RandomForest

内容列表

介绍

该文档是为了帮助你快速了解该项目的相关信息。

Random Forest（随机森林，简称RF）是一种基于树模型的Bagging的优化版本。本项目使用原生python实现了简单的随机森林用于西瓜分类，并将随机森林用于解决Titanic生存者的问题。

项目分为两个部分，请分别作为项目打开：

- Watermelon部分使用随机森林算法对西瓜进行分类，演示了如何训练决策树并构建随机森林进行分类。

- Titanic部分使用随机森林算法预测泰坦尼克中的幸存者，并与直接使用决策树进行分类的结果进行了对比

环境

~=表示兼容的版本

如果只运行Watermelon部分，只需要python~=3.10。

```
python~=3.10
pandas~=1.5.1
numpy~=1.23.3
scikit-learn~=1.1.3
matplotlib~=3.6.2
seaborn~=0.12.1
graphviz~=0.20.1
pydotplus~=2.0.2
```

使用说明

Watermelon

整个算法中包括三个部分：输入数据集，根据数据集随机创建多棵决策树，利用创建的决策树对输入的object类别进行判定

分别在三个.py文件中对其进行了实现

Step1：

运行createDataSet.py生成数据集并输出为dataSet.json和labels.json

数据集的最后一列应为该对象的类别，如：好瓜，坏瓜

数据集输入示例：

```
[
        # 1
        ['青绿', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', '好瓜'],
        # 2
        ['乌黑', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑', '好瓜'],
        # 3
        ['乌黑', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', '好瓜'],
        # 4
        ['青绿', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑', '好瓜'],
        # 5
        ['浅白', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', '好瓜'],
        # 6
        ['青绿', '稍蜷', '浊响', '清晰', '稍凹', '软粘', '好瓜'],
        # 7
        ['乌黑', '稍蜷', '浊响', '稍糊', '稍凹', '软粘', '好瓜'],
        # 8
        ['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '硬滑', '好瓜'],
```

```
        # 9
        ['乌黑', '稍蜷', '沉闷', '稍糊', '稍凹', '硬滑', '坏瓜'],
        # 10
        ['青绿', '硬挺', '清脆', '清晰', '平坦', '软粘', '坏瓜'],
        # 11
        ['浅白', '硬挺', '清脆', '模糊', '平坦', '硬滑', '坏瓜'],
        # 12
        ['浅白', '蜷缩', '浊响', '模糊', '平坦', '软粘', '坏瓜'],
        # 13
        ['青绿', '稍蜷', '浊响', '稍糊', '凹陷', '硬滑', '坏瓜'],
        # 14
        ['浅白', '稍蜷', '沉闷', '稍糊', '凹陷', '硬滑', '坏瓜'],
        # 15
        ['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '软粘', '坏瓜'],
        # 16
        ['浅白', '蜷缩', '浊响', '模糊', '平坦', '硬滑', '坏瓜'],
        # 17
        ['青绿', '蜷缩', '沉闷', '稍糊', '稍凹', '硬滑', '坏瓜']
]
```

label输入示例：

```
['色泽', '根蒂', '敲击', '纹理', '脐部', '触感', 'good/bad']
```

Step2：

运行RandomForest.py生成随机森林，并输出为treeSet.json

此步中需要step1中生成的dataSet.json和labels.json，当然你也可以用自己生成的dataSet.json文件和labels.json文件（要注意输入格式符合范例）

Step3:

运行DecideByRandomForest.py，读取dataSet.json labels.json 和treeSet.json

用户在控制台输入要判断的对象

输入对象应为一个属性列表，例：

```
['青绿', '蜷缩', '浊响', '清晰', '凹陷', '硬滑',]
```

控制台将输出随机森林对该对象的类别判断结果如：

```
好瓜
```

Titanic

背景

根据泰坦尼克号上乘客上的一些乘客的个人信息以及存活状况，生成合适的模型并预测其他人的存活状况。这里分别使用了随机森林、逻辑回归和朴素贝叶斯的方法训练模型并进行预测。

结构说明

文件/目录名	说明
Titanic.py	主程序文件
train.csv	训练数据集
test.csv	测试数据集
predict.csv	随机森林的预测结果
TreeGraph	存储随机森林中决策树可视化的结果

主要步骤

1. 数据清洗（Data Cleaning）
2. 特征工程（Feature Engineering）
3. 建模（Modeling）
4. 评估(Evaluation)

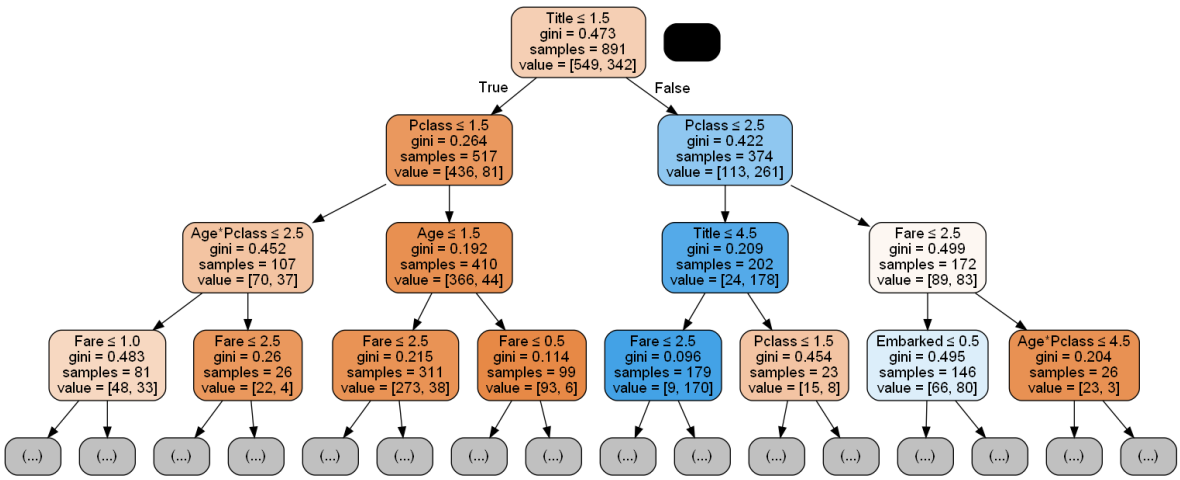
运行说明

指定随机森林中决策树的数量为100，使用的决策树为CART决策树(以基尼系数作为评判标准)

取消以下注释以生成可视化的决策树，结果保存在TreeGraph目录中

```
# 176行开始
# print(acc_random_forest)
# 可视化随机森林中的决策树
# m=0
# for per_estimator in random_forest.estimators_:
#     dot_data = tree.export_graphviz(per_estimator, out_file=None,
#                                     feature_names=X_train.columns,
#                                     class_names=['0','1'],
#                                     filled=True, rounded=True,
#                                     special_characters=True)
#     graph = pydotplus.graph_from_dot_data(dot_data)
#     m=m+1
#     graph.write_pdf("./TreeGraph/"+(str(m)+"DTtree.pdf"))
```

决策树可视化示例：



算法分析

算法介绍

Random Forest（随机森林，简称RF）是一种**基于树模型的Bagging的优化版本**。核心思想依旧是Bagging。**具体过程如下**：

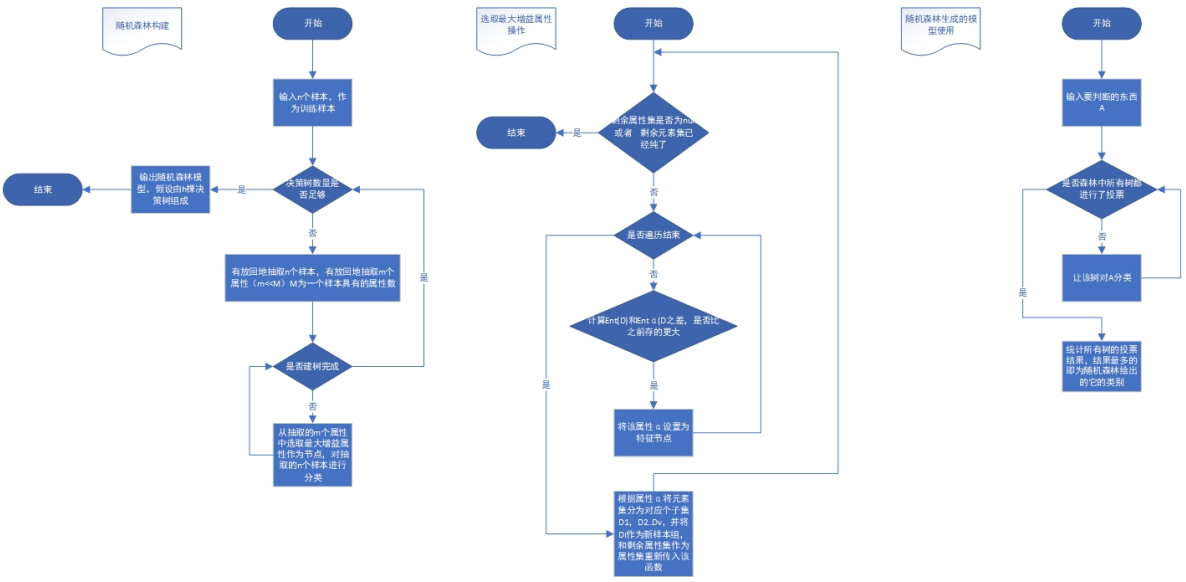
> 输入为样本集D={(x_1,y_1),(x_2,y_2),…,(x_m,y_m)}，弱分类器迭代次数T。
>
> 输出为最终的强分类器f(x)。
>
> 1. 对于t=1,2,…,T：
>
> 对训练集进行第t次随机采样，共采集m次，得到包含m个样本的采样集D_T。
>
> 用采样集D_t训练第t个决策树模型G_t(x)，在训练决策树模型的节点的时候，在节点上所有的样本特征中选择一部分　　样本特征，在这些随机选择的部分样本特征中选择一个最优的特征来做决策树的左右子树划分。
>
> 2.如果是分类算法预测，则T个弱学习器投出最多票数的类别或者类别之一为最终类别。如果是回归算法，T个弱学习 器得到的回归结果进行算术平均得到的值为最终的模型输出。

每棵树都按照如下规则生成：

> 1. 如果训练集大小为N，对于每棵树而言，随机且有放回地从 训练集中的抽取N个训练样本（这种采样方式称为bootstrap sample方法），作为该树的训练集；
> 2. 如果每个样本的特征维度为M，指定一个常数m<<M，随机地从M个特征中选取m个特征子集，每次树进行分裂时，从这m个特征中选择最优的；
> 3. 每棵树都尽最大程度的生长，并且没有剪枝过程。
>
> 随机森林中的“随机”就是指的这里的两个随机性。两个随机性的引入对随机森林的分类性能至关重要。由于它们的引入，使得随机森林不容易陷入过拟合，并且具有很好得抗噪能力。

算法流程

算法流程图：



算法复杂度受影响的可能因素分析

一般情况下，决策树的深度越深，叶节点个数越多，拟合效果越好，树的复杂度越高。

随机森林的复杂度与训练样本数和树的个数成正比，和树的深度成正比，和随机选取的特征数量成正比。

与其他分类算法的比较

比较了随机森林与传统的分类算法的优缺点。

随机森林的优缺点

优点

1. 它可以出来很高维度（特征很多）的数据，并且不用降维，无需做特征选择
2. 它可以判断特征的重要程度
3. 可以判断出不同特征之间的相互影响
4. 不容易过拟合
5. 训练速度比较快，容易做成并行方法
6. 实现起来比较简单
7. 对于不平衡的数据集来说，它可以平衡误差。
8. 如果有很大一部分的特征遗失，仍可以维持准确度。

缺点

1. 随机森林已经被证明在某些噪音较大的分类或回归问题上会过拟合。
2. 对于有不同取值的属性的数据，取值划分较多的属性会对随机森林产生更大的影响，所以随机森林在这种数据上产出的属性权值是不可信的

逻辑回归的优缺点

优点

1. 训练速度较快，分类的时候，计算量仅仅只和特征的数目相关；
2. 易理解，模型的可解释性非常好，从特征的权重可以看到不同的特征对最后结果的影响；
3. 适合二分类问题，不需要缩放输入特征；
4. 内存资源占用小，因为只需要存储各个维度的特征值；

缺点

1. 不能用Logistic回归去解决非线性问题，因为Logistic的决策面试线性的；
2. 对多重共线性数据较为敏感；
3. 很难处理数据不平衡的问题；
4. 准确率并不是很高，因为形式非常的简单(非常类似线性模型)，很难去拟合数据的真实分布；
5. 逻辑回归本身无法筛选特征，有时会用gbdt来筛选特征，然后再上逻辑回归

朴素贝叶斯的优缺点

优点

1. 朴素贝叶斯模型有稳定的分类效率。
2. 对小规模的数据表现很好，能处理多分类任务，适合增量式训练，尤其是数据量超出内存时，可以一批批的去增量训练。
3. 对缺失数据不太敏感，算法也比较简单，常用于文本分类。

缺点

1. 理论上，朴素贝叶斯模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为朴素贝叶斯模型给定输出类别的情况下,假设属性之间相互独立，这个假设在实际应用中往往是不成立的，在属性个数比较多或者属性之间相关性较大时，分类效果不好。而在属性相关性较小时，朴素贝叶斯性能最为良好。
2. 需要知道先验概率，且先验概率很多时候取决于假设，假设的模型可以有很多种，因此在某些时候会由于假设的先验模型的原因导致预测效果不佳。
3. 由于我们是通过先验和数据来决定后验的概率从而决定分类，所以分类决策存在一定的错误率。
4. 对输入数据的表达形式很敏感。

不同算法在Titanic中的效果对比

Titanic部分对比了随机森林算法与朴素贝叶斯、逻辑回归方法的模型得分，得分的计算方式为平均精确度*100。

模型	得分
随机森林	86.64
逻辑回归	81.37
朴素贝叶斯	76.88

可以看出随机森林算法在Titanic任务中的表现要优于逻辑回归方法和朴素贝叶斯方法。

相关文章

决策树总结（二）如何构建决策树 - 知乎 (zhihu.com)

一文看懂随机森林 - Random Forest（附 4 个构造步骤+10 个优缺点） - 知乎 (zhihu.com)

kaggle 泰坦尼克事件——随机森林算法实现*Ap21ril的博客-CSDN博客*泰坦尼克号随机森林

Kaggle竞赛 —— 泰坦尼克号（Titanic） - 知乎 (zhihu.com)