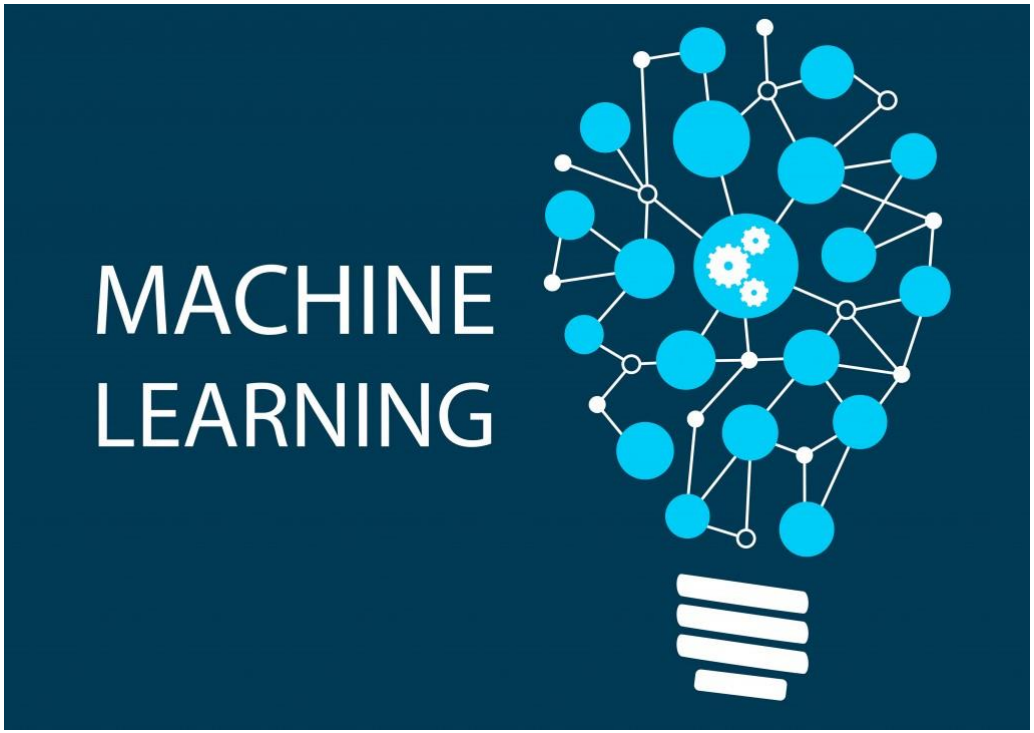
法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 **小象学院**

第五讲



机器学习（3）

--Robin

目录

- 特征工程
- 模型调优
- 模型评价指标
- 集成学习
- 实战案例4：动物种类识别

目录

特征工程

- 数值型特征，如：长度、宽度、像素值等
  - 可直接使用
  - 但是，对于有些模型来说，数值范围归一化（feature normalization）可以提高模型的性能，如：线性回归，kNN，SVM，神经网络等
  - sklearn.preprocessing.MinMaxScaler()　$x_i' = (x_i - x_i^{MIN})/(x_i^{MAX} - x_i^{MIN})$
- 有序型特征，如：等级（A，B，C）；级别（低、中、高）
  - 转换成有序数值即可，如A->1, B->2, C->3
- 类别型特征，如：性别（男、女）
  - 独热编码（One-Hot Encoding），如男->0 1, 女->1 0
  - sklearn.preprocessing.OneHotEncoder()

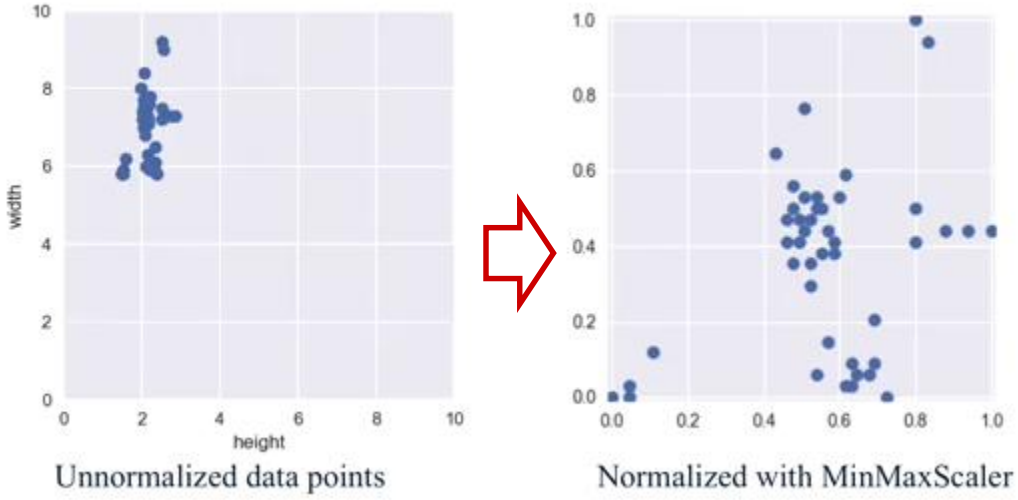
lect05_eg01.ipynb

ID	Gender
1	Male
2	Female
3	Not Specified
4	Not Specified
5	Female

ID	Male	Female	Not Specified
1	1	0	0
2	0	1	0
3	0	0	1
4	0	0	1
5	0	1	0

特征工程

- 特征范围归一化（feature normalization）



- 注意，在测试集上的scaler和训练集上的scaler要保持一致；不要在训练集和测试集分别使用不同的scaler
- 同理，对于One-Hot Encoding，也是一样，要保证测试集和训练集的encoder一致

```
lect05_eg01.ipynb
```

目录

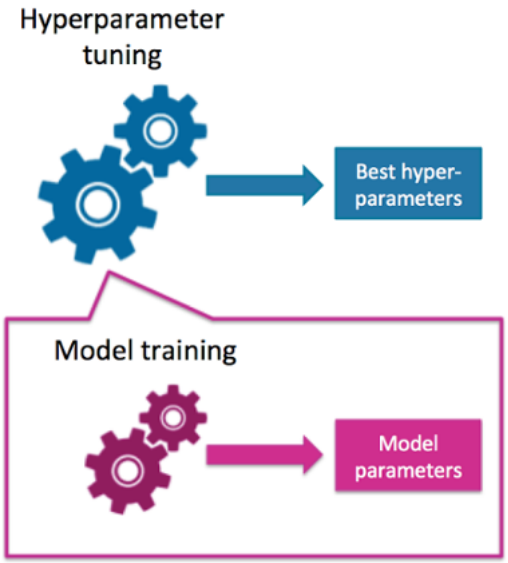
模型调优

参数调整

- 模型参数包括两种
  1. 模型自身参数，通过样本学习得到的参数。如：逻辑回归及神经网络中的权重及偏置的学习等
  2. 超参数，模型框架的参数，如kmeans中的k，神经网络中的网络层数及每层的节点个数。通常由手工设定
- 如何调整参数
  1. 交叉验证

     sklearn.model_selection.cross_val_score()
  2. 网格搜索(Grid Search)

     sklearn.model_selection.GridSearchCV()

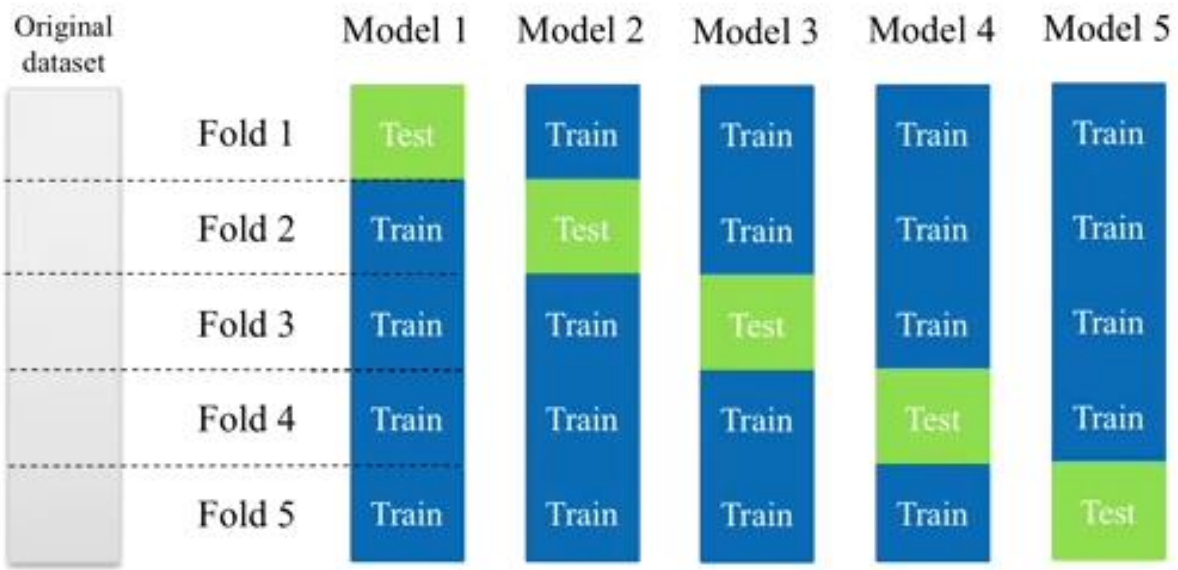


lect05_eg02.ipynb

模型调优

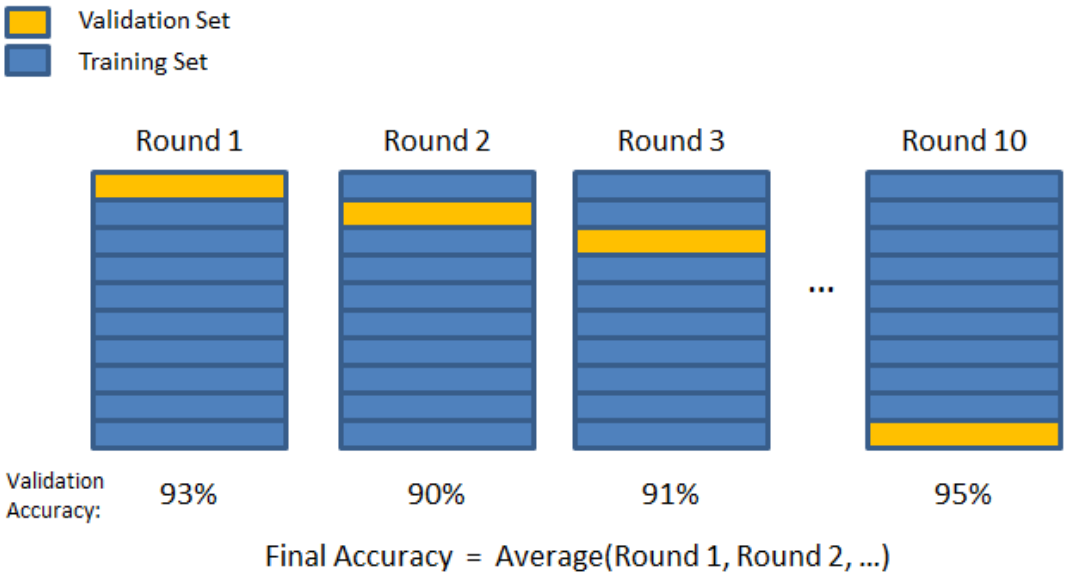
调整参数

- 依靠经验
- 依靠实验, 交叉验证 (cross validation)
- 例子：5折交叉验证



模型调优

例子：10折交叉验证



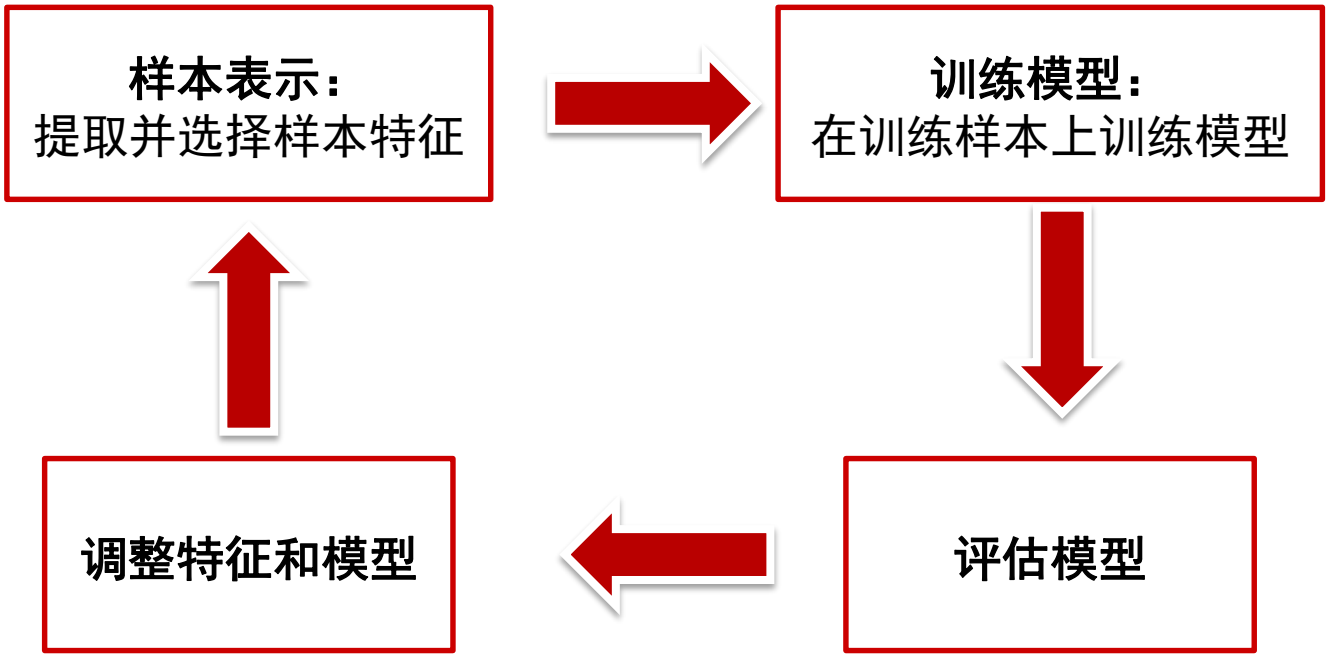
模型持久化（model persistence）

1. pickle
2. joblib

目录

模型评价指标及模型选择

建模过程



模型评价指标及模型选择

评估模型

- 不同的应用有着不同的目标，不同的评价指标
- **准确率（accuracy）**是最常见的一种
- 但是准确率越高，模型就越好么？
- 假设，在1000个样本中，有999个正样本，1个负样本（不均衡数据集）

  如果全部预测正样本，就可以得到准确率99.9%！这样的场景有：信用卡欺诈检测，离职员工检测等。
- 有些任务更关心的是某个类的准确率，而非整体的准确率

模型评价指标及模型选择

- 真正例(TP)，预测值是1，真实值是1。被正确分类的正例样本。
- 假正例(FP)，预测值是1，但真实值是0
- 真反例(TN)，预测值是0，真实值是0
- 假反例(FN)，预测值是0，但真实值是1。
- **TPR(Recall，召回率)**：TP/(TP + FN)，表示检测率
- **Precision**(精确率)： TP/(TP + FP)
- **FPR**: FP/(TN+FP)，在所有实际值是0的样本中，被错误地预测为1的比例。

lect05_eg03.ipynb

		Prediction	
		Positive	Negative
Ground truth	Positive	**True positive (TP)**	**False negative (FN)**
	Negative	**False positive (FP)**	**True negative (TN)**

模型评价指标及模型选择

- **F1值**
  - 将召回率和精确率用一个数值表示

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\mathrm{recall}} + \frac{1}{\mathrm{precision}}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

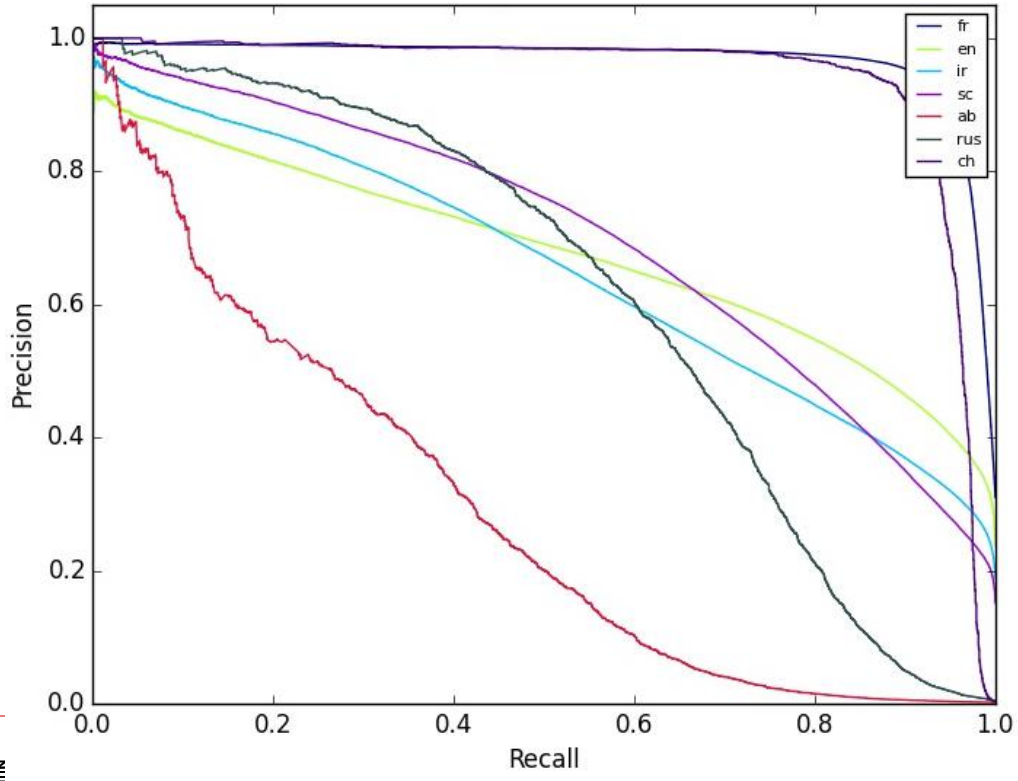
  - sklearn.metrics中包含常用的评价指标
    - accuracy_score
    - precision_score
    - recall_score
    - f1_score

lect05_eg03.ipynb

模型评价指标及模型选择

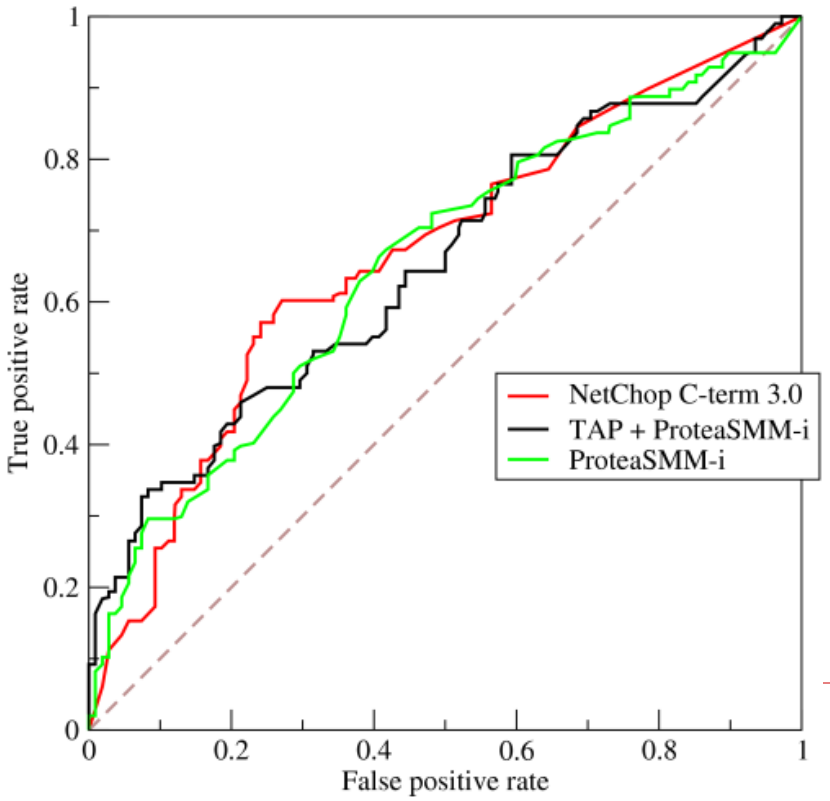
- **Precision-Recall Curve (PR曲线)**
- x轴：recall, y轴：precision（可交换）
- 右上角是“最理想”的点，precision=1.0, recall=1.0
- sklearn.metrics.precision_recall_curve()

lect05_eg03.ipynb



模型评价指标及模型选择

- **Receiver Operating Characteristic Curve (ROC曲线)**
- x轴：FPR, y轴：TPR
- 左上角是“最理想”的点，FPR=0.0, TPR=1.0
- sklearn.metrics.roc_curve()



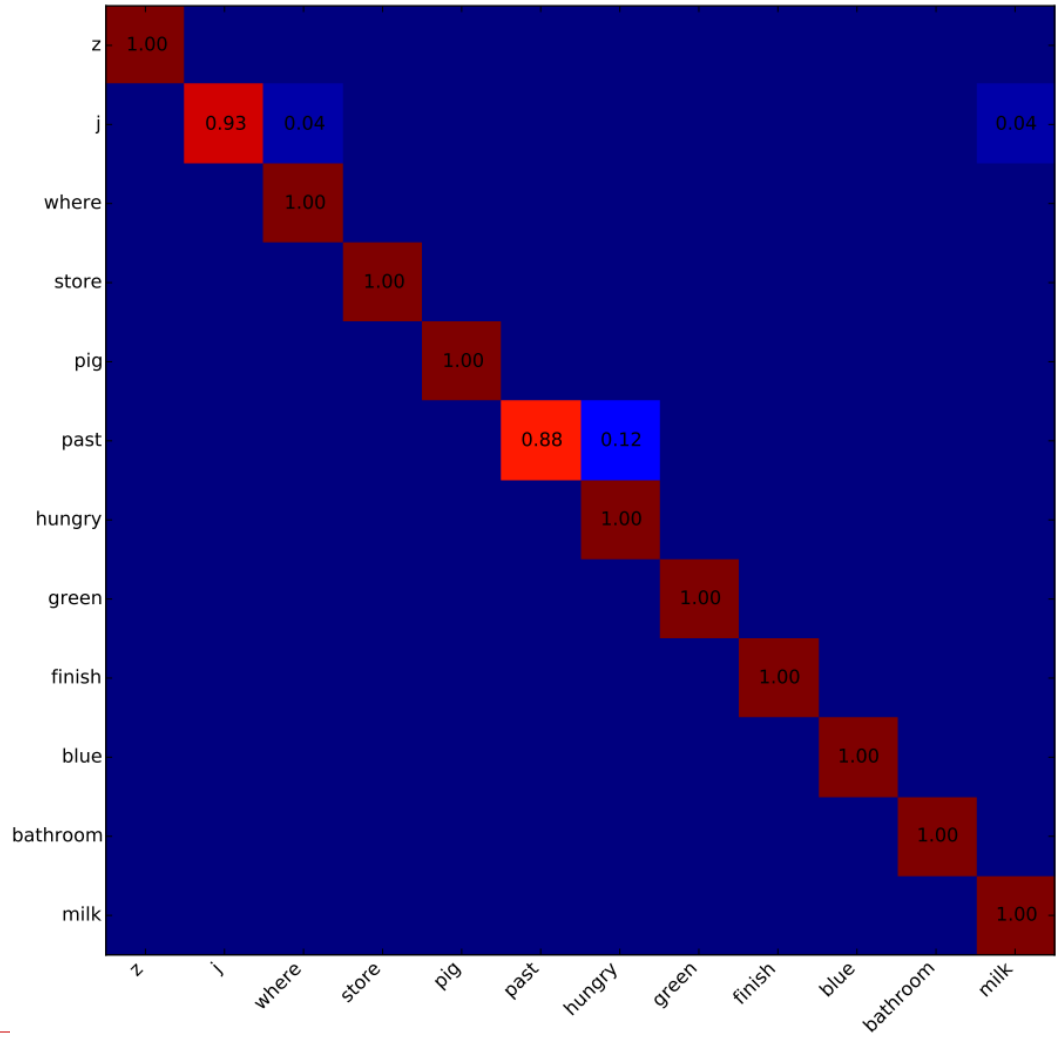
AUC的值就是ROC曲线下的面积

- AUC在0~1之间
- 0.5 < AUC < 1，优于随机猜测。这个分类器（模型）妥善设定阈值的话，能有预测价值。
- AUC = 0.5，跟随机猜测一样（例：丢铜板），模型没有预测价值。
- AUC < 0.5，比随机猜测还差；但只要总是反预测而行，就优于随机猜测。

lect05_eg03.ipynb

模型评价指标及模型选择

- 混淆矩阵（**confusion matrix**）
- 可用于**多分类**模型的评价
- sklearn.metrics.confusion_matrix()




```
lect05_eg03.ipynb
```

模型评价指标及模型选择

- 回归模型中常用的评价指标
- sklearn.metrics.r2_score()
- sklearn.metrics.mean_absolute_error()
- sklearn.metrics.mean_squared_error()
- sklearn.metrics.median_absolute_error()

- 更多模型评价指标请参考：

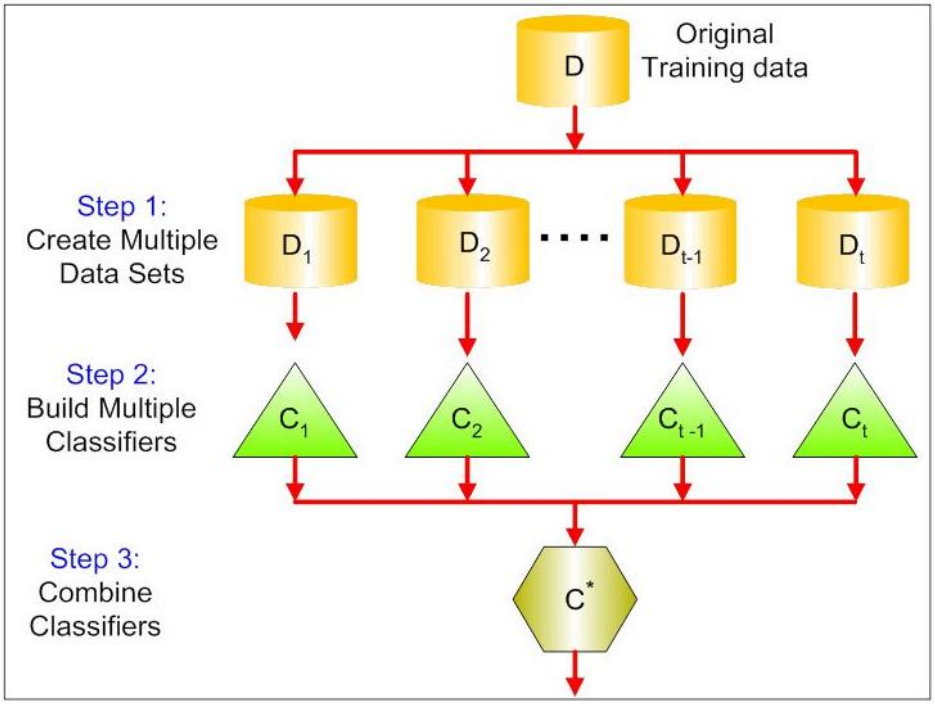
  http://scikit-learn.org/stable/modules/model_evaluation.html

目录

集成学习

- Ensemble learning
  - 通过构建并结合多个学习器来完成学习任务

- “同质”(homogeneous)集成：集成中包含同种类型的学习器->“基学习器”(base leaner)
- “异质”(heterogeneous)集成：集成中包含不同类型的学习器->“组件学习器”(component leaner)



集成学习

- 集成策略
  - 简单平均法

$$H(\boldsymbol{x}) = \frac{1}{T}\sum_{i=1}^{T} h_i(\boldsymbol{x})$$

  - 加权平均法

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}) \qquad w_i \geqslant 0, \sum_{i=1}^{T} w_i = 1$$

  - 绝对数投票法，某标记投票过半，则预测为该标记
  - 相对数投票法，预测为投票最多的标记

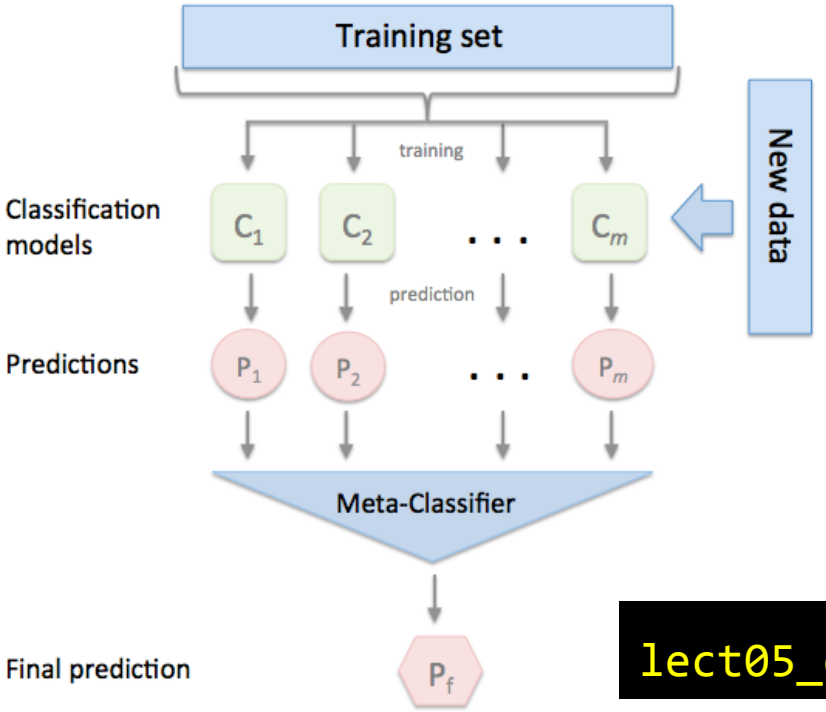
$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} h_i^j(\boldsymbol{x})}$$

  - 加权投票法

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} w_i h_i^j(\boldsymbol{x})}$$

集成学习

- 集成策略
  - stacking
  - pip install mlxtend
  - conda install mlxtend



lect05_eg03.ipynb

Algorithm 19.7 Stacking

Input: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^m$ ($\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathcal{Y}$)
Output: An ensemble classifier H

1: Step 1: Learn first-level classifiers
2: **for** $t \leftarrow 1$ to T **do**
3: Learn a base classifier h_t based on $\mathcal{D}$
4: **end for**
5: Step 2: Construct new data sets from $\mathcal{D}$
6: **for** $i \leftarrow 1$ to m **do**
7: Construct a new data set that contains $\{\mathbf{x}'_i, y_i\}$, where $\mathbf{x}'_i = \{h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \ldots, h_T(\mathbf{x}_i)\}$
8: **end for**
9: Step 3: Learn a second-level classifier
10: Learn a new classifier h' based on the newly constructed data set
11: **return** $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x}))$

集成学习

- Ensemble learning
  - 好的集成，个体学习器应“好而不同”：个体学习器要有一定的“准确性”，并且还要有“多样性”

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	×	√	√
h_3	√	×	√
集成	√	√	√

(a) 集成提升性能

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
集成	√	√	×

(b) 集成不起作用

	测试例1	测试例2	测试例3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
集成	×	×	×

(c) 集成起负作用

  - 个体学习器间不存在强依赖关系、可同时生成的并行化方法（Bagging）
  - 个体学习器间存在强依赖关系、串行生成的序列化方法（Boosting）

集成学习

- Boosting
  1. 从初始训练集训练出一个基学习器
  2. 根据基学习器的表现对训练样本分布进行调整，使得之前基学习器做错的训练样本在之后得到更多关注
  3. 基于调整后的样本分布训练下一个基学习器
  4. 直至基学习器数目达到预设值*T*，最终将得到的*T*个基学习器进行加权结合
- 代表算法
  - Adaboost，“加性模型”(additive model)，基学习器的线型组合

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$$

lect05_eg03.ipynb

集成学习

- Adaboost算法

Input :

- A training set $S = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m))$.

Initialization :

- Maximum number of iterations T;
- initialize the weight distribution $\forall i \in \{1, \ldots, m\}, D^{(1)}(i) = \frac{1}{m}$.

for $t = 1, \ldots, T$ **do**

- Learn a classifier $f_t : \mathbb{R}^d \to \{-1, +1\}$ using distribution $D^{(t)}$
- Set $\epsilon_t = \sum_{i: f_t(\mathbf{x}_i) \neq y_i} D^{(t)}(i)$
- Choose $a_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
- Update the weight distribution over examples

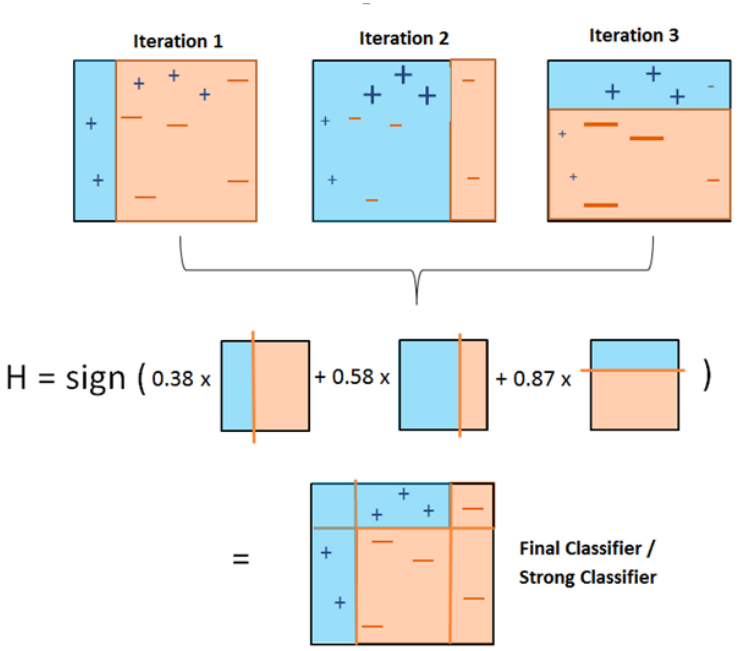
$$\forall i \in \{1, \ldots, m\}, D^{(t+1)}(i) = \frac{D^{(t)}(i) e^{-a_t y_i f_t(\mathbf{x}_i)}}{Z^{(t)}}$$

where $Z^{(t)} = \sum_{i=1}^{m} D^{(t)}(i) e^{-a_t y_i f_t(\mathbf{x}_i)}$ is a normalization factor such that $D^{(t+1)}$ remains a distribution.

Output : The voted classifier $\forall \mathbf{x}, F(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} a_t f_t(\mathbf{x})\right)$

集成学习

- Adaboost算法



- Adaboost的正则化:

$$f_k(x) = f_{k-1}(x) + \alpha_k G_k(x) \Rightarrow f_k(x) = f_{k-1}(x) + \nu\alpha_k G_k(x)$$

  - 取值范围0-1
  - 较小的值意味着更多的弱学习器的迭代次数

集成学习

- Adaboost例子:

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

以第一次迭代为例：

1.1 个体学习器及样本权重

$$G_1(x) = \begin{cases} 1 & x < 2.5, \\ -1 & x > 2.5. \end{cases}$$

x	0	1	2	3	4	5	6	7	8	9
w	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
y	1	1	1	-1	-1	-1	1	1	1	-1

1.2 分类错误率 e_1 =0.1+0.1+0.1=0.3

1.3 计算学习器系数，根据 $\alpha_i = \frac{1}{2} log \frac{1-e_i}{e_i}$ 可得 α_1 =0.4236

x	0	1	2	3	4	5	6	7	8	9
w	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16667	0.16667	0.16667	0.07143
y	1	1	1	-1	-1	-1	1	1	1	-1

$$f_1(x) = \alpha_1 G_1(x) = 0.4236 G_1(x) \quad 0.4236 G_1(x) = \begin{cases} 0.4236 * 1 & x < 2.5, \\ 0.4236 * (-1) & x > 2.5. \end{cases}$$

2. …

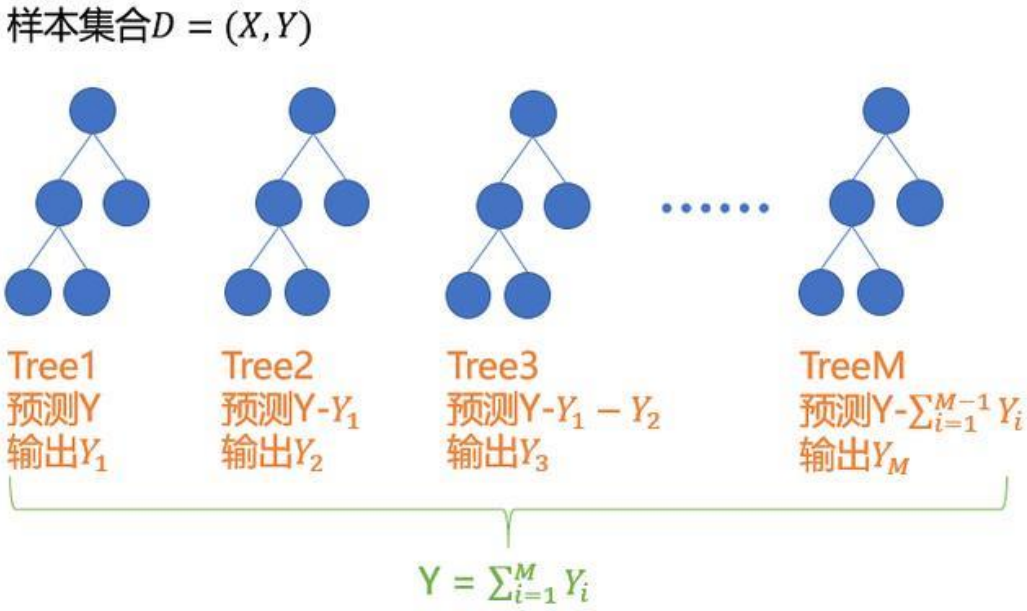
集成学习

- Gradient Boosted (-ing) Decision Tree
- 传统Boosting:

  从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多关注，然后基于调整后的样本分布来训练下一个基学习器；直至基学习器数目达到预设的*T*值，最终将这*T*个基学习器进行加权结合。
- Gradient Boost是一个框架，可以套入不同的模型，与传统Boosting的区别是，每一次的计算是为了减少上一次的残差，而为了消除残差，可以在残差减少的梯度方向上建立新的模型。

集成学习

- Gradient Boosted (-ing) Decision Tree
- 构造一系列决策树，每棵树尝试去纠正之前的错误
- 每棵树学习的是之前所有树的结论和残差，拟合得到一个当前的树。整个迭代过程生成的树的累加就是GBDT。
- 学习率决定新的树去纠正错误的程度
  - 高学习率：生成复杂的树
  - 低学习率：生成简单的树



集成学习

- GBDT
- 关键参数
  1. n_estimators: 包含的决策树（弱分类器）的个数
  2. learning_rate: 学习率，控制从上一次迭代中纠错的强度
  3. max_depth: 通常不会设的过大，比如在大多数应用中设置为3-5

优点	缺点
• 在多数任务中都能取得不错的效果 • 不需要过多的使用特征归一化和参数调整 • 可以使用不同类型的特征	• 结果不容易解释 • 需要注意对学习率的调参 • 训练会需要大量的计算 • 由于计算量的问题，不太适用于文本分类或者其他高维稀疏的特征

集成学习

- Bagging，基于自助采样法（bootsrap sampling）

  给定包含m个样本的数据集，先随机取出一个样本放入采样集中，再把该样本放回初始数据集，使得下次采样时该样本仍有可能被选中，这样经过m次随机采样操作，得到包含m个样本的采样集，初始训练集中有的样本在采样集中多次出现，有的则未出现。（初始训练集中约有63.2%的样本出现在样本集中。）

  样本在m次采样中始终不被采到的概率是$\left(1-\frac{1}{m}\right)^m$，取极限$\lim_{m\mapsto\infty}\left(1-\frac{1}{m}\right)^m \mapsto \frac{1}{e}\approx 0.368$

- 基于此，采样出T个含m个训练样本的采样集，然后基于每个采样集训练出一个基学习器。

集成学习

- 随机森林在构建Bagging采样的基础上，进一步在决策树的训练过程中引入随机属性选择。对基决策树的每个结点，先从该结点的属性集合（d个属性）中随机选择一个包含k个属性的子集，然后再从子集中选择最优属性进行划分。
  - k=d，基决策树的构建与传统决策树相同
  - k=1，随机选择一个属性用于划分；通常 $k = \log_2 d$
- k对应于sklearn中的参数max_features
- max_features会影响随机森林的学习过程
- max_features=1，构造出的是完全不同的森林，其中的树较复杂
- max_features=接近特征的个数，构造出相似的森林，其中的树较简单

集成学习

- 随机森林
- 关键参数
  1. n_estimators: 包含决策树的个数
  2. max_features: 对于模型效果有很大影响。森林中决策树的差异性
     - 通常选择sklearn中默认的即可，个别情况需要人为调整
  3. max_depth: 每棵决策树的深度
  4. random_state: 如果需要复现实验结果，需要设置相同的random_state

优点	缺点
• 应用广泛，在多数任务中都能取得不错的效果 • 不需要过多的使用特征归一化和参数调整 • 可以使用不同类型的特征	• 结果不容易解释 • 对于高维数据，可能没有线性模型快速很准确

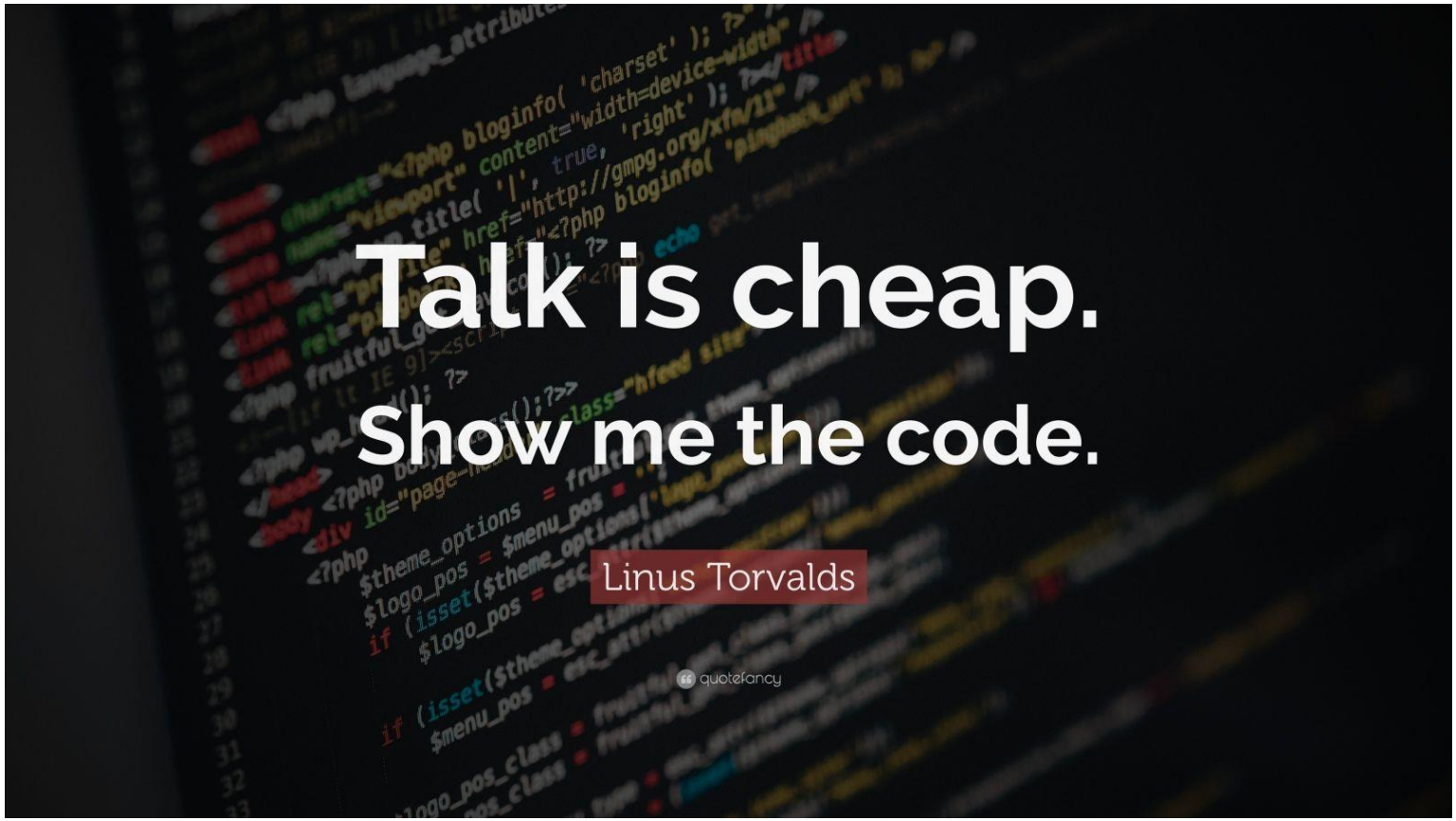
lect05_eg03.ipynb

目录

实战案例 4

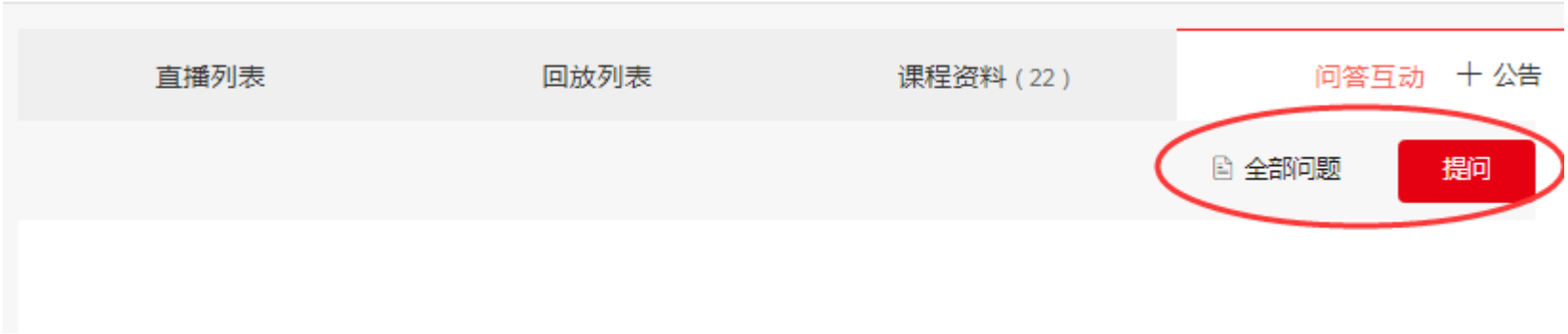
项目名称：动物种类识别

- 请参考相应的配套代码及案例讲解文档



问答互动

在所报课的课程页面，
1、点击“全部问题”显示本课程所有学员提问的问题。
2、点击“提问”即可向该课程的老师和助教提问问题。



联系我们

小象学院：互联网新技术在线教育领航者

– 微信公众号：**小象学院**