BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC KINH TẾ THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

ĐỒ ÁN MÔN HỌC

DATA SCIENCE TRENDS: KHÁM PHÁ XU HƯỚNG SÁCH KHOA HỌC DỮ LIỆU TRÊN AMAZON

Lớp - Chuyên ngành	**:** DS001 - Khoa Học Dữ Liệu
Khóa	**:** K47
Lớp học phần	**:** 23C1INF50907001
Môn học	**:** Lập trình phân tích dữ liệu

Giảng viên: TS. Nguyễn An Tế

TP Hồ Chí Minh, 2023

Thành viên nhóm

STT	Họ và tên	Lớp	MSSV
1	Nguyễn King	DS001	31211023531
2	Nguyễn Đức Huy	DS001	31201024483
3	Võ Ngọc Mỹ Kim	DS001	31211027646
4	Đặng Nhật Huy	DS001	31211027641

Mục lục

Lời nói đầu

Lĩnh vực Khoa học Dữ liệu (Data Science) đã trở thành một trong những lĩnh vực phổ biến và có sự tăng trưởng mạnh mẽ trong những năm gần đây. Điều này được chứng minh qua các chỉ số và xu hướng sau:

- Tăng trưởng của ngành công nghiệp: Ngành công nghiệp và doanh nghiệp ngày càng nhận ra giá trị của dữ liệu và sức mạnh của việc phân tích dữ liệu. Điều này đã tạo ra nhu cầu lớn cho chuyên gia Khoa học Dữ liệu, từ việc xử lý dữ liệu đến xây dựng các mô hình dự đoán và phân tích.
- Sự phát triển của công nghệ: Công nghệ ngày càng phát triển, đưa đến việc có thêm nhiều dữ liệu được tạo ra hàng ngày từ các nguồn khác nhau như mạng xã hội, các thiết bị IoT, và nhiều nguồn dữ liệu khác. Điều này tạo ra cơ hội mới trong việc khai thác và phân tích dữ liệu.
- Sự lan rộng của học thuật: Sự lan rộng của lĩnh vực Khoa học Dữ liệu trong giáo dục và học thuật đã tăng cường sự quan tâm và tìm hiểu về lĩnh vực này. Nhiều trường đại học và tổ chức đào tạo cung cấp các chương trình và khóa học về Khoa học Dữ liệu để đáp ứng nhu cầu tăng của ngành công nghiệp.
- Sự phổ biến của sách và tài liệu về Khoa học Dữ liệu: Sách và tài liệu về Khoa học Dữ liệu đã trở thành một nguồn tài nguyên quan trọng. Sự phổ biến của các cuốn sách về lĩnh vực này có thể thể hiện thông qua số lượng sách được xuất bản và sự quan tâm của người đọc.

Tuy vậy, các nhà xuất bản sách về Khoa học dữ liệu hiện nay vẫn phải đối mặt với nhiều khó khăn và thách thức như việc hiểu rõ nhu cầu của độc giả và dự đoán xu hướng và nội dung sách. Các nhà xuất bản cần nắm rõ nhu cầu và mong muốn của độc giả đối với sách về Khoa học Dữ liệu, đồng thời việc phân tích các xu hướng đọc giả, chủ đề được quan tâm và loại sách được ưa chuộng có thể giúp họ tạo ra các sách phù hợp với nhu cầu thị trường. Ngoài ra, các nhà xuất bản có thể sử dụng dữ liệu về các xu hướng trong lĩnh vực Khoa học Dữ liệu để dự đoán các chủ đề nổi bật và cung cấp thông tin cho các tác giả và nhà xuất bản về nội dung cần thiết cho sách mới. Những bài toán này có thể được giải quyết thông qua việc sử dụng dữ liệu có sẵn về các sách và thông tin liên quan, áp dụng các phương pháp phân tích dữ liệu, và xây dựng các mô hình dự đoán để hỗ trợ quyết định trong ngành xuất bản sách về Khoa học Dữ liệu.

Trong báo cáo đồ án này, chúng tôi đã thu thập được một tập mang dữ liệu tên "Amazon Data Science Books Dataset"(Tạm dịch: Bộ dữ liệu về sách khoa học dữ liệu của Amazon) được đăng tải trên Kaggle. Bộ dữ liệu được sử dụng nghiên cứu được khai thác bằng việc sử dụng thuật toán máy học trên bộ dữ liệu Amazon Data Science Books, hình thành những phân tích liên quan đến xu hướng về nội dung cũng như chất lượng của các cuốn sách về lĩnh vực khoa học dữ liệu. Đây cũng chính là lý do nhóm chúng tôi quan tâm và chọn đề tài “Data Science Trends: Khám phá xu hướng

sách khoa học dữ liệu trên Amazon" làm đồ án cuối kỳ học phần Lập trình phân tích dữ liệu khoa Công nghệ thông tin Kinh doanh trường Đại học Kinh tế (UEH). Do điều kiện về mặt thời gian và nhận thức còn nhiều hạn chế nên đồ án không khỏi có phần sai sót, kính mong quý giảng viên nhiệt tình góp ý để nhóm chúng tôi có những tiếp thu tốt hơn trong môn học này. Qua bài đồ án, nhóm chúng tôi xin cảm ơn giảng viên phụ trách môn học Lập Trình Phân Tích Dữ Liệu, thầy Nguyễn An Tế đã truyền đạt những kiến thức quý báu cho học viên bằng cả tấm lòng nhiệt tình và sự tận tâm của thầy.

1 Tổng quan đề tài

1.1 Xác định đề tài

Khóa luận này nhằm mục đích phân tích và khai thác bộ dữ liệu "Amazon Data Science Books Dataset"để đạt được các mục tiêu nghiên cứu sau đây:

- Phân tích xu hướng phát triển của lĩnh vực Khoa học Dữ liệu: Bằng cách sử dụng dữ liệu về số lượng sách mới xuất bản theo thời gian, khóa luận sẽ xác định xu hướng phát triển của lĩnh vực Khoa học Dữ liệu và đưa ra những nhận định về sự thay đổi, phổ biến của các chủ đề, công nghệ trong lĩnh vực này.
- Phân tích mối tương quan giữa điểm đánh giá và số lượng đánh giá: Bằng việc sử dụng dữ liệu về điểm đánh giá và số lượng đánh giá của các cuốn sách, khóa luận sẽ phân tích mối quan hệ giữa chúng để hiểu rõ hơn về sự ảnh hưởng của số lượng đánh giá đến chất lượng đánh giá của một cuốn sách.
- Xây dựng mô hình dự đoán điểm đánh giá của sách: Sử dụng các thuộc tính của sách như tác giả, nhà xuất bản, ngày phát hành, giá bán, khóa luận sẽ xây dựng một mô hình dự đoán điểm đánh giá của sách, từ đó cung cấp thông tin hữu ích cho độc giả khi lựa chọn sách.

1.2 Mục đích nghiên cứu

Mục tiêu của khóa luận là từ bộ dữ liệu "Amazon Data Science Books Dataset"thu thập được, đưa ra những phân tích sâu sắc, nhận định chính xác và xây dựng mô hình có thể ứng dụng trong thực tế, góp phần nâng cao chất lượng và hiệu quả trong việc tìm kiếm, đánh giá sách về Khoa học Dữ liệu trên nền tảng thương mại điện tử."

1.3 Giới hạn phạm vi đề tài

Phạm vi thời gian: Nghiên cứu sẽ tập trung vào dữ liệu từ một khoảng thời gian cụ thể, ví dụ như từ năm 2010 đến năm 2020, để phản ánh xu hướng phát triển của lĩnh vực Khoa học Dữ liệu trong một khoảng thời gian đủ lớn nhưng vẫn có thể quản lý được.

Phạm vi dữ liệu: Dữ liệu sẽ tập trung vào các cuốn sách liên quan đến lĩnh vực Khoa học Dữ liệu, bao gồm thông tin về tên sách, tác giả, nhà xuất bản, ngày phát hành, giá bán, số đánh giá và điểm đánh giá trung bình.

Phạm vi phân tích: Phần lớn phân tích sẽ tập trung vào việc thăm dò dữ liệu, phân tích thống kê mô tả, xây dựng mô hình dự đoán điểm đánh giá. Khóa luận sẽ không đi sâu vào việc phân tích nội dung của từng cuốn sách . Qua việc giới hạn phạm vi đề tài, khóa luận mong muốn đảm bảo tính

chính xác và khả thi của nghiên cứu, đồng thời tập trung vào những khía cạnh quan trọng nhất của bộ dữ liệu để đạt được mục tiêu nghiên cứu.

1.4 Phương pháp nghiên cứu

Sử dụng các thuật toán máy học Machine Learning, phương pháp phân cụm - phân lớp cho các giá trị thu được cho bộ dữ liệu Amazon Data Science Books và biểu diễn trực quan kết quả theo các biến mục tiêu đặt ra.

1.5 Ý nghĩa nghiên cứu

Việc nghiên cứu và phân tích bộ dữ liệu này có ý nghĩa quan trọng đối với việc nghiên nên cứu xu hướng phát triển của lĩnh vực Khoa học Dữ liệu. Kết quả nghiên cứu có thể cung cấp thông tin tham khảo cho các tác giả, nhà xuất bản trong việc lựa chọn định hướng nghiên cứu, biên soạn sách phù hợp. Đồng thời, kết quả cũng giúp ích cho độc giả trong việc lựa chọn các cuốn sách uy tín.

2 Tổng quan bộ dữ liệu

2.1 Tổng quan bộ dữ liệu thu thập

2.1.1 Giới thiệu bộ dữ liệu

Bộ dữ liệu "Amazon Data Science Books Dataset"là một tập dữ liệu chứa thông tin về các cuốn sách liên quan đến lĩnh vực Khoa học Dữ liệu, được thu thập từ trang thương mại điện tử Amazon. Bộ dữ liệu này bao gồm các thông tin chi tiết về hơn 10.000 cuốn sách, bao gồm tên sách, tác giả, nhà xuất bản, ngày phát hành, giá bán, số đánh giá và điểm đánh giá trung bình.

2.1.2 Các thuộc tính của bộ dữ liệu

Bộ dữ liệu Amazon Data Science Books chứa 18 cột bao gồm:

STT	Tên thuộc tính	Mô tả	Ghi chú
1	title	Tiêu đề của cuốn sách	
2	author	Tên tác giả	
3	price (including used books)	Giá	Tính theo đồng đô-la
4	pages	Số trang	
5	avg_reviews	Số điểm đánh giá trung bình	Theo thang điểm 5
6	reviews done for each book	Tổng số lượt đánh giá	
7	star5	Tỷ lệ đánh giá 5 sao	0% đến 100%
8	star4	Tỷ lệ đánh giá 4 sao	0% đến 100%
9	star3	Tỷ lệ đánh giá 3 sao	0% đến 100%
10	star2	Tỷ lệ đánh giá 2 sao	0% đến 100%
11	star1	Tỷ lệ đánh giá 1 sao	0% đến 100%
12	dimensions	Kích cỡ cuốn sách	Tính theo inches
13	weight	Trọng lượng cuốn sách	Tính theo pounds hoặc ounces
14	language	Ngôn ngữ của cuốn sách	
15	publisher	Tên nhà xuất bản	
16	ISBN-13	Mã ISBN-13	
17	link	Đường dẫn đến Amazon	/sach-a
18	complete_link	Đường dẫn hoàn chỉnh đến Amazon	https://amazon.com/sach-a

Bảng 1: Bảng mô tả các thuộc tính của bộ dữ liệu

2.2 Tiến trình xử lý bộ dữ liệu thu thập

Ta sử dụng bộ dữ liệu "Amazon Data Science Books"để hình thành những phân tích liên quan đến xu hướng sách khoa học dữ liệu qua các năm. Để làm được điều này, ta chia dự án này thành các nhiệm vụ nhỏ có thể tổ chức và quản lý được thông qua sáu giai đoạn của quy trình phân tích dữ liệu:

- Giai đoạn 1 - Problem Understanding: Làm rõ vấn đề và bối cảnh hiện tại để xác định mục tiêu khai thác dữ liệu. Ở đây mục tiêu chúng ta hướng đến là làm sao có thể thực hiện được một phân tích dữ liệu về xu hướng sách khoa học dữ liệu trên trang Amazon. Khi đã hiểu được xu hướng cũng như sở thích của người đọc với lĩnh vực này thể hiện qua số lượng đánh giá, mức độ đánh giá (rating) của từng cuốn sách, giá sách,.. ta sẽ tiếp cận được mục tiêu quan trọng là cung cấp thông tin về các cuốn sách chất lượng đối với lĩnh vúc khoa học dữ liệu, đồng thời có thể giúp nhà xuất bản đánh giá chất lượng của như giá thành của mỗi cuốn sách.
- Giai đoạn 2 - Data Understanding: Kiểm tra tình trạng dữ liệu để xác định dữ liệu đang có liệu có phù hợp với mục tiêu khai thác hay không. Bước làm này ta sẽ xác định những dữ liệu cần thiết cho quá trình phân tích cũng như kiểm tra mức độ “sạch” của bộ dữ liệu với các tiêu chuẩn đề ra: không có chứa dữ liệu missing data, lọc nhiễu và lượt bớt dữ liệu.
- Giai đoạn 3 - Data preparation: Thực hiện các bước tiền xử lý để chuẩn hóa dữ liệu sẵn sàng cho các giai đoạn tiếp theo. Giai đoạn này thường chiếm đến 90% thời gian của cả quy trình. Ta sẽ thực hiện xử lý bộ dữ liệu theo những phát hiện tìm được ở giai đoạn 2 bằng lập trình phân tích dữ liệu - tiền xử lý dữ liệu.
- Giai đoạn 4 - Mô hình hóa: Sử dụng các mô hình thống kê, máy học để xác định các mẫu/ quy luật của dữ liệu. Bước làm này rất quan trọng trong việc đưa ra gợi ý và đánh giá cho người đọc. Đối với bộ "Amazon Data Science Books", ta cần thực hiện lập trình phân tích các chỉ số của bộ dữ liệu gốc cũng như thể hiện nó qua biểu đồ biểu diễn trực quan dữ liệu giúp người đọc quan sát dễ dàng, thuận tiện đánh giá và phân tích.
- Giai đoạn 5 - Đánh giá: Kiểm tra tính hiệu quả của mô hình có đáp ứng với mục tiêu kinh doanh hay không, có đủ tin cậy hay không.
- Giai đoạn 6 - Triển khai: Đưa mô hình giải pháp vào ứng dụng cho việc đánh giá chất lượng sách và gợi ý thông tin hữu ích cho các nhà xuất bản. Bước làm này ta có thể đưa ra những giải pháp dựa theo đánh giá và mô hình ở giai đoạn 4 - 5.

3 Tiền xử lý dữ liệu

3.1 Định nghĩa

Tiền xử lý dữ liệu (Data Preprocessing) là quá trình tiền chuẩn bị và làm sạch dữ liệu từ nguồn dữ liệu ban đầu để chuẩn bị cho các phương pháp phân tích, khai thác thông tin, hoặc huấn luyện mô hình máy học. Quá trình này là một bước quan trọng và cần thiết trước khi tiến hành bất kỳ phân tích hoặc mô hình hóa nào.

Quá trình tiền xử lý dữ liệu bao gồm nhiều bước:

- Thu thập dữ liệu: Thu thập dữ liệu từ nhiều nguồn khác nhau như cơ sở dữ liệu, tệp tin, API, hoặc bất kỳ nguồn nào chứa thông tin cần thiết.
- Kiểm tra dữ liệu: Xác định và phân tích dữ liệu để kiểm tra tính chính xác, hoàn chỉnh và tính nhất quán. Điều này bao gồm kiểm tra dữ liệu bị thiếu, dữ liệu ngoại lệ, và kiểm tra định dạng dữ liệu.
- Xử lý dữ liệu thiếu: Xử lý các giá trị thiếu bằng cách điền giá trị thiếu hoặc loại bỏ các mẫu dữ liệu không đủ thông tin.
- Chuẩn hóa dữ liệu: Chuẩn hóa các đơn vị đo lường, biến đổi dữ liệu về cùng một phạm vi, hoặc chuyển đổi dữ liệu để đảm bảo đồng nhất trong phân phối.
- Chuyển đổi dữ liệu: Chuyển đổi dữ liệu sang định dạng phù hợp cho việc phân tích, như mã hóa biến phân loại, chuyển đổi ngày tháng sang định dạng datetime, hoặc chuyển đổi số liệu theo cách tiện lợi hơn cho mô hình hóa.
- Loại bỏ dữ liệu ngoại lệ: Xác định và xử lý các giá trị ngoại lệ không phản ánh chân thực của tập dữ liệu.
- Tạo và xử lý biến mới: Tạo các biến mới từ các biến hiện có thông qua các phép tính hoặc kết hợp giữa các biến để tăng cường thông tin và cải thiện mô hình.

3.2 Kiểm tra tình trạng bộ dữ liệu gốc

Bộ dữ liệu Amazon Data Science Books Dataset được thu thập cần phải kiểm tra các yếu tố về tình trạng đầu vào để làm sạch dữ liệu (Data cleaning/cleansing), loại bỏ nhiễu (remove noise) cũng như hiệu chỉnh những phần dữ liệu không nhất quán (correct data inconsistencies).

Code

```
print('Chi tiết bộ data')
print(df.info())
```

```
print(f'Số lượng phần tử bộ dữ liệu: {df.size}')
print(f'Số dòng và cột của bộ dữ liệu - (dòng, cột): {df.shape}')
```

Kết quả

```
Chi tiết bộ data
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 830 entries, 0 to 829
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   title                           830 non-null    object
 1   author                          657 non-null    object
 2   price                           722 non-null    float64
 3   price (including used books)    722 non-null    object
 4   pages                           745 non-null    object
 5   avg_reviews                     702 non-null    float64
 6   n_reviews                       702 non-null    object
 7   star5                           702 non-null    object
 8   star4                           635 non-null    object
 9   star3                           554 non-null    object
 10  star2                           451 non-null    object
 11  star1                           328 non-null    object
 12  dimensions                      644 non-null    object
 13  weight                          651 non-null    object
 14  language                        759 non-null    object
 15  publisher                       714 non-null    object
 16  ISBN_13                         665 non-null    object
 17  link                            830 non-null    object
 18  complete_link                   830 non-null    object
dtypes: float64(2), object(17)
memory usage: 123.3+ KB
None
Số lượng phần tử bộ dữ liệu: 15770
Số dòng và cột của bộ dữ liệu - (dòng, cột): (830, 19)
```

Ta thấy bộ dữ liệu gồm 830 dòng và 19 cột. Với 2 kiểu dữ liệu chính là: Float (2 thuộc tính), Object (17 thuộc tính).

3.2.1 Kiểm tra dữ liệu lặp lại

Định nghĩa:

Quá trình kiểm tra dữ liệu bị lặp trong tiền xử lý dữ liệu là một bước quan trọng để xác định và loại bỏ các mẫu dữ liệu trùng lặp hoặc tương đồng cao trong tập dữ liệu. Dữ liệu lặp có thể xuất hiện trong các tình huống khác nhau như thu thập dữ liệu không chính xác hoặc quá trình thu thập dữ liệu không đảm bảo việc loại bỏ các mẫu dữ liệu trùng lặp, gây ảnh hưởng tiêu cực đến kết quả cuối cùng hoặc hiệu suất của mô hình.

Code

```
import pandas as pd
import numpy as np

# Check for duplicates
duplicates = df[df.duplicated()]
if duplicates.empty:
    print("No duplicates found.")
else:
    print("Duplicates found. Rows to be removed:")
    print(duplicates)
```

Kết quả

```
No duplicates found.
```

Giải thích: Kết quả từ đoạn code trên cho thấy tập dữ liệu "Amazon Data Science Books Dataset"không xuất hiện các dòng dữ liệu nào bị trùng lặp. Điều này có ý nghĩa tích cực, bởi việc không có dữ liệu trùng lặp giúp duy trì tính toàn vẹn và độ chính xác của tập dữ liệu, đồng thời tránh được những sai sót trong quá trình xử lý và phân tích dữ liệu.

3.2.2 Kiểm tra dữ liệu bị thiếu (Missing values)

Định nghĩa:

Quá trình kiểm tra dữ liệu bị thiếu (Missing Values) trong tiền xử lý dữ liệu là bước quan trọng để phát hiện và xử lý các giá trị còn thiếu trong tập dữ liệu. Dữ liệu bị thiếu có thể xuất hiện do nhiều nguyên nhân như quá trình thu thập dữ liệu không hoàn chỉnh, lỗi trong quá trình ghi nhận thông tin, hoặc do nguyên nhân khác như quá trình nhập liệu.

Code

```python
# Check for missing values
missing_values = df.isnull().sum()
if missing_values.sum() == 0:
    print("No missing values found.")
else:
    print("Missing values found. Columns with missing values:")
    print(missing_values[missing_values > 0])
```

Kết quả

```
Missing values found. Columns with missing values:
author                          173
price                           108
price (including used books)    108
pages                            85
avg_reviews                     128
n_reviews                       128
star5                           128
star4                           195
star3                           276
star2                           379
star1                           502
dimensions                      186
weight                          179
language                         71
publisher                       116
ISBN_13                         165
dtype: int64
```

Giải thích: Kết quả của đoạn code ở trên thể hiện thông tin về số lượng giá trị bị thiếu (missing values) trong từng cột của DataFrame. Mỗi hàng trong output đều thể hiện số lượng giá trị thiếu của mỗi cột. Ta có thể nhận thấy hầu hết các cột đều có số lượng giá trị bị thiếu không quá lớn (<25%), ngoại trừ 3 cột *star3*, *star2* và *star1* có số lượng giá trị bị thiếu cao hơn hẳn so với các cột khác.

3.2.3 Kiểm tra dữ liệu ngoại lệ (Outliers)

Định nghĩa:

Quá trình kiểm tra dữ liệu ngoại lệ (Outliers) trong tiền xử lý dữ liệu là quá trình xác định và xử lý các giá trị không phù hợp hoặc không thường xuyên xuất hiện trong tập dữ liệu. Dữ liệu ngoại lệ có thể xuất hiện do nhiều nguyên nhân như lỗi trong quá trình thu thập dữ liệu, đo lường không chính xác, hoặc là dữ liệu thực tế nhưng không phản ánh đặc điểm chung của tập dữ liệu.

Đối với bộ dữ liệu "Amazon Data Science Books Dataset", ta sẽ thực hiện kiểm tra dữ liệu ngoại lệ sử dụng *quy tắc 3-sigma* đối với các biến định lượng là *price* và *avg_reviews*.

Code

```
# Validate numeric columns for outliers
numeric_cols = df.select_dtypes(include=np.number).columns
for col in numeric_cols:
    # Assuming a simple Z-score based outlier detection method
    z_scores = (df[col] - df[col].mean()) / df[col].std()
    outliers = df[abs(z_scores) > 3][col]
    if outliers.empty:
        print(f"No outliers found in column '{col}'.")
    else:
        print(f"Outliers found in column '{col}':")
        print(outliers)
```

Kết quả

```
Outliers found in column 'price':
248     250.63
638     743.47
677     287.14
734    1318.74
Name: price, dtype: float64
Outliers found in column 'avg_reviews':
56     3.0
137    1.0
258    3.0
260    3.0
311    2.8
343    3.0
399    1.0
405    3.0
637    1.0
```

```
724    2.7
Name: avg_reviews, dtype: float64
```

Giải thích:

- Cột 'price': Có 4 giá trị ngoại lệ được xác định với các giá trị nằm ngoài giới hạn 3-sigma. Các giá trị này là: 250.63, 743.47, 287.14, và 1318.74.
- Cột 'avg_reviews': Xuất hiện 10 cuốn sách nằm ngoài giới hạn 3-sigma của biến này.

Các giá trị được xác định là ngoại lệ dựa trên việc tính toán z-score của từng giá trị trong cột. Z-score được sử dụng để đo độ lệch của một điểm dữ liệu so với trung bình của dữ liệu và được chuẩn hóa theo độ lệch chuẩn (standard deviation). Trong trường hợp này, các giá trị với z-score lớn hơn 3 (hoặc nhỏ hơn -3) được coi là ngoại lệ.

3.2.4 Kiểm tra định dạng và tính nhất quán của dữ liệu

Định nghĩa:

Việc kiểm tra định dạng và tính nhất quán của dữ liệu trong quá trình tiền xử lý là bước quan trọng để đảm bảo dữ liệu đầu vào đáp ứng các tiêu chuẩn cần thiết cho việc phân tích hoặc xử lý sau này. Điều này bao gồm việc kiểm tra và chuẩn hóa định dạng của các cột dữ liệu, đồng thời đảm bảo tính nhất quán giữa các giá trị trong các cột.

Kiểm tra định dạng cột `ISBN-13`:

ISBN (viết tắt của International Standard Book Number, Mã số tiêu chuẩn quốc tế cho sách) là mã số tiêu chuẩn quốc tế có tính chất thương mại duy nhất để xác định một quyển sách. ISBN-13 sẽ có 13 số đại diện cho 1 cuốn sách với định dạng '<3 chữ số>-<10 chữ số>'. Ta thực hiện kiểm tra xem các số ISBN trong cột *ISBN-13* đã đúng định dạng hay chưa, và nếu chưa thì ta sẽ thực hiện in ra màn hình và kiểm tra xem liệu dữ liệu bị nhập sai (ví dụ thiếu dấu '-') hay nội dung hoàn toàn khác so với mã số thông thường.

Code

```python
# Check for data format and consistency (specific to columns)
# Validating ISBN-13 format
invalid_isbn = df[~df['ISBN_13'].astype(str).str.match(r'^\d{3}-\d{10}$')]
if invalid_isbn.empty:
    print("All ISBN-13 codes are in valid format.")
else:
    print("Invalid ISBN-13 format found:")
    print(list(invalid_isbn['ISBN_13'].unique()))
```

Kết quả

```
Invalid ISBN-13 format found:
[nan, 'Research in Drama Education', ' ', '$13.99 ', '-34%', '     ',
↪  'Usually ships within 2 to 3 days.', '#NAME?', '#250 in ', '#1 in ', '
↪  Second Edition, Second edition ', '$113.19 ', '-22%', 'The Data
↪  Revolution', 'Regulating Alcohol around the World: Policy Cocktails',
↪  '-19%', '#35 in ', 'x', '-27%', '79', '99', '-21%', '75', 'Pratip
↪  Samanta', '49', '69', '29', '66', '-30%', '  Kindle Edition ',
↪  '$139.99 ', '$13.79 ', 'Python', '#372 in ', 'PennyLane', 'JOSH
↪  TYSON', '-', '-39%', '59', '39', '-25%', '30', '56', '25']
```

Giải thích: Từ kết quả trên, ta có thể thấy ngoài *nan* là giá trị bị thiếu thì tất cả các giá trị còn lại đều có nội dung khác hoàn toàn so với mã số ISBN thông thường (như *#NAME?*, *-34%*...). Do đó ta có thể gom các giá trị này lại thành một giá trị duy nhất là *Unknown* trong quá trình tiền xử lý tiếp theo.

Kiểm tra định dạng cột `complete_link`:

Tiếp theo, ta sẽ thực hiện kiểm tra định dạng cột *complete_link*. Cột này chứa các giá trị là URL dẫn đến trang Amazon của cuốn sách đó. URL (hay Uniform Resource Locator, nghĩa tiếng Việt: Hệ thống định vị tài nguyên thống nhất; được gọi một cách thông thường là một địa chỉ web) là một tham chiếu đến tài nguyên web chỉ định vị trí của nó trên một mạng máy tính và cơ chế để truy xuất nó. Định dạng của một URL đối với bộ dữ liệu "Amazon Data Science Books Dataset"sẽ có dạng "https://www.amazon.com/<tên cuốn sách>".

Code

```python
# Validate URLs in link columns
# Check for valid URL format in 'complete_link'
link_columns = ['complete_link']
for col in link_columns:
    invalid_urls = df[~df[col].str.startswith('https://www.amazon.com')]
    if invalid_urls.empty:
        print(f"All URLs in column '{col}' are valid Amazon URLs.")
    else:
        print(f"Invalid URLs found in column '{col}':")
        print(invalid_urls[col])
```

Kết quả

```
All URLs in column 'complete_link' are valid Amazon URLs.
```

Giải thích: Từ kết quả trên, ta có thể thấy tất cả các URL đều có định dạng chính xác với tiêu chuẩn.

3.3 Tiền xử lý dữ liệu

3.3.1 Xử lý định dạng dữ liệu (Data format)

Đối với bộ dữ liệu "Amazon Data Science Books Dataset", ta sẽ thực hiện xử lý định dạng theo từng biến (cột).

Biến `weight`

Đây là cột thể hiện khối lượng của một cuốn sách. Ta sẽ thực hiện in ra các giá trị độc nhất (unique values) của cột này để thực hiện phân tích định dạng và chuẩn hóa nó.

```
...'9.1 ounces', '2.37 pounds', '1.99 pounds', '2.41 pounds',
'3.52 ounces', '15.7 ounces', '1.29 pounds', '1.73 pounds',
'4.73 pounds', '5.51 pounds',
'I love the way Seth Stephens-Davidowitz explains how we can better live
↪  our lives by exploiting the small advantages in life. On the
↪  basketball court, I made a career out of finding these types of minor
↪  advantages, and I've found that most successful individuals in life
↪  value the accumulation of small advantages. In the end, they add up to
↪  significant life benefits. -- ',
'3.44 pounds', '2.61 pounds', '6.9 ounces', '2.78 pounds',...
```

Ở đây ta có thể thấy, hầu hết các cột đều có định dạng '<khối lượng> <đơn vị>' với đơn vị có thể là *ounces* hoặc *pounds*. Tuy nhiên, trong cột lại xuất hiện các giá trị nhiễu như '*I love the way Seth Stephens-Davidowitz explains...*'. Ta sẽ sử dụng biểu thức chính quy (regular expression) để trích xuất các giá trị số trong cột, chuyển về kiểu *float* và quy đổi tất cả các giá trị có đơn vị *pounds* về *ounces* (1 pound = 16 ounces). Đồng thời, ta cũng chuyển tất cả các giá trị nhiễu về *nan*.

Code

```python
import re
def preprocess_weight(weight):
    try:
        # Extract numeric values
        numeric_value = float(re.findall(r'^(.+) \w+', weight)[0])
```

```
        # Check for 'pounds' and convert to ounces
        if 'pounds' in weight:
            numeric_value *= 16.0  # 1 pound = 16 ounces

        return numeric_value
    except:
        return np.nan  # Return NaN for non-convertible values

processed_weight = [preprocess_weight(wei) for wei in df['weight']]
df['weight'] = processed_weight

# Display the processed weight column
print(df['weight'])
```

Kết quả

```
0      40.48
1      31.36
2      22.40
3      23.52
4      20.80
       ...
825    15.50
826    36.00
827    17.60
828      NaN
829      NaN
Name: weight, Length: 830, dtype: float64
```

Giải thích: Như vậy, ta có thể thấy tất cả các các giá trị đã được quy đổi về kiểu *float* với cùng đơn vị là *ounces* và không còn giá trị nhiễu trong cột.

Biến `star5 - star1`

Ta sẽ sử dụng hàm *unique()* để in ra các giá trị độc nhất (unique values) của các cột *star5*, *star4*, *star3*, *star2*, *star1*.

```
array(['55%', '61%', '87%', '75%', '52%', '84%', '100%', '78%', '73%',
       '91%', '63%', '81%', '48%', '72%', nan, '74%', '90%', '76%', '85%',
       '86%', '94%', '70%', '97%', '67%', '79%', '82%', '68%', '65%',
       '80%', '69%', '83%', '77%', '53%', '50%', '60%', '66%', '95%',
       '58%', '59%', '22%', '64%', '44%', '34%', '98%', '62%', '93%',
       '56%', '57%', '38%', '54%', '92%', '37%', '71%', '43%', '47%',
       '51%', '99%', '36%', '24%', '42%', '89%', '88%', '39%', '46%',
       '40%', '49%', '20%', '33%'], dtype=object)
```

Đối với cột này, ngoại trừ *nan* là giá trị bị thiếu thì định dạng của các cột 'star' đang ở kiểu *string*. Ta sẽ thực hiện trích xuất phần trăm từ các cột này, chuyển về dạng *float* và biểu diễn dưới dạng số thập phân.

Code

```
star_columns = ['star5', 'star4', 'star3', 'star2', 'star1']
def preprocess_star(value):
    try:
        # Extract numeric values
        numeric_value = float(value) / 100.0
        return numeric_value
    except:
        return np.nan  # Return NaN for non-convertible values

for col in star_columns:
    df[col] = [preprocess_star(value) for value in
    ↪  df[col].str.replace(r'%', '')]
```

Kết quả

```
0      0.55
1      0.61
2      0.87
3      0.75
4      0.52
       ...
825    0.72
826    0.78
```

```
827    0.83
828     NaN
829    0.79
Name: star5, Length: 830, dtype: float64
```

Giải thích: Như vậy, ta có thể thấy dữ liệu về phần trăm đánh giá của các cột 'star' đã được định dạng về dạng số thập phân với kiểu dữ liệu *float*.

Biến `n_reviews`

Đây là cột thể hiện số lượng các đánh giá cho mỗi cuốn sách. Ta tiếp tục in ra màn hình các giá trị độc nhất (unique values) của cột này để đánh giá định dạng và kiểu dữ liệu.

```
array(['77', '4,388', '4,228', '807', '621', '579',
'932', '236', '326', '1,374', '59', '146', '336', '210', '64',
'141', '905', '159', '120', '185', '66', '303', '71', '238',
'1,001', '132', '1,068', '65', '149', '3,310', '271', '3,129',
'315', '594', '505', '145', '178', '121', '209', '109', '501',
'67', '155', '989', '163', '3,678', '3,142', '114', '1,386', '43', '688',
 '213', '606', '135', '74', '184', '1,291', '111', '123', '88', '408',
 '323', '3,337', '75', '324', '106', '591', '372', '414', '1,512',
 '845', '150', '390', '197', '1,724', '1,505', '7,767', '186', '263',
 '98', '402', '2,906', '68', '608', '476', '280', '5,517', '47',
 '1,123', '8,819', '839', '94', '687', '388', '1,338', '1,192', '988',
 '35', '177', '7,953', '262', '1,537', '237', '164', '404'],
 dtype=object)
```

Các giá trị trong cột này hiện đang có kiểu dữ liệu *string*, đồng thời các số hàng nghìn đang được phân tách với các số khác bằng dấu phẩy. Ta sẽ thực hiện xóa dấu phẩy và chuyển tất cả về kiểu dữ liệu *int*.

Code

```
df['n_reviews'] = df['n_reviews'].str.replace(r',', '',
 regex=True).astype("Int64")
```

Kết quả

```
0        23
1       124
```

```
2        10
3      1686
4        12
       ...
825      74
826      93
827       8
828    <NA>
829     142
Name: n_reviews, Length: 830, dtype: Int64
```

Giải thích: Như vậy, tất cả các giá trị trong cột này đã được chuyển về kiểu dữ liệu *int*.

Biến `pages`

Đây là cột thể hiện số trang sách của mỗi cuốn sách. Ta thực hiện in ra các giá trị độc nhất của cột để xác định các kiểu định dạng của cột và thực hiện chuẩn hóa về định dạng chung.

```
array(['425', '135', '723', '570', '502', '196', '290',
'Parse complicated HTML  ', '455', '590', '310', '171', '398',
'172', '714', '231', '77', '163', '241', '142', '493', '191',
'  of high quality paper', '308', '48', '262', '601', '32', '232',
'190', '507', '1650', '1643', '705', '372', '2962', '614', '521',
'166', '512', '446', '318', '326', '370', '462', '170', '572',
'31', '415', '340', '430', '59', '314', '98', '243', '316', '822',
'555', '460', '364', '311', '186', '94', '482', '748', '181',
'210', '270', '558', '687', '158', '522', '258', '667', '195',
'378', 'Full-color , showcasing all 24 final data visualizations',
'774', '573'], dtype=object)
```

Ta thấy rằng hầu hết các cột đều thể hiện số trang và các giá trị đang ở kiểu dữ liệu *string*. Tuy nhiên, cột này có xuất hiện các giá trị nhiễu (như '*Parse complicated HTML*', '*Full-color , showcasing...*). Do đó, ta sẽ thực hiện chuyển kiểu dữ liệu của cột về *float* và chuyển tất cả giá trị nhiễu thành *NaN*.

Code

```
# Xóa dấu phẩy xuất hiện trong cột
df['pages'] = df['pages'].str.replace(r',', '', regex=True)

def preprocess_pages(value):
```

```
    try:
        # Trích xuất giá trị số
        numeric_value = float(value)
        return numeric_value
    except:
        return np.nan

df['pages'] = [preprocess_pages(value) for value in df['pages']]
df['pages'] = df['pages'].astype("Int64") # Thực hiện chuyển sang kiểu int
```

Kết quả

```
0       500
1       484
2       274
3       547
4       368
       ...
825     208
826     573
827     288
828    <NA>
829    <NA>
Name: pages, Length: 830, dtype: Int64
```

Giải thích: Phân tích các bước xử lý cột 'pages'

- Xóa dấu phẩy (,) trong cột: Sử dụng phương thức str.replace để loại bỏ dấu phẩy nếu có trong các giá trị của cột 'pages'. Phương pháp này giúp chuẩn hóa định dạng số, ví dụ làm cho dữ liệu số có dạng '1,000' thành '1000'.
- Hàm preprocess_pages: Đây là một hàm được tạo ra để chuyển đổi các giá trị của cột 'pages' thành dạng số. Hàm này cố gắng chuyển đổi chuỗi thành dạng số float. Nếu giá trị không thể chuyển đổi, hàm sẽ trả về NaN.
- Áp dụng hàm preprocess_pages: Với mỗi giá trị trong cột 'pages', hàm preprocess_pages được áp dụng để chuyển đổi dữ liệu về kiểu dữ liệu số. Kết quả trả về là một Series mới.
- Chuyển kiểu dữ liệu: Cuối cùng, sau khi xử lý, cột 'pages' được chuyển đổi sang kiểu dữ liệu số nguyên (Int64) bằng cách sử dụng .astype("Int64").

Kết quả cuối cùng là cột 'pages' với các giá trị đã được chuyển đổi thành dạng số nguyên (Int64). Các giá trị không thể chuyển đổi hoặc là giá trị nhiễu đã được thay thế bằng giá trị NaN để làm sạch dữ liệu.

Biến `price (including used books)`

Ta in ra các kiểu định dạng hiện tại trong cột này bằng cách sử dụng hàm 'unique()' trong pandas.

```
array(['6.75', ' 21.49 - 33.72 ', '32.07', '53.99', '24.49', nan, '40.49',
       '90', ' 53.98 - 54.19 ', ' 47.97 - 56.99 ', '20', '15.97', '28.6',
       '39.99', '24.99', '10.69', '19.38', ' 36.00 - 49.99 ', '29.99',
       '24.2', '9.99', '68.59', ' 47.10 - 51.95 ', ' 25.50 - 50.99 '],
       ↪  dtype=object)
```

Định dạng hiện tại của cột 'price (including used books)' là '<giá thấp nhất> - <giá gốc>' hoặc chỉ xuất hiện giá gốc, tất cả đều đang có kiểu dữ liệu *object*. Vì giá gốc đã được lưu trữ trong cột 'price', ta sẽ thực hiện trích xuất giá thấp từ các giá trị cột này, đổi kiểu dữ liệu thành *float* và chuyển sang cột mới là *min_price*.

Code

```
# Lấy giá thấp nhất trong cột 'price (including used books)' và đưa sang
↪  cột min_price
def preprocess_price(price):
    try:
        min_price = float(re.split(r'-', price)[0])
        return min_price
    except:
        return np.nan  # Return NaN for non-convertible values
df['min_price'] = [preprocess_price(price) for price in df['price
↪  (including used books)']]
```

Kết quả

```
0       6.75
1      21.49
2      32.07
3      53.99
4      24.49
       ...
```

```
825     8.55
826    52.41
827    44.99
828      NaN
829    38.49
Name: min_price, Length: 830, dtype: float64
```

Giải thích:

Đoạn code trên thực hiện việc chuyển đổi dữ liệu từ cột 'price (including used books)' sang cột mới 'min_price', chứa giá thấp nhất của mỗi cuốn sách.

- Hàm preprocess_price: Đây là hàm được định nghĩa để chuyển đổi giá trị trong cột 'price (including used books)' thành giá thấp nhất trong trường hợp giá đó là một khoảng giá (ví dụ: '6.75 - 20.00'). Hàm này sử dụng biểu thức chính quy re.split(r'-', price)[0] để tách phần số đầu tiên trong trường hợp có khoảng giá. Nếu không thể chuyển đổi, hàm sẽ trả về NaN.
- Áp dụng hàm preprocess_price vào cột 'price (including used books)': Một vòng lặp for được sử dụng để áp dụng hàm preprocess_price cho từng giá trị trong cột 'price (including used books)'. Kết quả là một Series mới có giá trị là giá thấp nhất từ các giá trị trong mỗi mẫu dữ liệu.
- Tạo cột mới 'min_price': Kết quả trả về từ hàm preprocess_price được gán vào cột mới 'min_price' trong DataFrame.

Kết quả cuối cùng là cột 'min_price' mới chứa giá trị giá thấp nhất trong mỗi mẫu dữ liệu từ cột ban đầu 'price (including used books)'. Các giá trị không thể chuyển đổi đã được thay thế bằng giá trị NaN.

Biến `author`:

Ta thực hiện in ra các giá trị độc nhất (unique values) trong cột 'author' để tiến hành phân tích định dạng hiện tại của cột và tìm hướng xử lý.

```
array(['[ Dr Dhaval Maheta]', nan, '[ Oz du Soleil,  and , Bill Jelen]',
       '[ William McKinney]', '[ Paul McFedries]', '[ Cathy Tanimura]',
       '[ Matthew B. Miles, A. Michael Huberman, et al.]',
       '[ Gunnar Carlsson,  and , Mikael Vejdemo-Johansson]',
       '[ Robert I. Kabacoff]', '[ Chad Knowles]',
       '[ Programming Languages Academy, Matthew Kinsey, et al.]',
```

```
       '[ JOE WEBINAR]', '[ Jules Damji, Brooke Wenig, et al.]'],...
      dtype=object)
```

Ta có thể nhận thấy, ngoại trừ giá trị *nan* thì các giá trị của cột 'author' đang ở kiểu 'object', tên của các tác giả được đặt trong dấu ngoặc vuông [] và được phân tách với nhau bằng dấu phẩy. Tuy nhiên, một vài giá trị có xuất hiện các từ liên kết như *and* hay *et al.* và ta cần phải loại bỏ chúng trong quá trình xử lý. Sau khi trích xuất tên các tác giả, ta thực hiện chuẩn hóa bằng cách sử dụng hàm 'title()' để in hoa chữ cái đầu tiên của từng từ trong tên.

Code

```python
# Preprocessing function to clean author names
def preprocess_authors(author):
    if pd.isnull(author):  # Handling NaN values
        return np.nan

    # Remove unwanted characters, split multiple authors and normalize
    ↪ the names (title)
    author = re.sub(r'[\[\]]|et al.|\sand', '', author)
    cleaned_author = [name.strip().title() for name in
    ↪ re.split(r'\s*,+\s*', author) if name != '']

    return cleaned_author  # Convert to title case

# Apply the preprocessing function to the author data
df['author'] = [preprocess_authors(a) for a in df['author']]
print(df['author'])
```

Kết quả

```
0                    [Dr Dhaval Maheta]
1                                   NaN
2             [Oz Du Soleil, Bill Jelen]
3                     [William Mckinney]
4                       [Paul Mcfedries]
                      ...
825       [Michael Fullan, Joanne Quinn]
826    [Matthew F. Dixon, Igor Halperin]
827        [Yong Liu, Dr. Matei Zaharia]
```

```
828                                    NaN
829                      [Laurence Moroney]
Name: author, Length: 830, dtype: object
```

Giải thích:

1. Hàm preprocess_authors: Đây là hàm xử lý trước khi áp dụng cho dữ liệu trong cột 'author'. Hàm này thực hiện các bước sau:
   - Kiểm tra xem giá trị có phải là NaN không và trả về NaN nếu là vậy.
   - Sử dụng các biểu thức chính quy để loại bỏ các ký tự không mong muốn như dấu ngoặc vuông, từ 'et al.', 'and' trong dữ liệu tác giả.
   - Tách tác giả thành từng phần riêng biệt dựa trên dấu phẩy và loại bỏ khoảng trắng xung quanh.
   - Chuẩn hóa tên tác giả bằng cách viết hoa chữ cái đầu của từng từ trong tên tác giả.
2. Áp dụng hàm *preprocess_authors* vào cột 'author': Một vòng lặp for được sử dụng để áp dụng hàm preprocess_authors cho từng giá trị trong cột 'author'. Kết quả là một Series mới chứa danh sách các tác giả được xử lý theo cách mô tả trong hàm preprocess_authors.

Kết quả cuối cùng là cột 'author' mới chứa danh sách tác giả đã được xử lý, với mỗi tên tác giả được chuẩn hóa viết hoa chữ cái đầu của mỗi từ. Các giá trị NaN được duy trì nếu giá trị ban đầu là NaN.

Biến `dimensions`:

Ta in ra các giá trị trong cột *dimensions* để tiến hành phân tích định dạng của cột và xác định hướng xử lý.

```
array(['8.5 x 1.01 x 11 inches', '8 x 0.98 x 9.25 inches',
       '8.25 x 0.6 x 10.75 inches', '7 x 1.11 x 9.19 inches',
       '7.38 x 0.83 x 9.25 inches', nan, '6.75 x 0.75 x 8.75 inches',
       '8.5 x 0.92 x 11 inches', '6.75 x 0.75 x 9.75 inches',
       '7.38 x 1.5 x 9.25 inches', '6 x 0.9 x 9 inches',
       '7 x 1.25 x 9.19 inches', '6.69 x 2.44 x 2.4 inches; 7.37 Ounces',
       '9.3 x 8.2 x 3 inches; 1.8 Pounds', '7.5 x 1.34 x 9.25 inches',
       '8.19 x 5.67 x 4.02 inches; 1.5 Pounds'], ...
       dtype=object)
```

Ta có thể thấy định dạng chung của hầu hết các giá trị trong cột này là '<dài> x <rộng> x <cao> inches'. Tuy nhiên, cột này có xuất hiện các giá trị là khối lượng của sách ở cuối (ví dụ như '*9.3 x 8.2 x 3 inches; 1.8 Pounds*'...) và điều này có thể là sai sót trong quá trình thu thập dữ liệu. Như vậy,

bước đầu tiên ta cần trích xuất giá trị khối lượng sách từ cột *dimensions*, đưa sang cột *weight* và thực hiện xóa các giá trị khối lượng này trong cột *dimensions*.

Code

```
# Extract weight from the dimensions column and impute to the weight column
to_replace = df[df['dimensions'].str.contains(r'Pounds|Ounces', na=False)][['dimensions', 'weight']]
extract_weight = [preprocess_weight(wei) for wei in to_replace['dimensions'].str.extract(r';\s(.*)$')[0]]
df.loc[to_replace.index, 'weight'] = extract_weight

# Delete weights from dimensions column
df.loc[:, 'dimensions'] = df['dimensions'].str.replace(r';.+$', '', regex=True)
```

Tiếp theo, ta sẽ thực hiện trích xuất chiều dài, chiều rộng và chiều cao từ cột này, đổi đơn vị từ inches sang cm, cuối cùng thực hiện tính thể tích của cuốn sách với công thức *thể tích = chiều dài x chiều rộng x chiều cao*, sau đó ta đưa các giá trị thể tích này sang cột mới là *volume*.

Code

```
# Loại bỏ từ 'inches', tách dimensions thành length, width, height, đổi inches -> cm và tạo column 'volume'
def preprocess_dimensions(dimension):
    try:
        dimension = re.sub(r'inches', '', dimension)
        # Chuyển thành kiểu float và chuyển inches -> cm
        length, width, height = [float(num) * 2.54 for num in re.split(r'x', dimension)]
        volume = length * width * height
        return volume
    except:
        return np.nan  # Return NaN for non-convertible values

df['volume'] = [preprocess_dimensions(dim) for dim in df['dimensions']]
```

Kết quả

```
0      1547.512389
1      1188.389881
2      871.996643
3      1170.139608
4      928.490227
          ...
825    539.134406
826    1158.348296
827    738.954167
828    NaN
829    NaN
Name: volume, Length: 830, dtype: float64
```

Giải thích: Như vậy, ta có thể thấy cột 'volume' đã chứa thể tích của mỗi cuốn sách sau khi đã được tính toán dựa trên kích thước chiều dài, rộng và cao được chuyển từ inches sang cm và tính toán thể tích theo công thức hình hộp. Các giá trị không thể tính được sẽ được thay thế bằng NaN.

Biến `language`

Ta in ra màn hình các giá trị độc nhất (unique values) trong cột *language*.

```
array(['English', nan, 'Spanish',
       'Unqualified, Japanese (Dolby Digital 2.0 Mono), English (Dolby
       ↪  Digital 5.1), English (Dolby Digital 2.0 Mono)',
       'you will discover all you need ',
       '• How to make better business decisions using ',
       'Concepts are presented in a "to-the-point" style to cater to the
       ↪  busy individual. With this book, you can learn Python in just
       ↪  one day and start coding immediately. ',
       'standard library',
       'This Python programming guide assumes certain level of programming
       ↪  knowledge. It is not a beginner textbook.',
       'Scroll to the top of the page and click the ',
       'English (Dolby Digital 2.0 Mono)',
       'English (DTS-HD Master Audio 5.1), French (DTS-HD 2.0)',
       '"Brilliant."'], dtype=object)
```

Ta có thể thấy ngoài các giá trị là ngôn ngữ (như *English*, *Spanish*...) thì cột này xuất hiện khá nhiều giá trị nhiễu (như *standard library*, *"Brilliant."*...). Ngoài ra, có rất nhiều giá trị thể hiện ngôn ngữ của cuốn sách, tuy nhiên nó lại đi kèm với các phiên bản khác nhau của cuốn sách đó (*English (Dolby Digital 2.0 Mono)*, *English (DTS-HD Master Audio 5.1), French (DTS-HD 2.0)*...). Ta sẽ đưa các giá trị nhiễu về cùng một giá trị là *Unknown*, sau đó thực hiện trích xuất ngôn ngữ của cuốn sách đối với các giá trị có nhiều phiên bản.

Code

```
def preprocess_language(lang):
    try:
        lang = re.findall(r'English|Japanese|French|Spanish', lang)
        lang = set(lang) if lang[0] != '' else np.nan
        lang = ', '.join(lang)
        return lang
    except:
        return np.nan  # Return NaN for non-convertible values
df['language'] = [preprocess_language(lang) for lang in df['language']]
print(df['language'].unique())
```

Kết quả

```
['English' nan 'Spanish' 'Japanese, English' 'French, English']
```

Giải thích:

Trong đoạn mã trên, quá trình tiền xử lý dữ liệu cho cột *language* được thực hiện như sau:

- Loại bỏ giá trị nhiễu và xác định ngôn ngữ.
- Sử dụng biểu thức chính quy (regular expression) để trích xuất các ngôn ngữ phổ biến như 'English', 'Japanese', 'French', 'Spanish' từ các giá trị có trong cột 'language'.
- Nếu không có bất kỳ ngôn ngữ nào được trích xuất từ giá trị ban đầu, hoặc có giá trị không xác định, nó sẽ trả về *NaN*.
- Nếu có ngôn ngữ được trích xuất, chúng sẽ được gộp lại thành một chuỗi duy nhất sử dụng set để loại bỏ trùng lặp và sau đó đưa về chuỗi thông thường sử dụng 'join'.

Kết quả đầu ra là một mảng chứa các giá trị ngôn ngữ đã được xử lý. Mỗi giá trị đại diện cho ngôn ngữ hoặc một kết hợp ngôn ngữ nhất định (nếu có) từ dữ liệu ban đầu. Ví dụ, trong kết quả ta có ['English', nan, 'Spanish', 'Japanese, English', 'French, English']:

- 'English': Đại diện cho sách viết bằng tiếng Anh.

- 'Spanish': Đại diện cho sách viết bằng tiếng Tây Ban Nha.
- 'Japanese, English': Đại diện cho sách viết bằng tiếng Nhật và tiếng Anh.
- 'French, English': Đại diện cho sách viết bằng tiếng Pháp và tiếng Anh.

Kết quả này đã loại bỏ các giá trị nhiễu và tạo ra một chuỗi ngôn ngữ rõ ràng và dễ hiểu từ dữ liệu ban đầu.

Biến `publisher`

Ta tiếp tục thực hiện in ra màn hình các giá trị độc nhất (unique values) trong cột *publisher* để phân tích các kiểu định dạng và xác định hướng xử lý đối với dữ liệu.

```
array(['Notion Press Media Pvt Ltd (November 22, 2021)',
       "O'Reilly Media; 1st edition (August 18, 2009)",
       'Holy Macro! Books; Third edition (August 1, 2022)',
       "O'Reilly Media; 2nd edition (November 14, 2017)",
       'For Dummies; 5th edition (February 3, 2022)', nan,
       "O'Reilly Media; 1st edition (October 5, 2021)",
       'SAGE Publications, Inc; 4th edition (January 22, 2019)',
       'Cambridge University Press; 1st edition (March 24, 2022)',
       'Manning; 3rd edition (May 3, 2022)',
       'University of Chicago Press (October 15, 2021)'
```

Từ đây ta có thể thấy định dạng của mỗi giá trị trong cột *publisher* là '<publisher>; <edition> (time)'. Ta sẽ thực hiện tách tên nhà xuất bản (publisher), số lần tái bản (edition) và thời gian phát hành (time) thành các cột riêng biệt.

Code:

```python
def preprocess_time(publisher):
    try:
        time = re.findall(r'.*\(?.*\)?.*\((.+)\)$', publisher)
        return time[0].strip()
    except:
        return np.nan

def preprocess_edition(publisher):
    try:
        edition = re.findall(r';(.+)edition|ed\.', publisher)
        return edition[0].strip()
```

```
    except:
        return np.nan

def preprocess_publisher(publisher):
    try:
        publisher = [p.strip() for p in re.findall(r'^(.*);|^(.*)\(',
        ↪  publisher)[0] if p != '']
        return publisher[0].strip()
    except:
        return np.nan

df['time'] = [preprocess_time(publisher) for publisher in df['publisher']]
df['edition'] = [preprocess_edition(publisher) for publisher in
↪  df['publisher']]
df['publisher'] = [preprocess_publisher(publisher) for publisher in
↪  df['publisher']]
```

Giải thích: Ta thực hiện các công việc sau:

1. Xử lý tách thông tin
   - Tạo ba hàm riêng biệt (preprocess_time, preprocess_edition, preprocess_publisher) để trích xuất thông tin về thời gian, phiên bản và tên nhà xuất bản từ các giá trị trong cột "publisher".
   - Mỗi hàm sử dụng biểu thức chính quy để trích xuất thông tin tương ứng và xử lý nếu có bất kỳ ngoại lệ nào xảy ra.
2. Áp dụng các hàm xử lý:
   - Hàm preprocess_time trích xuất thông tin về thời gian từ mỗi giá trị trong cột "publisher"và gán kết quả vào cột "time".
   - Hàm preprocess_edition trích xuất thông tin về phiên bản từ mỗi giá trị trong cột "publisher"và gán kết quả vào cột "edition".
   - Hàm preprocess_publisher trích xuất thông tin về tên nhà xuất bản từ mỗi giá trị trong cột "publisher"và gán kết quả vào cột "publisher".

Kết quả sẽ là ba cột mới "time", "edition", và "publisher", mỗi cột chứa thông tin đã được xử lý tương ứng từ cột gốc "publisher".

Tiếp theo, ta thực hiện tách năm xuất bản (publish_year) từ cột *time*

Code

```python
def preprocess_year(time):
    try:
        year = int(re.findall(r'(\d+)$', time)[0])
        return year
    except:
        return np.nan

df['publish_year'] = [preprocess_year(time) for time in df['time']]
```

Kết quả

```
0      2021
1      2009
2      2022
3      2017
4      2022
       ...
825    2017
826    2020
827    2022
828    NaN
829    NaN
Name: publish_year, Length: 830, dtype: float64
```

Giải thích: Ta có thể thấy cột "publish_year"chứa các giá trị năm xuất bản được trích xuất từ cột "time", với các giá trị đã được chuyển thành kiểu dữ liệu số nguyên hoặc NaN nếu không thể tìm thấy năm trong chuỗi.

Biến `ISBN_13`

Tiếp theo, ta thực hiện gom các dữ liệu bị nhiễu trong cột "ISBN-13"thành một giá trị là *NaN*.

Code

```python
# Validating ISBN-13 format
invalid_isbn = df[~df['ISBN_13'].str.contains(r'^\d{3}-\d{10}$',
 ↪  na=False)]['ISBN_13']
df.loc[invalid_isbn.index, 'ISBN_13'] = np.nan
```

Cuối cùng, ta thực hiện loại bỏ các cột không cần thiết:

Code

```
drop_columns = ['price (including used books)', 'dimensions', 'ISBN_13',
↪  'link', 'complete_link', 'time', 'edition']
df.drop(columns=drop_columns, inplace=True)
```

Tình trạng bộ dữ liệu sau khi xử lý định dạng:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 830 entries, 0 to 829
Data columns (total 19 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   title          830 non-null    object
 1   author         657 non-null    object
 2   price          722 non-null    float64
 3   pages          737 non-null    Int64
 4   avg_reviews    702 non-null    float64
 5   n_reviews      702 non-null    Int64
 6   star5          702 non-null    float64
 7   star4          635 non-null    float64
 8   star3          554 non-null    float64
 9   star2          451 non-null    float64
 10  star1          328 non-null    float64
 11  weight         660 non-null    float64
 12  language       752 non-null    object
 13  publisher      713 non-null    object
 14  min_price      722 non-null    float64
 15  volume         644 non-null    float64
 16  publish_date   713 non-null    float64
 17  publish_month  713 non-null    object
 18  publish_year   713 non-null    float64
dtypes: Int64(2), float64(12), object(5)
memory usage: 124.9+ KB
```

3.3.2 Xử lý dữ liệu bị thiếu (Missing values)

Bước 1: Xóa những dòng có số lượng giá trị bị thiếu quá lớn

```
# Xóa những dòng có hơn 60% các giá trị bị thiếu
threshold = 0.4
df.dropna(thresh=df.shape[1] * threshold, inplace=True, axis=0)
```

Giải thích:

1. Thiết lập ngưỡng:
   - Ngưỡng được đặt bằng biến threshold, trong đoạn mã là 0.4, tương đương với 40%.
   - Ngưỡng này quyết định tỷ lệ tối thiểu số giá trị không bị thiếu cho mỗi dòng trong DataFrame mà chúng được chấp nhận không bị xóa.
2. Xóa dòng có tỷ lệ giá trị bị thiếu vượt quá ngưỡng:
   - "df.dropna()"dùng để loại bỏ các dòng chứa giá trị NaN.
   - "thresh=df.shape[1] * threshold"thiết lập ngưỡng cho số lượng giá trị không bị thiếu cho mỗi dòng, được tính bằng cách nhân số cột của DataFrame với ngưỡng. Tất cả những dòng có số lượng giá trị không bị thiếu ít hơn 40% sẽ bị loại bỏ.
   - "inplace=True"để áp dụng thay đổi trực tiếp lên DataFrame hiện tại, không cần phải gán kết quả vào một DataFrame mới.
   - "axis=0"chỉ định xóa dòng chứa giá trị NaN.

Kết quả là các dòng trong DataFrame mà có tỷ lệ giá trị bị thiếu (NaN) vượt quá 40% đã được loại bỏ, giúp làm sạch dữ liệu và giảm thiểu ảnh hưởng của dữ liệu bị thiếu lên quá trình phân tích và xử lý sau này.

Bước 2: Xác định các cột có giá trị bị thiếu và in ra tên cột cũng như số giá trị bị thiếu trong cột.

```
# Xác định các cột có giá trị bị thiếu
columns_with_missing = df.columns[df.isnull().any()]
missing_counts = df[columns_with_missing].isnull().sum()
print("Các cột có dữ liệu bị thiếu:")
print(missing_counts)
```

Kết quả

```
Các cột có dữ liệu bị thiếu:
author          149
price           64
```

```
pages           56
avg_reviews     85
n_reviews       85
star5           85
star4           151
star3           231
star2           334
star1           457
weight          125
language        42
publisher       73
min_price       64
volume          141
publish_year    73
dtype: int64
```

Bước 3: Điền giá trị "Unknown"cho các giá trị bị thiếu đối với các cột có kiểu dữ liệu "object"

```
unknown_cols = ['author', 'language', 'publisher', 'publish_year']
for col in unknown_cols:
    df[col].fillna(value='Unknown', inplace=True)
df.isna().sum()
```

Bước 4: Điền giá trị trung vị cho các giá trị bị thiếu đối với các cột có kiểu dữ liệu *float*

```
median_cols = ['price', 'avg_reviews', 'weight', 'min_price', 'volume']
for col in median_cols:
    df[col].fillna(value=df[col].median(), inplace=True)
df.isna().sum()
```

Bước 5: Điền giá trị trung vị cho các giá trị bị thiếu đối với các cột có kiểu dữ liệu *int*

```
int_median_cols = ['pages', 'n_reviews']
for col in int_median_cols:
    df[col].fillna(value=int(df[col].median()), inplace=True)
df.isna().sum()
```

Bước 6: Điền giá trị "0"cho các giá trị bị thiếu đối với các cột *star*

```
star_columns = ['star5', 'star4', 'star3', 'star2', 'star1']
for col in star_columns:
    df[col].fillna(value=0, inplace=True)
df.isna().sum()
```

Tình trạng của bộ dữ liệu sau khi hoàn tất tiền xử lý:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 785 entries, 0 to 829
Data columns (total 17 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   title         785 non-null    object
 1   author        785 non-null    object
 2   price         785 non-null    float64
 3   pages         785 non-null    Int64
 4   avg_reviews   785 non-null    float64
 5   n_reviews     785 non-null    Int64
 6   star5         785 non-null    float64
 7   star4         785 non-null    float64
 8   star3         785 non-null    float64
 9   star2         785 non-null    float64
 10  star1         785 non-null    float64
 11  weight        785 non-null    float64
 12  language      785 non-null    object
 13  publisher     785 non-null    object
 14  min_price     785 non-null    float64
 15  volume        785 non-null    float64
 16  publish_year  785 non-null    object
dtypes: Int64(2), float64(10), object(5)
memory usage: 111.9+ KB
```

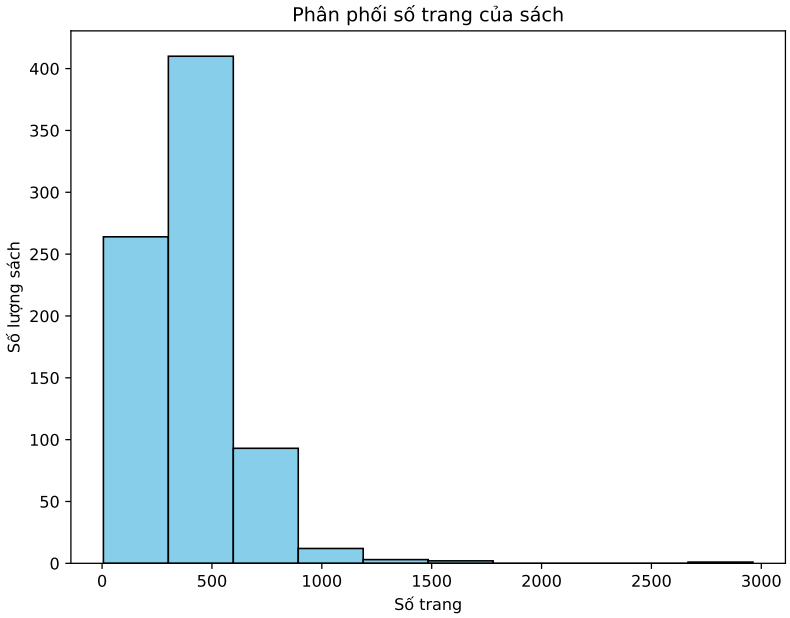
4 Phân tích bộ dữ liệu sau tiền xử lý

4.1 Phân tích đơn biến

4.1.1 Phân tích số trang (pages)

Biểu diễn bằng biểu đồ histogram để thể hiện sự phân phối của số trang của những quyển sách trong bộ dữ liệu.

```
# Biểu đồ phân phối số trang
plt.figure(figsize=(8, 6))
plt.hist(df['pages'], bins=10, color='skyblue', edgecolor='black')
plt.title('Phân phối số trang của sách')
plt.xlabel('Số trang')
plt.ylabel('Số lượng sách')
plt.show()
```

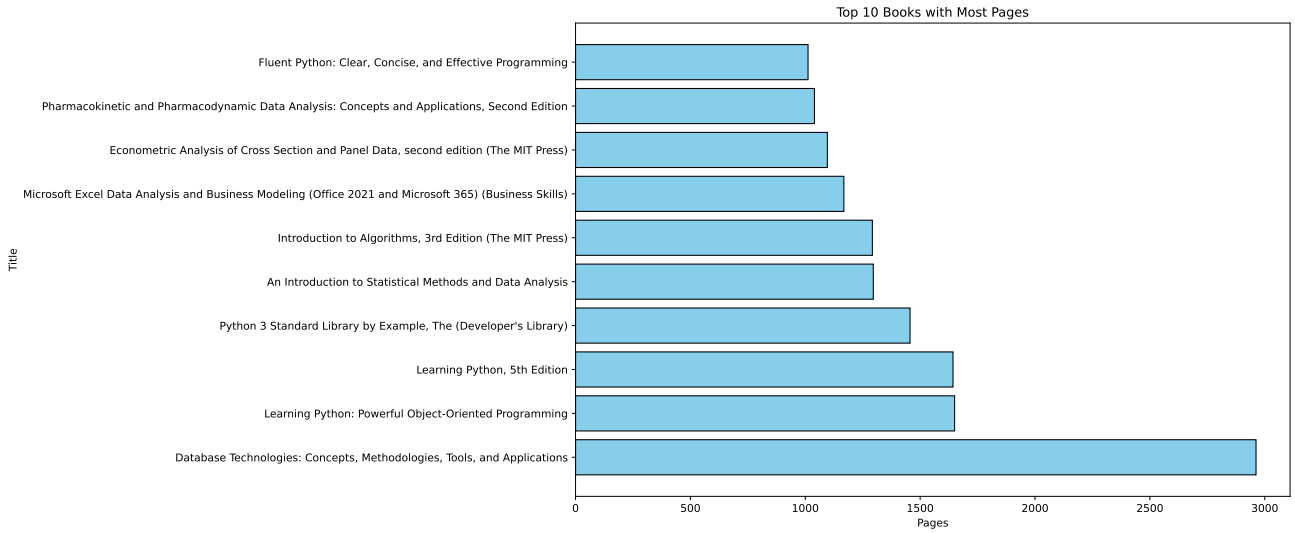


Nhận xét: Biểu đồ cho thấy hầu hết các quyển sách trong bộ dữ liệu có số trang dưới 1000 trang, đây là khoảng phân bố chính của số trang sách. Phần lớn thì các quyển sách sẽ có khoảng trên dưới 500 trang.

Ngoài ra còn có những quyển sách đặc biệt với số trang nhiều hơn hẳn những quyển khác. Để hiểu rõ hơn những quyển sách đặc biệt này có chủ đề là gì, tiếp theo ta cho in ra thông tin chi tiết những quyển sách dày nhất.

```
pd.set_option('display.max_colwidth', True)
# Chọn top 10 sách có nhiều trang nhất dựa trên cột 'pages'
books_most_pages = df.nlargest(10, 'pages')
# In ra kết quả
books_most_pages[['title', 'author', 'pages']]

# Vẽ biểu đồ
plt.figure(figsize=(12, 8))
plt.barh(books_most_pages['title'], books_most_pages['pages'],
    color='skyblue', edgecolor='black')
plt.xlabel('Pages')
plt.ylabel('Title')
plt.title('Top 10 Books with Most Pages')
plt.show()
```

Kết quả:



Nhận xét: Nhìn chung, top 10 sách có nhiều trang nhất phản ánh nhu cầu cao về kiến thức chuyên sâu trong việc học ngôn ngữ lập trình thông dụng nhất trong ngành phân tích dữ liệu là Python (có 4 trên 10 quyển). Ngoài ra ta cũng có thể thấy rằng trong đó còn có những quyển sách tham khảo chuyên ngành chẳng hạn như "Econometric Analysis of Cross Section and Panel Data, second edition (The MIT Press)" ...

4.1.2 Phân tích ngôn ngữ (language)

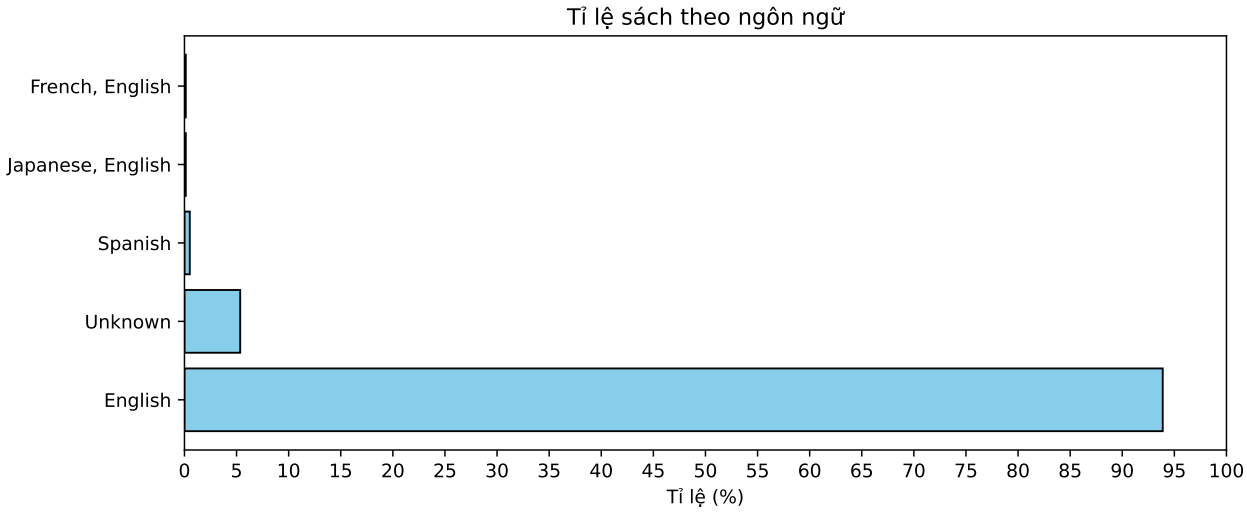
Đầu tiên chúng ta sẽ tính tỷ lệ các ngôn ngữ viết sách có trong bộ dữ liệu này.

```
# Tính tỷ lệ sách theo ngôn ngữ
language_counts = df['language'].value_counts()
total_books = len(df)
language_percentages = (language_counts / total_books) * 100

# Hiển thị thông tin tỷ lệ theo ngôn ngữ
print('Tỉ lệ sách theo ngôn ngữ:')
print(language_percentages)

# Tạo biểu đồ cột
plt.figure(figsize=(10, 4))
plt.barh(language_percentages.index, language_percentages,
 ↪  color='skyblue', edgecolor='black')
plt.xlabel('Tỉ lệ (%)')
plt.xticks(range(0, 101, 5))
plt.title('Tỉ lệ sách theo ngôn ngữ')
plt.show()
```

Kết quả

```
Tỉ lệ sách theo ngôn ngữ:
English              93.885350
Unknown              5.350318
Spanish              0.509554
Japanese, English    0.127389
French, English      0.127389
Name: language, dtype: float64
```

Nhận xét: Hầu hết những quyển sách trong bộ dữ liệu này đều được viết bằng tiếng Anh (hơn 90%). Qua đó cho thấy tiếng Anh là ngôn ngữ được sử dụng rộng rãi và phổ biến nhất trong ngành Khoa học dữ liệu này.

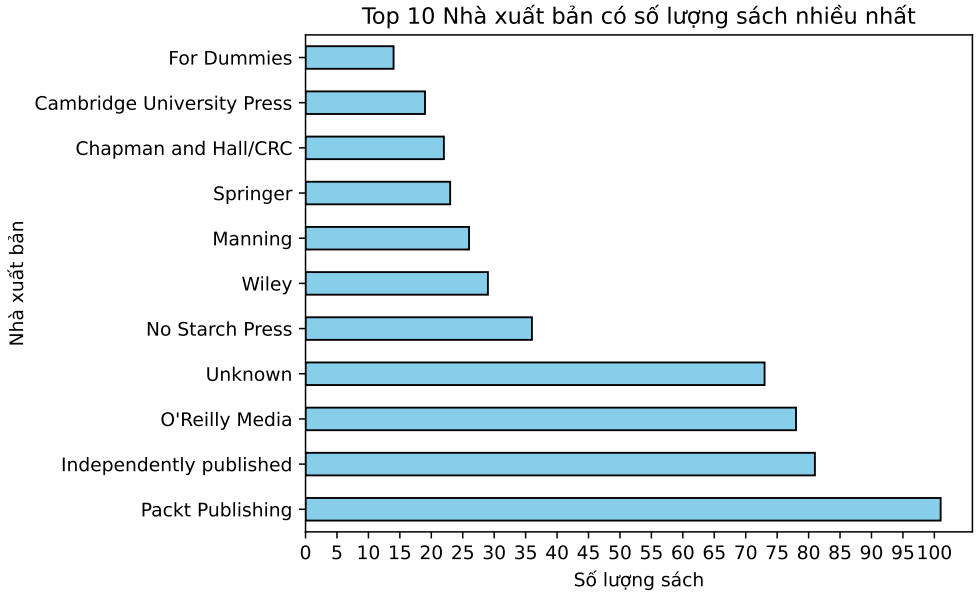


4.1.3 Phân tích nhà xuất bản (publisher)

Một trong những yếu tố để chọn được quyển sách tốt là nhà xuất bản phải uy tín và có danh tiếng. Bởi vì họ thường có chất lượng sách tốt, sách được biên tập và kiểm duyệt kỹ lưỡng hơn hẳn những nhà xuất bản nhỏ.

```
# Phân tích nhà xuất bản:
# Đếm số sách từng nhà xuất bản
publisher_counts= df['publisher'].value_counts()
# Tạo DataFrame mới từ Series 'publisher_data'
publisher_df = publisher_counts.reset_index()
publisher_df.columns = ['publisher', 'count']
publisher_df.head(11)

# Tạo biểu đồ
plt.figure(figsize=(10, 10))
publisher_df[0:11].plot(kind='barh', x='publisher', y='count',
 ↪  color='skyblue', edgecolor='black', legend=False)
plt.xlabel('Số lượng sách')
plt.xticks(range(0, max(publisher_df['count']) + 1, 5))
plt.ylabel('Nhà xuất bản')
plt.yticks(rotation=0)
plt.title('Top 10 Nhà xuất bản có số lượng sách nhiều nhất')
plt.show()
```

Kết quả:



Nhận xét: Biểu đồ cho thấy sự chênh lệch lớn về số lượng sách giữa các nhà xuất bản. Điều này có thể phản ánh sự khác biệt về quy mô, chủ đề mà các nhà xuất bản tập trung, hoặc chiến lược xuất bản của họ. Packt Publishing là nhà xuất bản có số lượng sách nhiều nhất, tiếp theo là Independently published và O'Reilly Media.

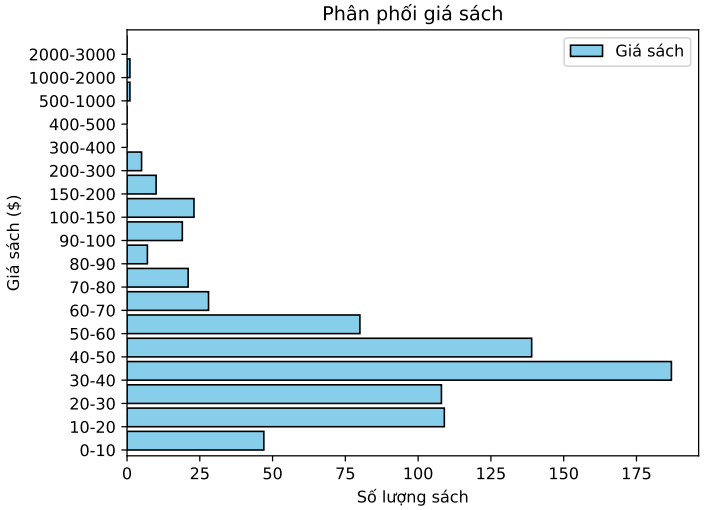
4.1.4 Phân tích giá cả (price)

Phân tích giá cả giúp độc giả xác định được mức giá hợp lý cho sách dự định mua, từ đó đưa đưa ra quyết định tiết kiệm hơn. Ngoài ra chúng ta cũng có thể nhìn thấy xu hướng và nhu cầu của thị trường về các dòng sách dành cho ngành Khoa học dữ liệu.

```
col = ['price']
price_data = df[col]

# Gom nhóm giá sách vào các khoảng
price_bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 300, 400, 500, 1000, 2000, 3000]
price_labels = [f'{i}-{j}' for i, j in zip(price_bins[:-1], price_bins[1:])]
price_data['price_group'] = pd.cut(price_data['price'], bins=price_bins, labels=price_labels, right=False)

# Tạo histogram cho phân phối giá sách
```

```
price_data['price_group'].value_counts().sort_index().plot(kind='barh',
    color='skyblue', edgecolor='black', width=0.8, position=0, label='Giá
    sách')
plt.xlabel('Số lượng sách')
plt.ylabel('Giá sách ($)')
plt.title('Phân phối giá sách')
plt.legend()
plt.show()
```

Kết quả:



Nhận xét: Biểu đồ cho thấy phạm vi sách có thể trải dài từ $10 đến $3000. Phần lớn sách có giá dưới $70 đô và số lượng sách nhiều nhất có giá từ $20 đến $50. Điều này cho thấy xu hướng giá sách của ngành Khoa học dữ liệu tập trung ở các mức giá thấp.

4.2 Phân tích đa biến

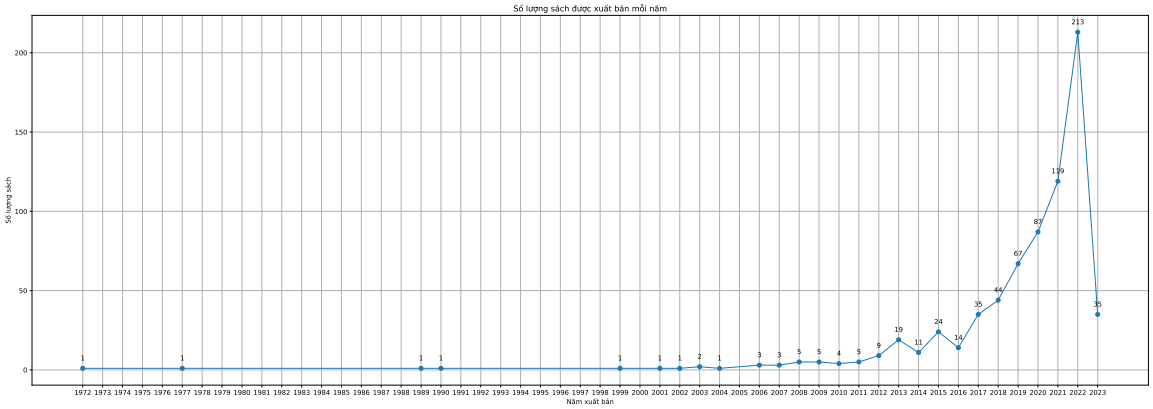
4.2.1 Phân tích năm xuất bản (publish_year)

Phân tích số sách về data science được xuất bản mỗi năm giúp chúng ta nắm bắt được sự phát triển và thay đổi của lĩnh vực này theo thời gian.

```
book_counts = df.groupby('publish_year')['title'].count()
print('Số lượng sách được xuất bản mỗi năm là: ')

# Vẽ biểu đồ đường
years = list(range(1972, 2024, 1))
```

```
plt.figure(figsize=(30, 10))
plt.plot(book_counts.index, book_counts.values, marker='o')
plt.xlabel('Năm xuất bản')
plt.xticks(years)
plt.ylabel('Số lượng sách')
plt.title('Số lượng sách được xuất bản mỗi năm')
plt.grid(True)
# Hiển thị số lượng sách từng năm trên mỗi chấm
for i, v in enumerate(book_counts.values):
    plt.text(book_counts.index[i], v + 5, str(v), ha='center')
plt.show()
```

Kết quả:



Nhận xét: Biểu đồ cho thấy ở thập niên 90 và những năm 2000 sự xuất bản sách về ngành Khoa học dữ liệu cực kì ít. Sau đó theo xu hướng phát triển của internet, việc xuất bản sách tăng đột biến về số lượng, nhất là ở khoảng thời gian từ 2017 đến 2023. Năm 2022 có số lượng sách được xuất bản nhiều nhất với 213 quyển.

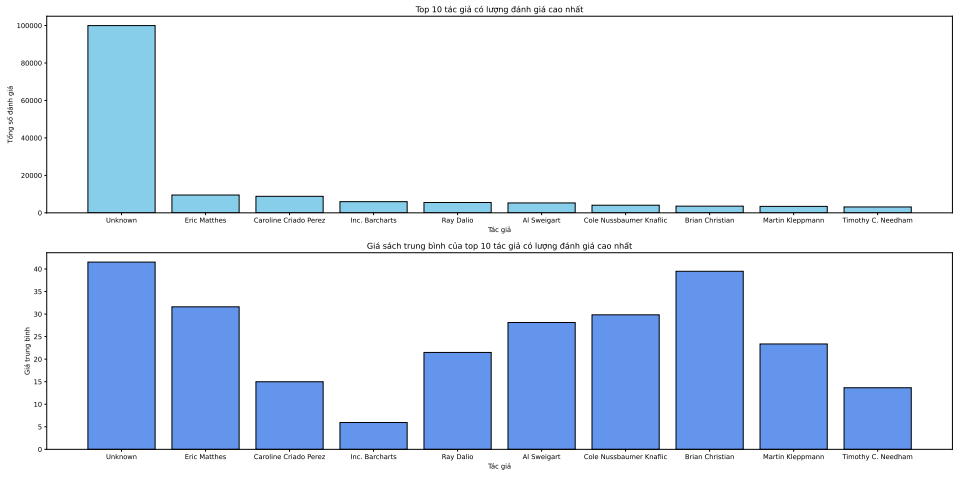
4.2.2 Phân tích tác giả (author)

Lý do của việc phân tích này vì các tác giả phổ biến thường được độc giả yêu thích và có những yếu tố đặc biệt như chất lượng sách tốt hay giá sách lại rẻ...

```
# Chuyển đổi danh sách tác giả thành chuỗi
df['author'] = df['author'].apply(lambda x: x[0] if isinstance(x, list)
 ↪  else x)

# Tính tổng số đánh giá và giá sách trung bình cho mỗi tác giả
```

```
author_analysis = df.groupby('author').agg(
    total_reviews=('n_reviews', 'sum'),
    average_price=('price', 'mean')
).reset_index()

# Sắp xếp theo tổng số đánh giá giảm dần
author_analysis = author_analysis.sort_values(by='total_reviews',
↪   ascending=False)

# Hiển thị thông tin
author_analysis.head(10)
```

Kết quả:



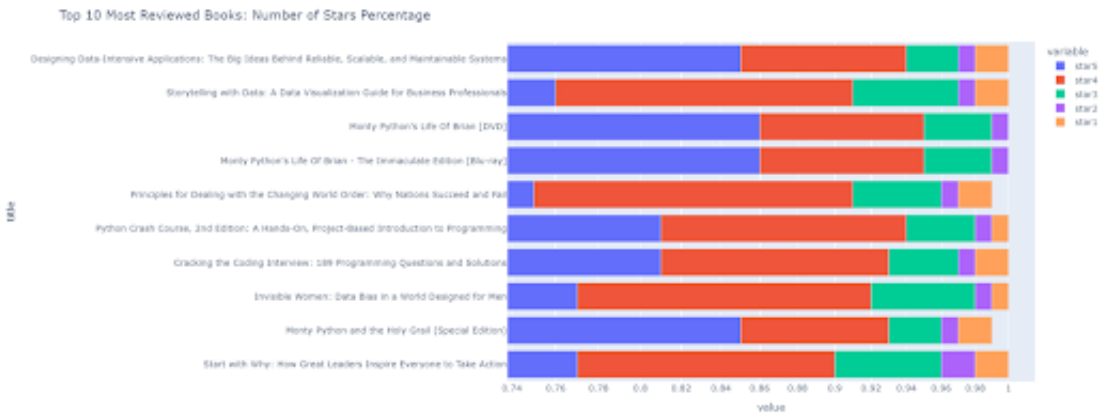
Nhận xét: Biểu đồ cho thấy ngoài những tác giả không được biết thì Eric Matthes là tác giả có nhiều lượt đánh giá nhất, nghĩa là sách của người này cũng có thể được mua nhiều nhất, tiếp đến là tác giả Caroline Criado Perez và Inc. Barcharts. Thêm vào đó khi nhìn vào giá sách trung bình có thể thấy, giá sách của các tác giả nằm trong top 10 này đều có mức giá thấp rơi vào khoảng dưới $40, và nằm ở mức phổ biến như phân tích phân phối giá ở trên.

4.2.3 Phân tích đánh giá (n_reviews vs avg_reviews)

Xếp hạng sách dựa trên số lượng đánh giá và trung bình đánh giá cao nhất để đưa ra gợi ý mua sắm cho độc giả:

```python
# Chọn các sách có số lượng đánh giá (n_reviews) và đánh giá trung bình (avg_reviews) cao
selected_books = df[(df['n_reviews'] > df['n_reviews'].mean()) & (df['avg_reviews'] > df['avg_reviews'].mean())]
# Sắp xếp kết quả theo thứ tự giảm dần của n_reviews
high_review = selected_books.sort_values(by='n_reviews', ascending=False)

print("Các sách có số n_reviews và avg_reviews cao, sắp xếp theo n_reviews giảm dần: ")
high_review[['title', 'author', 'n_reviews', 'avg_reviews']].head(10)

# Tạo DataFrame most_reviewed_books từ DataFrame high_review
most_reviewed_books = high_review.nlargest(10, 'n_reviews')
# Tạo biểu đồ bar
fig = px.bar(most_reviewed_books, y='title', x=['star5', 'star4', 'star3', 'star2', 'star1'], log_x=True, orientation='h')
fig.update_layout(
    width=1000,
    height=400,
    title="Top 10 sách có review cao nhất và phần trăm đánh giá của nó"
)
fig.show()
```

Kết quả:



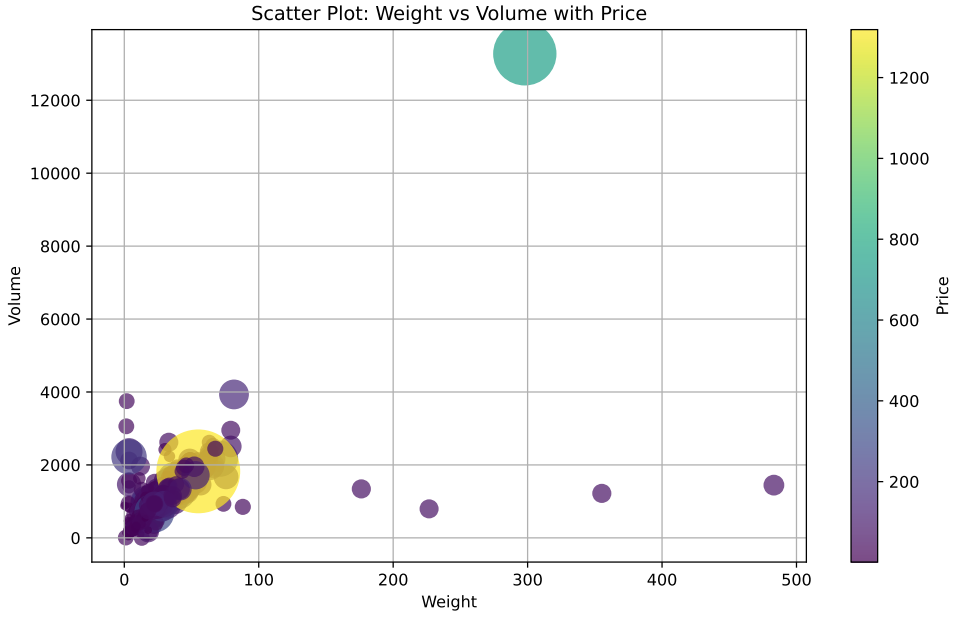
Nhận xét: Nhìn chung cả 10 quyển sách đều được đánh giá cao, với tỷ lệ phần trăm số lượng đánh giá 4 sao là cao nhất, kế đến là 5 sao. Bên cạnh đó những đánh giá 1, 2 sao chiếm tỷ lệ rất nhỏ so

với tổng thể. Với lượng đánh giá lên đến hơn 3000 đánh giá cho mỗi quyển sách nhưng chúng vẫn có đánh giá trung bình là hơn 4,6 sao. Điều này cho thấy những quyển sách này được độc giả yêu thích và đánh giá cao về chất lượng.

4.2.4 Phân tích thể tích và trọng lượng (volume vs weight)

```
# Vẽ biểu đồ scatter plot với kích thước điểm thay đổi theo giá sách
plt.figure(figsize=(10, 6))
scatter = plt.scatter(df['weight'], df['volume'], c=df['price'],
 ↪  cmap='viridis', alpha=0.7, s=df['price']*2)
plt.title('Scatter Plot: Weight vs Volume with Price')
plt.xlabel('Weight')
plt.ylabel('Volume')
plt.colorbar(scatter, label='Price')
plt.grid(True)
plt.savefig('weight-volume.pdf', bbox_inches='tight')
# plt.show()
```

Kết quả:



Nhận xét: Biểu đồ cho thấy xu hướng chung là sách có trọng lượng và thể tích nhỏ hơn thì có giá thấp hơn. Đa phần các quyển sách có thể tích dưới 4000 inches và trọng lượng dưới 100 pounds thì có giá dưới $200. Tuy nhiên cũng có 1 số điểm ngoại lệ, chẳng hạn như với thể tích và trọng lượng như vừa nhắc nhưng giá thành lại lên đến tầm $1200. Điều này có thể là do những yếu tố khác ví

dụ như chất liệu giấy và bìa sách, nội dung được đầu tư về mặt thiết kế mang tính thẩm mỹ... Và ngược lại, một số điểm ngoại lệ có trọng lượng lớn nhưng thể tích và giá cũng gần như ngang bằng xu hướng chung, có lẽ vì chất liệu giấy mỏng, rẻ tiền hơn...

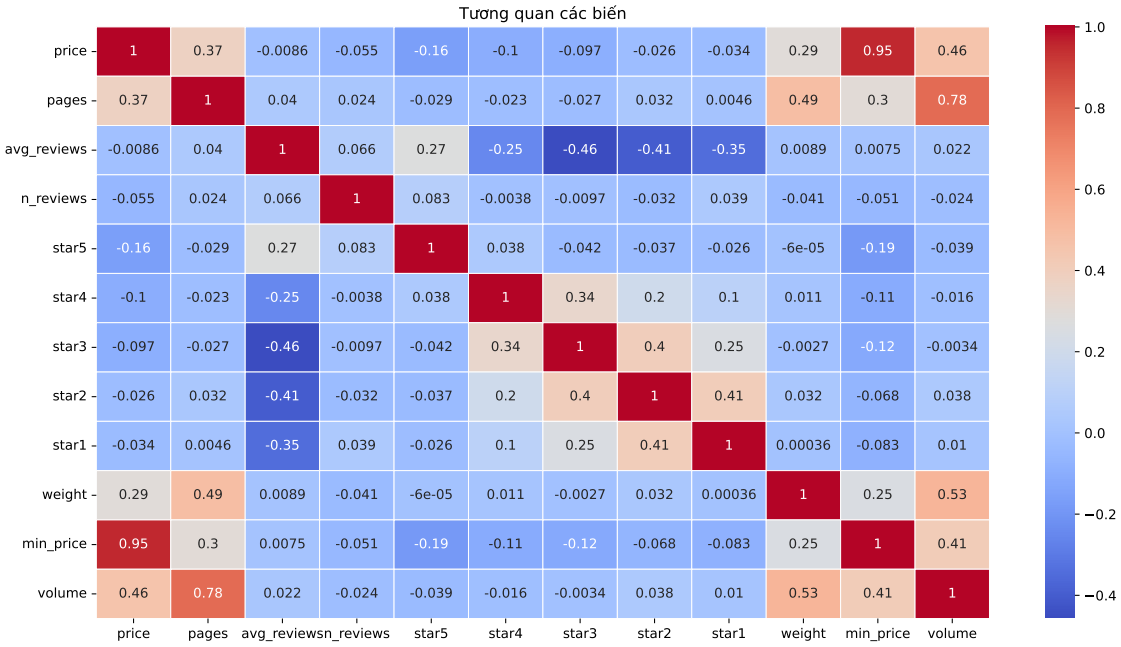
4.2.5 Phân tích tương quan

a. Phân tích sự tương quan của tất cả các biến

Tạo biểu đồ heatmap để mô tả các mối tương quan giữa tất cả các biến hiện tại:

```
# Tính toán ma trận tương quan
correlation_matrix = df.corr()
# Tạo biểu đồ heatmap
plt.figure(figsize=(15, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
 ↪  linewidths=.5)
plt.title('Tương quan các biến')
plt.show()
```

Kết quả:



Nhận xét: Theo biểu đồ cho thấy dường như các biến ít có sự tương quan dương mạnh, đa phần đều là tương quan âm và không rõ ràng. Ở đây đặc biệt có biến price và min_price có chỉ số tương quan cực cao là bởi vì min_price là giá sách có bao gồm sách cũ. Các biến có tương quan dương mạnh

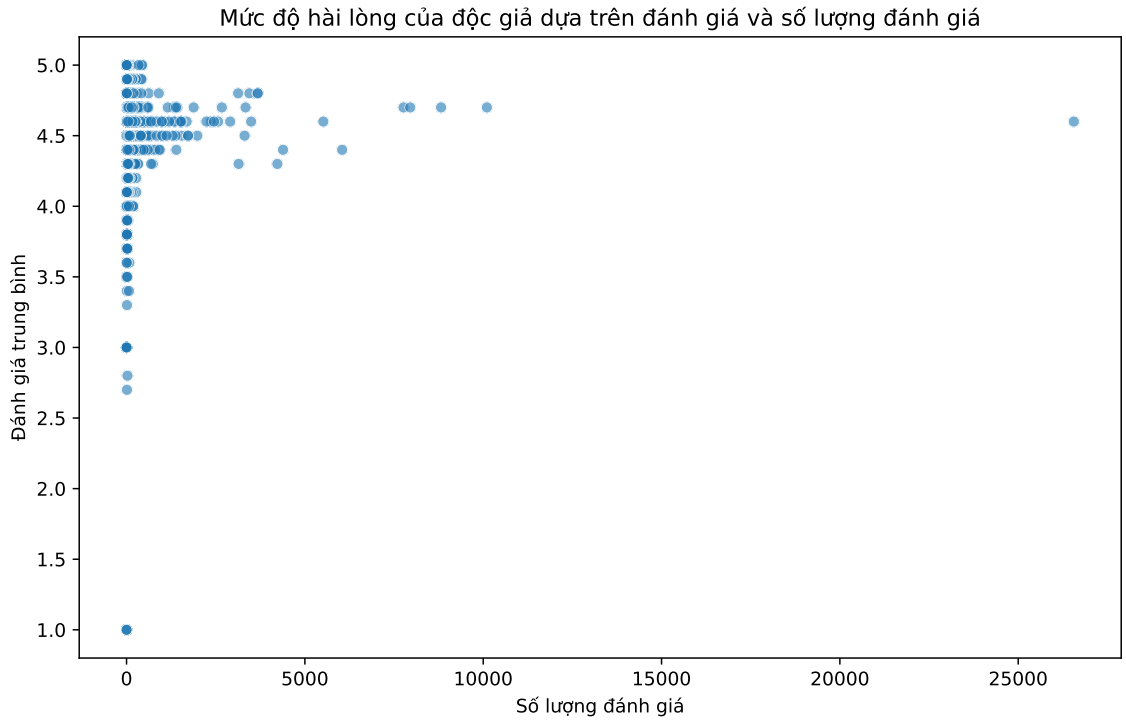
tiếp theo là pages và volume với 0.78, weight và volume với 0.53. Điều này có nghĩa là số trang và trọng lượng càng lớn thì thể tích cũng sẽ càng tăng.

b. Phân tích mức độ hài lòng của độc giả

Dùng biểu đồ scatter plot để mô tả mối quan hệ giữa số lượng đánh giá (n_reviews) và đánh giá trung bình (avg_reviews) của những quyển sách.

```
# Tổng quan về mức độ hài lòng của độc giả dựa trên đánh giá và số lượng đánh giá.
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='n_reviews', y='avg_reviews', alpha=0.6)
plt.title('Mức độ hài lòng của độc giả dựa trên đánh giá và số lượng đánh giá')
plt.xlabel('Số lượng đánh giá')
plt.ylabel('Đánh giá trung bình')
plt.show()
```

Kết quả:



Nhận xét: Biểu đồ cho thấy sự phân bố dữ liệu tập trung ở phần trên bên trái của biểu đồ. Điều này cho thấy rằng phần lớn các sản phẩm có số lượng đánh giá ít nhưng trung bình đánh giá lại cao. Để đánh giá đây có phải là quyển sách hay hay không trong trường hợp này còn phải thêm dữ liệu

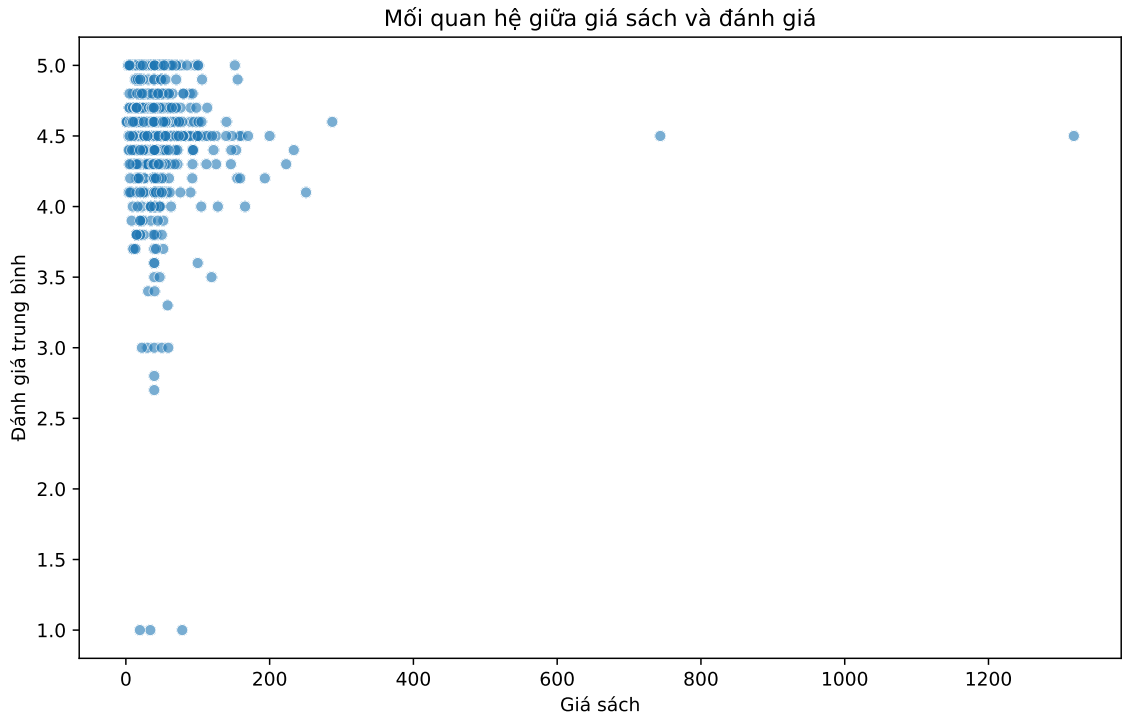
và phân tích chi tiết hơn. Tuy nhiên có một số ngoại lệ có số lượng đánh giá rất cao cùng với đó là trung bình đánh giá cũng cao. Điều này chứng tỏ đây là những quyển sách hay, được rất nhiều người trải nghiệm qua và hài lòng với nó.

c. Phân tích mối quan hệ giữa giá sách và đánh giá

Dùng biểu đồ scatter plot để mô tả mối quan hệ giữa giá sách (price) và đánh giá trung bình (avg_reviews) của những quyển sách.

```
# Xem xét mối quan hệ giữa giá sách và đánh giá.
plt.figure(figsize=(10,6))
sns.scatterplot(data=df, x='price', y='avg_reviews', alpha=0.6)
plt.title('Mối quan hệ giữa giá sách và đánh giá')
plt.xlabel('Giá sách')
plt.ylabel('Đánh giá trung bình')
plt.show()
```

Kết quả:



Nhận xét: Dựa trên biểu đồ ta có thể thấy một xu hướng tương quan dương âm, tức là khi giá sách tăng thì đánh giá trung bình có xu hướng giảm. Điều này có nghĩa là độc giả có khả năng mua sách giá thấp và chỉ có thể đánh giá sách giá thấp thôi. Tuy nhiên cũng có một số điểm không thuộc xu

hướng chung, chẳng hạn như một số sách có giá thấp nhưng đánh giá trung bình cao, điều này có thể phụ thuộc vào yếu tố khác ảnh hưởng đến.

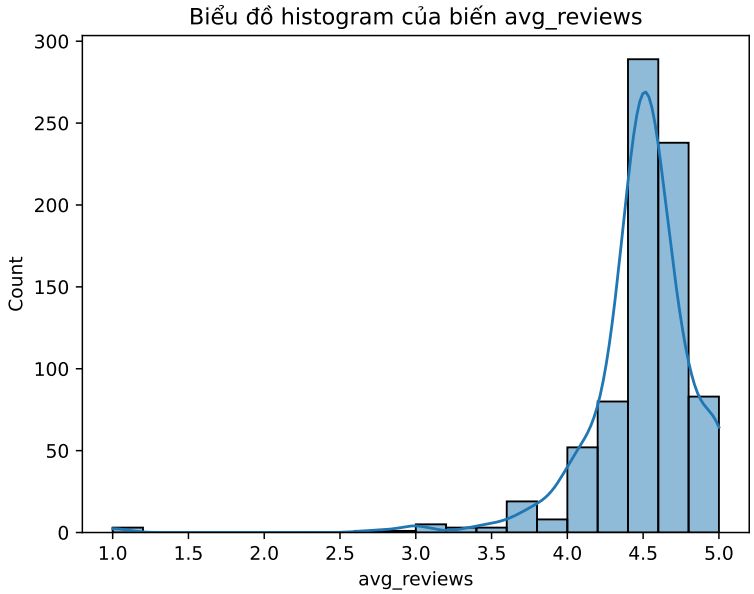
5 Khai thác dữ liệu nghiên cứu

5.1 Kiểm định giả thuyết

5.1.1 Kiểm định t (t-test)

Trong thống kê, "kiểm định t là một công cụ giúp so sánh giá trị trung bình (the mean) của một hoặc hai tổng thể bằng cách sử dụng phương pháp kiểm định giả thuyết" (Paul, 2008). Phương pháp kiểm định t-test được sử dụng để:

- Xác định khác biệt về giá trị trung bình của một tổng thể so với giá trị cho trước nào đó (thường gọi là giá trị trung bình giá trị trung bình giả thuyết - α hypothesized mean)
- Kiểm định sự khác biệt về giá trị trung bình giữa hai tổng thể trong mẫu.



Trong biểu đồ trên, ta thấy điểm trung bình đánh giá sách tập trung xung quanh giá trị 4,5. Ta muốn tìm hiểu xem liệu giá trị trung bình của biến avg_reviews có lớn hơn 4,5 hay không.

Mục tiêu 1: Kiểm định giá trị trung bình của biến 'avg_reviews' lớn hơn hoặc bằng 4,5 Giả sử chúng ta muốn kiểm định giả thuyết rằng giá trị trung của 'avg_reviews' lớn hơn hoặc bằng 4.5. với độ tin cậy là 98%; nghĩa là, chúng ta sẽ bác bỏ giả thuyết H0 và ủng hộ giả thuyết thay thế H1 nếu $p - value < 0.02$.

$$H_0 : \mu(avg\_reviews) < 4.5$$

$$H_1 : \mu(avg\_reviews) >= 4.5$$

Code

```python
from scipy.stats import ttest_1samp

print('H0: trung bình của avg_reviews >= 4.5')
print('H1: trung bình của avg_reviews <  4.5')
null_hypothesis_mean = 4.5
confidence_level = 0.98

t_statistic, p_value = ttest_1samp(df['avg_reviews'],
 ↪  null_hypothesis_mean)

print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")

alpha = 1 - confidence_level
if p_value < alpha:
    print("Bác bỏ H0.")
    print('--> Trung bình của avg_reviews < 4.5 .')

else:
    print('Chấp nhận H0')
    print("Không có đủ bằng chứng để kết luận giả thuyết trung bình của
     ↪  avg_reviews < 4.5")
```

Output

```
H0: trung bình của avg_reviews >= 4.5
H1: trung bình của avg_reviews <  4.5
T-statistic: -1.799585519368086
P-value: 0.07231049424561395
Chấp nhận H0
Không có đủ bằng chứng để kết luận giả thuyết trung bình của avg_reviews <
 ↪  4.5
```

Kết quả: Thực hiện kiểm định t cho thấy kết quả p-value = 0.0723 lớn hơn ngưỡng $\alpha = 0.02$ đã chọn.

Giải thích: Khi p-value lớn hơn ngưỡng alpha, chúng ta không có đủ bằng chứng để bác bỏ giả thuyết H_0: "giá trị trung bình của biến avg_reviews nhỏ hơn hoặc bằng 5". Điều này cho biết rằng giá trị trung bình của biến avg_reviews thực sự không lớn hơn 4.5 với mức ý nghĩa thống kê.

Dựa trên kết quả kiểm định, chúng ta không có đủ bằng chứng để ủng hộ giả thuyết thay thế (H_1): "Giá trị trung bình của biến avg_reviews lớn hơn 4.5" với độ tin cậy 98%.

5.1.2 Kiểm định chi-square (chi bình phương)

Định nghĩa: Kiểm định Chi bình phương là phương pháp thống kê được dùng để xác định mối quan hệ giữa các biến độc lập rời rạc. Phương pháp này thường được dùng trong các trường hợp:

- So sánh mức độ khác biệt giữa các tần số quan sát và tần số dự kiến trong cùng một bảng tần số và thông qua đó, đánh giá mức độ khác biệt có ý nghĩa gì so với thống kê hay không
- Kiểm tra các giả thuyết về các biến rời rạc để xem mức độ liên quan, sự đồng đều và mối qua hệ

Mục tiêu: Kiểm định mối tương quan giữa biến 'language' và 'publish_year'

Giả sử chúng ta muốn kiểm định mức độ tương quan giữa 2 biến phân loại trong tập dữ liệu về trang bán hàng sách là biến 'language' và 'publish_year' với độ tin cậy 95%. Trong các giá trị của kết quả trả về, nhóm chủ yếu sử dụng giá trị p_value để xác định kết quả kiểm định.

$$H_0 : \text{Ngôn ngữ và năm phát hành của sách là độc lập.}$$

$$H_1 : \text{Ngôn ngữ và năm phát hành của sách không độc lập.}$$

Code

```
from scipy.stats import chi2_contingency
print('H0: Ngôn ngữ và năm sách phát hành là độc lập.')
print('H1: Ngôn ngữ và năm sách phát hành không độc lập.')
contingency_table = pd.crosstab(df['language'], df['publish_year'])

chi2, p, _, _ = chi2_contingency(contingency_table)

print(f"Chi-square value: {chi2 :.4f}")
print(f"P-value: {p}")

alpha = 0.05
if p < alpha:
    print('Bác bỏ H0')
```

```
    print("Ngôn ngữ và năm sách phát hành không độc lập.")
else:
    print('Chấp nhận H0')
    print("Không có đủ bằng chứng để kết luận Ngôn ngữ và năm sách phát
    ↪  hành không độc lập.")
```

Output

```
H0: Ngôn ngữ và năm sách phát hành là độc lập.
H1: Ngôn ngữ và năm sách phát hành không độc lập.
Chi-square value: 317.2738
P-value: 1.8618677535783751e-22
Bác bỏ H0
Ngôn ngữ và năm sách phát hành không độc lập.
```

Kết quả: $p\_value = 1.8619e^{-22} < 0.05$ nhận được từ thực hiện kiểm định chi squared. Từ đây ta có thể khẳng định ngôn ngữ và năm phát hành của sách không độc lập với nhau.

5.1.3 Kiểm định Shapiro-Wilk

Định nghĩa: Kiểm định Shapiro - Wilk là một phương pháp thống kê được sử dụng để kiểm tra giả thiết về sự phân phối chuẩn của một biến ngẫu nhiên. Phương pháp này được thiết kế để kiểm tra xem dữ liệu có tuân theo phân phối chuẩn hay không. Phương pháp sẽ thực hiện so sánh các đặc tính phân phối của mẫu dữ liệu với phân chuẩn được kỳ vọng.

Cách thức thực hiện kiểm định:

Ta thực hiện so sánh giá trị thống kê được hình thành theo giả thuyết H0 với trọng số lấy từ phân phối chuẩn. Đối với kiểm định này, không dễ tính toán chính xác phân phối của giả thuyết H0, vì vậy phân phối thường sử dụng phương pháp Monte Carlo để lấy xấp xỉ. Phương pháp này sẽ lấy nhiều mẫu có cùng kích thước với x từ phân phối chuẩn và tình toán các giá trị của thống kê cho mỗi mẫu.

Nếu giá trị p (p-value) nhỏ hơn một ngưỡng ý nghĩa thống kê, ta có đủ bằng chứng để bác bỏ giả thuyết không và kết luận rằng dữ liệu không tuân theo phân phối chuẩn. Ngược lại, nếu giá trị p lớn hơn ngưỡng ý nghĩa, ta không có đủ bằng chứng để bác bỏ giả thuyết không và kết luận rằng có thể coi dữ liệu tuân theo phân phối chuẩn.

Lưu ý: Nếu giá trị p là rất nhỏ, thì có khả năng dữ liệu không tuân theo phân phối chuẩn. Tuy nhiên, kiểm định Shapiro-Wilk có thể trở nên không chính xác khi kích thước mẫu lớn.

Mục tiêu: Với độ tin cậy 95% (alpha = 0.05), ta cần kiểm định giả thuyết:

$$H_0 : \text{Biến 'price' có tuân theo dạng phân phối chuẩn.}$$

$$H_1 : \text{Biến 'price' không tuân theo dạng phân phối chuẩn.}$$

Code

```
import numpy as np
from scipy.stats import shapiro

sample_size = len(df['price'])

num_samples = 1000

shapiro_statistics = []

# Lấy nhiều mẫu từ phân phối chuẩn
for _ in range(num_samples):
    # Tạo mẫu ngẫu nhiên từ phân phối chuẩn
    random_sample = np.random.normal(size=sample_size)

    statistic, _ = shapiro(random_sample)
    shapiro_statistics.append(statistic)

observed_statistic, observed_p_value = shapiro(df['price'])

# Tính toán p-value theo phương pháp Monte Carlo
monte_carlo_p_value = (np.sum(np.array(shapiro_statistics) >=
↪  observed_statistic) + 1) / (num_samples + 1)

print(f"Shapiro-Wilk statistic (observed): {observed_statistic}")
print(f"P-value (observed): {observed_p_value}")
print(f"Monte Carlo p-value: {monte_carlo_p_value}")

if observed_p_value < 0.05:
    print("Bác bỏ H0: Biến price không tuân theo phân phối chuẩn.")
```

```
else:
    print("Không đủ bằng chứng để bác bỏ H0: Biến price tuân theo phân
    ↪ phối chuẩn.")
```

Output

```
Shapiro-Wilk statistic (observed): 0.33458107709884644
P-value (observed): 0.0
Monte Carlo p-value: 1.0
Bác bỏ H0: Biến price không tuân theo phân phối chuẩn.
```

Kết quả: Ta có thể thấy $p - value \approx 0$, như vậy ta có đủ bằng chứng thống kê chống lại H0 và ủng hộ giả thuyết H_1: các trọng số không được rút ra từ phân phối chuẩn.

Giải thích: Khi p-value nhỏ hơn ngưỡng ý nghĩa thống kê, chúng ta sẽ có đủ bằng chứng để bác bỏ H0: biến price tuân theo phân phối chuẩn. Dựa trên kết quả kiểm định, chúng ta có đủ thông tin để kết luận có đủ bằng chứng để phủ định giả thuyết biến price tuân theo phân phối chuẩn.

5.2 Phân lớp dữ liệu

5.2.1 Định nghĩa

Học có giám sát (Supervised learning) là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên các cặp (input, outcome) đã biết từ trước. Cặp dữ liệu này còn được gọi là (data, label), tức (dữ liệu, nhãn). Supervised learning là nhóm phổ biến nhất trong các thuật toán Machine Learning.

Một cách toán học, học có giám sát là khi chúng ra có một tập hợp biến đầu vào $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ và một tập hợp nhãn tương ứng $\mathbf{Y} = \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_N}\}$, trong đó $\mathbf{x}_i, \mathbf{y}_i$ là các vector. Các cặp dữ liệu biết trước $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbf{X} \times \mathbf{Y}$ được gọi là tập training data (dữ liệu huấn luyện). Từ tập training data này, chúng ta cần tạo ra một hàm số ánh xạ mỗi phần tử từ tập $\mathbf{X}$ sang một phần tử (xấp xỉ) tương ứng của tập $\mathbf{Y}$:

$$\mathbf{y}_i \approx f(\mathbf{x}_i), i = 1, 2, \ldots, N$$

Mục đích là xấp xỉ hàm số f thật tốt để khi có một dữ liệu $\mathbf{x}$ mới, chúng ta có thể tính được nhãn tương ứng của nó $\mathbf{y} = f(\mathbf{x})$

Ví dụ 1: Trong nhận dạng chữ viết tay, ta có ảnh của hàng nghìn ví dụ của mỗi chữ số được viết bởi nhiều người khác nhau. Chúng ta đưa các bức ảnh này vào trong một thuật toán và chỉ cho nó biết mỗi bức ảnh tương ứng với chữ số nào. Sau khi thuật toán tạo ra một mô hình, tức một hàm số mà

đầu vào là một bức ảnh và đầu ra là một chữ số, khi nhận được một bức ảnh mới mà mô hình chưa nhìn thấy bao giờ, nó sẽ dự đoán bức ảnh đó chứa chữ số nào.

Ví dụ này khá giống với cách học của con người khi còn nhỏ. Ta đưa bảng chữ cái cho một đứa trẻ và chỉ cho chúng đây là chữ A, đây là chữ B. Sau một vài lần được dạy thì trẻ có thể nhận biết được đâu là chữ A, đâu là chữ B trong một cuốn sách mà chúng chưa nhìn thấy bao giờ.

Ví dụ 2: Thuật toán dò các khuôn mặt trong một bức ảnh đã được phát triển từ rất lâu. Thời gian đầu, facebook sử dụng thuật toán này để chỉ ra các khuôn mặt trong một bức ảnh và yêu cầu người dùng tag friends - tức gán nhãn cho mỗi khuôn mặt. Số lượng cặp dữ liệu (khuôn mặt, tên người) càng lớn, độ chính xác ở những lần tự động tag tiếp theo sẽ càng lớn.

Thuật toán học có giám sát (supervised learning) còn được tiếp tục chia nhỏ ra thành hai loại chính là Phân loại (Classification) và Hồi quy (Regression). Kỹ thuật phân lớp được tiến hành bao gồm 2 bước:

1. Học, mục đích của bước này là xây dựng mô hình xác định tập các lớp dữ liệu.

2. Kiểm tra và đánh giá, bước này sử dụng mô hình phân lớp đã được xây dựng ở bước 1 vào việc phân lớp.

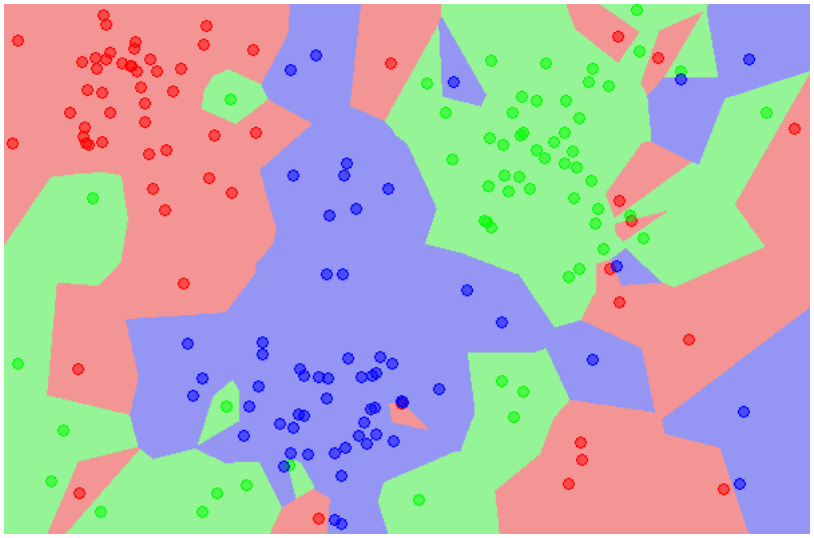
5.2.2 Các mô hình phân lớp

Đối với bộ dữ liệu "Amazon Data Science Books", ta sẽ sử dụng 4 mô hình học máy để tiến hành phân loại các cuốn sách với biến "Target"là các nhà xuất bản sách.

K-nearest neighbor (KNN)

K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi training, thuật toán này không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

Một cách ngắn gọn, KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách chỉ dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu. Hình dưới đây là một ví dụ về KNN trong classification với K = 1.



Hình 1: Bản đồ của 1NN

Ví dụ trên đây là bài toán Classification với 3 classes: Đỏ, Lam, Lục. Mỗi điểm dữ liệu mới (test data point) sẽ được gán label theo màu của điểm mà nó thuộc về. Trong hình này, có một vài vùng nhỏ xem lẫn vào các vùng lớn hơn khác màu. Ví dụ có một điểm màu Lục ở gần góc 11 giờ nằm giữa hai vùng lớn với nhiều dữ liệu màu Đỏ và Lam. Điểm này rất có thể là nhiễu. Dẫn đến nếu dữ liệu test rơi vào vùng này sẽ có nhiều khả năng cho kết quả không chính xác.

Support Vector Machine - SVM

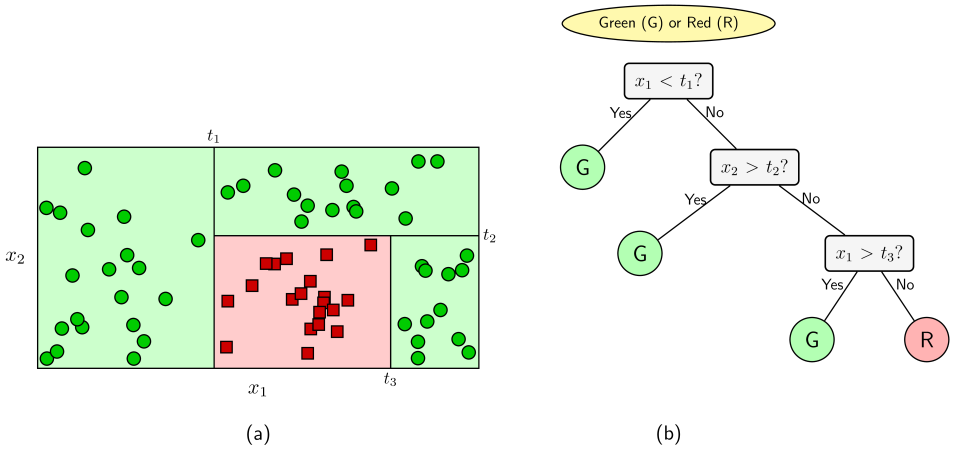
Support Vector Machine - SVM là một thuật toán học máy phổ biến được sử dụng cho cả bài toán phân loại và hồi quy. Thuật toán này được sử dụng để tạo ra ranh giới quyết định (decision boundary) giữa các lớp hoặc các nhóm dữ liệu.

SVM tìm kiếm ranh giới quyết định tốt nhất bằng cách tìm một siêu phẳng (hyperplane) tối ưu nhất để phân chia các điểm dữ liệu thành các lớp khác nhau. Siêu phẳng này cố gắng tối đa khoảng cách (margin) từ siêu phẳng đến các điểm dữ liệu gần nhất của mỗi lớp, các điểm này được gọi là các vector hỗ trợ (support vectors). Mục tiêu chính của SVM là tìm ra siêu phẳng có margin lớn nhất, tức là giữ khoảng cách giữa các lớp càng xa càng tốt, giúp mô hình có khả năng tổng quát hóa tốt với dữ liệu mới.

Decision Tree

Cây Quyết Định (Decision Tree) là một thuật toán học máy thuộc nhóm học có giám sát (supervised learning), được sử dụng cho cả bài toán phân loại (classification) và dự đoán (regression). Thuật toán này hoạt động bằng cách tạo ra một cấu trúc cây quyết định, mô phỏng các quyết định và hành động dựa trên các điều kiện dữ liệu.

Cách hoạt động:



Hình 2: Bản đồ của 1 cây quyết định

- Chọn thuộc tính tốt nhất: Bắt đầu từ nút gốc, thuật toán chọn thuộc tính tốt nhất để phân chia dữ liệu, tối ưu hóa việc phân loại hoặc dự đoán.
- Tạo cây quyết định: Tiếp tục phân chia dữ liệu tại mỗi nút con, tạo ra cây quyết định dựa trên các tiêu chí phân loại tốt nhất cho từng nút.
- Dừng quá trình xây dựng cây: Quá trình tiếp tục cho đến khi mọi điều kiện dừng được đáp ứng, ví dụ như đạt đến độ sâu tối đa của cây, không còn dữ liệu hoặc không có thêm thuộc tính nào để phân chia.

Cây Quyết Định được sử dụng rộng rãi trong nhiều lĩnh vực như hệ thống chuyên gia, dự đoán tài chính, nhận dạng hình ảnh, và trong các ứng dụng lĩnh vực khác nhau trong khoa học dữ liệu.

Naive Bayes

Thuật toán Naive Bayes là một thuật toán phân loại trong machine learning, dựa trên việc áp dụng Định lý Bayes với giả định "ngây thơ"(naive), tức là các biến độc lập với nhau khi đã biết lớp của chúng.

Một vài điểm chính về thuật toán "Naive Bayes":

- Naive Bayes Classifiers (NBC) thường được sử dụng trong các bài toán Text Classification.
- NBC có thời gian training và test rất nhanh. Điều này có được là do giả sử về tính độc lập giữa các thành phần, nếu biết class.
- Nếu giả sử về tính độc lập được thoả mãn (dựa vào bản chất của dữ liệu), NBC được cho là cho kết quả tốt hơn so với SVM và logistic regression khi có ít dữ liệu training.
- NBC có thể hoạt động với các feature vector mà một phần là liên tục (sử dụng Gaussian Naive Bayes), phần còn lại ở dạng rời rạc (sử dụng Multinomial hoặc Bernoulli).

- Khi sử dụng Multinomial Naive Bayes, Laplace smoothing thường được sử dụng để tránh trường hợp 1 thành phần trong test data chưa xuất hiện ở training data.

5.2.3 Xây dựng mô hình phân lớp và thực hiện đánh giá

Đối với bộ dữ liệu "Amazon Data Science Books", ta sẽ tiến hành xây dựng mô hình phân loại và dự đoán các cuốn sách về Khoa học dữ liệu được phát hành bởi nhà xuất bản nào.

Bước 1: Ta thực hiện mã hóa (encode) biến target là *publisher*

```
def label_encoder(x):
    if x == 'Packt Publishing':
        return 1
    elif x == 'Independently published':
        return 2
    elif x == "O'Reilly Media":
        return 3
    else:
        return 4

X = df[df.columns[df.dtypes != 'object']]
y = np.array([label_encoder(publisher) for publisher in df['publisher']])

# Chia tập dữ liệu thành training, test sets theo tỷ lệ 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .2,
↪  random_state = 1)
```

Ta thực hiện mã hóa tên của 3 nhà xuất bản có số lượng sách Khoa học dữ liệu được xuất bản nhiều nhất là Packt Publishing, Independently published và O'Reilly Media lần lượt là 1, 2 và 3. Đối với sách được phát hành bởi các nhà xuất bản khác, ta sẽ mã hóa nhãn các cuốn sách đó là 4. Tiếp theo, ta sẽ lấy tất cả các biến có dữ liệu dạng số để làm biến độc lập bao gồm: price, pages, avg_reviews, n_reviews, star5, star4, star3, star2, star1, weight, min_price, volume.

Bước 2: Ta thực hiện khởi tạo các mô hình được sử dụng

```
# Các mô hình ứng viên
models = [KNeighborsClassifier(n_neighbors = int(pow(X_train.shape[0],
↪  1/2) / 2)),
          DecisionTreeClassifier(),
```

```
        GaussianNB(),
        SVC(kernel='rbf')]
```

Bước 3: Ta chia tập huấn luyện thành 10-folds và thực hiện cross-validation đối với từng mô hình, sau đó thực hiện đánh giá và so sánh độ chính xác của các mô hình với nhau để lựa chọn mô hình tốt nhất.

```
folds = 10
for model in models:
    model_name = model.__class__.__name__
    accuracies = cross_val_score(model, X, y, scoring = 'accuracy', cv =
    ↪ folds)
    print(f'Accuracy (trung bình) của mô hình {model_name} =
    ↪ {accuracies.mean():.4f}')
```

Kết quả:

```
Accuracy (trung bình) của mô hình KNeighborsClassifier = 0.6560
Accuracy (trung bình) của mô hình DecisionTreeClassifier = 0.6638
Accuracy (trung bình) của mô hình GaussianNB = 0.3769
Accuracy (trung bình) của mô hình SVC = 0.6688
```

Đánh giá

Kết quả cross-validation đã cung cấp một cái nhìn tổng quan về độ chính xác của các mô hình trên dữ liệu. Độ chính xác trung bình của các mô hình được tính bằng phương pháp cross-validation, và mô hình có độ chính xác trung bình cao nhất là SVM với độ chính xác khoảng 66.88%.

Việc lựa chọn mô hình tốt nhất cần xem xét nhiều yếu tố khác nhau, không chỉ dựa trên độ chính xác mà còn dựa trên yếu tố như tính diễn giải, độ phức tạp của mô hình, khả năng xử lý dữ liệu lớn, và cách mà mô hình phù hợp với bản chất của dữ liệu.

Trong trường hợp này, SVM có độ chính xác cao nhất, nhưng việc chọn mô hình cuối cùng cần xem xét các yếu tố sau:

- Độ phức tạp: Decision Tree thường có khả năng overfitting nếu không kiểm soát được độ sâu của cây.
- Tính diễn giải: KNN và Naive Bayes thường dễ hiểu hơn và có tính diễn giải tốt hơn so với SVM hay Decision Tree.
- Tính linh hoạt: SVM có thể phù hợp với dữ liệu có cấu trúc phức tạp hơn và có khả năng chống lại overfitting.

Với một bộ dữ liệu về sách khoa học dữ liệu, việc lựa chọn mô hình cuối cùng cần phản ánh các yếu tố trên cùng với yếu tố thực tế trong việc áp dụng mô hình vào thực tế. Ví dụ, nếu tính diễn giải và khả năng giải thích dữ liệu là yếu tố quan trọng, có thể một mô hình như KNN hoặc Naive Bayes sẽ phù hợp hơn. Tuy nhiên, nếu độ chính xác là ưu tiên hàng đầu và dữ liệu có cấu trúc phức tạp, SVM có thể là lựa chọn tốt nhất.

6 Kết luận

6.1 Vì sao các phương pháp của nhóm hiệu quả

Nhóm đã tiến hành nghiên cứu và phân tích bộ dữ liệu "Amazon Data Science Books Dataset" thông qua việc sử dụng nhiều phương pháp phân tích và biểu diễn trực quan. Dưới đây là một số lý do chính giúp giải thích tại sao phương pháp của nhóm được coi là hiệu quả:

1. Tiền xử lý dữ liệu: Giai đoạn tiền xử lý dữ liệu giúp làm sạch dữ liệu, loại bỏ nhiễu và chuẩn hóa thông tin, tạo điều kiện thuận lợi cho các phương pháp phân tích và mô hình hóa.
2. Phân tích đơn biến và đa biến: Nhóm thực hiện các phương pháp phân tích đơn biến và đa biến cũng như mối tương quan giữa chúng. Điều này giúp chúng ta nhận diện xu hướng, đặc điểm và sự phụ thuộc giữa các yếu tố trong bộ dữ liệu
3. Biểu diễn trực quan: Dữ liệu được biểu diễn một cách trực quan, dễ hiểu để trình bày thông tin. Điều này giúp người xem hiểu rõ hơn về xu hướng và biến động trong dữ liệu.
4. Kiểm định thống kê: Sử dụng các kiểm định t-test, chi-square test và Shapiro-Wilk giúp nhóm đưa ra những kết luận hợp lý về sự khác biệt và mối quan hệ giữa các biến. Điều này tăng tính xác thực và giúp hỗ trợ quyết định dựa trên dữ liệu.

6.2 Ý nghĩa đề tài

Qua việc thực hiện đề tài này, nhóm đã có cơ hội áp dụng và mở rộng kiến thức về quá trình phân tích dữ liệu, từ việc xử lý dữ liệu đến xây dựng mô hình dự đoán. Đây là đề tài có ý nghĩa vô cùng quan trọng, không chỉ với những người đọc quan tâm đến Khoa học dữ liệu và có nhu cầu tìm hiểu thông tin về những cuốn sách phổ biến, mà còn là nguồn thông tin quan trọng đối với các nhà xuất bản sách liên quan đến lĩnh vực này.

- Ứng dụng vào thực tế: Môn học và đề tài đã giúp nhóm áp dụng những kỹ năng và phương pháp học được vào việc nghiên cứu và giải quyết vấn đề thực tế, trong trường hợp này là phân tích xu hướng của sách trong ngành Data Science.
- Có ý nghĩa trong lĩnh vực xuất bản và đánh giá sách: Kết quả của đề tài mang lại thông tin hữu ích cho ngành xuất bản và đánh giá sách, giúp họ hiểu rõ hơn về sự phổ biến và chất lượng của các sách liên quan đến Khoa học Dữ liệu trên thị trường.
- Hỗ trợ quyết định kinh doanh: Các phương pháp và kết quả nghiên cứu có thể hỗ trợ quyết định kinh doanh trong việc chọn lựa chiến lược xuất bản, quảng bá sách, và cải thiện chất lượng sách để thu hút độc giả.

7 Tài liệu liên quan

Link Source Code: colab

Link powerpoint: ppt