神经网络&深度学习
TENSORFLOW
Google

6.4 波士顿房价数据集可视化

中国大学MOOC

Keras

- 是一个高层的**神经网络和深度学习库**。
- 可以**快速搭建神经网络模型**，非常**易于调试和扩展**。
- TensorFlow的**官方API**
- 内置了一些**常用的公共数据集**，可以通过keras.datasets模块加载和访问。

Keras中集成的数据集

序号	名 称	说 明
1	boston_housing	波士顿房价数据集
2	CIFAR10	10种类别的图片集
3	CIFAR100	100种类别的图片集
4	MNIST	手写数字图片集
5	Fashion-MNIST	10种时尚类别的图片集
6	IMDB	电影点评数据集
7	reuters	路透社新闻数据集

波士顿房价数据集

- 卡内基梅隆大学，StatLib库，1978年
- 涵盖了麻省波士顿的506个不同郊区的房屋数据
- 404条训练数据集，102条测试数据集
- 每条数据14个字段，包含13个属性，和1个房价的平均值

序号	变量名	说 明	示 例
1	CRIM	城镇人均犯罪率	0.00632
2	ZN	超过25000平方英尺的住宅用地所占比例	18.0
3	INDUS	城镇非零售业的商业用地所占比例	2.31
4	CHAS	是否被Charles河流穿过（取值1：是；取值0：否）	0
5	NOX	一氧化碳浓度	0.538
6	RM	每栋住宅的平均房间数	6.575
7	AGE	早于1940年建成的自住房屋比例	65.2
8	DIS	到波士顿5个中心区域的加权平均距离	4.0900
9	RAD	到达高速公路的便利指数	1
10	TAX	每10000美元的全值财产税率	296
11	PTRATIO	城镇中师生比例	15.3
12	B	反映城镇中的黑人比例的指标，越靠近0.63越小；$B=1000*(BK-0.63)^2$，其中BK是黑人的比例。	396.90
13	LSTAT	低收入人口的比例	7.68
14	MEDV	自住房屋房价的平均房价（单位为1000美元）	24.0

加载数据集——.load_data()方法

前缀：tensorflow.keras.datasets

数据集名称：boston_housing

```
import tensorflow as tf
boston_housing = tf.keras.datasets.boston_housing

(train_x, train_y), (test_x, test_y) = boston_housing.load_data()
```

提示信息：

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/boston_housing.npz
57344/57026 [==============================] - 1s 23us/step
```

本地默认路径：

C:\Users\user_name\.keras\datasets
C:\Users\Administrator\.keras\datasets\boston_housing.npz

训练集合测试集

```
import tensorflow as tf
boston_housing = tf.keras.datasets.boston_housing

(train_x, train_y), (test_x, test_y) = boston_housing.load_data()
```

训练数据集　测试数据集　test_split=0.2

```
print("Training set:",len(train_x))
print("Testing set:",len(test_x))
```

运行结果：

```
Training set: 404
Testing set: 102
```

改变数据集划分比例

提取出全部数据作为训练集

```
(train_x, train_y), (test_x, test_y) = boston_housing.load_data(test_split=0)
```

```
print("Training set:",len(train_x))
print("Testing set:",len(test_x))
```

运行结果：

```
Training set: 506
Testing set: 0
```

访问数据集中的数据

```
>>>type(train_x)
numpy.ndarray
>>>type(train_y)
numpy.ndarray

>>>print("Dim of train_x:",train_x.ndim)
Dim of train_x: 2
>>>print("Shape of train_x:",train_x.shape)
Shape of train_x: (506, 13)

>>>print("Dim of train_y:",train_y.ndim)
Dim of train_y: 1
>>>print("Shape of train_y:",train_y.shape)
Shape of train_y: (506,)
```

- 访问数据集中的数据——输出train_x中的**前5行**数据

```
>>>print(train_x[0:5])
[[1.23247e+00  0.00000e+00  8.14000e+00  0.00000e+00  5.38000e-01  6.14200e+00
  9.17000e+01  3.97690e+00  4.00000e+00  3.07000e+02  2.10000e+01  3.96900e+02
  1.87200e+01]
 [2.17700e-02  8.25000e+01  2.03000e+00  0.00000e+00  4.15000e-01  7.61000e+00
  1.57000e+01  6.27000e+00  2.00000e+00  3.48000e+02  1.47000e+01  3.95380e+02
  3.11000e+00]
 [4.89822e+00  0.00000e+00  1.81000e+01  0.00000e+00  6.31000e-01  4.97000e+00
  1.00000e+02  1.33250e+00  2.40000e+01  6.66000e+02  2.02000e+01  3.75520e+02
  3.26000e+00]
 [3.96100e-02  0.00000e+00  5.19000e+00  0.00000e+00  5.15000e-01  6.03700e+00
  3.45000e+01  5.98530e+00  5.00000e+00  2.24000e+02  2.02000e+01  3.96900e+02
  8.01000e+00]
 [3.69311e+00  0.00000e+00  1.81000e+01  0.00000e+00  7.13000e-01  6.37600e+00
  8.84000e+01  2.56710e+00  2.40000e+01  6.66000e+02  2.02000e+01  3.91430e+02
  1.46500e+01]]
```

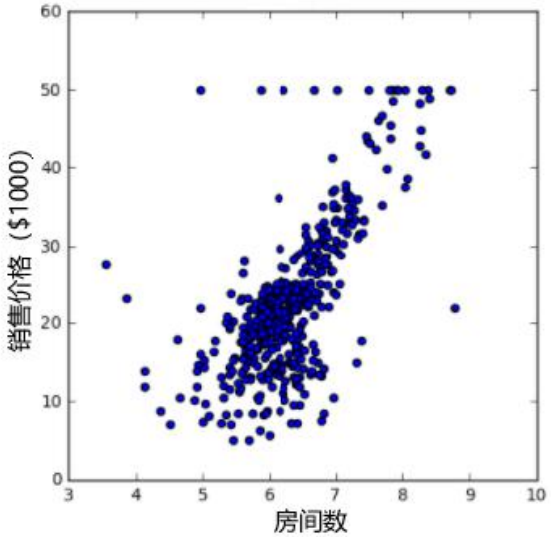
- 访问数据集中的数据——输出train_x中的**第6列**数据

```
>>>print(train_x[:, 5])
[6.142 7.61  4.97  6.037 6.376 5.708 5.536 5.468 5.628 5.019 6.404 4.628
 5.572 6.251 5.613 5.957 7.016 6.345 6.162 6.727 6.202 6.595 7.135 6.575
 5.895 6.794 6.012 7.185 5.813 5.569 6.315 6.297 6.301 5.935 7.024 6.415
 5.599 5.701 6.041 6.279 5.454 6.211 6.316 6.411 5.887 5.924 5.822 6.674
 6.842 5.713 5.968 6.461 7.358 6.565 5.88  5.87  6.348 6.193 6.854 6.546
 ......
 5.036 5.813 7.185 6.63  6.343 8.297 6.758 6.421 6.98  6.471 6.852 6.019
 6.376 6.108 6.417 6.209 5.093 5.987 6.395 6.957 6.229 5.414 6.495 6.009
 5.885 6.375 6.968 4.88  5.981 7.52  5.593 6.485 5.705 6.172 6.229 5.951
 6.593 7.061 6.03  5.884 6.897 8.259 6.812 6.122 7.333 8.78  6.273 7.802
 6.951 6.101]
```

- 访问数据集中的数据——输出train_y中的全部数据

```
>>>print(train_y)
[15.2 42.3 50.  21.1 17.7 18.5 11.3 15.6 15.6 14.4 12.1 17.9 23.1 19.9
 15.7  8.8 50.  22.5 24.1 27.5 10.9 30.8 32.9 24.  18.5 13.3 22.9 34.7
 16.6 17.5 22.3 16.1 14.9 23.1 34.9 25.  13.9 13.1 20.4 20.  15.2 24.7
 22.2 16.7 12.7 15.6 18.4 21.  30.1 15.1 18.7  9.6 31.5 24.8 19.1 22.
 14.5 11.  32.  29.4 20.3 24.4 14.6 19.5 14.1 14.3 15.6 10.5  6.3 19.3
 ......
 32.5 29.6 28.4 19.8 20.2 25.  35.4 20.3  9.7 14.5 34.9 26.6  7.2 50.
 32.4 21.6 29.8 13.1 27.5 21.2 23.1 21.9 13.  23.2  8.1  5.6 21.7 29.6
 19.6  7.  26.4 18.9 20.9 28.1 35.4 10.2 24.3 43.1 17.6 15.4 16.2 27.1
 21.4 21.5 22.4 25.  16.6 18.6 22.  42.8 35.1 21.5 36.  21.9 24.1 50.
 26.7 25. ]
```

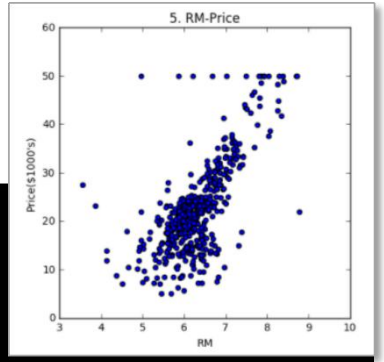
平均房间数与房价之间的关系



绘制散点图　　平均房间数　　房价

```
plt.scatter(train_x[:, 5],train_y)
```

例：将平均房间数与房价之间的关系可视化

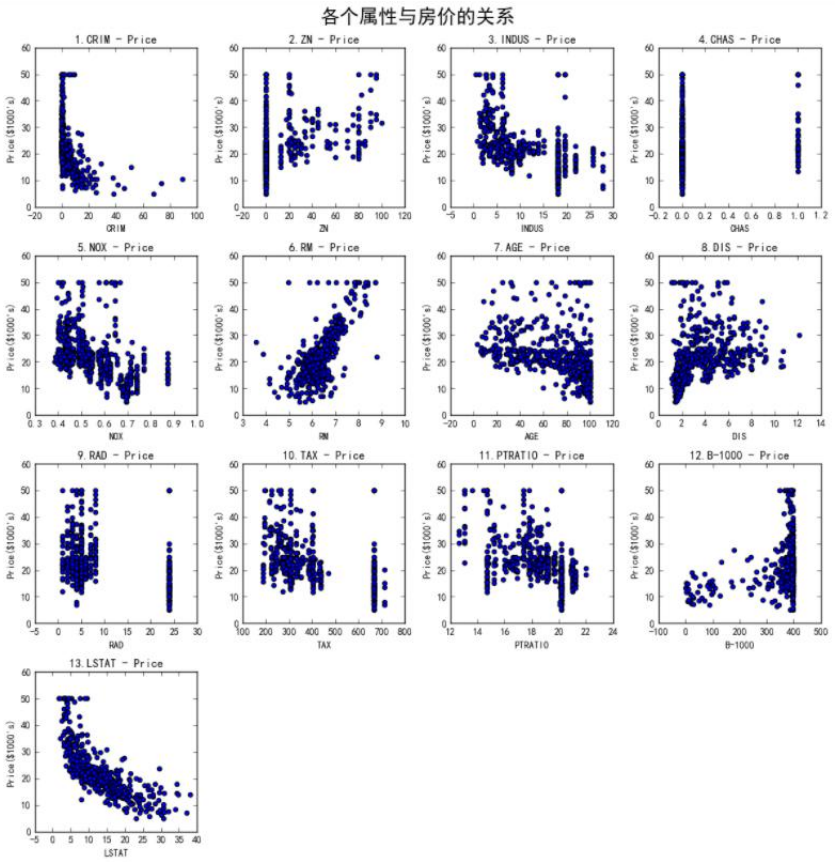



```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

boston_housing = tf.keras.datasets.boston_housing
(train_x, train_y), (_,_) = boston_housing.load_data(test_split=0)

plt.figure(figsize=(5,5))                    #设置绘图尺寸
plt.scatter(train_x[:, 5],train_y)           #绘制散点图
plt.xlabel("RM")                             #设置x轴标签文本
plt.ylabel("Price($1000's)")                 #设置y轴标签文本
plt.title("5. RM-Price")                     #设置标题
plt.show()                                   #显示绘图
```

例：将**所有属性**与房价之间的关系可视化



```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

boston_housing = tf.keras.datasets.boston_housing
(train_x, train_y), (_, _) = boston_housing.load_data(test_split=0)

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

titles = ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE",
          "DIS", "RAD", "TAX", "PTRATIO", "B-1000", "LSTAT", "MEDV"]

plt.figure(figsize=(12,12))

for i in range(13):
    plt.subplot(4,4,(i+1))
    plt.scatter(train_x[:,i], train_y)
    plt.xlabel(titles[i])
    plt.ylabel("Price($1000's)")
    plt.title(str(i+1)+ "." +titles[i]+" - Price")

plt.tight_layout()
plt.suptitle("各个属性与房价的关系", x=0.5, y=1.02, fontsize=20)
plt.show()
```