Intelligent Integration of Enterprise Project 3

房屋售價預測模型之建構與驗證

108034540 吳欣蓉

國立清華大學 工業工程與工程管理學系

指導教授：邱銘傳 教授

摘要

房屋通常是一個人一生中所購買的最大和最昂貴的商品，而其價格是影響購買意願的最關鍵因素。因此本專題欲建立房價預測模型，幫助購屋者進行決策。藉由蒐集房屋類型、周邊環境及地點等特徵資料以預測房屋的售價。本研究使用深度學習方法中的神經網路回歸器建立模型，並使用 Ames Housing dataset 的公開資料集進行結果與效度驗證。

目錄

一、 緒論

房屋通常是一個人一生中所購買的最大和最昂貴的商品，而其價格是影響購買意願的最關鍵因素。房地產投資是近年很熱門的投資事業之一，對於投資客而言，未來房價的漲跌會顯著影響其投資意願。另外對於房仲業者而言，當其取得具備說服力的房屋未來價值資料時，將可增加其房屋銷售的成功率。因此本研究欲建立房價預測模型，幫助購屋者審慎評估購屋決策。藉由蒐集房屋類型、周邊環境及地點等特徵資料以預測房屋的售價。

針對此研究主題，5W1H 分析說明如下表：

表一、5W1H 分析

項目	內容
What	房價預測
Where	美國艾姆斯州的房屋資料
Who	欲購買房屋的民眾、房仲業者、投資客
When	欲出售或購買房子時
Why	房價是影響房屋購買意願的最關鍵因素
How	透過深度學習分析房屋的特徵資料，以預測房屋未來價值

二、 研究方法

本研究使用深度學習方法中的神經網路回歸器建立模型，並使用 Ames Housing dataset 的公開資料集進行結果與效度驗證。以下段落將分別說明資料來源及其架構分析、資料前處理過程以及模型建立與方法。

1. 資料來源與結構分析

本研究資料來源為 Ames Housing dataset 公開資料集，內有關於美國 Ames 州房屋的特徵資料及其售價，共 1460 筆訓練集資料、1459 筆測試集資料。此外公開資料集中也有一份欄位與代碼說明檔案，其部分內容如表二所示。

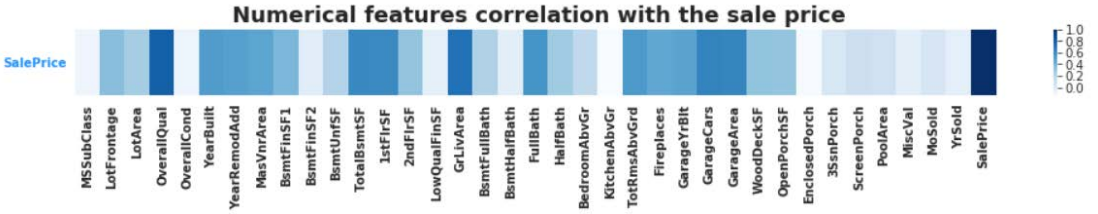
Id	MSSubClass	MSZoning	LotFrontag	LotArea	Street	Alley	LotShape	LandConto	Utilities	LotConfig	LandSlope
1	60	RL	65	8450	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl
2	20	RL	80	9600	Pave	NA	Reg	Lvl	AllPub	FR2	Gtl
3	60	RL	68	11250	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl
4	70	RL	60	9550	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl
5	60	RL	84	14260	Pave	NA	IR1	Lvl	AllPub	FR2	Gtl
6	50	RL	85	14115	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl
7	20	RL	75	10084	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl
8	60	RL	NA	10382	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl
9	50	RM	51	6120	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl
10	190	RL	50	7420	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl

圖一、Ames Housing dataset

欄位	說明
SalePrice	the property's sale price in dollars.
MSZoning	The general zoning classification
MSSubClass	The building class
LotFrontage	Linear feet of street connected to property
LotArea	Lot size in square feet
Street	Type of road access
Alley	Type of alley access
LotShape	General shape of property
LandContour	Flatness of the property
Utilities	Type of utilities available

表二、資料欄位說明示例

訓練集資料中共有 81 個欄位，如房屋類型、建造與售出年分、地坪與建坪、周邊道路類型，甚至是與鄰近主要道路的遠近等等，資料鉅細靡遺。其中包含數值型及類別型的資料，分別為 38 欄及 43 欄。首先針對數值型欄位觀察其與房屋售價之間的關聯性。由圖所示可發現”OverallQual”欄位，也就是房屋的材質與完工品質，與房價之間的關聯性最高。其次是”GrLivArea”欄位，也就是房屋的建物面積。



圖二、數值型特徵與房價之間的相關性

2. 資料前處理

2.1 資料清洗

針對訓練集與測試集資料進行清洗。

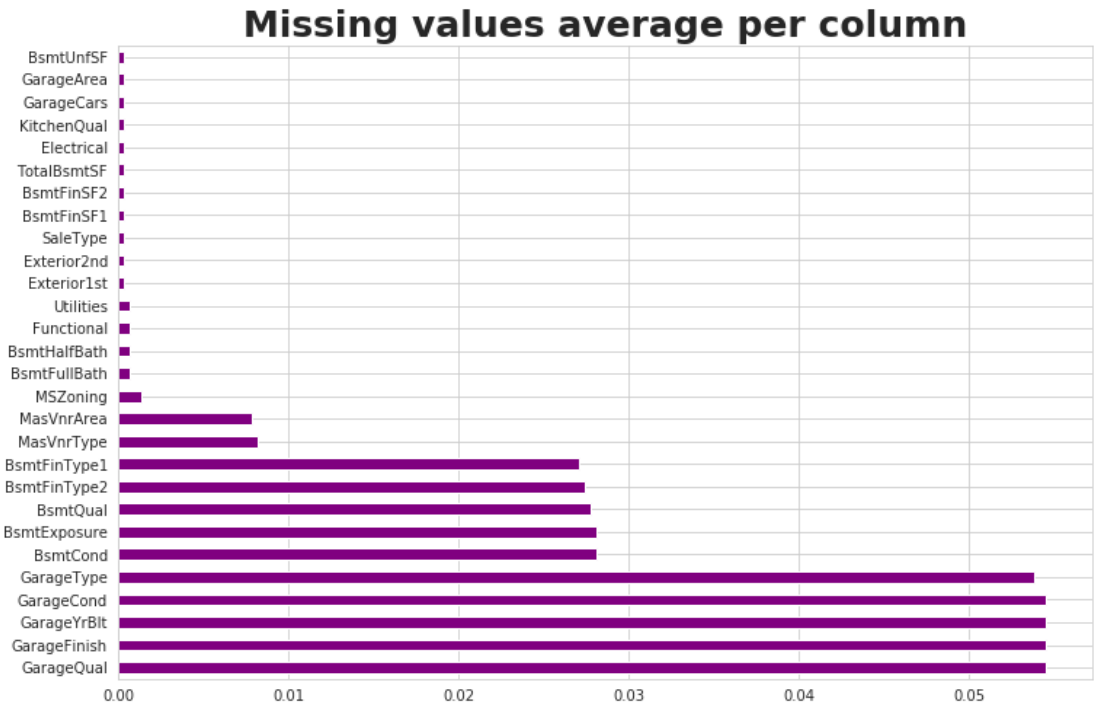
A. 刪除缺失值大於 90%的特徵，共刪除 6 個欄位

```
c=whole.dropna(thresh=len(whole)*0.9, axis=1)
print('We dropped ',whole.shape[1]-c.shape[1], ' features in the combined set')
```

```
We dropped  6  features in the combined set
```

B. 統計並補齊剩餘欄位的缺失值

```
allna = (c.isnull().sum() / len(c))
allna = allna.drop(allna[allna == 0].index).sort_values(ascending=False)
plt.figure(figsize=(12, 8))
allna.plot.barh(color='purple')
plt.title('Missing values average per column', fontsize=25, weight='bold' )
plt.show()
```



■ 數值型欄位

車庫建造年代：使用中位數替補缺失值

```
c['GarageYrBlt']=c["GarageYrBlt"].fillna(1980)
```

其餘欄位的缺失值均補 0。

■ 類別型欄位

針對有較少缺失值的欄位，用缺失欄位的前一筆資料替代

```
c['Electrical']=c['Electrical'].fillna(method='ffill')
c['SaleType']=c['SaleType'].fillna(method='ffill')
c['KitchenQual']=c['KitchenQual'].fillna(method='ffill')
c['Exterior1st']=c['Exterior1st'].fillna(method='ffill')
c['Exterior2nd']=c['Exterior2nd'].fillna(method='ffill')
c['Functional']=c['Functional'].fillna(method='ffill')
c['Utilities']=c['Utilities'].fillna(method='ffill')
c['MSZoning']=c['MSZoning'].fillna(method='ffill')
```

其餘欄位的缺失值均改為”None”

2.2 資料篩選與分割

- 首次建模只選用數值型特徵，因此從資料中篩選出其資料

```
print('Number of features in the whole data :', c.shape[1])
c1 = c.select_dtypes(exclude=['object'])
print('Shape of the whole data with numerical features:', c1.shape)
print("List of numerical features:",list(c1.columns))
```

```
Number of features in the whole data : 73
Shape of the whole data with numerical features: (2919, 35)
List of numerical features: ['MSSubClass', 'LotArea', 'OverallQual',
```

- 分割訓練集與測試集

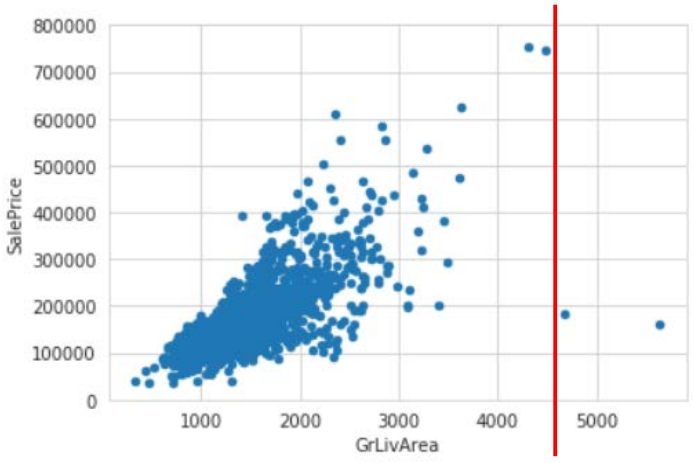
```
na = train.shape[0] # the number of rows of the original training set
train = c1[:na]
test = c1[na:]
```

```
train = pd.concat((train, y_train), sort=False,axis=1).reset_index(drop=True)
```

2.3 刪除離群值

公開資料集的敘述中，作者提及可針對”GrLivArea”欄位進行分析並刪除其離群值。

```
var = 'GrLivArea'
train.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```



如圖所示，刪除右下角中建物面積大，售價卻很低的兩筆資料。

```
train = train.drop(train[(train['GrLivArea']>4000) & (train['SalePrice']<300000)].index)
```

2.4 資料正規化

使用 MinMaxScalar 進行正規化，將資料轉化為 0~1 之間的數值。其中數值最大的轉換為 1，最小的轉換為 0。

■ Code >

```
prepro_y = MinMaxScaler()
prepro_y.fit(mat_y)

prepro = MinMaxScaler()
prepro.fit(mat_train)

prepro_test = MinMaxScaler()
prepro_test.fit(mat_new)

#將array轉換成dataframe
train = pd.DataFrame(prepro.transform(mat_train),columns = col_train)
test  = pd.DataFrame(prepro_test.transform(mat_test),columns = col_train_bis)
```

■ Output >

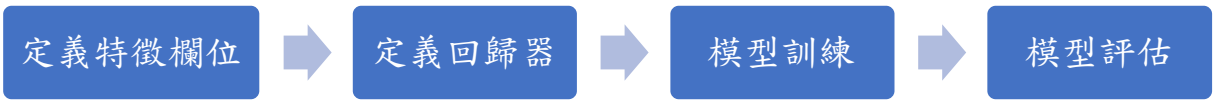
	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd
0	0.235294	0.033420	0.666667	0.500	0.949275	0.883333
1	0.000000	0.038795	0.555556	0.875	0.753623	0.433333
2	0.235294	0.046507	0.666667	0.500	0.934783	0.866667
3	0.294118	0.038561	0.666667	0.500	0.311594	0.333333
4	0.235294	0.060576	0.777778	0.500	0.927536	0.833333

3. 模型建立與方法介紹

3.1 模型建立與應用方法論

一般而言，類神經網路多用於處理分類問題，但是同樣可以用於回歸問題和非監督式學習的問題。此研究使用類神經網路回歸器進行建模，使用的是 Tensorflow 套件中的 DNNRegressor 模組。

DNNRegressor 模型建構步驟如下：



1. 定義特徵欄位

只選用數值型特徵值，並取得其特徵欄位。

```
feature_cols = [tf.contrib.layers.real_valued_column(k) for k in FEATURES]
```

2. 定義回歸器

```
regressor = tf.contrib.learn.DNNRegressor(feature_columns=feature_cols,
                                          activation_fn = tf.nn.relu, hidden_units=[200, 100, 50, 25, 12])
```

- 5 hidden layers with repsectly200, 100, 50, 25 and 12 units
- activation function: Relu
- optimizer: Adagrad (default)

3. 模型訓練

(1) 定義 input function (定義傳給模型訓練的資料格式)

```
def input_fn(data_set, pred = False):
    if pred == False:
        feature_cols = {k: tf.constant(data_set[k].values) for k in FEATURES}
        labels = tf.constant(data_set[LABEL].values)
        return feature_cols, labels

    if pred == True:
        feature_cols = {k: tf.constant(data_set[k].values) for k in FEATURES}
        return feature_cols
```

feature_cols：dictionary，key 是特徵的名字，value 是對應的資料

labels: 要預測的變數 (SalePrice)

(2) 模型訓練

將 input_fn 函數轉換為數值，迭代次數設定為 2000 次

```
regressor.fit(input_fn=lambda: input_fn(training_set), steps=2000)
```

4. 模型評估

```
ev = regressor.evaluate(input_fn=lambda: input_fn(testing_set), steps=1)
```

```
Final Loss on the testing set: 0.001301
```

3.2 模型調適

以下將透過改變部分模型參數或調整模型架構來提升模型效度。

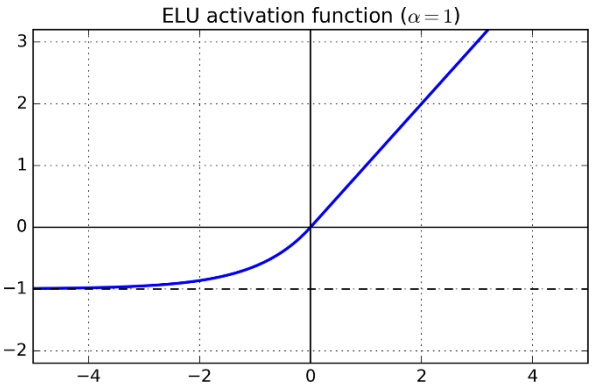
(1) Activation function: Leaky Relu

```
regressor = tf.contrib.learn.DNNRegressor(feature_columns=feature_cols,
                                          activation_fn = tf.nn.leaky_relu, hidden_units=[200, 100, 50, 25, 12])
```

Loss: 0.001061

(2) Activation function: Elu (Exponential Linear Unit)

$$R(z) = \begin{cases} z & z > 0 \\ \alpha.(e^z - 1) & z <= 0 \end{cases}$$




```
regressor = tf.contrib.learn.DNNRegressor(feature_columns=feature_cols,
                                          activation_fn = tf.nn.elu, hidden_units=[200, 100, 50, 25, 12])
```

Loss: 0.001328

(3) 加入類別型特徵，activation function: Leaky Relu

I. 取得離散型特徵值的欄位

```
for categorical_feature in FEATURES_CAT:
    sparse_column = tf.contrib.layers.sparse_column_with_hash_bucket(categorical_feature, hash_bucket_size=1000)
```

II. 定義新的 input function

```
def input_fn_new(data_set, training = True):
    continuous_cols = {k: tf.constant(data_set[k].values) for k in FEATURES}
    categorical_cols = {k: tf.SparseTensor(
        indices=[[i, 0] for i in range(data_set[k].size)], values = data_set[k].values, dense_shape = [data_set[k].size, 1]) for k in FEATURES_CAT}

    # Merges the two dictionaries into one.
    feature_cols = dict(list(continuous_cols.items()) + list(categorical_cols.items()))
    if training == True:
        label = tf.constant(data_set[LABEL].values)
        return feature_cols, label

    return feature_cols
```

Loss: 0.001253

(4) Shallow Network (淺層神經網路)

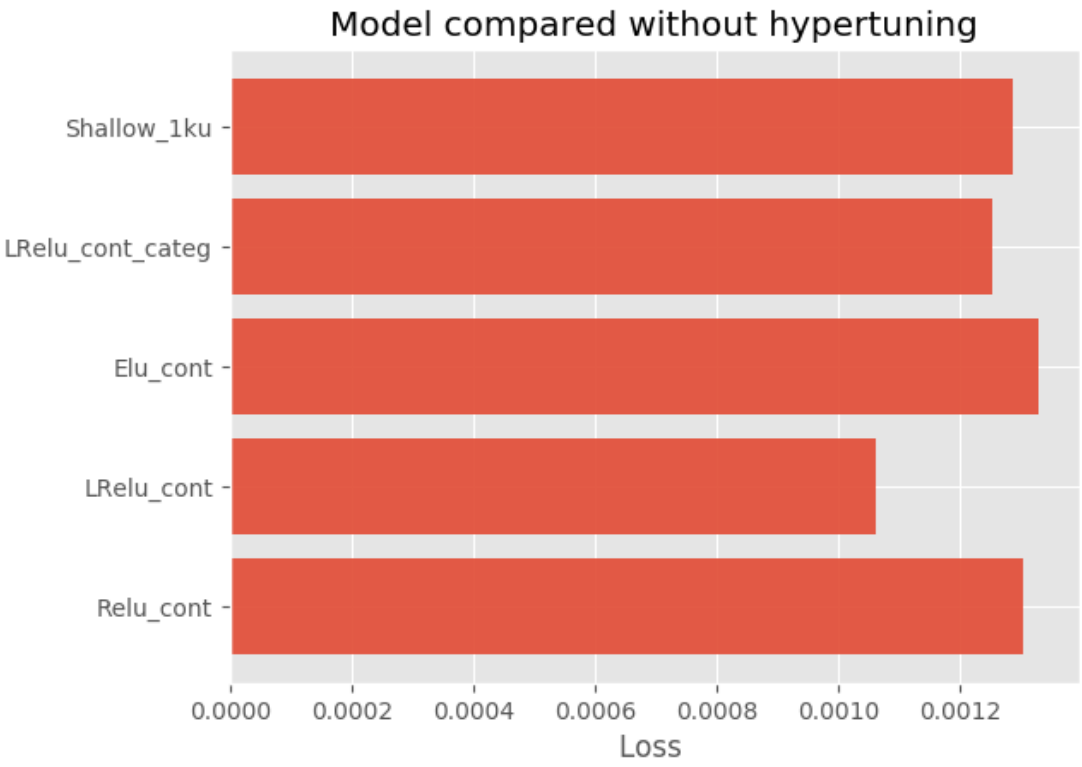
深層神經網路理論上模型表現會更強，但相對的運算量也會比較大、容易有 overfitting 的問題。因此嘗試淺層神經網路的方法，透過設定一層足夠多神經元的隱藏層來測試模型表現。

→ one hidden layer with 1000 units

```
regressor = tf.contrib.learn.DNNRegressor(feature_columns = engineered_features,
                                          activation_fn = tf.nn.relu, hidden_units=[1000])
```

Loss: 0.001285

三、 小結與未來展望



綜合以上模型調整結果，可得知僅保留數值型特徵，並使用 Leaky Relu 為激活函數時，模型表現最佳。加入類別型特徵後，模型表現反而較差，可能的原因是大部分類別型特徵的資料相似，(均屬於同一類別)，無法進行有效的區別。

本研究使用深度學習的類神經網路回歸器進行建模，其模型預測結果具良好的效度。未來可以在資料前處理上可加入特徵萃取的方法，以取得更有效的判斷特徵。此外在模型建構上可再做更多參數的調適，使得模型效度更加提升。

四、 參考資料

1. 【TensorFlow】DNNRegressor 的简单使用：
   https://blog.csdn.net/u010099080/article/details/72824899
2. Activation Functions: ELU
   https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#relu