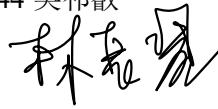


大數據專題報告

指導教授:林長鑒

學生:4110053144 吳祁叡

教授簽名:



競賽題目說明

競賽題目

Russian Car Plates Prices Prediction

題目說明

在本競賽中，將根據車牌號碼、註冊地區和註冊日期等資訊，預測俄羅斯車牌的價格。資料集包含了大量已知價格的俄羅斯車牌，其特徵包括車牌內容（如英文字母和數字組成）、地區編號、註冊日期等。目標是建立一個機器學習模型，能夠根據這些資訊準確預測尚未標價車牌的市場價值。

評估目標

本競賽以 ****SMAPE（對稱平均絕對百分比誤差，Symmetric Mean Absolute Percentage Error）**** 作為評估指標。SMAPE 會計算對測試集所有預測值與實際值的誤差，SMAPE 越低代表預測效果越佳。

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

from lightgbm import LGBMRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_absolute_error, make_scorer
from sklearn.impute import SimpleImputer
```

套件與工具說明

基礎資料處理與視覺化

- 1) pandas、numpy: 負責資料的載入、處理與運算。
- 2) matplotlib.pyplot、seaborn: 用於資料視覺化，協助進行資料探索與結果展示。

- 3) warnings: 用於管理 Python 執行過程中可能出現的警告訊息。

A. 機器學習模型

- lightgbm.LGBMRegressor、xgboost.XGBRegressor: 實現先進的梯度提升樹（GBDT）回歸模型。
- sklearn.ensemble.RandomForestRegressor、sklearn.ensemble.GradientBoostingRegressor: 隨機森林與梯度提升樹模型。
- sklearn.linear_model.Ridge、sklearn.linear_model.Lasso: 線性迴歸模型中的 Ridge（脊迴歸）及 Lasso（套索迴歸）方法。

B. 機器學習流程輔助

- sklearn.model_selection: 包含資料集切分（train_test_split）、交叉驗證（cross_val_score）、超參數搜尋（GridSearchCV）等。
- sklearn.preprocessing: 用於資料前處理，如標準化（StandardScaler）、類別特徵編碼（OneHotEncoder）。
- sklearn.compose.ColumnTransformer: 整合多種特徵處理流程。
- sklearn.pipeline.Pipeline: 將資料處理、模型訓練等流程組合成管線，提高程式可讀性與重現性。
- sklearn.impute.SimpleImputer: 處理缺失值。

C. 模型評估指標

- sklearn.metrics.mean_absolute_error、sklearn.metrics.make_scorer: 計算模型預測誤差，並自訂評估指標。

I. 資料讀取與來源

本研究所使用的資料集分為訓練集（train.csv）與測試集（test.csv）。資料以 CSV 格式儲存，透過 pandas 套件中的 read_csv() 函數進行讀取。

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

II. 資料初步檢查

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51635 entries, 0 to 51634
Data columns (total 4 columns):
#   Column   Non-Null Count  Dtype
---  ---
0    id      51635 non-null  int64
1   plate   51635 non-null  object
2    date   51635 non-null  object
3   price   51635 non-null  int64
dtypes: int64(2), object(2)
memory usage: 1.6+ MB
```

在資料讀取後，使用 pandas 的 info() 方法，快速檢視訓練資料集的整體結構。此步驟有助於了解每個欄位的資料型態（如 int、float、object），以及是否存在缺失值（null/NaN），並確認資料集的總筆數。

III. 資料描述性統計分析

為進一步了解訓練資料集的數值型欄位分布情形，本研究使用 pandas 的 describe() 方法來產生基本統計量。此方法可快速獲得各數值型欄位的平均值、標準差、最大值、最小值、四分位數等資訊，有助於初步掌握資料的整體分布與潛在異常值。

```
train.describe()
```

	id	price
count	51635.000000	5.163500e+04
mean	25818.000000	4.532253e+05
std	14905.884912	1.793287e+06
min	1.000000	2.000000e+04
25%	12909.500000	7.000000e+04
50%	25818.000000	1.500000e+05
75%	38726.500000	3.600000e+05
max	51635.000000	1.000000e+08

大多數價格集中在較低區間，但也存在明顯的高價異常值（例如最大值 1 億）。

IV. 缺失值檢查

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

```
0
id      0
plate    0
date     0
price    0
dtype: int64
```

V. 特徵工程設計

為提升機器學習模型對於資料的解讀與預測能力，本研究針對原始資料進行特徵萃取與轉換。以車牌資料與日期為例，自訂 extract_plate_features() 函數，將原始資料中的車牌號碼與日期拆解並轉換為多項數值與類別特徵。

A. 車牌特徵

- prefix: 萃取車牌開頭 1 至 3 個英文字母（表示車輛類型或所屬單位）
- number: 萃取中間 3 位數字，轉換為整數
- region_code: 萃取車牌末端 2 至 3 位地區碼，轉換為整數

B. 日期特徵

- year、month、day: 分別表示車牌註冊的年份、月份與日期
- weekday: 對應星期幾（0 為星期一）
- day_of_year: 當年度的第幾天

C. 車牌特殊指標

- is_gov: 判斷是否屬於政府車牌，透過 prefix 與指定清單比對
- is_repdigit: 判斷車牌數字是否為重複（如 111、222），常被視為特殊號碼
- is_moscow、is_spb: 標記是否為莫斯科（77）、聖彼得堡（78）地區車牌

D. 程式流程說明

該函數會：

- 輸入含有 plate 與 date 欄位的 DataFrame。
- 對 date 欄位進行型別轉換與多重時間特徵提取。
- 使用正則表達式從 plate 欄位擷取各部分。
- 依據車牌號碼規則與特殊類型定義，產生多種標記類特徵。
- 輸出包含新增特徵的 DataFrame。

透過上述特徵工程設計，模型可利用更多結構化資訊，提升預測準確度。

```
def extract_plate_features(df):
    |
    | df = df.copy()
    |
    | # 將 date 欄位轉為 datetime 類型，方便後續提取時間特徵
    | df['date'] = pd.to_datetime(df['date'])
    |
    | # -----
    | # 擷取車牌組成部分
    | # -----
    | # prefix: 擷取開頭 1-3 個英文字母
    | df['prefix'] = df['plate'].str.extract(r'^([A-ZA-Z]{1,3})')[0]
    |
    | # number: 擷取中段連續 3 位數字，並轉為整數
    | df['number'] = df['plate'].str.extract(r'(\d{3})')[0].astype(int)
    |
    | # region_code: 擷取末尾 2-3 位數字，並轉為整數
    | df['region_code'] = df['plate'].str.extract(r'(\d{2,3})$')[0].astype(int)
    |
    | # -----
    | # 產生日期相關特徵
    | # -----
    | # year: 年份
    | df['year'] = df['date'].dt.year
    | # month: 月份 (1-12)
    | df['month'] = df['date'].dt.month
    | # day: 日 (1-31)
    | df['day'] = df['date'].dt.day
    | # weekday: 星期幾 (0=Mon, 6=Sun)
    | df['weekday'] = df['date'].dt.weekday
    | # day_of_year: 當年的天
    | df['day_of_year'] = df['date'].dt.dayofyear
    |
    | # -----
    | # 車牌特殊類別指標
    | # -----
    | # is_gov: 是否為政府車牌 (前綴屬於設定清單)
    | df['is_gov'] = df['prefix'].isin(supplement.GOVERNMENT_CODES).astype(int)
    |
    | # is_repdigit: 數字是否為重複數字 (如111, 222)
    | df['is_repdigit'] = (
    |     (df['number'].astype(str).str[0] == df['number'].astype(str).str[1]) &
    |     (df['number'].astype(str).str[1] == df['number'].astype(str).str[2])
    | ).astype(int)
    |
    | # -----
    | # 熱門地區車牌標記
    | # -----
    | # is_moscow: 地區碼為 77
    | df['is_moscow'] = (df['region_code'] == 77).astype(int)
    | # is_spb: 地區碼為 78
    | df['is_spb'] = (df['region_code'] == 78).astype(int)
    |
    | # 回傳新增特徵後的 DataFrame
    |
    | return df
```

E. 特徵工程應用

前述特徵工程函式 `extract_plate_features()`，會被套用至訓練資料與測試資料。

```
# Apply feature engineering
train = extract_plate_features(train)
test = extract_plate_features(test)
```

VI. 特徵選擇與目標變數設定

為進行監督式學習，本研究將資料中的「特徵」與「目標變數」明確區分。具體做法如下：

- **特徵變數 (Features)**：選取經過特徵工程處理後、與預測任務直接相關的欄位，排除不具預測意義或為標識用途的欄位，包括 id、plate、date、price。以 `train.drop([...], axis=1)` 取得所有預測用特徵，命名為 X，測試集同理命名為 `X_test`。
- **目標變數 (Target)**：以 `train['price']` 取出目標欄位（即要預測的價格），命名為 y。

```
# Define features and target
X = train.drop(['id', 'plate', 'date', 'price'], axis=1)
y = train['price']
X_test = test.drop(['id', 'plate', 'date', 'price'], axis=1)
```

VII. 特徵型態區分

資料中的特徵依性質可分為「類別型特徵」與「數值型特徵」。不同型態的特徵在前處理與建模過程中需採用不同處理策略。

- **類別型特徵 (Categorical Features)**：如 prefix(車牌前綴)、region_code(地區碼)，屬於分類屬性，需進行編碼轉換。
- **數值型特徵 (Numerical Features)**：其餘經特徵工程產生且不屬於類別型的欄位（如 number、year、month、is_gov 等），均視為數值型特徵。

```
# Identify categorical and numerical features
categorical_features = ['prefix', 'region_code']
numerical_features = [col for col in X.columns if col not in categorical_features]
```

VIII. 資料視覺化

A. 目標變數（價格）分布視覺化

為深入了解目標變數 `price` 的分布情形，本研究以直方圖對價格數據進行視覺化分析。考慮到價格資料存在極端高值，分布高度偏態，特別針對常見價格區間進行放大觀察。

1) 9.1 最高頻率的價格區間：首先，統計各區間（bin）的最大頻次，以便設定合適的 y 軸範圍進行視覺化。

```
counts, bin_edges = np.histogram(train['price'], bins=50)
print("各 bin 最大頻次: ", counts.max())
```

2) 9.2 常見價格區間分布（頻率 0-50）：設定 y 軸上限為 50，聚焦於大部分「常見」價格區間，以觀察價格較為集中的區段。

```
plt.figure(figsize=(12, 6))
sns.histplot(train['price'], bins=50, kde=False, color='skyblue', edgecolor='black')
plt.title('Price Distribution (Y-axis zoomed to 0-50)')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.ylim(0, 50) # 只看 0 到 50 的頻次
plt.show()
```

3) 9.3 熱門價格小峰值分析（頻率 0-10）：若需更細緻地觀察「最熱門」的小峰值區段，可將 y 軸上限再下調至 10。

```
plt.figure(figsize=(12, 6))
sns.histplot(train['price'], bins=50, kde=False, color='salmon', edgecolor='black')
plt.title('Price Distribution (Y-axis zoomed to 0-10)')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.ylim(0, 10) # 只看 0 到 10 的頻次
plt.show()
```

4) 9.4 分析說明：由視覺化結果可知，`price` 分布極為偏態，大多數價格集中於低價區間，而高價部分雖佔少數，但對分布有明顯拉長（右尾）的現象。

a) 第一張圖：Y 軸上限 50（*Price Distribution (Y-axis zoomed to 0-50)*）：

- **描述：**這張圖將 Y 軸最高頻率設定為 50，聚焦於大部分資料的「常見」價格分布區間。
- **觀察重點：**
 - 價格資料分布極為偏態（右偏態），大多數車牌的價格集中在左側（低價區間）。
 - 低於約 0.1 億（即 100,000,000 以下）的價格區間內，有明顯的高頻分布。

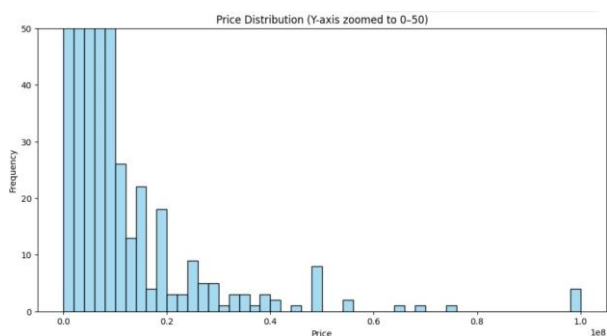
- 當價格提升至 0.2 億、0.3 億甚至更高時，頻率迅速降低，僅剩少數車牌落在高價區間。
- 圖的最右側可見極少數「天價」車牌，為潛在異常值或特殊號碼。

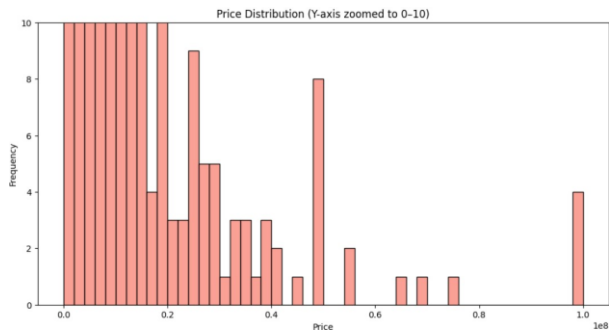
b) 第二張圖：Y 軸上限 10（*Price Distribution (Y-axis zoomed to 0-10)*）：

- **描述：**這張圖將 Y 軸最高頻率進一步下調至 10，更細緻地觀察罕見但價格極高的車牌分布。
- **觀察重點：**
 - 可以看到高價車牌的分布型態，儘管每個價格區間的頻數都極低，但分布範圍極廣。
 - 在高價區間（約 0.2 億至 1 億）仍能觀察到一些小型峰值，顯示雖然數量極少，但高價車牌的存在並非偶然，可能有其市場需求或特殊性。
 - 右側「天價」車牌（接近 1 億）依然清晰可見。

c) 綜合解讀：

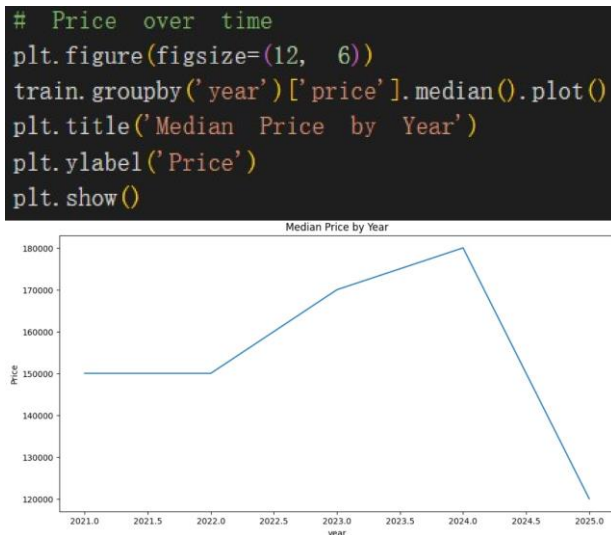
- **分布特性：**
 - 數據高度右偏，大部分車牌屬於「平價」或「常見」價格區間，僅少數特殊車牌價格極高，極可能為特殊號碼或具有特定收藏價值。





IX. 價格隨時間變化趨勢分析

為探討價格（price）是否隨年份變動而產生趨勢，本研究將資料依年份（year）分組，計算各年中位數價格，並繪製折線圖進行視覺化。

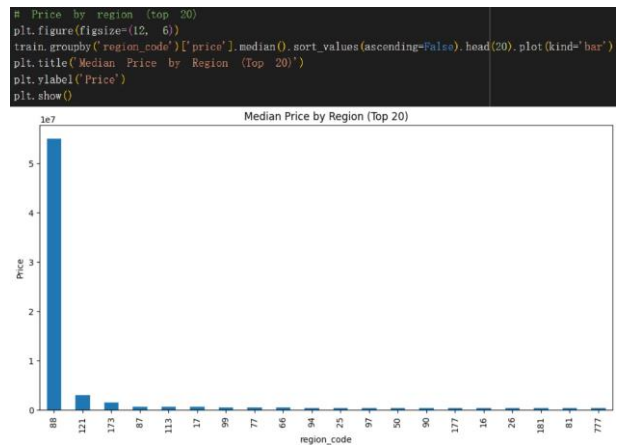


a) 分析說明:

- 1) 圖中每一點代表該年度所有資料的價格中位數，能反映不同年度價格的整體趨勢。
- 2) 若觀察到價格逐年上升或下降，可能與經濟環境、政策或特定市場事件相關，有助於後續模型納入時間因素進行預測。
- 3) 透過中位數而非平均值，可以避免極端值（如高價異常點）對整體趨勢判斷的影響，使趨勢線更具代表性。

X. 各地區價格分布分析（TOP 20）

為瞭解不同地區（region_code）之間的價格差異，本研究計算每個地區的價格中位數，並選取前 20 個價格最高的地區繪製長條圖。

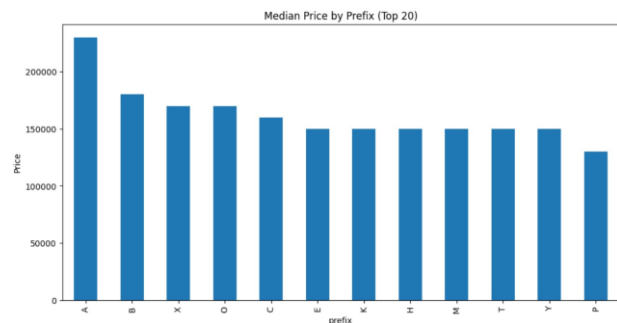


XI. 車牌前綴（PREFIX）與價格關係分析

為分析車牌前綴（prefix）對價格的影響，本研究將資料依據 prefix 分組，計算每一類的價格中位數，並取前 12 種價格最高的前綴，繪製長條圖。

a) 分析說明:

- X 軸為車牌前綴（prefix），Y 軸為該前綴的價格中位數。
- 從圖中可辨識哪些車牌前綴的車輛價值較高，這通常與車種、車齡、用途（如政府、公務、特殊用途）等因素有關。
- 若特定前綴的價格明顯高於其他類型，模型可考慮將其設為重要分類特徵。



XII. 數值型特徵相關性分析

為瞭解各數值型特徵與目標變數（price）之間的關聯性，本研究計算了各數值型特徵間的皮爾森相關係數，並以熱力圖（heatmap）視覺化呈現。



a) 分析說明：

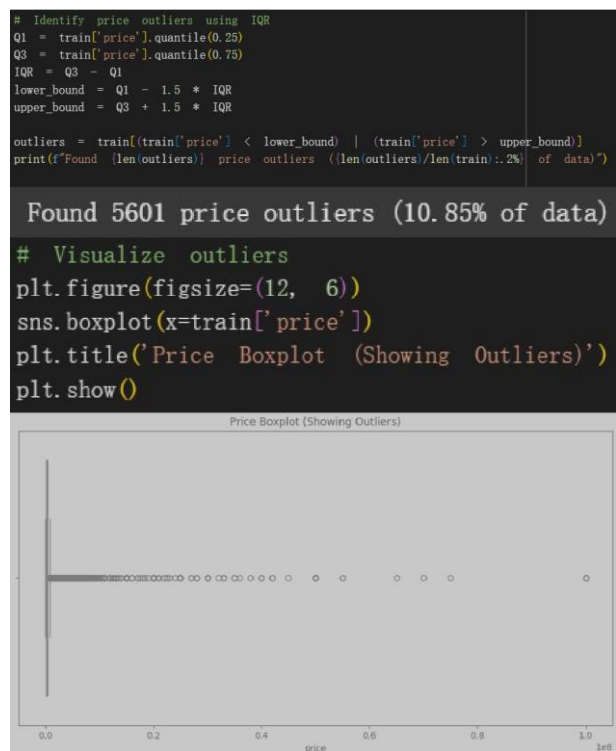
- 熱力圖顯示各特徵彼此之間及與價格的相關性，顏色由紅（正相關）到藍（負相關）。
- 觀察結果發現，所有特徵與 **price** 的相關性普遍較弱，最大相關係數（如 **is_repdigit**）僅約 **0.13**，表示沒有明顯的線性相關特徵。
- 部分時間特徵（如 **year**、**month**、**day_of_year**）間有較強的負相關或正相關，這是因資料本身的日期結構所致。
- 熱力圖有助於判斷哪些特徵較有機會提升模型表現，並作為後續特徵選擇與工程的參考依據。

XIII. 價格異常值檢查與統計（IQR 法）

為提升模型穩健性並減少極端值對預測造成的影響，本研究利用 **四分位距（Interquartile Range, IQR）** 法檢查目標變數 **price** 的異常值（outliers）。

A. 步驟說明

- 計算第一四分位數（**Q1, 25%**）及第三四分位數（**Q3, 75%**）。
- 四分位距 $IQR = Q3 - Q1$ 。
- 設定異常值判斷上下界：
 - 下界： $Q1 - 1.5 \times IQR$
 - 上界： $Q3 + 1.5 \times IQR$
- 標記超出上述範圍的資料為異常值。



B. 移除離群值

```
train = train[(train['price'] >= lower_bound) & (train['price'] <= upper_bound)]
```

XIV. 資料前處理流程設計

為提升模型的學習效能及泛化能力，本研究針對數值型與類別型特徵設計不同的前處理管線（pipeline），並以 **ColumnTransformer** 統一應用至整個資料集。

A. 數值型特徵處理（*numeric_transformer*）

- 缺失值補齊：以中位數（**median**）填補數值型特徵的缺失值，降低極端值影響。
- 標準化：利用 **StandardScaler** 進行 Z 分數標準化。

B. 類別型特徵處理（*categorical_transformer*）

- 缺失值補齊：以眾數（**most_frequent**）填補類別型特徵的缺失值。
- 獨熱編碼：採用 **OneHotEncoder**，參數 **handle_unknown='ignore'** 可忽略測試集中出現但訓練集未見的新類別。

C. 前處理流程統整

使用 `ColumnTransformer` 統一指定數值型與類別型特徵的處理方式，確保整體資料前處理流程標準化、模組化。

```
# Define preprocessing
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)])
```

XV. 評估指標：SMAPE（對稱平均絕對百分比誤差）

為符合競賽評分標準，本研究自訂 SMAPE（Symmetric Mean Absolute Percentage Error）作為模型預測誤差指標。

A. 指標定義

SMAPE 衡量預測值與實際值之間的對稱性百分比誤差，介於 0 - 200，數值越低表示預測準確度越高。SMAPE 能有效處理目標值範圍廣、極端值多的資料情境。

$$\text{SMAPE} = \frac{100}{n} \sum_{i=1}^n \frac{2|y_{\text{pred},i} - y_{\text{true},i}|}{|y_{\text{true},i}| + |y_{\text{pred},i}|}$$

```
# Define SMAPE scoring (competition metric)
def smape(y_true, y_pred):
    return 100/len(y_true) * np.sum(2 * np.abs(y_pred - y_true) / (np.abs(y_true) + np.abs(y_pred)))

smape_scoring = make_scorer(smape, greater_is_better=False)
```

B. 實務應用

透過該指標可公平比較不同模型或調參結果，聚焦於「預測誤差」的降低。

XVI. 多模型比較架構

為全面評估不同機器學習方法在本問題上的表現，本研究選用多種主流回歸模型，並統一設定隨機種子（`random_state=42`）以確保實驗結果可重現。各模型包括：

- 隨機森林回歸（`RandomForestRegressor`）：集成式樹狀模型，擅長捕捉非線性與高維特徵關係。

- 梯度提升樹回歸（`GradientBoostingRegressor`）：逐步優化、組合弱學習器強化預測能力。
- Ridge 回歸（`Ridge`）：加入 L2 正則化以抑制過度擬合。
- Lasso 回歸（`Lasso`）：加入 L1 正則化，兼具特徵選擇效果。
- LightGBM 回歸（`LightGBMRegressor`）：高效梯度提升樹框架，適合大規模資料。
- XGBoost 回歸（`XGBRegressor`）：業界常用高效梯度提升樹實現。

```
models = {
    'Random Forest': RandomForestRegressor(random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'Ridge': Ridge(random_state=42),
    'Lasso': Lasso(random_state=42),
    'LightGBM': LightGBMRegressor(random_state=42, n_estimators=500, learning_rate=0.05),
    'XGBoost': XGBRegressor(random_state=42, n_estimators=500, learning_rate=0.05, tree_method='hist'),
}
```

本架構便於後續進行多模型訓練、驗證及 SMAPE 指標下的效能比較。

```
models = {
    'Random Forest': RandomForestRegressor(random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'Ridge': Ridge(random_state=42),
    'Lasso': Lasso(random_state=42),
    'LightGBM': LightGBMRegressor(random_state=42, n_estimators=500, learning_rate=0.05),
    'XGBoost': XGBRegressor(random_state=42, n_estimators=500, learning_rate=0.05, tree_method='hist'),
}
```

XVII. 多模型交叉驗證與比較

本研究採用三折交叉驗證（3-fold cross-validation），評估不同回歸模型的 SMAPE 表現，並保持一致的資料處理流程。

A. 主要結果摘要

模型	Mean SMAPE	標準差
Random Forest	59.60	±0.26
XGBoost	71.30	±0.40
LightGBM	72.36	±0.69
Gradient Boosting	76.31	±0.86
Ridge	96.50	±0.33
Lasso	96.49	±0.38

B. 分析與討論

- 隨機森林回歸表現最佳，SMAPE 約 59.6。
- XGBoost 與 LightGBM 表現相近，次於隨機森林。

- Ridge、Lasso 表現最差，說明資料複雜性高，需用非線性方法捕捉特徵。
- 交叉驗證標準差小，代表評估穩定。

C. 額外觀察

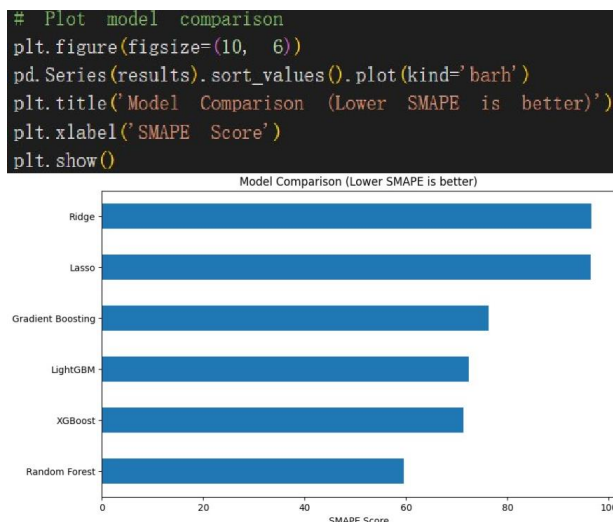
- Lasso 出現 `ConvergenceWarning`，可能需調整 `max_iter` 或進行特徵選擇/縮放。
- LightGBM、XGBoost 執行穩定，並有多執行緒與記憶體調適建議。

XVIII. 多模型 SMAPE 表現視覺化比較

以橫條圖 (barh) 方式繪製 SMAPE 排名，直觀呈現模型表現差異。

a) 分析說明：

- SMAPE 越低表示預測越佳。
- Random Forest 領先，XGBoost 與 LightGBM 次之。
- Ridge、Lasso 顯示表現較差，推測資料包含複雜的非線性關係。



XIX. 最終模型建構與完整訓練流程

根據多模型交叉驗證評估結果，選出 SMAPE 最佳的隨機森林回歸模型，並與前處理流程串接成最終 pipeline。

A. 最佳模型選取

程式自動取得評分最佳模型，整合至預測流程。

```
best_model_name = min(results, key=results.get)
best_estimator = models[best_model_name]
print(f"使用最佳模型: {best_model_name}")
```

B. 建立最終預測流程

將資料前處理 (preprocessor) 與最佳模型 (RandomForestRegressor) 串接為同一 pipeline。

```
final_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('model', best_estimator)
])
```

C. 完整資料訓練

以完整訓練資料 (X, y) 重新訓練最終模型，確保模型充分學習所有特徵。

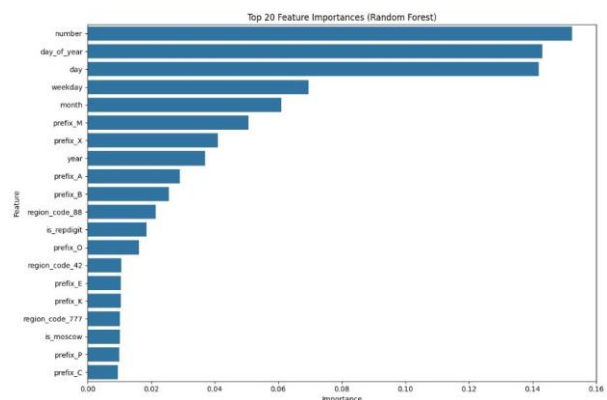
```
final_pipeline.fit(X, y)
```

XX. 特徵重要性分析 (FEATURE IMPORTANCE)

使用 Random Forest 模型，可評估各特徵對於最終預測結果的影響力。此分析有助於後續的特徵工程與模型簡化。

A. 分析流程

- 由 `final_pipeline` 取出模型的 `feature_importances_`。
- 合併數值型與類別型特徵名稱。
- 建立 DataFrame，依據 importance 降冪排序，繪製前 20 名特徵圖。



XXI. 結果解讀

- 從圖中可見，number（車牌數字）、day_of_year（年度天數）、day（日期）、weekday（星期幾）、month（月份）等特徵為最具預測力的前五大特徵。
- 某些 OneHot 編碼的類別型特徵（如 prefix_M、prefix_X），以及特定地區碼（如 region_code_88），也顯示出不容忽視的重要性。
- 這些結果顯示，價格預測不僅與車牌數字相關，亦受到註冊日期、特殊號碼、地區與部分類別指標顯著影響。

XXII. 測試集預測與結果提交







訓練完成的最終預測流程（pipeline）可直接對測試集進行預測，產生預測結果並依指定格式輸出 CSV 檔案，以便後續 kaggle 競賽提交。

```
predictions = final_pipeline.predict(X_test)

# 建立 submission DataFrame
submission = pd.DataFrame({
    'id': test['id'],          # 確定 test DataFrame 裡有 id 欄
    'price': predictions
})

#將結果存成 CSV
submission.to_csv('submission.csv', index=False)
```

XXIII. SUBMISSION 結果

YOUR RECENT SUBMISSION						
		submission (7).csv			Score: 60.7837	
		Submitted by RayWu576 · Submitted a day ago			Public score: 58.1192	
408	- 2	Nouredine RIDA		60.7837	1	8d
409	- 2	Reynaldi Holtrop		60.7837	1	6d
410	+ 3	Ben Hamner		60.8912	14	2mo
411	—	Farhan Wicaksono		60.9125	6	16d
412	+ 10	Md Arafat		61.1955	1	20d