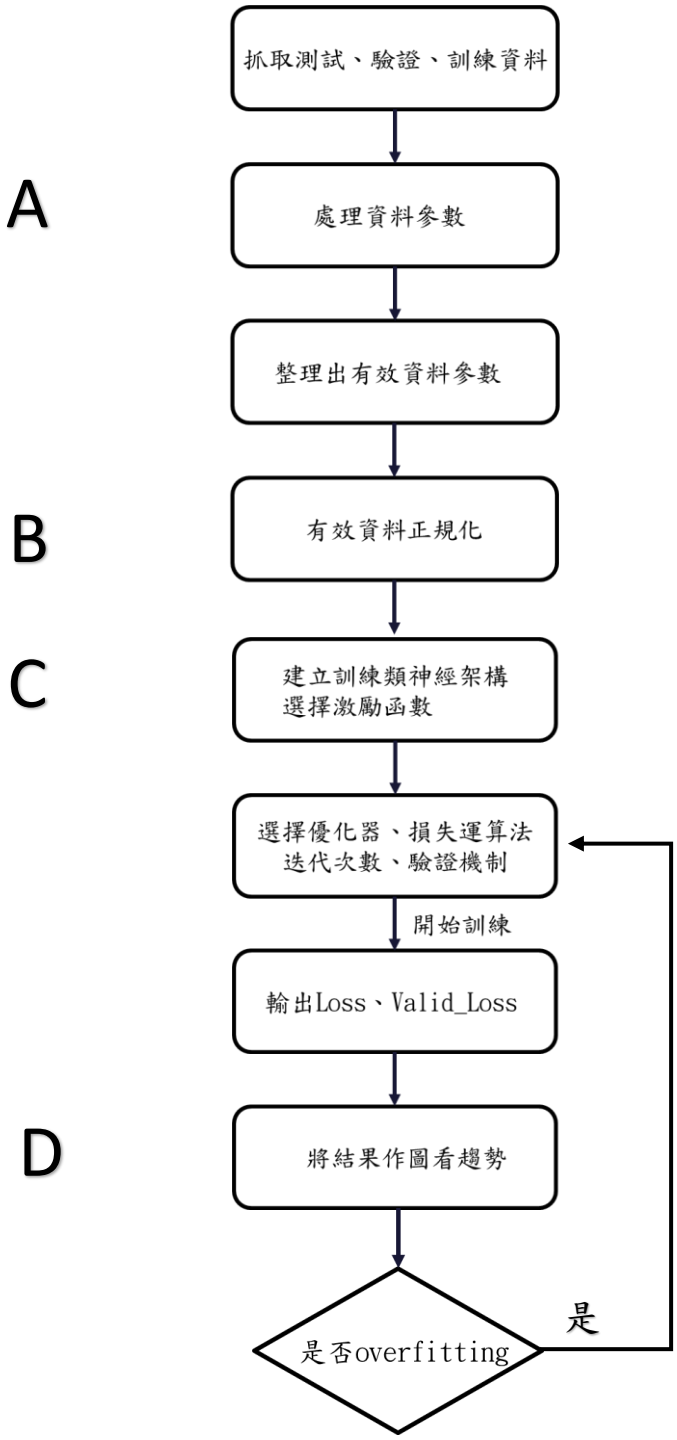
Machine Learning@NTUT – Regression

指導教授:廖元甫
學生:陳藝鵬

1. 實踐說明:先作出流程圖再撰寫程式，取得結果後進行優化找出更小誤差。

2. 程式方塊圖與寫法



```python
def getData(path):
    df = pd.read_csv(path)
    Y = np.array([])
    zip = pd.get_dummies(df['zipcode'])
    df = df.join(zip)
    #df = df.drop(columns=['id', 'zipcode', 'sale_yr', 'sale_month', 'sale_day'])
    df.drop(columns=['id', 'zipcode', 'sale_yr', 'sale_month', 'sale_day'])
    dataset = np.array(df.values)
    if "price" in df.columns:
        Y = dataset[:, 1]
        dataset = np.delete(dataset, 1, 1)
        #dataset = np.delete(dataset, 0, 1)
    dataset = np.delete(dataset, 0, 1)
    return dataset, Y

# Read training dataset into X and Y
X_train, Y_train = getData('./train-v3.csv')

np.savetxt('./X_train.csv', X_train, delimiter=',', fmt='%i')
# Read validation dataset into X and Y
X_valid, Y_valid = getData('./valid-v3.csv')

# Read test dataset into X
X_test, _ = getData('./test-v3.csv')
```

A:抓取 train、valid、test.csv 檔
並且將郵遞區號進行編碼
重新丟回訓練用資料參數，
目的是為了增加模型可靠度
最後將 price、id 從.csv 移除
得到只有訓練需要的參數。

```python
def normalize(train,valid,test):
    tmp=train
    mean=tmp.mean(axis=0)
    std=tmp.std(axis=0)
    # print("tmp.shape=",tmp.shape)
    # print("mean.shape=",mean.shape)
    # print("std.shape=",std.shape)
    # print("mean=",mean)
    # print("std=",std)
    train=(train-mean)/std
    valid=(valid-mean)/std
    test=(test-mean)/std
    return train,valid,test

X_train,X_valid,X_test=normalize(X_train,X_valid,X_test)
```

B:正規化，以 train 為中心進行
處理，先取平均、標準差。
接著再用各筆資料減去 train
的平均並且除以 train 的標準
差，得到的結果就會是同樣
的基準。

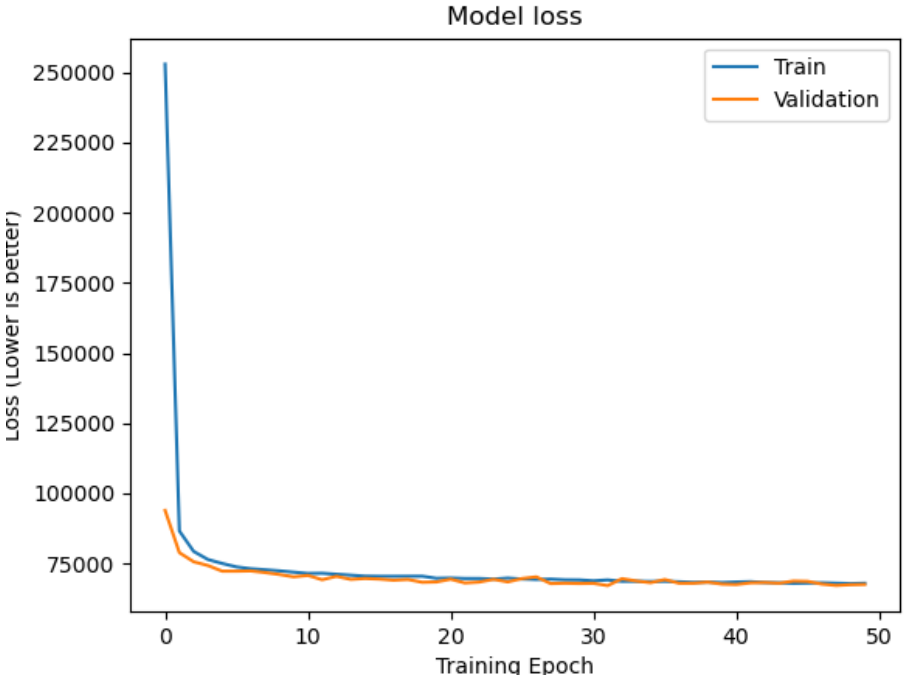
```python
from tensorflow import keras
import tensorflow as tf

model = keras.Sequential([
    keras.layers.Dense(40, input_dim=X_train.shape[1]),
    keras.layers.Dense(60, activation='relu'),
    keras.layers.Dense(100, activation='relu'),
    keras.layers.Dense(150, activation='relu'),
    keras.layers.Dense(50, activation='relu'),
    keras.layers.Dense(1)
])


model.compile(optimizer='adam',
              loss='mae')

history = model.fit(X_train, Y_train, batch_size=30, epochs=50, validation_data=(X_valid, Y_valid))
```

C:類神經訓練，首先建立隱藏層
架構，輸入維度以 train 來
看，激勵函數使用’relu’
優化器使用’adam’
損失函數運算使用‘Mae’
隨機抓取:30、迭代次數:50
最後用 valid 輔佐 train 訓練。

```python
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss (Lower is better)')
plt.xlabel('Training Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.savefig('loss.png')
plt.show()
```

D:作圖進行輔助判 overfitting 或
者 underfitting，如果有問
題，再從訓練資料或隱藏層調
整，如果沒問題，則繼續優化
使 Loss、Valid Loss 更小。

3. 畫圖做結果分析




```
Epoch 48/50
433/433 [==============================] - 0s 819us/step - loss: 67984.7891 - val_loss: 67197.6328
Epoch 49/50
433/433 [==============================] - 0s 813us/step - loss: 67846.0078 - val_loss: 67461.7500
Epoch 50/50
433/433 [==============================] - 0s 799us/step - loss: 67967.4453 - val_loss: 67552.8047
val_loss: 67234.8125
loss: 67552.8046875
```

4. 討論預測值差很大:由於此圖並沒有 overfitting 的趨勢，因此先撇除這個原因再來可能是類神經訓練的神經元不夠多或者隱藏層數量不夠，導致模型學得不夠深或者學錯方向，但是經過多次調整類神經學習測試，兩個誤差的結果都會差不多，不然就是 Overfitting。因此，問題最大的地方仍然在於訓練資料，期間有嘗試未將郵遞區號進行編碼去訓練，跟有編碼過的比起，就相差了將近 40000，所以可能是還有其他有利因素並沒有被找到或經過處理，才會使結果仍然差了 70000，又或者將某幾筆錯誤的訓練參數當成有利因素，間接導致模型往沒意義的方向訓練，形成反效果，仍需多加嘗試並找出原因。

5. 如何改進:嘗試以下方法 1-隨機抓取或迭代數量增加確定沒有 underfitting
   2-找出對於模型更有效的訓練參數
   3-增加隱藏層數或者神經元