

DỰ ÁN:



# DỰ BÁO DOANH SỐ BÁN HÀNG (PREDICT FUTURE SALES)

# NỘI DUNG

## PHẦN 1: PHÂN TÍCH DỮ LIỆU VÀ KHÁM PHÁ (DATA ANALYSIS EDA)

- Tóm tắt tình hình doanh nghiệp và tầm quan trọng của công tác dự báo doanh số.
- Trực quan hóa các dữ liệu thành các biểu đồ và rút ra nhận xét

## PHẦN 2: LẬP VÀ ĐÁNH GIÁ MÔ HÌNH (MODELING)

- Dữ liệu **dạng chuỗi thời gian**. Yêu cầu tạo các mô hình để dự đoán doanh số tháng tiếp theo của **từng mặt hàng tại từng cửa hàng** nằm trong chuỗi 60 đại lý trực thuộc công ty.
- Chọn một số mô hình có kết quả tốt và thời gian dự đoán hợp lý để tham dự cuộc thi trên trang chủ Kaggle
- Để thống nhất kết quả giữa các mô hình và cuộc thi, ta lựa chọn thang đo RMSE, mục tiêu đề ra là **RMSE < 1.5**

## PHẦN 3: CẢI THIỆN MÔ HÌNH – CÁC CÔNG VIỆC KHÁC (FUTURE WORK)

# PHẦN 1: PHÂN TÍCH DỮ LIỆU VÀ KHÁM PHÁ

# 1. HIẾU VỀ NGHIỆP VỤ (BUSINESS UNDERSTANDING):

## ❖ **Tổng quan về doanh nghiệp**

- Công ty 1C là một công ty phần mềm lớn của Nga được thành lập vào năm 1991
- Là nhà phân phối chính thức của Microsoft, Novell, Borland, Symantec, ABBYY, Kaspersky Lab, ProMT, Eset Software ... cho thị trường Nga.

## ❖ **Tầm quan trọng của công tác dự báo doanh số**

- Dự báo doanh số là công việc dự đoán số lượng sản phẩm có khả năng bán được trong một khoảng thời gian tương lai.
- Trong kinh doanh, dự báo doanh số bán hàng giúp cho doanh nghiệp sản xuất đúng số lượng sản phẩm cần thiết vào đúng thời điểm, tạo được lợi thế trước các đối thủ cạnh tranh.

## ❖ **Đối với doanh nghiệp 1C nói riêng, việc dự báo doanh số giúp công ty xác định được đâu là thế mạnh của mình, định hướng marketing, tập trung nguồn lực vào sản phẩm chính, giảm chi phí sản xuất hàng hóa doanh thu kém, phí lưu kho và các chi phí khác.**

## 2. HIỂU VỀ DỮ LIỆU (DATA UNDERSTANDING):

- ❖ **Nguồn dữ liệu:** Là lịch sử bán hàng từ **T1/2013** đến **T10/2015**. Dữ liệu gồm các thuộc tính:
  - **date** - Ngày tháng với định dạng dd/mm/yyyy
  - **date\_block\_num** - Số tháng liên tiếp, bắt đầu từ tháng 1/2013 là 0, tháng 2/2013 là 1...
  - **shop\_id** - Mã định danh của cửa hàng
  - **item\_id** - Mã định danh của sản phẩm
  - **item\_price** - Giá bán của sản phẩm
  - **item\_cnt\_day** - Số lượng sản phẩm đã bán
- ❖ **Ngoài ra còn một số thông tin bổ sung:**
  - **item\_category\_id** - Mã định danh của loại mặt hàng
  - **item\_name** - Tên mặt hàng
  - **shop\_name** - Tên cửa hàng, vị trí cửa hàng
  - **item\_category\_name** - Tên của loại mặt hàng

### 3. KHÁM PHÁ DỮ LIỆU:

Dữ liệu khá chi tiết với gần 3 triệu bản ghi

```
# Xem thông tin của bảng dữ liệu  
sales.info()  
<class 'pandas.core.frame.DataFrame'> RangeIndex:  
2935849 entries, 0 to 2935848
```

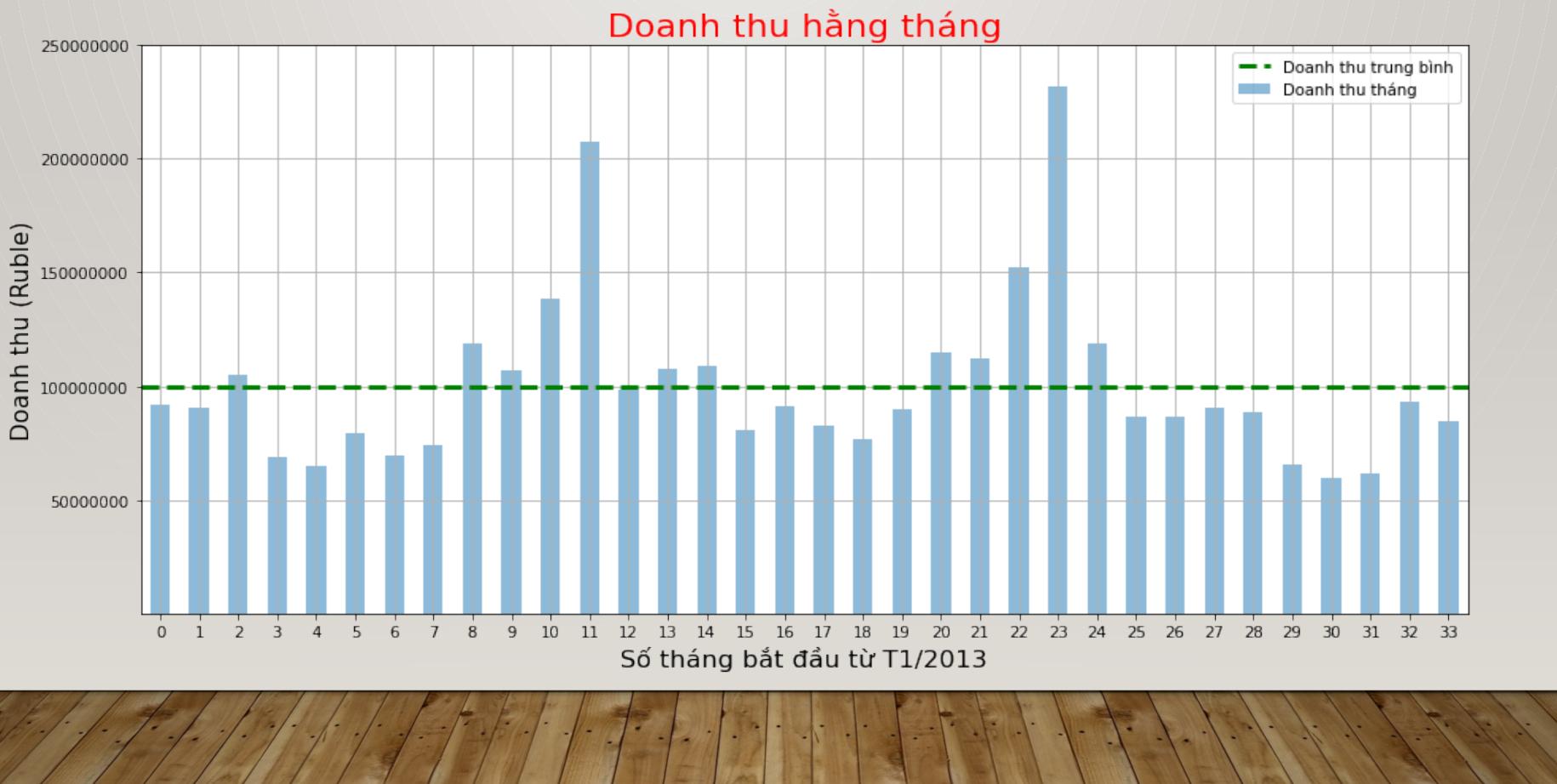
Trong tập dữ liệu có 60 cửa hàng

```
# Xem các thông tin về cửa hàng  
shops.info()  
<class 'pandas.core.frame.DataFrame'> RangeIndex: 60 entries, 0 to 59
```

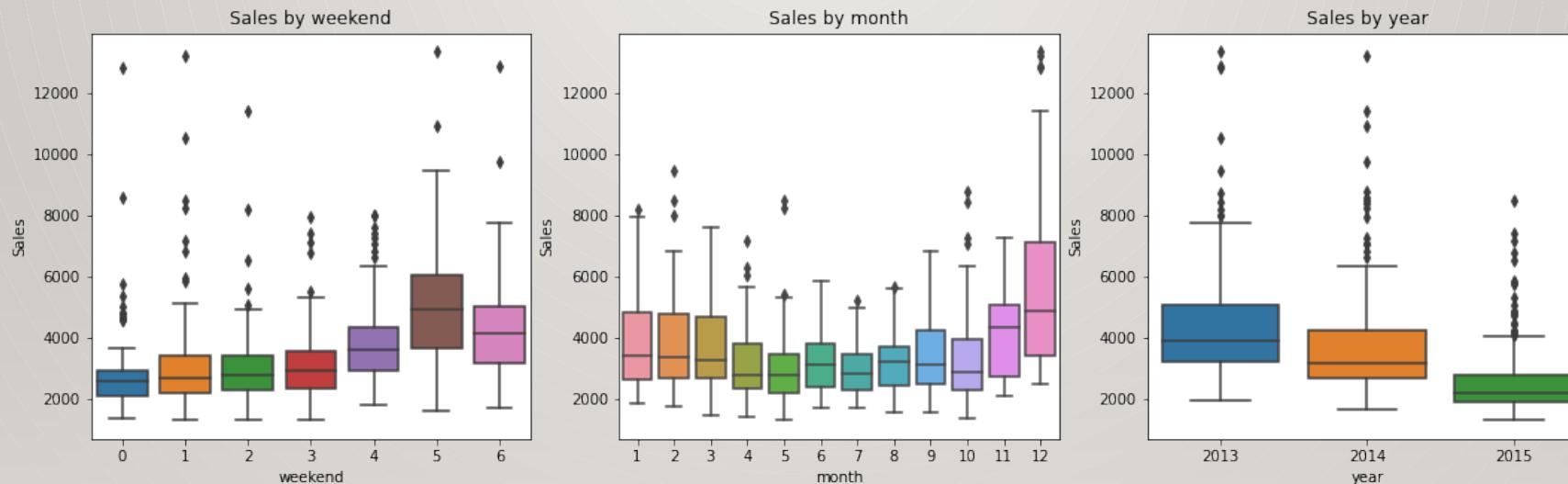
Công ty đang kinh doanh khoảng 22170 sản phẩm với 84 loại mặt hàng khác nhau

```
# Số lượng sản phẩm trong danh mục items  
items['item_id'].nunique()  
>>> 22170  
# Số chủng loại mặt hàng trong danh mục categories  
categories['item_category_id'].nunique()  
>>> 84
```

### 3.1. BIỂU ĐỒ BAR DOANH THU THEO THÁNG TỪ 2013-2015

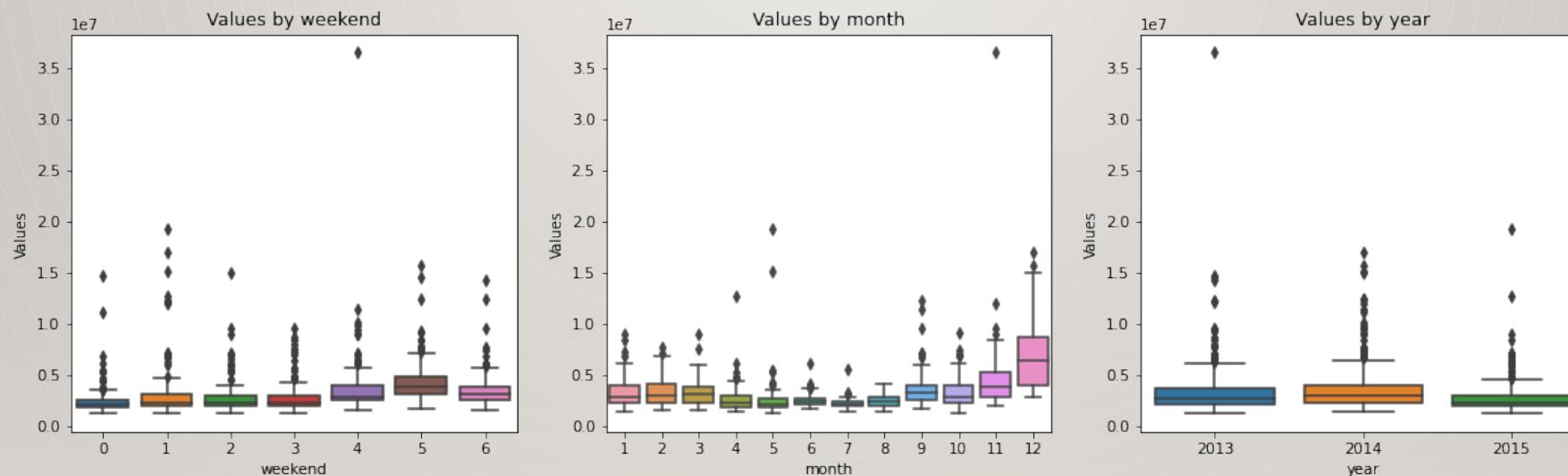


### 3.2. BIỂU ĐỒ BOX DOANH SỐ TRONG 3 NĂM 2013-2015



- Với biểu đồ **Tuần** ta thấy doanh số cao nhất vào ngày thứ 7 và thấp nhất vào ngày thứ 2. Vậy khách hàng có thói quen mua vào các ngày thứ 7 và chủ nhật
- Với biểu đồ **Tháng**, doanh số tăng cao nhất vào tháng 12, thấp nhất vào các tháng giữa năm, như vậy khách hàng thường mua hàng nhiều hơn vào dịp noel, các kỳ nghỉ lễ tết mùa đông
- Với biểu đồ **Năm**, doanh số của công ty đang có dấu hiệu sụt giảm, tuy năm 2015 mới có dữ liệu đến tháng 10 và 2 tháng cuối năm có doanh số lớn nhất nhưng tổng cả năm khó có thể bằng năm 2013 được, điều này cũng thể hiện rõ từ biểu đồ **Sale by date** phía trên.

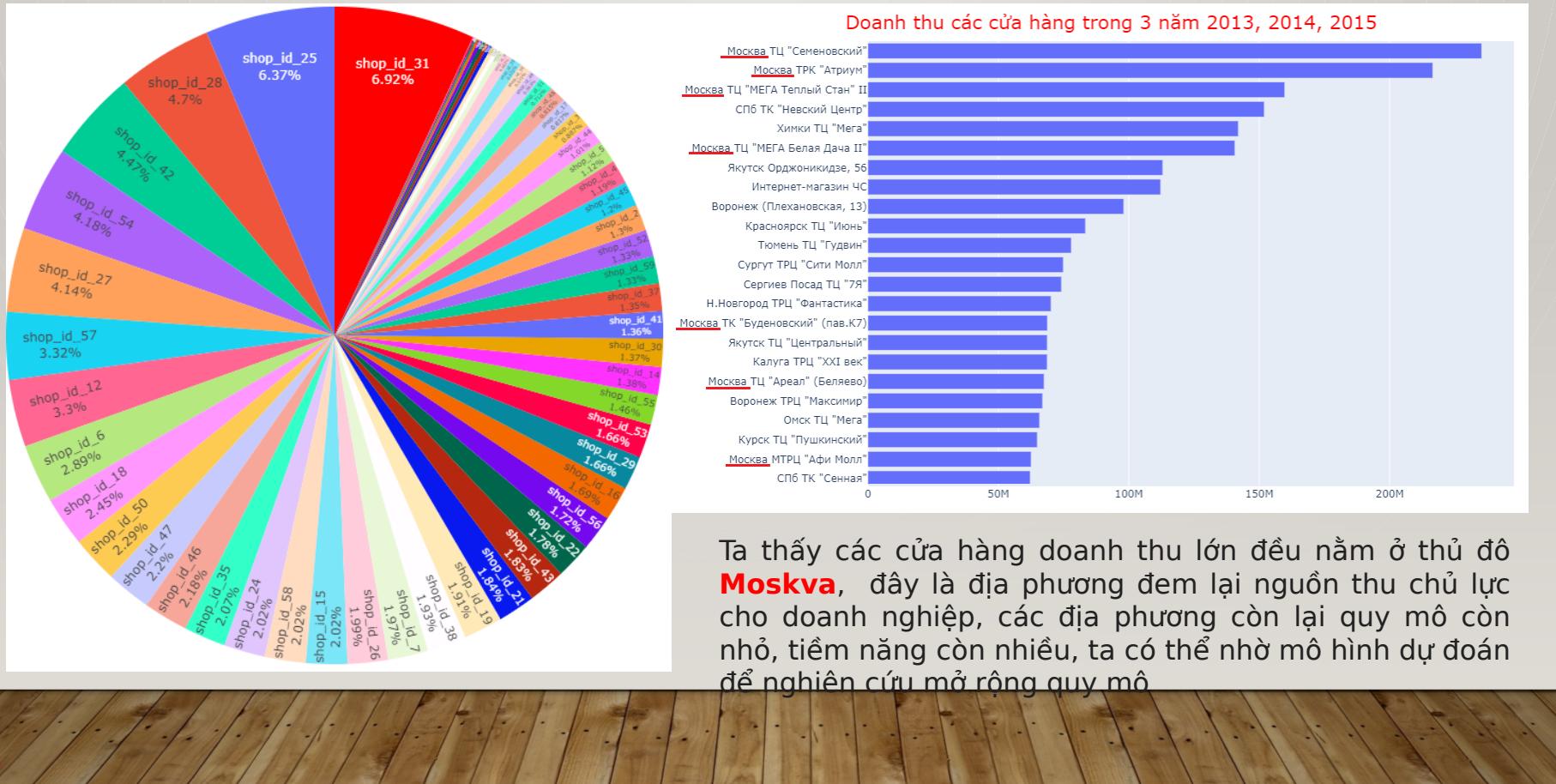
### 3.3. BIỂU ĐỒ BOX DOANH THU TRONG 3 NĂM 2013-2015



Qua biểu đồ box doanh thu ta thấy điều tương tự như box doanh số, tức doanh thu tăng vào dịp cuối tuần và cuối năm, doanh thu năm 2015 có sự sụt giảm so với các năm trước. Điều này cũng thể hiện rõ từ biểu đồ **doanh thu tháng** phía trên

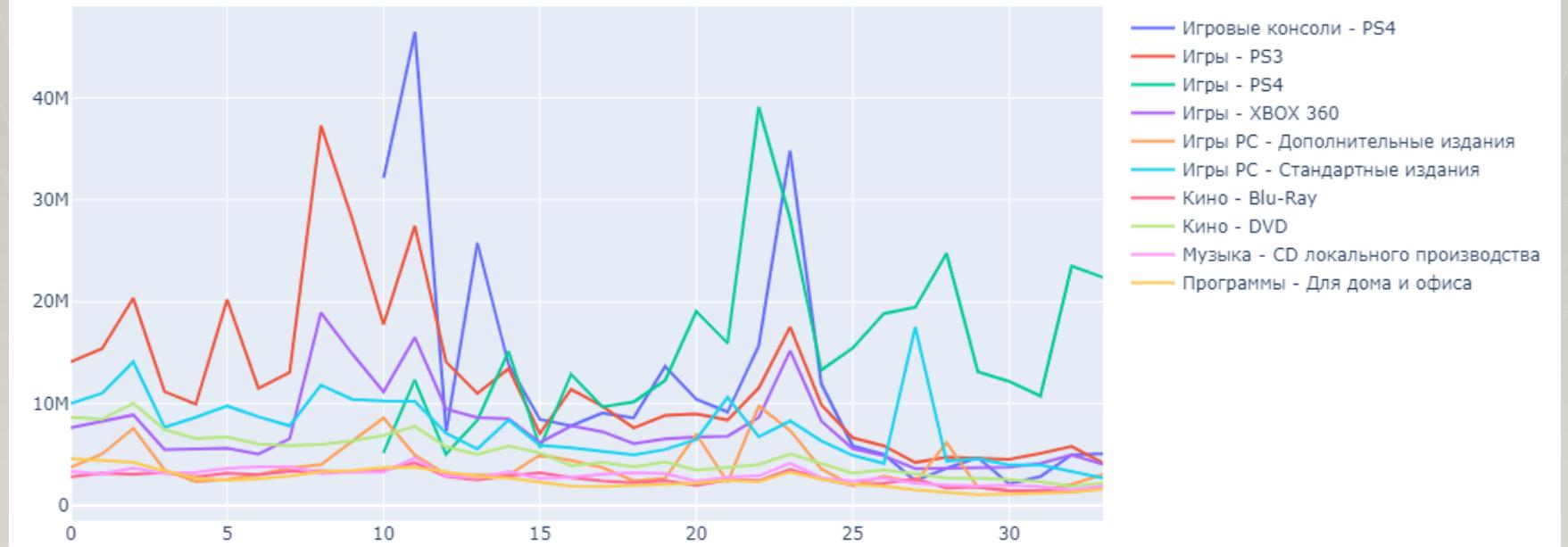
Ngoài ra ta thấy có một sự đột biến doanh thu lớn vào tháng 11 năm 2013. Ngày 29/11/2013 đều đến từ các máy PlayStation4 với item\_ID 6675, điều này đã đưa về doanh thu đột biến cho công ty

### 3.4. BIỂU ĐỒ TỶ LỆ DOANH THU 2013-2015



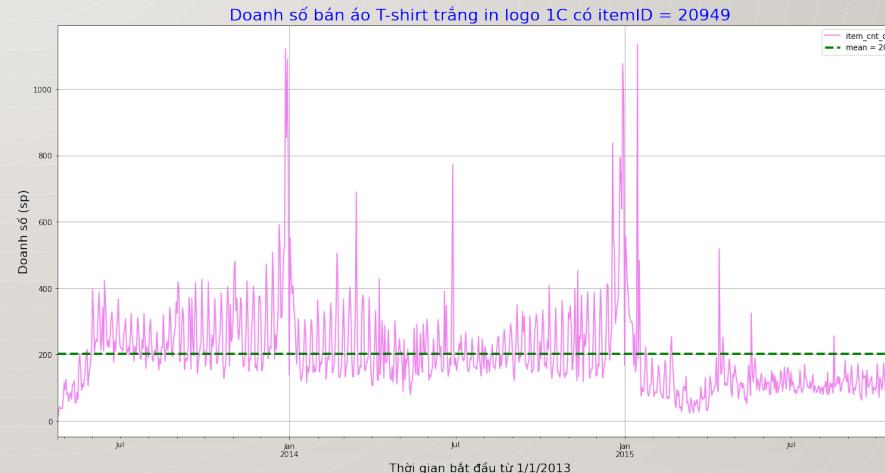
### 3.5. MƯỜI DANH MỤC CÓ DOANH THU CAO NHẤT

Doanh thu hàng tháng của 10 danh mục có doanh thu tốt nhất



Doanh thu đến từ các danh mục bán máy chơi game PS4, PS3, Xbox, máy tính PC và danh mục đầu đĩa Bluray, CD, DVD chiếm áp đảo các sản phẩm khác, **đây là các sản phẩm thế mạnh** mà công ty cần tập trung

### 3.6. TOP 15 SẢN PHẨM BÁN CHẠY NHẤT



Khá bất ngờ, sản phẩm bán chạy nhất của 1C có itemID=20949 là những chiếc áo T-shirt trắng in logo 1C có giá trung bình 5 rúp - khoảng 2000 VNĐ một chiếc)

Doanh số sản phẩm thứ 1 này vượt gấp nhiều lần sản phẩm bán chạy thứ 2

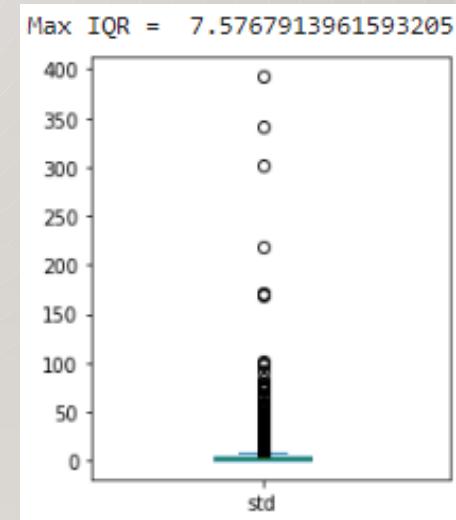
Những ngày bán hàng có doanh số tăng đột biến là các ngày cuối năm, tháng 3/2014, tháng 4/2014, tháng 10/2014 nên chắc đây là những thời điểm diễn ra sự kiện marketing của công ty

## **PHẦN 2: LẬP VÀ ĐÁNH GIÁ CÁC MÔ HÌNH DỰ ĐOÁN**

## 2.1. CHUẨN BỊ DỮ LIỆU ĐÀO TẠO VÀ DỮ LIỆU KIỂM TRA

Trong tập dữ liệu chúng ta biết có khoảng 3 triệu bản ghi, tập hợp được thành 400.000 chuỗi dữ liệu, nhưng phần lớn trong đó là các chuỗi rỗng (giá trị bằng 0, hoặc những sản phẩm đã dừng kinh doanh, hoặc sản phẩm quảng cáo mang tính sự kiện (chỉ bán khi công ty có sự kiện)... Do đó ta sẽ cần phải lọc dữ liệu để lấy ra những chuỗi phù hợp cho công tác dự báo doanh số

```
# Lọc chuỗi có doanh số trung bình lớn hơn 0
mask1 = data_sales.loc[:,0:33].median(axis = 1)>0
# Lọc chuỗi mà 3 tháng cuối bán được ít nhất 1 sp
mask2 = data_sales.iloc[:, -3: ].sum(axis = 1) > 1
# Lọc chuỗi có độ lệch chuẩn trong khoảng từ 0 đến max_IQR
Q1 = data['std'].quantile(0.25)
Q3 = data['std'].quantile(0.75)
IQR = Q3 - Q1; max_IQR = Q3 + 1.5 * IQR
mask3 = data['std'] < max_IQR
data = data_sales[mask1 & mask2 & mask3].copy()
data.shape
>>> (3825, 38)
```



Sử dụng bộ lọc sản phẩm có **mức bình quân dương** và 3 tháng cuối bán được **ít nhất 1 sản phẩm**, và doanh số các tháng có độ lệch chuẩn nằm trong khoảng min-max của biểu đồ BOX. Kết quả ta có được hơn 3800 chuỗi time\_series các sản phẩm đưa vào mô hình tính toán.

## 2.1. CHUẨN BỊ DỮ LIỆU (2)

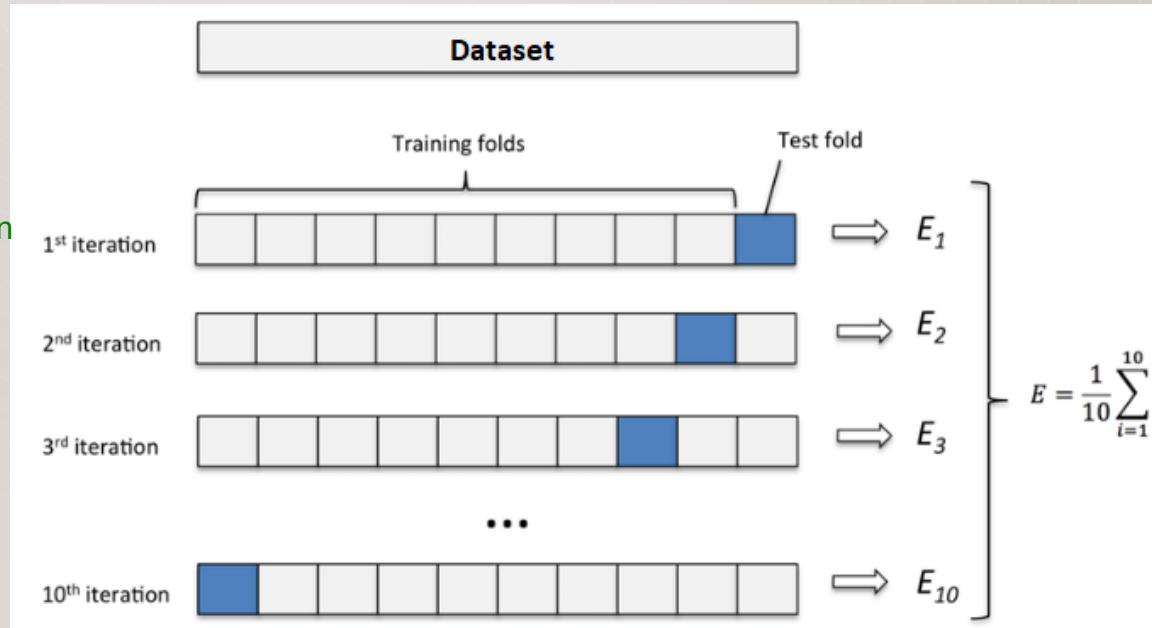
Index	idx	Dữ liệu input (X)																															Label (y)			
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32		
Train 90%	0	0	0.0	18.0	6.0	2.0	2.0	1.0	0.0	3.0	0.0	2.0	0.0	1.0	0.0	0.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	2.0	1.0	0.0	0.0	0.0	1.0	1.0	2.0
	1	1	0.0	14.0	7.0	1.0	0.0	2.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	1.0	0.0	1.0
	2	2	0.0	43.0	10.0	5.0	2.0	2.0	2.0	3.0	1.0	1.0	4.0	3.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	0.0	3.0	2.0	4.0	1.0	0.0	0.0	0.0	3.0	0.0	1.0	
	3	3	0.0	33.0	19.0	8.0	7.0	5.0	2.0	3.0	3.0	2.0	2.0	1.0	1.0	2.0	2.0	1.0	0.0	1.0	2.0	0.0	1.0	2.0	0.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	1.0			
	4	4	0.0	39.0	12.0	3.0	2.0	1.0	2.0	2.0	1.0	1.0	2.0	2.0	3.0	0.0	1.0	2.0	2.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	3.0	2.0	1.0	1.0	0.0	5.0	1.0	7.0	0.0	2.0
Test 10%	3820	3820	0.0	0.0	0.0	0.0	2.0	4.0	2.0	1.0	2.0	3.0	5.0	6.0	10.0	7.0	8.0	10.0	8.0	3.0	13.0	7.0	4.0	5.0	5.0	4.0	8.0	6.0	7.0	4.0	3.0	4.0	10.0	9.0	4.0	3.0
	3821	3821	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	1.0	1.0	3.0	3.0	2.0	1.0	0.0	2.0	4.0	1.0	1.0	3.0	2.0	4.0	2.0	2.0	0.0	0.0	5.0	4.0	
	3822	3822	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	4.0	2.0	1.0	3.0	2.0	3.0	3.0	4.0	7.0	5.0	5.0	2.0	8.0	4.0	5.0	0.0	1.0
	3823	3823	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	3.0	1.0	6.0	8.0	6.0	7.0	3.0	4.0	3.0	10.0	3.0	3.0	5.0	4.0	0.0	2.0	
	3824	3824	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	1.0	0.0	2.0	1.0	0.0	0.0	1.0	1.0	2.0	4.0	2.0	3.0	0.0	1.0	2.0	4.0	0.0	4.0	1.0	2.0	

Do số lượng mẫu ít (gần 4000 mẫu) nên ta sử dụng phương pháp K-Fold ( $k=10$ ) và kết quả mô hình được tính bằng điểm trung bình của 10 Fold.

## 2.1. CHUẨN BỊ DỮ LIỆU (3)

- **Phương pháp K-Fold với (k=10)** : Ta chia dữ liệu thành các tập dữ liệu train – test như hình dưới. Điểm số mô hình xác định bằng trung bình cộng của các Fold

```
# Tạo bộ chia KFold  
kfolds = KFold(n_splits=10,  
    shuffle=True,  
    random_state = 47)  
  
# Chia input và dữ liệu nhãn  
dataset = dataset.values  
inputs = dataset[:, :-1]  
labels = dataset[:, -1]  
  
# Tạo biến lưu số Fold  
fold_no = 1
```



## 2.2. MÔ HÌNH CƠ SỞ BASE\_MODEL

Mô hình cơ sở là kết quả ban đầu đưa ra để làm thước đo tham chiếu, đánh giá chất lượng giữa các mô hình một cách trực quan. Có nhiều mô hình cơ sở khác nhau, như **Naïve Forecast, EMA, SMA...** Đặc điểm chung của các mô hình cơ sở là sự đơn giản, các giá trị dự đoán được tính theo công thức xuất phát từ dữ liệu ban đầu mà không cần trải qua quá trình đào tạo phức tạp.

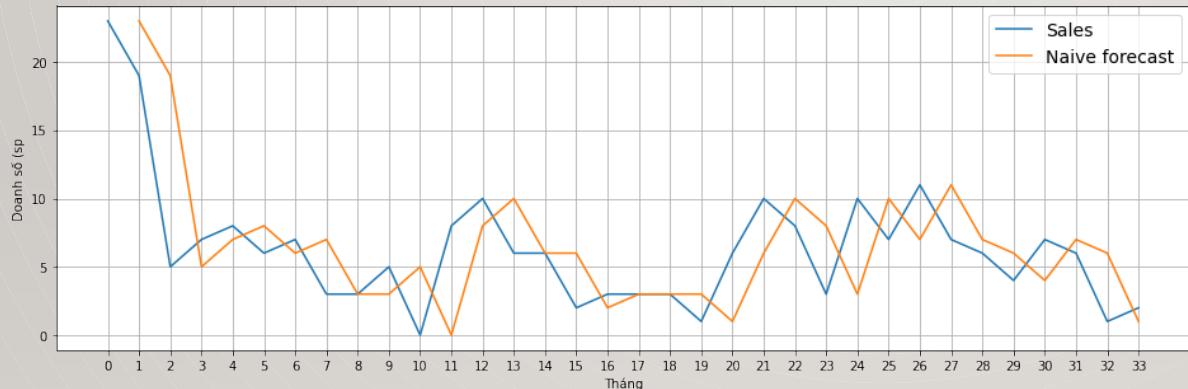
Ví dụ mô hình Naïve forecast:

Trong đó  $y_t$  là giá trị của chuỗi tại thời điểm  $t$   
Trong đó  $\hat{y}_t$  là giá trị dự đoán tại thời điểm  $t$

Ví dụ mô hình EMA

Ứng với bài toán của chúng ta =  $y_t$  và  $\hat{y}_t$  =  $\hat{y}_{t-1}$  nên công thức tính điểm RMSE

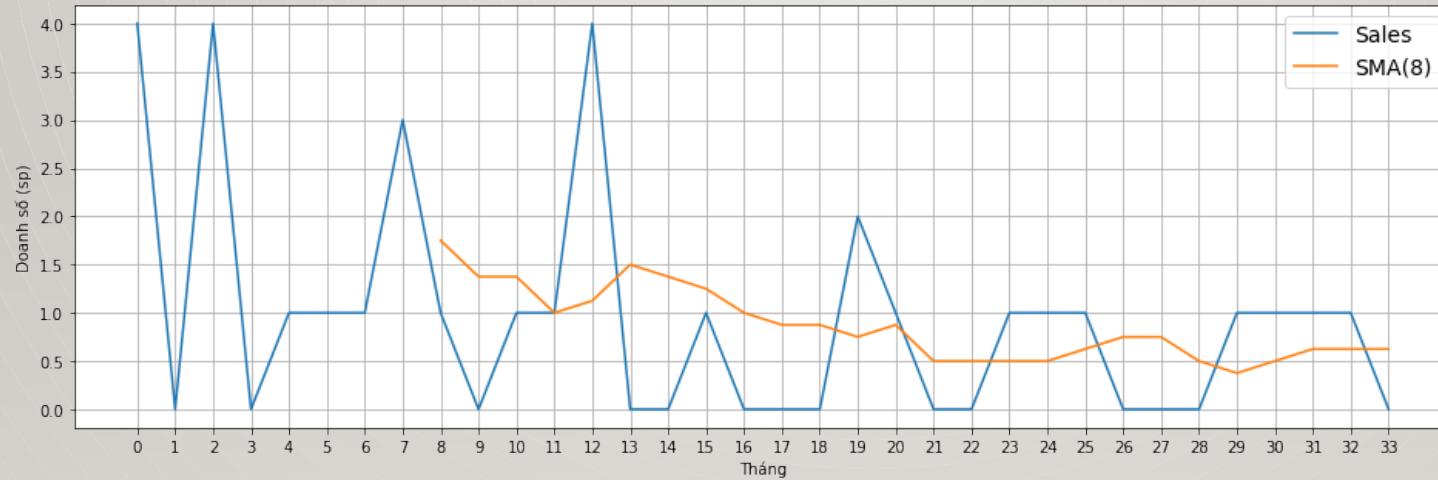
## 2.2.1 MÔ HÌNH NAÏVE FORECAST



Một trong những mô hình cơ sở phổ biến nhất cho dự báo chuỗi thời gian là mô hình Naïve. Không cần bất kỳ huấn luyện nào, tất cả những gì mà mô hình naïve làm là sử dụng giá trị bước thời gian trước đó để dự đoán giá trị bước thời gian tiếp theo với công thức:

```
# Tạo bảng lưu kết quả mô hình Naive_forecast
Naive_forecast_df['y33_pred'] = Naive_forecast_df[32]
# Tính điểm mse
Naive_forecast_df['mse'] = (Naive_forecast_df[32] - Naive_forecast_df[33])**2
# Lấy mức trung bình làm điểm số cho mô hình
Naive_forecast_mse_tb = round(Naive_forecast_df['mse'].mean(), 6)
Naive_forecast_rmse_tb = round(np.sqrt(Naive_forecast_mse_tb), 6)
print(f'Diểm số Naive_forecast: rmse: {Naive_forecast_rmse_tb}')
>>> Điểm số Naive_forecast: rmse: 2.029392
Điểm số Naive_forecast rmse = 2.029, đây là điểm số mà chúng ta cần phải vượt qua trong các mô hình tiếp theo
```

## 2.2.2 MÔ HÌNH TRUNG BÌNH TRƯỢT ĐƠN SMA

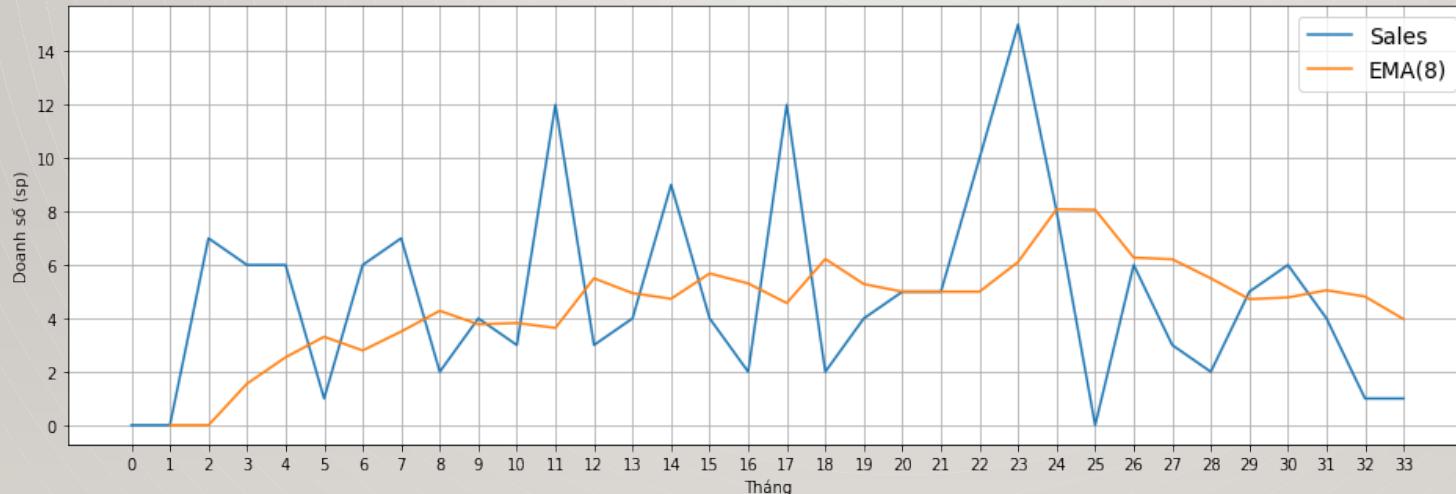


### Mô hình SMA (Simple Moving Average)

Hiểu đơn giản, đường trung bình trượt đơn được tạo bởi các điểm là trung bình cộng của chuỗi trong khoảng

```
# Ta tính sai số RMSE với các khoảng thời gian khác nhau, từ 6 đến 12 tháng thời gian W
for w in range(6,13):
    SMA_rt[w] = sma(w)           # sma(w) là hàm tính điểm RMSE tại khoảng thời gian w
    print(f'W = {w}, rmse = {SMA_rt[w]}')
print(f'Trung vị = {np.median(SMA_rt.values())}')
>>> Trung vị = 1.744606
Điểm số Single Moving Average SMA(9) rmse = 1.744
```

## 2.2.3 MÔ HÌNH TRUNG BÌNH TRƯỢT CÓ TRỌNG SỐ EMA



```
for w in range(6,13):
    rt[w] = ema(w)
    kq.append(rt[w][1])
print(f'Trung vị = {np.median(kq)}')
>>>
Trung vị = 1.677714
```

Điểm số Exponential Moving Average EMA(9) rmse = 1.677

### Mô hình EMA (Exponential Moving Average)

Đường trung bình trượt có trọng số giống đường trung bình trượt đơn nhưng các thời điểm gần được đánh trọng số lớn hơn, do vậy nó có thể là một mô hình tốt hơn và nắm bắt tốt hơn chuyển động của xu thế.

## 2.3. MÔ HÌNH MẠNG NƠRON HỌC SÂU DNN (1)

Quay trở lại với dữ liệu dataset gồm dữ liệu input (X) và dữ liệu nhãn label (y)

Dữ liệu bao gồm **3825** chuỗi, mỗi chuỗi có 34 điểm giá trị tương ứng doanh số của 34 tháng liên tục từ tháng 1/2013 và được **đánh số từ 0 đến 33**.

- Các đặc trưng (X) bao gồm 33 giá trị đầu tiên của chuỗi (từ cột 0 đến cột 32) và nhãn dự đoán (y) là giá trị của tháng cuối

Index	idx	Dữ liệu input (X)																																Label (y)					
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32					
Train 90%	0	0.0	18.0	6.0	2.0	2.0	1.0	0.0	1.0	3.0	0.0	2.0	0.0	1.0	0.0	0.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	2.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	2.0				
	1	1.0	0.0	14.0	7.0	1.0	0.0	2.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0	0.0	2.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0		
	2	2.0	0.0	43.0	10.0	5.0	2.0	2.0	2.0	3.0	1.0	1.0	4.0	3.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	2.0	0.0	3.0	2.0	4.0	1.0	0.0	0.0	0.0	0.0	3.0	0.0	1.0	1.0	0.0		
	3	3.0	0.0	33.0	19.0	8.0	7.0	5.0	2.0	3.0	3.0	2.0	2.0	1.0	1.0	2.0	1.0	2.0	2.0	1.0	0.0	1.0	2.0	0.0	1.0	2.0	0.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	1.0	1.0	0.0		
	4	4.0	0.0	39.0	12.0	3.0	2.0	1.0	2.0	2.0	1.0	2.0	2.0	2.0	3.0	0.0	1.0	2.0	2.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	3.0	2.0	1.0	1.0	0.0	5.0	1.0	7.0	0.0	2.0	2.0		
		3820	3820	0.0	0.0	0.0	0.0	2.0	4.0	2.0	1.0	2.0	3.0	5.0	6.0	10.0	7.0	8.0	10.0	8.0	3.0	13.0	7.0	4.0	5.0	5.0	4.0	8.0	6.0	7.0	4.0	3.0	4.0	10.0	9.0	4.0	3.0		
		3821	3821	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	1.0	1.0	3.0	3.0	2.0	1.0	0.0	2.0	4.0	1.0	1.0	3.0	2.0	4.0	2.0	0.0	5.0	4.0	1.0
		3822	3822	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	4.0	2.0	1.0	3.0	2.0	3.0	3.0	4.0	7.0	5.0	5.0	2.0	8.0	4.0	5.0	0.0	1.0
		3823	3823	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	3.0	1.0	6.0	8.0	6.0	7.0	3.0	4.0	3.0	10.0	3.0	3.0	5.0	4.0	0.0	2.0		
		3824	3824	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	1.0	0.0	1.0	0.0	2.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	2.0	4.0	0.0	4.0	1.0	2.0

Cột **index** được bổ sung vào bảng để dễ dàng cho việc ghép kết quả predict của model sau mỗi lần chia K-Fold chứ không mang ý nghĩa là một đặc trưng để đưa vào mô hình đào tạo.

## 2.3. MÔ HÌNH MẠNG NƠRON HỌC SÂU DNN (3)

```
# Tạo hàm fit model DNN
def DNN_fit(X_train, y_train, X_val, y_val):
    DNN_model = Sequential()
    DNN_model.add(Dense(128, activation="relu"))
    DNN_model.add(Dense(64, activation="relu"))
    DNN_model.add(Dense(16, activation="relu"))
    DNN_model.add(Dense(1, activation="linear"))
    DNN_model.compile(optimizer='adam', loss='mse',
                       metrics=[tf.keras.metrics.RootMeanSquaredError(name='rmse')])
    # fit model
    DNN_model.fit(X_train, y_train, epochs=250, verbose=0,
                  validation_data=(X_val, y_val),
                  callbacks=[tf.keras.callbacks.EarlyStopping(monitor="val_loss",
                                                              patience=30,
                                                              restore_best_weights=True)])
    return DNN_model
```

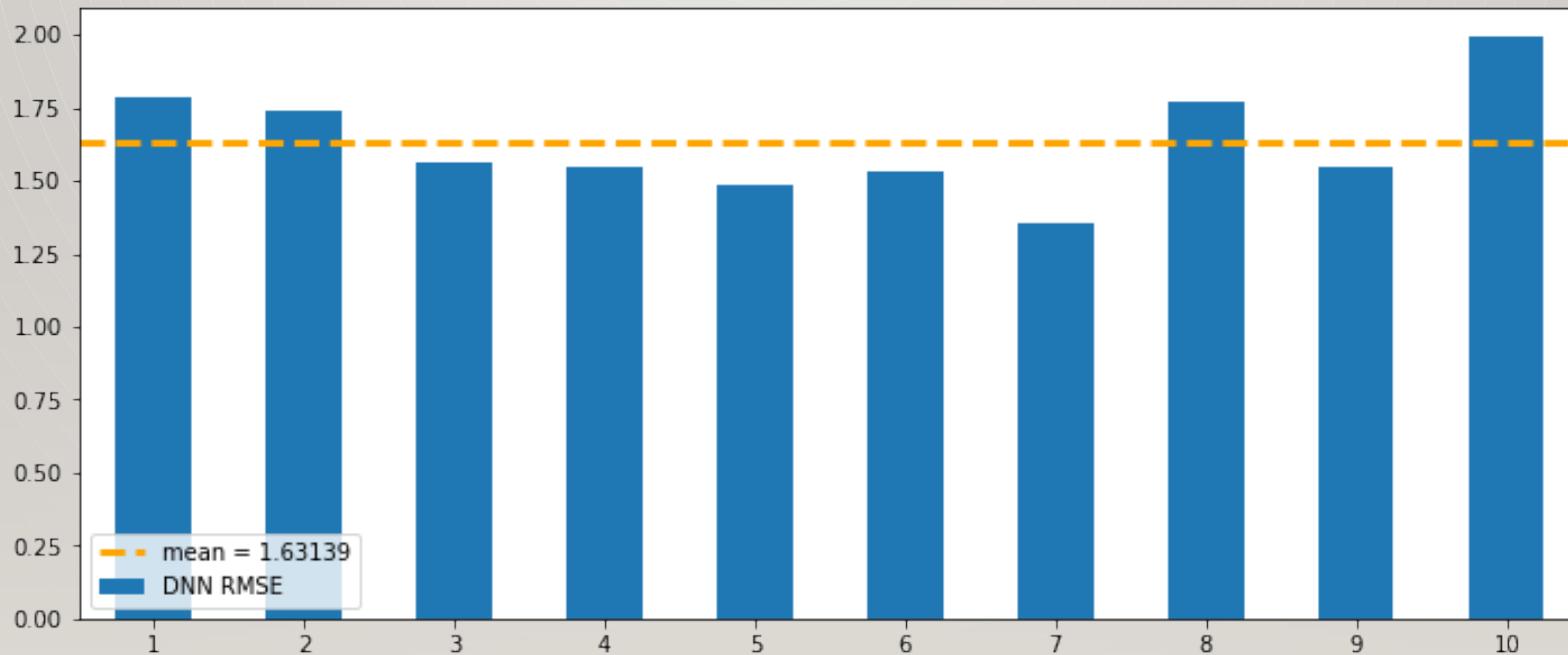
- Việc tuning các tham số của mô hình xem thêm trong file *Lua\_chon\_layer\_model\_DNN\_LSTM.ipynb* đi kèm đồ án

## 2.2. MÔ HÌNH MẠNG NƠRON HỌC SÂU DNN (4)

```
# Chạy mô hình DNN qua các Fold
for train, test in kfold.split(inputs, labels):
    print('-----')
    print(f'Training for fold {fold_no} ...')
    X_train, X_val, y_train, y_val = train_test_split(inputs[train], labels[train],
                                                       test_size =0.2, random_state=47)
    X_test, y_test = inputs[test], labels[test]
    DNN_model = DNN_fit(X_train, y_train, X_val, y_val)
    mse, rmse = DNN_model.evaluate(X_test, y_test)
    DNN_rmse[fold_no] = rmse
>>> -----
Training for fold 1 ... 12/12 [=====] - 0s 1ms/step - loss:
3.1820 - rmse: 1.7838 RMSE_per_fold 1= 1.7838101387023926
-----
Training for fold 2 ... 12/12 [=====] - 0s 1ms/step - loss:
3.0296 - rmse: 1.7406 RMSE_per_fold 2= 1.7405788898468018
...
-----
Training for fold 10 ... 12/12 [=====] - 0s 1ms/step - loss:
3.9774 - rmse: 1.9943 RMSE_per_fold 10= 1.9943475723266602
```

## 2.2. MÔ HÌNH MẠNG NƠRON HỌC SÂU DNN (5)

Điểm số RMSE của các K-Fold của mô hình DNN



Mô hình DNN có điểm **rmse = 1.631**, có cải thiện hơn so với kết quả của mô hình cơ sở SMA(9) có **rmse = 1.677**

(Kết quả dự đoán của mô hình DNN được lưu trong tập tin [DNN\\_predict\\_df.csv](#))

## 2.3 MÔ HÌNH MẠNG NƠRON HỒI TIẾP LSTM

### Mô hình Long short term memory (LSTM)

LSTM là một dạng đặc biệt của mô hình mạng hồi tiếp RNN (Recurrent Neural Network). Bằng thiết kế bổ sung thêm các cổng kết nối, mô hình LSTM có khả năng học được các phụ thuộc xa tốt hơn mô hình RNN truyền thống, và với cách thức hoạt động hồi tiếp lấy đầu ra của nút này làm đầu vào của nút kế tiếp, mô hình LSTM khá phù hợp cho việc học tập và dự đoán các chuỗi thời gian.

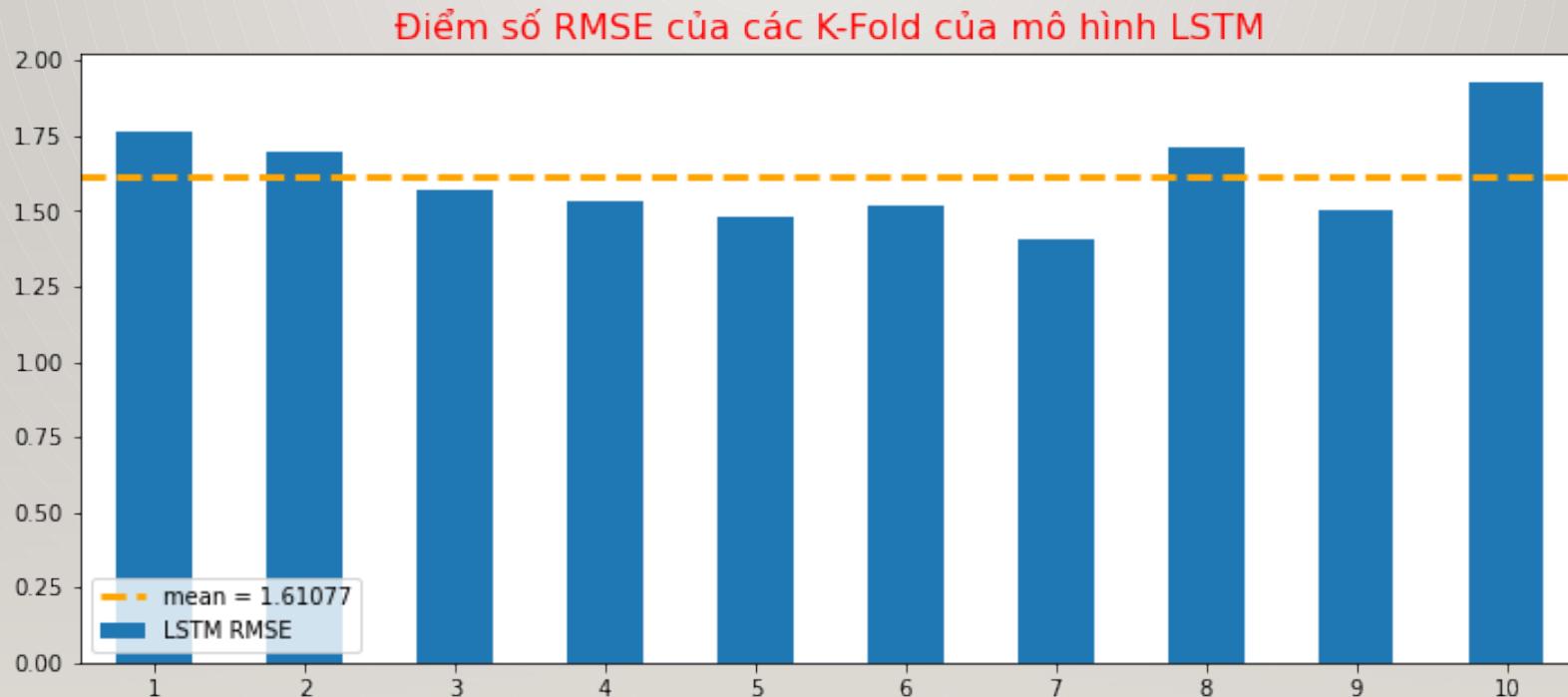
```
# Tạo hàm fit model LSTM
def LSTM_fit(X_train, y_train, X_val, y_val):
    X_train = np.expand_dims(X_train, axis=1)
    X_val = np.expand_dims(X_val, axis=1)
    LSTM_model_fit = Sequential()
    LSTM_model_fit.add(Bidirectional(LSTM(128, activation="relu", return_sequences=True,
                                         input_shape=(1, 33))))
    LSTM_model_fit.add(LSTM(64, activation="relu"))
    LSTM_model_fit.add(Dense(1, activation="linear"))
    LSTM_model_fit.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.00005),
                           loss='mse', metrics=[tf.keras.metrics.RootMeanSquaredError(name='rmse')])
    # fit model
    LSTM_model_fit.fit(X_train, y_train, epochs=50, verbose=0,
                        return LSTM_model_fit
```

- Việc tuning các tham số của mô hình xem thêm trong file *Lua\_chon\_layer\_model\_DNN\_LSTM.ipynb* đi kèm đồ án

## 2.3 MÔ HÌNH MẠNG NƠRON HỒI TIẾP LSTM (2)

```
# Chạy mô hình LSTM qua các Fold
for train, test in kfold.split(inputs, labels):
    print('-----')
    print(f'Training for fold {fold_no} ...')
    X_train, X_val, y_train, y_val = train_test_split(inputs[train], labels[train],
                                                       test_size =0.2, random_state=47)
    X_test, y_test = inputs[test], labels[test]
    LSTM_model = LSTM_fit(X_train, y_train, X_val, y_val)
    mse, rmse = LSTM_model.evaluate(X_test, y_test)
    LSTM_rmse[fold_no] = rmse
>>> -----
Training for fold 1 ... 12/12 [=====] - 0s 3ms/step - loss:
3.1181 - rmse: 1.7658
-----
Training for fold 2 ... 12/12 [=====] - 0s 2ms/step - loss:
2.8804 - rmse: 1.6972
...
-----
Training for fold 10 ... 12/12 [=====] - 0s 3ms/step - loss:
3.7156 - rmse: 1.9276
```

## 2.3 MÔ HÌNH MẠNG NƠRON HỒI TIẾP LSTM (3)



Mô hình LSTM có điểm **rmse = 1.611**, có cải thiện hơn so với kết quả của mô hình cơ sở SMA(9) có **rmse = 1.677**

(Kết quả dự đoán của mô hình LSTM được lưu trong tập tin *LSTM\_predict\_df.csv*)

## 2.4 THUẬT TOÁN XGBOOST

### Thuật toán XGBoost:

XGBoost là viết tắt của Extreme Gradient Boosting, một thuật toán với nhiều ưu điểm và được sử dụng thường xuyên trong các cuộc thi Kaggle. Thuật toán XGB cũng dùng để giải quyết các bài toán học có giám sát giống mô hình Deep learning, nhưng đầu vào được tối ưu để có thể nhận input gồm các định dạng không phải dạng số như str, categorical... tốc độ huấn luyện nhanh, hỗ trợ GPU và nhiều ưu điểm khác.

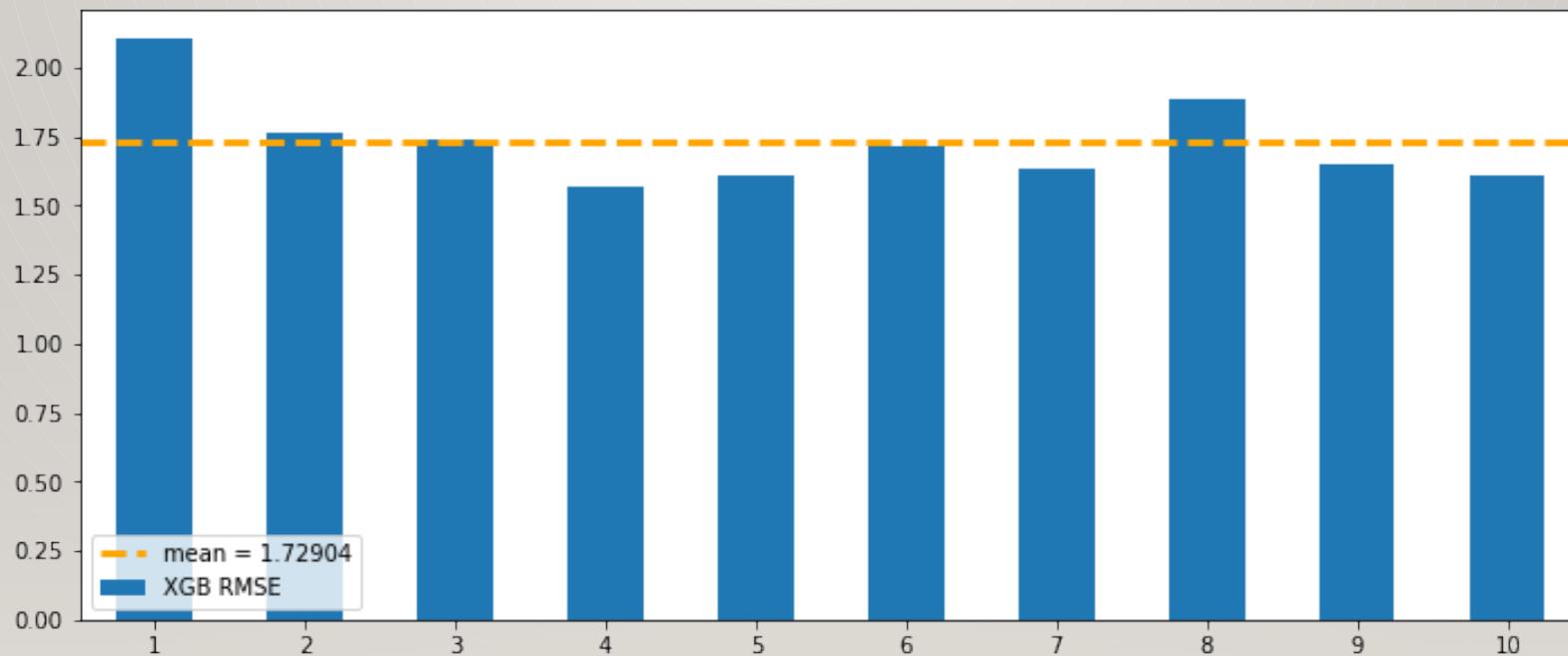
```
import xgboost as xgb
# Chia tách tập train và test
features_name = data1_xgb.columns.values
X = data1_xgb[features_name[:-1]]
y = data1_xgb[features_name[-1:]]
# Tinh chỉnh tham số cho param
param = {'max_depth': 26,                      # Độ sâu tối đa của cây, default=6
          'obj':'gbtree',                     # Loại thuật toán tăng cường
          'min_child_weight': 1,               # tương tự learning_rate: default=0.3
          'eta':0.31,                         # Số lượng vòng lặp
          'num_round':1000,                   # Thước đo rmse cho hồi quy
          'eval_metric':'rmse',                # n_estimators : Số lượng trees
          'n_estimators': 10}
```

## 2.4 THUẬT TOÁN XGBOOST (2)

```
# Chạy mô hình XGB qua các K-Fold
for train, test in kfold.split(X, y):
    print('-----')
    print(f'Training for fold {fold_no} ...')
    id_train, id_val = train_test_split(train, test_size =0.2, random_state=42)
    X_train, y_train = X.iloc[id_train], y.iloc[id_train]
    X_val, y_val = X.iloc[id_val], y.iloc[id_val]
    X_test, y_test = X.iloc[test], y.iloc[test]
    # Tạo và train mô hình
    train_matrix = xgb.DMatrix(X_train.values, y_train.values)
    val_matrix = xgb.DMatrix(X_val.values, y_val.values)
    evallist = [(train_matrix, 'train'), (val_matrix, 'val')]
    xgb_model = xgb.train(param, train_matrix, evals=evallist, early_stopping_rounds=20)
    # Dự đoán kết quả tập test
    y_true = pd.Series(y_test.values.reshape(-1,))
    y_pred = xgb_model.predict(xgb.DMatrix(X_test.values))
    # Tính toán điểm số
    xgb_mse = np.mean((y_true - y_pred)**2)
    xgb_rmse = round(np.sqrt(xgb_mse), 6)
    print(f'RMSE_per_fold {fold_no}= {xgb_rmse}')
```

## 2.4 THUẬT TOÁN XGBOOST (3)

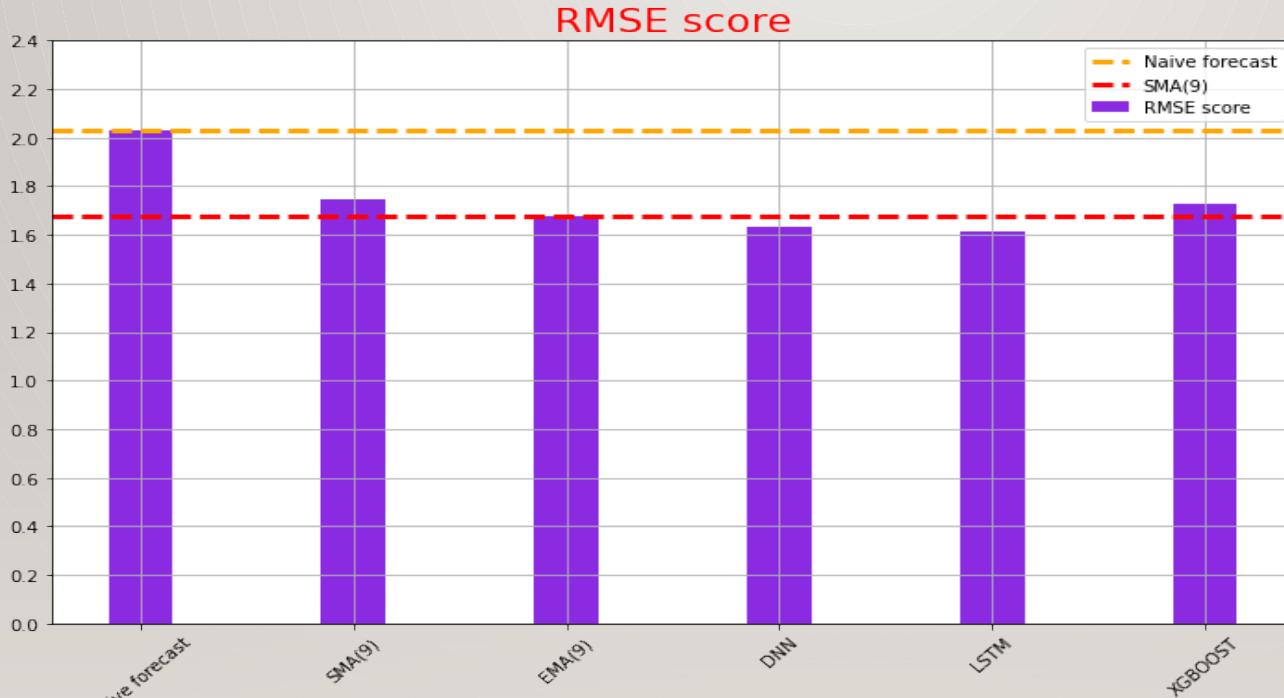
Điểm số RMSE của các K-Fold của mô hình XGB



Mô hình LSTM có điểm **rmse = 1.729**

(Kết quả dự đoán của mô hình LSTM được lưu trong tập tin *XGB\_predict\_df.csv*)

## 2.5. TỔNG HỢP KẾT QUẢ ĐẠT ĐƯỢC



Kiểm tra kết quả đã lưu trong bảng **score result**

	rmse
Naive forecast	2.02939
SMA(9)	1.74460
EMA(9)	1.67771
DNN	1.63138
LSTM	1.61077
XGBOOST	1.72904

- Các mô hình khá đồng đều về mặt chất lượng, ta thấy mô hình LSTM có kết quả **1.610**

- Tuy nhiên chênh lệch kết quả giữa các mô hình không nhiều, và ta sẽ tìm hiểu nguyên nhân và cách thức để cải thiện chất lượng mô hình ở phần sau

## 2.6. DỰ ĐOÁN CUỘC THI VỚI MÔ HÌNH XGBOOST

### Mô hình XGBOOST

Mô hình **XGBOOST** sử dụng khi dữ liệu có mức độ phức tạp lớn, với nhiều ưu điểm là thời gian đào tạo nhanh, có thể kết hợp nhiều định dạng vào tập đào tạo mà không cần các bước trung gian như one\_hot encode...

### Định dạng dữ liệu đầu vào

Ngoài thông tin về doanh số của 33 tháng liền kề, ta sẽ bổ sung thêm một số thông tin vào tập dữ liệu đào tạo như **shop\_id** của cửa hàng, **item\_id** của sản phẩm, **item\_catalogue\_id** của loại\_mặt\_hàng, và **ID** là index của dòng. Để mô hình không hiểu nhầm các thuộc tính có ID=2 gấp đôi thuộc tính có ID=1, ta đưa

```
# Tạo dữ liệu data_xgb
# data_xgb = data_sales.copy()
# data_xgb["ID"] = data_xgb["ID"].astype("str")
# data_xgb["shop_id"] = data_xgb["shop_id"].astype("str")
# data_xgb["item_id"] = data_xgb["item_id"].astype("str")
# data_xgb["item_category_id"] = data_xgb["item_category_id"].astype("str")
# data_xgb = data_xgb.drop_duplicates()
```

## 2.6. DỰ ĐOÁN CUỘC THI VỚI MÔ HÌNH XGBOOST (2)

```
9 data_xgb["ID"] = data_xgb["ID"].astype("str")
10 data_xgb["shop_id"] = data_xgb["shop_id"].astype("str")
11 data_xgb["item_id"] = data_xgb["item_id"].astype("str")
12 data_xgb["item_category_id"] = data_xgb["item_category_id"].astype("str")
13 data_xgb = data_xgb.drop_duplicates()
14 data_xgb
```

Ta thấy dataset có hơn 420.000 chuỗi dữ liệu, đa phần các chuỗi có rất nhiều số 0 thể hiện tại các tháng đó, mặt hàng đang không có doanh số (Sản phẩm chưa lên kệ hoặc sản phẩm trong tháng đó không có người mua, hoặc đã dừng kinh doanh).

## 2.6. DỰ ĐOÁN CUỘC THI VỚI MÔ HÌNH XGBOOST (3)

### Chia tập train và test

Mỗi dòng đều là một chuỗi thời gian riêng biệt không có thứ tự nên ta có thể sử dụng hàm `train_test_split` có sẵn để chia tách dữ liệu gốc thành 2 tập train và test theo tỷ lệ 80-20

```
X = data_xgb[features_name[:-1]]  
y = data_xgb[features_name[-1:]]  
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

### Tạo dữ liệu train và test với DMatrix

`DMatrix` là điểm mạnh của mô hình XGBOOST, dù mô hình có phương thức `.fit()` như các mô hình khác nhưng với phương thức `.train()` kết hợp với ma trận `Dmatrix` dữ liệu đầu vào giúp cho mô hình giảm được nhiều bước tính trung gian với hiệu suất cao. Ta sẽ tiết kiệm được nhiều thời gian để tinh chỉnh tham số nếu sử dụng `Dmatrix`.

```
train_matrix = xgb.DMatrix(X_train.values, y_train.values)  
test_matrix = xgb.DMatrix(X_test.values, y_test.values)
```

### Đào tạo mô hình và tinh chỉnh tham số thông qua param

```
evallist = [(train_matrix, 'train'), (test_matrix, 'test')]  
model = xgb.train(param, train_matrix, evals=evallist, early_stopping_rounds=10)
```

Ta sẽ train các bộ số và theo dõi kết quả evallist để tìm bộ số cho kết quả tối ưu nhất

( như param trên cho **test-rmse:1.24085** là tối ưu )

## 2.6. DỰ ĐOÁN CUỘC THI VỚI MÔ HÌNH XGBOOST (4)

## Tạo dữ liệu test\_xgb

Dữ liệu test của cuộc thi chỉ bao gồm `shop_ID` và `item_ID`, do đó ta cần ghép các thuộc tính còn lại của bộ dữ liệu đào tạo với dữ liệu test, các thôna số còn thiếu được `fillna = 0`

```
6 col2 = [i for i in range(-1,33)]
7 test_xgb = test_xgb[['ID', 'shop_id', 'item_id', 'item_category_id']+col2]
8 test_xgb.columns = ['ID', 'shop_id', 'item_id', 'item_category_id']+col2
9 test_xgb.drop(columns = -1, axis = 1, inplace = True)
10 test_xgb["ID"] = test_xgb["ID"].astype("str")
11 test_xgb["shop_id"] = test_xgb["shop_id"].astype("str")
12 test_xgb["item_id"] = test_xgb["item_id"].astype("str")
13 test_xgb["item_category_id"] = test_xgb["item_category_id"].astype("str")
14 test_xgb
```

## 2.6. DỰ ĐOÁN CUỘC THI VỚI MÔ HÌNH XGBOOST (5)

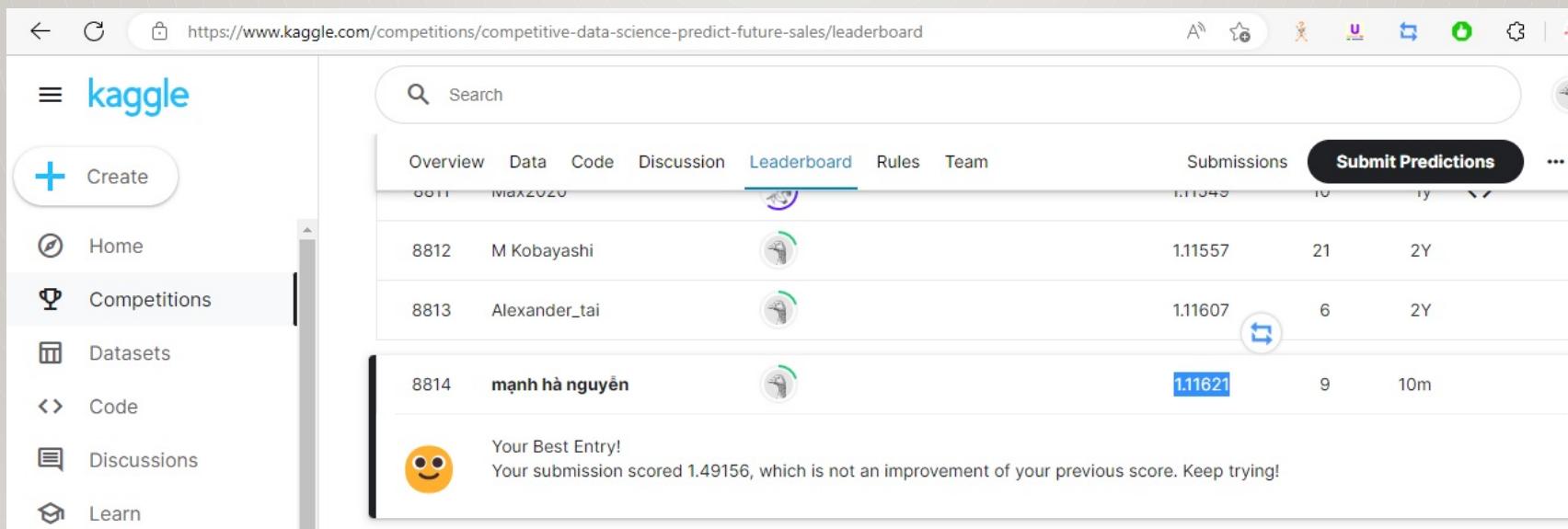
### Dự đoán kết quả với tập test\_xgb

```
# Dự đoán kết quả từ dữ liệu test_xgb
predictions = model.predict(xgb.DMatrix(test_xgb.values))
predictions_remix = list(map(lambda x: min(20, x), list(predictions)))
# Tạo bảng DataFrame lưu kết quả dự đoán
submit_remix = pd.DataFrame({'item_cnt_month' : predictions_remix,
                             'ID' : test_xgb.index.tolist()})
submit_remix.set_index('ID', inplace = True)
```

### Lưu kết quả thành file csv để post lên trang chủ cuộc thi

```
submit_remix.to_csv('/content/drive/MyDrive/DSP305/Challenger/Challenger_XGBOOST/
Submission.csv')
```

## 2.6. DỰ ĐOÁN CUỘC THI VỚI MÔ HÌNH XGBOOST (6)



The screenshot shows the Kaggle competition page for "competitive-data-science-predict-future-sales". The user is on the Leaderboard tab, which lists submissions from other participants and the user's own entry. The user's entry, submission ID 8814, is highlighted with a yellow border. The submission details are as follows:

Rank	User	Score	Submissions	Team
8812	M Kobayashi	1.11557	21	2Y
8813	Alexander_tai	1.11607	6	2Y
8814	mạnh hà nguyễn	<b>1.11621</b>	9	10m

A message box at the bottom left of the list area says: "Your Best Entry! Your submission scored 1.49156, which is not an improvement of your previous score. Keep trying!"

Điểm submit với trang chủ kaggle đạt được là từ 1.116 - 1.498, phù hợp với kết quả các mô hình của chúng ta, tuy kết quả không tệ nhưng còn kém thứ hạng top **0.753** một khoảng cách lớn, sẽ còn nhiều việc cần làm để nâng cao chất lượng mô hình.

## **PHẦN 3: (FUTURE WORK)**

**CẢI THIỆN MÔ HÌNH  
CÁC CÔNG VIỆC CẦN LÀM TIẾP THEO**

### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

Kiểm tra những mẫu có sai số dự đoán lớn để tìm hiểu nguyên nhân dự đoán sai

```
# Tạo bảng Predict_df tổng hợp kết quả dự đoán của các mô hình
Pred_df = data.iloc[:,[0,1,2,3,-1]].copy()

# Nối kết quả dự đoán các mô hình vào bảng Predict_df
Pred_df = Pred_df.merge(EMA_result.iloc[:,1], left_index = True, right_index = True)
Pred_df = Pred_df.merge(DNN_predict_df['y_pred'], left_index = True, right_index = True)
Pred_df = Pred_df.merge(LSTM_predict_df['y_pred'], left_index = True, right_index = True)
Pred_df = Pred_df.merge(XGB_predict_df['y_pred'], left_index = True, right_index = True)
Pred_df.columns = ['ID','shop_id','item_id','item_category_id',
                   'y_true','EMA','DNN','LSTM','XGBOOST']

# Tính sai số mse của mỗi dự đoán và tính sai số trung bình
Pred_df['DNN_mse'] = (Pred_df['y_true'] - Pred_df['DNN'])**2
Pred_df['LSTM_mse'] = (Pred_df['y_true'] - Pred_df['LSTM'])**2
Pred_df['XGB_mse'] = (Pred_df['y_true'] - Pred_df['XGBOOST'])**2
Pred_df['mean_mse'] = (Pred_df['DNN_mse']+Pred_df['LSTM_mse']+Pred_df['XGB_mse'])/3
Pred2_df = Pred_df.sort_values(by = 'mean_mse', ascending = False).head(100)
```

### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

```
# Xem 10 chuỗi có mean_mse lớn nhất
```

```
Pred2_df head(10)
```

ID	shop_id	item_id	item_category_id	y_true	EMA	DNN	LSTM	XGBOOST	DNN_mse	LSTM_mse	XGB_mse	mean_mse
1258	61977	24	12134	30	25.0	1.923217	1.487519	1.522902	1.048439	552.83...	551.17413	573.67...
852	28364	42	5823	35	28.0	12.685307	8.958644	6.855683	9.812649	362.57...	447.08...	330.77...
2909	205457	42	11655	41	18.0	4.080380	3.769355	3.337561	2.502535	202.51...	214.98...	240.17...
3771	423350	55	6429	31	17.0	4.439772	3.468325	2.030247	2.803882	183.10...	224.09...	201.52...
2803	186468	42	7872	30	20.0	7.217845	7.526989	7.517146	5.260485	155.57...	155.82...	217.25...
837	28333	7	5823	35	14.0	3.267650	2.004588	2.735103	2.538277	143.88...	126.89...	131.37...
843	28344	19	5823	35	12.0	3.509104	2.347533	3.003057	2.803618	93.170124	80.944975	84.573438
1824	114249	56	22088	83	12.0	5.840967	2.275194	3.742481	2.50828	94.571858	68.186612	90.092743
2364	164968	21	4806	30	10.0	2.131471	2.163293	1.45604	1.215498	61.413971	72.999258	77.16748
2798	186452	24	7872	30	11.0	3.614228	3.827188	2.958329	1.377317	51.449235	64.668473	92.596029

### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

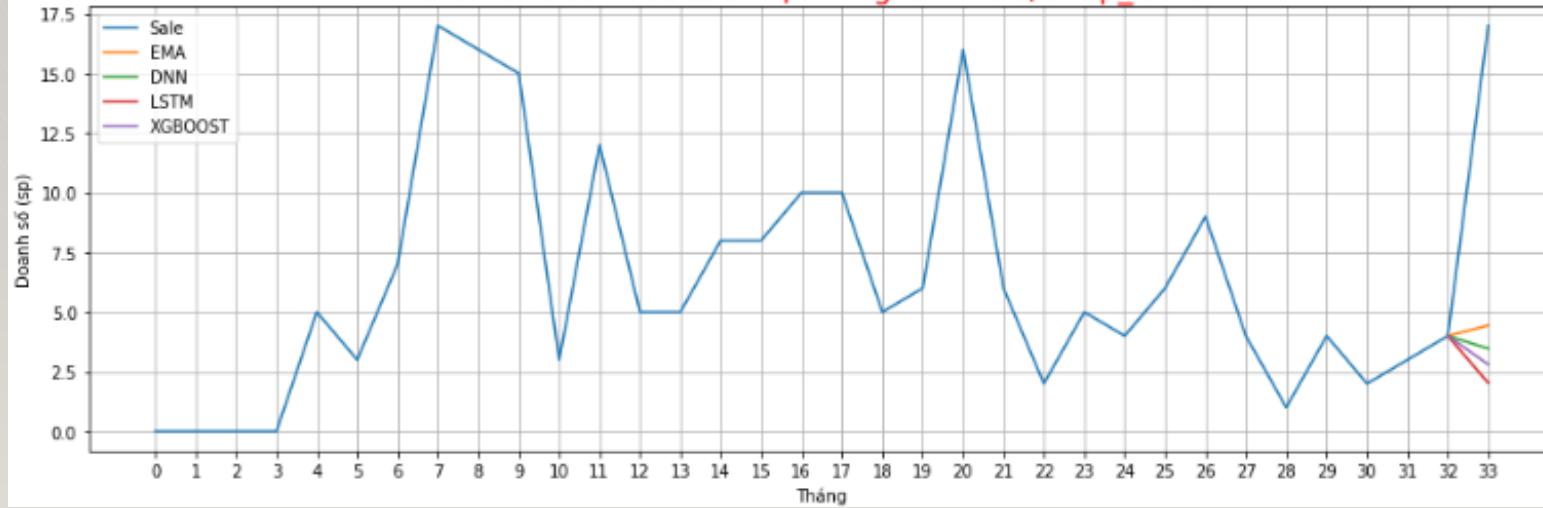
```
1 id = 1258
2 plot_time_series(data.loc[id,0:], label='Sale')
3 plot_time_series(pd.Series(data=[data.loc[id,32], k.loc[id,'EMA']]), index=[32,33]), label='EMA')
4 plot_time_series(pd.Series(data=[data.loc[id,32], k.loc[id,'DNN']]), index=[32,33]), label='DNN')
5 plot_time_series(pd.Series(data=[data.loc[id,32], k.loc[id,'LSTM']]), index=[32,33]), label='LSTM')
6 plot_time_series(pd.Series(data=[data.loc[id,32], k.loc[id,'XGBOOST']]), index=[32,33]), label='XGBOOST')
7 plt.legend();
```



### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

```
1 id = 3771
2 plot_time_series(data.loc[id,0:], label='Sale')
3 plot_time_series(pd.Series(data=[data.loc[id,32], k.loc[id,'EMA']], index=[32,33]), label='EMA')
4 plot_time_series(pd.Series(data=[data.loc[id,32], k.loc[id,'DNN']], index=[32,33]), label='DNN')
5 plot_time_series(pd.Series(data=[data.loc[id,32], k.loc[id,'LSTM']], index=[32,33]), label='LSTM')
6 plot_time_series(pd.Series(data=[data.loc[id,32], k.loc[id,'XGBOOST']], index=[32,33]), label='XGBOOST')
7 plt.legend();
```

Biểu đồ doanh số của mặt hàng id=6429, shop\_id=55



### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

Ta thấy các mẫu có dự đoán sai số lớn đều có doanh số tháng cuối tăng đột biến, các mô hình dự đoán của chúng ta có **mức độ tổng quát** vừa đủ để hạn chế các trường hợp overfit, nhưng lại không đủ chi tiết để dự đoán cho các trường hợp đột biến như một số mẫu trên. Để cải thiện chất lượng dự đoán, ta bổ sung phương án dự đoán doanh số đối với 1 mẫu cụ thể bằng mô hình **DNN\_single\_time\_series**

```
# Tạo hàm để tách chuỗi dưới dạng windowing
def make_windows(x, window_size=3, horizon=1):
    window_step = np.expand_dims(np.arange(window_size+horizon), axis=0)
    window_indexes = window_step + np.expand_dims(np.arange(len(x)-
                                                               (window_size+horizon-
                                                               1)), axis=0).T
    windowed_array = x[window_indexes]
    windows = windowed_array[:, :-horizon]
    labels = windowed_array[:, -horizon:]
    return (windows, labels)
```

### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

#### Định dạng dữ liệu

##### Windowing

Windowing là một phương pháp để biến một tập dữ liệu chuỗi thời gian thành một bài toán học có giám sát. Phương pháp Windowing có 2 thuộc tính là **Window size** và **horizon size**.

- Window size là kích thước của chuỗi thời gian được tách ra từ chuỗi thời gian ban đầu.
- Horizon size là số đơn vị thời gian mà ta muốn dự đoán.

**Chuỗi id = 1258**

[ 7. 4. 3. 0. 3. 1. 2. 3. 3. 2. 5. 3. 8. 0. 3. 3. 3. 3.

2. 1. 3. 2. 4. 3. 1. 2. 1. 0. 0. 2. 0. 1. 5. 25.]

Tách chuỗi bằng phương pháp windowing với WS=15 ta được dataset

| WS=15                | Window (X)   | Label (y)   |
|----------------------|--|---|
| Train<br>(18<br>mẫu) | [[7., 4., 3., 0., 3., 1., 2., 3., 3., 2., 5., 3., 8., 0., 3.], [4., 3., 0., 3., 1., 2., 3., 3., 2., 5., 3., 8., 0., 3., 3.], [3., 0., 3., 1., 2., 3., 3., 2., 5., 3., 8., 0., 3., 3., 3.], [0., 3., 1., 2., 3., 3., 2., 5., 3., 8., 0., 3., 3., 3., 3.], [3., 1., 2., 3., 3., 2., 5., 3., 8., 0., 3., 3., 3., 3., 2.], [1., 2., 3., 2., 5., 3., 8., 0., 3., 3., 3., 3., 3., 2., 1.], [2., 3., 2., 5., 3., 8., 0., 3., 3., 3., 3., 2., 1., 3.], [3., 2., 5., 3., 8., 0., 3., 3., 3., 3., 2., 1., 3., 2.], [3., 2., 5., 3., 8., 0., 3., 3., 3., 3., 2., 1., 3., 2., 4.], [2., 5., 3., 8., 0., 3., 3., 3., 2., 1., 3., 2., 4., 3.], [5., 3., 8., 0., 3., 3., 3., 2., 1., 3., 2., 4., 3., 1.], [3., 8., 0., 3., 3., 3., 2., 1., 3., 2., 4., 3., 1., 2.], [8., 0., 3., 3., 3., 2., 1., 3., 2., 4., 3., 1., 2., 1.], [0., 3., 3., 3., 3., 2., 1., 3., 2., 4., 3., 1., 2., 1., 0.], [3., 3., 3., 3., 2., 1., 3., 2., 4., 3., 1., 2., 1., 0., 0.], [3., 3., 3., 2., 1., 3., 2., 4., 3., 1., 2., 1., 0., 0., 2.], [3., 3., 2., 1., 3., 2., 4., 3., 1., 2., 1., 0., 0., 2., 0.], [3., 2., 1., 3., 2., 4., 3., 1., 2., 1., 0., 0., 2., 0., 1.], [2., 1., 3., 2., 4., 3., 1., 2., 1., 0., 0., 2., 0., 1., 5.]] | [[ 3.], [ 3.], [ 3.], [ 2.], [ 1.], [ 3.], [ 2.], [ 4.], [ 3.], [ 1.], [ 2.], [ 0.], [ 2.], [ 0.], [ 1.], [ 5.], [25.]] |
|                      | Test   |   |

### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

Ta thấy các mẫu có dự đoán sai số lớn đều có doanh số tháng cuối tăng đột biến, các mô hình dự đoán của chúng ta có **mức độ tổng quát** vừa đủ để hạn chế các trường hợp overfit, nhưng lại không đủ chi tiết để dự đoán cho các trường hợp đột biến như một số mẫu trên. Để cải thiện chất lượng dự đoán, ta bổ sung phương án dự đoán doanh số đối với 1 mẫu cụ thể bằng mô hình **DNN\_single\_time\_series**

```
def DNN_single_fit(time_series, horizon = 1):
    # Xây dựng mô hình
    model_DNN = tf.keras.Sequential([
        Dense(64, activation="relu"),
        Dense(32, activation="relu"),
        Dense(1, activation="linear")])
    # Biên dịch mô hình
    model_DNN.compile(loss="mae", optimizer=tf.keras.optimizers.Adam(),
                      metrics=[tf.keras.metrics.RootMeanSquaredError(name='rmse')])
    # Khớp mô hình
    model_DNN.fit(x=X_train, y=y_train,
                  epochs=500, verbose=0, batch_size=1, validation_data=(X_val, y_val),
                  callbacks=[tf.keras.callbacks.EarlyStopping(monitor="val_loss",
                                                patience=100, restore_best_weights=True)])
    return predict_with_WS
```

### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

Chạy mô hình DNN dự đoán chuỗi đơn với nhiều kích thước WS khác nhau và vẽ biểu đồ trực

quản hóa

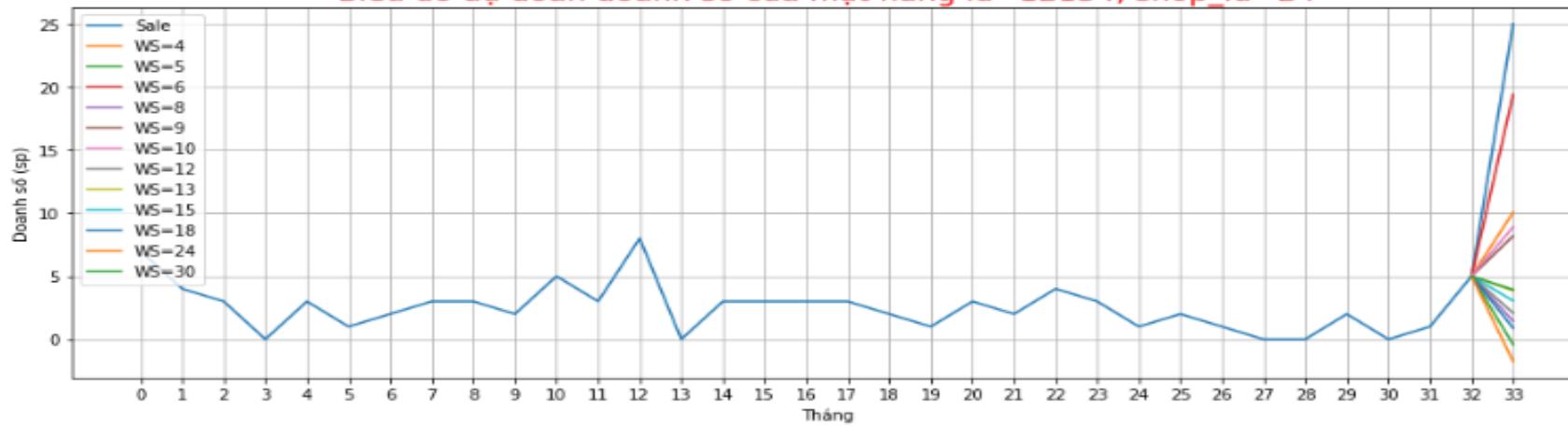
```
row_series = data.loc[id, 0: ].astype(np.float16).values
DNN_predict_series_with_WS = DNN_single_fit(row_series)
# Vẽ biểu đồ minh họa kết quả
plot_time_series(data.loc[id, 0: ], label='Sale')
for WS, value in DNN_predict_series_with_WS.items():
    plot_time_series(pd.Series(data=[data.loc[id, 32], value], index=[32, 33]),
                     label=f'WS={WS}')
plt.title(f'Biểu đồ dự đoán doanh số của mặt hàng id={item_id}, shop_id={shop_id}', fontsize = 18, color = 'red')
plt.legend('upper left');
```

### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

```
WS = 4, mse = 4.9690327644348145, y_pred = 10.05998
WS = 5, mse = 5.048611164093018, y_pred = -0.423091
WS = 6, mse = 4.610026836395264, y_pred = 19.405867
WS = 8, mse = 5.460358142852783, y_pred = 1.430526
WS = 9, mse = 5.67873477935791, y_pred = 8.189894
WS = 10, mse = 5.622107028961182, y_pred = 8.904579
WS = 12, mse = 5.845906734466553, y_pred = 2.067379
WS = 13, mse = 5.844075679779053, y_pred = 3.847542
WS = 15, mse = 7.322249412536621, y_pred = 3.045058
WS = 18, mse = 7.1055803298950195, y_pred = 0.880294
WS = 24, mse = 13.412101745605469, y_pred = -1.762187
WS = 30, mse = 22.48355484008789, y_pred = 3.92613
```

Row\_id = 1258

Biểu đồ dự đoán doanh số của mặt hàng id=12134, shop\_id=24

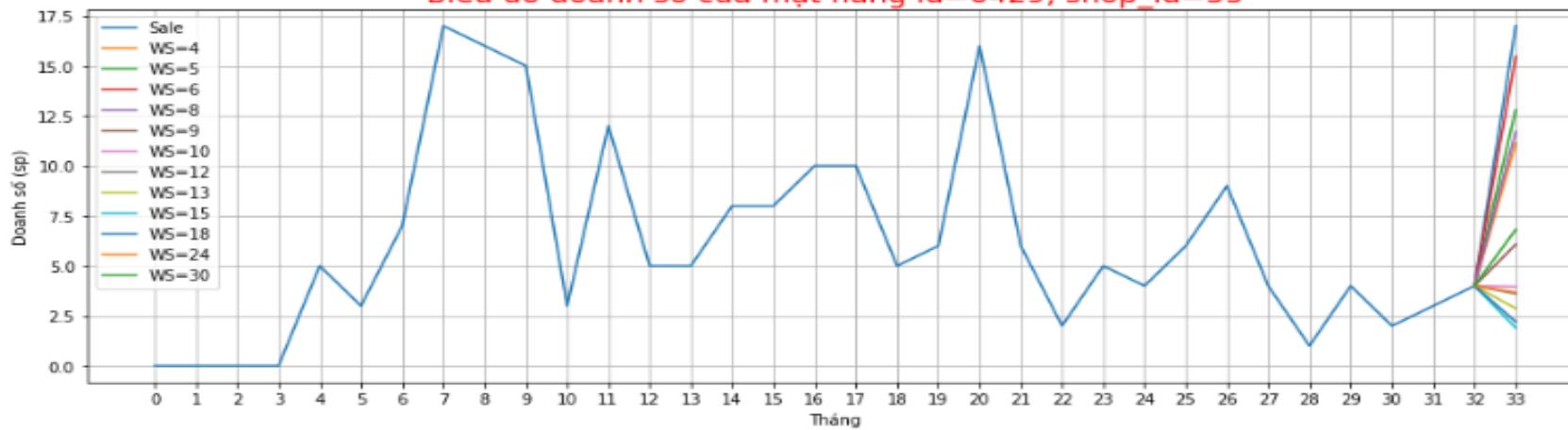


### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

```
WS = 4, mse = 2.775439500808716, y_pred = 11.136115
WS = 5, mse = 2.73844838142395, y_pred = 12.795407
WS = 6, mse = 2.8111751079559326, y_pred = 15.465499
WS = 8, mse = 3.517531633377075, y_pred = 11.710583
WS = 9, mse = 2.9801106452941895, y_pred = 6.071167
WS = 10, mse = 2.8801393508911133, y_pred = 3.960135
WS = 12, mse = 4.0892415046691895, y_pred = 3.608191
WS = 13, mse = 4.195652961730957, y_pred = 2.875553
WS = 15, mse = 4.4993000003051758, y_pred = 1.906225
WS = 18, mse = 4.4152984619140625, y_pred = 2.20387
WS = 24, mse = 7.292539596557617, y_pred = 3.682189
WS = 30, mse = 10.583730697631836, y_pred = 6.810515
```

Row\_id = 3771

Biểu đồ doanh số của mặt hàng id=6429, shop\_id=55



### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

Ta thấy mô hình **DNN dự đoán chuỗi riêng lẻ** sẽ cho các kết quả dự đoán sát thực tế hơn khi ta chọn được kích thước **Window Size** phù hợp. Tuy nhiên để thiết lập cho mỗi chuỗi thời gian một mô hình độc lập là khá tốn kém và không khả thi để có thể dự đoán cho toàn bộ các chuỗi mà cuộc thi đề ra.

Một số mô hình dự đoán chuỗi riêng lẻ khác bằng phương pháp **ARIMA, SARIMA** cũng có thể được sử dụng nhưng do mô hình cần đạt các điều kiện nhất định nên ta không đưa vào nội dung đồ án này (*Xem thêm trong phần source code ARIMA, SARIMA và kết quả dự đoán đi kèm*)

(Kết quả dự đoán của khoảng 1000 mô hình **DNN** ngẫu nhiên được lưu trong tập tin **DNN\_result\_df.csv**)

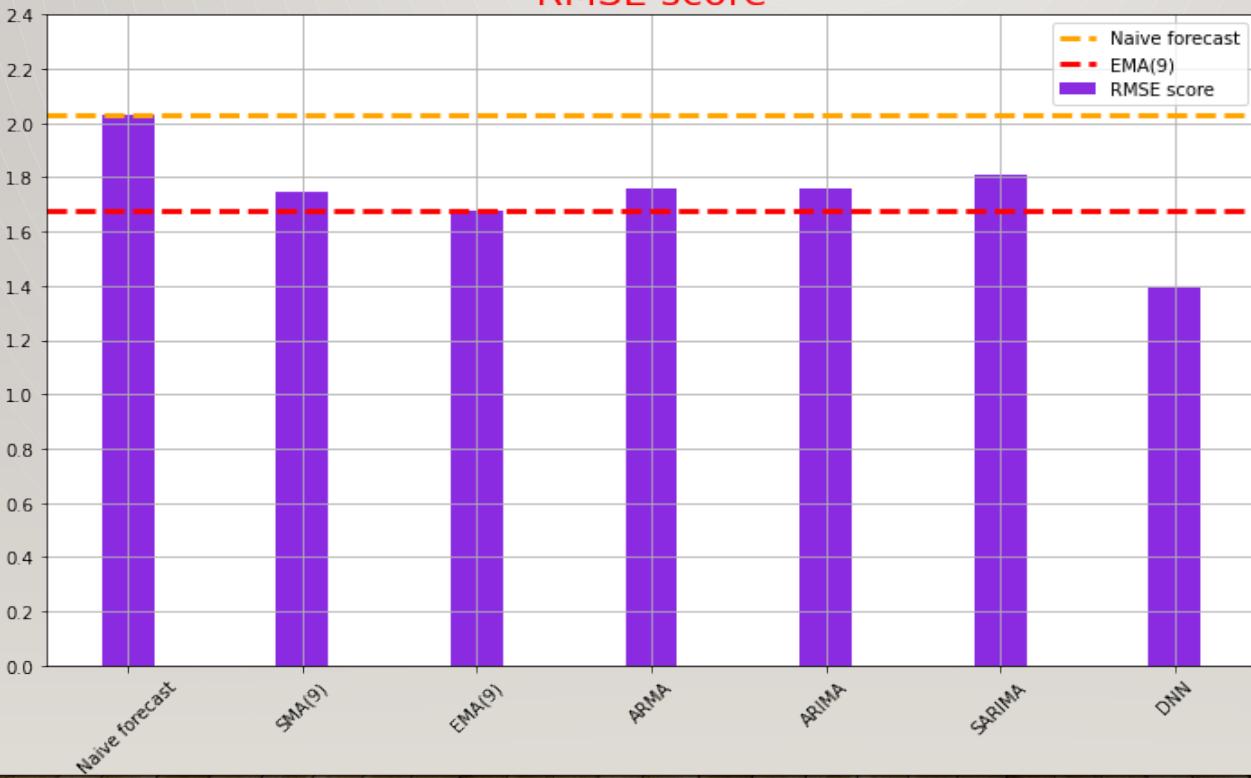
(Kết quả dự đoán của khoảng 1000 mô hình **ARMA** ngẫu nhiên được lưu trong tập tin **ARMA\_result\_df.csv**)

(Kết quả dự đoán của khoảng 1000 mô hình **ARIMA** ngẫu nhiên được lưu trong tập tin **ARIMA\_result\_df.csv**)

(Kết quả dự đoán của khoảng 1000 mô hình **SARIMA** ngẫu nhiên được lưu trong tập tin **SARIMA\_result\_df.csv**)

### 3.1. CẢI THIỆN MÔ HÌNH HIỆN TẠI

RMSE score



Bảng KQ lập mô hình  
cho từng chuỗi riêng

|                | rmse     |
|----------------|----------|
| Naive forecast | 2.029392 |
| SMA(9)         | 1.744606 |
| EMA(9)         | 1.677714 |
| ARMA           | 1.756987 |
| ARIMA          | 1.761397 |
| SARIMA         | 1.811252 |
| DNN            | 1.394935 |

Tuy nhiên kết quả DNN đã  
cải thiện được đáng  
kể.

# NGUỒN THAM KHẢO

## Link cuộc thi Kaggle

<https://www.kaggle.com/competitions/competitive-data-science-predict-future-sales/overview>

## Bài tham khảo time series

<https://www.kaggle.com/code/jagangupta/time-series-basics-exploring-traditional-ts>  
<https://forum.machinelearningcoban.com/t/moi-quan-he-danh-doi-giua-bias-va-variance/4173>

[https://machinelearningcoban.com/tabml\\_book/ch\\_data\\_processing/timeseries\\_data.html](https://machinelearningcoban.com/tabml_book/ch_data_processing/timeseries_data.html)

<https://online.stat.psu.edu/stat510/lesson/5/5.1>

<https://online.stat.psu.edu/stat510/lesson/5/5.2>

<https://phamdinhkhanh.github.io/2019/12/12/ARIMAmodeL.html>

[https://maths.uel.edu.vn/Resources/Docs/SubDomain/maths/TaiLieuHocTap/ToanUngDung/m\\_hnh\\_arima.html](https://maths.uel.edu.vn/Resources/Docs/SubDomain/maths/TaiLieuHocTap/ToanUngDung/m_hnh_arima.html)

<https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>

## Mô hình LSTM

<https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/>

## Mô hình XGBOOST

<https://xgboost.readthedocs.io/en/stable/parameter.html#parameters-for-tree-booster>  
[https://xgboost.readthedocs.io/en/stable/python/python\\_intro.html](https://xgboost.readthedocs.io/en/stable/python/python_intro.html)