National Astronomical Data Center
国家天文科学数据中心

China-VO

Python in Astronomy
机器学习Python实践

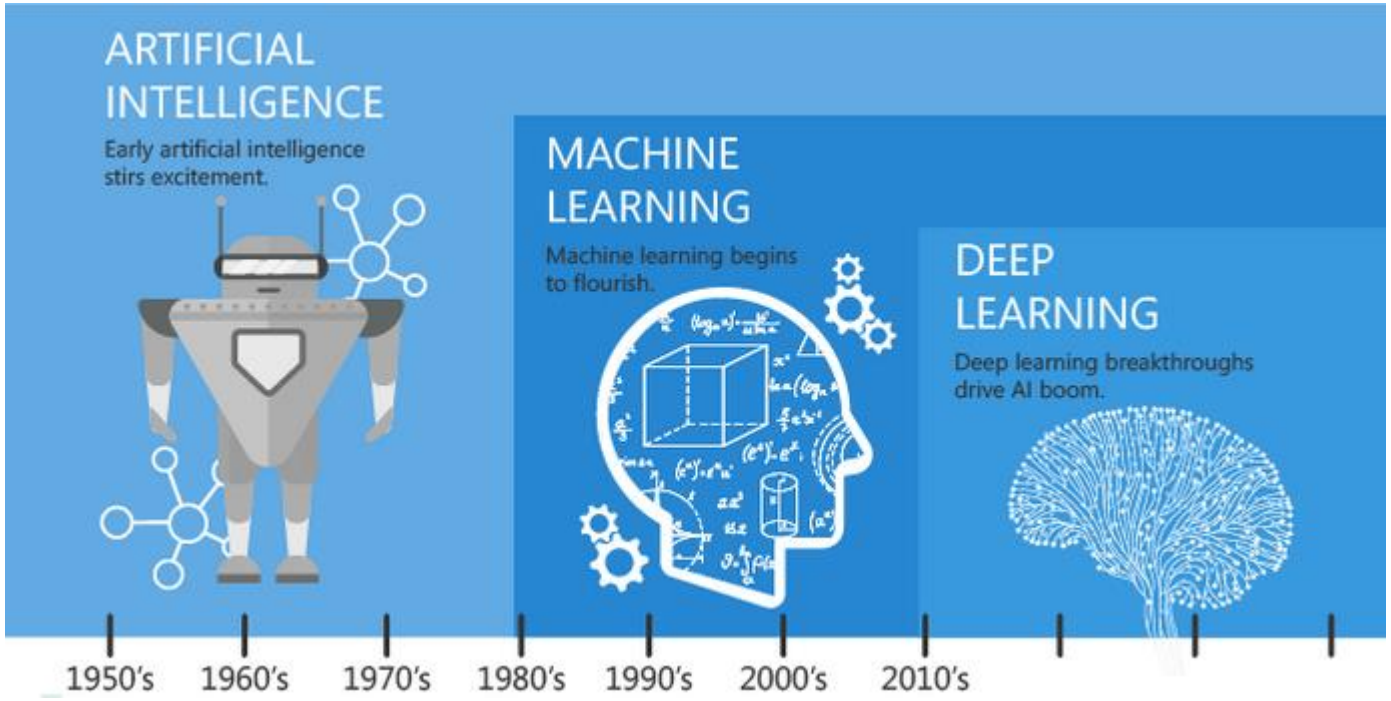
陶一寒
y.tao@nao.cas.cn

2020.12.9

主要内容

- 机器学习基本概念、类型、流程等
- 传统机器学习常用库scikit-learn
- 利用scikit-learn进行光谱分类实例
- 深度学习常用库
- 利用tf.keras搭建卷积神经网络进行光谱分类实例

机器学习基本概念



人工智能

- 计算机科学的一个分支，研究如何让机器模拟人类的智能（包括感知能力）来完成复杂的任务。

机器学习

- 人工智能的一个分支，研究如何使机器通过算法和统计模型从数据中学习建模，模拟或实现人类的学习行为。

深度学习

- 机器学习的一个分支。与传统机器学习的区别主要在于算法从原始特征自动提取抽象特征，而不是人工设计特征。

机器学习类型

- 按形式
  - 监督学习：根据带标签的训练数据建立模型以预测标签
  - 无监督学习：识别无标签数据中的结构
  - 半监督学习：使用少量标记数据，以及大量的未标记数据帮助提高算法的准确性和效率
  - 强化学习：使用稀疏且具有延时的标签，即奖励，基于当前环境做出行动决策，以最大化长期奖励。主要应用于游戏、机器人、智能驾驶、智能医疗等领域
- 按任务/目标
  - 分类：预测离散标签
  - 回归：预测连续标签
  - 聚类：在数据中检测和识别显著分组结构
  - 降维：将较高维数据转化为低维空间中表达，检测和识别低维空间中的结构

机器学习Python库

- 传统机器学习
  - Scikit-Learn （https://scikit-learn.org/stable/ https://github.com/scikit-learn/scikit-learn）

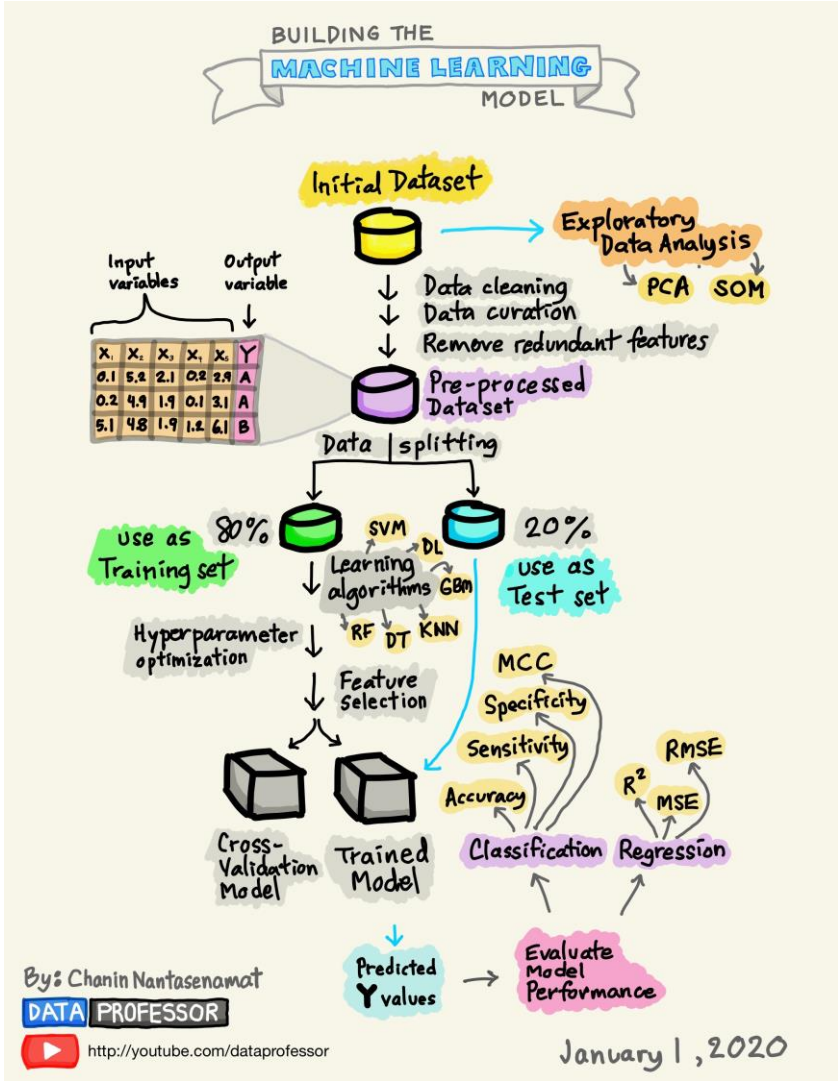
  提供了常用的机器学习算法（分类、回归、聚类、降维）的高效版本，以及数据预处理和模型选择的工具包

  - AstroML （https://www.astroml.org/index.html https://github.com/astroML/astroML）

  基于numpy, scipy, scikit-learn, matplotlib, astropy, 包含一些天文数据集的加载模块
- 深度学习框架
  - Tensorflow、Keras、PyTorch、Theano、Caffe…
- 其他
  - Numpy、Pandas、Matplotlib、Seaborn 等辅助数据预处理和可视化等环节

（监督）机器学习流程

- 数据准备
  数据预处理、数据集划分
- 特征工程
  特征表示、特征提取
- 创建模型
  选择合适的算法并建立模型、特征选择
- 模型训练
  将数据输入模型训练参数
- 模型评估和优化
  测试模型在新数据上的预测性能
- 模型部署
  保存和加载模型



https://github.com/dataprofessor/infographic

数据准备 → 特征工程 → 创建模型 → 模型训练 → 模型评估 → 模型部署

数据准备-sklearn

- 数据预处理 sklearn.preprocessing
  - 处理缺失值

    用适当的值（均值、中位数、众数）等来替换缺失值

    - Imputer `SimpleImputer(missing_values='NaN', strategy='mean')`
  - 特征缩放

    将特征映射到指定范围以去除不同维度量纲差异的影响。除了决策树和随机森林少数不需要特征缩放的机器学习算法，对大部分机器学习算法和优化算法来说，如果特征都在同一范围内，会获得更好的结果

    - 归一化 MinMaxScaler->[0, 1]，MaxAbsScaler->[-1, 1]
    - 标准化 StandardScaler 特征数据分布调整为标准正态分布，即均值为0，方差为1
  - 特征编码

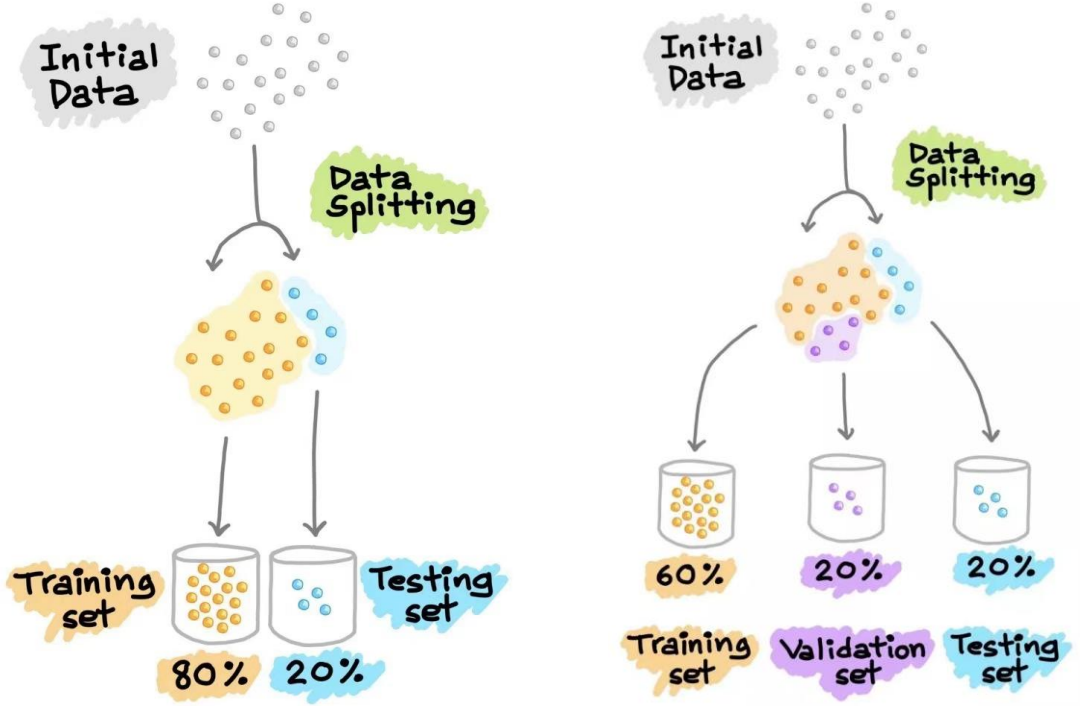
    将文本类别转换为数字, 能直接被模型使用 OneHotEncoder

数据准备-sklearn

- 数据集划分 sklearn.model_selection.train_test_split

  将数据分为训练集和测试集

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)



特征工程-sklearn

- 特征提取
  - sklearn.feature_extraction
  - 主成分分析（PCA）

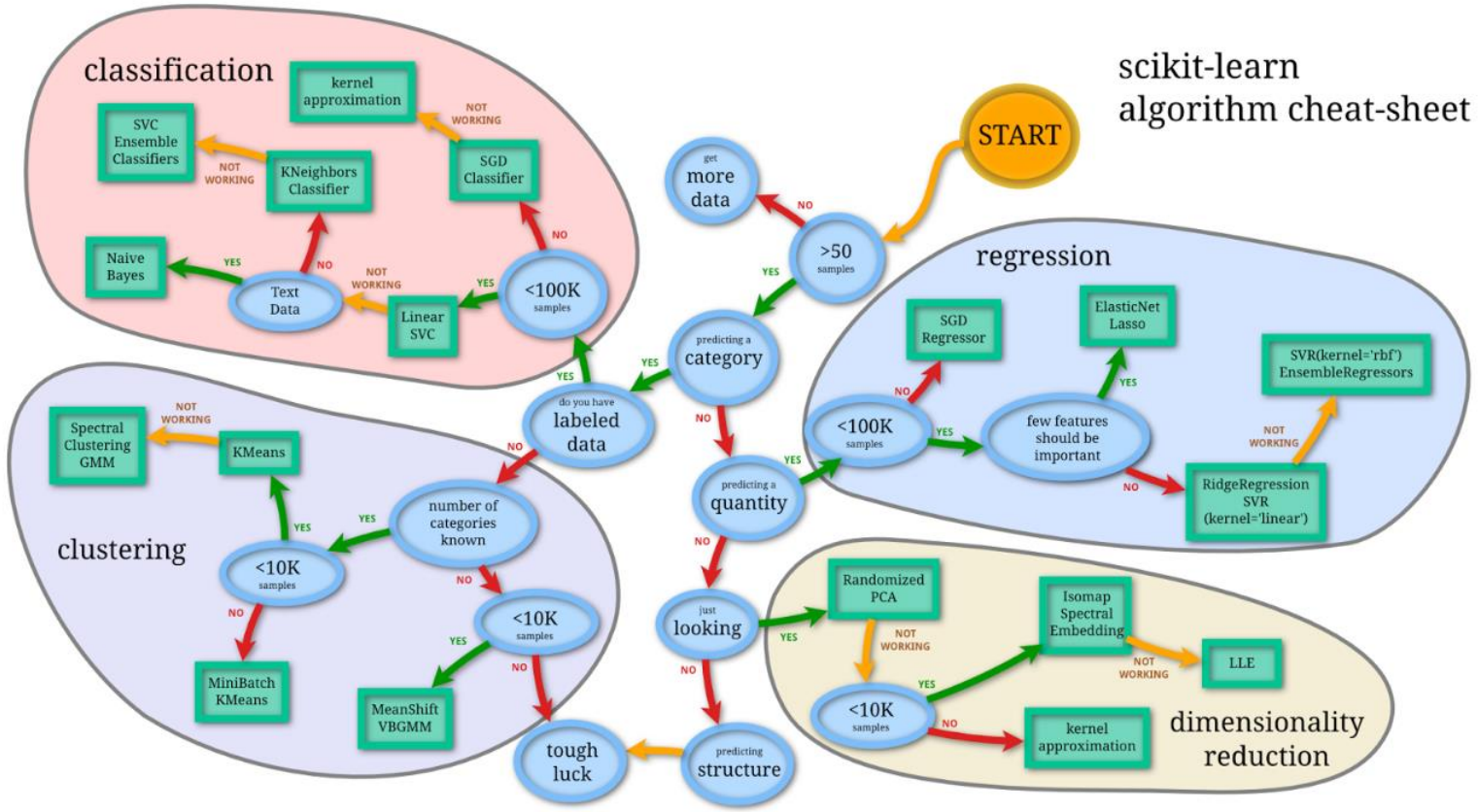
    常用的特征提取和降维算法，分析数据的低维结构

    将初始样本映射到维度更低的样本空间中
- 特征选择

  从大量特征中选择一个特征子集，剔除不相关或冗余的特征
  - sklearn.feature_selection
  - 决策树模型特征重要性

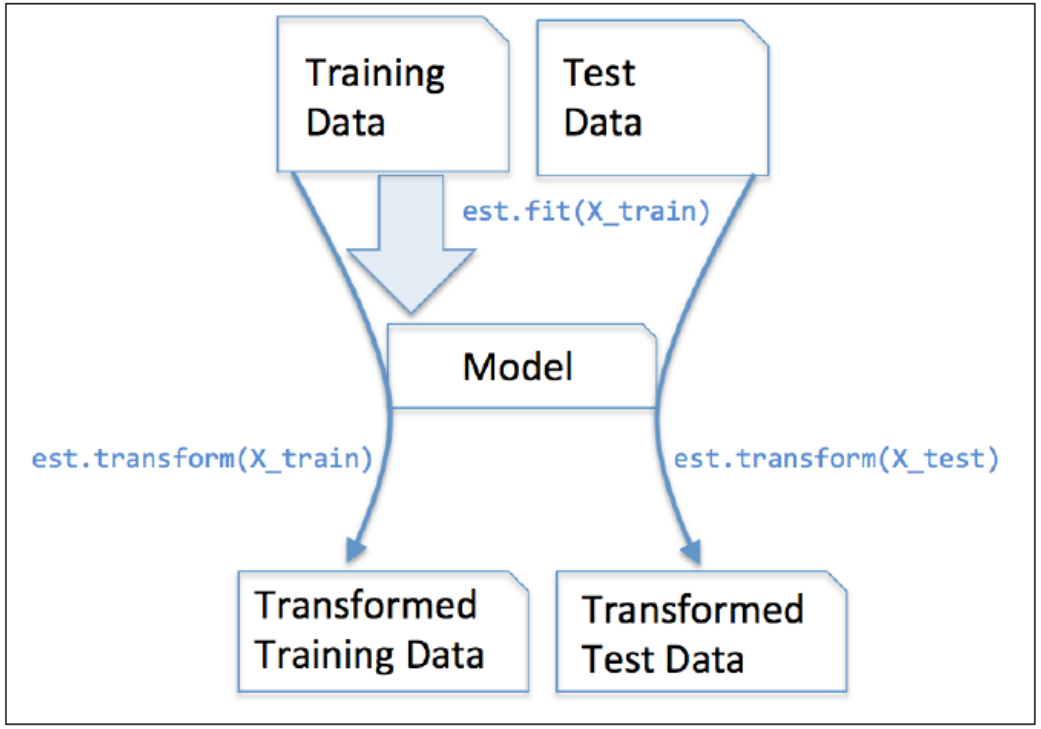
创建模型-sklearn

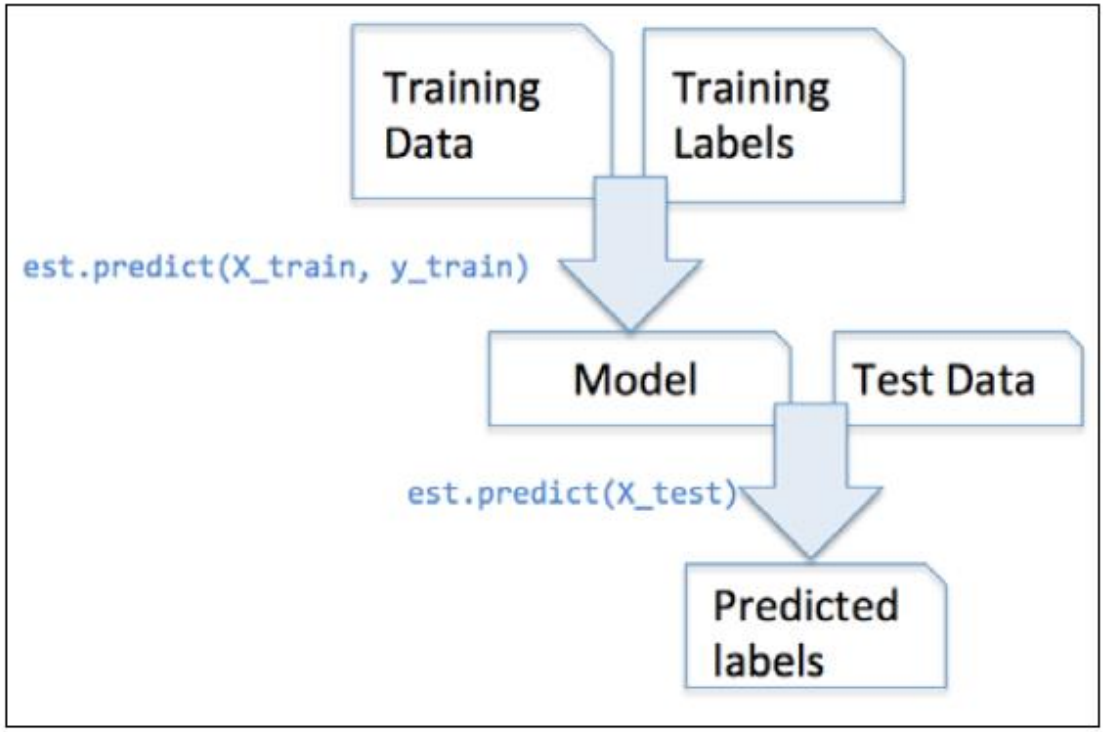
- 常用算法
  - 分类和回归
    - 线性回归
    - K近邻
    - 支持向量机
    - 朴素贝叶斯
    - 决策树
    - 随机森林
    - …
  - 聚类
    - K均值
    - DBSCAN
    - 层次聚类
    - …
  - 降维
    - 主成分分析（PCA）
    - 独立成分分析（ICA）
    - 等距特征映射（ISOMAP）
    - …
    - …



模型训练-sklearn

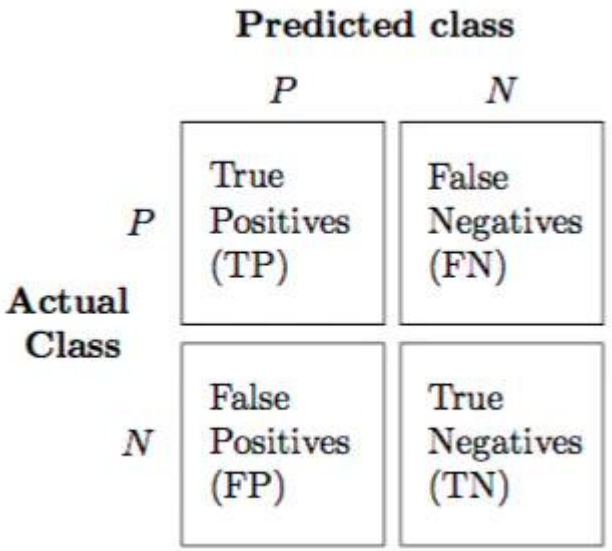
- sklearn.estimator
- fit(), transform()





模型评估-sklearn

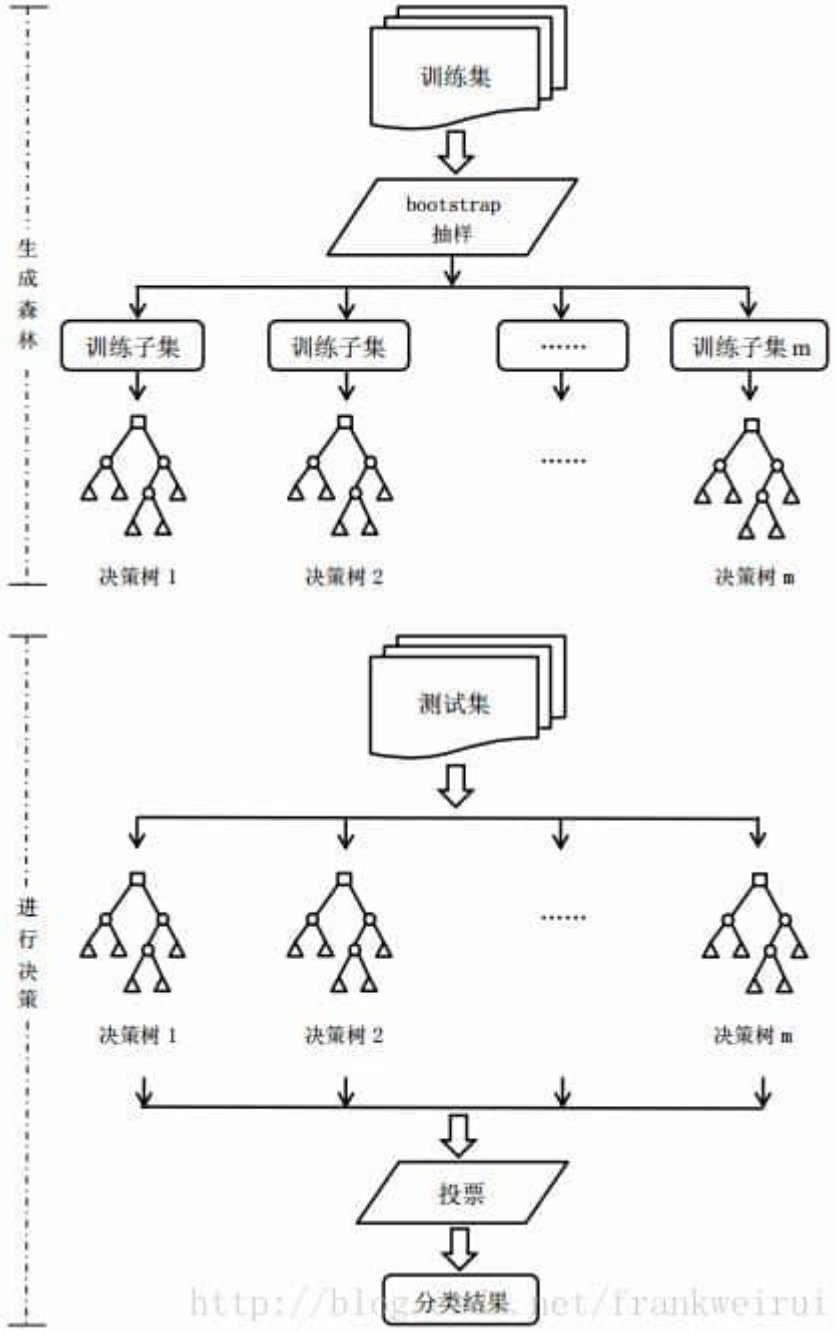
- from sklearn import metrics
- 预测
  - predict
  - predict_proba
- 模型性能指标
  - confusion_matrix
  - classification_report
    - Precision：TP/TP+FP，正确预测的正例数 /预测为正例的总数
    - Recall：TP/TP+FN ，正确预测的正例数 /实际正例总数
    - F1-score：
  - roc_auc_score，roc曲线
    - True Positive Rate = TP/TP+FN,被预测为正的正样本数 /正样本实际数
    - False Positive Rate = FP/FP+TN,被预测为负的正样本数 /正样本实际数
    - ROC曲线左上角（0，1），分类器性能越好
    - AUC-ROC曲线下的面积（积分），值越大分类性能越好
    - 不受不同类别样本数量不平衡的影响



随机森林

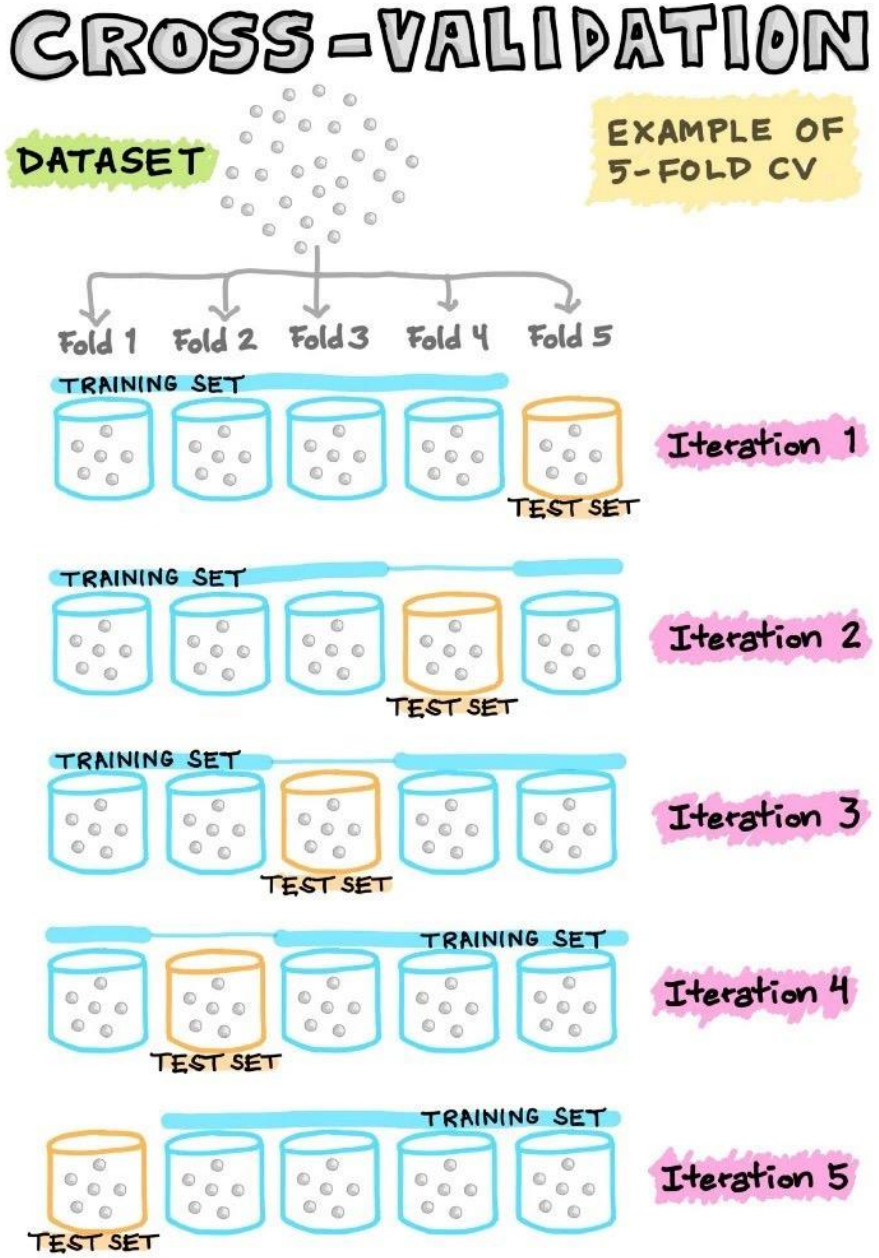
- 随机森林是一种集成算法，由多个决策树通过bagging方式结合起来
- Bagging集成策略主要是随机采样独立生成多个树模型，最终通过投票或均值获得预测结果。
- 另一种集成方法是boosting，代表算法如GBDT，XGBoost
- from sklearn.ensemble import RandomForestClassifier
- 主要参数：
  - n_estimators: 决策树的个数
  - oob_score: 是否采用袋外数据（有放回随机抽样未被抽到的样本）来评估模型的好坏
  - max_depth: 决策树的最大深度
  - min_samples_leaf: 叶子节点的最少样本数

  …
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html



模型优化-sklearn

- 交叉验证
  - sklearn.model_selection.cross_val_score
- 网格搜索
  - 搜索模型的超参数
  - sklearn. model_selection. grid_search



https://github.com/dataprofessor/infographic

模型部署-sklearn

- 模型保存和恢复
  - 不跨平台 joblib->.pkl
    - from sklearn.externals import joblib
    - dump
    - load
  - 跨平台 ->.pmml
    - sklearn2pmml
    - pickle

深度学习框架

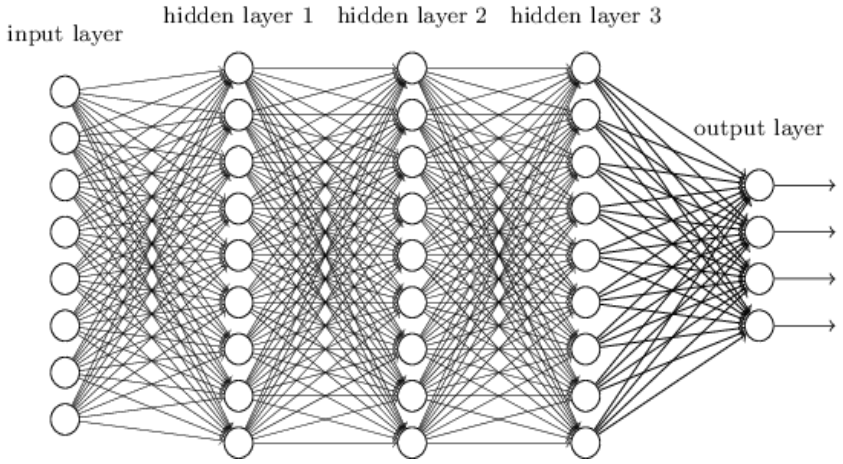
- Tensorflow (https://www.tensorflow.org https://github.com/tensorflow/tensorflow)
  - 端到端开源机器学习平台
  - 支持多种语言，Python的API是最完整及简单易用的
- Keras （https://keras.io/ https://github.com/keras-team/keras）
  - 基于Python的深度学习框架
  - 用于构建和训练深度学习模型的高阶 API，支持Tensorflow、Theano、CNTK等后端
  - 模块化易扩展，方便机器学习新手和研究人员使用快速搭建网络
- PyTorch （https://pytorch.org/ https://github.com/keras-team/keras）
  - 基于Torch的Python开源机器学习框架
  - 支持更先进的定制化

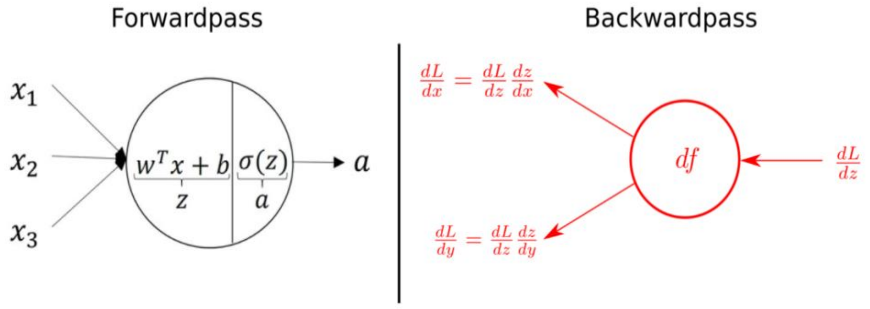
数据准备-Keras

- Keras models接受三种类型的输入：
  - Numpy arrays
  - TensorFlow Dataset –如果数据集很大，需要分批训练并使用GPU
  - Python generators
- 读入数据集：
  - tensorflow.keras.preprocessing.image_dataset_from_directory
  - tensorflow.keras.preprocessing.text_dataset_from_directory
  - tensorflow.data.experimental.make_csv_dataset
- 数据预处理
  - tensorflow.keras.layers.experimental.preprocessing
  - 特征编码：生成独热编码 tf.keras.utils.to_categorical

深度神经网络

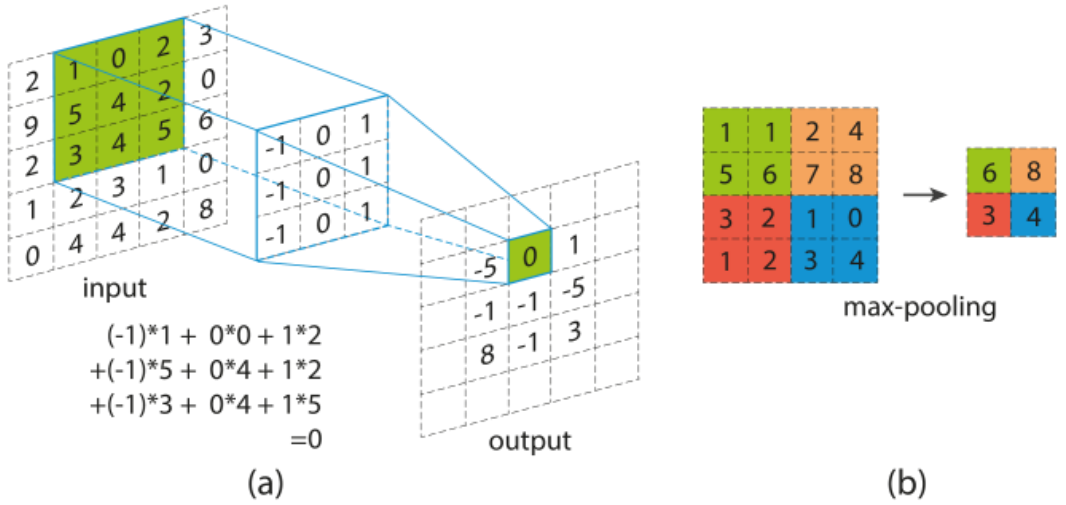
- 全连接神经网络：上下层每个节点均有连接
  - 前向传播：每一层的输出是下一层的输入
  - 激活函数（Activations）：每个节点的输入与该节点的做矩阵乘法（线性变换），之后通过激活函数转换为高阶非线性特征
  - 目标函数（loss/cost）：优化的目标，度量标签和预测之间的误差，如欧氏距离、交叉熵等
  - 反向传播：更新权重，确定每层的W、b参数；通过链式法则求偏导，计算目标函数对每个权重的偏导（即梯度）。
  - 梯度下降优化：寻找使目标函数最小的参数。梯度下降法需要给定一个初始点，并求出该点的梯度向量，然后以负梯度方向为搜索方向，以一定的步长进行搜索，从而确定下一个迭代点，再计算该新的梯度方向，如此重复直到收敛。





深度神经网络

- 卷积神经网络（CNN)
  - 应用卷积运算代替矩阵乘法的神经网络
  - 卷积层：由一组卷积核（滤波器）组成，依次对输入进行卷积操作
  - 池化层：最大池化、均值池化、平移不变性、降维



图片来源：https://www.arxiv-vanity.com/papers/1604.07043/

模型构建-Keras

- Keras中的模型构建方式
  - Sequential，一系列网络层按顺序构成模型，Sequential()
  - Functional，函数式模型，Model()，可定义多输出、非循环有向模型等复杂模型
- 网络层(layers)、目标函数(loss)、优化器（Optimizer）、初始化策略（Initializers）、激活函数（Activation）、正则化方法（Regularizer）等都是独立的模块，可用来自由组合构建模型
- layers 网络层
  - 全连接层（Dense）
  - 卷积层（Conv1D, Conv2D, Conv3D…）
  - 池化层（MaxPooling1D, MaxPooling2D, MaxPooling3D, AveragePooling1D, AveragePooling2D, AveragePooling3D…）
  - Dropout层
  - Flatten层
  - 局部连接层（ LocallyConnceted）
  - 循环层（Reurrent）

  …

  自己定义新的层
- summary() 显示网络结构 每层的输出和参数等信息

模型训练-Keras

- 编译 compile

  主要参数:

  - 优化器optimizer-可指定预定义的优化器名或Optimizer对象, 可选的优化器如SGD（随机梯度下降），RMSprop，Adam，Adadelta，Adamax，Nadam等
  - 目标函数loss-即模型试图最小化的目标函数,可指定预定义的目标函数名如categorical_crossentropy, mean_squared_error, mean_absolute_error，kullback_leibler_divergence, binary_crossentropy, 也可以为一个损失函数
  - 指标列表metrics-指标可以是一个预定义指标的名字,也可以是一个定制的评估函数。对分类问题，一般将该列表设置为metrics=['accuracy']

- 训练 fit

  主要参数:

  - batch_size-批大小，即每次反向传播的数据量
  - epochs -遍历数据集次数
  - validation_data –输入验证数据
  - validation_split-划分出验证数据

- history对象记录训练过程中训练和验证的准确率

模型评估-keras

- predict()
- evaluate ()
  - 返回loss和 acc

模型保存-keras

- save()
  - 模型的结构，以便重构该模型
  - 模型的权重
  - 训练配置（损失函数，优化器等参数）
  - 优化器的状态，以便于从上次训练中断的地方开始
- load_model()
- save_weights()

  只保存模型的参数，不保存图结构
- load_weights()