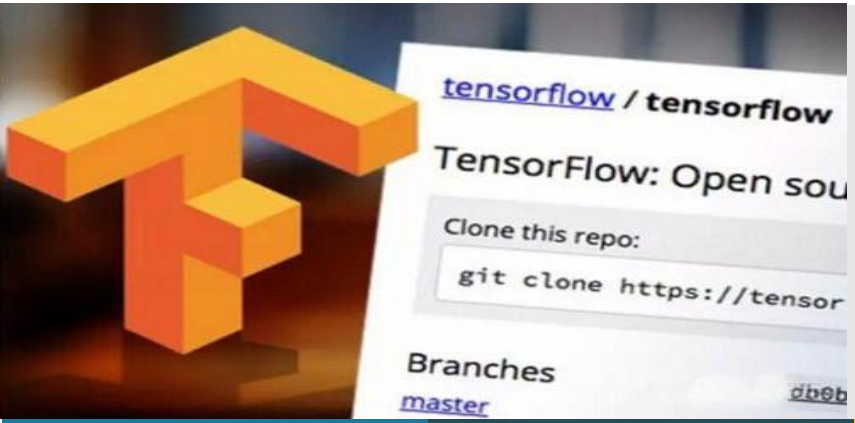
神经网络&深度学习
TENSORFLOW
Google



6.5.2 Pandas访问csv数据集

中国大学MOOC

- **Pandas库**（**Panel Data** & **Data Analysis**）
  - 用于**数据统计**和**分析**
  - 可以高效、方便地操作大型数据集

- 导入Pandas库

```
import pandas as pd
```

读取csv数据集文件

文件名参数——filepath_or_buffer

pd.read_csv(filepath_or_buffer, header, names)

直接使用绝对路径

```
pd.read_csv("C:/Users/Administrator/.keras/datasets/iris_training.csv")
```

```
TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
train_path = tf.keras.utils.get_file("iris_trainning.csv", TRAIN_URL)
pd.read_csv(train_path)
```

使用get_file()函数的返回值

运行结果：

	120	4	setosa	versicolor	virginica
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0
5	4.4	3.2	1.3	0.2	0
...	...	...	...	...	...
115	5.5	2.6	4.4	1.2	1
116	5.7	3.0	4.2	1.2	1
117	4.4	2.9	1.4	0.2	0
118	4.8	3.0	1.4	0.1	0
119	5.5	2.4	3.7	1.0	1

120 rows × 5 columns

二维数据表——DataFrame

是一种Pandas中常用的数据类型

```
TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
train_path = tf.keras.utils.get_file("iris_trainning.csv", TRAIN_URL)
df_iris=pd.read_csv(train_path)
```

```
>>>type(df_iris)
pandas.core.frame.DataFrame
```

设置列标题——header参数

```
pd.read_csv( filepath_or_buffer, header, names)
```

header 的取值是行号，行号**从0开始**

header=0，第1行数据做为列标题(默认设置)

header=None, 没有列标题

- **header=0，第1行数据作为列标题**

```
df_iris = pd.read_csv(train_path, header=0)
df_iris.head()
```

读取前5行数据

运行结果：

	120	4	setosa	versicolor	virginica
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

- **header=None**，数据文件中没有表头

```
df_iris = pd.read_csv(train_path, header=None)
df_iris.head()
```

运行结果：

	0	1	2	3	4
0	120.0	4.0	setosa	versicolor	virginica
1	6.4	2.8	5.6	2.2	2
2	5.0	2.3	3.3	1.0	1
3	4.9	2.5	4.5	1.7	2
4	4.9	3.1	1.5	0.1	0

系统自动的加上了数字序列 0,1,2,3,4 作为列标题

第一行数据被作为数据样本

设置列标题——names参数

自定义列标题，代替header参数指定的列标题

```
pd.read_csv( filepath_or_buffer, header, names)
```

- **header=0**，第1行做为列标题

	120	4	setosa	versicolor	virginica
0	6.4	2.8	5.6	2.2	2
1	5	2.3	3.3	1	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

- 设置**names参数**，指定新的列标题

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	6.4	2.8	5.6	2.2	2
1	5	2.3	3.3	1	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

指定列标题

```
COLUMN_NAMES = ['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Species']
df_iris = pd.read_csv(train_path, names=COLUMN_NAMES, header=0)
df_iris.head()
```

运行结果：

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

访问数据

head(n)

- **head()函数**：读取**前n行**数据

  参数为**空**时，默认读取二维数据表中的**前5行**数据

```
df_iris.head(8)
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0
5	4.4	3.2	1.3	0.2	0
6	5.4	3.4	1.5	0.4	0
7	6.9	3.1	5.1	2.3	2

tail()函数：读取后n行数据

tail(n)

```
df_iris.tail(8)
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
112	5.0	3.0	1.6	0.2	0
113	6.3	3.3	6.0	2.5	2
114	5.0	3.5	1.6	0.6	0
115	5.5	2.6	4.4	1.2	1
116	5.7	3.0	4.2	1.2	1
117	4.4	2.9	1.4	0.2	0
118	4.8	3.0	1.4	0.1	0
119	5.5	2.4	3.7	1.0	1

使用索引和切片

- 读取**索引值为10-15**的行

```
df_iris[10:16]
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
10	5.2	2.7	3.9	1.4	1
11	6.9	3.1	4.9	1.5	1
12	5.8	4.0	1.2	0.2	0
13	5.4	3.9	1.7	0.4	0
14	7.7	3.8	6.7	2.2	2
15	6.3	3.3	4.7	1.6	1

显示统计信息

describe()方法：显示二维数据的统计信息

```
df_iris.describe()
```

		SepalLength	SepalWidth	PetalLength	PetalWidth	Species
总数	count	120.000000	120.000000	120.000000	120.000000	120.000000
平均值	mean	5.845000	3.065000	3.739167	1.196667	1.000000
标准差	std	0.868578	0.427156	1.822100	0.782039	0.840168
最小值	min	4.400000	2.000000	1.000000	0.100000	0.000000
1/4	25%	5.075000	2.800000	1.500000	0.300000	0.000000
1/2	50%	5.800000	3.000000	4.400000	1.300000	1.000000
3/4	75%	6.425000	3.300000	5.100000	1.800000	2.000000
最大值	max	7.900000	4.400000	6.900000	2.500000	2.000000

DataFrame的常用属性：ndim、size、shape

属　性	描　述
ndim	数据表的维数
shape	数据表的形状
size	数据表元素的总个数

```
>>>df_iris.ndim
2

>>>df_iris.shape
(120, 5)

>>>df_iris.shape
600
```

转化为NumPy数组

- 使用NumPy中的创建数组函数array()

```
>>>iris=np.array(df_iris)

>>>type(df_iris)
pandas.core.frame.DataFrame

>>>type(iris)
numpy.ndarray
```

- 使用DataFrame中的values方法或as_matrix()方法

```
>>>iris=df_iris.values
>>>iris=df_iris.as_matrix()
```

访问数组元素——索引和切片

一维切片
读取**前6行**

```
>>>iris[0:6]
array([[6.4, 2.8, 5.6, 2.2, 2. ],
       [5. , 2.3, 3.3, 1. , 1. ],
       [4.9, 2.5, 4.5, 1.7, 2. ],
       [4.9, 3.1, 1.5, 0.1, 0. ],
       [5.7, 3.8, 1.7, 0.3, 0. ],
       [4.4, 3.2, 1.3, 0.2, 0. ]])
```

二维切片
读取**前6行**中的**前4列**

```
>>>iris[0:6,0:4]
array([[6.4, 2.8, 5.6, 2.2],
       [5. , 2.3, 3.3, 1. ],
       [4.9, 2.5, 4.5, 1.7],
       [4.9, 3.1, 1.5, 0.1],
       [5.7, 3.8, 1.7, 0.3],
       [4.4, 3.2, 1.3, 0.2]])
```

二维切片
所有行中的
鸢尾花种类

```
>>>iris_y=iris[:,4]
array([2., 1., 2., 0., 0., 0., 0., 2., 1., 0., 1., 1., 0., 0., 2., 1., 2.,
       2., 2., 0., 2., 2., 0., 2., 2., 0., 1., 2., 1., 1., 1., 1., 1., 2.,
       2., 2., 2., 2., 0., 0., 2., 2., 2., 0., 0., 2., 0., 2., 0., 2., 0.,
       1., 1., 0., 1., 2., 2., 2., 2., 1., 1., 2., 2., 2., 1., 2., 0., 2.,
       2., 0., 0., 1., 0., 2., 2., 0., 1., 1., 1., 2., 0., 1., 1., 1., 2.,
       0., 1., 1., 1., 0., 2., 1., 0., 0., 2., 0., 0., 2., 1., 0., 0., 1.,
       0., 1., 0., 0., 0., 0., 1., 0., 2., 1., 0., 2., 0., 1., 1., 0., 0.,
       1.])
```