UNIVERSITÀ
DI PAVIA

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 19, 2023

Abstract —This report delves into the Amazon Books Review dataset using data science techniques. Our goal was to uncover insights, sentiments, and correlations within this extensive collection of reviews. Leveraging tools like Hadoop, Spark, MongoDB, and Python libraries, we explored factors influencing review helpfulness, including review length, sentiment, and ratings. We also ventured into helpfulness prediction with Word2Vec and machine learning, highlighting Random Forest as a standout model. User profiles and book categories provided valuable insights into user behavior and preferences. This report underscores the power of data science in understanding book reviews, emphasizing data-driven decision-making and future exploration of hidden patterns in data.

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

Contents

1. Introduction

In 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

To initiate our data science initiative, it was important to assemble our team, precisely define our project's objectives, and conduct a comprehensive assessment of the available tools.

Team

The team is composed by three members:
Andrea Alberti: github.com/AndreaAlberti07
Davide Ligari: github.com/DavideLigari01
Cristian Andreoli: github.com/CristianAndreoli94

Framing

The primary **objective** of this project is to craft a **scalable** solution for the comprehensive analysis of a dataset comprising Amazon book reviews. Ultimately, our aim is to construct a predictive model capable of assessing the helpfulness of a review based on its content.

Tools

The selection of our tools was driven by the overarching objective of crafting a scalable solution that can effectively operate within a **Big Data** environment.

- **Virtual Machine**: Employed to establish a controlled working environment.
- **Hadoop**: Utilized for the storage of data within a distributed file system and for executing MapReduce operations.
- **Spark**: Chosen as an enhanced alternative to MapReduce, facilitating operations on distributed datasets.

- **Python**: Adopted as the primary programming language due to its extensive library support.
- **MongoDB**: Implemented as a NoSQL database sandbox, ensuring secure handling of local data.
- **GitHub**: Employed for seamless project sharing and collaborative development.
- **LaTeX**: Utilized for the creation of the project report, ensuring professional and structured documentation.

3. Data Preparation

To commence our project, we initiated the process of data retrieval and preparation.

Data Retrieval and Preliminary Analysis

The selected dataset comprises two tables and approximately three million reviews, accessible at the following link: Amazon Books Reviews. After acquiring the dataset, we executed the following steps:

1. **HDFS Loading:** We loaded the data into HDFS using the following commands:

```
# Create HDFS directories
hdfs dfs -mkdir -p "$HDFS_PATH/ratings"
hdfs dfs -mkdir -p "$HDFS_PATH/books_info"

# Copy local files to HDFS
hdfs dfs -copyFromLocal "$LOCAL_PATH/ratings.csv"
    "$HDFS_PATH/ratings/"
hdfs dfs -copyFromLocal "$LOCAL_PATH/books_info.
    csv" "$HDFS_PATH/books_info/"
```

2. **Preliminary Analysis:** We utilized PySpark to gain a comprehensive understanding of the data. During this phase, we defined a schema for our data and computed essential statistics, including the percentage of missing values and unique values for each field in our dataset.

Hypothesis Generation

Following the preliminary analysis, we formulated several hypotheses for testing:

- **H1:** Reviews with longer text exhibit higher helpfulness ratings.
- **H2:** Reviews containing more positive sentiment words receive higher helpfulness ratings.
- **H3:** Reviews associated with higher book ratings correlate with higher helpfulness ratings.
- **H4:** Rating scores are influenced by individual users, potentially leading to overestimation or underestimation of a book's quality. Anonymous users may tend to overrate books.
- **H5:** The review score is influenced by the category of the book.
- **H6:** An increase in the number of books published within a category or by a particular publisher results in higher review scores.

Data Cleaning

In this phase, we meticulously cleaned the data, addressing duplicates, eliminating extraneous columns for our analysis, and removing any symbols that could potentially interfere with the reading of the CSV files. All cleaning operations were executed using PySpark.

Data Aggregation

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.

Mapper

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

Upon completion of all previous operations, the next step involved loading the data into MongoDB. The process included the following steps:

- Connect to MongoDB using the ‘pymongo‘ library.
- Establish a connection to HDFS and read the data using the ‘spark.read.csv‘ method.
- Select a subset of the Spark DataFrame for import, employing the ‘sample‘ method.
- Transform the data into a dictionary format using the ‘to_dict‘ method.
- Insert the transformed data into MongoDB using the ‘insert_many‘ method.

We imported both the ‘ratings‘ and ‘books_info‘ tables into MongoDB, along with the resultant joined table generated through the MapReduce process. These datasets were instrumental in conducting the local hypothesis testing described below.

4. Local Hypotheses Testing

Hypothesis 1

H0 (Null Hypothesis): There is a positive correlation between the length of a review and its helpfulness score.
The data cleaning and 'review/helpfulness' transformation process (helpfulness score $= \frac{x}{y}\sqrt{y}$) was executed using the

‘pymongo‘ library to leverage the efficiency of MongoDB. Specifically, we designed a pipeline to perform the necessary operations. Regarding the ’review/text’ transformation, we employed the ‘nltk‘ library to tokenize the text, remove punctuation, stopwords, and subsequently 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 representation confirming this correlation can be found in Figure 1. There is a positive correlation observed until approximately 400 words, beyond which the boxplot stabilizes. Consequently, we conducted an analysis of the correlation within specific review length groups. As a result (Table 1), we observed a positive and statistically significant correlation for reviews with lengths between 0 and 400 words. However, for reviews longer than 750 words, the correlation becomes negative and statistically significant. For reviews falling in the intermediate range (between 400 and 750 words), the correlation is negligible.
Conclusion: The hypothesis is confirmed, but the correlation is not very strong and varies depending on the length of the review.

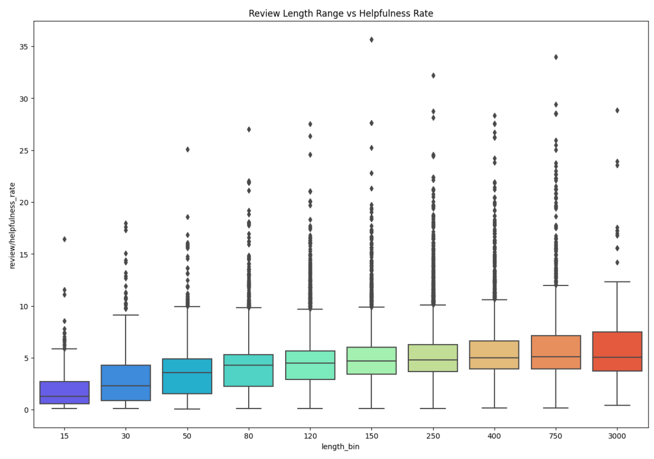

Fig. 1: Correlation between review length and helpfulness score for different review length groups

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

H0 (Null Hypothesis): There is no correlation between the rating of a review and its helpfulness score. Similar to the previous hypothesis, we addressed missing values and data transformations directly with a MongoDB query. With the data prepared for analysis, we conducted an initial examination of the distribution of votes across the four rating categories. Figure 2 reveals a **positive bias** wherein individuals tend to vote more for positive reviews than negative ones. Specifically, a significant portion of votes for rating 5 consists of reviews with a total vote count equal to 1. This introduces bias into our results because, based on the formula used to compute the helpfulness score, a small total vote count would lead to a low helpfulness score. To mitigate this, we retained only reviews with a total vote count greater than 20.

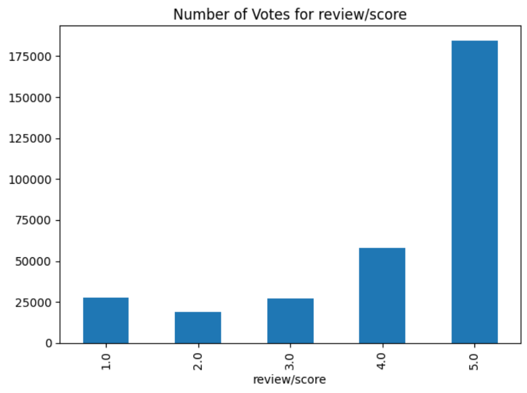

Fig. 2: Distribution of votes across the four rating categories

The Spearman correlation coefficient between the two variables is 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 review rating and helpfulness score. This finding is further supported by the boxplot in Figure 3.

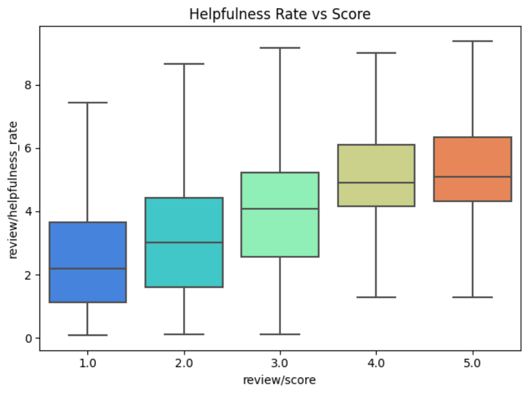

Fig. 3: Boxplot illustrating the correlation between review rating and helpfulness score

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.

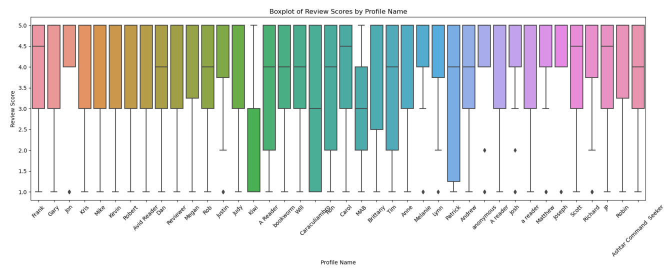

Fig. 4: Distribution of rating scores across users

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.

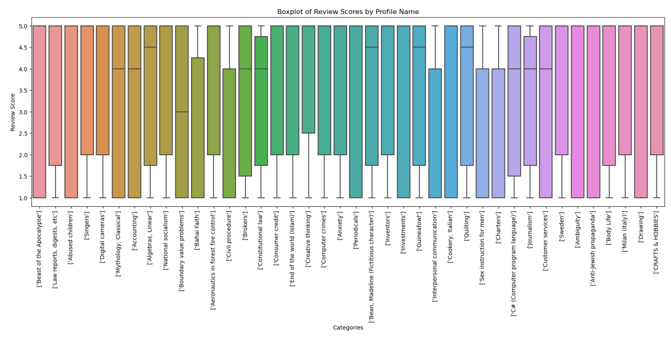

Fig. 5: Distribution of rating scores across categories

Hypothesis 6

H0 (Null Hypothesis): There is no correlation between the number of books published for a category (or publisher) and the review score.
All data cleaning and transformation steps were executed using MongoDB's aggregation pipeline to ensure efficient and rapid computation. Specifically, data cleaning involved the use of the *$match* operator, while data transformation steps were carried out with the *$group* operator, utilizing the *$avg* operation. Finally, the *$project* operator was applied to select the relevant fields. To reduce bias, we excluded categories with fewer than 50 books and publishers with fewer than 20 books. The results are presented in Table 2.
Conclusion: The hypotheses are rejected as the metrics reveal no significant correlation between the number of books published for a category (or publisher) and the review score 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 executed two complex MongoDB queries to answer two intriguing questions:

- **Which are the best publishers?** (i.e., those capable of achieving average scores above 4.5 in multiple categories)
- **In which categories are the best publishers focused?**

The results of these queries are presented in Figure 6 and Figure 7.

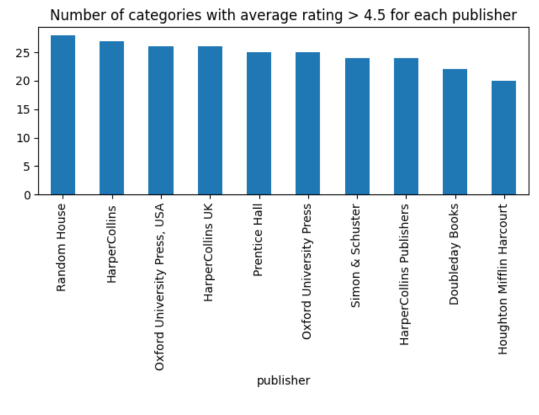

Fig. 6: Identifying the Best Publishers

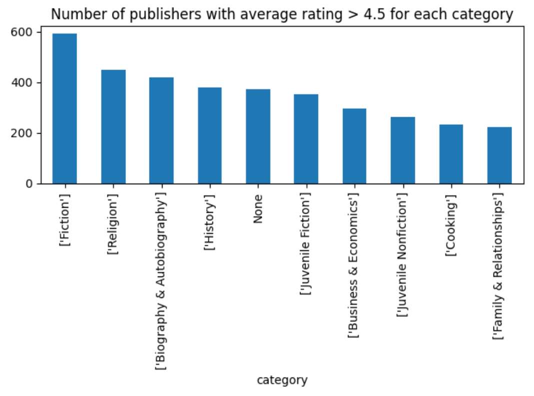

Fig. 7: Categories Favored by the Best Publishers

5. Spark Hypotheses Testing

To showcase the feasibility of implementing data analysis within a Big Data context, we opted to replicate some hypothesis testing using Spark, focusing particularly on Hypotheses 1 and 3.

Hypothesis 1

Addressing this hypothesis involved several key steps:

- **Compute the Helpfulness Score:** This was straightforwardly achieved by leveraging the 'WithColumn' method of the Spark DataFrame, creating a new column with the updated values.
- **Compute the Text Length:** Text length computation was accomplished by utilizing 'regexp_replace' to eliminate punctuation, along with 'Tokenizer' and 'StopWordsRemover' to tokenize the text and remove stop words. Subsequently, a new column containing the text length was generated.
- **Bucketize the Text Length:** To address this requirement, we utilized the 'Bucketizer' class from Spark MLlib in conjunction with a User Defined Function (UDF) to assign appropriate labels to the classes.
- **Compute the Correlation Coefficient:** Finally, the correlation coefficient was computed using the 'Correlation.corr' method from Spark MLlib, specifically the Spearman correlation coefficient. The data was reshaped to conform to the required format using 'VectorAssembler'.

Hypothesis 3

This hypothesis involved computing the helpfulness score and correlation coefficient, both of which were calculated using the same methods described in the previous hypothesis.

Results

In both test cases, the results closely mirrored those obtained in the local environment.

6. Helpfulness Prediction

Our ambitious objective was to build a model capable of predicting the helpfulness of a review based solely on the review text. To create such a model, we first needed to convert the text into a machine-readable format, a process known as *feature extraction*. Subsequently, we explored different models to identify the one best suited for our needs, ultimately selecting the most appropriate one.

Feature Extraction

Given the complexity of the problem, we opted to employ a *Word Embedding* technique called *Word2Vec*, which converts words into vectors of real numbers while preserving their semantic meaning. We utilized the *Gensim* library to perform this task, using the following parameters for the model:

- **Size**: 30
- **Window**: 5
- **Min Count**: 2
- **Workers**: -1

Consequently, each word is represented by a 30-dimensional vector, and the average of these vectors for all words in a review forms the vector representation of the review.

Models

We evaluated three different models:*Random Forest*,*Support Vector Regressor (RBF kernel)*, and*Multi-Layer Perceptron (MLP)*. We employed the*Scikit-Learn* library for model training and testing, utilizing the*GridSearchCV* class to perform cross-validation on the training set to identify the best model parameters. The results are presented in Table 3 and visualized in Figure 8.

Table 3: Model Results

Model	MSE	RMSE	R^2
RF	0.0259	0.1609	0.2532
SVR	0.0279	0.1670	0.1955
MLP	0.0282	0.1680	0.1858

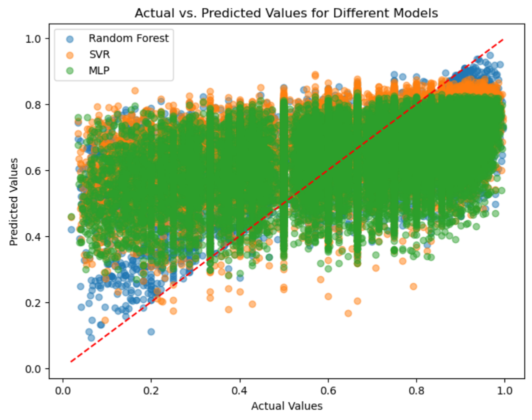

Fig. 8: Model Results

The results and metrics used indicate that the *Random Forest* model outperforms the others in terms of Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The Random Forest model achieved the lowest MSE of approximately 0.026 and RMSE of approximately 0.161, suggesting that its predictions are the closest, on average, to the actual values. This implies that the Random Forest model offers the best overall predictive performance among the three models. To further enhance model performance, we experimented with increasing the number of features from 30 to 150. However, the performance improvement was not substantial, so we chose to retain 30 features due to the optimal balance between performance and computational cost.

Results Interpretation

Figures 9 and 10 aid in interpreting the results. The scatter plot visually represents the distribution of errors, revealing that the model tends to overestimate the helpfulness of reviews with high helpfulness scores and underestimate those with low scores. A comprehensive analysis of the underlying causes of this behavior remains a subject for future investigation.

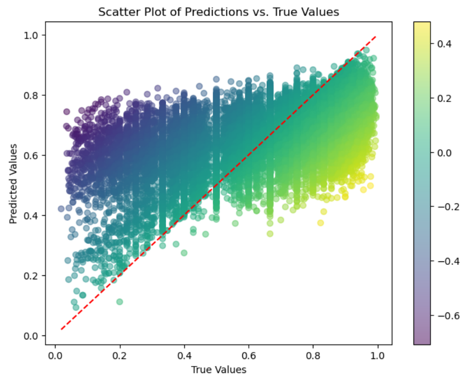

Fig. 9: Errors of the Best Model

The line plot, on the other hand, provides insights into the meaning of a given helpfulness score by translating it into Total Votes and Helpfulness Votes. The blue line represents the values (Total Votes, Helpfulness Votes) corresponding to a helpfulness score close to 0.8, while the red and green lines represent values corresponding to a helpfulness score of 0.8 plus or minus the RMSE. For instance, for a base of 100 Total Votes, the RMSE of our model corresponds to an excess or deficit of approximately 13 votes.

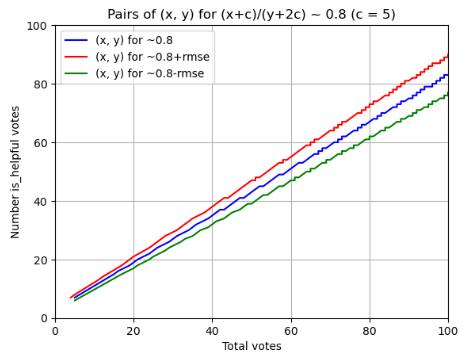

Fig. 10: Translation of Helpfulness Score Errors

7. Conclusion

In our exploration of the Amazon Books Review dataset, we've demonstrated the power of data science to unearth valuable insights. Our journey began with a dedicated team and ambitious goals, leveraging cutting-edge tools such as Hadoop, Spark, MongoDB, and Python to tackle big data challenges efficiently.

We delved into various hypotheses, examining factors like review length, sentiment, ratings, and their impact on review helpfulness. Our investigation also ventured into user profiles, book categories, and their influence on review scores, shedding light on user behavior and preferences.

While our journey faced challenges, we persisted. Some hypotheses were confirmed, while others refined our understanding of book reviews' complexities.

In conclusion, our project exemplifies the expanding horizons of data science and the importance of data-driven decision-making. We remain committed to further exploration and application of these insights, believing that data science will continue to illuminate our world and shape a data-driven future.

8. Python Script

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

```
# Deal with missing values
pipeline_missing = {'$match': {
    'review/score': {'$exists': True, '$ne': 0},
    'publisher': {'$exists': True, '$ne': None},
    'categories': {'$exists': True},
}
}

# Compute average rating for each tuple category,
    publisher
pipeline_average_rating = {'$group': {
    '_id': {
        'category': '$categories',
        'publisher': '$publisher',
    },
    'avg_score': {'$avg': '$review/score'},
    'count': {'$sum': 1}
}
}

# Show average rating for category for each
    publisher
pipeline_publisher = {'$group': {
    '_id': '$_id.category',
    'avg_score/publisher': {
        '$push': {
            'publisher': '$_id.publisher',
            'avg_score': '$avg_score',
            'count': '$count'
        }
    }
}
}

# Unwind the list of categories
pipeline_unwind = {'$unwind': '$avg_score/
    publisher'}

# Remove categories or publisher with less than '
    threshold' reviews
threshold = 0
pipeline_remove = {'$match': {
    'avg_score/publisher.count': {'$gte':
        threshold}
}
}

# Count the number of categories with average
    rating > 4.5
pipeline_counts = {'$project': {
    'category': '$_id',
    '_id': 0,
    'publisher': '$avg_score/publisher.publisher',
    'count': {
        '$sum': {
            '$cond': {

                'if': {'$gt': ['$avg_score/
                    publisher.avg_score', 4.5]},
                'then': 1,
                'else': 0
            }
        }
    }
}
}

# Sum the results for each publisher. If Total >
```

```
    10, then the hypothesis is False
pipeline_sum = {'$group': {
    '_id': '$category',
    'total': {'$sum': '$count'}
}
}

pipeline_sort = {'$sort': {
    'total': -1
}
}

results = books.aggregate([pipeline_missing,
    pipeline_average_rating, pipeline_publisher,
                          pipeline_unwind,
                              pipeline_remove,
                              pipeline_counts,
                          pipeline_sum,
                              pipeline_sort])
```