此文件用于课堂辅助同学学习机器学习。

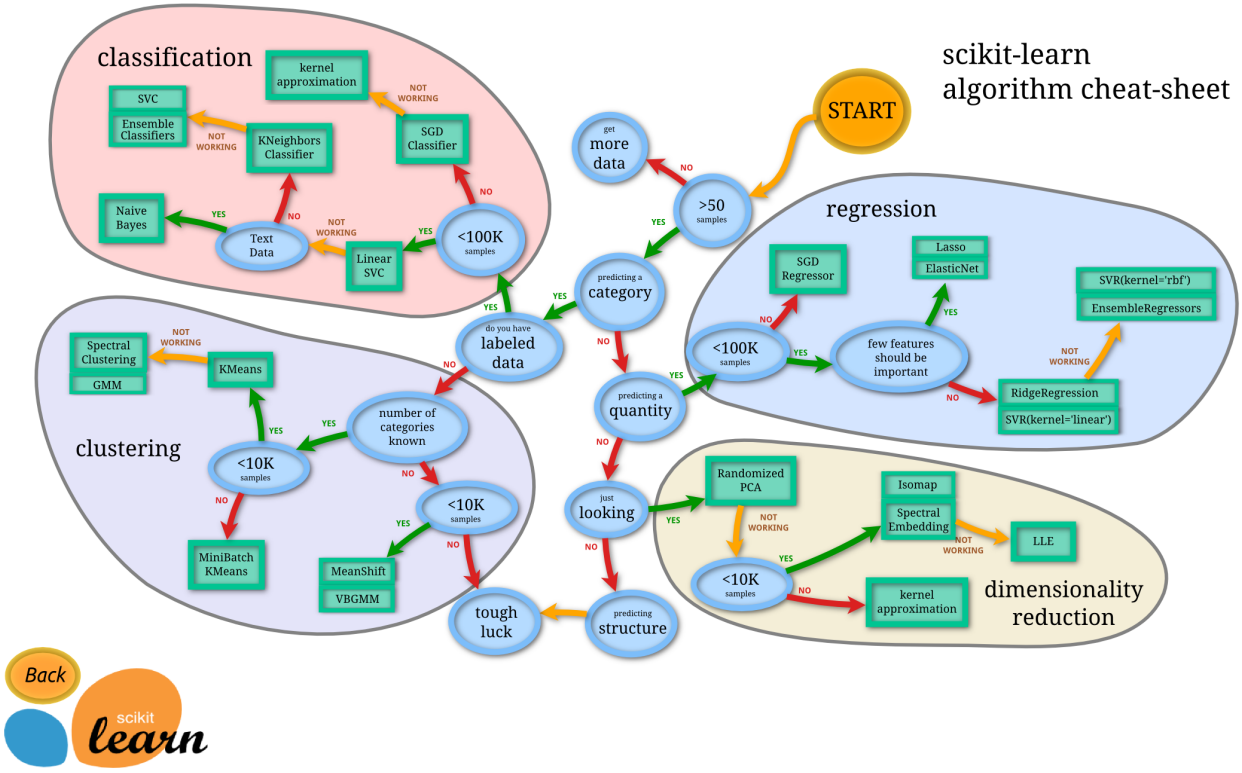
1. Scikit-learn 简单介绍

`sklearn` (scikit-learn简称) 是基于 `numpy` 和 `scipy` 扩展的模块，项目最早由`David Cournapeau` 于2007年谷歌编程大赛发起，后来越来越多的贡献者加入到模块的开发，经过多年发展，成为了python里机器学习最强大的工具包。 基本涵盖了一个简单的数据挖掘流程中所有可能需要用到的算法。

项目代码传送门：[Github/scikit-learn]

1.1. 包含什么

此软件包几乎包含了所有数据挖掘或机器学习的常用场景。如图所示。



1.2. 安装

```
pip3 install -U scikit-learn
```

在conda的环境中，sklearn已经内置。

2. 以一个分类任务来讲

我们以分类问题为例，来介绍一个使用 `sklearn` 的完整的机器学习过程。 本章节主要包含相关功能介绍，以及一些常用的函数。 其中不同函数都有着许多可选参数，在此不赘述，如有需求可以查阅相关文档。

2.1. 训练前（数据）

2.1.1. 获取数据

`sklearn` 内置了很多常用的数据集，可以直接在 `sklearn.datasets` 中进行调用。

此处我们使用 iris 数据集来作为示例。

```
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target
```

对于自己导入的数据集，仅需使其满足相应的维度要求即可。
对于一般的 data vector ， `X` 应该为 (m,n)， `Y` 应该为 (m,)。
其中 m 为数据的数量，n 为数据的特征的数量。

2.1.2. 数据预处理

数据通常需要进行简单的预处理，包括但不限于：

- 标准化/归一化
- 缺失值补全
- 维度选择

此处对于此数据集，我们需要对其进行标准化，sklearn 中对此类处理已经有相应的函数。

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = scaler.fit_transform(X)

# 之后遇到的新数据依旧需要用此scaler标准化
New_data_standard = scaler.fit_transform(New_data)
```

2.1.3. 训练及测试集构建

有些数据集或者一些现成的 benchmark 已经提供现成的 train/test 划分。 但是如果没有提供划分的话，我们需要手动来对其进行划分。

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
random_state=2)
```

2.1.4. 数据可视化

有的时候我们会需要对数据可视化来查看其特性。 通常情况下我们需要将数据转换为2维或3维向量。 这个可以通过单独的维度选择得到，也可以通过降维来得到。

```
X_2d = X[:, :2]  # 选择两个维度

# 或者使用 PCA 降维
pca = PCA(n_components=2)
pca.fit(X)
X_2d = pca.transform(X)
```

降维过后便可以将数据点画图表示出来，此处需要用到的软件包为 matplotlib，请见此文档。

2.2. 训练中（模型）

2.2.1. 模型选用

本文仅以分类问题为例，这是一个常见的 supervised learning 的问题。

sklearn 中有很多内置的相关模型实现，eg. LinearRegression, SVC, LinearSVC, DecisionTreeClassifier, etc.

此处我们选用 SVC 作为我们的样例。

```
from sklearn.svm import SVC

clf = SVC()
clf.fit(X_train, y_train)
```

2.2.2. 模型调参及训练

当然不同的模型会存在不同的参数，这些参数会对模型的表现产生至关重要的影响。一般来说，我们会用 cross validation 的方式来对模型进行训练和调参。

此处我们介绍一种 sklearn 内置的参数选择方法。 我们依旧以 SVC 为例，使用 RBF 核，对其 C 和 gamma 进行选择。

```
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.model_selection import GridSearchCV

C_range = np.logspace(-2, 10, 13)
gamma_range = np.logspace(-9, 3, 13)
param_grid = dict(gamma=gamma_range, C=C_range)

cv = StratifiedShuffleSplit(n_splits=5, test_size=0.2, random_state=42)
clf = GridSearchCV(SVC(), param_grid=param_grid, cv=cv)
clf.fit(X, y)

print("The best parameters are %s with a score of %0.2f" % (grid.best_params_,
grid.best_score_))
```

2.3. 训练后（表现）

2.3.1. 模型表现评估

模型训练结束之后我们需要对其进行评估，评估需要在测试集上进行。 常见的评估方式有 accuracy/recall/precision/F1/ROC/AUC 等。

对于训练好的模型，一般可以直接使用 score 来进行评估。 一般默认是返回 accuracy，这个样本标签较为平衡的任务中一般已经够用。

```
score = clf.score(X_test, y_test)
```

sklearn 还提供了 classification_report 功能，可以提供更多的评估信息。

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

2.3.2. 模型表现可视化

有时我们仍然需要对分类的结果进行可视化，此处的可视化一般来说是在最开始数据可视化的基础上再加入预测的标签。 此处仍属于matplotlib的范围，请查阅相关文档。