垃圾短信识别

1. 问题重述

本次的实验需要实现的是一个二分类模型。训练数据集为众多中文短信内容和短信是否为垃圾短信的分类结果。训练模型能满足，当读入新的短信时，可对此短信是否为垃圾短信进行判断。

2. logistics回归和决策树设计思想

2.1 logistics回归的设计思想

逻辑回归往往选择通过一个 sigmod 函数将样本映射到某个区间上，以此来估计样本属于某种类别的概率从而达到分类的目的。

假定我们对多维向量拟定的参数为ω，对数据集的表示用D。我们使用极大似然法来估计参数ω，并拥有极大似然法参数估计的损失函数E(ω)如下：

$$: \sum_{i \in I}(-y_i \omega^T x_i + \ln(1 + e^{\omega^T x_i}))$$

我们可以证明 E(ω) 是一个关于ω的凸函数，且括号内的式子为g(ω)也是关于ω的凸函数（因为g(ω)的二阶导为正）。因此，可以用梯度下降的方法求得其最优解，即：

$$\omega* = argminE(\omega)$$

因此，可以得到推导式：

$$\omega i + 1 = \omega i + \eta(i) X(\sigma(\omega i, X) - y)$$

其中的σ(x) 是 sigmod 函数，而 η(i) 是学习率。

2.2 决策树的设计思想

决策树 (Decision Tree)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。

下面简要介绍CART(classification and regression tree)树，涉及三个部分，分别是：基尼指数的特征选择，树的生成，剪枝。

基尼指数：假设数据集D中有K个类（二分类问题中K=2），样本属于第k类的概率为pk，基尼指数Gini(D)表示集合D的不确定性(纯度)，其涉及的公式如下：

$$G(p) = \sum_{k=0}^{K} pk(1 - pk)$$

当我们选择A作为分类特征（例如D1为A一维全为0的，D2为A一维全为1的），将D分成

$$G(D,A)=\sum_{i=1}^{2}\frac{|Di|}{|D|}G(Di)$$

树的生成：当我们遍历所有的可能分法后，可以选择基尼指数最小的分类特征A，将节点做一次二叉分类。

剪枝：为了防止模型发生过拟合，可以对"完全生长"的CART树底端剪去一些枝，使得决策树变小从而变得简单。

3. logistics回归和决策树代码内容

针对本题，下面讲解程序设计思路如下：

3.1 预处理部分

数据处理：

本题涉及数据转换与数据清洗两方面。数据转换上，牵涉到从文本到向量的算法，这里统一使用了TfidfVectorizer，基于词频和逆向文件频率来对文本进行提取。

数据清洗：

由于文章中有很多不重要的词汇，因此在做转换的时候要把这些词语剔除，也就是去除掉stopwords。为了提高准确率，这里我使用了中文停用词表, 哈工大停用词表, 百度停用词表, 四川大学机器智能实验室停用词库合并去重后的停词库。

此外，在拟合效果上，由于原训练集的正样本和负样本数目有着极大的不统一，负样本过多，因此我随机提取了和正样本数目一样多的负样本，生成了新的训练样本。

此外，还有对于分词数

数据清洗的代码如下：

```
data_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv"
sms = pd.read_csv(data_path, encoding='utf-8')

# 添加列用于计数分词数
cnt_vals = []
for i in sms['msg_new'].values:
    cnt_vals.append(len(i))
sms.insert(loc=3, column='cnt', value=cnt_vals)
# print(sms[cnt'].dtypes) #

#统一正负样本量
sms1 = sms[sms['label'] == 1]
bool_multi = ((sms['label'] == 0) & (sms['cnt'] >95) & (sms['cnt'] <115))
sms0 = sms[bool_multi].sample(n=len(sms1),replace=True)
sms = pd.concat([sms0,sms1], axis=0).reset_index(drop=True)
sms.to_csv('new_sms_pub.csv')

sms.groupby('label').describe() #
```

3.2 logistics代码

由于代码实现时，可以直接使用sklearn的函数，因此我们先给出代码，再讲解运行思路：

```
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords)),
    ('clf', LogisticRegression(penalty='l2', C=5.0, max_iter=10000,
solver='liblinear'))
])
```

pipeline第一步，读取数据，去除stopwords，再用TfidfVectorizer得到文本的向量。

pipeline第二步，进行训练，这里着重讲解选择的几个参数的意义：

- penalty：选择L2正则化（Ridge），避免参数过大，从而有效防止模型的过拟合问题
- C：用于控制正则化强度。C越小，则正则化强度越大，能够更好地防止过拟合；C越大，则正则化强度越小，模型能够更好地拟合数据。这里发现，当C较小时，得到的结果不好，因此增大了C的数值
- solver：liblinear是一个基于坐标轴下降法的算法，常用于二分类问题

3.3 决策树代码

```
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords)),
    ('clf',
DecisionTreeClassifier(min_samples_leaf=10,max_depth=200,min_samples_split=100))
])
```

pipeline第一步，读取数据，去除stopwords，再用TfidfVectorizer得到文本的向量。

pipeline第二步，进行训练，这里着重讲解选择的几个参数的意义：

- max_depth：决策树的最大深度，若节点的深度超过最大深度则剪枝，防止过拟合
- min_samples_split：一个节点能够做拆分时所需的最小样本数，默认值2。
- min_samples_leaf：叶子节点需要有的最小样本数

4. 实验结果呈现

当使用logistics算法时，得到的结果如下：

```
in stop_words.
  'stop_words.' % sorted(inconsistent))
在测试集上的混淆矩阵:
[[7693  180]
 [ 470 7487]]
在测试集上的分类结果报告:
              precision    recall  f1-score   support

           0       0.94      0.98      0.96      7873
           1       0.98      0.94      0.96      7957

    accuracy                           0.96     15830
   macro avg       0.96      0.96      0.96     15830
weighted avg       0.96      0.96      0.96     15830

在测试集上的 f1-score :
0.9583973374295955
```

当使用决策树算法时，得到的结果如下：

```
在测试集上的混淆矩阵：
[[7869    4]
 [ 215 7742]]
在测试集上的分类结果报告：
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      7873
           1       1.00      0.97      0.99      7957

    accuracy                           0.99     15830
   macro avg       0.99      0.99      0.99     15830
weighted avg       0.99      0.99      0.99     15830

在测试集上的 f1-score ：
0.9860536203273259
```

用例测试

测试点	状态	时长	结果
测试模型预测结果	✓	1s	通过测试，训练的分类器具备检测恶意短信的能力，分类正确比例:8/10
测试读取停用词库函数结果	✓	1s	read_stopwords 函数返回的类型正确

5. 总结

Logistic Regression 和决策树都是常用的分类算法，它们在原理、实现和性能等方面有着不同的异同点。

相同点：

1. 都可以处理二分类和多分类问题。
2. 都可用于特征选择和降维。
3. 都可以通过正则化和剪枝等方式来避免过拟合。

不同点：

1. 原理：Logistic Regression 是一种基于概率模型的分类算法，通过最大似然估计来确定模型的参数；而决策树是一种基于树形结构的分类算法，通过判断样本特征的取值来进行分类。
2. 实现：Logistic Regression 的实现比较简单，只需要求解一个凸优化问题即可得到模型参数；而决策树的实现比较复杂，需要考虑如何选择最佳的分裂特征和分裂点等问题。
3. 性能：Logistic Regression 的优点在于它通常比决策树计算速度更快，对于大规模数据集的分类问题，Logistic Regression 可以更好地发挥其优势；而决策树的优点在于它能够处理非线性决策边界和高维特征空间，并且易于解释和可视化。在本题中，决策树展现了较高的性能，我对Logistic Regression的参数做了许多调整，却依旧保持在0.6的正确率，但是我对决策树的参数进行了几次调整后，则有了较大的提示。