Statistical Learning and Deep Learning, Project 1

Jie Lin 林杰 (B11705048)[1]

[1]Department of Information Management, National Taiwan University

November 21, 2024

1 檔案說明

主要程式檔案:

- `SUB_gdboost.ipynb`: 訓練 Gradient Boosting Decision Tree 模型的 Jupyter Notebook 檔案。
- `SUB_xgboost.ipynb`: 訓練 XGBoost 模型的 Jupyter Notebook 檔案。
- `SUB_pipeline.ipynb`: 結合 Gradient Boosting Decision Tree 和 XGBoost 進行預測的 Jupyter Notebook 檔案。我的最終預測結果是使用這個檔案產生的。

資料集與預測結果:

- `X_train.csv`: 訓練資料集。
- `y_train.csv`: 訓練資料集的標籤（房價）。
- `X_test.csv`: 測試資料集。
- `final_pred.csv`: 我的最終預測結果。

我的結果是使用 Gradient Boosting Decision Tree 和 XGBoost 共同產出的,所以在 `SUB_pipeline.ipynb` 中，我將這兩個模型的預測結果進行了加權平均，並且將最終的預測結果輸出到 `final_pred.csv` 中。

2 訓練與執行方式

1. 在 `SUB_gdboost.ipynb` 中執行所有的 cell，會產生 `final_model_gdboost.pkl`。
2. 在 `SUB_xgboost.ipynb` 中執行所有的 cell，會產生 `final_model_xgboost.pkl`。
3. 執行 `SUB_pipeline.ipynb` 中的所有 cell,會產生 `final_pred.csv`。此步驟會需要 `final_model_gdboost.pkl` 和 `final_model_xgboost.pkl`。

詳細的資料處理流程、模型選擇流程可以在第 **4** 部分中找到。

3 實驗設計與數據分割

在所有的實驗中，我將 `X_train.csv` 和 `y_train.csv` 分割為訓練集和驗證集，具體比例為 80% 訓練集和 20% 驗證集。

- 訓練集：用於模型的訓練。
- 驗證集：用於模型的性能評估以及超參數調整。

測試集（`X_test.csv`）僅用於生成最終的預測結果。

4 模型嘗試與對比實驗

在這次的實驗中，我嘗試了多個模型及前處理方法，並通過交叉驗證和驗證集上的 RMSE 進行比較。以下是具體實驗過程與結果：

4.1 前處理技巧嘗試

我嘗試的技巧包括：

- **I.** 將資料集中的「建築完成年月」欄位轉換成新的欄位「建築年齡」（以年為單位），並且在完成轉換後刪除原本的「建築完成年月」欄位。
- **II.** 將資料集中的「交易年」、「交易月」、「交易日」欄位整合成單一的「交易年」欄位，並表示成帶有小數的年份。
- **III.** 對於「地鐵站」、「超商」、「公園」、「托兒所」、「國小」、「國中」、「高中職」、「大學」、「金融機構」、「醫院」、「大賣場」、「超市」、「百貨公司」、「警察局」與「消防局」欄位進行 min-max 標準化。
- **IV.** Feature Selection: 使用 `Forward Selection` 方法選擇特徵，threshold 為所有特徵重要性的平均。

4.2 模型嘗試

嘗試的模型包括：

1. Gradient Boosting Decision Tree (GBDT)
2. XGBoost
3. Gradient Boosting Decision Tree (GBDT) + XGBoost

4.3 性能比較

以下表格總結了前處理方法在 GBDT 的性能表現（基於驗證集 RMSE）：

模型	前處理方法	驗證集 RMSE
GBDT	**I**	33559.43695000462
GBDT	**I**, **II**	33077.47809409529
GBDT	**I**, **II**, **III**	33099.13748192837
GBDT	**I**, **II**, **IV**	34001.23414142309

Table 1: 不同與前處理方法的性能比較

Table 1 顯示，前處理方法 **I** 和 **II** 的組合可以帶來最好的結果。在這裡，Data Normalization 和 Feature Selection 並沒有帶來顯著的性能提升，甚至有些微的下降。

找出較有效的前處理方法後，我接著嘗試了不同模型的組合。以下表格總結了各個不同模型組合的性能表現（基於驗證集 RMSE）：

模型	前處理方法	驗證集 RMSE
GBDT	**I**, **II**	32786.071429620235
XGBoost	**I**, **II**	32418.572258500837
GBDT + XGBoost	**I**, **II**	**32417.53458909182**

Table 2: 不同模型的性能比較

Table 2 顯示，XGBoost 模型的性能優於 GBDT 模型，而 GBDT 和 XGBoost 兩個模型的組合可以帶來最好的結果，代表這兩個模型小有些微的互補效果。

透過以上實驗，我找到最佳的前處理與模型的組合：前處理方法 **I** 和 **II**，以及 GBDT 和 XGBoost 兩個模型的組合。最終的預測結果是這兩個模型的預測結果的平均。這也是我提交的程式碼中包含的內容。

5 訓練與驗證過程

在模型訓練過程中，我使用了 Optuna 來進行超參數優化。以下是超參數調整的範圍：

5.1 Gradient Boosting Decision Tree

參數範圍：

- `n_estimators`: 500 至 2000（步長 500）
- `learning_rate`: 0.01 至 0.1，
- `max_depth`: 3 至 8
- `subsample`: 0.8 至 1.0
- `min_samples_split`: 2 至 10
- `min_samples_leaf`: 1 至 5

最佳模型參數通過多次調參，Gradient Boosting Decision Tree 模型的最佳參數如下：

- `n_estimators`: 1500
- `learning_rate`: 0.025469889165697028
- `max_depth`: 8
- `subsample`: 0.8517843833202309
- `min_samples_split`: 10
- `min_samples_leaf`: 4

5.2 XGBoost

參數範圍：

- `n_estimators`: 500 至 2000（步長 100）
- `learning_rate`: 0.01 至 0.3
- `max_depth`: 3 至 12
- `subsample`: 0.6 至 1.0
- `colsample_bytree`: 0.6 至 1.0
- `min_child_weight`: 1 至 20
- `reg_alpha`: 1×10^{-8} 至 10.0（對數刻度）
- `reg_lambda`: 1×10^{-8} 至 10.0（對數刻度）
- `gamma`: 0.0 至 5.0

經過多次調整，XGBoost 模型的最佳參數如下：

- `n_estimators`: 1500
- `learning_rate`: 0.0413
- `max_depth`: 11
- `subsample`: 0.6537
- `colsample_bytree`: 0.6771

- `min_child_weight`: 5
- `reg_alpha`: 0.7594
- `reg_lambda`: 0.0004
- `gamma`: 0.6680

最終的預測結果是平均 Gradient Boosting Decision Tree 和 XGBoost 兩個模型的預測結果，權重分別為 0.5。

6 Kaggle 預測結果

我在 Kaggle 上的 public score 為 29249.78450，排名為 13/79。