本科实验报告

课程名称：	人工智能
实验名称：	朴素 Bayes 方法
专业名称：	保密管理
学生姓名：	武自厚
学生学号：	20336014
实验地点：	东校园实验楼 D502
实验成绩：	
报告时间：	2022 年 5 月 10 日

一、 实验要求

在给定的文本数据集完成文本情感分类训练，在测试集完成测试，计算准确率。

- 文本的特征可以使用 TF-IDF, 对 TF 均使用拉普拉斯平滑技巧 (可以使用 `sklearn` 库提取特征).
- 利用朴素 Bayes 完成对测试集的分类, 并计算准确率.

二、 实验过程

1. 朴素 Bayes 的数学原理

朴素 Bayes 是一种基于 Bayes 定理以及假设特征条件相互独立的分类方法. 在该算法中, 我们先使用一些带有标签 C_i 和特征 x_j 的数据去训练模型 $\boldsymbol{x}$, 并且准备一个测试样例 $\boldsymbol{x'} = (x'_0, \cdots, x'_{n-1})$. 根据具体问题得到某分类下遇到某特征的条件概率 $\Pr(x_j = x'_j | y = C_i)$, 这个值可以根据观测的频数近似.

由于朴素 Bayes 的一大假设就是特征条件相互独立, 那么可以得到

$$\Pr(\boldsymbol{x} = \boldsymbol{x'} | y = C_i) = \prod_{j=1}^{n} \Pr(x_j = x'_j | y = C_i)$$

根据 Bayes 定理, 得到已知测试样例特征时, 其标签为特定值的概率.

$$\Pr(y = C_i | \boldsymbol{x} = \boldsymbol{x'}) = \frac{\Pr(y = C_i) \cdot \Pr(\boldsymbol{x} = \boldsymbol{x'} | y = C_i)}{\Pr(\boldsymbol{x} = \boldsymbol{x'})}$$

最后计算不同标签的概率, 选择概率最大的标签作为 $\hat{y}$ 返回. 另外由于对于每一种标签,$\Pr(\boldsymbol{x} = \boldsymbol{x'})$ 都是一样的, 所以可以直接忽略掉它的影响.

$$\begin{aligned}
\hat{y} &= \arg_{C_i} \Pr(y = C_i | \boldsymbol{x} = \boldsymbol{x'}) \\
&= \arg_{C_i} \frac{\Pr(y = C_i) \cdot \Pr(\boldsymbol{x} = \boldsymbol{x'} | y = C_i)}{\Pr(\boldsymbol{x} = \boldsymbol{x'})} \\
&= \arg_{C_i} \Pr(y = C_i) \cdot \Pr(\boldsymbol{x} = \boldsymbol{x'} | y = C_i) \\
&= \arg_{C_i} \Pr(y = C_i) \cdot \prod_{j=1}^{n} \Pr(x_j = x'_j | y = C_i)
\end{aligned}$$

2. 文本特征 TF

由上面的推导过程我们可以根据 $\Pr(x_j = x'_j | y = C_i)$ 推导出 $\hat{y}$, 而如何求出 $\Pr(x_j = x'_j | y = C_i)$ 则是在不同领域有着不同方法的问题. 在文本处理方面可以使用文字频率 (TF) 来估算 $\Pr(x_j = x'_j | y = C_i)$.

给定训练集 $\boldsymbol{X}$, 包含带有情感标签 C_k 的训练文本 $\boldsymbol{x_k}$. TF 可以定义为 “特定情感标签的文本下特定关键词出现的频率”, 理论上这个值就代表着我们需要的概率, 但是它还没有归一化处理, 所以需要如下步骤:

简单起见设 C_i 情感标签文本下关键词 x_j 出现的频率为 $\mathrm{tf}_{i,j}$, 那么条件概率可以用频率除以频数得到:

$$\Pr(x_j = x'_j | y = C_i) = \frac{\mathrm{tf}_{i,j}}{\sum_k \mathrm{tf}_{i,k}}$$

然而 $\mathrm{tf}_{i,j}$ 可能为 0 导致估算结果被完全否定, 这里采用平滑技术, 也就是

$$\Pr(x_j = x'_j | y = C_i) = \frac{\mathrm{tf}_{i,j} + \lambda}{\sum_{k=1}^{m} \mathrm{tf}_{i,k} + \lambda \cdot m}$$

m 为特征词的数量. 取 $\lambda = 1$(称为 Laplace 平滑), 这样就可以正确处理含 0 测试集.

3. 关键代码

代码中采用 `sklearn` 库中的 `CountVectorizer` 来提取语料库. 具体预测代码如下:

```
def predict_unit(self, words):
    # 学霸题，预测标签值
    temp = []
    # 遍历单词语
    for word in words:
        if self.vocabulary.get(word):
            # 获取词频列
            temp.append(self.x_train[:,self.vocabulary[word]])
        else:
            # 没有加个零
            temp.append(np.zeros(6))
    # 矩阵叠起来
    tf = np.vstack(temp)
    # 特征词数量，特征总频率
    size = np.sum(tf, axis=1, keepdims=True) + np.size(tf, axis=0)
    # 拉氏来平滑，加一除上去
    p_unit = (tf + 1) / size
    # 结果乘起来
    p = np.prod(p_unit, axis=0, keepdims=True) * self.emotion_size
    # 最大看出来
    return np.argmax(p)
    # 你学会了吗
```

三、 实验结果

采用 246 个句子的训练集以及 1000 个句子的测试集来测试代码. 先后展示测试集样本中真实以及预测的情感标签, 再计算出预测成功率.

```
Real Label:
[3 4 3 3 2 4 5 4 5 4 5 5 5 3 5 3 5 3 4 5 5 5 0 1 3 5 3 5 1 5 3 5 3 2 3 5 4
 2 3 0 3 5 4 3 2 1 4 4 0 0 3 2 4 5 2 5 3 0 3 3 5 0 2 2 3 4 3 3 2 3 4 5 3 5
 2 3 1 3 2 4 3 0 2 2 5 4 3 5 3 4 2 2 3 5 2 2 5 3 2 3 5 2 0 2 3 4 2 3 4 2 2
 5 3 0 3 0 0 3 3 5 4 1 5 5 2 3 5 3 3 4 5 3 2 4 5 3 4 3 5 4 2 5 3 0 1 4 5 3
 2 3 4 5 4 5 3 0 2 2 2 4 5 5 4 5 4 5 3 2 4 4 0 5 3 5 3 3 3 4 4 2 3 5 3 2 5
 3 5 5 3 2 4 4 5 4 4 3 4 2 2 5 5 0 3 5 3 2 2 5 0 4 3 3 4 4 3 3 0 0 4 0 3 3
 5 2 4 3 4 5 3 3 3 3 0 4 2 3 4 3 0 5 5 5 4 5 2 4 3 3 4 5 3 3 4 4 4 2 4 1 2
```

```
3 3 4 3 5 3 3 4 2 3 3 2 2 3 4 4 3 0 3 4 2 5 4 2 3 3 3 2 3 4 1 2 2 2 4 3 5
3 4 4 5 3 3 2 4 5 2 4 5 3 4 2 4 4 2 3 0 4 4 4 0 3 2 0 0 3 3 5 5 2 3 4 3 2
4 4 3 4 3 3 3 0 2 5 0 2 2 2 5 5 3 2 2 3 5 5 2 4 0 3 3 3 1 2 2 3 5 3 2 5 3
4 4 2 3 4 3 4 3 3 3 2 2 4 3 3 2 4 4 5 2 3 0 3 3 3 2 4 2 2 3 4 5 4 2 1 5 4
0 2 5 3 3 0 3 4 4 5 2 3 3 4 4 3 3 3 2 3 5 3 4 5 3 3 3 3 2 3 4 3 2 2 3 4 4
4 4 5 3 3 2 3 5 3 3 3 3 3 2 0 3 3 3 3 3 5 0 4 3 0 5 2 3 5 4 0 2 3 1 4 3 5
5 5 4 4 3 3 0 3 1 5 2 3 5 4 3 3 3 0 3 2 5 3 3 2 2 2 2 0 4 2 0 2 3 2 3 3 0
2 4 5 5 5 3 5 3 5 5 5 1 3 4 2 3 2 2 3 3 2 4 4 2 4 4 1 4 4 1 3 4 1 0 3 3 3
2 5 2 2 2 2 3 2 4 3 5 0 3 3 4 4 3 3 3 3 0 4 3 2 5 5 3 2 3 3 3 2 2 2 5 5 3
3 2 2 4 4 2 4 0 0 2 2 3 3 3 3 2 5 3 5 2 4 5 5 5 2 3 3 3 0 3 5 0 4 2 5 3 3
4 3 3 3 3 4 3 0 3 4 3 3 5 0 4 5 3 3 3 5 5 4 3 3 3 3 3 4 2 3 4 4 4 3 3 3 4
3 5 0 5 3 3 3 4 5 4 3 3 2 3 4 1 2 0 3 2 5 0 3 3 5 2 5 5 3 3 3 3 5 4 3 2 3
3 4 3 3 4 3 2 3 3 2 3 3 3 4 5 5 3 4 3 4 3 3 3 3 5 4 4 3 5 3 3 2 4 3 3 3 3
0 4 5 2 4 5 3 5 3 5 4 3 5 4 3 3 2 2 4 3 2 4 5 3 4 3 2 3 3 3 3 3 5 3 3 4 3
3 4 3 4 4 2 1 3 3 4 5 3 3 4 3 0 4 3 4 5 3 3 3 5 4 3 4 5 4 1 5 5 4 5 5 5 3
0 5 4 2 3 4 4 4 3 4 5 3 3 3 2 3 5 2 5 3 2 3 3 5 5 3 4 2 5 2 2 2 4 5 3 3 5
3 2 5 1 3 5 3 2 3 5 3 2 4 5 5 4 3 3 3 3 4 2 4 4 3 4 3 4 4 0 2 4 3 5 3 0 3
4 4 3 0 3 4 5 1 1 3 3 3 3 4 3 4 5 0 5 3 3 0 3 3 3 3 4 2 4 1 3 3 5 5 2 3 3
4 2 4 3 5 4 2 2 3 5 5 5 3 0 2 0 4 3 4 3 3 5 3 5 4 2 3 5 2 3 1 5 3 3 4 0 5
5 5 3 3 4 3 2 3 5 3 1 3 3 3 4 5 4 5 5 0 1 3 5 4 4 3 3 4 0 4 4 3 5 4 4 5 5
3]
Predict Label:
[2 4 3 3 4 4 3 4 3 3 3 4 4 3 3 4 5 4 4 3 3 4 3 3 3 3 4 3 4 4 3 3 3 4 3 3 3
 3 3 3 4 3 3 3 4 3 3 3 4 3 3 4 4 4 3 3 2 2 3 3 3 4 3 3 4 4 3 3 3 3 4 3 3 4
 3 3 3 4 4 3 3 3 2 4 3 4 3 3 3 4 4 4 4 3 3 3 4 3 3 4 3 4 3 4 4 4 4 3 3 4 3
 3 4 3 3 3 4 3 3 4 3 3 3 3 4 3 3 2 3 4 3 3 4 3 4 3 4 3 3 3 4 5 3 4 3 4 4 2
 3 3 4 3 3 3 4 3 4 3 4 4 3 4 4 4 2 3 3 3 3 3 4 3 3 4 3 3 4 3 4 3 3 3 4 4 3 3
 3 3 4 4 2 4 3 4 3 4 3 4 3 4 3 4 4 4 4 3 3 4 3 3 4 4 3 4 4 3 3 3 3 4 4 4 3
 3 3 4 4 4 3 3 4 3 3 4 3 2 4 3 3 3 3 2 4 4 3 4 4 4 4 3 3 4 3 3 3 4 4 4 3 4
 5 3 3 3 3 4 3 4 3 3 3 3 3 4 3 4 3 4 3 3 3 3 4 4 3 3 4 4 3 3 3 3 4 2 4 3 4
 3 3 3 3 3 4 4 4 4 4 4 3 4 5 4 4 3 3 4 3 4 4 4 4 4 3 3 4 3 5 3 3 3 2 3 4 3 3
 3 4 4 3 3 3 4 3 4 3 4 2 4 5 4 3 3 3 4 2 4 3 3 4 4 3 3 4 3 3 3 3 3 4 3 4 5 4
 3 4 4 3 3 3 4 4 3 3 3 4 3 3 3 4 3 4 4 4 3 3 3 3 3 4 3 3 3 3 3 3 3 4 3 3 4
 3 4 3 4 4 4 3 3 3 3 3 3 3 4 4 4 4 3 4 3 3 3 4 4 3 3 2 3 3 3 4 3 3 3 3 3 4
 4 3 4 3 3 3 4 3 3 3 3 3 4 2 2 4 3 4 4 3 3 4 3 3 4 4 4 3 4 3 3 4 3 3 3 3 3 4
 3 3 4 2 3 3 4 5 3 3 3 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 3 3 4 3 4 2 3 4
 2 3 3 3 3 3 3 3 4 4 3 3 4 3 3 3 2 4 4 3 2 2 4 4 4 4 4 3 4 4 3 3 4 4 3 3 3
 4 4 2 3 3 3 2 2 3 3 3 4 3 3 4 4 2 3 3 4 3 3 3 4 4 3 4 3 4 3 4 3 4 4 4 3 4 4
 4 4 4 4 2 3 4 4 4 4 3 4 4 3 3 3 4 3 4 4 3 4 3 3 3 4 3 3 3 3 4 3 4 4 4 4 3 4 4
 3 4 5 3 3 4 3 3 3 3 4 4 4 3 3 3 5 3 3 4 3 4 3 3 3 3 4 3 3 3 3 4 4 3 3 4 3
 3 3 3 4 3 3 3 3 3 4 2 4 4 3 4 3 3 2 4 3 3 3 3 4 4 2 3 3 3 4 4 3 4 3 3 3 4
 4 4 4 3 3 4 4 3 3 4 4 4 3 3 3 3 3 3 3 4 3 3 4 3 3 3 3 3 3 4 3 4 3 3 3 3 3
 3 4 4 3 3 3 3 4 3 4 4 3 3 4 2 3 4 3 4 3 3 4 3 5 4 3 3 2 3 4 3 2 3 3 4 3 3
 3 4 3 3 3 4 4 3 2 4 3 3 3 4 3 4 3 3 3 3 3 3 3 4 3 4 4 4 4 3 3 3 4 4 4 3 4
 3 4 4 4 3 3 0 4 3 3 4 3 3 4 3 4 3 3 3 3 4 3 3 4 4 3 2 3 4 4 4 4 4 3 4 3 4
 4 4 4 3 5 3 3 4 3 3 4 4 4 4 3 4 4 3 3 3 4 4 4 4 3 3 3 4 3 3 3 3 3 3 3 3 3
 4 3 3 2 3 4 3 3 3 3 3 3 3 3 4 4 3 3 3 3 3 4 4 3 3 3 4 4 4 3 2 3 3 4 3 3 4
 4 4 3 3 3 3 4 2 4 4 4 3 3 3 1 4 5 3 3 4 3 3 3 3 5 3 3 4 3 3 3 4 4 4 3 4 3 3
 3 3 3 3 4 3 3 4 3 3 3 3 4 4 3 3 3 3 3 2 3 4 3 4 3 4 3 3 3 4 3 4 3 3 3 4 4 4
 3]
```

```
Accuracy: 0.374

进程已结束,退出代码0
```

最后得出成功率为 0.374, 符合朴素 Bayes 分类器的一般效果.