ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Báo cáo bài tập thực hành 1

Đề tài: Dự đoán lương dựa theo dữ liệu việc làm

Môn học: Nhập môn học máy

Sinh viên thực hiện:

Tô Hữu Danh (23127336)

Nguyễn Đăng Hưng (23127050)

Lê Phú Cường (23127164)

Nguyễn Bá Đăng Khoa (23127392)

Giáo viên hướng dẫn:

Thầy Võ Nhật Tân

Ngày 13 tháng 11 năm 2025

Mục lục

Danh sách bảng

Danh sách hình vẽ

1 Mô tả đồ án và lý do chọn đề tài

1.1 Mô tả đồ án

Đồ án này sẽ tập trung nghiên cứu bài toán xây dựng mô hình hồi quy tuyến tính để dự doán giá xe ô tô dựa trên tập hợp các thuộc tính kỹ thuật và đặc điểm của xe.
Sau khi nghiên cứu đặc trưng của tập dữ liệu, nhóm sẽ khám phá 2 hướng tiếp cận chính sau:

- Mô hình Chính quy hoá (*Regularizaiton*): Triển khai các kỹ thuật bao gồm Ridge (L2), Lasso (L1) và Elastic Net (kết hợp L1 và L2) nhằm xử lý đa cộng tuyến và lựa chọn thuộc tính thông qua các chuẩn của sai số.

- Mô hình Lựa chọn thuộc tính tường minh (*Explicit Feature Selection*): Triển khai mô hình hồi quy tuyến tính cơ bản, nhưng được tối ưu hoá bằng cách áp dụng thuật toán Forward Selection dựa trên Bayesian Information Criterion để chủ động loại bỏ các thuộc tính nhiễu, không cần thiết ngay từ bước tiền xử lý.

Đầu tiên, dữ liệu sẽ được chuẩn hóa và chia ra các tập train (60%), validation (20%), và test (20%). Sau đó, các mô hình sẽ được đem train dựa trên tập train hoặc tập train đã được đưa qua BIC trong hướng tiếp cận thứ 2. Với hướng tiếp cận thứ nhất, các thông số sẽ được chọn thông qua grid search. Cuối cùng, các mô hình sẽ được đánh giá dựa trên các chỉ số R^2, MAE và RMSE trên tập validation và tập test, nhằm tìm ra mô hình có hiệu quả tốt nhất.

1.2 Lý do chọn đề tài

Trong các bài toán hòi quy thực tế, dữ liệu hiếm khi ở trạng thái lý tưởng. Các mô hình OLS (*Ordinary Least Square*) thường hoạt động không tốt khi đối mặt với dữ liệu có hiện tượng đa cộng tuyến (các biến dự đoán tương quan mạnh với nhau). Dẫn đến việc mô hình trở nên thiếu ổn định, có phương sai cao (*high variance*), nhạy cảm với dữ liệu huấn luyện và mất khả năng diễn giải.
Do đó, nhóm được thúc đẩy bởi mong muốn tìm ra câu trả lời dựa trên dữ liệu thực nghiệm: Liệu việc chấp nhận một lượng độ chệch nhỏ trong các mô hình chính quy hoá có hiệu quả hơn việc loại bỏ hoàn toàn các thuộc tính không? Hay một phương pháp lựa chọn thuộc tính thống kê cẩn thận kết hợp với mô hình đơn giản mới là giải pháp tối ưu? Kết quả của đồ án này sẽ là câu trả lời về sự đánh đổi giữa các kỹ thuật và đưa ra kết luận cụ thể cho bài toán dự đoán trên tập dữ liệu này.

2 Mô tả và kỹ thuật dữ liệu

2.1 Mô tả dữ liệu

Bộ dữ liệu về giá xe dựa trên các thuộc tính được lấy từ Kaggle của tác giả Ashish (đường dẫn đến bộ dữ liệu: Dữ liệu giá xe). Toàn bộ thuộc tính của dữ liệu được mô tả ở bảng sau:

Tên thuộc tính	Kiểu dữ liệu	Mô tả
`car_ID`	Integer	Chỉ số định danh cho từng chiếc xe trong tập dữ liệu, đếm từ 0 - 205 (tương ứng với 206 chiếc xe trong tập dữ liệu).
`symboling`	Integer	Mức xếp hạng rủi ro bảo hiểm của chiếc xe, giá trị này càng lớn thì xe có rủi ro càng cao (các loại xe thể thao, xe địa hình, ...) và ngược lại.
`CarName`	String	Tên của chiếc xe.
`fueltype`	String	Loại nhiên liệu, gồm 2 loại duy nhất là `gas` và `diesel`.
`aspiration`	String	Mô tả cách không khí được nạp vào xi-lanh của động cơ xe, gồm 2 loại duy nhất là `std` (tiêu chuẩn) và `turbo` (tăng áp).
`doornumber`	String	Số lượng cửa của xe, cũng chỉ có 2 loại duy nhất là `two` và `four`.
`carbody`	String	Kiểu dáng của chiếc xe (`sedan`, `hatchback`, ...).
`drivewheel`	String	Cho biết bánh xe nào nhận lực kéo từ động cơ để đẩy chiếc xe đi, gồm 2 loại duy nhất là `fwd` (bánh trước) và `rwd` (bánh sau).
`enginelocation`	String	Vị trí đặt động cơ, gồm 2 loại duy nhất là `front` và `rear`.
`wheelbase`	Decimal	Chiều dài trục cơ sở.

Bảng 1: Bảng mô tả chi tiết các thuộc tính

2.2 Tiền xử lí dữ liệu

2.2.1 Loại bỏ thuộc tính ID không mang giá trị dự đoán

Trước tiên, ta cần loại bỏ thuộc tính ID (chỉ số định danh của mẫu) khỏi bộ dữ liệu. Thuộc tính ID thường là một mã định danh duy nhất cho mỗi mẫu và không chứa thông tin hữu ích để dự đoán biến mục tiêu. Việc giữ lại ID có thể gây nhiễu cho mô hình học máy do bản chất duy nhất của nó

– mỗi giá trị ID chỉ xuất hiện một lần và không liên quan đến nhãn dự đoán. Loại bỏ cột ID giúp giảm số chiều dữ liệu và tránh việc mô hình học máy học những mẫu giả tạo từ ID.

2.2.2 Khảo sát kích thước và kiểu dữ liệu

Tiếp theo, ta tiến hành khảo sát tổng quan về bộ dữ liệu nhằm nắm rõ kích thước và kiểu dữ liệu của các trường. Cụ thể:

- Ta dùng phương thức `.shape` để xác định số lượng mẫu và số thuộc tính trong tập dữ liệu. Kết quả dữ liệu có 205 mẫu và 25 thuộc tính.
- Ta dùng phương thức `.info()` của pandas để liệt kê các cột, kiểu dữ liệu từng cột, số lượng giá trị không rỗng cũng như lượng bộ nhớ sử dụng. Kết quả kiểu dữ liệu của các cột phù hợp với dữ liệu, dữ liệu không bị thiếu và không có sai lệch trong định dạng dữ liệu.

2.2.3 Kiểm tra dữ liệu trùng lặp

Sau khi xem xét cấu trúc dữ liệu, ta kiểm tra sự trùng lắp trong tập dữ liệu. Bằng việc sử dụng phương thức `.duplicated()` của pandas, ta xác định được các bản ghi không có dữ liệu trùng lắp. Bước kiểm tra này là cần thiết vì nếu dữ liệu trùng lắp quá nhiều có thể làm cho mô hình tin rằng mẫu đó quan trọng hơn, từ đó mô hình có thể bị thiên lệch và giảm khả năng tổng quát hóa

2.2.4 Chia dữ liệu thành các tập huấn luyện, xác thực và kiểm thử

Sau các bước làm sạch dữ liệu, ta tiến hình chia bộ dữ liệu thành ba tập: tập huấn luyện (60%), tập xác thực (20%) và tập kiểm thử (20%). Tập huấn luyện được sử dụng để huấn luyện mô hình, tập validation dùng để điều chỉnh siêu tham số và đánh giá mô hình một cách công bằng trong quá trình phát triển (do mô hình không nhìn thấy tập này khi huấn luyện), và tập kiểm thử được giữ riêng để đánh giá cuối cùng về khả năng tổng quát hóa của mô hình. Việc phân chia theo tỉ lệ 60/20/20 được lựa chọn nhằm đảm bảo đủ dữ liệu cho huấn luyện (giảm phương sai của mô hình) đồng thời vẫn dành một phần hợp lý cho việc xác thực và kiểm thử. Việc giữ tập kiểm thử tách biệt đến cuối cùng mô phỏng tình huống áp dụng mô hình trên dữ liệu thực tế chưa từng được thấy, do đó độ chính xác thu được trên tập này phản ánh sát hơn hiệu năng kỳ vọng trong thực tiễn.

2.2.5 Mã hóa one-hot cho biến không có kiểu dữ liệu là số

Đối với các thuộc tính có kiểu dữ liệu là Object mà ta đã khảo sát ở trên, ở bước này ta áp dụng kỹ thuật mã hóa one-hot để biến chúng thành các đặc trưng nhị phân. Phương pháp one-hot encoding sẽ tạo ra các biến giả (dummy variables) cho mỗi hạng mục có thể có của thuộc tính phân loại: mỗi cột mới đại diện cho một hạng mục, nhận giá trị 1 nếu mẫu thuộc hạng mục đó và 0 nếu không. Cụ thể, với pandas, chúng tôi sử dụng hàm `pd.get_dummies` trên tập dữ liệu huấn luyện để chuyển đổi mỗi cột phân loại thành nhiều cột dummy tương ứng. Kỹ thuật này cho phép mô hình học máy của nhóm xử lý được dữ liệu phân loại một cách hiệu quả, bởi vì 4 mô hình của nhóm chỉ làm việc với dữ liệu số.

2.2.6 Căn chỉnh ma trận đặc trưng giữa các tập dữ liệu bằng reindex

Sau khi thực hiện one-hot encoding, một vấn đề quan trọng là đảm bảo tập validation và tập kiểm thử có cùng cấu trúc đặc trưng như tập huấn luyện. Do quá trình `get_dummies` tạo cột dựa trên các hạng mục xuất hiện trong từng tập, nên các tập khác nhau có thể lệch nhau về số lượng hoặc tên cột dummy (ví dụ: một hạng mục nhất định có thể xuất hiện ở tập kiểm thử mà không có ở tập huấn luyện, hoặc ngược lại).

Để giải quyết việc này, chúng ta sử dụng phương pháp reindex của pandas để căn chỉnh cột của tập validation và kiểm thử theo đúng tập cột của tập huấn luyện. Cụ thể, sau khi mã hóa one-hot trên tập huấn luyện và tập con cần căn chỉnh, chúng ta thực hiện: `X_val = X_val.reindex(columns=X_train.columns, fill_value=0) (tương tự cho X_test)`. Cách làm này bổ sung những cột còn thiếu trong tập validation/test (nếu tập huấn luyện có cột dummy mà tập khác không có, thì cột đó sẽ được thêm vào tập kia và điền giá trị 0 cho tất cả các hàng), đồng thời loại bỏ những cột thừa không có trong tập huấn luyện (tương ứng với hạng mục không xuất hiện khi huấn luyện). Kết quả là mọi tập dữ liệu đều có cùng số chiều và tên cột đặc trưng, sẵn sàng để đưa vào mô hình.

Bước căn chỉnh này đặc biệt quan trọng bởi nếu one-hot encoding tạo ra số cột khác nhau giữa tập huấn luyện và tập kiểm thử, mô hình sẽ gặp lỗi hoặc hoạt động sai; thậm chí lỗi này không hiển thị rõ ràng mà có thể chỉ làm giảm độ chính xác dự báo do bất nhất trong ma trận đặc trưng

2.2.7 Chuẩn hóa đặc trưng bằng StandardScaler

Cuối cùng, trước khi huấn luyện mô hình, chúng ta tiến hành chuẩn hóa các đặc trưng đầu vào bằng phương pháp StandardScaler. Bộ chuẩn hóa StandardScaler sẽ loại bỏ giá trị trung bình của mỗi đặc trưng và chia cho độ lệch chuẩn, đưa các đặc trưng về cùng một thang đo với trung bình 0 và phương sai đơn vị.

Có hai lợi ích chính từ việc chuẩn hóa dữ liệu:

- các thuật toán học máy hồi quy tuyến tính thường hội tụ nhanh hơn và cho kết quả tốt hơn khi các đặc trưng được đưa về cùng một quy mô
- tránh hiện tượng một số thuộc tính có giá trị tuyệt đối lớn lấn át ảnh hưởng của các thuộc tính khác chỉ vì đơn vị đo hoặc phạm vi khác nhau.

Quá trình chuẩn hóa được thực hiện cẩn trọng để không gây rò rỉ dữ liệu. Cụ thể, chúng ta fit đối tượng StandardScaler trên tập huấn luyện (tính toán trung bình và độ lệch chuẩn của từng thuộc tính dựa trên dữ liệu huấn luyện), sau đó dùng scaler này để transform tập validation và tập kiểm thử theo cùng thông số. Lưu ý, chúng ta tuyệt đối không fit StandardScaler trên toàn bộ dữ liệu hay trên tập kiểm thử, bởi làm như vậy đồng nghĩa với việc sử dụng thông tin của tập kiểm thử trong quá trình chuẩn hóa tập huấn luyện – một dạng "data leakage" có thể dẫn đến đánh giá quá mức hiệu năng mô hình. Thay vào đó, việc chỉ dùng thống kê của tập huấn luyện để chuẩn hóa mọi tập khác đảm bảo rằng quy trình tiền xử lý của chúng ta tuân thủ chặt chẽ nguyên tắc huấn luyện trên dữ liệu quá khứ và kiểm thử trên dữ liệu chưa thấy. Nhờ bước chuẩn hóa này, dữ liệu đầu vào cho mô hình có phân phối ổn định, giúp mô hình học được trọng số tối ưu mà không bị ảnh hưởng bởi khác biệt về thang đo của các thuộc tính.

2.3 Chọn lọc thuộc tính

2.3.1 Mục tiêu

Với tập dữ liệu được chọn, nếu sử dụng tất cả các thuộc tính có sẵn sẽ vô tình chứa các thuộc tính gây nhiễu, làm cho kết quả sau cùng của mô hình chua phải là tốt nhất. Vì thế, cần xây dựng được một mô hình hôi quy tốt nhất bắng cách xác định một tập con các thuộc tính (hay biến dự đoán) phù họp X để dự đoán biến mục tiêu y.

2.3.2 Thách thức

Tuy vậy, quá trình lựa chọn thuộc tính không phải chỉ là áp dụng công thức có sẵn mà phải đảm bảo mô hình vẫn cân bằng được các yếu tố sau:

- ***Độ vừa vặn (Goodness-of-Fit):*** Mô hình phải giải thích được sự biến thiên trong dữ liệu, điều đó sẽ được thể hiện bằng độ lớn của R^2.
- ***Tính đơn giản (Parismony):*** Mô hình nên càng đơn giản càng tốt, càng phức tạp (hay số lượng biến lớn) thì càng khó diễn giải và dễ dẫn đến *overfitting*.

Ngoài ra, thách thức về mặt tính toán cũng quan trọng không kém khi với p thuộc tính, sẽ có 2^p mô hình tập con có thể. Việc kiểm tra toàn bộ là bất khả thi về mặt tính toán nếu p lớn (từ 40 trở lên).

2.3.3 Xây dựng mô hình chọn tập con và đánh giá

Ta cần xây dựng một mô hình chọn lọc từng thuộc tính, do đó mỗi khi mô hình chọn được một thuộc tính thì nó sẽ có thêm một biến dẫn đến cần kỹ thuật nâng cao hơn trong việc so sánh nó với phiên bản cũ của chính nó. Để so sánh các mô hình có số lượng biến khác nhau, không thể chỉ sử dụng R^2 (vì chỉ số này sẽ luôn tăng khi thêm biến) mà cần thêm các tiêu chí có "điều khoản phạt" (*penalty*) cho sự phức tạp. Đề xuất tiêu biểu nhất là **Tiêu chí thông tin (*Information Criterion*)** [1], phương pháp này yêu cầu các tiêu chí phải cân bằng trực tiếp giữa độ vừa vặn (đo bằng Log-Likelihood, liên quan đến RSS) và số lượng tham số p để tối thiểu hoá giá trị tiêu chí chỉ bằng một con số duy nhất dùng cho việc đánh giá mô hình. Có hai loại chỉ số:

- **AIC (Akaike Information Criterion):** [1] Công thức tổng quát dựa trên Log-Likelihood (L) của mô hình. Đối với mô hình hồi quy tuyến tính sử dụng ước tính tổng bình phương phần dư (RSS), công thức có thể được viết là

$$AIC = n \ln \frac{RSS}{n} + 2k.$$

  Trong đó $\left(n \ln \frac{RSS}{n}\right)$ là thành phần dùng để đo mức độ vừa vặn, $(2k)$ là một khoảng trừng phạt (*penalty*), với k là tổng số tham số mà mô hình phải ước tính (bao gồm tất cả các thuộc tính và hệ số chặn). Với mỗi thuộc tính được thêm vào, k sẽ tăng thêm 1 đơn vị dẫn đến AIC

bị tăng 2 đơn vị. Điều đó dẫn đến khi thêm một thuộc tính vào mô hình, RSS sẽ giảm nhưng $2k$ sẽ tăng để phạt mô hình. AIC chỉ giảm (tức là mô hình sẽ tốt hơn) khi mức độ cải thiện RSS đủ lớn để bù lại khoản phạt.

- **BIC (Bayesian Information Criterion):** [1] Tương tự như AIC, nhưng xuất phát từ lý thuyết xác suất Bayes. Được thiết kế để tìm ra mô hình có xác suất cao nhất là "mô hình thực sự" (*true model*) tạo ra dữ liệu. Công thức của BIC rất giống AIC nhưng có một khoản phạt khác biệt

$$BIC = n \ln \frac{RSS}{n} + k \ln n.$$

Trong đó $\left(n \ln \frac{RSS}{n}\right)$ giống hệt AIC, khoản phạt $(k \ln n)$ được coi là nặng hơn khá nhiều vì n (số lượng mẫu) thường rất lớn. Điều đó chứng tỏ rằng BIC yêu cầu mô hình đưa ra thuộc tính mang lại sự cải thiện rất lơn cho RSS, đây là một điểm có thể được coi là lợi vì khi đó BIC có thể đưa ra mô hình đơn giản hơn hay ít thuộc tính hơn so với AIC. Vì vậy, BIC sẽ phù hợp vơi một mô hình cô đọng, dễ diễn giải và tránh các thuộc tính có ảnh hưởng yếu.

2.3.4 Thuật toán tìm kiếm tự động

Như đã trình bày ở trên, vì số lượng mô hình cần kiểm tra là quá lớn (2^p mô hình) nên cần áp dụng các thuật toán tìm kiếm tham lam để khám phá không gian mô hình theo cách tối ưu nhất, tránh lãng phí tài nguyên tính toán hay tối thiểu nhất là có thể "khả thi hoá" quy trình tính toán.
Một thuật toán được đề xuất trong sách *"Applied Linear Regression"* là **Forward Selection** [2], với ý tưởng chính là tiếp cận từ dưới lên bằng cách xây dựng mô hình từng bước một từ mô hình ban đầu chỉ có một biến hằng số. Với mỗi lần lặp, thuật toán sẽ chọn mô hình với bộ thuộc tính mới (vừa mới được thêm một thuộc tính) có chỉ số AIC/BIC tốt hơn so với mô hình trước đó (chỉ số AIC/BIC càng thấp là mô hình càng tốt), điều kiện dừng của thuật toán là khi tất cả các thuộc tính đều đã được kiểm tra.
Để trực quan hơn, dưới đây là mã giả của thuật toán được viết lại từ mã nguồn của tác giả **talhahascelik** được đăng trong một repository Github [3]

Algorithm 1 Mã giả của thuật toán Forward Selection

```
remaining_features ← list(X.columns)
remaining_features.remove('intercept')
selected_features ← ['intercept']
current_best_bic ← OLS(y, X[selected_features]).fit().bic
while remaining_features is not empty do
    best_feature_to_add ← None                ▷ Biến dùng để chọn ra thuộc tính tốt nhất
    best_new_bic ← current_best_bic           ▷ Biến để tính toán chỉ số BIC tốt nhất
    for feature ∈ remaining_features do       ▷ Xét từng thuộc tính trong tập thuộc tính còn lại
        model_features ← selected_features + [feature]   ▷ Tạo danh sách các thuộc tính tạm
thời gồm thuộc tính đã chọn và thuộc tính đang xét
        model ← OLS(y, X[model_features]).fit()          ▷ Huấn luyện mô hình OLS mới với bộ
thuộc tính tạm thời và lấy BIC mới
        new_bic ← model.bic
        if new_bic < best_new_bic then ▷ So sánh với BIC tốt nhất để hoán đổi nếu nó tốt hơn
            best_new_bic ← new_bic
            best_feature_to_add ← feature
        end if
    end for
    if best_feature_to_add ≠ None then        ▷ Nếu có thuộc tính mới được chọn
        selected_features.append(best_new_to_add)   ▷ Thêm vào danh sách được chọn và loại
khỏi danh sách thuộc tính còn lại
        remaining_features.remove(best_feature_to_add)
        current_best_bic ← best_new_bic       ▷ Cập nhật giá trị BIC mới
    end if
    if !best_feature_to_add ≠ None then return selected_features   ▷ Dừng khi không có
thuộc tính nào làm cho BIC tốt hơn
    end if
end while
```

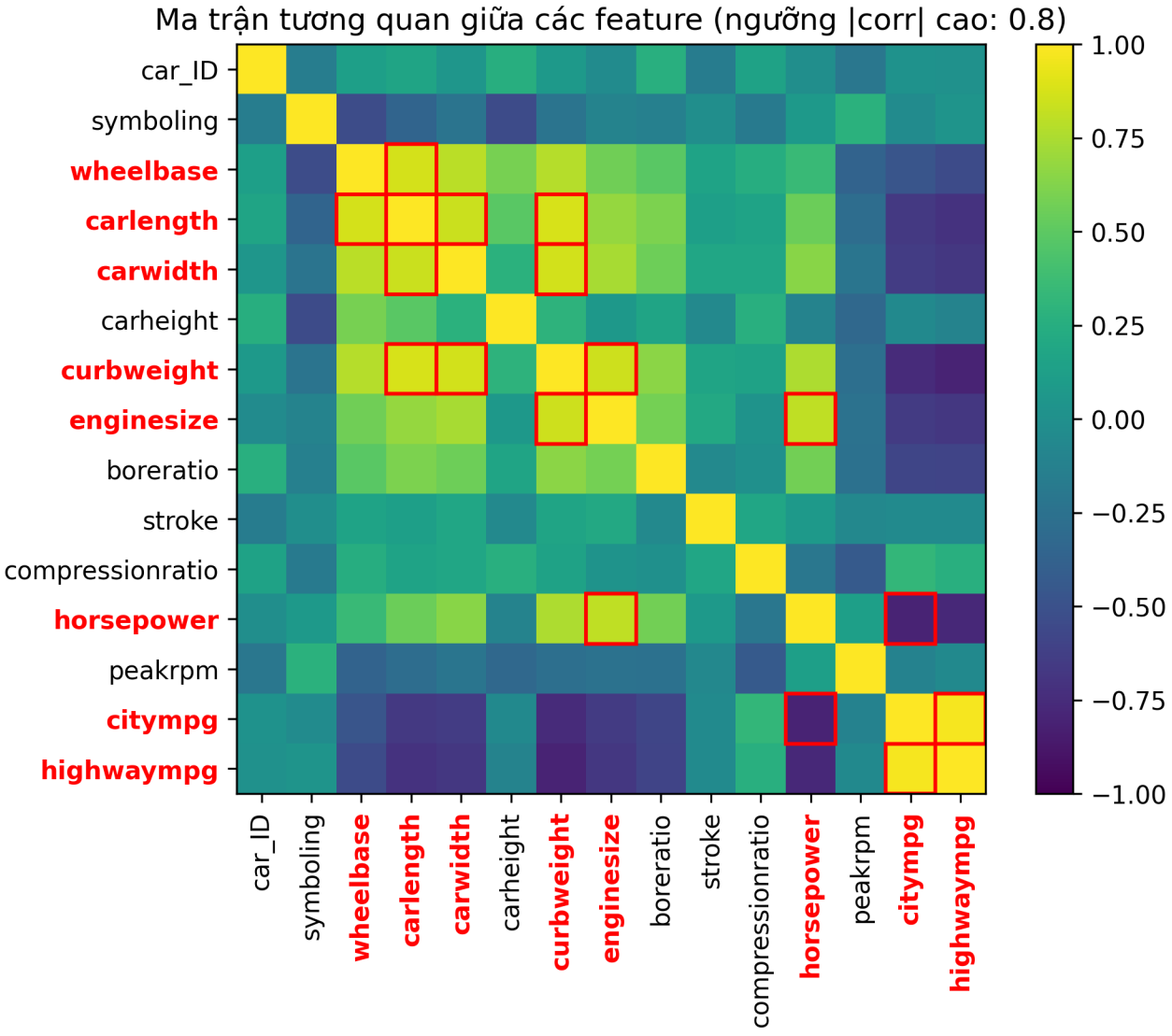
Thuật toán Foward Selection là một giải pháp hoàn toàn khả thi và hợp lí dành cho tập dữ liệu lớn ($p > n$) mà tại đó số thuộc tính nhiều hơn số mẫu. Lựa chọn tiến chỉ yêu cầu đánh giá tối đa $\frac{p(p+1)}{2}$ [2] mô hình so với số lượng mô hình thực sự có là 2^p. Sự giảm thiểu đáng kể về mặt chi phí tính toán sẽ làm thuật toán trở nên khả thi mặc cho p rất lớn. Hơn nữa, Forward Selection có một lợi thế cấu trúc so với phương pháp **Backward Elimination** [2] vì nó bắt đầu từ mô hình rỗng (*null model*) và xây dựng dần lên thay vì yêu cầu $n > p$ như Backward Elimination để ước tính mô hình đầy đủ.

Tuy nhiên, hạn chế cơ bản của Forward Selection nằm ở bản chất tham lam (*greedy*) của thuật toán vốn chỉ tìm kiếm giải pháp tối ưu cục bộ (*local optimum*) ở mỗi bước mà không đảm bảo tìm được giải pháp tối ưu toàn cục (*global optimum*). Forward Selection có sự thiếu sót về bước lùi hay nói

cách khác là nó không có khả năng quay lui (*backtrack*), một thuộc tính khi đã được thêm vào mô hình sẽ không bao giờ bị loại bỏ ở các bước sau, ngay cả khi nó trở nên thừa thãi. Điều đó dẫn đến tiềm ẩn vấn đề che khuất (*masking problem*), nơi một biến ban đầu được chọn vì nó là đại diện tạm thời tốt nhất cho một cấu trúc dữ liệu, nhưng sau đó trở nên không còn ý nghĩa thống kê khi một biến tương quan mạnh hơn khác được thêm vào. Do đó, không thể đảm bảo rằng mô hình k-biến với bộ thuộc tính sau cùng là mô hình k-biến tốt nhất.

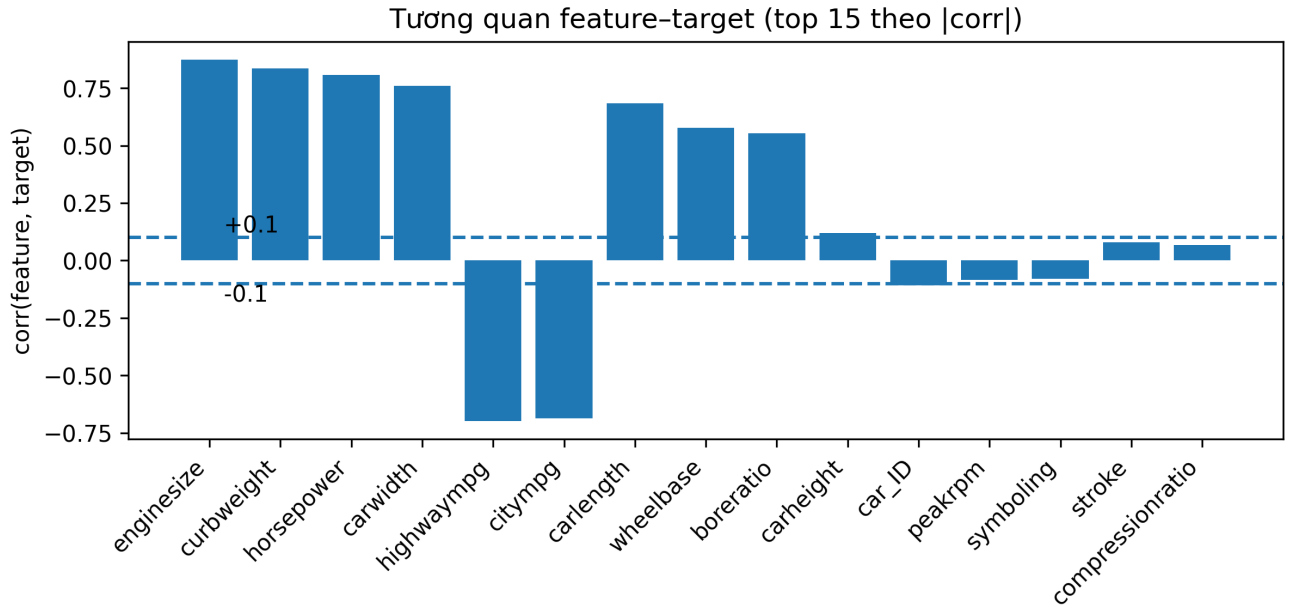
2.4 Phân tích dữ liệu và lựa chọn mô hình

Ta tiến hành trích xuất một số đặc trưng của các cột dữ liệu (trừ carname) của dataset:



Hình 1: Heatmap correlation giữa các feature

Có 8 cặp feature có correlation cao được highlight trên hình như là carlength với wheelbase, carwidth, curbweight; carwidth và curbweight, enginesize và curbweight, horsepower và enginesize, citympg và horsepower, và highwaympg với citympg.



Hình 2: Bar chart correlation giữa feature và target (price)

Ngược lại, khi kiểm tra correlation giữa các feature và target, ta thấy có những biến có correlation rất là thấp: carhieght, car_ID, peakrpm, symboling, stroke, và compressionratio.

=> Đây là một dataset vừa có các feature có correlation với nhau rất cao, vừa có các feature khác có correlation với target gần như bằng 0.

Với vấn đề đầu tiên, ta có thể giải quyết bằng Ridge. Với vấn đề thứ 2, ta có thể giải quyết bằng Lasso. Ngoài ra, thay vì thêm một chuẩn bậc nhất hoặc bậc 2 vào hàm loss nhằm giải quyết các vấn đề liên quan đến dataset, ta cũng có thể thử tối ưu hóa dataset ngay từ bước đầu tiên thông qua việc chọn các feature cho phù hợp.

Vì vậy, nhóm quyết định sẽ thử chạy dataset này thông qua các model: Ridge, Lasso, Elastic Net (Ridge + Lasso), và Linear Regression cơ bản nhưng các feature được chọn trước thông qua thuật toán Forward Selection BIC.

Với các model: Ridge, Lasso, Elastic, hệ số λ_1 và λ_2 được chọn đơn giản bằng grid search với target là R^2 cao nhất.

3 Lý thuyết về các mô hình và lựa chọn

3.1 Mô hình Linear Regression

Linear regression là một mô hình học có giám sát với khả năng tìm mối liên hệ tuyến tính giữa input và output.

Vấn đề: Model này nhận vào một vector $x \in \mathbb{R}^{D+1}$ gồm các đặt trưng của một mẫu và trả về giá trị dự đoán $y \in \mathbb{R}$. Ví dụ: Nhận vào các giá trị thuộc tính của một ngôi nhà như Diện tích sàn (m^2), số lượng phòng ngủ, khoảng cách tới trung tâm thành phố (m), tuổi nhà (năm) và trả về dự đoán giá của căn nhà đó.

Đánh giá hiệu năng: Mô hình Linear Regression thường được đánh giá bằng Mean Square Error (MSE) trên tập train:

$$MSE_{\text{train}} = \frac{1}{N}\|\hat{\mathbf{y}} - \mathbf{y}\|^2 = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

với $\hat{\mathbf{y}}$ là tập giá trị dự đoán, $\mathbf{y}$ là tập giá trị thực tế của tập train.

Huấn luyện mô hình: Các phương pháp huấn luyện mô hình Linear Regression được dựa trên cách tính toán vector trọng số ω để giảm MSE xuống thấp nhất. Một cách làm phổ biến là tính toán đạo hàm của hàm MSE theo ω và cho nó bằng 0 để tìm cực tiểu.

$$\nabla_{\mathbf{w}}(MSE_{\text{train}}) = \nabla_{\mathbf{w}}(\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{y}^T\mathbf{y}) = 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y}$$

$$\begin{aligned}
\nabla_{\mathbf{w}}(MSE_{\text{train}}) &= 0 \\
\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} &= 0 \\
\mathbf{X}^T\mathbf{X}\mathbf{w} &= \mathbf{X}^T\mathbf{y} \\
\mathbf{w} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}
\end{aligned}$$

3.2 Mô hình Ridge

3.2.1 Cơ sở lý thuyết

Hồi quy Ridge (***Ridge Regression***) là một kỹ thuật hồi quy tuyến tính đuợc điều chỉnh (*regularized*) nhằm giải quyết hiện tượng quá khớp (*overfitting*) bằng cách dùng các kỹ thuật Chính quy hoá (*Regularization*). Phương pháp này thêm một số hạng (thường là *penalty*) vào hàm mất mát (*loss function*) của mô hình. Thành phần phạt này dùng để đánh giá và kiểm soát độ phức tạp của mô hình

Cụ thể, mô hình Ridge sử dụng kỹ thuật *L2 Regularization*, thay vì chỉ tối thiểu hoá tổng bình phương sai số (*Mean Square Error - MSE*) như mô hình hồi quy tuyến tính cơ bản thì Ridge sẽ tối thiểu hoá một hàm mục tiêu mới:

$$J(w) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda.\|\mathbf{w}\|_2^2 \text{ [4]}$$

Trong đó, $\mathbf{w}$ là vector trọng số của mô hình và λ là siêu tham số (*hyperparameter*) điều chỉnh độ mạnh của thành phần phạt $\|\mathbf{w}\|_2^2$.

[5] Việc thêm thành phần phạt sẽ ép các trọng số của mô hình phải giữ ở mức nhỏ, co về gần 0. Điều này làm giảm sự phụ thuộc của mô hình vào bất kỳ một biến đầu vào cụ thể nào, khiến mô hình đơn giản hơn, ít nhạy cảm với nhiễu dữ liệu và tăng khả năng tổng quát hoá. Vì thế, bài toán tối ưu hàm mất mát của hồi quy Ridge thực chất là tối ưu song song hai thành phần bao gồm tổng bình phương sai số và thành phần phạt hay thành phần điều chuẩn (*regularization term*).

- Trường hợp $\lambda = 0$, thành phần điều chuẩn bị tiêu giảm và chúng ta quay về hồi quy tuyến tính.
- Trường hợp $\lambda \approx 0$, thành phần điều chuẩn trở nên ít quan trọng, mức độ kiểm soát quá khớp trở nên kém.
- Trường hợp λ lớn, mức độ kiểm soát lên độ lớn của các hệ số ước lượng tăng lên qua đó giảm bớt quá khớp.

Khi λ tăng dần, hồi quy Ridge có xu hướng thu hẹp hệ số ước lượng $\mathbf{w}$ từ mô hình.

3.2.2 Triển khai thực nghiệm

Mô hình Ridge được triển khai và cấu hình như sau:

- Pipeline: Mô hình được gói trong một `sklearn.pipeline.Pipeline` để chuẩn hoá quy trình xử lý.
- Chuẩn hoá (Scaling): Bước đầu tiên trong pipeline là chuẩn hoá dữ liệu, rất quan trọng đối với các mô hình được chính quy hoá như Ridge, vì thành phần phạt nhạy cảm với sự chênh lệch về thang đo (*scale*) của các biến đầu vào.
- Cấu hình: Mô hình Ridge được khởi tạo với siêu tham số `alpha=33.6` và `random_state=42` (được dùng để đảm bảo kết quả có thể được tái lập).
- Dữ liệu: Không giống như mô hình `baseline_linear` sử dụng bộ dữ liệu được chọn từ phương pháp chọn lọc thuộc tính ở phần trước, mô hình Ridge được huấn luyện trên bộ dữ liệu đầy đủ (đã được lưu lại vào `X_train` và `y_train`).

Siêu tham số `alpha` lúc này sẽ được lựa chọn thông qua các bước sau:

- Định nghĩa không gian tìm kiếm: Một tập hợp các giá trị `alpha` tiềm năng được định nghĩa bằng `numpy.logspace(-4, 3, 20)`, tức là một mảng chứa 20 giá trị thực phân bố theo thang logarit từ 10^{-4} đến 10^{3}. Việc sử dụng thang logarit cho phép khám phá hiệu qua các giá trị ở nhiều bậc độ lớn khác nhau.
- Đóng gói Pipeline: Dùng chính mô hình Ridge để chuẩn hoá dữ liệu trước khi huấn luyện mô hình, mục tiêu là tìm ra và đánh giá R^2.
- Kiểm định chéo (*Cross-Validation*): `GridSearchCV` được cấu hình để sử dụng phương pháp kiểm định chéo. Toàn bộ tập dữ liệu huấn luyện sẽ được chia thành 5 phần (*folds*). Quá trình tìm kiếm sẽ lặp lại 5 lần, mỗi lần sẽ có một phàn được giữ lại làm tập validation tạm thời và 4 phần còn lại sẽ dùng để huấn luyện.
- Tiêu chí đánh giá: Chỉ số được sử dụng để đánh giá là R^2
- Lựa chọn tối ưu: quy trình `GridSearchCV` tự động huấn luyện và đánh giá mô hình Ridge với từng giá trị `alpha` trong không gian tìm kiếm. Giá trị nào mang R^2 trung bình cao nhất (qua 5 lượt) sẽ được chọn làm siêu tham số.

Từ quy trình trên, mô hình Ridge chọn được giá trị `alpha` $= 33.6$.

3.3 Mô hình Lasso

3.3.1 Cơ sở lý thuyết

Lasso (Least Absolute Shrinkage and Selection Operator) là một phương pháp hồi quy tuyến tính sử dụng chuẩn hóa L1 (L1 regularization). Kỹ thuật này đồng thời thực hiện lựa chọn biến (variable selection) và điều chuẩn hóa (regularization) để nâng cao độ chính xác dự báo và khả năng diễn giải của mô hình. Nói cách khác, Lasso bổ sung một khoản phạt L1 vào hàm mất mát của hồi quy thường, khiến một số hệ số hồi quy bị kéo về 0, từ đó đơn giản hóa mô hình và giúp tránh hiện tượng quá khớp. [6]

Hàm mục tiêu của hồi quy Lasso (với dữ liệu có N mẫu, p đặc trưng) có dạng:

$$\min_{\beta_0,\boldsymbol{\beta}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

trong đó vế thứ nhất là lỗi bình phương trung bình (MSE) và vế thứ hai là khoản phạt $L1$ nhân với hệ số điều chuẩn λ. Tham số λ (ký hiệu điều chuẩn, trong scikit-learn tham số này được gọi là `alpha`) kiểm soát độ mạnh của hình phạt $L1$, qua đó quyết định mức độ phức tạp của mô hình. λ càng lớn thì mô hình bị phạt càng nhiều: các hệ số β_j bị kéo về 0 đáng kể hơn, nhiều hệ số nhỏ gần như bị triệt tiêu; kết quả là mô hình giữ lại rất ít biến quan trọng (tránh overfitting). Ngược lại, λ nhỏ chỉ phạt nhẹ, mô hình sẽ giữ lại nhiều đặc trưng hơn (gần với hồi quy thường). Tài liệu từ IBM mô tả: *"Larger values of lambda increase the penalty, shrinking more of the coefficients towards zero; this subsequently reduces the importance of (or altogether eliminates) some of the features from the model, resulting in automatic feature selection. Conversely, smaller values of lambda reduce the effect of the penalty, retaining more features within the model."*[7] (Dịch: Giá trị λ lớn làm tăng mức phạt, kéo nhiều hệ số hơn về gần 0; điều này làm giảm tầm quan trọng (hoặc thậm chí loại bỏ hoàn toàn) một số đặc trưng khỏi mô hình, dẫn đến việc tự động chọn lọc biến. Ngược lại, giá trị λ nhỏ làm giảm ảnh hưởng của mức phạt, giữ lại nhiều đặc trưng hơn trong mô hình). Nhìn trên phương diện bias–variance, λ đóng vai trò cân bằng giữa độ chệch và phương sai của mô hình. Cũng theo IBM: *"As λ increases, the bias increases, and the variance decreases, leading to a simpler model with fewer parameters. Conversely, as λ decreases, the variance increases, leading to a more complex model with more parameters. If λ is zero, then one is left with an OLS function – that is, a standard linear regression model without any regularization."*[7] (Dịch: Khi λ tăng, bias tăng và

variance giảm, mô hình đơn giản hơn với ít tham số hơn. Ngược lại, khi λ giảm, variance tăng lên, mô hình phức tạp hơn với nhiều tham số hơn. Nếu $\lambda = 0$ thì hàm mục tiêu trở thành OLS – tức là mô hình hồi quy tuyến tính thông thường không có regularization). Trường hợp $\lambda = 0$ nghĩa là không áp dụng phạt, Lasso lúc này tương đương mô hình hồi quy thường và sẽ không có tác dụng chống overfitting; ngược lại, λ quá lớn sẽ phạt mạnh đến mức hầu hết các hệ số bị triệt tiêu về 0, mô hình khi đó có thể bị underfitting (thiếu độ linh hoạt). Do đó, việc lựa chọn λ tối ưu là rất quan trọng để mô hình đạt hiệu năng cao nhất.

Một ưu điểm nổi bật của Lasso là khả năng chọn lọc đặc trưng tự động nhờ vào chuẩn hóa $L1$. Khoản phạt $L1$ thúc đẩy nghiệm *thưa* (sparse solution), nhiều hệ số hồi quy có thể bị đẩy về đúng bằng 0. Điều này có nghĩa là mô hình Lasso sẽ loại bỏ hẳn những biến không quan trọng, chỉ giữ lại những biến thật sự có đóng góp lớn. Nói cách khác, Lasso vừa giảm overfitting vừa đơn giản hóa mô hình bằng cách bỏ qua các đặc trưng dư thừa. Như IBM mô tả: *"Some variables will shrink exactly to zero, leaving the model with a subset of the most important variables to make predictions."*[7] (Dịch: Một số biến sẽ được kéo về đúng 0, khiến mô hình chỉ còn một tập hợp các biến quan trọng nhất để dự đoán). Nhờ đó, Lasso đặc biệt hữu ích khi xử lý dữ liệu có số lượng đặc trưng rất lớn hoặc có nhiều biến ít liên quan – mô hình sẽ tự động bỏ qua những biến ít liên quan, giảm nguy cơ overfitting và cải thiện tính diễn giải (model interpretability) do mô hình trở nên gọn nhẹ hơn.

3.3.2 Triển khai thực nghiệm

Trong thực nghiệm, chúng tôi xây dựng một pipeline gồm hai bước: (1) Chuẩn hóa dữ liệu bằng `StandardScaler` và (2) Hồi quy Lasso (scikit-learn). Việc chuẩn hóa thang đo các đặc trưng là cần thiết trước khi áp dụng Lasso, nhằm đảm bảo các hệ số bị phạt công bằng giữa các đặc trưng. Một blog khoa học dữ liệu nhấn mạnh: *"It is crucial to scale (e.g. StandardScaler) input features because regression models are sensitive to them."*[8] (Dịch: Việc chuẩn hóa các đặc trưng đầu vào (ví dụ dùng StandardScaler) là cực kỳ quan trọng vì các mô hình hồi quy rất nhạy cảm với đặc trưng có đơn vị hay độ lớn khác nhau). Nếu không chuẩn hóa, đặc trưng có độ lớn lớn sẽ bị phạt nặng hơn đặc trưng nhỏ, dẫn đến mức phạt $L1$ không đồng đều và ảnh hưởng xấu đến kết quả hồi quy Lasso. Do đó, toàn bộ features được chuẩn hóa về trung bình 0 và phương sai 1 trước khi huấn luyện mô hình.

Tiếp theo, để tìm giá trị điều chuẩn tối ưu cho mô hình Lasso, chúng tôi sử dụng phương pháp tìm kiếm lưới kết hợp cross-validation. Cụ thể, chúng tôi thực hiện `GridSearchCV` (5-fold cross-

validation, scoring theo R^2) trên tham số α của Lasso (tương ứng với λ) trong khoảng logarithmic từ 10^{-4} đến 10^3. Việc tìm kiếm trên không gian log-space giúp thử nhiều cấp độ regularization, từ rất nhẹ đến rất mạnh. Kết quả cho thấy $\alpha \approx 78.48$ là giá trị tối ưu cho mô hình (đạt R^2 cao nhất trên tập validation). Với α này, mô hình Lasso giữ lại được độ đơn giản cần thiết đồng thời vẫn giải thích tốt phương sai của dữ liệu. Mô hình cuối cùng được huấn luyện với `alpha=78.48`, `random_state=42` (để kết quả tái lập) và `max_iter=10000`. Việc tăng `max_iter` lên 10000 vòng lặp nhằm đảm bảo thuật toán Lasso (coordinate descent) hội tụ, nhất là khi α khá lớn.

3.4 Mô hình Elastic Net

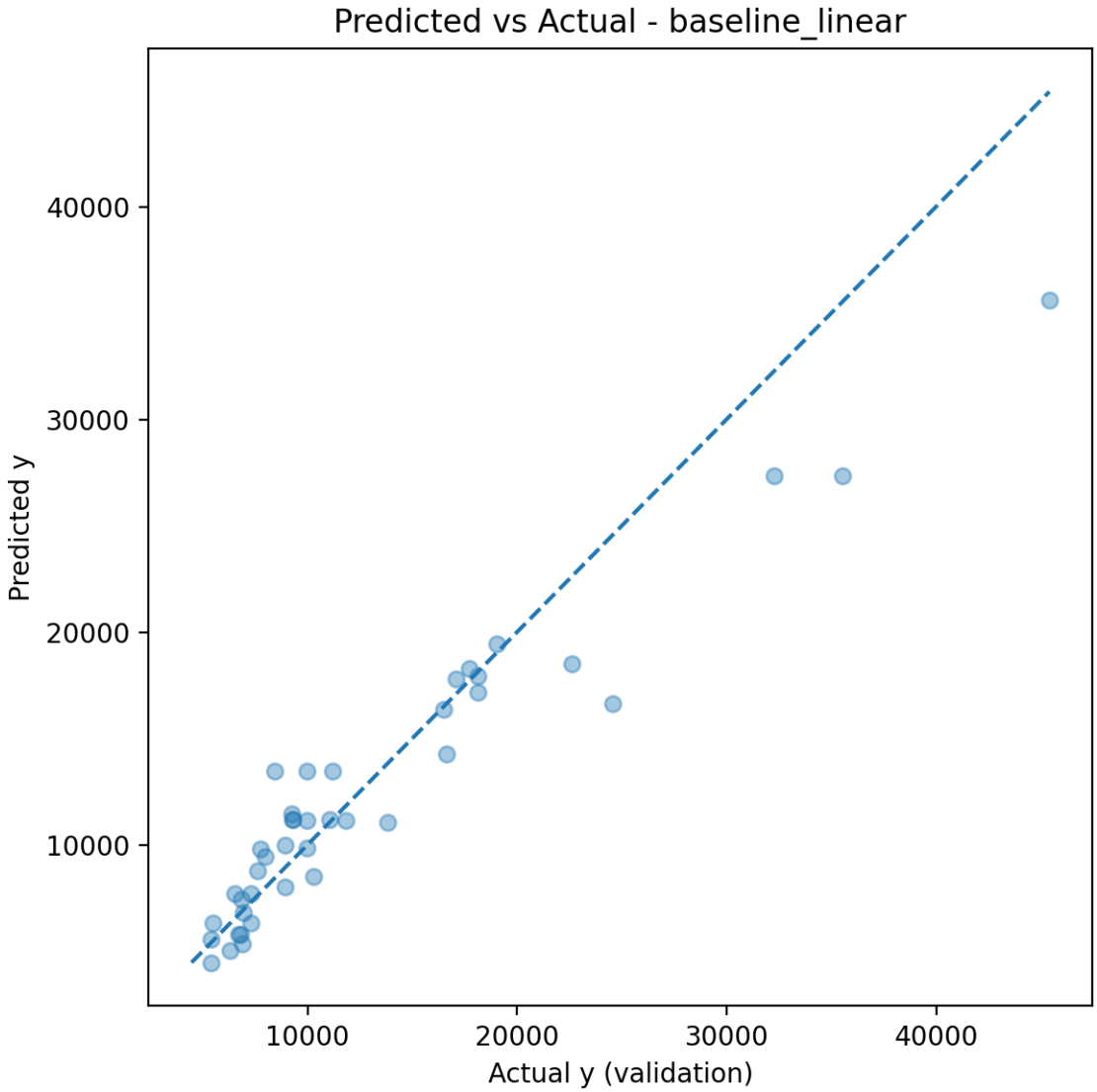
Theo [9], khi kết hợp cả hai dạng regularization l_1 và l_2, ta thu được mô hình Elastic Net Regression. Lúc đó, **hàm loss của Elastic Net** sẽ có dạng:

$$J(w) = \frac{1}{2}\|y - Xw\|_2^2 + \lambda_1\|w\|_1 + \lambda_2\|w\|_2^2$$

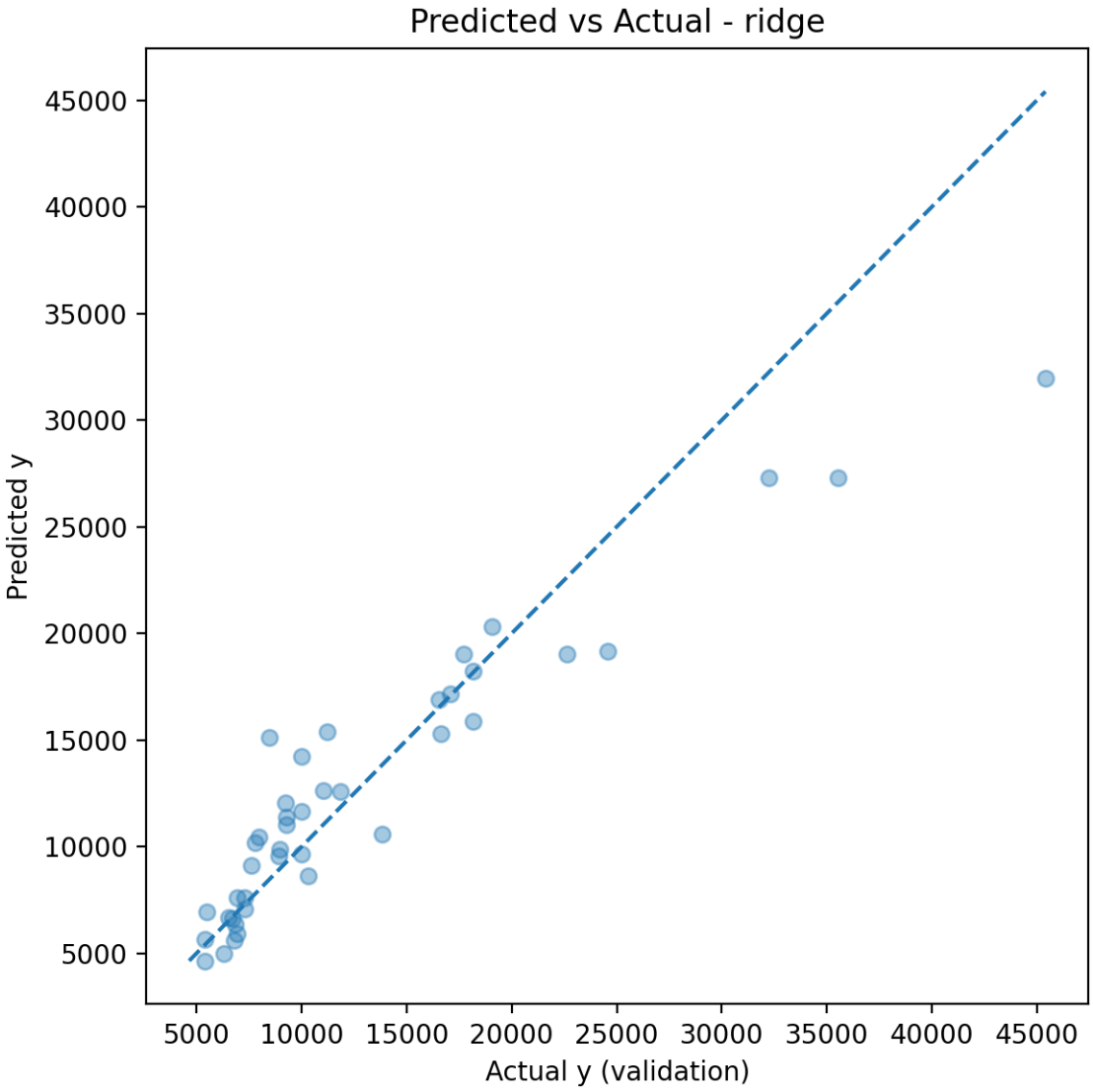
trong đó, λ_1 và λ_2 lần lượt là các hệ số điều chuẩn tương ứng với l_1 và l_2 regularization, giúp cân bằng giữa khả năng chọn lọc đặc trưng và việc giảm độ phức tạp của mô hình. Nhờ đó, Elastic Net đặc biệt hữu ích khi dữ liệu vừa chứa nhiều đặc trưng không quan trọng, vừa tồn tại hiện tượng đa cộng tuyến (multicollinearity) giữa các biến.

4 Đánh giá và so sánh kết quả

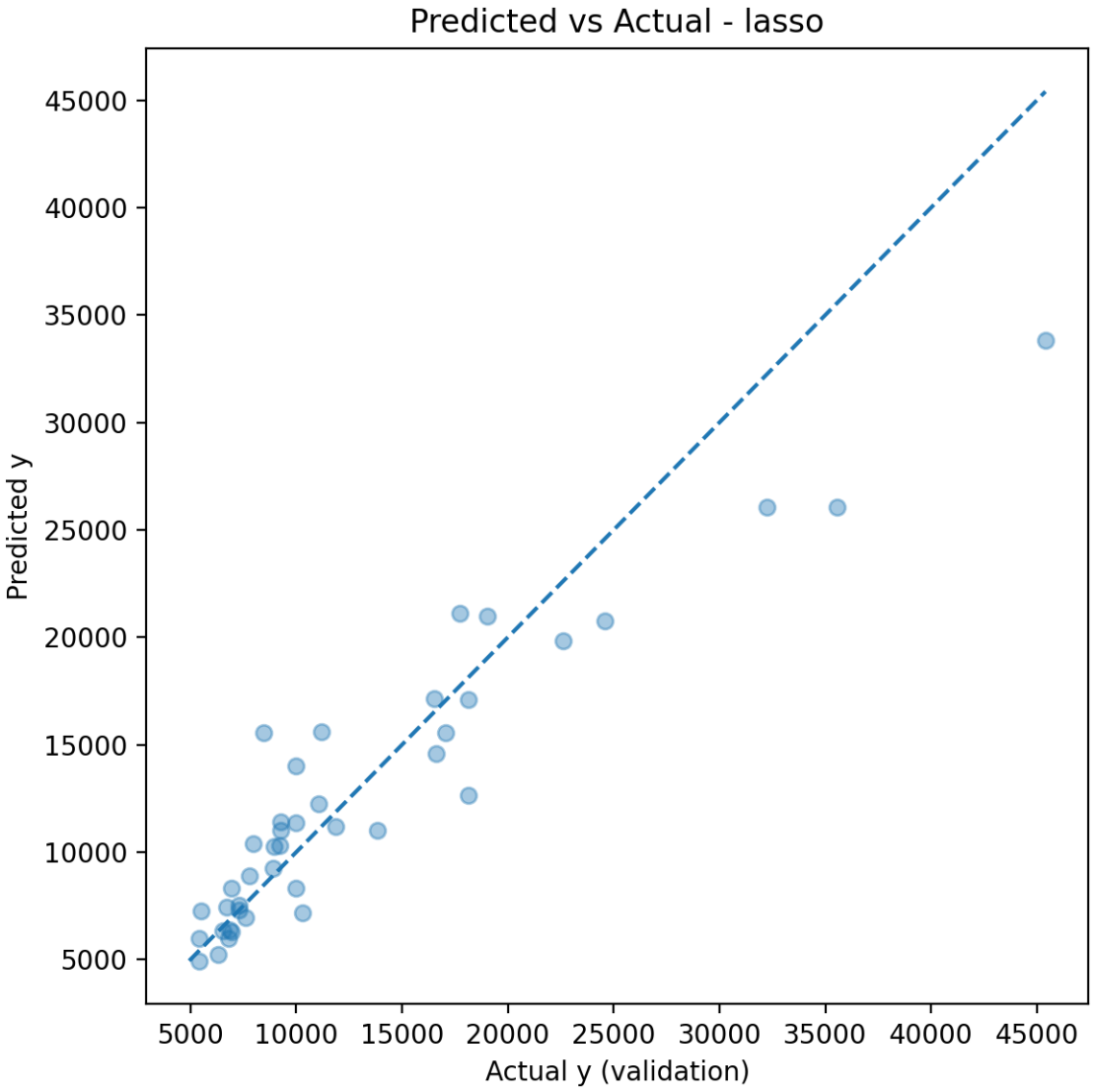
4.1 Kết quả dự đoán của 4 mô hình trên tập validation



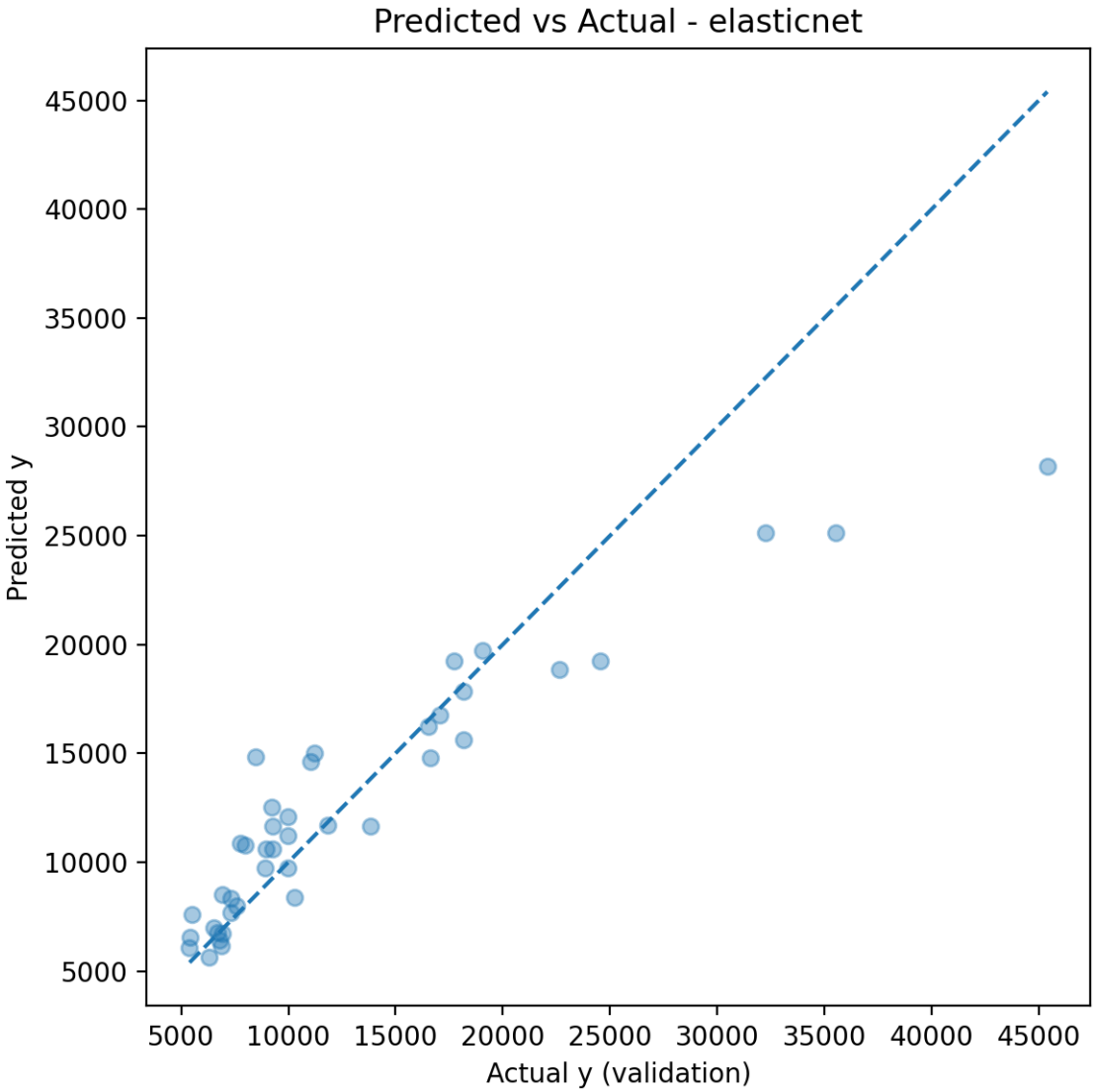
Hình 3: So sánh giá trị dự đoán và giá trị thực tế – Mô hình Linear Regression



Hình 4: So sánh giá trị dự đoán và giá trị thực tế – Mô hình Ridge



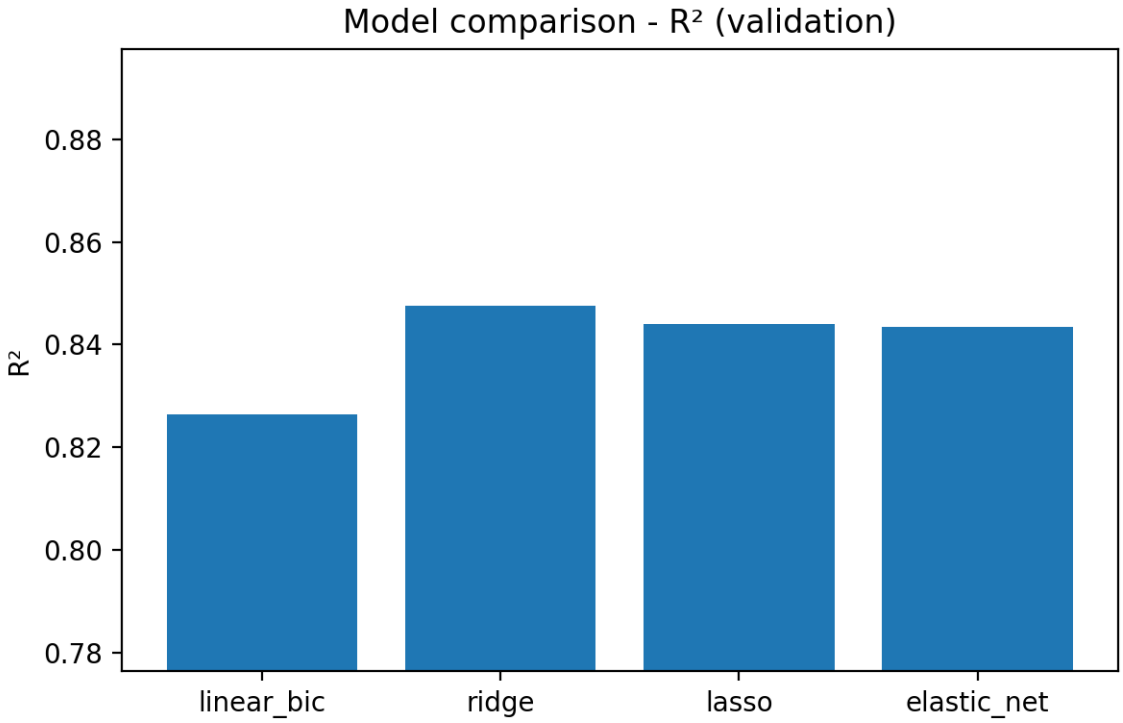
Hình 5: So sánh giá trị dự đoán và giá trị thực tế – Mô hình Lasso



Hình 6: So sánh giá trị dự đoán và giá trị thực tế – Mô hình Elastic Net

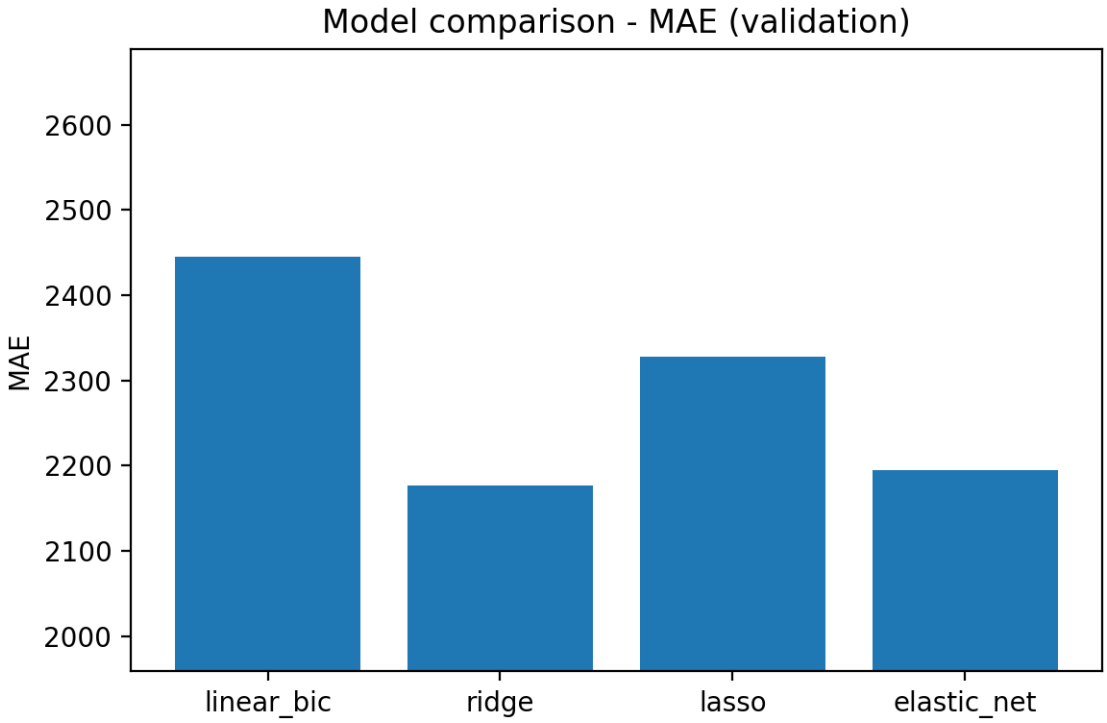
Đầu tiên, ta có thể thấy 4 mô hình đều đưa ra dự đoán giá khá là tốt ở khoảng giá thấp và có xu hướng dự đoán giá thấp hơn giá thực tế với các xe ở giá cao.

4.2 So sánh giữa các mô hình với nhau



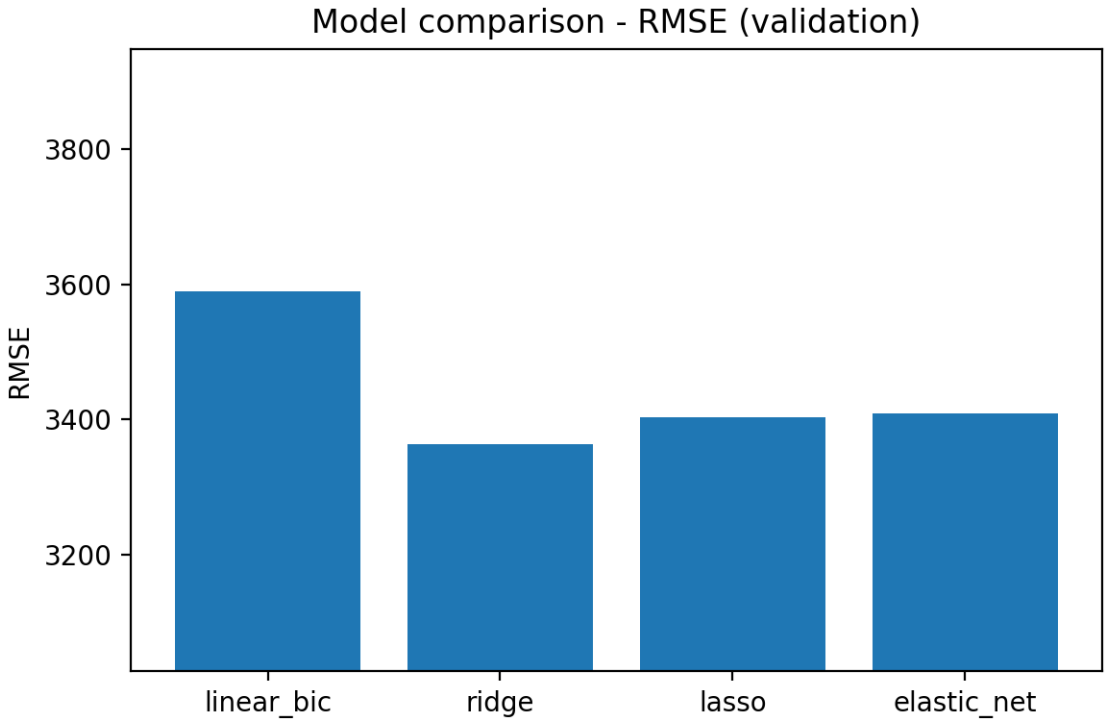
Hình 7: So sánh giá trị R^2 giữa các mô hình

Mô hình Ridge cho ra kết quả tốt nhất với $R^2 = 0.85$, theo sau là Lasso và ElasticNet không chênh lệch nhiều lắm với $R^2 = 0.84$ và Linear Regression sử dụng BIC là tệ nhất với $R^2 = 0.83$.



Hình 8: So sánh giá trị MAE giữa các mô hình

Tương tự như R^2, MAE thể hiện một bức tranh gần tương tự với Ridge cũng là mô hình cho kết quả tốt nhất với MAE nhỏ nhất chỉ gần 2200, tương đương với ElasticNet, trong khi Lasso và Linear Regression sử dụng BIC có sai số cao hơn.



Hình 9: So sánh giá trị RMSE giữa các mô hình

RMSE cao hơn ở tất cả các mô hình cho thấy các ngoại lệ bị dự đoán sai nhiều hơn. Trong đó, Ridge cũng là mô hình có dự đoán tốt nhất. Ridge perform tốt nhất ở cả 3 chỉ số, tuy nhiên performance có thể nói là tương đương giữa Ridge, Lasso, và ElasticNet khi cả 3 chỉ lệch nhau một khoảng rất nhỏ so với chênh lệch giữa Ridge và Linear Regression sử dụng BIC (mô hình tệ nhất). => Việc chỉ chọn một số attribute theo BIC là không đủ để làm tăng hiệu suất mô hình so với các phương pháp thêm norm của w vào hàm loss. Nếu chỉ chọn một mô hình thì Ridge là mô hình perform tốt nhất trên tập validation này. Ngoài ra Lasso và ElasticNet cũng có thể được cân nhắc khi performance chênh lệch là không đáng kể.

4.3 Chạy thực tế trên tập test

Khi chạy lại trên tập test, mô hình Lasso thể hiện tốt nhất với MAE = 2308 và $R^2 = 0.841$, cho thấy tính ổn định và khả năng tránh overfitting tốt hơn trên dữ liệu chưa thấy. Ridge cũng duy trì kết quả khá tốt, Elastic net có hiệu năng trung bình, trong khi Linear Regression sử dụng BIC tiếp tục cho kết quả thấp nhất với $R^2 = 0.7$.

Mô hình	Split	MAE	RMSE	R^2
linear_bic	val	2444.636397649265	3589.5463960434304	0.8263884998258851
linear_bic	test	2783.1648824842	4882.289746350901	0.698054990083924
ridge	val	2176.1437858332592	3363.04433725391	0.8476071702262462
ridge	test	2740.724017729538	3932.558881631839	0.8041014482771742
lasso	val	2327.5662605452317	3403.213561521712	0.8439449762866457
lasso	test	2307.516077153252	3547.5767977943	0.8405794258258836
elastic_net	val	2194.2041057989272	3407.9831095515196	0.8435072526540173
elastic_net	test	2785.465939343897	4010.4200760797594	0.7962674182615903

Bảng 2: Metric của các mô hình trên tập validation và test

Do đó, Ridge là mô hình chạy tốt nhất ở phần validation và Lasso là mô hình chạy tốt nhất ở phần test.

5 Kết luận và phân tích kết quả

Một trong những phát hiện cốt lõi là đối với tập dữ liệu cụ thể này, phương pháp lựa chọn thuộc tính rõ ràng và có chiến lược (Forward Selection với BIC) đã mang lại nột mô hình dự đoán mạnh mẽ hơn so với các kỹ thuật chính quy hoá (*regularization*). Điều đó cho thấy việc chủ động loại bỏ thuộc tính không mang thông tin cho phép mô hình tuyến tính đơn giản tìm thấy một tín hiệu dự đoán rõ ràng và mạnh mẽ.

Mặc dù R^2 nghiêng về mô hình hồi quy với BIC, các chỉ số độ lỗi lại cho thấy sự đánh đổi. Các mô hình Ridge và Elastic Net đạt được MAE thấp nhất cho thấy dự đoán trung bình của chúng chính xác hơn. Tuy nhiên, mô hình hồi quy vớ BIC lại đảm bảo được RMSE thấp nhất, vì RMSE sẽ tăng rất mạnh nếu như mô hình dự đoán sai lệch lớn, cho thấy mô hình này dự đoán outliers tốt hơn và tránh được tình trạng dự đoán lệch giá (quá thấp) so với xe siêu đắt tiền, một vấn đề mà các mô hình khác gặp phải.

Riêng về Lasso, có lẽ ở ngữ cảnh của tập dữ liệu này thì hình phạt của L1 regularization đã quá mạnh, khiến các chỉ số của mô hình này luôn là tệ nhất và chênh lệch rất lớn so với các mô hình khác.

Tóm lại, mô hình Hồi quy tuyến tính cơ bản sử dụng thuật toán Forward Selection với chỉ số BIC là mô hình tốt nhất, được chọn để kiểm tra và là mô hình chính của phần mềm ứng dụng Web. Trên tập test, mô hình có sự sụt giảm nhẹ về hiệu năng so với tập validation, nhưng điều này là hoàn toàn hợp lí vì nó chứng tỏ rằng mô hình đã không bị quá khớp (overfiiting).

6 Mô tả ứng dụng

6.1 Phân tích và trực quan hoá mức độ ảnh hưởng của thuộc tính

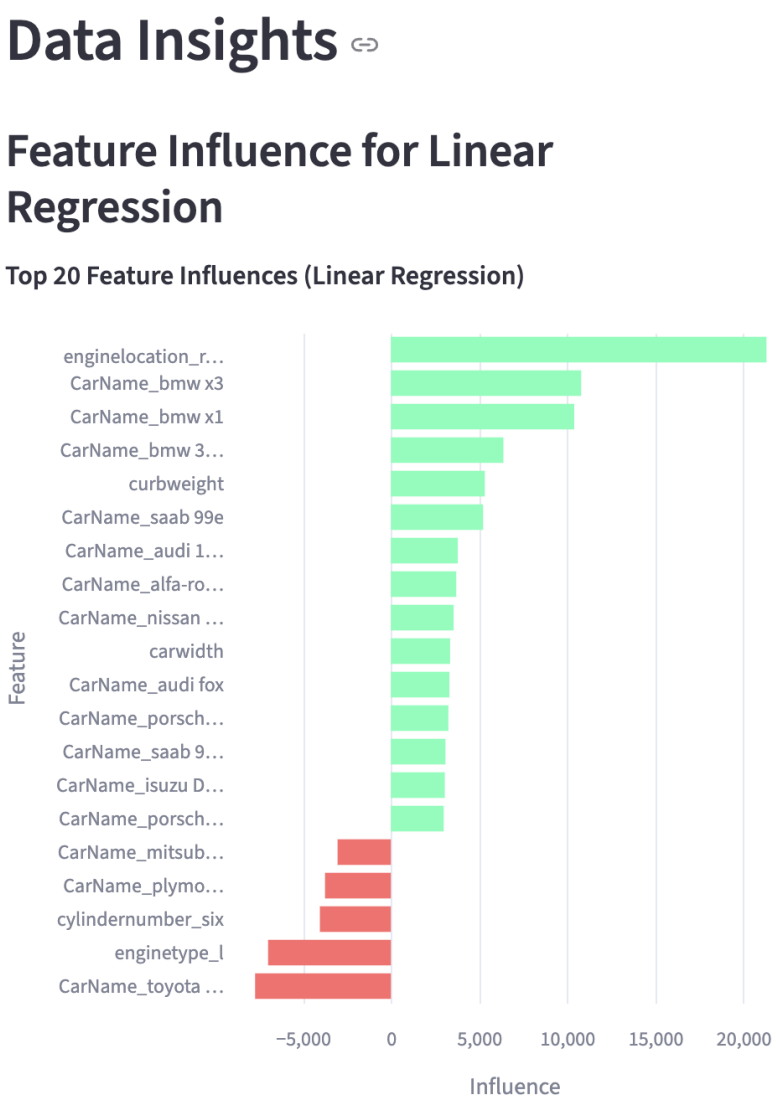
6.1.1 Mục tiêu

Mục tiêu chính của hệ thống là cho phép người dùng trải nghiệm trực tiếp tính năng dự đoán của mô hình, đồng thời giúp họ hiểu được phần nào cách thức mà mô hình đưa ra dự đoán. Chính vì thế, giao diện ứng dụng sẽ đưa ra các cơ chế trực quan hoá để làm rõ mối quan hệ giữa các biến đầu vào và kết quả dự đoán giá xe hay phân tích sự ảnh hưởng của các biến lên giá xe sau cùng.

6.1.2 Triển khai

Khi người dùng chọn một mô hình trên giao diện, ứng dụng sẽ tải tệp mô hình `.pkl` tương ứng. Quá trình trích xuất thông tin diễn ra như sau:

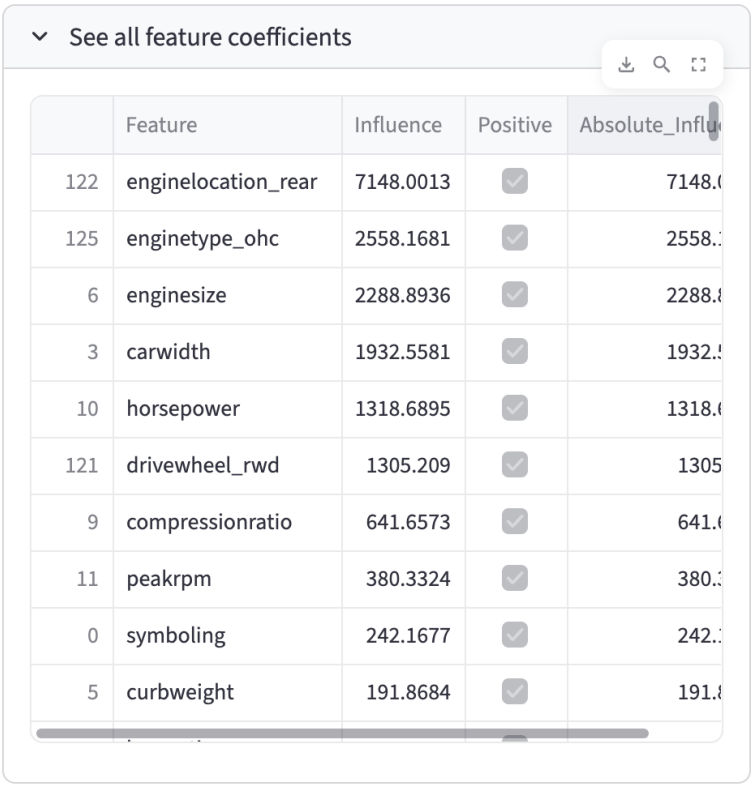
- **Trích xuất hệ số:** Các mô hình hồi quy tuyến tính lưu trữ tầm quan trọng của mỗi thuộc tính dươi dạng mảng các hệ số (`coef_`). Trích xuất mảng này từ pipeline của mỗi mô hình.
- **Trích xuất tên thuộc tính:** Ánh xạ các hệ số vừa trích xuất xong về lại tên ban đầu của nó bằng cách truy cập vào bước tiền xử lý trong pipeline:
  - Đối với các mô hình chính quy hoá (Ridge, Lasso, Elastic Net), tên thuộc tính được trích xuất từ thuộc tính `feature_columns_` của bước `"preprocess"` (lớp `TabularPreprocessor`).
  - Đối với mô hình hồi quy tuyến tính sử dụng BIC, tên đặc trưng được trích xuất từ phương thức `get_feature_names_out()` của bước chọn lọc thuộc tinh, đảm bảo chỉ có các thuộc tính được lựa chọn mới hiển thị.
- **Trực quan hoá:** Sau khi có được danh sách các tên thuộc tính ánh xạ với mảng chỉ số `coef_`, kết hợp chúng thành một cấu trúc dữ liệu `DataFrame` duy nhất, sau đó dùng `Altair` để vẽ biểu đồ. Biểu đồ được sắp xếp dựa trên giá trị tuyệt đối của hệ số và chỉ hiển thị 20 đặc trưng có ảnh hưởng lớn nhất.



Hình 10: Biểu đồ ví dụ thể hiện độ ảnh hưởng của thuộc tính lên kết quả dự đoán sau cùng được trích xuất từ giao diện ứng dụng web.

Trong biểu đồ trên, các thuộc tính có đường biểu diễn chỉ số tầm ảnh hưởng màu xanh tức là chúng có ảnh hưởng dương hay đồng biến với giá xe (giá xe sẽ tăng nếu các thuộc tính này tăng) và các thuộc tính có chỉ số tầm ảnh hưởng màu đỏ nghĩa là chúng có ảnh hưởng âm hay nghịch biến đến giá xe sau cùng.

Ngoài ra, giao diện ứng dụng web cũng cung cấp một cách xem khác để có thể thấy được chi tiết toàn bộ thuộc tính và chỉ số tầm ảnh hưởng của chúng, được cài đặt để hiển thị khi người dùng ấn nút "See all feature coefficients".



Hình 11: Bảng chi tiết ví dụ thể hiện độ ảnh hưởng của thuộc tính lên kết quả dự đoán sau cùng được trích xuất từ giao diện ứng dụng web.

Như hình trên, giao diện bảng này cung cấp một nút tải xuống để cho người dùng có thể xuất bảng ra file `.csv` để dễ dàng thực hiện các nghiệp vụ tính toán khác.

7 Tài liệu tham khảo

[1] Sanford Weisberg. "Applied Linear Regression". In: 4th. 2014. Chap. Chapter 10: Variable Selection, pp. 238–239. ISBN: 978-1-118-38608-8.

[2] Sanford Weisberg. "Applied Linear Regression". In: 4th. 2014. Chap. Chapter 10: Variable Selection, p. 240. ISBN: 978-1-118-38608-8.

[3] Talhahascelik. *Automated Stepwise Backward and Forward Selection*. en. URL: https://github.com/talhahascelik/python_stepwiseSelection/blob/master/stepwiseSelection.py.

[4] Tiep Vu. *Bài 15: Overfitting*. Mar. 2017. URL: https://machinelearningcoban.com/2017/03/04/overfitting.

[5] *2.2.2. Hồi qui Ridge — Deep AI KhanhBlog*. en. URL: https://phamdinhkhanh.github.io/deepai-book/ch%5C_ml/RidgedRegression.html.

[6] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[7] *What is lasso regression?* IBM. URL: https://www.ibm.com/think/topics/lasso-regression (visited on 11/11/2025).

[8] Arjun Mota. *Lasso Regression*. URL: https://arjun-mota.github.io/posts/lasso-regression/ (visited on 11/11/2025).

[9] Machine Learning Cơ Bản. *Bài 15: Overfitting*. vi. https://machinelearningcoban.com/2017/03/04/overfitting/. Truy cập ngày 11 tháng 7 năm 2025. Mar. 2017.