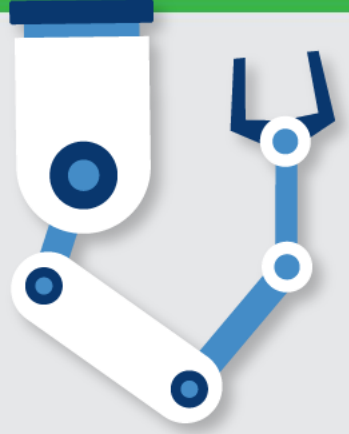
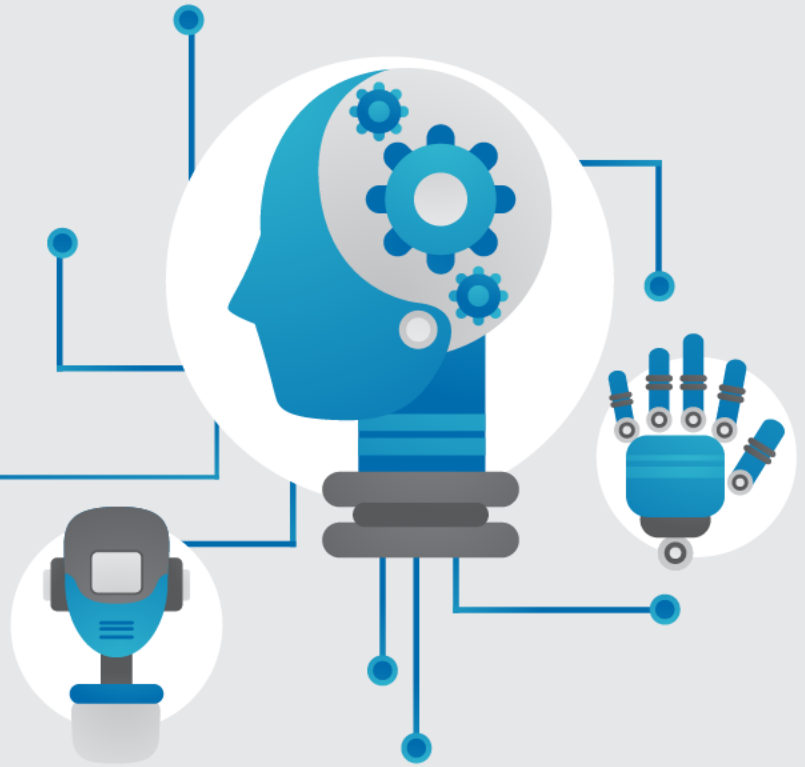


波斯頓房價預測實作



Boston Housing資料集介紹

- 為U.S Census Service所蒐集，由Harrison, D.和Rubinfeld, D.L.建立，內容為波斯頓區域各城鎮的資料，能在出處StatLib (http://lib.stat.cmu.edu/datasets/)找到。
- 記錄14種城鎮資料，總共506筆
- 能應用於一氧化碳濃度或房價的預測，通常使用於**房價預測**

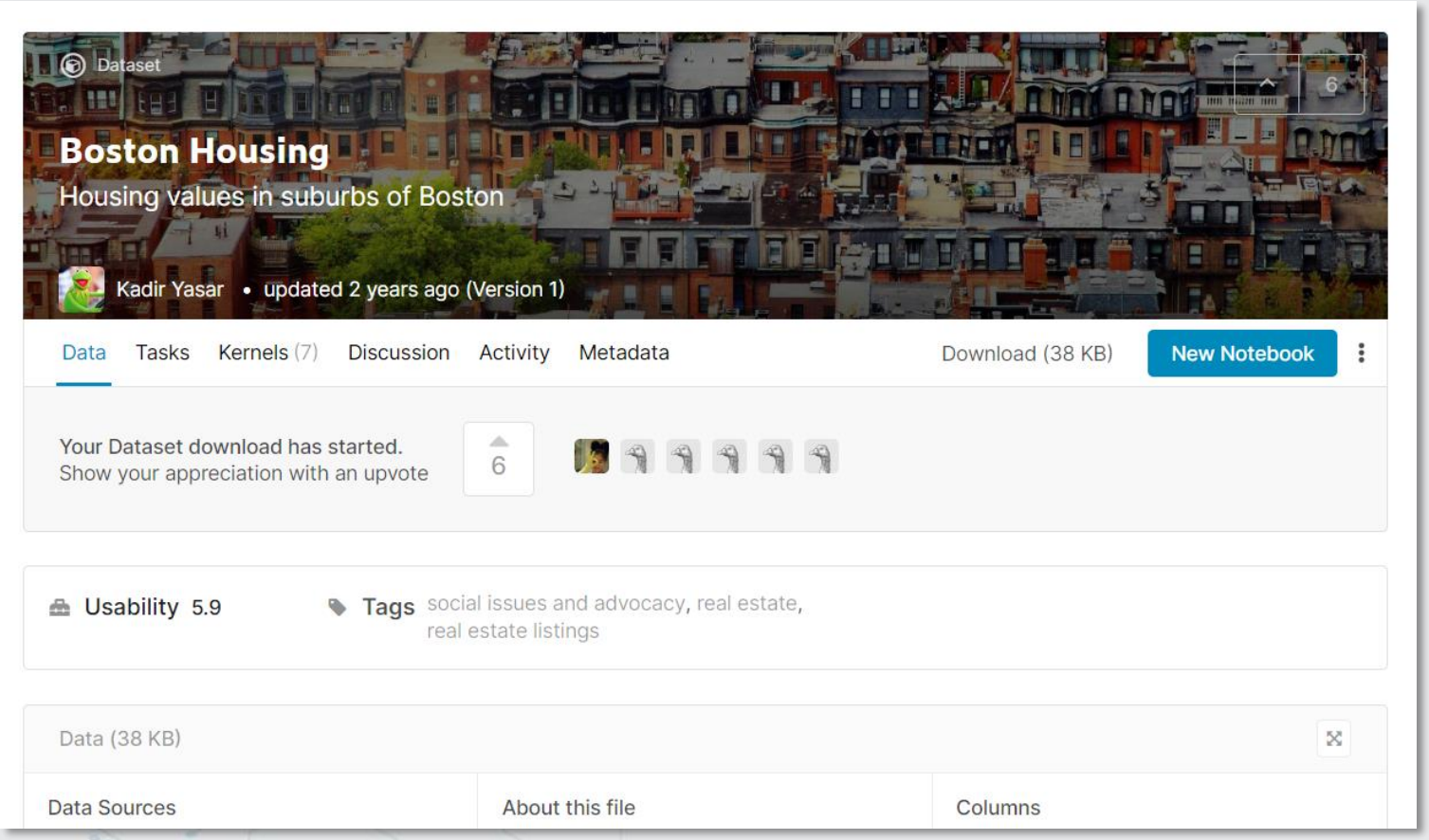
Boston Housing資料集介紹

14種城鎮資料

- 城鎮人均犯罪率
- 佔地25,000平方英尺以上的住宅用地比例
- 城鎮非零售業比例（每英畝）
- 是否在查爾斯河邊
- 一氧化碳濃度
- 每個住宅的平均房間數
- 1940年之前建造的自有住房的比例
- 與五個波士頓就業中心的加權距離
- 高速公路通行性指數
- 財產稅率（每10,000美元）
- 城鎮師生比例
- 1000（黑人比例−0.63）^{2}
- 人口地位較低的百分比
- **自有住房價值的中位數(單位1000美元)**

Kaggle資料集 - Boston Housing

https://www.kaggle.com/kyasar/boston-housing



Kaggle資料集 - Boston Housing

機器學習實務

資料檔案　boston_housing.csv

- 共506筆資料（row）
- 每個row有14 columns，第一row為描述縮寫
- 欄位描述順序對應於前面的說明

1	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
2	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
3	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
4	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
5	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
6	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
7	0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
8	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
9	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
10	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
11	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
12	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
13	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
14	0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7
15	0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4
16	0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2
17	0.62739	0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21	395.62	8.47	19.9
18	1.05393	0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21	386.85	6.58	23.1
19	0.7842	0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21	386.75	14.67	17.5
20	0.80271	0	8.14	0	0.538	5.456	36.6	3.7965	4	307	21	288.99	11.69	20.2
21	0.7258	0	8.14	0	0.538	5.727	69.5	3.7965	4	307	21	390.95	11.28	18.2
22	1.25179	0	8.14	0	0.538	5.57	98.1	3.7979	4	307	21	376.57	21.02	13.6

MLP模型建置流程

1.資料前處理

2.決定模型架構

3.編譯與訓練模型

4.模型評估

資料前處理

- 從檔案boston_housing.csv讀進資料，並去掉不需要的欄位
  - ✓佔地25,000平方英尺以上的住宅用地比例 (zn)
  - ✓一氧化碳濃度 (nox)
  - ✓每個住宅的平均房間數 (rm)
  - ✓1940年之前建造的自有住房的比例 (age)
  - ✓城鎮師生比例 (ptratio)
  - ✓1000（黑人比例−0.63）(black)

```
1 import pandas as pd
2 data=pd.read_csv('boston_housing.csv')
3
4 unuse=['zn', 'nox', 'rm', 'age', 'ptratio', 'black']
5 data=data.drop(unuse, axis=1)
```

資料前處理

› 訓練資料和label轉成np.array資料型態

```
7 y=data.pop('medv').values.astype('float32')
8 x=data.values.astype('float32')
```

資料前處理

- 切出訓練和測試資料

```
10 from sklearn.model_selection import train_test_split
11 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

資料前處理

› 正規化

```
13 # 用train的max正規化
14 import numpy as np
15 x_train_max=np.max(x_train, axis=0)
16 x_test_max=np.max(x_test, axis=0)
17
18 x_train=x_train/x_train_max
19 x_test=x_test/x_test_max
```

決定模型架構

模型建置

```
21 from keras.layers import Dense, Dropout
22 from keras.models import Sequential
23
24 model=Sequential()
25 model.add(Dense(1024, activation='relu',input_shape=(7,)))
26 model.add(Dropout(0.3))
27 model.add(Dense(1024, activation='relu'))
28 model.add(Dropout(0.3))
29 model.add(Dense(1024, activation='relu'))
30 model.add(Dropout(0.3))
31 model.add(Dense(1024, activation='relu'))
32 model.add(Dropout(0.3))
33 model.add(Dense(64, activation='relu'))
34 model.add(Dropout(0.1))
35 model.add(Dense(1, activation='linear'))
```

決定模型架構

- 自訂loss：均方根誤差

```
37 # y_true, y_pred為tensor物件
38 import keras.backend as K
39 def rmse(y_true, y_pred):
40         return K.sqrt(K.mean(K.square(y_pred - y_true)))
```

編譯與訓練模型

› 設定參數，訓練模型

```
42 model.compile(
43         loss=rmse,
44         optimizer='rmsprop')
45 history=model.fit(
46         x_train,y_train,
47         validation_data=(x_test,y_test),
48         epochs=60,
49         shuffle=True)
```

模型評估

› 顯示訓練歷程

```
51 import matplotlib.pyplot as plt
52 def show_train_history(train_history):
53     plt.figure(figsize=(10,5))
54     plt.plot(train_history.history['loss'])
55     plt.plot(train_history.history['val_loss'])
56     plt.xticks([i for i in range(len(train_history.history['loss']))])
57     plt.title('Train History')
58     plt.ylabel('rmse')
59     plt.xlabel('epoch')
60     plt.legend(['train', 'validation'], loc='upper left')
61     plt.show()
62 show_train_history(history)
```

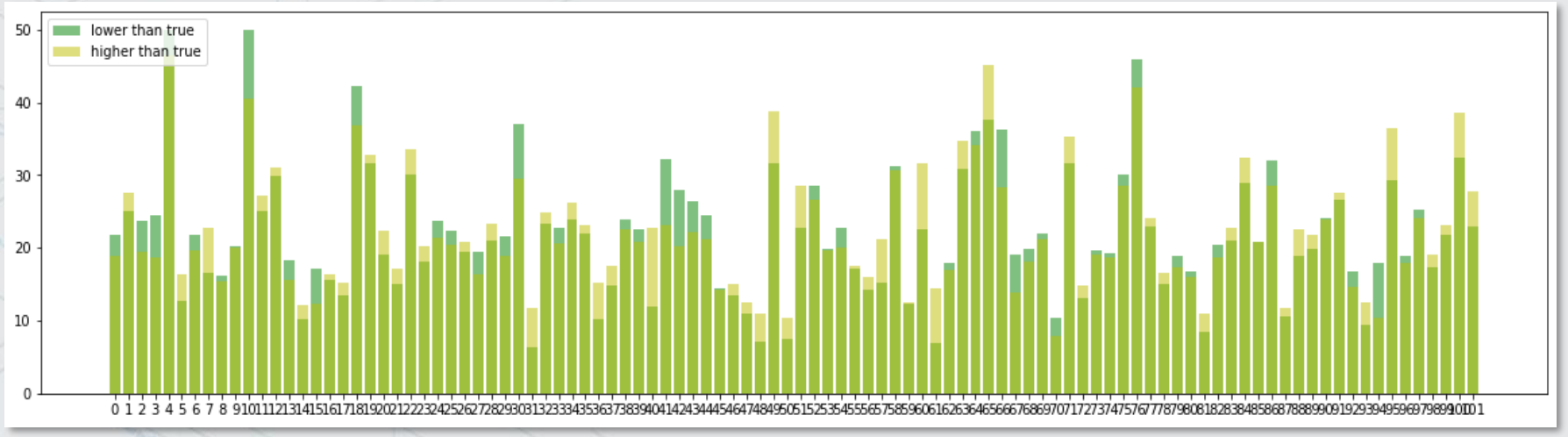
模型評估

- 顯示訓練歷程

模型評估

- 顯示實際值和預測值的差異

```
64 y_true=y_test
65 y_pred=model.predict(x_test).ravel()
66
67 plt.figure(figsize=(20,5))
68 plt.bar([i for i in range(len(y_true))], y_true, alpha=0.5, color='g')
69 plt.bar([i for i in range(len(y_pred))], y_pred, alpha=0.5, color='y')
70 plt.xticks([i for i in range(len(y_true))])
71 plt.legend(['lower than true', 'higher than true'], loc='upper left')
72 plt.show()
```



模型評估

顯示預測差異

```
74 from sklearn.metrics import mean_squared_error
75 print('rmse: ', np.sqrt(mean_squared_error(y_true,y_pred)))
```

```
In [8]: from sklearn.metrics import mean_squared_error
   ...: print('rmse: ', np.sqrt(mean_squared_error(y_true,y_pred)))
rmse:  3.9048777
```