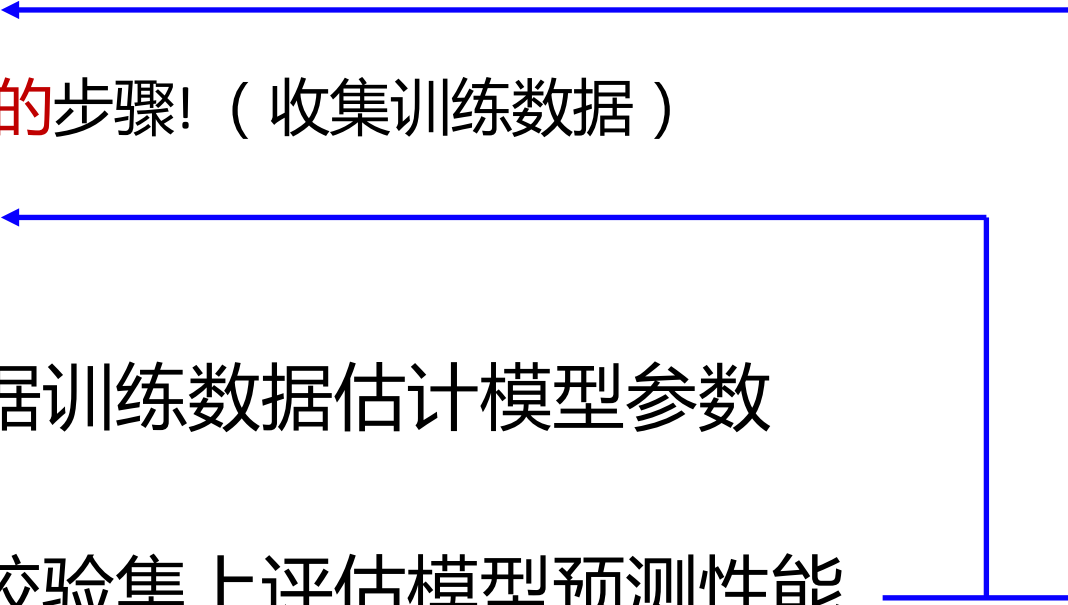


1.7 波士顿房价预测案例详解 ——数据探索

CSDN学院

► 机器学习任务的一般步骤

- 确定特征
 - 可能是**最重要的**步骤! (收集训练数据)
 - 确定模型
 - 目标函数
 - 模型训练：根据训练数据估计模型参数
 - 优化计算
 - 模型评估：在校验集上评估模型预测性能
- 
- ```
graph LR; A[确定特征] --> B[确定模型]; B --> C[模型训练]; C --> D[模型评估]; D --> A; D --> B;
```

# ► 波士顿房价预测

- 训练数据： $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ 
  - 训练样本数目 $N$ ：506个样本
  - 输入房屋属性 $\mathbf{x}$ ：13个特征（CRIM、...、LSTAT）
  - 输出房价 $y$ ：MEDV（ $y$ 为连续值，所以这是一个回归问题）

| CRIM    | ZN | INDUS | CHAS | NOX   | RM    | AGE  | DIS    | RAD | TAX | PTRATIO | B      | LSTAT | MEDV |
|---------|----|-------|------|-------|-------|------|--------|-----|-----|---------|--------|-------|------|
| 0.00632 | 18 | 2.31  | 0    | 0.538 | 6.575 | 65.2 | 4.09   | 1   | 296 | 15      | 396.9  | 4.98  | 24   |
| 0.02731 | 0  | 7.07  | 0    | 0.469 | 6.421 | 78.9 | 4.9671 | 2   | 242 | 17      | 396.9  | 9.14  | 21.6 |
| 0.02729 | 0  | 7.07  | 0    | 0.469 | 7.185 | 61.1 | 4.9671 | 2   | 242 | 17      | 392.83 | 4.03  | 34.7 |
| 0.03237 | 0  | 2.18  | 0    | 0.458 | 6.998 | 45.8 | 6.0622 | 3   | 222 | 18      | 394.63 | 2.94  | 33.4 |
| 0.06905 | 0  | 2.18  | 0    | 0.458 | 7.147 | 54.2 | 6.0622 | 3   | 222 | 18      | 396.9  | 5.33  | 36.2 |

## ► 第一步：理解任务，准备数据

- 任务描述
- 数据读取
- 数据探索
- 数据工程

# ► 特征描述

- 输入地区的属性：13个特征  $x$ 
  - CRIM：城镇人均犯罪率
  - ZN：住宅用地超过 25000 sq.ft. 的比例
  - INDUS：城镇非零售商用土地的比例
  - CHAS：是否在查理斯河边（如果边界是河流，则为1；否则为0）
  - NOX：一氧化氮浓度
  - RM：住宅平均房间数
  - AGE：1940 年之前建成的自用房屋比例
  - DIS：到波士顿五个中心区域的加权距离
  - RAD：辐射性公路的接近指数
  - TAX：每 10000 美元的全值财产税率
  - PTRATIO：城镇师生比例
  - B：1000  $(B_k - 0.63)^2$ ，其中  $B_k$  指代城镇中黑人的比例
  - LSTAT：人口中地位低下者的比例
- 输出：地区房价均值  $y$ 
  - MEDV：自住房的平均房价（单位：千美元）

- Pandas支持多种格式的数据

| Format Type | Data Description                     | Reader                         | Writer                       |
|-------------|--------------------------------------|--------------------------------|------------------------------|
| text        | <a href="#">CSV</a>                  | <a href="#">read_csv</a>       | <a href="#">to_csv</a>       |
| text        | <a href="#">JSON</a>                 | <a href="#">read_json</a>      | <a href="#">to_json</a>      |
| text        | <a href="#">HTML</a>                 | <a href="#">read_html</a>      | <a href="#">to_html</a>      |
| text        | Local clipboard                      | <a href="#">read_clipboard</a> | <a href="#">to_clipboard</a> |
| binary      | <a href="#">MS Excel</a>             | <a href="#">read_excel</a>     | <a href="#">to_excel</a>     |
| binary      | <a href="#">HDF5 Format</a>          | <a href="#">read_hdf</a>       | <a href="#">to_hdf</a>       |
| binary      | <a href="#">Feather Format</a>       | <a href="#">read_feather</a>   | <a href="#">to_feather</a>   |
| binary      | <a href="#">Msgpack</a>              | <a href="#">read_msgpack</a>   | <a href="#">to_msgpack</a>   |
| binary      | <a href="#">Stata</a>                | <a href="#">read_stata</a>     | <a href="#">to_stata</a>     |
| binary      | <a href="#">SAS</a>                  | <a href="#">read_sas</a>       |                              |
| binary      | <a href="#">Python Pickle Format</a> | <a href="#">read_pickle</a>    | <a href="#">to_pickle</a>    |
| SQL         | <a href="#">SQL</a>                  | <a href="#">read_sql</a>       | <a href="#">to_sql</a>       |
| SQL         | <a href="#">Google Big Query</a>     | <a href="#">read_gbq</a>       | <a href="#">to_gbq</a>       |

```
dpath = './data/'
data = pd.read_csv(dpath + "boston_housing.csv")
```

# ► 数据探索&特征工程

- 数据规模
- 确定数据类型，是否需要进一步编码
  - 特征编码（以后讲解）
- 数据是否有缺失值
  - 数据填补
- 查看数据分布，是否有异常数据点
  - 离群点处理
- 查看两两特征之间的关系，看数据是否有冗余 / 相关
  - 降维

- pandas : DataFrame
  - Head(): 数据前5行，可查看每一列的名字及数据类型
  - Info() :
    - 数据规模：行数&列数
    - 每列的数据类型、是否有空值
    - 占用存储量
  - shape : 行数&列数
    - 例：输入命令 data.shape , 输出(506, 14)



```
data = pd.read_csv(dpath + "boston_housing.csv")
data.head()
```

|   | CRIM    | ZN | INDUS | CHAS | NOX   | RM    | AGE  | DIS    | RAD | TAX | PTRATIO | B      | LSTAT | MEDV |
|---|---------|----|-------|------|-------|-------|------|--------|-----|-----|---------|--------|-------|------|
| 0 | 0.00632 | 18 | 2.31  | 0    | 0.538 | 6.575 | 65.2 | 4.0900 | 1   | 296 | 15      | 396.90 | 4.98  | 24.0 |
| 1 | 0.02731 | 0  | 7.07  | 0    | 0.469 | 6.421 | 78.9 | 4.9671 | 2   | 242 | 17      | 396.90 | 9.14  | 21.6 |
| 2 | 0.02729 | 0  | 7.07  | 0    | 0.469 | 7.185 | 61.1 | 4.9671 | 2   | 242 | 17      | 392.83 | 4.03  | 34.7 |
| 3 | 0.03237 | 0  | 2.18  | 0    | 0.458 | 6.998 | 45.8 | 6.0622 | 3   | 222 | 18      | 394.63 | 2.94  | 33.4 |
| 4 | 0.06905 | 0  | 2.18  | 0    | 0.458 | 7.147 | 54.2 | 6.0622 | 3   | 222 | 18      | 396.90 | 5.33  | 36.2 |

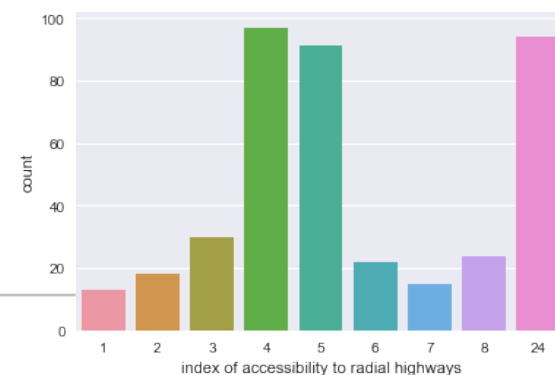
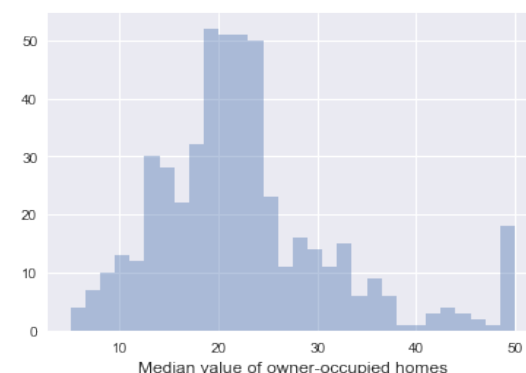
## ► 各属性的统计特性

data.describe()

|       | CRIM       | ZN         | INDUS      | CHAS       | NOX        | RM         | AGE        | DIS        | RAD        | TAX        | PTRATIO    |
|-------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| count | 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  |
| std   | 8.596783   | 23.322453  | 6.860353   | 0.253994   | 0.115878   | 0.702617   | 28.148861  | 2.105710   | 8.707259   | 168.537116 | 2.164946   |
| min   | 0.006320   | 0.000000   | 0.460000   | 0.000000   | 0.385000   | 3.561000   | 2.900000   | 1.129600   | 1.000000   | 187.000000 | 12.600000  |
| 25%   | 0.082045   | 0.000000   | 5.190000   | 0.000000   | 0.449000   | 5.885500   | 45.025000  | 2.100175   | 4.000000   | 279.000000 | 17.400000  |
| 50%   | 0.256510   | 0.000000   | 9.690000   | 0.000000   | 0.538000   | 6.208500   | 77.500000  | 3.207450   | 5.000000   | 330.000000 | 19.050000  |
| 75%   | 3.647423   | 12.500000  | 18.100000  | 0.000000   | 0.624000   | 6.623500   | 94.075000  | 5.188425   | 24.000000  | 666.000000 | 20.200000  |
| max   | 88.976200  | 100.000000 | 27.740000  | 1.000000   | 0.871000   | 8.780000   | 100.000000 | 12.126500  | 24.000000  | 711.000000 | 22.000000  |

# 直方图

- 直方图：每个取值在数据集中出现的样本数目，可视为概率函数（PDF）的估计（seaborn可视化工具比较简单）
  - import seaborn as sns
  - %matplotlib inline（seaborn 是基于matplotlib）
- 连续型特征
  - sns.distplot(data.MEDV.values, bins=30, kde=False)
- 离散型特征
  - sns.countplot(X\_train.RAD)



- 离群点：或称奇异点（outlier），指远离大多数样本的样本点。通常认为这些点是噪声，对模型有坏影响
- 可以通过直方图或散点图发现奇异点
  - 直方图的尾巴
  - 散点图上孤立的点
- 可以通过只保留某些分位数内的点去掉奇异点
  - 如0.5%-99.5%，或>99%
  - `ulimit = np.percentile(train.price.values, 99)`
  - `train['price'].ix[train['price']>ulimit] = ulimit`

# 相关性

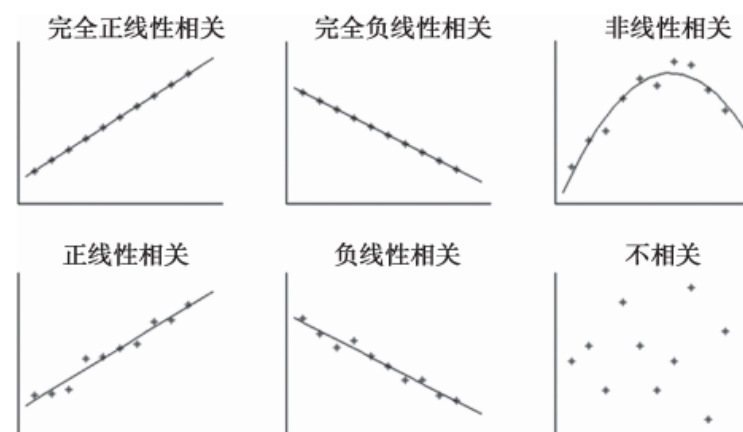
- 相关性可以通过计算相关系数或打印散点图来发现
- 相关系数：两个向量（所有样本在该特征的取值构成一个向量） $x, y$ 之间的线性相关程度

$$r = \frac{\sum_{i=0}^N (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^N (x_i - \bar{x})^2 \sum_{i=0}^N (y_i - \bar{y})^2}}$$

$$-1 \leq r \leq 1$$

通常 $|r| > 0.5$ ，认为两者相关性比较强

$$\begin{cases} r = 0 & \text{完全不线性相关} \\ r > 0 & \text{正相关} \\ r < 0 & \text{负相关} \end{cases}$$

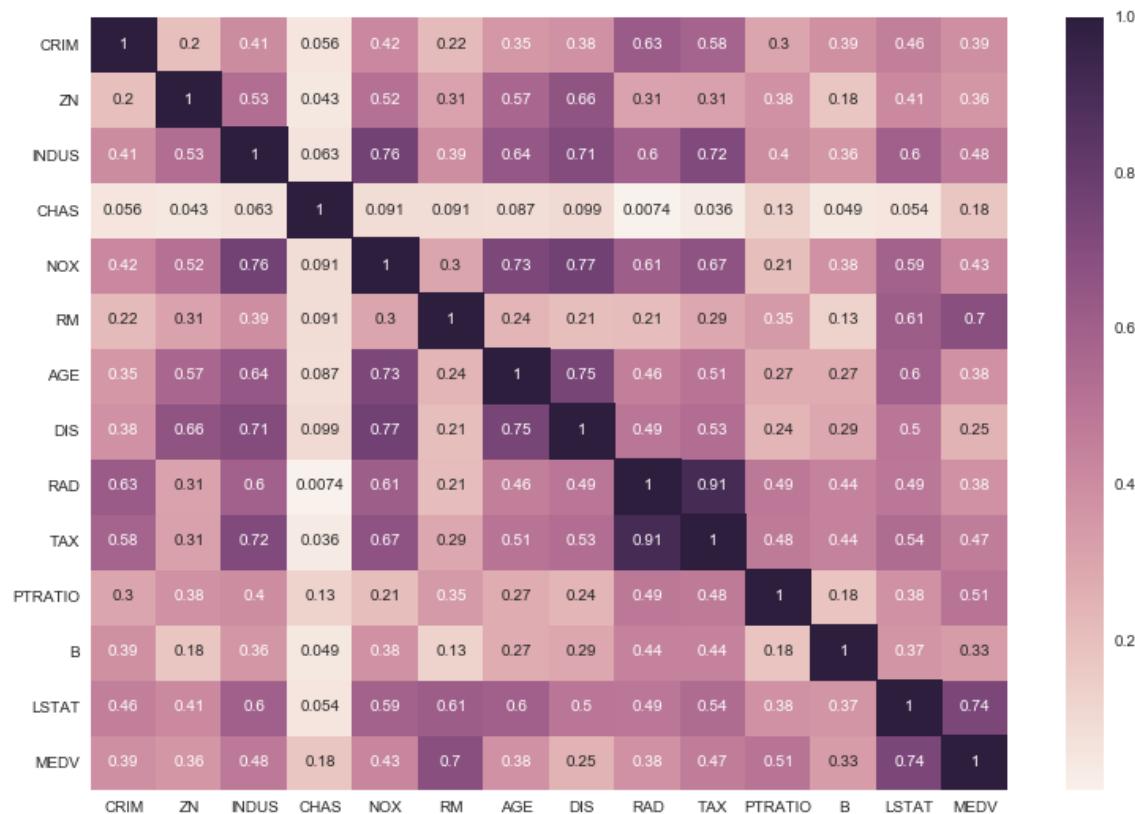


不线性相关并不代表不相关，可能高阶相关，如 $y = x^2$

## ► 相关性(cont.)

- 我们希望特征与标签强相关
- 特征与特征之间强相关的话意味着信息冗余
  - 可以两个特征可以只保留一个特征
  - 或采用主成分分析 ( PCA ) 等降维
    - 后续课程讲解

# ► Boston数据集各属性的相关系数



**RAD and TAX = 0.91**

NOX and DIS = 0.77

INDUS and NOX = 0.76

AGE and DIS = 0.75

**LSTAT and MEDV = 0.74**

NOX and AGE = 0.73

INDUS and TAX = 0.72

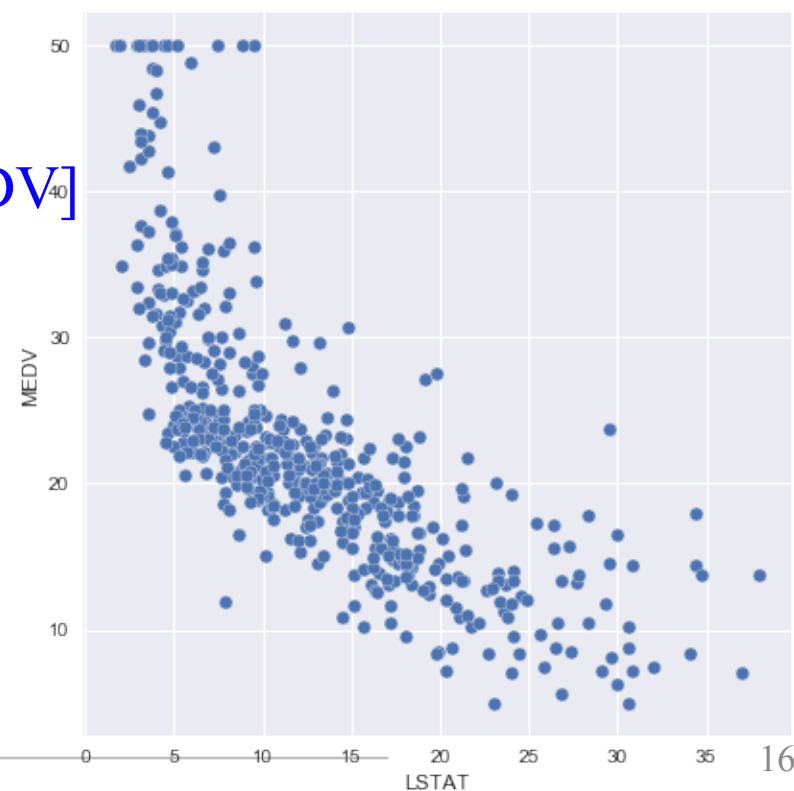
INDUS and DIS = 0.71

**RM and MEDV = 0.70**

## ► 散点图

- 可以通过两个变量之间的散点图直观感受二者的相关性

```
sns.pairplot(data, size=6, x_vars=LSTAT, y_vars=MEDV)
```





## ► 数据探索小结

- 任务类型
- 特征含义
- 单变量特征分布
  - 直方图 / 散点图
  - 离群点检测
  - 缺失值处理
- 多变量之间的关系探索
  - 相关系数
  - 散点图