1 python机器学习/数据挖掘概述

1.1 如何用python做机器学习/数据挖掘?

1.2 准备python环境

In [ ]:

```
# 加载numpy库，偶尔需要使用相应的函数
import numpy as np
```

In [ ]:

```
# 加载pandas库（用于数据管理）
import pandas as pd
```

In [ ]:

```
# 加载matplotlib.pyplot库
from matplotlib import pyplot as plt

# 要求图形在notebook中直接显示
% matplotlib inline
```

In [ ]:

```
# 加载seaborn库
import seaborn as sns
# 加载seaborn默认格式设定
sns.set()
```

In [ ]:

```
# 解决中文显示问题
plt.rcParams["font.family"] = "STXIHEI"
```

In [ ]:

```
# 如果需要使用statsmodels，可以考虑用api接口简化后续调用
import statsmodels.api as sm
```

1.3 sklearn的样本数据集

In [ ]:

```
from sklearn import datasets

boston = datasets.load_boston()
```

In [ ]:

```
# 显示数据对象的内容
boston.
```

In [ ]:

```
# 转换为数据框便于使用
bostondf = pd.DataFrame(boston.data, columns = boston.feature_names)
bostondf.head()
```

1.4 sklearn基本操作入门

sklearn中集成了各种数据挖掘所需的变量变换、变量信息处理、统计建模、模型优化、模型评估方法，为便于使用，这些操作基本上都封装成了具有统一API的类，调用时都遵循统一的操作规范。

标准的类参数

class sklearn.大类名称.Modelclass(类参数列表)

Modelclass中基本通用的类参数：

```
fit_intercept = True : 模型是否包括常数项
    使用该选项就不需要在数据框中设定cons
n_jobs = 1 : 使用的例程数，为-1时使用全部CPU
max_iter = 200 : int，模型最大迭代次数
tol = 0.0001  模型收敛标准
warm_start = False : 是否使用上一次的模型拟合结果作为本次初始值
sample_weight = None : 案例权重
random_state = None : int/RandomState instance/None, 随机器的设定
shuffle = True : 是否在拆分前对样本做随机排列
```

)# 大多数类参数都会有默认值

In [ ]:

```
from sklearn import preprocessing

# 完整的类名称为sklearn.preprocessing.StandardScaler()
std = preprocessing.StandardScaler()
std
```

In [ ]:

```
from sklearn import linear_model

# 完整的类名称为sklearn.linear_model.LinearRegression()
reg = linear_model.LinearRegression()
reg
```

Modelclass中基本通用的类方法

get_params([deep]) : 获取模型的具体参数设定
set_params(**params) : 重新设定模型参数
fit(X, y[, sample_weight]) : 使用数据拟合模型/方法

特征处理class: Preprocessing、降维、Feature extraction/selection
transform(X[, y]) : 使用拟合好的模型对指定数据进行转换
fit_transform(X[, y]) : 对数据拟合相应的方法，并且进行转换

建模分析class: Classification、Regression、Clustering
predict(X) : 使用拟合好的模型对数据计算预测值
predict_proba(X) : 模型给出的每个案例（各个类别）的预测概率
score(X, y[, sample_weight]) : 返回模型决定系数/模型准确度评价指标

In [ ]:

```
std.get_params()
```

In [ ]:

```
# 使用fit方法，使std类获取数据中相应的信息
std.fit(boston.data)
```

In [ ]:

```
std.mean_
```

In [ ]:

```
std.var_
```

In [ ]:

```
ZX = std.transform(boston.data)
ZX[:2]
```

In [ ]:

```
std.fit_transform(boston.data)[:2]
```

In [ ]:

```
# 使用fit方法，使reg类基于指定数据估计出回归模型的相应参数
reg.fit(boston.data, boston.target)
```

In [ ]:

```
reg.coef_
```

In [ ]:

```
pred = reg.predict(boston.data)
pred[:10]
```

In [ ]:

```
reg.score(boston.data, boston.target)
```

ModelClass中基本通用的类属性

注意：模型拟合前这些属性可能不存在

```
coef_ : array, 多因变量时为二维数组
intercept_ : 常数项

classes_ : 每个输出的类标签
n_classes_ : int or list, 类别数
n_features_ : int, 特征数

loss_ : 损失函数计算出来的当前损失值
n_iter_ : 迭代次数
```

In [ ]:

```
std.mean_, std.scale_
```

In [ ]:

```
reg.intercept_, reg.coef_
```

简化的调用函数

特征处理class往往会有简化版本的函数可供调用，功能类似，但使用上更简单。

```
class sklearn.preprocessing.StandardScaler()
sklearn.preprocessing.scale()
```

In [ ]:

```
preprocessing.scale(boston.data)[:2]
```

模型的保存（持久化）

可以直接使用通过使用Python的pickle模块将训练好的模型保存为外部文件，但最好使用sklearn中的joblib模块进行操作。

In [ ]:

```
# 保存为外部文件
from sklearn.externals import joblib

joblib.dump(std, 'f:/std.pkl')
joblib.dump(reg, 'f:/reg.pkl')
```

In [ ]:

```
# 读入外部保存的模型文件
reg2 = joblib.load('f:/reg.pkl')
reg2.coef_
```

1.5 实战练习

加载sklearn自带的iris数据集，熟悉该数据集的各种属性，并尝试将其转换为数据框。

尝试在不参考任何帮助文档的情况下，按照sklearn中的标准API操作方式，使用BP神经网络对iris数据进行拟合，并返回各案例的预测类别、预测概率等结果。

> BP神经网络对应的类为：class sklearn.neural_network.MLPClassifier()
> 此处只为API操作演示，不进一步讨论模型拟合前的数据预处理问题

将上题中生成的模型存储为外部文件，并重新读入。

2 数据的预处理

2.1 数值变量的标准化

数据标准化可以去除均数、离散程度量纲差异太大的影响。

> 减去均数：去除均数的影响。
> 除以标准差：去除离散程度的影响。

但是标准化对离群值的影响无能为力，其结果仍然受离群值的严重影响。

2.1.1 对单个数据集进行标化

sklearn.preprocessing.scale(

> X : {array-like, sparse matrix}，需要进行变换的数据阵
> axis = 0 : 指定分别按照列(0)还是整个样本(1)计算均数、标准差并进行变换
> with_mean = True : 是否中心化数据（移除均数）
> with_std = True : 是否均一化标准差（除以标准差）
> copy = True : 是否生成副本而不是替换原数据

)

In [ ]:

```
bostondf.head()
```

In [ ]:

```
bostondf.describe()
```