**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN**

# BÀI TẬP LỚN

## HỌC MÁY CƠ BẢN

## Dự Đoán Thời Gian Di Chuyển Dựa Trên Dữ Liệu Hành Trình Sử Dụng Xe Đạp

NGÀNH: KHOA HỌC MÁY TÍNH

SINH VIÊN: **ĐỖ VĂN LINH**

MÃ SINH VIÊN: **12423040**

LỚP: **12423TN**

HƯỚNG DẪN: **PGS.TS. NGUYỄN VĂN HẬU**

HƯNG YÊN - 2025

# NHẬN XÉT

**Nhận xét của giảng viên hướng dẫn:**

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

**GIÁO VIÊN HƯỚNG DẪN**

**Nguyễn Văn Hậu**

# LỜI CAM ĐOAN

Em cam kết rằng bài tập **"Dự Đoán Thời Gian Di Chuyển Dựa Trên Dữ Liệu Hành Trình Sử Dụng Xe Đạp"** là kết quả nghiên cứu do chính em thực hiện. Nội dung sử dụng trong bài tập không sao chép từ bất kỳ tài liệu nào. Các nội dung được trích dẫn đều được thực hiện đúng theo quy định về bản quyền và tránh vi phạm bản quyền. Những kết quả trình bày trong bài tập hoàn toàn là kết quả nghiên cứu của cá nhân em dưới sự hướng dẫn của giảng viên. Nếu có bất kỳ sai sót nào, em xin hoàn toàn chịu trách nhiệm trước khoa và nhà trường.

Hung Yen, 05 tháng 01 năm 2026

**Sinh viên thực hiện**

Đỗ Văn Linh

# LỜI CẢM ƠN

Để có thể hoàn thành bài tập này, trước hết em xin bày tỏ lòng biết ơn sâu sắc đến **Khoa Công nghệ Thông tin – Trường Đại học Sư phạm Kỹ thuật Hưng Yên** đã tạo điều kiện thuận lợi để em thực hiện bài tập.

Đặc biệt, em xin gửi lời cảm ơn chân thành đến **thầy Nguyễn Văn Hậu** đã tận tình hướng dẫn và chỉ bảo em trong suốt quá trình thực hiện. Những ý kiến chuyên môn quý báu cùng sự hỗ trợ liên tục của thầy đã góp phần quan trọng vào việc hoàn thành thành công đề tài này.

Em cũng xin chân thành cảm ơn toàn thể các thầy cô trong nhà trường đã tận tâm giảng dạy, trang bị cho em những kiến thức cần thiết và quý báu, giúp em có nền tảng vững chắc để thực hiện bài tập này. Sự tận tụy của các thầy cô trong công tác giảng dạy là nguồn động viên và cảm hứng lớn trong suốt quá trình học tập của em.

Mặc dù đã cố gắng hết sức, song do năng lực còn hạn chế, bài tập không tránh khỏi những thiếu sót trong quá trình thực hiện. Em rất mong nhận được những ý kiến đóng góp và nhận xét từ các thầy cô để có thể hoàn thiện hơn và rút ra nhiều kinh nghiệm quý báu.

Xin trân trọng cảm ơn các thầy cô đã dành thời gian, sự quan tâm và hướng dẫn cho em!

# MỤC LỤC

# DANH MỤC HÌNH ẢNH

# CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN

## 1.1 Bài toán

Thành phố Oslo là một trong những đô thị tiên phong tại châu Âu trong việc phát triển giao thông bền vững, đặc biệt là khuyến khích sử dụng xe đạp như một phương tiện di chuyển thân thiện với môi trường. Cùng với sự gia tăng mạnh mẽ của các hệ thống xe đạp công cộng và hạ tầng dành riêng cho xe đạp, lượng dữ liệu hành trình được thu thập ngày càng lớn và có giá trị cao cho việc phân tích và nghiên cứu. Dựa trên dữ liệu hành trình xe đạp tại Oslo, chúng ta có thể khai thác nhiều thông tin quan trọng liên quan đến thời gian di chuyển, hành vi người dùng, cũng như các yếu tố ảnh hưởng đến hiệu quả và tốc độ di chuyển trong đô thị.

Dữ liệu hành trình xe đạp thường được thu thập từ các hệ thống xe đạp công cộng, thiết bị định vị GPS và các nền tảng giao thông thông minh, đảm bảo độ tin cậy và tính cập nhật cao. Việc phân tích dữ liệu này cho phép nhận diện các xu hướng di chuyển phổ biến, đánh giá ảnh hưởng của các yếu tố như khoảng cách, thời gian trong ngày, điều kiện thời tiết, địa hình và mật độ giao thông đến thời gian di chuyển của người dùng xe đạp. Qua đó, bài toán dự đoán thời gian di chuyển không chỉ mang ý nghĩa kỹ thuật mà còn phản ánh thực tiễn vận hành của hệ thống giao thông đô thị.

Việc dự đoán chính xác thời gian di chuyển bằng xe đạp mang lại nhiều lợi ích thiết thực. Đối với người sử dụng, kết quả dự đoán giúp họ lựa chọn lộ trình tối ưu, chủ động sắp xếp thời gian và nâng cao trải nghiệm di chuyển hàng ngày. Đối với các nhà quản lý đô thị và đơn vị vận hành hệ thống xe đạp công cộng, phân tích và dự đoán dữ liệu hành trình hỗ trợ việc quy hoạch hạ tầng, phân bổ xe hợp lý và cải thiện hiệu suất vận hành. Ngoài ra, kết quả nghiên cứu còn góp phần hỗ trợ các nhà hoạch định chính sách trong việc thúc đẩy giao thông xanh và phát triển đô thị bền vững.

## 1.2 Trình bày dữ liệu bài toán

- Dữ liệu bài toán được lấy từ Kaggle với tên : Oslobysykkel-2025

| | started_at | ended_at | duration | start_station_id | start_station_name | start_station_description | start_station_latitude | start_station_lon |
|---|---|---|---|---|---|---|---|---|
| 0 | 2025-06-01 03:00:18.285000+00:00 | 2025-06-01 03:13:57.293000+00:00 | 819 | 555 | Griffenfeldts gate | ved Colletts gate | 59.933703 | 10. |
| 1 | 2025-06-01 03:00:20.204000+00:00 | 2025-06-01 03:16:36.936000+00:00 | 976 | 735 | Oslo Hospital | ved trikkestoppet | 59.903213 | 10. |
| 2 | 2025-06-01 03:03:45.692000+00:00 | 2025-06-01 03:12:55.683000+00:00 | 549 | 485 | Sommerfrydhagen | langs Jens Bjelkes gate | 59.911453 | 10. |
| 3 | 2025-06-01 03:05:09.876000+00:00 | 2025-06-01 03:23:25.633000+00:00 | 1095 | 387 | Studenterlunden | langs Karl Johan | 59.914586 | 10. |
| 4 | 2025-06-01 03:06:09.611000+00:00 | 2025-06-01 03:12:56.455000+00:00 | 406 | 623 | 7. juni-plassen | langs Henrik Ibsens gate | 59.915080 | 10. |

Hình 1. 1: Dữ liệu của bộ Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025963 entries, 0 to 1025962
Data columns (total 13 columns):
 #   Column                      Non-Null Count    Dtype
---  ------                      --------------    -----
 0   started_at                  1025963 non-null  object
 1   ended_at                    1025963 non-null  object
 2   duration                    1025963 non-null  int64
 3   start_station_id            1025963 non-null  int64
 4   start_station_name          1025963 non-null  object
 5   start_station_description   1025925 non-null  object
 6   start_station_latitude      1025963 non-null  float64
 7   start_station_longitude     1025963 non-null  float64
 8   end_station_id              1025963 non-null  int64
 9   end_station_name            1025963 non-null  object
 10  end_station_description     1025925 non-null  object
 11  end_station_latitude        1025963 non-null  float64
 12  end_station_longitude       1025963 non-null  float64
dtypes: float64(4), int64(3), object(6)
memory usage: 101.8+ MB
```

Hình 1. 2: Thông tin chi tiết từng cột của Dataset

- Dữ liệu bài toán gồm các cột sau:

  started_at: Thời gian bắt đầu chuyến đi

  ended_at: Thời gian kết thúc chuyến đi

  duration: Thời gian di chuyển (s)

  start_station_id: Mã trạm bắt đầu

  start_station_name: Tên trạm bắt đầu

  start_station_description: Mô tả trạm bắt đầu

  start_station_latitude: Vĩ độ trạm đầu

  start_station_longitude: Kinh độ trạm đầu

end_station_id: Mã trạm kết thúc

end_station_name: Tên trạm kết thúc

end_station_description: Mô tả trạm kết thúc

end_station_latitude: Vĩ độ trạm cuối

end_station_longitude: Kinh độ trạm cuối

- Dữ liệu của bài toán là 1 file csv gồm 1025963 rows x 13 columns
    - File csv: oslobysykkel-2025.csv
    - Có 13 cột và mỗi cột có 1025963 bản ghi
- Sau khi mô tả dữ liệu, ta có:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| duration | 1025963.0 | 770.474425 | 850.175764 | 61.000000 | 366.000000 | 564.000000 | 865.000000 | 17888.000000 |
| start_station_id | 1025963.0 | 731.631984 | 750.564761 | 377.000000 | 438.000000 | 501.000000 | 591.000000 | 5431.000000 |
| start_station_latitude | 1025963.0 | 59.921159 | 0.010675 | 59.898434 | 59.912713 | 59.919524 | 59.928067 | 59.953411 |
| start_station_longitude | 1025963.0 | 10.746971 | 0.024395 | 10.651118 | 10.730589 | 10.750847 | 10.762213 | 10.814314 |
| end_station_id | 1025963.0 | 746.309637 | 756.369919 | 377.000000 | 442.000000 | 500.000000 | 593.000000 | 5431.000000 |
| end_station_latitude | 1025963.0 | 59.918741 | 0.009577 | 59.898434 | 59.911776 | 59.916065 | 59.924732 | 59.953411 |
| end_station_longitude | 1025963.0 | 10.745458 | 0.023320 | 10.651118 | 10.731219 | 10.750462 | 10.760804 | 10.814314 |

Hình 1. 3: Mô tả dữ liệu từng cột của Dataset

### 1.3 Tiền xử lý dữ liệu

a) Xử lý thông tin các cột: started_at, ended_at

b) Xử lý dữ liệu trùng lặp, Nan

c) Logarit hóa dữ liệu cột duration

d) Taget_encoding dữ liệu cột start_station_id theo cột duration

e) Taget_encoding dữ liệu cột end_station_id theo cột duration

f) Xóa các cột không cần thiết: start_station_name, start_station_description, end_station_name, end_station_description

g) Loại bỏ dữ liệu ngoại lệ

### 1.4 Trực quan hóa dữ liệu

a) Biểu đồ boxplot cho phân phối thời gian di chuyển

b) Biểu đồ phân bố Trạm

c) Biểu đồ số lượng chuyến đi theo Trạm

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1 Pandas

### 2.1.1 Giới thiệu về Pandas

Pandas là một thư viện mã nguồn mở phổ biến trong Python, được xây dựng trên nền tảng NumPy, chuyên dùng để thao tác và phân tích dữ liệu. Với khả năng xử lý các loại dữ liệu đa dạng như chuỗi thời gian, bảng không đồng nhất và ma trận dữ liệu, Pandas mang lại trải nghiệm làm việc trực quan và linh hoạt cho người dùng.

Thư viện này cho phép import dữ liệu từ nhiều nguồn như CSV, cơ sở dữ liệu SQL, và các định dạng đặc biệt khác. Sau khi nhập dữ liệu, người dùng có thể thực hiện các thao tác như chọn lọc (subsetting), cắt nhỏ (slicing), lọc (filtering), hợp nhất (merging), nhóm (groupBy), sắp xếp (re-ordering), và tái cấu trúc (re-shaping) dữ liệu một cách dễ dàng. Pandas cũng hỗ trợ xử lý dữ liệu mất mát bằng cách loại bỏ hoặc thay thế bằng giá trị khác.

Ngoài ra, Pandas tích hợp tốt với các thư viện như Matplotlib, SciPy và Scikit-Learn, giúp mở rộng khả năng phân tích và trực quan hóa. Với hiệu suất cao, Pandas có thể xử lý khối lượng dữ liệu lớn hiệu quả, là công cụ không thể thiếu cho các nhà phân tích dữ liệu.

### 2.1.2 Cài đặt Pandas

Để cài đặt Pandas, bạn có thể sử dụng hai phương pháp chính: Anaconda và Pip.

- **Anaconda**: Anaconda là một nền tảng phổ biến cho khoa học dữ liệu và học máy, cung cấp một môi trường dễ sử dụng để quản lý các gói và môi trường ảo. Để cài đặt Pandas thông qua Anaconda, bạn có thể sử dụng lệnh sau:`conda install pandas`
- **Pip**: Pip là trình quản lý gói chuẩn cho Python. Nếu bạn không sử dụng Anaconda, bạn có thể cài đặt Pandas thông qua Pip bằng lệnh sau:`pip install pandas`.

Sau khi cài đặt, để sử dụng Pandas chúng ta cần khai báo: `import pandas as pd`. Cách viết ngắn gọn này là tiêu chuẩn phổ biến, giúp mã nguồn dễ đọc và tuân

theo các quy ước trong cộng đồng Python. Dù có thể đặt tên khác, nhưng dùng pd là lựa chọn tối ưu và được khuyến khích.dễ đọc và tuân theo các quy ước chung trong tài liệu và ví dụ.

## 2.2 Numpy

### 2.2.1 Giới thiệu về Numpy

NumPy là thư viện cốt lõi của Python dành cho khoa học máy tính, hỗ trợ xử lý các mảng nhiều chiều với kích thước lớn. Thư viện này cung cấp các hàm tối ưu hóa để thao tác trên các mảng, giúp thực hiện các phép tính nhanh chóng và hiệu quả. NumPy đặc biệt hữu ích trong các bài toán liên quan đến đại số tuyến tính, xử lý tín hiệu, và tính toán ma trận, làm cho việc thao tác và phân tích dữ liệu số trở nên đơn giản và hiệu quả hơn.

### 2.2.2 Cài đặt Numpy

Để cài đặt NumPy, bạn có thể sử dụng hai phương pháp chính: Anaconda và Pip.

- **Anaconda**: Anaconda là một nền tảng phổ biến cho khoa học dữ liệu và học máy, cung cấp một môi trường dễ sử dụng để quản lý các gói và môi trường ảo. Để cài đặt NumPy thông qua Anaconda, bạn có thể sử dụng lệnh sau`conda install numpy`
- **Pip**: Pip là trình quản lý gói chuẩn cho Python. Nếu không sử dụng Anaconda, bạn có thể cài đặt NumPy thông qua Pip bằng lệnh sau`pip install numpy`

Sau khi cài đặt, cần khai báo thư viện NumPy để sử dụng trong mã nguồn Python. Thông thường, NumPy được khai báo gọn lại bằng lệnh: `import numpy as np` . Việc sử dụng np làm bí danh đã trở thành quy ước chung trong cộng đồng Python, giúp mã ngắn gọn, dễ đọc và phù hợp với các tài liệu hướng dẫn tiêu chuẩn. Điều này không chỉ tiện lợi khi viết mã mà còn hỗ trợ tốt hơn trong việc cộng tác và tham khảo tài liệu.

## 2.3 Matplotlib

### 2.3.1 Giới thiệu về Matplotlib

Để hỗ trợ các suy luận thống kê, trực quan hóa dữ liệu là bước không thể thiếu, và Matplotlib là một công cụ phổ biến trong Python. Đây là thư viện vẽ đồ thị mạnh mẽ, đặc biệt hữu ích khi kết hợp với Python và NumPy. Trong Matplotlib, module Pyplot thường được sử dụng nhất, cung cấp giao diện thân thiện, tương tự MATLAB nhưng dựa trên Python và hoàn toàn mã nguồn mở.

Matplotlib cho phép tạo ra nhiều loại biểu đồ như biểu đồ đường, biểu đồ thanh, biểu đồ phân tán và hơn thế nữa, giúp phân tích dữ liệu hiệu quả và trực quan. Với sự linh hoạt và khả năng tùy chỉnh cao, người dùng dễ dàng tạo các hình ảnh trực quan đáp ứng nhu cầu cụ thể của mình.

### 2.2.2 Cài đặt Matplotlib

Để cài đặt Matplotlib, bạn có thể sử dụng hai phương pháp chính: Anaconda và Pip.

- **Anaconda**: Anaconda là một nền tảng phổ biến cho khoa học dữ liệu và học máy, cung cấp một môi trường dễ sử dụng để quản lý các gói và môi trường ảo. Lệnh cài đặt qua Annaconda : `conda install matplotlib`
- **Pip**: Pip là trình quản lý gói chuẩn cho Python. Lệnh cài đặt qua Pip: `pip install matplotlib`

Sau khi cài đặt, bạn có thể bắt đầu sử dụng Matplotlib trong mã nguồn Python bằng cách khai báo: import matplotlib.pyplot as plt. Đây là cách viết tắt phổ biến, được cộng đồng Python sử dụng rộng rãi, giúp mã nguồn dễ đọc và tuân theo các quy ước chuẩn trong tài liệu và ví dụ.

## 2.4 Joblib

Joblib là một thư viện Python được sử dụng để lưu trữ và tải lại các đối tượng Python, đặc biệt là các mô hình học máy, dưới dạng tệp nhị phân. Nó hỗ trợ việc nén và lưu trữ các đối tượng lớn như các mô hình học máy, ma trận, và các mảng NumPy. Joblib giúp tối ưu hóa quá trình lưu trữ và tái sử dụng các đối tượng mà không cần phải tái huấn luyện lại mô hình hoặc tính toán lại các giá trị.

Một trong những ưu điểm của Joblib là khả năng lưu trữ các đối tượng hiệu quả, đặc biệt khi đối tượng chứa các mảng NumPy lớn. Joblib thường được sử dụng

trong các bài toán học máy để lưu mô hình sau khi huấn luyện và dễ dàng tải lại khi cần thiết mà không mất thời gian tính toán lại từ đầu.

Cách sử dụng phổ biến của Joblib: Lưu đối tượng và tải đối tượng. Joblib thường được sử dụng thay thế cho pickle khi cần lưu trữ các mô hình học máy hoặc các đối tượng có kích thước lớn.

## 2.5 Random Forest

Random Forest là một thuật toán học máy mạnh mẽ thuộc nhóm học máy giám sát (supervised learning) và thường được sử dụng cho các bài toán phân loại và hồi quy. Thuật toán này kết hợp nhiều cây quyết định (decision trees) để tạo thành một "rừng" cây, và kết quả cuối cùng được đưa ra dựa trên kết quả từ tất cả các cây trong rừng.

### 2.5.1 Cấu trúc của Random Forest

Random Forest là một bộ phân loại học theo phương thức ensemble learning, tức là nó xây dựng nhiều mô hình đơn lẻ (các cây quyết định) và kết hợp chúng lại để đưa ra dự đoán chính xác hơn.

- Cây quyết định (Decision Trees): Mỗi cây quyết định trong Random Forest là một cây riêng biệt được xây dựng bằng cách sử dụng một tập con ngẫu nhiên của dữ liệu và các đặc trưng.
- Quy trình Ensemble: Sau khi các cây quyết định được xây dựng, kết quả của mỗi cây sẽ được lấy (dựa trên đa số phiếu bầu cho phân loại hoặc giá trị trung bình cho hồi quy) để đưa ra dự đoán cuối cùng.

### 2.5.2 Cách thức hoạt động của Random Forest

a, Tạo ra các cây quyết định:

- Tạo ra một tập hợp các mẫu ngẫu nhiên từ tập huấn luyện bằng cách sử dụng Bootstrap sampling (hay còn gọi là sampling với thay thế).
- Tại mỗi nút trong cây, thay vì thử tất cả các đặc trưng để chia, thuật toán chỉ chọn một tập con ngẫu nhiên của các đặc trưng để tạo sự đa dạng giữa các cây.

b, Dự đoán từ mỗi cây:

- Phân loại: Mỗi cây đưa ra một dự đoán phân loại. Random Forest sẽ chọn kết quả dự đoán mà cây quyết định chiếm số phiếu bầu cao nhất (phương pháp "majority vote").
- Hồi quy: Kết quả dự đoán là giá trị trung bình của tất cả các cây trong rừng.

### 2.5.3 Ứng dụng của Random Forest

- Phân loại: Ví dụ như phân loại email thành thư rác và không phải thư rác, phân loại khách hàng thành nhóm có thể mua và không thể mua, hoặc phân loại bệnh

nhân có nguy cơ mắc bệnh hay không.

- Hồi quy: Ví dụ như dự đoán giá trị của bất động sản, giá cổ phiếu, hay giá trị của các biến liên tục khác.
- Phân tích tầm quan trọng của đặc trưng: Random Forest có thể được sử dụng để phân tích các yếu tố quan trọng trong tập dữ liệu, giúp xác định các đặc trưng có ảnh hưởng mạnh đến dự đoán.

## 2.6 Linear Regression

Linear Regression (Hồi Quy Tuyến Tính) là một trong những thuật toán học máy cơ bản và phổ biến nhất, được sử dụng để mô hình hóa mối quan hệ giữa một (hoặc nhiều) biến độc lập (còn gọi là đặc trưng, feature) và một biến phụ thuộc (còn gọi là mục tiêu, target). Mục tiêu của Linear Regression là tìm ra một hàm số tuyến tính sao cho có thể dự đoán được giá trị của biến mục tiêu dựa trên giá trị của các biến độc lập.

### 2.6.1 Mô hình hồi quy tuyến tính

Hồi quy tuyến tính mô hình hóa mối quan hệ giữa các biến thông qua một đường thẳng (trong trường hợp một biến độc lập) hoặc một siêu phẳng (trong trường hợp nhiều biến độc lập). Công thức của một mô hình hồi quy tuyến tính là:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Trong đó:

- y là giá trị dự đoán (biến phụ thuộc).
- $x_1, x_{2'} \ldots, x_n$ là các biến độc lập (đặc trưng).
- $\beta_0$ là hệ số chặn (intercept), tức là giá trị của yyy khi tất cả các biến độc lập đều bằng 0.
- $\beta_1, \beta_2, \ldots, \beta_n$ là các hệ số góc (coefficients), biểu thị ảnh hưởng của từng biến độc lập đối với biến phụ thuộc.
- $\epsilon$ là sai số ngẫu nhiên (error term), thể hiện sự khác biệt giữa giá trị thực tế và giá trị dự đoán.

Mục tiêu của hồi quy tuyến tính là tìm các giá trị tối ưu cho các hệ số $\beta_0, \beta_1, \ldots, \beta_n$ sao cho lỗi dự đoán (difference between predicted and actual values) là nhỏ nhất. Phương pháp thường được sử dụng để tối ưu các hệ số này là Least Squares (phương pháp bình phương tối thiểu), trong đó ta tìm cách giảm thiểu tổng bình phương sai số giữa giá trị dự đoán và giá trị thực tế.

### 2.6.2 Loss Function

Hàm mất mát trong hồi quy tuyến tính là Mean Squared Error (MSE), tính toán trung bình bình phương sai số giữa các giá trị thực tế và giá trị dự đoán:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Trong đó:

- $y_i$ là giá trị thực tế của quan sát thứ iii,
- $\hat{y_i}$ là giá trị dự đoán của mô hình cho quan sát thứ iii,
- $n$ là số lượng quan sát.

### 2.6.3 Các loại Linear Regression

- Simple Linear Regression (Hồi Quy Tuyến Tính Đơn Giản): Khi có chỉ một biến độc lập (feature), mô hình trở thành một đường thẳng trong không gian 2 chiều.
- Multiple Linear Regression (Hồi Quy Tuyến Tính Đa Biến): Khi có nhiều biến độc lập, mô hình trở thành một siêu phẳng trong không gian n chiều.

- Khả năng tùy chỉnh cao: XGBRegressor cung cấp rất nhiều tham số để tối ưu hóa mô hình, bao gồm các tham số điều chỉnh độ sâu của cây, tốc độ học (learning rate), số lượng cây, tỷ lệ phân bổ mẫu (subsampling), v.v.
- Regularization: Một trong những điểm mạnh của XGBRegressor là khả năng sử dụng regularization (làm đều hóa) thông qua các tham số như alpha và lambda để tránh overfitting, giúp mô hình tổng quát tốt hơn với dữ liệu chưa thấy.

### 2.6.4 Base models (Mô hình cơ sở)

- Mô hình yếu hoặc các mô hình cơ sở là những mô hình đơn lẻ mà chúng ta sử dụng để học trên dữ liệu ban đầu. Các mô hình cơ sở có thể là bất kỳ thuật toán học máy nào (Ví dụ: Random Forest, XGBoost, Logistic Regression, ...).

- Mỗi mô hình cơ sở có thể có các loại giả thuyết và lỗi khác nhau. Stacking kết hợp các mô hình này để giảm thiểu sai số tổng thể.

### 2.6.5 Meta model (Mô hình thứ hai)

- Meta model là mô hình được huấn luyện trên các dự đoán của các mô hình cơ sở. Mô hình này học cách kết hợp các dự đoán từ các mô hình cơ sở để đưa ra dự đoán cuối cùng.
- Meta model có thể là bất kỳ mô hình học máy nào, nhưng một lựa chọn phổ biến là Logistic Regression hoặc Linear Regression, vì chúng rất phù hợp để kết hợp các dự đoán của các mô hình cơ sở.

### 2.6.6 Quy trình hoạt động

- Bước 1: Huấn luyện các mô hình cơ sở trên dữ liệu huấn luyện.
- Bước 2: Dự đoán từ các mô hình cơ sở sẽ được sử dụng làm đặc trưng đầu vào cho meta model.
- Bước 3: Huấn luyện mô hình meta trên các dự đoán đầu ra của các mô hình cơ sở.
- Bước 4: Dự đoán mới được thực hiện bằng cách sử dụng mô hình meta, kết hợp các dự đoán từ các mô hình cơ sở.

# CHƯƠNG 3: GIẢI PHÁP

## 3.1 Mã nguồn tiền xử lý dữ liệu

```
print(df.info())
```

```
print(df.describe())
```

```
print(df.dtypes)
```

Đầu tiên em sử dụng hàm df.info() để kiểm tra các feature nào bị thiếu và có kiểu dữ liệu thế nào.Em hiển thị lại hàm df và dùng lại hàm info để kiểm tra xem dữ liệu. Và hình ảnh dưới là kết quả sau khi đã điền các dữ liệu:

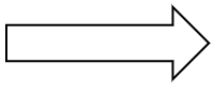
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025963 entries, 0 to 1025962
Data columns (total 13 columns):
 #   Column                    Non-Null Count    Dtype
---  ------                    --------------    -----
 0   started_at                1025963 non-null  object
 1   ended_at                  1025963 non-null  object
 2   duration                  1025963 non-null  int64
 3   start_station_id          1025963 non-null  int64
 4   start_station_name        1025963 non-null  object
 5   start_station_description 1025925 non-null  object
 6   start_station_latitude    1025963 non-null  float64
 7   start_station_longitude   1025963 non-null  float64
 8   end_station_id            1025963 non-null  int64
 9   end_station_name          1025963 non-null  object
 10  end_station_description   1025925 non-null  object
 11  end_station_latitude      1025963 non-null  float64
 12  end_station_longitude     1025963 non-null  float64
dtypes: float64(4), int64(3), object(6)
memory usage: 101.8+ MB
```

Hình 3. 1: Thông tin chi tiết từng cột của Dataset

a) Xử lý thông tin các cột:

- started_at: Thời gian bắt đầu chuyến đi
- ended_at: Thời gian kết thúc chuyến

| started_at | ended_at |
|---|---|
| 2025-06-01 03:00:18.285000+00:00 | 2025-06-01 03:13:57.293000+00:00 |
| 2025-06-01 03:00:20.204000+00:00 | 2025-06-01 03:16:36.936000+00:00 |
| 2025-06-01 03:03:45.692000+00:00 | 2025-06-01 03:12:55.683000+00:00 |
| 2025-06-01 03:05:09.876000+00:00 | 2025-06-01 03:23:25.633000+00:00 |
| 2025-06-01 03:06:09.611000+00:00 | 2025-06-01 03:12:56.455000+00:00 |



| start_time | start_dayofweek |
|---|---|
| 3.0 | 6.0 |
| 3.0 | 6.0 |
| 3.0 | 6.0 |
| 3.0 | 6.0 |
| 3.0 | 6.0 |

Hình 3. 2: Mã nguồn lọc và làm sạch dữ liệu

- Mã này chủ yếu chuyển kiểu dữ liệu của các cột về đúng định dạng để tạo thêm dữ liệu

```python
def formating_time(df):
    df['started_at'] = pd.to_datetime(df['started_at'], errors='coerce')
    df['ended_at'] = pd.to_datetime(df['ended_at'], errors='coerce')

    df['start_time'] = (df['started_at'].dt.hour)

    df['start_dayofweek'] = df['started_at'].dt.day_name()
    df['start_dayofweek'] = df['start_dayofweek'].map(
        {'Monday':0, 'Tuesday':1, 'Wednesday':2, 'Thursday':3, 'Friday':4, 'Saturday':5, 'Sunday':6}
    )
    # return df
formating_time(df)
```

Hình 3. 3: Mã nguồn chuyển đổi kiểu dữ liệu

- Hàm time_recovering() có chức năng đọc lại thời gian di chuyển một cách chính xác và nhất quán nếu chuyển kiểu dữ liệu tại một cột ra giá trị NaN

```python
def time_recovering(df):
    # for checking
    df['started_at_original'] = df['started_at'].copy()
    df['ended_at_original'] = df['ended_at'].copy()

    df.loc[df['ended_at'].isna() & df['started_at'].notna() & df['duration'].notna(), 'ended_at'] = \
      df['started_at'] + pd.to_timedelta(df['duration'], unit='s')

    df.loc[df['started_at'].isna() & df['ended_at'].notna() & df['duration'].notna(), 'started_at'] = \
      df['ended_at'] - pd.to_timedelta(df['duration'], unit='s')

    df['is_time_recovered'] = (
      (df['started_at_original'].isna() & df['started_at'].notna()) |
      (df['ended_at_original'].isna() & df['ended_at'].notna())
      )

    df.drop(columns=['started_at_original', 'ended_at_original'], inplace=True)

    recovered = df[df['is_time_recovered']]
    return recovered.head(3)
```

Hình 3. 4: Mã nguồn đọc lại thời gian di chuyển

- Đoạn mã này dùng để lọc các chuyến đi bất thường có điểm bắt đầu và kết thúc là

trùng nhau

```
suspicious_trips = df[
    (df['start_station_id'] == df['end_station_id'])
]

print(f"Number of suspicious trips: {len(suspicious_trips)}")
suspicious_trips.head()
```

Hình 3. 5: Mã nguồn lọc các chuyến đi bất thường

- Đoạn mã trên lọc các giá trị ngoại lai của cột duration

```
df = df[
    (df['duration'] > 200) &
    (df['duration'] < 7500)
]
```

Hình 3. 6: Mã nguồn lọc các giá trị ngoại lai

b) Xử lý dữ liệu trùng lặp

```
df = df.drop_duplicates()
df.duplicated().sum()
```

Hình 3. 7: Mã nguồn xử lý dữ liệu trùng lặp

- Xóa các giá trị trùng lặp

c) Logarit hóa và taget_encode dữ liệu

```
df['log_duration'] = np.log1p(df['duration'])
```

```
# Target Encoding
start_mean_map = y_train.groupby(X_train['start_station_id']).mean()
end_mean_map = y_train.groupby(X_train['end_station_id']).mean()

# Áp dụng Encoding và Xử lý cột
for data in [X_train, X_test]:
    # Tạo cột Encoding
    data['start_target_encoded'] = data['start_station_id'].map(start_mean_map)
    data['end_target_encoded'] = data['end_station_id'].map(end_mean_map)
    data.drop(['start_station_id', 'end_station_id'], axis=1, inplace=True)
```

Hình 3. 8: Mã nguồn Logarit hóa và taget_encode dữ liệu

- Logarit hóa và taget_encode dữ liệu

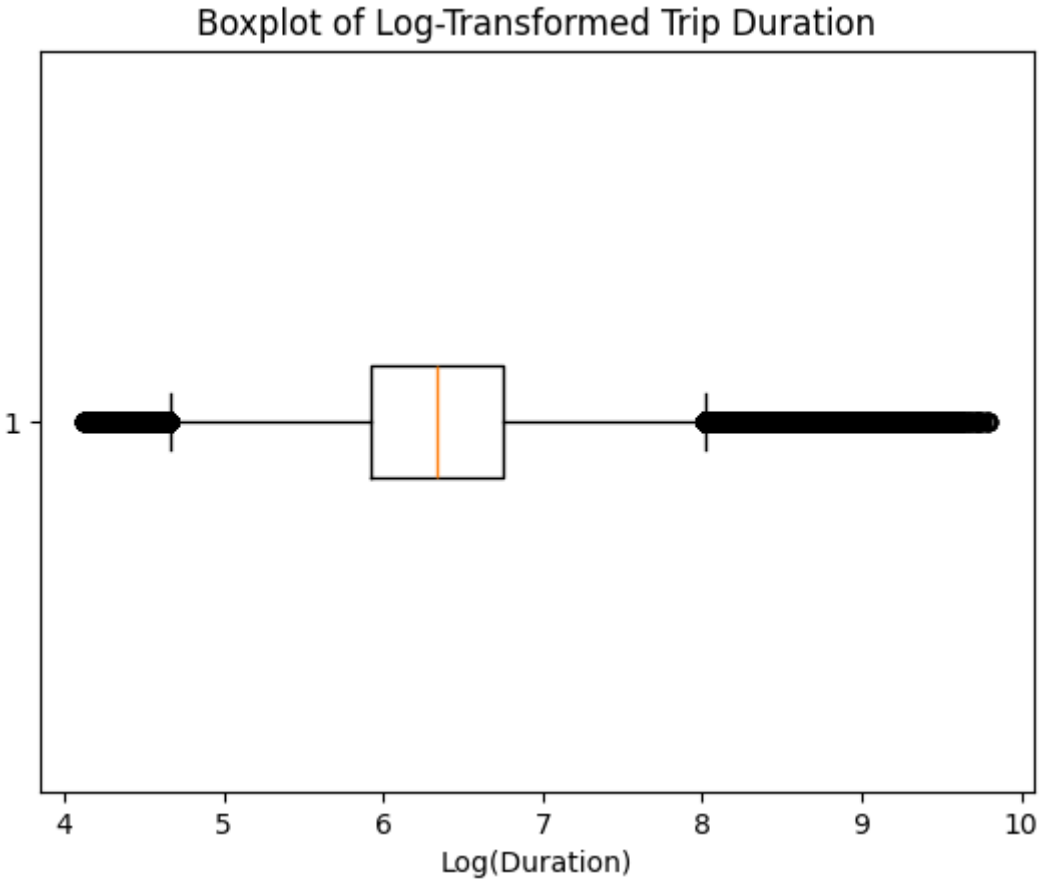
### 3.2 Mã nguồn chức năng trực quan hóa dữ liệu

a) Biểu đồ boxplot cho phân phối thời gian di chuyển

```
df['log_duration'] = np.log1p(df['duration'])
plt.figure()
plt.boxplot(df['log_duration'], vert=False)
plt.xlabel("Log(Duration)")
plt.title("Boxplot of Log-Transformed Trip Duration")
plt.show()
```

Hình 3. 9: Mã nguồn biểu đồ boxplot phân phối thời gian di chuyển

- Vẽ boxplot: Sử dụng thư viện Seaborn (sns) để vẽ boxplot, giúp hiển thị phân phối thời gian di chuyển
- Tùy chỉnh biểu đồ:
  - plt.title() để đặt tiêu đề cho biểu đồ
  - plt.xlabel() và plt.ylabel() để thêm nhãn cho các trục x và y.




Hình 3. 10: Biểu đồ boxplot cho phân phối thời gian di chuyển

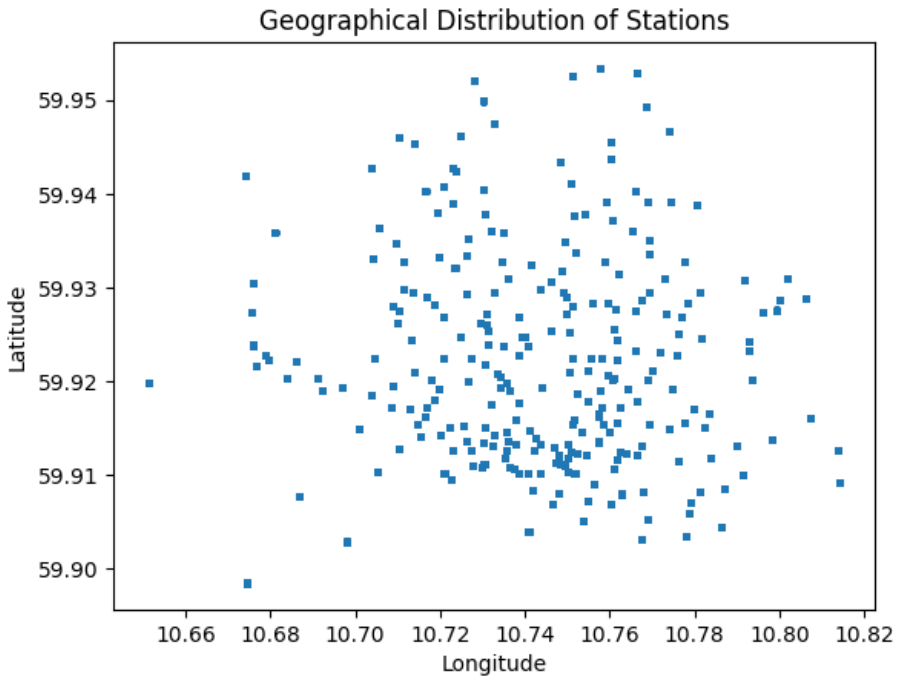
b) Biểu đồ phân bố Trạm

```
plt.figure()
plt.scatter(
    df['start_station_longitude'],
    df['start_station_latitude'],
    s=5,
    alpha=0.5
)
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.title("Geographical Distribution of Stations")
plt.show()
```

Hình 3. 11: Mã nguồn vẽ biểu đồ phân bổ trạm

- Vẽ biểu đồ phân phối trạm:

  - Sử dụng thư viện matplotlib.pyplot để vẽ biểu đồ phân tán (scatter plot) dựa trên tọa độ kinh độ và vĩ độ của các trạm từ DataFrame df.
  - plt.scatter(df['start_station_longitude'], df['start_station_latitude'], ...) sẽ tạo các điểm dữ liệu trên bản đồ:
    - s=5 thiết lập kích thước điểm ảnh nhỏ để biểu đồ thoáng mắt hơn.
    - alpha=0.5 thiết lập độ trong suốt cho các điểm, giúp hiển thị rõ mật độ tại các khu vực có nhiều trạm chồng lấn lên nhau (nơi màu sắc đậm hơn).




Hình 3. 12:Biểu đồ phân bố giá nhà

c) Biểu đồ số lượng chuyến đi theo Trạm

```
TOP_N = 20
start_counts = df['start_station_id'].value_counts().sort_values(ascending=True).head(TOP_N)

plt.figure()
start_counts.plot(kind='bar')
plt.xlabel("Start station ID")
plt.ylabel("Number of trips")
plt.title(f"Number of trips per START station (Top {TOP_N})")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

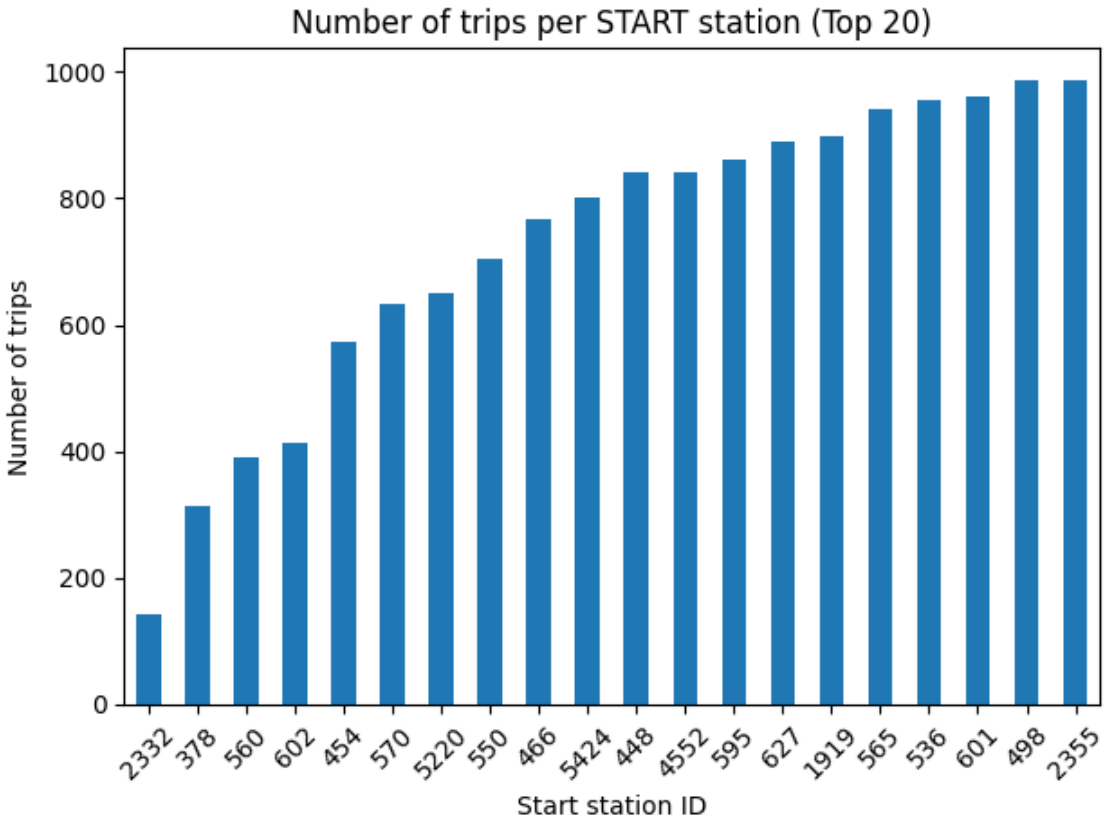
Hình 3. 13: Mã nguồn vẽ biểu đồ số lượng chuyến đi theo trạm bắt đầu

- Vẽ biểu đồ số lượng chuyến đi theo trạm bắt đầu (Top 20):

- Khai báo biến TOP_N = 20 để giới hạn số lượng trạm sẽ hiển thị trên biểu đồ.
- Chuẩn bị dữ liệu để vẽ:
    - df['start_station_id'].value_counts() đếm tần suất xuất hiện (số chuyến đi) của từng trạm khởi hành.
    - .sort_values(ascending=True).head(TOP_N) sắp xếp số lượng chuyến đi theo thứ tự tăng dần và lấy 20 trạm đầu tiên (Lưu ý: Code đang lấy 20 trạm có số lượng chuyến ít nhất do ascending=True).
- start_counts.plot(kind='bar') vẽ biểu đồ cột để so sánh số lượng giữa các trạm.
- Tinh chỉnh giao diện biểu đồ:
    - plt.xticks(rotation=45) xoay nhãn trục hoành một góc 45 độ để các mã

trạm (Station ID) không bị chồng chéo, dễ đọc hơn.

- plt.tight_layout() tự động căn chỉnh lề và khoảng cách để đảm bảo các nhãn trục và tiêu đề không bị cắt mất khi hiển thị.



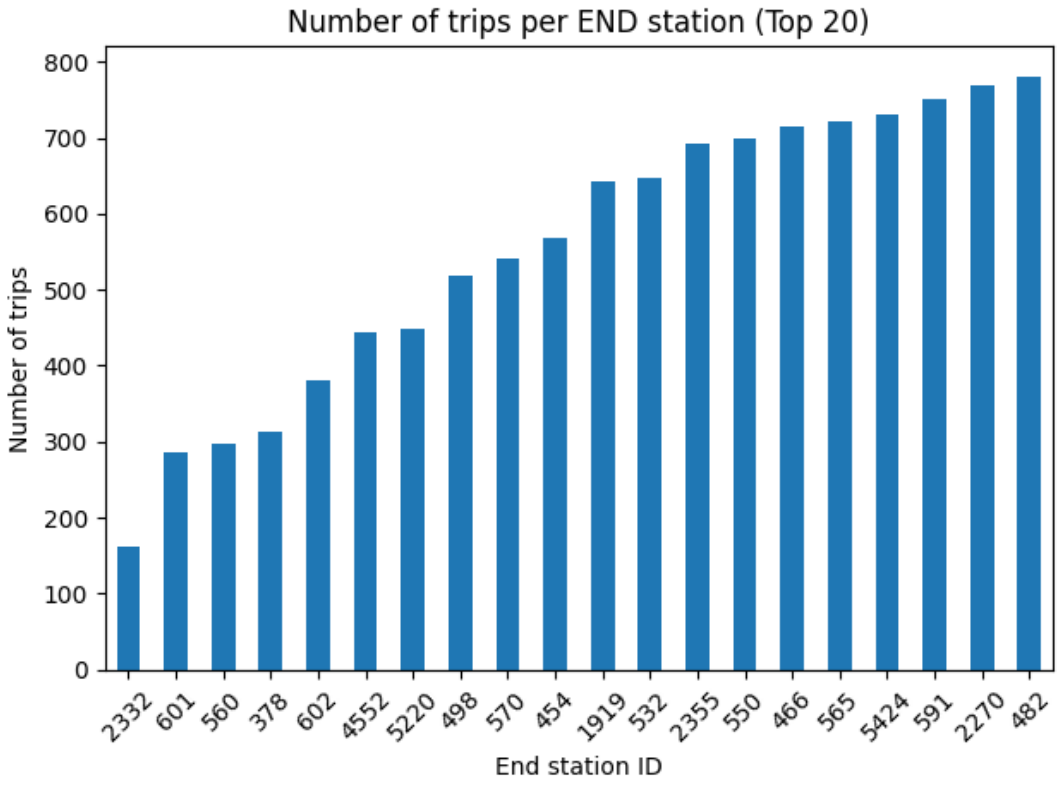

Hình 3. 14: Biểu đồ số lượng chuyến đi theo trạm bắt đầu

```
TOP_N = 20
end_counts = df['end_station_id'].value_counts().sort_values(ascending=True).head(TOP_N)

plt.figure()
end_counts.plot(kind='bar')
plt.xlabel("End station ID")
plt.ylabel("Number of trips")
plt.title(f"Number of trips per END station (Top {TOP_N})")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Hình 3. 15: Mã nguồn vẽ biểu đồ số lượng chuyến đi theo trạm kết thúc

- Vẽ biểu đồ số lượng chuyến đi theo trạm kết thúc (End Station):

- Tương tự như trên, nhưng áp dụng cho cột end_station_id.
- df['end_station_id'].value_counts() đếm số lần mỗi trạm xuất hiện làm điểm đến.
- end_counts.plot(kind='bar') vẽ biểu đồ cột thể hiện tần suất các trạm đích.
- Các lệnh plt.xlabel, plt.ylabel, plt.title được cập nhật tương ứng để chú thích cho "End station" (Trạm kết thúc).




Hình 3. 16: Biểu đồ số lượng chuyến đi theo trạm kết thúc

### 3.3 Áp dụng model

```
X = df.drop(['duration', 'log_duration'], axis=1)
y = df['log_duration']
```

Hình 3. 17: Chọn các cột đặc trưng và mục tiêu

- Chọn các cột đặc trưng (features): X. Kết quả là một DataFrame X chứa tất cả các đặc trưng (features) mà bạn sẽ sử dụng để xây dựng mô hình.
- Chọn cột mục tiêu (target): y = df['log_duration']. Biến này sẽ là đầu ra mà bạn muốn dự đoán.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Hình 3. 18: Chia tập train/test

- Chia dữ liệu thành tập huấn luyện (train) và tập kiểm tra (test):
  - X là các đặc trưng (features) và y là biến mục tiêu (target).
  - test_size=0.2 chỉ định rằng 20% dữ liệu sẽ được sử dụng cho tập kiểm tra, trong khi 80% còn lại sẽ được dùng cho tập huấn luyện.
  - random_state=42 đảm bảo rằng quá trình chia dữ liệu sẽ được lặp lại một cách nhất quán mỗi lần bạn chạy mã, giúp bạn có thể tái tạo kết quả.

```
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(max_depth=10, random_state=42),
    "Random Forest": RandomForestRegressor(n_estimators=100, max_depth=15, n_jobs=-1, random_state=42)
}
```

Hình 3. 19: Khởi tạo các model

- Khởi tạo mô hình Linear Regressor, Decision Tree, Random Forest Regressor từ thư viện sklearn. ensemble để khởi tạo một mô hình hồi quy.
- Huấn luyện mô hình: model.fit(X_train, y_train) sử dụng phương thức fit để huấn luyện mô hình với train data X_train và biến mục tiêu y_train.

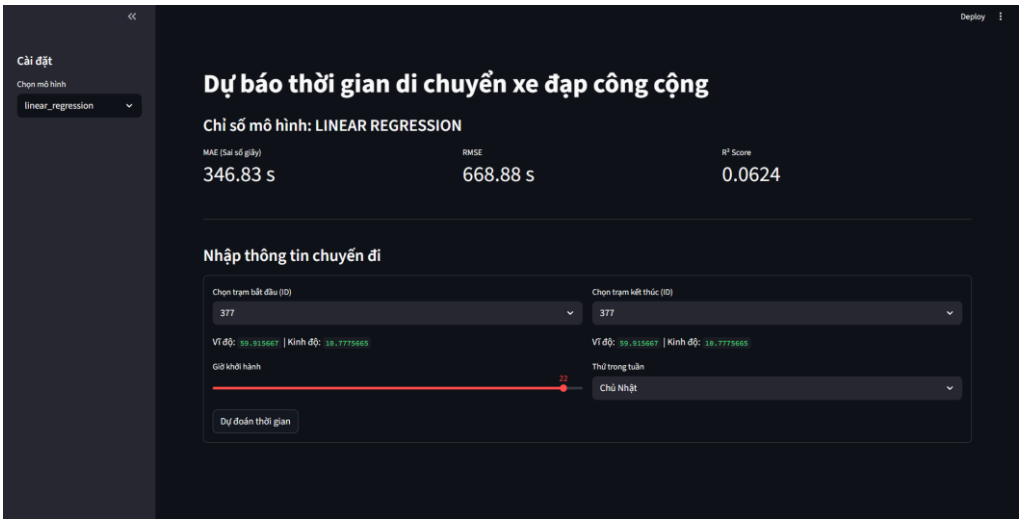
```
for name, model in models.items():
    # Huấn luyện
    model.fit(X_train, y_train)

    # Đánh giá
    y_pred_log = model.predict(X_test)
    y_actual = np.expm1(y_test)
    y_pred = np.expm1(y_pred_log)

    metrics = {
        "MAE": mean_absolute_error(y_actual, y_pred),
        "RMSE": np.sqrt(mean_squared_error(y_actual, y_pred)),
        "R2": r2_score(y_actual, y_pred)
    }
```

```
Đã lưu Linear Regression thành công! (MAE: 344.96s), (RMSE: 659.38s), (R2: 0.0583)
Đã lưu Decision Tree thành công! (MAE: 284.26s), (RMSE: 606.68s), (R2: 0.2029)
Đã lưu Random Forest thành công! (MAE: 241.51s), (RMSE: 571.87s), (R2: 0.2917)
```

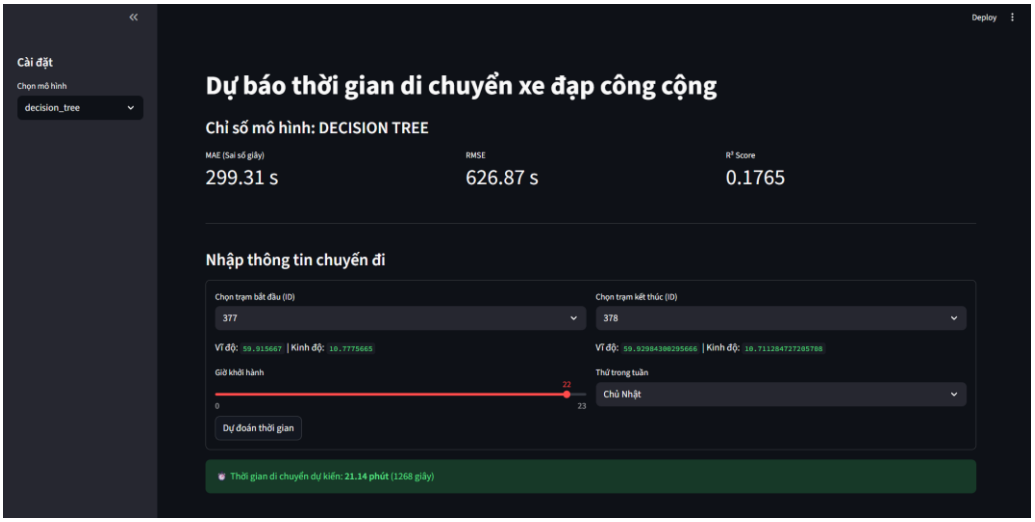
Hình 3. 20: Dự đoán và tính chỉ số đánh giá

- Tính toán các chỉ số đánh giá hiệu suất của mô hình:
    - mae_rf = mean_absolute_error(y_test, y_pred_rf): Tính giá trị lỗi tuyệt đối trung bình (Mean Absolute Error - MAE) giữa giá trị thực tế y_test và giá trị dự đoán y_pred_rf.
    - mse_rf = mean_squared_error(y_test, y_pred_rf): Tính giá trị lỗi bình phương trung bình (Mean Squared Error - MSE).
    - rmse_rf = np.sqrt(mse_rf): Tính căn bậc hai của MSE để có giá trị lỗi bình phương trung bình (Root Mean Squared Error - RMSE).
    - r2_rf = r2_score(y_test, y_pred_rf): Tính hệ số xác định ($R^2$ Score) để đo lường tỷ lệ biến thiên của giá trị mục tiêu mà mô hình giải thích

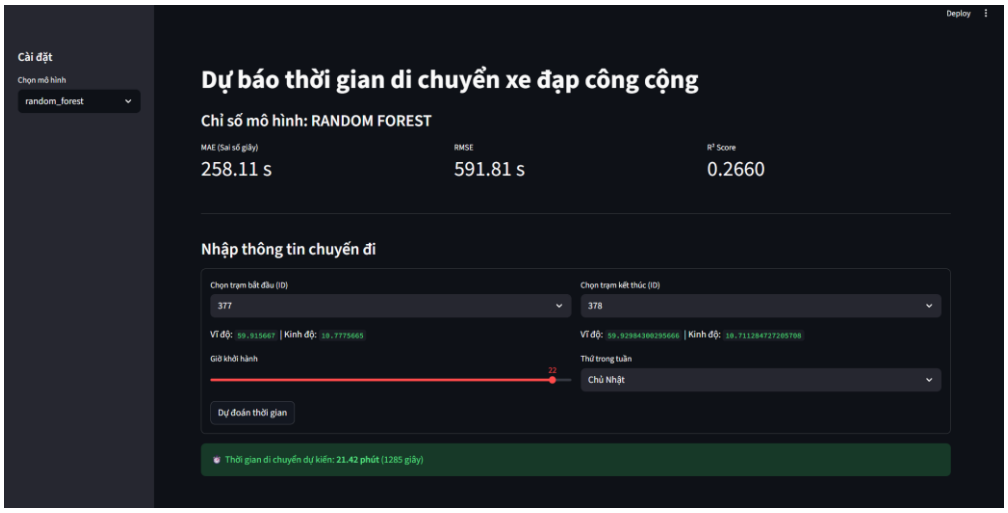
## 3.4 Sản Phẩm




Hình 3. 21: Giao diện cho mô hình Linear Regession




Hình 3. 22: Giao diện cho mô hình Decision Tree




Hình 3. 23: Giao diện cho mô hình Random Forest

# TÀI LIỆU THAM KHẢO

[1]. Matplotlib : Giới thiệu về Matplotlib - Nguyen Van Hoang (viblo.asia)

[2]. Pandas : Giới thiệu về Pandas - Nguyen Van Hoang (viblo.asia)

[3]. Numpy : Giới thiệu về Numpy - Nguyen Van Hoang (viblo.asia)

[4]. Dataset : Hanoi Housing price 2024 - huytrngquc (kaggle.com)

[5] N. V. Hậu, P. M. Chuẩn, và N. V. Quyết , "Giáo trình Học máy cơ bản", Nhà xuất bản Khoa học và Kỹ thuật, 2022.