# 國 立 臺 北 科 技 大 學

NATIONAL TAIPEI UNIVERSITY OF TECHNOLOGY

課程名稱: 機器學習

實習單元名稱: House Sale Price Prediction Challenge

班級: 電四乙

姓名及學號:劉昆琳 107310203

中 華 民 國 110 年 11 月 17 日

# 一、作法說明

此次實驗參考「Stacking」的作法，其是整合多個回歸模型的整合學習技術，模型架構分做兩個部分，第一為**基礎模型**，第二為**元模型**，其中基礎模型利用整個訓練集做訓練，而元模型則將基礎模型的輸出結果作為特徵進行訓練；在本實驗中，基礎模型採用 DNN、KNN、及 LightGBM，最後利用 3 層全連階層(Fully Connected Layer)整合三個基礎模型的輸出。

深度神經網路(DNN)：模擬人類神經網絡的運作方式，可以將其拆為輸入層(Input layer)、隱藏層(Hidden layer)及輸出層(Output layer)，網路中的每一層都有許多神經元(Neuron)，而各層之間神經元與神經元的連接都是倚賴權重(Weight)和偏差(Bias)，權重可以使用梯度下降（Gradient descent）更新。

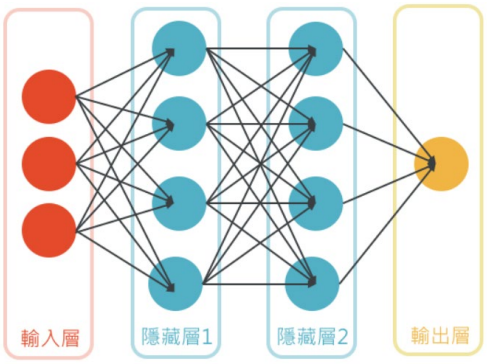


圖 1 深度神經網路(DNN)

K-近鄰算法(KNN)：KNN 最近鄰居法又譯為 K-近鄰算法，是所有機器學習算法中最簡單，也是使用最廣的演算法之一，其數學原理為透過 K 個距離最近的鄰居，並依照這些鄰居的房價決定賦予待測點的預測值；以下是 KNN 每個步驟的說明:

1. 利用歐基里德距離計算待測點與已知每個點之間的距離。
2. 計算完距離以後，需要決定常數 K，也就是有多少個鄰居被視為最接近的鄰居。舉例來說，如果選擇 K=3，那就會以距離最接近的 3 個點當作鄰居，並將這 3 個鄰居的房價取總和後平均，得到的值即為待測點的預測值。

LightGBM：其是基於樹的學習算法，所謂的決策樹是透過一連串的決策，分出不同結果所組成的樹狀圖，而隨機森林(Random Forest)則是以決策樹為基礎，透過多個隨機決策樹找到合適的答案，並將多個好結果的決策樹聚合成好的隨機森

林；隨機森林樹的生長方式是水平方向的，而 LightGBM 是垂直方向的，也就是說 LightGBM 生長的是樹的葉子，其他的算法生長的是樹的層次，這樣的好處是速度很快，可以處理大量的數據。

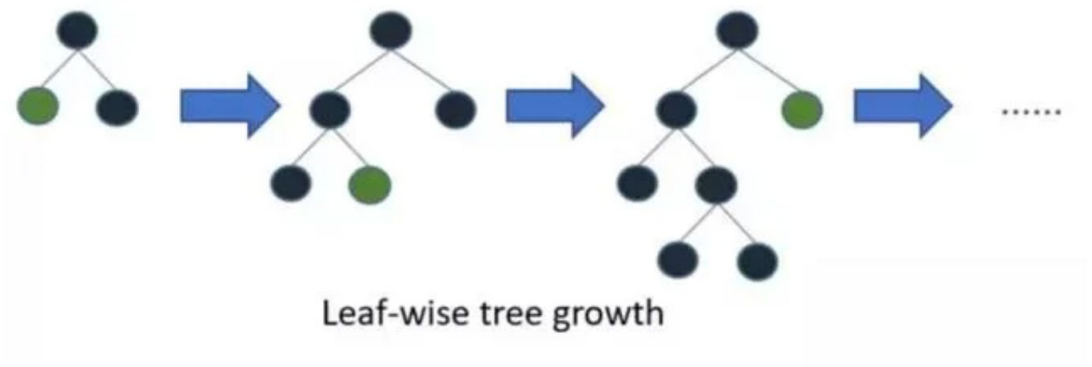


圖 2 LightGBM 樹生長方式

# 二、程式方塊圖與寫法

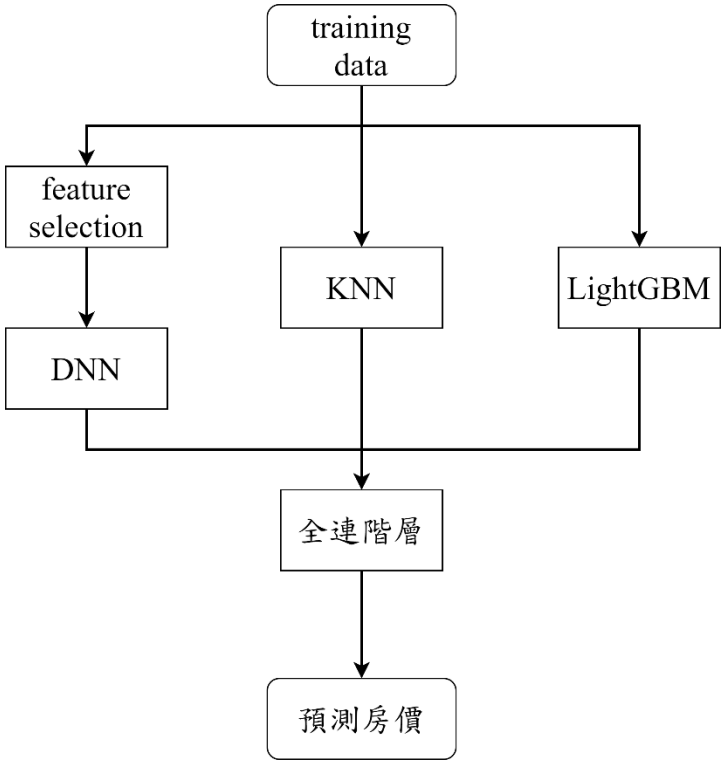


圖 3 實驗架構圖

feature selection： 由於提供的 22 種特徵，有些對於房價並無直接的關聯，如 year、month、day，所以須將這些特徵移除，如果沒有移除的話，可能會使模型預測錯誤，故使用 Pandas 套件中的 dataframe.corr()進行分析，並取出與房價相關性高的前 11 種特徵。

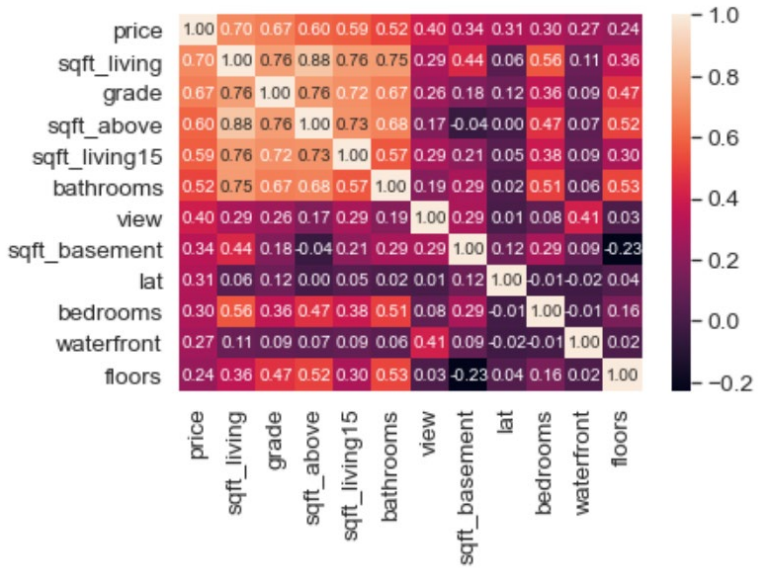


圖 4 與房價相關性高的前 11 種特徵

深度神經網路(DNN)：輸入層為 feature selection 後的 11 種特徵，並架設 5 層的隱藏層，神經元數分別是 64、128、512、128、64，激勵函數(Activation function)為 relu，此函數能將輸出轉為非線性方程式，公式為$f(x) = \max(0, x)$，最後由於輸出為房價預測，是一個線性的值，故輸出層的激勵函數將設為 linear。

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputs (InputLayer)          [(None, 11)]              0
_________________________________________________________________
dense (Dense)                (None, 64)                768
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_2 (Dense)              (None, 512)               66048
_________________________________________________________________
dense_3 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65
=================================================================
Total params: 149,121
Trainable params: 149,121
Non-trainable params: 0
_________________________________________________________________
```

圖 5 DNN 網路模型

K-近鄰算法(KNN)：本實驗經由多次的測試後，找到最佳值得 K 為 13，故將待測點距離最近的 13 個房價總和後平均，即為待測點的預測值。

LightGBM：採用 lightgbm 套件匯入，參數設計如圖 6，boosting_type 為指定學習器的類型，而設定為 gbdt 則是使用梯度提升樹；objective 為學習目標，由於此次實驗是回歸任務，故設為 regression；metric 是評估指標，設定為 l1、l2 分別是用 MAE 及 MSE 來評估；num_leaves 為葉子的各數，設定為 31；learning_rate 則是學習率，設定為 0.05；feature_fraction 為隨機對特徵採樣的比率，設定為 0.9；bagging_fraction 為在不進行重新採樣的情況下，隨機選擇部分數據，可以用來加速訓練及處理 overfitting，設定之值為 0.8；bagging_freq 即每 k 次迭代執行 bagging 算法，設定之值為 5；最後在訓練時則採用 early_stopping 的技術防止 Overfitting。

```
params = {
    'boosting_type': 'gbdt',
    'objective': 'regression',
    'metric': {'l2', 'l1'},
    'num_leaves': 31,
    'learning_rate': 0.05,
    'feature_fraction': 0.9,
    'bagging_fraction': 0.8,
    'bagging_freq': 5,
    'verbose': 0
}
```

圖 6 LightGBM 參數設定

全連階層整合上述三模型：將上述的輸出預測值作為整合模型的輸入層，並架設 3 層全連接層整合，神經元數分別為 8、32、8，且激勵函數為 relu，最後輸出層則是房價預測的最終結果，激勵函數將設為 linear。

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputs (InputLayer)          [(None, 3)]               0
_________________________________________________________________
dense_6 (Dense)              (None, 8)                 32
_________________________________________________________________
dense_7 (Dense)              (None, 32)                288
_________________________________________________________________
dense_8 (Dense)              (None, 8)                 264
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 9
=================================================================
Total params: 593
Trainable params: 593
Non-trainable params: 0
_________________________________________________________________
```

圖 7 整合模型

# 三、畫圖做結果分析

深度神經網路(DNN)：由輸出曲線圖可以看到，loss 及 val_loss 在最後的變化會變得平緩，且無 loss 下降，val_loss 上升的趨勢，故無 Overfitting。

```
loss= 79249.015625      val_loss= 82333.6328125
```

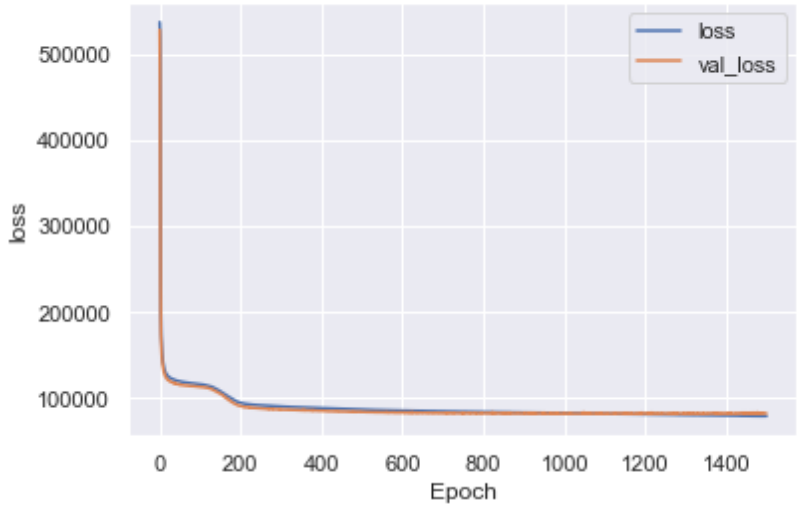


圖 8 DNN 的 loss 曲線圖

K-近鄰算法(KNN)：求得最佳值得 K 為 13，可以利用最近的 13 個鄰居算出待測點之值，並算出 loss 為 85526.24710782041。

LightGBM：由圖 9 可得知，LightGBM 的 loss 有在持續的下降，代表模型有學習到有用的特徵。

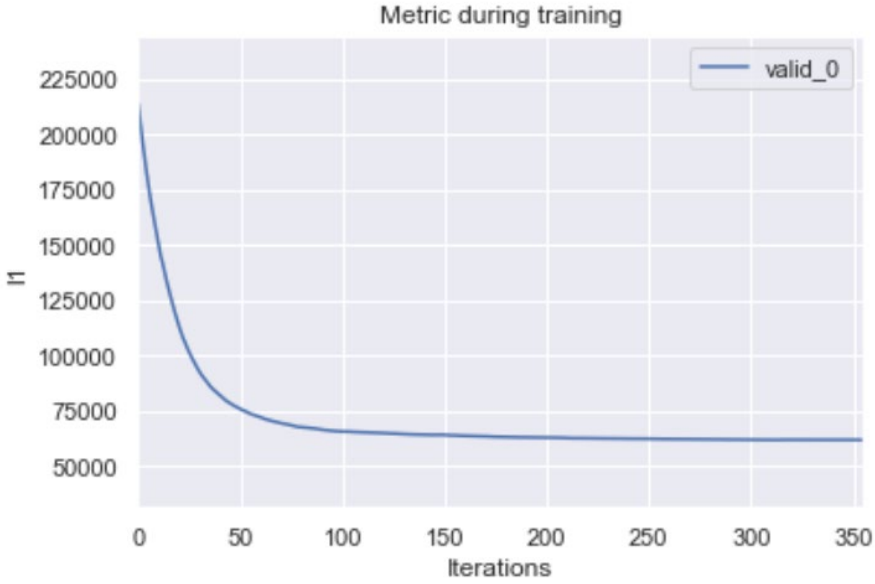


圖 9 LightGBM 的 loss 曲線圖

全連階層整合上述三模型 ： 透過簡單的模型將 DNN、KNN、LightGBM 的輸出結果整合，可以看到輸出的 loss 並無 Overfitting。

```
loss= 49721.25      val_loss= 61891.5078125
```

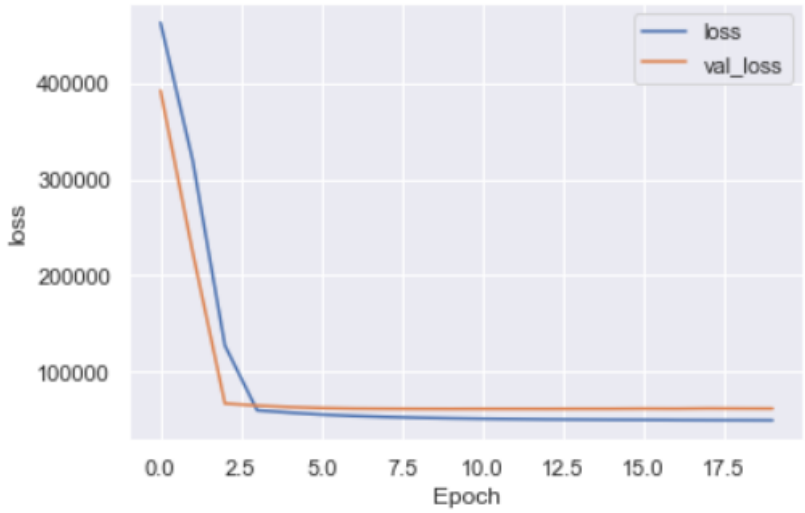


圖 10 整合模型的 loss 曲線圖

# 四、結論與改進方向

1. 加深網路，並增加 Dropout 層及 L2 正規化抑制 Overfitting，嘗試以此方式得到更佳的結果。
2. KNN 的做法雖然簡單且直觀，但只看距離就決定結果也過於單調，其與類神經網路雖同為監督式演算法，但 KNN 無法經過訓練過程學習資料特性，所以在準確度上會有天花板，故未來希望以其他種演算法取代 KNN，以不同的方式萃取特徵，以獲得更多更重要的訊息。