

Data Pre-processing Scripts

概述

- 資料來源: Kaggle - London House Price Prediction
- 訓練資料: 266,325 筆, 17 個特徵
- 測試資料: 16,547 筆, 16 個特徵
- 原始完整程式碼: [🔗 114552029_DL_lab1](#)

步驟一：原始資料分布視覺化

程式繪製了 6 個主要特徵的分布直方圖：

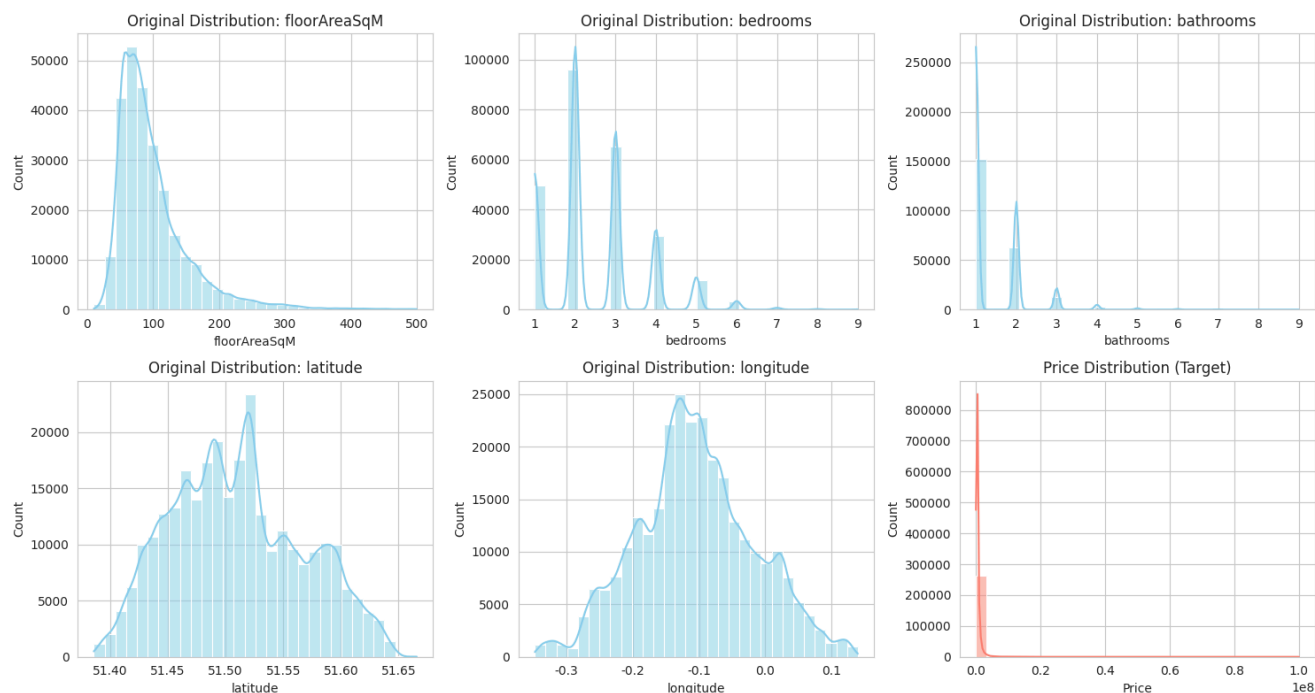
1. **floorAreaSqM** (房屋面積)
 - 觀察：明顯的右偏分布 (right-skewed)，大部分房屋集中在 50-150 平方公尺，但存在大於 400 的極端離群值 (outliers)。
2. **bedrooms** (臥室數量)
 - 觀察：數據集中在 2-4 間，峰值在 3 間。超過 6 間的樣本非常稀少，可視為離群值。
3. **bathrooms** (浴室數量)
 - 觀察：絕大多數為 1-3 間，超過 4 間的樣本極為罕見，呈現高度右偏。
4. **latitude** (緯度)
 - 觀察：接近常態分布，集中在 51.40 至 51.65 度之間，反映樣本集中於倫敦市區，無明顯離群值。
5. **longitude** (經度)
 - 觀察：呈現雙峰分布 (bimodal)，可能反映倫敦東西部房產分布的地理差異。
6. **price** (房價 - 目標變數)
 - 觀察：極端右偏分布，大部分房價偏低，但存在極少數價格極高的豪宅 (如 1e8 附近)，證實了 Log 轉換的必要性。

對應程式碼

```
def plot_original_distributions():  
    fig, axs = plt.subplots(2, 3, figsize=(15, 8))  
    axs = axs.ravel()  
    # 合併訓練與測試資料以觀察整體分布  
    combined_features = pd.concat([  
        train_df.drop(['price', 'ID'], axis=1, errors='ignore'),  
        test_df.drop('ID', axis=1, errors='ignore')  
    ], axis=0)  
    key_features = ['floorAreaSqM', 'bedrooms', 'bathrooms', 'latitude', 'longitude']  
    for i, feature in enumerate(key_features):  
        if feature in combined_features.columns:  
            sns.histplot(combined_features[feature], bins=30, kde=True,  
                          ax=axs[i], color='skyblue')  
            axs[i].set_title(f'Original Distribution: {feature}')  
    # 房價分布 (僅訓練集有)  
    sns.histplot(train_df['price'], bins=30, kde=True,  
                  ax=axs[5], color='salmon')  
    axs[5].set_title('Price Distribution (Target)')
```

```
plt.tight_layout()
plt.show()
```

執行結果

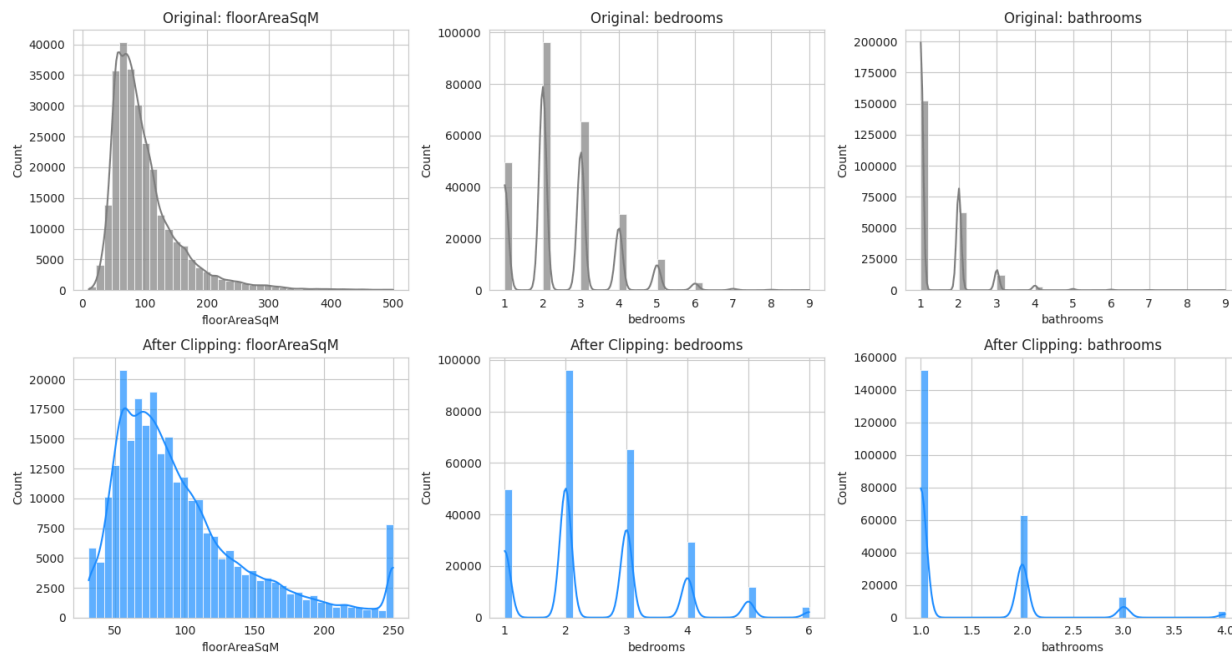


關鍵發現

- 數值特徵(面積、房數)存在明顯離群值，需進行裁剪處理。
 - 目標變數 price 高度右偏，必須進行 Log 轉換以穩定模型訓練。
 - 地理特徵分布合理，可用於後續的空間聚類分析。
-

步驟二：離群值處理 (Winsorization/Clipping)

執行結果



Step 2: Applying outlier treatment with before/after comparison...

→ Clipping removes extreme outliers and stabilizes distributions

處理前後對比

Before Clipping (處理前):

- floorAreaSqM: 存在 >400 的極端長尾。
- bedrooms: 出現 7-9 間的異常值。
- bathrooms: 存在超過 6 間的不合理數據。

After Clipping (處理後):

- floorAreaSqM: 限制在 20-250 範圍，再依 1%-99% 分位數截斷，分布更集中。
- bedrooms: 限制在 1-8 間，移除了 >8 的極端值。
- bathrooms: 限制在 1-6 間，分布更合理。

對應程式碼

```
def winsorize_clip(df: pd.DataFrame):  
    df = df.copy()  
    # 面積處理: 硬截斷 + 分位數截斷  
    if 'floorAreaSqM' in df.columns:  
        df['floorAreaSqM'] = df['floorAreaSqM'].clip(lower=20, upper=250)  
        q_low, q_high = df['floorAreaSqM'].quantile([0.01, 0.99])  
        df['floorAreaSqM'] = np.clip(df['floorAreaSqM'], q_low, q_high)  
    # 臥室處理  
    if 'bedrooms' in df.columns:  
        df['bedrooms'] = df['bedrooms'].clip(lower=1, upper=8)  
        q_low, q_high = df['bedrooms'].quantile([0.01, 0.99])  
        df['bedrooms'] = np.clip(df['bedrooms'], q_low, q_high)  
    # 浴室處理  
    if 'bathrooms' in df.columns:  
        df['bathrooms'] = df['bathrooms'].clip(lower=1, upper=6)
```

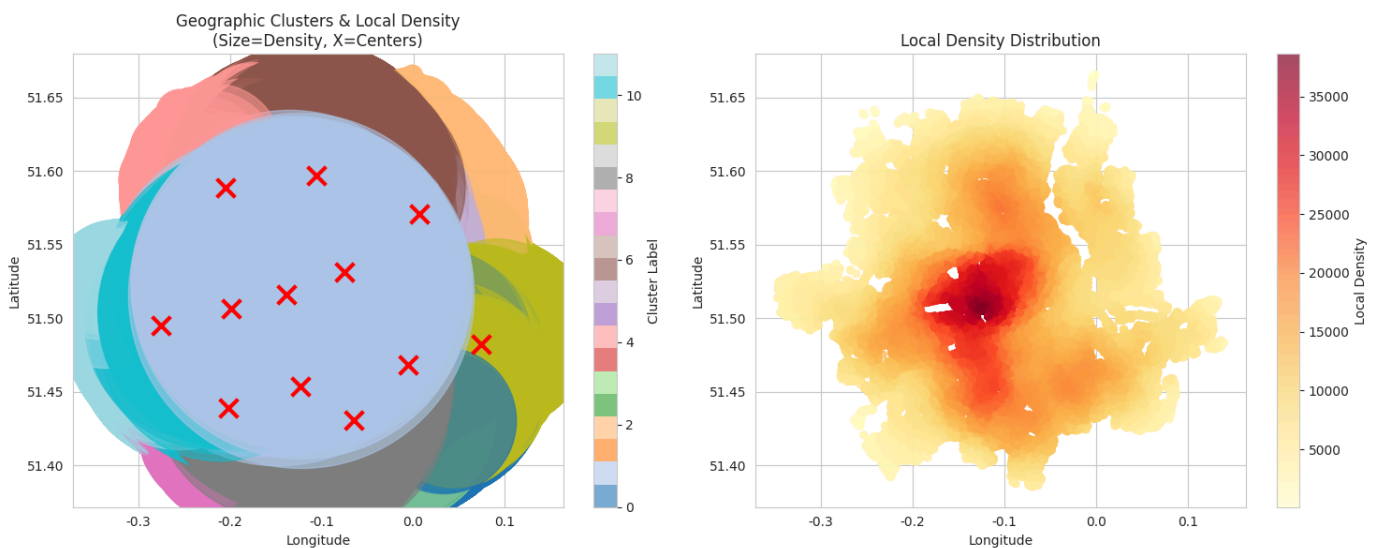
```
q_low, q_high = df['bathrooms'].quantile([0.01, 0.99])
df['bathrooms'] = np.clip(df['bathrooms'], q_low, q_high)
return df
```

為何執行此步驟？

1. 提升模型穩健性: 極端值會導致梯度計算不穩定, 影響模型收斂。
2. 反映真實市場: 移除不合理或極度稀有的樣本(如 9 房、500 平方公尺), 讓模型專注於學習一般市場的規律。

步驟三: 地理聚類與密度特徵

執行結果



Step 3: Adding geographic clustering and density features...

→ Geographic features capture spatial patterns and neighborhood effects

Feature count: 19 → 37 (+18 new features)

視覺化分析

左圖: 地理聚類 (Geographic Clusters & Local Density) ¹⁵

- 使用 KMeans 演算法將所有房產依經緯度分為 12 個地理群集(不同顏色)。
- 圖中紅色 'x' 標記為各群集的中心點。
- 散點的大小代表該位置的「局部密度」(geo_density), 可見中心區域的點較大, 代表房產密集。

右圖: 局部密度熱圖 (Local Density Distribution) ¹⁶

- 這是一張熱力圖, 顏色越深(偏紅)代表 3km 半徑內的房屋數量越多。
- 清楚顯示了倫敦的高密度住宅核心區(紅色)與中低密度的郊區(黃色)。

對應程式碼

```
def add_geo_cluster_features(df: pd.DataFrame, n_clusters=12, radius_km=3.0):
    df = df.copy()
    geo_cols = ['latitude', 'longitude']
    if all([col in df.columns for col in geo_cols]):
        points = df[geo_cols].values
        # 1. KMeans 地理聚類
```

```

kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
labels = kmeans.fit_predict(points)
df['geo_cluster'] = labels
# 2. 計算到群集中心的距離
centers = kmeans.cluster_centers_
df['center_distance'] = np.sqrt(np.sum((points - centers[labels]) ** 2, axis=1))
# 3. BallTree 計算局部密度 (3km半徑內鄰居數)
tree = BallTree(np.deg2rad(points), metric='haversine')
earth_radius = 6371.0 # 地球半徑(km)
dist_rad = radius_km / earth_radius
counts = tree.query_radius(np.deg2rad(points), r=dist_rad, count_only=True)
df['geo_density'] = counts
return df

```

新增特徵說明

- **geo_cluster**: 房產所屬的地理群集編號 (0-11), 此類別特徵能捕捉不同區域的房價基線。
- **center_distance**: 房產到其所屬群集中心的距離, 可反映其在該區域內的中心或偏僻程度。
- **geo_density**: 3km 半徑內的鄰居數量, 代表該社區的繁榮度或人口密度。

步驟四：進階特徵工程

新增特徵詳細列表

此步驟共新增 20 個特徵, 並刪除 2 個 (sale_year, sale_month), 淨增加 18 個特徵, 使特徵總數達到 37 個。

1. 時間相關特徵 (新增 3)

```

if 'sale_year' in df.columns:
    current_year = df['sale_year'].max()
    df['house_age'] = current_year - df['sale_year'] # 屋齡
    df['is_new_house'] = (df['house_age'] <= 5).astype(int) # 是否新屋
    df['age_squared'] = df['house_age'] ** 2 # 屋齡平方

```

2. 面積相關特徵 (新增 4)

```

if 'floorAreaSqM' in df.columns:
    df['area_log'] = np.log1p(df['floorAreaSqM']) # 面積對數
    df['area_squared'] = df['floorAreaSqM'] ** 2 # 面積平方
    df['area_per_bedroom'] = df['floorAreaSqM'] / (df['bedrooms'] + 1)
    df['area_per_bathroom'] = df['floorAreaSqM'] / (df['bathrooms'] + 1)

```

3. 房型組合與交互特徵 (新增 6)

```

if 'bedrooms' in df.columns and 'bathrooms' in df.columns:
    df['total_rooms'] = df['bedrooms'] + df['bathrooms']
    df['bed_bath_ratio'] = df['bedrooms'] / (df['bathrooms'] + 1)
    df['bedrooms_squared'] = df['bedrooms'] ** 2
    df['bed_area_interaction'] = df['bedrooms'] * df['floorAreaSqM']
if 'house_age' in df.columns and 'floorAreaSqM' in df.columns:
    df['area_age_interaction'] = df['floorAreaSqM'] * df['house_age']
if 'livingRooms' in df.columns and 'bedrooms' in df.columns:
    df['total_living_space'] = df['bedrooms'] + df['livingRooms']

```

4. 地理交互特徵 (新增 2)

```
if 'latitude' in df.columns and 'longitude' in df.columns:
    df['lat_lng_interaction'] = df['latitude'] * df['longitude']
    lat_mean = df['latitude'].mean()
    lng_mean = df['longitude'].mean()
    df['distance_from_center'] = np.sqrt(
        (df['latitude'] - lat_mean) ** 2 + (df['longitude'] - lng_mean) ** 2
    )
```

5. 季節特徵 (新增 3)

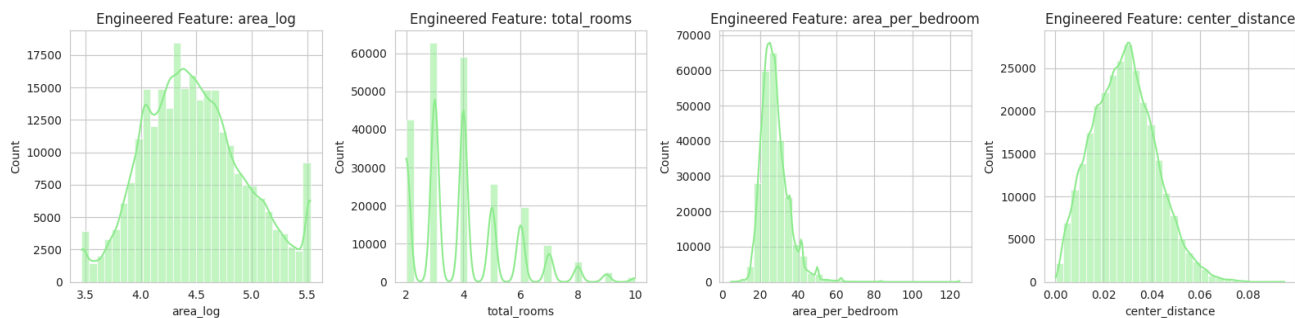
```
if 'sale_month' in df.columns:
    df['is_spring'] = df['sale_month'].isin([3, 4, 5]).astype(int)
    df['is_summer'] = df['sale_month'].isin([6, 7, 8]).astype(int)
    df['is_winter'] = df['sale_month'].isin([12, 1, 2]).astype(int)
```

6. 分箱特徵 (新增 2)

```
if 'floorAreaSqM' in df.columns:
    df['area_bin'] = pd.qcut(df['floorAreaSqM'], q=10, labels=False, duplicates='drop')
if 'house_age' in df.columns:
    df['age_bin'] = pd.qcut(df['house_age'], q=8, labels=False, duplicates='drop')
```

視覺化新特徵分布

執行結果



Step 4: Creating advanced engineered features...

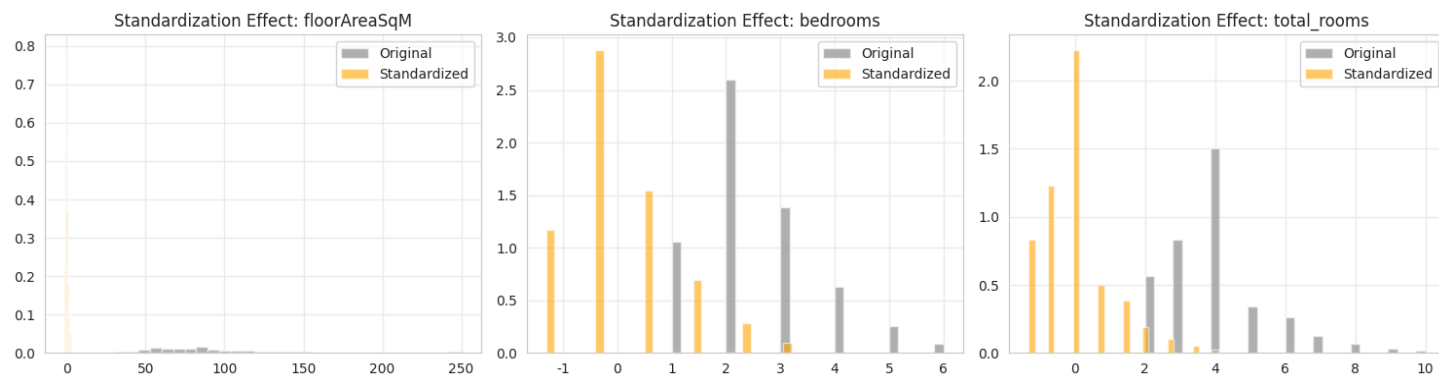
Feature count: 19 → 37 (+18 new features)

→ Engineered features provide additional predictive signals

1. **area_log**: 面積經 log1p 轉換後，分布更接近常態，有助於模型學習。
 2. **total_rooms**: 總房間數，提供了比單獨臥室或浴室更綜合的房型指標。
 3. **area_per_bedroom**: 每臥室平均面積，反映房屋的寬敞程度。
 4. **center_distance**: 到地理群集中心的距離，反映地理位置的優劣。
-

步驟五：最終預處理 (缺失值與標準化)

執行結果



Step 5: Final preprocessing (missing values, encoding)...

→ Standardization normalizes features for neural network training

1. 缺失值處理

- 數值型欄位 (num_cols): 使用該欄位的中位數 (**median**) 進行填補。中位數對離群值不敏感，是比平均值更穩健的選擇。
- 類別型欄位 (cat_cols): 先將 NaN 填補為 "Unknown" 字串，再使用 LabelEncoder 將所有類別 (如 'HDB', 'Condo') 轉換為 0, 1, 2... 等數值，以便神經網路讀取。

數值型欄位: 中位數填補

```
for col in num_cols:
    if all_df[col].isna().any():
        all_df[col] = all_df[col].fillna(all_df[col].median())
```

類別型欄位: 標籤編碼

```
for col in cat_cols:
    le = LabelEncoder()
    all_df[col] = le.fit_transform(all_df[col].fillna('Unknown').astype(str))
```

2. 標準化 (Standardization)

- **Original** (灰色): 特徵的數值範圍差異巨大 (例如 floorAreaSqM 範圍 0-250, 而 bedrooms 範圍 1-8)。
- **Standardized** (橘色): 所有特徵都被轉換為均值为 0、標準差為 1 的分布。
- 為何重要?: 此步驟對神經網路至關重要。它能防止數值範圍大的特徵 (如面積) 在梯度下降中主導權重更新，從而加速模型收斂並提高訓練穩定性。

標準化在 K-Fold 迴圈內部完成, 以防資料洩漏

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_fold)
X_val_scaled = scaler.transform(X_val_fold)
X_test_scaled = scaler.transform(X_test)
```

步驟六：目標變數轉換 (Target Transformation)

Log 轉換的必要性

```
y_log = np.log1p(y) # log(1 + price)
```

原因:

- 修正偏態: 如步驟一的圖表¹⁹所示, price 分布極端右偏。Log 轉換能使其分布接近常態, 更符合多數迴歸模型的假設。
- 穩定損失函數: 若不轉換, 極端高價的豪宅樣本會產生巨大的誤差 (Loss), 主導整個訓練過程, 使模型過度關注這些稀有樣本。
- 優化指標: 在 Log 尺度上計算 RMSE (RMSLE), 意味著模型更關注「相對誤差」(百分比) 而非「絕對誤差」(金額), 這在房價預測中通常更合理。

資料前處理總結

處理流程

原始資料 (266,325 × 17)

↓

步驟 1: 視覺化 (發現離群值與偏態)

↓

步驟 2: 離群值裁剪 (Clipping + Winsorization)

↓

步驟 3: 地理聚類 (KMeans + BallTree, +3 特徵)

↓

步驟 4: 特徵工程 (新增 20, 刪除 2, 淨 +18 特徵)

↓

步驟 5: 缺失值處理 (中位數/編碼) & 標準化

↓

步驟 6: 目標轉換 (np.log1p)

↓

最終訓練集 (特徵數 37)

Final dataset info:

Features: 37

Training samples: 266325

Test samples: 16547

關鍵成果

- 特徵數量: 17 → 37 (淨增加 18 個)。
- 數值穩定: 透過離群值處理、Log 轉換和標準化, 確保了神經網路訓練的穩定性與收斂速度。
- 特徵豐富度: 透過地理空間分析與交互特徵, 為模型提供了捕捉複雜非線性關係的基礎。