台鐵不動產租金預測模型

以下是這份 CatBoost 預測「每月租金」的完整程式碼的 **逐段說明與每行詳細解釋**：

第 1 段：匯入套件與圖表中文字設定

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from catboost import CatBoostRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import shap
import json
```

- `pandas` 和 `numpy`：資料處理用。
- `matplotlib.pyplot`，`seaborn`：繪圖用。
- `CatBoostRegressor`：使用 CatBoost 建立回歸模型。
- `train_test_split`，`r2_score`：資料切分與模型評估用。
- `shap`：模型可解釋性視覺化工具。
- `json`：讀取 `.json` 檔案格式。

```
plt.rcParams['font.family'] = 'Microsoft JhengHei'
plt.rcParams['axes.unicode_minus'] = False
```

- 設定 matplotlib 中文顯示為「微軟正黑體」。
- 避免座標軸上的負號顯示錯誤。

第 2 段：載入與清洗原始資料

```
file_path =
"C:/Users/user/Desktop/taiwan_railway_rent_predictor/taiwan_rent_data.json"
with open(file_path, "r", encoding="utf-8-sig") as f:
    data = json.load(f)
df = pd.DataFrame(data)
```

- 設定檔案路徑並讀取 JSON 檔案資料。
- 將 JSON 格式轉為 `DataFrame`。

```
df["建物面積"] = pd.to_numeric(df["建物面積"], errors="coerce")
df["每月租金"] = pd.to_numeric(df["每月租金"], errors="coerce")
```

```
df["總樓層數"] = pd.to_numeric(df["總樓層數"], errors="coerce")
df["租期屆滿"] = pd.to_datetime(df["租期屆滿"], errors="coerce")
```

- 將文字轉換為數值或時間格式，如果無法轉換則變成 NaN。

```
df = df[["縣市", "實際用途", "建物面積", "構造", "總樓層數", "建物現況", "房屋座落", "租期屆滿", "每月租金"]]
```

- 篩選出要分析的欄位，排除不需要的欄位。

```
df = df.dropna(subset=["建物面積", "每月租金"])
df = df[df["建物面積"] > 0]
df = df[df["每月租金"] < 300000]
```

- 去除建物面積或租金為缺值的資料。
- 篩除面積小於等於 0 或租金過高（>30萬）的異常值。

第 3 段：新增剩餘租期（月）

```
today = pd.Timestamp.today()
df["剩餘租期(月)"] = (df["租期屆滿"] - today).dt.days // 30
df["剩餘租期(月)"] = df["剩餘租期(月)"].fillna(0).astype(int)
```

- 使用今天日期與租期屆滿日相減，計算剩餘天數，並轉換為「整數月」。
- 若租期是空值，就填 0。

第 4 段：從地址中提取「區名」

```
df["區名"] = df["房屋座落"].str.extract(r"(..區)")
df["區名"] = df["區名"].fillna("未知區")
```

- 使用正規表示式從「房屋座落」中擷取區名，例如「大安區」、「信義區」。
- 若無法擷取到，填為「未知區」。

第 5 段：加入區域經緯度快取表（手動建立）

```
latlon_cache = pd.DataFrame([
    ["台北市", "大安區", 25.026, 121.543],
    ...
], columns=["縣市", "區名", "lat", "lon"])
```

- 建立手動經緯度快取表（避開 API 限制與錯誤），提供主要縣市的區名對應的緯度與經度。

```
df = df.merge(latlon_cache, on=["縣市", "區名"], how="left")
```

- 根據縣市與區名合併緯度與經度資料到原始 `df`。

第 6 段：對租金做 log 轉換（緩和極端值）

```
df["log_租金"] = np.log1p(df["每月租金"])
```

- 使用 `log1p()` 計算 `log(1 + x)`，讓模型更容易擬合且穩定。

第 7 段：訓練初始模型來剔除高殘差資料

```
X_raw = df[["縣市", "實際用途", "建物面積", "構造", "總樓層數", "建物現況",
"區名", "剩餘租期(月)", "lat", "lon"]]
y_raw = df["log_租金"]
cat_features = ["縣市", "實際用途", "構造", "建物現況", "區名"]
```

- 定義原始特徵與標籤，以及類別型特徵。

```
model_init = CatBoostRegressor(verbose=0, random_state=42)
model_init.fit(X_raw, y_raw, cat_features=cat_features)
```

- 建立初始模型（不輸出訓練過程），訓練用來判斷殘差。

```
y_pred_raw = model_init.predict(X_raw)
residuals_raw = y_raw - y_pred_raw
std = residuals_raw.std()
```

- 計算預測值與實際值的差（殘差），再求其標準差。

```
df["殘差"] = residuals_raw
df_filtered = df[np.abs(df["殘差"]) <= 2 * std].copy()
print("? 篩除高殘差樣本後剩餘筆數：", len(df_filtered))
```

- 將殘差超過 2 標準差的樣本移除，留下比較穩定的資料作為訓練集。

第 8 段：正式資料準備與切分

```
X = df_filtered[["縣市", "實際用途", "建物面積", "構造", "總樓層數", "建物現
況", "區名", "剩餘租期(月)", "lat", "lon"]]
y = df_filtered["log_租金"]
```

- 正式模型用特徵與標籤。

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

- 分割為訓練集、驗證集與測試集（三份）來做交叉驗證與防止過擬合。

第 9 段：訓練正式模型（含 early stopping）

```
model = CatBoostRegressor(
    depth=5,
    l2_leaf_reg=10,
    learning_rate=0.05,
    iterations=1000,
    early_stopping_rounds=50,
    random_state=42,
    verbose=100
)
```

- CatBoost 模型參數設定：
  - `depth=5`：決策樹深度
  - `l2_leaf_reg=10`：正則化
  - `early_stopping_rounds`：若 50 回合內驗證集沒有提升就提前停止訓練。

```
model.fit(
    X_train_,
    y_train_,
    eval_set=(X_valid, y_valid),
    cat_features=cat_features,
    use_best_model=True
)
```

- 正式訓練模型，並使用驗證集來做 early stopping。

第 10 段：預測與還原原始單位

```
y_train_pred = np.expm1(model.predict(X_train))
y_test_pred = np.expm1(model.predict(X_test))
y_train_true = np.expm1(y_train)
y_test_true = np.expm1(y_test)
```

- 預測值與真實值從 `log1p` 還原回原本的租金（使用 `expm1()`）。

第 11 段：模型評估

```
r2_train = r2_score(y_train_true, y_train_pred)
r2_test = r2_score(y_test_true, y_test_pred)
```

- 計算訓練與測試資料的 R? 分數（擬合程度指標）。

```
print("? 訓練 R?:", round(r2_train, 4))
print("? 測試 R? :", round(r2_test, 4))
if r2_train - r2_test > 0.1:
    print("?? 模型可能仍有過擬合")
else:
    print("? 模型表現穩定，沒有明顯過擬合")
```

- 顯示分數與簡單過擬合檢查邏輯（訓練與測試差距太大會警告）。

第 12 段：回歸預測圖

```
df_vis = X_test.copy()
df_vis["每月租金"] = y_test_true
df_vis["預測租金"] = y_test_pred
```

- 合併預測與實際結果以利繪圖。

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x="建物面積", y="每月租金", data=df_vis, label="實際值")
sns.lineplot(x="建物面積", y="預測租金", data=df_vis, color="red", label="預測值")
```

- 用散點圖與線圖比較預測與實際租金。

```
plt.title("建物面積對每月租金（測試資料）")
plt.xlabel("建物面積（平方公尺）")
plt.ylabel("每月租金（元）")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

第 13 段：殘差圖

```
residuals = df_vis["每月租金"] - df_vis["預測租金"]
plt.figure(figsize=(10, 6))
sns.residplot(x=df_vis["建物面積"], y=residuals, lowess=True,
color="purple")
```

- 計算預測誤差並畫出殘差圖（是否存在系統性偏差）。

```
plt.title("殘差分析圖（測試資料）")
plt.xlabel("建物面積（平方公尺）")
plt.ylabel("殘差（實際值 - 預測值）")
plt.grid(True)
plt.tight_layout()
plt.show()
```

第 14 段：SHAP 特徵重要性圖

```
explainer = shap.Explainer(model)
shap_values = explainer(X_test)
```

- 使用 SHAP 解釋模型並計算每個特徵對預測的貢獻。

```
shap.summary_plot(shap_values, X_test, show=False, plot_type="bar")
plt.title("特徵重要性分析（SHAP 值）")
plt.tight_layout()
plt.savefig("C:/Users/user/Desktop/taiwan_railway_rent_predictor/shap_summary
plt.show()
```

- 產生 SHAP summary bar chart 並儲存圖檔。
