永豐AI GO競賽-攻房戰

隊伍：星之卡比

外部資料

- 內政部不動產成交案件實際資訊資料供應系統 (使用2020Q1~2023Q3，CSV格式)
- 各警察(分)局分駐(派出)所地址電話經緯度資料
- 全國消防單位位置及災害應變中心位置
- 台灣中油公司加油站服務資訊
- 台糖加油站據點資訊
- 鄉鎮土地面積及人口密度(97)
- 低收入戶戶數及人數按鄉鎮市區別分

資料前處理

- 轉換原始經緯度座標
- 實價登錄資料轉換
- 外部資料轉換
  - 消防局之地點資訊，主要提取經緯度座標，以與原始資料集做關聯
  - 警察局之地點資訊，主要提取經緯度座標，以與原始資料集做關聯
  - 官方數據集中的額外地點資訊，主要提取經緯度座標，以與原始資料集做關聯
  - 加油站(中油、全國加油站)之地點資訊，主要提取經緯度座標，以與原始資料集做關聯
  - 城鎮人口數、人口密度之資訊，我們基於縣市、鄉鎮市區來與原始資料集做關聯
  - 低收入戶之資訊，我們基於縣市、鄉鎮市區來與原始資料集做關聯

特徵工程

- 基於縣市、鄉鎮市區之房價統計資訊mean、std、max、min
- 基於座標距離，篩選每筆資料最近之4、10、30、100筆其餘房價統計資訊mean、std、max、min作為額外特徵
- 基於座標距離，篩選每筆資料距離200、1000、5000公尺之其餘房價統計資訊mean、std、max、min作為額外特徵

特徵工程

使用了下列多種方式將實價登錄之房價資訊與原始資料集做關聯，除了採用實價登錄之不動產交易資料外，也使用了實價登錄的租屋資訊以及預售屋的資訊。

- 縣市、鄉鎮市區
- 縣市、鄉鎮市區、路名、
- 縣市、鄉鎮市區、路名、主要用途、建物型態
- 縣市、鄉鎮市區、路名、主要用途、建物型態、總樓層數
- 縣市、鄉鎮市區、路名、主要用途、建物型態、移轉層次、總樓層數
- 縣市、鄉鎮市區、路名、主要用途、建物型態、移轉層次、總樓層數、**屋齡**
- 縣市、鄉鎮市區、路名、主要用途、建物型態、移轉層次、總樓層數、**屋齡**、**附屬建物面積**

特徴工程

```
df_external_gov_data = pd.read_csv('../data/external_gov_data.csv')
df_external_gov_data['附屬建物面積'].value_counts().reset_index().head()
```

	附屬建物面積	count
0	-0.651622	117855
1	0.189057	535
2	0.179147	434
3	0.258440	404
4	0.053309	404

```
df_train['附屬建物面積'].value_counts().reset_index().head()
```

	附屬建物面積	count
0	-0.438452	6174
1	-0.118180	24
2	-0.171559	21
3	-0.062666	20
4	0.041956	20

```
df_valid['附屬建物面積'].value_counts().reset_index().head()
```

	附屬建物面積	count
0	-0.438452	3129
1	-0.171559	13
2	0.020604	11
3	-0.118180	10
4	-0.101099	10

```
df_test['附屬建物面積'].value_counts().reset_index().head()
```

	附屬建物面積	count
0	-0.438452	3087
1	-0.242019	15
2	-0.171559	14
3	-0.062666	12
4	-0.079748	10

特徵工程

```
new_col_name = 'externalkey'
df_train, df_valid = mapping_external_gov_data(
    df_train,
    df_valid,
    df_external_gov_data,
    by = ['縣市', '鄉鎮市區', '路名', '主要用途', '建物型態'],
    new_col_name = new_col_name
)
na_cnt = sum(df_train[f'{new_col_name}_price_mean'].isna())
mapping_rate = 1 - na_cnt / len(df_train)
corr = df_train[['單價', f'{new_col_name}_price_mean']].corr(
print(f'mapping_rate = {round(mapping_rate*100, 3)}%, corr =
```

```
100%|
62.51it/s]
100%|
24.77it/s]
mapping_rate = 91.099%, corr = 0.92604
```

```
new_col_name = 'externalkey_samebuilding'
df_train, df_valid = mapping_external_gov_data(
    df_train,
    df_valid,
    df_external_gov_data,
    by = ['縣市', '鄉鎮市區', '路名', '主要用途', '建物型態', '總樓層數'],
    new_col_name = new_col_name
)
na_cnt = sum(df_train[f'{new_col_name}_price_mean'].isna())
mapping_rate = 1 - na_cnt / len(df_train)
corr = df_train[['單價', f'{new_col_name}_price_mean']].corr().iloc[0].va
print(f'mapping_rate = {round(mapping_rate*100, 3)}%, corr = {round(corr,
```

```
100%|
86.68it/s]
100%|
69.37it/s]
mapping_rate = 87.584%, corr = 0.94854
```

```
new_col_name = 'externalkey_samefloor'
df_train, df_valid = mapping_external_gov_data(
    df_train,
    df_valid,
    df_external_gov_data,
    by = ['縣市', '鄉鎮市區', '路名', '主要用途', '建物型態', '總樓層數', '移轉層次'],
    new_col_name = new_col_name
)
na_cnt = sum(df_train[f'{new_col_name}_price_mean'].isna())
mapping_rate = 1 - na_cnt / len(df_train)
corr = df_train[['單價', f'{new_col_name}_price_mean']].corr().iloc[0].values[1]
print(f'mapping_rate = {round(mapping_rate*100, 3)}%, corr = {round(corr, 5)}')
```

```
100%|
27.00it/s]
100%|
07.89it/s]
mapping_rate = 66.505%, corr = 0.93587
```

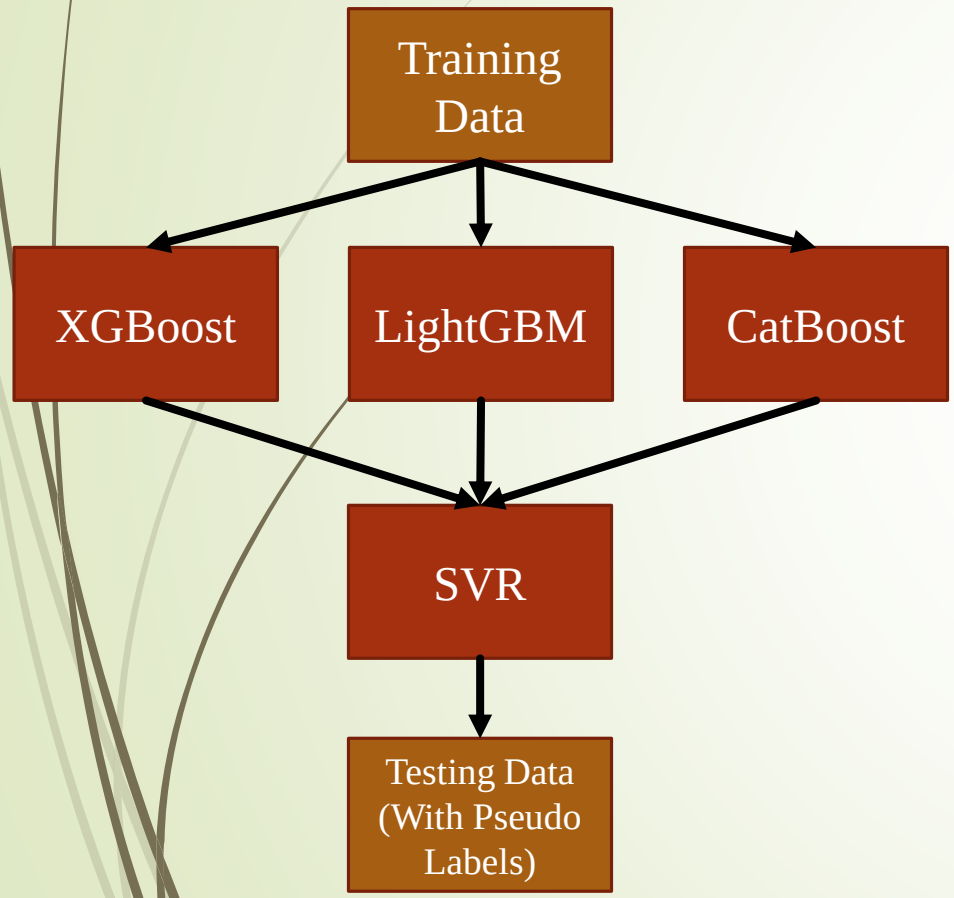
```
new_col_name = 'externalkey_samefloor_samecar'
df_train, df_valid = mapping_external_gov_data(
    df_train,
    df_valid,
    df_external_gov_data,
    by = ['縣市', '鄉鎮市區', '路名', '主要用途', '建物型態', '總樓層數', '移轉層次', '車位個數'],
    new_col_name = new_col_name
)
na_cnt = sum(df_train[f'{new_col_name}_price_mean'].isna())
mapping_rate = 1 - na_cnt / len(df_train)
corr = df_train[['單價', f'{new_col_name}_price_mean']].corr().iloc[0].values[1]
print(f'mapping_rate = {round(mapping_rate*100, 3)}%, corr = {round(corr, 5)}')
```

```
100%|
39.44it/s]
100%|
19.22it/s]
mapping_rate = 59.723%, corr = 0.94153
```

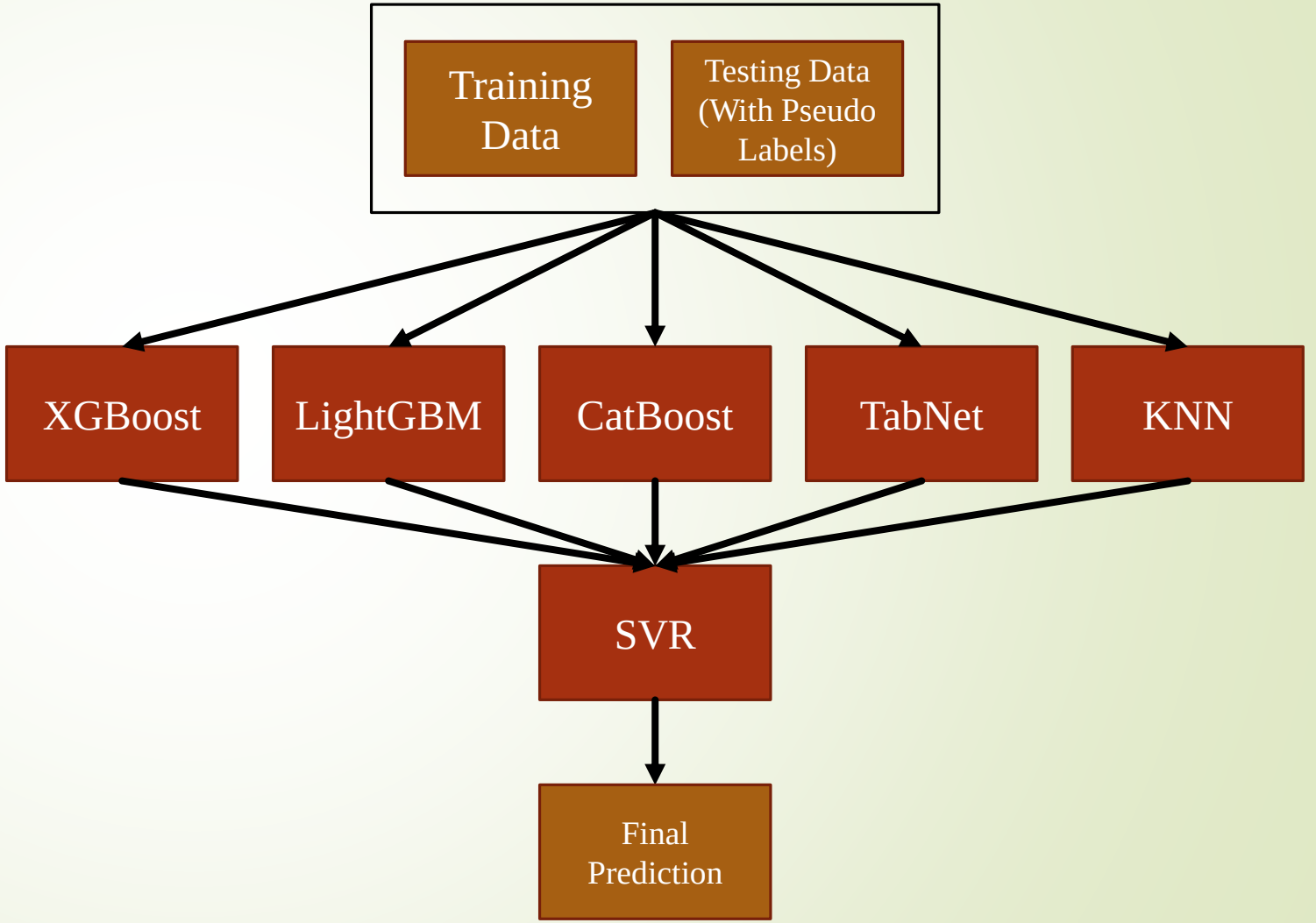
```
new_col_name = 'externalkey_sameage'
df_train, df_valid = mapping_external_gov_data(
    df_train,
    df_valid,
    df_external_gov_data,
    by = ['縣市', '鄉鎮市區', '路名', '主要用途', '建物型態'],
    new_col_name = new_col_name,
    age_in = 1.0
)
na_cnt = sum(df_train[f'{new_col_name}_price_mean'].isna())
mapping_rate = 1 - na_cnt / len(df_train)
corr = df_train[['單價', f'{new_col_name}_price_mean']].corr().iloc[0].values[1]
print(f'mapping_rate = {round(mapping_rate*100, 3)}%, corr = {round(corr, 5)}')
```

```
100%|
88.02it/s]
100%|
38.36it/s]
mapping_rate = 84.835%, corr = 0.95278
```

模型架構



(第一階段之模型架構，用以產生Pseudo Labels)



(第二階段之模型架構，以Training Data + Testing Data with Pseudo Labels完成模型訓練及預測)

實驗分享

在比賽的一開始，我嘗試了本地的Cross Validation，並在提交到Public LB之後，確認了Cross Validation結果以及Public LB的一致性，有關本地Cross Validation以及Public LB的結果關係，請參考下表。

Model Name	K-folds	Pseudo Labels	Cross Validation	Public LB
Model 20231106	5	No	7.0954	7.0084
Model 20231106	5	No	6.9612	6.9024
Model 20231107	10	No	6.869	6.8415
Model 20231108	20	No	6.7231	6.7254
Model 20231110	20	No	6.5918	6.5967
Model 20231112	20	No	6.3632	6.3141
Model 20231112	20	Yes	-	6.2868

(Cross Validation以及Public LB的結果關係)

實驗分享

- 實價登錄的資料與預測目標具有高度相關性，透過實價登錄資料，可以降低至少1%的成績 (e.g. 8.0 -> 7.0)
- 透過不同的資料對應、以及特徵間的統計計算(e.g. 同類型資料的相加/相減)方式，模型可以再降低0.3%左右的成績 (e.g. 7.0 -> 6.7)
- 根據LB回饋，Pseudo Labeling有助於降低0.03%左右的成績 (e.g. 6.7 -> 6.67)
- LightGBM + CatBoost + XGBoost 的ensemble可以降低0.2%左右的成績 (e.g. 6.67 -> 6.47)
- 加入非tree base模型(TabNet, KNN)對模型穩定性有幫助，同時也能小小降低MAPE (e.g. 6.47 -> 6.45)
- 針對官方資料的附屬建築物面積，以及實價登錄的附屬建築物面積對應，能再將低0.2%左右的成績 (e.g. 6.45 -> 6.25)

THANK YOU