Interview Project Report

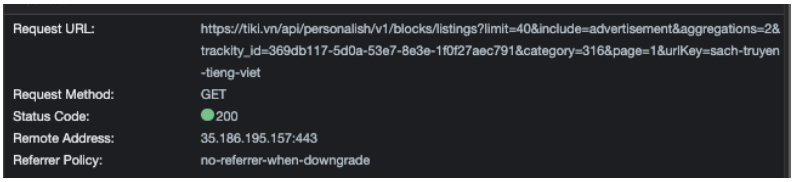
Interviewee: Đào Văn Hiếu

Contents

1 Crawling Tiki data

1.1 Finding the book categories to crawl

By inspecing this url I can see there are 24 main book categories for "Sách Tiếng Việt".




```
{'name': 'Sách kinh tế', 'id': 846},
{'name': 'Sách thiếu nhi ', 'id': 393},
{'name': 'Sách kỹ năng sống', 'id': 870},
{'name': 'Sách Bà mẹ - Em bé', 'id': 2527},
{'name': 'Sách Giáo Khoa - Giáo Trình', 'id': 2321},
{'name': 'Sách Học Ngoại Ngữ', 'id': 887},
{'name': 'Sách Tham Khảo', 'id':     2320},
{'name': 'Từ Điển', 'id': 897},
{'name': 'Sách Kiến Thức Tổng Hợp', 'id': 873},
{'name': 'Sách Khoa Học - Kỹ Thuật', 'id': 879},
{'name': 'Sách Lịch sử', 'id': 880},
{'name': 'Điện Ảnh - Nhạc - Họa', 'id': 881},
{'name': 'Truyện Tranh, Manga, Comic', 'id': 1084},
{'name': 'Sách Tôn Giáo - Tâm Linh', 'id'   : 861},
{'name': 'Sách Văn Hóa - Địa Lý - Du Lịch', 'id': 857},
{'name': 'Sách Chính Trị - Pháp Lý', 'id': 875},
{'name': 'Sách Nông - Lâm - Ngư Nghiệp', 'id': 882},
{'name': 'Sách Công Nghệ Thông Tin', 'id': 876},
{'name': 'Sách Y Học', 'id': 885},
{'name': 'Tạp Chí - Catalogue', 'id': 1468},
{'name': 'Sách Tâm lý - Giới tính', 'id': 868},
{'name': 'Sách Thường Thức - Gia Đình', 'id': 862},
{'name': 'Thể Dục - Thể Thao', 'id': 6905}
```

1.2 Crawling book from tiki

Let dive into each category and crawl about 400 items for that category.
1. I get the id of each product for each page

```
def get_product_per_page(book_cat, page):
    product_list_per_page = []
    product_per_page_link = f"https://tiki.vn/api/v2/products?category={book_cat}&limit={LIMIT}&page={page}"
    response  = requests.get(product_per_page_link, headers=HEADERS)
    products = json.loads(response.text)['data']
```

2. For each product in 1 page, I continue to crawl the infomation of that product.

```
products = json.loads(response.text)['data']
for product in (products):
    product_data = (get_product(product['id']))
    #print(product_data['name'])
```

```python
def get_product(product_id):
    time.sleep(1)
    product_info_link = f"https://tiki.vn/api/v2/products/{product_id}"
    response  = requests.get(product_info_link, headers=HEADERS)
    #print(response.status_code,response.text)
    product_info = json.loads(response.text)

    return product_info
```

And to prevent Tiki blocking my crawling, i need to set sleep time for each time i use request.get.

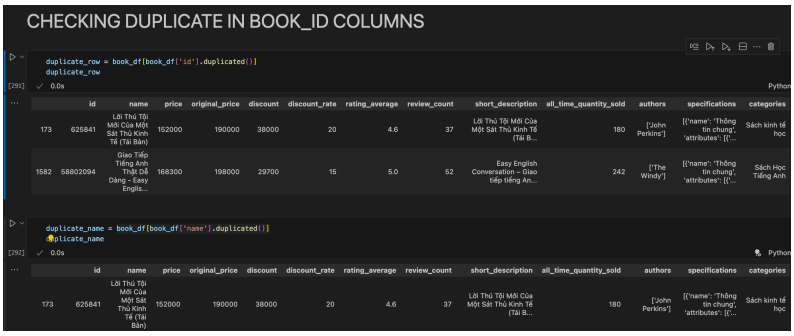
After doing that all categories i acquire around 6000 items

2 Data Cleaning

After crawling the items i need, i start perform data cleaning with following stage:

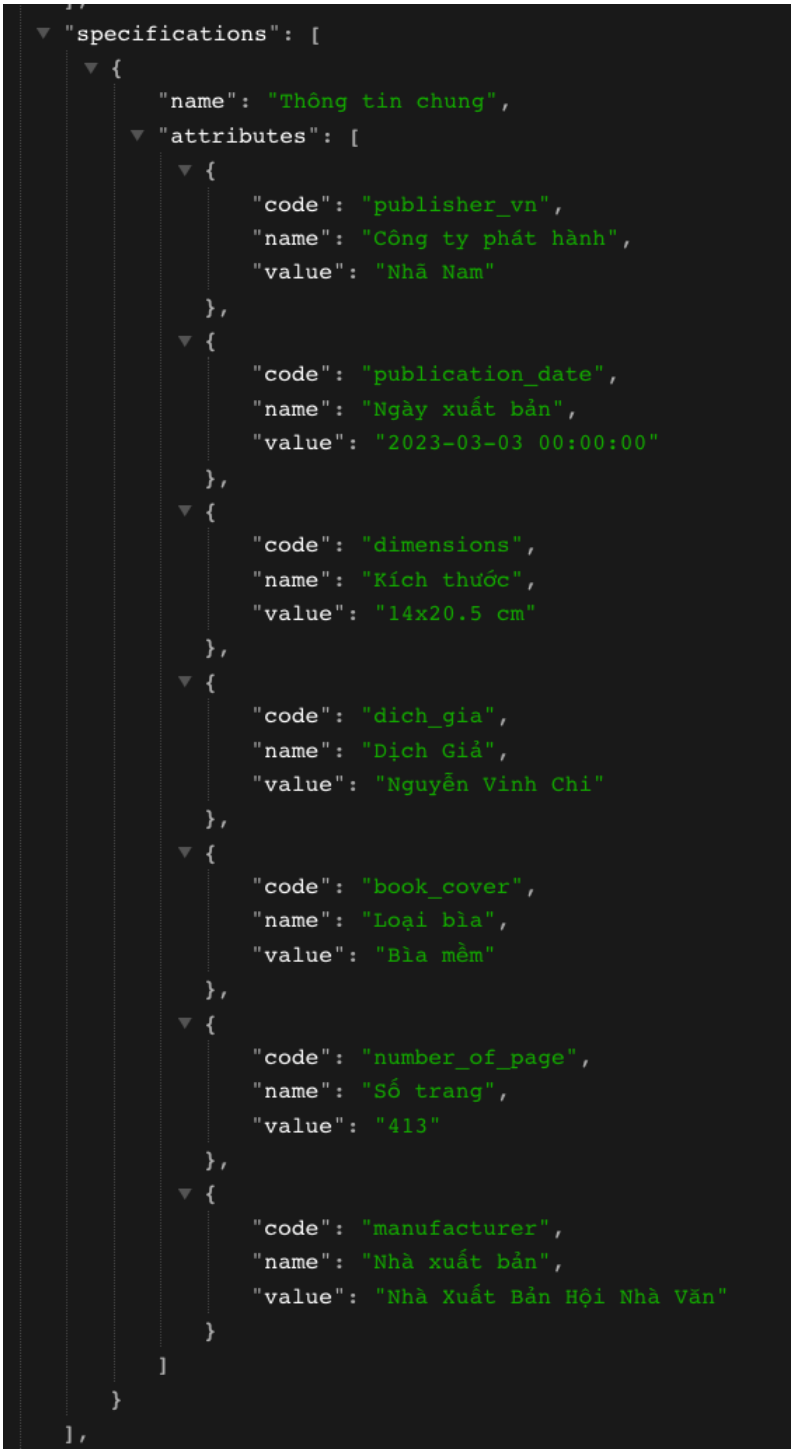
2.1 Checking duplicate

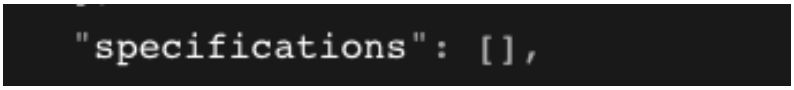
I start the cleaning by checking there is any duplicate book, by inspecting 2 columns, $book_id$ and $book_name$



And drop any that are duplicate

2.2 Working with specifications features





As you can see, the specifications features of each book doesnt have same value, so i need to preprocess it to improve my dataset.
After extract useful feature in the specifications attributes, drop any book that does not have $book_cover, number_of_page' manufacturer'$ in specifications. And we after the dataset with below attbiutes.

```
book_df = pd.read_csv('cleaned_book.csv')
print(book_df.shape)
book_df.columns
```

```
(3830, 16)

Index(['id', 'name', 'price', 'original_price', 'discount', 'discount_rate',
       'rating_average', 'review_count', 'short_description',
       'all_time_quantity_sold', 'authors', 'categories', 'publisher_vn',
       'book_cover', 'number_of_page', 'manufacturer'],
      dtype='object')
```

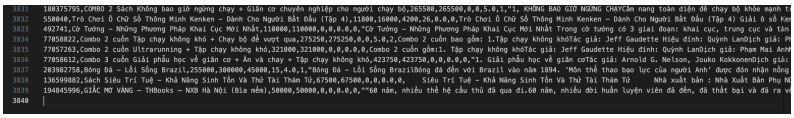
2.3 Weird value in $'numer_of_page'$ attribute

When i inspect the value of $'numer_of_page'$ attribute i relize that there are some value.

```
['384' '216' '280' '450' '559' '344' '536' '532' '612' '272' '268' '512'
 '644' '392' '211' '367' '257' '250' '116' '632' '346' '293' '1044' '436'
 '260' '236' '200' '336' nan '342' '504' '288' '416' '224' '228' '320'
 '312' '608' '192' '300' '420' '350' '244' '455' '306' '130' '332' '459'
 '276' '184' '168' '172' '352' '128' '310' '348' '750' '208' '196' '432'
 '296' '106' '159' '828' '399' '480' '256' '220' '592' '204' '412' '240'
 '556' '484' '492' '120' '308' '160' '372' '507' '232' '488' '278' '152'
 'combo' '300000' '473' '351' '376' '205' '316' '576' '456' '524' '1041'
 '780' '299' '206' '668' '1183' '177' '748' '328' '946' '435' '314' '428'
 '247' '687' '470' '424' '333' '452' '264' '1092' '284' '324' '222' '214'
 '270' '304' '323' '540' '714' '403' '1360' '448' '368' '582' '429' '548'
 '378' '112' '383' '396' '616' '784' '360' '382' '566' '258' '380' '327'
 '404' '692' '356' '558' '193' '286' '292' '408' '1268' '340' '366' '508'
 '102' '144' '215' '30' '203' '24' '444' '84' '400' '156' '40' '388' '138'
 '980' '16' '195' '96' '176' '52' '180' '36' '318' '108' '48' '386' '158'
 '72' '1312' '95' '80' '28' '174' '839' '143' '32' '364' '787' '136' '12'
 '6' '169' '167' '100' '248' '334' '170' '370' '164' '147' '76' '58' '295'
 '20' '10' '65' '98' '800' '145' '188' '44' '64' '379' '38' '476' '11'
 '406' '124' '162' '457' '45' '140' '291' '385' '330' '199' '230' '290'
 '496' '279' '209' '884' '148' '468' '422' '252' '302' '628' '794' '478'
 '343' '210' '226' '190' '298' '194' '440' '374' '146' '656' '728' '212'
 '410' '596' '564' '664' '123000' '271' '700' '246' '914' '294' '338'
 '281' '362' '329' '464' '289' '1169' '197' '660' '796' '249' '307' '503'
 '776' '1128' '568' '736' '339' '1050' '287' '242' '395' '267' '712' '263'
...
 '375' '2475' '227' '1482' '1808' '1027' '830' '1390' '1400' '282' '33'
 '245' '89000' '662' '675' '565' '221' '77' '620' '42' '122' '2372' '1728'
 '2812']
<class 'str'>
```

I thought that this data type should be numerical, but after inspect it, i see it

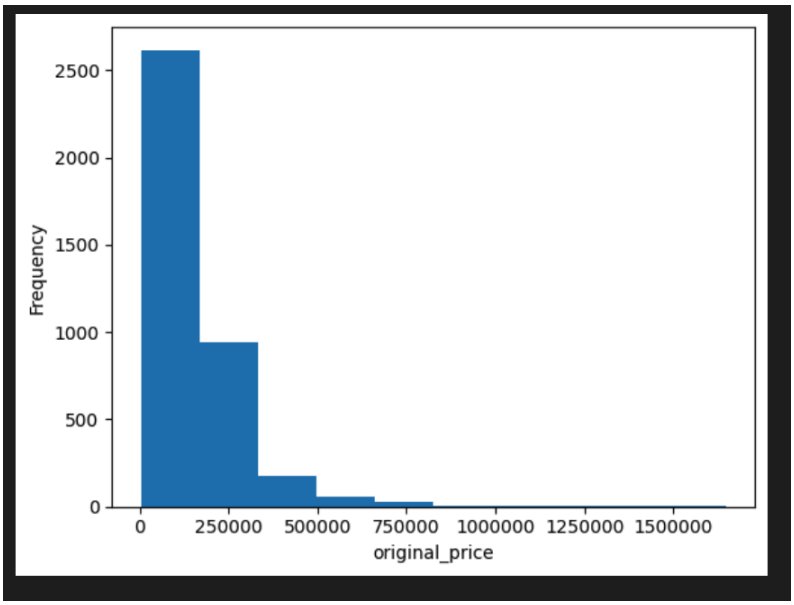
have string datatype. So first i need to convert it back to number, and drop any row that have value not a number.
Another weird thing i see is a book with 300000 pages. So i check that book on Tiki and turn out that is a mistake of the seller. That book only have around 300 not 300000. So to make sure i drop any book with >1000 page. And the data after I clean reduced to 4000 items



3 Perform EDA

3.1 EDA for price prediction

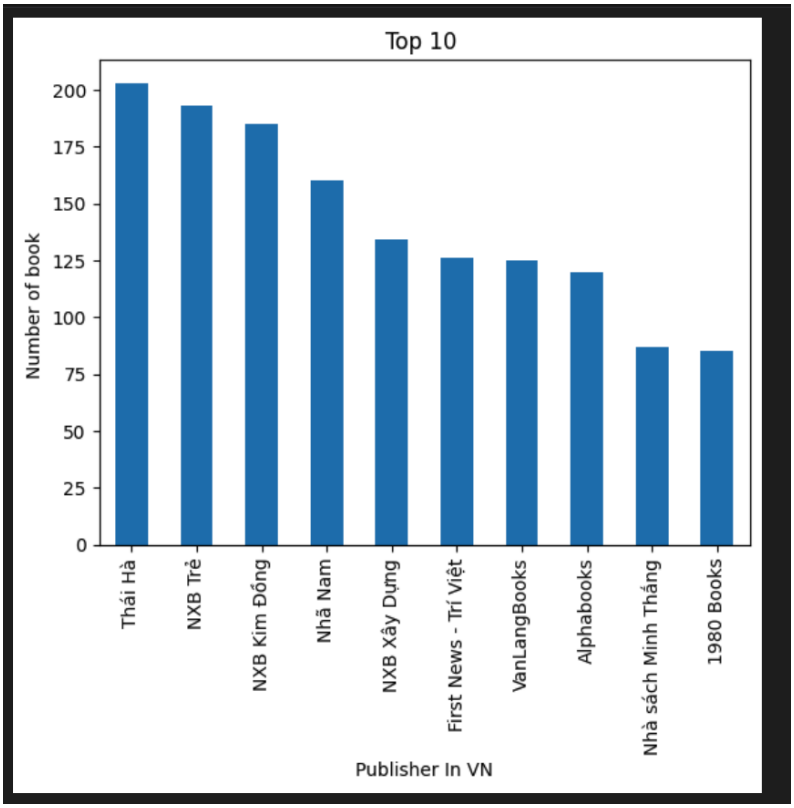
First I dive into our target variables and see its distribution



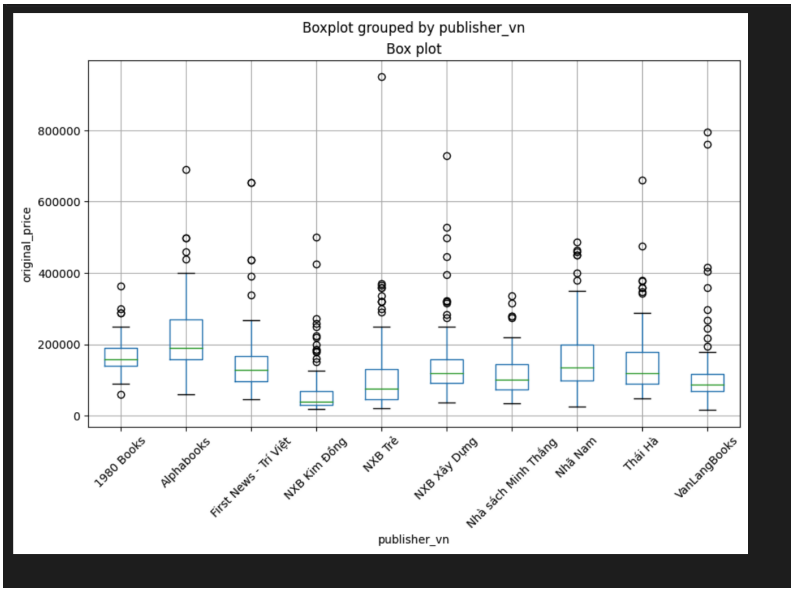
After that I also investigate other feature that contribute to the price of a book

3.1.1 *Publisher_vn*

Here are top 10 publisher in VN



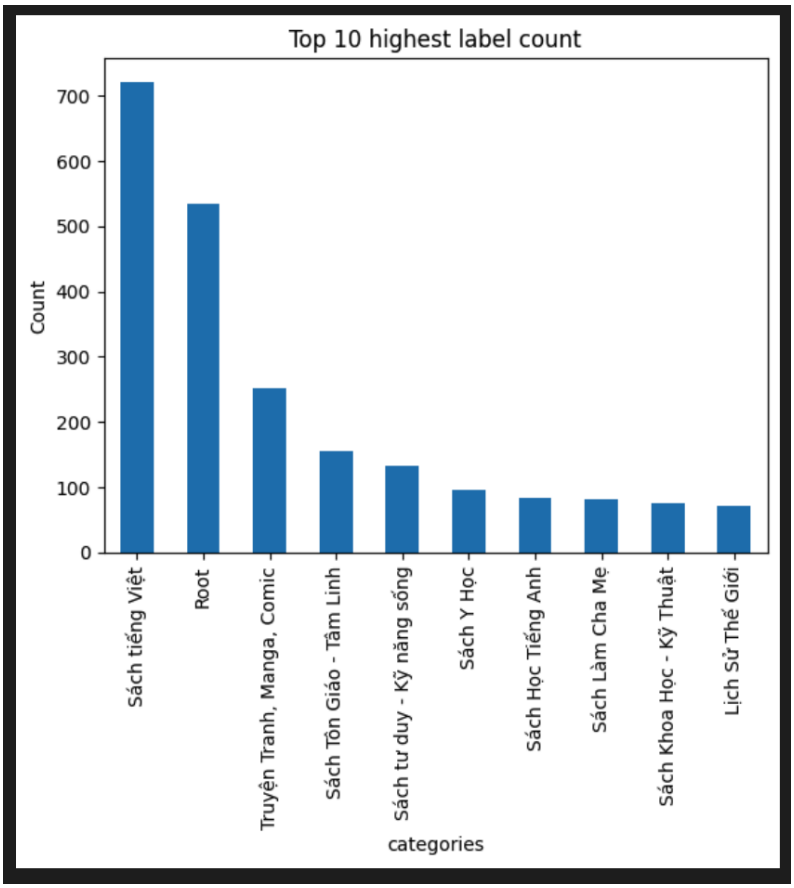
And we can see the price distribution for each publisher, and see that the price vary between publisher



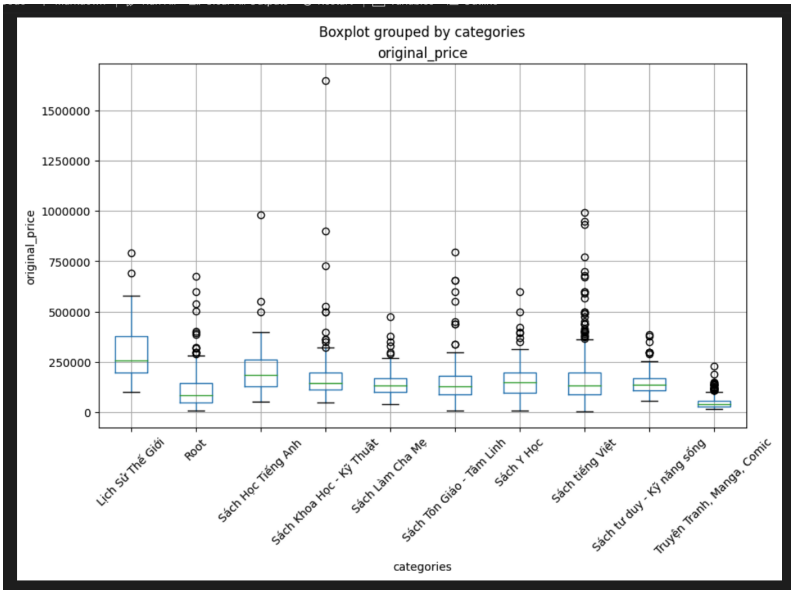
It implies that the price of a book may depend on the publisher

3.1.2 *Categories*

Here are top 10 most favourite book categories from in Tiki



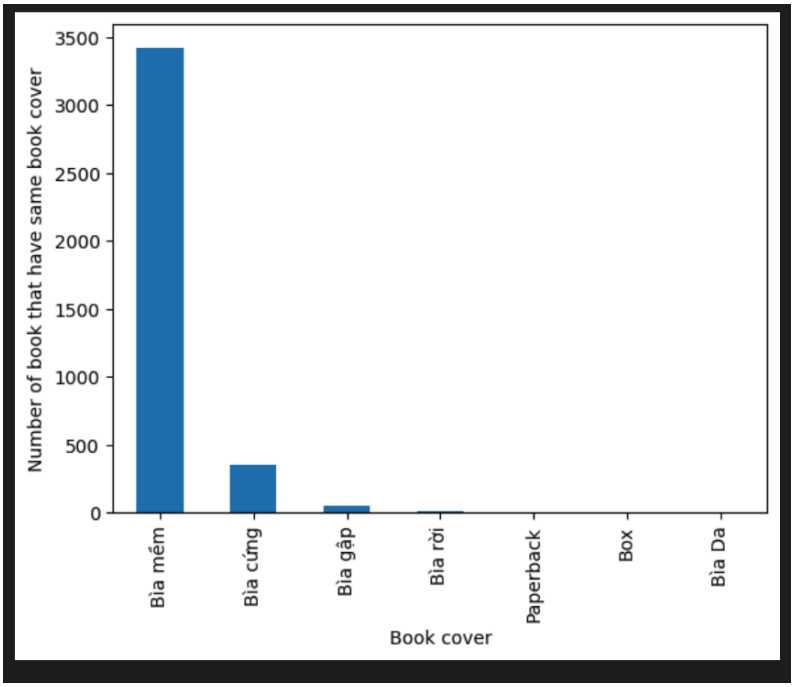
And we can see the price distribution for each category vary slightly



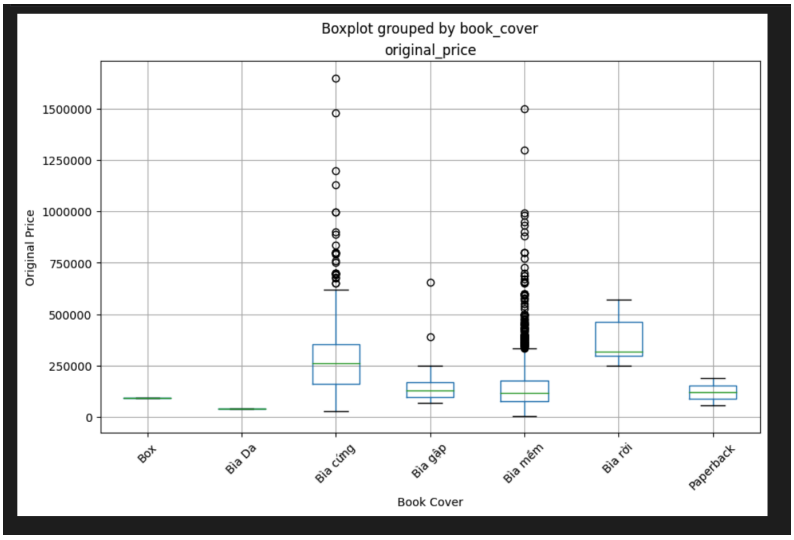
It may suggest that the price of a book may depend on the categrory

3.1.3 *Book_cover*

Here are top the count of book cover for each type of book

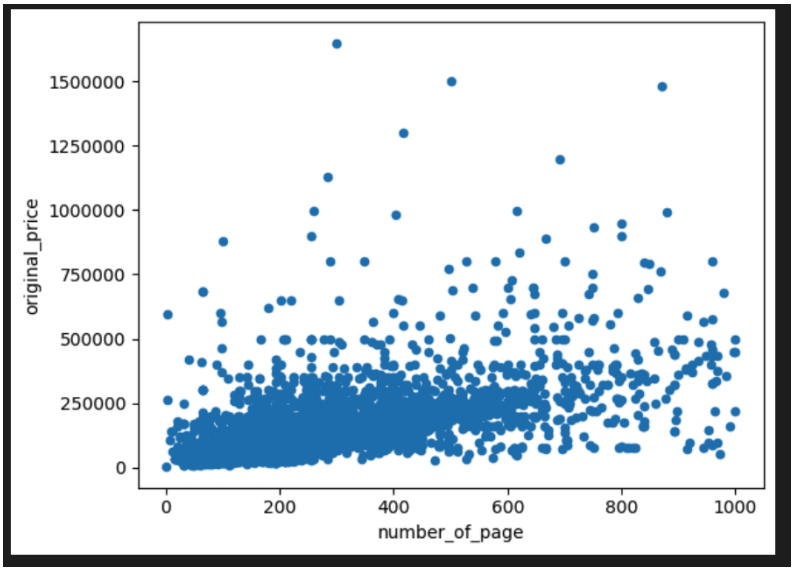


And each $book_cover$ type has different price distribution



3.1.4 *number_of_page*

Here we can see the distribution of price based on the number of page. We can see that have a slightly upward trend.



To make sure it has relation, we can use $correlation_coefficient$

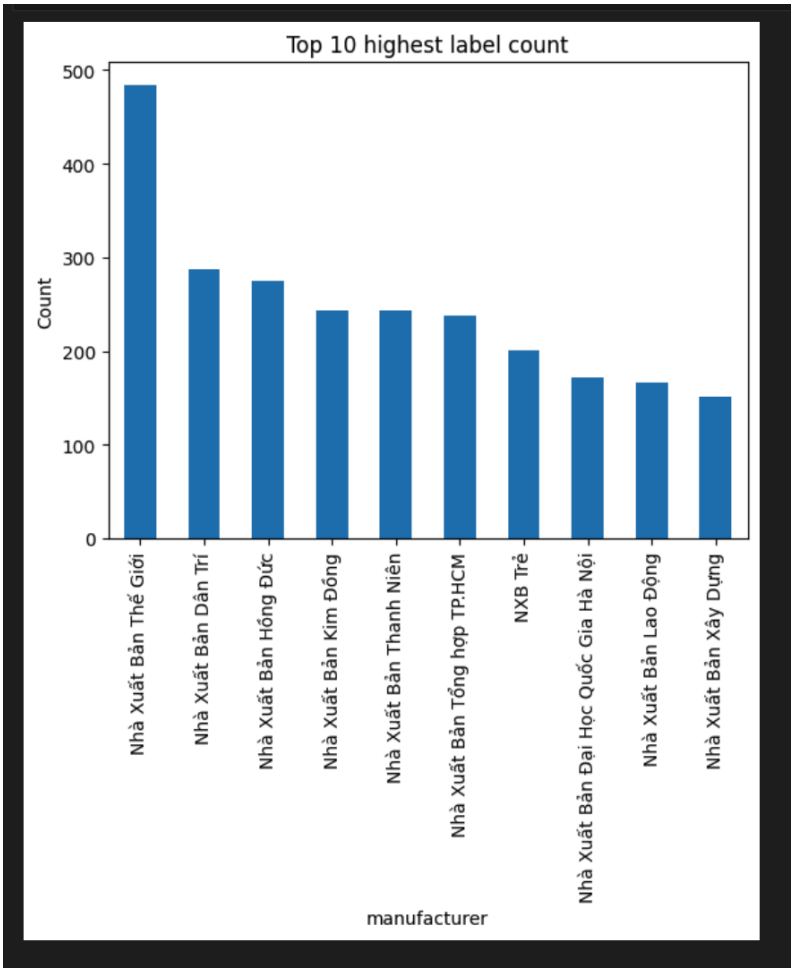
```
correlation_coefficient = book_df['number_of_page'].corr(book_df['original_price'])

print(correlation_coefficient)
```

✓ 0.0s

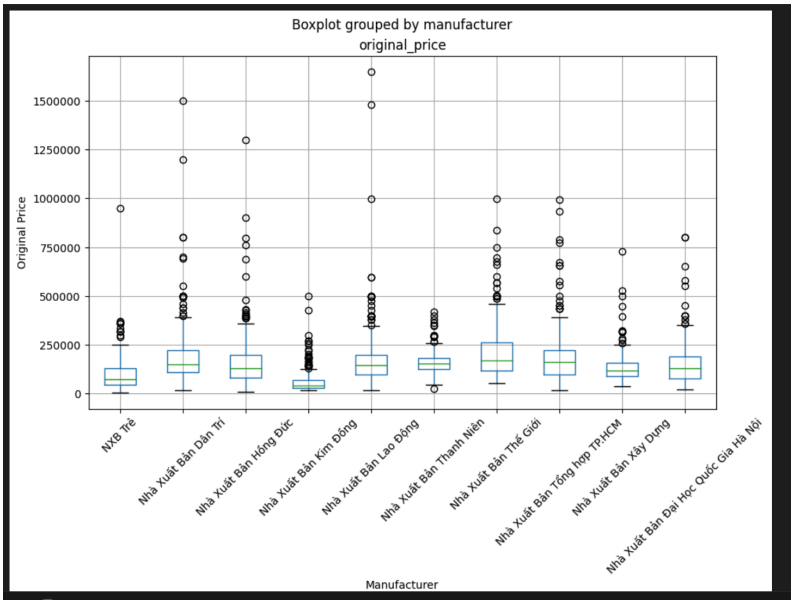
```
0.5425250296811849
```

3.1.5 *manufactures*

Here are 10 main manufacturer for most of the book



And each manufacturer has different price range on the price of the book



4 Predict book price

4.1 Feature engineering

4.1.1 Target Encoding

Problem with our a dataset is that, the label in publisher, and manufacturer is very high. Thus, using 1-hot encoding, or label encoding will not give us a good result. And to solve that problem, we need new encoding method that is Target Encoding
A target encoding is any kind of encoding that replaces a feature's categories with some number derived from the target.

Before encoding:

	categories	publisher_vn	book_cover	number_of_page	manufacturer
3142	Tin Học Văn Phòng	Thái Hà	Bìa mềm	360	Nhà Xuất Bản Công Thương
3263	Sách Y Học	Thái Hà	Bìa mềm	264	Nhà Xuất Bản Công Thương
829	Sách Kiến Thức - Kỹ Năng Cho Trẻ	Tân Việt	Bìa mềm	31	Nhà Xuất Bản Mỹ Thuật
700	Sách nghệ thuật sống đẹp	Công ty cổ phần Ahora	Bìa mềm	556	Nhà Xuất Bản Thế Giới
3779	Sách tiếng Việt	Công ty Sách Bảo Trang	Bìa mềm	192	Nhà Xuất Bản Tri Thức
...	...	...	...	...	...
1130	Sách Học Tiếng Anh	Zenbooks	Bìa mềm	327	Nhà Xuất Bản Đà Nẵng
1294	Sách tham khảo cấp III	Cty Sách Sách Hay	Bìa mềm	204	Nhà Xuất Bản Đại Học Quốc Gia Hà Nội
860	Sách Làm Cha Mẹ	Đinh Tị	Bìa mềm	396	Nhà Xuất Bản Thanh Niên
3507	Sách Tâm Lý Tuổi Teen	First News - Trí Việt	Bìa mềm	152	Nhà Xuất Bản Tổng Hợp
3174	Thiết Kế - Đồ Họa	Công Ty Cổ Phần Phát Triển Giáo Dục Ngọc Hà	Bìa mềm	120	Nhà Xuất Bản Đại Học Quốc Gia Hà Nội

3064 rows × 5 columns

After encoding:

```
(3064, 5)          categories   publisher_vn     book_cover  number_of_page  \
3142  177550.865403  146753.719575  141483.233209             360
3263  159453.944781  146753.719575  141483.233209             264
829   100696.923096  105234.412971  141483.233209              31
700   131412.820537  163476.923221  141483.233209             556
3779  162966.011930  188453.846443  141483.233209             192
...             ...            ...            ...             ...
1130  192190.446659  123287.044550  141483.233209             327
1294  185118.414936  112953.846443  141483.233209             204
860   150237.428252  119494.230805  141483.233209             396
3507  115283.076940  158884.835170  141483.233209             152
3174  150169.230805  159453.846443  141483.233209             120

       manufacturer
3142  152998.557701
3263  152998.557701
829   281648.160560
700   203974.463871
3779  194849.946346
...             ...
1130  140957.577506
1294  160615.908227
860   157428.836088
3507  197488.461611
3174  160615.908227

[3064 rows x 5 columns]
```

4.1.2 Scaling

We aslo do the normalization to the features improves the performance and training stability for the model.

4.2 Choosing model

4.2.1 Linear Regression

First model we use is Linear Regression due to it simplicity to understand and implement. And after that we got the score of the model using cross validation.

Linear Regression :

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score

linRegressor = LinearRegression()

scores = cross_val_score(linRegressor, X_train_normalized, y_train, cv=5, scoring='neg_mean_squared_error')

# Convert the negative MSE scores to positive RMSE scores
rmse_scores = np.sqrt(-scores)

# Print the RMSE scores for each fold
print("RMSE scores:", rmse_scores)
print("Mean RMSE:", rmse_scores.mean())
```

```
[9]  ✓ 0.3s

...  RMSE scores: [86010.72222065 88914.56453534 71216.49616304 90418.80553456
      88223.08345698]
     Mean RMSE: 84956.73438211411
```

4.2.2 Random Forest

Another model we can use is Random Forest

Random Forest :

```
from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor(n_estimators=100, random_state=42)
scores = cross_val_score(forest_reg, X_train_normalized, y_train, cv=5, scoring='neg_mean_squared_error')

# Convert the negative MSE scores to positive RMSE scores
rmse_scores = np.sqrt(-scores)

# Print the RMSE scores for each fold
print("RMSE scores:", rmse_scores)
print("Mean RMSE:", rmse_scores.mean())
```

[10] ✓ 3.2s

```
/usr/local/lib/python3.11/site-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change th
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.11/site-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change th
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.11/site-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change th
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.11/site-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change th
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.11/site-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change th
  estimator.fit(X_train, y_train, **fit_params)
RMSE scores: [83640.68260803 87327.12332847 74192.03021814 87631.80091609
 95775.10556408]
Mean RMSE: 85713.3485269628
```

As we can see both model have the same score, however we not fine tuning the random forest yet. So we need to tune it to get even better score

4.2.2.1 Random Search

To tune the model we can use Random Search, to search for the best combination of paramater for a model

RANDOMIZE SEARCH

```
from sklearn.model_selection import RandomizedSearchCV

params = {
        'n_estimators': [np.random.randint(low=1, high=100)],
        'max_features': [np.random.randint(low=1, high=5)],
    }

forest_reg = RandomForestRegressor(random_state=42)
rnd_search = RandomizedSearchCV(forest_reg, param_distributions=params,
                                n_iter=10, cv=5, scoring='neg_mean_squared_error', random_state=42)
rnd_search.fit(X_train_normalized, y_train)
```

11] ✓ 1.2s

And evaluating with cross validation

```
RMSE scores: [84858.31141495 85106.90452188 69829.65116489 87117.74967399
 90226.21622413]
Mean RMSE: 83427.76659996875
```

And the result we get using the test set

```
RMSE scores: 641195.3465878738
```

5 Book genres Classification based on Image

One additional issue I observe with the book dataset is the presence of a significant number of uncategorized books, indicating a lack of careful categorization.

```
Sách tiếng Việt                721
Root                           535
Truyện Tranh, Manga, Comic     252
Sách Tôn Giáo - Tâm Linh       156
Sách tư duy - Kỹ năng sống     133
                               ...
Sách Nông - Lâm - Ngư Nghiệp     1
Từ Điển Tiếng Hàn                1
11.11 Campaign                   1
Sách tài chính, kế toán          1
Tranh Truyện                     1
Name: categories, Length: 82, dtype: int64
```

5.1 Modelling

For this problem i have chosen 2 different model , one is from traditional machine learning, and other is from deep learning

5.1.1 Support vector machine

For this model to work, we need to load the images from the dataset and flatten it into vector array.

```
images_np = np.array(images, dtype=np.float32)
targets_np = np.array(targets, dtype=np.float32)
flatten_img_np =  np.array(flatten_img, dtype=np.float32)
print(images_np.shape)
print(targets_np.shape)
print(flatten_img_np.shape)
```

✓ 0.1s

```
(100, 280, 200, 3)
(100,)
(100, 168000)
```

And we tuning the model using grid search, and based on cross validation scores

```
from sklearn import svm
from sklearn.model_selection import GridSearchCV,cross_val_score

param_grid = [
    {'C':[1,10,100,1000], 'kernel':["linear"]},
    {'C':[1,10,100,1000], 'gamma':[0.001, 0.0001] ,'kernel':["rbf"]}
]
svc = svm.SVC()
grid_search = GridSearchCV(svc, param_grid, cv=5, scoring='accuracy')

grid_search.fit(X_train,y_train)
```

```
from sklearn.metrics import accuracy_score

svc_best =  grid_search.best_estimator_

cv_scrore = cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=5, scoring='accuracy')
print(cv_scrore)
print(np.mean(cv_scrore))
```

✓ 2.1s

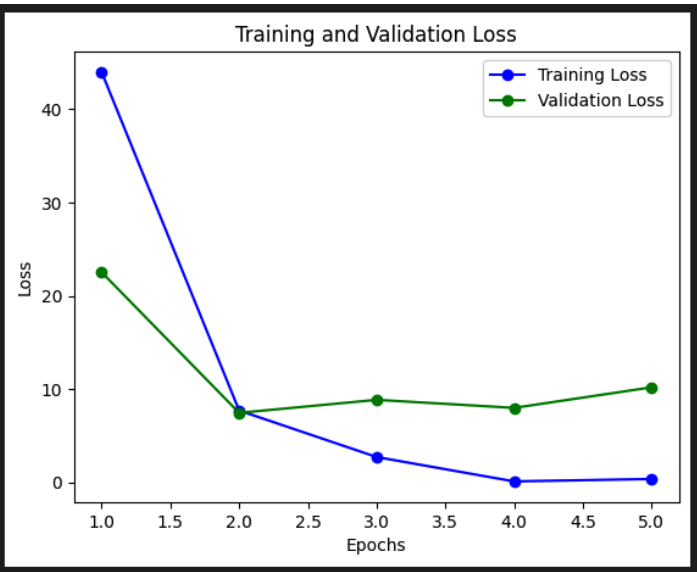
```
[0.5625 0.625  0.4375 0.4375 0.4375]
0.5
```

5.1.2 Transfer learning with VGG16

As we can see that the traditional machine learning not performing well on the image dataset, so we will explore other technique. Also due to the limitation of my hardware, i will get the pretrained model, VGG16 in this case, and train the last layer of this model only.

```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(280,200, 3), classes=3)
for layer in base_model.layers:
    layer.trainable = False

transfer_model = Sequential()
transfer_model.add(base_model)

transfer_model.add(Flatten())

transfer_model.add(Dense(256,activation="relu"))
transfer_model.add(Dense(len(CATS),activation="softmax"))

transfer_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history  =  transfer_model.fit(X_train_img, y_train_img,batch_size=16, epochs=5, validation_data=(X_val_img, y_val_img))
```

And here is the score the model obtain, we can see clearly that the deep learning model perform better on this type of dataset




```
loss, accuracy = transfer_model.evaluate(X_test_img, y_test_img)
print(accuracy)
```

```
1/1 [==============================] - 1s 1s/step - loss: 11.1550 - accuracy: 0.6000
0.6000000238418579
```