Dự đoán giá laptop

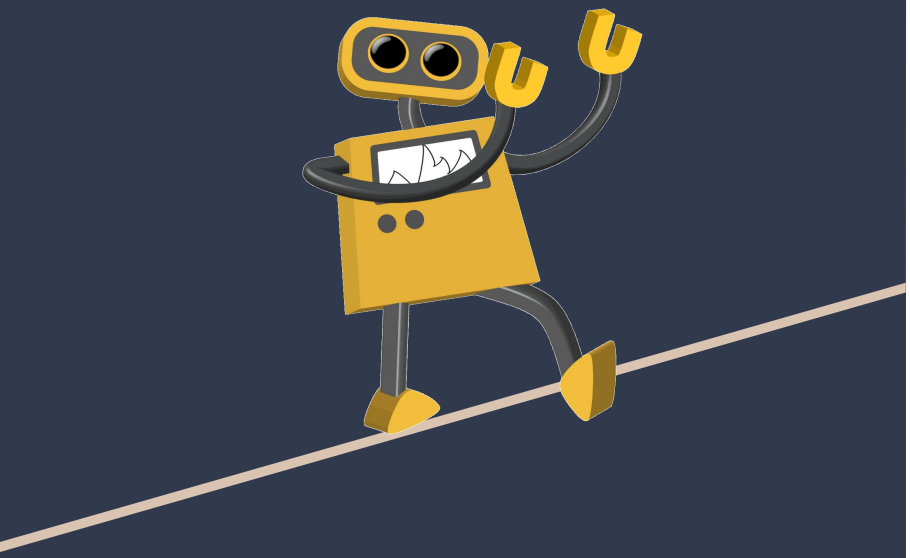
Đồ án cuối kỳ

Khoa học dữ liệu

TRUONG DAI HOC KHOA HOC TU NHIEN
KHTN
TP. HO CHI MINH

★ Nguyễn Thanh Tuấn 1612744
★ Hoàng Xuân Trường 1612899

Nội dung



1. Giới thiệu đề tài
2. Crawl data
3. Preprocessing
4. Model
5. Test và đánh giá model

Giới thiệu đề tài

Dự đoán giá laptop, tablet từ các thông tin cấu thành (hãng, CPU, ram, gpu,...)

- **Input**: các thông tin của máy tính: hãng, CPU, ram, ...
- **Output**: Giá sản phẩm.

Crawl data

Từ 2 nguồn chính:

- **Amazon** ~ 5600 items: nhiều nhưng không chi tiết (41 thuộc tính).
- **Best buy** ~ 1000 items: ít nhưng rất chi tiết (136 thuộc tính).

Tính hợp lệ của dữ liệu: Dữ liệu **được phép crawl** (Đã ***kiểm tra thủ công*** trên một số link cụ thể).

Crawl data

- Amazon Data:
  - Nhiều dữ liệu - gần 6000 items.
  - Số thuộc tính ít: 41 thuộc tính.
  - Thông tin cần thiết rút trích từ dữ liệu có nhiều thông tin thiếu, giá trị null nhiều.
  - Giá ảo khá nhiều. Ví dụ có một số giá 1$ hoặc 2$ (trường hợp này dễ nhận dạng, tuy nhiên các trường hợp khác như 300$ nhưng vẫn chưa chắc đó là giá bán thực tế).

Crawl data

- BestBuy Data:
  - Số lượng dữ liệu ít - hơn 1000 items.
  - Dữ liệu rất nhiều thuộc tính: 168 thuộc tính.
  - Các thuộc tính tuy nhiều, nhưng không rối, do có nhiều thuộc tính là triển khai chi tiết của thuộc tính khác.
  - Thông tin lấy từ dữ liệu đầy đủ hơn: Ít bị thiếu.
  - Dữ liệu khá chuẩn, không có dữ liệu nhiễu: Giá đồng bộ, không có giá ảo.

Crawl data

- Chọn dữ liệu cho việc xây dựng mô hình:
  - Gộp dữ liệu:
    - Dữ liệu trùng: Cùng sản phẩm nhưng giá khác -> Không biết lấy giá nào hợp lý hơn.
    - Thuộc tính không đồng bộ: Amazon có thuộc tính này nhưng BestBuy không có -> Khó ghép, nếu bỏ thì không còn nhiều thuộc tính có giá trị.

➔ Không gộp chung, chỉ dùng 1 bộ dữ liệu duy nhất.

➔ **Sử dụng dữ liệu BestBuy** để xây dựng mô hình: Do BestBuy dữ liệu ít thuộc tính null (tính trên tập dữ liệu mà các thuộc tính cần thiết được rút trích từ dữ liệu thô) và giá của BestBuy không có giá trị ảo như Amazon.

Preprocessing

- Chú ý: Giai đoạn này có thể gộp chung vào 1 class và đưa chung vào pipeline khi fit mô hình. Giai đoạn này chủ yếu là rút trích đặc trưng và clean data, không điền các giá trị null/nan trong giai đoạn này (điền null/nan thực hiện trong pipeline - fit trong giai đoạn train/val/test trên model).
- Các công việc cần làm:
  - Chọn/loại bỏ các thuộc tính.
  - Rút trích các dữ liệu cần thiết từ các thuộc tính (Ví dụ lấy giá từ string - “Your price for this item is 300$”-> lấy giá trị 300, ….).
  - Lựa chọn các giá trị điền cho các giá trị rỗng trong các cột (không điền luôn trong giai đoạn này).

Preprocessing

- Chọn, loại bỏ các thuộc tính:
  - Điều kiện loại bỏ các thuộc tính:
    - Các thuộc tính trùng (hoặc triển khai của nhau), chỉ lấy 1 thuộc tính đặc trưng, các cột còn lại drop. Ví dụ:
      - ’Key Specs__Total Storage Capacity’ Và ‘Storage__Total Storage Capacity’ thể hiện cùng một đặc trưng là tổng dung lượng lưu trữ.
      - ‘Processor__Processor Model’ đã bao gồm các thuộc tính ‘Processor__Processor Brand’, ‘Processor__Processor Speed (Base)’,... nên chỉ lấy ‘Processor__Processor Model’ là được.
    - Các thuộc tính: Null nhiều, có nhiều giá trị rỗng.
    - Các thuộc tính không ảnh hưởng đến giá sản phẩm nhiều (đưa vào chỉ làm phức tạp mô hình dễ overfit). Ví dụ: thuộc tính Bluetooth_version,.....
    - Các thuộc tính có rất nhiều giá trị khác nhau. Ví dụ: Number_series (hầu như mỗi máy có số series khác nhau).

Preprocessing

- Rút trích thông tin đặc trưng:
  - Rút trích đặc trưng cần thiết từ giá trị. Ví dụ:
    - Giá: “Your price for this item is 300$” -> rút trích ra giá trị 300.
    - Kích thước màn hình: “11.6 inches” -> Lấy giá trị 11.6.
  - Rút trích và thêm cột vào dữ liệu khi cần thiết: Ví dụ:
    - Độ phân giải: “1366 x 768” -> Tách thành 2 cột mới có giá trị lần lượt là 1366 và 768.
  - Đồng bộ hóa các giá trị về cùng format:
    - Ví dụ: “Samsung”/”SAMSUNG”/”samsung” -> đưa về chữ thường hết, ta được “samsung”.
    - …..

Preprocessing

- Dữ liệu cuối cùng:
  - Số thuộc tính: 26 (25 thuộc tính train, 1 thuộc tính nhãn).
    - Số thuộc tính dạng numeric: 13.
    - Số thuộc tính dạng category: 13.
  - Tỉ lệ dữ liệu thiếu:
    - RAM_type, Feature__Keyboard_Touch_Screen: Thiếu gần 35%.
    - Feature__Mac_Features: Thiếu gần 93% -> Không bỏ được, vì tính năng này ảnh hưởng nhiều đến giá sản phẩm.
    - Các cột còn lại: Thiếu <10%.

Preprocessing

- Lựa chọn giá trị sẽ điền cho các giá trị thiếu (trong pipeline):
  - Các thuộc tính số: Thuộc tính ‘Price’ không có null.
    - Các giá trị điền mean (trung bình): Vì các thông số này null điền mean sẽ lấy được tính tổng quan của thuộc tính.
      - 'Key_Specs__Screen_Size'
      - 'Key_Specs__System_Memory'
      - 'RAM_speed'
      - 'Screen_resolution1',
      - 'Screen_resolution2'
      - 'Port_Number_USB_Ports'
      - 'Dimension__Product_Depth'
      - 'Dimension__Product_Height'
      - 'Dimension__Product_Weight',
      - 'Dimension__Product_Width'
    - Các giá trị 0.0: Vì các thuộc tính này null/nan là không có chứ không phải thiếu (ví dụ: nếu máy tính không SSD có sẽ điền 0.0 vào thuộc tính *‘Storage__Solid_State_Drive_Capacity*).
      - 'Storage__eMMC_Capacity' - Có thể điền ngay lúc tiền xử lý, vì đây là giá trị cứng.
      - 'Storage__Solid_State_Drive_Capacity' - Có thể điền ngay lúc tiền xử lý, vì đây là giá trị cứng.

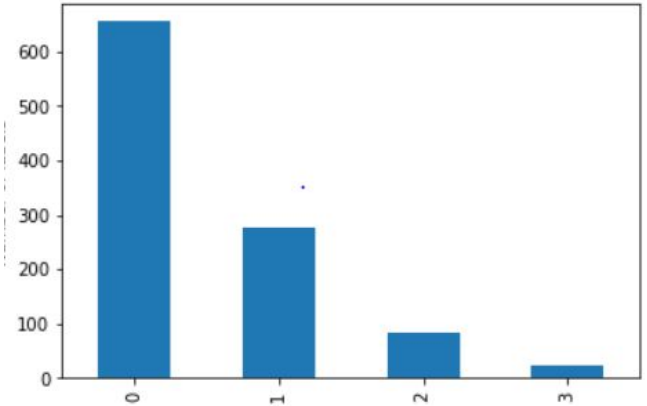
Preprocessing

- Lựa chọn giá trị sẽ điền cho các giá trị thiếu (trong pipeline):
  - Các thuộc category:
    - Điền các giá trị null bằng mode (giá trị có tầng suất xuất hiện nhiều nhất trong cột).
      - ‘Key_Specs__Touch_Screen'
      - 'Key_Specs__Storage_Type',
      - 'RAM_type'
      - 'Processor'
      - ‘Graphics__Graphics'
      - 'Key_Specs__Operating_System'
      - 'Key_Specs__Battery_Type'
      - 'General__Color_Category'
      - 'General__Brand'
      - 'Feature__Keyboard_Touch_Screen'
      - 'Feature__Backlit_Keyboard'
      - 'Display__Display_Type'
    - Các giá trị null/nan trong các cột này được điền bằng ‘**NotHave**’: Vì máy tính không có chứ không phải thiếu giá trị.
      - 'Feature__Mac_Features'

Preprocessing

- Biểu đồ kiểm tra sự phân bố giá sản phẩm trong các khoảng giá trị (trên toàn tập dữ liệu).
  - ***Trục tung***: Số lượng sản phẩm
  - ***Trục hoành***: ***Giá*** các sản phẩm trong data ta được ***chia thành 4 khoảng*** bằng nhau. ***Đánh theo mức độ từ 0 đến 3***.
- Đánh giá: Giá càng cao thì sản phẩm càng ít.

➔ Ít có sản phẩm có giá cao (các sản phẩm cao cấp).



Phân chia dữ liệu

- Dữ liệu được tách ra thành 3 phần chính:
  - 10%: Dùng để test (đánh giá cho mô hình chọn cuối cùng).
  - 90%: Dùng cho quá trình train. Tập này được chia thành 2 tập nhỏ trong quá trình train, gồm:
    - 15%: Dùng làm tập validation (đánh giá mô hình trong quá trình train).
    - 95%: Dùng làm tập train (tập dùng để huấn luyện mô hình).

Model

- Pipeline: Dữ liệu được đưa vào pipeline với các tác vụ:
  - dropAndAddColums: Thực hiện việc thêm hoặc xóa các cột trong dữ liệu (trường hợp này không dùng tới).
  - FilllNa: Điền các giá trị rỗng trong data (giá trị fill nói trong phần Preprocessing).
  - Chuẩn hóa.
  - Train với model.

➔ Việc tạo pipeline có tác dụng giúp dễ quả lý luồng công việc. Đồng thời giúp việc điền các giá trị rỗng (null/nan) vào tập validation/test dựa trên các giá trị lấy từ tập train (không phải lấy từ chính tập validation hay tập test. Vì 2 tập val/test khá nhỏ, đồng thời việc train thực hiện trên dữ liệu train - tức các giá trị tiêu chuẩn phải lấy từ tập train, nếu lấy từ tập val/test thì các giá trị điền này có thể sẽ khác nhau giữa tập train-val-test) -> Đảm bảo đồng bộ dữ liệu.

Model

- Nhóm thử nghiệm train dữ liệu qua 2 thuật toán chính:
  - Linear Regression (Để kiểm tra tính chuẩn của dữ liệu - sự phụ thuộc của các thuộc tính dữ liệu với các mô hình -> Dữ liệu có thực sự tốt không, nhóm đã thực hiện train bộ dữ liệu trên một số thuật toán cải tiến của LinearRegression là Lasso (thêm L1), Ridge (Thêm L1) và ElasticNet (Thêm L1 và L2) để đánh giá độ ổn định của data khi đưa vào các mô hình training).
  - Random forest.

Model

- LinearRegression:
  - Score:
    - 91.7% trên tập train.
    - 85.9% trên tập validation.
  - Nhận xét: Do mô hình ban đầu còn khá đơn giản, nên độ chính xác bên trên vẫn chưa thấy mô hình và dữ liệu có thực sự khớp hay không (chỉ biết score khá cao trên 2 tập -> tuy nhiên vẫn chưa xác định sự phụ thuộc thật sự của các thuộc tính với mô hình có ổn hay không).

```
Train score: 0.917766 - Val score: 0.858982
```

Score của LinearRegression trên tập train và validation

=> Thêm các thử nghiệm trên các mô hình tương tự (có thêm L1, L2 - Laso, Ridge, ElasticNet) -> Xem sự phụ thuộc dữ liệu vào các mô hình có tốt không -> Dữ liệu (các thuộc tính) có thật sự tốt để train hay chưa.

Model

- Lasso (L1):

```
alpha: 0.100000 - Train score: 0.925864 - Val score: 0.857608
alpha: 0.300000 - Train score: 0.925760 - Val score: 0.857826
alpha: 0.500000 - Train score: 0.925599 - Val score: 0.858002
alpha: 1.000000 - Train score: 0.925004 - Val score: 0.857760
alpha: 2.000000 - Train score: 0.923415 - Val score: 0.858468
alpha: 5.000000 - Train score: 0.917766 - Val score: 0.858982
```

Score trên tập train và validation khi dùng Lasso

- Ridge (L2):

```
alpha: 0.100000 - Train score: 0.925869 - Val score: 0.855210
alpha: 0.300000 - Train score: 0.925869 - Val score: 0.855241
alpha: 0.500000 - Train score: 0.925863 - Val score: 0.855169
alpha: 1.000000 - Train score: 0.925860 - Val score: 0.855277
alpha: 2.000000 - Train score: 0.925842 - Val score: 0.855450
alpha: 5.000000 - Train score: 0.925759 - Val score: 0.855774
```

Score trên tập train và validation khi dùng Ridge

Model

- ElasticNet:

```
alpha: 0.010000 - Train score: 0.925819 - Val score: 0.856021
alpha: 0.100000 - Train score: 0.923610 - Val score: 0.858918
alpha: 0.200000 - Train score: 0.920281 - Val score: 0.860010
alpha: 0.300000 - Train score: 0.916599 - Val score: 0.859828
alpha: 0.500000 - Train score: 0.908595 - Val score: 0.857177
alpha: 1.000000 - Train score: 0.886886 - Val score: 0.843799
```

Score trên tập train và validation khi dùng ElasticNet

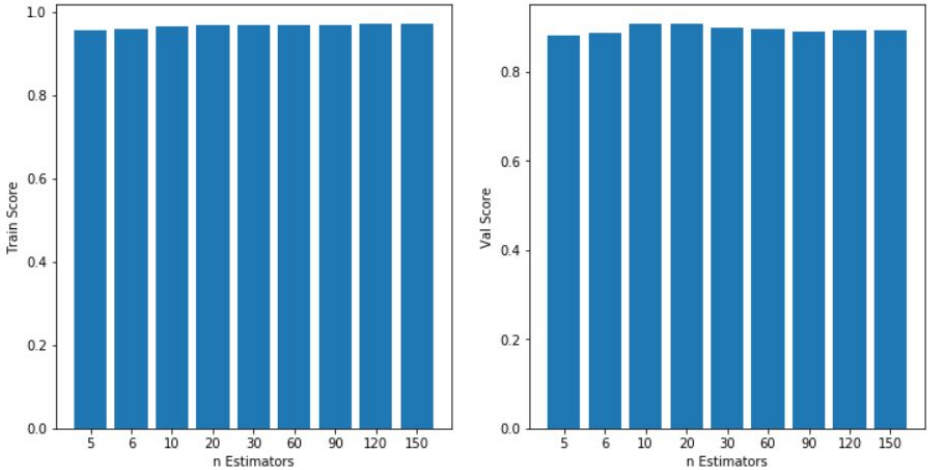
★ Nhận xét: Dù áp dụng Linear, hay thêm các L1, L2 thì score trên tập train và test vẫn khá ổn định (dường như sự chênh lệch không nhiều, chêch lệch chỉ khoảng 1%) => Các dữ liệu ít bị nhiễu, khá đồng bộ với nhau (chuẩn) => Phù hợp để fit với các mô hình huấn luyện (các mô hình tương tự (GradientBoosting - cải tiến tốc độ của Gradient) cũng như mô hình khác).

Model

- Randomforest:
  - Dùng randomforest để giúp hạn chế sự buộc chặt vào dữ liệu khi train (do được hình thành từ nhiều DecisionTree - vote để chọn sự giá trị có độ tin cậy cao nhất -> Không phải chỉ dồn hết vào 1 cây như DecisionTree) - đặc biệt khi dữ liệu train hiện tại khá ít - mà vẫn đảm bảo được độ chính xác ổn định.
  - Dữ liệu hiện tại ít: Nên chọn max_depth (độ sâu tối đa) cũng như n_estimators (số DecisionTree build để vote) không nên quá lớn. Vì nếu, chọn quá lớn có thể sẽ chỉ làm giảm tốc độ học mà kết quả không tốt hơn với trường hợp tham số này nhỏ, đồng thời có thể các DecisionTree sẽ khá giống nhau dẫn đến việc vote (và xây dụng nhiều cây không còn tác dụng) không khác gì với DecisionTree bình thường.

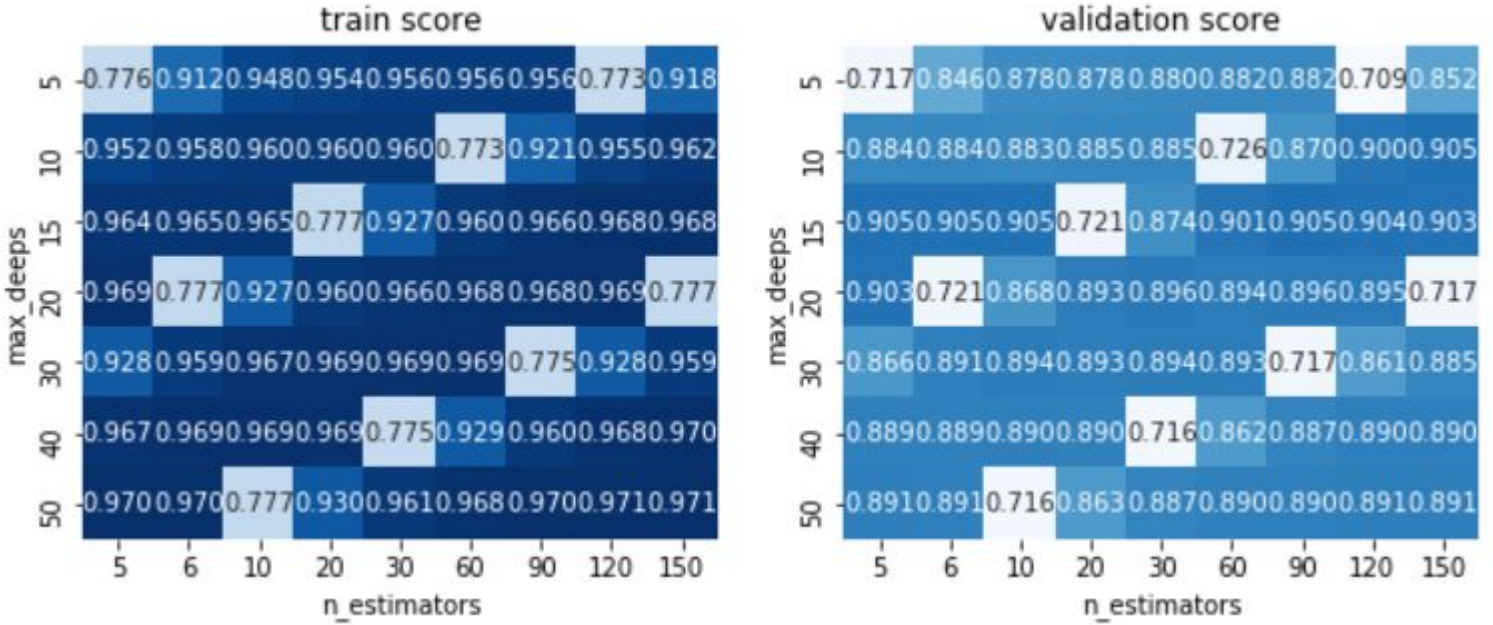
Model

- Randomforest:
  - Đánh giá score của mô hình (trên tập train và validation) chỉ dựa trên n_estimators:



Model

- Randomforest: (2 siêu tham số max_depth và n_estimators)
  - Score trên tập train và tập validation khi áp dụng với randomforest:

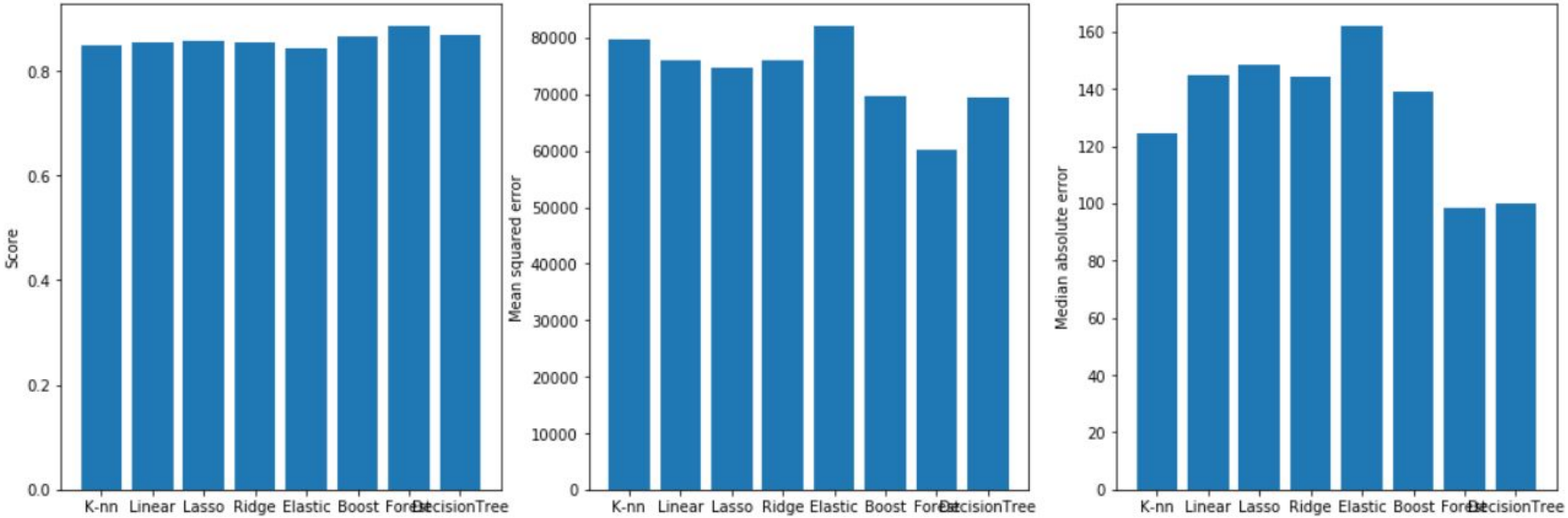


Model

- Randomforest:
  - Đánh giá:
    - Ta thấy một cách chọn 2 tham số max_depth, n_estimators làm cho mô hình có score khá thấp trên tập train và validation.
    - Ngoài ra, các giá trị còn lại có score khá ổn định, tuy nhiên sẽ rơi vào trường hợp giá trị của max_depth hoặc n_estimators quá cao (so với dữ liệu hiện tại) -> Dù 2 tham số này cao nhưng score gần như tương tự nhau.
    - Chọn tham số tốt nhất: Từ các phân tích trên, ta chọn n_estimators=10 và max_depth=15 là 2 giá trị khá hợp lý cho dữ liệu.

Model

- Đánh giá mô hình dựa trên (so sánh) với các mô hình Regression tương tự:
  - Biểu đồ so sánh nhanh về score (trên tập train và validation):



Model

- Đánh giá mô hình dựa trên (so sánh) với các mô hình Regression tương tự - đánh giá dựa trên validation_score:
  - Nhận xét: Đánh giá các mô hình theo các score metrics khác nhau, ta thấy Randomforest có khả năng fit mô hình phù hợp hơn các mô hình khác.
    - Theo độ lỗi: R^2, ta thấy sự chêch lệch không cao giữa RandomForest và các mô hình khác.
    - Theo mse (mean square error) mà mae (median absolute error), ta thấy sự khác biệt tổng thể giữa nhãn predict và nhãn đúng của các mô hình khác cao hơn khá nhiều so với RandomForest.

➔ Mô hình lựa chọn cuối cùng: RandomForest: n_estimator=10, max_depth=15.

Test và đánh giá model

- Dữ liệu test: Dữ liệu test được tách từ 10% tổng dữ liệu (Đã được clean - chưa chuẩn hóa, chưa fillNa - việc này được thực hiện trong pipeline khi tính score với model test).
- Dữ liệu test được fit vào mô hình RandomForest (n_estimators=10, max_depth=15).
- Quá trình được đưa vào pipeline (fit model với train data), do đó, các giá trị khi transform thực sự (tính score) thì test_data được chuẩn hóa từ các giá trị lấy từ tập train (giá trị mean, mode, …).

Test và đánh giá model

- Kết quả: ***Score trên tập test*** từ ***82-85%***.
- Hình bên phải là kết quả được so sánh giữa giá thực sự (nhãn đúng) và giá predict (mô hình đề xuất). Các cột:
  - ***Predicted***: Giá sản phẩm được predict từ mô hình.
  - ***True***: Giá sản phẩm thực sự (đúng).
  - ***Difference***: Sự chênh lệch giữa giá đúng và giá predict.
    - > 0: predicted cao hơn true.
    - < 0: predicted thấp hơn true.

	predicted	true	Difference
13	1071.88	1079.99	-8.11
54	705.94	699.99	5.95
63	1242.28	1269.99	-27.71
35	1507.87	1499.99	7.88
53	1096.86	769.99	326.87
16	573.01	499.99	73.02
25	1532.97	1699.99	-167.02
0	287.78	328.99	-41.21
14	583.97	599.99	-16.02
76	1305.52	1399.99	-94.47

Q&A

Cảm ơn thầy và các bạn đã lắng nghe!