● 摘要 [請簡單說明本次比賽所使用過之特徵、演算法以及訓練模型的方式]

此次比賽的建模過程為:1. 資料探索與想法實驗 2. Hyperparameters 搜尋以及其他模型設置測試 3. 訓練最終模型，進行 test set 預測，以及 kfold validation 預測 4. Stack 不同模型做出的預測。特徵方面，除了提供的數據直接得到的特徵，我們亦產生了一些新的特徵:經緯度編碼、時間相關特徵、樓層相關特徵，以及土地相關特徵。除此之外，我們使用特徵選擇和特徵分布調整的方式，使得特徵中的特徵更容易讓模型習得。比賽中使用 lightgbm (gradient boost tree)、類神經網路、隨機森林建模，並使用 linear stacking methods 組合個模型產生的預測。

● 環境 [請說明本次比賽所使用的系統平台、程式語言、函式庫]

我們使用的系統平台為 macOS、Windows 和 Ubuntu 皆有，記憶體為 16GB～32GB，CPU 為 4～20 cores，neural network 的機器則使用 Google cloud 提供的 nvidia K80 GPU。程式語言為 python 3.6+，主要使用的函式庫為 pandas、numpy、scipy、scikit-learn、lightgbm、keras、tensorflow、matplotlib。

● 特徵 [請說明本次比賽所使用的特徵]

原始資料分成 numerical、binary 和 categorical 特徵。numerical 特徵直接應用於模型訓練，binary 特徵則直接轉成 numerical (neural network) 或 categorical (lightgbm) 特徵。categorical 特徵則需要進行 categorical 編碼。不同的模型使用不同的 categorical 編碼技術:

- Lightgbm (gradient boost tree, GBT):lightgbm 內建 categorical 編碼。
- 類神經網路 (neural networks):使用 embedding 編碼。方法為建立一個 embedding layer 網路，將 categorical values 經由轉成適當的權重向量，再與其他特徵合併，進而在類神經網路中使用
- 隨機森林:使用 target encoding 的方法進行編碼。方法為使用目標值平均的順序作為編碼的依據。

除了原始的資料，我們也製作了額外的特徵:

- 經緯度編碼:將經緯度資料使用 K-means 進行 clustering，之後將 clusters 的 id 作為特徵。
- 時間相關特徵:從 building_complete_dt 和 transaction_dt 可以製作 building_age。而 transaction_dt 和 building_complete_dt 也可以製作年、月、日以及星期等特徵
- 樓層相關特徵:floor_ratio (txn_floor 除以 total_floor)、is_top_floor (如果 total_floor 等於 txn_floor)
- 土地相關特徵:area_ratio = building_area / land_area

特徵選擇
沒有相關性的特徵會使得模型的預測能力下降。因此，只選擇有幫助的特徵是會幫助模型學習的。在這次比賽中使用的特徵選擇方法步驟如下：
1. 先使用所有的特徵訓練一個模型。這個模型必須要有相對大的深度和葉子數量，使得所有的特徵都有被使用的可能性
2. 在 validation dataset 做預測時，將各 sample 的值亂序，測量 metric 改變的程度。在進行此步驟時，一次只有一個特徵 column 被亂序，因此可以測量出此特徵對模型的重要性，若是 metric 沒有變差或是變差的數量很小，則可辦定此特徵不重要；反之，則為重要特徵。
3. 根據前一步驟得到的特徵重要程度列表,我們可以將較為不重要的特徵刪除來提高模型的預測能力。移除特徵的量則使用 k-fold cross-validation 來決定。下面將會有更詳細的說明。

特徵分布調整
由於 neural network 對 input features 的分佈相當敏感，因此除了上述處理外，neural network model 的 numerical features 還有額外處理：
- 若該 feature 的 skewness > 2.2 則做 log10(0.1 + x - x.min()) 的轉換
- 做 standardize 的轉換讓所有的 numerical feature 皆有 mean = 0、stdev = 1。

● 訓練模型 [請說明本次比賽所使用的訓練模型、參數]

我們使用的 31 個 models stack 做為最後的結果，其中包含 26 個 Lightgbm models、5 個 neural networks 及 1 個 random forest。

Lightgbm
Lightgbm models 又分為以 log1p(total_price / building_area) 和直接以 log1p(total_price) 為 targets 兩種 (model 名包含 wo-per-area 的)，loss 為 target 的 mean square error 來訓練，但使用 mape (mean absolute percentage error) 做為 early stop 的 metric。詳細參數請見下表。平均來說我們使用 num_leaves 約為 250～300，max_depth 為 16～32，feature fraction 為 0.4～0.5，learning rate 為 0.0005~0.01，kfold CV 時的 number of iteration 由 validation set 的 mape 決定，並做 early stop (容許值為 2000 iterations、最大值為 100萬步)，結果的 stop iterations 從 1 萬 5000 到 76 萬步都有 (詳見下表)，final model 則使用由 kfold 所找到的最佳 iterations 做最終的 iterations 參數。

詳細 lightgbm 參數列表：

N	name	lr	max_depth	num_leaves	feature_fraction	min_data_in_leaf
1	lgb	0.01	16	300	0.5	10

3	lgb-corr-feats-selection-100	0.01	16	300	0.5	10
4	lgb-PCA	0.01	24	300	0.5	5
5	lgb-wo-per-area	0.01	24	258	0.4	6
6	lgb-lr0.001	0.001	16	300	0.5	10
9	lgb-feats-selection-75	0.01	24	300	0.5	10
10	lgb-feats-selection-75-lr-0.001	0.001	24	300	0.5	10
14	lgb-feats-selection-75-lr-0.001-rand	0.001	24	300	0.5	10
15	lgb-feats-selection-75-lr-0.001-rand323	0.001	24	300	0.5	10
16	lgb-feats-selection-68-lr-0.001-mix5	0.001	24	279	0.45	8
17	lgb-feats-selection-70-lr-0.001-mix5	0.001	24	279	0.45	8
18	lgb-feats-selection-70-lr-0.001-p5	0.001	24	258	0.4	6
19	lgb-search-bins-lr-0.0005	0.0005	30	300	0.45	5
20	lgb-lr-0.0008-mix5	0.0008	24	279	0.45	8
21	lgb-wo-per-area-long	0.0008	24	258	0.4	6
22	lgb-wo-per-area-long-2	0.0008	22	255	0.4	6
23	lgb-binary	0.0005	32	127	0.4	18
24	lgb-binary-augment	0.0008	16	127	0.3	2
25	lgb-search-bins-lr-0.0005-250	0.0005	26	250	0.45	5
26	lgb-search-bins-lr-0.0005-350	0.0005	26	350	0.45	5
27	lgb-feat_rm_new	0.0005	26	350	0.45	5
28	lgb-search-bins-lr-0.0005-255	0.0005	22	255	0.45	8
29	lgb-building_age-fillna	0.0005	22	255	0.45	8
30	lgb-binary-2	0.0005	24	127	0.4	6
31	lgb-3_groups	0.0005	23	85	0.4	4

詳細 lightgbm 參數列表 (續):

N	name	lambda_l1	lambda_l2	max_bin	min_sum_hessian_in_leaf	min_data_per_group	平均 early stop iterations	特徵移除數量
1	lgb	0.01	0.1				30733	
3	lgb-corr-feats-selection-100	0.01	0.1				30866	100
4	lgb-PCA	0.1	0				15245	
5	lgb-wo-per-area	0.04	0.02				26017	
6	lgb-lr0.001	0.01	0.1				296069	
9	lgb-feats-selection-75	0.1	0				16373	75

10	lgb-feats-selection-75-lr-0.001	0.1	0				159504	75
14	lgb-feats-selection-75-lr-0.001-rand	0.1	0				159803	75
15	lgb-feats-selection-75-lr-0.001-rand323	0.1	0				160513	75
16	lgb-feats-selection-68-lr-0.001-mix5	0.07	0.01				192753	68
17	lgb-feats-selection-70-lr-0.001-mix5	0.07	0.01				192315	70
18	lgb-feats-selection-70-lr-0.001-p5	0.04	0.02				248465	70
19	lgb-search-bins-lr-0.0005	0.05	0.001	383	0.0001		383077	68
20	lgb-lr-0.0008-mix5	0.07	0.01				234106	70
21	lgb-wo-per-area-long	0.04	0.02				311525	35
22	lgb-wo-per-area-long-2	0.04	0.02	511	0.01		318721	35
23	lgb-binary	0	0.05	511	0.1	50	669762, 376184	68
24	lgb-binary-augment	0.05	0.01	255	0.01	25	447854, 473562	68
25	lgb-search-bins-lr-0.0005-250	0.05	0.001	383	0.0001		456619	68
26	lgb-search-bins-lr-0.0005-350	0.05	0.001	383	0.0001		332877	68
27	lgb-feat_rm_new	0.05	0.001	383	0.0001		310508	85 new
28	lgb-search-bins-lr-0.0005-255	0.02	0.04	511	0.1	50	561421	68
29	lgb-building_age-fillna	0.02	0.04	511	0.1	50	552896	68
30	lgb-binary-2	0.01	0.01	255	0.001	10	764125, 728676	68
31	lgb-3_groups	0.01	0.01	255	0.001	7	643720, 611161, 613967	68

另外，由於我們看到在 total_price / building_area 的分佈重現雙波峰的奇特現象，因此我們嘗試先建立一個 binary classification 將兩波峰的 training data 分開成兩個 groups，再分別對兩個 groups 的 data train 兩個 models，以期望個別 model 所需要學習的目標較為接近 Gaussian 分佈 (此方法標示為 binary)。以此類推我們也做了 3 groups 的版本 (model 31)。

我們也發現過的多的特徵會降低學習效率，因此我們根據特徵的重要性移除較不重要的特徵，標示為 (特徵移除數量)。

<u>Neural network</u>
Neural network models 為基本的 dense layers 相疊，但 categorical features 先輸入到 embedding layers 再與其他數值 features 和 binary features 相併。5 個 neural networks 的 dense layers 結構如下：

- 6 層 dense layers (3548, 3548, 1774, 1774, 887, 887)

- 10 層 dense layers (1024,) * 5 + (512,) * 5
- 4 層 dense layers (3440, 6880, 1720, 860)
- 4 層 dense layers (7400, 3700, 1850, 925)
- 4 層 dense layers (4764, 9528, 2382, 1191)

每一層我們使用 batch normalization，optimizer 為 Adam 和 learning rate 0.001 左右。Loss function 為 mean square error。Nerual network 的 target 為 log1p(total_price / building_area)，再經過 standardization，讓輸入到 neural network 的 target 有平均值為 0，標準差為 1。Activation function 為 relu 或 prelu。5 個 neural networks 的其餘參數 (batch size, epochs, learning rate) 依序如下：

- Batch size = 256, epochs = 227, activation = relu, learning rate = 0.001
- Batch size = 64, epochs = 179, activation = relu, learning rate = 0.001
- Batch size = 256, epochs = 598, activation = relu, learning rate = 0.001
- Batch size = 256, epochs = 854, activation = prelu, learning rate = 0.0009
- Batch size = 256, epochs = 805, activation = relu, learning rate = 0.001

● 訓練方式及原始碼 [請說明本次比賽答案的產出方式並提供有效之原始碼(連結亦可)]

Cross-validation

我們使用 3-fold 和 5-fold cross validation (CV) 來確定個別 model 的功效，用以決定模型的 hyperparameters 和 training 時最佳的停止 epochs/iterations。在此方法中，每個 fold 的 model 都只 train on 4/5 的 training set，剩下的 1/5 做為 validation 使用。以此 valiadation 上的分數來決定最佳的 epochs/iterations 數或是stacking 比重。然後個別 model 再以找到的最佳 epochs/iterations 重新 train on 全部的 training set，對 test set 做預測，做為最終的輸出。Neural network 的方法較特別，因為 neural network noise 的關係，我們直接使用 kfold CV 時每個 fold 的 model 的結果取平均做為 stack 的輸出，而沒有使用全部的 training set 重新訓練。

Stacking

各個模型建立好之後，我們可以將模型預測的結果以各種不同的方式結合，以達到更好的預測。先使用k-fold cross-validations預測的值搜尋最佳的組合參數，之後將對於test set的predictions使用相同參數組合。在這次比賽中，我們使用的 stacking 方法均為線性相加，即是各個 model 的預測值乘以一個係數後相加，得到最後的預測。在此概念下，我們測試了各種不同的 stacking 方式：

- 預測的 total_price 直接線性相加
- 先使用 log1p 轉換 total_price 到 log_total_price，線性相加之後再做 expm1 得到最終預測
- total_price 先除以 building_area，再使用 log1p 做轉換；線性相加之後，再做 expm1 和乘以 building_area 還原成最終預測

而尋找最佳的線性相加權重則使用了以下的最佳化過程：

- 平均相加：使用貪婪算法搜尋最佳的模型組合，之後直接將預測值平均相加

- 線性回歸：使用 L1 (Lasso) regularization 線性回歸最佳化參數。Lasso 中的 hyperparameter 則使用 cross-validation 決定
- 亂數搜尋：不斷的產生隨機的參數組，並檢查是否能夠得到更好的 metric 結果。再我自行開得的程式碼中，亂數是產生於高斯分布。使用的初始值則可從平均相加的結果得到 (沒有被貪婪算法選到的模型初始化權重為 0，其他則平分 100%)
- 使用最佳化算法：因 scikit-learn 的 linear regression 只能最佳化 mean-squared error，因此我們使用 scipy.optimize.minimize 進行最佳化。最佳化的目標為 MAPE。在程式進行最佳化的過程中，hit-rate 也會被計算，最佳的 hit-rate 和相對應的權重組合也會被記錄下來

<u>原始碼使用方式</u>
先執行 dataset/gen_5_fold_cv.ipynb 來將 training set 分成 5 fold CV sets，接著執行各個 metamodel 為開頭的 code 來產生一部分 feature engineering。然後執行各個 model 為開頭的 code 訓練各個 model，會輸出。最終執行想要的 stacking 方法的 code (由 stack 為開頭)。結果輸出在 output 資料夾。

● 結論 [請簡易說明本次競賽後所得的結論]

非常感謝玉山銀行和趨勢科技舉辦這次的比賽，我們在這次的比賽中學習到很多。這次比賽的數據特性是：samples數量相對少，但是特徵很多。這種情況下 overfitting 的問題理論上會較為嚴重，因此，特徵選擇、數據降維和其他避免 overfitting 的方法非常的重要。這次比賽中，因為數據進行了神祕轉換，有些想像中可以進行的 feature engineering 無法達成，譬如說一般下半年房市就為冷淡，這部分只能隨意地使用天數除以 30 或是 90 來得到類似於月或是季的特徵。模型方面，不意外的 gradient boost tree 是最為重要的模型。但是類神經網路在 stacking 中也不可或缺，可提供另一個觀點。learning rate 則是相當意外的下降的 saturate 的值非常的小，因此耗費很多資源在建很多樹的模型。至於 leaderboard score，不能不提一下 shake up。此次比賽我們在公榜上第 18 名。最終私榜第四名，稍微體驗了 shake up 的震撼，不過其他獲得前六名的隊伍似乎沒有公榜和私榜的差距。當然，可以想像公榜評分的 dataset 較小，flunctuations 較大，因此出來的分數不一定那麼可信，不過要努力的不去相信公榜上的分數，或是不使用公榜分數 judge 模型好壞還是需要一點心理建設的。