ML2017 Final Project

Problem: 俄羅斯房地產 Sberbank Russian Housing Market
Team name: NTU_b04902112_捲起袖子開心幹
Members:

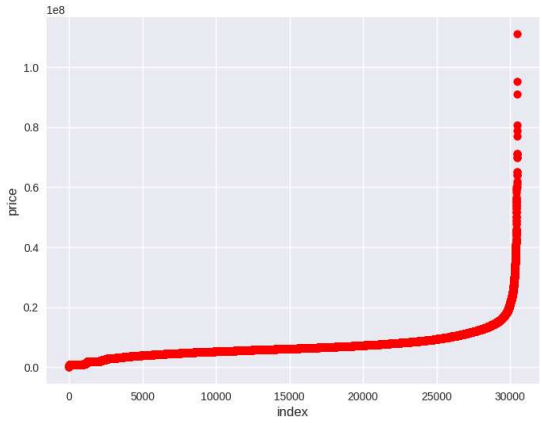
資工二 B04902112 張凱捷
資工二 B04902023 鄭士驤
資工二 B04902048 蔡毓聰
資工二 B04611015 陳佳佑

Work division:

B04902112 - Sberbank Strong Baseline and Final Submission, Report
B04902023 - Presentation，Sberbank DNN implementation and experiments, Report
B04902048 - Pump Simple Baseline, Pump Strong Baseline, Report
B04611015 - Sberbank Simple Baseline, DengAI Simple Baseline, Report

Preprocessing / Feature Engineering

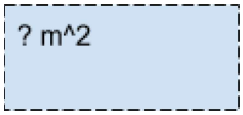
- Train data依照時序的價格分布，可以看出來是呈現指數成長，剛好符合Metric的RMSLE的性質，一般而言，可以不用特別對做轉換調整分布，不過為了更好的performance，還有2015俄羅斯有些突發狀況，我們還是有做了一些調整

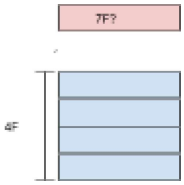


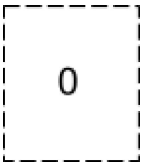
- 我們對於特徵的前處理，依其目的大概可以分為過濾錯誤資料、去除極端情形、防止overfit、新增特徵。以下敘述的前處理方式，都會用在兩個我們所使用的模型 (XGBoost, DNN)中。

❑ **過濾錯誤資料**

- 居住面積 > 總面積
- 廚房面積 > 居住面積
- 未知面積
- 最高樓層 = 0
- 樓層 > 最高樓層
- 房間數 = 0

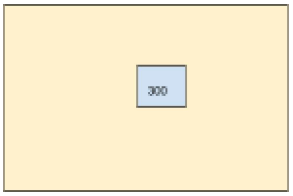


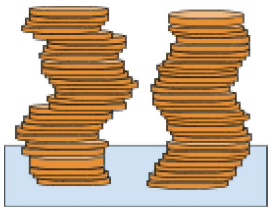


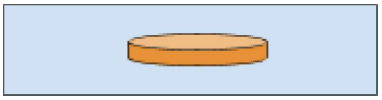


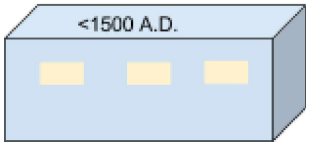
❑ **去除極端情形**

- 居住面積 < 5
- 居住面積 > 300
- 居住面積比例 (居住面積/總面績) < 0.3
- 每平方米價格 (總價 / 總面績) > 600000
- 每平方米價格 (總價 / 總面績) < 10000
- 非常老的房 (建造時間 < 1500)









❑ **防止Overfit**

- 去除id

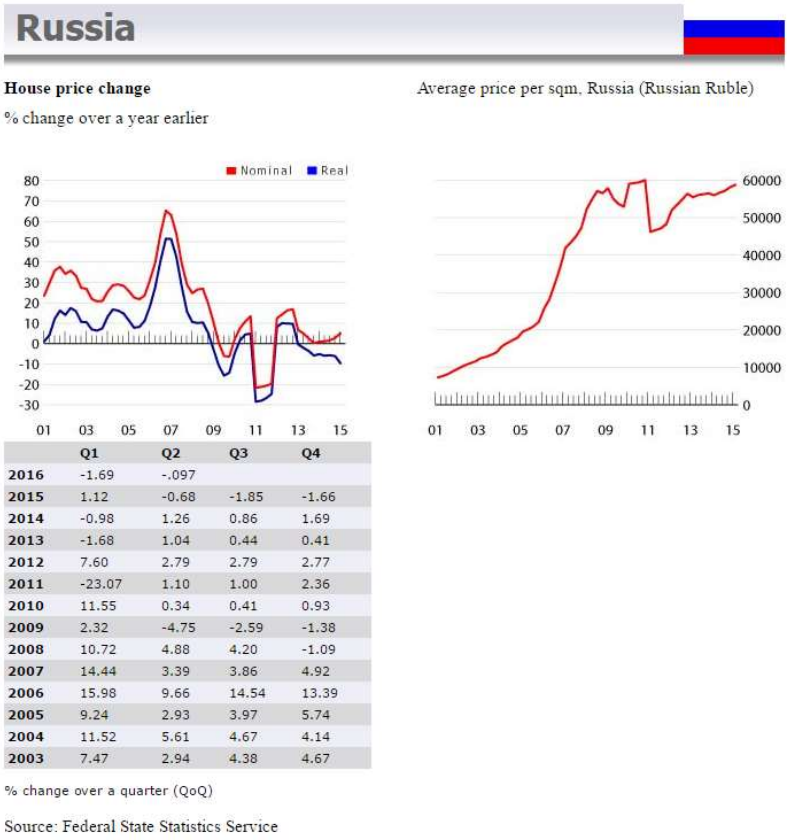
- 去除timestamp

❑ **新增其它特徵**

- 相對樓層 (樓層 / 最高樓層)
- 廚房比例 (廚房面積 / 總面積)
- 公寓名稱 (區域名稱 + 離地鐵站平均距離)

❑ **房價調整**

- 透過House Price Index對training data的房價做標準化



	Q1	Q2	Q3	Q4
2016	-1.69	-.097		
2015	1.12	-0.68	-1.85	-1.66
2014	-0.98	1.26	0.86	1.69
2013	-1.68	1.04	0.44	0.41
2012	7.60	2.79	2.79	2.77
2011	-23.07	1.10	1.00	2.36
2010	11.55	0.34	0.41	0.93
2009	2.32	-4.75	-2.59	-1.38
2008	10.72	4.88	4.20	-1.09
2007	14.44	3.39	3.86	4.92
2006	15.98	9.66	14.54	13.39
2005	9.24	2.93	3.97	5.74
2004	11.52	5.61	4.67	4.14
2003	7.47	2.94	4.38	4.67

% change over a quarter (QoQ)

Source: Federal State Statistics Service

Reference: http://www.globalpropertyguide.com/real-estate-house-prices/R#russia

❖ DNN模型額外前處理：除了上述的前處理之外，針對DNN模型，我們還另外對NaN做處理，處理方式如下：

❑ **新增NaN特徵及改寫NaN為0**

❖ 上述捨棄資料的前處理實作方式是將想要捨棄的資料填寫為NaN，但不適合NaN處理，所以我們把NaN改成一個實數，就取0，但是實際上它並不是仍然是NaN，所以我們決定新增維度

❖ 一共有54個維度是含有NaN，所以我們新增54個新維度，值只會是0或1，用來指該欄是否為NaN

❑ **切Validation Set**

- ❖ 切Validation Set的時候，我們想要盡量讓各種情況以相同的比例出現在Validation Set及Training Set中。我們讓對於所有54維NaN的組合，都有占總資料0.2的Validation Set及0.8的Training Set，簡單地來說就是分Case去切Validation Set，事實上，這裡的NaN並非原始資料的NaN，它包含著經過「去除極端情形」之後的NaN，所以這麼做等於是我們也它極端情形拉回討論，但是會至少有一個維度是跟非極端情形不同的。

Model Description (At least two different models)

- ❑ **Model 1 - XGBoost + shifting**
  - ❑ 我們用了三個model做training，分別以不同的考量，去做feature preprocess，然後再透過ensemble，平衡每組的bias。

  - ❖ **Parameter of XGB1**
    - Eta : 0.05
    - max_depth : 6
    - Subsample : 0.6
    - Colsample_bytree: 1
    - Objective: reg:linear
    - Eval_metric: rmse

  - ❖ **Parameter of XGB2**
    - Eta : 0.05
    - max_depth : 5
    - Subsample : 0.7
    - Colsample_bytree: 0.7
    - Objective: reg:linear
    - Eval_metric: rmse

  - ❖ **Parameter of XGB3**
    - Eta : 0.05
    - max_depth : 5
    - Subsample : 0.7
    - Colsample_bytree: 0.7
    - Objective: reg:linear
    - Eval_metric: rmse

- ❑ **Model 2 - DNN**
  - ❖ **DNN1**
    - Loss function: RMSLE, optimizer: adam
    - Layer: (350, 64, 64, 1) (input_dim = 350)
    - Validation Split = 0.2

- Validation RMSLE: 0.38
- Epoch: 500
- Batch Size: 32
- Activation Function: RELU

❖ **DNN2**

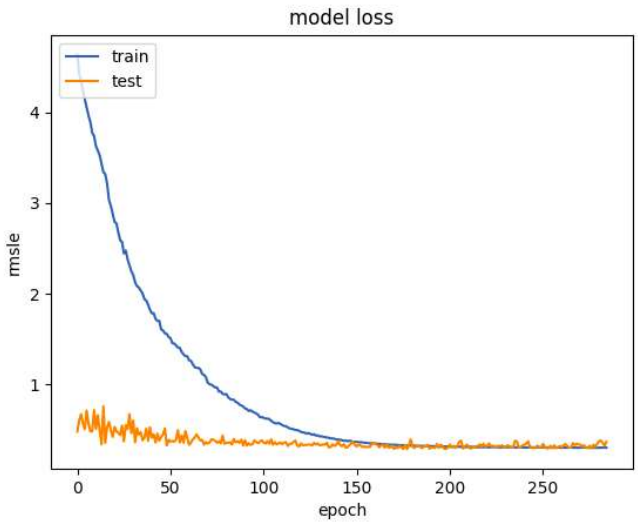
- 先將training price取log, Loss function: RMSE, optimizer: adam
- Validation Split = 0.2
- Epoch: 300
- Batch Size: 128
- Activation Function: RELU

- 結構

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 80)	23760
activation_1 (Activation)	(None, 80)	0
batch_normalization_1 (Batch	(None, 80)	320
dropout_1 (Dropout)	(None, 80)	0
dense_2 (Dense)	(None, 40)	3240
activation_2 (Activation)	(None, 40)	0
dropout_2 (Dropout)	(None, 40)	0
dense_3 (Dense)	(None, 20)	820
activation_3 (Activation)	(None, 20)	0
dropout_3 (Dropout)	(None, 20)	0
dense_4 (Dense)	(None, 1)	21

Total params: 28,161
Trainable params: 28,001
Non-trainable params: 160

- 過程



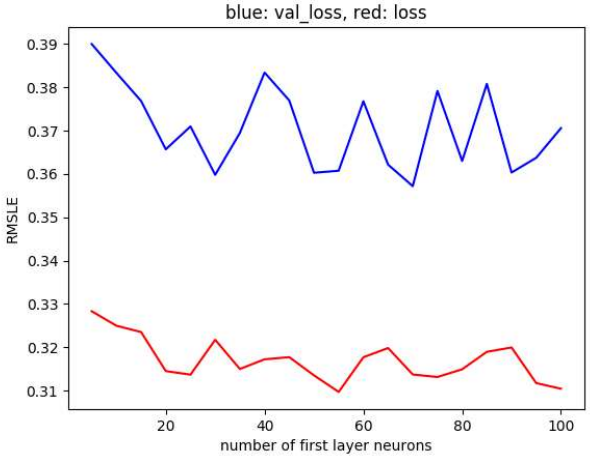
- Training loss: 0.3086

- Validation loss: 0.3110
- Kaggle public: 0.46182

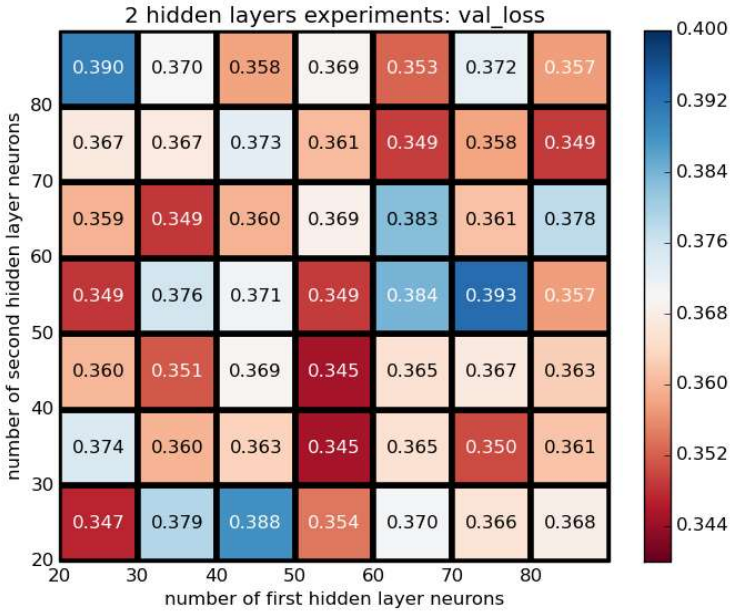
Experiments and Discussion

❑ **DNN 結構實驗**

❖ 以下實驗為搜尋DNN結構做的實驗，以訓練100個epoch做的正確率做為參考
❖ 由於實際在測試時，會遇到卡在很高的minima，所以實驗中每一種神經元數目組合都會跑好幾次，取loss最低者來評估(不使用val_loss，因為在訓練時最小化的函數是loss而不是val_loss)
❖ 無隱藏層 loss: 0.4356, val_loss: 0.4902
❖ 一個隱藏層:



❖ 一個隱藏層所需要的隱藏層神經元數目並沒有顯著影響，除非是太少神經元數目。另外可觀察到使用一個隱藏層的表現明顯好過無隱藏層的表現
❖ 兩個隱藏層：

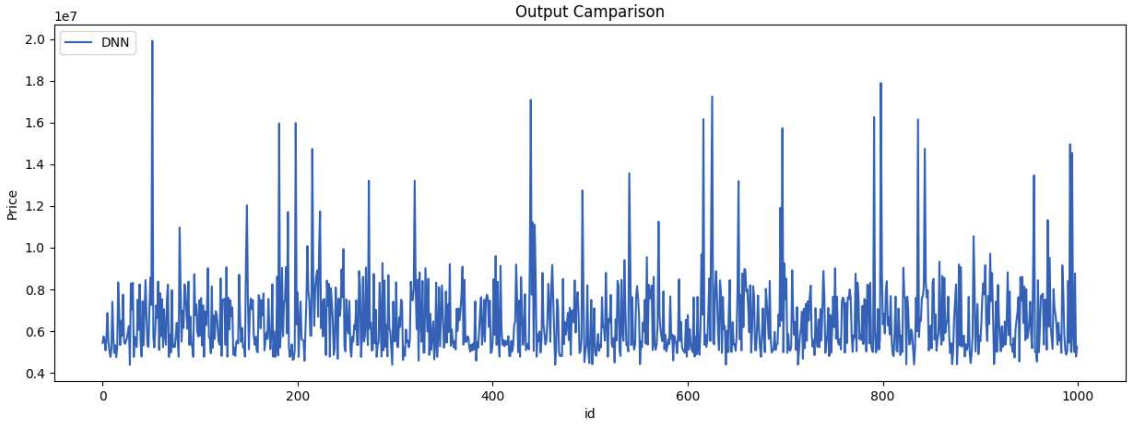


❑ DNN 易overfit

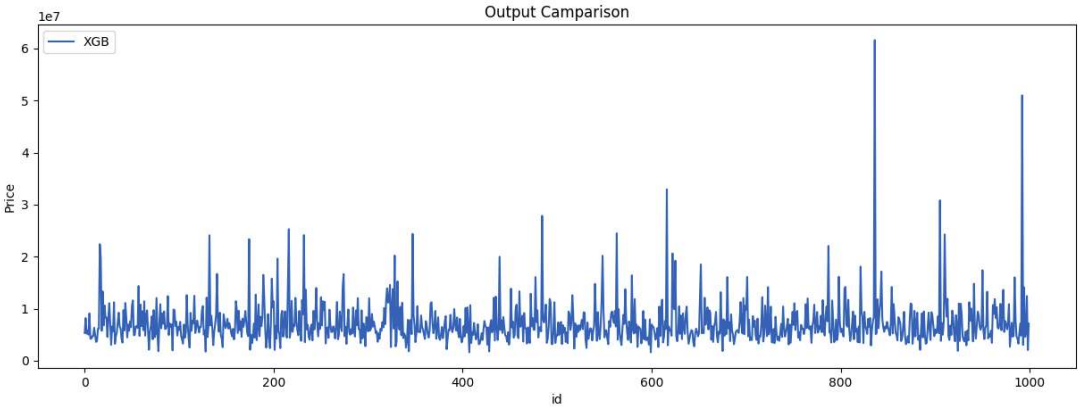
- ❖ 在我們嘗試使用DNN的時候發現，DNN很容易就會overfit，但有時候在validation set的上面表現又特別好，於是我們嘗試從validation的過程中選一次結果特別好的，也就是做到250epochs validation的分數大概在0.29~0.32之間，並將此model拿來預測test data，但是即便如此在kaggle public testdata上也只拿到大概0.46的分數，因此我們最後決定使用XGBoost。

❑ DNN & XGB output comparison, 極端值調整

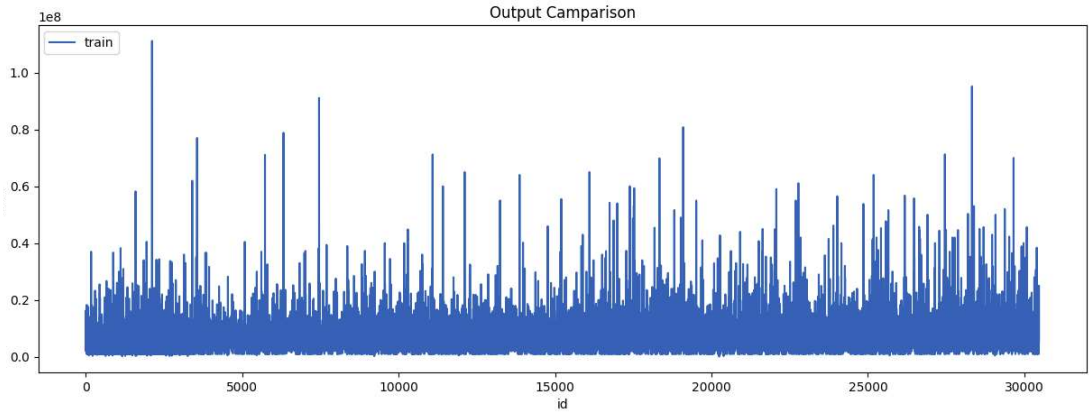
- DNN



- XGB



- Training data



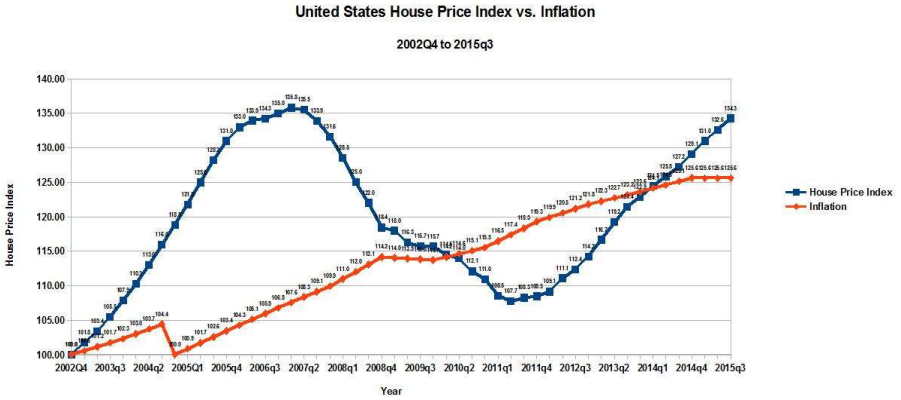
- ❖ 由上面DNN和XGBoot的圖可以知道，XGBoost會預測出一些比較極端的房價，而從training data的圖可以知道確實會有比較極端的情況存在，我認為DNN由於極端training data的資料量不夠大，因此再預測的時候沒有辦法應付這種狀況。

- ❖ 利用這個結論我們做了一個猜測，將XGBoost預測出price > 20,000,000的房價再乘上1.1，因為看起來極端房價的幅度沒有像training data那麼大，結果只用了一個submission public leader board的分數就從0.30900衝到0.30867。

❑ **捨棄macro.csv改而使用HPI以及fit public test data**

- ❖ 由training data每個月的平均成交價格，可發現每個月的平均值其實是相差很大的，但是會有一定的趨勢，造成這樣波動的原因有兩個，一個是國內的通貨膨脹率，另一個是投資客炒房的房價指數。由美國的次級房貸事件，可了解到房價與GDP、CPI、Inflation rate比較沒有直接的關係，而像這種房價崩盤是無法由過去的總體經濟資料所得知，所以使用俄羅斯官方所提供的House Price Index先對training data做初步的調整。

- ❖ 經過對public和private test data分布的測試，可以發現是隨機的，也就是說每個public test data每一個月都有，因此我們可以透過public test data來調整output的mean，期望能使private的分數提升。

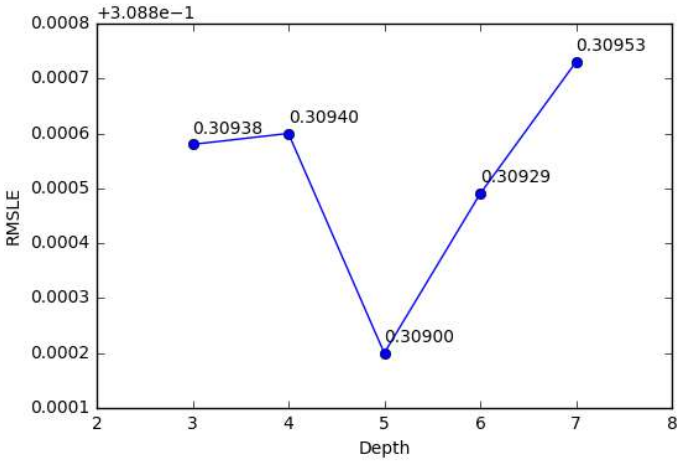


(USA Subprime mortgage crisis 2007 ~ 2010) Reference:
http://systemisbroken.blogspot.tw/2016/03/looks-like-australias-housing-market-is.html

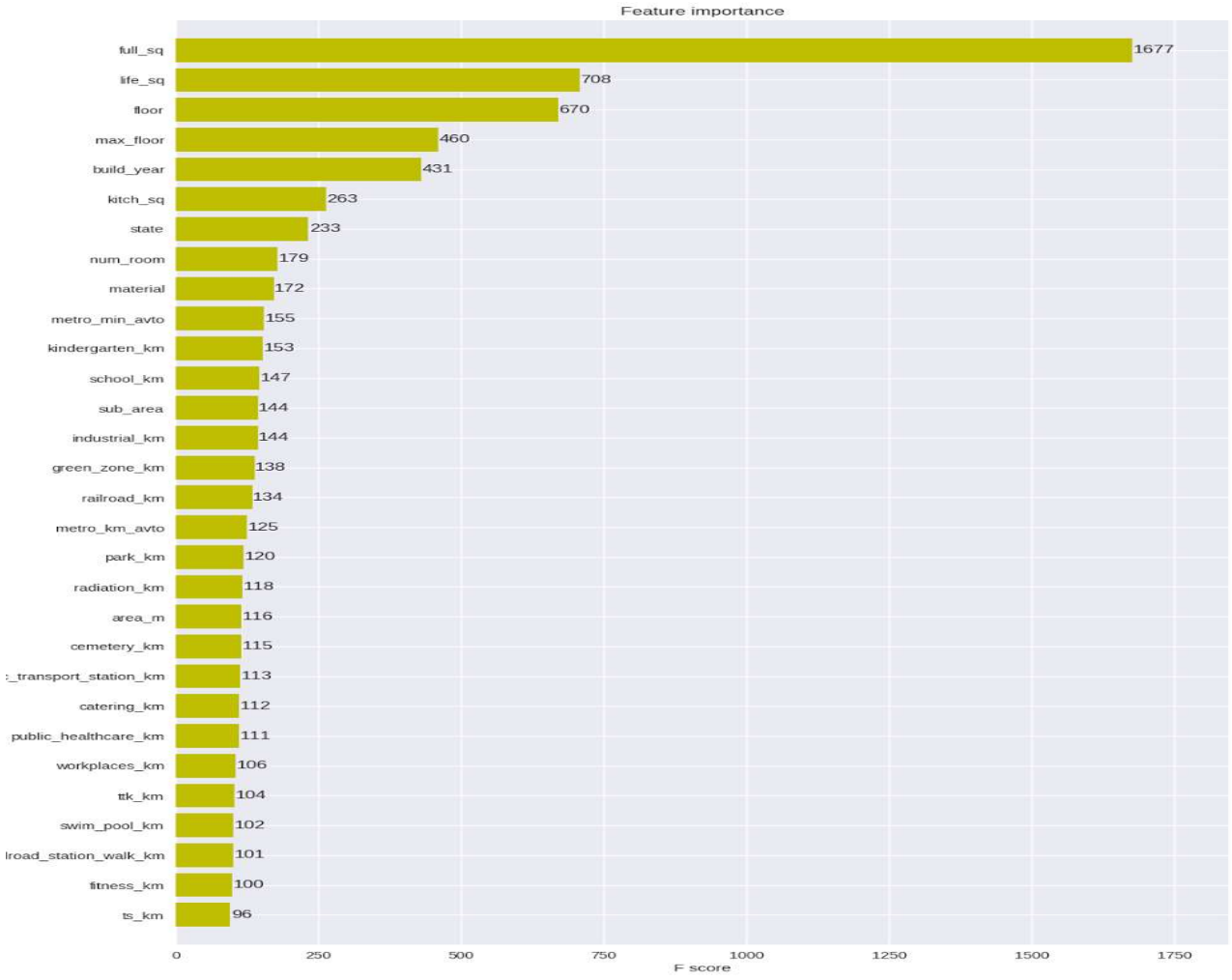
- ❖ 缺點: 這個方法雖然可以有效的提升準確度，但是通用性並不高，因為通常預測房價這件事，不應該要有未來的資料可以做validation (puclib)，也不應該要有外來的

總體經濟資訊，macro.csv給了2016的資訊相當不合理，House Price Index也被Kaggle認定為合法資料，我們認為主辦方經過這次比賽應該會好好檢討。

❑ **XGBoost 不同深度的分數分布**



❑ **XGBoost 做為base learner 篩選 feature importance (top 30)**



❑ **Feature 跟 price_doc 的 Correlation(top 30)**