ĐẠI HỌC BÁCH KHOA HÀ NỘI

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

A red and white flag with a yellow and red design

Description automatically generated

**BÁO CÁO BÀI TẬP LỚN AI**

**Chủ đề: Dự đoán giá nhà đất Hà Nội**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn:  Nhóm 12  Sinh viên tham gia: | **Trần Thế Hùng**  **Phạm Công Hào 20215045**  **Đặng Thái Tuấn 20210907**  **Phạm Đức Lưu 20215084** |
|  |  |

***Hà Nội, 6/2024***

# Giới thiệu bài toán

* Bài toán dự đoán giá nhà đất là một trong những bài toán quan trọng trong lĩnh vực bất động sản. Mục đích của bài toán là xây dựng một mô hình dự đoán giá nhà đất dựa trên các yếu tố đầu vào như vị trí, diện tích, số phòng, số tầng và nhiều yếu tố khác. Với sự phát triển không ngừng và nhu cầu nhà ở tăng cao, việc dự đoán chính xác giá nhà đất sẽ giúp các nhà đầu tư, người mua nhà và các bên liên quan đưa ra các quyết định chính xác và hiệu quả hơn.

Phân công công việc

|  |  |
| --- | --- |
| **Công việc** | **Tham gia** |
| Crawl data | Đặng Thái Tuấn,  Phạm Đức Lưu |
| Xử lý dữ liệu | Phạm Công Hào |
| Xây dựng mô hình | Phạm Công Hào,  Đặng Thái Tuấn,  Phạm Đức Lưu |
| Xây dựng giao diện | Phạm Đức Lưu,  Đặng Thái Tuấn |

# Cơ sở lý thuyết

1. Giới thiệu phương pháp Ensemble
   1. Phương pháp Ensemble là một kỹ thuật trong học máy nhằm cải thiện độ chính xác và hiệu suất của mô hình bằng cách kết hợp nhiều mô hình dự đoán khác nhau thay vì chỉ dựa vào một mô hình duy nhất, phương pháp Ensemble tận dụng sức mạnh của nhiều mô hình để tạo ra một mô hình tổng hợp mạnh mẽ hơn.

* Ưu điểm:
  + Cải thiện độ chính xác: kết hợp nhiều mô hình thường cho kết quả chính xác hơn so với 1 mô hình đơn lẻ.
  + Giảm overfitting: Ensemble giúp làm giảm nguy cơ overfitting, đặc biệt là với các mô hình phức tạp.
* Nhược điểm:
  + Tăng độ phức tạp: Ensemble nhiều mô hình làm tăng độ phức tạp của hệ thống. Việc triển khai và bảo trì trở nên khó khăn hơn.
  + Tài nguyên tính toán: Huấn luyện và dự đoán với nhiều mô hình đòi hỏi nhiều tài nguyên tính toán (CPU, GPU, bộ nhớ) và thời gian hơn.
  + Giải thích kết quả: Với những kỹ thuật phức tạp như stacking, thường khó giải thích hơn các mô hình đơn lẻ. Điều này có thể là một vấn đề trong các lĩnh vực đòi hỏi tính minh bạch và giải thích rõ ràng.
  1. Ensemble được chia thành các loại chính:
* Bagging: Sử dụng nhiều mô hình học máy giống nhau nhưng được huấn luyện trên các tập dữ liệu con khác nhau, được tạo ra bằng cách lấy mẫu ngẫu nhiên từ bộ dữ liệu train ban đầu. Những model này sẽ được train độc lập và song song với nhau nhưng đầu ra của chúng sẽ được trung bình cộng để cho ra kết quả cuối cùng.

Vd: RandomForest

* Boosting: Xây dựng một chuỗi các mô hình, trong đó mỗi mô hình cố gắng sửa lỗi của mô hình trước đó. Các mô hình sau được huấn luyện để tập trung vào những điểm dữ liệu mà các mô hình trước đó dự đoán sai.

Vd: AdaBoost, XGBoost, Gradient Boosting Machines (GBM).

* Stacking: Xây dựng một số model (thường là khác loại) và một meta model (supervisor model), train những model này độc lập, sau đó meta model sẽ học cách kết hợp kết quả dự báo của một số mô hình một cách tốt nhất. Stacking sẽ tập trung để giảm bias ( độ lệch, biểu thị sự chênh lệch giữa giá trị trung bình mà mô hình dự đoán và giá trị thực tế của dữ liệu ).

1. Giới thiệu RandomForessRegressor

* Cơ sở lý thuyết: RandomForest là một thuật toán học máy thuộc họ ensemble model được xây dựng theo phương pháp Bagging (các Decision trees trong model được huấn luyện độc lập trên các bộ dữ liệu con) , được sử dụng cho cả bài toán phân loại (classification) và hồi quy (regression). Nó kết hợp nhiều cây quyết định (decision trees) để tạo ra một mô hình dự đoán mạnh mẽ và chính xác hơn.
* RandomForest còn được hiểu là một rừng cây. Kết quả cuối cùng của model được tổng hợp lại từ các cây con này.
* RandomForest còn được hiểu là 1 rừng cây.

1. Giới thiệu XGBoost

* Cơ sở lý thuyết: XGBoost là một tập hợp của nhiều cây quyết định (decision trees). Mỗi cây trong XGBoost là một mô hình con và thường là các cây nhỏ có độ sâu nông. XGBoost sử dụng kỹ thuật boosting, trong đó các cây được xây dựng tuần tự. Mỗi cây mới cố gắng sửa các lỗi của các cây trước đó bằng cách giảm thiểu hàm mất mát (loss function) thông qua gradient descent.
* Ưu điểm: tốc độ tính toán nhanh, tự động cắt tỉa cây loại bỏ những leaves không mang giá trị tích cực trong quá trình mở rộng cây.

# Xây dựng mô hình

* Mô hình được xây dựng bằng cách sử dụng Stacking để kết hợp 2 mô hình XGBoost và RandomForestRegressor lấy điểm mạnh của mô hình này bù đắp vào sự thiếu sót của mô hình kia.
* Có một vài lý do chỉ lựa chọn hai model chứ không phải nhiều hơn:
  + **Đơn giản hơn**: Khi sử dụng ít mô hình, hệ thống sẽ đơn giản hơn, dễ triển khai và bảo trì hơn so với việc sử dụng nhiều mô hình.
  + **Tiết kiệm tài nguyên**: Sử dụng ít mô hình sẽ ít tốn kém tài nguyên tính toán hơn so với việc sử dụng nhiều mô hình.
  + **Thời gian huấn luyện và dự đoán**: Việc huấn luyện và sử dụng nhiều hay ít mô hình sẽ ảnh hưởng nhiều đến thời gian huấn luyện và dự đoán. Trong quá trình chạy thử nghiệm nhận thấy sử dụng nhiều hơn 2 model sẽ đạt được sự cân bằng giữa sai số và thời gian dự đoán. Cụ thể xem ở mục IV.
* **Lý do lựa chọn XGBoost chứ không phải các kiểu mô hình khác trong các mô hình đươc xây dựng theo Boosting**
  + **Khả năng giải thích tốt hơn**: do cả hai model Ran và XGB đều xây dựng các mô hình con dựa trên cây quyết định (Desision Tree) nên có thể dễ dàng hiểu và phân tích cách mỗi mô hình đóng góp vào dự đoán cuối cùng.
  + Tự động bỏ qua những leaves, node không mang giá trị tích cực trong quá trình mở rộng Tree dẫn đến việc giảm đáng kể lượng thời gian train mô hình cũng như thời gian dự đoán.
  1. Crawl data
* Dữ liệu gồm: Phố, Phường, Quận, Thành phố/ Tỉnh, Diện tích, Số phòng ngủ, Số tầng, WC( số nhà vệ sinh ), Mặt tiền (m2), Vị trí, Pháp lý, Kiểu nhà, Thang máy, Giá nhà.
  + Dữ liệu về nhà: 55271 bản ghi (1 bản ghi chưa các thông số của 1 căn nhà: diện tích, số tầng, vị trí, giá nhà,....)
  + Dữ liệu về chung cư: 1500 bản ghi
* Cách lấy dữ liệu: sử dụng phương pháp Selenium cho tester test các chức năng của 1 website nhugw cũng có thể dùng để crawl data thông qua các thẻ trong html.
  1. Xử lý dữ liệu
* Missing values

Sau khi kiểm tra lượng dữ liệu bị thiếu trong mỗi bản ghi là 7% > 5% => thực hiện sinh dữ liệu cho các Feature bị thiếu. Cụ thể:

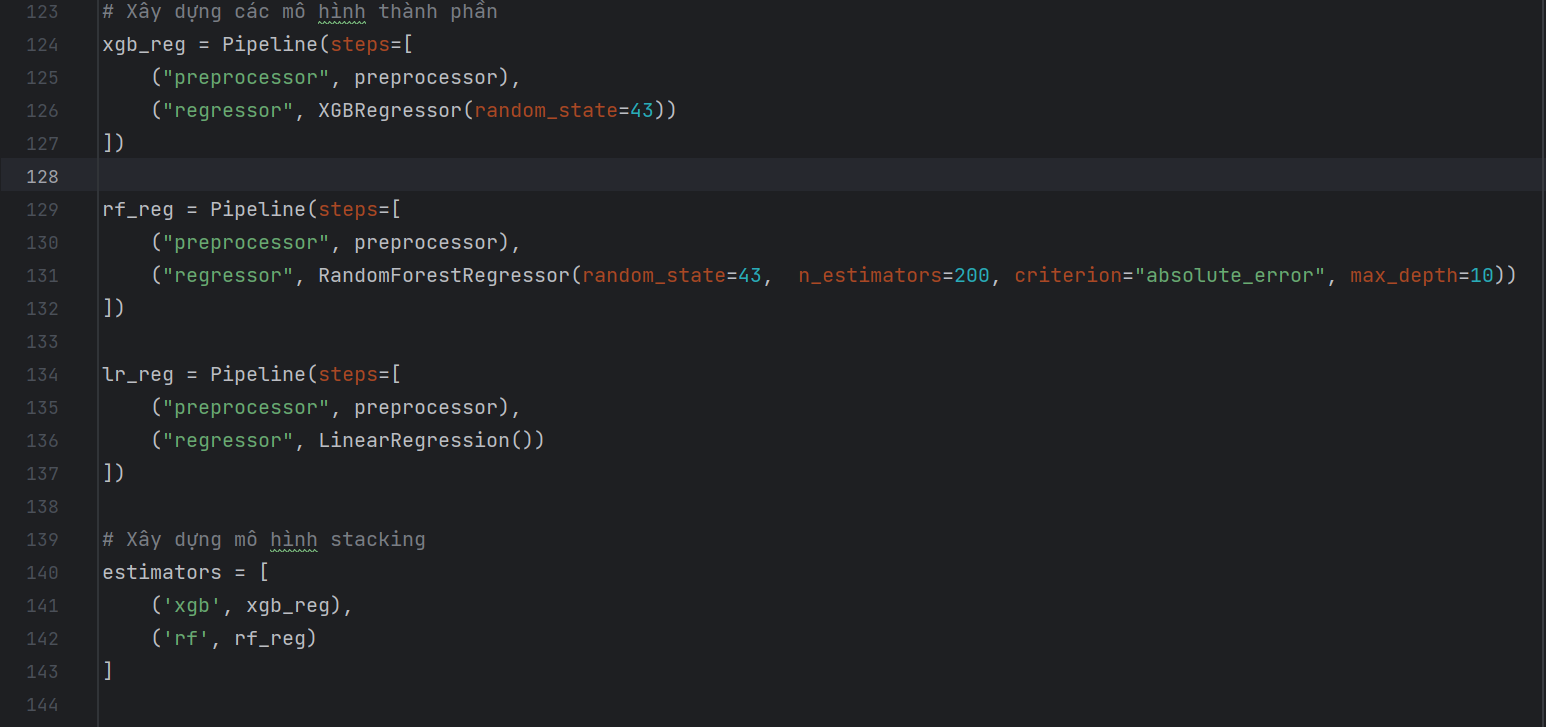
* + Diện tích: gán giá trị bằng giá trị trung bình diện tích của các căn nhà trong bộ dữ liệu.
  + Số tầng: gán giá trị bằng giá trị xuất hiện nhiều lần nhất trong Feature “Số tầng” của bộ dữ liệu.
  + Số phòng ngủ, WC, Thang máy: Xử lý tương tự như “Số tầng”.
  + Mặt tiền: việc gán giá trị bị thiếu cho Feature này tùy thuộc vào vị trí của căn nhà trong bộ dữ liệu:
    - 1. “Nhà hẻm, ngõ” => bằng 0.
      2. “Nhà mặt tiền, phố” => bằng 5.
      3. “Biệt Thự liền kề” => bằng giá trị trung bình cộng Mặt tiền của các căn nhà có vị trí “Biệt thự liền kề”.
  + Vị trí, Quận, Pháp lý: gán các giá trị bằng giá trị xuất hiện nhiều nhất trong mỗi Feature tương ứng.
* Loại bỏ các Outliers (xử lý các bản ghi có dữ liệu khác nhiều so với đa số): Sử dụng phương pháp Interquartile Range (IQR) tính giới hạn trên và giới hạn dưới sau đó loại bỏ các bản ghi có giá trị nằm ngoài khoảng từ giới hạn dưới đến giới hạn trên.
* Mã hóa dữ liệu dạng chữ: Do các Feature bên dưới đều là kiểu dữ liệu dạng Ordinal feature (có thứ tự) nên cần mã hóa phân cấp cho từng Feature. Cụ thể:
  + Quận: gán bằng giá trị cho 1m2 đất của mỗi Quận. Cụ thể:
    - Quận Đống Đa: 192
    - Quận Hai Bà Trưng: 214
    - Quận Hà Đông: 144
    - ....
  + Vị trí: có các loại giá trị (“Nhà hẻm, ngõ”, “nan”, “Nhà đường nội bộ, Cổ Nhuế”, “Biệt thự liền kề”, “Nhà mặt tiền, phố”) sẽ được mã hóa lần lượt (0, 1, 1, 2, 3).

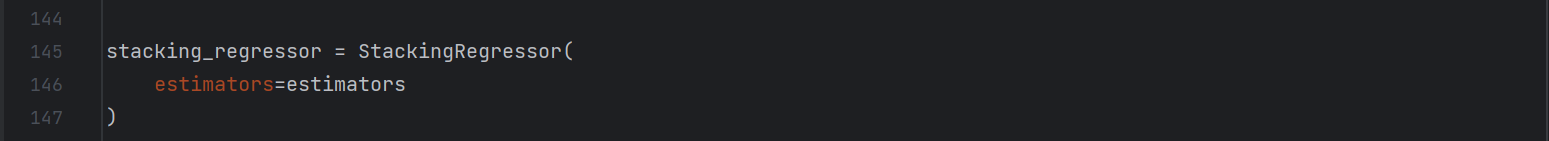
Giải thích giá trị “nan” do dữ liệu của Chung cư không có thuộc tính “Vị trí” => không xác định => gán giá trị “nan”.

* + Pháp lý: có các giá trị (“Sổ hồng, Sổ đỏ”) do chỉ có hai thuộc tính để tránh việc mô hình hiểu nhầm đây là dữ liệu kiểu Boolean sẽ mã hóa (0, 2) thay vì (0, 1).
  + Kiểu nhà: có hai giá trị (“Nhà”, Chung cư”) tương tự sẽ được mã hóa (0, 2).
* Bảng dữ liệu sau khi được loại bỏ các outliers, sinh dữ liệu cho các bản ghi bị thiếu (Missing values), mã hóa các Feature dạng chữ:



* Do độ dài miền dữ liệu của từng Feature chênh lệch nhau rất nhiều VD: quận(104 đến 514), kiểu nhà (0 đến 2). Nếu như không được xử lý thì khi train mô hình sẽ tập trung nhiều vào Feature có độ dài miền giá trị lớn và bỏ qua các Feature có độn dài miền dữ liệu bé. Để khắc phục điều này ta sẽ đưa tất cả các Feature về cùng 1 Range bằng cách sử dụng hàm Robustscaler().
* Xây dựng mô hình

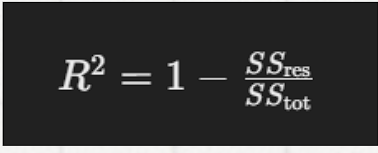
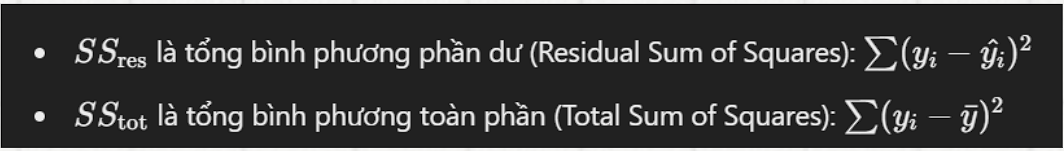


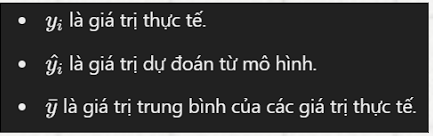


* Sử dụng stacking để gộp hai mô hình RandomForestRegressor (RFR) và XGBoost (XGB) lại thành 1 mô hình mới có độ chính xác, thời gian Train, thời gian dự đoán được cải thiện hơn rất nhiều.
* Bên trong mỗi model RFR, XGB đều được lựa chọn các tham số phù hợp nhất để tạo thành model cuối cùng.

# Đánh giá và giải thích các kết quả của các mô hình

1. Các Metrict được sử dụng để đánh giá mô hình
2. R2

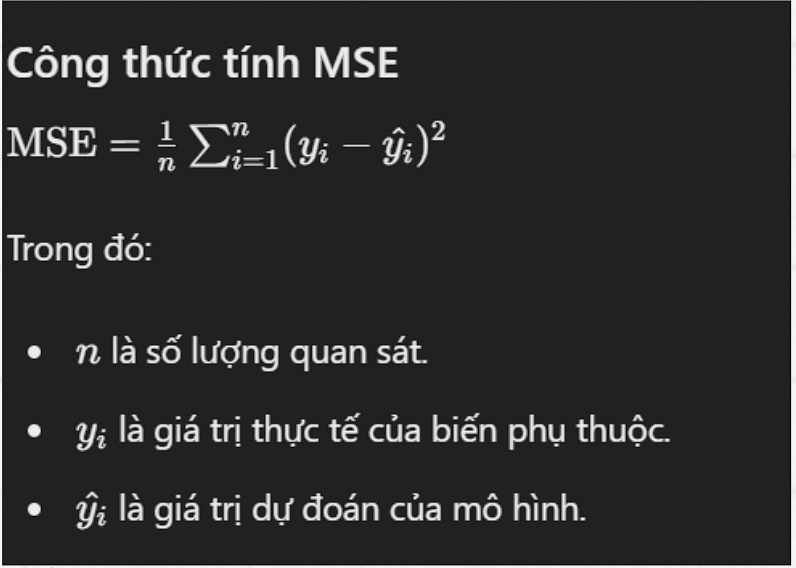
 



R2 nằm trong giá trị từ khoảng 0 -> 1.

R2 càng gần 1 thì mô hình sẽ càng tốt vì tất cả các dự đoán sẽ gần với thực tế.

1. MSE



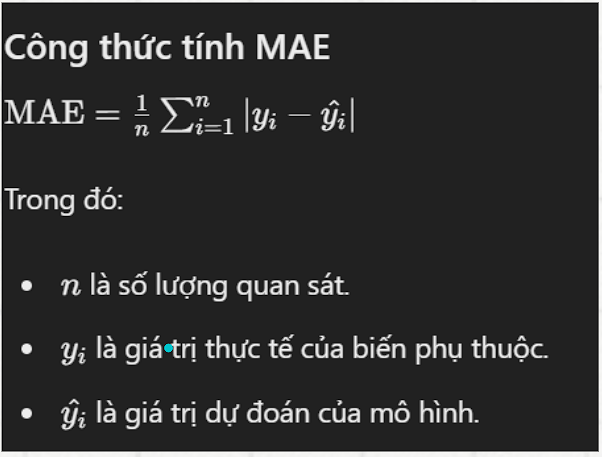
Được gọi là sai số bình phương ( phương sai )

Nếu MSE càng nhỏ thì mô hình dự đoán giá càng tốt

Vì MSE tính bình phương của sai số, các sai số lớn sẽ có tác động lớn hơn đến MSE so với các sai số nhỏ.

=> MSE rất nhạy cảm với các giá trị ngoại lệ (outliers).

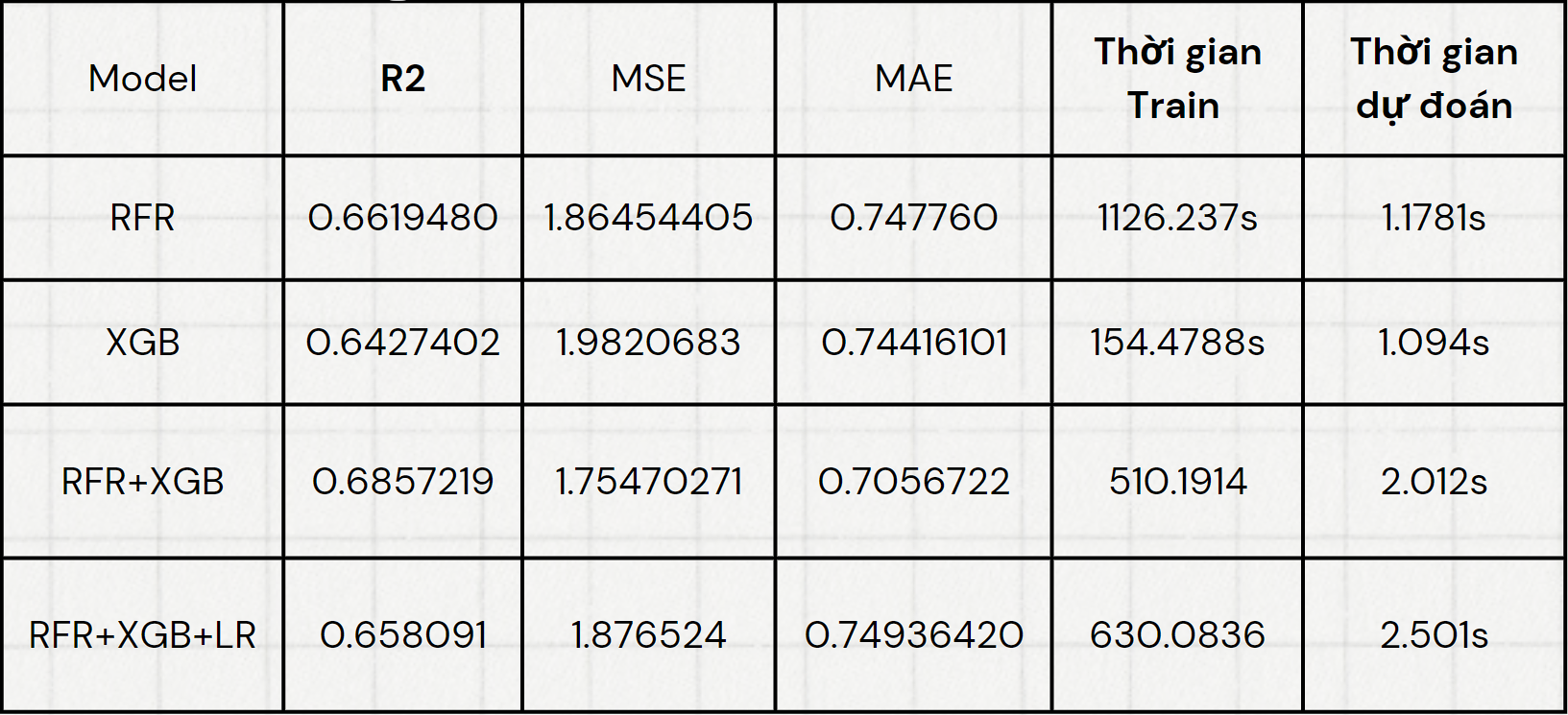
1. MAE



Độ lớn trung bình của các sai số dự đoán và thực tế

MAE càng nhỏ: Sai số giữa các giá trị thực tế và giá trị dự đoán càng nhỏ, nghĩa là mô hình càng chính xác.

1. So sánh giữa các mô hình



* Giải thích 1 vài kết quả có được từ bảng kết quả trên:
* Đầu tiên RFR không có khả năng tự động cắt tỉa cây nên có thể cây được tạo ra sẽ có độ chi tiết, bao phủ rộng hơn so với XGB dẫn đến thời gian chạy lâu hơn nhưng bù lại sẽ có kết quả dự đoán chính xác.
* Tuy nhiên khi RFR xây dựng cây quá chi tiết có thể dẫn đến Overfiting (kết quả ít sai lệch trong bộ train nhưng sẽ bị lệch khi test) dẫn đến kết quả cuối cùng chưa tối ưu.
* Khi sử dụng Stacking để hợp hai mô hình này thành một mô hình mới trong quá trình xây dựng cây các cây sẽ được tự động cắt tỉa tránh bị Overfitting tuy nhiên vẫn đảm bảo tính chi tiết, độ bao phủ rộng. => thời gian chạy của model mới sẽ nhanh hơn RFR và chậm hơn XGB.
* Với mô hình được tạo bởi RFR, XGB, LG: bản thân LG là một mô hình tuyến tính nó sẽ không năm bắt được các mối quan hệ phi tuyến tính của bộ dữ liệu Nhà do vậy nó sẽ làm cho dự đoán tổng thể của toàn bộ mô hình đi xuống. Và thời gian dự đoán sẽ bị tăng lên.
* LG là một mô hình đơn giản nó đã làm tăng thời gian dự đoán lên 0.5s, khi thêm các mô hình phức tạp tương tự như RFR thì thời gian dự đoán sẽ bị tăng lên đáng kể làm cho trải nghiệm của người dùng đối với phầm mềm đi xuống nên để cân bằng giữa tính chính xác và thời gian dự đoán phần mềm đã sử dụng 2 mô hình là RFR và XGB.

# Một số vấn đề gặp phải khi triển khai trong thực tế

* Dữ liệu không được update liên tục từng ngày
* Các yếu tố phức tập ảnh hưởng đến giá căn nhà nhưng khó mô tả bằng dữ liệu. Vd: Các yếu tố kinh tế vĩ mô như suy thoái kinh tế, lạm phát, tỷ lệ thất nghiệp có thể ảnh hưởng lớn đến thị trường bất động sản. Thay đổi trong chính sách nhà ở, thuế, và quy hoạch đô thị có thể ảnh hưởng đến giá nhà.