ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ

Thành viên nhóm:

Nguyễn Quốc Khoa - 20120511
Trần Đắc Khoa - 2012013

MÔN: Phân tích dữ liệu thông minh

Thành phố Hồ Chí Minh – 2023

Mục lục

Giới thiệu

- Dữ liệu được lấy từ Kaggle. Đây là một DataBook của sàn thương mại điện tử TiKi tại Việt Nam. Tập dữ liệu này chứa thông tin về 2024 cuốn sách bán chạy nhất trên TIKI: Trong đồ án cuối kì lần này, nhóm tập trung chủ yếu về dữ liệu về các cuốn sách (book_data.csv). Bên cạnh đó, tập dữ liệu này bao gồm:

+ Thông tin về sách;

+ Liên kết cho bìa sách;

+ Khoảng 50 bình luận cho mỗi cuốn sách

+ …

- License: : CC0 1.0 Universal (CC0 1.0)Public Domain Dedication

* Đặc biệt ta sẽ làm việc chủ yếu trên file book_data.csv. Data bao gồm các trường dữ liệu như sau:

product_id: id của sản phẩm trong cơ sở dữ liệu Tiki (duy nhất)

title: tiêu đề của cuốn sách, có thể chứa thời gian xuất bản lại

authors: tác giả

original_price: giá gốc

current_price: giá hiện tại nếu có chiết khấu

quantity: số sách đã bán mọi thời đại

category: thể loại sách

n_review: số lượng đánh giá

avg_rating: xếp hạng trung bình (tối đa 5,0)

pages: tổng số trang của mỗi cuốn sách

Tiền xử lý dữ liệu

Cài đặt thư viện

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import plotly.express as px
import cv2
from PIL import Image

data_df = pd.read_csv("prepared_data_book.csv")
```

Tải dữ liệu

- Ở phần này, ta thấy được 1796 cuốn sách không trùng lặp. Nhưng trong id_df, Ta lại có 2024 id. Lý do cho vấn đề này là trong bộ dữ liệu có một số vật phẩm không phải là sách, dù vậy thì số lượng của chúng khá ít và không đáng quan tâm.

```
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1796 entries, 0 to 1795
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   product_id      1796 non-null   int64  
 1   title           1796 non-null   object 
 2   authors         1653 non-null   object 
 3   original_price  1796 non-null   int64  
 4   current_price   1796 non-null   int64  
 5   quantity        1751 non-null   float64
 6   category        1796 non-null   object 
 7   n_review        1796 non-null   int64  
 8   avg_rating      1796 non-null   float64
 9   pages           1546 non-null   object 
 10  manufacturer    1523 non-null   object 
 11  cover_link      1796 non-null   object 
dtypes: float64(2), int64(4), object(6)
memory usage: 168.5+ KB
```

Xử lý dữ liệu:

Xử lý dữ liệu bị thiếu hoặc khiếm khuyết:

```
data_df.isnull().sum()

product_id          0
title               0
authors           143
original_price      0
current_price       0
quantity           45
category            0
n_review            0
avg_rating          0
pages             250
manufacturer      273
cover_link          0
dtype: int64
```

- Đầu tiên, ta phải xóa các mục bị trùng lặp, ở phần tác giả, ta nhận thấy rằng có một số tác giải khuyết danh đại diện bằng dấu . nên ta sẽ thay thế tên của họ bằng “Unknown”.

```
data_df = data_df.drop_duplicates(subset=['title'])
```

```
[10] data_df.authors.value_counts()

Nguyễn Nhật Ánh    24
Higashino Keigo    20
.                  18
Thích Nhất Hạnh    16
Haruki Murakami    15
                   ..
Urako Kanamori      1
Cổ Viên             1
Robert Winston      1
Yongchul Kwon       1
John C. Maxwell     1
Name: authors, Length: 1083, dtype: int64
```

Xóa các tác giả không đích danh, đại diện bằng dấu "."

```
[11] data_df.loc[data_df.authors == '.' , 'authors'] = "Unknown"
```

- Tiếp theo là về loại sách, vì trong mục category ta thấy có nhiều loại sách không rõ ràng nên ta không thể xét hết, chỉ ưu tiên xét 30 loại sách có số lượng nhiều nhất

```
0s  data_df.category.value_counts()

    Sách tư duy - Kỹ năng sống                292
    Tiểu Thuyết                               133
    Truyện ngắn - Tản văn - Tạp Văn           109
    Sách nghệ thuật sống đẹp                   58
    Sách kỹ năng làm việc                      55
                                              ...
    Shaman King - Tập 19                        1
    Blue Period - Tập 07                        1
    Komi - Nữ Thần Sợ Giao Tiếp - Tập 14        1
    Bên Rặng Tuyết Sơn (Tái Bản)                1
    Kaguya-Sama: Cuộc Chiến Tỏ Tình - Tập 3     1
    Name: category, Length: 354, dtype: int64
```

- Ở mục page , tức số trang cũng có một trường hợp bị sai, nhóm đã kịp thời chỉnh sửa và bổ sung lại

- Ta tạo thêm một cột tên “discount” để thể hiện % số tiền giảm giá, và cân nhắc việc chuyển đổi giữa VND sang USD để con số phù hợp hơn cho việc tính toán.

```
[31] 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))
```

- Cuối cùng ta xem lại loại dữ liệu của các trường dữ liệu trong dataset này, cân nhắc chuyển đổi một số thông tin như “author”, “title”, “categorical”, .. thành loại string. Và cuối cùng lưu lại data đã tiền xử lý thành 1 file csv mới tên là : “prepared_data_book”.

2. Phân tích dữ liệu

Cài đặt thư viện

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import plotly.express as px
import cv2
from PIL import Image

data_df = pd.read_csv("prepared_data_book.csv")
```

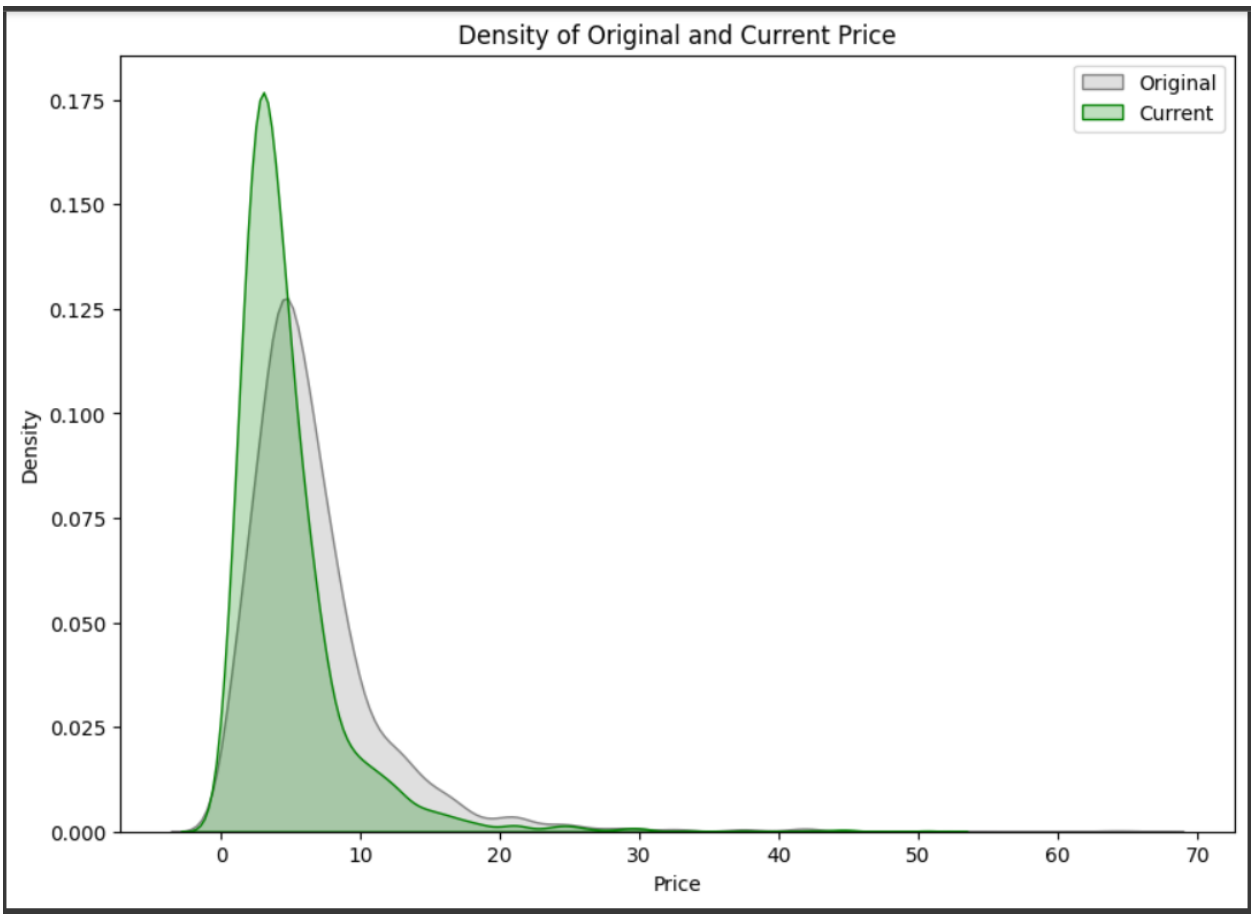
Phân tích sơ lược về tổng thể dữ liệu.

- Ta sẽ drop 2 thành phần đó là “product_id” và “ cover_link” để việc trực quan có thể dễ dàng hơn.

1. Giá

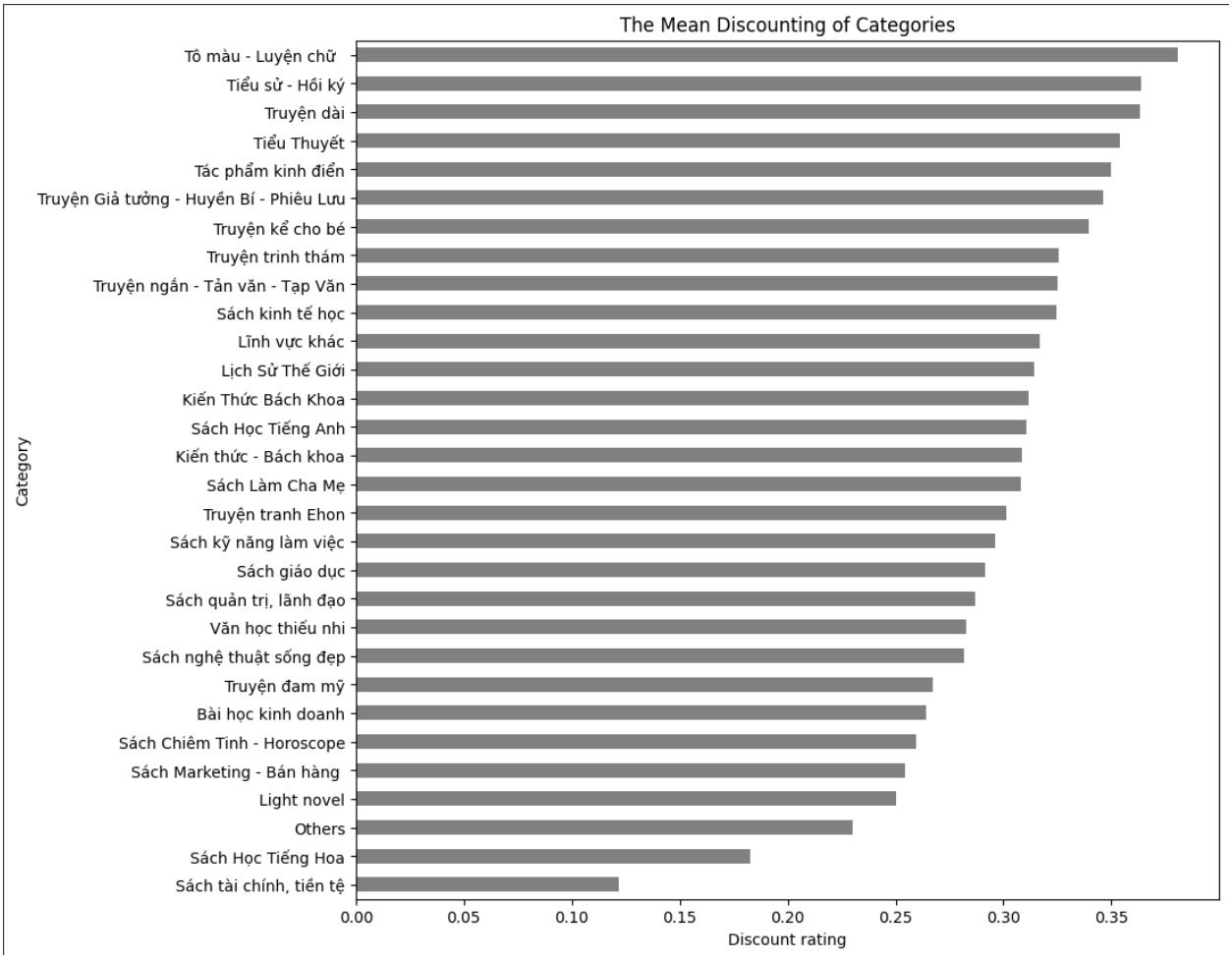
Đầu tiên ta sẽ đánh giá tổng quan về giá của những quyển sách trong dataset này.

```
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()
```

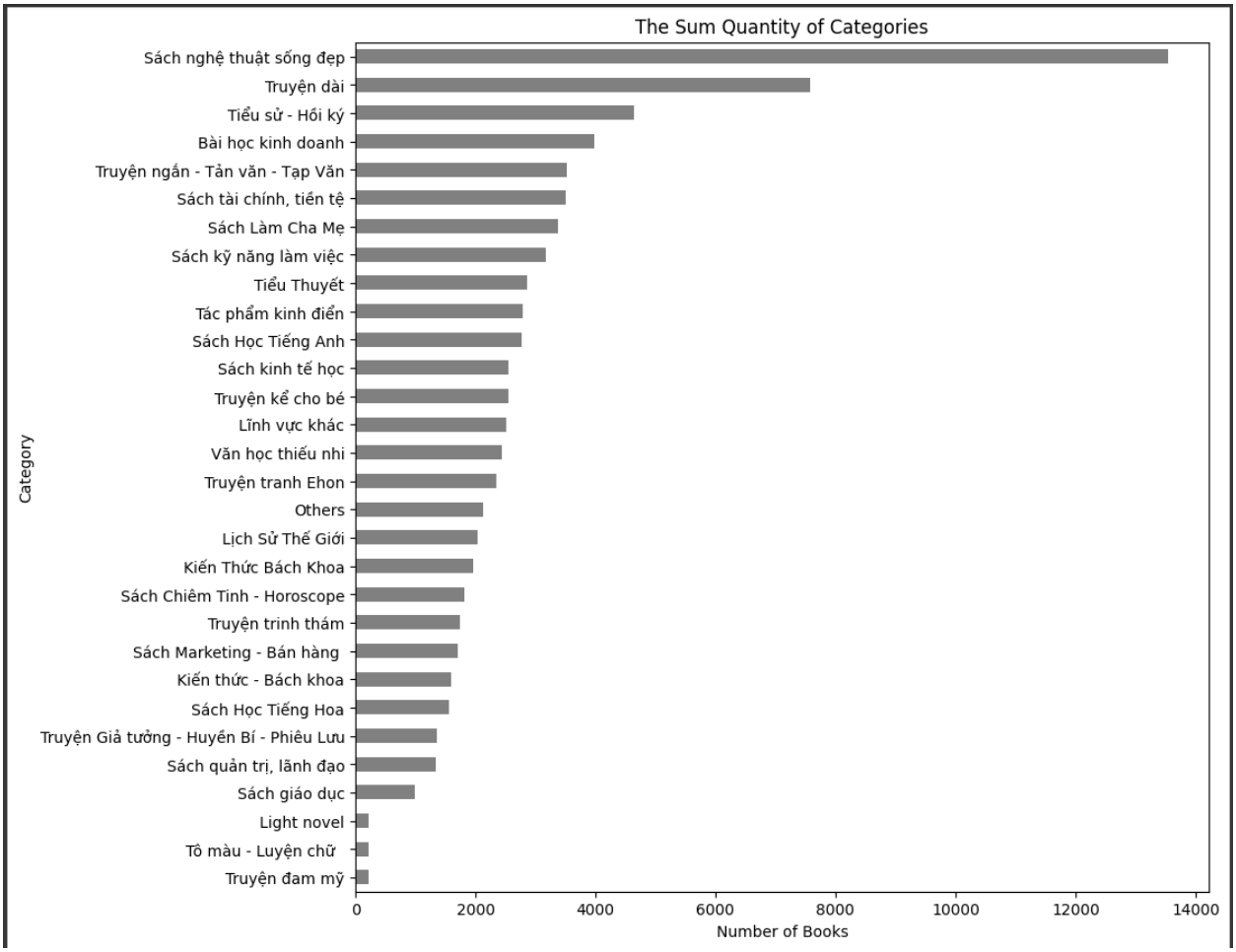


- Đoạn code trên dùng để trực quan một biểu đồ mật độ của giá sách ban đầu (original_price) và giá sách hiện tại (current_price) từ một bộ dữ liệu gọi là eda_df sử dụng thư viện seaborn để vẽ. Qua biểu đồ này ta có thể thấy giá của hầu hết những cuốn sách đều tầm dưới 20 đô. Chứng tỏ rằng giá sách ở Việt Nam khá rẻ.

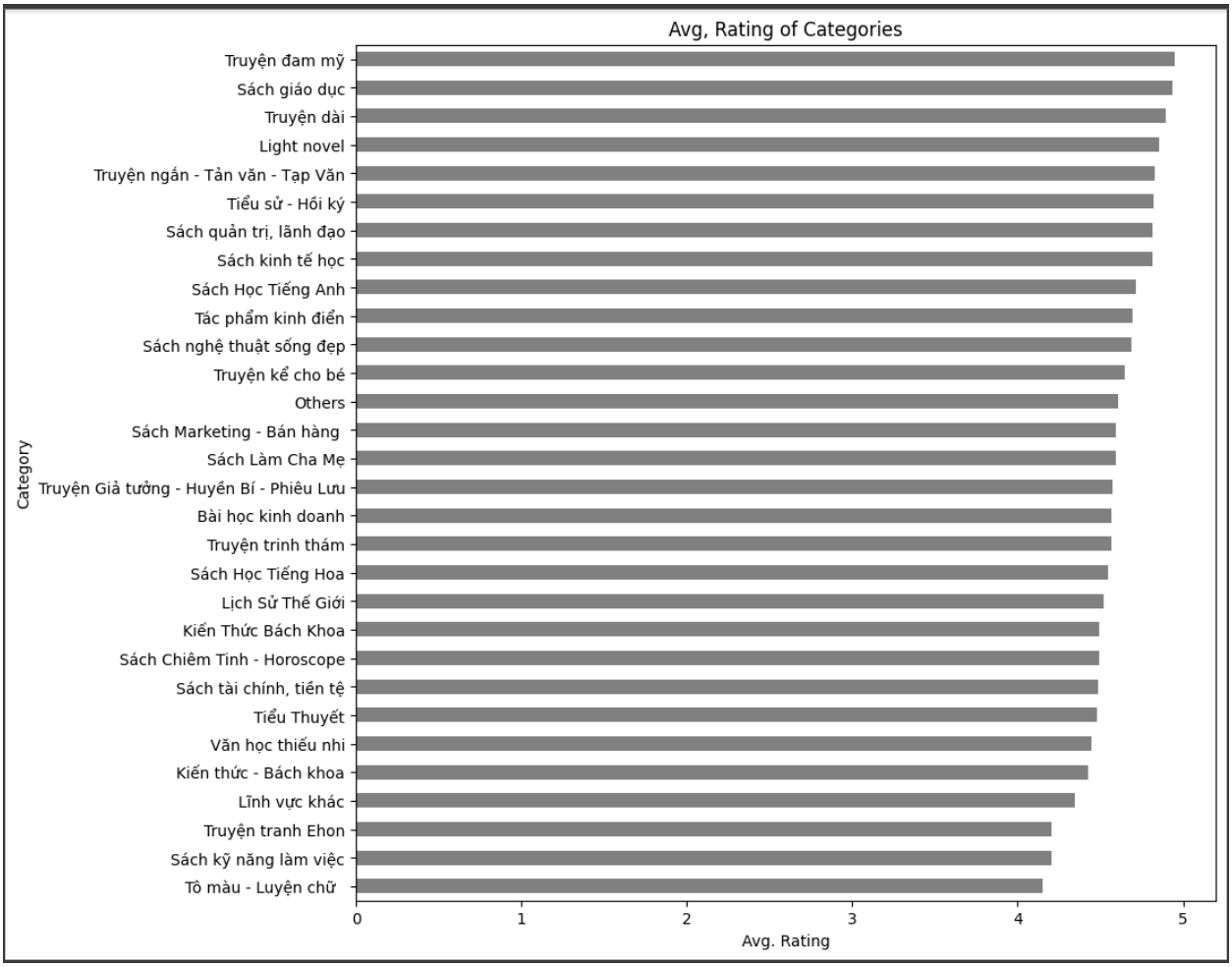
- Tiếp theo ta sẽ xem từng loại sách, mỗi loại sẽ có lượng discount bao nhiêu %



- Đa phần các loại sách giảm khoảng từ 10 % -> 40 %. Đặc biệt các cuốn sách liên quan đến các bé mầm non được giảm sâu nhất. Mặc dù giảm sâu như vậy, nhưng ta vẫn cần xem thử liệu rằng những cuốn sách đó có thực sự bán chạy không?



- Khá bất ngờ, mặc dù được giảm giá sâu nhất nhưng những cuốn sách như vậy lại không được người tiêu dùng mua nhiều, điều đó cũng có thể hiểu rằng việc chiến lược giảm giá để kích cầu có vẻ hoạt động không được tốt lắm.

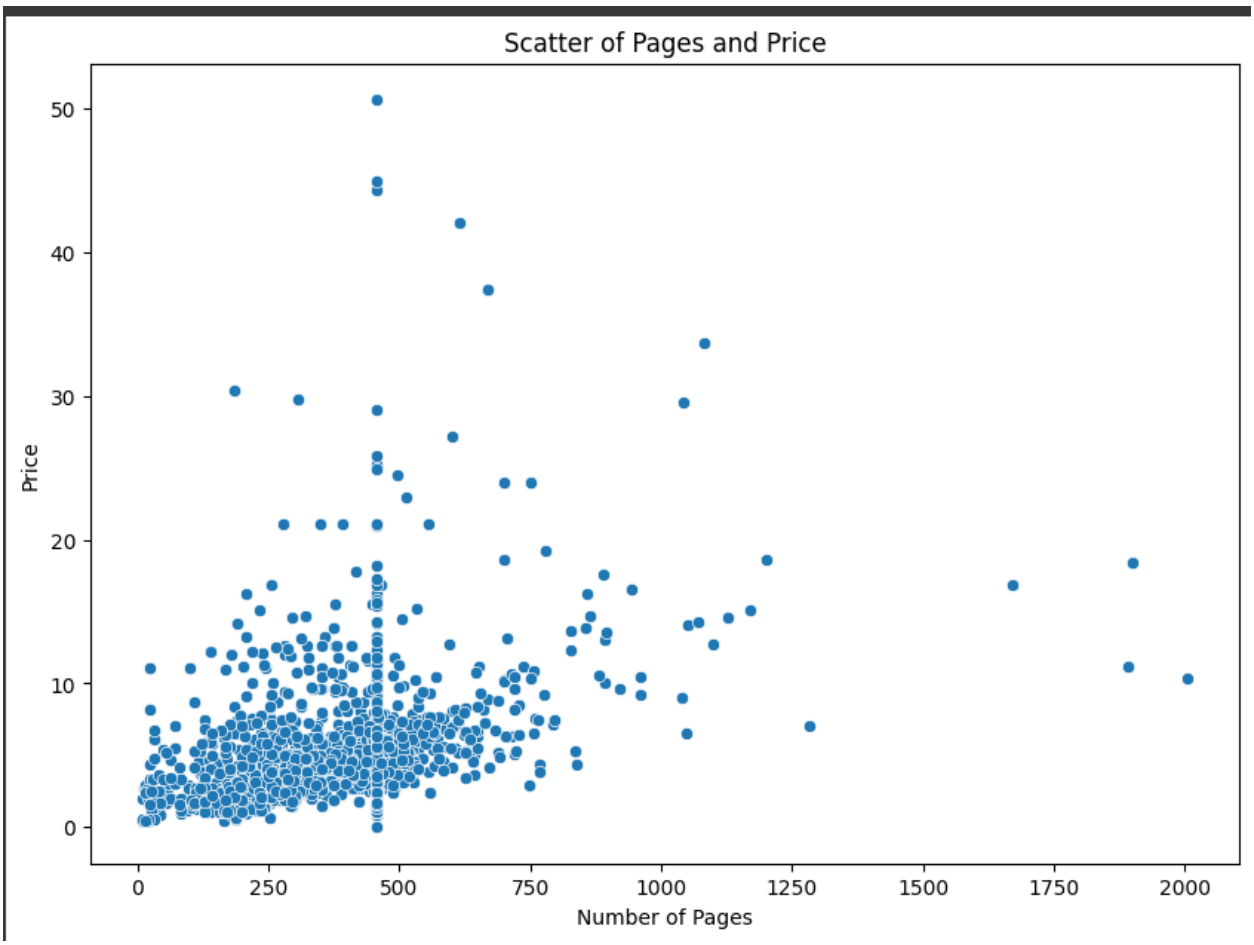


- Cuối cùng ta xem đánh giá của người tiêu dùng với từng loại sách. Các đánh giá đều ở mức tốt, rating đều trên 4. Có thể thấy chất lượng sách TIKI khá tốt và có thể làm hài lòng người mua hàng.

Đặt câu hỏi ?

Bây giờ ta sẽ đi qua một số câu hỏi đơn giản về dữ liệu trên.

- Câu 1: Có phải càng nhiều trang thí giá sách càng đắt ?



- Đây chính là biểu đồ tương quan giữa số lượng trang và giá sách, Qua đó ta có thể thấy, những cuốn sách có giá tiền và số trang hầu như không liên quan nhiều tới nhau. Có một số cuốn sách có khá nhiều trang nhưng giá thấp và ngược lại. Vì vậy kết luận trên là chưa đủ căn cứ.

Bây giờ ta sẽ chuyển sang câu hỏi tiếp theo:

- Câu 2: Tác giả nào có tầm ảnh hưởng nhất

Dựa vào các yếu tố, ta sẽ dựa vào ba tiêu chí: Quantity, Number of Reviews, Average Rating để chọn ra các tác giả có tầm ảnh hưởng lớn nhất

```
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
```

Cuối cùng kết quả là ta đã tìm thấy ba tác giả:

- Jack Canfield & Đ.D.Watkins
- LỚP GEORGE SAMUEL
- Thái Phạm

Và qua đó ta cũng xem được một số tác phẩm bán chạy nhất của ba nhà văn trên. Có thể kể đến như: Người Nam Châm, hiết Kế Cuộc Đời Thịnh Vượng, Người Giàu Có Nhất Thành Babylon.

Tiếp theo ta sẽ tìm hiểu đến những cuốn sách:

Câu 3: Những cuốn sách nào bán chạy nhất:

```
[ ]  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
```

Cũng dựa vào 3 tiêu chí, đó là Quantity, Number of Reviews, Average Rating. Ta cũng đã tìm ra được những cuốn sách bán chạy nhất, cụ thể như sau:

0	81	Giải Thích Ngữ Pháp Tiếng Anh (Với Bài Tập Và ...	Mai Lan Hương	9.28	6.41	5616	Sách Học Tiếng Anh	1014	5.0	560	Nhà Xuất Bản Đà Nẵng	0.31
1	128	Không Sợ Chậm Chỉ Sợ Dừng	Văn Tinh	5.44	3.50	13655	Sách nghệ thuật sống đẹp	2802	5.0	364	Nhà Xuất Bản Thế Giới	0.36
2	140	Từ Tốt Đến Vĩ Đại (Tái bản 2021)	Unknown	5.49	3.96	8235	Sách kinh tế học	1119	5.0	484	NXB Trẻ	0.28
3	289	Muôn Kiếp Nhân Sinh (Bìa Cứng)	Nguyên Phong	9.62	6.35	7391	Others	1421	5.0	424	Nhà Xuất Bản Tổng hợp TP.HCM	0.34
4	407	Làm Bạn Với Bầu Trời (Bìa Mềm) (Tặng Kèm Khung...	Nguyễn Nhật Ánh	4.64	3.16	26282	Truyện dài	2043	5.0	220	Unknown	0.32
5	503	Muôn Kiếp Nhân Sinh 2	Nguyên Phong	11.31	7.46	23606	Others	5447	5.0	540	Nhà Xuất Bản Tổng hợp TP.HCM	0.34

Bây giờ chúng ta sẽ đi qua phần chính của đồ án này, các vấn đề được đặt ra, trong báo cáo này, nhóm chúng em đã chọn ra 3 vấn đề. Cụ thể thế nào? nhóm chúng em xin phép được trình bày bên dưới

Vấn đề 1: Các yếu tố nào ảnh hưởng đến rating của những cuốn sách và xây dựng mô hình để dự đoán rating ?

1. Sử dụng mô hình phi noron để giải quyết vấn đề

- Đầu tiên, ta vẫn sẽ tiến hành các bước xử lý dữ liệu để có thể đưa vào mô hình, cụ thể như sau:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Load dataset
data = pd.read_csv('prepared_data_book.csv')

# Bỏ các trường dữ liệu không cần thiết
data.drop(columns=['Unnamed: 0', 'product_id', 'title', 'cover_link'], inplace=True)

# Bỏ các giá trị bị thiếu
data.dropna(inplace=True)

# Encode các biến categorical thành numeric
label_encoder = LabelEncoder()
data['category_encoded'] = label_encoder.fit_transform(data['category'])
data['manufacturer_encoded'] = label_encoder.fit_transform(data['manufacturer'])
data['authors_encoded'] = label_encoder.fit_transform(data['authors'])

# Vì đã encode nên ta sẽ bỏ đi các cột dữ liệu củ
data.drop(columns=['category', 'manufacturer', 'authors'], inplace=True)
```

- Tiếp sau đó ta sẽ đưa bộ dữ liệu đã được xử lý vào mô hình hồi quy tuyến tính (ở đây ta sẽ dùng mô hình của thư viện sklearn)

```python
# Split data thành dữ liệu x và target y
X = data.drop(columns=['avg_rating'])
y = data['avg_rating']

# Chia thành tập train và tập test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Ở phần mở đầu này ta sẽ dùng hồi quy tuyến tính ( linear regression)

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Khởi tạo mô hình
model = LinearRegression()

# Huấn luyện
model.fit(X_train, y_train)

# Đánh giá
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

Sau khi đã huấn luyện xong mô hình, ta cùng xem 2 độ đo MSE và R-square

Mean Squared Error: 1.1482770391515056

R-squared: 0.0676452042381217

1. Mean Squared Error (MSE):

MSE là 1.1482770391515056, có thể coi là một giá trị không quá lớn và không quá nhỏ. Tuy nhiên, nó cần được so sánh với phạm vi của biến mục tiêu (rating) để hiểu rõ hơn về mức độ dự đoán của mô hình.

2. R-squared (R^2):

R-squared nằm trong khoảng từ 0 đến 1. Giá trị càng gần 1 thì mô hình càng tốt, vì nó cho thấy mô hình giải thích một phần lớn sự biến động của biến mục tiêu. Trong trường hợp này, R-squared là 0.0676452042381217, có giá trị khá thấp, chỉ là khoảng 0.07. Điều này cho thấy mô hình không thể giải thích một phần lớn sự biến động của biến mục tiêu (rating).

Nhìn chung, từ kết quả trên, mô hình hồi quy tuyến tính hiện tại có thể cần cải thiện để dự đoán rating của cuốn sách tốt hơn. Có thể cần thêm một số điều chỉnh trong mô hình hoặc xem xét sử dụng các mô hình phức tạp hơn như mạng neural để cải thiện hiệu suất dự đoán.

Bây giờ chúng ta sẽ chuyển qua mô hình mạng nơ ron:

2. Sử dụng mô hình mạng nơ ron để giải quyết vấn đề

- Ta sẽ dùng một mạng nơ ron được xây dựng từ thư viện tensorflow, đầu tiên ta cũng phải xử lý dữ liệu và thêm các thư viện cần thiết như ở bài toán trên

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

```
# Các bước ở trên ta sẽ làm tương tự như với mô hình hồi quy tuyến tính

data = pd.read_csv('prepared_data_book.csv')

data.drop(columns=['Unnamed: 0', 'product_id', 'title', 'cover_link'], inplace=True)

data.dropna(inplace=True)


label_encoder = LabelEncoder()
data['category_encoded'] = label_encoder.fit_transform(data['category'])
data['manufacturer_encoded'] = label_encoder.fit_transform(data['manufacturer'])
data['authors_encoded'] = label_encoder.fit_transform(data['authors'])


data.drop(columns=['category', 'manufacturer', 'authors'], inplace=True)
```

- Sau đó ta chia dữ liệu thành các tập và xây dựng mạng nơ ron để tiến hành huấn luyện.

```python
# Xây dựng mạng neural
model = keras.Sequential([
    layers.Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dropout(0.2),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(32, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(16, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(8, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(1)
])


model.compile(optimizer='adam', loss='mean_squared_error')

# Huấn luyện mô hình
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
```

Tổng quan về mạng nơ ron này:

Mạng neural trong đoạn mã trên là một mô hình hồi quy sử dụng framework Keras của TensorFlow. Mạng này có một số đặc điểm như sau:

1. Đặc điểm của mạng neural:

- Mạng có 8 lớp ẩn (hidden layers) với số lượng đơn vị (neurons) lần lượt là: 256, 256, 128, 64, 32, 16, 8 và 1.

- Các lớp ẩn được kích hoạt bằng hàm kích hoạt ReLU (Rectified Linear Activation), được xác định bằng tham số `activation='relu'`.

- Các lớp sau lớp ẩn (hidden layers) đều có thêm dropout layer để giảm thiểu hiện tượng overfitting. Điều này được thực hiện bằng cách sử dụng lớp `Dropout` với tham số `dropout=0.2`.

2. Quá trình huấn luyện:

- Mô hình được biên dịch (compile) bằng cách sử dụng tối ưu hóa `Adam` (`optimizer='adam'`) và hàm mất mát `mean_squared_error` (`loss='mean_squared_error'`), vì đây là bài toán dự đoán giá trị liên tục (hồi quy) và mất mát sử dụng sai số bình phương trung bình (Mean Squared Error).

- Sau khi mô hình được biên dịch, quá trình huấn luyện bắt đầu. Mô hình được huấn luyện trên tập dữ liệu huấn luyện (`X_train`, `y_train`) trong 100 epochs (vòng lặp) với kích thước batch là 32 (`batch_size=32`). `validation_split=0.2` cho biết 20% dữ liệu huấn luyện sẽ được sử dụng làm tập xác thực (validation set) để đánh giá hiệu suất mô hình sau mỗi epoch.

Kết quả sau quá trình huấn luyện sẽ được lưu vào biến `history`, cho phép truy cập các thông tin như hàm mất mát trên tập huấn luyện và tập xác thực sau mỗi epoch.

```
# Đánh giá mô hình trên tập test
mse = model.evaluate(X_test, y_test)
print("Mean Squared Error:", mse)
```

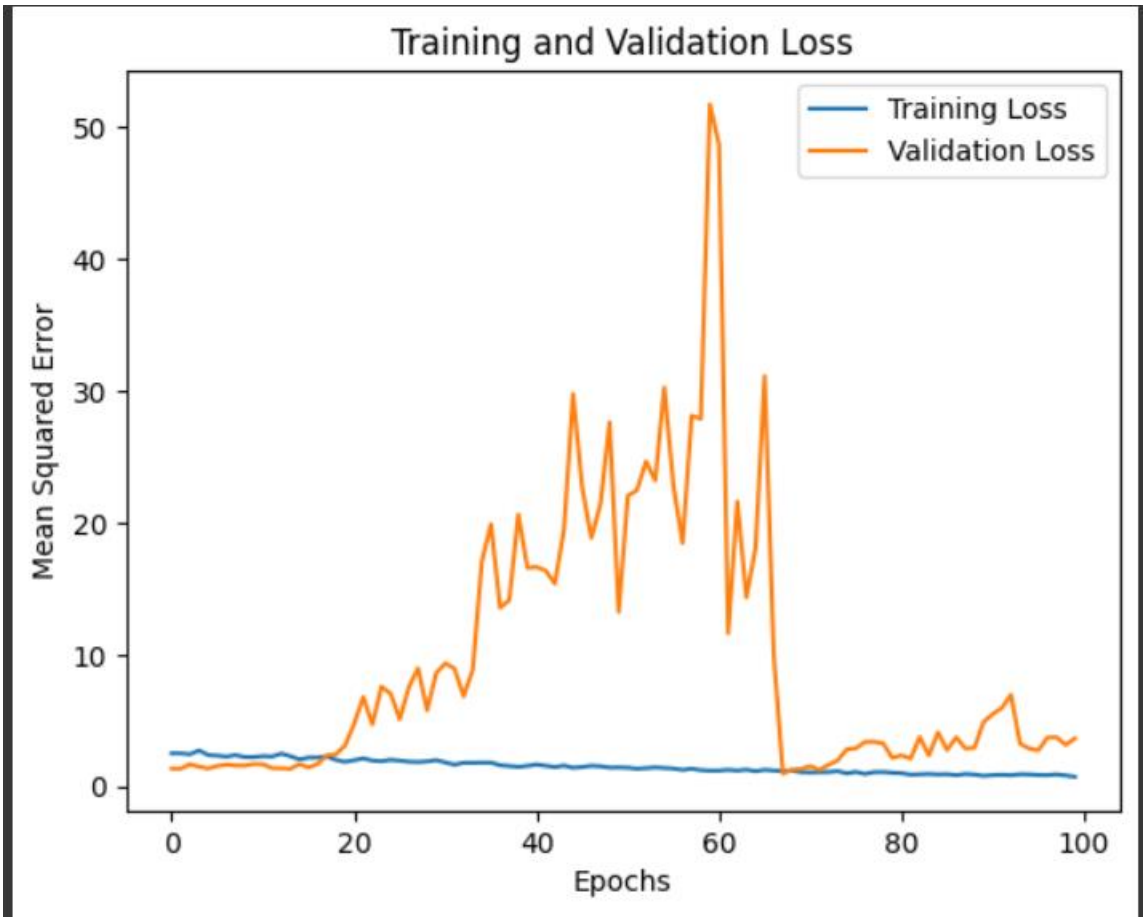
```
12/12 [==============================] - 0s 2ms/step - loss: 1.1870
Mean Squared Error: 1.1870346069335938
```

- Sau khi đã huấn luyện xong mô hình MSE của mô hình mạng neural trên là: 1.1870346069335938

* Nhận xét

Tiếp theo ta sẽ trực quan bằng một số biểu đồ.

- Biểu đồ đầu tiên sử dụng để trực quan hóa hàm mất mát (Mean Squared Error) của mô hình học máy qua từng epoch trong quá trình huấn luyện và đánh giá. Biểu đồ giúp hiểu rõ hơn về hiệu suất của mô hình và xác định các điểm cần cải thiện và điều chỉnh mô hình.



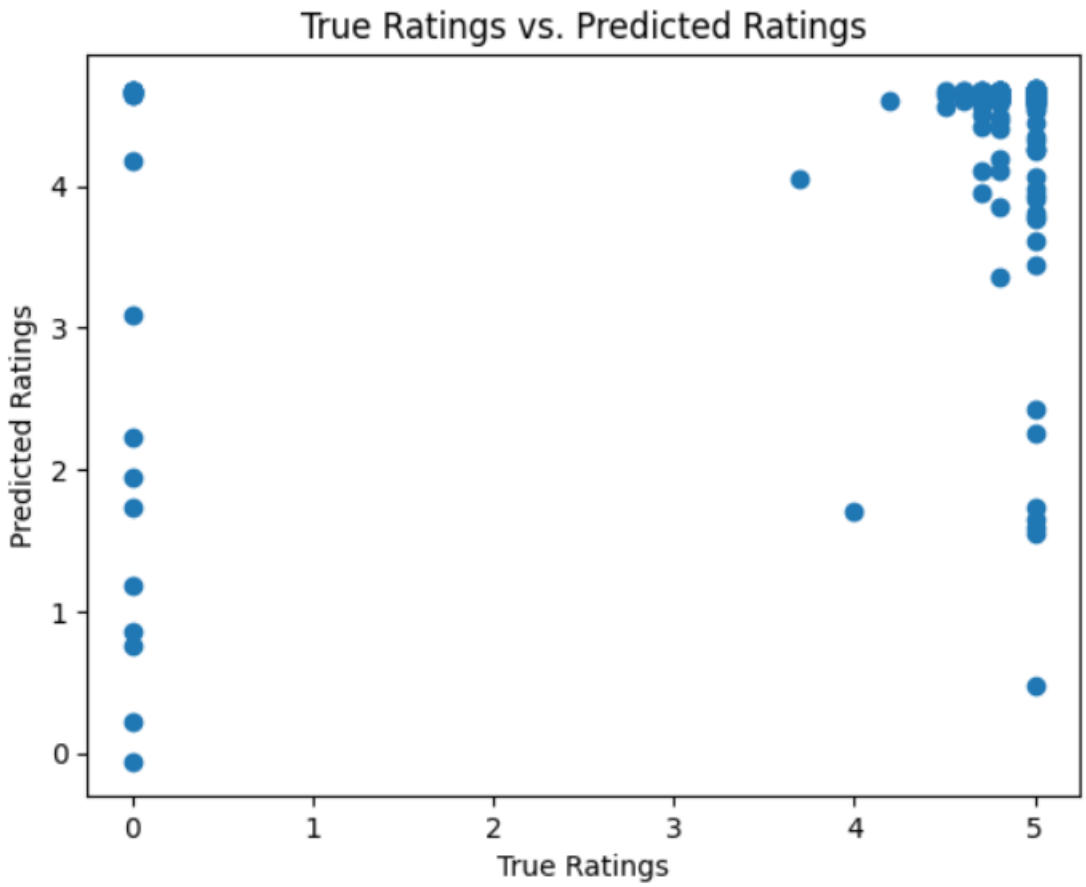
Khi train loss giảm dần từ từ trong quá trình huấn luyện mô hình, và val loss biến thiên liên tục nhưng sau đó giảm đều gần cuối, điều này có ý nghĩa như sau:

1. Train Loss giảm dần từ từ: Việc train loss giảm dần từ từ cho thấy mô hình đang học và tối ưu hóa các tham số để phù hợp tốt hơn với dữ liệu huấn luyện. Mô hình đang học cách biểu diễn dữ liệu và dự đoán kết quả gần với giá trị thực tế trên tập huấn luyện.

2. Val Loss biến thiên liên tục: Sự biến thiên liên tục của val loss có thể xuất phát từ sự không ổn định trong việc đánh giá hiệu suất trên tập xác thực (validation set) khi mô hình chưa hoàn toàn hội tụ. Có thể xuất hiện những biến động ngẫu nhiên do dữ liệu xác thực có thể khác nhau giữa các epoch.

3. Val Loss giảm đều gần cuối: Khi val loss giảm đều gần cuối quá trình huấn luyện, điều này cho thấy mô hình đã hội tụ và dự đoán trên tập xác thực ngày càng gần với giá trị thực tế. Nếu val loss tiếp tục giảm sau khi đã ổn định, có thể mô hình vẫn còn tiềm năng cải thiện và có thể tiếp tục huấn luyện một số epoch nữa.

Tổng quan, khi train loss giảm dần từ từ và val loss biến thiên liên tục nhưng giảm đều gần cuối, có thể cho thấy mô hình đang tiến triển tốt và không bị overfitting.



Biểu đồ phân tán (scatter plot) và giúp hiểu rõ hơn về hiệu suất dự đoán của mô hình trên dữ liệu mới. Như ta thấy mô hình này thực sự các điểm phân tán xa khỏi đường chéo và tập trung ở các vị trí không chính xác, có thể mô hình gặp khó khăn trong việc dự đoán chính xác giá trị rating của sách.

- Bây giờ ta sẽ nhận xét so sánh giữa 2 mô hình trong việc xử lý vấn đề này:

1. Hiệu suất dự đoán của mô hình hồi quy tuyến tính không tốt hơn so với mô hình mạng neural: MSE càng cao thể hiện mô hình không thực hiện dự đoán tốt trên dữ liệu kiểm tra, và sự sai khác giữa giá trị rating thực tế và giá trị rating dự đoán của mô hình lớn hơn so với mô hình mạng neural. Điều này có thể xuất phát từ sự đơn giản của mô hình hồi quy tuyến tính, nó không thể mô hình hóa mối quan hệ phức tạp giữa các biến đầu vào và biến mục tiêu (rating).

2. Mô hình hồi quy tuyến tính không khái quát hóa tốt trên dữ liệu mới: MSE cao cũng có thể xuất phát từ hiện tượng overfitting, khi mô hình học quá chính xác từ dữ liệu huấn luyện và không tổng quát hóa tốt trên dữ liệu kiểm tra. Trong khi đó, mô hình mạng neural có khả năng học mối quan hệ phức tạp hơn và có khả năng tổng quát hóa tốt hơn trên dữ liệu mới.

Vấn đề 2: Số sách bán ra có thể được dự đoán từ những thông tin đã được cung cấp không ?

1. Sử dụng mô hình phi noron để giải quyết vấn đề

- Đầu tiên, ta vẫn sẽ tiến hành các bước xử lý dữ liệu để có thể đưa vào mô hình, cụ thể như sau:

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load dataset
data = pd.read_csv('prepared_data_book.csv')

# Bỏ các trường dữ liệu không cần thiết
data.drop(columns=['Unnamed: 0', 'product_id', 'title', 'cover_link'], inplace=True)

# Bỏ các giá trị bị thiếu
data.dropna(inplace=True)

# Encode các biến categorical thành numeric
label_encoder = LabelEncoder()
data['category_encoded'] = label_encoder.fit_transform(data['category'])
data['manufacturer_encoded'] = label_encoder.fit_transform(data['manufacturer'])
data['authors_encoded'] = label_encoder.fit_transform(data['authors'])

# Vì đã encode nên ta sẽ bỏ đi các cột dữ liệu cũ
data.drop(columns=['category', 'manufacturer', 'authors'], inplace=True)

# Vì quantity khá lớn nên thay vì tính toán trực tiếp ta sẽ tính log của quantity
data['quantity'] = np.log1p(data['quantity'])
```

- Tiếp sau đó ta sẽ đưa bộ dữ liệu đã được xử lý vào mô hình hồi quy tuyến tính (ở đây ta sẽ dùng mô hình của thư viện sklearn). Ở đây, khác với vấn đề 1, vì quantity là một con số khá lớn nên thay vì tính toán trực tiếp ta sẽ tính log thay vào đấy

```
# Split data thành dữ liệu x và target y
X = data.drop(columns=['quantity'])
y = data['quantity']

# Chia thành tập train và tập test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale lại các đặc trưng
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Khởi tạo và huấn luyện mô hình hồi quy tuyến tính
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán dựa trên testing data
y_pred = model.predict(X_test)

# Tính toán MSE
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Sau khi đã huấn luyện xong mô hình, ta cùng xem độ đo MSE

Mean Squared Error: 1.9990549098231671

1. Mean Squared Error (MSE):

MSE = 1.9990549098231671 cho thấy mô hình hồi quy tuyến tính chưa dự đoán chính xác số lượng sách bán ra trên tập dữ liệu kiểm tra. Để cải thiện hiệu suất, có thể thử các phương pháp cải tiến mô hình hoặc sử dụng mô hình phức tạp hơn .Nhìn chung, từ kết quả trên, mô hình hồi quy tuyến tính hiện tại có thể miễn cưỡng dự đoán số lượng sách bán ra

Bây giờ chúng ta sẽ chuyển qua mô hình mạng nơ ron:

2. Sử dụng mô hình mạng nơ ron để giải quyết vấn đề

- Ta sẽ dùng một mạng nơ ron được xây dựng từ thư viện tensorflow, đầu tiên ta cũng phải xử lý dữ liệu và thêm các thư viện cần thiết như ở bài toán trên

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Các bước ở trên ta sẽ làm tương tự như với mô hình hồi quy tuyến tính
data = pd.read_csv('prepared_data_book.csv')

data.drop(columns=['Unnamed: 0', 'product_id', 'title', 'cover_link'], inplace=True)

data.dropna(inplace=True)

label_encoder = LabelEncoder()
data['category_encoded'] = label_encoder.fit_transform(data['category'])
data['manufacturer_encoded'] = label_encoder.fit_transform(data['manufacturer'])
data['authors_encoded'] = label_encoder.fit_transform(data['authors'])

data.drop(columns=['category', 'manufacturer', 'authors'], inplace=True)
```

- Sau đó ta chia dữ liệu thành các tập và xây dựng mạng nơ ron để tiến hành huấn luyện.

```
# Chia thành tập train và test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Xây dựng mạng neural
model = keras.Sequential([
    layers.Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dropout(0.2),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(32, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(16, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(8, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(1)
])

# compile model
model.compile(optimizer='adam', loss='mean_squared_error')

# huấn luyện mô hình trên tập train
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)

# Đánh giá MSE trên tập test
mse = model.evaluate(X_test, y_test)
print("Mean Squared Error:", mse)

# áp dụng trên tập test
y_pred = model.predict(X_test)
```

Tổng quan về mạng nơ ron này:

Mạng neural trong mã trên đã được xây dựng và huấn luyện để dự đoán số lượng sách bán ra (quantity) dựa vào các đặc trưng (features) của sách. Dưới đây là phân tích chi tiết về mô hình và quá trình huấn luyện:

1. Chia dữ liệu thành đặc trưng X và mục tiêu y:

- Dữ liệu được chia thành hai phần X (đặc trưng) và y (mục tiêu) bằng cách loại bỏ cột 'quantity' trong data và lưu giữ giá trị này vào y. Đây là bước chuẩn bị dữ liệu để xây dựng mô hình.

2. Logarithm hóa của số lượng sách (y):

- Dữ liệu số lượng sách đã được logarithm hóa (log1p) trước khi sử dụng làm mục tiêu (y_train, y_test). Logarithm hóa giúp giảm biến thiên lớn của dữ liệu và làm cho phân bố của y đồng đều hơn, giúp mô hình dự đoán tốt hơn trên các giá trị số lượng sách khác nhau.

3. Chia dữ liệu thành tập huấn luyện và tập kiểm tra:

- Dữ liệu được chia thành tập huấn luyện (X_train, y_train) và tập kiểm tra (X_test, y_test) với tỷ lệ 80% tập huấn luyện và 20% tập kiểm tra. Việc này giúp đánh giá hiệu suất của mô hình trên dữ liệu không được dùng để huấn luyện.

4. Tiêu chuẩn hóa dữ liệu:

- Dữ liệu trong tập huấn luyện và tập kiểm tra đã được tiêu chuẩn hóa sử dụng StandardScaler để đảm bảo các đặc trưng có cùng đơn vị và có phân phối gần với chuẩn (mean = 0, standard deviation = 1). Việc tiêu chuẩn hóa giúp tăng hiệu suất và tốc độ hội tụ của mô hình.

5. Xây dựng mạng neural:

- Mạng neural được xây dựng bằng thư viện Keras của TensorFlow.

- Mạng bao gồm nhiều layers gồm: 6 layers Dense với số lượng node lần lượt là 256, 128, 64, 32, 16, 8 và hàm kích hoạt ReLU (Rectified Linear Unit) cho các hidden layer.

- Giữa các layers Dense, có các layers Dropout với tỷ lệ 0.2 (20%) để giảm nguy cơ overfitting và tăng tính tổng quát hóa của mô hình.

- Layer cuối cùng có 1 node để dự đoán số lượng sách bán ra (quantity).

6. Compile mô hình:

- Mô hình được biên dịch với hàm loss là 'mean_squared_error' (MSE) và tối ưu hóa 'adam' (phương pháp tối ưu hóa dựa trên gradient).

7. Huấn luyện mô hình:

- Mô hình được huấn luyện trên tập huấn luyện với 50 epoch (vòng lặp huấn luyện) và batch size là 32 (số lượng mẫu được sử dụng để tính gradient một lần).

- Trong quá trình huấn luyện, mô hình sẽ cố gắng tối thiểu hóa MSE trên tập huấn luyện và kiểm tra.

8. Đánh giá mô hình:

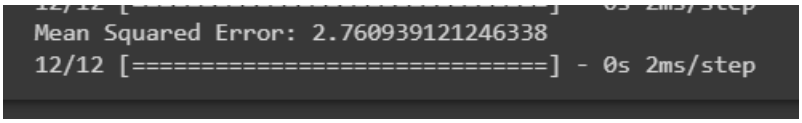
- Mô hình được đánh giá trên tập kiểm tra bằng cách tính MSE trên dữ liệu kiểm tra đã được tiêu chuẩn hóa.

- Giá trị MSE này sẽ đánh giá hiệu suất của mô hình trong việc dự đoán số lượng sách bán ra trên tập kiểm tra.

9. Dự đoán trên tập kiểm tra:

- Mô hình được sử dụng để dự đoán số lượng sách bán ra trên tập kiểm tra (y_pred).

Tóm lại, mô hình mạng neural đã được xây dựng và huấn luyện để dự đoán số lượng sách bán ra dựa vào các đặc trưng của sách. Quá trình huấn luyện mô hình được thực hiện qua nhiều vòng lặp (epochs), và đánh giá hiệu suất của mô hình dựa trên giá trị MSE trên tập kiểm tra.



- Sau khi đã huấn luyện xong mô hình MSE của mô hình mạng neural trên là: 2.7

* Nhận xét

Tiếp theo ta sẽ trực quan bằng một số biểu đồ.

- Biểu đồ đầu tiên là biểu đồ biểu thị giá trị của hàm mất mát (loss) trên tập huấn luyện và tập kiểm tra (validation) qua từng epoch (vòng lặp huấn luyện). Đây là biểu đồ quan trọng trong quá trình huấn luyện mô hình mạng neural, giúp theo dõi sự học tập và hiệu suất của mô hình theo thời gian.

- Ta thấy đường màu xanh (train loss) giảm mạnh qua từng epoch trong khi đường màu cam (val loss) giảm nhẹ hoặc biến động liên tục, điều này có thể có ý nghĩa như sau:

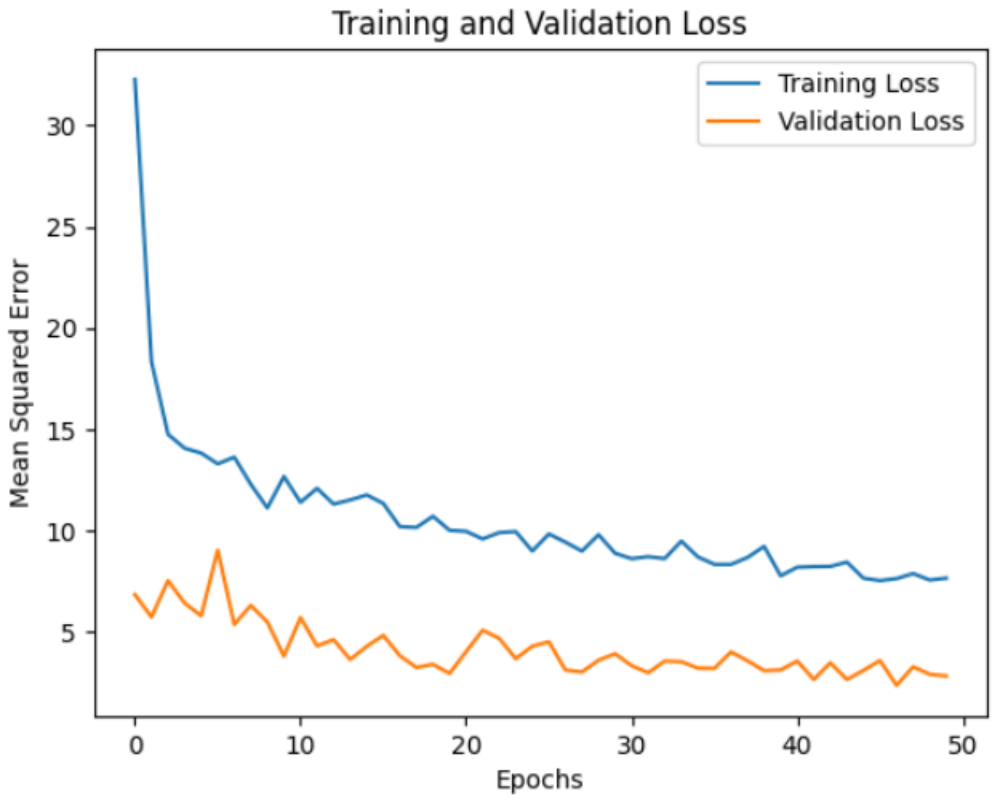
1. Overfitting: Đường màu xanh (train loss) giảm mạnh cho thấy mô hình đang học tốt từ dữ liệu huấn luyện và đang giảm thiểu sai số trên tập huấn luyện.

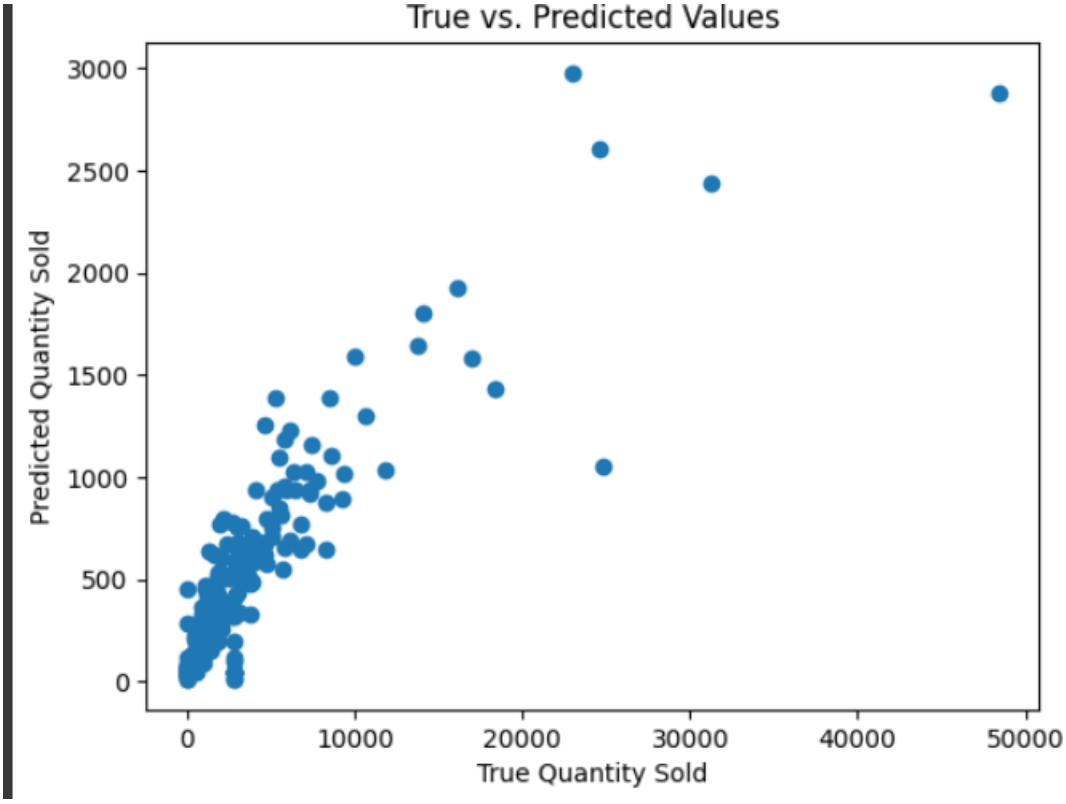
2. Khả năng tổng quát hóa: Nếu đường màu xanh (train loss) giảm mạnh và đường màu cam (val loss) giảm chậm hơn hoặc không giảm đều, điều này cho thấy mô hình có thể không tổng quát hóa tốt trên dữ liệu kiểm tra (tập kiểm tra).

3. Kiểm tra lại số lượng epochs: Trong trường hợp đường màu cam (val loss) không giảm đều, có thể xem xét giảm số lượng epochs hoặc điều chỉnh mô hình để tránh overfitting. Có thể mô hình đã học đủ thông tin từ dữ liệu huấn luyện sau một số epoch, việc tiếp tục huấn luyện có thể làm gia tăng overfitting.

4. Đánh giá hiệu suất: Hiệu suất của mô hình trên dữ liệu kiểm tra (val loss) quan trọng hơn so với hiệu suất trên dữ liệu huấn luyện (train loss). Nếu val loss không giảm đều và tiếp tục tăng sau một số epoch, điều này cần được xem xét và tối ưu hóa mô hình để đạt được kết quả tốt hơn trên dữ liệu mới.

Để cải thiện hiệu suất của mô hình, có thể xem xét giảm số lượng layers, tăng số lượng node trong các layers, điều chỉnh tỷ lệ dropout hoặc tăng số lượng epochs. Điều này giúp tăng khả năng tổng quát hóa và giảm hiện tượng overfitting.





Đồ thị này là đồ thị phân tán (scatter plot) giữa giá trị thực tế (True Quantity Sold) và giá trị dự đoán (Predicted Quantity Sold) của số lượng sách đã bán ra. Trong đồ thị, mỗi điểm biểu diễn một giá trị thực tế và giá trị dự đoán, trong đó trục x (X-axis) biểu thị giá trị thực tế và trục y (Y-axis) biểu thị giá trị dự đoán. Điều này cho phép ta quan sát mối quan hệ và sự tương đồng giữa giá trị thực tế và giá trị dự đoán trong việc dự đoán số lượng sách đã bán ra.

Các điểm dữ liệu nằm gần đường chéo, điều này có thể cho thấy mô hình dự đoán khá chính xác số lượng sách đã bán ra. Các sai số dự đoán có thể nhỏ và mô hình có thể cần được điều chỉnh hoặc cải thiện để đạt được kết quả tốt hơn trong việc dự đoán số lượng sách đã bán ra.

Kết luận khi sử dụng mô hình mạng nơ ron và phi nơ ron:

Mô hình mạng nơ-ron và phi nơ-ron là hai loại mô hình máy học được sử dụng rộng rãi trong phân tích dữ liệu và dự báo. Dưới đây, ta sẽ so sánh hai loại mô hình này dựa vào những gì thu thập được từ hai vấn đề trên

1. Mô hình mạng nơ-ron:

Ưu điểm:

- Khả năng học và biểu diễn các mối quan hệ phức tạp giữa các biến đầu vào và đầu ra.
- Có khả năng xử lý dữ liệu phi cấu trúc và không cấu trúc
- Đạt được hiệu suất tốt trên các bài toán phân loại và dự báo phức tạp
- Có thể được huấn luyện thông qua nhiều lớp và chức năng kích hoạt khác nhau.

Nhược điểm:

- Dễ bị overfitting (quá khớp) nếu không có các biện pháp kiểm soát như dropout và regularization.
- Đòi hỏi một lượng lớn dữ liệu huấn luyện để đạt được hiệu suất tốt.
- Cần nhiều tài nguyên tính toán, đặc biệt khi mô hình có nhiều lớp và tham số.

2. Mô hình phi nơ-ron:

Ưu điểm:

- Đơn giản và dễ triển khai, đặc biệt đối với các bài toán dự báo nhỏ và đơn giản.
- Thường đòi hỏi ít dữ liệu hơn để huấn luyện so với mạng nơ-ron.
- Dễ dàng hiểu và giải thích kết quả dự báo.

Nhược điểm:

- Giới hạn trong việc xử lý dữ liệu phức tạp và mối quan hệ phi tuyến tính giữa các biến.
- Hiệu suất thường không cao như mạng nơ-ron đối với các bài toán phức tạp.
- Không phù hợp với các bài toán có cấu trúc phức tạp hơn.

Vấn đề 3: Sử dụng PHOBERT để đánh giá nhận xét của người tiêu dùng khi đặt hàng

PhoBERT là mô hình ngôn ngữ tự nhiên dựa trên kiến trúc mạng nơ-ron biểu diễn từ (BERT) được điều chỉnh và huấn luyện lại trên dữ liệu tiếng Việt.

BERT là một trong những mô hình ngôn ngữ tự nhiên tiên tiến nhất, nổi tiếng với khả năng hiểu được ngữ nghĩa của ngôn ngữ tự nhiên thông qua việc biểu diễn từ (word embeddings) và đại diện ngữ cảnh (contextual embeddings). BERT đã tạo ra cách tiếp cận mới cho nhiều nhiệm vụ trong xử lý ngôn ngữ tự nhiên, như phân loại văn bản, dự đoán từ tiếp theo, và trích xuất thông tin.

PhoBERT, được phát triển bởi VinAI Research, là phiên bản được tinh chỉnh từ BERT để phù hợp với ngôn ngữ tiếng Việt. Nó đã được huấn luyện trên một lượng lớn dữ liệu tiếng Việt và cho phép xử lý ngôn ngữ tự nhiên đa tác vụ trên văn bản tiếng Việt. Điều này làm cho PhoBERT trở thành một trong những công cụ quan trọng và hiệu quả trong nghiên cứu và ứng dụng NLP cho tiếng Việt.

PhoBERT cung cấp các biểu diễn từ và đoạn văn bản có độ chính xác cao, giúp cải thiện hiệu suất trong nhiều tác vụ NLP như phân loại văn bản, dự đoán ngữ cảnh, và tóm tắt văn bản bằng tiếng Việt. Nó đã trở thành một công cụ hữu ích và quan trọng trong cộng đồng NLP và đóng góp vào sự phát triển của xử lý ngôn ngữ tự nhiên cho ngôn ngữ tiếng Việt.

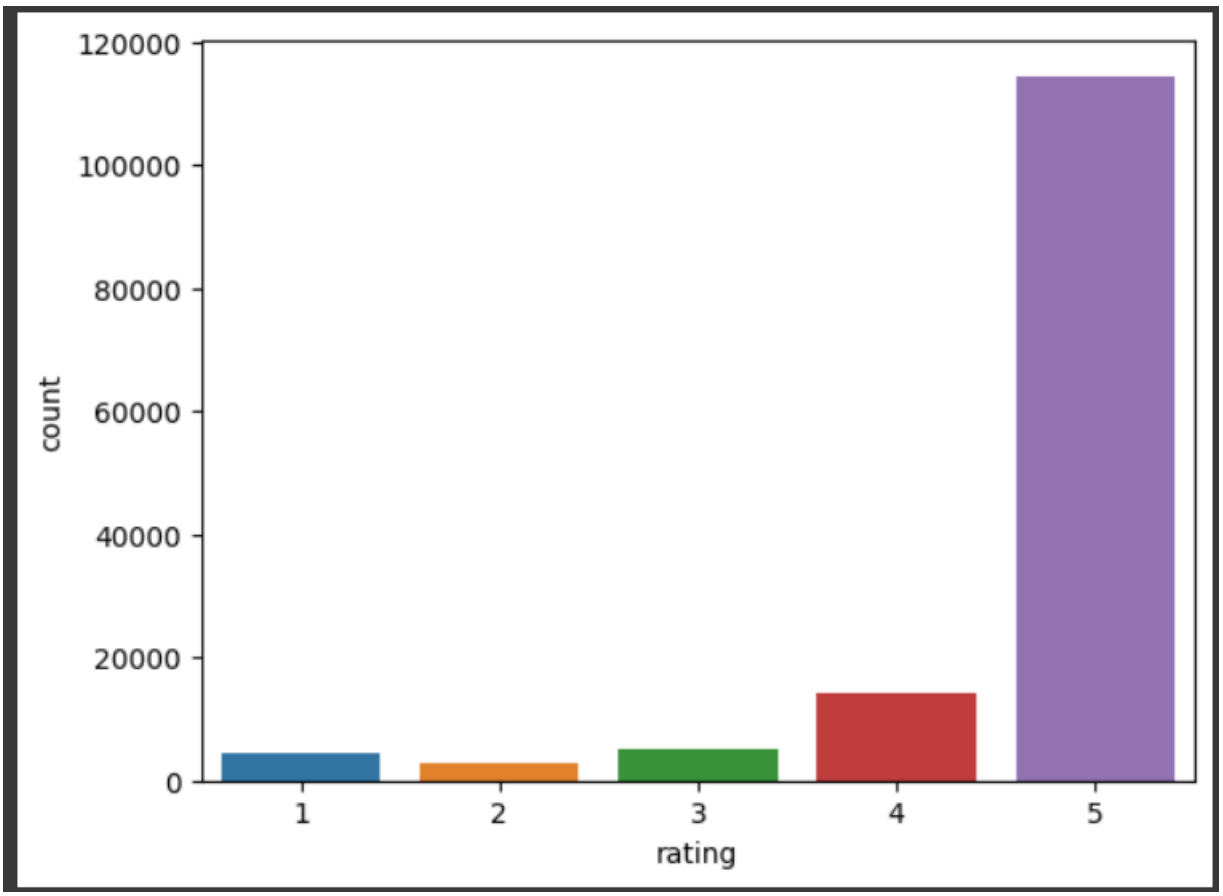
Trong đồ án lần này, chúng em có một dataset bao gồm các comments (bình luận) trên trang bán sách và xếp hạng của sách dựa vào các comments này. Sau đó sử dụng mô hình PhoBERT để xây dựng một mô hình dự đoán xếp hạng sách từ các bình luận (comments) của người dùng.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

import tensorflow as tf
import tensorflow_addons as tfa
```

```
df = pd.read_csv('comments.csv',
                 usecols = ['rating', 'content'])
print(df['content'])
```

- Đầu tiên ta vẫn cài đặt các thư viện hỗ trợ xây dựng mô hình mạng nơ ron như bình thường, đồng thời ta chỉ sử dụng hai cột trong tập dữ liệu đó là ‘rating’ và ‘content’.

- Sau đó ta nhìn nhận một cách tổng thể các rating của người tiêu dùng qua data này:



- Chủ yếu các bình luận này khá tích cực, ở mức 5*. Tiếp theo ta xử lý các tập dữ liệu bị thiếu và tiến hành chia tập dữ liêu. Phần này khá tương đồng với 2 vấn đề đã đề cập ở phần trước.

- Tiếp theo ta sẽ đến phần cấu hình TPU:

```
[ ]  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)
```

Đoạn code trên thực hiện các bước cấu hình để tận dụng tích hợp TPU (Tensor Processing Unit) hoặc nhiều GPU (Graphics Processing Unit) trong TensorFlow, nhằm tăng tốc độ huấn luyện mô hình trên nhiều thiết bị xử lý song song.

1. Tạo và kết nối đến TPU (nếu có): Đầu tiên, đoạn mã kiểm tra xem có TPU nào được phát hiện trong môi trường không. Nếu có, nó sẽ kết nối đến TPU bằng cách sử dụng tf.distribute.cluster_resolver. TPUClusterResolver()` để tìm TPU và `tf.config.experimental_connect_to_cluster(tpu)` để kết nối đến cụm TPU. Sau đó, đoạn mã sử dụng `tf.tpu.experimental.initialize_tpu_system(tpu)` để khởi tạo hệ thống TPU.

2. Xác định chiến lược phân phối (distribution strategy): Tiếp theo, đoạn mã xác định chiến lược phân phối dữ liệu trên các thiết bị xử lý. Có ba trường hợp có thể xảy ra:

a. Nếu TPU được phát hiện (`if tpu:`), chiến lược sẽ được định nghĩa là `tf.distribute.experimental.TPUStrategy(tpu)`. Điều này cho phép mô hình được huấn luyện song song trên các đơn vị TPU.

b. Nếu không có TPU nhưng có nhiều GPU (`elif len(gpus) > 1:`), chiến lược sẽ là `tf.distribute.MirroredStrategy([gpu.name for gpu in gpus])`. Điều này cho phép mô hình được huấn luyện song song trên các GPU.

c. Nếu chỉ có một GPU (`elif len(gpus) == 1:`), chiến lược sẽ là `tf.distribute.get_strategy()` để sử dụng một GPU duy nhất.

d. Nếu không có GPU và không có TPU (`else:`), chiến lược sẽ là `tf.distribute.get_strategy()` để sử dụng CPU.

3. In thông tin về chiến lược phân phối: Cuối cùng, đoạn mã in ra thông tin về chiến lược phân phối đã được chọn và số lượng accelerator (đơn vị xử lý, bao gồm cả TPU và GPU) sẽ được sử dụng trong quá trình huấn luyện.

- Tiếp theo ta sẽ cài đặt một số thư viện và package cần thiết:

```
!pip3 install transformers

from transformers import TFAutoModel, AutoTokenizer
```

1. Cài đặt thư viện `transformers`, sử dụng để cài đặt thư viện `transformers`. Thư viện này cung cấp các công cụ và kiến trúc mô hình liên quan đến xử lý ngôn ngữ tự nhiên (NLP) như BERT, GPT, RoBERTa, và nhiều mô hình NLP nổi tiếng khác.

2. Import lớp `TFAutoModel` và `AutoTokenizer` từ thư viện `transformers`: Sau khi đã cài đặt thành công, mã `from transformers import TFAutoModel, AutoTokenizer` được sử dụng để import lớp `TFAutoModel` và `AutoTokenizer` từ thư viện `transformers`. Lớp `TFAutoModel` được sử dụng để tạo các mô hình ngôn ngữ dựa trên kiến trúc BERT và các biến thể khác của nó trong TensorFlow. Lớp `AutoTokenizer` được sử dụng để chuyển đổi văn bản thành đầu vào số cho các mô hình NLP, giúp chuẩn hóa và biểu diễn các văn bản dưới dạng số để có thể đưa vào mô hình NLP.

- Tiếp theo ta sẽ tạo Tokenizer:

```
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()]
```

```
Downloading (…)lve/main/config.json: 100% 557/557 [00:00<00:00, 18.0kB/s]
Downloading (…)solve/main/vocab.txt: 100% 895k/895k [00:00<00:00, 11.4MB/s]
Downloading (…)solve/main/bpe.codes: 100% 1.14M/1.14M [00:00<00:00, 32.4MB/s]
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
```

Đoạn codetrên thực hiện các bước sau:

1. Xác định độ dài tối đa của câu (`MAX_LEN`): Đoạn code đầu tiên xác định độ dài tối đa của mỗi câu trong dữ liệu là 200 ký tự. Tức là, mỗi câu sẽ được cắt hoặc bổ sung chèn các ký tự đệm (padding) để có độ dài là 200 ký tự.

2. Xác định tên của mô hình PhoBERT và tạo tokenizer: Đoạn code thứ hai xác định tên của mô hình PhoBERT là `'vinai/phobert-base'`. Sau đó, nó sử dụng `AutoTokenizer.from_pretrained(model_name)` để tạo một tokenizer từ mô hình đã được huấn luyện trước của PhoBERT.

3. Token hóa dữ liệu: Đoạn code cuối cùng thực hiện việc token hóa dữ liệu. Với mỗi câu trong dữ liệu (được lưu trữ trong biến `X`), nó sử dụng tokenizer đã được tạo trước để chuyển đổi câu thành một dạng số (mã hóa bằng các ID của từng từ trong từ điển của mô hình). Các bước thực hiện token hóa là như sau:

- `tokenizer(item, max_length=MAX_LEN, padding='max_length', truncation=True, return_tensors='np')`: Sử dụng tokenizer để chuyển đổi câu `item` thành dạng số. Các tham số được cung cấp bao gồm `max_length` (độ dài tối đa của câu), `padding='max_length'` (thêm các ký tự đệm để câu có độ dài bằng `MAX_LEN`), `truncation=True` (cắt các câu quá dài về độ dài `MAX_LEN` nếu cần), và `return_tensors='np'` (trả về dạng numpy array).

- `'input_ids'`: Mã hóa các từ trong câu `item` thành các ID tương ứng trong từ điển của mô hình. Kết quả trả về là một numpy array chứa các ID của từng từ trong câu.

- `reshape(MAX_LEN)`: Sau khi token hóa, kết quả trả về có thể có độ dài thay đổi (nhỏ hơn `MAX_LEN` nếu câu quá ngắn). Bước này đảm bảo rằng độ dài của kết quả được đặt là `MAX_LEN` bằng cách thêm các giá trị đệm (padding) nếu cần.

Kết quả cuối cùng của đoạn mã là biến `inputs`, một danh sách các numpy array chứa các ID của từng từ trong từng câu trong dữ liệu, đã được mã hóa và chuẩn hóa độ dài thành `MAX_LEN`.

- Tiếp theo, ta sẽ xây dựng mô hình:

```
# Build model
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)
```

Đoạn code trên xây dựng mô hình dự đoán xếp hạng sách dựa trên các nhận xét từ người dùng sử dụng PhoBERT làm bộ mã hóa.

1. Tạo mô hình bộ mã hóa (encoder) từ mô hình PhoBERT: Đoạn mã `encoder = TFAutoModel.from_pretrained(model_name)` tạo mô hình bộ mã hóa bằng cách sử dụng `TFAutoModel.from_pretrained(model_name)` để tải mô hình PhoBERT được huấn luyện sẵn với tên là `model_name`. Mô hình bộ mã hóa sẽ được sử dụng để mã hóa các câu đầu vào thành các biểu diễn số học.

2. Định nghĩa lớp đầu vào và bộ mã hóa: Đoạn mã `input_ids = Input(shape=(MAX_LEN,), dtype=tf.int32)` định nghĩa lớp đầu vào cho mô hình. Các câu đầu vào sẽ có kích thước là `MAX_LEN`, được biểu diễn dưới dạng dãy các số nguyên (ID từ tokenizer). Tiếp theo, đoạn mã `embedding = encoder(input_ids)[1]` sử dụng mô hình bộ mã hóa để mã hóa các câu đầu vào thành biểu diễn số học. Mô hình bộ mã hóa trả về một tuple, trong đó phần tử thứ hai là biểu diễn của câu dưới dạng một tensor 2D. Trong đoạn mã này, chỉ lấy phần tử thứ hai từ tuple, tức là biểu diễn của câu.

3. Xây dựng mạng nơ-ron: Đoạn mã tiếp theo xây dựng mạng nơ-ron dự đoán xếp hạng sách từ các biểu diễn của câu. Trước tiên, đoạn mã `x = Dense(128, activation='relu', kernel_regularizer=regularizers.L2(0.1))(embedding)` tạo một lớp Dense với 128 đơn vị ẩn và hàm kích hoạt ReLU. Đây là lớp ẩn trong mạng nơ-ron.

4. Thêm lớp Dropout: Để tránh overfitting, đoạn mã `x = Dropout(0.2)(x)` thêm một lớp Dropout với tỷ lệ dropout là 0.2 sau lớp ẩn.

5. Xây dựng lớp đầu ra: Đoạn mã `x = Dense(y.shape[1], activation='softmax', kernel_initializer=tf.initializers.GlorotUniform(seed=1905), name='output_layer')(x)` tạo lớp đầu ra với số lượng đơn vị ẩn bằng số lượng xếp hạng (có được từ `y.shape[1]`) và hàm kích hoạt softmax. Đây là lớp cuối cùng của mô hình và sẽ đưa ra dự đoán xếp hạng cho sách.

6. Hoàn thành mô hình: Cuối cùng, đoạn mã `model = Model(inputs=[input_ids], outputs=x)` hoàn thành việc xây dựng mô hình dự đoán xếp hạng sách từ các câu đầu vào. Mô hình này nhận các câu đầu vào dưới dạng các ID từ tokenizer và trả về các dự đoán xếp hạng sách.

```
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)
```

```
model.compile(
    optimizer=tfa.optimizers.AdamW(weight_decay=wd,
                                   learning_rate=lr),
    loss=tf.keras.losses.CategoricalCrossentropy(),
    metrics=['accuracy'])
```

```
# Train model
history = model.fit(X_train,
                    y_train,
                    epochs = 20,
                    verbose = 1,
                    validation_split = 0.1,
                    batch_size = 256)
```

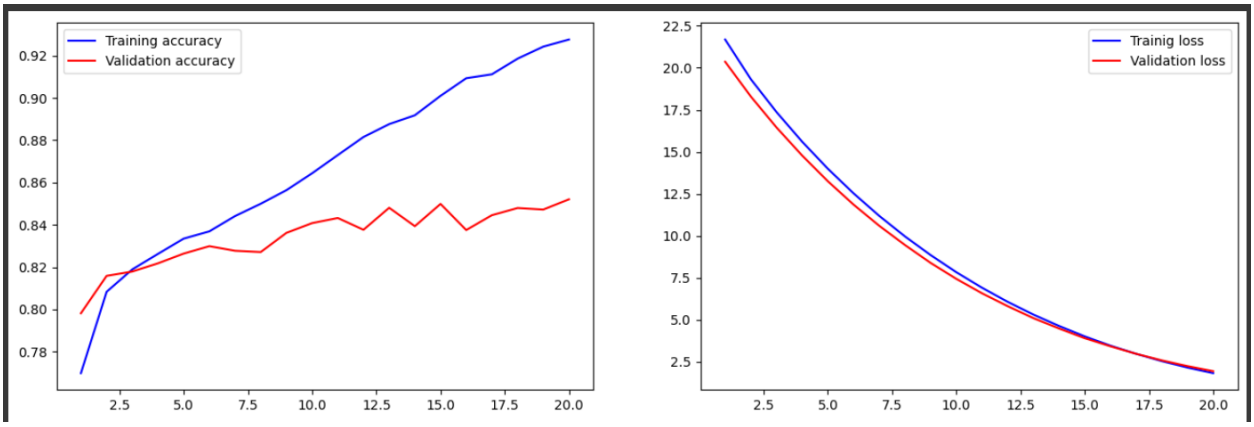
Sau đó ta xác định lịch learning rate và weight decay với optimizer là Adam sử dụng thư viện TensorFlow Addons. Tiếp theo, biên dịch mô hình với các siêu tham số tùy chỉnh như sau:

1. Learning rate schedule: Đoạn code `step = tf.Variable(0, trainable=False)` tạo một biến `step` đại diện cho số bước huấn luyện đã thực hiện. Biến này được đặt là không được huấn luyện (`trainable=False`). Sau đó, đoạn code `schedule = tf.optimizers.schedules.PiecewiseConstantDecay([10000, 15000], [1e-3, 1e-4, 1e-5])` xác định một lịch trình học tập tỷ lệ bằng `PiecewiseConstantDecay`, có nghĩa là tỷ lệ học tập sẽ được giữ nguyên cho 10000 bước, sau đó giảm xuống 1e-4 cho 5000 bước tiếp theo, và cuối cùng giảm xuống 1e-5 cho các bước huấn luyện còn lại.

2. Tính Learning rate và weight decay. Đoạn code `lr = 1e-2 * schedule(step)` tính tỷ lệ học tập cho bước hiện tại dựa vào lịch trình học tập `schedule` và `step`. Tương tự, đoạn code `wd = lambda: 1e-5 * schedule(step)` tính giá trị giảm trọng lượng (weight decay) cho bước hiện tại dựa vào lịch trình học tập `schedule` và `step`. Lưu ý rằng `wd` được định nghĩa dưới dạng hàm lambda để đảm bảo giá trị giảm trọng lượng được tính lại sau mỗi bước huấn luyện.

3. Biên dịch mô hình: Đoạn code cuối cùng biên dịch mô hình với các siêu tham số tùy chỉnh đã xác định ở trên. Cụ thể, mô hình được biên dịch với bộ tối ưu hóa AdamW (`tfa.optimizers.AdamW`) với các tham số weight_decay được tính bằng hàm lambda `wd`, learning_rate được tính bằng `lr`, và hàm mất mát là CategoricalCrossentropy. Ngoài ra, đánh giá mô hình sử dụng chỉ số accuracy (độ chính xác) trong quá trình huấn luyện.

Cuối cùng ta trực quan lần lượt accuray và loss của mô hình:



Như ta thấy, mô hình sau khi huấn luyện qua nhiều epoch đã có thể tối thiểu hóa hàm loss và tăng accuracy trên cả tập train và val. Accucary cuối cùng đạt được xấp xỉ 0.86. Chứng tỏ mô hình đã hoạt động tốt trên tập dữ liệu.