

ĐẠI HỌC UEH  
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ  
KHOA: CÔNG NGHỆ THÔNG TIN KINH DOANH



ĐỒ ÁN MÔN HỌC: MÁY HỌC

Đề tài:  
DỰ ĐOÁN DOANH THU BÁN HÀNG

Thành viên:

Trần Mạnh Tường : 31201024515  
Bùi Thành Công : 31201024459  
Phạm Nhật Hải : 31201024472  
Lê Trung Nguyên : 31201024505

GVHD: TS. ĐẶNG NGỌC HOÀNG THÀNH

TP Hồ Chí Minh, Ngày 22 tháng 4 năm 2023

# MỤC LỤC

MỤC LỤC.....	1
Chương 1: Giới thiệu .....	3
1. Lý do chọn đề tài.....	3
2. Giới thiệu bài toán.....	3
Chương 2: Cơ sở lý thuyết .....	5
1. Cài đặt thư viện pygsheets .....	5
1.1 Giới thiệu .....	5
1.2 Cách cài đặt.....	5
2. Tổng quan về bài toán dự đoán chuỗi thời gian.....	11
3. Một số phương pháp dùng để dự đoán chuỗi thời gian .....	12
3.1. MA .....	12
3.2. EMA .....	12
3.3. Auto Regression.....	13
3.4. ARIMA .....	14
4. Các mô hình máy học .....	16
4.1. Linear Regression .....	16
4.2. ElasticNet .....	19
4.3. Gradient boosting .....	20
5. Mô hình học sâu.....	23
5.1. Giới thiệu về LSTM.....	23
5.2. Cấu trúc của mô hình .....	24
5.4. Ứng dụng.....	29
Chương 3: Bộ dữ liệu.....	30
Chương 4: Thực nghiệm và kết quả.....	31
1. Tiền xử lý .....	31

1.1. Xử lý thời gian bằng datetime.....	31
1.2. Xử lý các biến thông tin khách hàng .....	32
1.3. Tính tổng giá trị đơn hàng.....	33
1.4. Gom dữ liệu theo ngày bằng groupby .....	33
1.5. Tính phần trăm .....	34
1.6. Rolling feature.....	34
1.7. Tính RSI.....	35
1.8. Scaling data.....	36
1.9. Correlation .....	37
2. EDA.....	39
2.1. Time Series decomposition:.....	39
2.2. Moving Average (MA): .....	43
2.3. Exponential Moving Average (EMA): .....	44
2.4. Auto Regression (AR): .....	46
2.5. Autoregressive Integrated Moving Average (ARIMA): .....	47
3. Mô hình máy học .....	50
3.1. Linear Regression .....	50
3.2. ElasticNet Regression .....	54
3.3. Gradient Boosting Regression .....	55
4. Mô hình học sâu - Long Short-Term Memory .....	57
Chương 5: Kết luận .....	63
1. Kết luận.....	63
2. Hướng phát triển.....	63
PHỤ LỤC.....	64
TÀI LIỆU THAM KHẢO .....	64

# Chương 1: Giới thiệu

## 1. Lý do chọn đề tài

Báo cáo nhằm mục đích dự đoán doanh thu bán hàng là giúp doanh nghiệp có thể dự đoán được doanh thu của các tháng sau từ đó ước tính số lượng hàng hóa hoặc dịch vụ được bán ra trong tương lai, từ đó lập ra các kế hoạch sản xuất, cung ứng, kinh doanh, và quản lý các chiến lược marketing cũng như quản lý nguồn lực một cách hiệu quả.

Lý do cần xây dựng mô hình máy học hoặc học sâu dùng dự đoán doanh thu thay vì sử dụng các phương pháp dự đoán như MA, hay EMA,... là vì các phương pháp dự đoán truyền thống này chỉ xét tới giá và xu hướng của nó, không xét tới tác động của các biến phụ thuộc liên quan, ví dụ doanh thu của một ngày vẫn có khả năng chịu sự tác động của doanh thu những ngày trước đó, hoặc tệp khách hàng ngày liền trước,...

## 2. Giới thiệu bài toán

Bài toán dự đoán chuỗi thời gian ( Time Series Forecasting) là một bài toán trong lĩnh vực dự đoán, mục tiêu là dự đoán giá trị của một chuỗi dữ liệu theo thời gian, dựa trên các mẫu và xu hướng quá khứ của chuỗi dữ liệu đó, với giả định rằng các xu hướng trong tương lai sẽ tương tự như vậy.

Vậy tại sao cần dự đoán doanh thu:

- Trước hết, nó là một benchmark. Chúng ta có thể sử dụng nó như công việc kinh doanh sẽ đạt được nếu không thay đổi chiến lược. Hơn nữa, chúng ta có thể tính toán giá trị gia tăng của các hành động mới trên benchmark này.
- Tiếp đến, nó có thể được sử dụng để lập kế hoạch. Chúng ta có thể lập kế hoạch hoạt động cung và cầu của mình bằng cách xem các dự báo.
- Cuối cùng, nó là một hướng dẫn để chúng ta có thể lập ra kế hoạch ngân sách và mục tiêu.

Với sự phát triển của trí tuệ nhân tạo và máy móc đã có nhiều nghiên cứu sử dụng các phương pháp ước lượng để lựa chọn mô hình phù hợp như Regression, Neural Networks, Support Vector Machines, Random Forests và XGBoost,.. để phát triển mô hình dự đoán chuỗi thời gian. Những mô hình dự đoán này không chỉ mang lại

lợi ích cho các doanh nghiệp mà còn cả khách hàng vì nó sẽ giúp họ nhận biết sự cung ứng của sản phẩm hoặc dịch vụ mà họ cần, từ đó đảm bảo số lượng hàng hóa hoặc dịch vụ được cung cấp đúng thời điểm đồng thời tạo được sự tin tưởng giữ khách hàng và doanh nghiệp.

# Chương 2: Cơ sở lý thuyết

## 1. Cài đặt thư viện pygsheets

### 1.1 Giới thiệu

Đây là thư viện Python đơn giản, trực quan để truy cập spreadsheets thông qua Google Sheets API

### 1.2 Cách cài đặt

- B1: Cài đặt thư viện pygsheets vào máy bằng lệnh cmd

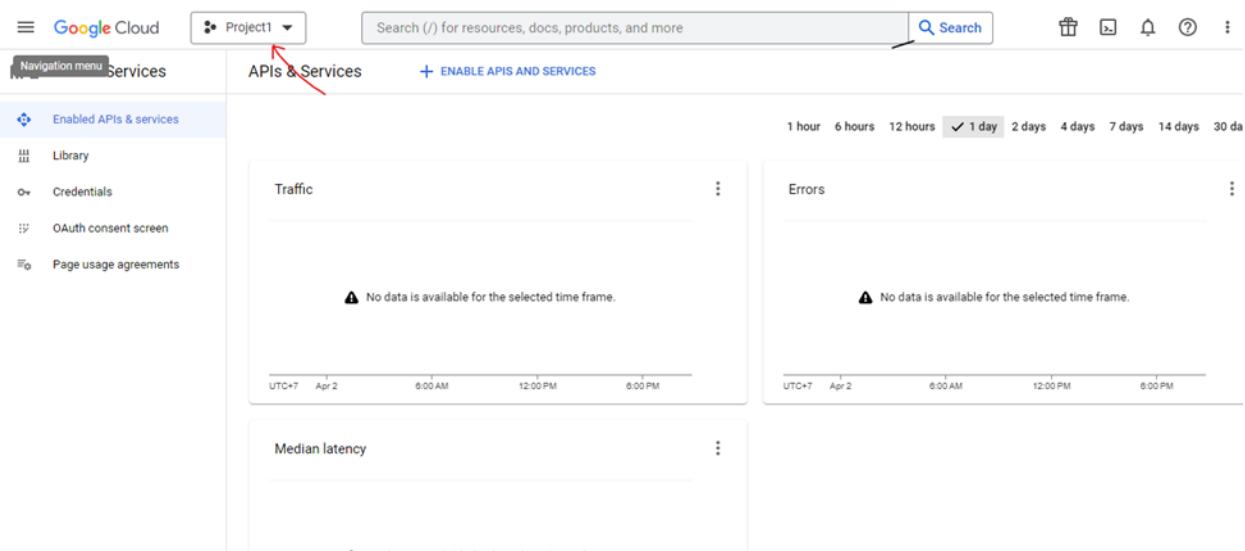
pip install pygsheets

- B2: Truy cập vào trang web sau và tiến hành kết nối thông qua Google API

<https://console.cloud.google.com/apis>

- B3: Tiến hành thực hiện từng bước bên dưới:

- Đầu tiên, tạo một project mới.



Select a project

NEW PROJECT

Search projects and folders

RECENT STARRED ALL

Name	ID
Project1	project1-378014

- Tiếp đó enable một số product trước khi sử dụng.

#### Product details



#### Google Drive API

[Google Enterprise API](#)

Create and manage resources in Google Drive.

ENABLE

TRY THIS API

#### Product details



#### Google Sheets API

[Google Enterprise API](#)

Read and write Google Sheets data

MANAGE

TRY THIS API

API Enabled

- Sau khi đã enable tất cả xong, đi về lại APIs overview và chọn Credentials.

APIs & Services

Credentials + CREATE CREDENTIALS DELETE RESTORE DELETED CREDENTIALS

Enabled APIs & services

Library

**Credentials** ←

OAuth consent screen

Page usage agreements

Create credentials to access your enabled APIs. [Learn more](#)

Remember to configure the OAuth consent screen with information about your application.

CONFIGURE CONSENT SCREEN

API Keys

Name	Creation date	Restrictions	Actions
No API keys to display			

OAuth 2.0 Client IDs

- Tiếp tục chọn Create credentials > Service Account

Credentials + CREATE CREDENTIALS DELETE RESTORE DELETED CREDENTIALS

Create credentials to access your enabled APIs

Remember to configure the OAuth consent screen with information about your application.

**Service account** ←

API key

Identifies your project using a simple API key to check quota and access

OAuth client ID

Requests user consent so your app can access the user's data

Service account

Enables server-to-server, app-level authentication using robot accounts

Help me choose

Asks a few questions to help you decide which type of credential to use

Restrictions

- Đặt tên và hoàn tất quá trình này

## 1 Service account details

Service account name

test

Display name for this service account

Service account ID \*

test-519

x c

Email address: test-519@project1-378014.iam.gserviceaccount.com 

Service account description

Describe what this service account will do

**CREATE AND CONTINUE**

## 2 Grant this service account access to project (optional)

## 3 Grant users access to this service account (optional)

**DONE**

CANCEL

- Lấy địa chỉ email bot mà GG cung cấp thông qua Service Account

The screenshot shows the Google Cloud Platform dashboard. At the top, there is a navigation bar with the Google Cloud logo, a dropdown for 'Project1' set to 'Project1', and a tab for 'google sheet'. Below the navigation bar, the main menu is visible, with 'Cloud overview' and 'View all products' options. A 'PINNED' section lists several services: API, APIs & Services, Billing, IAM & Admin (which is currently selected and highlighted in grey), Marketplace, Compute Engine, Kubernetes Engine, Cloud Storage, BigQuery, VPC network, Cloud Run, SQL, Security, and Google Maps Plat... . A dropdown menu for 'IAM & Admin' is open, showing options like IAM, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policies, Service Accounts (with a red arrow pointing to it), Workload Identity Federation, Labels, Tags, Settings, Privacy & Security, Identity-Aware Proxy, Roles, Audit Logs, Manage Resources, Create a Project, Essential Contacts, Asset Inventory, Quotas, and Groups.

Google Cloud

Project1

google sheet

Cloud overview >

View all products

PINNED

API APIs & Services >

Billing

IAM & Admin >

Marketplace

Compute Engine

Kubernetes Engine

Cloud Storage

BigQuery

VPC network

Cloud Run

SQL

Security

Google Maps Plat...

IAM

Identity & Organization

Policy Troubleshooter

Policy Analyzer

Organization Policies

Service Accounts

Workload Identity Federation

Labels

Tags

Settings

Privacy & Security

Identity-Aware Proxy

Roles

Audit Logs

Manage Resources

Create a Project

Essential Contacts

Asset Inventory

Quotas

Groups

Service accounts    + CREATE SERVICE ACCOUNT    DELETE    HELP ASSISTANT    LEARN

Service accounts for project "Project1"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more about service accounts.](#)

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. [Learn more about service account organization policies.](#)

<input type="checkbox"/> Email	Status	Name	Description	Key ID	Key creation date	Actions
<a href="#">ass-868@project1-378014.iam.gserviceaccount.com</a>		ass		No keys		
<a href="#">test-395@project1-378014.iam.gserviceaccount.com</a>		test		No keys		
<a href="#">test-519@project1-378014.iam.gserviceaccount.com</a>		test		No keys		

- Lấy một email vừa này tạo và gắn share trên Google Spreadsheets mà bạn muốn đẩy data lên
- Sau đó nhấn, Manage keys > Add Keys > Create new key

**Keys**

⚠ Service account keys could pose a security risk if compromised. We recommend you avoid downloading service account keys and instead use environment variables or service account credentials.

You can learn more about setting up service account keys.

Add a new key pair or upload an existing one.

Block service account key

[Learn more about setting up service account keys.](#)

**Create private key for "test"**

Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

**Key type**

**JSON**  
Recommended

**P12**  
For backward compatibility with code using the P12 format

**CANCEL**    **CREATE**

- Chọn Create và ta sẽ tải xuống 1 file json. Ở đây, ta chỉ cần thực hiện một số lệnh trong thư viện pygsheets là ta có thể upload dữ liệu ta mong muốn lên spreadsheet với Python.

## 2. Tổng quan về bài toán dự đoán chuỗi thời gian

Dự đoán chuỗi thời gian (time series forecasting) là quá trình dự đoán giá trị của một biến dữ liệu trong tương lai dựa trên các giá trị quá khứ của cùng một biến dữ liệu đó. Có nhiều phương pháp dự đoán chuỗi thời gian được sử dụng trong khoa học dữ liệu và dự đoán, và chúng dựa trên một số cơ sở chính sau đây:

- Tính chu kỳ: Một số chuỗi thời gian có tính chu kỳ, tức là chúng có một mô hình lặp lại theo chu kỳ nhất định trong dữ liệu, chẳng hạn như chu kỳ hàng ngày, hàng tuần, hàng tháng, hoặc hàng năm. Các phương pháp dự đoán chuỗi thời gian có thể sử dụng tính chu kỳ này để dự đoán các giá trị trong tương lai.
- Xu hướng và mùa vụ: Chuỗi thời gian thường có xu hướng tăng hoặc giảm theo thời gian, cũng như có sự biến động theo mùa vụ. Các phương pháp dự đoán chuỗi thời gian thường dựa trên việc xác định xu hướng và mùa vụ trong dữ liệu để dự đoán các giá trị trong tương lai.
- Tương quan: Chuỗi thời gian có thể bị ảnh hưởng bởi các yếu tố bên ngoài hoặc các biến khác trong hệ thống. Các phương pháp dự đoán chuỗi thời gian có thể sử dụng tương quan giữa chuỗi thời gian và các biến khác để dự đoán giá trị trong tương lai.
- Phân tích thống kê: Các phương pháp dự đoán chuỗi thời gian thường sử dụng các kỹ thuật phân tích thống kê để xác định mô hình dự đoán phù hợp, chẳng hạn như mô hình ARIMA (AutoRegressive Integrated Moving Average), mô hình GARCH (Generalized Autoregressive Conditional Heteroskedasticity), hay mô hình máy học như LSTM (Long Short-Term Memory) trong mạng nơ-ron hồi quy (Recurrent Neural Networks).
- Đánh giá và lựa chọn mô hình: Các phương pháp dự đoán chuỗi thời gian thường đánh giá và lựa chọn mô hình dự đoán tốt nhất dựa trên các tiêu chí như độ chính xác, có thể còn dựa trên những yếu tố khác như độ đơn giản của mô hình, khả năng diễn giải, tính khả thi trong việc triển khai, và khả năng xử lý dữ liệu lớn.

- Ngoài ra ta có thể dựa trên lịch sử dữ liệu: Chuỗi thời gian dự đoán dựa trên dữ liệu quá khứ, vì vậy lịch sử dữ liệu là một cơ sở quang trọng để dự đoán tương lai. Phân tích và đánh giá lịch sử dữ liệu là một phần quan trọng trong việc xây dựng mô hình dự đoán.
- Kiến thức chuyên gia: Kiến thức chuyên gia trong lĩnh vực liên quan cũng có thể dự đoán chuỗi thời gian. Sự hiểu biết về ngành, kiến thức về các yếu tố ảnh hưởng, dự báo thị trường, hoặc các sự kiện địa phương hoặc toàn cầu có thể sử dụng để dự đoán chuỗi thời gian hiệu quả

### **3. Một số phương pháp dùng để dự đoán chuỗi thời gian**

Mô hình dự đoán chuỗi thời gian truyền thống thường bao gồm hai thành phần chính là thành phần trend và thành phần seasonality. Thành phần trend mô tả xu hướng tăng hoặc giảm của chuỗi thời gian theo thời gian, trong khi thành phần seasonality mô tả các sự biến đổi định kỳ theo thời gian.

#### 3.1. MA

Phương pháp dự đoán bằng MA (Moving Average) là một trong những phương pháp đơn giản nhất để dự đoán chuỗi thời gian. Phương pháp này dựa trên việc tính toán giá trị trung bình của một số giá trị gần nhất trong chuỗi thời gian để dự đoán giá trị tiếp theo.

Cụ thể, phương pháp dự đoán bằng MA sử dụng một cửa sổ trượt (window) để tính toán giá trị trung bình của các giá trị trong cửa sổ này. Ví dụ, nếu ta chọn cửa sổ có kích thước là 5, thì giá trị dự đoán cho thời điểm tiếp theo sẽ là giá trị trung bình của 5 giá trị gần nhất.

Phương pháp này có thể được sử dụng để dự đoán chuỗi thời gian có tính chất định kỳ và ổn định. Tuy nhiên, phương pháp này không phù hợp để dự đoán các chuỗi thời gian có tính chất phi định kỳ và biến động mạnh.

#### 3.2. EMA

EMA (Exponential Moving Average) là một trong những biến thể của MA (Moving Average). Tuy nhiên, EMA sử dụng trọng số khác nhau cho các giá trị trong chuỗi thời gian, với trọng số giảm dần theo thời gian.

Cụ thể, để tính EMA, ta bắt đầu với giá trị trung bình động (SMA) của chuỗi thời gian trong cửa sổ xác định. Sau đó, ta tính toán giá trị EMA cho mỗi thời điểm bằng cách sử dụng công thức sau:

$$EMA(t) = \alpha * y(t) + (1 - \alpha) * EMA(t-1)$$

Trong đó:

- $EMA(t)$  là giá trị dự đoán cho thời điểm  $t$
- $y(t)$  là giá trị thực tế của chuỗi thời gian tại thời điểm  $t$
- $EMA(t-1)$  là giá trị EMA của thời điểm trước đó
- $\alpha (0 \leq \alpha \leq 1)$  là hệ số trọng số, được tính toán theo công thức:  $\alpha = 2 / (n + 1)$ , trong đó  $n$  là số lượng giá trị trong cửa sổ

Ta có thể thấy rằng, giá trị EMA cho thời điểm  $t$  phụ thuộc vào giá trị EMA của thời điểm trước đó và giá trị thực tế của chuỗi thời gian tại thời điểm  $t$ , với trọng số của giá trị thực tế giảm dần theo thời gian.

Phương pháp EMA thường được sử dụng để dự đoán các chuỗi thời gian có tính chất phi định kỳ và biến động mạnh, vì nó cho phép trọng số của các giá trị gần nhất giảm dần theo thời gian, giúp giảm thiểu ảnh hưởng của các giá trị nhiễu hoặc các giá trị bất thường trên dự đoán.

### 3.3. Auto Regression

Auto Regression (AR) là một trong những phương pháp dự đoán chuỗi thời gian phổ biến. AR dựa trên giả định rằng giá trị của chuỗi thời gian tại một thời điểm phụ thuộc vào các giá trị của chuỗi thời gian trong quá khứ.

Cụ thể, một mô hình AR( $p$ ) (Auto Regression of order  $p$ ) được xác định bằng công thức sau:

$$y(t) = c + \sum(\phi_i * y(t-i)) + e(t)$$

Trong đó:

- $y(t)$  là giá trị thực tế của chuỗi thời gian tại thời điểm  $t$
- $c$  là hằng số
- $\phi_i$  là các hệ số, đại diện cho mức độ ảnh hưởng của các giá trị trong quá khứ đến giá trị hiện tại, với  $i = 1, 2, \dots, p$
- $e(t)$  là giá trị nhiễu tại thời điểm  $t$

Với mô hình AR(p), ta cần xác định giá trị của các hệ số  $\phi_i$  và hằng số  $c$  để có thể dự đoán giá trị của chuỗi thời gian trong tương lai.

Cách phổ biến để xác định giá trị của các hệ số và hằng số này là sử dụng phương pháp tối thiểu hóa sai số bình phương (least squares method), với mục tiêu là tối thiểu hóa sai số giữa giá trị thực tế của chuỗi thời gian và giá trị dự đoán của mô hình AR(p).

AR thường được sử dụng để dự đoán các chuỗi thời gian có tính chất định kỳ, vì chúng ta có thể xác định một chu kỳ định kỳ trong chuỗi thời gian và sử dụng mô hình AR để dự đoán giá trị trong các chu kỳ tương lai. Tuy nhiên, AR có thể không hiệu quả cho các chuỗi thời gian có tính chất phi định kỳ và biến động mạnh, vì nó không xử lý được các yếu tố bất thường hoặc nhiễu trong chuỗi thời gian.

### 3.4. ARIMA

ARIMA (Autoregressive Integrated Moving Average) là một trong những phương pháp phổ biến nhất để dự đoán chuỗi thời gian, và nó là sự kết hợp của 3 phương pháp khác nhau: Auto Regression (AR), Integration (I), và Moving Average (MA).

- Auto Regression (AR): giống như mô hình AR đã nói ở trên, ARIMA sử dụng các giá trị của chuỗi thời gian trong quá khứ để dự đoán giá trị trong tương lai. AR sẽ dựa vào một số giá trị của chuỗi thời gian trong quá khứ để dự đoán giá trị tại thời điểm hiện tại.
- Integration (I): Chuỗi thời gian có thể không được ổn định và cần phải được chuyển đổi để trở nên ổn định. Quá trình chuyển đổi này được gọi là Integration, và nó liên quan đến tích hợp đạo hàm của chuỗi thời gian. Nếu chuỗi thời gian không ổn định, sử dụng ARIMA thay vì AR sẽ cải thiện đáng kể độ chính xác của mô hình.
- Moving Average (MA): MA cũng dự đoán giá trị của chuỗi thời gian tại một thời điểm dựa trên các giá trị của chuỗi thời gian trong quá khứ. MA sử dụng các sai số dự đoán (khác biệt giữa giá trị thực tế và giá trị dự đoán) trong quá khứ để dự đoán giá trị tương lai.

Với ARIMA, ta có thể xác định ba thông số quan trọng:

- $p$ : Số lượng lags (giá trị chuỗi thời gian trong quá khứ) được sử dụng cho AR model

- d: Số lần tích hợp (integration) được áp dụng cho chuỗi thời gian để trở nên ổn định
- q: Số lượng lags (sai số dự đoán trong quá khứ) được sử dụng cho MA model
- Thông thường, ta sử dụng các phương pháp như ACF (Auto Correlation Function) và PACF (Partial Auto Correlation Function) để xác định giá trị của p và q. Giá trị của d phụ thuộc vào tính ổn định của chuỗi thời gian, và ta cần sử dụng kiểm định ADF (Augmented Dickey-Fuller) để xác định giá trị của d. Sau khi xác định được các giá trị của p, d, và q, ta có thể sử dụng ARIMA để dự đoán giá trị của chuỗi thời gian

### 3.5. Mô hình máy học và học sâu

Ngoài các phương pháp truyền thống như MA, EMA, AR, ARIMA, có thể sử dụng các mô hình máy học và học sâu để dự đoán chuỗi thời gian.

Các mô hình máy học và học sâu có nhiều ưu điểm so với các phương pháp truyền thống trong xử lý bài toán dự đoán chuỗi thời gian, bao gồm:

- **Khả năng xử lý dữ liệu phi tuyến tính:** Các mô hình máy học và học sâu có khả năng xử lý dữ liệu phi tuyến tính và học các mẫu phức tạp hơn trong dữ liệu, giúp cải thiện độ chính xác của mô hình dự đoán.
- **Khả năng học các mối quan hệ phức tạp trong dữ liệu:** Các mô hình máy học và học sâu có khả năng học được các mối quan hệ phức tạp giữa các biến và các yếu tố không rõ ràng trong dữ liệu.
- **Tính linh hoạt:** Các mô hình máy học và học sâu có khả năng thích nghi với nhiều loại dữ liệu và có thể được tùy chỉnh để phù hợp với nhiều loại bài toán.
- **Khả năng xử lý dữ liệu lớn:** Các mô hình máy học và học sâu có thể xử lý được các tập dữ liệu lớn và phức tạp, giúp cải thiện độ chính xác và độ tin cậy của mô hình dự đoán.

Tuy nhiên, các mô hình máy học và học sâu cũng có một số nhược điểm, bao gồm:

- **Khả năng overfitting:** Do số lượng tham số của các mô hình máy học và học sâu rất lớn, nên chúng có nguy cơ bị overfitting với các tập dữ liệu nhỏ hoặc không đại diện cho dữ liệu thực tế.

- Khó hiểu và khó giải thích: Các mô hình máy học và học sâu có thể khó hiểu và khó giải thích vì chúng không đơn giản như các phương pháp truyền thống và có quá nhiều tham số để giải thích.

## 4. Các mô hình máy học

### 4.1. Linear Regression

Là một phương pháp thống kê để hồi quy dữ liệu với biến phụ thuộc có giá trị liên tục trong khi các biến độc lập có thể có một trong hai giá trị liên tục hoặc là giá trị phân loại. Nói cách khác "Hồi quy tuyến tính" là một phương pháp để dự đoán biến phụ thuộc y dựa trên giá trị của biến độc lập X. Nó có thể được sử dụng cho các trường hợp chúng ta muốn dự đoán một số lượng liên tục

$$y \approx f(\mathbf{x}) = \hat{y}$$

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + w_3x_3 + w_0 \quad (*)$$

Hay nó một cách đơn giản chúng sẽ như thế này

$$\text{Dữ liệu} = \text{Hồi quy (Regression)} + \text{Phản dư (Residual)}$$

Vậy phản dư là gì

Là khác biệt giữa giá trị quan sát và giá trị dự đoán . Khi chúng ta xem xét một biểu đồ hồi quy, phản dư là khoảng cách từ điểm dữ liệu đến đường hồi quy được fit

Quay trở lại với (\*)

trong đó,  $w_1, w_2, w_3, w_0$  là các hằng số,  $w_0$  còn được gọi là bias. Mỗi quan hệ  $y \approx f(\mathbf{x}) = \hat{y}$  bên trên là một mối quan hệ tuyến tính (linear). Bài toán chúng ta đang làm là một bài toán thuộc loại regression. Bài toán đi tìm các hệ số tối ưu  $\{w_1, w_2, w_3, w_0\}$  chính vì vậy được gọi là bài toán Linear Regression

$y$  là giá trị thực của outcome (dựa trên số liệu thông kê chúng ta có trong tập training data), trong khi  $\hat{y}$  là giá trị mà mô hình Linear Regression dự đoán được.

Nhìn chung,  $y$  và  $\hat{y}$  là hai giá trị khác nhau do có sai số mô hình, tuy nhiên, chúng ta mong muốn rằng sự khác nhau này rất nhỏ

Linear hay tuyến tính hiểu một cách đơn giản là thẳng, phẳng. Trong không gian hai chiều, một hàm số được gọi là tuyến tính nếu đồ thị của nó có dạng một đường thẳng. Trong không gian ba chiều, một hàm số được gọi là tuyến tính nếu đồ thị của nó có dạng một mặt phẳng. Trong không gian nhiều hơn 3 chiều, khái niệm mặt phẳng không còn phù hợp nữa, thay vào đó, một khái niệm khác ra đời được gọi là siêu mặt phẳng (hyperplane). Các hàm số tuyến tính là các hàm đơn giản nhất, vì chúng thuận tiện trong việc hình dung và tính toán

Và có một lưu ý là nó hoàn toàn khá nhạy cảm với outliers

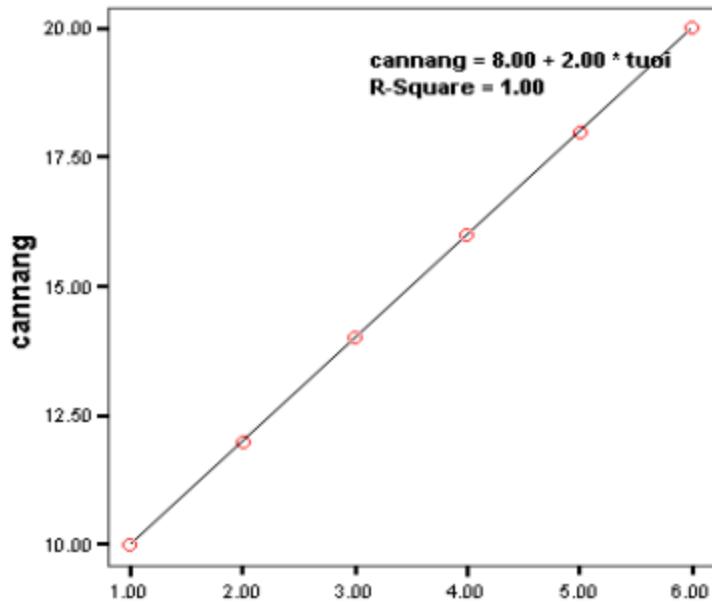
Và từ công thức trên, ta hoàn toàn có một phương trình đường thẳng như sau:

$$y = mx + b$$

trong đó độ dốc (slope) "m" và hệ số chặn y (y-intercept) "b". Hệ số chặn y "b" chỉ đơn giản là giá trị của "y" khi  $x = 0$ . Độ dốc "m" được tính từ hai điểm bất kỳ ( $x_1, y_1$ ) và ( $x_2, y_2$ ) là:  $m = (y_2 - y_1) / (x_2 - x_1)$

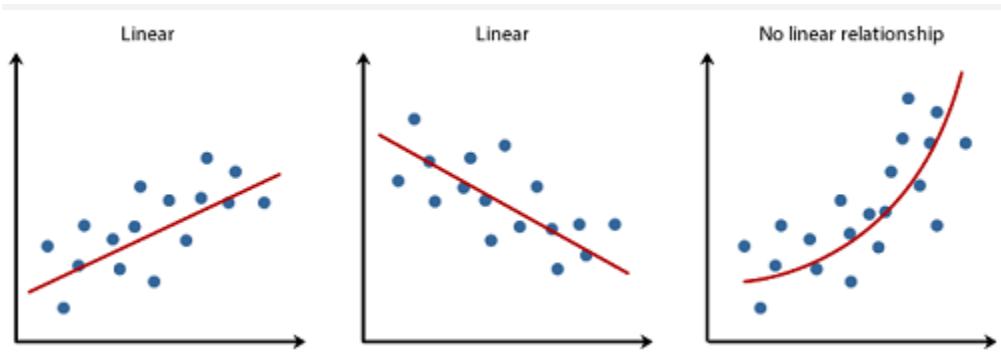
Ví dụ: Ta có 1 mẫu gồm 6 trẻ từ 1-6 tuổi, có cân nặng như bảng sau:

Tuổi	Cân nặng (kg)
1	10
2	12
3	14
4	16
5	18
6	20



Nối các cặp (x,y) này ta thấy có dạng 1 phương trình bậc nhất:  $y=2x+8$

Sau ví dụ này, nhóm em sẽ phân loại đường hồi quy



Với đồ thị thứ nhất, mô hình là hoàn toàn tuyến tính nhưng là tuyến tính thuận, tức là khi x tăng thì y cũng tăng và ngược lại

Còn với đồ thi thứ hai thì lại là tuyến tính nghịch, khi đó x tăng sẽ làm y giảm và ngược lại

Còn với mô hình thứ ba là mô hình phi tuyến tính

Đến với mô hình tuyến tính, trong mô hình tuyến tính sẽ được chia làm hai loại mô hình: Simple Linear Regression và Multiple Linear Regression

Simple Linear Regression

Là tìm sự liên hệ giữa 2 biến số liên tục: biến độc lập (biến dự đoán) trên trục hoành x với biến phụ thuộc (biến kết cục) trên trục tung y. Sau đó vẽ một đường thẳng hồi quy và từ phương trình đường thẳng này ta có thể dự đoán được biến y

### Multiple Linear Regression

Dùng để sử dụng khi nhiều biến độc lập x để dự đoán một biến phụ thuộc y, phương pháp này dùng để giải thích mối quan hệ giữa một biến phụ thuộc và nhiều biến độc lập

Và ứng dụng của hồi quy trong thực tế cũng khá là phổ biến:

Dự đoán giá nhà, dự đoán tài chính, dự đoán liều lượng thuốc, dự đoán sản lượng và năng suất cây trồng, dự đoán chi phí phần mềm, chi phí đảm bảo chất lượng phần mềm, ...

### 4.2. ElasticNet

Hồi quy ElasticNet là một phương pháp kết hợp của cả Lasso Regression và Ridge Regression, để tận dụng các ưu điểm của cả hai phương pháp. Hồi quy ElasticNet có thể loại bỏ các biến không cần thiết hoặc giảm giá trị của tất cả các hệ số hồi quy tùy thuộc vào đặc tính của tập dữ liệu đầu vào.

Lasso Regression và Ridge Regression là hai phương pháp hồi quy khác nhau, cùng được sử dụng để giảm thiểu overfitting trong mô hình hồi quy tuyến tính.

Lasso Regression, còn được gọi là L1 regularization, giúp đưa các hệ số của biến không cần thiết về 0 trong phương trình hồi quy, dẫn đến việc giảm số lượng biến đầu vào của mô hình. Lasso sử dụng hàm mất mát để tối thiểu hóa tổng giá trị tuyệt đối của các hệ số hồi quy, và do đó, các biến không cần thiết có thể được loại bỏ khỏi mô hình.

Ridge Regression, còn được gọi là L2 regularization, tương tự như Lasso Regression, giúp giảm thiểu overfitting bằng cách thêm một thành phần giảm thiểu vào phương trình mục tiêu. Tuy nhiên, thay vì sử dụng giá trị tuyệt đối của hệ số hồi quy, Ridge Regression sử dụng giá trị bình phương của chúng. Điều này dẫn đến việc mô hình sẽ giảm giá trị của tất cả các hệ số hồi quy, chứ không phải loại bỏ các biến không cần thiết.

Hồi quy ElasticNet sử dụng cả hai thành phần regularization (giảm thiểu overfitting) là L1 regularization (dùng trong Lasso Regression) và L2

regularization (dùng trong Ridge Regression). Điều này giúp giảm thiểu các ảnh hưởng của các biến không cần thiết trong quá trình huấn luyện mô hình.

Cụ thể, phương trình hồi quy ElasticNet là:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p + \varepsilon$$

Trong đó:

- $y$  là biến phụ thuộc (được dự đoán)
- $x_1, x_2, \dots, x_p$  là các biến độc lập
- $\beta_0, \beta_1, \beta_2, \dots, \beta_p$  là các hệ số hồi quy cần tìm
- $\varepsilon$  là sai số ngẫu nhiên

Phương pháp này giải quyết vấn đề overfitting của mô hình hồi quy thông thường bằng cách đưa thêm hai thành phần regularization L1 và L2 vào phương trình mục tiêu. Kết quả là hồi quy ElasticNet sẽ giúp tìm ra một tập hợp các hệ số hồi quy tốt hơn, giảm thiểu ảnh hưởng của các biến không cần thiết trong mô hình, giúp cải thiện độ chính xác và khả năng tổng quát hóa của mô hình.

#### 4.3. Gradient boosting

Gradient Boosting là một dạng tổng quát hóa của AdaBoost và thường được sử dụng cho phân loại và hồi quy. Nó đưa ra một mô hình dự đoán dưới dạng một tập hợp các mô hình dự đoán yếu, thường là các cây quyết định (decision tree). Khi cây quyết định là yếu, thuật toán kết quả được gọi là gradient-boosted trees; nó thường vượt trội so với random forest. Mô hình gradient-boosted trees được xây dựng tốt như trong các phương pháp boosting khác, nhưng nó tổng quát hóa các phương pháp khác bằng cách cho phép tối ưu hóa tùy ý hàm mất khả vi

Giống như các phương pháp boosting khác, nó tăng cường độ dốc kết hợp cách build model yếu thành một model mạnh theo kiểu lặp đi lặp lại. Và nó thường được áp dụng trong hồi quy bình phương nhỏ nhất, trong đó mục tiêu là build model  $F$  để dự đoán các giá trị của  $\hat{y} = F(x)$  bằng cách giảm thiểu mean squared error (MSE). Công thức tổng quát

$$\min_{c_n, w_n} L(y, W_{n-1} + c_n w_n))$$

Ý tưởng

Phương pháp Gradient Boosting giả định giá trị thực  $y$ . Nó tìm kiếm một hàm  $\hat{F}(x)$  và trong hàm đó sẽ xuất hiện một hàm  $h_m(x)$  từ một mô hình học máy khác (và mô hình này sẽ có kết quả không được tốt)

$$\hat{F}(x) = \sum_{m=1}^M \gamma_m h_m(x) + \text{const.}$$

Và sẽ có một bộ dữ liệu train của các giá trị mẫu đã biết của  $x$  và các giá trị tương ứng của  $y$ . Phương pháp này cố gắng tìm ra hàm  $\hat{F}(x)$  và làm giảm thiểu giá trị trung bình của hàm mất mát trên tập huấn luyện, tức là giảm thiểu rủi ro thực nghiệm. Nó làm như vậy bằng cách bắt đầu với một mô hình, bao gồm một hàm hằng và dần dần mở rộng nó:

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma),$$

$$F_m(x) = F_{m-1}(x) + \left( \arg \min_{h_m \in \mathcal{H}} \left[ \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + h_m(x_i)) \right] \right)(x),$$

Ý tưởng tiếp theo là áp dụng steepest descent (một chức năng có trong Gradient Boosting) cho vấn đề tối thiểu hóa này. Steepest descent là tìm cực tiểu cục bộ của hàm mất mát bằng cách lặp lại trên  $F_{m-1}(x)$ . Trên thực tế, hướng giảm dần cục bộ của hàm mất mát là độ dốc âm

Và sau đó sẽ có được hàm như sau:

$$F_m(x) = F_{m-1}(x) - \gamma \sum_{i=1}^n \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))$$

### Gradient tree boosting

Là một thuật toán học máy được sử dụng để đưa ra các dự đoán cho các vấn đề liên quan đến dự đoán. Thuật toán này kết hợp giữa việc sử dụng các cây quyết định (decision trees) và gradient boosting, trong đó việc xây dựng các cây quyết định được tối ưu hóa theo hướng giảm độ lỗi của mô hình. Gradient tree boosting là một phương pháp hiệu quả để giải quyết các vấn đề liên quan đến dự đoán và thường được sử dụng trong các bài toán như phân loại và dự báo

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}_{R_{jm}}(x), \quad \gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma).$$

## Regularized Gradient Boosting

Là một phương pháp cải tiến của Gradient Boosting, trong đó sử dụng thêm các kỹ thuật regularization nhằm giảm thiểu hiện tượng overfitting. Các kỹ thuật regularization được sử dụng bao gồm:

- L1 regularization (Lasso regularization) để đưa các trọng số gần bằng 0 và loại bỏ các đặc trưng không quan trọng.
- L2 regularization (Ridge regularization) để giảm tổng bình phương các trọng số để tránh các trọng số quá lớn.
- Early stopping để dừng quá trình training khi độ chính xác trên tập validation không cải thiện nữa.
- Dropout để ngẫu nhiên bỏ đi một số nơ-ron trong quá trình training.

Tất cả các kỹ thuật này giúp giảm thiểu thiểu sót trong dữ liệu huấn luyện và giúp mô hình trở nên ổn định và chính xác hơn

## Ưu điểm của Gradient Boosting:

- Hiệu suất cao: Gradient Boosting là một trong những thuật toán học máy tốt nhất hiện nay, có thể đạt được độ chính xác cao và hiệu quả tốt với các tập dữ liệu lớn.
- Tính tổng quát cao: Gradient Boosting là một thuật toán tập trung vào việc tối ưu hóa mô hình cho toàn bộ tập dữ liệu, thay vì chỉ tập trung vào từng điểm dữ liệu riêng lẻ. Điều này giúp tăng tính tổng quát của mô hình.
- Khả năng xử lý dữ liệu có nhiễu: Gradient Boosting có khả năng xử lý dữ liệu có nhiễu tốt hơn các thuật toán khác.
- Dễ dàng triển khai: Gradient Boosting thường được hỗ trợ bởi các thư viện học máy như Scikit-Learn hay XGBoost, vì vậy triển khai và sử dụng Gradient Boosting

Một số nhược điểm của Gradient Boosting là:

- Overfitting: Gradient Boosting có thể dẫn đến overfitting nếu không được điều chỉnh đúng cách. Điều này xảy ra khi mô hình quá phức tạp và tập dữ liệu huấn luyện quá nhỏ.
- Tốc độ huấn luyện: Gradient Boosting tốn nhiều thời gian hơn để huấn luyện so với các mô hình học máy khác. Điều này là do quá trình tìm kiếm các cây quyết định phù hợp.
- Cân định cấu hình thủ công: Để đạt được hiệu suất tốt, Gradient Boosting yêu cầu nhiều định cấu hình thủ công. Điều này có thể làm cho mô hình trở nên khó khăn để sử dụng đối với những người mới bắt đầu trong lĩnh vực học máy.
- Nhạy cảm với nhiễu: Gradient

Gradient Boosting là một kỹ thuật học máy được sử dụng rộng rãi để giải quyết các vấn đề liên quan đến dự đoán và phân loại. Đây là một phương pháp ensemble learning, nghĩa là nó kết hợp nhiều mô hình để cải thiện độ chính xác dự đoán.

Các ứng dụng của Gradient Boosting bao gồm:

- Dự đoán giá cổ phiếu: Gradient Boosting có thể được sử dụng để dự đoán giá cổ phiếu bằng cách sử dụng các feature như giá trị thị trường, chỉ số kinh doanh, dữ liệu tài chính,...
- Phân loại ảnh: Gradient Boosting có thể được sử dụng để phân loại ảnh bằng cách sử dụng các feature như hình dạng, kích thước, màu sắc của các đối tượng trong ảnh.
- Xử lý ngôn ngữ tự nhiên:

## 5. Mô hình học sâu

### 5.1. Giới thiệu về LSTM

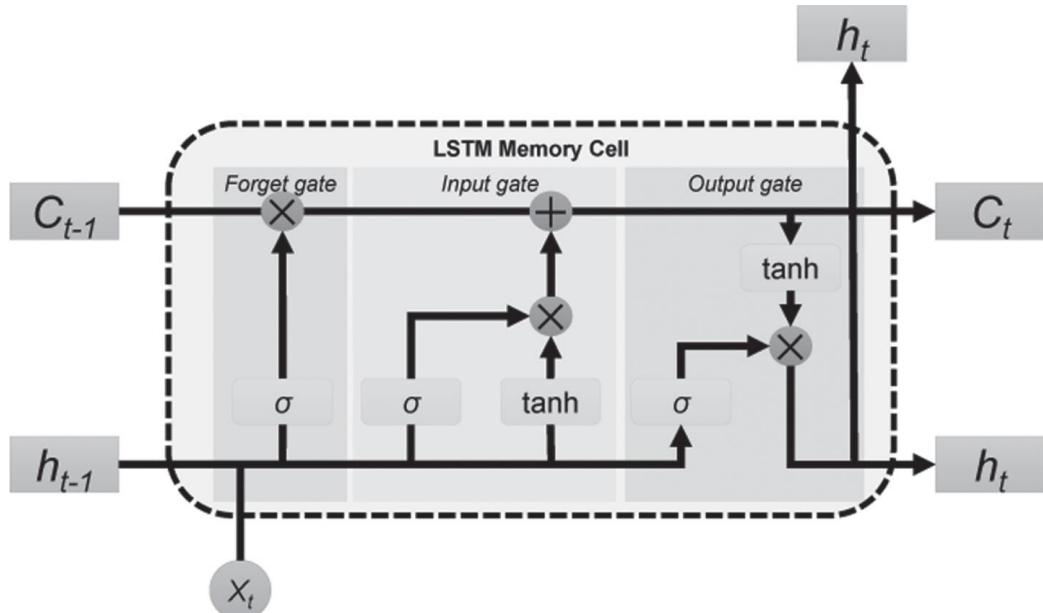
LSTM là một dạng đặc biệt của mạng nơ-ron hồi quy (RNN - Recurrent Neural Network) với khả năng giải quyết được vấn đề đuôi theo độ dài phụ thuộc (vanishing gradient problem) trong RNN, được đề xuất vào năm 1997 và được phát triển bởi Sepp Hochreiter và Jürgen Schmidhuber.

- Năm 1982: Paul Werbos đề xuất mạng nơ-ron hồi quy (RNN) với khả năng giữ lại thông tin liên quan trong dữ liệu chuỗi, tuy nhiên, mạng RNN gốc vẫn tồn tại vấn đề đuôi theo độ dài phụ thuộc, làm giảm khả năng học và dự đoán đúng.

- Năm 1991: Lần đầu tiên ý tưởng của LSTM được đề cập đến bởi công trình của Gers và Schmidhuber, trong đó họ giới thiệu kiến trúc RNN với một thành phần quan trọng là cổng (gate) để điều chỉnh luồng thông tin.
- Năm 1997: Hochreiter và Schmidhuber công bố công trình về LSTM, đưa ra một kiến trúc hoàn thiện của mạng LSTM gồm các cổng đầu vào (input gate), cổng quên (forget gate), và cổng đầu ra (output gate). Các cổng này cho phép mạng LSTM có khả năng điều chỉnh luồng thông tin trong quá trình lan truyền thuận và lan truyền ngược, giúp tránh vấn đề đuôi theo độ dài phụ thuộc.
- Năm 2000: Bengio, Simard và Frasconi tiếp tục cải tiến LSTM bằng việc đưa ra một kiến trúc biến thể của LSTM gọi là mạng nơ-ron hồi quy hoạt hóa (LSTM-activation), tận dụng tính chất của các hàm hoạt hóa để điều chỉnh thông tin.
- Từ năm 2000 đến nay: Công nghệ LSTM đã được tiếp tục nghiên cứu và phát triển, cùng với các biến thể như LSTM kép (bidirectional LSTM), LSTM nhúng (embedding LSTM), LSTM đa tầng)

## 5.2. Cấu trúc của mô hình

### 5.2.1. Cấu trúc của mô hình Long Short-Term Memory

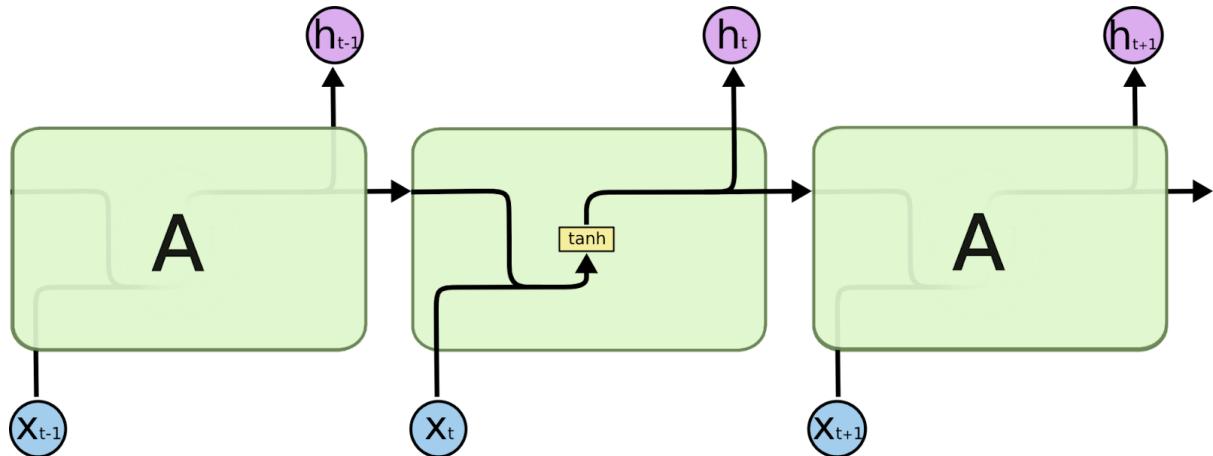


Hình: Mô hình LSTM

LSTM dựa trên một số khái niệm chính, bao gồm:

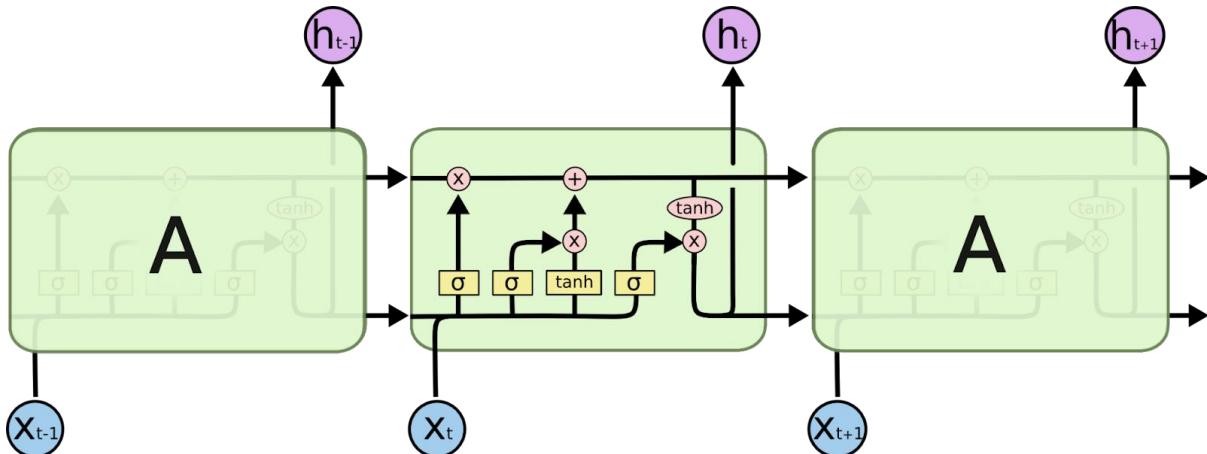
- Cảng vào (Input gate): LSTM sử dụng một cảng vào để quyết định thông tin mới nào sẽ được đưa vào trong ô nhớ của LSTM. Cảng vào được điều khiển bởi một hàm kích hoạt sigmoid, giúp xác định mức độ chấp nhận của thông tin mới.
- Cảng quên (Forget gate): LSTM sử dụng một cảng quên để quyết định thông tin cũ nào sẽ được loại bỏ khỏi ô nhớ của LSTM. Cảng quên cũng được điều khiển bởi một hàm kích hoạt sigmoid, giúp xác định mức độ quên của thông tin cũ.
- Cảng ra (Output gate): LSTM sử dụng một cảng ra để quyết định thông tin nào sẽ được đưa ra từ ô nhớ của LSTM. Cảng ra được điều khiển bởi một hàm kích hoạt sigmoid, giúp xác định mức độ đóng góp của thông tin trong ô nhớ đến đầu ra của LSTM.
- Ô nhớ (Cell state): LSTM duy trì một ô nhớ để lưu trữ thông tin trạng thái đọc theo chuỗi đầu vào. Ô nhớ có thể được cập nhật thông qua cảng vào, cảng quên và các phép toán tuyến tính.
- Hàm kích hoạt tanh (Tanh activation): LSTM sử dụng hàm kích hoạt tanh để biến đổi các giá trị đầu vào và đầu ra, giúp giới hạn giá trị đầu ra của LSTM trong khoảng  $[-1, 1]$ .
- Các thành phần này cho phép LSTM có khả năng tự học và lưu trữ thông tin dài hạn trong ô nhớ của nó, đồng thời điều chỉnh mức độ đóng góp của các thông tin mới và cũ vào đầu ra của LSTM. Đây là những cơ sở lý thuyết cơ bản của LSTM, giúp nó trở thành một kiến trúc mạng nơ-ron mạnh mẽ trong xử lý dữ liệu chuỗi.

### 5.2.2. Sự khác biệt giữa cấu trúc LSTM và cấu trúc RNN



*Hình: Sự lặp lại kiến trúc module trong mạng RNN chứa một tầng ẩn*

Như hình trên ta thấy được một mạng RNN tiêu chuẩn chỉ gồm một tầng ẩn và hàm tanh như hình điêu này có thể gây khó khăn trong việc nhớ lại thông tin từ quá khứ khi chuỗi có độ dài lớn, cũng như việc học được các quy luật phức tạp trong dữ liệu dẫn đến việc nhớ lại không chính xác. Khi chỉ có một ẩn đây cũng là khó khăn trong việc giải quyết vấn đề gradient biến mất hoặc gradient bùng nổ

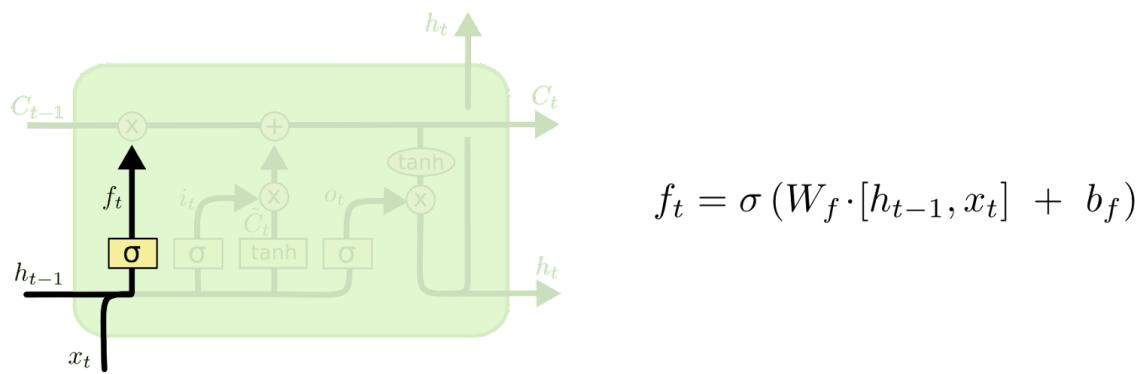


*Hình: Sự lặp lại kiến trúc module trong mạng LSTM chứa 4 tầng ẩn (3 sigmoid và 1 tanh) tương tác*

Với mạng LSTM ở trên đã có thể thấy rõ được sự khác biệt với mạng RNN, điều này có thể giúp cải thiện kết quả đầu ra của mô hình cũng như bù đắp vào những thiếu sót của mô hình trên.

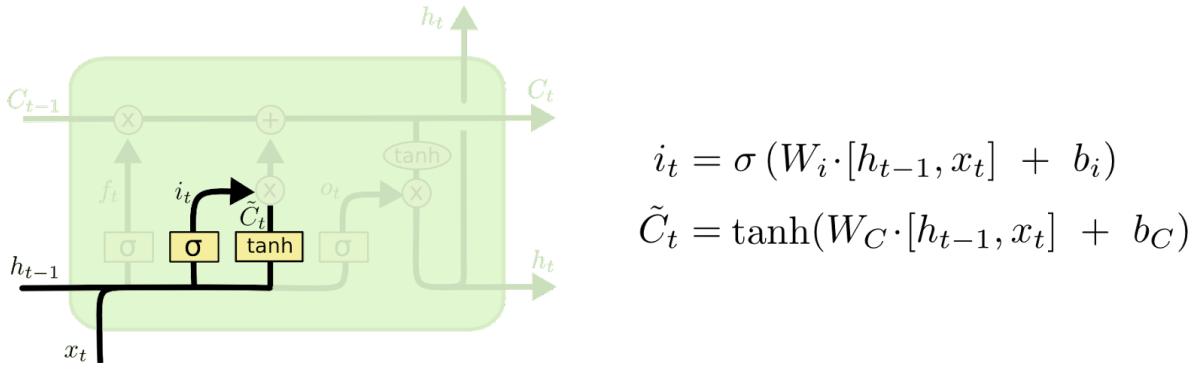
### 5.3. Quy trình thực hiện

Bước đầu tiên trong LSTM sẽ quyết định xem thông tin nào chúng ta sẽ cho phép đi qua ô trạng thái (cell state). Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (forget gate layer). Đầu tiên nó nhận đầu vào là 2 giá trị  $h_{t-1}$  và  $x_t$  và trả về một giá trị nằm trong khoảng 0 và 1 cho mỗi giá trị của ô trạng thái  $C_{t-1}$ . Nếu giá trị bằng 1 thể hiện ‘giữ toàn bộ thông tin’ và bằng 0 thể hiện ‘bỏ qua toàn bộ chúng’.



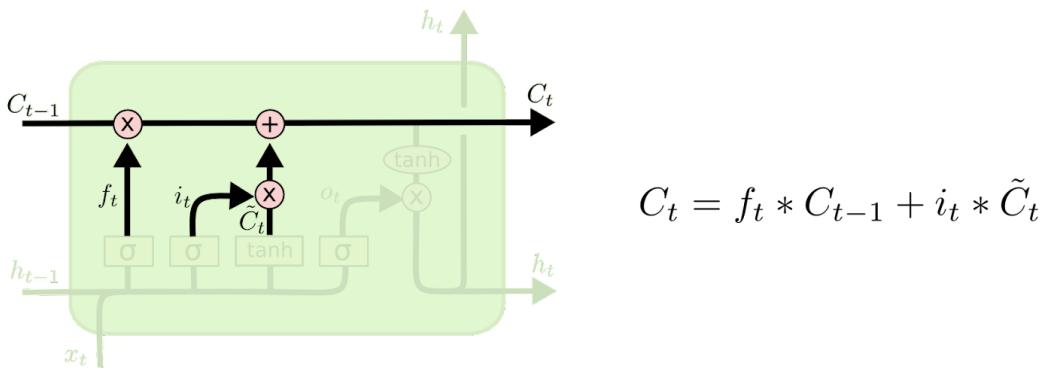
Hình: Tầng cổng quên (forget gate layer)

Bước tiếp theo chúng ta sẽ quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái. Bước này bao gồm 2 phần. Phần đầu tiên là một tầng ẩn của hàm sigmoid được gọi là tầng cổng vào (input gate layer) quyết định giá trị bao nhiêu sẽ được cập nhật. Tiếp theo, tầng ẩn hàm tanh sẽ tạo ra một vectơ của một giá trị trạng thái mới  $\tilde{C}_t$  mà có thể được thêm vào trạng thái. Tiếp theo kết hợp kết quả của 2 tầng này để tạo thành một cập nhật cho trạng thái.



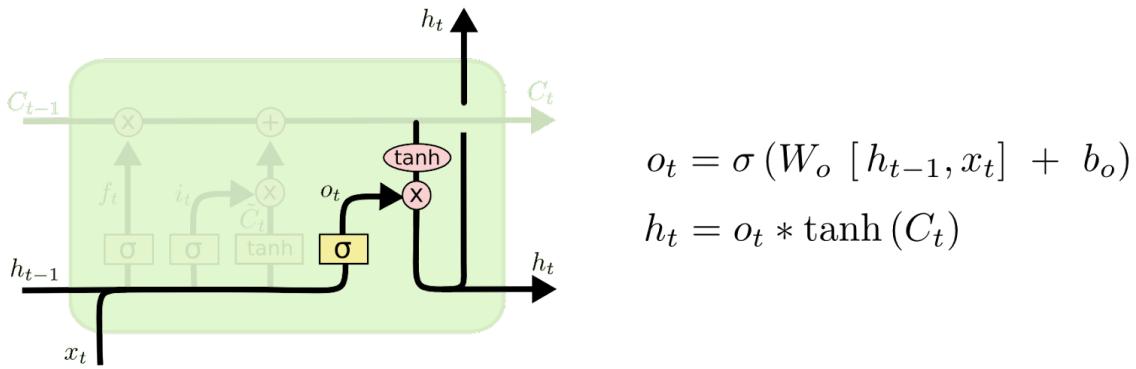
Hình: Cập nhật giá trị cho ô trạng thái bằng cách kết hợp 2 kết quả từ tầng cổng vào và tầng ẩn hàm tanh

Chúng ta nhân trạng thái cũ với  $f_t$  tương ứng với việc quên những thứ quyết định được phép quên sớm. Phần tử để cử  $i_t * \tilde{C}_t$  là một giá trị mới được tính toán tương ứng với bao nhiêu được cập nhật vào mỗi giá trị trạng thái.



Hình: Ô trạng thái mới

Cuối cùng cần quyết định xem đầu ra sẽ trả về bao nhiêu. Kết quả ở đầu ra sẽ dựa trên ô trạng thái, nhưng sẽ là một phiên bản được lọc. Đầu tiên, chúng ta chạy qua một tầng sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra. Sau đó, ô trạng thái được đưa qua hàm tanh (để chuyển giá trị về khoảng -1 và 1) và nhân nó với đầu ra của một cổng sigmoid, do đó chỉ trả ra phần mà chúng ta quyết định.



Hình: Điều chỉnh thông tin ở đầu ra thông qua hàm tanh

#### 5.4. Ứng dụng

Mạng không chỉ xử lý các điểm dữ liệu đơn lẻ (như các hình ảnh), mà còn xử lý toàn bộ chuỗi dữ liệu (chẳng hạn như lời nói hoặc video). Ví dụ, LSTM có thể áp dụng cho các tác vụ nhận dạng chữ viết tay, nhận dạng tiếng nói và phát hiện bất thường có tính chất kết nối, không phân đoạn trong giao thông mạng hoặc các IDS (hệ thống phát hiện xâm nhập). Ngoài ra, LSTM còn có một số ứng dụng cụ thể khác như quản lý robot, dự đoán chuỗi thời gian, học giai điệu (nhiều điệu), sáng tác nhạc, học ngữ pháp, nhận dạng hành động của con người, phát hiện tương đồng protein, dự đoán vị trí dưới tế bào của protein, phát hiện chuỗi thời gian bất thường, một số tác vụ dự đoán trong lĩnh vực quản lý quy trình kinh doanh, dự đoán trong các lô trình chăm sóc y tế, phân tích ngữ nghĩa, đồng phân đoạn đối tượng, quản lý hành khách sân bay, thiết kế thuốc.

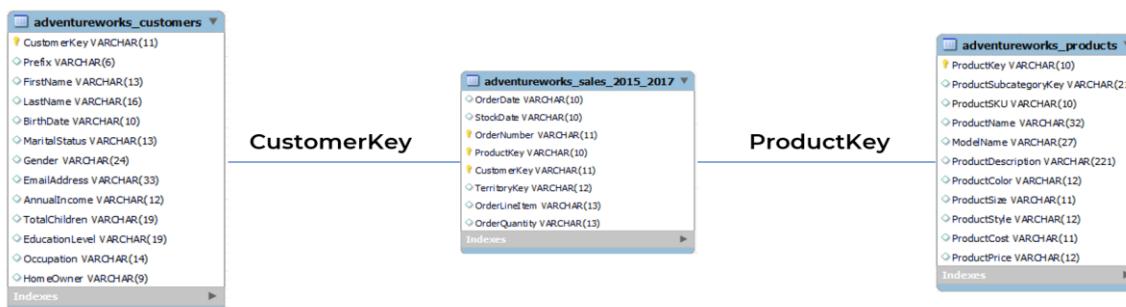
## Chương 3: Bộ dữ liệu

Bộ dữ liệu của công ty AdventureWork gồm nhiều bảng để lưu trữ thông tin khách hàng, sản phẩm,... Bảng fact là bản ghi lại thông tin bán hàng từ năm 2015-2017.

Bộ dữ liệu AdventureWorks bao gồm các tập dữ liệu khác nhau về hoạt động kinh doanh, bao gồm:

- Đơn đặt hàng (Orders): Bao gồm thông tin về các đơn đặt hàng của khách hàng, chẳng hạn như mã đơn hàng, khách hàng, ngày đặt hàng, sản phẩm được đặt hàng, giá cả, số lượng, và các thông tin khác liên quan.
- Khách hàng (Customers): Bao gồm thông tin về các khách hàng, chẳng hạn như mã khách hàng, tên, địa chỉ, số điện thoại, email, và các thông tin khác liên quan.
- Sản phẩm (Products): Bao gồm thông tin về các sản phẩm, chẳng hạn như mã sản phẩm, tên sản phẩm, mô tả, giá cả, và các thông tin khác liên quan.
- Nhân viên (Employees): Bao gồm thông tin về các nhân viên, chẳng hạn như mã nhân viên, tên, chức vụ, phòng ban, ngày vào làm, lương, và các thông tin khác liên quan.
- Các tập dữ liệu khác: Bên cạnh các tập dữ liệu chính, AdventureWorks còn bao gồm các tập dữ liệu khác như địa điểm, danh mục, giảng viên, hóa đơn, v.v.

Từ bộ dữ liệu ở trên nhóm đã sử dụng một số bảng là customers, sales, product các bảng này được nối với nhau qua các khóa như CustomerKey và Productkey



Hình: Mô hình liên hệ giữa các bảng dữ liệu

## Chương 4: Thực nghiệm và kết quả

### 1. Tiết xử lý

Đầu tiên, chúng ta xem xét bộ dữ liệu thông qua phương thức “info()”.

```
df.info()
```

Kết quả hiển thị:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 56046 entries, 0 to 56045
Data columns (total 30 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   ProductKey      56046 non-null   int64  
 1   ProductSubcategoryKey 56046 non-null   int64  
 2   ProductSKU       56046 non-null   object  
 3   ProductName      56046 non-null   object  
 4   ModelName        56046 non-null   object  
 5   ProductDescription 56046 non-null   object  
 6   ProductColor     56046 non-null   object  
 7   ProductSize      56046 non-null   object  
 8   ProductStyle     56046 non-null   object  
 9   ProductCost       56046 non-null   float64 
 10  ProductPrice     56046 non-null   float64 
 11  CustomerKey      56046 non-null   int64  
 12  Prefix            56046 non-null   object  
 13  FirstName         56046 non-null   object  
 14  LastName          56046 non-null   object  
 15  BirthDate         56046 non-null   object  
 16  MaritalStatus     56046 non-null   object  
 17  Gender             56046 non-null   object  
 18  EmailAddress      56046 non-null   object  
 19  AnnualIncome      56046 non-null   object  
 20  TotalChildren     56046 non-null   int64  
 21  EducationLevel    56046 non-null   object  
 22  Occupation         56046 non-null   object  
 23  HomeOwner          56046 non-null   object  
 24  OrderDate          56046 non-null   object  
 25  StockDate          56046 non-null   object  
 26  OrderNumber         56046 non-null   object  
 27  TerritoryKey       56046 non-null   int64  
 28  OrderLineItem      56046 non-null   int64  
 29  OrderQuantity      56046 non-null   int64
```

#### 1.1. Xử lý thời gian bằng datetime

Định dạng lại biến OrderDate về dạng date bằng hàm “pd.to\_datetime”, format = ‘%m/%d/%Y’

Lấy năm từ OrderDate bằng “dt.year”

Lấy tháng từ OrderDate bằng “dt.month”

Lấy ra thứ mấy từ OrderDate bằng “dt.weekday”

```
df['OrderDate'] = pd.to_datetime(df['OrderDate'], format='%m/%d/%Y')
df['year'] = df['OrderDate'].dt.year
df['month'] = df['OrderDate'].dt.month
df['day'] = df['OrderDate'].dt.day
df['weekday'] = df['OrderDate'].dt.weekday
```

### 1.2. Xử lý các biến thông tin khách hàng

Xử lý biến thu nhập hàng năm: Do biến này đang ở định dạng string, còn nhiều kí tự đặc biệt như “,” và “\$” và chuyển về dạng “float”

```
df['AnnualIncome'] = df['AnnualIncome'].astype(str).str.replace(",","");
df['AnnualIncome'] = df['AnnualIncome'].astype(str).str.replace("$","");
df['AnnualIncome'] = df['AnnualIncome'].astype('float')
```

Kết quả:

```
df['AnnualIncome'].head()

0    40000.0
1    30000.0
2    70000.0
3    30000.0
4    30000.0
Name: AnnualIncome, dtype: float64
```

Xử lý biến “BirthDate”: chuyển về định dạng date, với format = “%m/%d/%Y” và tính tuổi của các khách hàng lưu thành cột “age”

```
df['BirthDate'] = pd.to_datetime(df['BirthDate'], format='%m/%d/%Y')
df['age'] = [2017 - i for i in df['BirthDate'].dt.year]
```

	BirthDate	age
0	1960-03-18	57
1	1970-02-27	47
2	1960-10-08	57
3	1979-06-02	38
4	1978-02-14	39

Xử lý lại các cột “Gender”, “HomeOwner” và “TotalChildren”. Hiện tại các cột này rất nhiều dữ liệu khác nhau.

- Gender: “M” là nam và “F” là nữ. Tiến hành encode biến “**M**” là **1** và biến “**F**”:**0**
- HomeOwner: “Y” là có sở hữu nhà và “N” là không có sở hữu nhà. Tiến hành Encode biến “**Y**”: **1** và biến “**N**”: **0**
- TotalChildren: chứa thông tin gia đình có bao nhiêu con. Đối với xử liệu này nhóm em sẽ chia làm 2 loại là gia đình đã có con và gia đình chưa có con. Đối với gia đình đã có con, nhóm sẽ toàn bộ dữ liệu  $> 0$  là tiến hành encode = 1 và gia đình không có con = 0.

```
df['is_male'] = [1 if i=='M' else 0 for i in df['Gender']]  
df['have_home'] = [1 if i=='Y' else 0 for i in df['HomeOwner']]  
df['have_child'] = [1 if i > 0 else 0 for i in df['TotalChildren']]
```

Kết quả:

	is_male	have_home	have_child
0	1	0	0
1	1	1	0
2	1	1	1
3	0	1	1
4	1	1	1

### 1.3. Tính tổng giá trị đơn hàng

(total\_bill): total\_bill = ProductPrice \* OrderQuantity

```
df['total_bill'] = [i*j for i,j in zip(df['ProductPrice'],df['OrderQuantity'])]
```

### 1.4. Gom dữ liệu theo ngày bằng groupby

```
df_gb = df.groupby(['month','year','day','weekday']).agg({'total_bill':'sum','OrderNumber':'nunique',  
'CustomerKey':'nunique','AnnualIncome':'mean','age':'mean',  
'is_male':'sum','have_home':'sum','have_child':sum}).reset_index()  
df_gb.sort_values(['year','month','day','weekday'], inplace=True, ignore_index=True)
```

## 1.5. Tính phần trăm

Do biến CustomerKey đã được groupby unique ở trên, nên hiện tại biến CustomerKey là số lượng khách hàng do tính phần trăm sẽ chia cho CustomerKey

- 1) Phần trăm khách hàng là nam giới (df['male\_per'])

```
df['male_per'] = df['is_male'] / df['CustomerKey']
```

- 2) Phần trăm khách hàng sở hữu nhà (df['home\_per'])

```
df['home_per'] = df['have_home'] / df['CustomerKey']
```

- 3) Phần trăm khách hàng đã có con (df['child\_per'])

```
df['child_per'] = df['have_child'] / df['CustomerKey']
```

```
df_gb['male_per'] = [i/j for i,j in zip(df_gb['is_male'],df_gb['CustomerKey'])]
df_gb['home_per'] = [i/j for i,j in zip(df_gb['have_home'],df_gb['CustomerKey'])]
df_gb['child_per'] = [i/j for i,j in zip(df_gb['have_child'],df_gb['CustomerKey'])]
```

## 1.6. Rolling feature

Phương thức Rolling có một số đối số, bao gồm kích thước window và loại thống kê để tính toán. Trong trường hợp này, kích thước window được đặt thành 7, có nghĩa là số liệu thống kê luôn phiên được tính toán trên 7 dòng, mà do dữ liệu ở trên đã group by và sort theo date nên 7 dòng ở đây có nghĩa là gom 7 ngày lại và tính toán thống kê.

Nhóm em lần lượt tính toán các giá trị trung bình và độ lệch chuẩn của “total\_bill” trong 7 ngày và thêm các giá trị này dưới dạng cột mới vào DataFrame. Tiếp tục thực hiện tương tự cho các biến: “OrderNumber”, “CustomerKey”, “male\_per”, “home\_per” và “child\_per”.

```
df_gb['rolling_mean_bill'] = df_gb.total_bill.rolling(window=7,closed = 'left').mean()
df_gb['rolling_std_bill'] = df_gb.total_bill.rolling(window=7,closed = 'left').std()

df_gb['rolling_mean_order'] = df_gb.OrderNumber.rolling(window=7,closed = 'left').mean()
df_gb['rolling_std_order'] = df_gb.OrderNumber.rolling(window=7,closed = 'left').std()

df_gb['rolling_mean_cus'] = df_gb.CustomerKey.rolling(window=7,closed = 'left').mean()
df_gb['rolling_std_cus'] = df_gb.CustomerKey.rolling(window=7,closed = 'left').std()

df_gb['rolling_mean_gender'] = df_gb.male_per.rolling(window=7,closed = 'left').mean()
df_gb['rolling_mean_home'] = df_gb.home_per.rolling(window=7,closed = 'left').mean()
df_gb['rolling_mean_child'] = df_gb.child_per.rolling(window=7,closed = 'left').mean()
```

Phương pháp Rolling trong Pandas cung cấp một công cụ mạnh mẽ để tính toán số liệu thống kê cuộn trên một cửa sổ các điểm dữ liệu liên tiếp. Bằng cách sử dụng số liệu thống kê, nhóm em có thể xác định các xu hướng và mẫu trong dữ liệu chuỗi thời gian, đồng thời làm mịn các dao động và nhiễu. Điều này có thể đặc biệt hữu ích trong phân tích tài chính và kinh tế, nơi các xu hướng và mô hình dài hạn thường được quan tâm.

### 1.7. Tính RSI

Các bước tính toán RSI:

- 1) Định nghĩa thời gian cho chu kỳ RSI, trong đoạn code này được đặt là 7.
- 2) Tính toán sự thay đổi giá và hướng giá cho từng ngày dữ liệu bằng cách tính hiệu của cột “total\_bill” theo từng ngày.
- 3) Tính toán lợi nhuận (gains) và mất mát (losses) riêng biệt bằng cách lấy các giá trị dương và âm của delta, tương ứng với sự thay đổi giá.
- 4) Tính toán giá trị trung bình cho lợi nhuận và mất mát bằng rolling mean trong một chu kỳ thời gian được định nghĩa trước đó (period = 7).
- 5) Tính toán giá trị RSI bằng cách tính giá trị RS, được tính bằng cách lấy giá trị trung bình của lợi nhuận chia cho giá trị trung bình của mất mát.
- 6) Tính toán giá trị RSI thực tế bằng cách sử dụng công thức  $RSI = 100 - (100 / (1 + RS))$ .
- 7) Thêm giá trị RSI vào cột mới của dataframe df\_gb.

```

# Define the time period for the RSI
period = 7

delta = df_gb['total_bill'].diff()
direction = np.where(delta > 0, 1, np.where(delta < 0, -1, 0))

gain = delta.where(delta > 0, 0)
loss = -delta.where(delta < 0, 0)

avg_gain = gain.rolling(window=period).mean()
avg_loss = loss.rolling(window=period).mean()

rs = avg_gain / avg_loss
rsi = 100 - (100 / (1 + rs))

df_gb['rsi'] = rsi

```

### 1.8. Scaling data

Đầu tiên, nhóm sẽ chia dữ liệu làm 2 bộ riêng biệt: “train\_data” và “no\_scale”.

- Ở bộ “train\_data”, nhóm sẽ loại bỏ các cột: ‘month’, ‘year’, ‘day’, ‘weekday’, ‘OrderNumber’, ‘CustomerKey’, ‘AnnualIncome’, ‘age’, ‘is\_male’, ‘have\_home’, ‘have\_child’, ‘male\_per’, ‘home\_per’, ‘child\_per’, ‘total\_bill’
- Ở bộ “no\_train”, nhóm chỉ lấy các cột đã bỏ phía trên

“train\_data” sẽ chứa các dữ liệu được chuẩn hóa, trong khi “no\_scale” sẽ chứa các cột dữ liệu không được chuẩn hóa.

```

train_data = df_gb[df_gb.columns.drop(['month', 'year', 'day', 'weekday', 'OrderNumber', 'CustomerKey', \
                                         'AnnualIncome', 'age', 'is_male', 'have_home', 'have_child', \
                                         'male_per', 'home_per', 'child_per', 'total_bill'])]

no_scale = df_gb[['month', 'year', 'day', 'weekday', 'OrderNumber', 'CustomerKey', 'AnnualIncome', 'age', \
                  'is_male', 'have_home', 'have_child', 'male_per', 'home_per', 'child_per', 'total_bill']]

```

Để scaler bộ data này, nhóm em dùng RobustScaler để chuẩn hóa cho tập “train\_data”. RobustScaler là một phương pháp chuẩn hóa dữ liệu theo dạng scale theo phạm vi (interquartile range - IQR). Với phương pháp này, các giá trị bất thường sẽ được xử lý tốt hơn so với phương pháp chuẩn hóa MinMaxScaler hay

StandardScaler do data của nhóm có nhiều giá trị outlier nhưng mà quan trọng nên không loại bỏ được.

Sau đó, nhóm em sử dụng phương thức “fit\_transform()” để fit và transform dữ liệu. Dữ liệu sau khi được chuẩn hóa được lưu vào scaled\_data. Sau đó, ta sử dụng pd.DataFrame để tạo một DataFrame mới “scaled\_df” từ “scaled\_data”. Các cột của “scaled\_df” được đổi tên bằng cách thêm tiền tố “S\_” vào trước tên cột.

Cuối cùng, ta dùng phương thức concat() để ghép “train\_data” và “scaled\_df” theo chiều dọc (axis=1), để tạo ra “train\_data” mới chứa các cột dữ liệu gốc và các cột dữ liệu đã được chuẩn hóa.

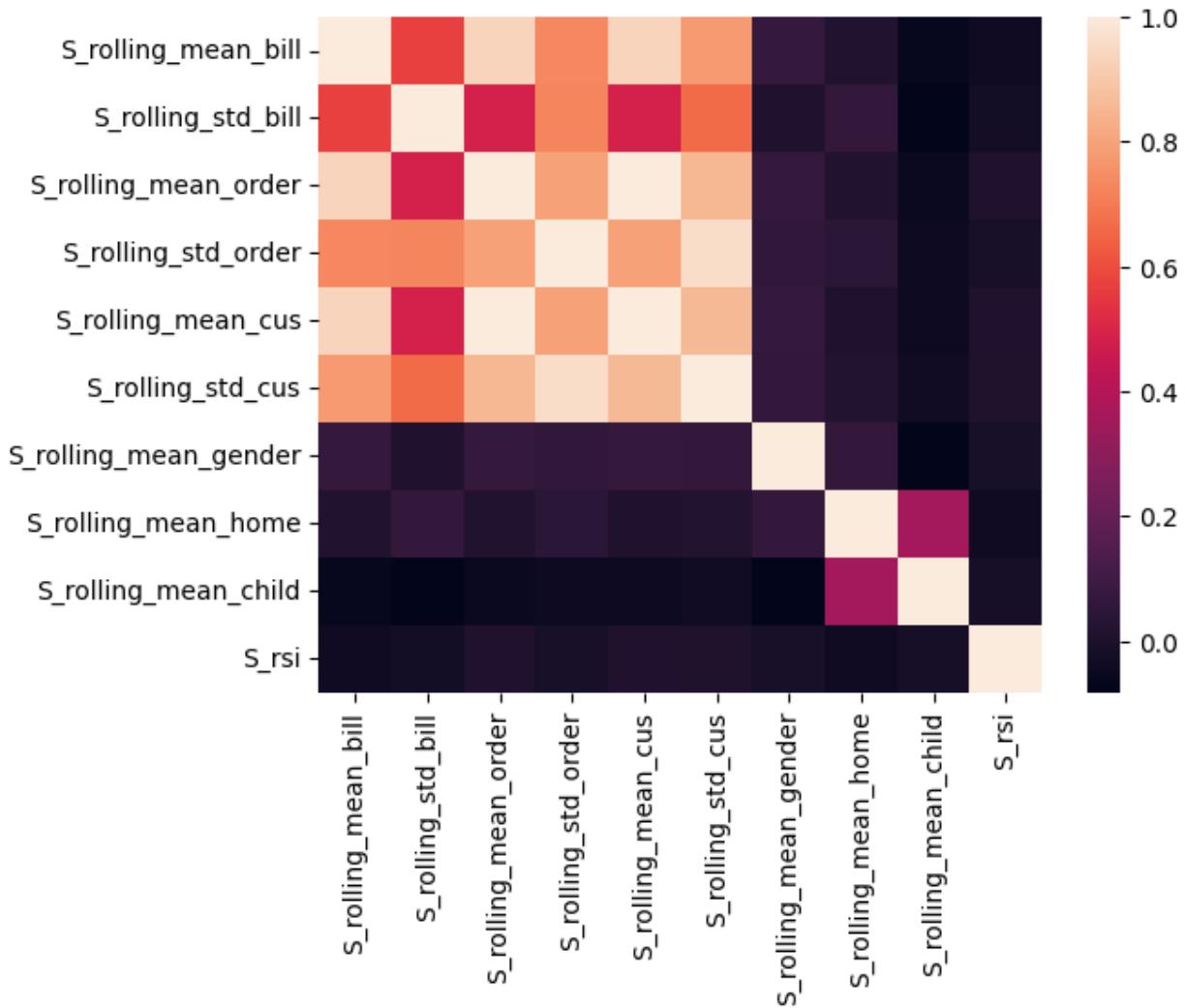
```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
scaled_data = scaler.fit_transform(train_data)
scaled_df = pd.DataFrame(scaled_data, columns=train_data.columns).add_prefix('S_')
train_col = scaled_df.columns
train_data = pd.concat([no_scale,scaled_df], axis=1)
```

### 1.9. Correlation

Nhóm em sẽ lấy các cột đã scale bên trên và vẽ heatmap để vẽ tương quan giữa các biến

```
corr_matrix = train_data.drop(columns=['month', 'year', 'day', 'weekday', 'OrderNumber', \
                                         'CustomerKey', 'AnnualIncome', 'age', 'is_male', \
                                         'have_home', 'have_child', 'male_per', 'home_per', \
                                         'child_per', 'total_bill']).get_numeric_data().corr()
sns.heatmap(corr_matrix)
```

Kết quả:



Tiếp theo, nhóm sẽ loại các biến mà có độ tương quan quá cao (trên 0.85) để tránh hiện tượng đa cộng tuyến. Cách thực hiện:

- Đầu tiên, tạo ra biến threshold để xác định vượt quá bao nhiêu thì tương quan sẽ được coi là quá cao. Sau đó, tạo ra một ma trận tương quan bằng cách tính toán hệ số tương quan Pearson tuyệt đối giữa tất cả các cặp tính năng trong tập dữ liệu.
- Tiếp theo, chọn tam giác trên của ma trận tương quan, loại bỏ đường chéo, vì tam giác dưới chỉ là hình ảnh phản chiếu của tam giác trên. Điều này được thực hiện để tránh xem xét cùng một tương quan hai lần.
- Cuối cùng, tìm chỉ mục của các cột tính năng có tương quan lớn hơn ngưỡng tương quan bằng cách sử dụng list comprehension và lưu chúng vào danh

sách `to_drop_corr`. Các tính năng này sẽ được loại bỏ khỏi tập dữ liệu để giảm đa cộng tuyến giữa các tính năng còn lại.

```
# threshold above which is considered too high of a correlation
corr_threshold = 0.85

corr_matrix = corr_matrix.abs()

upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

to_drop_corr = [column for column in upper.columns if any(upper[column] >= corr_threshold)]
print(to_drop_corr)
```

In kết quả các cột tương quan  $> 0.85$ :

```
['S_rolling_mean_order', 'S_rolling_mean_cus', 'S_rolling_std_cus']
```

Tiếp theo, nhóm sẽ loại các cột này thông qua hàm `drop`

```
train_col.drop(to_drop_corr)

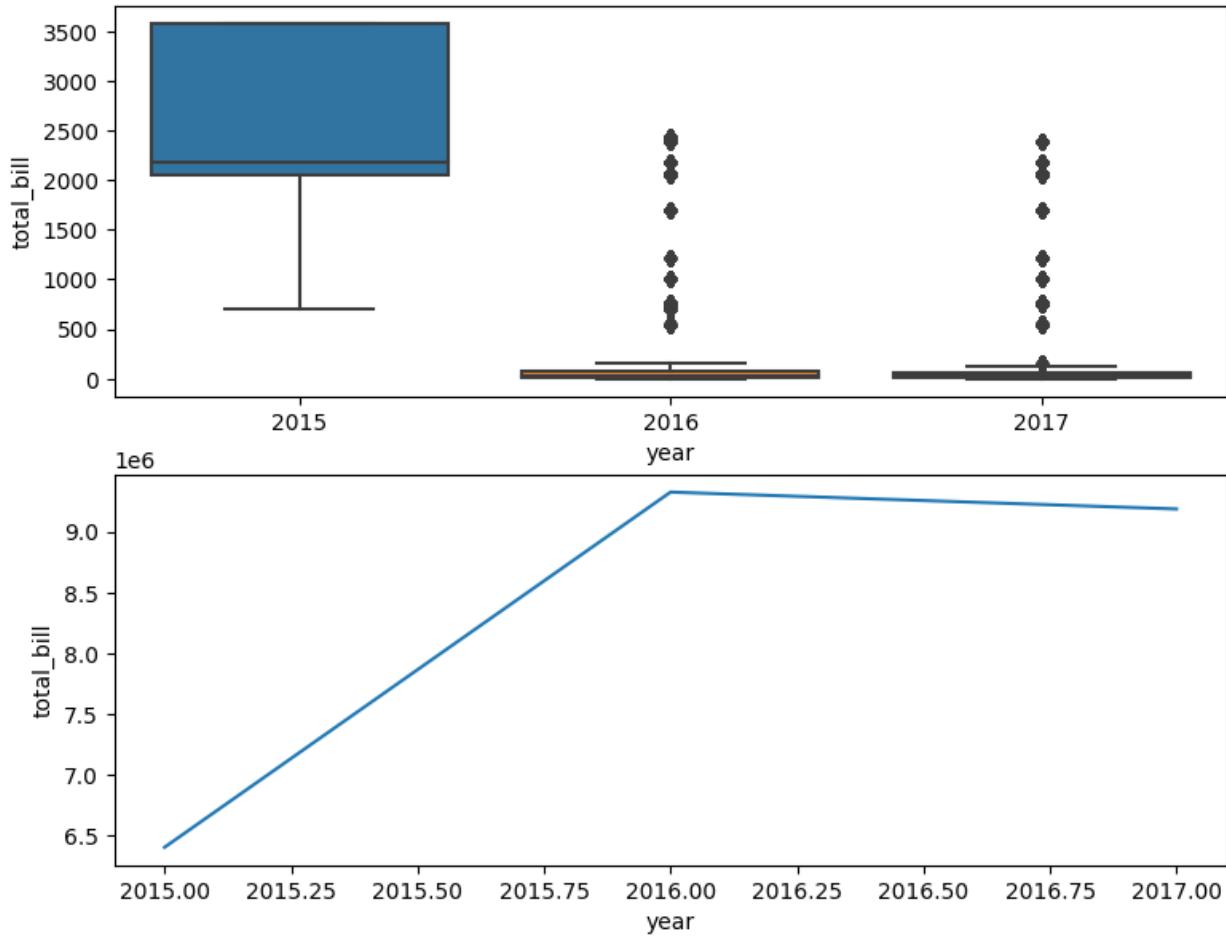
Index(['S_rolling_mean_bill', 'S_rolling_std_bill', 'S_rolling_std_order',
       'S_rolling_mean_gender', 'S_rolling_mean_home', 'S_rolling_mean_child',
       'S_rsi'],
      dtype='object')
```

## 2. EDA

### 2.1. Time Series decomposition:

a. Year:

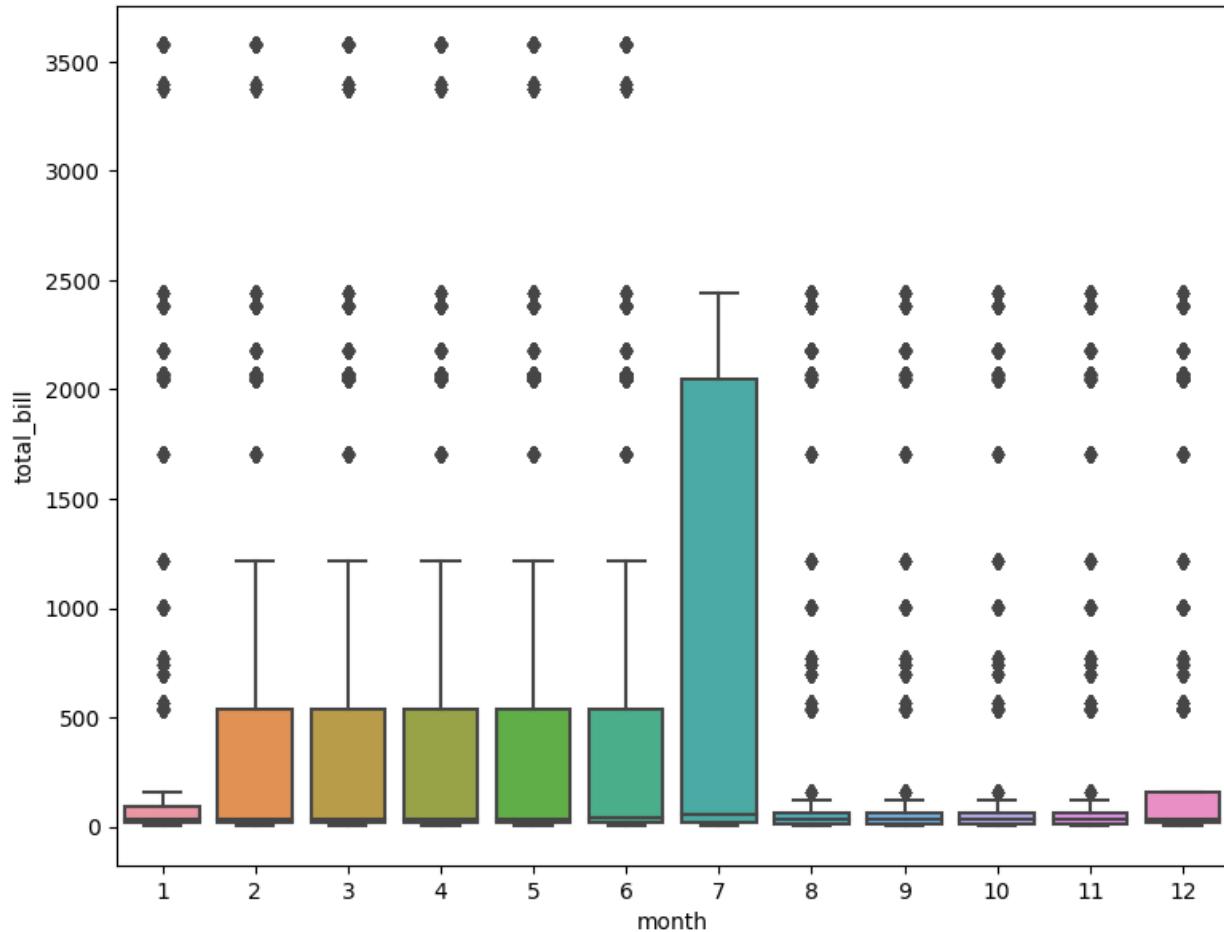
```
yearly_agg = df.groupby('year')['total_bill'].sum().reset_index()
fig, axs = plt.subplots(nrows=2, figsize=(9,7))
sns.boxplot(x='year', y='total_bill', data=df, ax=axs[0])
_ = sns.lineplot(x='year', y='total_bill', data=yearly_agg, ax=axs[1])
```



Ở biểu đồ hộp, ta thấy giá trị từng đơn hàng ở năm 2015 là cao nhất, tuy vậy nếu xét về tổng giá trị đơn hàng cả năm thì, theo lineplot, ta có thể dễ dàng thấy được tổng giá trị đơn hàng tăng dần theo từng năm. Đối với năm 2017, mặc dù data chỉ có 6 tháng nhưng tổng giá trị đơn hàng đã gần chạm đỉnh năm 2016.

### 1. Month

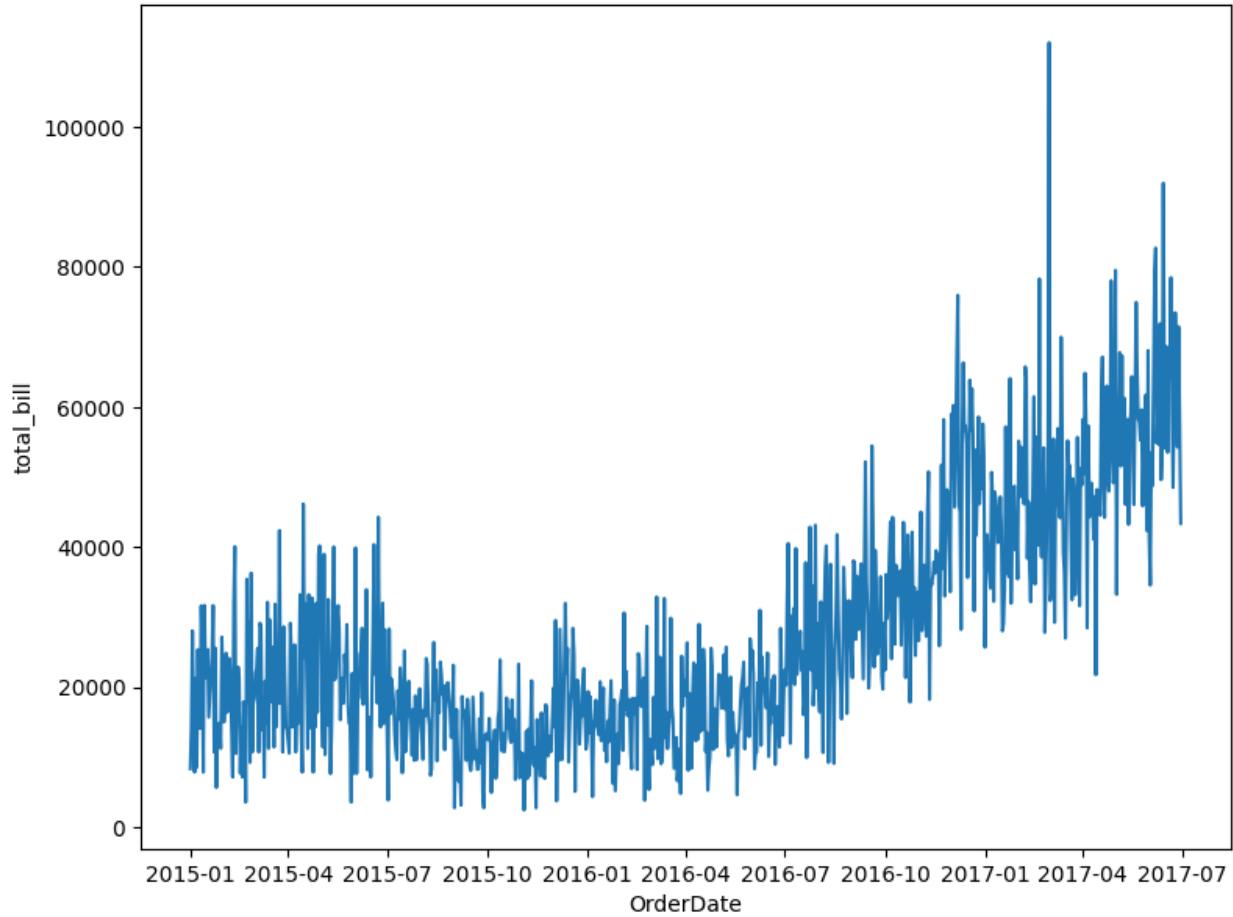
```
monthly_agg = df.groupby('month')['total_bill'].sum().reset_index()
fig, axs = plt.subplots(nrows=1, figsize=(9,7))
sns.boxplot(x='month', y='total_bill', data=df)
```



Dựa vào biểu đồ hộp gộp theo tháng, ta có thể thấy tổng giá trị đơn hàng của từng tháng biến động khá nhiều, có những tháng 1 - 6, giá trị đơn hàng có thể chậm đinh nhưng có thể một số đơn hàng thì giá trị xấp xỉ không, biến động tương đối lớn. Riêng chỉ tháng 7, tổng giá trị đơn hàng trên hình ta thấy khá ổn định, không có các đơn hàng quá ngoại biên. Cũng vì lẽ đó, ta có thể kết luận doanh thu từ tháng 7 này sẽ ổn định hơn nhiều so với các tháng còn lại.

## 2. Date:

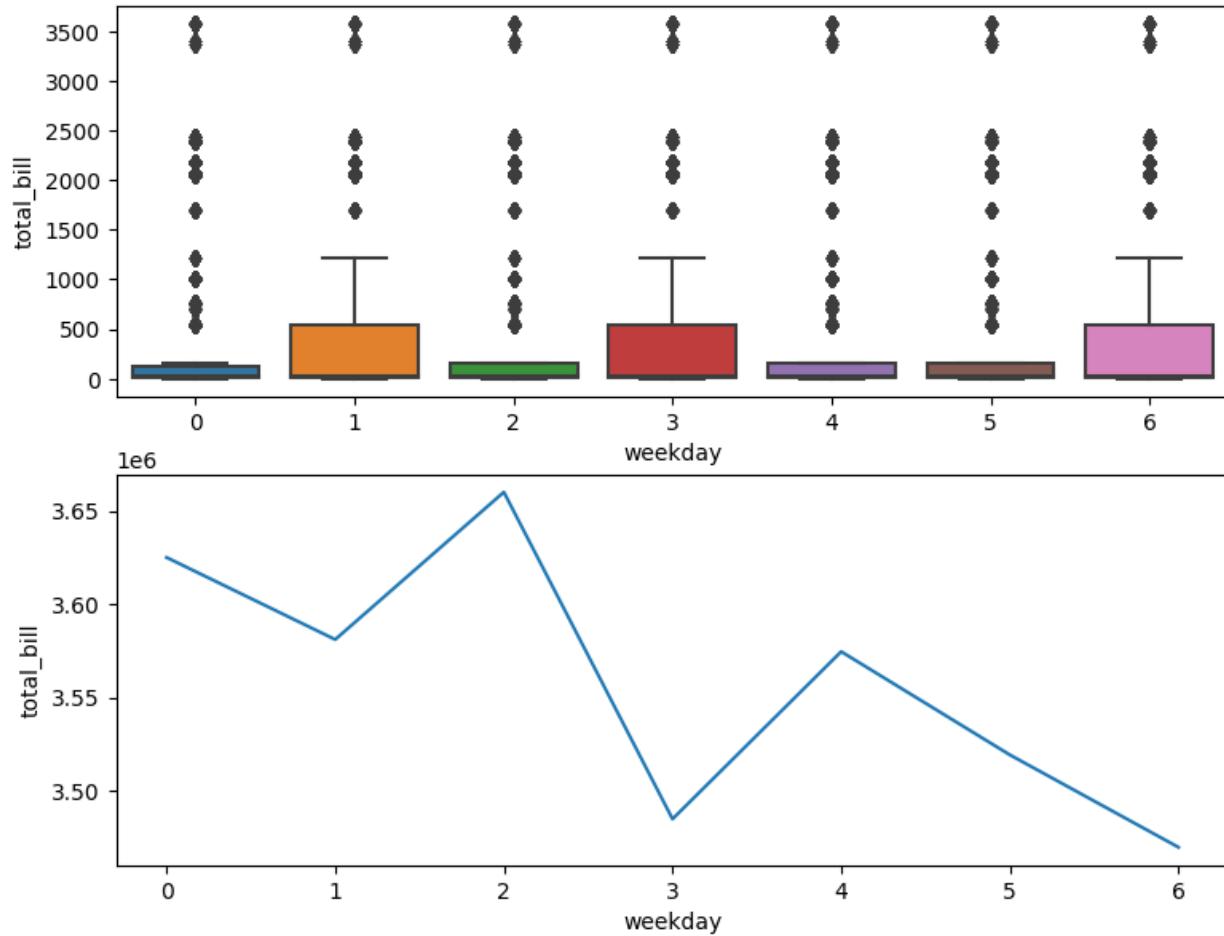
```
monthly_agg = df.groupby('OrderDate')['total_bill'].sum().reset_index()
fig, axs = plt.subplots(nrows=1, figsize=(9,7))
_ = sns.lineplot(x='OrderDate', y='total_bill', data=monthly_agg)
```



Tương tự như dữ liệu gom theo năm, ta có thể thấy tổng giá trị đơn hàng tăng dần theo thời gian.

### 3. Weekday:

```
wd_agg = df.groupby('weekday')['total_bill'].sum().reset_index()
fig, axs = plt.subplots(nrows=2, figsize=(9,7))
sns.boxplot(x='weekday', y='total_bill', data=df, ax=axs[0])
_ = sns.lineplot(x='weekday', y='total_bill', data=wd_agg, ax=axs[1])
```



Ở đây, ta có thể thấy mỗi thứ trong tuần đều có các đơn hàng nhiều giá trị khác nhau, theo như lineplot thì ta thấy thứ 4 là ngày có tổng giá trị đơn hàng cao nhất. Ta có thể kích cầu, tăng cá khuyến mãi vào ngày thứ 4.

## 2.2. Moving Average (MA):

**Moving Average (MA)** là giá trị trung bình của một tập hợp các quan sát trong một khoảng thời gian nhất định. Nó được sử dụng để loại bỏ những biến động ngắn hạn và nhìn vào xu hướng chung của dữ liệu.

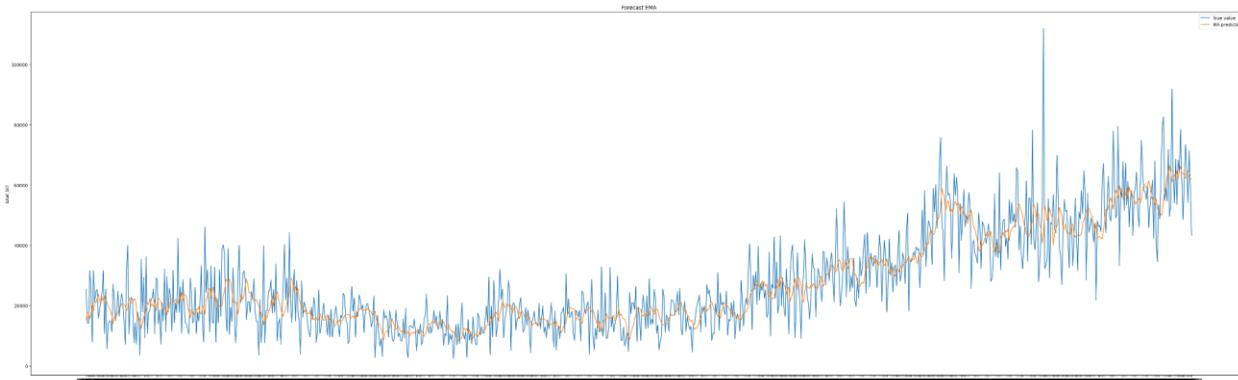
Trong Python, để tính toán Moving Average, nhóm em sử dụng thư viện Pandas. Thư viện này cung cấp hàm `rolling()` để tính toán trung bình trượt. Hàm này sẽ tạo ra một đối tượng "trượt" với kích thước cửa sổ (window) được chỉ định và tính toán giá trị trung bình của các phần tử trong cửa sổ đó. Nhóm đã tính hàm này ở phía trên: `df['rolling_mean_bill']`

```

plt.figure(figsize=(50,15))
plt.plot(dfgb['OrderDate'], dfgb['total_bill'], label = 'True value')
plt.plot(dfgb['OrderDate'], dfgb['rolling_mean_bill'], label = 'MA prediction')
plt.legend(loc='best')
plt.xlabel('date')
plt.ylabel('total_bill')
plt.title('Forecast EMA')

```

Kết quả:



Theo như đồ thị, ta có thể thấy đường dự đoán giá MA không bám sát với đường giá trị đơn hàng thực tế cho lắm.

Tiếp đó, nhóm em sẽ tính các chỉ số MSE và MAE để so sánh với các chỉ báo khác. Cụ thể, **mean\_squared\_error (MSE)** tính toán bình phương sai số trung bình giữa giá trị thực tế và giá trị dự đoán, trong khi **mean\_absolute\_error (MAE)** tính toán trung bình giá trị tuyệt đối của sai số.

```

mse_lst.append(mean_squared_error(dfgb['total_bill'],dfgb['rolling_mean_bill']))
mae_lst.append(mean_absolute_error(dfgb['total_bill'],dfgb['rolling_mean_bill']))
model_lst.append('MA')

print(f'{model_lst[-1]}\nMAE:{mae_lst[-1]}\nMSE:{mse_lst[-1]}')

```

MA  
MAE:6848.75683596713  
MSE:83388914.67321005

### 2.3. Exponential Moving Average (EMA):

**Exponential Moving Average (EMA)** là một công cụ quan trọng trong phân tích dữ liệu và dự đoán thời gian. EMA là một phiên bản cải tiến của Moving Average

(MA), trong đó các giá trị gần đây được đánh giá cao hơn và có ảnh hưởng lớn hơn đến giá trị trung bình.

Trong Python, để tính toán EMA, nhóm sử dụng thư viện Pandas. Thư viện này cung cấp hàm `ewm()` để tính toán trung bình trượt theo phương pháp EMA. Hàm này sẽ tạo ra một đối tượng "trượt" với kích thước cửa sổ (window) được chỉ định và tính toán giá trị trung bình theo phương pháp EMA.

Nhóm em ở đây sẽ gom trung bình 7 ngày 1 lần để tính EMA.

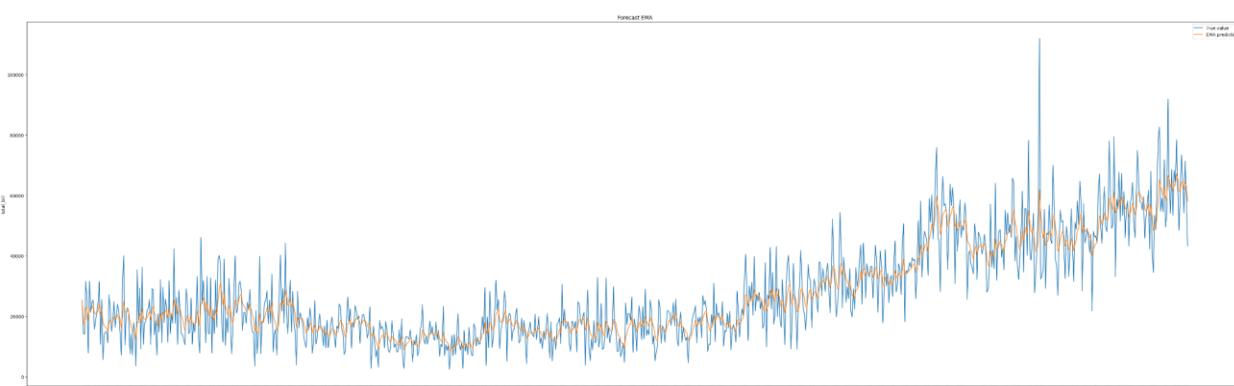
```
# Define the number of days for the EMA
num_days = 7

ema = dfgb['total_bill'].ewm(span=num_days, adjust=True).mean()

dfgb['ema'] = emma
dfgb
```

Visualize EMA:

```
plt.figure(figsize=(50,15))
plt.plot(dfgb['OrderDate'], dfgb['total_bill'], label ='True value')
plt.plot(dfgb['OrderDate'], dfgb['ema'], label ='EMA prediction')
plt.legend(loc='best')
plt.xlabel('date')
plt.ylabel('total_bill')
plt.title('Forecast EMA')
```



Ở đây, ta thấy đường EMA dự đoán vẫn không fit nhiều so với giá trị đơn hàng thực tế lắm như đường MA.

Tiếp tục tính các chỉ số MSE và MAE

```
mse_lst.append(mean_squared_error(dfgb['total_bill'],dfgb['ema']))
mae_lst.append(mean_absolute_error(dfgb['total_bill'],dfgb['ema']))
model_lst.append('EMA')

print(f'{model_lst[-1]}\nMAE:{mae_lst[-1]}\nMSE:{mse_lst[-1]}')

EMA
MAE:5143.066816073632
MSE:47052381.350058496
```

#### 2.4. Auto Regression (AR):

Auto Regression (AR) là một phương pháp dự đoán thời gian trong đó giá trị tương lai được dự đoán dựa trên các giá trị quá khứ. Trong AR, giá trị tại một thời điểm được xác định dựa trên giá trị của một số lượng giá trị quá khứ được gọi là "thứ tự p" (order p).

Trong Python, để thực hiện mô hình AR, nhóm có thể sử dụng model ARIMA của thư viện statsmodels. Model này cung cấp lớp ARIMA để xây dựng và khớp mô hình ARIMA trên các chuỗi dữ liệu.

Nhóm sử dụng model AutoReg của thư viện statsmodels để xây dựng mô hình Auto Regression trên chuỗi dữ liệu 'total\_bill'. Sử dụng đối số 'lags=7' để chỉ định rằng chúng ta muốn xây dựng một mô hình AR với 7 ngày trong quá khứ. Sau đó, fit mô hình và sử dụng nó để dự đoán trên toàn bộ tập dữ liệu.

```
from statsmodels.tsa.ar_model import AutoReg
model = AutoReg(dfgb['total_bill'], lags=7).fit()

# Use the fitted model to make predictions on the testing data
predictions = model.predict(start=dfgb.index[0], end=dfgb.index[-1])

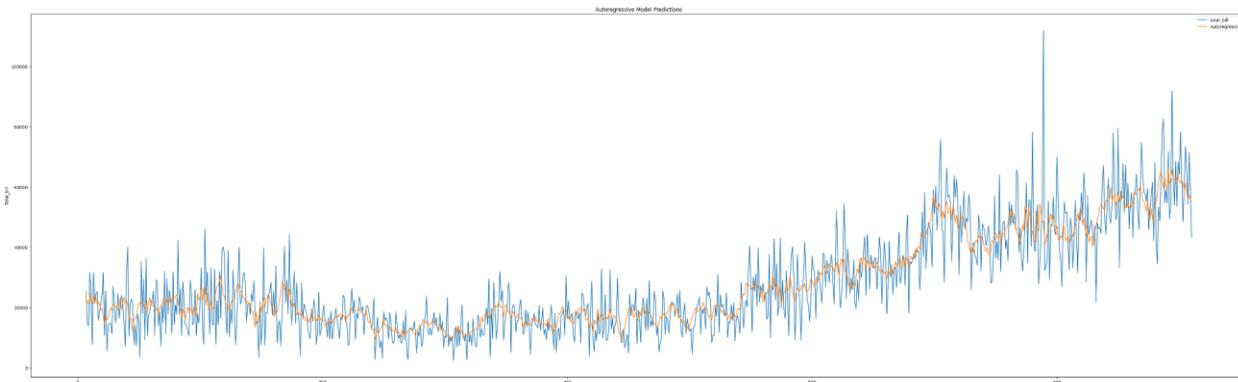
dfgb['autoreg'] = predictions
```

Visualize AR:

```

plt.figure(figsize=(50,15))
plt.plot(dfgb.index, dfgb['total_bill'], label='total_bill')
plt.plot(predictions.index, predictions, label='Autoregression')
plt.legend(loc='best')
plt.title('Autoregressive Model Predictions')
plt.xlabel('Date')
plt.ylabel('Total_bill')
plt.show()

```



Tương tự như MA và EMA phía trên, model này bằng việc sử dụng dữ liệu trong quá khứ để dự đoán xu hướng vẫn chưa không fit lắm so với dữ liệu thực tế

Tiếp tục tính toán các chỉ số MSE và MAE của model:

```

mse_lst.append(mean_squared_error(dfgb['total_bill'],dfgb['autoreg']))
mae_lst.append(mean_absolute_error(dfgb['total_bill'],dfgb['autoreg']))
model_lst.append('AutoReg')

print(f'{model_lst[-1]}\nMAE:{mae_lst[-1]}\nMSE:{mse_lst[-1]}')

```

```

AutoReg
MAE:6000.889916790978
MSE:63247582.72123052

```

## 2.5. Autoregressive Integrated Moving Average (ARIMA):

ARIMA (Autoregressive Integrated Moving Average) là một phương pháp dự đoán chuỗi thời gian trong đó giá trị tương lai được dự đoán dựa trên giá trị của các quan sát quá khứ và sai số của chuỗi đó. ARIMA kết hợp ba phương pháp: mô hình autoregressive (AR), mô hình moving average (MA) và mô hình integration (I). ARIMA được sử dụng để dự đoán các giá trị trong tương lai dựa trên các giá trị quá khứ.

Trong Python, để thực hiện ARIMA, nhóm sử dụng model ARIMA của thư viện statsmodels. Model này cung cấp lớp ARIMA để xây dựng và khớp mô hình ARIMA trên các chuỗi dữ liệu.

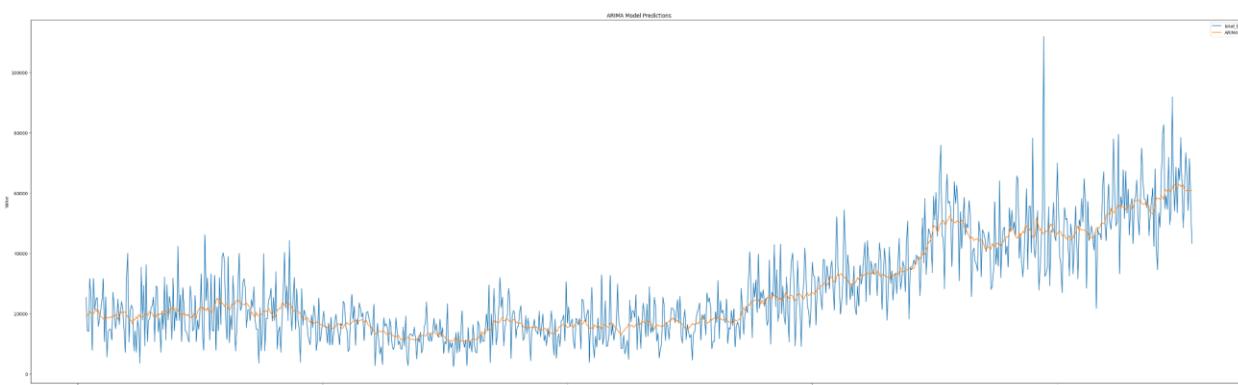
```
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(dfgb['total_bill'], order=(1,1,1)).fit()

# Use the fitted model to make predictions on the testing data
predictions = model.predict(start=dfgb.index[0], end=dfgb.index[-1])

dfgb['arima'] = predictions
```

Visualize ARIMA:

```
plt.figure(figsize=(50,15))
plt.plot(dfgb.index, dfgb['total_bill'], label='total_bill')
plt.plot(predictions.index, predictions, label='ARIMA')
plt.legend(loc='best')
plt.title('ARIMA Model Predictions')
plt.xlabel('Date')
plt.ylabel('Value')
plt.show()
```



Tiếp tục tính các chỉ số MSE và MAE của model ARIMA

```
mse_lst.append(mean_squared_error(dfgb['total_bill'],dfgb['arima']))
mae_lst.append(mean_absolute_error(dfgb['total_bill'],dfgb['arima']))
model_lst.append('ARIMA')

print(f'{model_lst[-1]}\nMAE:{mae_lst[-1]}\nMSE:{mse_lst[-1]}')
```

```
ARIMA
MAE:6216.605856110092
MSE:69150893.43518661
```

Dựa vào giá trị MSE và MAE của 4 model trên ta có:

```
MA
MAE:6848.75683596713
MSE:83388914.67321005
```

```
EMA
MAE:5143.066816073632
MSE:47052381.350058496
```

```
AutoReg
MAE:6000.889916790978
MSE:63247582.72123052
```

```
ARIMA
MAE:6216.605856110092
MSE:69150893.43518661
```

Chỉ số MAE và MSE càng thấp thì model dự báo càng tốt và mang lại kết quả tốt nhất. Ở đây ta có thể thấy, chỉ số MAE và MSE của model EMA là nhỏ nhất so

với các model còn lại. Do vậy với dữ liệu này, ta có thể dùng chỉ báo EMA để dự đoán “total\_bill”.

### 3. Mô hình máy học

#### 3.1. Linear Regression

Và sau khi tiền xử lý dữ liệu xong, nhóm em sẽ tiến hành xây dựng mô hình Đầu tiên là đến với phần chia dữ liệu, nhóm em sẽ chọn biến output chính là biến target của nhóm em là cột total\_bill, còn biến input là tất cả các biến nhóm em đã scaler như ở trên

```
[ ] train_col  
Index(['S_rolling_mean_bill', 'S_rolling_std_bill', 'S_rolling_mean_order',  
       'S_rolling_std_order', 'S_rolling_mean_cus', 'S_rolling_std_cus',  
       'S_rolling_mean_gender', 'S_rolling_mean_home', 'S_rolling_mean_child',  
       'S_rsi'],  
      dtype='object')  
  
[ ] output = train_data[['total_bill']]  
input = train_data[train_col]
```

Sau đó nhóm em sẽ chia dữ liệu ra thành 2 tập là tập train và tập test

```
[ ] x_train,x_test,y_train,y_test = train_test_split(input,output, test_size = 0.3, random_state=1)  
  
[ ] x_train.info()  
  
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 632 entries, 779 to 44  
Data columns (total 10 columns):  
 #   Column           Non-Null Count  Dtype     
---  
 0   S_rolling_mean_bill    632 non-null   float64  
 1   S_rolling_std_bill     632 non-null   float64  
 2   S_rolling_mean_order   632 non-null   float64  
 3   S_rolling_std_order    632 non-null   float64  
 4   S_rolling_mean_cus     632 non-null   float64  
 5   S_rolling_std_cus      632 non-null   float64  
 6   S_rolling_mean_gender  632 non-null   float64  
 7   S_rolling_mean_home    632 non-null   float64  
 8   S_rolling_mean_child   632 non-null   float64  
 9   S_rsi                 632 non-null   float64  
dtypes: float64(10)  
memory usage: 54.3 KB
```

Tiếp theo nhóm em sẽ xây dựng mô hình

Đầu tiên nhóm em sẽ xây dựng mô hình linear regression

```
## Linear Regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

```
LinearRegression
LinearRegression()
```

Ở đây nhóm em sẽ sử dụng tập train để xây dựng mô hình

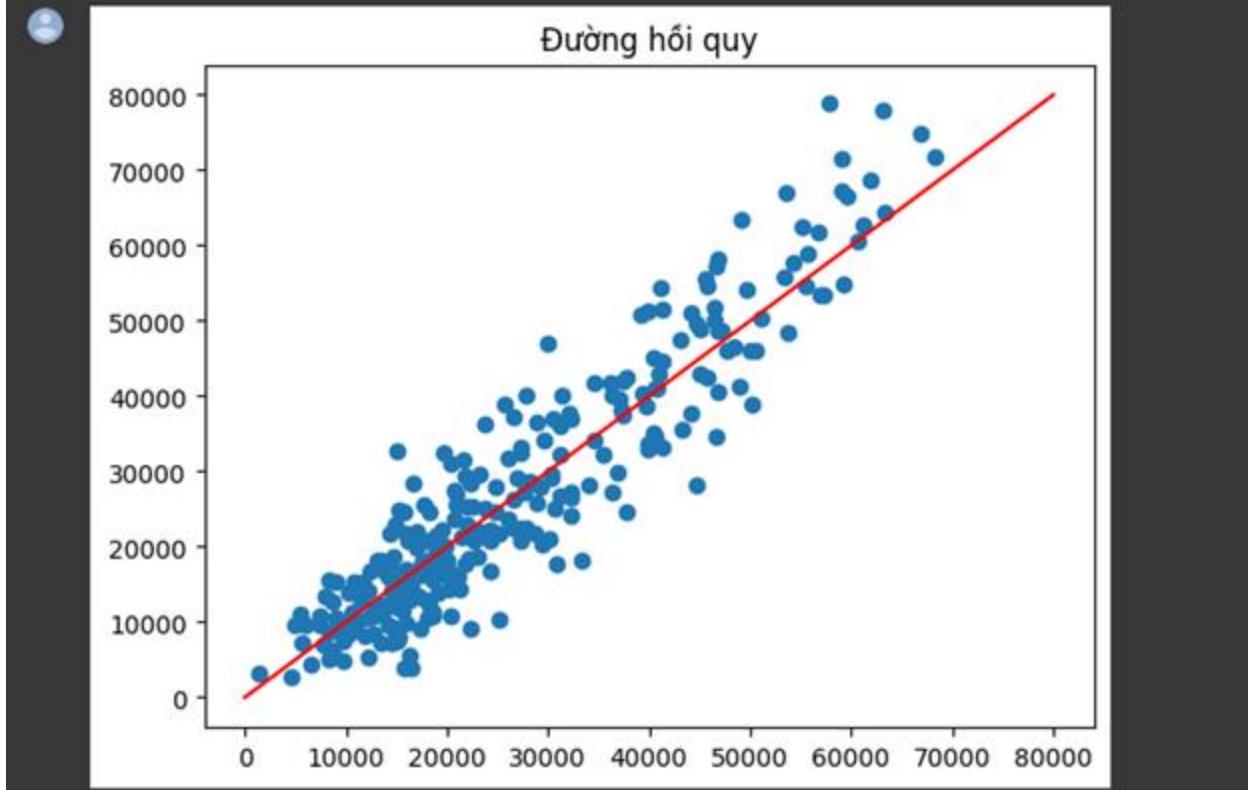
Tiếp đó nhóm em sẽ sử dụng MSE, MAE, R2

```
[ ] y_predict_li = model.predict(x_test)
print('MSE:', mean_squared_error(y_test,y_predict_li))
print('MAE:',mean_absolute_error(y_test,y_predict_li))
print('R2:',r2_score(y_test,y_predict_li))

MSE: 39102626.562452555
MAE: 4905.573760164137
R2: 0.8616368591611696
```

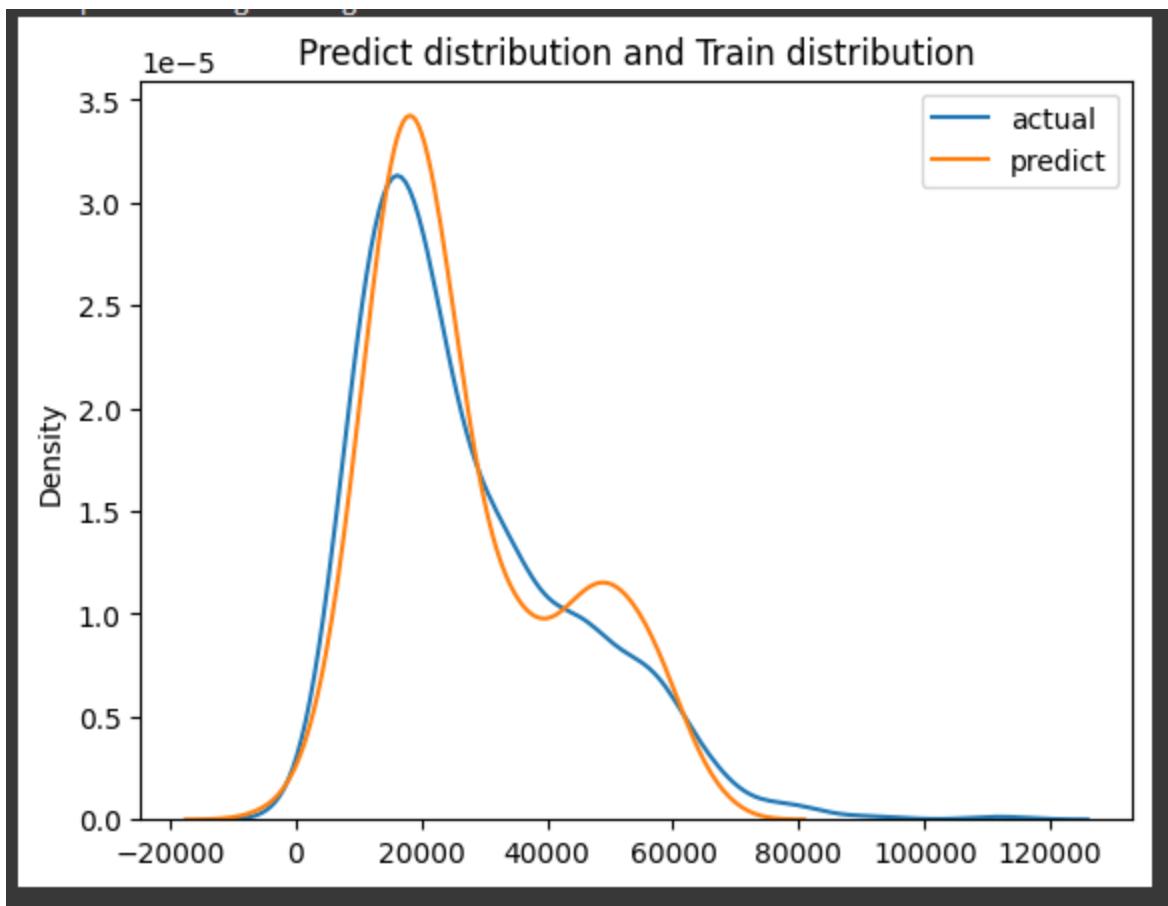
Và kết quả đạt được hoàn toàn cho thấy mô hình trên là mô hình hoàn toàn ổn khi R2 ở ngưỡng hoàn toàn chấp nhận được

```
plt.scatter(y_predict_li,y_test)
plt.plot([0,80_000],[0,80_000],color = 'r')
plt.title('Đường hồi quy')
plt.show()
```



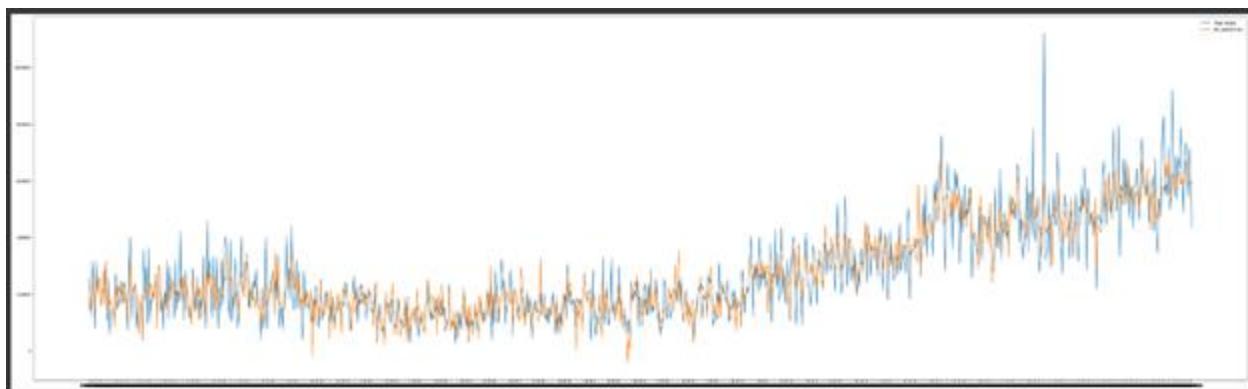
Sau đó em sẽ vẽ đường hồi quy để so sánh sự tương quan giữa tập test và tập dự đoán, thì có thể thấy rằng 2 tập này khá tương quan với nhau và có tương quan tuyến tính thuận

Và tiếp theo nhóm em sẽ so sánh giữa tập train và tập dự đoán



Và có thể thấy rằng 2 tập trên hoàn toàn có sự chênh lệch với nhau nhưng không nhiều

Và sau đó nhóm em sẽ so sánh độ chênh lệch của thực tế và dự đoán theo từng ngày của dữ liệu là có nhiều hay không bằng cách nhóm em sẽ sử dụng toàn bộ mô hình để so sánh



Có thể thấy rằng mức độ chênh lệch của thực tế và dự đoán là hoàn toàn không đáng kể, điều đó cho thấy rằng mô hình trên là một mô hình khá ổn

Nhưng để chắc chắn hơn, nhóm em sẽ tính MSE và MAE

```
[ ] mse_lst.append(mean_squared_error(train_data['total_bill'],train_data['lin_pred']))  
mae_lst.append(mean_absolute_error(train_data['total_bill'],train_data['lin_pred']))  
model_lst.append('LinearRegression')  
  
print(f'{model_lst[-1]}\nMAE:{mae_lst[-1]}\nMSE:{mse_lst[-1]}')  
  
LinearRegression  
MAE:4883.085353874862  
MSE:43039676.8961422
```

### 3.2. ElasticNet Regression

Gọi mô hình Elastic với tham số truyền vào với:

- Tham số điều chỉnh sự phụ thuộc vào hệ số như trong Ridge Regression và Lasso Regression (alpha) là 0.1.
- Tham số điều chỉnh tỉ lệ giữa thành phần L1 và L2 trong regularization là 0.5. Nghĩa là dùng 1 nửa mô hình Lasso và 1 nửa Ridge.

```
elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5, random_state=1)  
elastic_net.fit(x_train, y_train)
```

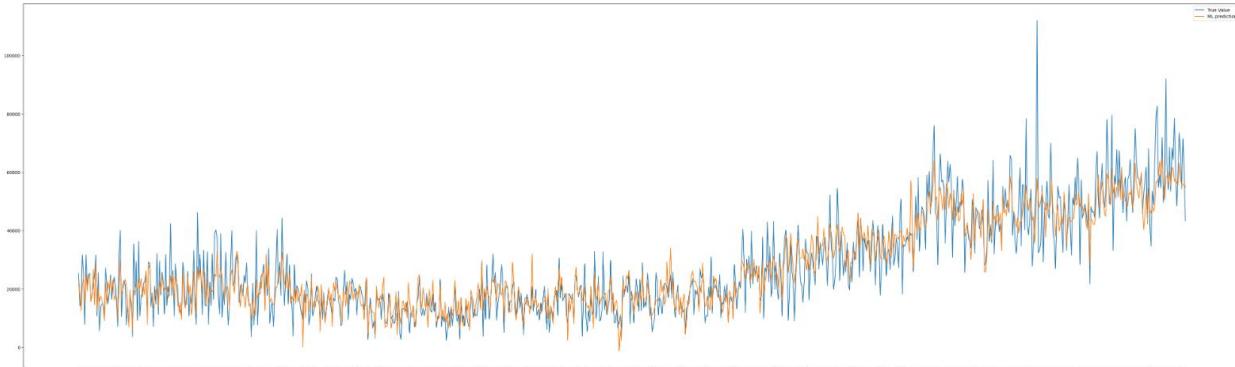
Đánh giá mô hình bằng MSE, MAE, R2, kết quả thu được dù mô hình không tốt hơn mô hình hồi quy tuyến tính trên tuy nhiên vẫn cao hơn các mô hình truyền thống.

```
y_pred = elastic_net.predict(x_test)  
mse = mean_squared_error(y_test, y_pred)  
print('MSE:', mean_squared_error(y_test,y_pred))  
print('MAE:',mean_absolute_error(y_test,y_pred))  
print('R2:',r2_score(y_test,y_pred))  
  
MSE: 43231834.869703226  
MAE: 5135.72100533715  
R2: 0.8470258143083003
```

Dùng mô hình để dự đoán cho toàn bộ khoảng thời gian.

```
[ ] elas_pred = elastic_net.predict(train_data[train_col])  
train_data['elas_pred'] = elas_pred  
  
plt.figure(figsize=(50,15))  
  
plt.plot(train_data['month_year_date'].values, train_data['total_bill'].values, label="True Value")  
plt.plot(train_data['month_year_date'].values, train_data['elas_pred'].values, label="ML prediction")  
plt.legend()  
plt.show()
```

Kết quả thu được cho thấy mô hình dự đoán tốt ở những điểm dữ liệu có độ lớn vừa và nhỏ, tuy nhiên đối với các điểm dữ liệu lớn thì chỉ đúng xu hướng tăng giảm.



### 3.3. Gradient Boosting Regression

Cuối cùng nhóm em sẽ xây dựng mô hình Gradient Boosting Regression

```
[ ] from sklearn.ensemble import GradientBoostingRegressor  
  
[ ] model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.022, max_depth=3)  
model.fit(x_train, y_train)  
  
/usr/local/lib/python3.9/dist-packages/sklearn/ensemble/_gb.py:437: DataConversionWarning  
y = column_or_1d(y, warn=True)  
  GradientBoostingRegressor  
GradientBoostingRegressor(learning_rate=0.022)
```

Trong đó sẽ có những tham số:

- n\_estimators: Số lượng các giai đoạn boosting để thực hiện. Tăng cường độ dốc khá mạnh đối với quá khớp, vì vậy một số lượng lớn thường mang lại hiệu suất tốt hơn. Gradient boosting mạnh đối với over-fitting vì vậy một số lượng lớn thường mang lại hiệu suất tốt hơn. Do vậy ở đây nhóm em sẽ cho tham số này là 100
- learning\_rate: Tỷ lệ học thu hẹp đóng góp của mỗi cây được đo bằng learning\_rate. Có sự đánh đổi giữa learning\_rate và n\_estimators. Nhóm em cho giá trị của tham số này là 0.022
- max\_depth: Độ sâu tối đa của các công cụ ước tính hồi quy riêng lẻ và nó để giới hạn số nút trong cây. Điều chỉnh tham số này để có hiệu suất tốt nhất;

giá trị tốt nhất phụ thuộc vào sự tương tác của các biến đầu vào. Ở đây nhóm em sẽ cho giá trị tham số này bằng 3

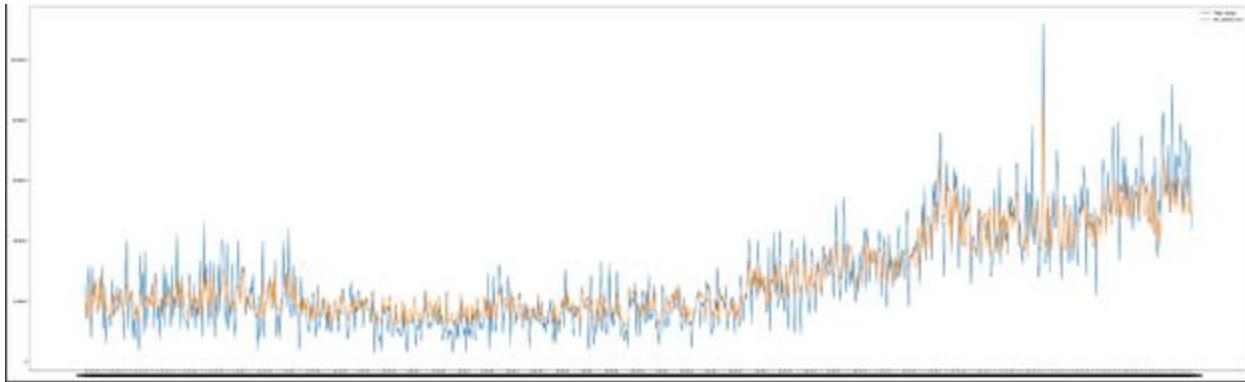
Sau đó nhóm em sẽ tính MSE, MAE, R2 để đánh giá mô hình

```
y_pred = model.predict(x_test)
mse = mean_squared_error(y_test, y_pred)
print('MSE:', mean_squared_error(y_test,y_pred))
print('MAE:',mean_absolute_error(y_test,y_pred))
print('R2:',r2_score(y_test,y_pred))

MSE: 51625940.50323122
MAE: 5606.864427870124
R2: 0.8173235942251333
```

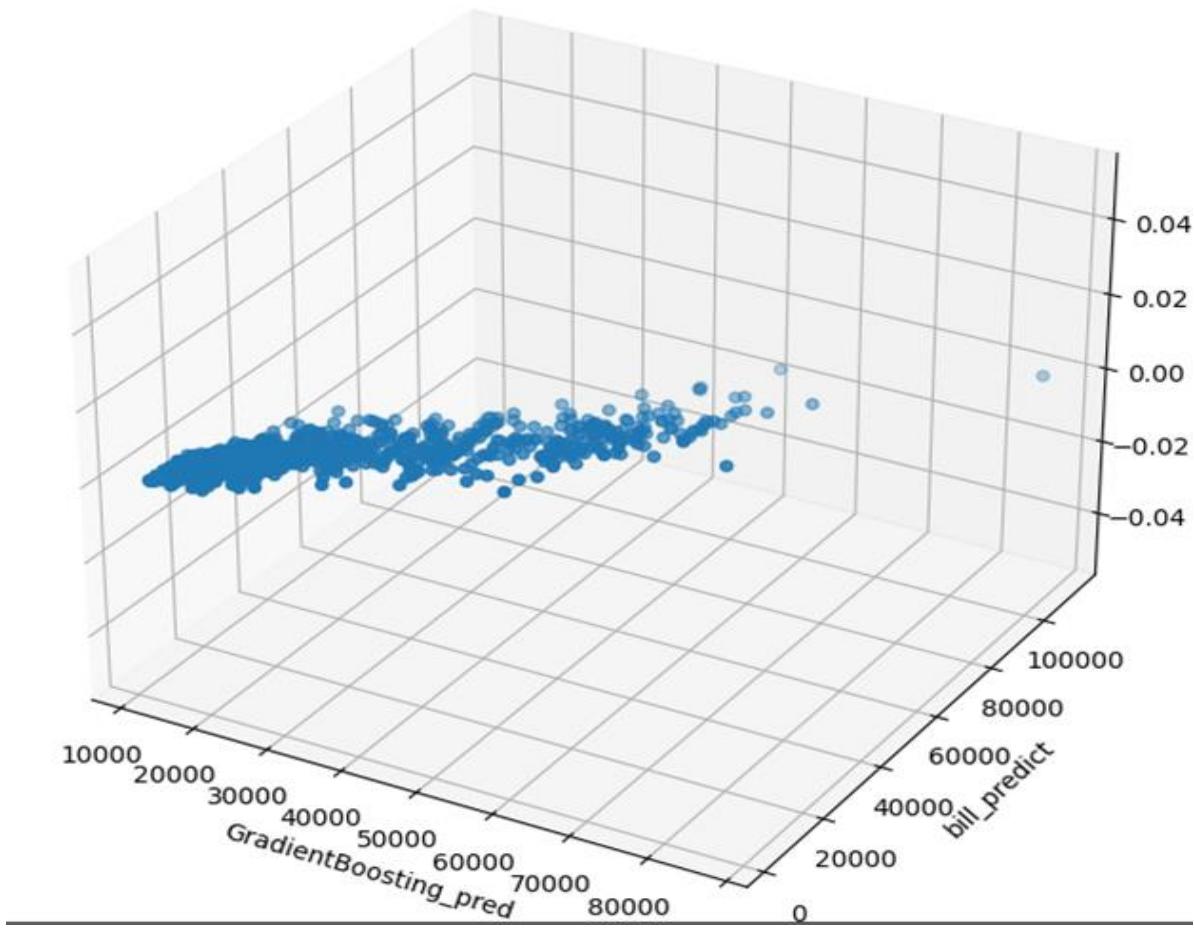
Kết quả thu được cho nhóm em thấy rằng mô hình này cũng là 1 mô hình hoàn toàn có thể chấp nhận được, nhưng mức độ học của mô hình này vẫn thua hơn mô hình hồi quy tuyến tính một xíu nhưng không quá quan ngại

Và sau đó nhóm em sẽ so sánh độ chênh lệch của thực tế và dự đoán theo từng ngày của dữ liệu là có nhiều hay không bằng cách nhóm em sẽ sử dụng toàn bộ mô hình để so sánh



Nhóm em thấy được sự chênh lệch giữa chúng là hoàn toàn có nhưng không quá đáng kể nên nhóm em thấy rằng đây là một mô hình hoàn toàn có thể chấp nhận được

Nhưng để chắc chắn hơn nhóm em sẽ trực quan chúng bằng 1 đồ thị 3d



Có thể thấy rằng các điểm dữ liệu khá gần nhau, điều này cho thấy rằng mức độ tương đồng của mô hình thực và dự đoán là hoàn toàn giống nhau

Và cuối cùng nhóm em sẽ tính MAE và MSE của chúng

```

▶ mse_lst.append(mean_squared_error(train_data['total_bill'],train_data['GradientBoosting_pred']))
mae_lst.append(mean_absolute_error(train_data['total_bill'],train_data['GradientBoosting_pred']))
model_lst.append('GradientBoosting')

print(f'{model_lst[-1]}\nMAE:{mae_lst[-1]}\nMSE:{mse_lst[-1]}')

● GradientBoosting
MAE:4830.464875369504
MSE:40400247.77896728

```

#### 4. Mô hình học sâu - Long Short-Term Memory

Sử dụng các thư viện phổ biến hỗ trợ xây dựng mạng LSTM, cụ thể ở đây là Keras và TensorFlow

```
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
import tensorflow as tf
```

Định nghĩa kiến trúc của LSTM gồm các bước sau:

```
model = Sequential()
model.add(tf.keras.layers.LSTM(1024, activation='relu', input_shape=(1, x_train.shape[1]), return_sequences=True))
model.add(tf.keras.layers.Dense(1024, activation='relu'))
model.add(LSTM(512, activation='relu'))
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dense(64, activation='relu'))
model.add(tf.keras.layers.Dense(32, activation='relu'))
model.add(tf.keras.layers.Dense(16, activation='relu'))
model.add(Dense(1, 'relu'))
model.compile(optimizer=adam, loss=MeanAbsoluteError())
```

- Khởi tạo mô hình LSTM

```
model = Sequential()
```

- Thêm lớp LSTM đầu tiên với 1024 đơn vị, lớp tiếp theo sẽ là một lớp ẩn có 1024 đơn vị, lớp tiếp theo là lớp LSTM có 512 đơn vị và tiếp theo là một chuỗi lớp ẩn có số đơn vị lần lượt là 256, 128, 64, 32, 16 và lớp cuối cùng là có 1 đơn vị là đầu ra của mô hình. Hàm kích hoạt cho mỗi lớp là ‘relu’. Hàm kích hoạt được sử dụng để đưa ra quyết định về giá trị đầu ra của mỗi đơn vị trong mạng, hàm các hoạtReLU thường dùng trong các lớp Fully Connecte của mạng nơ-ron để giúp mô hình học được các đặc trưng phức tạp từ dữ liệu đầu vào.

```
model.add(tf.keras.layers.LSTM(1024, activation='relu', input_shape=(1, x_train.shape[1]), return_sequences=True))
model.add(tf.keras.layers.Dense(1024, activation='relu'))
model.add(LSTM(512, activation='relu'))
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dense(64, activation='relu'))
model.add(tf.keras.layers.Dense(32, activation='relu'))
model.add(tf.keras.layers.Dense(16, activation='relu'))
model.add(Dense(1, 'relu'))
```

- Biên dịch mô hình với hàm mất mát và thuật toán tối ưu. tham số thiết lập thuận toán tối ưu cho quá trình huấn luyện là Adam, và hàm mất mát ở đây là Mean Absolute Error.

```
model.compile(optimizer=adam, loss=MeanAbsoluteError())
```

- Đối với mô hình trên thì nhóm có tinh chỉnh các tham số trong tham số của thuật toán Adam như sau:

```
adam = tf.keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=True,
    weight_decay=None,
    clipnorm=True,
    clipvalue=1.0,
    global_clipnorm=None,
    use_ema=True,
    ema_momentum=0.99,
    ema_overwrite_frequency=None,
    jit_compile=True,
    name='Adam'
)
```

- learning\_rate: Tốc độ học của mô hình, tức là mức độ điều chỉnh các trọng số của mạng dựa trên độ lớn của đạo hàm của hàm mất mát. Giá trị mặc định là 0.001.
- beta\_1 và beta\_2: Hệ số giảm gradient cho momentum của gradient và momentum của độ dốc vuông. Giá trị mặc định lần lượt là 0.9 và 0.999.
- epsilon: Đây là một giá trị nhỏ được thêm vào mẫu số trong công thức tính toán của Adam để tránh chia cho 0. Giá trị mặc định là 1e-07.
- amsgrad: Đây là một cờ đánh dấu để sử dụng phương pháp AMSGrad (Adam with AMSGrad) hay không. Giá trị mặc định là False.
- weight\_decay: Đây là hệ số giảm trọng lượng (weight decay) được áp dụng để giảm quá khớp (overfitting) trong quá trình đào tạo. Giá trị mặc định là None, tức là không áp dụng weight decay.
- clipnorm: Đây là một cờ đánh dấu để sử dụng việc cắt (clipping) theo norm của gradient hay không. Giá trị mặc định là True.
- clipvalue: Đây là giá trị tối đa của gradient sau khi được cắt (clipped). Giá trị mặc định là 1.0.
- global\_clipnorm: Đây là giá trị tối đa của norm của gradient toàn cục sau khi được cắt (clipped). Giá trị mặc định là None.
- use\_ema: Đây là một cờ đánh dấu để sử dụng Exponential Moving Average (EMA) hay không. Giá trị mặc định là True.
- ema\_momentum: Đây là hệ số momentum của EMA. Giá trị mặc định là 0.99.

- ema\_overwrite\_frequency: Số lần cập nhật trước khi ghi đè lên các giá trị EMA trước đó. Giá trị mặc định là None, tức là không ghi đè.
- jit\_compile: Đây là một cờ đánh dấu để sử dụng Just-In-Time (JIT) compilation hay không. Giá trị mặc định là True.
- name: Tên của đối tượng tối ưu

Tổng kết lại mô hình:

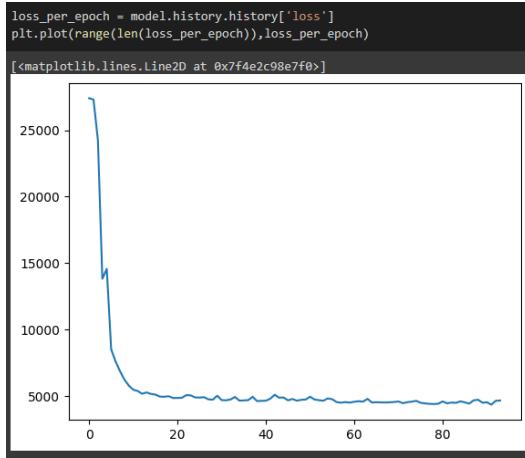
```
model.summary()
Model: "sequential"
Layer (type)      Output Shape       Param #
=====
lstm (LSTM)      (None, 1, 1024)    4239360
dense (Dense)    (None, 1, 1024)    1049600
lstm_1 (LSTM)    (None, 512)        3147776
dense_1 (Dense)  (None, 256)        131328
dense_2 (Dense)  (None, 128)        32896
dense_3 (Dense)  (None, 64)         8256
dense_4 (Dense)  (None, 32)         2080
dense_5 (Dense)  (None, 16)         528
dense_6 (Dense)  (None, 1)          17
=====
Total params: 8,611,841
Trainable params: 8,611,841
Non-trainable params: 0
```

Mô hình có tổng cộng 8,611,841 tham số, gồm các lớp LSTM, Dense với đầu vào và đầu ra tương ứng, và đều có khả năng được đào tạo (trainable).

Huấn luyện mô hình bằng những dữ liệu chuẩn bị ở bước trước để huấn luyện mô hình LSTM, cần phải chia dữ liệu thành dữ liệu huấn luyện và dữ liệu kiểm tra (validation data) để đánh giá hiệu suất của mô hình. Ngoài ra nhóm có sử dụng thêm tham số callback trong keras để kiểm soát quá trình đào tạo, cụ thể ở đây là EarlyStopping: Dừng quá trình đào tạo sớm nếu mất mát trên tập kiểm tra không cải thiện sau một số lượng epoch nhất định.

```
# Train the LSTM model
model.fit(x_train_reshaped, y_train_test, epochs=500, batch_size=64, validation_data=(x_test_reshaped, y_test_reshaped), callbacks=[early_stopping])
```

Sau khi hoàn thành quá trình huấn luyện, để đánh giá mô hình bằng cách sử dụng dữ liệu kiểm tra và độ đo đánh giá như ở đây độ mất mát (loss)



Từ hình trên ta có thể thấy được trong quá trình đào tạo mô hình, giá trị loss trên tập huấn luyện giảm dần và không có dấu hiệu overfitting (quá khớp), và giá trị loss cũng giảm và ổn định dần, thì có thể cho rằng mô hình hoạt động tốt.

Tiến hành kiểm tra thêm bằng các độ đo như MSE, MAE, R2 ta có thể thấy được mô hình LSTM cho ra một kết quả khá tốt và vượt trội và vẫn có thể tìm cách để cải thiện điểm số này.

```

# Predict on the test data
y_Pred = model.predict(x_train_reshaped)

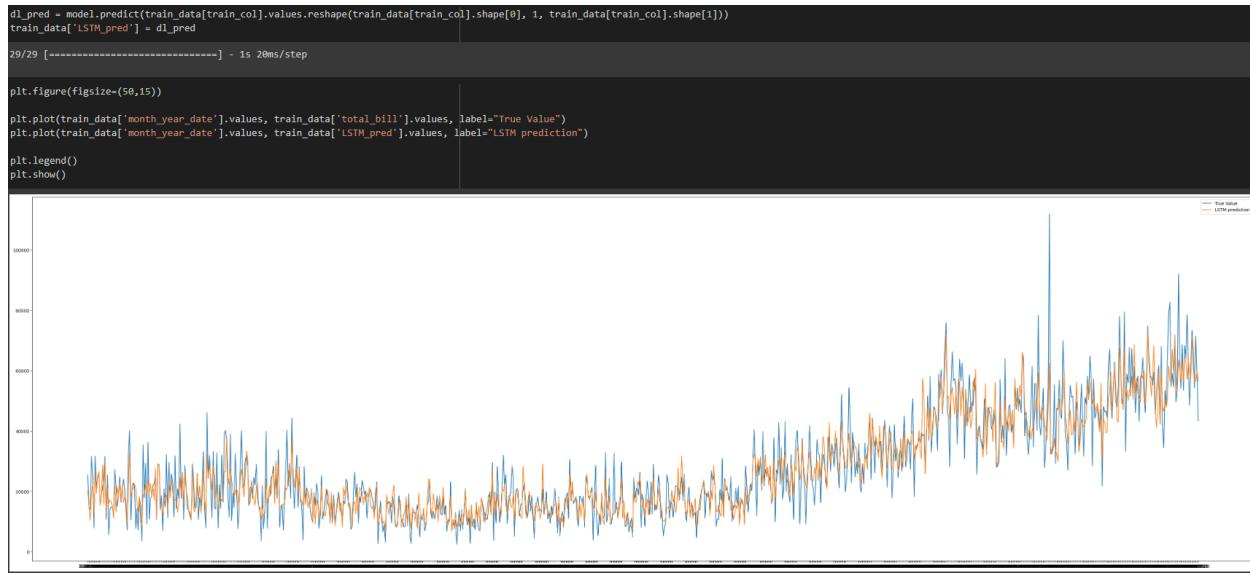
# Evaluate the model
mse = mean_squared_error(y_train_test, y_pred)
mae = mean_absolute_error(y_train_test, y_pred)
r2 = r2_score(y_train_test, y_pred)

print('Mean Squared Error (MSE):', mse)
print('Mean Absolute Error (MAE):', mae)
print('R2 Score:', r2)

20/20 [=====] - 1s 21ms/step
Mean Squared Error (MSE): 38893421.44177594
Mean Absolute Error (MAE): 4313.049484335443
R2 Score: 0.8628479261318911

```

Dùng mô hình để dự đoán cho toàn bộ chuỗi thời gian:



Kết quả thu được cho thấy mô hình dự đoán tốt ở toàn bộ dữ liệu nhưng ta có thể dễ dàng thấy được mô hình vẫn không thể bám sát vào những điểm dữ liệu cực đại hay cực tiểu, tuy nhiên đây vẫn được coi là một mô hình khá ổn định.

## Chương 5: Kết luận

### 1. Kết luận

	model	MSE	MAE
0	MA	8.338891e+07	6848.756836
1	EMA	4.705238e+07	5143.066816
2	AutoReg	6.324758e+07	6000.889917
3	ARIMA	6.915089e+07	6216.605856
4	LinearRegression	4.303968e+07	4883.085354
5	ElasticNet	4.586759e+07	5003.442356
6	GradientBoosting	4.040025e+07	4830.464875
7	LSTM	3.929122e+07	4506.485007

So sánh MSE và MAE giữa các mô hình truyền thống và các mô hình học máy, học sâu, dễ dàng thấy được lỗi giữa các giá trị thực tế và dự đoán của các mô hình máy học, học sâu là nhỏ hơn các mô hình truyền thống. Từ đó thấy được sự tối ưu của việc sử dụng máy học và học sâu cùng với các thuộc tính phụ thuộc khác có tác động đến doanh thu để dự đoán doanh thu.

Đối với các mô hình truyền thống việc dự đoán sử dụng EMA cho ra kết quả tối ưu nhất với điểm lỗi ghi nhận được là xấp xỉ mô hình học máy Elastic Net

Đối với mô hình học máy, mô hình Gradient Boosting cho ra ít lỗi nhất khi so sánh dự đoán với thực tế. Mô hình học sâu LSTM là mô hình tốt nhất trong tất cả mô hình dù vẫn chưa đạt được sự tối ưu tốt nhất trong MSE nhưng lại ghi nhận sự khác biệt đáng kể trong MAE

### 2. Hướng phát triển

Tối ưu mô hình học sâu, xây dựng số layers hợp lý giúp làm giảm thời gian và tài nguyên thực hiện.

Điều chỉnh optimizer để tối ưu kết quả cho mô hình.

Sử dụng thêm các biến khác có tác động đến kết quả để tăng độ chính xác mô hình.

Điều chỉnh các tham số liên quan để tránh hiện tượng overfit và đa cộng tuyến.

## PHỤ LỤC

Link source code	<a href="https://github.com/bthcong2002/-n-ML/blob/master/ML_Final.ipynb">https://github.com/bthcong2002/-n-ML/blob/master/ML_Final.ipynb</a>
Link data	<a href="https://docs.google.com/spreadsheets/d/16PAysBTrwTOj2bJAkVgxJBlqJouMJbsVnd3fpY3VAVM/edit#gid=0">https://docs.google.com/spreadsheets/d/16PAysBTrwTOj2bJAkVgxJBlqJouMJbsVnd3fpY3VAVM/edit#gid=0</a>

## TÀI LIỆU THAM KHẢO

- [https://phamdinhkhanh.github.io/2019/04/22/Ly\\_thuyet\\_ve\\_mang\\_LS\\_TM.html](https://phamdinhkhanh.github.io/2019/04/22/Ly_thuyet_ve_mang_LS_TM.html)
- [https://vi.wikipedia.org/wiki/B%e1%bb%99\\_nh%e1%bb%9b\\_d%C3%A0i-ng%E1%BA%AFn\\_h%E1%BA%A1n#Li%C3%AAn\\_k%E1%BA%BFt\\_ng%C3%A0i](https://vi.wikipedia.org/wiki/B%e1%bb%99_nh%e1%bb%9b_d%C3%A0i-ng%E1%BA%AFn_h%E1%BA%A1n#Li%C3%AAn_k%E1%BA%BFt_ng%C3%A0i)
- <https://viblo.asia/p/tong-quan-ve-du-bao-chuoi-thoi-gian-y37LdwGYJov>
- <https://cafedev.vn/tu-hoc-mlcac-loai-ky-thuat-hoi-quyregrression/>
- [https://viblo.asia/p/du-doan-doanh-thu-phim-tai-box-office-voi-hoi-quy-tuyen-tinh-Eb85oGQ252G#\\_1-quy-trinh-lam-viec-khoa-hoc-du-lieu-1](https://viblo.asia/p/du-doan-doanh-thu-phim-tai-box-office-voi-hoi-quy-tuyen-tinh-Eb85oGQ252G#_1-quy-trinh-lam-viec-khoa-hoc-du-lieu-1)
- <https://200lab.io/blog/predicting-sales/>
- <https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp#:~:text=What%20Is%20ARIMA%20Used%20for,points%20influence%20future%20data%20points.>
- <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.ewm.html>
- <https://viblo.asia/p/linear-regression-hoi-quy-tuyen-tinh-trong-machine-learning-4P856akRIY3>
- <https://machinelearningcoban.com/2016/12/28/linearregression/>
- [https://bvag.com.vn/wp-content/uploads/2013/01/k2\\_attachments\\_PHAN-TICH-HOI-QUY-TUYEN-TINH-DON-GIAN.pdf](https://bvag.com.vn/wp-content/uploads/2013/01/k2_attachments_PHAN-TICH-HOI-QUY-TUYEN-TINH-DON-GIAN.pdf)
- [https://en.wikipedia.org/wiki/Gradient\\_boosting](https://en.wikipedia.org/wiki/Gradient_boosting)