

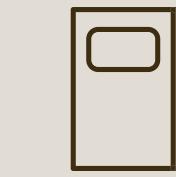
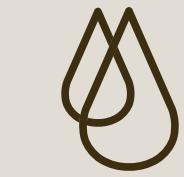
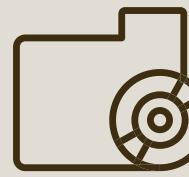
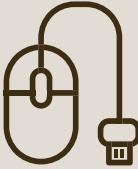
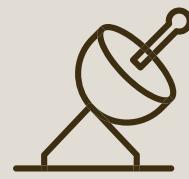
NAME	QUANTITY	PRICE
 <b>Horizon Case</b> Counter-Strike: Global Offensive	79,396	Starting at: 4.634,89đ
 <b>AK-47   Redline (Field-Tested)</b> Counter-Strike: Global Offensive	981	Starting at: 331.952,42đ
 <b>Operation Wildfire Case</b> Counter-Strike: Global Offensive	16,974	Starting at: 14.687,80đ
 <b>M4A1-S   Leaded Glass (Field-Tested)</b> Counter-Strike: Global Offensive	591	Starting at: 115.487,56đ
 <b>Operation Bravo Case</b> Counter-Strike: Global Offensive	864	Starting at: 903.979đ

Chủ đề : Phân  
tích thị trường  
CS:GO Skins  
trong Steam  
Community  
Market

## Bảng phân công thành viên nhóm 11

MSSV	Tên thành viên	Phần phụ trách	Mức độ hoàn thành
20120236	Phạm Tấn Anh Vũ	Tìm hiểu thị trường CSGO + Crawl dữ liệu	100%
19120151	Nguyễn Trí Tuệ	Tiền xử lí + Trực quan hóa + Phân tích dữ liệu Làm slide	100%
20120130	Đinh Thị Hoàng Linh	Tiền xử lí + Xây dựng mô hình dự đoán	100%

# GIỚI THIỆU ĐỒ ÁN



# I. Giới thiệu đồ án

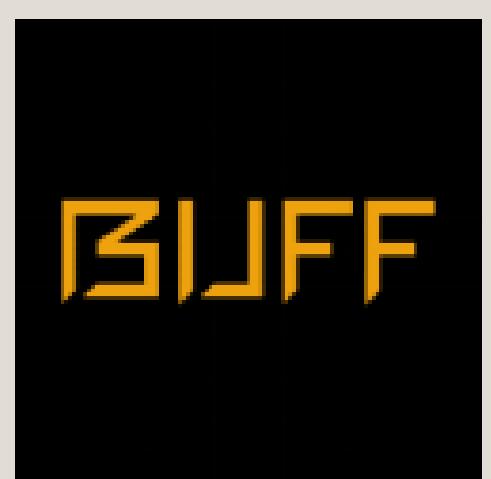


CS:GO là tựa game bắn súng góc nhìn thứ nhất được phát triển bởi Valve.

Skins là 1 phần không thể thiếu của game, được ra mắt vào năm 2013, nó đã tạo nên tác động vô cùng mạnh mẽ và đã trở thành “xương sống” của tựa game. Tính đến 12/2021, tổng giá trị toàn bộ skins vượt mốc 1,3 tỷ dollars.

Skins là các thiết kế tùy chọn với họa tiết được vẽ lên các loại vũ khí trong game, tạo nên sự độc đáo, mang lại cảm giác hứng thú cho người chơi.

# I. Giới thiệu đồ án



Ngoài skins ra còn có nhiều vật phẩm khác như stickers, agents, music kits,.....  
Tuy nhiên đồ án này nhóm chỉ tập trung vào skins và stickers.

Valve thiết lập cơ chế trade vật phẩm giữa người chơi với nhau thông qua nền tảng steam, người chơi có thể trao đổi vật phẩm trực tiếp hoặc bán trên chợ cộng đồng (community market) để đổi lấy tiền trên steam (Tuy nhiên steam không cho phép rút tiền steam ra tiền thật nên từ đó đã tạo ra nhiều marketplace bên thứ ba để trao đổi mua bán skins, nổi tiếng như buff163, skinsport, dmarket,...).

# I. Giới thiệu đồ án

---

Trong đồ án này, nhóm sẽ tiến hành các công đoạn thu thập dữ liệu, khám phá dữ liệu, trực quan hóa dữ liệu và mô hình hóa dữ liệu bằng học máy.

Mục tiêu : Tìm mối liên hệ giữa các thuộc tính của skins với giá, sử dụng mô hình để dự đoán giá skins từ đó đưa ra quyết định đầu tư phù hợp.

# Quy trình khoa học dữ liệu

---



# I. Thu thập & tiền xử lý dữ liệu

## ● Thu thập dữ liệu :

Việc thu thập dữ liệu sẽ được chia ra thành 2 công đoạn : thu thập thông tin về tất cả skins và thu thập lịch sử bán của chúng.

Import các thư viện phục vụ việc thu thập dữ liệu :

```
1 import requests
2 import json
3 import sys
4
5 import pandas as pd
6 import numpy as np
7
8 import datetime
9 import time
10 import random
11
12 import warnings
13 from IPython.display import Image
14 warnings.filterwarnings(action = 'ignore', category = FutureWarning)
```

✓ 9.3s

Chuẩn bị cookies :

Vì một vài request tới market của steam yêu cầu phải có thông tin tài khoản (login credentials) của người thu thập dữ liệu, nên ta cần chuẩn bị nó.

```
steamLoginSecure = '76561198329918048%7C%7CeyAidHlwIjogIkpxVCIsICJhbGciOiAiRWREU0EiIh0  
cookie = {'steamLoginSecure': steamLoginSecure}
```

Lưu ý : Vì mỗi lần đóng/mở trang sẽ cấp 1 cookie khác nên cần phải cập nhật cookie cho mỗi lần lấy dữ liệu.

## - Thu thập dữ liệu :

Công đoạn 1 : Tiến hành thu thập dữ liệu tất cả skins được đăng bán trên market

Ban đầu ta thu thập số lượng vật phẩm tồn tại trên market và lưu vào biến total\_Items :

```
1 URL = 'https://steamcommunity.com/market/search/render/?query=&start=0&count=100&search_descriptions=0&sort_column=price&sort_dir=desc&appid=730&norender=1'
2 Items_get = requests.get(url = URL, cookies = cookie)
3 Items = json.loads(Items_get.text)
4
5 data = Items['results']
6 total_Items = Items['total_count']
```

-Tiếp theo, ta tiến hành thu thập dữ liệu raw về từng loại vật phẩm trên market :

```
1 # get all items information, because steam only allow 100 items/request, and ~20 requests/minute so everytime request return code 400,
2 pos = 0
3 while (pos < total_Items) :
4     while (True) :
5         URL = 'https://steamcommunity.com/market/search/render/?query=&start=' + str(pos) + '&count=100&search_descriptions=0&sort_col
6         Items_get = requests.get(url = URL, cookies = cookie)
7         if(Items_get.status_code != 200) :
8             time.sleep(70)
9             break
10
11     Items = json.loads(Items_get.text)
12     data = Items['results']
13     for item in data :
14         Items_list.append(item['asset_description'])
15
16     pos += 100
```

-Cột asset\_description sẽ là nơi chứa dữ liệu hữu ích phục vụ việc phân tích, ta tạo 1 Items\_list để chứa các asset\_description của từng vật phẩm.

Các thuộc tính của từng vật phẩm thu thập được :

```
1 print([col for col in Items_list[0]])  
✓ 0.9s  
Python  
['appid', 'classid', 'instanceid', 'currency', 'background_color', 'icon_url', 'icon_url_large', 'descriptions', 'tradable', 'actions', 'name', 'name_color', 'type', 'market_name',  
'market_hash_name', 'market_actions', 'commodity', 'market_tradable_restriction', 'marketable']
```

-Công đoạn 2 : Tiền xử lý dữ liệu và thu thập thêm dữ liệu về giá

Ta sẽ loop qua hết mọi vật phẩm, tiền xử lý để trích xuất các thuộc tính ý nghĩa đồng thời thu thập thêm dữ liệu về lịch sử giá bán tương ứng của từng vật phẩm. Cuối cùng ta bỏ vào DataFrame items\_df và xuất ra file 'CSGO\_Market.csv'

Tiền xử lý dữ liệu thuộc tính của vật phẩm :

```
for cur_item in Items_list :
    item_name      = cur_item['market_hash_name']           # Lấy tên vật phẩm
    item_info       = cur_item['type']                      # Lấy đoạn string chứa các thông tin rarity, extra, type
    item_description = cur_item['descriptions'][0]['value']  # Lấy đoạn string chứa thông tin exterior

    item_exterior = _get_exterior(item_description)        # Lấy exterior
    item_rarity   = _get_rarity(item_info)                  # Lấy rarity
    item_extra    = _get_extra(item_info)                   # Lấy extra
    item_type     = _get_type(item_info)                    # Lấy type
```

Dữ liệu về giá có dạng lịch sử bán theo từng ngày, do đó các thuộc tính mới được tạo ra để phục vụ các công việc phân tích & mô hình hóa sau này.  
Các thuộc tính được tạo mới :

```
'Average volume sale',
'Average price (VND)', 'Max price (VND)', 'Days to reach max',
'Min price (VND)', 'Days to reach min', 'Price change (%)',
| 'Time on market (days)']
```

Lấy dữ liệu về giá của từng skin :

```
def _get_price_request(name) :          # get price data from a skin
    while (True) :
        URL                  = 'http://steamcommunity.com/market/pricehistory/?country=PT&currency=3&appid=730&market_hash_name=' + name
        item_price_data = requests.get(url = URL, cookies = cookie);
        if (item_price_data.status_code != 200) :
            #print(name, item_price_data.status_code)
            continue
    return json.loads(item_price_data.text)
```

## Tiền xử lý thông tin về giá cả :

```
while (r <= len(item_prices)) :
    cur_date = datetime.date(1, 1, 1)
    if (r < len(item_prices)) :
        cur_date = datetime.datetime.strptime(item_prices[r][0][0 : 11], '%b %d %Y').date()

    if (cur_date != last_date and r != 0) :
        avg_price = np.mean(tmp_price_list[l : r])      # tính giá trung bình trong ngày last_date
        avg_vol   = np.mean(tmp_vol_list[l : r])         # tính số lượng bán trung bình trong ngày last_date

        price_list.append(avg_price)
        vol_list.append(avg_vol)
        dates.append(last_date)

        l = r

    if (r < len(item_prices)) :
        tmp_price_list.append(item_prices[r][1])
        tmp_vol_list.append(float(item_prices[r][2]))
```

## Tiền xử lý thông tin về giá cả :

```
    last_date = cur_date
    r += 1

    avg_price = np.mean(price_list)          # tính giá trung bình của vật phẩm
    avg_vol   = np.mean(vol_list)            # tính số lượng bán trung bình của vật phẩm

    max_price = max(price_list)             # tính giá max
    min_price = min(price_list)             # tính giá min

    max_idx    = price_list.index(max_price)
    days_to_max = (dates[max_idx] - dates[0]).days  # tính số lượng ngày để đạt max

    min_idx    = price_list.index(min_price)
    days_to_min = (dates[min_idx] - dates[0]).days  # tính số lượng ngày để đạt min

    if (price_list[0] == 0) : price_list[0] = 1
    price_change = (price_list[-1] - price_list[0]) / price_list[0]          # tính tỉ lệ biến động vật phẩ
    price_change = round(price_change, 2)

    time_on_market = (dates[-1] - dates[0]).days           # tính số ngày vật phẩm đã tồn tại
```

Ta append toàn bộ vào DataFrame items\_df

```
item_dict = {'Name' : item_name, 'Type' : item_type, 'Exterior' : item_exterior, 'Rarity' : item_rarity,
             'Extra' : item_extra, 'Average volume sale' : avg_vol, 'Average price (VND)' : avg_price,
             'Max price (VND)' : max_price, 'Days to reach max' : days_to_max, 'Min price (VND)' : min_price,
             'Days to reach min' : days_to_min, 'Price change (%)' : price_change, 'Time on market (days)' : time_on_market}

cur_item_df = pd.DataFrame(item_dict, index = [0])
items_df    = items_df.append(cur_item_df, ignore_index = True)      # Bỏ vào dataframe items_df
```

Cuối cùng, xuất ra file csv

```
• 1 items_df.to_csv('CSGO_Market.csv', index = False)
```

### **III. Khám phá dữ liệu :**

- Dữ liệu đầu vào : Tập CSGO\_Market.csv

- Thông tin về tập dữ liệu :

### **Mỗi dòng có ý nghĩa gì ?**

Dữ liệu ở mỗi dòng thể hiện một đối tượng có đầy đủ các thuộc tính như Tên , Loại skin , Độ bền , Độ hiếm , Extra , Số lượng bán ra trung bình trên ngày , Giá ( trung bình ) , Giá ( cao nhất ) , Số ngày giữ giá cao nhất , Giá thấp nhất , Số ngày giữ giá thấp nhất , % Thay đổi giá , Thời gian xuất hiện trên market .

### **Mỗi cột có ý nghĩa gì?**

Name : object - Thể hiện tên của skin

Type : object - Loại skin thu thập là gì

Exterior : object - Độ bền của skin

Rarity : object - Độ hiếm của skin đó

Extra : object - Phân loại đặc điểm skin

Average volume sale : float64 - Số lượng bán ra trung bình trong 1 ngày

Average price (VND) : float64 - Giá tiền trung bình mà skin đó bán ra

Max price (VND) : float64 - Giá tiền cao nhất mà skin đó bán ra

Days to reach max : int64 - Số ngày skin đó giữ giá cao nhất

Min price (VND) : float64 - Giá tiền thấp nhất mà skin đó bán ra

Days to reach min : int64 - Số ngày skin đó giữ giá thấp nhất

Price change (%) : float64 - Tỉ lệ thay đổi giá của skin

Time on market (days) : int64 - Số ngày skin đã xuất hiện trên market

Kiểu phân bố dữ liệu ở mỗi cột là ngẫu nhiên không theo tỉ lệ ! và không có chiều hướng tăng hay giảm

**Có tổng cộng 19170 dòng và 13 cột**

### **III. Đặt câu hỏi :**

## 1.Type nào được đăng bán nhiều nhất trên market

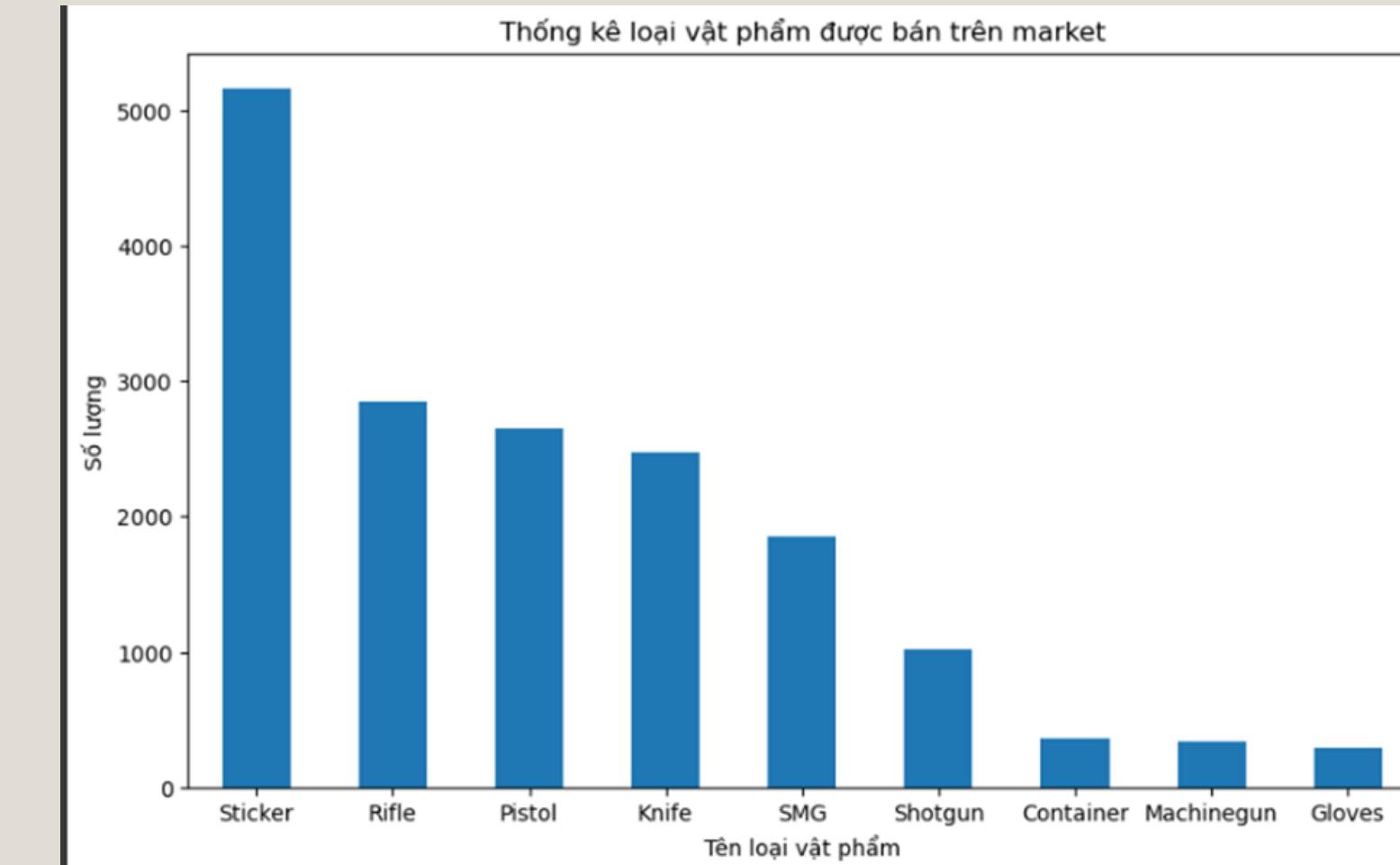
Nhằm để hiểu hơn về market đang được giao dịch đăng bán nhiều nhất ở kiểu nào của skin

Về mặt Type , khi biết market đang có những Type nào được bán nhiều , có thể từ đó khai thác các skin tiềm năng mà người chơi đang sở hữu

Nhận xét :

Sticker là vật phẩm được giao dịch nhiều nhất trên market , cùng với đó là Rifle , Pistol , Knife . Tuy nhiên Sticker chiếm đa số với hơn 5000 giao dịch Vậy khi bạn đang sở hữu Sticker tức là bạn đang có những cơ hội lớn để tham gia vào market và thu lại lợi nhuận từ việc bán chúng Tuy nhiên , Sticker cũng như các Type khác , chúng có nhiều phân loại về độ hiếm cũng như Extra đặc biệt , và nó cũng là một trong những điểm đánh giá xem Sticker của bạn đang sở hữu có đang là nhu cầu của người mua hay không ?

Để biết được điều đó , ta cần thống kê giữa Type và Extra



## 2.Type có skin Extra nào thì được bán nhiều nhất ?

Extra được chia thành 3 loại 'Souvenir' , 'StatTrak™' , 'Normal' , biết được đặc điểm này , chúng ta sẽ chia dữ liệu của Type theo 3 loại này . Từ đó thống kê ra các thông số xem thị trường thích nhất loại skin có đặc điểm nào !

rcParams[] # đặt kích thước hiển thị của biểu đồ  
biểu đồ dạng line sẽ hiển thị df\_Type được thống kê bên trên , bên dưới là các biểu đồ dạng cột thống kê cho các "Type" có Extra là "StatTrak", "Souvenir", "Normal"

Dữ liệu thống kê được thể hiện bằng cách lưu các Type có giá trị "StatTrak", "Souvenir", "Normal" vào các list tương ứng thông qua từng dòng lặp . Với mỗi vòng lặp tương ứng là một giá trị i trong df\_Type

```
plt.rcParams["figure.figsize"] = [12, 6]
df_Type.plot.line(marker = '.')

X = list(df_Type.keys())
list_Extra = ['Souvenir', 'StatTrak™', 'Normal']
list_Souvenir = []
list_Stattrak = []
list_Normal = []
for i in X:
    a = df.loc[df['Type'] == i]
    temp = a['Extra'].value_counts(dropna=False) # count value in columns Extra
    if ('StatTrak™' in list(temp.keys())):
        list_Stattrak.append(temp['StatTrak™']) # sum value = Stattrak in column
    else:
        list_Stattrak.append(0)

    if ('Souvenir' in list(temp.keys())):
        list_Souvenir.append(temp['Souvenir'])
    else:
        list_Souvenir.append(0)

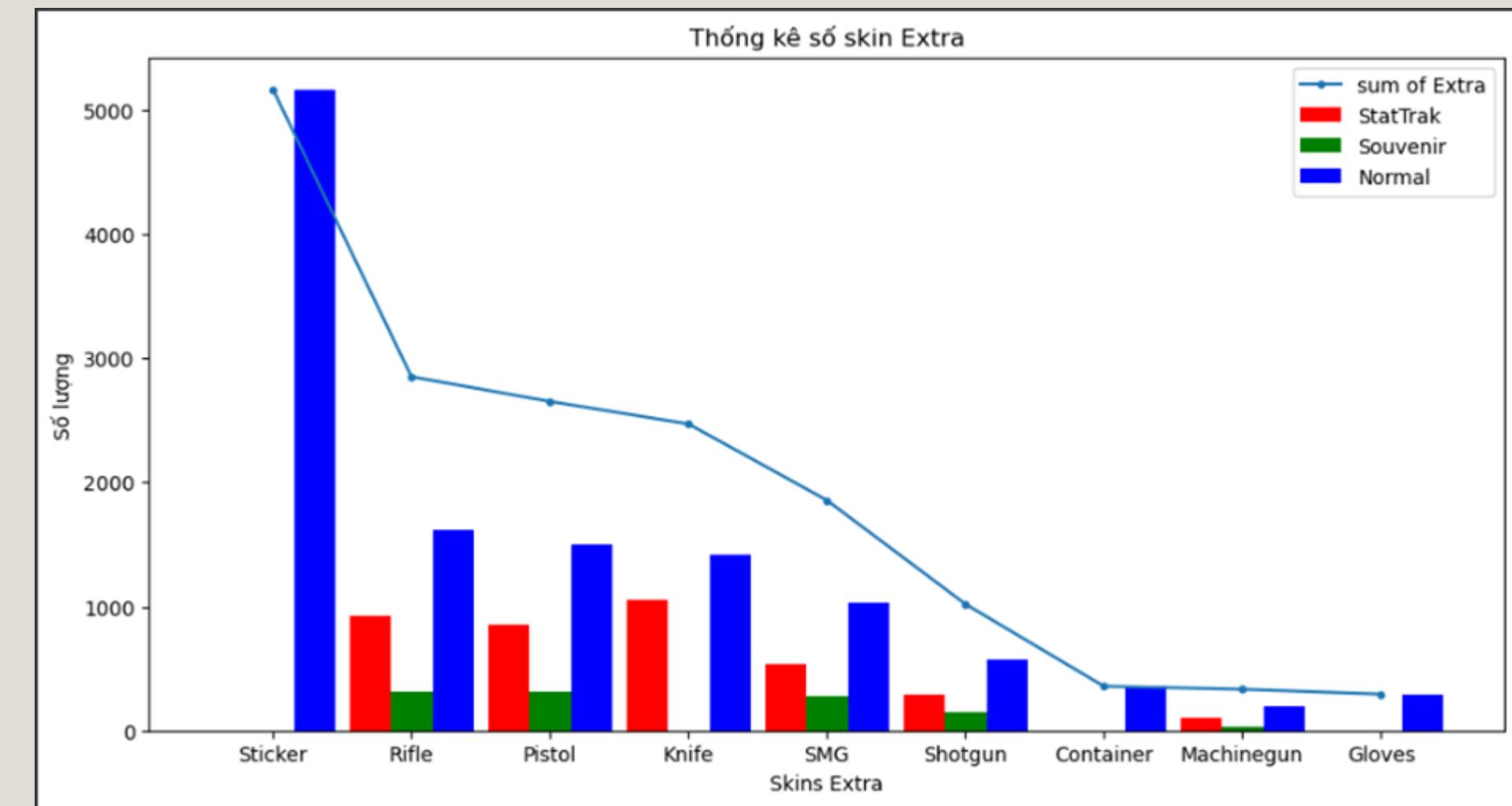
    if ('Normal' in list(temp.keys())):
        list_Normal.append(temp['Normal'])
    else:
        list_Normal.append(0)

X_axis = np.arange(len(X))

plt.bar(X_axis - 0.3, list_Stattrak, 0.3, label = 'StatTrak', color = 'r')
plt.bar(X_axis , list_Souvenir, 0.3, label = 'Souvenir', color = 'g')
plt.bar(X_axis + 0.3, list_Normal, 0.3, label = 'Normal', color = 'b')

plt.xticks(X_axis, X)
plt.xlabel("Skins Extra")
plt.ylabel("Số lượng")
plt.title("Thống kê số skin Extra")
plt.legend()
```

## 2.Type có skin Extra nào thì được bán nhiều nhất ?



Nhận xét :

Có thể thấy skin Normal xuất hiện nhiều trên Market , chiếm tỉ lệ cao nhất trong từng "Type"

Cho thấy skin Normal dễ kiếm hơn so với StatTrak và Souvenir . Trong đó Souvenir là ít xuất hiện nhất !

Skin Souvenir ít xuất hiện do chỉ có trong các event , vậy liệu nó có được người mua chú ý hay có hiếm trên thị trường không ?

### 3. Liệu skin Souvenir có nhận được sự quan tâm hơn so với các skin khác không ?!

Điều đó sẽ trả lời xem skin Souvenir có thực sự  
được ưa chuộng trong market

Thực hiện khá giống bên trên , tuy nhiên tham số thống kê là giá trị  
max của Type với "StatTrak", "Souvenir","Normal"

```
plt.rcParams["figure.figsize"] = [12, 6]

X = list(df_Type.keys())
list_Extra = ['Souvenir','StatTrak™','Normal']
list_Souvenir = []
list_Stattrak = []
list_Normal = []
list_max = {}
for i in X:
    a = df.loc[df['Type'] == i]
    list_max[i]=(a['Days to reach max'].mean())

    Extra = a.loc[a['Extra'] == 'Souvenir']
    list_Souvenir.append(Extra['Days to reach max'].mean())

    Extra1 = a.loc[a['Extra'] == 'StatTrak™']
    list_Stattrak.append(Extra1['Days to reach max'].mean())

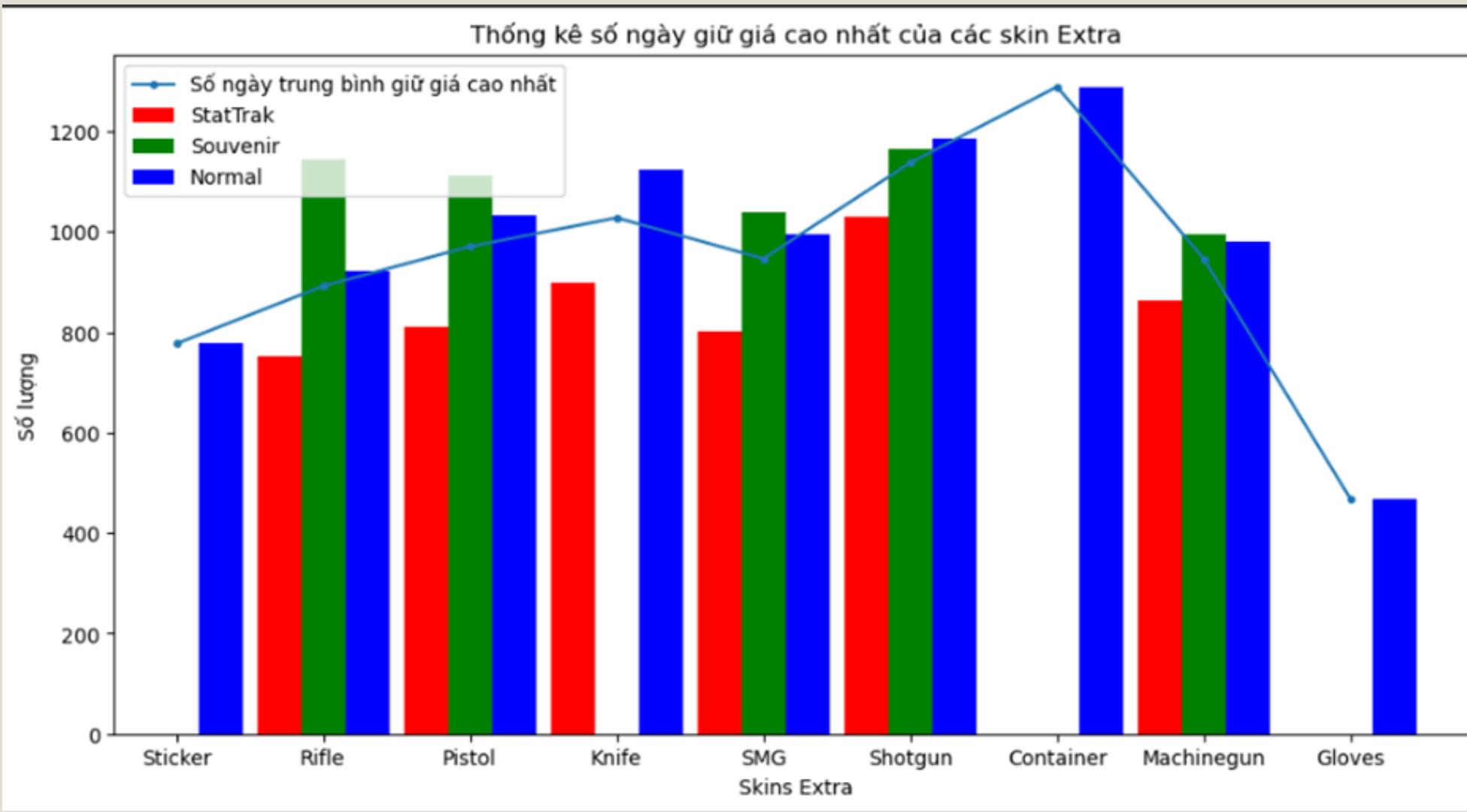
    Extra2 = a.loc[a['Extra'] == 'Normal']
    list_Normal.append(Extra2['Days to reach max'].mean())


plt.plot(list(list_max.keys()), (list_max.values()),marker ='.')
X_axis = np.arange(len(X))

plt.bar(X_axis - 0.3, list_Stattrak, 0.3, label = 'StatTrak',color = 'r')
plt.bar(X_axis , list_Souvenir, 0.3, label = 'Souvenir',color ='g')
plt.bar(X_axis + 0.3, list_Normal, 0.3, label = 'Normal',color = 'b')

plt.xticks(X_axis, X)
plt.xlabel("Skins Extra")
plt.ylabel("Số lượng")
plt.title("Thống kê số skin Extra")
plt.legend()
```

### 3. Liệu skin Souvenir có nhận được sự quan tâm hơn so với các skin khác không ?!

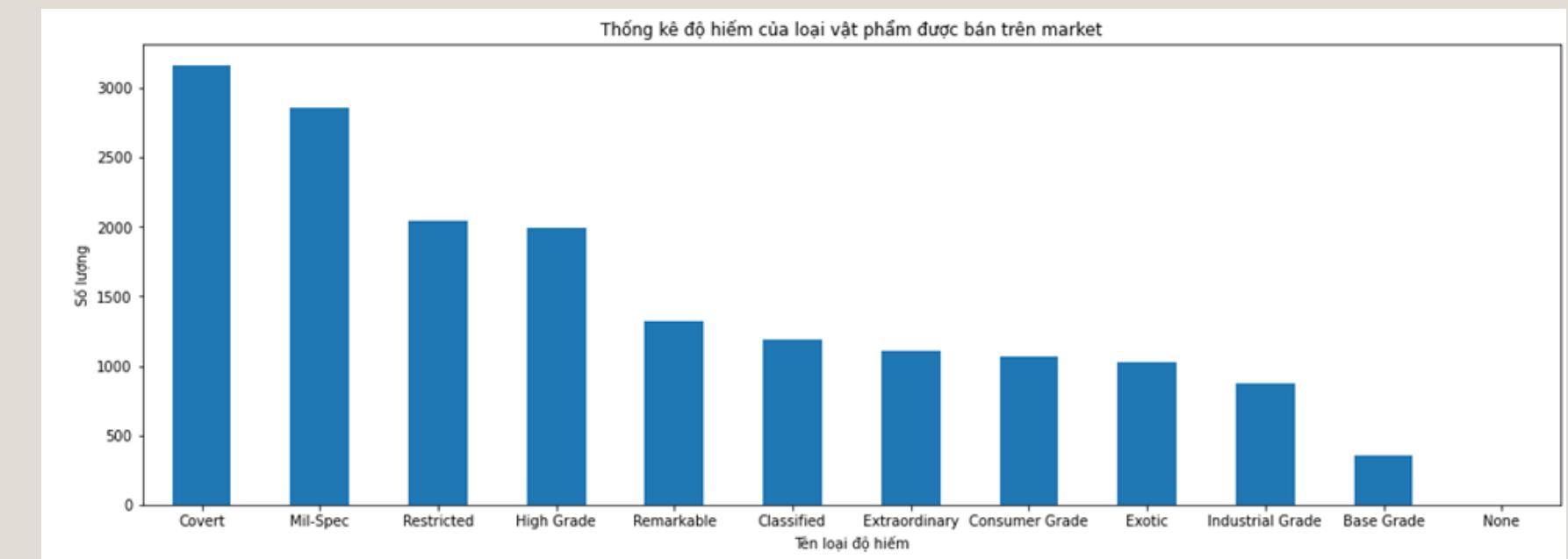


Nhận xét :

Giá của các vật phẩm Souvenir luôn cao và giữ được mức giá cao so với các Extra khác ngoài trừ Normal vì số lượng quá phổ biến  
Chứng tỏ các vật phẩm Souvenir được người dùng thích và là Extra hiếm có trong game  
Bên cạnh đó thì StatTrak™ khá thấp , thấp nhất trong các Extra , cho thấy nó khá kén người thích !

#### 4. Bây giờ vấn đề quan tâm của chúng ta là độ hiếm , có phải các vật phẩm xuất hiện trên market thì đa phần rất hiếm và rất khó tìm ?

```
df_Rarity = df['Rarity'].value_counts(dropna=False)# get dict _ of Type  
  
plt.rcParams["figure.figsize"] = [18, 6]  
ax = df_Rarity.plot.bar(rot=0)  
plt.xlabel('Tên loại độ hiếm')  
plt.ylabel('Số lượng')  
plt.title('Thống kê độ hiếm của loại vật phẩm được bán trên market')
```



Nhận xét :

Có thể thấy theo thống kê , các độ hiếm xuất hiện nhiều trên market là Covert , Mil-Spec , ... Bất ngờ là không có độ hiếm WeaPonS ( hiếm nhất ) xuất hiện trên market , có thể nó rất khó kiếm nên hầu như không có giao dịch nào hoặc có thể thị trường không ưa thích loại vật phẩm này .

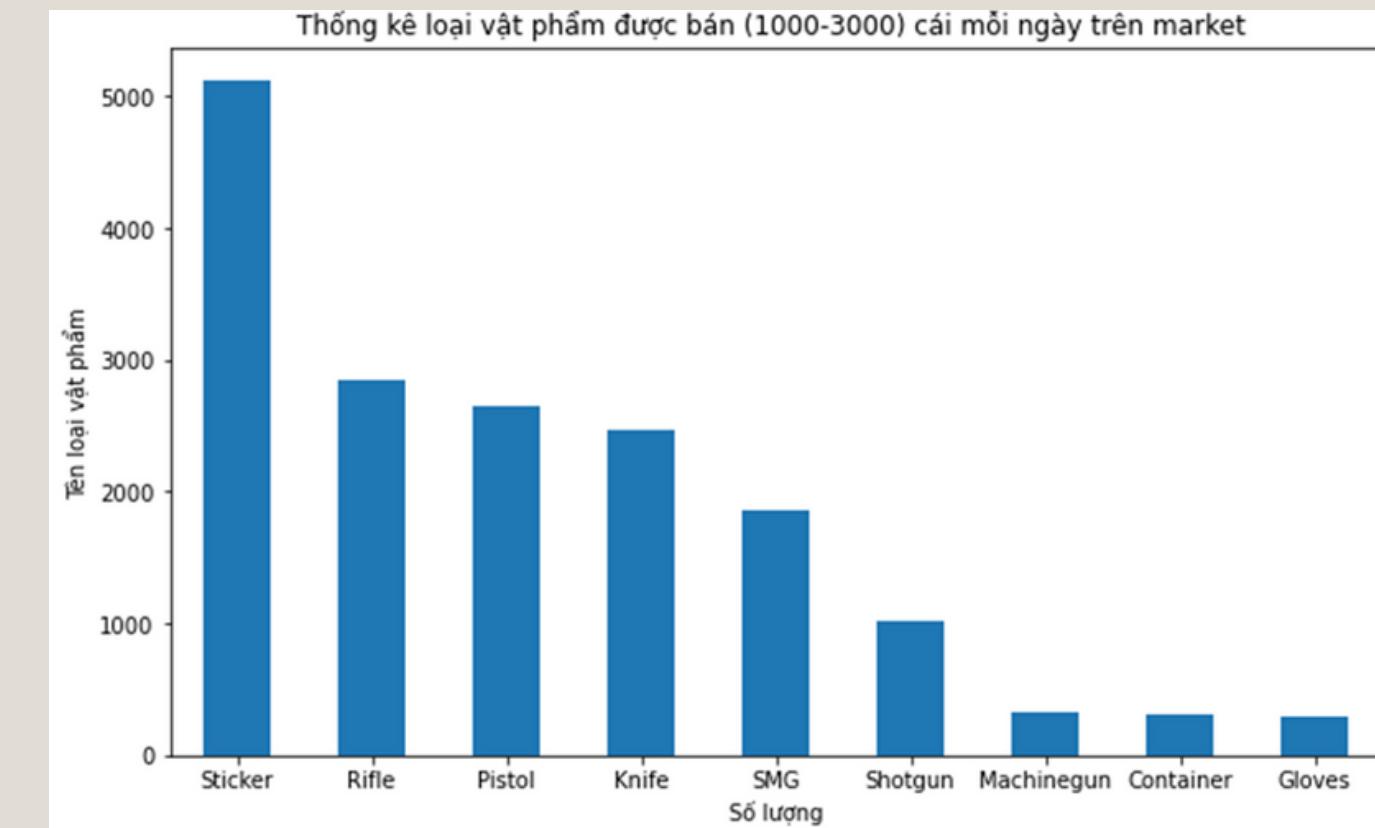
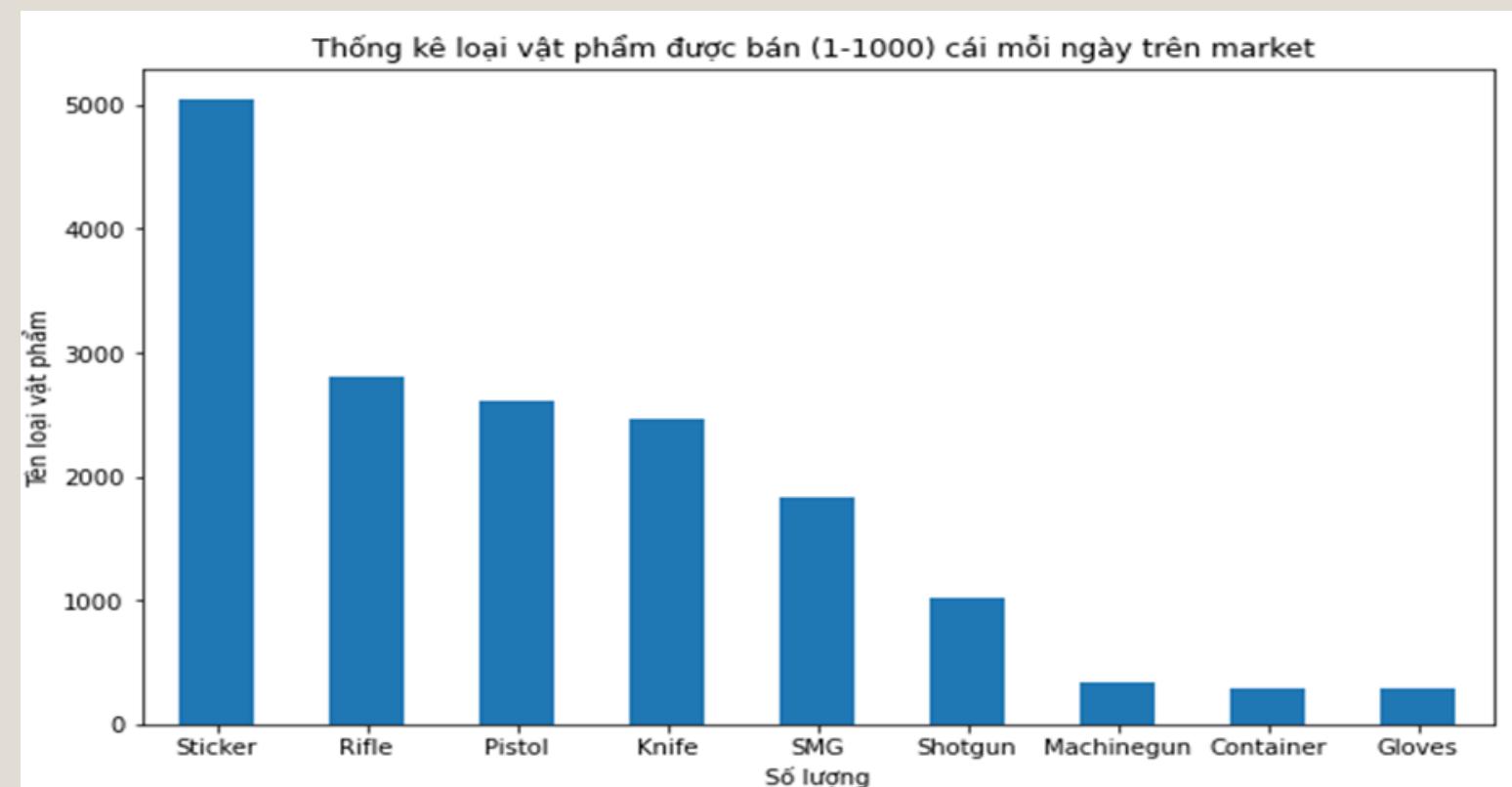
Loại vật phẩm có độ hiếm Covert (khá hiếm-red) được giao dịch nhiều nhất với số lượng hơn 3000 , Mil-Spec cũng được giao dịch khá nhiều trên market

Loại vật phẩm đại trà ( Base Grade) được giao dịch khá thấp dưới 500

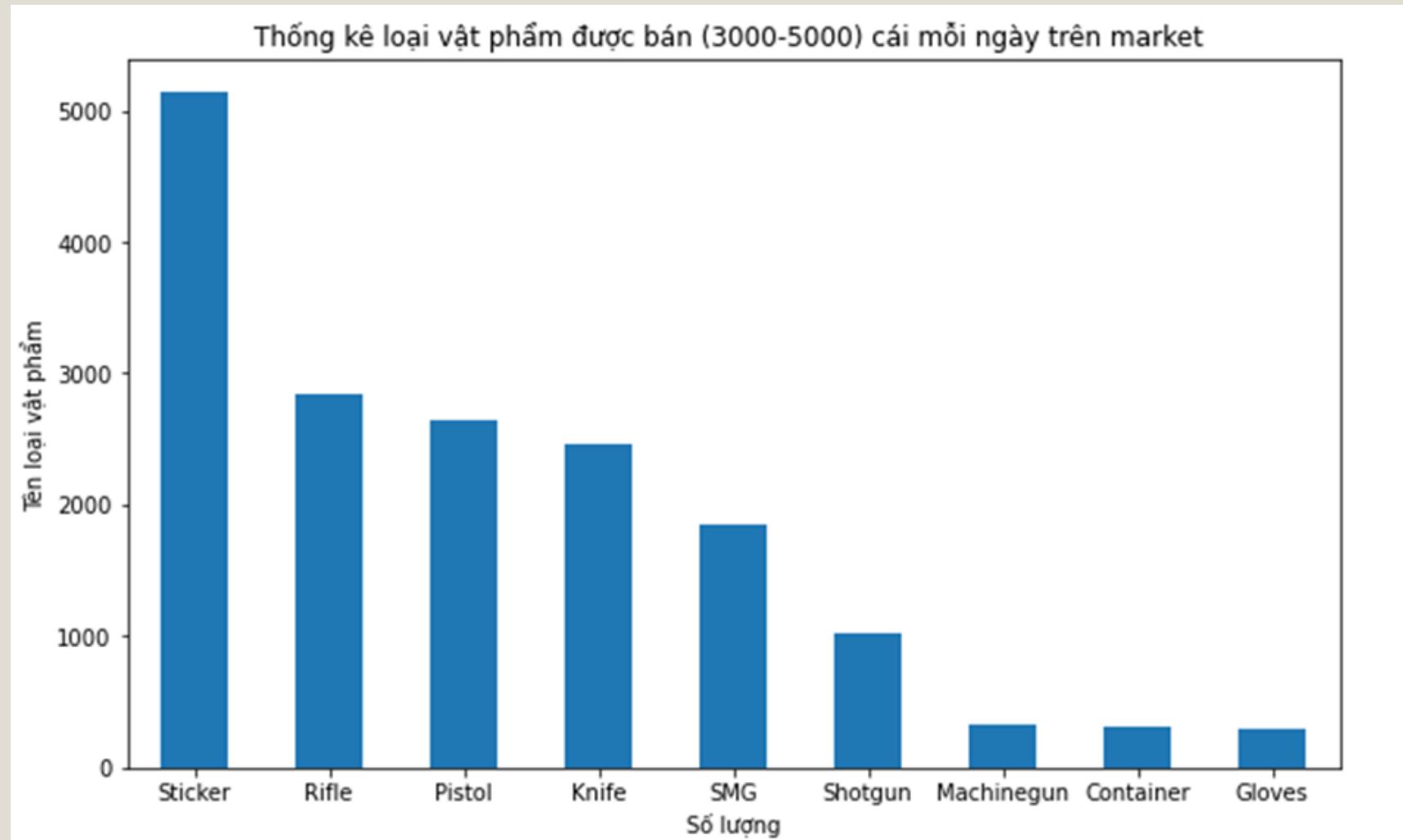
## 5. Tỉ lệ mua trong 1 ngày có phản ảnh đó là món đó "hot" không ?

Thống các skin bán được trong 1 ngày theo các mức thấp  
(1-1000) ,trung bình (1000-3000) , cao (>= 3000)

```
def volume (min_value,max_value):  
  
    df_volume = df.loc[df['Average volume sale']>= min_value]  
    df_volume = df.loc[df['Average volume sale']< max_value]  
  
    df_volume = df_volume['Type'].value_counts()  
    plt.rcParams["figure.figsize"] = [10, 6]  
    ax = df_volume.plot.bar(rot=0)  
    plt.ylabel('Tên loại vật phẩm')  
    plt.xlabel('Số lượng')  
    plt.title('Thống kê loại vật phẩm được bán (' + str(min_value) +'-' +str(max_value) +') cái mỗi ngày trên market ')
```



## 5. Tỉ lệ mua trong 1 ngày có phản ảnh đó là món đó "hot" không ?



Nhận xét :

Dữ liệu có xu hướng không thay đổi nhiều ở mức phân chia các giá trị theo mức bán ra hằng ngày Có thể thấy các skin được bán ra khá ổn định

Với lượt giao dịch nhiều nhất là Sticker -> Rifle -> Pistol -> Knife -> SMG -> Shotgun -> Machinegun -> Container -> Gloves

## 6.Trường hợp tôi có một skin , tôi cần tham khảo giá trên thị trường , thì tôi cần biết giá thấp nhất cao nhất trung bình của nó

Có được những thống kê về dữ liệu trên có thể dễ dàng đưa ra lựa chọn cho người khảo sát

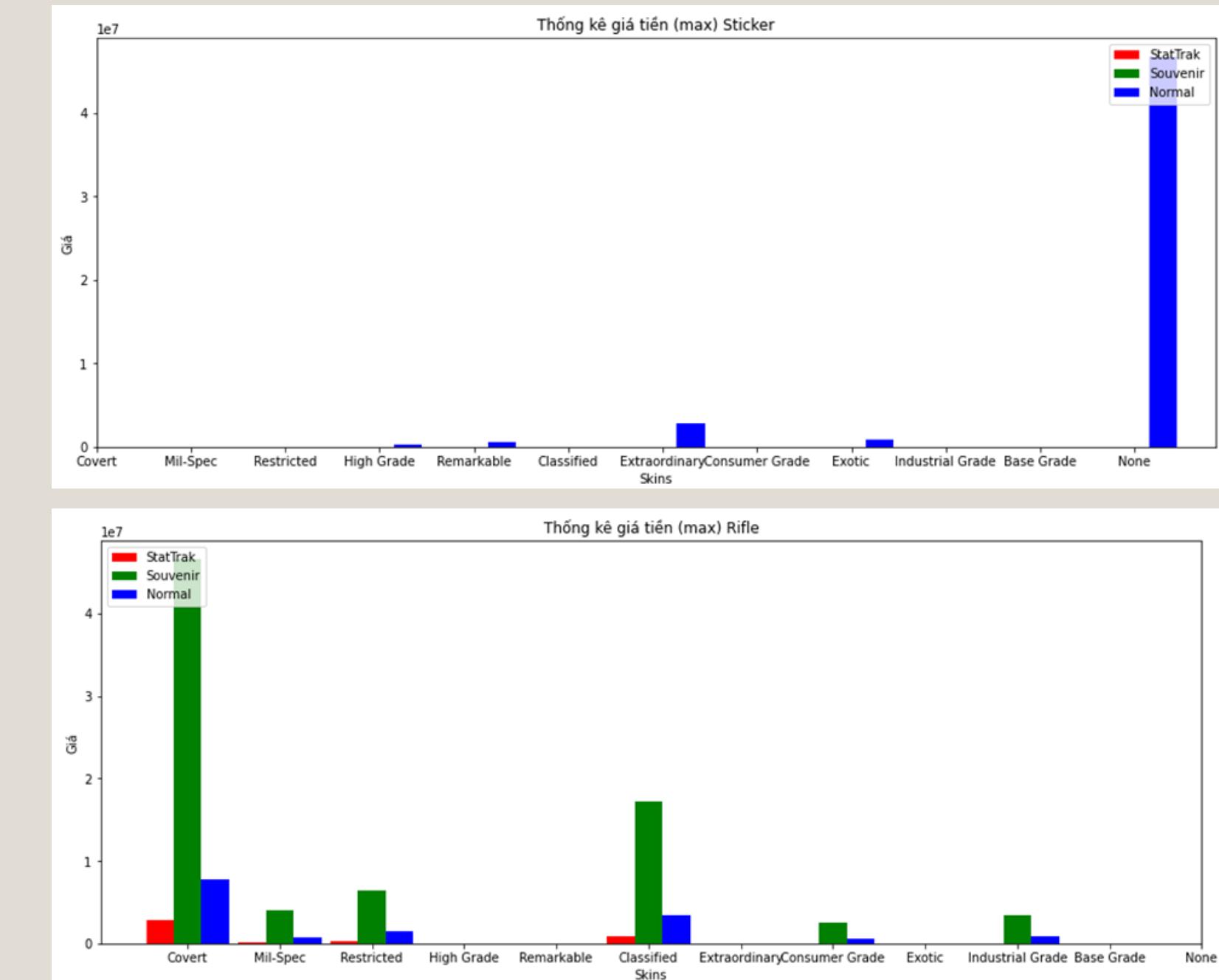
```
X = ['Covert','Mil-Spec','Restricted','High Grade','Remarkable','Classified','Extraordinary',
      'Consumer Grade','Exotic','Industrial Grade','Base Grade','None']
Y = ['Normal', 'StatTrak™', 'Souvenir']
def findskin(name_Type):

    plt.rcParams["figure.figsize"] = [16, 6]
    list_Souvenir = []
    list_Stattrak = []
    list_Normal = []
    df_skin = df.loc[df['Type'] == name_Type]

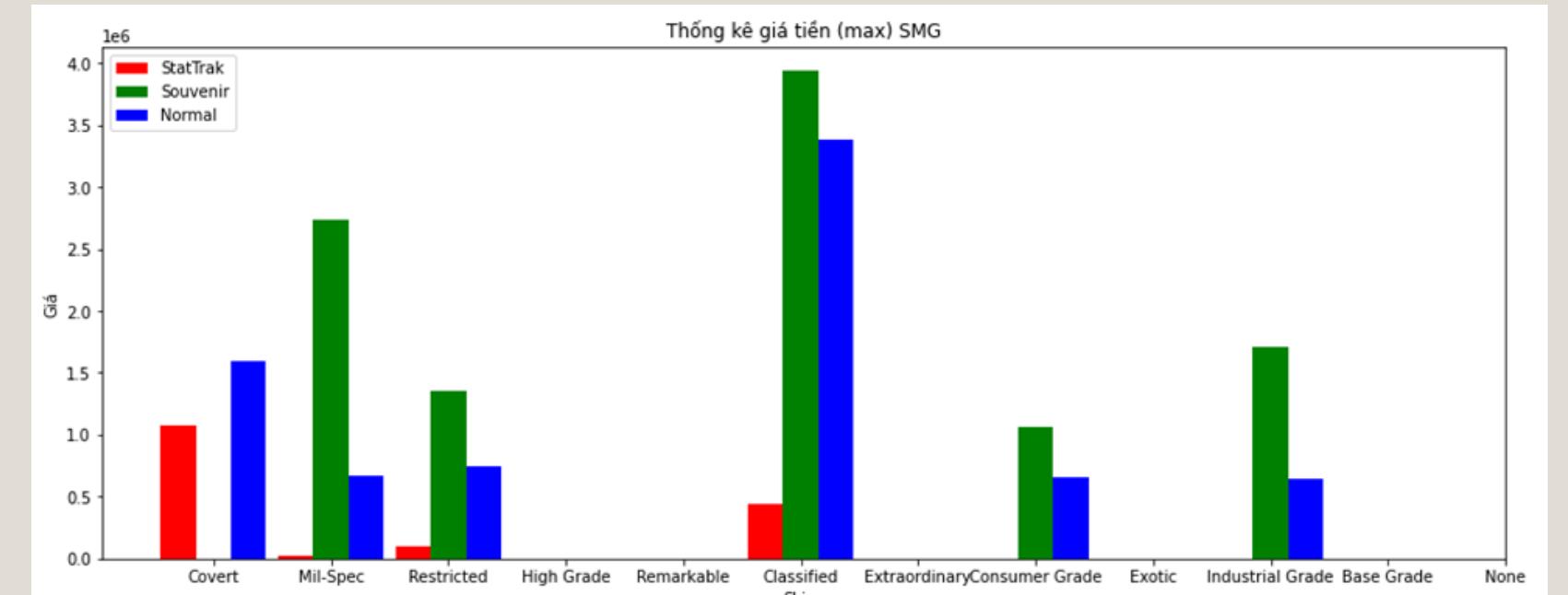
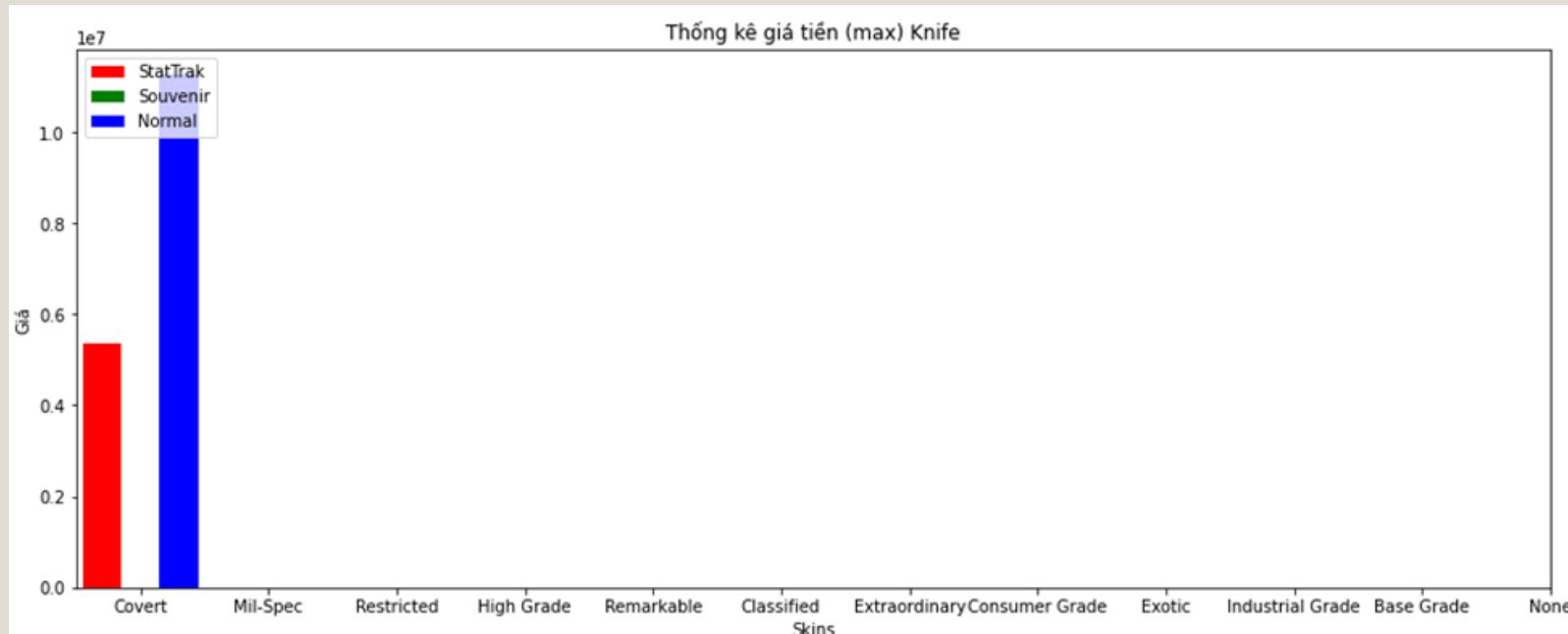
    for i in X:
        df_skin2 = df_skin.loc[df_skin['Rarity'] == i]
        for j in Y:
            if (j == 'Normal'):
                df_skin1 = df_skin2.loc[df_skin2['Extra'] == j]
                list_Normal.append(df_skin1['Max price (VND)'].mean())
            if (j == 'StatTrak™'):
                df_skin1 = df_skin2.loc[df_skin2['Extra'] == j]
                list_Stattrak.append(df_skin1['Average price (VND)'].mean())
            if (j == 'Souvenir'):
                df_skin1 = df_skin2.loc[df_skin2['Extra'] == j]
                list_Souvenir.append(df_skin1['Max price (VND)'].mean())

    X_axis = np.arange(len(X))
    plt.bar(X_axis - 0.3, list_Stattrak, 0.3, label = 'StatTrak',color = 'r')
    plt.bar(X_axis , list_Souvenir, 0.3, label = 'Souvenir',color ='g')
    plt.bar(X_axis + 0.3, list_Normal, 0.3, label = 'Normal',color = 'b')

    plt.xticks(X_axis, X)
    plt.xlabel("Skins")
    plt.ylabel("Giá")
    plt.title("Thống kê giá tiền (max) " + name_Type)
    plt.legend()
```



## 6.Trường hợp tôi có một skin , tôi cần tham khảo giá trên thị trường , thì tôi cần biết giá thấp nhất cao nhất trung bình của nó



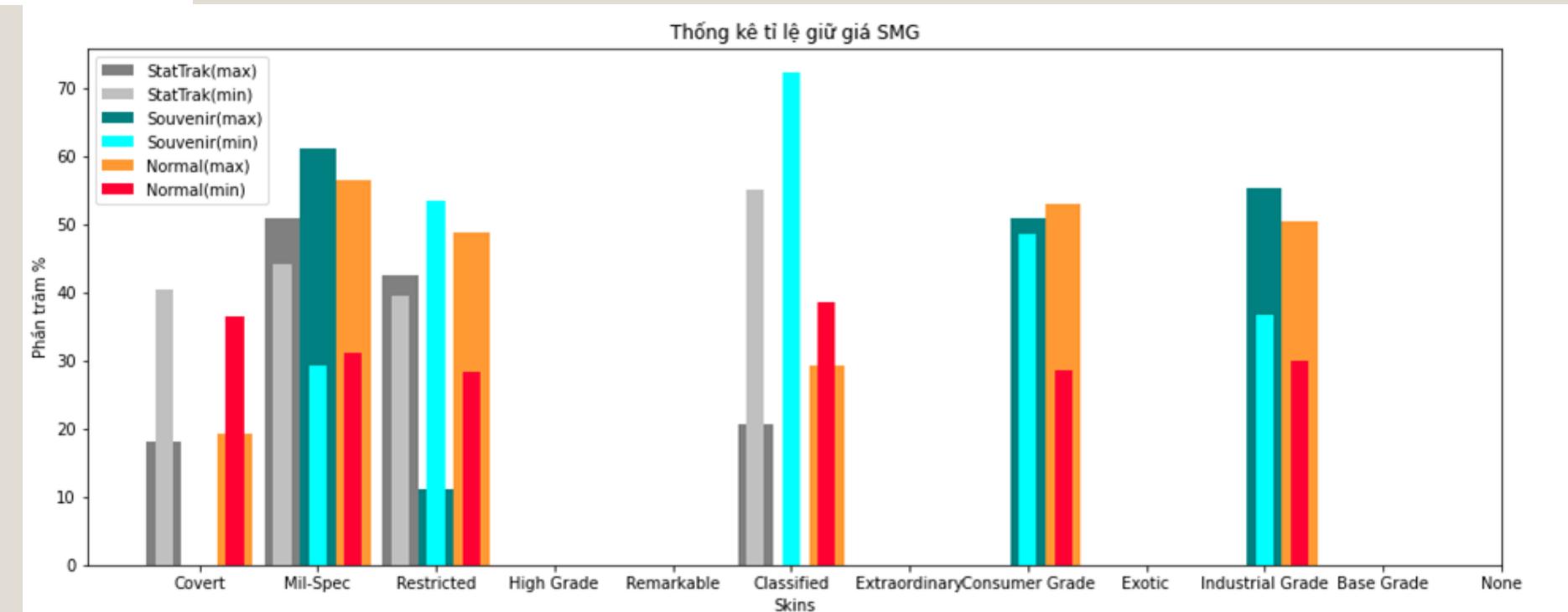
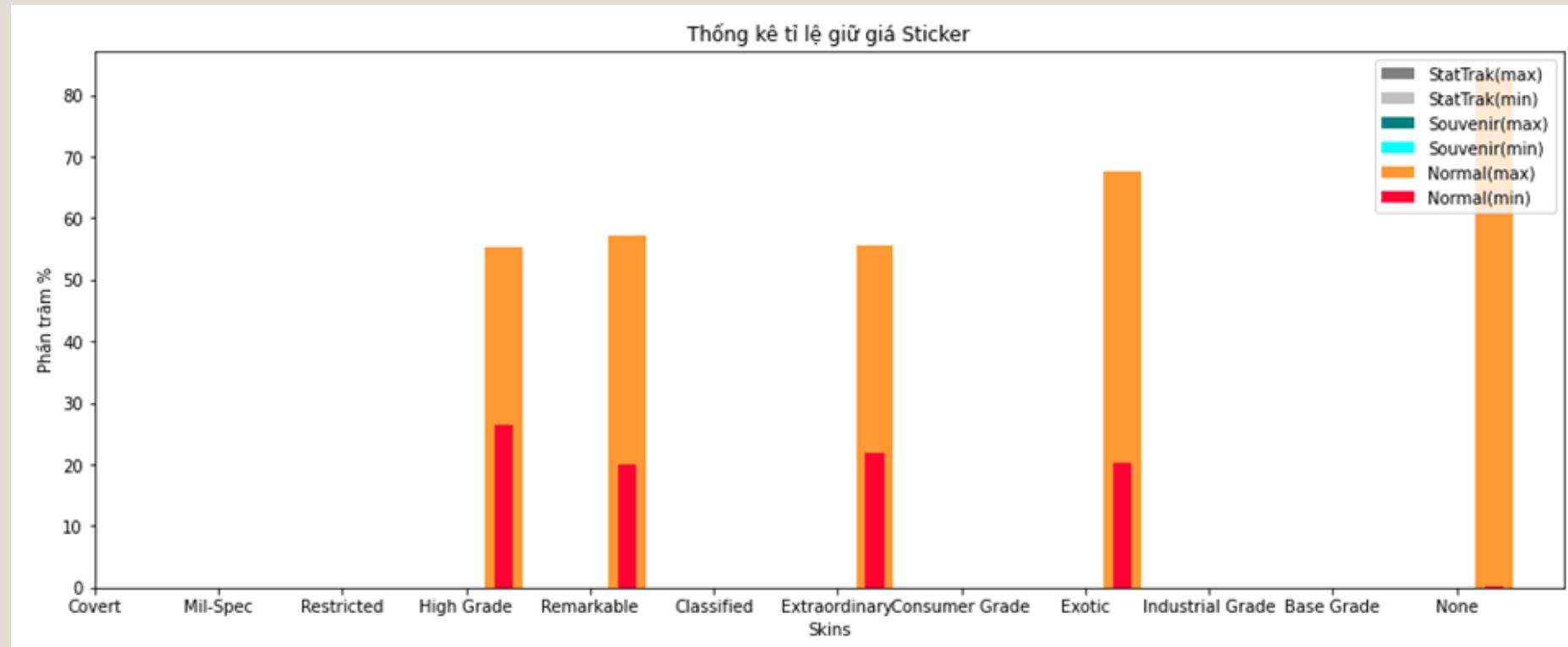
Nhận xét :

Như có thể thấy nhờ vào trực quan hóa có thể biết Type , Rarity và Extra đang muốn tham khảo giá bán trên market  
Từ đó có thể đưa ra giá thích hợp cho Skin định bán hoặc tránh tình trạng mua nhầm giá !

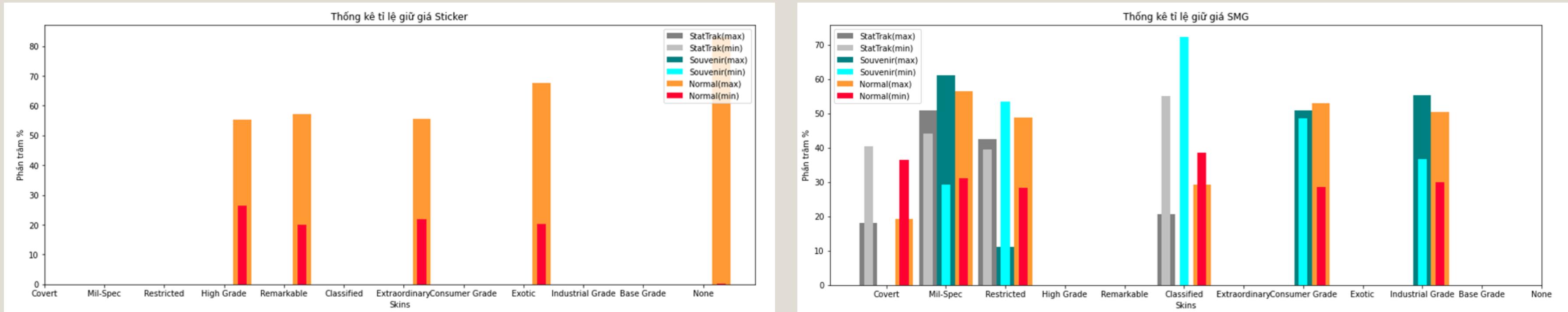
## 7. Làm sao tôi biết mình đang sở hữu một skin có giá trị như thế nào ?

Sẽ phải nhìn vào thị trường trên market ưa chuộng như thế nào ?

Một skin được ưa chuộng sẽ giữ được giá của nó lâu hơn ?



## 7. Làm sao tôi biết mình đang sở hữu một skin có giá trị như thế nào ?



Nhận xét :

Biểu đồ thống kê phần trăm (%) giá của các skin với các độ hiếm và Extra khác nhau

Tùy vào Type , Rarity , Extra sẽ có tỉ lệ phần trăm khác nhau

Biểu đồ thể hiện % Reach max và Reach min phản ảnh mức độ giữ giá của skin , qua đó thể hiện giá trị của skin đối với thị trường



3 trên 6

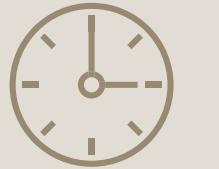
## Mô hình hóa dữ liệu



5 trên 6

# 1. Các thư viện sử dụng

---



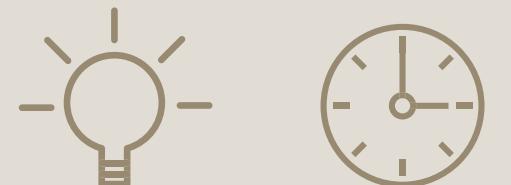
```
# Thư viện
import pandas as pd
from matplotlib import pyplot as plt
import warnings
from sklearn.linear_model import LinearRegression

warnings.filterwarnings('ignore')
✓ 0.1s
```

Python

## 2.Lựa chọn phương thức model data:

---



Đối với dataset này thì mục đích dễ thấy sẽ là dự đoán giá skin trong tương lai, nên nhóm quyết định sử dụng phương thức linear regression để model data, do phương thức này thường được sử dụng để dự đoán 1 biến phụ thuộc y dựa trên một hay nhiều biến phụ thuộc  $x_0, x_1, \dots$

# Lọc các vật phẩm theo từng loại

---

## Các vật phẩm thuộc loại Container

ID	Name	Type	Category	Rarity	Grade	Average volume (m³)	Average price (VNĐ)	Mean price (VNĐ)	Days until
116	ESL One Cologne 2014 Cobblestone Souvenir Package	Container	None	Base	Grade Normal	21.81162325	2843256.482	44631844	
335	Cologne 2016 Cobblestone Souvenir Package	Container	None	Base	Grade Normal	43.67946824	3635430.1	46116788	
396	EMS One 2014 Souvenir Package	Container	None	Base	Grade Normal	26.73472949	1990544.186	45921648	
405	ESL One Katowice 2015 Cobblestone Souvenir Package	Container	None	Base	Grade Normal	29.94109494	2592857.76	20741880	
419	DreamHack Cluj-Napoca 2015 Cobblestone Souvenir Package	Container	None	Base	Grade Normal	50.28034067	3046093.177	45142820	
424	Krakow 2017 Cobblestone Souvenir Package	Container	None	Base	Grade Normal	31.28313796	4607436.642	42592340	
449	MLG Columbus 2016 Cobblestone Souvenir Package	Container	None	Base	Grade Normal	26.58494405	2924097.742	22468634	
450	ESL One Cologne 2015 Cobblestone Souvenir Package	Container	None	Base	Grade Normal	50.16413994	3290374.682	44631788	
507	Boston 2018 Cobblestone Souvenir Package	Container	None	Base	Grade Normal	36.97135417	5525537.32	21450286	
519	Atlanta 2017 Cobblestone Souvenir Package	Container	None	Base	Grade Normal	47.06033058	4181406.268	47209024	
816	ESL One Katowice 2015 Challengers (Holo-Foil)	Container	None	Base	Grade Normal	26.43802537	3164272.929	24527790	
851	DreamHack 2014 Overpass Souvenir Package	Container	None	Base	Grade Normal	12.7962963	1030628.445	16071547	
967	DreamHack 2013 Souvenir Package	Container	None	Base	Grade Normal	21.81771281	1202198.805	10955368	
1123	ESL One Cologne 2014 Cache Souvenir Package	Container	None	Base	Grade Normal	11.42748092	565872.687	6291346.5	
1387	DreamHack 2014 Nuke Souvenir Package	Container	None	Base	Grade Normal	11.15804598	760195.3331	17378804	
1499	ESL One Cologne 2014 Overpass Souvenir Package	Container	None	Base	Grade Normal	10.34677419	928256.1921	5510042	
1500	Boston 2018 Minor Challengers Autograph Capsule	Container	None	Base	Grade Normal	13.62980769	914596.1767	5972288.5	
1731	Boston 2018 Legends (Holo-Foil)	Container	None	Base	Grade Normal	3.354312354	1150783.427	5391740	
1912	DreamHack 2014 Dust II Souvenir Package	Container	None	Base	Grade Normal	11.6247505	382910.8207	6940689.5	
1926	ESL One Katowice 2015 Legends (Holo-Foil)	Container	None	Base	Grade Normal	36.31473409	1527950.853	9623754	
1971	ESL One Katowice 2015 Overpass Souvenir Package	Container	None	Base	Grade Normal	18.30872483	595737.9441	6719518	
2048	ESL One Cologne 2014 Dust II Souvenir Package	Container	None	Base	Grade Normal	15.81223629	321522.7924	5453704.5	
2133	DreamHack 2014 Mirage Souvenir Package	Container	None	Base	Grade Normal	11.76294821	343935.9089	4711133	

# Lọc các vật phẩm theo từng loại

## Các vật phẩm thuộc loại Gloves

	A	B	C	D	E	F	G	H
1	Name	Type	Exterior	Rarity	Extra	Average volume sa	Average price (VN)	Max price (VN)
6	À™_ Specialist Gloves   Emerald Web (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.130434783	21938172.68	46029248
26	À™_ Moto Gloves   Spearmint (Well-Worn)	Gloves	Well-Worn	Extraordinary	Normal	1.145454545	12166266.35	37357056
27	À™_ Sport Gloves   Vice (Field-Tested)	Gloves	Field-Tested	Extraordinary	Normal	1.576709797	31272973.63	47962064
29	À™_ Driver Gloves   Imperial Plaid (Factory New)	Gloves	Factory New	Extraordinary	Normal	1	38383019	47991816
32	À™_ Specialist Gloves   Mogul (Factory New)	Gloves	Factory New	Extraordinary	Normal	1	38562630.93	49305480
54	À™_ Moto Gloves   Spearmint (Field-Tested)	Gloves	Field-Tested	Extraordinary	Normal	1.714644351	17793173.71	47688320
55	À™_ Specialist Gloves   Foundation (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.211505922	14568917.44	44149236
68	À™_ Sport Gloves   Pandora's Box (Battle-Scarred)	Gloves	Battle-Scarred	Extraordinary	Normal	1.271144279	15486954.95	41052072
80	À™_ Driver Gloves   Snow Leopard (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.238888889	42458037.81	49803328
115	À™_ Sport Gloves   Vice (Well-Worn)	Gloves	Well-Worn	Extraordinary	Normal	1.10031348	25081495.21	40614432
121	À™_ Sport Gloves   Omega (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.1806998152	25078836.59	45749664
125	À™_ Specialist Gloves   Fade (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.141280353	32839189.36	48187068
127	À™_ Moto Gloves   Polygon (Factory New)	Gloves	Factory New	Extraordinary	Normal	1	27043617.64	37020828
134	À™_ Driver Gloves   Crimson Weave (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.161425577	19172689.38	33954980
138	À™_ Specialist Gloves   Marble Fade (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.166666667	38653957.27	47884204
146	À™_ Specialist Gloves   Crimson Kimono (Field-Tested)	Gloves	Field-Tested	Extraordinary	Normal	1.763565891	18378320.4	44631868
156	À™_ Sport Gloves   Arid (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.319371728	11606447.84	30810492
191	À™_ Hand Wraps   Cobalt Skulls (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.169336384	25524573.65	40148772
197	À™_ Hand Wraps   Leather (Factory New)	Gloves	Factory New	Extraordinary	Normal	1.05	16420570.65	26414832
222	À™_ Moto Gloves   POW! (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.155902004	24163704.84	40975688
224	À™_ Moto Gloves   3rd Commando Company (Factory New)	Gloves	Factory New	Extraordinary	Normal	1	24033977	47680972
227	À™_ Broken Fang Gloves   Yellow-banded (Factory New)	Gloves	Factory New	Extraordinary	Normal	1	22870591.82	26886210
235	À™_ Bloodhound Gloves   Bronzed (Factory New)	Gloves	Factory New	Extraordinary	Normal	1	22651214.4	29237192

## -Một số đặc trưng của từng loại vật phẩm:

- Container: vật phẩm thuộc nhóm này đều có Exterior, Extra và Rarity giống nhau, và chỉ được phân biệt bằng sự kiện có liên quan tới vật phẩm đó.
- Knife: vật phẩm thuộc nhóm này đều có Rarity là Covert, nhưng Extra và Exterior vẫn được phân bậc như bình thường.
- Gloves: vật phẩm thuộc nhóm này đều có Rarity cũng như Extra lần lượt là Extraordinary và Normal, riêng Exterior vẫn được phân bậc như bình thường.
- Sticker: vật phẩm thuộc nhóm này đều có Extra cũng như Exterior lần lượt là Normal và None. Tuy nhiên, Rarity của vật phẩm loại này sử dụng hệ thống phân bậc riêng với độ hiếm tăng dần như sau: High Grade < Remarkable < Exotic < Extraordinary, riêng sticker Howling Dawn có độ hiếm None.
- Các loại súng nói chung (Machinegun, Rifle, Shotgun, SMG, Pistol): các skin súng sử dụng hệ thống phân bậc Rarity, Exterior và Extra chung của CS:GO.

## Kết luận

---

- Đối với vật phẩm thuộc loại Container, nhóm sẽ không áp dụng linear regression, do giá bán của vật phẩm phụ thuộc phần lớn vào sự kiện có liên quan tới nó (dẫn tới việc container này chứa những vật phẩm nào, có giá trị ra sao).
- Đối với các loại vật phẩm có Rarity/Extra/Exterior giống nhau thì nhóm sẽ sử dụng một model riêng cho từng loại, do dùng chung sẽ ảnh hưởng tới độ chính xác của model.
- Các loại vật phẩm còn lại sẽ sử dụng một model chung.

## **Số hóa các cột ordinal data**

---

- Mục đích của việc số hóa để đưa vào linear regression model
- Ta có thể thấy cột Rarity, Exterior và Extra đều thuộc dạng ordinal data, tức dữ liệu phân loại có thứ tự. Tuy nhiên, cách dễ nhất để số hóa data dạng này sẽ là số hóa chúng theo kiểu 1, 2, 3, 4, 5... dựa trên mức độ của chúng, nhưng chúng ta không thể chắc chắn rằng khoảng cách giữa mỗi giá trị thuộc cột có kiểu ordinal data là bằng nhau được. Vì vậy, trong trường hợp này, nhóm đề xuất việc sử dụng mean của average price để xác định “khoảng cách” giữa từng giá trị trong mỗi cột ordinal data một cách chính xác hơn

## Số hóa các cột ordinal data

Ví dụ: (số hóa cột Exterior cho các vật phẩm thuộc dạng Glove):

```
print('Avg price by exterior type: ', '\n')
for item in gloves['Exterior'].unique():
    print(item, ":", gloves[gloves['Exterior'] == item]['Average price (VND)'].mean())
print(' ')
✓ 0.1s
```

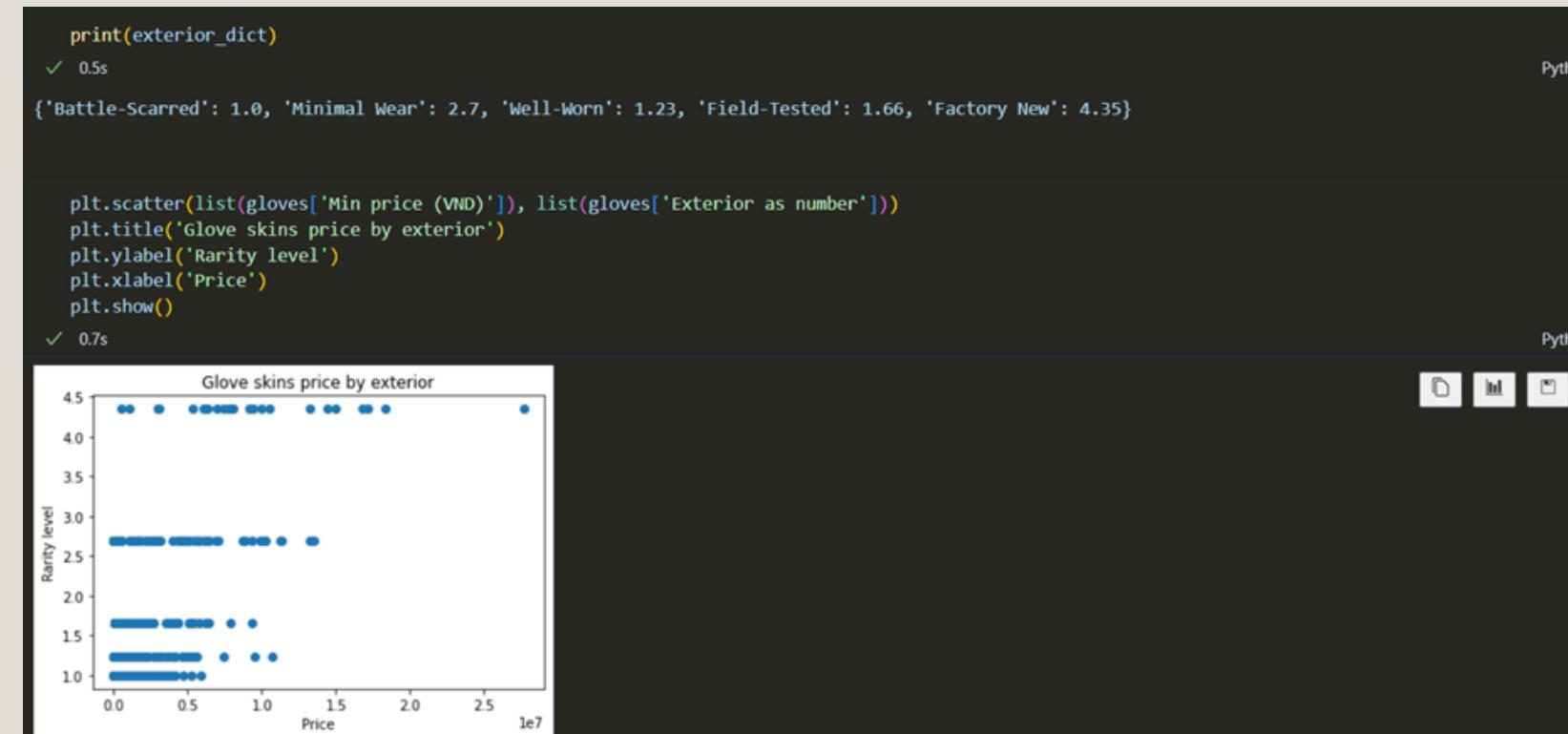
Avg price by exterior type:

Minimal Wear : 12699671.111338714  
Well-Worn : 5796134.967970149  
Field-Tested : 7789873.190521739  
Factory New : 20442106.835999995  
Battle-Scarred : 4699719.490577466

```
exterior_dict = {}
exterior_dict['Battle-Scarred'] = 1 # since the average price for battle-scarred gloves are lowest, we will enumerate it as 1.
for item in gloves['Exterior'].unique():
    exterior_dict[item] = round((gloves[gloves['Exterior'] == item]['Average price (VND)'].mean() /
    gloves[gloves['Exterior'] == 'Battle-Scarred']['Average price (VND)'].mean()), 2)
exterior_dict
exterior_num = []
for item in list(gloves['Exterior']):
    exterior_num.append(exterior_dict[item])
gloves['Exterior as number'] = exterior_num
gloves.head()
✓ 0.4s
```

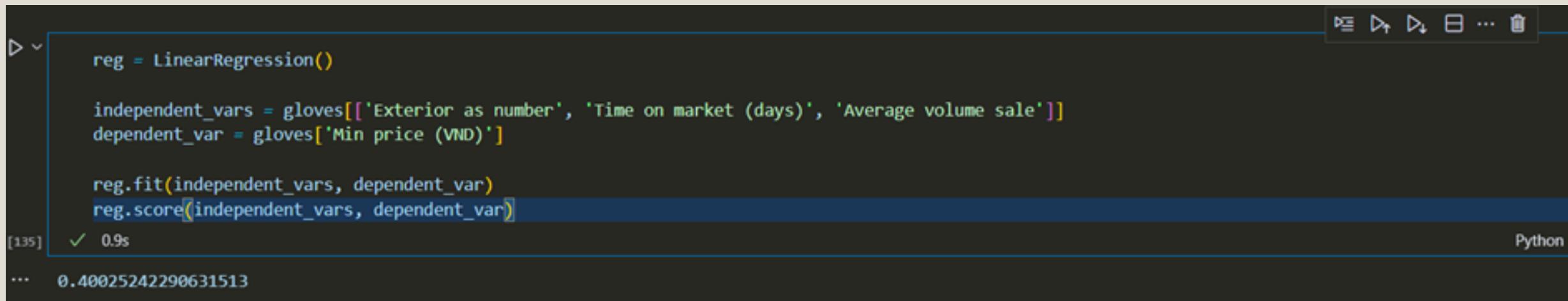
# Số hóa các cột ordinal data

## Kết quả số hóa cột Exterior



	Name	Type	Exterior	Rarity	Extra	Average volume sale	Average price (VND)	Max price (VND)	Days to reach max	Min price (VND)	Days to reach min	Price change (%)	Time on market (days)	Exterior as number
4	★ Specialist Gloves   Emerald Web (Minimal Wear)	Gloves	Minimal Wear	Extraordinary	Normal	1.130435	21938172.68	46029248.0	2192	7.007052e+06	453	3.71	2192	2.70
24	★ Moto Gloves   Spearmint (Well-Worn)	Gloves	Well-Worn	Extraordinary	Normal	1.145455	12166266.35	37357056.0	2111	1.012486e+04	698	4.91	2182	1.23
25	★ Sport Gloves   Vice (Field-Tested)	Gloves	Field-Tested	Extraordinary	Normal	1.576710	31272973.63	47962064.0	1629	4.114059e+06	1556	0.29	1744	1.66
27	★ Driver Gloves   Imperial Plaid (Factory New)	Gloves	Factory New	Extraordinary	Normal	1.000000	38383019.00	47991816.0	1434	6.293330e+06	0	6.09	1539	4.35
30	★ Specialist Gloves   Mogul (Factory New)	Gloves	Factory New	Extraordinary	Normal	1.000000	38562630.93	49305480.0	0	2.766229e+07	1643	-0.44	1643	4.35

## Mô hình hóa dữ liệu



```
reg = LinearRegression()

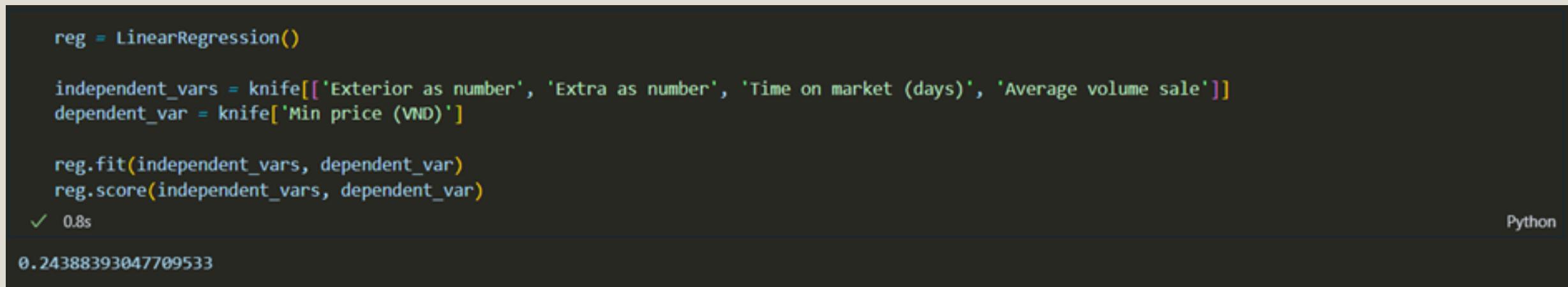
independent_vars = gloves[['Exterior as number', 'Time on market (days)', 'Average volume sale']]
dependent_var = gloves['Min price (VND)']

reg.fit(independent_vars, dependent_var)
reg.score(independent_vars, dependent_var)

[135] ✓ 0.9s
...
... 0.40025242290631513
```

The screenshot shows a Jupyter Notebook cell with Python code. The code imports the LinearRegression class from scikit-learn, defines independent variables (Exterior as number, Time on market (days), Average volume sale) and a dependent variable (Min price (VND)) from a 'gloves' dataset. It then fits the model and calculates the score. The output shows a score of 0.9s and a detailed output of 0.40025242290631513.

## Mô hình hóa dữ liệu cho các vật phẩm thuộc dạng Gloves



```
reg = LinearRegression()

independent_vars = knife[['Exterior as number', 'Extra as number', 'Time on market (days)', 'Average volume sale']]
dependent_var = knife['Min price (VND)']

reg.fit(independent_vars, dependent_var)
reg.score(independent_vars, dependent_var)

✓ 0.8s
0.24388393047709533
```

The screenshot shows a Jupyter Notebook cell with Python code. The code imports the LinearRegression class from scikit-learn, defines independent variables (Exterior as number, Extra as number, Time on market (days), Average volume sale) and a dependent variable (Min price (VND)) from a 'knife' dataset. It then fits the model and calculates the score. The output shows a score of 0.8s and a detailed output of 0.24388393047709533.

# Mô hình hóa dữ liệu

## Mô hình hóa dữ liệu cho các vật phẩm thuộc dạng Knife

```
reg = LinearRegression()  
  
independent_vars = sticker[['Rarity as number', 'Time on market (days)', 'Average volume sale']]  
dependent_var = sticker['Min price (VND)']  
  
reg.fit(independent_vars, dependent_var)  
reg.score(independent_vars, dependent_var)  
✓ 0.1s  
0.24231556546953625
```

Python

## Mô hình hóa dữ liệu cho các vật phẩm thuộc dạng Sticker

```
reg = LinearRegression()  
  
independent_vars = guns[['Exterior as number', 'Extra as number', 'Type as number', 'Time on market (days)', 'Average volume sale']]  
dependent_var = guns['Min price (VND)']  
  
reg.fit(independent_vars, dependent_var)  
reg.score(independent_vars, dependent_var)  
✓ 0.1s  
0.03535414779513246
```

Python

# Tổng hợp kết quả:

## a. Đánh giá kết quả:

Nhìn chung thì giá skin có vẻ không có mối quan hệ tuyến tính với các biến độc lập như Rarity, Extra, Exterior, Time on Market và Average volume sale như nhóm dự đoán mà còn phụ thuộc vào nhiều yếu tố khác, dẫn tới kết quả model không như kỳ vọng.

## b. Khó khăn và kinh nghiệm rút ra:

Khó khăn gặp phải: tìm ra đặc trưng của dữ liệu, số hóa các cột ordinal data và việc model chạy không đúng như kỳ vọng.

Những điều đã học được: cách sử dụng Python để tạo ra một model linear regression cơ bản (chia tập train/test, multiple linear regression...), trực quan hóa dữ liệu (vẽ biểu đồ 3D)...

Những gì sẽ làm nếu có thêm thời gian: tìm một hướng đi khác để dự đoán giá skin