CSDN
不 止 于 代 码

1.7 波士顿房价预测案例详解

CSDN学院

2017年10月

▶第一步：理解任务，准备数据

- 任务描述
- 数据读取
- 数据探索
- 特征工程

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

数据预处理

- from sklearn.preprocessing import …
  - 数据标准化（ Standardization ）
    - 某个特征的所有样本取值为0均值、1方差
  - 数据归一化（ Scaling ）
    - 某个特征的所有样本取值在规定范围内
  - 数据正规化（ Normalization ）
    - 每个样本模长为1
  - 数据二值化
    - 根据特征值取值是否大于阈值将特征值变为0或1，可用类Binarizer 实现
  - 数据缺失
  - 数据类型变换（以后讲解）
    - 有些模型只能处理数值型数据。如果给定的数据是不同的类型，必须先将数据变成数值型。

数据标准化

- 在很多模型中，假设各特征的取值区间相同。如果数据不满足该假设，需要将数据进行变换。
- 标准化是一种常用数据转换方式，将输入特征减去该特征的均值后，在除以其标准差。可用类StandardScaler实现。

```
# 数据标准化
from sklearn.preprocessing import StandardScaler
# 构造输入特征的标准化器
ss_X = StandardScaler()

# 分别对训练和测试数据的特征进行标准化处理
X_train = ss_X.fit_transform(X_train)
X_test = ss_X.transform(X_test)
```

对每维特征单独处理

$$x_i' = \frac{x_i - \mu}{\sigma}$$

$$\mu = \frac{1}{N}\sum_{i=0}^{N} x_i, \sigma = \sqrt{\frac{1}{N-1}\left(\sum_{i=0}^{N} x_i - \mu\right)^2}$$

数据归一化

- 另一种数据预处理的方式是将特征取值范围归一化到某个区间（scaling），即将样本数据取值限定在特定范围，如[0,1]，可用类MinMaxScaler实现；或 MaxAbsScaler将特征取值缩放到[-1,1]。

- 动机：
  - 对非常小的标准偏差的特征更鲁棒
  - 在稀疏数据中保留零条目

$$x_i' = \frac{x_i - \min}{\max},$$

$$where \quad \min = \min\{x_1, ..., x_N\}$$

$$\max = \max\{x_1, ..., x_N\}$$

数据正规化

- 数据正规化（ normalization ），将每个样本的模的长度变为单位长度1。可用类Normalizer实现。

$$\mathbf{x}'_i = \mathbf{x}_i / \|\mathbf{x}_i\|_2 = \mathbf{x}_i \Big/ \sqrt{\sum_{j=1}^{D} x_{ij}^2},$$

- 在求欧式距离（相似度度量指标）时就很必要

缺失值填补

- 由于各种原因，实际应用中总是存在一些缺失值，通常表示为NaN。
- scikit-learn的类Imputer提供一些常见填补方法
  - 均值mean（默认方法）
  - 中位数median
  - 众数most_frequent
- pandas库的fillna函数也可以处理缺失值，而且更加灵活，但是重用性较弱

▶第二步：模型确定和模型训练

- 1. 确定模型类型
  - 目标函数（损失函数、正则）
- 2、模型训练
  - 优化算法（解析法、梯度下降、随机梯度下降…）

线性回归模型

- 学习从输入$\mathbf{x}$到输出y的映射f：$\hat{y} = f(\mathbf{x})$
  - 输入$\mathbf{x}$：13维的房屋属性
  - 输出y：房屋价格
  - 回归器 / 模型（scikit learn中的estimator）
    - 若采用线性回归模型，则 $y = f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\mathbf{x}$

- 尝试三种不同的线性回归模型：OLS、岭回归、Lasso

Scikit learn 中的线性回归模型

sklearn.linear_model.LinearRegression(*fit_intercept=True, normalize=False, copy_X=True, n_jobs=1*)

```
from sklearn.linear_model import LinearRegression

# 使用默认配置初始化
lr = LinearRegression()

# 训练模型参数
lr.fit(X_train, y_train)

# 预测
lr_y_predict = lr.predict(X_test)
```

Scikit learn 中estimator使用三部曲：

1. 构造estimator （可设置参数）
2. 训练模型：fit
3. 利用模型进行预测： predict

随机梯度下降线性回归模型

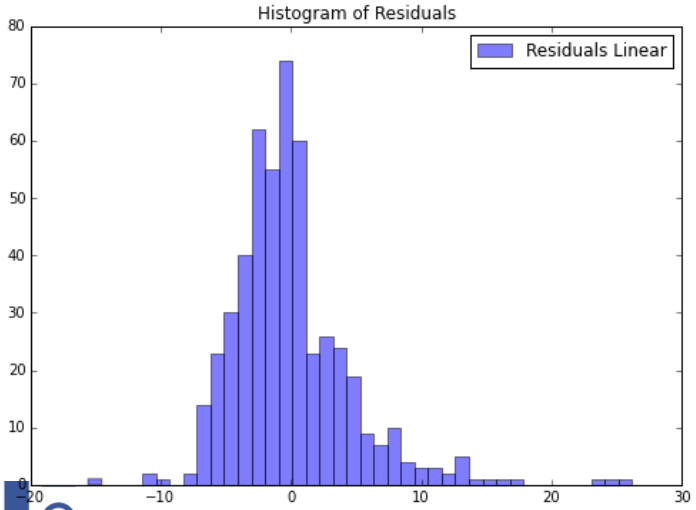

```
# 线性模型，随机梯度下降优化模型参数
from sklearn.linear_model import SGDRegressor

# 使用默认配置初始化线
sgdr = SGDRegressor()

# 训练：参数估计
sgdr.fit(X_train, y_train)

# 预测
sgdr_y_predict = sgdr.predict(X_test)
```

第四步：模型评估与模型选择

- 模型训练好后，需要在校验集上采用一些度量准则检查模型预测的效果
  - 校验集划分（train_test_split、交叉验证）
  - 评价指标（sklearn.metrics）
  - 也可以检查残差的分布
  - 还可以打印预测值与真值的散点图
- 模型选择：选择预测性能最好的模型
  - 模型中通常有一些超参数，需要通过模型选择来确定
  - 参数搜索范围：网格搜索（GridSearch）

预测残差分布


```
%matplotlib inline
from matplotlib import pyplot as plt
f, ax = plt.subplots(figsize=(7, 5))
f.tight_layout()
ax.hist(boston.target - predictions,bins=40, label='Residuals Linear', color='b', alpha=.5);
ax.set_title("Histogram of Residuals")
ax.legend(loc='best');
```



回忆：极大似然估计假设残差的分布围0均值的正态分布

残差近似0均值的正态分布，看起来拟合得不错☺，只是有点偏

预测值与真值散点图

```
plt.figure(figsize=(4, 3))
plt.scatter(y_train, lr_y_predict_train)
plt.plot([0, 3], [0, 3], '--k')  #数据已经标准化，3倍标准差即可
plt.axis('tight')
plt.xlabel('True price ($1000s)')
plt.ylabel('Predicted price ($1000s)')
plt.tight_layout()
```

二者之间的相关性越高越好

线性回归中的模型选择

sklearn.model_selection

- Scikit learn中的model selection模块提供模型选择功能
  - 对于线性模型，留一交叉验证（*N*折交叉验证，亦称为leave-one-out cross-validation，LOOCV）有更简便的计算方式，因此Scikit learn提供了RidgeCV类和LassoCV类
  - 后续课程将讲述一般模型的交叉验证和参数调优GridSearchCV

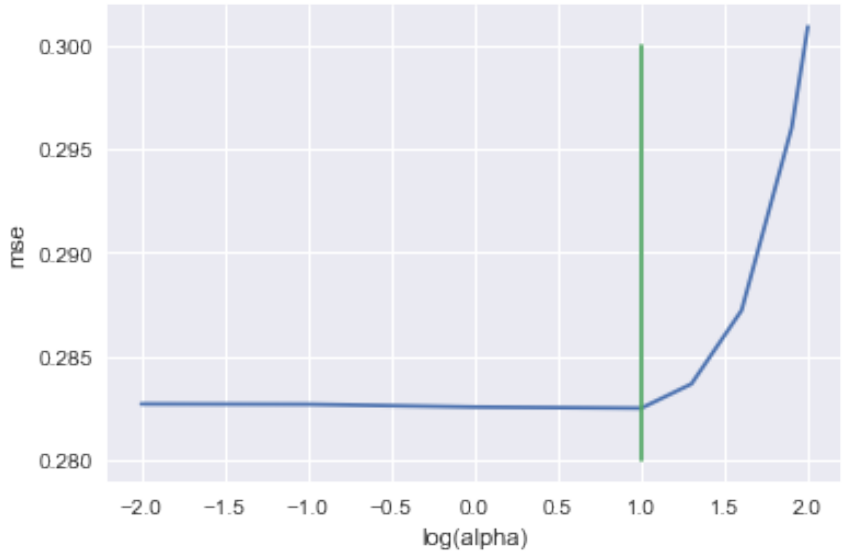
RidgeCV

- RidgeCV中超参数λ用alpha表示
- RidgeCV(*alphas=(0.1, 1.0, 10.0), fit_intercept=True, normalize=False, scoring=None, cv=None, gcv_mode=None, store_cv_values=False)*

```
from sklearn.linear_model import  RidgeCV

alphas = [0.01, 0.1, 1, 10,20, 30, 50, 60, 80,100]

reg = RidgeCV(alphas=alphas, store_cv_values=True)
reg.fit(X_train, y_train)
```

模型选择

- 可以通过在交叉验证误差曲线上找最佳值找到最佳模型



LassoCV

- LassoCV的使用与RidgeCV类似

- Scikit learn还提供一个与Lasso类似的LARS（least angle regression，最小角回归），二者仅仅是优化方法不同，目标函数相同。
- 当数据集中特征维数很多且存在共线性时，LassoCV更合适。

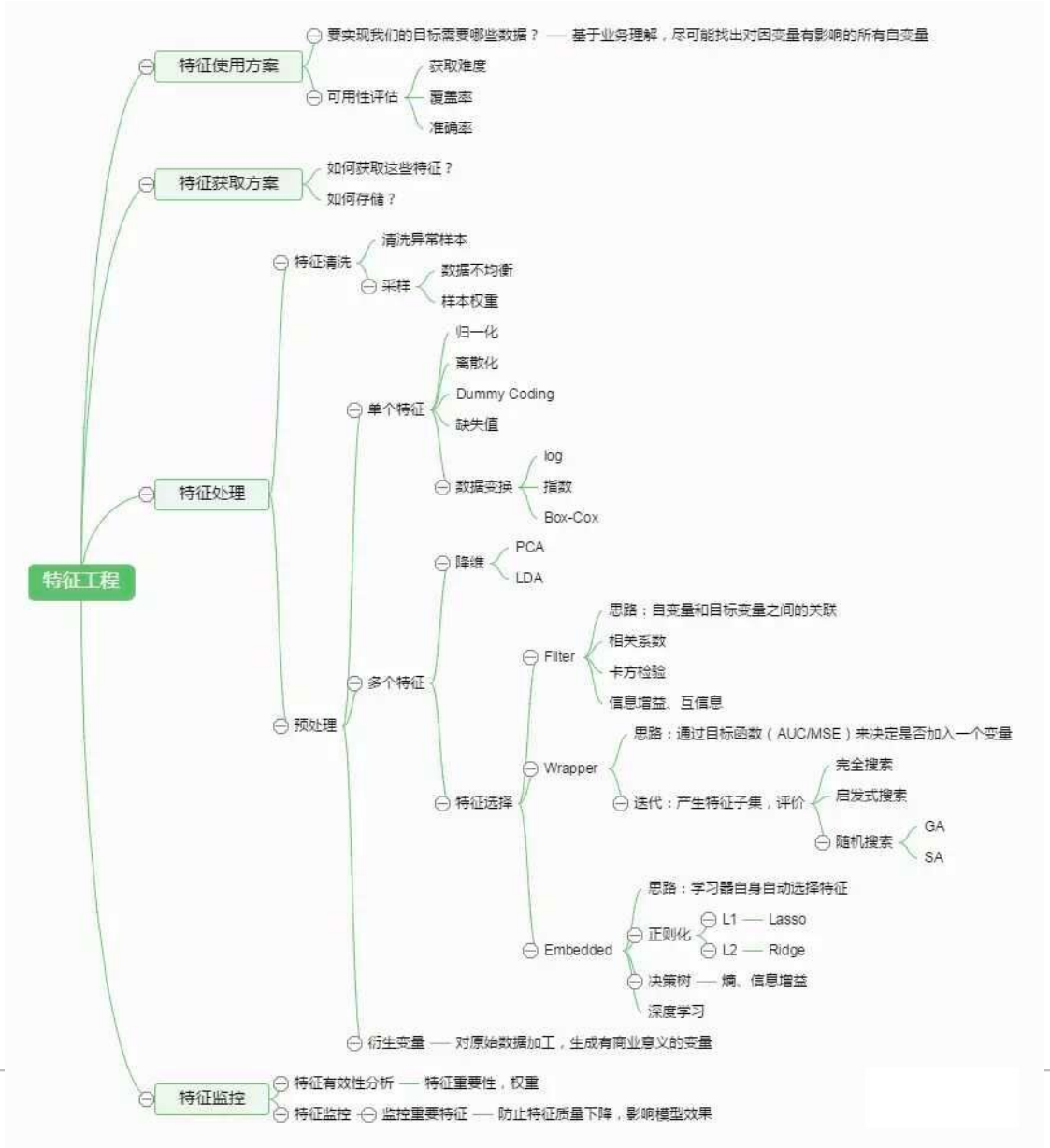
各模型比较

	线性回归	岭回归（alpha=10）	LASSO（alpha=0.01）
CRIM	-0.10643777	-0.09781781	-0.07849129
ZN	0.13238196	0.11357219	0.09347763
INDUS	0.0252063	-0.00069185	0
CHAS	0.08244512	0.08574201	0.0807981
NOX	-0.17705123	-0.14911378	-0.12532951
RM	0.30530892	0.31400952	0.31939438
AGE	-0.00429841	-0.00946359	0
DIS	-0.33726245	-0.30439602	-0.26700207
RAD	0.2942942	0.22375134	0.15577565
TAX	-0.24568977	-0.17767457	-0.12534733
PTRAIO	-0.18931585	-0.18290525	-0.17643178
B	0.08015874	0.07997069	0.07117083
LATAT	-0.43340828	-0.41743638	-0.43518032

CSDN
不 止 于 代 码

THANK YOU

AI100


特征工程
特征使用方案
要实现我们的目标需要哪些数据？ — 基于业务理解，尽可能找出对因变量有影响的所有自变量
可用性评估
获取难度
覆盖率
准确率
特征获取方案
如何获取这些特征？
如何存储？
特征处理
特征清洗
清洗异常样本
采样
数据不均衡
样本权重
预处理
单个特征
归一化
离散化
Dummy Coding
缺失值
数据变换
log
指数
Box-Cox
多个特征
降维
PCA
LDA
特征选择
Filter
思路：自变量和目标变量之间的关联
相关系数
卡方检验
信息增益、互信息
Wrapper
思路：通过目标函数（AUC/MSE）来决定是否加入一个变量
迭代：产生特征子集，评价
完全搜索
启发式搜索
随机搜索
GA
SA
Embedded
思路：学习器自身自动选择特征
正则化
L1 — Lasso
L2 — Ridge
决策树 — 熵、信息增益
深度学习
衍生变量 — 对原始数据加工，生成有商业意义的变量
特征监控
特征有效性分析 — 特征重要性，权重
特征监控 — 监控重要特征 — 防止特征质量下降，影响模型效果