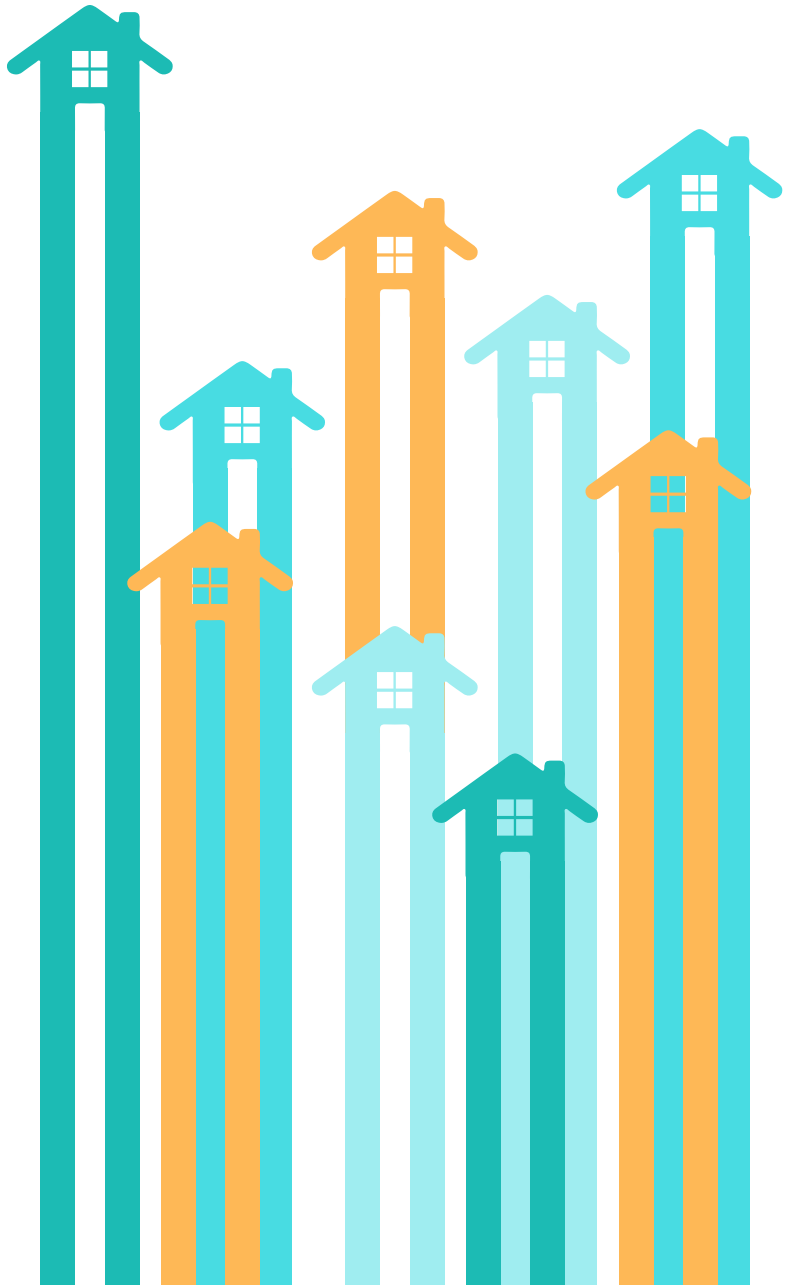
2019 IIE Project 3

房屋售價預測模型之建構與驗證

108034540 吳欣蓉



Agenda

01

研究主題

5W1H

02

資料分析與前處理

資料來源與結構分析/資料前處理

03

模型建立與方法介紹

模型架構、應用方法與模型調適及訓練

04

小結與未來展望

研究主題

房價預測 | What

美國艾姆斯州的房屋資料 | Where

欲購買房屋的民眾、房仲業者、投資客 | Who

欲出售或購買房子時 | When

房價是影響房屋購買意願的最關鍵因素 | Why

透過深度學習分析房屋的特徵資料 | How

資料來源與結構分析

資料來源：Ames Housing dataset公開資料集

- 美國艾姆斯州房屋的特徵資料及其售價
- 共1460筆資料

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

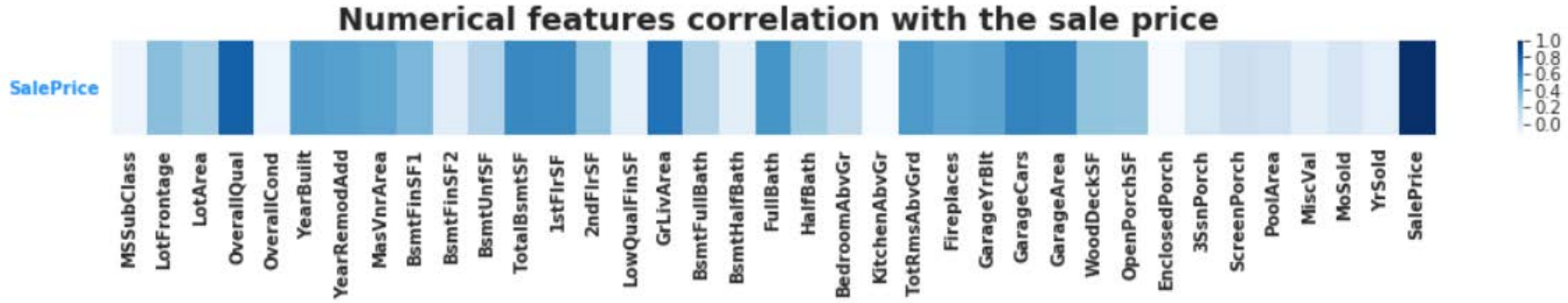
REAL ESTATE

資料來源與結構分析

- 資料中共有81個欄位，包含數值型及類別型的資料

  Ex: 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

- 分析數值型欄位與房價之間的關聯性



REAL ESTATE

" OverallQual" ： 房屋的材質與完工品質

" GrLivArea" ： 房屋的建物面積

資料分析與前處理

1. 資料清洗

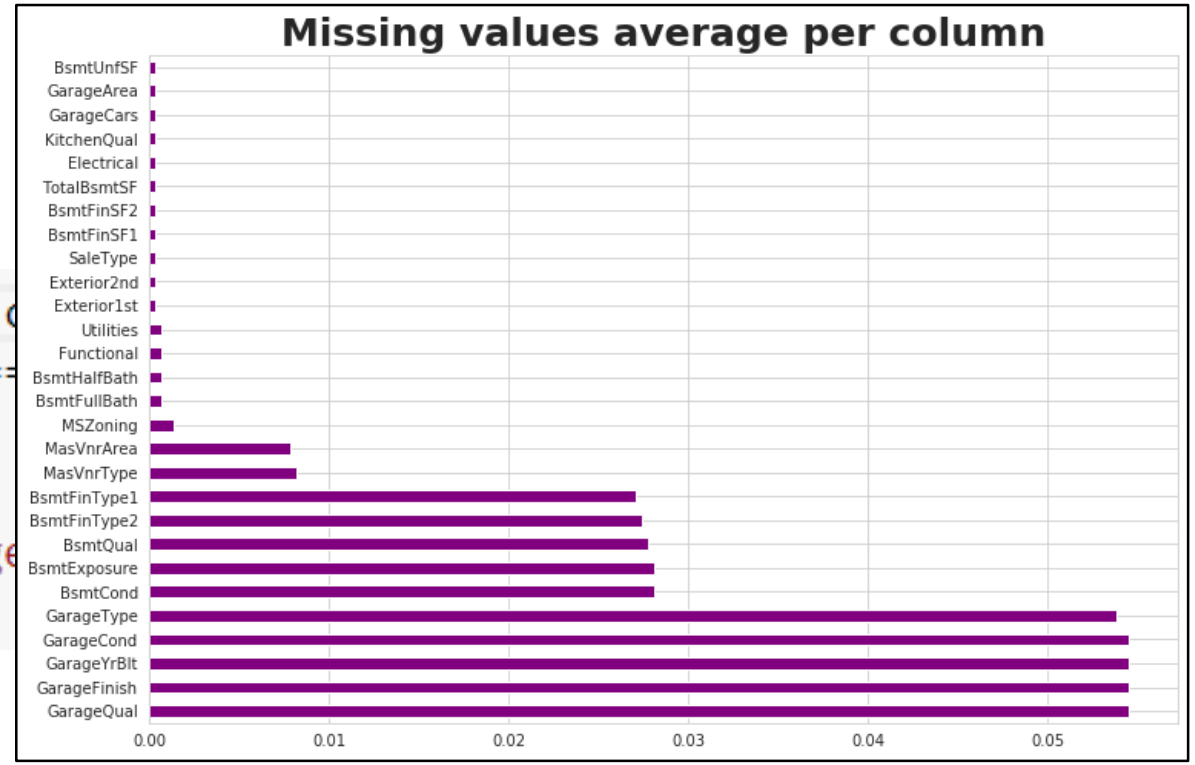
1. 刪除缺失值大於90%的特徵

```
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
```

2. 統計剩餘欄位的缺失值比例

```
allna = (c.isnull().sum() / len(
allna = allna.drop(allna[allna ==
plt.figure(figsize=(12, 8))
allna.plot.barh(color='purple')
plt.title('Missing values average
plt.show()
```



1. 資料清洗

3. 補齊剩餘欄位的缺失值

- 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：補0

- 類別型欄位

針對只有1~2筆缺失值的欄位，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')
```

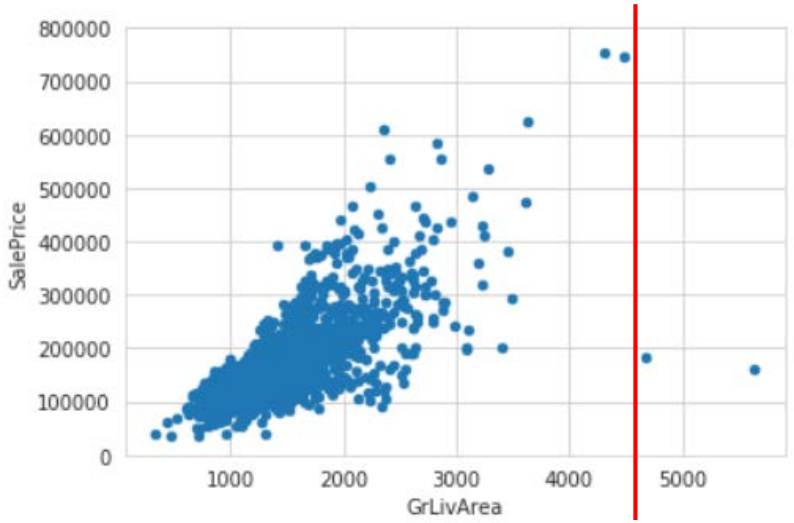
其餘欄位：改為”None”

Two

刪除離群值

資料集的敘述中，作者提及可分析”GrLivArea”欄位並刪除其離群值

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



建物面積大，售價卻很低？ 刪除右下角的兩筆資料。

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

3. 資料篩選

- 首次建模只選用數值型特徵，篩選出數值型欄位

```
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',
```

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

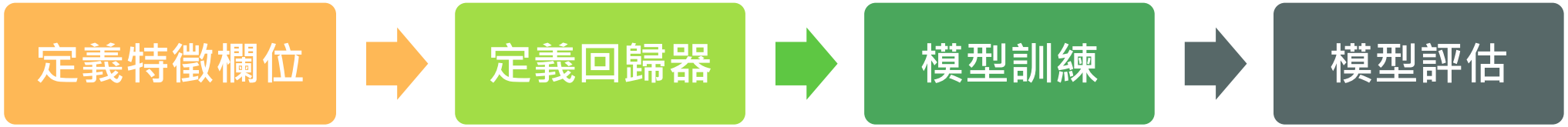
```
prepro_y = MinMaxScaler()
prepro_y.fit(mat_y)

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)
```

	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

模型建立與方法介紹

模型建立與應用方法論

- 應用方法：類神經網路回歸器
  使用Tensorflow套件中的DNNRegressor模組
- DNNRegressor模型建構步驟：



1. 定義特徵欄位：取得數值型特徵欄位

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

2. 分割訓練與測試集

```
x_train, x_test, y_train, y_test = train_test_split(training_set[FEATURES] , prediction_set, test_size=0.33, random_state=42)
```

模型建立

3. 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])
```

- activation function: Relu
- 5 hidden layers with repsectly 200, 100, 50, 25 and 12 units
- optimizer: Adagrad (default)

4. 模型訓練

- 定義input function (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：特徵的名字
value：其對應的資料

模型建立

5. 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)
```

6. 模型評估

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

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

模型調適

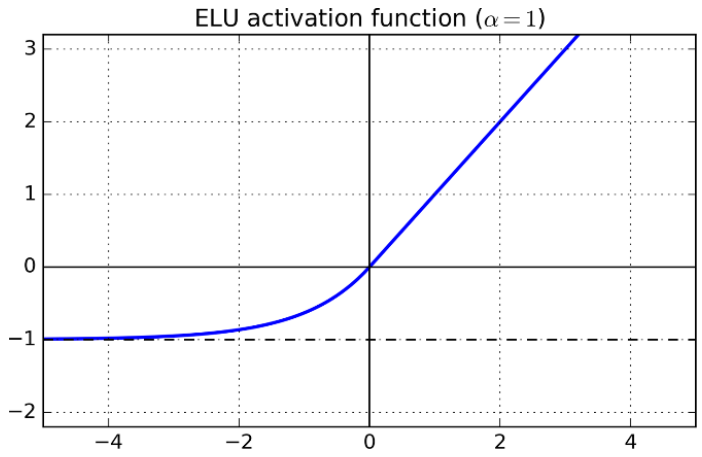
1) Activation function: Leaky Relu **Loss: 0.001061**

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

2) Activation function: Elu (Exponential Linear Unit) **Loss: 0.001328**

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

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



模型調適

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

- 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)
```

- 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

模型調適

4) Shallow Network (淺層神經網路)

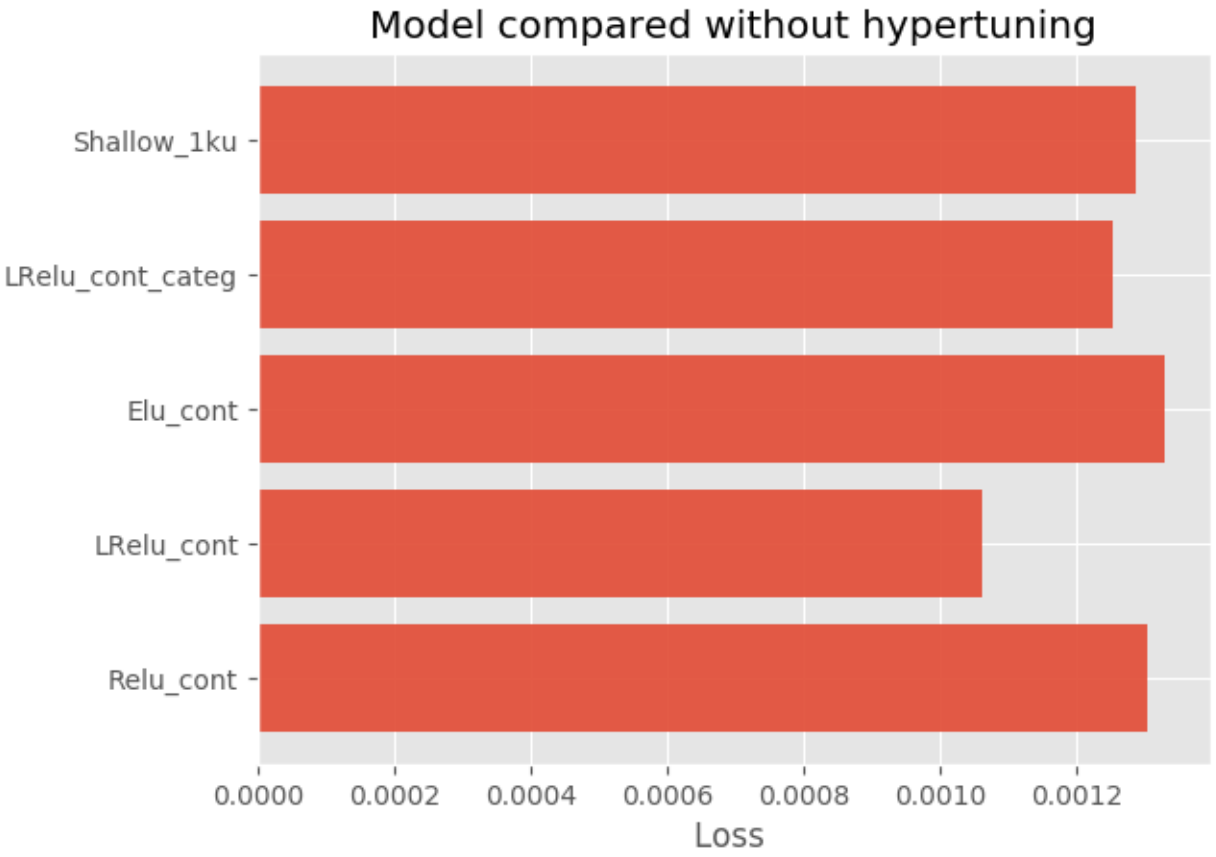
深層神經網路理論上模型表現會更強，但相對的運算量也會比較大、容易有overfitting的問題。
因此嘗試淺層神經網路的方法，透過設定一層足夠多神經元的隱藏層來測試模型表現。

→ one hidden layer with 1000 units

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

Loss: 0.001253

模型調適結果比較



- 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。

未來展望

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


Thank you