 }
72
73 results = books.aggregate([pipeline_missing,
    pipeline_average_rating, pipeline_publisher,
74     pipeline_unwind,
        pipeline_remove,
        pipeline_counts,
75     pipeline_sum,
        pipeline_sort])

```