



# Introduction to Python for Data Science

## 资料科学入门 – 数据维护与清理

Jan 2021

Microsoft Reactor | Ryan Chung

```
led by player to  
s.load_image("kg.png")  
  
[self]:  
ialize Dog object and create Text o  
g, self).__init__(image = Dog.image  
x = games.mouse.x  
bottom = games.sc  
  
re = games.Text(value = 0, size = 24  
top = 5, right = gam  
  
reen.add(self.score)  
1 = games.Text(value = 0, size = 24  
top = 5, left = gam
```



# Ryan Chung

Instructor / DevelopIntelligence  
Founder / MobileDev.TW

@ryanchung403 on WeChat







# Reactor



[developer.microsoft.com/reactor/](https://developer.microsoft.com/reactor/)  
@MSFTReactor on Twitter

# Overview 综览

- DataFrame 资讯探索
- 遗漏值处理
  - 确认、移除、填补
- 数据集合并
  - Numpy、Pandas 序列、Pandas DataFrame
- 案例分析
  - 数据集观察、关联性分析、数据视觉化

真实世界中  
数据分析型的专案  
八成以上的时间都花在  
数据的清理与准备





# Anderson's Iris data set / Iris flower data set

## 安德森鸢尾花卉数据集

样本数：150

类别：0-Setosa 山鸢尾、1-Versicolour 变色鸢尾、2-Virginica 维吉尼亚鸢尾

	花萼长度	花萼宽度	花瓣长度	花瓣宽度	类别
index ▲	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0
10	5.4	3.7	1.5	0.2	0

# DataFrame 资讯探索

dataFrameExplor.py > ...

```
1 import numpy as np
2 import pandas as pd
3 from sklearn import datasets
4 iris = datasets.load_iris()
5 iris_df = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
6                        columns= iris['feature_names'] + ['class'])
```

iris\_df.head()

head() 取最前面几笔(预设值5笔)

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

iris\_df.tail()

tail() 取最后面几笔(预设值5笔)

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
145	6.7	3.0	5.2	2.3	2.0
146	6.3	2.5	5.0	1.9	2.0
147	6.5	3.0	5.2	2.0	2.0
148	6.2	3.4	5.4	2.3	2.0
149	5.9	3.0	5.1	1.8	2.0

iris\_df.info()

info() 资料集摘要资讯

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal length (cm)    150 non-null float64
sepal width (cm)     150 non-null float64
petal length (cm)    150 non-null float64
petal width (cm)     150 non-null float64
class                150 non-null float64
dtypes: float64(5)
memory usage: 5.9 KB
```

iris\_df.shape  
shape 维度  
(150笔, 5个栏位)  
(150, 5)

iris\_df['sepal length (cm)'].mean()  
mean() 计算平均值  
5.8433333333333335

# 遗漏值处理

- Python 的空值: None
- NumPy/pandas 的空值: NaN

```
import numpy as np
example1 = np.array([2, None, 6, 8])
```

```
[14] example1
array([2, None, 6, 8], dtype=object)
```

```
[15] type(example1[0])
int
[16] type(example1[1])
NoneType
```

```
[17] type(example1[2])
int
[18] type(example1[3])
int
```

```
[19] example1.sum()
```



```
TypeError: unsupported operand type(s)
for +: 'int' and 'NoneType'
```

None的存在导致加总运算无法正常运行



# 遗漏值处理

- Python 的空值: None
- NumPy/pandas 的空值: NaN

```
import numpy as np  
example2 = np.array([2, np.nan, 6, 8])
```

```
[29] example2  
array([ 2., nan,  6.,  8.])
```

```
[30] type(example2[0])  
numpy.float64
```

```
[31] type(example2[1])  
numpy.float64
```

```
[32] type(example2[2])  
numpy.float64
```

```
[33] type(example2[3])  
numpy.float64
```

```
[34] example2.sum()
```

```
nan
```

加总运算可以运行但结果是....

有缺失值时不论哪一种都造成了问题！

# 遗失值处理方式：判断、舍去与填补

方法	用途	备注
isnull()	判断是否为空值	在原本每个值的位置回传布尔值
notnull()	判断是否不是空值	与isnull()反义

方法	用途	备注
dropna()	把遗失值该列删去	资料量够大
fillna(x)	以指定值x来替代遗失值	可指定参数决定替代值
ffill()	用前面一个的值来替代	全名：forward fill
bfill()	用后面一个的值来替代	全名：backward fill

# 遗漏值处理

- Python 的空值: None
- NumPy/pandas 的空值: NaN

```
import numpy as np
import pandas as pd
example3 = pd.Series([0,np.nan,' ',None])
```

原始内容

```
[43] example3
0      0
1     NaN
2
3     None
dtype: object
```

判断是否为空值

```
[44] example3.isnull()
0      False
1       True
2      False
3       True
dtype: bool
```

判断是否不为空值

```
[46] example3.notnull()
0      True
1     False
2      True
3     False
dtype: bool
```

扔掉空值

```
[47] example3.dropna()
0      0
2
dtype: object
```

# 遗漏值舍去与填补

- Python 的空值: None
- NumPy/pandas 的空值: NaN

```
import numpy as np
import pandas as pd
example4 = pd.DataFrame([[1,np.nan,7],
[2,5,8],[np.nan,6,9]])
```

原始内容

有空值就扔掉(横列为单位)

有空值就扔掉(直栏为单位)

[49] example4



	0	1	2
0	1.0	NaN	7
1	2.0	5.0	8
2	NaN	6.0	9

[50] example4.dropna()



	0	1	2
1	2.0	5.0	8

[51] example4.dropna(axis='columns')



	2
0	7
1	8
2	9



# 条件式遗漏值舍去

- thresh : 需要有n个非空值
- NumPy/pandas 的空值: NaN

```
import numpy as np
import pandas as pd
example4 = pd.DataFrame([[1,np.nan,7],
[2,5,8],[np.nan,6,9]])
example4[3] = np.nan
```

原始内容

[53] example4



	0	1	2	3
0	1.0	NaN	7	NaN
1	2.0	5.0	8	NaN
2	NaN	6.0	9	NaN

[54] example4.dropna(axis='rows',thresh=3)



	0	1	2	3
1	2.0	5.0	8	NaN

第0列跟第2列的非空值未达3个，扔掉

# fillna(x) : 填补特定值

```
import numpy as np
import pandas as pd
```

```
example5 = pd.Series([1,np.nan,2,None,3],index=list('abcde'))
```

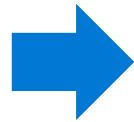
```
example5.fillna(0)
```

[20] example5



a	1.0
b	NaN
c	2.0
d	NaN
e	3.0

dtype: float64



[21] example5.fillna(0)



a	1.0
b	0.0
c	2.0
d	0.0
e	3.0

dtype: float64

# ffill() : forward fill

```
import numpy as np
import pandas as pd
```

```
example5 = pd.Series([1,np.nan,2,None,3],index=list('abcde'))
```

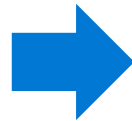
```
example5.fillna(method='ffill')
```

[20] example5



a	1.0
b	NaN
c	2.0
d	NaN
e	3.0

dtype: float64



[29] example5.fillna(method='ffill')



a	1.0
b	1.0
c	2.0
d	2.0
e	3.0

dtype: float64

# bfill() : backward fill

```
import numpy as np
import pandas as pd
```

```
example5 = pd.Series([1,np.nan,2,None,3],index=list('abcde'))
```

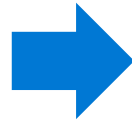
```
example5.fillna(method='bfill')
```

[20] example5



a	1.0
b	NaN
c	2.0
d	NaN
e	3.0

dtype: float64



[30] example5.fillna(method='bfill')



a	1.0
b	2.0
c	2.0
d	3.0
e	3.0

dtype: float64



# ffill() : forward fill 指定方向

```
import numpy as np
import pandas as pd
example4 = pd.DataFrame([[1,np.nan,7],[2,5,8],[np.nan,6,9]])
example4[3] = np.nan
example4.dropna(axis='rows',thresh=3)
example4
example4.fillna(method='ffill',axis=1)
```

[31] example4



	0	1	2	3
0	1.0	NaN	7	NaN
1	2.0	5.0	8	NaN
2	NaN	6.0	9	NaN



[32] example4.fillna(method='ffill',axis=1)



	0	1	2	3
0	1.0	1.0	7.0	7.0
1	2.0	5.0	8.0	8.0
2	NaN	6.0	9.0	9.0

我左边没坐人没得抄😭😭

# ffill() : forward fill

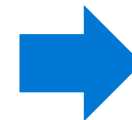
```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        "A": [5, 3, None, 4],
        "B": [None, 2, 4, 3],
        "C": [4, 3, 8, 5],
        "D": [5, 4, 2, None]
    }
)

df.ffmpeg(axis=0)
```

用前面一个row来填补

	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	NaN	4.0	8	2.0
3	4.0	3.0	5	NaN



	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	3.0	4.0	8	2.0
3	4.0	3.0	5	2.0

我上面没坐人没得抄



# ffill() : forward fill

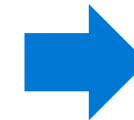
```
import numpy as np
import pandas as pd
```

```
df = pd.DataFrame(
    {
        "A": [5, 3, None, 4],
        "B": [None, 2, 4, 3],
        "C": [4, 3, 8, 5],
        "D": [5, 4, 2, None]
    }
)
```

```
df.ffill(axis=1)
```

用前面一个column来填补

	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	NaN	4.0	8	2.0
3	4.0	3.0	5	NaN



	A	B	C	D
0	5.0	5.0	4.0	5.0
1	3.0	2.0	3.0	4.0
2	NaN	4.0	8.0	2.0
3	4.0	3.0	5.0	5.0

# bfill() : backward fill

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        "A": [5, 3, None, 4],
        "B": [None, 2, 4, 3],
        "C": [4, 3, 8, 5],
        "D": [5, 4, 2, None]
    }
)

df.bfill(axis=0)
```

用后面一个row来填补

	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	NaN	4.0	8	2.0
3	4.0	3.0	5	NaN



	A	B	C	D
0	5.0	2.0	4	5.0
1	3.0	2.0	3	4.0
2	4.0	4.0	8	2.0
3	4.0	3.0	5	NaN

我后面没坐人没得抄





# bfill() : backward fill

```
import numpy as np
import pandas as pd
```

```
df = pd.DataFrame(
    {
        "A": [5, 3, None, 4],
        "B": [None, 2, 4, 3],
        "C": [4, 3, 8, 5],
        "D": [5, 4, 2, None]
    }
)
```

```
df.bfill(axis=1)
```

用后面一个column来填补

	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	NaN	4.0	8	2.0
3	4.0	3.0	5	NaN



	A	B	C	D
0	5.0	4.0	4.0	5.0
1	3.0	2.0	3.0	4.0
2	4.0	4.0	8.0	2.0
3	4.0	3.0	5.0	NaN



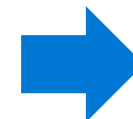
# fillna() 带参数用法

```
import numpy as np
import pandas as pd

df = pd.DataFrame({
    'ColA': [1, np.nan, np.nan, 4, 5, 6, 7],
    'ColB': [1, 1, 1, 1, 2, 2, 2]
})

df.fillna(value=0)
```

	ColA	ColB
0	1.0	1
1	NaN	1
2	NaN	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2



	ColA	ColB
0	1.0	1
1	0.0	1
2	0.0	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

# fillna() 带参数用法

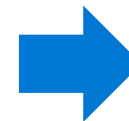
```
import numpy as np
import pandas as pd

df = pd.DataFrame({
    'ColA': [1, np.nan, np.nan, 4, 5, 6, 7],
    'ColB': [1, 1, 1, 1, 2, 2, 2]
})
```

```
df.fillna(value=df.mean())
```

$$(1+4+5+6+7)/5 = 4.6$$

	ColA	ColB
0	1.0	1
1	NaN	1
2	NaN	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2



	ColA	ColB
0	1.0	1
1	4.6	1
2	4.6	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

# fillna() 带参数用法

```
import numpy as np
import pandas as pd

df = pd.DataFrame({
    'ColA': [1, np.nan, np.nan, 4, 5, 6, 7],
    'ColB': [1, 1, 1, 1, 2, 2, 2]
})
```

```
df.fillna(method='ffill', axis=0)
```

用前面一个row来填补

	ColA	ColB
0	1.0	1
1	NaN	1
2	NaN	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2



	ColA	ColB
0	1.0	1
1	1.0	1
2	1.0	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2



# fillna() 带参数用法

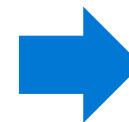
```
import numpy as np
import pandas as pd

df = pd.DataFrame({
    'ColA': [1, np.nan, np.nan, 4, 5, 6, 7],
    'ColB': [1, 1, 1, 1, 2, 2, 2]
})
```

```
df.fillna(method='bfill', axis=0)
```

用前面一个row来填补

	ColA	ColB
0	1.0	1
1	NaN	1
2	NaN	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2



	ColA	ColB
0	1.0	1
1	4.0	1
2	4.0	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

# 重复值

- 判断是否有重复值
- 删去重复值

```
import numpy as np
import pandas as pd
example6 = pd.DataFrame({'letters': ['A', 'B']*2 + ['B'],
                          'numbers': [1, 2, 1, 3, 3]})
```

[53] example6



	letters	numbers
0	A	1
1	B	2
2	A	1
3	B	3
4	B	3

[55] example6.duplicated()



```
0    False
1    False
2     True
3    False
4     True
dtype: bool
```

[56] example6.drop\_duplicates()



	letters	numbers
0	A	1
1	B	2
3	B	3

[57] example6.drop\_duplicates(['letters'])



	letters	numbers
0	A	1
1	B	2

# 资料集合并(一) Numpy

concatenate()

```
import numpy as np
array_one = np.array([1,2,3])
array_two = np.array([4,5,6])
array_concat = np.concatenate([array_one,array_two])
array_concat
array_stack = np.array([array_one,array_two])
array_stack
```

```
[67] array_one
array([1, 2, 3])
```

```
[68] array_two
array([4, 5, 6])
```

```
[69] array_concat
array([1, 2, 3, 4, 5, 6])
```

```
[70] array_stack
array([[1, 2, 3],
       [4, 5, 6]])
```

# 资料集合并(二) Pandas 序列合并

concatSeries.py > ...

```
1 import numpy as np
2 import pandas as pd
3 series1=pd.Series({'王明':90,'柳宇':40,'張三':70})
4 series2=pd.Series({'范武':60,'陳實':30,'張揚':90})
5 series3=pd.concat([series1,series2])
```

series1

王明	90
柳宇	40
張三	70
dtype:	int64

series2

范武	60
陳實	30
張揚	90
dtype:	int64

series3

王明	90
柳宇	40
張三	70
范武	60
陳實	30
張揚	90
dtype:	int64

# 资料集合并(三) Pandas DataFrame合并

```
import numpy as np
import pandas as pd
df0a = pd.DataFrame({'employee': ['Gary', 'Stu'], 'group': ['Accounting', 'Marketing']})
df0b = pd.DataFrame({'employee': ['Mary', 'Sue'], 'group': ['Marketing', 'HR']})
df1 = pd.concat([df0a, df0b], ignore_index=True)
```

也可以写成

```
df1 = df0a.append(df0b, ignore_index=True)
```

[14] df0a

	employee	group
0	Gary	Accounting
1	Stu	Marketing

[15] df0b

	employee	group
0	Mary	Marketing
1	Sue	HR

[19] df1

	employee	group
0	Gary	Accounting
1	Stu	Marketing
2	Mary	Marketing
3	Sue	HR

# 资料集合并(三) Pandas DataFrame合并

```
import numpy as np
import pandas as pd
df1 = pd.DataFrame({'employee': ['Gary', 'Stu', 'Mary', 'Sue'],
                    'group': ['Accounting', 'Marketing', 'Marketing', 'HR']})
df1
df2 = pd.DataFrame({'employee': ['Gary', 'Stu', 'Mary', 'Sue'],
                    'hire_date': [2008, 2012, 2017, 2018]})
df2
df3 = pd.merge(df1, df2)
df3
```

[4] df1

	employee	group
0	Gary	Accounting
1	Stu	Marketing
2	Mary	Marketing
3	Sue	HR

[6] df2

	employee	hire_date
0	Gary	2008
1	Stu	2012
2	Mary	2017
3	Sue	2018

[8] df3

	employee	group	hire_date
0	Gary	Accounting	2008
1	Stu	Marketing	2012
2	Mary	Marketing	2017
3	Sue	HR	2018

# 资料集合并(三) Pandas DataFrame合并

## 明确指定合并用栏位

```
df4 = pd.DataFrame({'group': ['Accounting', 'Marketing', 'HR'],  
                    'supervisor': ['Carlos', 'Giada', 'Stephanie']})
```

df4

```
pd.merge(df3, df4, on='group')
```

[8] df3



	employee	group	hire_date
0	Gary	Accounting	2008
1	Stu	Marketing	2012
2	Mary	Marketing	2017
3	Sue	HR	2018

[12] pd.merge(df3, df4, on='group')



	employee	group	hire_date	supervisor
0	Gary	Accounting	2008	Carlos
1	Stu	Marketing	2012	Giada
2	Mary	Marketing	2017	Giada
3	Sue	HR	2018	Stephanie

[11] df4



	group	supervisor
0	Accounting	Carlos
1	Marketing	Giada
2	HR	Stephanie



# 资料集合并(三) Pandas DataFrame合并

## 明确指定合并用栏位

```
df5 = pd.DataFrame({'group': ['Accounting', 'Accounting', 'Marketing', 'Marketing', 'HR', 'HR'],  
                    'core_skills': ['math', 'spreadsheets', 'writing', 'communication', 'spreadsheets', 'organization']})
```

df1

df5

```
pd.merge(df1, df5, on='group')
```

[19] df1



	employee	group
0	Gary	Accounting
1	Stu	Marketing
2	Mary	Marketing
3	Sue	HR

[20] df5



	group	core_skills
0	Accounting	math
1	Accounting	spreadsheets
2	Marketing	writing
3	Marketing	communication
4	HR	spreadsheets
5	HR	organization

[21] pd.merge(df1, df5)



	employee	group	core_skills
0	Gary	Accounting	math
1	Gary	Accounting	spreadsheets
2	Stu	Marketing	writing
3	Stu	Marketing	communication
4	Mary	Marketing	writing
5	Mary	Marketing	communication
6	Sue	HR	spreadsheets
7	Sue	HR	organization

# 资料集合并(三) Pandas DataFrame合并

合并栏位名称不同但意义相同时

```
df6 = pd.DataFrame({'name': ['Gary', 'Stu', 'Mary', 'Sue'], 'salary': [70000, 80000, 120000, 90000]})
df6
df1
df7 = pd.merge(df1, df6, left_on='employee', right_on='name')
df7
df7.drop('name', axis=1, inplace=True)
```

[23] df6

	name	salary
0	Gary	70000
1	Stu	80000
2	Mary	120000
3	Sue	90000

[24] df1

	employee	group
0	Gary	Accounting
1	Stu	Marketing
2	Mary	Marketing
3	Sue	HR

[25] pd.merge(df1, df6, left\_on='employee', right\_on='name')

	employee	group	name	salary
0	Gary	Accounting	Gary	70000
1	Stu	Marketing	Stu	80000
2	Mary	Marketing	Mary	120000
3	Sue	HR	Sue	90000

[31] df7

	employee	group	salary
0	Gary	Accounting	70000
1	Stu	Marketing	80000
2	Mary	Marketing	120000
3	Sue	HR	90000

# 资料集合并(三) Pandas DataFrame合并

以index来进行合并

df1

```
df1a = df1.set_index('employee')
```

df1a

```
df2a = df2.set_index('employee')
```

df2a

```
pd.merge(df1a, df2a, left_index=True, right_index=True)
```

[32] df1



	employee	group
0	Gary	Accounting
1	Stu	Marketing
2	Mary	Marketing
3	Sue	HR

[41] df2



	employee	hire_date
0	Gary	2008
1	Stu	2012
2	Mary	2017
3	Sue	2018

[34] df1a



	group
employee	
Gary	Accounting
Stu	Marketing
Mary	Marketing
Sue	HR

[36] df2a



	hire_date
employee	
Gary	2008
Stu	2012
Mary	2017
Sue	2018

[40]



```
pd.merge(df1a, df2a, left_index=True, right_index=True)
```

	group	hire_date
employee		
Gary	Accounting	2008
Stu	Marketing	2012
Mary	Marketing	2017
Sue	HR	2018

# 资料集合并(三) Pandas DataFrame合并

## index与on混合用法

```
df7 = pd.merge(df1a, df6, left_index=True, right_on='name')
df7 = df7.set_index('name')
df7.index.name = 'employee'
```

[34] df1a

	group
employee	
Gary	Accounting
Stu	Marketing
Mary	Marketing
Sue	HR

[42] df6

	name	salary
0	Gary	70000
1	Stu	80000
2	Mary	120000
3	Sue	90000

[72] df7

	group	name	salary
0	Accounting	Gary	70000
1	Marketing	Stu	80000
2	Marketing	Mary	120000
3	HR	Sue	90000

	group	salary
name		
Gary	Accounting	70000
Stu	Marketing	80000
Mary	Marketing	120000
Sue	HR	90000

	group	salary
employee		
Gary	Accounting	70000
Stu	Marketing	80000
Mary	Marketing	120000
Sue	HR	90000

# 资料集合并(三) Pandas DataFrame合并

how : inner / outer / left / right

```
df5 = pd.DataFrame({'group': ['Engineering', 'Marketing', 'Sales'],  
                    'core_skills': ['math', 'writing', 'communication']})  
pd.merge(df1, df5, on='group', how='inner')  
pd.merge(df1, df5, on='group', how='outer')  
pd.merge(df1, df5, how='left')  
pd.merge(df1, df5, how='right')
```

```
[89] df1
```

	employee	group
0	Gary	Accounting
1	Stu	Marketing
2	Mary	Marketing
3	Sue	HR

```
[90] df5
```

	group	core_skills
0	Engineering	math
1	Marketing	writing
2	Sales	communication

```
[91] pd.merge(df1, df5, on='group', how='inner')
```

	employee	group	core_skills
0	Stu	Marketing	writing
1	Mary	Marketing	writing

```
[92] pd.merge(df1, df5, on='group', how='outer')
```

	employee	group	core_skills
0	Gary	Accounting	NaN
1	Stu	Marketing	writing
2	Mary	Marketing	writing
3	Sue	HR	NaN
4	NaN	Engineering	math
5	NaN	Sales	communication

```
[93] pd.merge(df1, df5, how='left')
```

	employee	group	core_skills
0	Gary	Accounting	NaN
1	Stu	Marketing	writing
2	Mary	Marketing	writing
3	Sue	HR	NaN

```
[94] pd.merge(df1, df5, how='right')
```

	employee	group	core_skills
0	NaN	Engineering	math
1	Stu	Marketing	writing
2	Mary	Marketing	writing
3	NaN	Sales	communication

# 资料集合并(三) Pandas DataFrame合并

## 冲突合并

```
df_rank1 = pd.DataFrame({'name': ['Gary', 'Stu', 'Mary', 'Sue'], 'rank': [1, 2, 3, 4]})
df_rank2 = pd.DataFrame({'name': ['Gary', 'Stu', 'Mary', 'Sue'], 'rank': [3, 1, 4, 2]})
pd.merge(df_rank1, df_rank2, on='name')
pd.merge(df_rank1, df_rank2, on='name', suffixes=['_A', '_B'])
```

[100] df\_rank1

	name	rank
0	Gary	1
1	Stu	2
2	Mary	3
3	Sue	4

[101] df\_rank2

	name	rank
0	Gary	3
1	Stu	1
2	Mary	4
3	Sue	2

	name	rank_x	rank_y
0	Gary	1	3
1	Stu	2	1
2	Mary	3	4
3	Sue	4	2

	name	rank_A	rank_B
0	Gary	1	3
1	Stu	2	1
2	Mary	3	4
3	Sue	4	2

# 资料集合并(三) Pandas DataFrame合并

```
import numpy as np
import pandas as pd
df10 = pd.DataFrame({'A': ['a', 'd'], 'B': ['b', 'e'], 'C': ['c', 'f']})
df11 = pd.DataFrame({'B': ['u', 'x'], 'C': ['v', 'y'], 'D': ['w', 'z']})
```

```
pd.concat([df10, df11])
pd.concat([df10, df11], join='inner')
```

```
[22] df10
```

	A	B	C
0	a	b	c
1	d	e	f

```
[23] df11
```

	B	C	D
0	u	v	w
1	x	y	z

```
[24] pd.concat([df10, df11])
```

	A	B	C	D
0	a	b	c	NaN
1	d	e	f	NaN
0	NaN	u	v	w
1	NaN	x	y	z

```
[25] pd.concat([df10, df11], join='inner')
```

	B	C
0	b	c
1	e	f
0	u	v
1	x	y



# 数据清理与维护

- 数据内容确认
  - 看个几笔资料 `head()`、`tail()`
  - 摘要信息 `info()`
  - 维度 `shape`
- 遗漏值处理
  - 确认遗漏值 `isnull()`
  - 移除 – 数据量允许情况最合适处理方式 `dropna()`
  - 填补 – 前后左右、平均值、频率最高、其他模型产生 (均为替代折衷方案) `fillna()`、`ffill()`、`bfill()`
- 资料集合并
  - Numpy `concatenate()`
  - Pandas 序列 `concat()`
  - Pandas DataFrame `concat()`、`append()`、`merge()`



# 数据集探索：Boston Housing Dataset



Source : [boston.curbed.com](https://boston.curbed.com)



# 数据载入与初步观察

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
df.head()
df.info()
df.describe()
```

[6] df.head()



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	5.33	36.2

# 数据载入与初步观察

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
df.head()
df.info()
df.describe()
```

```
[7] df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   CRIM        506 non-null    float64
 1   ZN          506 non-null    float64
 2   INDUS       506 non-null    float64
 3   CHAS        506 non-null    float64
 4   NOX         506 non-null    float64
 5   RM          506 non-null    float64
 6   AGE         506 non-null    float64
 7   DIS         506 non-null    float64
 8   RAD         506 non-null    float64
 9   TAX         506 non-null    float64
10  PTRATIO     506 non-null    float64
11  LSTAT       506 non-null    float64
12  MEDV        506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

# 数据载入与初步观察

```
[7] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
#   Column      Non-Null Count  Dtype
---  -
0   CRIM        506 non-null    float64
1   ZN          506 non-null    float64
2   INDUS       506 non-null    float64
3   CHAS        506 non-null    float64
4   NOX         506 non-null    float64
5   RM          506 non-null    float64
6   AGE         506 non-null    float64
7   DIS         506 non-null    float64
8   RAD         506 non-null    float64
9   TAX         506 non-null    float64
10  PTRATIO     506 non-null    float64
11  LSTAT       506 non-null    float64
12  MEDV        506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

```
[9] df.shape

(506, 13)
```

#	英文栏位名称	中文栏位名称	说明	对房价的影响
0	CRIM	犯罪率	该城镇的人均犯罪率	-
1	ZN	住宅用地比例	25,000 平方英尺	+
2	INDUS	商业用地比例	非零售商业区、英亩	-
3	CHAS	河畔	1靠河、0不靠河	+
4	NOX	一氧化氮浓度	千万分点浓度	-
5	RM	平均房间数	每间住宅中	+
6	AGE	屋龄	自1940年计	-
7	DIS	就业中心距离	5个中心加权计算	-
8	RAD	邻近高速公路	等级区分	+
9	TAX	房屋税率	每1万美元的税率	-
10	PTRATIO	生师比	学生/老师的比例	-
11	LSTAT	蓝领比例	劳工、较低教育程度	-
12	MEDV	自用宅中位数	单位：1,000美元	目标值

# 数据载入与初步观察

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
df.head()
df.info()
df.describe()
```

[8] df.describe()

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.593761	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	12.653063	22.532806
std	8.596783	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	11.360000	21.200000
75%	3.647423	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	37.970000	50.000000

# 数据载入与初步观察

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
df.head()
df.info()
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
CRIM	506.0	3.593761	8.596783	0.00632	0.082045	0.25651	3.647423	88.9762
ZN	506.0	11.363636	23.322453	0.00000	0.000000	0.00000	12.500000	100.0000
INDUS	506.0	11.136779	6.860353	0.46000	5.190000	9.69000	18.100000	27.7400
CHAS	506.0	0.069170	0.253994	0.00000	0.000000	0.00000	0.000000	1.0000
NOX	506.0	0.554695	0.115878	0.38500	0.449000	0.53800	0.624000	0.8710
RM	506.0	6.284634	0.702617	3.56100	5.885500	6.20850	6.623500	8.7800
AGE	506.0	68.574901	28.148861	2.90000	45.025000	77.50000	94.075000	100.0000
DIS	506.0	3.795043	2.105710	1.12960	2.100175	3.20745	5.188425	12.1265
RAD	506.0	9.549407	8.707259	1.00000	4.000000	5.00000	24.000000	24.0000
TAX	506.0	408.237154	168.537116	187.00000	279.000000	330.00000	666.000000	711.0000
PTRATIO	506.0	18.455534	2.164946	12.60000	17.400000	19.05000	20.200000	22.0000
LSTAT	506.0	12.653063	7.141062	1.73000	6.950000	11.36000	16.955000	37.9700
MEDV	506.0	22.532806	9.197104	5.00000	17.025000	21.20000	25.000000	50.0000



# 数据载入与初步观察 - 个别数据取出

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
df.head()
df.info()
df.describe().T
```

```
df['MEDV'].mean()
df['MEDV'].max()
df['AGE'].median()
df['AGE'].count()
```

```
[11] df['MEDV'].mean()
22.532806324110698
```

```
[12] df['MEDV'].max()
50.0
```

```
[13] df['AGE'].median()
77.5
```

```
[15] df['AGE'].count()
506
```

# 数据载入与初步观察 – 特定族群观察

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
df.head()
df.info()
df.describe().T

df.groupby(['AGE'])['MEDV'].mean()
```

依据屋龄，来观察自用宅的中位数平均

```
AGE
2.9      26.600000
6.0      24.100000
6.2      23.400000
6.5      24.700000
6.6      24.750000
...
98.8     14.500000
98.9     13.066667
99.1     10.900000
99.3     17.800000
100.0     16.920930
Name: MEDV, Length: 356, dtype: float64
```

# 数据载入与初步观察 – 特定族群观察

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
df.head()
df.info()
df.describe().T

df.groupby(['AGE'])[ 'MEDV' ].mean()
```

依据屋龄，来观察自用宅的中位数平均

练习：  
依据自用宅中位数，列出屋龄的中位数

```
MEDV
5.0      100.00
5.6      100.00
6.3       77.80
7.0       99.15
7.2      100.00
...
46.7      17.00
48.3      70.40
48.5      33.20
48.8      91.50
50.0      90.20
Name: AGE, Length: 229, dtype: float64
```

# 数据转换 – 依据特定条件产生新的栏位

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')

df['AGE_50'] = df['AGE'].apply(lambda x: x>50)
```

屋龄是否大于50

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV	AGE_50
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	4.98	24.0	True
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	9.14	21.6	True
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	4.03	34.7	True
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	2.94	33.4	False
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	5.33	36.2	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	9.67	22.4	True
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	9.08	20.6	True
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	5.64	23.9	True
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	6.48	22.0	True
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	7.88	11.9	True

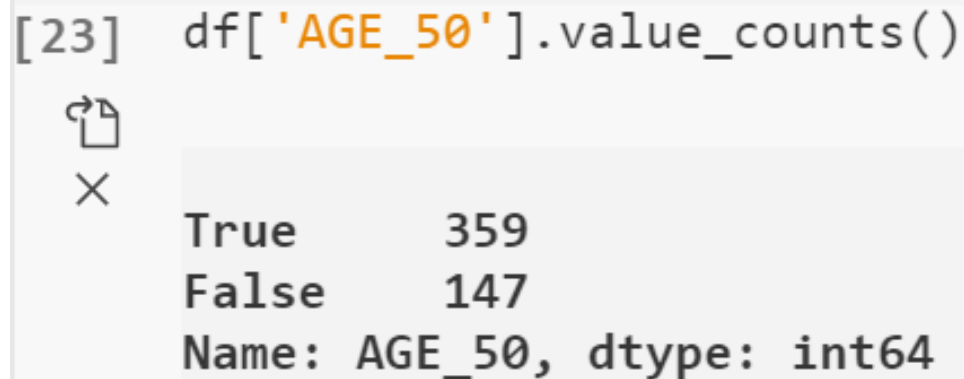
# 数据转换 – 依据特定条件产生新的栏位

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')

df['AGE_50'] = df['AGE'].apply(lambda x: x>50)

df['AGE_50'].value_counts()
```

```
[23] df['AGE_50'].value_counts()
```



True	359
False	147

Name: AGE\_50, dtype: int64

# 数据转换 – 依据特定条件产生新的栏位

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')

df['AGE_50'] = df['AGE'].apply(lambda x: x>50)
df['AGE_50'].value_counts()

df.groupby(['AGE_50'])['MEDV'].mean()
```

结合前面的语法，列出依据屋龄是否大于50，各别的自用宅中位数平均

```
[24] df.groupby(['AGE_50'])['MEDV'].mean()
```



```
AGE_50
False    26.693197
True     20.829248
Name: MEDV, dtype: float64
```

# 数据转换 – 多重groupby

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
df['AGE_50'] = df['AGE'].apply(lambda x: x>50)
df['AGE_50'].value_counts()
df.groupby(['AGE_50', 'RAD', 'CHAS'])['MEDV'].mean()
```

依据屋龄是否大于50、高速公路邻近、河畔三个因素来列房价的平均值

# 数据转换 – 多重groupby

依据屋龄是否大于50、高速公路邻近、河畔三个因素来列房价的平均值

## Number talks

- 屋龄低、近高速公路、河畔 => 高价
- 屋龄高，是不是河畔差异不显着

AGE_50	RAD	CHAS	
False	1.0	0.0	24.666667
		1.0	50.000000
		0.0	33.300000
		0.0	26.505556
	4.0	0.0	25.376744
		1.0	32.900000
	5.0	0.0	26.302857
		1.0	46.000000
	6.0	0.0	23.575000
	7.0	0.0	28.563636
	8.0	0.0	29.220000
	24.0	0.0	20.766667
True	1.0	0.0	20.185714
		0.0	24.170588
		0.0	29.350000
		1.0	27.950000
	4.0	0.0	17.879661
		1.0	21.560000
	5.0	0.0	25.124638
		1.0	25.610000
	6.0	0.0	19.822222
	7.0	0.0	24.433333
	8.0	0.0	32.321429
		1.0	26.000000
	24.0	0.0	15.306612
		1.0	31.362500

Name: MEDV, dtype: float64



# 数据转换 – 多重groupby拆解

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
df['AGE_50'] = df['AGE'].apply(lambda x: x>50)
df['AGE_50'].value_counts()

groupby_three = df.groupby(['AGE_50', 'RAD', 'CHAS'])['MEDV'].mean()
groupby_three.unstack()
```

# 数据转换 – 多重groupby拆解

groupby\_three.unstack()

	CHAS	0.0	1.0
AGE_50	RAD		
False	1.0	24.666667	50.0000
	2.0	33.300000	NaN
	3.0	26.505556	NaN
	4.0	25.376744	32.9000
	5.0	26.302857	46.0000
	6.0	23.575000	NaN
	7.0	28.563636	NaN
	8.0	29.220000	NaN
	24.0	20.766667	NaN
True	1.0	20.185714	NaN
	2.0	24.170588	NaN
	3.0	29.350000	27.9500
	4.0	17.879661	21.5600
	5.0	25.124638	25.6100
	6.0	19.822222	NaN
	7.0	24.433333	NaN
	8.0	32.321429	26.0000
	24.0	15.306612	31.3625

groupby\_three.unstack(-2)

		RAD	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	24.0
AGE_50	CHAS										
False	0.0	24.666667	33.300000	26.505556	25.376744	26.302857	23.575000	28.563636	29.220000	20.766667	
	1.0	50.000000	NaN	NaN	32.900000	46.000000	NaN	NaN	NaN	NaN	
True	0.0	20.185714	24.170588	29.350000	17.879661	25.124638	19.822222	24.433333	32.321429	15.306612	
	1.0	NaN	NaN	27.950000	21.560000	25.610000	NaN	NaN	26.000000	31.362500	

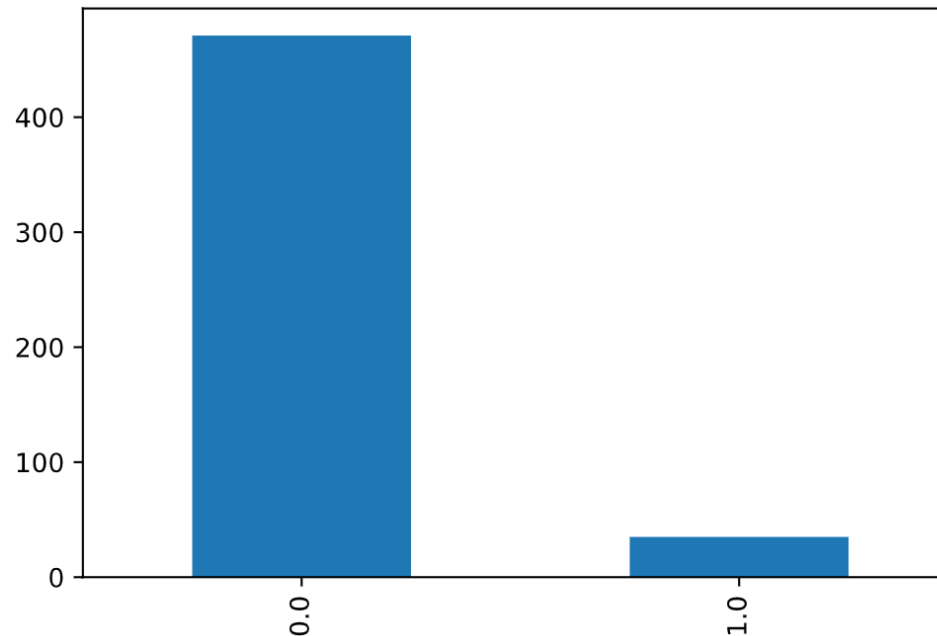
# 数据转换 – 视觉化

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')

df['CHAS'].unique()
df['CHAS'].value_counts()
%matplotlib inline
df['CHAS'].value_counts().plot(kind='bar')
```

```
[40] df['CHAS'].unique()
array([0., 1.])

[41] df['CHAS'].value_counts()
0.0    471
1.0     35
Name: CHAS, dtype: int64
```



# 关联性分析

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')

corr = df.corr(method='pearson')
corr_with_homevalue = corr.iloc[-2]
corr_with_homevalue.sort_values(ascending=False)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV	AGE_50
CRIM	1.000000	-0.199458	0.404471	-0.055295	0.417521	-0.219940	0.350784	-0.377904	0.622029	0.579564	0.288250	0.452220	-0.385832	0.254574
ZN	-0.199458	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	-0.412995	0.360445	-0.590769
INDUS	0.404471	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	0.603800	-0.483725	0.516001
CHAS	-0.055295	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	-0.053929	0.175260	0.088659
NOX	0.417521	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	0.590879	-0.427321	0.597644
RM	-0.219940	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	-0.613808	0.695360	-0.164465
AGE	0.350784	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	0.602339	-0.376955	0.870348
DIS	-0.377904	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	-0.496996	0.249929	-0.673813
RAD	0.622029	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	0.488676	-0.381626	0.361191
TAX	0.579564	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	0.543993	-0.468536	0.381395
PTRATIO	0.288250	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	0.374044	-0.507787	0.236216
LSTAT	0.452220	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	1.000000	-0.737663	0.468146
MEDV	-0.385832	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	-0.737663	1.000000	-0.289750
AGE_50	0.254574	-0.590769	0.516001	0.088659	0.597644	-0.164465	0.870348	-0.673813	0.361191	0.381395	0.236216	0.468146	-0.289750	1.000000

# 关联性分析

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')

corr = df.corr(method='pearson')
corr_with_homevalue = corr.iloc[-2]
corr_with_homevalue.sort_values(ascending=False)
```

CRIM	-0.385832
ZN	0.360445
INDUS	-0.483725
CHAS	0.175260
NOX	-0.427321
RM	0.695360
AGE	-0.376955
DIS	0.249929
RAD	-0.381626
TAX	-0.468536
PTRATIO	-0.507787
LSTAT	-0.737663
MEDV	1.000000
AGE_50	-0.289750

Name: MEDV, dtype: float64



MEDV	1.000000
RM	0.695360
ZN	0.360445
DIS	0.249929
CHAS	0.175260
AGE_50	-0.289750
AGE	-0.376955
RAD	-0.381626
CRIM	-0.385832
NOX	-0.427321
TAX	-0.468536
INDUS	-0.483725
PTRATIO	-0.507787
LSTAT	-0.737663

Name: MEDV, dtype: float64

**Number talks**  
房间多、住宅区、上班近 => 高房价

# 关联性分析 – 另类观察

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')

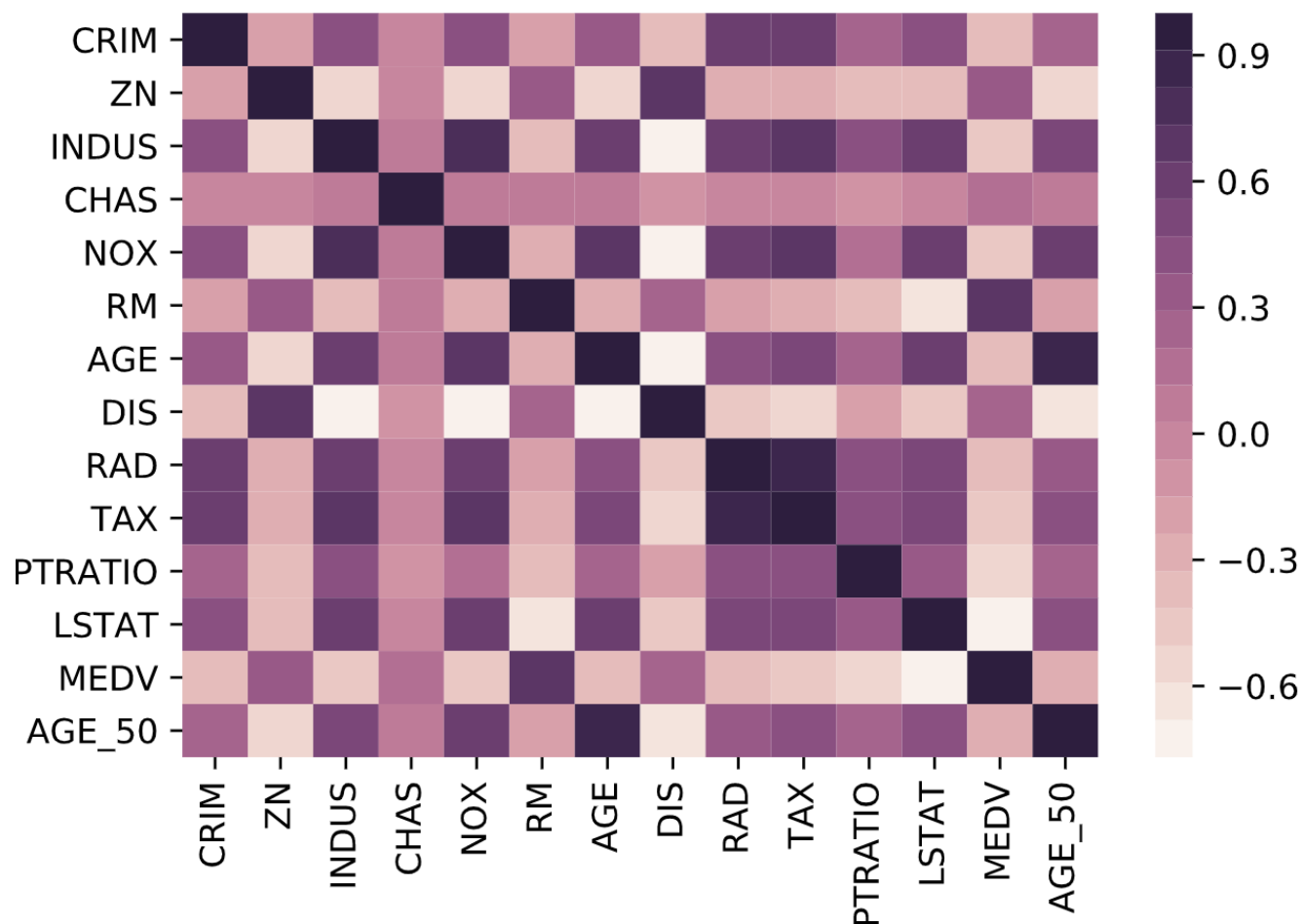
corr = df.corr(method='pearson')
corr_with_homevalue = corr.iloc[-2]
corr_with_homevalue.sort_values(ascending=False)
```

Number talks  
早期的房子邻近工业区 => 高污染

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV	AGE_50
CRIM	1.000000	-0.199458	0.404471	-0.055295	0.417521	-0.219940	0.350784	-0.377904	0.622029	0.579564	0.288250	0.452220	-0.385832	0.254574
ZN	-0.199458	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	-0.412995	0.360445	-0.590769
INDUS	0.404471	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	0.603800	-0.483725	0.516001
CHAS	-0.055295	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	-0.053929	0.175260	0.088659
NOX	0.417521	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	0.590879	-0.427321	0.597644
RM	-0.219940	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	-0.613808	0.695360	-0.164465
AGE	0.350784	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	0.602339	-0.376955	0.870348
DIS	-0.377904	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	-0.496996	0.249929	-0.673813
RAD	0.622029	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	0.488676	-0.381626	0.361191
TAX	0.579564	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	0.543993	-0.468536	0.381395
PTRATIO	0.288250	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	0.374044	-0.507787	0.236216
LSTAT	0.452220	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	1.000000	-0.737663	0.468146
MEDV	-0.385832	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	-0.737663	1.000000	-0.289750
AGE_50	0.254574	-0.590769	0.516001	0.088659	0.597644	-0.164465	0.870348	-0.673813	0.361191	0.381395	0.236216	0.468146	-0.289750	1.000000

# 关联性分析 – 绘制heatmap

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
import seaborn as sns
sns.heatmap(df.corr(), cmap=sns.cubehelix_palette(20, light=0.95, dark=0.15))
```

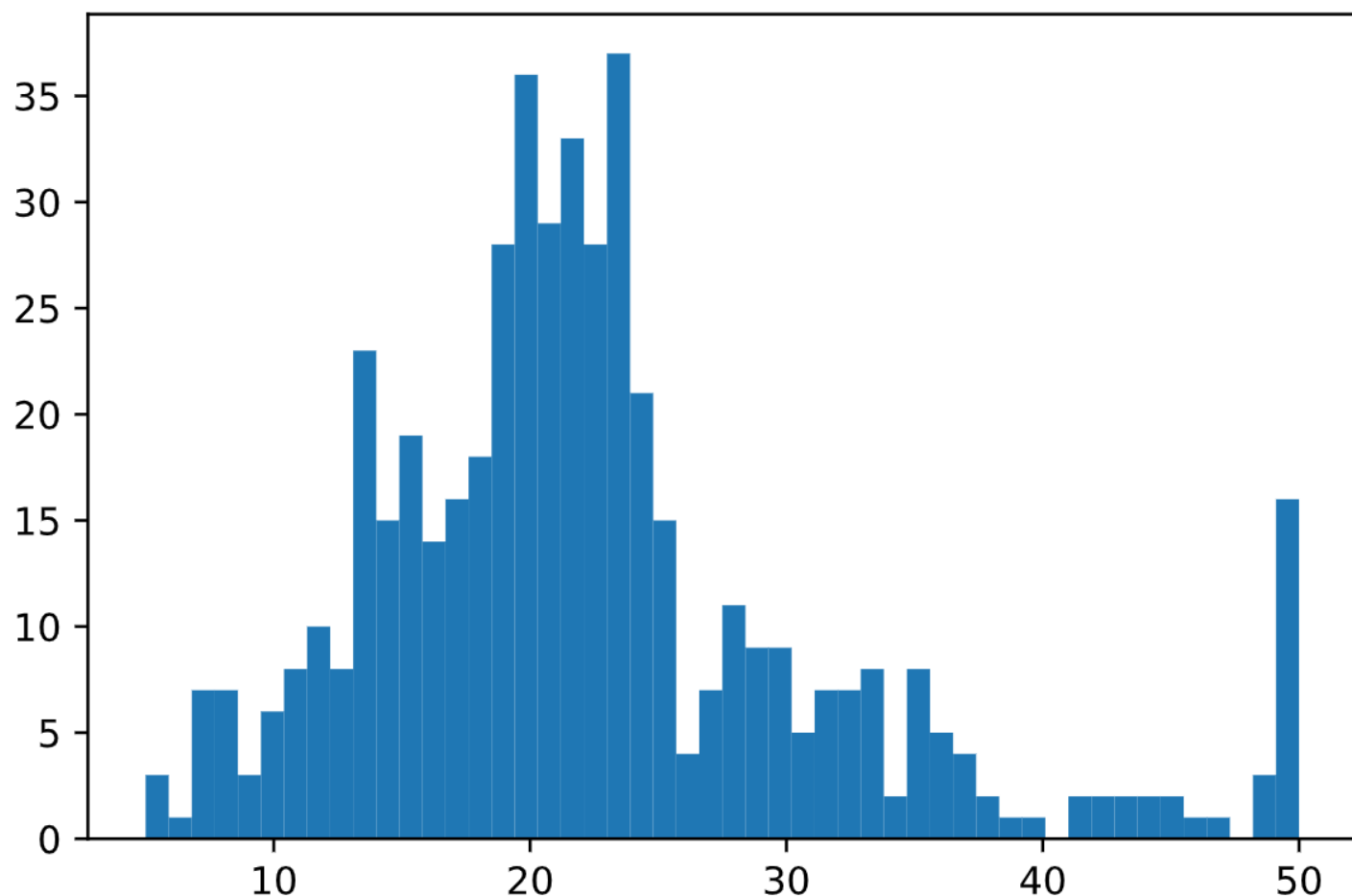


n\_colors 颜色数量  
light 最浅色色调  
dark 最深色色调

# 关联性分析 – 绘制histogram

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
import matplotlib.pyplot as plt
plt.hist(df['MEDV'], bins=50)
```

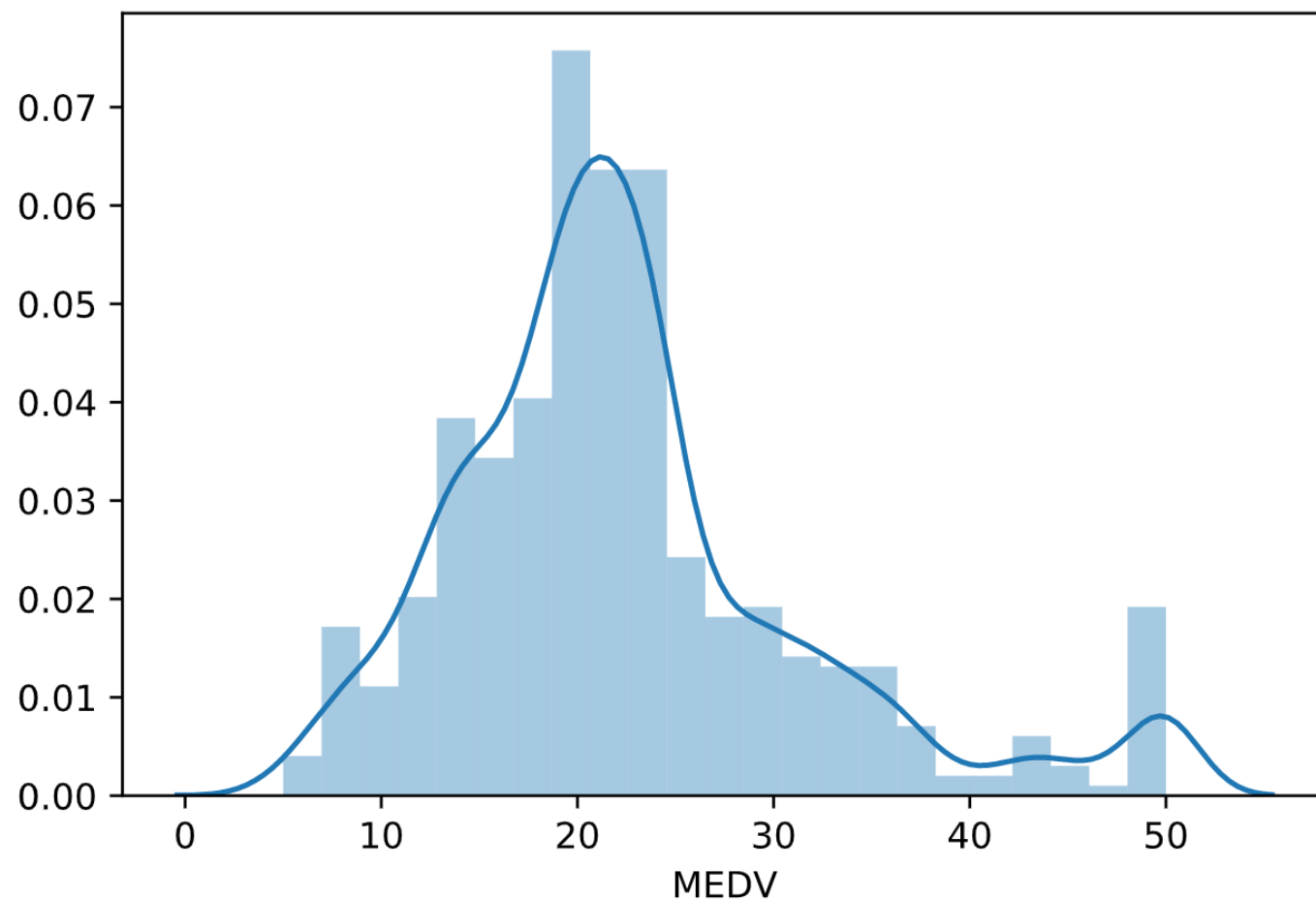
bins 条柱数量





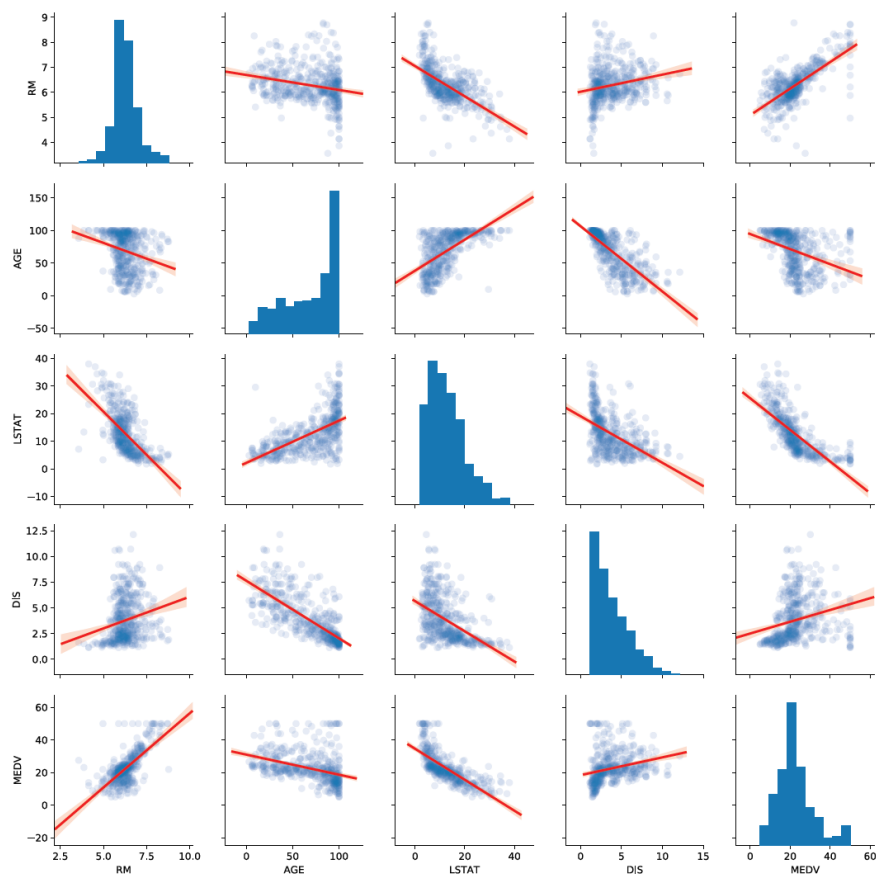
# 关联性分析 – SNS的分布图

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
import seaborn as sns
sns.distplot(df['MEDV'])
```



# 关联性分析 – SNS的大绝招

```
import numpy as np
import pandas as pd
df = pd.read_csv('housing_dataset.csv')
import seaborn as sns
sns.pairplot(df[['RM', 'AGE', 'LSTAT', 'DIS', 'MEDV']], kind='reg',
plot_kws={'line_kws': {'color': 'red'}, 'scatter_kws': {'alpha': 0.1}})
```



kind 绘图种类 scatter/kde/hist/reg  
kws 关键字参数字典

# 小结

- 了解数据集背景以及各项栏位意义，有助于寻找后续分析方向
- 简化问题，先从一个因子、两个因子慢慢增加，评估关联性
- 当单纯看数字无法理解时，试着将数据视觉化，可能会发现不同的线索





# Reactor



[developer.microsoft.com/reactor/](https://developer.microsoft.com/reactor/)  
[@MSFTReactor](https://twitter.com/MSFTReactor) on Twitter



# 议程结束 感谢聆听



请记得填写课程回馈问卷 (Event ID : **XXXXXX**)  
<https://aka.ms/Reactor/Survey>

© 2019 Microsoft Corporation. All rights reserved. The text in this document is available under the Creative Commons Attribution 3.0 License, additional terms may apply. All other content contained in this document (including, without limitation, trademarks, logos, images, etc.) are not included within the Creative Commons license grant. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. Microsoft makes no warranties, express or implied, with respect to the information provided here.