Báo cáo kỹ thuật

Dự án: Xây dựng mô hình hồi quy tuyến tính dự đoán giá xe ô tô cũ

Giới thiệu bài toán:

Dự án này sử dụng bộ dữ liệu Car Price Prediction (Kaggle) để xây dựng các mô hình học máy dự đoán giá xe ô tô cũ.

Dữ liệu bao gồm hãng xe, năm xe được bán ra và một số thông tin của chiếc xe và người bán.

Mục tiêu của dự án là xây dựng một số mô hình hồi quy hỗ trợ người mua xe cũ tốt nhất dựa trên những đặc trưng của chiếc xe được bán, giúp người mua và người bán có thể định giá chiếc xe

Phân tích và khám phá dữ liệu:

Bộ dữ liệu Car Price Prediction bao gồm 8128 dòng và 13 cột mô tả các thông tin về chiếc xe như là hãng xe, năm sản xuất, loại nhiên liệu sử dụng, người bán, loại hộp số (tự động hoặc thủ công),... và giá bán

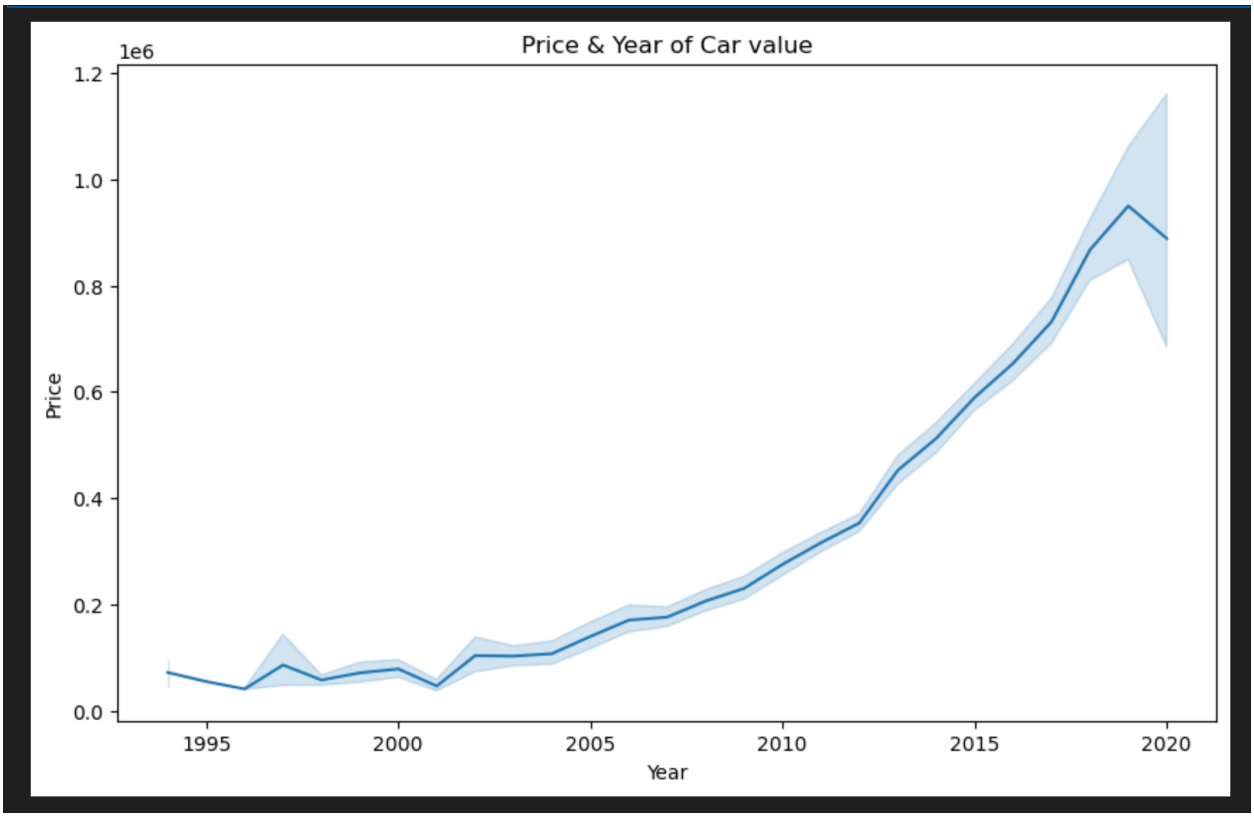
Trong giai đoạn phân tích và khám phá dữ liệu, dữ liệu đã được làm sạch bằng việc kiểm tra các giá trị bị trùng hoặc thiếu để đảm bảo chất lượng dữ liệu trước khi tiếp tục các bước phân tích khác.

Sau khi tiến hình xử lý, dữ liệu còn lại 6926 dòng và 12 cột

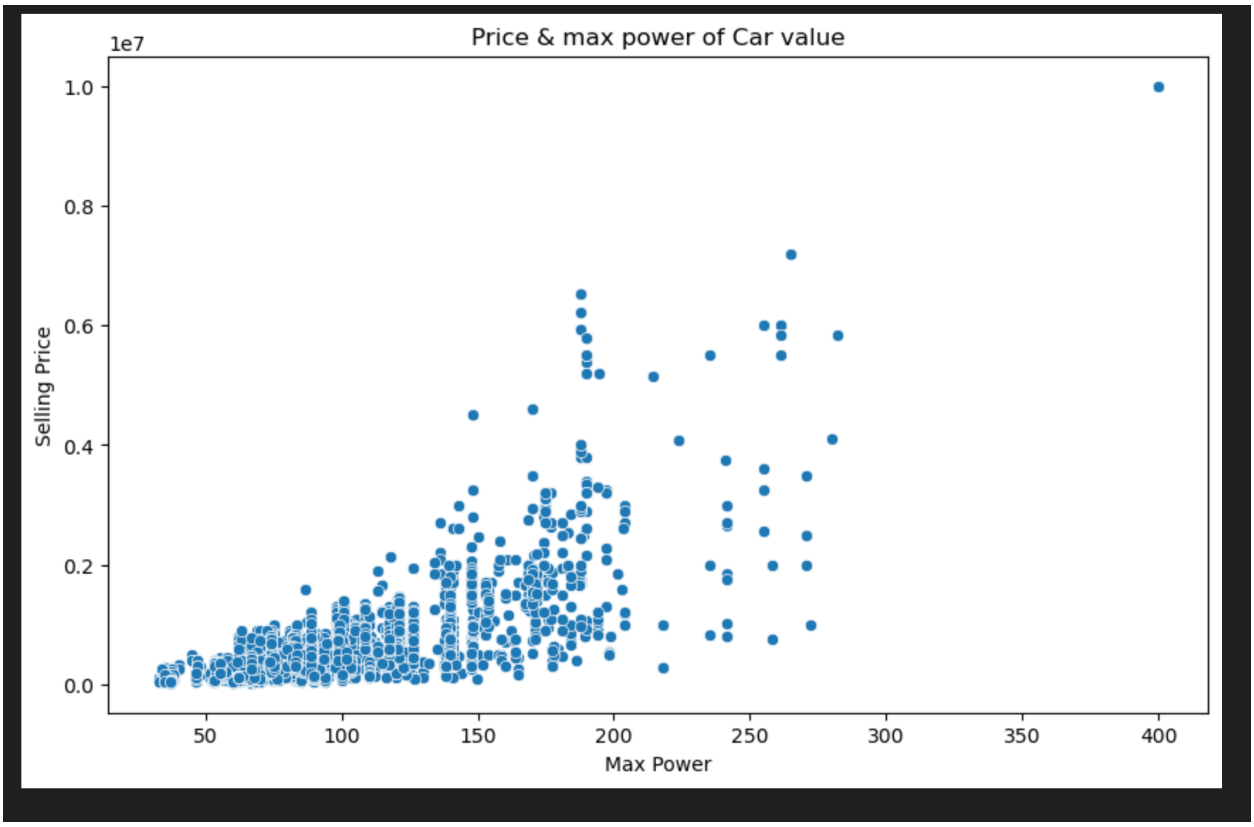
Bên cạnh đó, việc kiểm tra các cột như mileage, engine, max power,... giúp xác định các thang đo khác nhau trong bộ dữ liệu có thống nhất hay không để biến đổi các đơn vị đo này thành biểu diễn dạng số để có thể dễ dàng để mô phỏng.

Sau khi phân tích và biến đổi dữ liệu, việc mô phỏng bằng một số biểu đồ để kiểm tra sự liên kết giữa giá bán và một số thông tin khác của xe đã tối ưu hơn

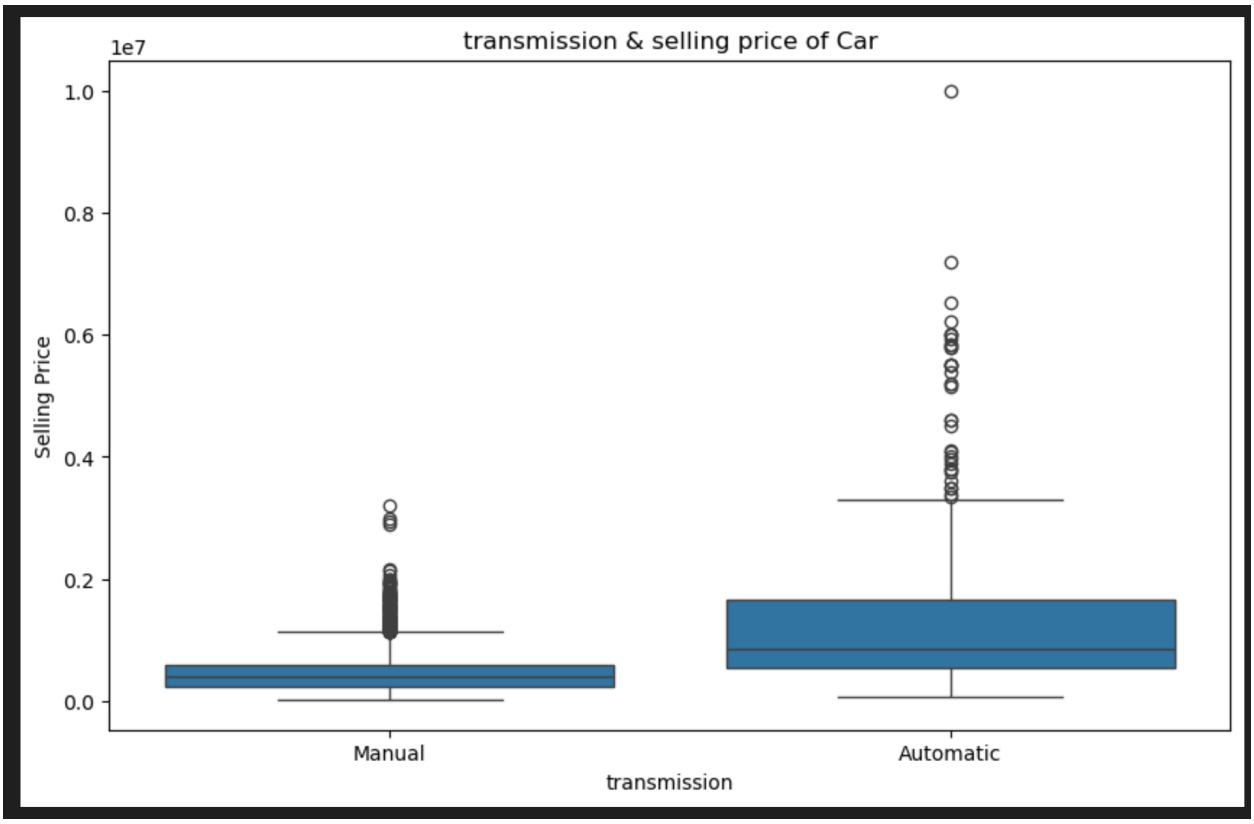
Như: năm sản xuất và giá bán xe



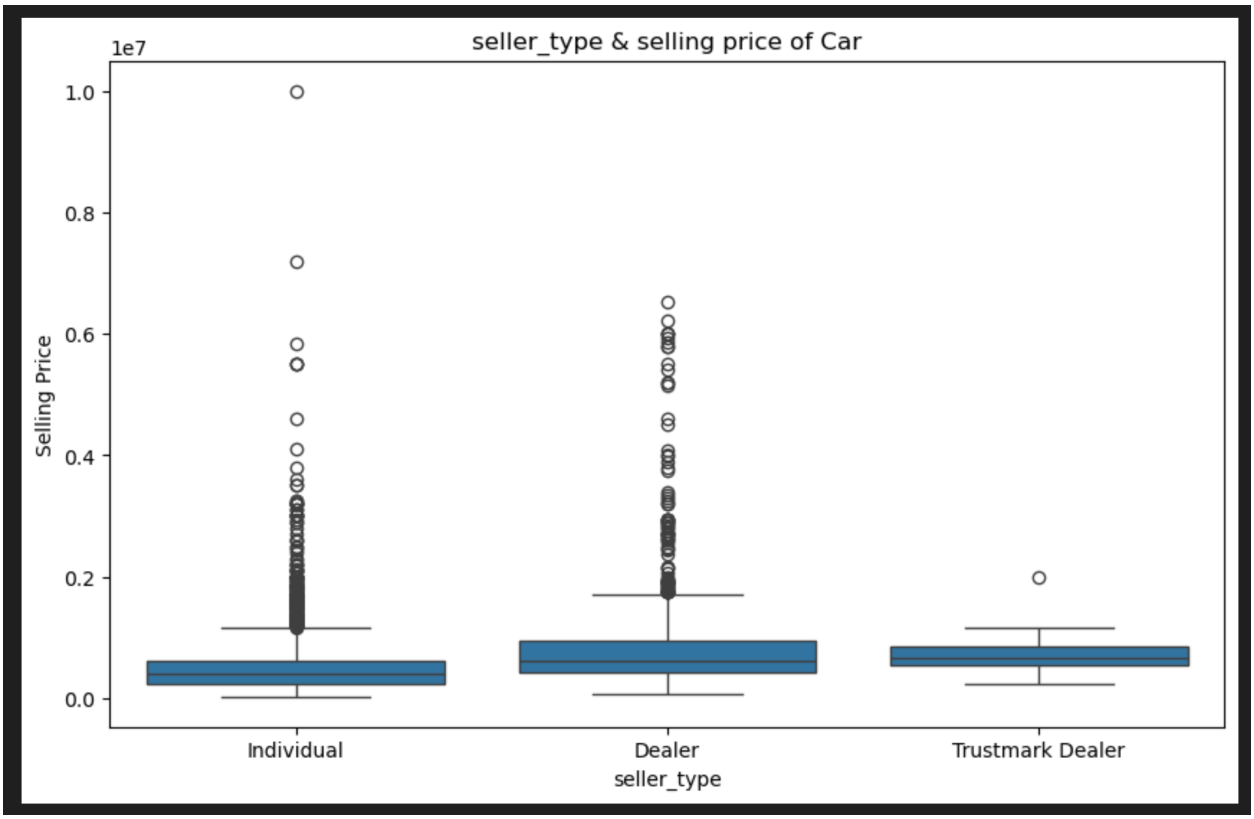
Hiệu năng và giá bán xe:



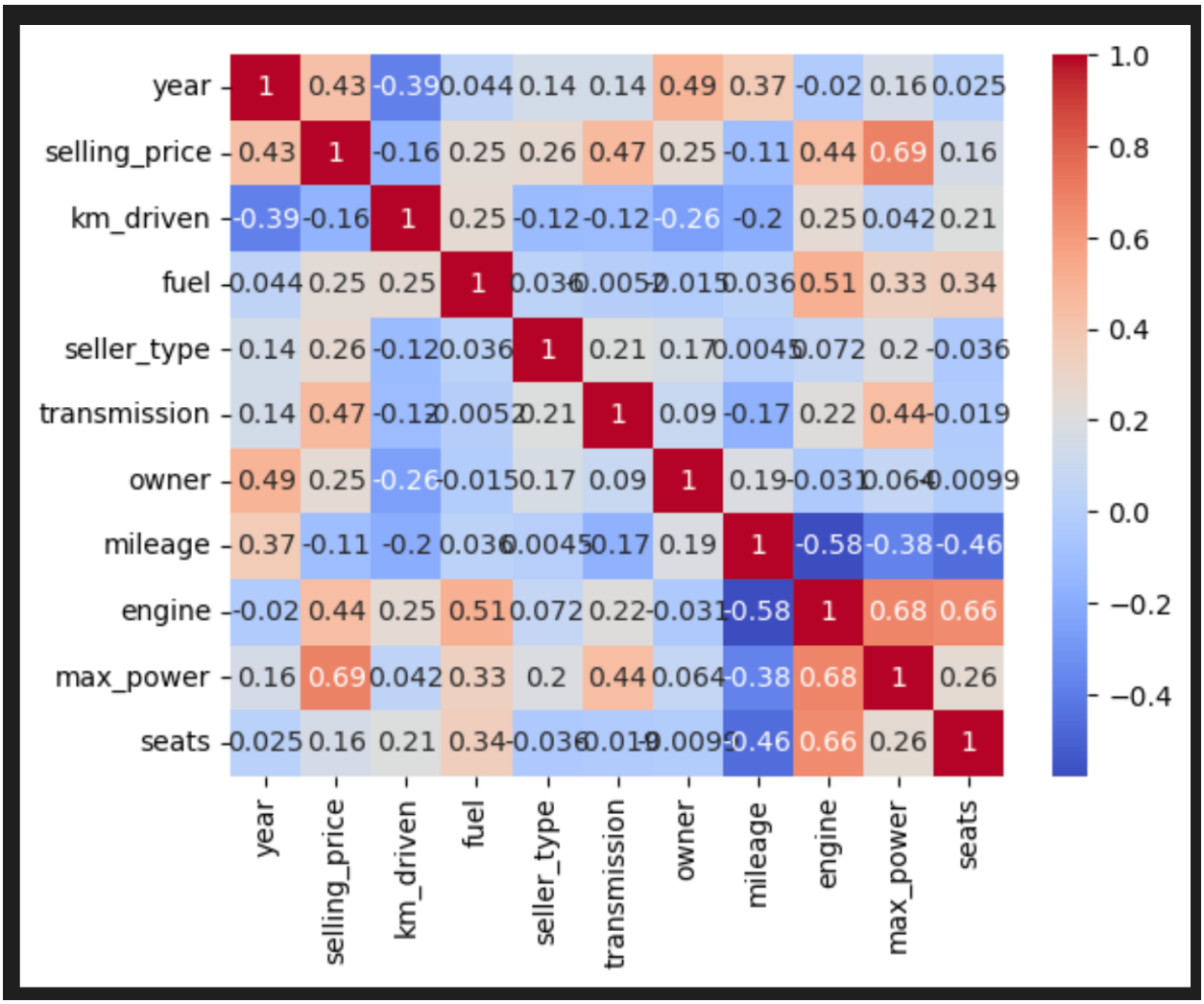
loại hộp số và giá xe:



Loại người bán và giá tiền của họ đề ra:



Việc xây dựng biểu đồ nhiệt dựa trên những quan sát này giúp chúng ta thấy được quan hệ giữa các đặc trưng có thể ảnh hưởng đến giá bán xe như thế nào



Vậy ta rút ra được hiệu năng của xe (0.69), loại hộp số (0.47), loại động cơ (0.44), năm sản xuất (0.43) liên quan mật thiết đến giá bán của xe và một số lượng các đặc trưng khác cũng giúp cho việc dự đoán giá xe chính xác hơn

Pipeline:

Tiền xử lý và chuyển đổi đặc trưng:

Tiền xử lý:

Bỏ đi cột torque do đặc trưng không nhất quán và có thể gây nhiễu

Kiểm tra số lượng dòng bị trùng và loại bỏ chúng

Kiểm tra số lượng dòng mà một số đặc trưng quyết định bị thiếu, vì chỉ chiếm 3.1% tổng số dữ liệu, một số lượng khá nhỏ, việc loại bỏ những dòng có giá trị thiếu sẽ giúp việc xây dựng mô hình dễ dàng hơn

Sau đó, kiểm tra thang đo của các đơn vị trong dữ liệu gốc và chuyển đổi các đơn các đặc trưng thành dạng số thực để dễ tính toán

Sau đó, chuyển đổi tên xe thành những tên hãng xe để dễ dàng phân loại

Biến đổi đặc trưng:

Áp dụng One-hot encoding cho đặc trưng hãng xe

Áp dụng Ordinal Encoding để biến đổi các đặc trưng người bán, loại hộp số, loại người sở hữu, và loại xăng thành các số 0,1,...

Áp dụng chuẩn hoá RobustScaler để chuẩn hoá cột max_power và engine dựa trên training data và áp dụng vào testing data, biến đổi các năm thành số năm tuổi của xe với 2020 là năm lớn nhất

Áp dụng log transformation lên giá bán của xe để co lại khoảng giá tiền outlier trong dữ liệu gốc để mô hình ổn định hơn

Lựa chọn đặc trưng:

Dựa trên heatmap, các đặc trưng như hiệu năng của xe, loại hộp số, người bán, năm sản xuất, loại nhiên liệu, người sở hữu, và động cơ ảnh hưởng lớn đến giá tiền của xe

Vậy nên các đặc trưng được chọn là: max_power, transmission, seller_type, owner, year, fuel, engine sẽ được đưa vào mô hình để học và dự đoán

Huấn luyện mô hình:

Phân chia tập dữ liệu với tỉ lệ 80/20 (với 80% là training data, 20% là test data)

Sử dụng 3 mô hình là Linear Regression, Random Forest, và Neural Network để tiến hành dự đoán và lựa chọn mô hình tốt nhất

Linear Regression: Mô hình được xây dựng áp dụng thư viện scikit-learn để làm baseline

Random Forest:

Cài đặt: số lượng cây quyết định: 100, trạng thái ngẫu nhiên: 42 (đảm bảo nhất quán trong quá trình huấn luyện)

Neural Network:

Cài đặt: (cài đặt bằng PyTorch)

Biến đổi các đặc trưng như X_train, X_test, y_train, y_test thành dạng Tensor để PyTorch có thể xử lý

Mạng nơ-ron bao gồm 4 lớp: 1 lớp nhận dữ liệu, 2 lớp ẩn, và 1 lớp output. Tất cả được áp dụng Batch Normalization để giúp quá trình huấn luyện ổn định hơn và tránh vấn đề explosive gradient

Với các lớp là hoàn toàn được kết nối với nhau được cài đặt như sau:

Input: Nhận đầu vào với kích thước là cột của dữ liệu, đầu ra có kích thước 256

Lớp ẩn thứ 1: Nhận đầu vào với kích thước 256, đầu ra có kích thước 128

Lớp ẩn thứ 2: Nhận đầu vào với kích thước 128, đầu ra có kích thước 64

Output: Nhận đầu vào với kích thước 64, đầu ra có kích thước 1 (là kết quả cuối cùng)

Mạng nơ-ron được huấn luyện với 500 epoch, sử dụng loại mất mát MSE, loại kích hoạt ReLU (để giúp mô hình học được các dạng dữ liệu phức tạp hơn), và sử dụng tối ưu hoá Adam (adaptive learning rate) với lr bắt đầu ở 0.01

Các mô hình sẽ được so sánh dựa trên kết quả huấn luyện để được chọn serving

Kết quả của quá trình huấn luyện: (dựa trên tập kiểm tra)

(Trên dữ liệu thực tế, đơn vị tiền tệ được tính theo đồng rupee - Ấn Độ)

Chỉ số	R^2	RMSE	MAE
Linear Regression	0.822	161219	93408
Random Forest	0.92	132284	77122
Neural Network	0.91	144320	84070

Việc kiểm tra sử dụng cross-validation (gồm 5 lần chia) để đảm bảo đầu ra đã chuẩn: (với giá trị trung bình trong 5 lần)

Chỉ số	R^2	RMSE	MAE
Linear Regression	0.83	213013	99488
Random Forest	0.9	163648	80157
Neural Network	0.88	179855	84324

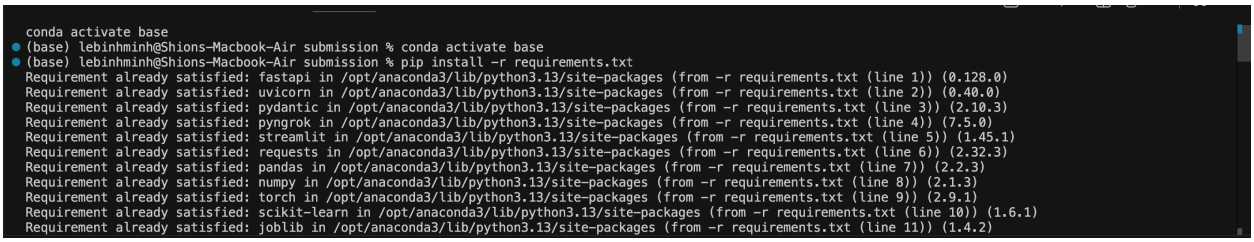
(RMSE được tính dựa trên dữ liệu khi đã được chuẩn hoá)

Vậy mô hình Random Forest cho ra kết quả tốt nhất, vậy nên random forest được dùng để triển khai.

Cách chạy ứng dụng:

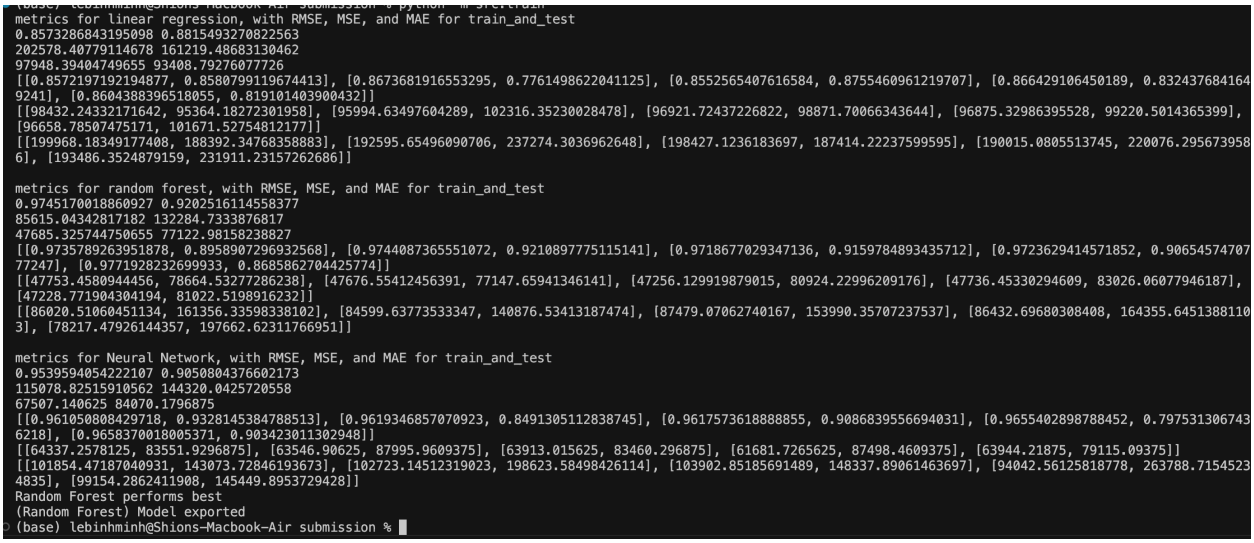
Bước 1: Cài đặt những thư viện cần dùng để chạy:

pip install -r requirements.txt



Bước 2: Chạy file huấn luyện (để lưu mô hình mới): (không bắt buộc)

python -m src.train



Bước 3: Khởi tạo API:

uvicorn app.API:app --reload

```
(base) lebinhminh@Shions-Macbook-Air submission % uvicorn app.API:app --reload
INFO:     Will watch for changes in these directories: ['/Users/lebinhminh/Downloads/FoML-main/submission']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [74623] using StatReload
INFO:     Started server process [74626]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

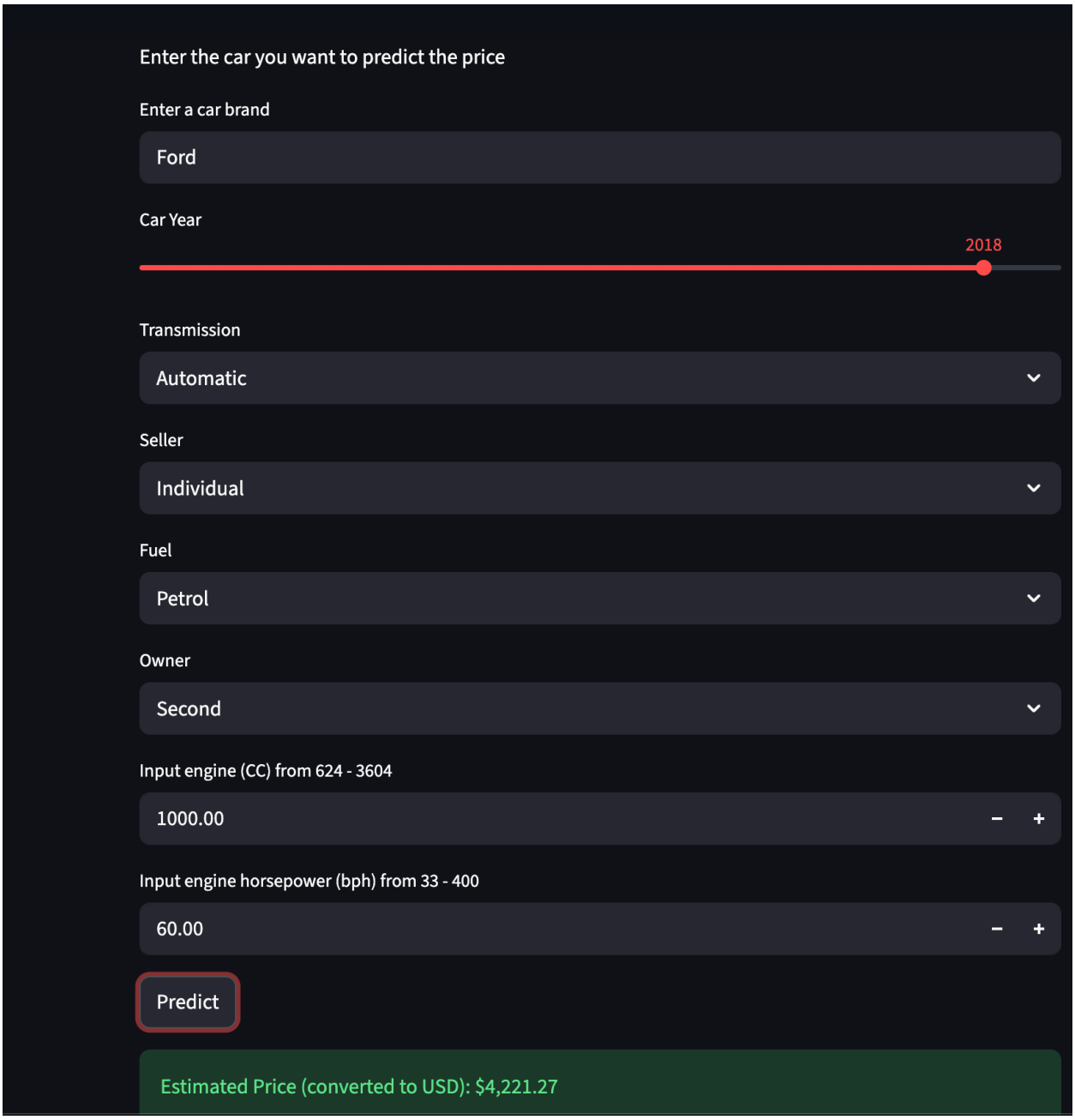
Bước 4: Khởi động ứng dụng:

streamlit run app/interface.py

```
conda activate base
(base) lebinhminh@Shions-Macbook-Air submission % conda activate base

(base) lebinhminh@Shions-Macbook-Air submission % streamlit run app/i
nterface.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.200.26:8501

  For better performance, install the Watchdog module:

  $ xcode-select --install
  $ pip install watchdog
```

Giao diện ứng dụng sẽ được khởi động

Bước 5: Nhập các thông tin và bấm nút Predict để tiến hành dự đoán giá xe (với các hãng xe trong 33 hãng đề cập trong file README.md trên github)



(Trong ứng dụng, giá tiền được biến đổi thành USD để dễ ước lượng)

Kết luận và hướng phát triển:

Kết luận:

Dự án này đã xây dựng hoàn chỉnh quy trình xây dựng mô hình học máy để dự đoán giá xe ô tô cũ. Quy trình bao gồm các bước làm sạch dữ liệu, tiền xử lý, biến đổi đặc trưng và lựa chọn đặc trưng cũng như việc lựa chọn các kiến trúc tối ưu để giải quyết bài toán dự đoán.

Hướng phát triển:

Dự án này còn có thể tối ưu việc dự đoán giá xe cũ dựa trên việc tiến hành tạo thêm dữ liệu bằng kỹ thuật data augmentation. Lý do của việc sử dụng data augmentation là để sinh ra thêm nhiều dữ liệu huấn luyện phong phú hơn, giúp mô hình hiểu được các mẫu dữ liệu phức tạp và đa dạng hơn, giúp cho việc dự đoán tốt hơn cũng như áp dụng các mô hình tiên tiến hơn, ví dụ như TabNet, giảm thiểu khả năng overfitting trên mô hình phức tạp hơn.

Việc áp dụng các kiến trúc DNN (mô hình mạng nơ-ron sâu) để tối ưu việc lựa chọn các đặc trưng và khái quát như việc transfer learning các kiến trúc Neural Network tối ưu cho dữ liệu bảng như mô hình TabNet [1]. Lý do việc lựa chọn TabNet là vì TabNet sử dụng cơ chế attention, giúp mô hình có thể tuỳ biến lựa chọn đặc trưng phù hợp qua nhiều bước suy luận, giúp mô hình có thể tối ưu và khái quát hoá tốt hơn trong thực tế.

Tham khảo:

1. Arik, S. Ö., & Pfister, T. (2021). TabNet: Attentive Interpretable Tabular Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8), 6679-6687.