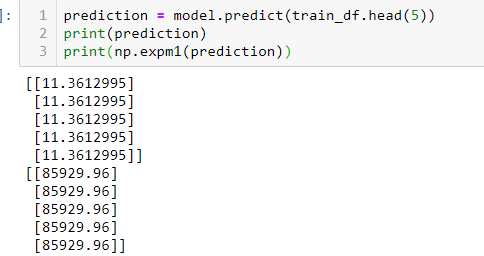
[碰到問題](" https://github.com/keras-team/keras/issues/6447)連結-模型為所有輸入提供相同的輸出

訓練出的其中一次模型預測前五個資料值輸出(輸出結果為自然底數的次方，房價為自然底數做次方完後)

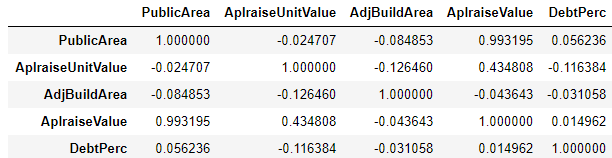


可能原因:

一 所提供的輸入代表的狀況，具有太多的結果(其他意義)，且這些輸入欄位除了公共區域面積(公設面積)具有高度相關性之外，其餘皆為低度相關性，但是沒有公設案件有1907筆。

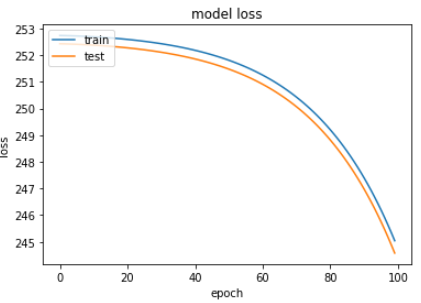
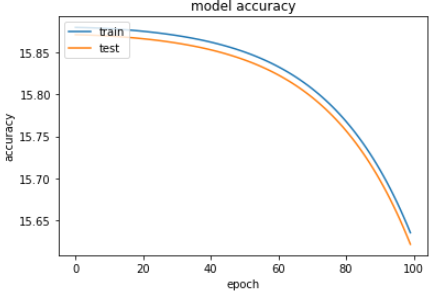
圖為從EXCEL塞選出的結果

下圖相關係數矩陣



這邊顯示公共區域有極高的相關性，但具有一半左有的數據都沒有公共區域面積所以所訓練出來的模型權重就呈現權重低的狀況，後面會提到。

二 輸入的參數範圍過廣(城市、鄉鎮)，導致預測收斂後無法得到該有的預測水準，即收斂後並無法代表真實情況。每個城市中的鄉鎮，各地區的價格都處在不同的水平上，代表要同時使用同一種模型達成預測效果是顯著是困難的。



MSE在每一次學習狀況 LOSS函數在每一次的學習狀況

三 無論輸入如何，都可以預測相同的“最佳”輸出。認為這可能是在學習訓練案例的平均目標值，或者類似的東西(patterns模式)，驗證 / 測試的MSE也是如此之低，從上方兩個圖中可以看到，模型在每一次的學習當中，似乎都有學習到甚麼，這似乎很奇怪，但是對於任何給定的案例，數據均具有相似的平均值，代表模型認為當所預測都為相同時可最小化損失函數，在給定輸入條件下，去符合所有輸入該有的輸出預測(有點像達到最小成本，但失去客製化分類的概念)，而這樣有可能的意味著輸入資料太過於一班、攏統，不具備到可以劃分的程度。

四　網絡越複雜，網絡調整的權重就越大，也代表類神經網路權重變調的機率越大。(EX:最後所更新的權重，無法具有分別能力，使之權重為零，指有激勵Bias到下一層)，認為此為主要原因。

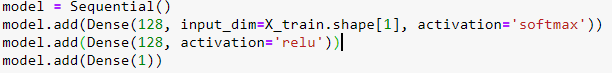
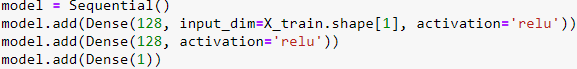
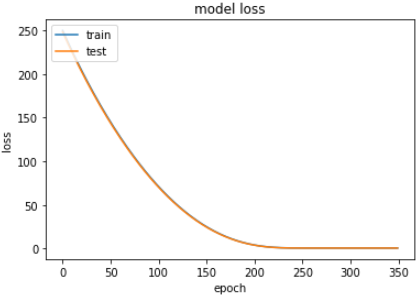
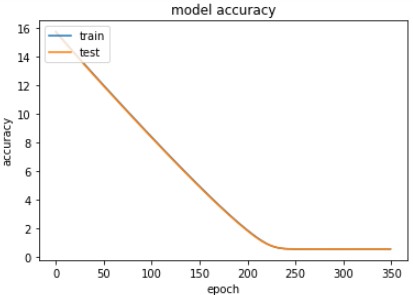
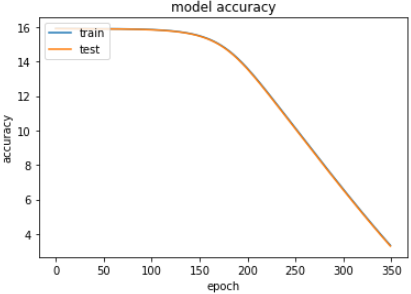
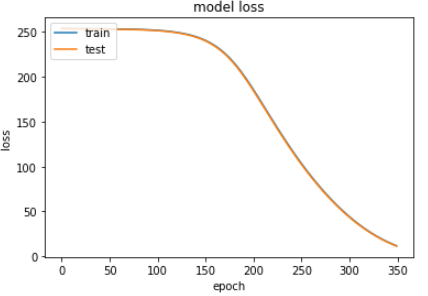
推測可以看下圖上方，我先以一次訓練模型，儘管第一次，我只用批次1個資料值去訓練，模型在訓練時LOSS函數值是如此高，MSE也是很高，可是驗證結果卻顯示LOSS值掉落非常低，驗證的MSE也是依樣掉落至非常低。而在後續訓練十次的結果卻顯示進步的相對非常慢，而再訓練十次的第一次結果也是一樣具有非常好的數值，與先前第一次驗證時的資料是相當近的，可以代表的就是訓練的模型非常有效，或是訓練的模型非常LOSS值的評估值。

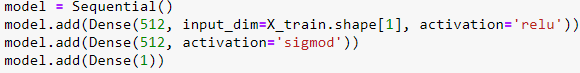
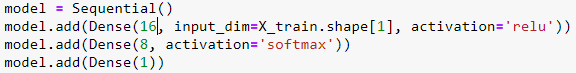
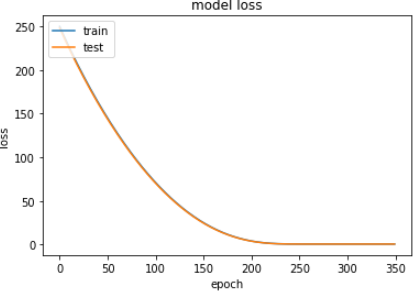
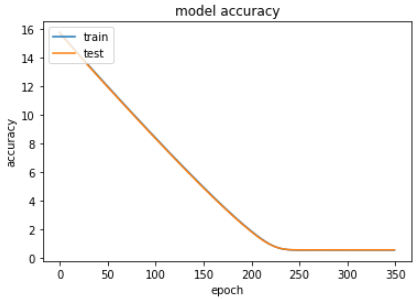
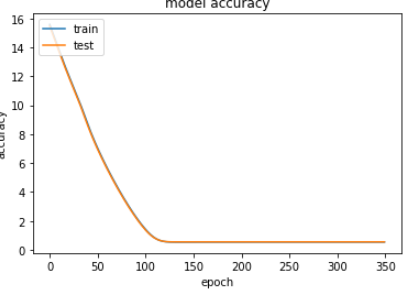
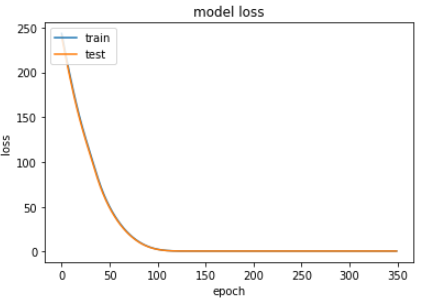


曾嘗試過的方案

1. 使用不同的激勵函數：目的是希望傳遞的過程不會因激勵的結果不好，而傳遞不出該有的資訊。包括:RELU、SOFTMAX、SIGMOID

下面我只列幾種(可以先往下看)

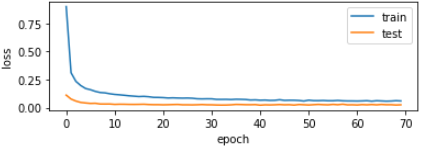
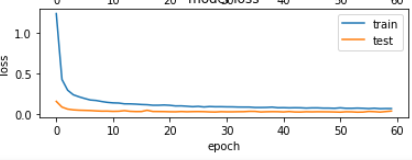
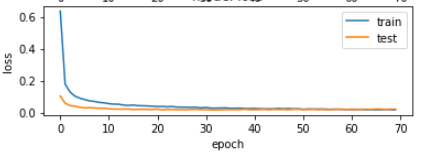
    



以上幾種是模型最後都落在平穩的LOSS值上，且都介在0.55左右，除了第一個還有學習的空間。另外他們的曲線似乎都是完美跌在一起，。

下圖是正常情況的loss function



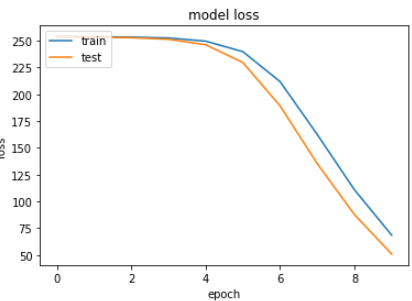
1. 採取較低的批次訓練：較低批次訓練已達成模型更新的方式。





1. 以線上學習的方式訓練資料：即batch\_size = 1，每筆資都會更新權重。這可

以很確定模型的學習狀況，學習狀況可以知道 ，不過線上學習的方法在學習是由於只有一筆，訓練模型都會花非常久，所以這邊我只用非常少的EPOCH去跑。

可以看到上圖，以批次資料是一次的線上學習模式，一般情況下測模型通常會在前期具有較高的學型狀況，隨著模型的複雜越來越深，所學的的將會越來越慢，也就是學習率將下降(可以看前面幾張正常良好模型情況下所會出的圖形)，而這邊顯示前三、四次的訓練，對預測結果是沒什麼進步的，而在後期的訓練結果卻進步卻異常前面的快，這也不符合常理。

注意:這邊繪圖會拿調最先前先跑一次的數據，因為後續跑十次的結果是相對相近的，前面進步的太多想要看學習率的狀況會看不到(Y軸比例尺的問題)，所以先拿掉第一次的繪圖數據，從的二次開始畫圖也就是指畫後續十次訓練的結果。

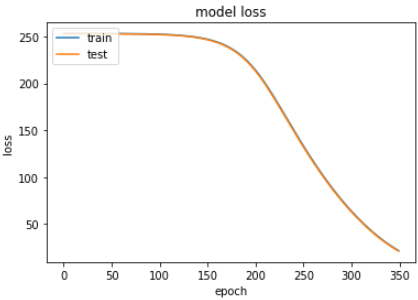
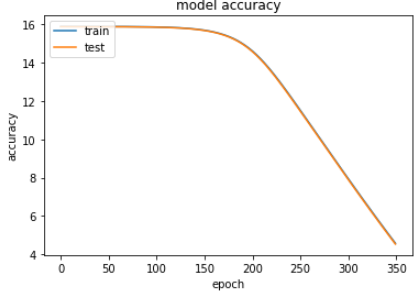
1. 使用較多層的神經元：已達到可以捕捉潛在更複雜的形式。

我有使用256、512個神經元去跑。

1. 使用較少層的神經元：目的是讓模型預測出的結果相對於多層來更簡單（通常會用此方式，是認為訓練完模型過度複雜，但實際資料維度過高，多神經元無法反映真實情況，改以較少的神經元只代表所有的特徵，這使得特徵較高的會稍較重要，反而可有較好的預測）。

我有使用8、16、64、128、256個神經元去跑。

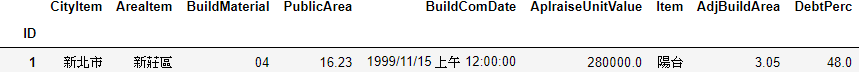
1. 使用較淺的網絡，然後使用較深的網絡：捕捉潛在更複雜的形式。這是AUTO ENCODER的概念。
2. 以單層感知積預測資料：MLP可能使模型過於複雜。

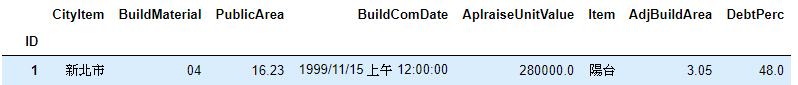


但模型最後還是預測出類似的結果

1. 以不同的欄位資料當作訓練的輸入:雖然大部分的欄位與房價的相關度不高，但還是看看有不同的欄位，是否有甚麼不同。

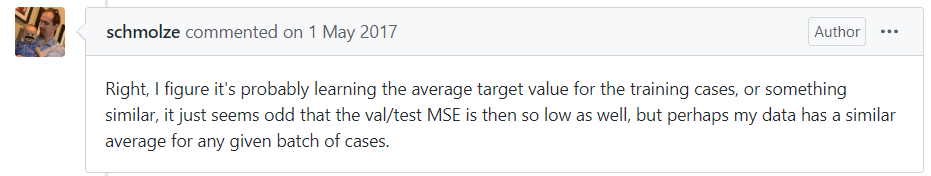
以下列幾個





注意:你可能會問，不嘗試DROP OUT類神經網路，因為我們的資料大部分都是單筆與此次相關的資料，跟分類是不同的，類別可能上一筆與下一筆是同一類別或是不同類別，再以批次訓練的方式將更好的訓練模型，所以DROP OUT在回歸問題上通常會倒置當次的訓練學習的狀況效率變的非常不好或是學習不到關聯，因為每次資料彼此都是不同的結果。

其中我所想到的方式與網路上給的意見(因為也有很多人遇到跟我一樣的問題)，我能試過的都試了，但所預測出來的結果就是所有輸入都會獲得同樣的預測輸出(也有很多人也是這樣，始終沒有改善這個狀況，讓模型可有更好的預測，因為這就是模型目前最好的結果)，其最根本的原因就是前面所提及的。

最初發問的人也提出了他的想法。

就是認為模型自己認為這樣對於所有資料能有最好的LOSS值。