Part 0. 請清楚標示你選的題目 (1)

Sberbank Russian Housing Market

Part 1. Team name, members and your work division (1)

Team name：

NTU_r05922085_紗路的名字來自非洲的吉力馬札羅山

Members & Work division：

蘇俞睿 (r05922026)：

- XGBoost - Linear Regression 架構的程式碼撰寫與調整參數實驗
- 資料前處理之程式碼撰寫與實驗

蕭至恆 (r05944033)：

- 各維特徵分析與作圖
- 統計數據並撰寫報告

周宗鴻 (r05922085)：

- XGBoost - Linear Regression 架構的程式碼撰寫與調整參數實驗
- 統計數據並撰寫報告

陳奕瑄 (r05922168)：

- DNN 架構的程式碼撰寫與調整參數實驗
- 資料前處理之程式碼撰寫與實驗

Part 2. Preprocessing / Feature Engineering (3)

我們把 train.csv 與 test.csv 的所有特徵讀進來之後，會進行下列的前處理：

(a) data.join(macro_data)

總體經濟會影響單一房屋的成交價格，而 macro.csv 裡，提供了許多總體經濟學上的特徵，將是良好的參考依據。這裡以 timestamp 為基準，將 macro.csv 裡的特徵附加於相對應 timestamp 的房屋上。

(b) if life_sq > full_sq, then life_sq = full_sq * 0.6

full_sq 表示住所總面積，而 life_sq 則表示生活區域(living area)面積，故 life_sq 包含於 full_sq，所以 life_sq 不應大於 full_sq。若碰到 life_sq > full_sq 的情形，一定是登記錯了嘛，因此經實驗(詳見後)，我們決定將這樣的 life_sq 替換為 full_sq * 0.6。

(c) if kitch_sq > life_sq, then kitch_sq = life_sq * 0.25

kitch_sq 表示房子裡廚房的面積，故 kitch_sq 包含於 life_sq，所以 kitch_sq 不應大於 life_sq。若碰到 kitch_sq > life_sq 的情形，一定是登記錯了嘛，因此經實驗(詳見後)，我們決定將這樣的 kitch_sq 替換為 life_sq * 0.25。

(d) if build_year < 1910 or build_year > 2017, then build_year = NaN

build_year 表示房子的建造年份，建造年份越久遠，則越有可能因為當時的登記制度不完善而有錯誤；而建造年份古老的房子的價格也有可能會因為特殊情況而與眾不同(比如原本古老破爛的房子因為變成了古蹟而價格攀升)。因此經實驗(詳見後)，我們決定直接忽略 build_year < 1910 的情形；另外，建造年份若大於今年也是不可能發生的情形，我們也選擇忽略這樣的情況。

(e) if floor <= 0 or max_floor <= 0, then floor = NaN, max_floor = NaN

floor 表示住處的所在樓層數，max_floor 則表示該棟建築的最大樓層數。以下考慮一些情形：

(i) floor == 0 or max_floor == 0: 我們所說的的樓層不會有「0 樓」這種東西，所以碰到 floor 或 max_floor 等於 0 的情形，一定是登記錯了。

(ii) max_floor < 0: max_floor 不可能是負數，因為就算整棟房子都在地下，我們也不能直接說這棟房子的 max_floor 是在地下幾層，所以這種情形也一定是登記錯了。

(iii) floor < 0: floor 是負數有可能是登記錯誤，就算住處所在樓層真的在地下，這種情形的變數也很多，使之難以成為一個很好的參考依據。

綜上所述，我們決定直接忽略上述三種情形。

(f) if floor > max_floor, then floor = max_floor * 0.6

住處的所在樓層數不可能大於該棟建築的最大樓層數，若碰到 floor > max_floor 的情形，一定是登記錯了嘛，因此經實驗(詳見後)，我們決定將這樣的 floor 替換為 max_floor * 0.6。

(g) month, month_cnt = process_timestamp(timestamp)

timestamp 表示房子的成交日期，它可以被轉成兩個比較有用的資

訊，詳列如下：

(i) month_year_cnt: 成交日期可以當成一個參考特徵，而這裡以（年份 * 12 + 月份）當作是成交時間。

(ii) month: 各國的房屋交易總有淡旺季，比如台灣國的鬼月成交量會比較低，農曆年前後的成交量會較高，而成交量便會影響到成交價格。成交月份是最能反映淡旺季的因素，因此以它來當作一個參考特徵。

(h) floor_rate = floor / max_floor

住處在該建築內的高度相對位置，可能影響到視野、逃生安全等，進而反應在房價上。這裡將 floor 除以 max_floor，當作一個新的參考特徵。

(i) kitch_rate, life_rate = kitch_sq / full_sq, life_sq / full_sq

廚房面積與生活區域面積佔住所總面積皆會對房價產生若干程度影響，因此這裡將上述兩者各當成一個新的參考特徵。

(j) data = process_category(data)

有許多特徵並不是以數值而是以類別的方式存在，因此這裡將這種類型的特徵統一轉換成數值，以方便做運算。

(k) data = normalize(data)

再訓練 DNN 模型的時候，若各維特徵的尺度大小差異很大，將可能大幅影響收斂速度，因此在訓練 DNN 模型時，我們會對資料進行標準化，即 data = (data − mean(data)) / std(data)。

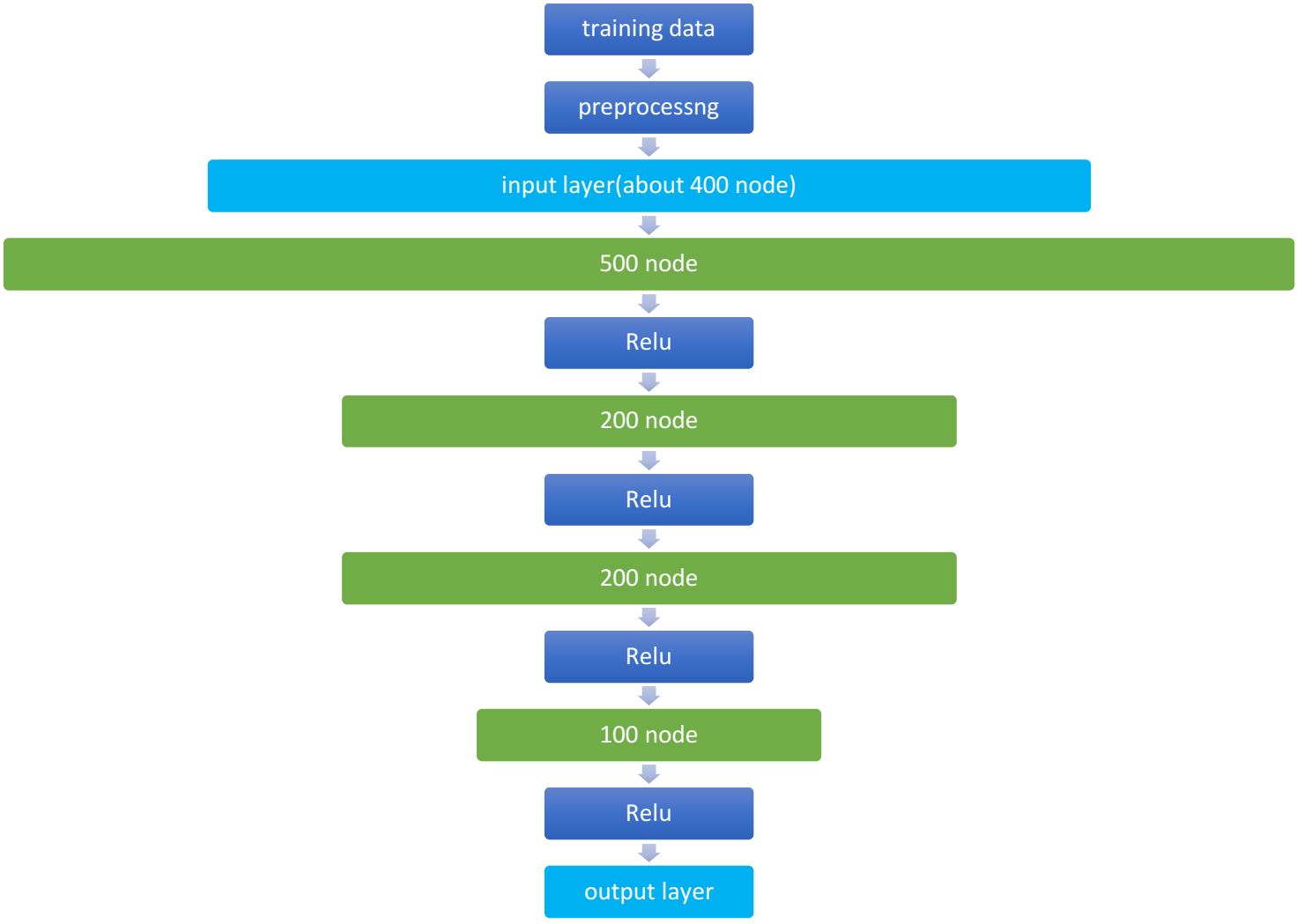
(l) data = fill_nan(data)

在 train.csv 與 test.csv 中，有許多資料是空白(NA)的，因此我們會將這些資料替換為 0。唯 XGBoost 訓練模型時自己會處理這些空白資料，因此訓練 XGBoost 模型時我們不會進行此項前處理。

Part 3. Model Description (At least two different models) (7)

模型一：DNN

我們有嘗試過用 DNN 模型來預測答案，模型架構為 4 層隱藏層，各層 node 數為 500、200、200、100，每層後面接 Relu activation function，model.compile 之參數為{loss=rmse, optimizer='adam', batch_size=256}，其整體架構如下圖：

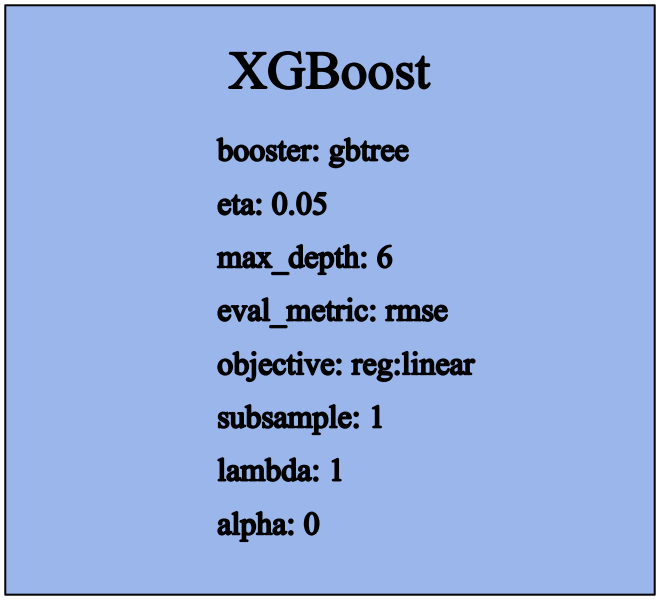


模型二：XGBoost - Linear Regression

Gradient Boosting 架構是透過許多 weak learner 組合而成，透過每次 iteration 來更新往最小化 objective function’s loss 的方向前進。而 XGBoost 是 Extreme Gradient Boosting 的縮寫，與上述傳統的 Gradient Boosting Tree 主要差別在於 Gradient Boosting Tree 只有使用到一階導函數且並無 regularization 的部分，但 XGBoost 則多使用了二階泰勒展開、加入二階導函數、regularization 等部分，下面為 XGBoost 的幾個重要參數及說明：

參數	預設值	說明
booster	gbtree	gradient boosting decision tree
eta	0.3	Learning rate
max_depth	6	每棵 gbtree 的最大高度
eval_metric	依據 objective 定義	用來評測模型的 function
objective	reg:linear	模型學習的目標
subsample	1	控制每棵 gbtree 的資料比例
lambda	1	L2 regularization 的 weight
alpha	0	L1 regularization 的 weight

下圖為我們最終使用的 XGBoost



Part 4. Experiments and Discussion (8)

以下實驗是將 train.csv 隨機切成 8 份，每次以其中 1 份作為 training data，另外 7 份作為 validation data，共實驗 8 次後取平均所得到的數據。表格中字體顏色為浪漫的粉紅色者代表最後決定所採用的參數。

DNN 部分

實驗一：有無標準化測資

此實驗中的模型架構為: input → 500 → 200 → 200 → 100 → output；有進行標準化的 rmlse 為 0.48173，沒進行標準化的 rmlse 則為 0.58293，可見標準化的確會對 DNN 模型的訓練造成影響。

實驗二：不同架構的 DNN 模型

2 層隱藏層 (input → 500 → 200 → output)：0.52817

3 層隱藏層 (input → 500 → 200 → 100 → output)：0.49471

4 層隱藏層 (input → 500 → 200 → 200 → 100 → output)：0.48173

5 層隱藏層 (input → 500 → 400 → 200 → 200 → 100 → output)：0.48819

※ 由於簡單測試了幾個不同架構的 DNN 模型，validation data 與 kaggle 上的表現皆沒有明顯進步，因此我們認為 DNN 也許並不是一個很好的方向，於是便沒有繼續鑽研改進 DNN 模型的方法。

XGBoost - Linear Regression 部分

實驗一：決定 model 參數

此實驗中，先不做特徵的前處理，在調整其中一項參數時，固定其他參數為預設值。

eta: learning rate.

eta	0.0001	0.0005	0.001	0.005
rmlse	3.26793	1.76418	1.19466	0.60293

eta	0.01	0.05	0.1	0.3 (default)
rmlse	0.48603	0.44124	0.47506	0.46724

eta	0.5
rmlse	(爆了)

max_depth: maximum depth of a tree.

max_depth	3	4	5	6 (default)
rmlse	0. 48678	0.47749	0.46992	0.46724

max_depth	7	8
rmlse	0.48992	0.61928

subsample: subsample ratio of the training instance.

subsample	0.5	0.6	0.7	0.8
rmlse	0.46881	0.46871	0.47123	0.46418

subsample	0.9	1 (default)
rmlse	0.46931	0.46724

※ 由於此參數對 rmlse 影響似乎不大，而且此參數最佳值與預設值之 rmlse 相差微小，因此雖然預設參數並非最好，我們最後仍採用預設參數。

colsample_bytree: subsample ratio of columns when constructing each tree.

colsample_bytree	0.5	0.6	0.7	0.8
rmlse	0.47182	0.47311	0.46612	0.46819

colsample_bytree	0.9	1 (default)
rmlse	0.46692	0.46724

※ 由於此參數對 rmlse 影響似乎不大，而且此參數最佳值與預設值之 rmlse 相差微小，因此雖然預設參數並非最好，我們最後仍採用預設參數。

實驗二：決定各前處理之方式

此實驗中，模型參數設為步驟一中試出來最好的參數，調整其中一項前處理的參數時，其他前處理均不做。

data.join(macro_data)

做不做？	不做 (default)	做
rmlse	0.44124	0.42992

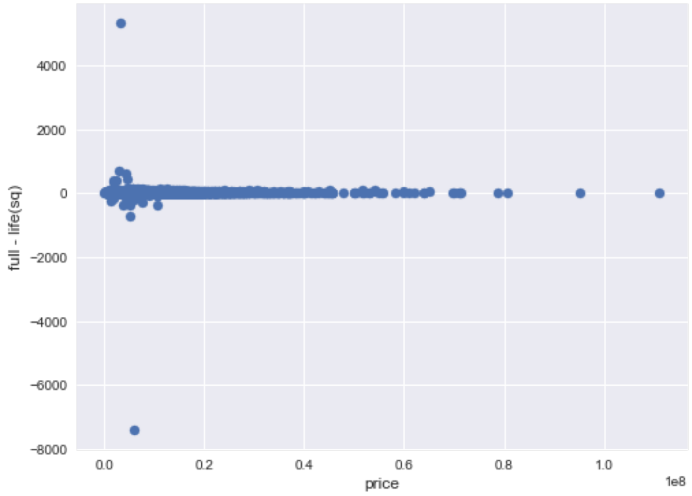
if life_sq > full_sq, then life_sq = full_sq * syaro

syaro	0.4	0.45	0.5	0.55
rmlse	0.44182	0.45182	0.43981	0.44819

syaro	0.6	0.65	0.7	0.75
rmlse	0.43117	0.45172	0.45192	0.44281

syaro	0.8	不做(default)
rmlse	0.45178	0.44124

下圖為 full_seq - life_seq 對 price_doc 的分佈圖，由圖中可發現的確有許多樣本之 full_sq 小於 life_sq，因此此項前處理是必要的。

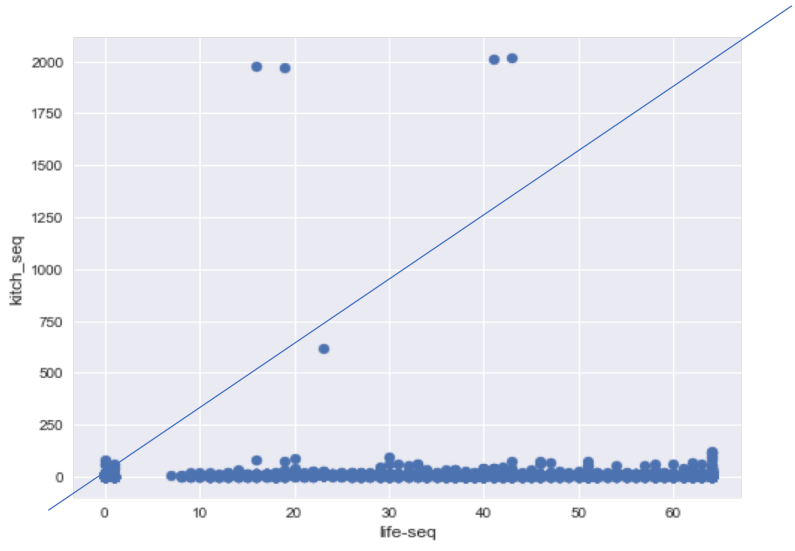


if kitch_sq > life_sq, then kitch_sq = life_sq * syaro

syaro	0.1	0.15	0.2	0.25
rmlse	0.48122	0.47218	0.45123	0.43910

syaro	0.3	0.35	0.4	0.45
rmlse	0.45721	0.44112	0.44110	0.45174
syaro	0.5	不做(default)		
rmlse	0.45999	0.44124		

下圖為 kitch_sq 對 life_sq 之分布圖，圖中高於對角線的點即為不正常的 data，有替換之必要。



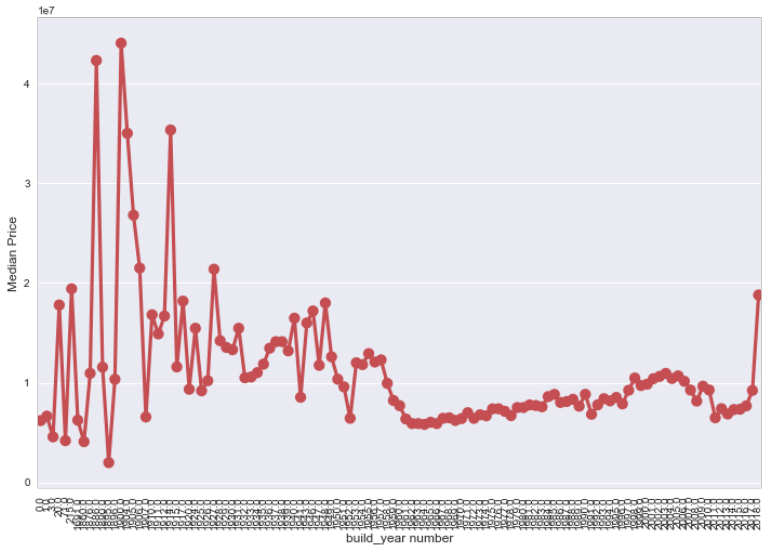
if build_year < syaro or build_year > 2017, then build_year = NaN

syaro	1830	1840	1850	1860
rmlse	0.44313	0.44313	0.44313	0.44313

syaro	1870	1880	1890	1900
rmlse	0.44313	0.44313	0.44313	0.44100

syaro	1910	1920	1930	不做(default)
rmlse	0.44087	0.44255	0.44789	0.44124

下圖為 build_year 對 prince_doc 平均數之分布圖，由圖中可以發現在約 1900 出頭之前隻房屋成交價格跳動非常大，故非常可能為登記錯誤之樣本，有忽略其建造年份之必要。



if floor <= 0 or max_floor <= 0, then floor = NaN, max_floor = NaN

做不做？	不做 (default)	做
rmlse	0.44124	0.44115

if floor > max_floor, then floor = max_floor * syaro

syaro	0.5	0.55	0.6	0.65
rmlse	0.43970	0.43787	0.43447	0.44105

syaro	0.7	0.75	0.8	0.85
rmlse	0.44128	0.44121	0.45002	0.45011

syaro	0.9	不做(default)
rmlse	0.45046	0.44124

month, month_cnt = process_timestamp(timestamp)

做不做？	不做 (default)	做
rmlse	0.44124	0.43974

floor_rate = floor / max_floor

做不做？	不做 (default)	做
rmlse	0.44124	0.44109

kitch_rate, life_rate = kitch_sq / full_sq, life_sq / full_sq

做不做？	不做 (default)	做
rmlse	0.44124	0.44120

另外，下圖為各個特徵之重要性統計圖，是以 xgboost.plot_importance()這個函數畫出來的。由圖中可以發現 full_sq、floor、life_sq、build_year、max_floor、kitch_sq 等特徵對房價之影響街站很重要的地位，可進一步證明我們之前做的這些前處理方向是正確的。