**社群平台觀看次數預測**

臺大國企所 陳帝文 r10724050

**摘要**

本次作業是要預測社群平台上的觀看次數。在特徵選取方面主要是參考給予的資料特徵並且依照自身對社群平台觀看次數的認知進行一些假設後，計算出新的特徵，再用此些特徵利用lightGBM預測觀看次數。除此之外，利用遷移學習的想法訓練一個可以讀取照片後預測出觀看次數的神經網路。最後再結合兩者的分數進行優化後得到最終結果。

1. **數據觀察**

我們有的特徵有userID，Title，Tags，

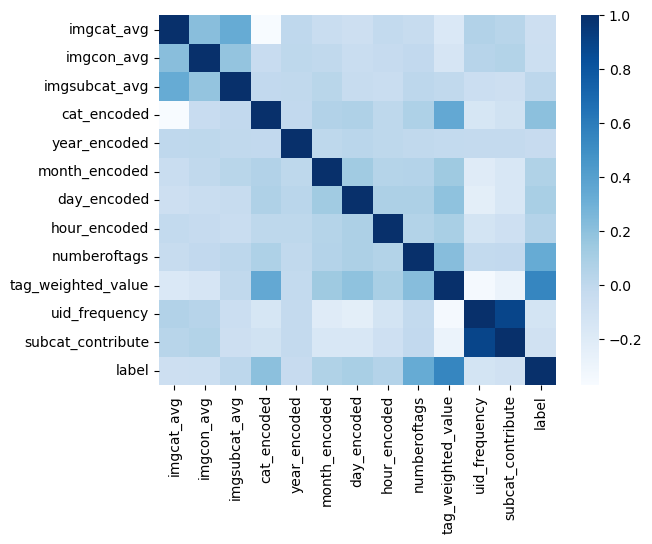
Category，Concept，Subcategory，Postdate和照片。首先，我對這些特徵進行下列假設:

1. 照片為主的社群平台，像是instagram，主要是照片影響觀看次數。人們不會因為Title的好壞決定是否觀看，因此這個特徵省略。
2. User如果有比較多的發文，會累積一定的粉絲，因此觀看次數較高。所以我計算userID的出現次數給予新的特徵，uid\_frequency。
3. Tags的數量越多，越有機會被人看到。但也要計算tags品質的好壞，假如tags都是一些自創或冷門的文字，並不會吸引到其他人。依照此邏輯，我計算numberoftags和tag\_weighted\_value。前者主要是代表tags的數量，後者使用target encoded的方法先去計算每個tag的value，再依照該tag在所有資料中出現的次數給予權重進行平均。
4. 為了提取Subcategory的訊息，我假設如果有某個用戶對某一Subcategory的貢獻太大，也就是大部分的人對此分類並無興趣，只有少部分的人使用此分類特徵，因此受到觀看的次數較少。依照此邏輯我計算了subcat\_contribution的特徵。
5. 若照片的內容和分類相關性較低，則有可能是分錯類，影響觀看次數。例如照片的內容是狗，但分類卻是Travel，mitt，Baseball，則會使喜愛狗的人沒辦法觀看到該照片。依照此邏輯我計算了imgcat\_avg，imgcon\_avg，imgsubcat\_avg，計算方法會在實驗方法中解釋。

除了上述的假設外，我也計算了category和postdate的target encoded feature，已提取其中訊息。為了觀察依照人為概念計算出來的特徵是否有效，表(一)是將上述特徵和labels進行線性迴歸後的結果，可以看到imgcat\_avg，numberoftags，tag\_weighted\_value和uid\_frequency十分顯著。雖然其他特徵的顯著性較低，但線性迴歸只能捕捉線性關係，使用機器學習的方法可以更好捕捉非線性關係，因此仍然可以嘗試將這些特徵用來預測。圖(一)則可以看到個特徵和觀看次數的相關性。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | coef | std | t |  |
| imgcat\_avg | 1.583 | 0.476 | 3.325 | 0.001 |
| imgcon\_avg | -0.073 | 0.249 | -0.292 | 0.771 |
| imgsubcat\_avg | 0.243 | 0.277 | 0.879 | 0.380 |
| cat\_encoded | 0.120 | 0.036 | 3.306 | 0.001 |
| year\_encoded | -1.085 | 0.156 | -6.967 | 0.000 |
| month\_encoded | -0.043 | 0.086 | -0.495 | 0.620 |
| day\_encoded | -0.015 | 0.058 | -0.269 | 0.788 |
| hour\_encoded | -0.032 | 0.089 | -0.363 | 0.716 |
| numberoftages | 0.033 | 0.001 | 32.370 | 0.000 |
| tag\_weighted\_value | 1.593 | 0.024 | 67.548 | 0.000 |
| uid\_frequency | 0.003 | 0.001 | 4.365 | 0.000 |
| subcat\_contribute | -0.072 | 0.255 | -0.280 | 0.779 |

表(一)特徵線性迴歸結果

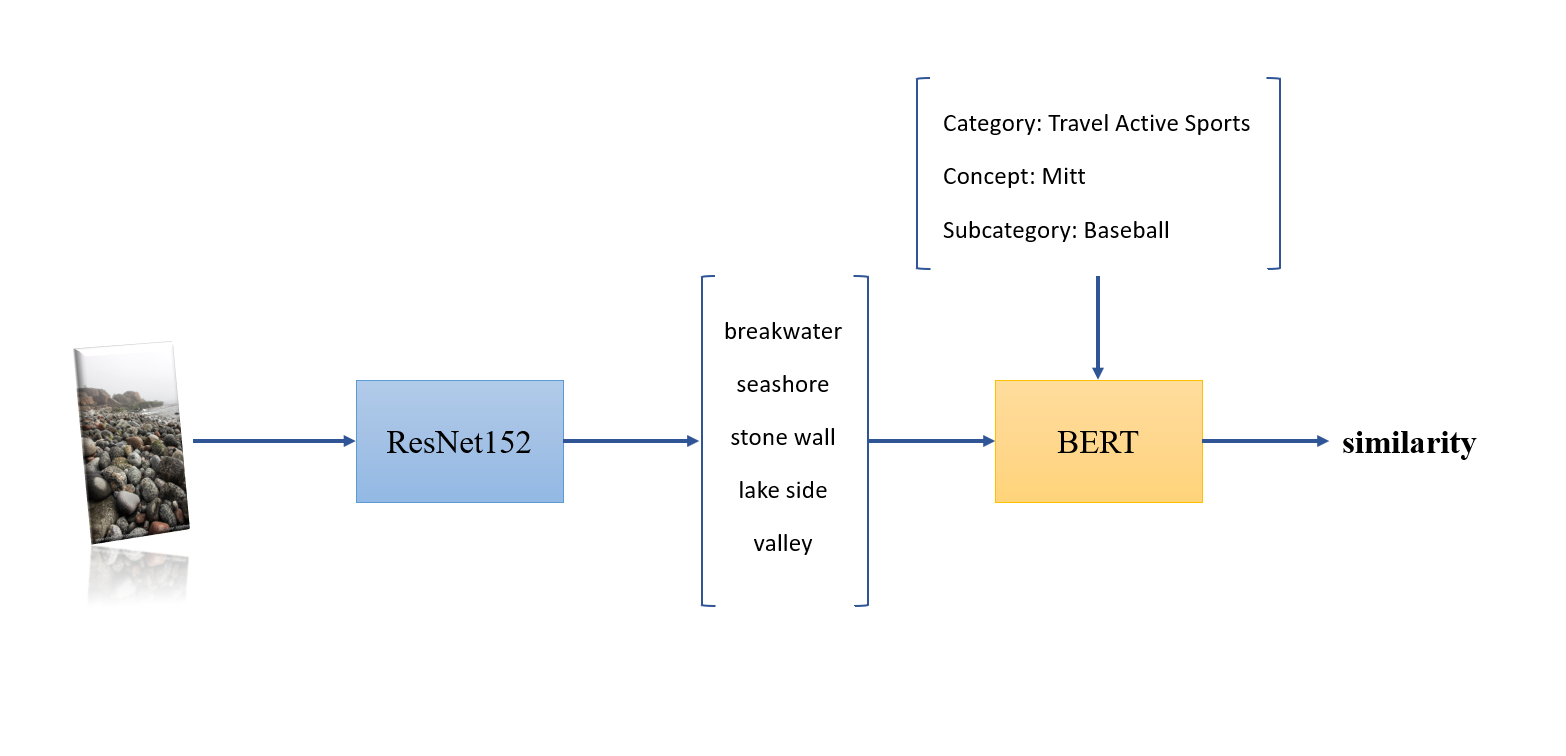


圖(一) 各特徵和觀看次數的相關性

1. **實驗方法**

* *相關性模型*

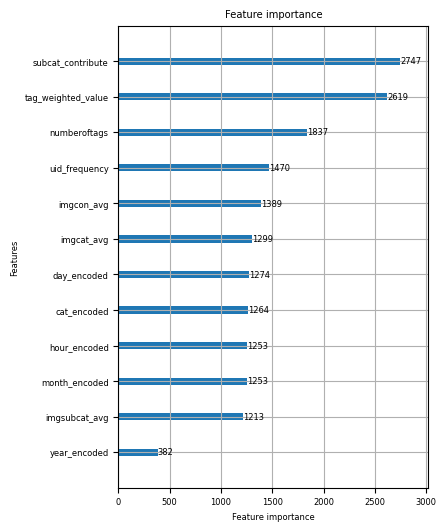
前面有提到我有計算圖片與Category，Concept和Subcategory的相關性。首先，我使用pretrained ResNet152模型和imagenet1000文本，讓模型分析圖片後預測出代表圖片的字，並且取前五大機率高者；接著使用BERT去生成代表這些字和Category，Concept，Subcategory的向量。最後去計算Category，Concept和Subcategory分別和這五個字的cosine similarity，並取平均。圖(二)是整個模型的流程圖。



圖(二) 相關性模型流程圖

* *結構化特徵模型*

得到結構化特徵資料後，我使用lightGBM去預測觀看次數[[1]](#footnote-1)，圖(三)是該模型對各參數的重要性排序圖。

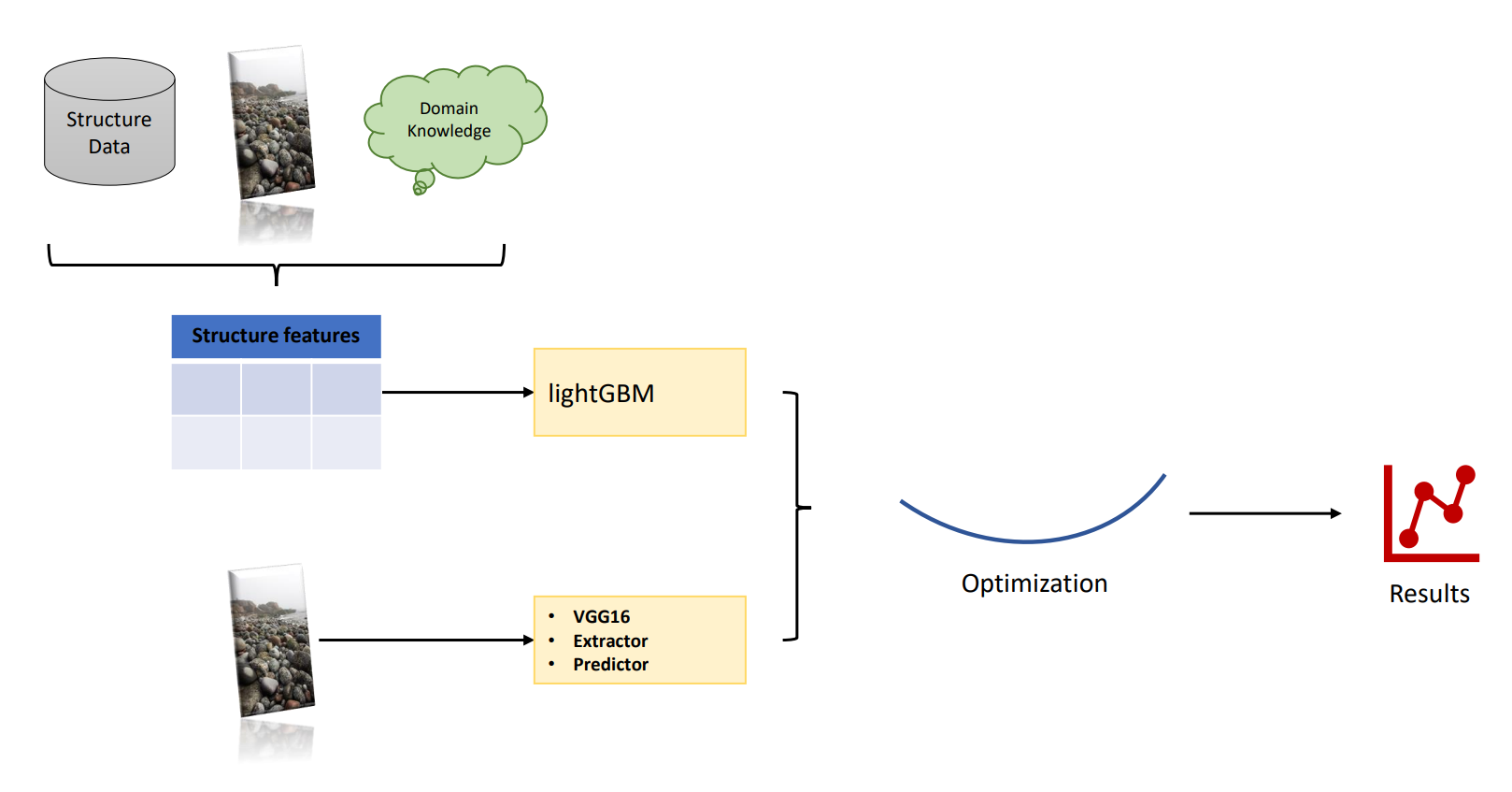


圖(三) 特徵重要性比較

圖(三)中可以觀察到對於lightGBM模型而言，貢獻度特徵和tags相關的特徵都很重要，這符合前面假設的情況。

* *非結構化資料模型*

除了結構化特徵外，仍然有充滿資訊的圖片資料。為了處理此資料，我使用vgg16 pretrained model接extractor和predictor。Extractor的部分主要是使用convolution layers去將pretrained model的結果引導到這份資料集；predictor則是將extractor萃取出的向量使用linear layer去預測觀看次數[[2]](#footnote-2)。最後，圖(四)為整個預測模型的成圖。



圖(四) 預測流程圖

1. **實驗結果與分析**

* *圖片模型預測*

在訓練時，我將資料分成training data和validation data，然而不論我嘗試幾次都會發現epoch 1的validation error會比繼續訓練下去的還要小，例如epoch 20或epoch 50等。為了觀察是否是因為特定validation data造成的緣故，我使用k-folds，使 去計算各epoch的validation error分布，圖(五)為此結果。由圖(五)可以看到在epoch 1的validation error會比訓練更多次的結果還要來的好，因此我選擇使用epoch 1來當作testing data的預測模型。

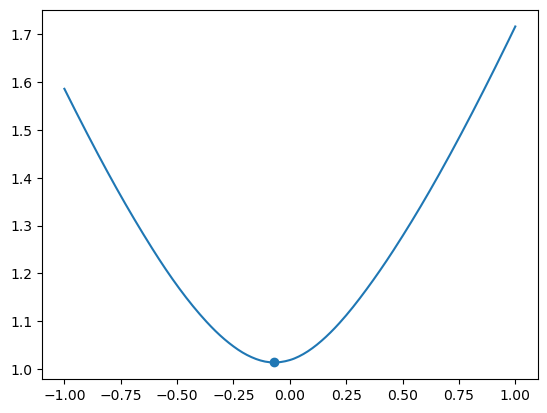
* *結構化特徵預測*

在非結構化特徵預測中，我並未將資料分成training data和validation data，而是直接使用public testing data當作我調整超參數的方法。在不加入圖片模型，只考慮結構化特徵的模型預測，其在testing data的最佳結果為2.0165。

* *最佳化結果*

在得到分別訓練不同資料的模型和 後，可以解下列方程式:

此處為了方便起見，我使用brute force法算出下，所有訓練集中的MAE，得到圖(六)。



圖(六) 圖片模型的最佳權重

由圖(六)可以看到最佳的權重是圖片模型為-0.07，但加上可能overfitting的結果，基本上此結果表示完全使用結構化特徵的預測結果即可。將此最佳化結果丟入testing data觀察可以得到2.02101，比完全使用結構化特徵的結果來的差。

為了使預測模型能夠加入圖片特徵進行預測，我將圖片模型給予的產出從觀看次數的預測改變成只擷取extractor輸出的向量，。將該向量與非結構化特徵合併後，丟入一個新的預測模型進行最後預測。

* *不分群預測結果*

表(一)展示了不分群下直接使用模型預測房價，訓練資料中MAE前五大的鄉鎮縣市和總訓練資料的MAE。由表中可以看到只有四筆資料的石門區誤差極大，仔細觀察後發現石門區的預測結果嚴重高估。另外，信義區，中正區等高價區則是被嚴重低估，因此有分開處理的打算。

|  |  |
| --- | --- |
| 鄉鎮縣市 | MAE |
| 士林區 | 19152.24 |
| 大安區 | 19315.72 |
| 中正區 | 19728.81 |
| 信義區 | 21251.44 |
| 石門區 | 22977.26 |
| 訓練資料MAE: 12160.59 | |

表(一) 不分群模型的MAE比較

* *分群預測結果*

首先，觀察從分聚和分類預測出的0和1[[3]](#footnote-3)，可以發現大多數新北市的房子都被分類在群，而大部分臺北市的房子和新莊，板橋等新北高房價區則是被分類在群。表(二)則呈現了先分群再預測的模型下的MAE前五大鄉鎮市區和總訓練集的MAE。和表(一)相比，表(二)中信義，中正，大安的誤差大幅下降且總訓練集的MAE也大幅下降。

|  |  |
| --- | --- |
| 鄉鎮縣市 | MAE |
| 大同區 | 12292.83 |
| 中正區 | 12473.18 |
| 松山區 | 12534.11 |
| 信義區 | 12811.44 |
| 大安區 | 13928.54 |
| 訓練資料MAE: 8485.91 | |

表(二) 分群模型的MAE比較

雖然兩階段模型的預測結果大幅進步，但從分類結果中可以看到仍有少數新北市較偏遠的地區被分類到群，像是石門區有一個，蘆洲有兩個，金山有一個等。因此我決定使用武斷的方式將訓練集分成臺北市和新北市，再訓練兩個預測模型。

* *武斷分群預測結果*

結果[[4]](#footnote-4)顯示用此方法下，大安區，信義區等臺北市的房價誤差相較於前一個方法高，但新北市的各鄉鎮市區則是下降，比較明顯的有淡水區，林口區和中和區等，如表(三)所示。我猜測由於此訓練集大多數都是新北市的房子，若使用此方法，臺北市的訓練集則減少，因此臺北市房價誤差上升。

|  |  |  |  |
| --- | --- | --- | --- |
| 區域 | 模型 | 武斷 | 數量 |
| 淡水 | 4920 | 4823 | 2322 |
| 林口 | 5015 | 4981 | 1104 |
| 中和 | 8297 | 8136 | 2036 |

表(三) 新北市部分區域MAE變化

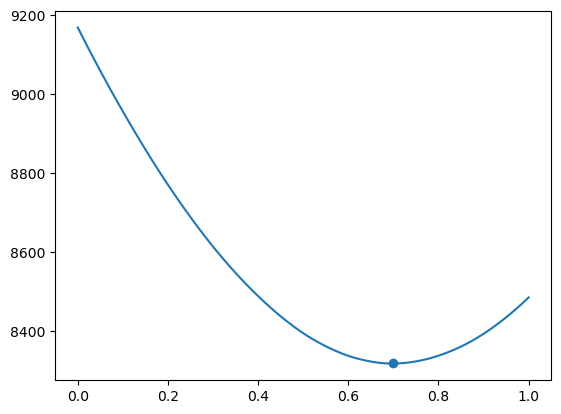
由於武斷的方法可以將新北市的房屋價格預測的不錯，加上資料中大部分屬於新北市，因此我決定結合前一個分群法和武斷分群法，有點類似ensemble的概念得出最終結果。

* *集成模型*

此處集成模型為兩個模型建立，分別標示為 和 。而我所需解的如下方程式表示，

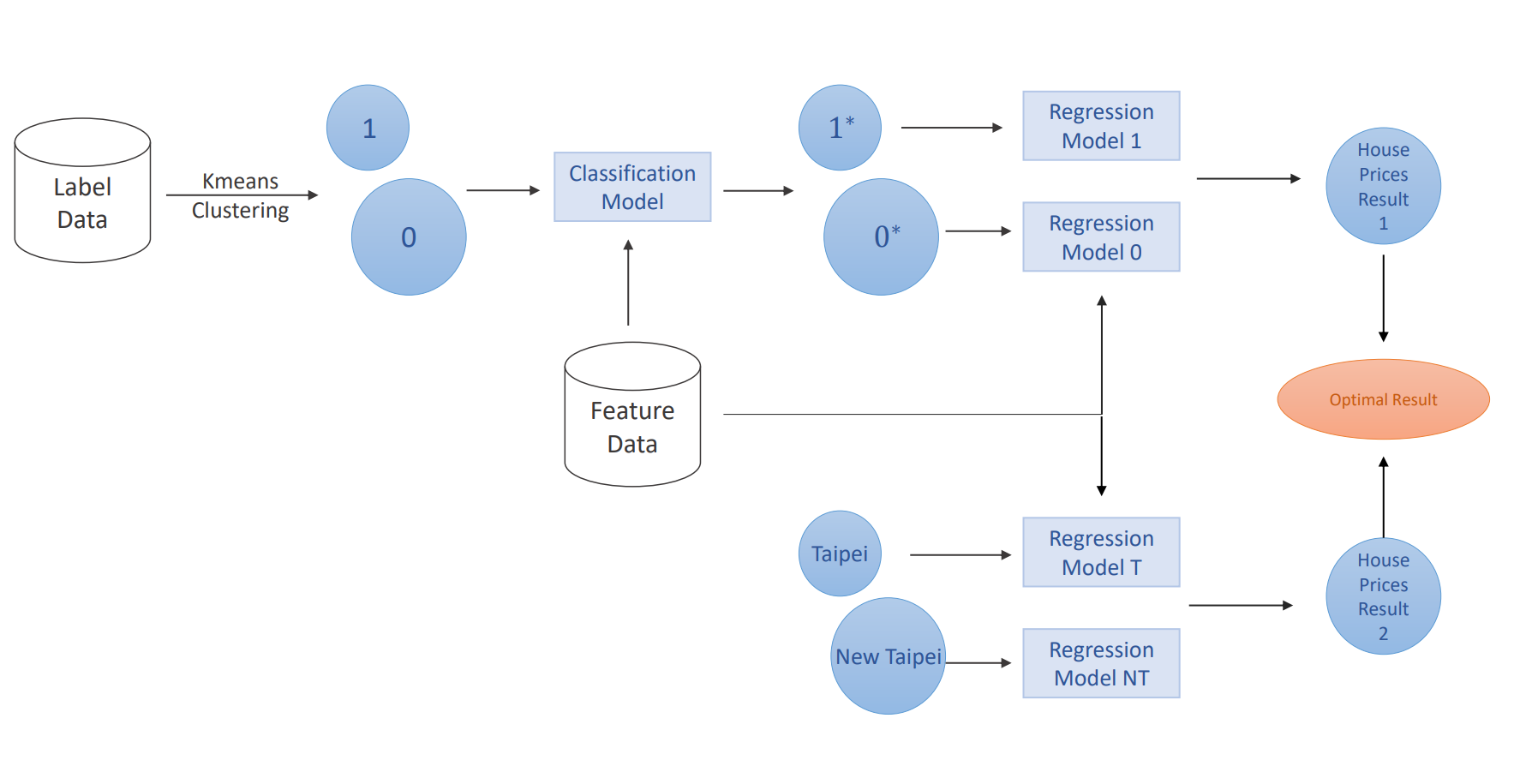
此處為了方便起見，我使用brute force法算出下，所有訓練集中的MAE，得到圖(六)。最後，將得到的最小訓練MAE下的權重當最佳權重得到最終結果。

從結果中可以看到，大安，信義，松山等房價誤差雖然仍比模型分群的模型大，但比武斷模型預測的進步。更重要的是，不知道什麼緣故，新北市各區的訓練資料房價在集成模型中預測誤差比武斷模型來的小很多，結果在ipynb檔中可以觀察到。



圖(六) 各權重下的訓練集MAE

最後，圖(七)為最終預測的流程圖。



圖(七) 最終模型流程圖

1. **總結與討論**

此次作業中，我認識到使用兩階段模型是如何找出資料中的隱狀態並且藉由此優化預測結果。但是，不可避免地，本次實驗仍然存在一些問題。首先，由於訓練資料太少，尤其是在分群後只會剩下6000筆資料，若是我再事先將資料分成training data和validation data則會造成誤差太大，自認為用此調參數也失去意義。因此，我選擇使用全訓練集訓練，並且使用public testing dataset來觀察是否overfitting。需注意的是，我使用public testing dataset並不是用來調整參數，而是用來觀察上述不同模型想法是否有更佳。表(四)是各種模型在public testing dataset的表現。

另外，集成模型使用的是訓練集資料下的最優MAE所得到的權重，有overfitting的風險。但由於我將此最佳權重丟入public testing dataset後得到的結果仍比模型分群，武斷分群和權重各半優異，因此我選擇相信此最佳權重。最後，我也嘗試使用CAT boost model來完成兩階段預估模型，可以看見在相同number of estimators下，其效果比lightGBM優異。

|  |  |
| --- | --- |
| 模型 | testing dataset MAE |
| 不分群 | 62338.63 |
| 模型分群 | 62210.34 |
| 武斷分群 | 61665.12 |
| 最佳權重 | 60905.67 |
| 最佳CAT | 60556.43 |

表(四) 各模型在public testing dataset之MAE

1. **參考文獻**

[1] DiCiurcio, K. J., Wu, B., Xu, F., Rodemer, S., & Wang, Q. (2024). Equity Factor Timing: A Two-Stage Machine Learning Approach. *The Journal of Portfolio Management*, *50*(3), 132-148.

1. 模型參數可以參考StructureDataModel.ipynb檔。 [↑](#footnote-ref-1)
2. 模型架構和參數可以參考Image Model.ipynb檔。 [↑](#footnote-ref-2)
3. 由於版面限制，欲檢視者可以查看ipynb檔。 [↑](#footnote-ref-3)
4. 由於版面限制，欲檢視者可以查看ipynb檔。 [↑](#footnote-ref-4)