Dự đoán giá nhà

Võ Nhật Thanh - 19522245, Lê Vinh Quang - 19522093,
Trần Trung Tín - 19522351, Hồ Thịnh - 19522274

***Tóm tắt nội dung*—Có một thời điểm trong cuộc sống của mọi người khi người đó muốn mua hoặc bán một ngôi nhà. Đầu tiên hãy xem xét tình huống một người cần mua một ngôi nhà. Người đó sẽ tìm kiếm ngôi nhà mong muốn của mình với mức giá hợp lý. Người đó sẽ có một số yêu cầu về kiến trúc hay vị trí hay một số đặc điểm khác của ngôi nhà để đánh giá xem mức giá hợp lý để mua. Tương tự, hãy xem xét tình huống một người cần bán một căn nhà. Bằng cách sử dụng hệ thống dự đoán giá nhà, người bán sẽ có thể quyết định tất cả các tính năng mà họ có thể thêm vào ngôi nhà để ngôi nhà có thể được bán với giá cao hơn. Do đó, từ cả hai kịch bản trên, chúng tôi có thể khẳng định rằng dự đoán giá nhà là hữu ích cho cả người mua và người bán. Bài viết này sẽ giúp dự đoán giá nhà dựa trên các thông số khác nhau thông qua một số mô hình hồi quy chúng tôi sử dụng dưới đây.**

I. Giới thiệu bài toán

Với nhu cầu đời sống ngày càng được cải thiện, cùng với nhu cầu mở rộng vị trí để kinh doanh cũng như sinh sống. Gắn liền với những nhu cầu đó thì nhu cầu về mua nhà ở cũng được sinh ra. Khi đi mua nhà, người mua sẽ quan tâm đến kích thước, vị trí, ngôi nhà còn mới hay đã cũ, nội thất bên trong của ngôi nhà ra sao và giá của ngôi nhà đó như thế nào?.... Nhưng quan trọng nhất và được ưu tiên hàng đầu đối với người mua nhà vẫn là giá của ngôi nhà đó có phù hợp với túi tiền của mình hay không.

Như đã đề cập ở trên, nhóm chúng tôi cài đặt một số mô hình hồi quy để giải quyết bài toán dự đoán giá nhà. Dự đoán giá nhà giúp cho khách hàng biết được ngôi nhà đó sẽ có giá rẻ hơn, hay đắt hơn so với giá trị thực của ngôi nhà.

Để thực hiện được mục tiêu chúng tôi chọn bộ dữ liệu Ames Housing. Bộ dữ liệu này gồm danh sách các ngôi nhà được đánh thứ tự bằng Id, bên cạnh đó mỗi ngôi nhà còn có những thông tin nhằm phục vụ cho việc dự đoán giá ngôi nhà ví dụ như: đặc điểm ngôi nhà, vị trí, thông tin lô đất, đánh giá về tình trạng, chất lượng.

II. Phương pháp

A. *Random Forest Regressor*

SVTH: Võ Nhật Thanh

Giới thiệu: Random Forest Regression là một thuật toán học có giám sát (supervised learning) sử dụng phương pháp học đồng bộ để hồi quy. Phương pháp học đồng bộ là một kỹ thuật kết hợp các dự đoán từ nhiều thuật toán học máy nhằm đưa ra dự đoán chính xác hơn so với một mô hình độc lập duy nhất. Cụ thể là Random Forest Regression sử dụng kết hợp nhiều loại Random Decision Tree mà trong đó mỗi cây được dùng để đào tạo một tập con của tập dữ liệu huấn luyện. Tại mỗi nút, một mẫu các đặc trưng khác nhau được chọn để phân tách và các cây sau đó được huấn luyện độc lập song song. Các dự đoán từ các cây sau đó sẽ được tính trung bình và tổng hợp lại thành kết quả dự đoán của Random Forest Regression. Việc sử dụng nhiều cây để huấn luyện bộ dữ liệu giúp cho thuật toán thêm phần ổn định cũng như giảm phương sai. Thuật toán này là một mô hình được sử dụng rộng rãi do khả năng hoạt động tốt trên hầu hết các loại dữ liệu lớn bao gồm các tính năng có mối quan hệ phi tuyến tính. Tuy nhiên, mô hình cũng tồn tại một số nhược điểm như không có khả năng diễn giải,dễ xảy ra overfit và quan trọng hơn là Random Forest Regression không thể ngoại suy các giá trị nằm ngoài bộ dữ liệu huấn luyện tức là khi ta đưa một dữ liệu mới ngoài dữ liệu đã huấn luyện cho mô hình thì mô hình chỉ có thể dự đoán giá trị trung bình từ các giá trị đã thấy trước đó và tất nhiên giá trị này không thể nằm ngoài khoảng nhỏ nhất-lớn nhất trong mẫu. Chính vì những nhược điểm như thế, chúng ta chỉ nên sử dụng mô hình này trên các bộ dữ liệu phi tuyến tính hoặc trong trường hợp việc ngoại suy những dữ liệu ngoài bộ dữ liệu đã huấn luyện là không quan trọng.
Thuật toán Random Forest cho bài toán hồi quy tuân theo một quy trình như sau: 1. Vẽ các mẫu bootstrap ntree từ dữ liệu gốc. 2. Đối với mỗi mẫu bootstrap, hãy phát triển một cây hồi quy chưa cắt tỉa, với sự sửa đổi như sau: tại mỗi nút, thay vì chọn phần tách tốt nhất trong số tất cả các biến dự đoán, hãy lấy mẫu ngẫu nhiên mtry của các biến dự đoán và chọn phần tách tốt nhất trong số các biến đó. (Đóng bao có thể được coi là trường hợp đặc biệt của Random Forest thu được khi mtry = p, số lượng yếu tố dự đoán). 3. Dự đoán dữ liệu mới bằng cách tổng hợp các dự đoán của cây ntree (trung bình kết quả thu được).

Mô hình bao gồm một số tham số quan trọng như: 1. **n_estimators**: số lượng Decision Tree ta sẽ dùng trong mô hình 2. **criterion**: biến này cho phép chúng ta chọn tiêu chí đánh giá kết quả mô hình (hàm mất mát). Giá trị mặc định là MSE, trong báo cáo này chúng tôi dùng R2_score. 3. **max_depth**: độ sâu tối đa của cây. 4. **max_features**: số lượng đặc trưng tối đa mô hình sử dụng. 5. **bootstrap**: gía trị mặc định là True, nghĩa là mô hình tuân theo nguyên tắc boostrapping. 6. **max_samples**: kích thước lớn nhất cho mỗi mẫu của mỗi cây, xác định khi giá trị bootstrap là True. 7. **min_samples_split**: số lượng mẫu tối thiểu để tách một nút. 8. **min_samples_leaf**: số lượng mẫu tối thiểu phải có tại nút lá.Một điểm phân chia ở bất kỳ độ sâu nào sẽ chỉ được xem xét nếu nó để lại ít nhất 'min samples leaf' mẫu huấn luyện trong mỗi nhánh trái và phải. Điều này có thể có tác dụng làm trơn mô hình, đặc biệt là trong hồi quy. 9. **n_jobs**: số lượng cây chạy song song

Tunning: Như đã đề cập ở phần các tham số, trong báo cáo này chúng tôi sử dụng R2_score do đó mục tiêu chính của chúng tôi sẽ là tối đa hóa giá trị này. Chúng tôi sử dụng

kỹ thuật Grid Searh để thực hiện quá trình tunning một trong các tham số đã đề cập ở phần trước. Các tham số dùng để tune được cài đặt như sau: 1. **n_estimators**: thuộc khoảng 100 đến 500 bước nhảy 5 2. **max_features**: 'auto', 'sqrt' 3. **max_depth**: thuộc khoảng 5 đến 30 bước nhảy 6 4. **min_samples_split**: thuộc (2,5,10,15,100) 5. **min_samples_leaf**: thuộc (1,2,5,10)

B. *Gradient Boosting Regression*

SVTH: Lê Vinh Quang

Giới thiệu: Gradient Boosting là một thuật toán máy học hoạt động dựa trên kỹ thuật tổng hợp có tên là Boosting. Giống như Random Forest cũng như các mô hình Boost khác, Gradient Boost kết hợp nhiều đơn vị weak learner để tạo ra một strong learner. Thông thường, Gradient Boost sử dụng Decision Tree làm các weak learner. Ý tưởng của Boosting là đạo tạo các weak learner một cách tuần tự bằng cách chỉnh sửa các weak learning trước learner đang xét. Điều này có nghĩa là, thuật toán sẽ luôn học điều gì đó không hoàn toàn chính xác nhưng chỉ là một bước nhỏ đi đúng hướng. Khi thuật toán tiến về phía trước bằng cách sửa liên tục các lỗi trước đó, nó sẽ cải thiện khả năng dự đoán.

Để hiểu Gradient Boost dưới đây là các bước thực hiện của thuật toán: Bước 1. Xây dựng cây cơ sở với một nút gốc. Đó là phỏng đoán ban đầu cho tất cả các mẫu. Bước 2. Xây dựng cây từ lỗi của cây trước đó. Bước 3. Chia tỷ lệ cây theo tỷ lệ học tập (giá trị từ 0 đến 1). Tỷ lệ học tập này xác định sự đóng góp của cây trong dự đoán. Bước 4. Kết hợp cây mới với tất cả các cây trước đó để dự đoán kết quả và lặp lại Bước 2 cho đến khi đạt được số lượng cây tối đa hoặc cho đến khi các cây mới không cải thiện độ vừa vặn. Mô hình dự đoán cuối cùng là sự kết hợp của tất cả các cây. Gradient Boosting là một kỹ thuật tăng cường mạnh mẽ. Nó cải thiện độ chính xác của mô hình bằng cách kết hợp tuần tự các cây yếu để tạo thành một cây mạnh. Bằng cách này, nó đạt được độ lệch thấp và phương sai thấp.

Gradient Boosting Regression có một số ưu điểm nổi trội cố thể kể đến như nó có thể sử dụng nhiều loại độ đo để đánh giá độ hiệu quả của mô hình, điều này giúp cho nó dễ dàng so sánh với các mô hình khác khi giải quyết cùng loại bài toán; phương pháp này huấn luyện dữ liệu nhanh kể cả trên các tập dữ liệu lớn; ngoài ra nó còn cung cấp hỗ trợ xử lý các tính năng phân loại, một số trong số họ xử lý các giá trị bị thiếu nguyên bản. Tuy nhiên, mô hình cũng tồn tại một số nhược điểm dễ thấy như dễ bị overfit, điều này có thể được giải quyết bằng cách áp dụng các chuẩn L1 và L2 hoặc giảm learning_rate. khó giải thích các mô hình cuối cùng; một nhược điểm khác đó là chi phí tính toán cao cũng như là tính khó giải thích của mô hình.

Tuning: Trong mô hình Gradient Boosting Regression có một số tham số quan trọng cần lưu ý và đây cũng là những tham số chúng ta sẽ tune nhằm tối ưu mô hình: 1. **n_estimators**: số lượng cây dùng trong mô hình. Càng nhiều cây thì càng có nhiều khả năng bị quá tải. 2. **learning_rate**: trọng số mà mỗi cây có trên dự đoán cuối cùng. 3. **sub_sample**: tỷ lệ của mẫu để sử dụng. 4. **max_depth**: độ sâu tối đa của cây. Những gì chúng ta sẽ làm bây giờ là tạo một phiên bản của Gradient Boosting Regressor. Tiếp theo, chúng ta sẽ tạo grid với các giá trị khác nhau cho siêu tham số. Sau đó, chúng tôi sẽ lấy grid này và đặt nó bên trong chức năng GridSearchCV để chúng tôi có thể chuẩn bị chạy mô hình của mình.

C. *Linear Regression*

SVTH: Trần Trung Tín

Giới Thiệu: Linear Regression là một thuật toán Machine Learning học có giám sát (supervised learning). Mô hình hồi quy một giá trị dự đoán mục tiêu dựa trên các biến độc lập. Nó chủ yếu được sử dụng để tìm ra mối quan hệ giữa các biến và dự đoán. Các mô hình hồi quy khác nhau khác nhau dựa trên – loại mối quan hệ giữa các biến phụ thuộc và biến độc lập mà chúng đang xem xét và số lượng biến độc lập được sử dụng. Có nhiều tên gọi cho biến phụ thuộc của hồi quy. Nó có thể được gọi là biến kết quả, biến tiêu chí, biến nội sinh hoặc biến hồi quy. Các biến độc lập có thể được gọi là biến ngoại sinh, biến dự đoán hoặc biến hồi quy.Linear Regression thực hiện nhiệm vụ dự đoán giá trị của biến phụ thuộc (y) dựa trên biến độc lập (X) cho trước. Vì vậy, kỹ thuật hồi quy này tìm ra mối quan hệ tuyến tính giữa X (đầu vào) và y (đầu ra). Trong báo cáo này, X (đầu vào) là các thông tin liên quan về ngôi nhà và Y (đầu ra) là tiền nhà. Đường hồi quy là đường phù hợp nhất cho mô hình của chúng ta. Có khá nhiều phiên bản Linear Regression khác nhau ví dụ như Simple Linear Regression, Logistic Regression, Ordinal Regression,... trong báo cáo này chúng tôi chọn sử dụng Multiple Linear Regression bởi vì dữ liệu đầu vào gồm nhiều biến độc lập. Linear Regression sử dụng để đưa ra dự đoán cho một số đầu vào. Nhưng ngoài việc đưa ra dự đoán, Linear Regression còn có một số ưu điểm khác như linh hoạt hơn và có khả năng ứng dụng rộng rãi; có ít **Black box** và dễ sử dụng; giúp ta hiểu rõ hơn về luận thống kê tổng thể.

Vậy khi nào thì dùng Linear Regression? Linear Regression cung cấp một tính toán khoa học để xác định và dự đoán kết quả trong tương lai. Khả năng tìm dự đoán và đánh giá chúng có thể giúp mang lại lợi ích cho nhiều doanh nghiệp và cá nhân, chẳng hạn như các hoạt động tối ưu hóa và tài liệu nghiên cứu chi tiết. Ví dụ: Bạn có thể thu nhập dữ liệu để giúp bạn tối ưu hóa các hoạt động tiếp thị hoặc sản xuất của mình bằng cách sử dụng quy trình này để phân tích mối quan hệ giữa các yếu tố góp phần khác nhau. Tương tự như vậy, bạn cũng có thể có được các tài liệu nghiên cứu dữ liệu chi tiết về mối quan hệ giữa các yếu tố quan trọng và đưa chúng vào các bài thuyết của các bên liên quan, kế hoạch cải tiến hoặc tài liệu nghiên cứu.

- **Siêu tham số**
  - **Fit_intercept: bool, default=True**: Có nên tính toán hệ số chặn cho mô hình này không. Nếu được đặt thành False sẽ không có phần chặn nào được sử dụng cho phép.
  - **copy_X: bool, default=True**: Nếu True, X sẽ được sao chép; nếu không, nó có thể bị đè.
  - **n_jobs: int, default=False**: Số lượng công việc được sử dụng tính toán. Điều này sẽ chỉ cung cấp khả năng

tăng tốc trong trường hợp có vấn đề đủ lớn, đó là nếu trước tiên n_targets >1 và thứ hai là X thư thớt hoặc nếu dương đặt thành True. False có nghĩa là 1 trừ khi trong ngữ cảnh joblib.parallel_backend - 1 có nghĩa là sử dụng tất cả các bộ xử lý.
  - **positive: bool, default=False**: Khi được đặt thành **True**, buộc các hệ số phải dương. Tùy chọn này chỉ được hỗ trợ cho các mảng dày đặc

Ưu điểm, hạn chế và tối ưu:

- **Ưu điểm**:
  - **Dễ dàng thực hiện**: Mô hình Linear Regression dễ thực hiện về mặt tính toán vì nó không đòi hỏi nhiều chi phí kỹ thuật, kể cả trước khi khởi chạy mô hình cũng như trong quá trình bảo trì mô hình.
  - **Khả năng diễn giải**: Không giống như các mô hình **deep learning** khác (neural networks), Linear Regression tính tương đối đơn giản. Kết quả là thuật toán này vượt trội so với các mô hình black-box thiếu sot trong việc chứng minh biến đầu vào nào khiến biến đầu ra thay đổi.
  - **khả năng mở rộng**: Linear Regression không nặng về mặt tính toán và do đó, rất phù hợp trong các trườn hợp cần mở rộng quy mô. Ví du: Mô hình có thể mở rộng quy mô tốt liên quan đến khối lượng dữ liệu tăng lên (big data).
- **Hạn chế**:
  - Hạn chế đầu tiên của Linear Regression là nó rất **nhạy cảm với nhiễm** (sensitive to noise). Vì vậy, trước khi thực hiện Linear Regression, các nhiễu (outlier) cần phải được loại bỏ. Bước này được gọi là tiền xử lý (pre-processing).
  - Hạn chế thứ hai của Linear Regression là nó **không biểu diễn được các mô hình phức tạp**. Mặc dù chúng ta thấy rằng phương pháp này có thể được áp dụng nếu quan hệ outcome và input không nhất thiết phải là tuyến tính, nhưng mối quan hệ này vẫn đơn giản nhiều so với các mô hình thực tế.
- **Tối ưu**:
  - Linear Regression là một mô hình đơn giản, lời giải cho phương trình đạo hàm bằng 0 cũng khá đơn giản. Trong hầu hết các trường hợp, chúng ta không thể giải được phương trình đạo hàm bằng 0.
  - Những có một điều chúng ta nên nhớ, **còn tính được đạo hàm là còn có hy vọng**.

Tuning: Đối với mô hình Multiple Linear Regression, các tham số đầu vào không quá nhiều nên việc tune các tham số này không đưa đến việc chuẩn hóa mô hình. Vì vậy nên chúng tôi không thực hiện điều chỉnh tham số đối với mô hình này.

D. XGBoost Regressor

SVTH: Hồ Thịnh

Giới thiệu: XGBoost hay Extreme Gradient Boost là một phiên bản cải tiến của Gradient Boost. XGBoost hiệu quả hơn Gradient Boost ở điểm nó xem xét khả năng mất mát đối với tất cả các lần phân tách có thể xảy ra để tạo một nhánh mới (đặc biệt nếu bạn xem xét trường hợp có hàng nghìn tính năng và do đó có hàng nghìn lần phân tách có thể xảy ra). XGBoost giải quyết vấn đề không hiệu quả này bằng cách xem xét sự phân bổ các đặc trưng trên tất cả các điểm dữ liệu trong một lá và sử dụng thông tin này để giảm không gian tìm kiếm của các phần tách tính năng có thể có. Mặc dù XGBoost thực hiện một số regularize tricks, nhưng cho đến nay, việc tăng tốc này là tính năng hữu ích nhất của thư viện, cho phép điều tra nhanh chóng nhiều cài đặt siêu tham số. Điều này rất hữu ích vì có rất nhiều siêu tham số để điều chỉnh. Gần như tất cả chúng đều được thiết kế để hạn chế việc overfitting (cho dù các mô hình cơ sở của bạn có đơn giản đến đâu, nếu bạn dán hàng nghìn mô hình lại với nhau thì chúng sẽ overfit).

Ưu nhược điểm

- **Ưu điểm**:
  - **Regularization**: XGBoost có quy trình chuẩn hóa L1 (Lasso Regression) và L2 (Ridge Rergession) tích hợp giúp ngăn mô hình overfit. Đó là lý do tại sao, XGBoost còn được gọi là dạng GBM thông thường (Gradient Boosting Machine).
  - **Khả năng xử lý các giá trị bị thiếu**: XGBoost có khả năng tích hợp sẵn để xử lý các giá trị bị thiếu. Khi XGBoost gặp một giá trị bị thiếu tại một nút, nó sẽ thử tách cả nhánh trái và nhánh phải và tìm hiểu cách dẫn đến tổn thất cao hơn cho mỗi nút. Sau đó, nó cũng làm như vậy khi làm việc trên dữ liệu thử nghiệm.
  - **Khả năng xử lý song song**: XGBoost sử dụng xử lý song song và đó là lý do tại sao nó nhanh hơn nhiều so với GBM. Nó sử dụng nhiều lõi CPU để thực hiện mô hình.
- **Nhược điểm**:
  - **Overfitting**: dễ xảy ra khi các tham số không được điều chỉnh hợp lý.
  - Thời gian đào tạo khá cao đối với tập dữ liệu lớn hơn, nếu bạn so sánh với catboost/lightgbm.

Tuning: Danh sách các siêu tham số cực kỳ đáng sợ đối với chúng tôi khi chúng tôi bắt đầu làm việc với XGBoost, vì vậy chúng tôi sẽ thảo luận về 4 tham số mà chúng tôi thấy quan trọng nhất khi đào tạo các mô hình của mình cho đến nay. 1. n_estimators(and early stopping): số lượng Decision Tree trong mô hình. 2. max_depth: chiều cao tối đa của Decision Tree. 3. learning_rate: biến thể hiện khả năng học của mô hình. Chúng tôi nhận thấy rằng khi tăng biến này giúp tăng hiệu suất mô hình mặc dù sẽ tốn thời gian huấn luyện hơn. 4. reg_alpha và reg_lambda: biến có chức năng kiểm soát thông số L1 và L2 nhằm giới hạn trọng số (extreme weights) trả về.

III. THỰC NGHIỆM

A. Bộ dữ liệu

Trong báo cáo này nhóm chúng tôi xử dụng bộ dữ liệu Ames Housing được giới thiệu bởi Dean De Cock (2011) để tiến hành thực nghiệm trên các mô hình đã giới thiệu ở phần trước. Bộ dữ liệu chứa các thông tin (thuộc tính) của từng

ngôi nhà thông qua các cột trong bảng liên quan đến: 1. Đặc điểm ngôi nhà (phòng ngủ, nhà để xe, lò sưởi, hồ bơi, hiên nhà, v.v.), 2. Vị trí (khu phố), 3. Thông tin lô (phân vùng, hình dạng, kích thước, v.v.), 4. Đánh giá về tình trạng và chất lượng, 5. Giá bán.
Chi tiết về các thuộc tính trong bộ dữ liệu được trình bày ở trong tệp sau: Link

B. Tiền xử lý

Từ dữ liệu thô, chúng tôi tiến hành thống kê thông tin từ các cột thông tin trong bộ dữ liệu và nhận thấy có một số cột chứa nhiều thông tin NaN cụ thể là các cột MiscFeature, PoolQC, Fence, Alley; những thông tin này không giúp ích cho quá trình dự đoán nên chúng tôi tiến hành bỏ các cột này ra khỏi bộ dữ liệu. Sau đó trên bộ dữ liệu còn lại chúng tôi chuẩn hóa các cột dữ liệu kiểu số, kiểu chuỗi thông qua thư viện pandas nhằm giảm số chiều dữ liệu cũng như hạn chế các lỗi có thể xảy ra trong quá trình train. Đó là toàn bộ các bước chúng tôi xử lý để tiến hành bước train dữ liệu trên các mô hình đã trình bày ở trên.

C. Kết quả

Source code: Link
Sau khi thực nghiệm huấn luyện cũng như là điều chỉnh các siêu tham số trên bộ dữ liệu Ames Housing với các mô hình đã để cập ở phần trước chúng tôi nhận thấy được có hai mô hình overfitting với các tham số ở giá trị mặc định đó là Gradient Boosting Regressor và XGB Regressor, điều này được thể hiện thông qua Score của mô hình trên tập test trước và sau khi điều chỉnh siêu tham số. Bảng 1 dưới đây là bảng kết quả chúng tôi tổng hợp lại sau quá trình thực nghiệm. Xem chi tiết hơn ở đường dẫn source code.

	R2 Score	Best Params	R2 Score after Tuning
Random Forest	0.81	n_estimators': 400, 'min_samples_split': 5, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 20	0.80
Gradient Boost	0.82	learning_rate': 0.01, 'max_depth': 4, 'n_estimators': 500, 'random_state': 1, 'subsample': 0.5	0.81
Linear Regression	0.72	-	-
XG Boost	0.79	n_estimators': 100, 'min_child_weight': 3, 'max_depth': 10, 'learning_rate': 0.1, 'booster': 'gbtree', 'base_score': 0.25	0.81

Bảng I
BẢNG KẾT QUẢ TOÀN BỘ QUÁ TRÌNH THỰC NGHIỆM.

IV. KẾT LUẬN

Qua báo cáo này, chúng tôi đã tìm hiểu được về tổng quan các mô hình như đã giới thiệu trên bao gồm Random Forest Regressor, Gradient Boosting Regression, Linear Regression và XGBoost Regressor cũng như ưu nhược điểm của từng loại mô hình, sâu hơn nữa chúng tôi đã tìm hiểu về ngữ cảnh để áp dụng các mô hình để giải quyết những bài toán hồi quy. Cuối cùng và cũng không kém phần quan trọng, chúng tôi đã tiến hành cài đặt mô hình kèm với đó là điều chỉnh các siêu tham số nhằm giúp cho mô hình có được kết quả chính xác hơn và đưa ra nhận xét thông qua thực nghiệm đó.

TÀI LIỆU THAM KHẢO

[1] MLOps BLog, Random Forest Regression: When Does It Fail and Why?
[2] sklearn Documents/RandomForestRegressor
[3] Classification and Regression by RandomForest
[4] Using Gradient Boosting Regressor to Predict Stress Intensity Factor of a Crack Propagating in Small Bore Piping
[5] A Gradient-Based Boosting Algorithm for Regression Problems
[6] sklearn Documents/LinearRegression
[7] Gradient Boosting and XGBoost
[8] The Ames housing dataset