scikit-learn

——江昭鹏

## 一：概念

scikit-learn是一个由Python实现的机器学习的库，其中包含大量机器学习算法、数据集，是数据挖掘方便的工具。

里面对一些常用的机器学习方法进行了封装，在进行机器学习任务时，并不需要每个人都实现所有的算法，只需要简单的调用sklearn里的模块就可以实现大多数机器学习任务。

机器学习任务通常包括分类（Classification）和回归（Regression），常用的分类器包括SVM、KNN、贝叶斯、线性回归、逻辑回归、决策树、随机森林、xgboost、GBDT、boosting、神经网络NN。

常见的降维方法包括TF-IDF、主题模型LDA、主成分分析PCA等等。

官方网站：<http://scikit-learn.org/>

## 特征抽取

现实世界中多数特征都不是连续变量，比如分类、文字、图像等，为了对非连续变量做特征表述，需要对这些特征做数学化表述，因此就用到了特征提取。

特征选择(排序)对于数据科学家、机器学习从业者来说非常重要。好的特征选择能够提升模型的性能，更能帮助我们理解数据的特点、底层结构，这对进一步改善模型、算法都有着重要作用。

特征选择主要有两个功能：

* 减少特征数量、降维，使模型泛化能力更强，减少过拟合
* 增强对特征和特征值之间的理解

拿到数据集，一个特征选择方法，往往很难同时完成这两个目的。通常情况下，我们经常不管三七二十一，选择一种自己最熟悉或者最方便的特征选择方法（往往目的是降维，而忽略了对特征和数据理解的目的）。

在许多机器学习相关的书里，很难找到关于特征选择的内容，因为特征选择要解决的问题往往被视为机器学习的一种副作用，一般不会单独拿出来讨论。

sklearn.feature\_extraction提供了特征提取的很多方法

### 对字典数据进行特征抽取

把字典中的一些类别数据，分别进行转换为特征数据

类：

|  |
| --- |
| sklearn.feature\_extraction.DictVectorizer(sparse = True) |

这个方法将映射列表即字典转换为Numpy数组或scipy.sparse矩阵

参数：

sparse： 是否转换为scipy.sparse矩阵表示，默认开启

如果以ndarray类型展示可指定sparse=False，或对结果值调用.toarray()方法。

导入：

|  |
| --- |
| **from** sklearn.feature\_extraction **import** DictVectorizer |

实例化：

|  |
| --- |
| dict = DictVectorizer() |

应用并转化列表为目标类型：

|  |
| --- |
| data = dict.fit\_transform(dir) |

其中，dir是一个字典列表或DataFrame

返回值data默认是一个sparse的矩阵格式，这是为了节约内存

他是使用one-hot编码进行计算的

获取键值：

|  |
| --- |
| dict.get\_feature\_names() |

获取映射列表：

|  |
| --- |
| dict.inverse\_transform(data) |

示例数据：

|  |
| --- |
| dir = [  {**'city'**:**'北京'**,**'temperature'**:100},  {**'city'**:**'上海'**,**'temperature'**:60},  {**'city'**:**'广州'**,**'temperature'**:30} ] |

结果：

|  |
| --- |
| [**'city=上海'**, **'city=北京'**, **'city=广州'**, **'temperature'**]  [[ 0. 1. 0. 100.]  [ 1. 0. 0. 60.]  [ 0. 0. 1. 30.]] |

### 对文本数据进行特征抽取

文本的特征提取应用于很多方面，比如说文档分类、垃圾邮件分类和新闻分类。那么文本分类是通过词是否存在、以及词的概率（重要性）来表示。

类：

|  |
| --- |
| sklearn.feature\_extraction.text.CountVectorizer() |

导入：

|  |
| --- |
| **from** sklearn.feature\_extraction.text **import** CountVectorizer |

它的方法参考DictVectorizer(),大致一样的。

示例数据：

|  |
| --- |
| text = [**'life is short, i like python'**, **'life is long, i dislike python'**] |

结果：

|  |
| --- |
| [**u'dislike'**, **u'is'**, **u'life'**, **u'like'**, **u'long'**, **u'python'**, **u'short'**]  [[0 1 1 1 0 1 1]  [1 1 1 0 1 1 0]] |

注意，对文本数据进行特征抽取时，单个字母或字符不统计。

英文文档中都是使用空格来分开各个单词，因此可以方便的进行统计计算，但中文文档中，各个词语之间没有分隔符，所以要对中文文档进行特征抽取就要先对文档进行分词，中文分词可以使用jieba分词。

导入jieba:

|  |
| --- |
| **import** jieba |

文本数据：

|  |
| --- |
| text = [**'人生苦短，我用Python'**,**'人生漫长，不用Python'**] |

jieba分词：

|  |
| --- |
| text = map(**lambda** t:**' '**.join(list(jieba.cut(t))),text) |

分词结果：

|  |
| --- |
| text = [**"人生 苦短 ， 我用 Python"**, **"人生 漫长 ， 不用 Python"**] |

接下来的操作就和英文文档特征抽取一样的了。

### 使用TF-IDF表示词的重要性

概念：

如果某个词或短语在一篇文章总出现的概率高，并且在其他的文章总很少出现，则认为此词或短语具有很好的类别区分能力，适合用来分类。

作用：

用以评估一字词对于一个文件集或一个词料库中的其中一份文件的重要性

tf:term frequency:  
 词的频率：出现的次数  
idf:inverse document frequency：  
 逆文档频率：log(总文档数量/该词出现的文档数量)

重要性 = tf \* idf

类：

|  |
| --- |
| sklearn.feature\_extraction.text.TfidfVectorizer() |

TfidfVectorizer会根据指定的公式将文档中的词转换为概率表示

它的方法同于其他的特征抽取方法，不再一一列举。

## 特征数据预处理

数据预处理即通过特定的统计方法（数学方法）将数据转换成算法要求的数据。

对于数值型数据，采用标准缩放：

* 归一化
* 标准化

类别型数据：

* one-hot编码

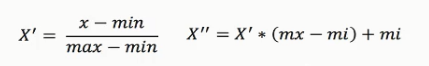
时间类型

* 时间的切分

### 归一化

特点：通过对原始的数据进行变换把数据映射到(默认[0,1])之间

目的：特征对最终结果不会造成更大的影响



公式：

注：作用于每一列，max为每一列的最大值，min为每一列的最小值，那么x’’为最终结果，mx,mi分别为指定区间值，默认mx为1，mi为0

示例：

|  |  |  |  |
| --- | --- | --- | --- |
| 特征1 | 特征2 | 特征3 | 特征4 |
| 1. | 0. | 0. | 0 |
| 0. | 1. | 1. | 0.83 |
| 0.5 | 0.5 | 0.6 | 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| 特征1 | 特征2 | 特征3 | 特征4 |
| 90 | 2 | 10 | 40 |
| 60 | 4 | 15 | 45 |
| 75 | 3 | 13 | 46 |

导入：

|  |
| --- |
| **from** sklearn.preprocessing **import** MinMaxScaler |

实例化：

|  |
| --- |
| mm = MinMaxScaler() |

参数：

feature\_range：默认为(0,1),可修改区间

压缩和结果：

|  |
| --- |
| data = mm.fit\_transform(train)  **print** data |
| **[[ 1. 0. 0. 0. ]**  **[ 0. 1. 1. 0.83333333]**  **[ 0.5 0.5 0.6 1. ]]** |

注意：在特定的情景下最小值和最大值是变化的，另外，最大值和最小值非常容易受异常的影响，此时结果显然也很容易受到影响，所以这种方法鲁棒性较差，只适合传统精确小数据场景。那么，我们就需要使用标准化。

### 标准化

相对于归一化，标准化的适用范围更大，算法也更加的精确，如果出现了异常点，由于标准化具有一定的数据量，少量的异常点对于平均值的影响并不大，从而方差改变较小。

处理之后每列数据来说所有的数据都聚集在均值0附近标准差差为1

特点：通过对原始数据进行变换把数据变化到均值为0，标准差为1的范围内



公式：

注：作用于每一列，mean为平均值，为标准差

其中，方差（考虑数据的稳定性）



var成为方差，

导入：

|  |
| --- |
| **from** sklearn.preprocessing **import** StandardScaler |

压缩：

|  |
| --- |
| train = [[1., -1, 3.], [2., 4., 3.], [4., 6., -1.]] ss = StandardScaler() data = ss.fit\_transform(train) **print** data |
| [[-1.06904497 -1.35873244 0.70710678]  [-0.26726124 0.33968311 0.70710678]  [ 1.33630621 1.01904933 -1.41421356]] |

## 数据降维

### 特征选择

特征选择就是单纯的从提取到的所有特征中选择部分特征作为训练集特征，特征在选择前和选择后可以改变值、也可以不改变值，但是选择后的特征维度肯定比选择前小，毕竟我们只选择了其中的一部分特征。

主要方法（三大武器）：

* Filter（过滤式）：VarianceThreshold
* Embedded（嵌入式）：正则化，决策树
* Wrapper（包裹式）

过滤式

导入：

|  |
| --- |
| **from** sklearn.feature\_selection **import** VarianceThreshold |

过滤：threshold=0.0:方差为0，即把所有相同的数据特征删除掉

|  |
| --- |
| vt = VarianceThreshold(threshold=0.0) train = [[0,2,0,3],[0,1,4,3],[0,1,1,3]] data = vt.fit\_transform(train) **print** data |
| [[2 0]  [1 4]  [1 1]] |

此时，将第一列和第四列删除了（这两列相同）

### 主成分分析PCA

本质:PCA是一种分析，简化数据集的技术。

目的：是数据维度压缩，尽可能减低源数据的维数（复杂度），损失少量信息。

作用：可以削减回归分析或者聚类分析中特征的数量

当数据特征达到上百个的时候，这时考虑数据的简化，才会使用PCA。

高维度数据之间，特征数据通常是相关的。

相似特征数据压缩简化后，数据的相似性非常大，这时就可以删除其中一个特征。

导入：

|  |
| --- |
| **from** sklearn.decomposition **import** PCA |

降维：（n\_components表示压缩率，当为小数时，表示去除率达到百分之几，比如0.9表示去除相似度达到90%的数据，一般这个值选择在0.9-0.95之间，具体大小还要看具体分析；而当为整数时，表示去除整数个最相同的特征。）

|  |
| --- |
| pca = PCA(n\_components=0.9) train = [[2,8,4,5],[6,3,0,8],[5,4,9,1]] data = pca.fit\_transform(train) **print** data |

变成了2列，缩减了90%

|  |
| --- |
| [[ 1.28620952e-15 3.82970843e+00]  [ 5.74456265e+00 -1.91485422e+00]  [ -5.74456265e+00 -1.91485422e+00]] |

## 机器学习任务

### 1、数据类型

在机器学习任务中，数据类型分为两种，即离散型数据和连续型数据。

* 离散型数据：有记录不同类别个体的数目所得到的数据，又称计数数据，所有这些数据全部都是整数，而且不能再细分，也不能进一步提高他们的精确度。如汽车数量、人口数量、班级数等等。
* 连续型数据：变量可以在某个范围内取任一数，即变量的取值可以是连续的，如长度、时间、质量值等等，这类整数通常是非整数，含有小数部分。

注 ：离散型是区间内不可分，连续型是区间内可分。

### 2、监督学习和无监督学习

那么，根据这些不同数据，我们在机器学习中所使用到的算法也就各有不同，而机器学习分为监督学习和无监督学习：

**监督学习（Supervised learning）**使用分类和回归算法，输入的数据有特征有标签，即有标准答案。可以由输入数据中学到或建立一个模，并依次模型推测出新的结果。输入数据输入特征值和目标值所组成。函数的输出可以是一个连续的值（称为回归），或是输出是有限个离散值（称作分类）

**无监督学习（unSupervised learning）**使用聚类算法，输入的数据有特征无标签，即无标准答案。可以由输入数据中学到或建立一个模型，并依次模型推测出新的结果。输入的数据是有输入特征值所组成。

下面是机器学习的算法分类。

* 监督学习（预测）：
* 分类算法：分类是监督学习的一个核心问题，在监督学习中，当输出变量为有限个离散值时，预测问题变为分类问题。最基础的便是二分类问题，即判断是非，从两个类别中选择一个作为预测结果。
* 回归算法：回归是监督学习中的另一个重要问题，回归用于预测输入变量和输出变量之间的关系，输出是连续型的值。
* 无监督学习
* 聚类算法：k-means

**如果目标值是离散型，难么使用分类算法**

**如果目标值是连续型，那么使用回归算法**

**如果没有目标值，使用聚类算法**

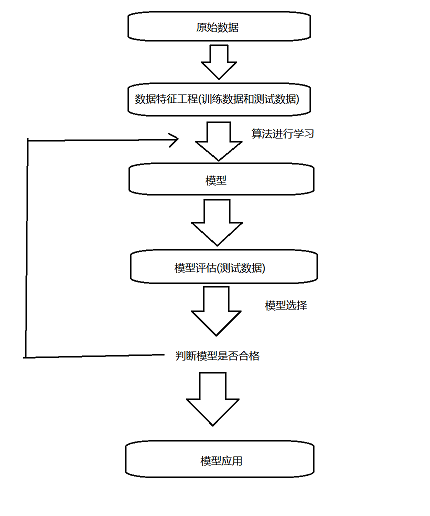
sklearn库的常用分类算法如下：

|  |  |  |
| --- | --- | --- |
| **模块名称** | **函数名称** | **算法名称** |
| linear\_model | LogisticRegression | 逻辑斯蒂回归 |
| svm | SVC | 支持向量机 |
| neighbors | KNeighborsClassifier | K最近邻分类 |
| naive\_bayes | GaussianNB | 高斯朴素贝叶斯 |
| tree | DecisionTreeClassifier | 分类决策树 |
| ensemble | RandomForestClassifier | 随机森林分类 |
| ensemble | GradientBoostingClassifier | 梯度提升分类数 |

sklearn库的常用回归算法如下：

|  |  |  |
| --- | --- | --- |
| **模块名称** | **函数名称** | **算法名称** |
| linear\_model | LinearRegression | 线性回归 |
| svm | SVR | 支持向量回归 |
| neighbors | KNeighborsRegression | K最近邻回归 |
| tree | DecisionTreeRegression | 回归决策树 |
| ensemble | RandomForestRegression | 随机森林回归 |
| ensemble | GradientBoostingRegression | 梯度提升回归树 |

### 机器学习的开发流程：



<https://www.cnblogs.com/wj-1314/p/10179741.html>