

# 第一章

## 第一节：数据载入及初步观察（以泰坦尼克数据为例）

- 数据的载入方式

`pandas.read_filename(file)`，例如 `pandas.read_csv('train.csv')` 和 `pandas.read_table('train.csv')`。

当然，这只是最基础的载入方式，在涉及到文件编码和分割等问题时还需要用到其他的参数。涉及到路径时需要注意相对路径和绝对路径，其中从文件中复制的路径往往是以"\"分割的，而python读文件的路径则以'/'分割，在使用\"时需要在路径前加上 'r'才能正常读入文件。其中，`pandas.read_csv()`是从csv文件读入数据，`pandas.read_table()`则是从文本文件中读入数据。前者是后者的包装，将后者中的分割符置为','便可得到相同的结果，这其实也与csv文件的编码方式有关，如果将csv文件用文本方式打开，你会发现它是一个以逗号为分隔符的文本列表。如下图所示。

```
PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked
1, 0, 3, "Braund, Mr. Owen Harris", male, 22, 1, 0, A/5 21171, 7.25,, S
2, 1, 1, "Cumings, Mrs. John Bradley (Florence Briggs Thayer)", female, 38, 1, 0, PC 17599, 71.2833, C85,
3, 1, 3, "Heikkinen, Miss. Laina", female, 26, 0, 0, STON/O2. 3101282, 7.925,, S
4, 1, 1, "Futrelle, Mrs. Jacques Heath (Lily May Peel)", female, 35, 1, 0, 113803, 53.1, C123, S
5, 0, 3, "Allen, Mr. William Henry", male, 35, 0, 0, 373450, 8.05,, S
6, 0, 3, "Moran, Mr. James", male,, 0, 0, 330877, 8.4583,, Q
7, 0, 1, "McCarthy, Mr. Timothy J", male, 54, 0, 0, 17463, 51.8625, E46, S
8, 0, 3, "Palsson, Master. Gosta Leonard", male, 2, 3, 1, 349909, 21.075,, S
9, 1, 3, "Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)", female, 27, 0, 2, 347742, 11.1333,, S
```

另外，如果遇到量级特别大的数据，直接将数据全部读入则显得不是那么明智了，此时我们可以使用 `chunksize` 参数将数据分块导入，“分而治之”：

#写入代码

```
chunk=pd.read_csv('train.csv',chunksize=500)
print(type(chunk))
bench=chunk.get_chunk(5)
print(bench)
```

```
<class 'pandas.io.parsers.TextFileReader'>
```

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	
4	5	0	3	

	Name	Sex	Age	SibSp	\
0	Braund, Mr. Owen Harris	male	22	1	
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	
2	Heikkinen, Miss. Laina	female	26	0	
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	
4	Allen, Mr. William Henry	male	35	0	

	Parch	Ticket	Fare	Cabin	Embarked
0	0	A/5 21171	7.2500	NaN	S
1	0	PC 17599	71.2833	C85	C
2	0	STON/O2. 3101282	7.9250	NaN	S
3	0	113803	53.1000	C123	S
4	0	373450	8.0500	NaN	S

分块读出数据之后，使用 `chunk.get_chunk()` 将数据取出转换为df类型。

- 

- 数据的初步探索

数据的探索方式有许多，比如说查看数据的行数和列数、数据的特征名称、标签名等，pandas 也为我们提供了许多的接口供使用。下面将简要介绍几个探索数据的函数。

`df.head()`：查看DataFrame的部分数据，默认显示前5行，可以改变函数的参数调整显示的行数。

`df.info()`：查看数据的基本信息，包括行数、列数、df的数据类型等。

`df.tail()`：其用法与`df.head()`类似，不过`df.tail()`是查看数据的末尾数据，默认也是5行。

`df.isnull()`：返回df中是否为空的布尔值，一般可以结合 `.sum()` 等函数一起使用。

`df.columns`：返回df列名。

- 数据保存

一般而言，我们对数据处理完之后需要将处理好的数据进行保存（提交测试集），pandas也提供了保存数据的函数，`df.to_filename()`，比如说`df.to_csv()`，第一个参数为保存文件的名称，需要注意的是，如果保存的文件中有着特殊的字符（比如说中文）则可能需要调整文件的编码以免出现乱码问题，事实上，在读取文件时也需要注意文件的编码格式。

## 第二节：pandas基础

- DataFrame和Series

DataFrame与Series均是pandas中的基础数据类型，按照我的个人理解，DataFrame是一个张量，而Series则是向量，二者的维度有所区别，通俗点将，DataFrame大一些，它的组成元素是Series。两个函数均可以字典作为参数，将字典转化为pandas的数据结构。

- 查看列的值

在第一节笔记中我写到了`df.columns`对象可用于查看数据的所有列名称，这里不过多赘述；下面记录查看某一列的方法：

`df['列名']` 取出该列的所有元素，返回Series类型的数据，可使用切片的方式取出想要数目的数据。

`df['列名'].unique()` 不重复的查看列中元素，找出所出现值的集合。

- 删除多余的列

在教程中，实例的代码为`del test_1['a']`，在后文中提到了`df.drop()`函数，由此可以想到，`df.drop()`函数也可以用于删除列（记得将`inplace=True`），不过在使用该函数时需要对列进行操作，将`axis`置1，

- pandas中的布尔运算

为了更好地理解教程中的代码，不妨先将教程中的代码拆开看：

```
data["Age"]<10
```

```
0    False
```

```
1    False
```

```
2    False
```

```
3    False
```

```
4    False
```

```
...
```

```
886  False
```

```
887  False
```

```
888  False
```

```
889  False
```

```
890  False
```

```
Name: Age, Length: 891, dtype: bool
```

，发现`data['Age']<10` 返回了Age列的布尔值，如此，`data[data["Age"]<10]` 的含义就很明显了：从data中取出Age<10 的数据。任务

二的代码含义也于此类似。

值得注意的是，取出的数据的索引仍是原来数据的索引，如需重新改变索引值，可以使用

`df.reset_index()` 函数

- 取出特定的行列

`df.iloc[参数1,参数2];df.loc[参数1,参数2]`：其中iloc支持字符取数据，loc支持用索引取数据，参数1取特定行，参数二取特定列。

## 第三节：探索性数据分析

- 对df进行排序

`df.sort_index(axis=0,ascending=True)` 按照索引进行排序，默认按照行索引升序排序，axis=1按照列索引排序，ascending=False时按降序排

`df.sort_value(by='',ascending=True)` 按照列的值进行排序，'by'的参数即所选择的列，默认升序排。

- 数据相加

在pandas中可以使用"+"将2个df中的数据进行相加，返回一个新的df，对应的行和列会变成相加后的结果，没有对应的行和列会成为nan

- 查看数据基本信息

`describe()` 函数，查看数据的均值、均差和四分位数等，DataFrame和Series均有这个函数。

## 第二章

### 第一节：数据清洗及特征处理

#### 查看缺失值：

- `df.info`:返回df的行列信息，可以通过查看每列值的总数判断是否有缺失值。
- `Series.isnull()`:返回一个bool型的array，如果是空值则会在相应的索引出置True，否则为False，可以与`sum()`连用查看缺失值的数目。（True在运算时为1，False为0.）

#### 处理缺失值：

- 最简单的方法：`pd.fillna()`，使用括号里的参数填充缺失值。
- 使用sklearn：`sklearn.preprocessing.SimpleImpute`处理，不过sklearn处理缺失值相比pandas较麻烦，所以不常用。

另外，在处理缺失值时，可以使用预测算法进行缺失值填充，将缺失数量最多的特征作为标签，特征中未缺失的作为train，有缺失值的作为test，将原来的标签加入特征中训练填充缺失值。（具体的案例可以看菜菜的机器学习课堂中用随机森林处理缺失值）

## 处理重复值：

- sklearn处理：sklearn.feature\_select.VarianceThreshold:VarianceThreshold () 可以用于删除方差小于threshold的特征，默认threshold为0，所以可以用来删除重复值。
- df.drop\_duplicates(): 实际上df.duplicate返回的是个bool型的张量，df.drop\_duplicates()则是丢掉了其中值为True的部分。

## 对文本类型进行转换（字符型数据编码）：

- 使用sklearn.preprocessing.OrdinalEncoder: 对数据进行编码，默认从0开始，可将编码后的数字都+1运算得到从1开始的编码，返回np.array数组。使用.categories\_查看特征名。
- sklearn.preprocessing.OneHotEncoder: 独热编码，记得用toarray函数将数值取出，使用.get\_feature\_names()查看特征名。

## 分箱

- pd.cut (x,bins,right=True,labels=None,include\_lowest=False) :按照所传参数进行分箱，默认等位分5箱，当传入的列表时按照列表划分区间分箱（默认前开后闭），可通过将include\_lowest置True调成闭区间。
- Sklearn.preprocessing.KBinsDcretizer: 待处理数据必须大于一维（pd.cut,pd.qcut只能处理一维），有'Ordinal'/'Onehot'的方式给箱子编码，但是KBinsDcretizer在处理一维数据时无法像pd.cut()一样能够按照指定区间或者指定百分数区间划分，它只能在等宽（strategy='uniform', 箱中极差相等）等位（strategy='quantile',箱中特征数目相等）和聚类（strategy='kmeans'）中选择分箱方法。