浙江大学

本科实验报告

课程名称：数据挖掘导论

姓　　名：张溢弛

学　　院：计算机科学与技术学院

专　　业：软件工程

学　　号：3180103772

指导教师：李石坚

2021 年 5 月 28 日

浙江大学 实验报告

课程名称：数据挖掘导论　　实验类型：综合

实验项目名称：决策树分类器的实现与性能对比

学生姓名：张溢弛　专业：软件工程　学号：3180103772

同组学生姓名：无　指导老师：李石坚

实验地点：曹西-503　实验日期：2021 年 5 月 28 日

目录

一　实验基本信息

1.1　实验要求

自己编程实现一个分类器（如决策树、朴素贝叶斯、基于规则的分类器等），自行确定实验数据集，在数据集上与商用系统中的同类分类器在不同指标上开展性能对比，并用 t 检验确定你的分类器和对比的分类器的性能差异是否显著。

1.2　实验环境

实验的基本环境如下：

- 操作系统：Windows 10
- 编程环境：Anaconda 4.10+Jupyter Notebook 4.4.0
- 编程语言：Python 3.6.5
- Python 库版本：Pandas 0.23.0，numpy 1.14.3

1.3　实验内容

本次实验我自己编程实现了一个基于信息增益作为分类标准的决策树 (通常被成为 ID3 树)，并使用 Titanic 数据集进行训练和测试，与开源机器学习库 sklearn 中的决策树模型进行了指标的对比，最后采用了 T 检验的方式检验了两个模型 (自己实现的决策树和 sklearn 中的决策树) 的性能差异是否显著。

二　决策树实现

2.1　基本理论

决策树 (Decision Tree) 实际上是通过树形的结构来进行分类或者回归，在分类问题中可以根据样本的不同特征属性的不同属性值来进行多次分类的决策最终确定输入样本的类别，也就相当于执行了一连串嵌套的 if-else 语句，也可以被认为是定义在特征空间和类空间上的条件概率分布。决策树的优点是模型具有较好的可解释性，分类的速度比较快。

决策树往往采用递归的方式来构建，并且需要在每一层选取一个最优的特征并用这个特征来进行分类，这也是决策树最关键的地方，而如何选择最优的划分特征，需要通过一定的评价标准。常见的评价标准由信息熵，信息增益，信息增益比等。

2.1.1 信息熵 Entropy

熵可以用来表示随机变量的**不确定性**。如果 X 是一个取值个数为有限个值即 $P(X = x_i) = p_i$，那么随机变量 X 的熵的定义就是：

$$H(x) = -\sum_{i=1}^{n} p_i \log p_i \in [0, \log n] \tag{1}$$

而对于随机变量 X 和 Y，如果 $P(X = x_i, Y = y_i) = p_{ij}$，那么在 X 给定的情况下随机变量 Y 的条件熵代表了 X 给定条件下 Y 的条件概率分布的熵对于 X 的数学期望：

$$H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i) \tag{2}$$

当用数据估计得到熵和条件熵的时候，他们又分别被称为经验熵和经验条件熵。

2.1.2 信息增益

信息增益 (information gain) 表示**已经知道特征 X 的信息而使得类 Y 的信息的不确定度减少的程度**。特征 A 对于数据集 D 的信息增益 $g(D, A)$ 定义为 D 的经验熵和 D 在 A 给定条件下的经验条件熵的差值，即：

$$g(D, A) = H(D) - H(D|A) \tag{3}$$

信息增益又叫做**互信息 (mutual information)**，两个概念是等价的。基于信息增益的特征选择方法一般都是对于当前训练集 D，计算其每个特征的信息增益，并比较大小，选择信息增益最大的特征来进行当前层的分类。

假设训练的数据集是 D，有 K 个不同的分类 C_i 满足 $\sum_{i=1}^{K} |C_k| = |D|$，其中特征 A 有 n 个不同的取值 a_i，根据特征的取值将 D 划分成了 n 个不同的子集 D_i，而每个子集 D_{ik} 中表示属于类别 C_k 的样本，那么数据集 D 的经验熵可以表示为：

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|} \tag{4}$$

特征 A 对于数据集 D 的条件经验熵可以表示为：

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|} \tag{5}$$

2.1.3 ID3 决策树

ID3 算法的核心是在决策树的各个节点上用信息增益来进行特征的选择，并且递归地建立决策树，从根结点开始，对于当前节点计算所有可能的特征值的信息增益，选择信息增

益最大的特征作为划分的一句并建立字节点，直到所有的特征的信息增益都很小或者没有子节点位置停止调用

2.2 算法描述

ID3 算法可以对输入的训练数据集 D，特征集合 A 生成一棵决策树 T，其基本步骤描述如下所示 (参考统计学习方法书中的算法描述)：

1. 如果 D 中的所有样本属于同一类，那么 T 就是一棵单节点树，用这个类别作为树的标签并返回
2. 如果 A 是空集那么 T 是单节点树，并用 D 中出现次数最多的类别 C 作为树的类别标签并返回
3. 否则计算每个 A 中的特征对应于数据集 D 的信息增益，选择出信息增益最大的特征 A_g
4. 对于 A_g 的每一种可能值 a_i 依据 $A_g = a_i$ 对数据集进行划分，得到若干个非空子集 D_i，将 D_i 中出现次数最大的类作为标记，并构建子节点，由节点及其子节点构成树 T，并返回 T
5. 对第 i 个子节点，以 D_i 作为训练集，$A - A_g$ 作为特征集，递归调用 ID3 构建函数，得到子树 T_i 并返回。

2.3 核心模块实现

我使用 Python 编写了一个类来实现决策树的结构，具体的代码可以在提交的压缩文件中看到，这个类主要有如下几个关键函数：

```
1  def entropy(y, sample_weights) # 计算信息熵
2  def information_gain(X, y, index, sample_weights) # 计算信息增益
3  def majority_vote(y, sample_weights=None) # 使用极大似然法进行投票
4  def get_label_distribute(y, weight) # 得到样本的概率分布
5
6  class DecisionTree:
7      def __init__(self,
8                   max_depth,
9                   min_samples_leaf):
10         # 初始化决策树，设定最大深度，节点最少样本数等超参数
11
12     def fit(self, X, y):
13         # 模型的训练
14         # 给样本赋予相同的权重
15
16     def split_data(self, X, y, index, value, sample_weights, X_valid=None, y_valid=None):
17         # 按照属性和属性值的标准分割数据
18
```

```
19    def choose_best_feature(self, X, y, sample_weights):
20        # 选择当前情况下最好的特征
21
22    def build_tree(self, X, y, feature_names, depth, sample_weights, X_valid=None, y_valid=None):
23        # 用dict的形式构建一棵树
24
25    def predict(self, X):
26        # 预测样本的结果
```

每个功能都和 ID3 决策树的构造算法一一对应，更详细的注释和代码实现可以查看提交的代码。

三　模型训练与测试

在完成了决策树分类器的代码编写之后，我使用 Kaggle 上的 Titanic 数据集对决策树的效果进行了测试。

3.1　数据集介绍

我使用的测试数据集是 Kaggle 上的公开数据集 Titanic，是一个二分类问题的数据集，用泰坦尼克号乘客的特征来预测乘客的存活情况，我仿照 Kaggle 讨论区公开的一些预处理方法对数据进行了简单的预处理和特征工程，最后得到的数据格式如下图所示：

	Pclass	Survived	Sex	Age	Parch	Fare	Embarked	Has_Cabin	FamilySize	IsAlone	Title
0	1	1	0	1	0	3	1	1	1	1	4
1	3	0	1	2	0	0	0	0	1	1	1
2	3	0	1	2	0	1	0	0	3	0	1
3	3	0	1	1	0	0	2	0	1	1	1
4	2	0	1	2	0	1	0	0	1	1	1

3.2　在 Titanic 数据集上的表现

3.2.1　模型准确率

我们使用划分好的训练集对模型进行训练之后，计算了模型在训练集和测试集上的准确率，得到的结果如下：

自己编写的决策树模型

```
# 使用自己编写的分类器测试数据集的准确性
from DecisionTree import DecisionTree
dt = DecisionTree(max_depth=2, min_samples_leaf=1)
dt.fit(X, y)
y_pred_train = dt.predict(X)
y_pred_test = dt.predict(X_test)
print("Accuracy on train data: {}".format(accuracy(y, y_pred_train)))
print("Accuracy on test data: {}".format(accuracy(y_test, y_pred_test)))
```

```
Accuracy on train data: 0.8013371537726839
Accuracy on test data: 0.8129770992366412
```

模型在训练集和测试集上的准确率分别是 80.1% 和 81.3%

3.2.2 召回率和查准率

在此基础上我们计算了模型在 Titanic 数据集上的召回率和查准率，得到的结果如下：

```
# 计算查准率和召回率
TP, FP, FN, TN = 0, 0, 0, 0
for i in range(y_test.shape[0]):
    if y_test[i] == 1 and y_pred_test[i] == 1:
        TP += 1
    elif y_test[i] == 1 and y_pred_test[i] == 0:
        FN += 1
    elif y_test[i] == 0 and y_pred_test[i] == 1:
        FP += 1
    else:
        TN += 1
P = TP / (TP + FP)
R = TP / (TP + FN)
print("precison: {}".format(P))
print("recall: {}".format(R))
```

```
precison: 0.8260869565217391
recall: 0.6063829787234043
```

模型的查准率约为 82.6%，而召回率约为 60.6%

3.3 与 sklearn 库的性能对比

下面我们调用 Python 的开源机器学习库 sklearn 中的决策树模型来进行训练和测试(这里我统一了决策树的分类标准和一些超参数，排除这些因素对结果的影响)，并计算其准确率，查准率和召回率，得到的结果如下：

```
from sklearn import tree
dt2 = tree.DecisionTreeClassifier(criterion='entropy', max_depth=2, min_samples_leaf=1)
dt2.fit(X, y)
y_pred_train = dt2.predict(X)
y_pred_test = dt2.predict(X_test)
print("Accuracy on train data: {}".format(accuracy(y, y_pred_train)))
print("Accuracy on test data: {}".format(accuracy(y_test, y_pred_test)))
```

```
Accuracy on train data: 0.7822349570200573
Accuracy on test data: 0.7709923664122137
```

```
TP, FP, FN, TN = 0, 0, 0, 0
for i in range(y_test.shape[0]):
    if y_test[i] == 1 and y_pred_test[i] == 1:
        TP += 1
    elif y_test[i] == 1 and y_pred_test[i] == 0:
        FN += 1
    elif y_test[i] == 0 and y_pred_test[i] == 1:
        FP += 1
    else:
        TN += 1
P = TP / (TP + FP)
R = TP / (TP + FN)
print("precison: {}".format(P))
print("recall: {}".format(R))
```

```
precison: 0.6847826086956522
recall: 0.6702127659574468
```

可以看到自己编写得到的决策树模型在相同的参数设定下,准确率和查准率要略好于 sklearn 中的决策树，而召回率低于 sklearn 中的决策树，这表明自己编写的决策树分类器出现了过拟合现象，而 sklearn 中防过拟合的措施做的比较好。

四　T 检验

我们使用 T 检验来判断自己编写的决策树模型和 sklearn 中的决策树模型的差距是否很大，这里我们选取 $\alpha = 0.05, k = 10$ 作为检验标准，通过将数据集进行十等分来进行 K-Fold 检验 (这一部分数据集的划分操作可以通过调用 sklearn 中的 model_selection.KFold 实现)，并分别得到两个模型的预测错误率 $\epsilon_i^{(1)}, \epsilon_i^{(2)} (i = 1, 2, \dots k)$，并计算每一组错误率的差值：

$$\Delta_i = \epsilon_i^{(2)} - \epsilon_i^{(1)} \tag{6}$$

然后计算出 $\Delta_i (i = 1, 2, \dots k)$ 的方差 μ 和均值 σ^2，并计算检验量的值：

$$\lambda_t = |\frac{\sqrt{k}\mu}{\sigma}| \tag{7}$$

我们计算得到的结果如下：

```
error = err2 - err1
check_t = np.sqrt(10 / np.var(error)) * np.mean(error)
# 计算T检验所需的关键变量
print(check_t)
```

```
3.690161492654147
```

然后查表得 $t_{\alpha/2,k-1} = 2.262$，我们计算得到的检验量超过了这个值，因此可以认为两个分类器具有显著的差异。

五　实验总结

本次实验中，我不仅学习了决策树算法的基本内容，还动手尝试了决策树的代码实现，但其实决策树也有多种形式比如 ID3,C4.5,CART 等等，目前我只实现了其中一种，此外还复习了之前学过的数据预处理的知识，并使用预处理后的数据集对决策树算法进行了训练和测试，此外还和 sklearn 中的决策树进行了 T 检验，通过 K 折交叉检验，验证了两个模型的差异性，可以说内容还是非常丰富的。