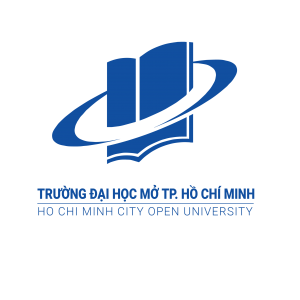
**BỘ GIÁO DỤC & ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**



**BÁO CÁO BÀI TẬP LỚN**

**Đề tài: Tìm hiểu về cơ sở dữ liệu Tiki book.**

**Giáo viên hướng dẫn: Nguyễn Văn Bảy**

**Sinh viên thực hiện:**

**Phạm Vĩnh Khánh 2051012047**

**Nguyễn Ngọc Linh 2051012053**

**Bùi Thị Mỹ Linh 2051052073**

**MỤC LỤC**

[**A. MÔ TẢ DỮ LIỆU**](#_heading=) 2

[**B. CHUẨN BỊ DỮ LIỆU**](#_heading=) 3

[**C. PHÂN TÍCH DỮ LIỆU KHÁM PHÁ (EDA)**](#_heading=) 6

[I. Tổng quan về dữ liệu](#_heading=h.ya403a9hde68) 6

[II. EDA](#_heading=h.kvkq8w8c4131) 9

[- Top books have a number of reviews larger than 3000.](#_heading=h.hov83q4qjuk7) 11

[- More expensive books have more reviews?](#_heading=h.gklrko8sv2k) 12

[- More reviewed books have higher average ratings?](#_heading=h.heaidsw9yij7) 12

[- The more pages, the more expensive?](#_heading=h.p6hjswnct86j) 13

[- What are the largest authors?](#_heading=h.ev81uqef72yz) 14

[- What is the best seller book?](#_heading=h.r451rjb6nnw) 15

[III. Xóa dữ liệu ngoại lai (outliner)](#_heading=h.ptf6bpfu8yb0) 19

[**D. XÂY DỰNG MÔ HÌNH**](#_heading=h.rrjfwc2olbty) 22

[I. Mô hình phân loại văn bản :](#_heading=h.6nq3t6z7xn8b) 22

[1. Công cụ khai phá dữ liệu:](#_heading=h.7yfhderg38yz) 22

[2. Đặc tả bài toán](#_heading=h.4drnc3m9nhv6) 22

[Chuẩn bị dữ liệu:](#_heading=h.dtrijru8tmyn) 22

[Kiểm tra các giá trị thiếu và xóa:](#_heading=h.mi7rqtmce0qg) 23

[Gán cột content cho X và cột rating cho y:](#_heading=h.rrrc2tmfvuxl) 24

[Chuyển đổi y thành dạng ma trận nhị phân (one-hot encoding):](#_heading=h.403lsmi8c5hi) 24

[TPU setup:](#_heading=h.gq6wsaqhym8e) 24

[Build Model:](#_heading=h.snjmqkegbwg9) 25

[II. Mô hình dự đoán giá sách:](#_heading=h.161kasvlmyym) 31

[1. Công cụ khai phá dữ liệu](#_heading=h.bb6333p413r0) 31

[2. Đặc tả bài toán](#_heading=h.dioxuqy5r5zp) 31

[Chuẩn bị Dữ liệu:](#_heading=h.up0bxs19ju8i) 32

[Chuẩn hóa Dữ liệu:](#_heading=h.17x84n1kv9ki) 32

[Xây dựng Mô hình ANN](#_heading=h.arhhakotg1vi) 32

[Huấn luyện Mô hình](#_heading=h.pkwyaxxhv6s8) 33

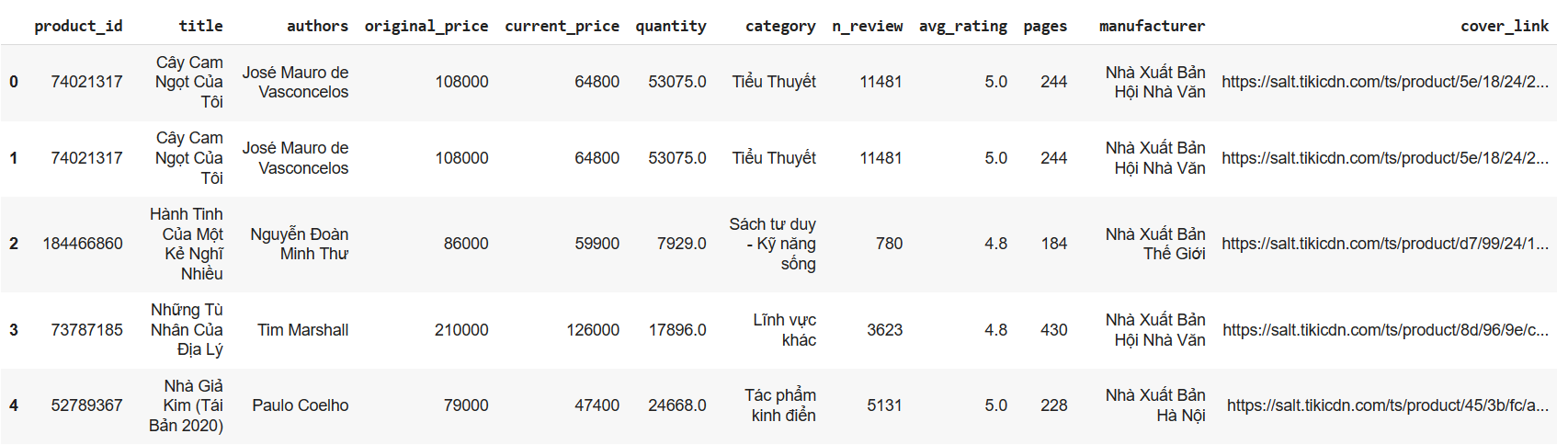
[Đánh giá Mô hình](#_heading=h.1gyjkzilc6q4) 34

[Dự đoán giá sách cho toàn bộ dữ liệu](#_heading=h.8rgobtrzso7t) 35

# MÔ TẢ DỮ LIỆU

Bộ dữ liệu sách của **Tiki E-commerce Việt Nam**:

Tên file: [**book\_data.csv**](https://drive.google.com/file/d/1FLsY-LJiYWy3fwQqa_BASy7mSGEcjBJ5/view?usp=drive_link) (1796 thể hiện)



Gồm các thuộc tính đặc trưng:

* **product\_id** (id của sản phẩm trong cơ sở dữ liệu của tiki, khóa chính)
* **title** (Tên của cuốn sách, có thể chứa thời gian tái bản)
* **authors** (Tên tác giả)
* **original\_price** (Giá gốc, tính bằng đơn vị USD)
* **current\_price** (Giá hiện tại nếu có chiết khấu, tính bằng đơn vị USD)
* **quantity** (Tổng số sách đã bán ra)
* **category** (Thể loại sách)
* **n\_review** (Tổng số đánh giá của khách hàng)
* **avg\_rating** (Đánh giá trung bình, tối đa 5)
* **pages** (Tổng số trang của mỗi cuốn sách)
* **manufacturer** (nhà xuất bản)
* **cover\_link** (link bìa sách)

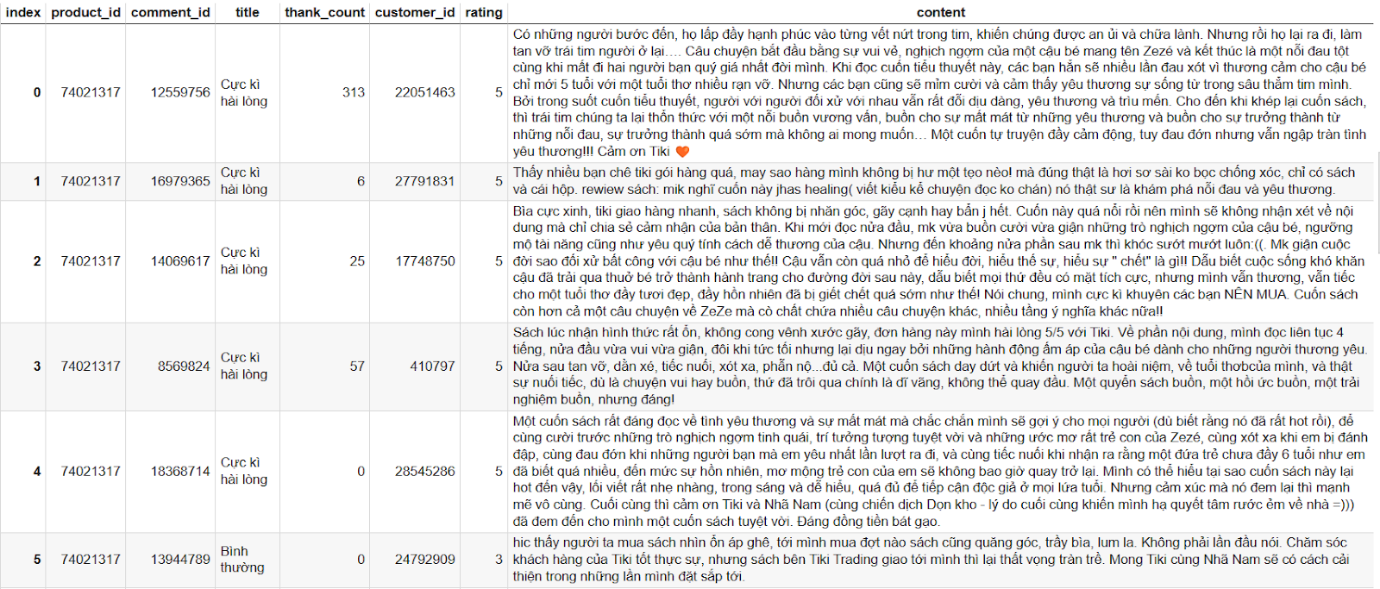
Thông tin các cột được giữ lại:

* **product\_id**, **title** , **authors**, **original\_price**, **current\_price**,

**quantity**, **category**, **n\_review**, **avg\_rating, pages.**

Nhận xét:

Dữ liệu đánh giá của khách hàng khi mua sách, tên file:  [comments.csv](https://drive.google.com/file/d/1QaKdTIi4gwTj_-PS7U4RjVFrgrcBl8le/view?usp=drive_link) (141281 thể hiện)



Gồm các thuộc tính đặc trưng: **product\_id** (id của sản phẩm trong cơ sở dữ liệu của tiki), **comment\_id** (id của từng nhận xét, khóa chính), **title** (từ khóa của nhận xét), **thank\_count** (Số lượt thích của những khách hàng khác dành cho nhận xét), **customer\_id** (Mỗi khách hàng có một id riêng), **rating** (Đánh giá trung bình của nhận xét, từ 1 đến 5), **content** (Nhận xét của khách hàng)

# CHUẨN BỊ DỮ LIỆU

1. **Các công việc chuẩn bị dữ liệu :**

Đầu tiên để phát hiện các lỗi trong dữ liệu ta đếm các giá trị **NULL** có trong file:

data\_df.isnull().sum()

Kiểm tra cho thấy có : **authors** 143, **quantity** 45, **pages** 250, **manufacturer** 273 có giá trị **NULL**.

Tiếp theo đi lần lượt từng cột dữ liệu:

Trong **Title** : xóa tiêu đề trùng lặp

data\_df = data\_df.drop\_duplicates(subset=['title'])

Trong **Author**: Thay "." thành "Unknown"

data\_df.authors.value\_counts()

Thay giá trị null thành "Unknown"

data\_df.loc[data\_df.authors == '.' , 'authors'] = "Unknown"

Trong **Quantity**, đối với các ô dữ liệu bị thiếu xử lí bằng cách gán các giá trị trung bình của toàn bộ cột quantity, bao gồm các giá trị NaN.

data\_df.quantity = data\_df.quantity.fillna(np.mean(data\_df.quantity))

Trong **Category**:

Sau khi nhìn tổng quát ta nhìn thấy có một số thể loại sách có tên giống nhau nhưng cách viết khác nhau ví dụ: "Kiến Thức Bách Khoa" thành "Kiến thức - Bách khoa"

Sửa : data\_df.loc[data\_df.category == 'Kiến Thức Bách Khoa' , 'category'] = "Kiến thức - Bách khoa"

data\_df.loc[data\_df.category == 'Kiến Thức Bách Khoa' , 'category'] = "Kiến thức - Bách khoa"

Vì có rất nhiều thể loại khác nhau, để làm gọn lại dữ liệu ta chỉ lấy 30 thể loại đầu tiên.. Thay giá trị của các thể loại ngoài danh sách 30 thể loại đầu tiên thành "Others"

def handle\_category(category):

    if category not in keeping\_values:

        return "Others"

    return category

data\_df.category = data\_df.category.apply(lambda category: handle\_category(category))

Trong **Pages**:

Thay đổi kiểu dữ liệu: chuyển đổi các dữ liệu thành số và là số nguyên làm tròn. Sau khi đã chuyển đổi tất cả các giá trị trong cột "pages" thành số nguyên, đoạn mã này dùng để thay đổi kiểu dữ liệu của cột "pages" thành "int64". Điều này đảm bảo rằng cột "pages" chứa các giá trị số nguyên kiểu dữ liệu 64-bit.

data\_df.pages = data\_df.pages.apply(lambda page: round(float(page)))

data\_df.pages = data\_df.pages.astype("int64")

Chỉ giữ lại những quyển sách dưới 3000 trang trở xuống, để giảm tải những dữ liệu không cần thiết:

data\_df = data\_df.query('pages < 3000')

Trong **Manufacturer**:

Sửa lại những tên NXB  có tên giống nhau nhưng cách viết khác nhau ví dụ: hong duc – Nhà Xuất Bản Hồng Đức:

Tạo thêm cột Discount : Để hiển thị số tiền được giảm giá:

data\_df['discount'] = round(1 - data\_df['current\_price']/data\_df['original\_price'],2)

Thay đổi đơn vị **VND USD**

def convert\_to\_usd(vietnamdong):

    usd = vietnamdong/23700

    usd = round(usd, 2)

    return usd

data\_df.original\_price = data\_df.original\_price.apply(lambda vietnamdong: convert\_to\_usd(vietnamdong))

data\_df.current\_price = data\_df.current\_price.apply(lambda vietnamdong: convert\_to\_usd(vietnamdong))

* **Thay đổi kiểu dữ liệu:**

Cho biết định dạng và quan sát được số lượng **NON-NULL** trong file:

data\_df.info()

* **product\_id** 1767(int64), **title** 1767(object), **authors** 1767(object), **original\_price** 1767(float64), **current\_price** 1767(float64), **quantity** 1767(float64), **category** 1767(object), **n\_review** 1767(int64), **avg\_rating** 1767(float64), **pages** 1767(int64), **manufacturer** 1767(object), **cover\_link** 1767(object), **discount** 1766(float64).

Sau đó thay đổi đổi kiểu dữ liệu của các cột **title**, **authors**, **category**, **manufacturer**, **cover\_link** thành kiểu dữ liệu string và **quantity** thành int64.

# PHÂN TÍCH DỮ LIỆU KHÁM PHÁ (EDA)

Tạo df mới để dễ trực quan hóa hơn

eda\_df = data\_df.drop(['product\_id', 'cover\_link'], axis=1)

eda\_df.authors.value\_counts()

## Tổng quan về dữ liệu

Về **Price** (giá):  chia làm 2 loại original\_price, current\_price.  
 Vẽ biểu đồ để có cái nhìn tổng quát.

fig = plt.subplots(figsize=(10, 7))

ax = sns.kdeplot(eda\_df.original\_price, color='gray', shade=True, label='Original')

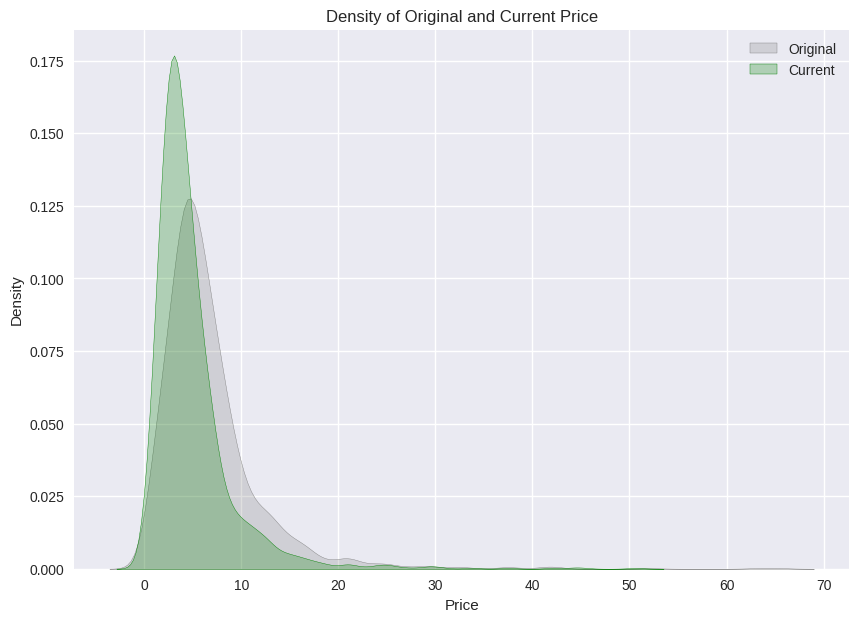
ax = sns.kdeplot(eda\_df.current\_price, color='g', shade=True, label='Current')

plt.xlabel('Price')

plt.title('Density of Original and Current Price')

plt.legend()

plt.show()



Nhìn vào biểu đồ ta có thể thấy đa số sách sẽ có giá từ **20$** trở xuống. Có thể kết luận giá sách ở Việt Nam là quá rẻ.

Về **Discount per Category**: Giảm giá cho mỗi danh mục. Xem biểu đồ để nhìn rõ các mục được giảm giá.

fig, ax = plt.subplots(figsize=(10,10))

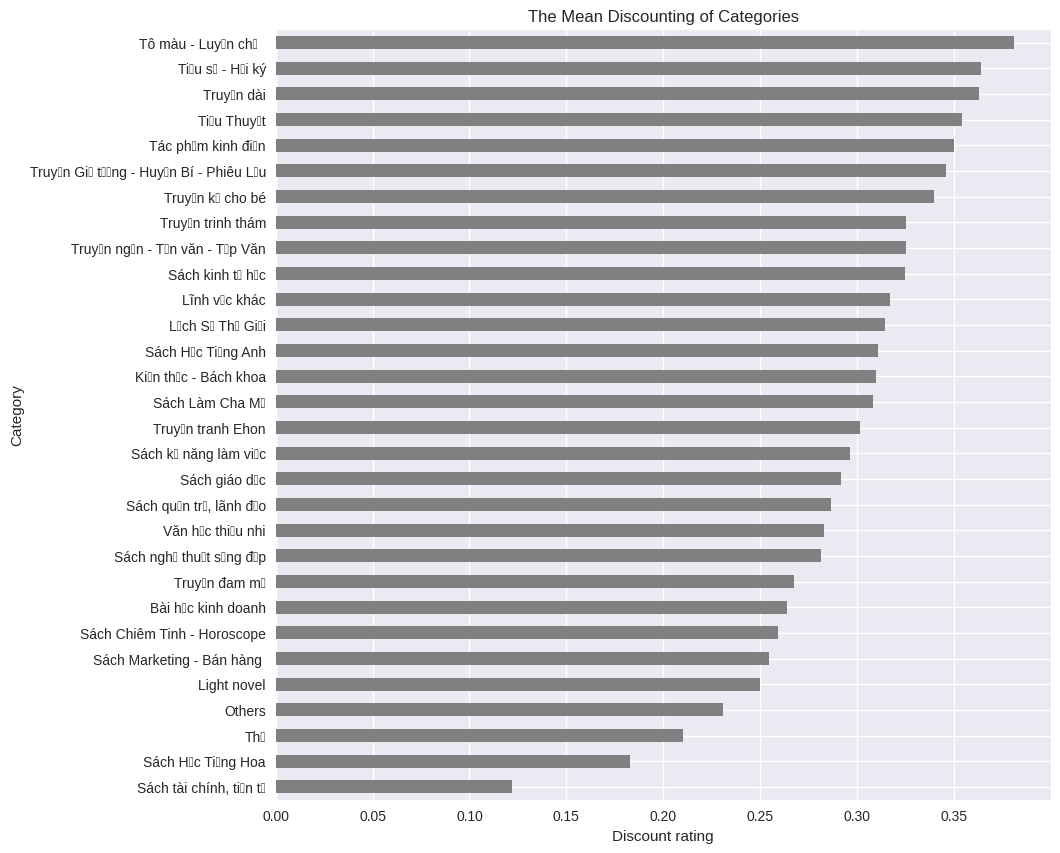
ax=eda\_df.groupby('category').discount.mean().sort\_values(ascending=True).plot(kind='barh', color='gray')

plt.xlabel('Discount rating')

plt.ylabel('Category')

plt.title('The Mean Discounting of Categories')

plt.show()



Hầu hết các danh mục giảm giá là truyện (chứa '**Truyen**').

Về **Quantity per Category All of Time**: Số lượng mỗi danh mục Mọi lúc

Về **Rating per Category**: Xếp hạng cho mỗi danh mục

fig, ax = plt.subplots(figsize=(10,10))

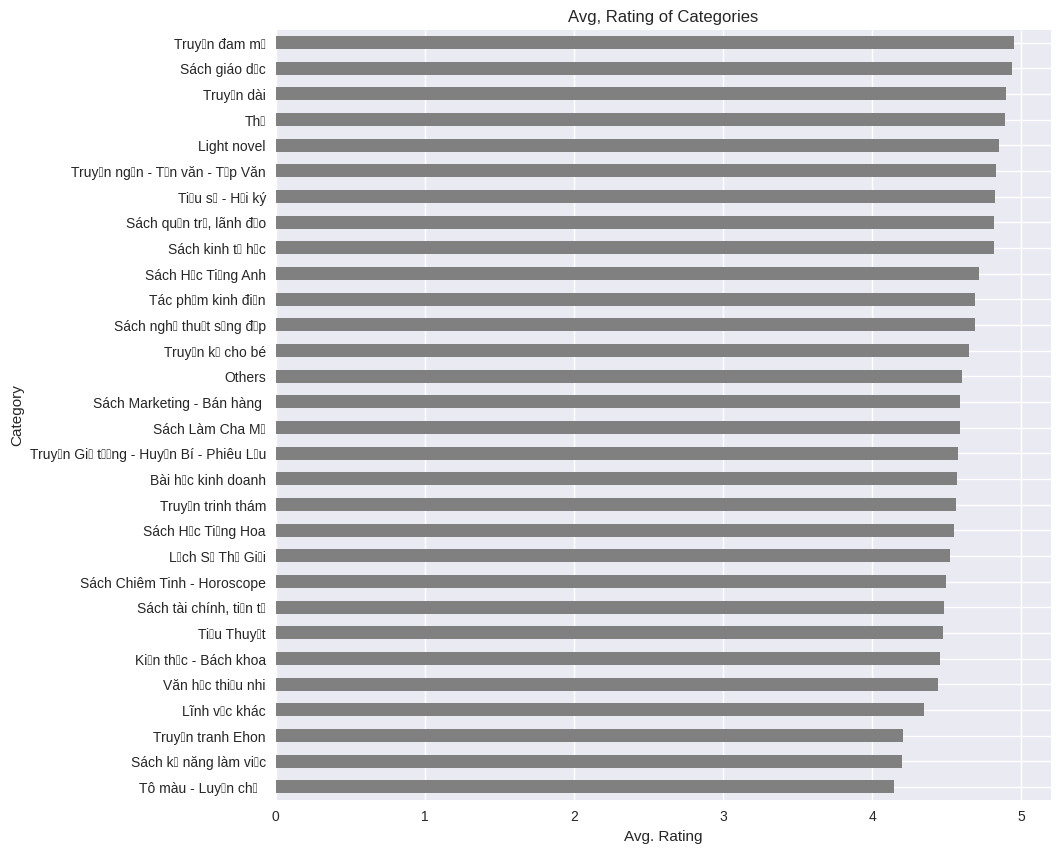
ax=eda\_df.groupby('category').avg\_rating.mean().sort\_values(ascending=True).plot(kind='barh', color='gray')

plt.xlabel('Avg. Rating')

plt.ylabel('Category')

plt.title('Avg, Rating of Categories')

plt.show()

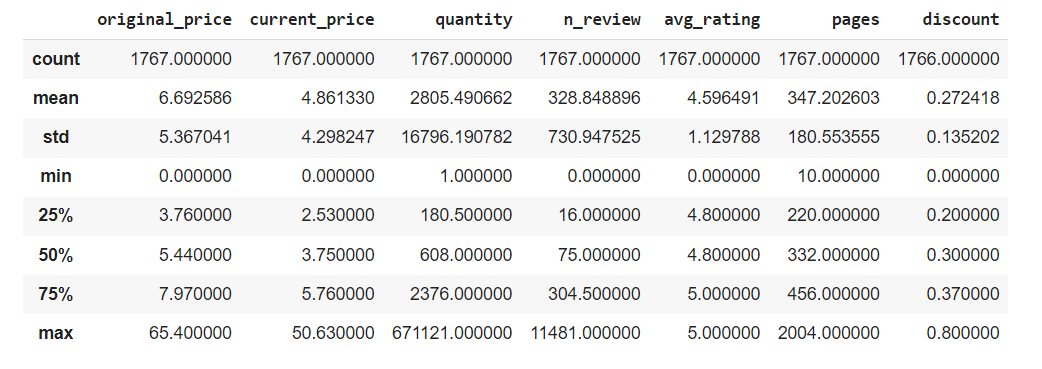


Qua biểu đồ có thể thấy được tất cả thể loại đều có trung bình đánh giá trên 4

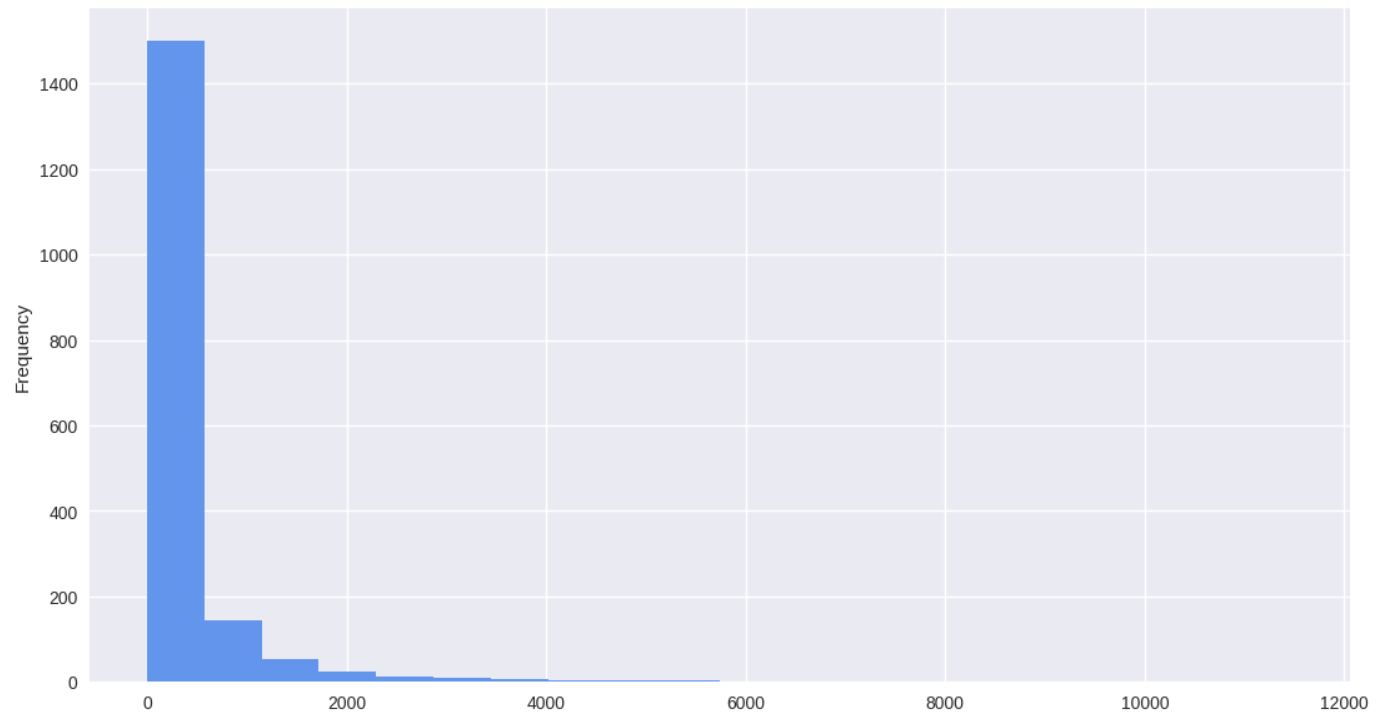
## EDA

eda\_df.describe()

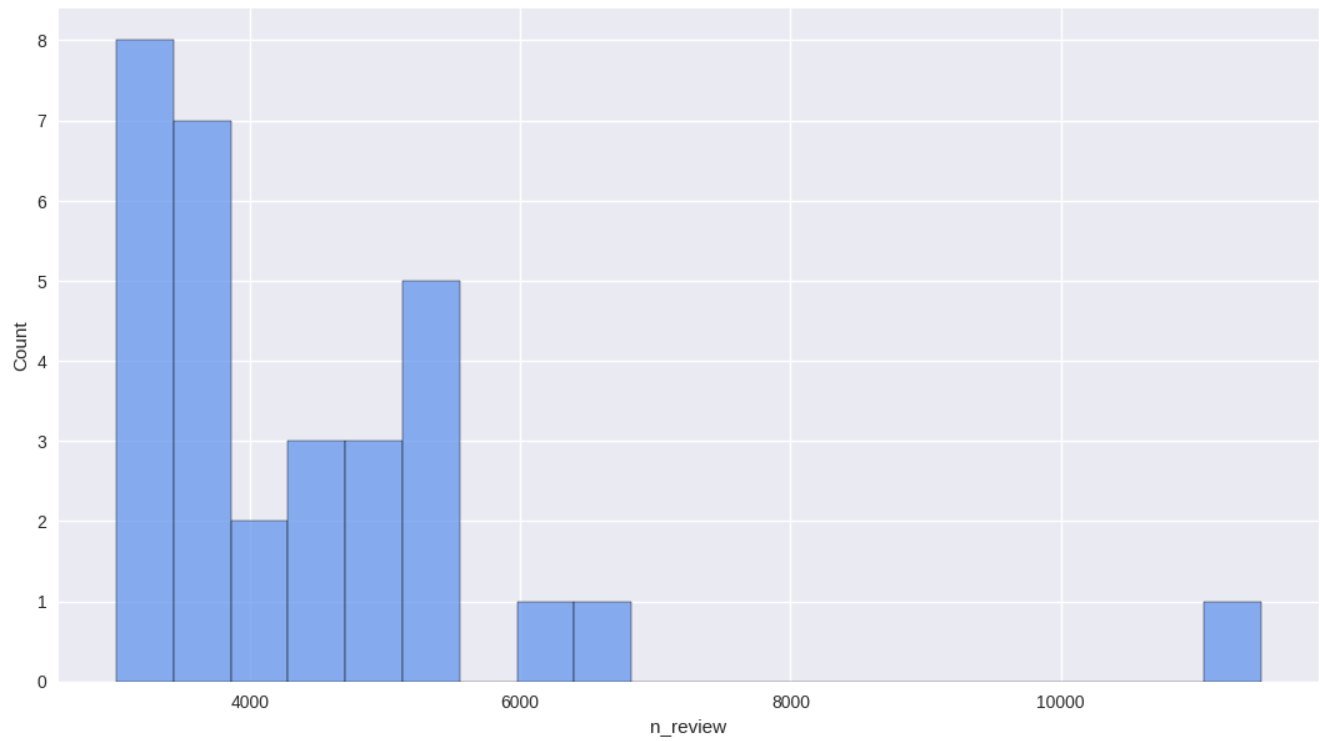
Thống kê số liệu của các dataframe gồm các thành phần: **count** (tổng số record), **mean** (trung bình cộng), **std** (độ lệch chuẩn), **phân vị**, **min** , **max**.



Ta có biểu đồ về số lượng đánh giá của khách hàng:



=> 90% số sách có dưới 3000 lượt đánh giá.

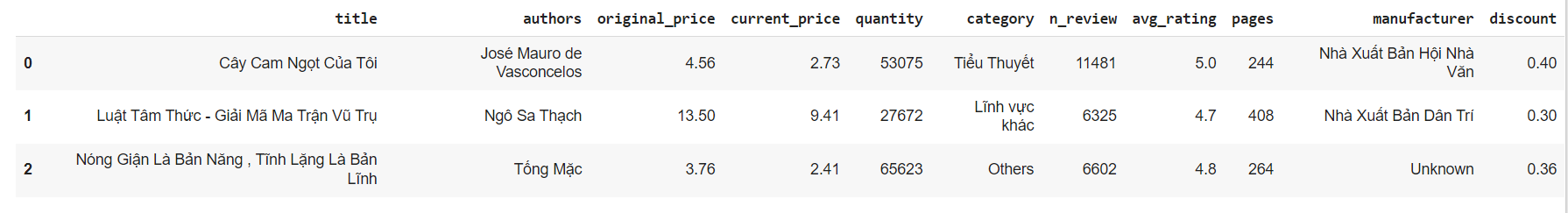


=> Trong 10% số sách có trên 3000 lượt đánh giá, ta thấy có 10% số sách sẽ có trên 6000 lượt đánh giá.

### Top books have a number of reviews larger than 3000.

larger\_3000\_reviews\_df.query('n\_review >= 6000').reset\_index(drop=True)

Có 3 quyển sách có hơn 6000 lượt đánh giá.



Cây Cam Ngọt của Tôi (11481 lượt đánh giá với xếp hạng 5/5)

Luật Tâm Thức - Giải Mã Ma Trận Vũ Trụ (6325 lượt đánh giá với xếp hạng 4.7/5)

Nóng Giận Là Bản Năng, Tĩnh Lặng Là Bản Tính (6602 lượt đánh giá với xếp hạng 4.8/5)

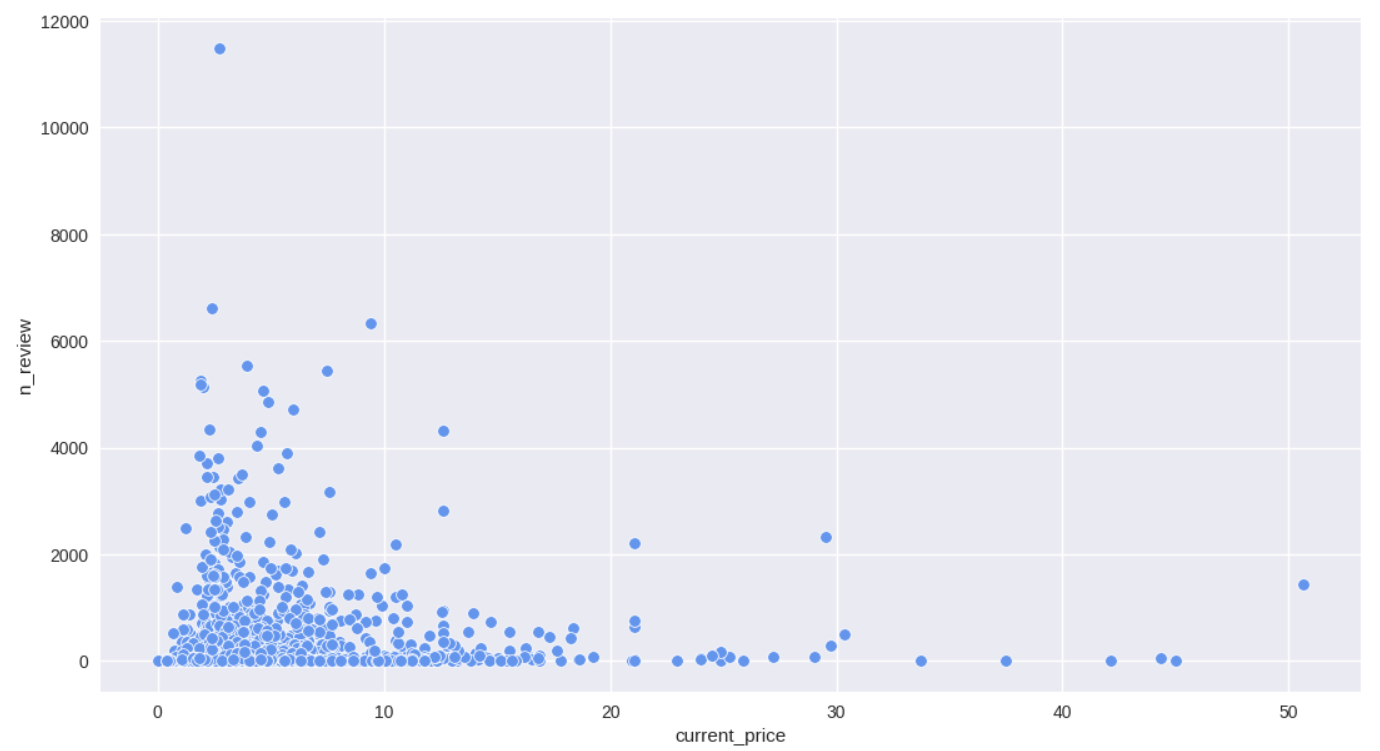
larger\_3000\_reviews\_df

=> Có tổng 31 quyển sách có trên 3000 lượt đánh giá và hầu hết tất cả đều có xếp hạng cao.

### More expensive books have more reviews?

sns.scatterplot(data=eda\_df, x='current\_price',y='n\_review', color='cornflowerblue');

Sau khi thực hiện ta có biểu đồ phân tán:

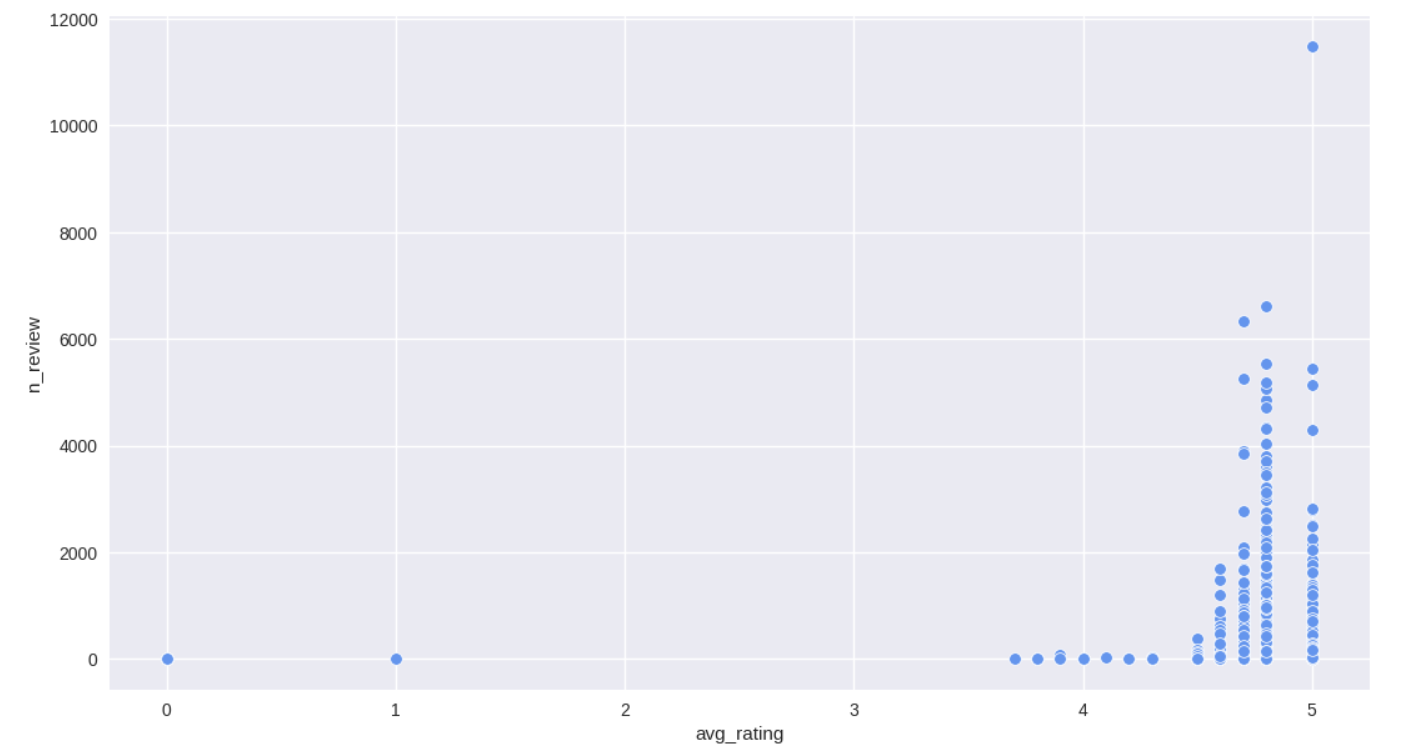


Hầu hết các cuốn sách có đánh giá lớn đều xoay quanh 0 và 10.

Do đó, chúng ta không thể coi những cuốn sách càng đắt tiền là có càng bài đánh giá hơn.

### More reviewed books have higher average ratings?

sns.scatterplot(data=eda\_df, x='avg\_rating',y='n\_review', color='cornflowerblue');



Biểu đồ phân tán đã cho thấy rõ xếp hạng của sách chủ yếu nằm ở khoảng lớn hơn 4 và bé hơn hoặc bằng 5.

pd.set\_option('display.max\_colwidth', True)

larger\_3000\_reviews\_df.sort\_values(by="n\_review",ascending=False).reset\_index(drop=True)[['n\_review','avg\_rating','title']]

pd.reset\_option('display.max\_colwidth')

Tất cả các cuốn sách có hơn 3000 đánh giá đều có hơn 4,7 xếp hạng trung bình.

### The more pages, the more expensive?

fig = plt.subplots(figsize=(10, 7))

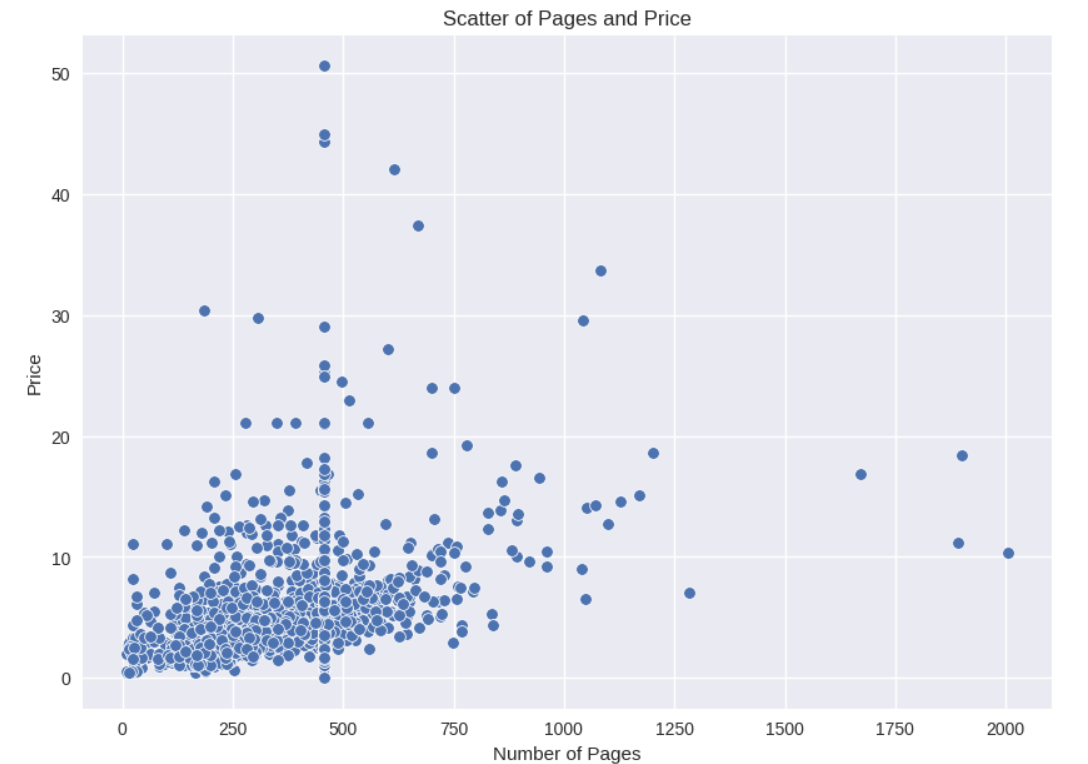
ax=sns.scatterplot(data=eda\_df, x='pages', y='current\_price')#, color="",color\_continuous\_scale='RdBu\_r')

plt.xlabel('Number of Pages')

plt.ylabel('Price')

plt.title('Scatter of Pages and Price')

plt.show()



Vẫn có một số quyển sách có nhiều trang nhưng giá thấp và một số quyển có ít trang nhưng giá cao.

### What are the largest authors?

Sử dụng **quantity**, **n\_review** và **avg\_rating**, giả định rằng các tác giả nằm trong top 5 của các dataframe đó.

def sort\_authors\_by\_col(col):

res = eda\_df.groupby('authors')[col]\

.mean()\

.sort\_values(ascending=False)\

.index.tolist()

return res

quantity\_lst = sort\_authors\_by\_col("quantity")

n\_reviews\_lst = sort\_authors\_by\_col("n\_review")

avg\_rating\_lst = sort\_authors\_by\_col("avg\_rating")

def top\_k\_common\_elements(list1, list2, list3, k=10):

list1, list2, list3 = list1[:k], list2[:k], list3[:k]

return list(set(list1) & set(list2) & set(list3))

result = top\_k\_common\_elements(quantity\_lst, n\_reviews\_lst, avg\_rating\_lst, 100)

result

Sau khi thực hiện đã tìm thấy 3 tác giả trong top 100 giả định được xác định:

* Jack Canfield & DD.Watkins
* GEORGE SAMUEL CLASON
* Thái Phạm

### What is the best seller book?

Để biết được quyển sách nào được bán chạy nhất ta sử dụng **quantity**, **n\_review** và **avg\_rating**. Tiếp tục giả định quyển sách luôn nằm trong top 5 của các dataframe trên.

def sort\_title\_by\_col(col):

# I used a trick here, because we group by title (without duplicated value).

# The aim is to map with title against.

res = eda\_df.groupby("title")[col].max()\

.sort\_values(ascending=False)\

.index.tolist()

return res

quantity\_lst = sort\_title\_by\_col("quantity")

n\_reviews\_lst = sort\_title\_by\_col("n\_review")

avg\_rating\_lst = sort\_title\_by\_col("avg\_rating")

def top\_k\_common\_elements(list1, list2, list3, k=10):

list1, list2, list3 = list1[:k], list2[:k], list3[:k]

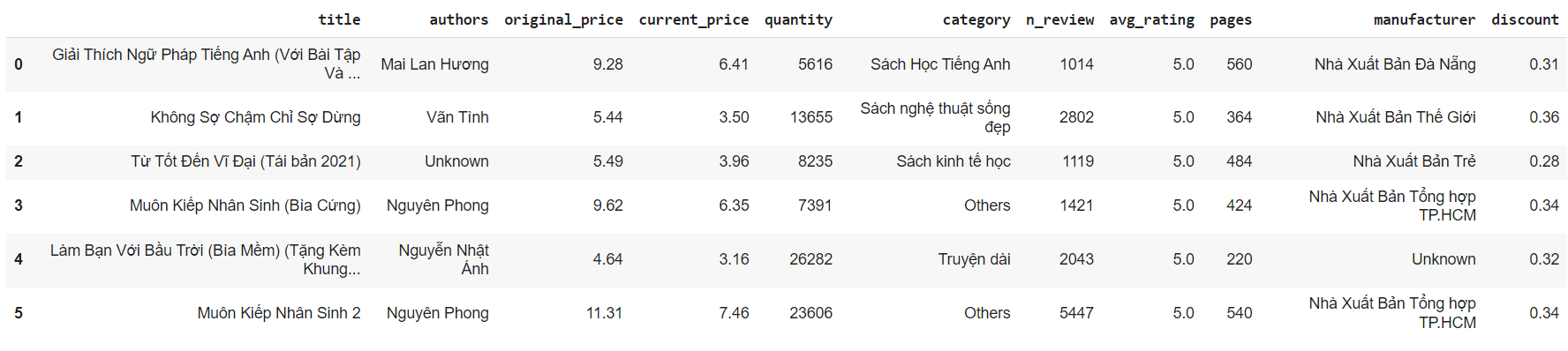
return list(set(list1) & set(list2) & set(list3))

result = top\_k\_common\_elements(quantity\_lst, n\_reviews\_lst, avg\_rating\_lst, 200)

result

eda\_df.loc[eda\_df.title.isin(result)].reset\_index(drop=True)

Tìm được 5 quyển sách trong số sách giả định được xác định:



* **Correlation between price and category (Mối tương quan giữa giá cả và thể loại):** Sử dụng thuật toán **KMeans** để phân cụm.

# Tiền xử lý dữ liệu: chuyển cụm thể loại thành số

encoder = LabelEncoder()

correlation\_df = eda\_df

correlation\_df['CategoryEncoded'] = encoder.fit\_transform(correlation\_df['category'])

# Sử dụng thuật toán KMeans để phân cụm

num\_clusters = 5

kmeans = KMeans(n\_clusters=num\_clusters, random\_state=42)

kmeans.fit(correlation\_df[['CategoryEncoded']])

# Thêm cột phân cụm vào dữ liệu

correlation\_df['Cluster'] = kmeans.labels\_

# In ra một số thông tin về các cụm

for cluster\_id in range(num\_clusters):

cluster\_data = correlation\_df[correlation\_df['Cluster'] == cluster\_id]

print(f"Cluster {cluster\_id} - Size: {len(cluster\_data)}")

print(cluster\_data['category'].value\_counts())

print()

# Vẽ biểu đồ phân cụm

plt.figure(figsize=(12, 6))

plt.scatter(correlation\_df['CategoryEncoded'], correlation\_df['original\_price'], c=correlation\_df['Cluster'], cmap='rainbow')

plt.title('Book Clustering based on Categories and Prices')

plt.xlabel('Category')

plt.ylabel('Price')

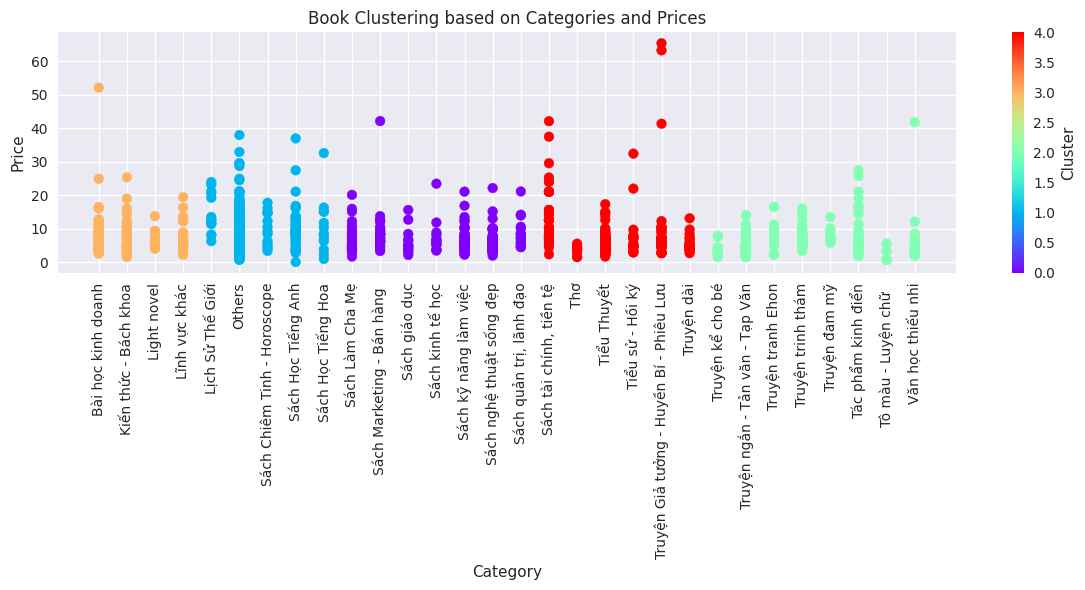
plt.xticks(range(len(encoder.classes\_)), encoder.classes\_, rotation=90)

plt.colorbar(label='Cluster')

plt.tight\_layout()

plt.show()

Nhìn biểu đồ để có cái nhìn tổng quan:



## Xóa dữ liệu ngoại lai (outliner)

**Sử dụng phương pháp IQR (Interquartile Range)**: Đây là phương pháp sử dụng khoảng giữa phần một (Q1) và phần ba (Q3) của dữ liệu. Các giá trị nằm ngoài khoảng từ **Q1 - 1.5 \* IQR** đến **Q3 + 1.5 \* IQR** được xem là outlier và có thể bị loại bỏ.

Xóa outliner cho **original\_price**

**# Tính Q1 và Q3**

**Q1 = data\_df['original\_price'].quantile(0.25)**

**Q3 = data\_df['original\_price'].quantile(0.75)**

**# Tính IQR**

**IQR = Q3 - Q1**

**# Xác định ngưỡng để xác định outlier**

**lower\_bound = Q1 - 1.5 \* IQR**

**upper\_bound = Q3 + 1.5 \* IQR**

**# Loại bỏ outlier**

**data\_df = data\_df[(data\_df['original\_price'] >= lower\_bound) & (data\_df['original\_price'] <= upper\_bound)]**

**Tương tự cho pages**

**# Tính Q1 và Q3**

**Q1 = data\_df['pages'].quantile(0.25)**

**Q3 = data\_df['pages'].quantile(0.75)**

**# Tính IQR**

**IQR = Q3 - Q1**

**# Xác định ngưỡng để xác định outlier**

**lower\_bound = Q1 - 1.5 \* IQR**

**upper\_bound = Q3 + 1.5 \* IQR**

**# Loại bỏ outlier**

**data\_df = data\_df[(data\_df['pages'] >= lower\_bound) & (data\_df['pages'] <= upper\_bound)]**

# 

# XÂY DỰNG MÔ HÌNH

## Mô hình phân loại văn bản :

### Công cụ khai phá dữ liệu:

Sử dụng mạng nơ-ron biến đổi ngôn ngữ (BERT) trong TensorFlow để phân loại văn bản (**chương trình chạy trên** [**ggcolab**](https://colab.research.google.com/drive/1p850I2ih99FgJKHU7q-0DvlN5NRdyVpd?usp=sharing))

### Đặc tả bài toán

Dựa vào file [comment.csv](https://drive.google.com/file/d/1QaKdTIi4gwTj_-PS7U4RjVFrgrcBl8le/view?usp=drive_link) lấy các đặc trưng: **rating, content.**

df = pd.read\_csv('/content/drive/MyDrive/DataMining/BTL/data/comments.csv',

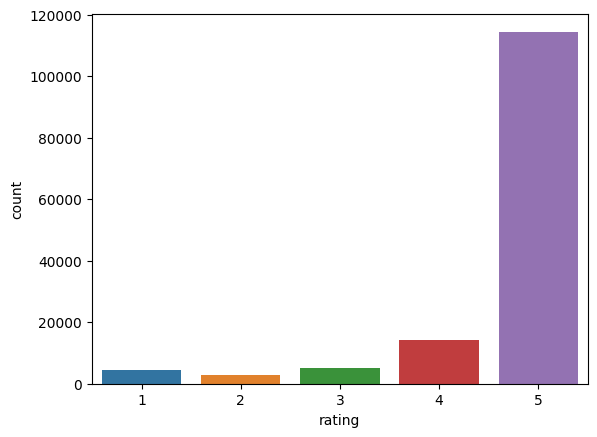
usecols = ['rating', 'content'])

#### Chuẩn bị dữ liệu:

print(f'Len of train dataset: {len(df)}')

output: Len of train dataset: 141281

sns.countplot(x = df.rating);



=> Thấy được lượt đánh giá 5 rất nhiều ( chiếm phần lớn). Việc này sẽ ảnh hưởng đến kết quả của mô hình về sau.

#### Kiểm tra các giá trị thiếu và xóa:

df.isna().sum()

df = df.dropna()

df.shape

output: (103263, 2)

=> 103263 mẫu và 2 đặc trưng

#### 

#### Gán cột content cho X và cột rating cho y:

X = df.content.copy()

y = df.rating.copy()

=> y là target value cho mô hình

#### Chuyển đổi y thành dạng ma trận nhị phân (one-hot encoding):

y = tf.keras.utils.to\_categorical(y)

y.shape

output: (103263, 6)

=> sau khi chuyển đổi y lúc này có 103263 mẫu và 6 đặc trưng

#### TPU setup:

Xác định và tạo các chiến lược phân phối dữ liệu và tính toán phù hợp cho môi trường huấn luyện dựa trên sự có mặt của TPU, GPU hoặc CPU trong hệ thống.

try:

tpu = tf.distribute.cluster\_resolver.TPUClusterResolver()

tf.config.experimental\_connect\_to\_cluster(tpu)

tf.tpu.experimental.initialize\_tpu\_system(tpu)

strategy = tf.distribute.experimental.TPUStrategy

except ValueError:

strategy = tf.distribute.get\_strategy()

print('Number of replicas:', strategy.num\_replicas\_in\_sync)

try:

tpu = tf.distribute.cluster\_resolver.TPUClusterResolver() # TPU detection

except ValueError:

tpu = None

gpus = tf.config.experimental.list\_logical\_devices("GPU")

if tpu:

tf.tpu.experimental.initialize\_tpu\_system(tpu)

strategy = tf.distribute.experimental.TPUStrategy(tpu,)

print('Running on TPU ', tpu.cluster\_spec().as\_dict()['worker'])

elif len(gpus) > 1:

strategy = tf.distribute.MirroredStrategy([gpu.name for gpu in gpus])

print('Running on multiple GPUs ', [gpu.name for gpu in gpus])

elif len(gpus) == 1:

strategy = tf.distribute.get\_strategy()

print('Running on single GPU ', gpus[0].name)

else:

strategy = tf.distribute.get\_strategy()

print('Running on CPU')

print("Number of accelerators: ", strategy.num\_replicas\_in\_sync)

=> Sau khi chạy đoạn chương trình trên ggcolab, thì sẽ sử dụng TPU có sẵn của ggcolab để chạy mô hình.

#### Build Model:

MAX\_LEN = 200

model\_name = 'vinai/phobert-base'

# Tokenizing

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

inputs = [tokenizer(item,

max\_length = MAX\_LEN,

padding = 'max\_length',

truncation = True,

return\_tensors = 'np')['input\_ids'].reshape(MAX\_LEN)

for \_, item in X.items()]

=> Chuyển đổi dữ liệu văn bản thành input IDs thông qua tokenizer của mô hình PhoBERT và áp dụng cắt hoặc padding để đảm bảo độ dài của mỗi câu là tối đa 200.

Tách df train, test theo tỉ lệ 8:2:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(np.asarray(inputs),

y,

random\_state=1905,

test\_size=0.2,

shuffle=True)

Build:

with strategy.scope():

encoder = TFAutoModel.from\_pretrained(model\_name)

input\_ids = Input(shape=(MAX\_LEN,), dtype=tf.int32)

embedding = encoder(input\_ids)[1]

x = Dense(128, activation = 'relu', kernel\_regularizer = regularizers.L2(0.1))(embedding)

x = Dropout(0.2)(x)

x = Dense(y.shape[1],

activation = 'softmax',

kernel\_initializer=tf.initializers.GlorotUniform(seed=1905),

name='output\_layer')(x)

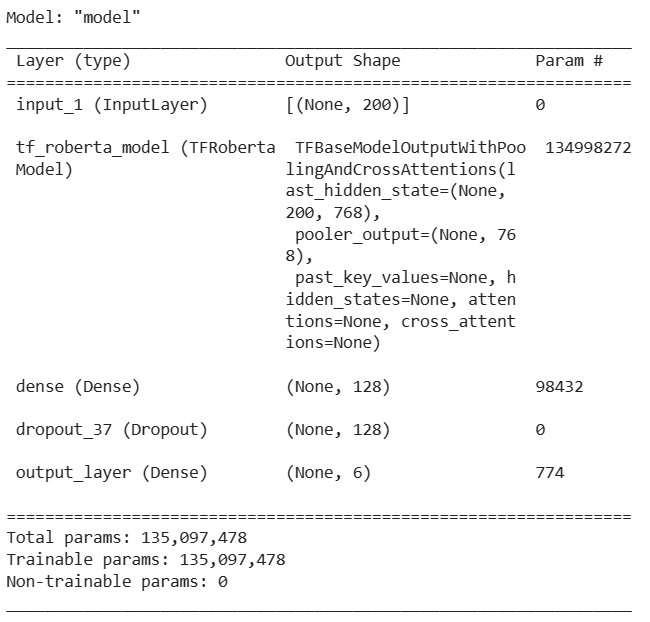
model = Model(inputs=[input\_ids], outputs = x)

=> Sử dụng mô hình PhoBERT đã được huấn luyện sẵn để trích xuất thông tin về nội dung của đoạn văn bản. Sau đó, đưa qua một vài lớp mạng neural như Dense và Dropout để học cách phân loại đánh giá. Cuối cùng, xây dựng một mô hình toàn cục thông qua việc định nghĩa các lớp đầu vào và đầu ra.

Bản tóm tắt của mô hình:

model.summary()

output:



Tạo một lịch trình học tập (learning rate schedule) và mức trọng số giảm (weight decay):

step = tf.Variable(0, trainable=False)

schedule = tf.optimizers.schedules.PiecewiseConstantDecay(

[10000, 15000], [1e-3, 1e-4, 1e-5])

lr = 1e-2 \* schedule(step)

wd = lambda: 1e-5 \* schedule(step)

=> Mục đích của việc sử dụng lịch trình học tập và trọng số giảm là để điều chỉnh learning rate và trọng số giảm theo thời gian, giúp mô hình học tốt hơn và tránh overfitting trong quá trình huấn luyện.

Thiết lập các thông số của mô hình:

model.compile(

optimizer=tfa.optimizers.AdamW(weight\_decay=wd,

learning\_rate=lr),

loss=tf.keras.losses.CategoricalCrossentropy(),

metrics=['accuracy'])

Huấn luyện mô hình:

history = model.fit(X\_train,

y\_train,

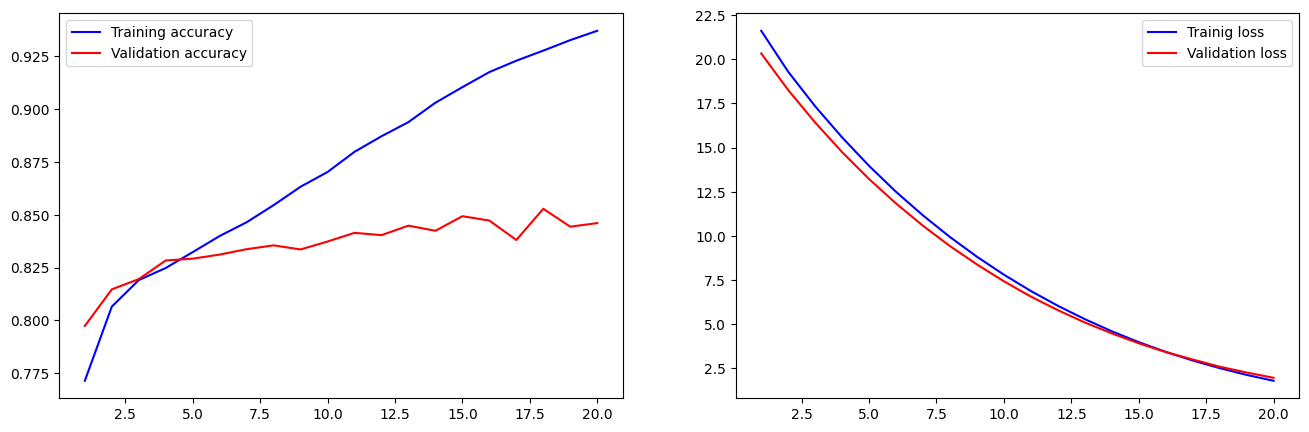
epochs = 20,

verbose = 1,

validation\_split = 0.1,

batch\_size = 256)

Đồ thị biểu diễn sự thay đổi của độ chính xác và hàm mất mát trên tập huấn luyện và tập kiểm tra qua các epochs:



=> Mô hình có khả năng học tốt từ tập dữ liệu huấn luyện và cải thiện kết quả theo số lần thực hiện. Mô hình có xu hướng bị overfitting, tức là nó hoạt động tốt trên tập dữ liệu huấn luyện nhưng không hoạt động tốt trên tập dữ liệu mới hoặc chưa biết. Điều này có thể được nhận biết qua sự chênh lệch giữa hai đường xanh và đỏ trong biểu đồ bên trái, và hai đường đỏ và tím trong biểu đồ bên phải, nhưng sự chênh lệ này không quá lớn. Điều này cho thấy mô hình không bị overfitting nghiêm trọng, và mức độ overfitting có vẻ khá ổn định.

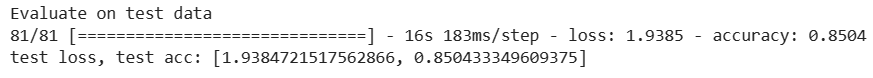
Đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra:

print("Evaluate on test data")

results = model.evaluate(X\_test, y\_test, batch\_size=256)

print("test loss, test acc:", results)

output:



=> Mô hình đã có kết quả tốt trên tập dữ liệu kiểm tra, với mức loss thấp và độ chính xác cao. Kết quả này cho thấy mô hình có khả năng phân loại các dữ liệu đầu vào (câu nhận xét) thành các lớp đúng (các mức đánh giá) với hiệu suất tương đối tốt. Tuy nhiên, muốn đưa mô hình vào thực tế chúng ta cần có thêm dữ liệu để huấn luyện mô hình tránh bị overfitting và có thể cung cấp một cái nhìn toàn diện hơn về hiệu suất thực sự của mô hình.

## Mô hình dự đoán giá sách:

### Công cụ khai phá dữ liệu

Sử dụng mạng nơ-ron nhân tạo (ANN) để dự đoán giá sách (**chương trình chạy trên** [**ggcolab**](https://colab.research.google.com/drive/1j4KZrD-tD3SRtkBDcQwl1vK7-Onx4WJS?authuser=2))

### Đặc tả bài toán

Dựa vào file [prepared\_data\_book.csv](https://drive.google.com/file/d/1n1oV-0TxIorBfDOFq94tqxe4v8JAXfet/view?usp=drive_link) lấy các đặc trưng: **authors, category, pages, original\_price.**

columns\_to\_keep = ['authors', 'category', 'pages', 'original\_price']

df=pd.read\_csv('/content/drive/MyDrive/DataMining/BTL/data/prepared\_data\_book.csv', usecols=columns\_to\_keep)

#### Chuẩn bị Dữ liệu:

X = df[['authors', 'category', 'pages']]

y = df['original\_price']

=> y là target value cho mô hình

X\_encoded = pd.get\_dummies(X)

=> Mã hóa các thuộc tính categorical trong X thành dạng one-hot encoding. Điều này giúp biến các thuộc tính này thành dạng số học mà mô hình có thể hiểu được.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_encoded, y, test\_size=0.2, random\_state=42)

=> Tách df train, test theo tỉ lệ 8:2.

#### Chuẩn hóa Dữ liệu:

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

=> Sau khi chuẩn hóa, dữ liệu đầu vào đã có dạng chuẩn hóa với trung bình bằng 0 và độ lệch chuẩn bằng 1, giúp mô hình dễ dàng học và thích nghi với các biến đầu vào.

#### Xây dựng Mô hình ANN

model = keras.Sequential([

layers.Input(shape=(X\_train\_scaled.shape[1],)),

layers.Dense(64, activation='relu'),

layers.Dense(32, activation='relu'),

layers.Dense(1)

])

model.compile(optimizer='adam', loss='mean\_squared\_error')

=> Xây dựng một mô hình mạng nơ-ron với ba layer fully connected và hàm kích hoạt ReLU, được huấn luyện bằng thuật toán Adam với mục tiêu là giảm thiểu sai số dự đoán bằng cách tối thiểu hóa hàm mất mát MSE.

#### Huấn luyện Mô hình

history = model.fit(X\_train\_scaled,

y\_train,

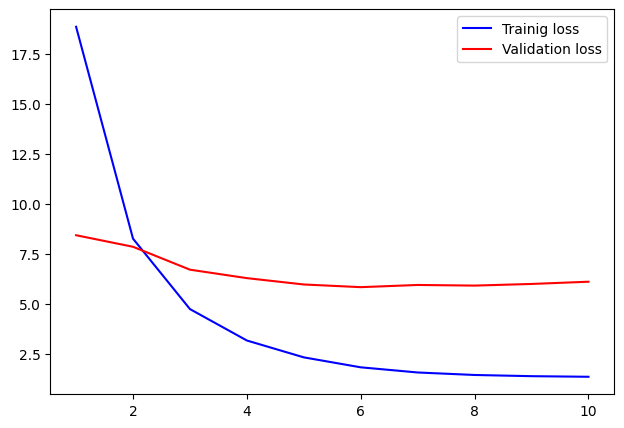
epochs=5,

verbose = 1,

batch\_size=32,

validation\_split=0.1)

Biểu đồ biểu thị mất mát (loss) trên tập dữ liệu đào tạo và tập dữ liệu kiểm tra (validation loss) qua các epoch:



=> Sự biến đổi của mất mát kiểm tra không quá lớn và không có dấu hiệu rõ ràng của overfitting

#### Đánh giá Mô hình

y\_pred = model.predict(X\_test\_scaled)

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

output:



=> Điều này cho thấy mô hình có thể dự đoán giá sách với sai số trung bình khá thấp trên tập dữ liệu kiểm tra.

#### Dự đoán giá sách cho toàn bộ dữ liệu

X\_all\_scaled = scaler.transform(X\_encoded)

predicted\_prices = model.predict(X\_all\_scaled)

df['predicted\_price\_ann'] = predicted\_prices

df[['original\_price', 'predicted\_price\_ann']]

output:

