Python 数据挖掘入门与实践

一、开启数据挖掘之旅

1.3.1 简单的排序规则

- **支持度**：指数据集中规则应验的次数，统计起来很简单。有时候，还需要对支持度进行规范化，即再除以规则有效前提下的总数量。
- **置信度：**符合给定条件（即规则的"如果"语句所表示的前提条件）的所有规则里，跟当前规则结论一致的比例有多大。计算方法为首先统计当前规则的出现次数，再用它来除以条件（"如果"语句）相同的规则数量。

1.5.2 OneR算法

根据已有数据中，具有相同特征值的个体最可能属于哪个类别进行分类

- 算法首先遍历每个特征的每一个取值，对于每一个特征值，统计它在各个类别中的出现次数，找到它出现次数最多的类别，并统计它在其他类别中的出现次数。
- 统计完所有的特征值及其在每个类别的出现次数后，我们再来计算每个特征的错误率。

  计算方法为把它的各个取值的错误率相加，选取错误率最低的特征作为唯一的分类准则（OneR），用于接下来的分类。

二、scikit-learn

- 估计器（Estimator）：用于分类、聚类和回归分析。
- 转换器（Transformer）：用于数据预处理和数据转换。
- 流水线（Pipeline）：组合数据挖掘流程，便于再次使用。

2.1 估计器

- fit()：训练算法，设置内部参数。该函数接收训练集及其类别两个参数。
- predict()：参数为测试集。预测测试集类别，并返回一个包含测试集各条数据类别的数组。

2.2.2 标准预处理

MinMaxScaler类进行基于特征的规范化

```
from sklearn.preprocessing import MinMaxScaler
```

把每个特征的值域规范化为0到1之间。最小值用0代替，最大值用1代替，其余值介于两者之间。

```
X_transformed = MinMaxScaler().fit_transform(X)
```

- 为使每条数据各特征值的和为1，使用sklearn.preprocessing.Normalizer。
- 为使各特征的均值为0，方差为1，使用sklearn.preprocessing.StandardScaler，常用作规范化的基准。
- 为将数值型特征的二值化，使用sklearn.preprocessing.Binarizer，大于阈值的为1，反之为0。

2.3 流水线

流水线把这些步骤保存到数据挖掘的工作流中。之后你就可以用它们读入数据，做各种必要的预处理，然后给出预测结果 `from sklearn.pipeline import Pipeline`

```
scaling_pipeline = Pipeline([('scale', MinMaxScaler()),          #规范特征取值
（scale）
                             ('predict', KNeighborsClassifier())])   #预测
（predict）

scores = cross_val_score(scaling_pipeline, X_broken, y, scoring='accuracy')
```

四、用亲和性分析方法推荐电影

4.1.1 亲和性分析算法

Apriori算法可以说是经典的亲和性分析算法。它只从数据集中频繁出现的商品中选取共同出现的商品组成频繁项集（frequent itemset）。一旦找到频繁项集，生成关联规则就很容易了。生成频繁项集后，将不再考虑其他可能的却不够频繁的项集（这样的集合有很多），从而大大减少测试新规则所需的时间。

其他亲和性分析算法有Eclat和频繁项集挖掘算法（FP-growth）

1. 需要为Apriori算法指定一个项集要成为频繁项集所需的最小支持度。任何小于最小支持度的项集将不再考虑。如果最小支持度值过小，Apriori算法要检测大量的项集，会拖慢的运行速度；最小支持度值过大的话，则只有很少的频繁项集。
2. 找出频繁项集后，在第二个阶段，根据置信度选取关联规则。可以设定最小置信度，返回一部分规则，或者返回所有规则，让用户自己选。

- 置信度过低将会导致规则支持度高，正确率低；
- 置信度过高，导致正确率高，但是返回的规则少。

五、用转换器抽取特征

5.2 特征选择

- 降低复杂度：随着特征数量的增加，很多数据挖掘算法需要更多的时间和资源。减少特征数量，是提高算法运行速度，减少资源使用的好方法。
- 降低噪音：增加额外特征并不总会提升算法的表现。额外特征可能扰乱算法的正常工作，这些额外特征间的相关性和模式没有实际应用价值（这种情况在小数据集上很常见）。只选择合适的特征有助于减少出现没有实际意义的相关性的几率。
- 增加模型可读性：根据成千上万个特征创建的模型来解答一个问题，对计算机来说很容易，但模型对我们自己来说就晦涩无比。因此，使用更少的特征，创建我们自己可以理解的模型，就很有必要

```
#去除特征中方差为0的列，方差为0对分类没有意义
from sklearn.feature_selection import VarianceThreshold
vt = VarianceThreshold()
Xt = vt.fit_transform(X)

输出每一列的方差。
print(vt.variances_)
```

选择最佳特征

- SelectKBest返回*k*个最佳特征，
- SelectPercentile返回表现最佳的前r%个特征。

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
transformer = SelectKBest(score_func=chi2, k=3)
Xt_chi2 = transformer.fit_transform(X, y)
```

九、作者归属问题

- 作者画像：根据作品界定作者的年龄、性别或其他特性。
- 作者验证：根据作者已有作品，推断其他作品是否也是他写的。
- 作者聚类：作者验证问题的延伸，用聚类分析方法把作品按照作者进行分类。