Đại học quốc gia TP.HCM
Trường đại học công nghệ thông tin

Ngành: Khoa học máy tính

Môn học: CS116.P21

BÁO CÁO ĐỒ ÁN MÔN HỌC

Sinh viên:
Nguyễn Võ Ngọc Bảo –
MSSV: 23520131
Vũ Việt Cương –
MSSV: 23520213
Ngô Phương Nam –
MSSV: 23520974

Giảng viên:
TS. Nguyễn Vinh Tiệp

Lời mở đầu

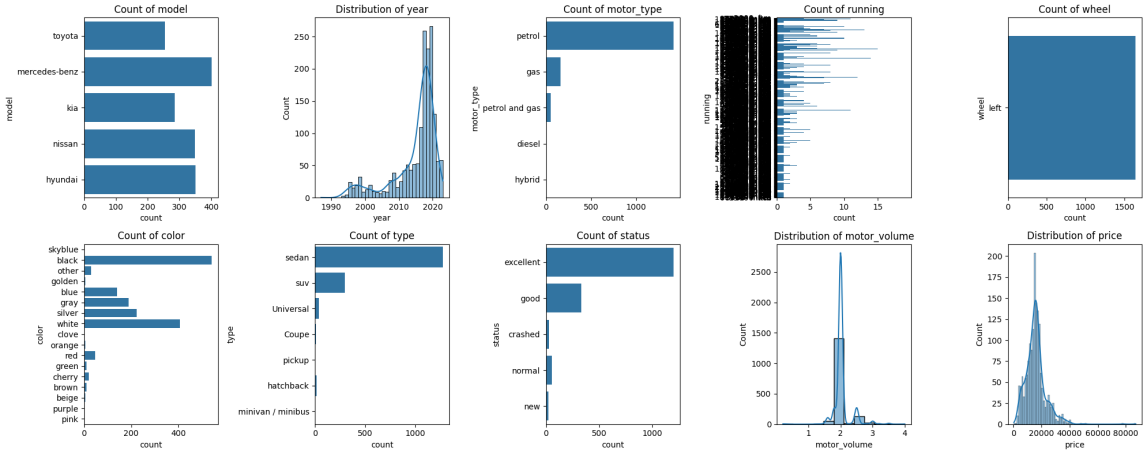
Tài liệu này trình bày kết quả đồ án môn học **CS116 – Lập trình Python cho Máy học**, với đề tài là về **Car Price Prediction**, bao gồm các yêu cầu, thiết kế, triển khai và đánh giá kết quả. Mục tiêu của đồ án là áp dụng các kiến thức đã học để xây dựng model, đồng thời rèn luyện kỹ năng làm việc nhóm, báo cáo và trình bày kết quả một cách bài bản.

Chúng em xin chân thành cảm ơn thầy **Nguyễn Vinh Tiệp** đã quan tâm theo dõi, hướng dẫn tận tình, truyền đạt kiến thức và kinh nghiệm bổ ích cho chúng em trong suốt thời gian học tập môn học này.

Mục lục

1 Giới thiệu chung

– Tên đề tài: Car Price Prediction
– Thông tin bộ dữ liệu: Car Price Prediction Dataset
– Jupyter Notebook của nhóm: Google Colab



Hình 1: Mô tả số liệu của cột trong dữ liệu

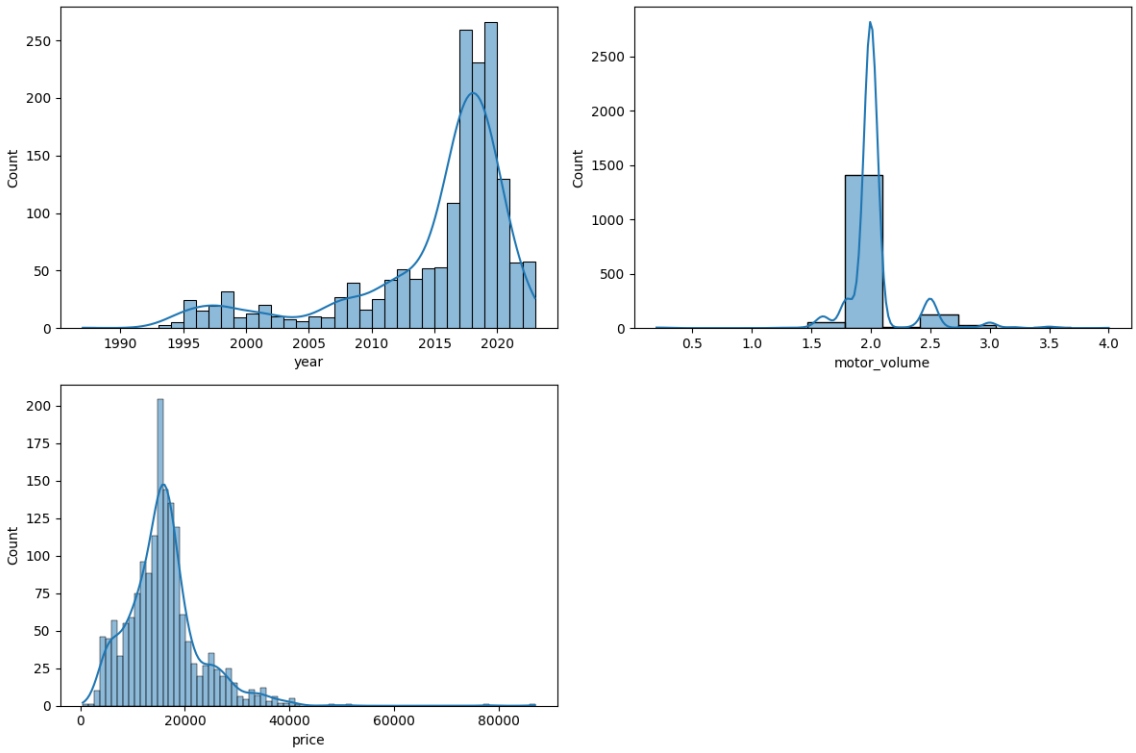
– Bộ dữ liệu bao gồm 10 đặc trưng dữ liệu được trực quan hoá ở *Hình 1*:

- *model*: Hãng xe
- *year*: Năm sản xuất
- *motor_type*: Loại xăng xe sử dụng
- *running*: Số km/miles xe đã chạy
- *wheel*: Hướng đặt vô lăng xe
- *color*: Màu sắc của xe
- *type*: Loại xe
- *status*: Tình trạng xe
- *motor_volume*: Dung tích xe
- *price*: Giá bán của xe

2 Phân tích dữ liệu với EDA

2.1 Phân tích sự phân phối dữ liệu

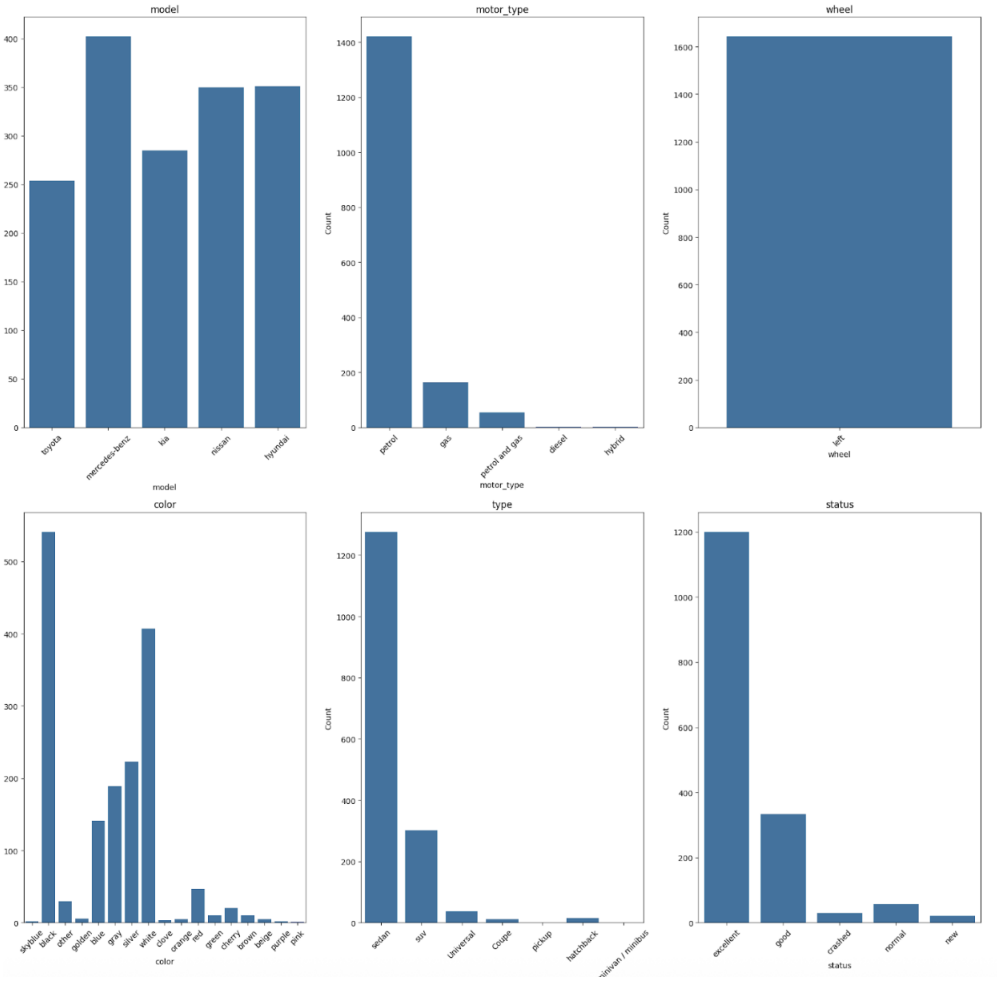
• Các trường dữ liệu số: year, running và motor_volume



Hình 2: Mô tả sự phân phối của các đặc trưng số

– Đối với `year`, sự phân phối có phần lệch phải nhẹ. Số lượng mẫu giảm dần rõ rệt từ khoảng năm 2005. Giai đoạn trước năm 2000 có rất ít mẫu → Có thể phản ánh rằng xe được sản xuất từ những năm gần đây.
– `running`: phân phối lệch trái mạnh. Phần lớn xe có số km đã chạy thấp, số xe chạy trên 400000 miles rất ít.
– `motor_value`: tập trung ở một khoảng giá trị nhất định. Đa phần xe có dung tích xi lanh khoảng 2.0L. Có một số ít xe có dung tích xi lanh lớn hơn 3.0L hoặc nhỏ hơn 1.5L. Cho thấy rằng thị trường ở đây là các dòng xe phổ thông, không thiên về xe có phân phối lớn.

• Các đặc trưng dữ liệu phân loại: model, motor_type, wheel, color, type, status

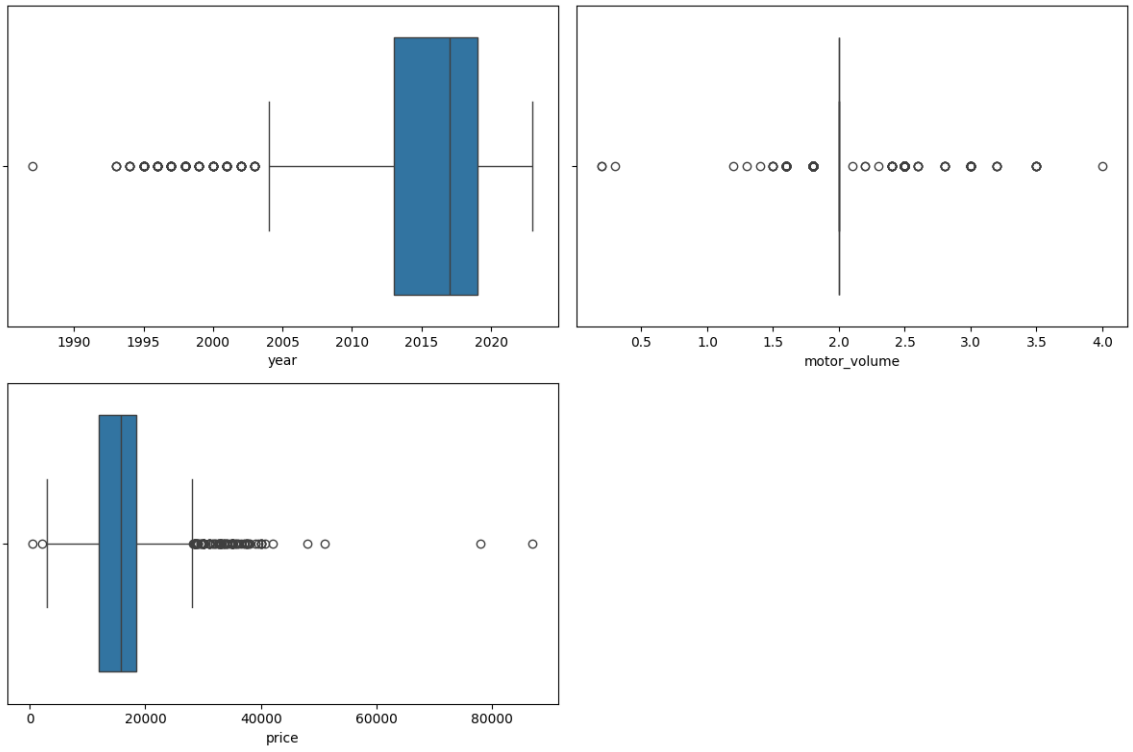


Hình 3: Mô tả sự phân phối của các đặc trưng phân loại

– `model`: các hãng xe phổ biến là mercedes-benz với 400 mẫu, theo sau là kia, nissan và hyundai 350 mẫu và toyota ít mẫu hơn 250 mẫu.
– `motor_type`: áp đảo là petrol (xăng) với hơn 1400 mẫu. Rất ít xe sử dụng gas, petrol/gas hoặc diesel. Gần như không có xe hybrid –> Dataset chủ yếu là xe chạy xăng truyền thống.

– `wheel`: tất cả các xe đều có vị trí tay lái ở bên trái.
– `color`: màu xe phổ biến nhất là màu đen (>500) sau đó là white, sliver, gray và blue. Các màu pink, gold, orange, green rất hiếm.
– `type`: chủ yếu là sedan và suv. Các loại xe như universal, coupe, pickup chiếm tỉ lệ rất nhỏ.
– `status`: phần lớn là excellent, good (tình trạng tốt) và new hoặc normal. Một số ít là crashed.

2.2 Phân tích outlier



Hình 4: Mô tả các dữ liệu ngoại lệ ở 3 đặc trưng quan trọng

– Biến `year`: tập trung chủ yếu từ năm 2012 đến 2022. Với outliers là một vài giá trị nằm trước năm 2005, đặc biệt trước 1995.
– Biến `running`: phân bố chính dưới 300000. Có nhiều giá trị vượt biên 400000 thậm chí hơn 1000000.

– Biến `motor_volume`: phân bố chính quanh 2.0L. Một số dòng xe có giá trị nhỏ hơn 1.0L hoặc lớn hơn 3.5L.

2.3 Phân tích đơn biến

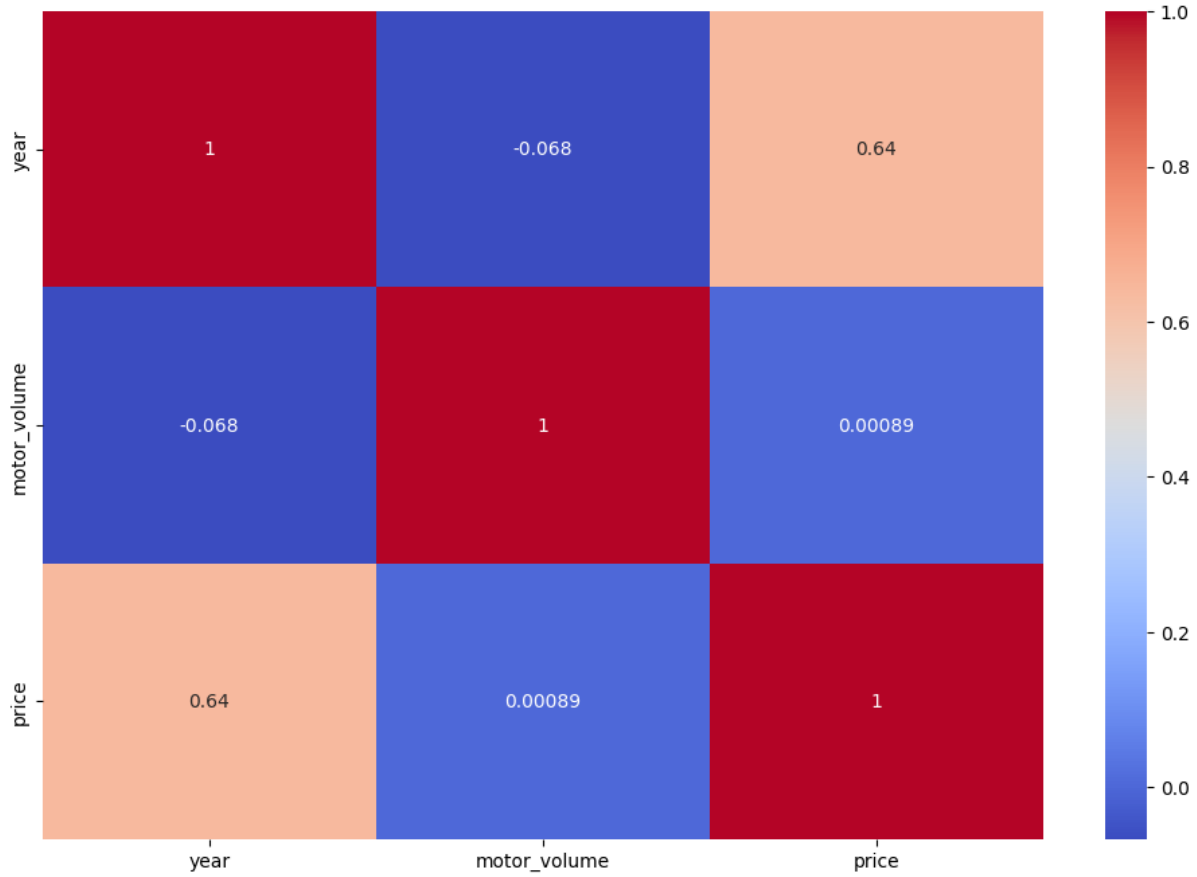
Bảng thống kê các đơn biến quan trọng

	year	motor_volume	price
count	1642.00	1642.00	1642.00
mean	2014.81	2.0350	15982.63
std	6.5876	0.2531	7176.08
min	1987.00	0.2000	462.00
25%	2013.00	2.0000	12000.00
50%	2017.00	2.0000	15750.00
75%	2019.00	2.0000	18500.00
max	2023.00	4.0000	87000.00

– **Số lượng**: 1642 mẫu cho mỗi biến.
– **Năm sản xuất trung bình** đạt 2014.8, với năm bé nhất là 1987 và năm lớn nhất là 2023.
– **Quãng đường**: Trung bình đạt 119,210 km, với min 10 và max 1,251,708, phân tán lớn.
– **Dung tích động cơ**: nằm trong khoảng [0.2, 4.0] với trung bình 2.035 lít.
– **Giá**: Trung bình cao đến 15,982.63$ trong khoảng [462, 87000], có sự phân tán.

2.4 Phân tích đa biến

– Đối với *Feature - Feature*:



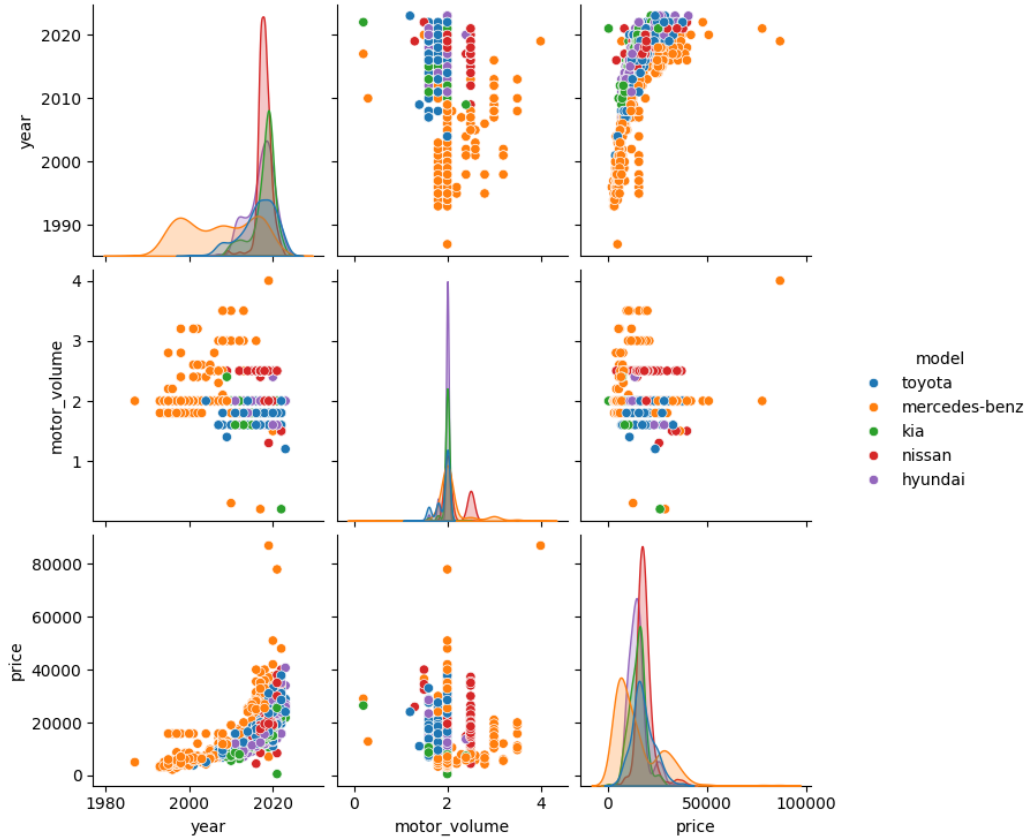
Hình 4: Phân tích độ tương quan giữa feature - feature

- **Tương quan giữa `year` và `running`**: hệ số $r = -0.66$.
  - Cho thấy mối tương quan nghịch mạnh: xe đời mới thường đi ít km hơn, ngược lại xe đời cũ đã đi nhiều km.
  - Kết luận: có quan hệ tuyến tính nghịch khá mạnh.
- **Tương quan giữa `year` và `motor_volume`**: hệ số $r = -0.068$.

  - Giá trị rất gần 0, cho thấy hầu như không có mối liên hệ tuyến tính giữa năm sản xuất và dung tích động cơ.
  - Kết luận: gần như không có quan hệ tuyến tính.

- **Tương quan giữa `running` và `motor_volume`**: hệ số $r = 0.10$.
  - Cho thấy mối tương quan thuận rất yếu: có xu hướng nhẹ là xe động cơ lớn đi nhiều km hơn nhưng không đáng kể.
  - Kết luận: chỉ có quan hệ tuyến tính thuận rất yếu.

– Đối với *Feature - Target*:



Hình 5: Phân tích độ tương quan giữa feature - target

- **year – price**: Quan sát điểm rải trên cột (`year` vs. `price`) cho thấy rõ xu hướng tăng dần của giá ô tô theo năm sản xuất. Các mẫu xe đời từ

2015 trở về sau đa phần rơi vào vùng giá cao (>20.000 USD), trong khi xe đời cũ hơn (trước 2005) thường có giá thấp (<10.000 USD). Đường KDE dọc trục `price` cũng thể hiện phân bố "đỉnh" ở mức trung bình (15.000–25.000 USD) và đuôi dài về phía giá cao do một số mẫu xe sang.

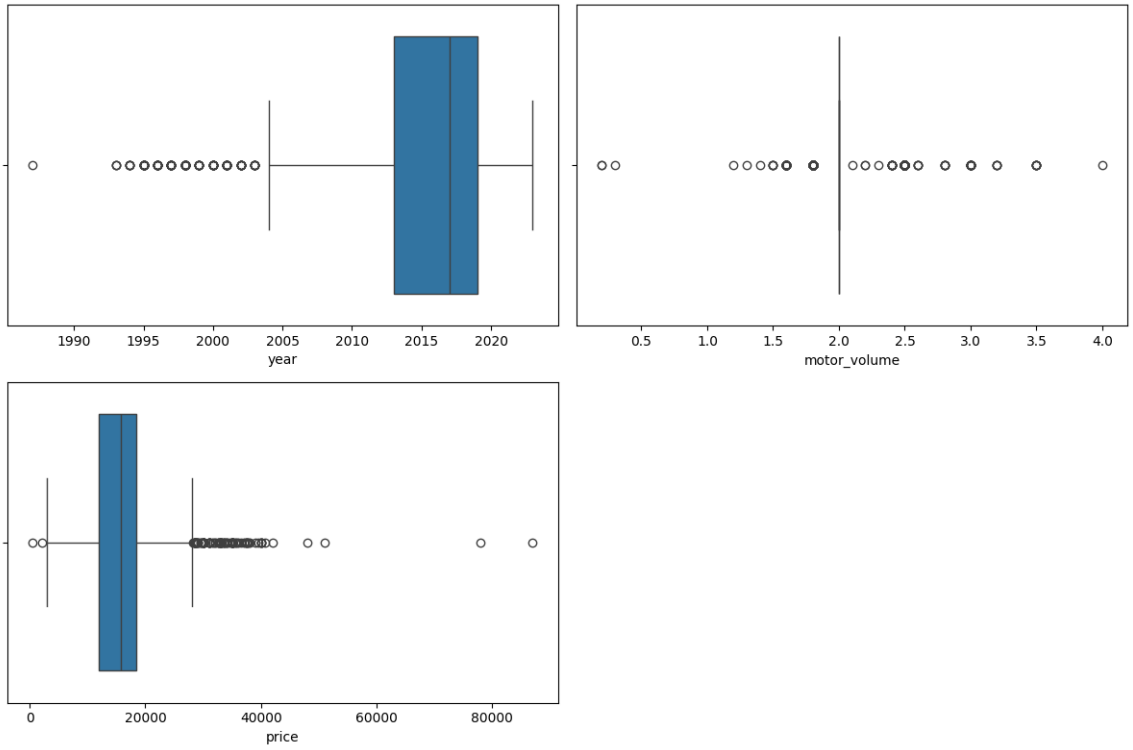
- **motor_volume – price**: Ở hàng (`motor_volume` vs. `price`), ta thấy dung tích động cơ càng lớn (2.5 L) thì giá trung bình thường cao hơn, nhiều điểm dữ liệu nằm trong khoảng 30.000–80.000 USD. Người mua sẵn sàng trả thêm cho các mẫu động cơ lớn, đặc biệt là với thương hiệu Mercedes-Benz.

- **year – motor_volume**: Mối quan hệ giữa hai biến này không quá chặt chẽ: xe đời mới có dung tích động cơ trải dài từ 1.0 đến 4.0 L, nhưng phần lớn tập trung quanh 1.5–2.0 L. Điều này cho thấy trong cùng một khoảng thời gian, phân khúc xe gia đình (1.5–2.0 L) vẫn chiếm ưu thế.

- >So sánh theo thương hiệu: Mercedes-Benz (màu cam) thể hiện độ biến thiên giá và dung tích động cơ lớn nhất, với nhiều mẫu "cao cấp" giá lên đến 80.000–100.000 USD. Toyota (xanh dương), Kia (xanh lá), Hyundai (tím) và Nissan (đỏ) chủ yếu nằm trong phân khúc tầm trung, dung tích 1.2–2.0 L và giá từ 5.000 đến 35.000 USD.

3 Tiền xử lý dữ liệu

3.1 Chiến thuật xử lý dữ liệu ngoại lệ



Hình 5: Mô tả các dữ liệu ngoại lệ ở 3 trường quan trọng

- Các mô hình Catboost và XGBoost ít nhạy cảm với outlier nên giá trị dự đoán không bị kéo lệch.
- Giữ lại outlier để mô tả toàn diện bộ dữ liệu, cần biết tình trạng thị trường thực tế.

3.2 Chiến thuật xử lý dữ liệu bị khuyết

– Chuẩn bị cho trường hợp phát sinh là có chứa NaN thì ta sẽ sử dụng SimpleImputer từ thư viện Scikit-learn:

```
import numpy as np
from sklearn.impute import SimpleImputer

category_cols = train.select_dtypes(exclude=[np.number])
    .columns.tolist()

imputer_mean = SimpleImputer(strategy='mean')
train[numeric_cols] = imputer_mean.fit_transform(train[
    numeric_cols])

imputer_mode = SimpleImputer(strategy='most_frequent')
train[category_cols] = imputer_mode.fit_transform(train[
    category_cols])

train.isnull().sum()
```

• Cột số (numeric): Thay thế NaN bằng giá trị trung bình (strategy='mean').
• Cột phân loại (category): Thay thế NaN bằng giá trị xuất hiện thường xuyên nhất (strategy='most_frequent').

Lưu ý: Trong tập test, cột running_km có 262/411 giá trị non-null, điều này cho thấy dữ liệu test chứa NaN nhưng chưa được xử lý. Cần kiểm tra lại quy trình áp dụng Imputer cho tập test.

3.3 Chiến thuật encode dữ liệu

– Đầu tiên, dữ liệu trong cột running không cùng đơn vị (mile, km). Do đó, cần chuyển dữ liệu về cùng một đơn vị là km.

```
2         95000  miles
6         49000  miles
9         58000  miles
11          135800  km
13          220000  km
              ...
1635        180000  km
1637    120000  miles
1638        170000  km
1639     68900  miles
1640     31000  miles
Name: running, Length: 833, dtype: object
```

Hình 6.1: Dữ liệu của running trước khi xử lý

– Sau khi xử lý, do đã cùng một đơn vị nên bỏ phần đơn vị để trở thành dạng số.

```
2       152887.300
6        78857.660
9        93341.720
11      135800.000
13      220000.000
           ...
1635    180000.000
1637    193120.800
1638    170000.000
1639    110883.526
1640     49889.540
Name: running_km, Length: 833, dtype: float64
```

Hình 6.2: Dữ liệu của $running_k msaukhixl$

```
def convert_to_km(dist):
  val, unit = re.match(r'(\d+)\s*([a-zA-Z]+)', dist).
     groups()
  if unit.lower() == 'miles':
    return float(val) * 1.60934
  else:
    return float(val)

train['running_km'] = train['running'].apply(
   convert_to_km)
test['running_km'] = train['running'].apply(
   convert_to_km)
sns.histplot(train['running_km'], kde=True)
```

– Tiếp theo là sử dụng LabelEncoding để biến đổi dữ liệu dạng danh mục (phân loại).
– Với biến `model`:

- hyundai –> 0
- kia –> 1
- mercedes-benz –> 2
- nissan –> 3
- toyota –> 4

– Với biến `motor_type`:

- gas –> 1
- petrol –> 2
- petrol and gas –> 3

– Với biến `wheel`, do chỉ có một giá trị duy nhất là 'left', nghĩa là tất cả các xe đều có vô lăng nằm bên trái nên không có sự biến thiên trong dữ liệu –> không mang thông tin hữu ích cho mô hình –> sẽ được loại bỏ.

– Với biến `color`:

- black –> 1
- blue –> 2
- brown –> 3
- cherry –> 4
- clove –> 5
- golden –> 6
- gray –> 7
- green –> 8
- orange –> 9
- other –> 10
- pink –> 11
- purple –> 12
- red –> 13
- silver –> 14
- sky blue –> 15
- white –> 16

– Với biến `type`:

- universal –> 1
- hatchback –> 2
- minivan minibus –> 3
- pickup –> 4
- sedan –> 5
- suv –> 6

– Cuối cùng là biến `status`:

- crashed –> 0
- excellent –> 1
- good –> 2
- new –> 3
- normal –> 4

3.4 Chiến thuật thêm đặc trưng mới

– Ta thêm cột `running_per_year` thể hiện số km chạy được mỗi năm.

```
train['run_per_year'] = train['running_km'] / (2025-
    train['year'])
test['run_per_year'] = test['running_km'] / (2025-test['
    year'])
print(train[['running_km','year','run_per_year']])
```

Bảng thống kê mô tả các biến `running_km`, `year` và `run_per_year`

	running_km	year	run_per_year
0	3000.000	2022	1000.000000
1	132000.000	2014	12000.000000
2	152887.300	2018	21841.042857
3	220479.580	2002	9586.068696
4	130000.000	2017	16250.000000
...	...	...	...
1637	193120.800	2017	24140.100000
1638	170000.000	2014	15454.545455
1639	110883.526	2018	15840.503714
1640	49889.540	2019	8314.923333
1641	20.000	2022	6.666667

– Giá trị `running_per_year` cao hơn cho biết xe đã được sử dụng nhiều hơn mỗi năm, điều này có thể cho thấy mức độ hao mòn cao hơn.

– Giá trị `running_per_year` thấp hơn cho biết mức độ sử dụng ít hơn, điều này có thể cho thấy mức độ bảo dưỡng tốt hơn và có khả năng giá trị bán lại cao hơn.
–> Đặc trưng này năng này giúp nắm bắt mối quan hệ giữa tuổi của xe và mức độ sử dụng của xe, đây có thể là yếu tố quan trọng trong việc xác định giá của xe.

4 Chọn lọc các đặc trưng

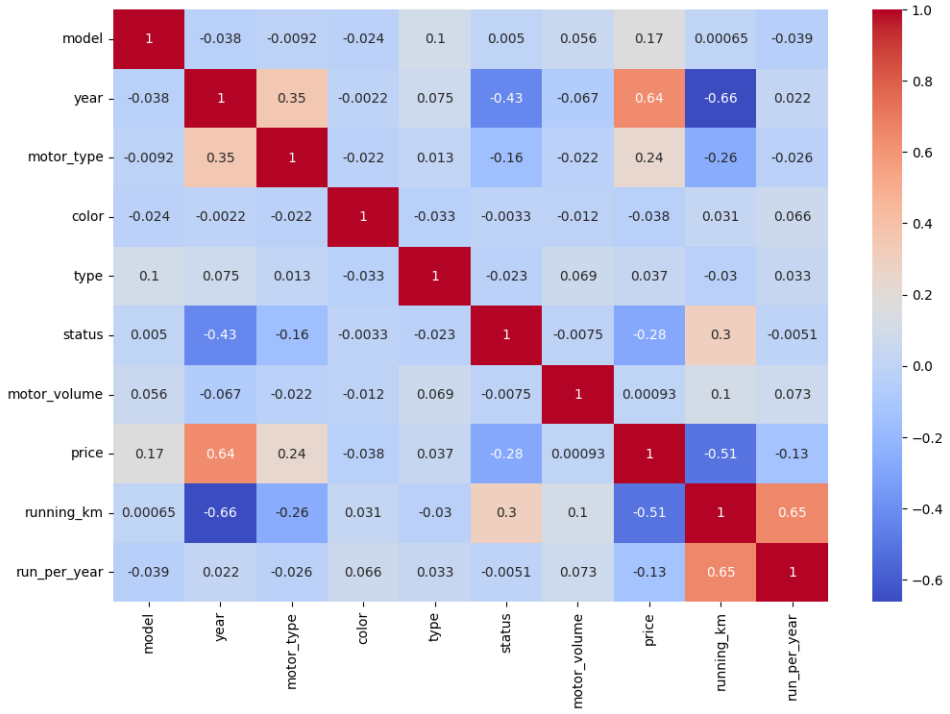
– Ta sẽ xóa bỏ cột `running` và giữ lại cột `running_km` vì cột `running` không đồng nhất đơn vị nên ta chuyển về chung một đơn vị ở cột `running_km`.

```
train.drop(columns=['running'], inplace=True)
test.drop(columns=['running'], inplace=True)
```

– Bên cạnh đó, ta thấy cột `wheel` chỉ mang 1 giá trị, không phục vụ trong việc dự đoán nên ta cũng bỏ cột `wheel`.

```
train.drop(columns=['wheel'], inplace=True)
test.drop(columns=['wheel'], inplace=True)
```

– Tiếp theo, ta đưa các đặc trưng vào correlation_matrix để chọn các đặc trưng phù hợp cho việc dự đoán. Nếu đặc trưng nào có độ tương quan cao (> 0.9) thì loại bỏ.



Hình 7: Correlation matrix của các đặc trưng

– Có thể thấy, không có đặc trưng nào có độ tương quan cao (> 0.9) để ta có thể loại bỏ.
– Ta sẽ sử dụng RandomForestRegressor để biết được các đặc trưng chiếm độ quan trọng bao nhiêu % trong việc dự đoán.

```
X = train.drop(columns=['price'], axis=1)
y = train['price']
```

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(random_state=42)
model.fit(X, y)

feature_importances = pd.DataFrame({
    'Feature': X.columns,
```

```
    'Importance': model.feature_importances_
})
feature_importances = feature_importances.sort_values(by
    ='Importance', ascending=False)
top_features = feature_importances.head(10)

print(top_features)
X.drop(columns=['motor_type'], inplace=True)
```

Bảng thống kê mức độ quan trọng của các đặc trưng

	Feature	Importance
1	year	53.9%
0	model	22.5%
7	running_km	6.1%
6	motor_volume	5.7%
8	run_per_year	5.04%
5	status	3.1%
3	color	2.2%
4	type	1.2%
2	motor_type	0.36%

– Ở đây, cột year chiếm tỉ lệ cao nhất (53,9%).
– Cuối cùng, ta áp dụng phương pháp Recursive Feature Elimination (RFE) để loại bỏ dần các đặc trưng chiếm tỉ lệ thấp và sau đó train lại.

```
from sklearn.ensemble import RandomForestRegressor

rfe = RFE(estimator=RandomForestRegressor(),
    n_features_to_select=10)
rfe.fit(X, y)

selected_features = X.columns[rfe.support_]
print(selected_features)
```

– Sau khi loại bỏ, ta chọn ra được 5 đặc trưng cao nhất bao gồm: `model`, `year`, `motor_volume`, `running_km` và `run_per_year`.

5 Triển khai các model trên dữ liệu

– Trước hết, ta sẽ chia tập dữ liệu huấn luyện theo tỉ lệ 8:2.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y
    , test_size=0.2, random_state=42)
```

5.1 Sử dụng model CatBoost

5.1.1 Giới thiệu

– CatBoost (viết tắt của "Categorical Boosting") là thư viện học máy do Yandex phát triển, thuộc dòng thuật toán Gradient Boosting trên cây quyết định. Điểm nổi bật của CatBoost là khả năng xử lý trực tiếp biến phân loại mà không cần chuyển sang one-hot encoding, nhờ đó giảm thiểu sai số và cải thiện hiệu năng. Ngoài ra, CatBoost áp dụng cơ chế "Ordered Boosting" giúp tránh hiện tượng overfitting, đặc biệt hiệu quả trên các tập dữ liệu có nhiều biến phân loại. Thư viện cũng hỗ trợ huấn luyện trên GPU, cho phép tăng tốc độ xử lý khi làm việc với khối lượng dữ liệu lớn.

5.1.2 Các bước thực hiện

– Dữ liệu được chia làm tập huấn luyện và kiểm tra theo tỉ lệ 8:2 với random_state=42 để tái lập kết quả.
– Các cột dạng phân loại được nhận diện và khai báo trong Pool (CatBoost format).

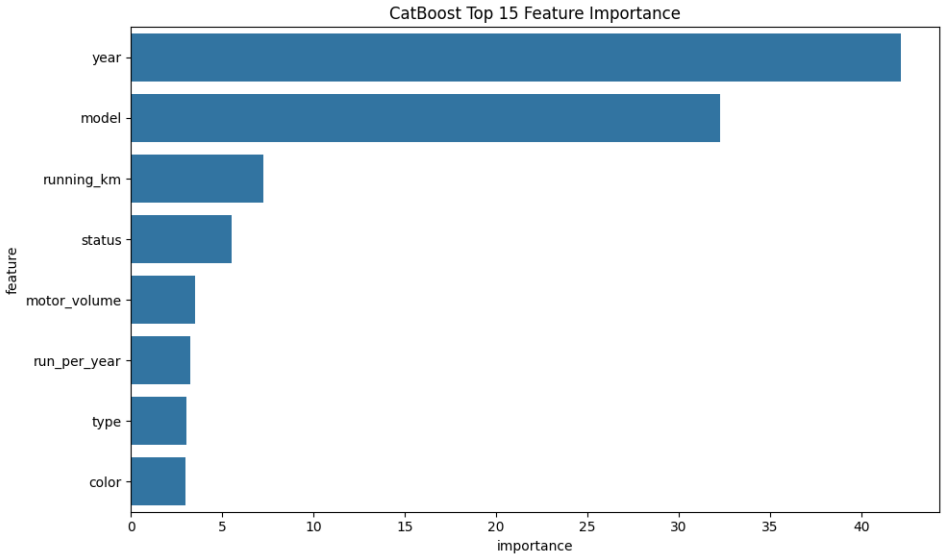
```
from catboost import CatBoostRegressor, Pool

category_cols = [c for c in category_cols if c in
    X_train.columns]

cat_model = CatBoostRegressor(loss_function='MAE',
    verbose=0, iterations=1200, random_state=42,
    cat_features=category_cols)

train_pool = Pool(data=X_train, label=y_train,
    cat_features=category_cols)
```

```
eval_pool = Pool(data=X_test, label=y_test, cat_features
    =category_cols)

cat_model.fit(train_pool, eval_set=eval_pool,
    use_best_model=True)
```

5.1.3 Đánh giá MAE, RMSE

– Chỉ số RMSE đạt được: 3199.2004.
– Chỉ số MAE đạt được: 1846.8110.

5.1.4 Phân tích các đặc trưng quan trọng



Hình 8: Top 15 đặc trưng quan trọng đối với model CatBoost

– Biểu đồ feature importance cho thấy các đặc trưng quan trọng nhất là: `year`, `model`, `running_km` và `status`.
–> Các biến liên quan đến loại xe và mức độ sử dụng ảnh hưởng nhiều nhất đến kết quả dự đoán.

5.2 Sử dụng model XGBoost

5.2.1 Giới thiệu

– XGBoost (eXtreme Gradient Boosting) là một thư viện mã nguồn mở nổi tiếng về thuật toán Gradient Boosting, phát triển bởi Tianqi Chen. XGBoost được tối ưu hoá cao về hiệu năng và khả năng mở rộng, tận dụng tính song song (parallel processing) và các kỹ thuật như pruning, regularization (L1/L2) để giảm overfitting. Bên cạnh đó, XGBoost hỗ trợ huấn luyện trên CPU và GPU, xử lý dữ liệu thiếu tự động và cung cấp API linh hoạt cho Python, R, Java, Scala,..., nên rất phổ biến trong các cuộc thi Kaggle và ứng dụng thực tế.

5.2.2 Các bước thực hiện

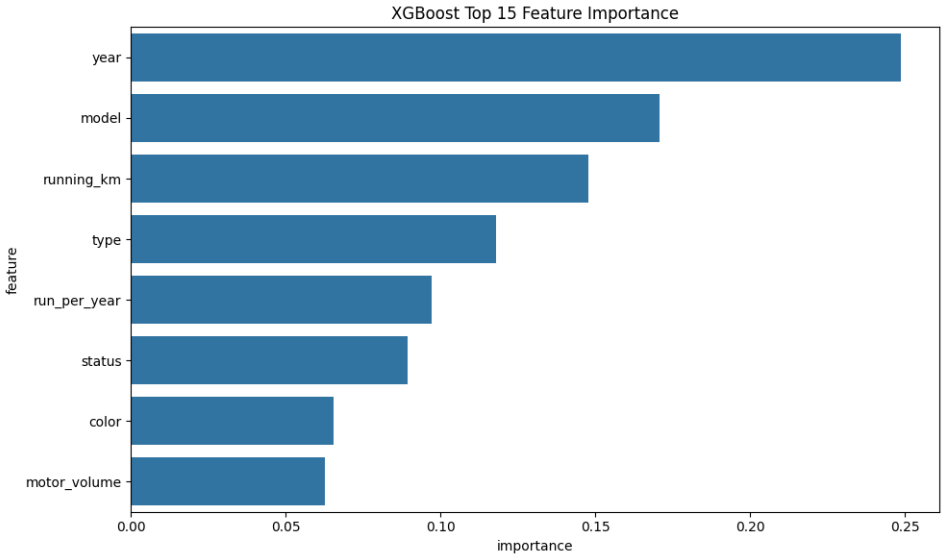
– Sử dụng các tham số được tối ưu bao gồm: max_depth, learning_rate, n_estimators, subsample, colsample_bytree, reg_alpha, reg_lambda, v.v...

```
from xgboost import XGBRegressor

params_xgb = {'booster': 'gbtree',
          'max_depth': 3,
          'max_leaves': 769,
          'learning_rate': 0.04538451353216046,
          'n_estimators': 1171,
          'min_child_weight': 13,
          'subsample': 0.6578720167306904,
          'reg_alpha': 0.4622943878867952,
          'reg_lambda': 0.6211309481623339,
          'colsample_bylevel': 0.7985625445322192,
          'colsample_bytree': 0.9634723040072963,
          'colsample_bynode': 0.49814271378837316,
          'objective': 'reg:absoluteerror',
          'n_jobs': -1,
          'random_state': 42
         }

xgb_model = XGBRegressor(**params_xgb)
xgb_model.fit(X_train, y_train)
```

5.2.3 Đánh giá MAE, RMSE

– Chỉ số RMSE đạt được: 3296.6259.
– Chỉ số MAE đạt được: 1903.0657.
–> Hiệu suất tệ hơn so với CatBoost.

5.2.4 Phân tích các đặc trưng quan trọng



Hình 9: Top 15 đặc trưng quan trọng đối với model XGBoost

– Biểu đồ feature importance cho thấy các đặc trưng quan trọng nhất là: `year`, `model`, `running_km` và `type`.

5.3 Sử dụng kết hợp model CatBoost và XGBoost

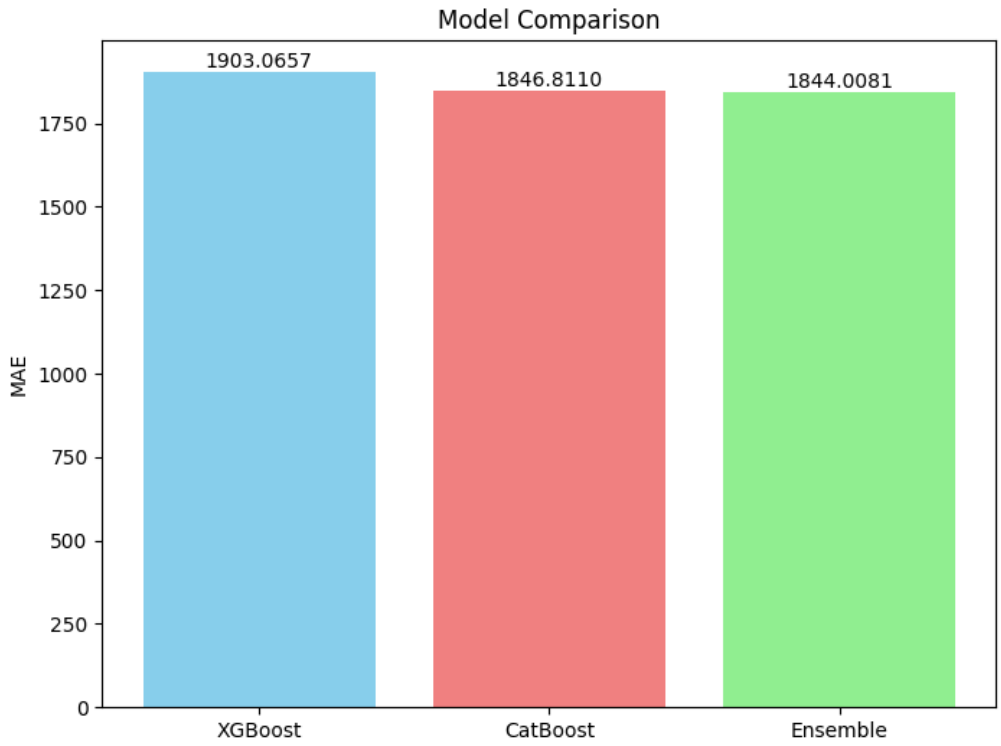
– Sau khi đánh giá riêng lẻ hai mô hình XGBoost và CatBoost, ta tiến hành kết hợp chúng để cải thiện hiệu suất dự đoán bằng cách tận dụng ưu điểm của từng mô hình.
– XGBoost hoạt động tốt trên dữ liệu đã qua xử lý và tối ưu siêu tham số.
– CatBoost rất mạnh khi làm việc với dữ liệu chứa nhiều đặc trưng phân loại (categorical features) và không cần mã hóa.

– Hai mô hình có cấu trúc và cơ chế học khác nhau –> sai số dự đoán có thể mang tính bù trừ.
–> Kết hợp dự đoán của cả hai mô hình thường dẫn đến kết quả ổn định và chính xác hơn.
– Ta sẽ sử dụng phương pháp trung bình đơn giản theo tỉ lệ 35:65.

$$\hat{y}_{\text{ens}} = 0.35\,\hat{y}_{\text{xgb}} + 0.65\,\hat{y}_{\text{cat}}\,.$$

– Chỉ số RMSE đạt được: 3202.6772.
– Chỉ số MAE đạt được: 1844.0081.
–> Hiệu suất tốt nhất trong 3 phương pháp.

6 Kết luận



Hình 10: So sánh chỉ số MAE

Mô hình	**MAE**	**RMSE**
CatBoost	1 846.81	3 199.20
XGBoost	1 903.07	3 296.63
Ensemble (0.35/0.65)	1 844.01	3 202.68

- Mô hình CatBoost cho kết quả MAE thấp nhất trong hai mô hình đơn lẻ, với MAE = 1 846.81 và RMSE = 3 199.20.
- XGBoost có độ chính xác hơi kém hơn, MAE = 1 903.07 và RMSE = 3 296.63.
- Mô hình kết hợp CatBoost và XGBoost (ensemble) đạt hiệu năng tốt nhất, giảm MAE xuống 1 844.01 và RMSE xuống 3 202.68 (giảm sai số khoảng 2–3% so với CatBoost).
- Kết quả cho thấy việc kết hợp dự đoán bù trừ giữa hai mô hình giúp tăng độ ổn định và chính xác, phù hợp cho bài toán dự báo giá ô tô.
- Hướng phát triển: tiếp tục tinh chỉnh siêu tham số, thử các thuật toán khác (LightGBM, Random Forest, SVR), áp dụng kỹ thuật stacking/blending, và xây dựng pipeline deploy để đưa mô hình vào ứng dụng thực tế.

References

Machine Learning Cơ Bản
Scikit-learn