Bayes

1. 实验目的

了解朴素贝叶斯算法的原理，并且可以简单应用

2. 算法原理

1）介绍

在所有的机器学习分类算法中，朴素贝叶斯和其他绝大多数的分类算法都不同。对于大多数的分类算法，比如决策树,KNN,逻辑回归，支持向量机等，他们都是判别方法，也就是直接学习出特征输出 Y 和特征 X 之间的关系，要么是决策函数 Y=f(X),要么是条件分布 P(Y|X)。但是朴素贝叶斯却是生成方法，也就是直接找出特征输出 Y 和特征 X 的联合分布 P(X,Y),然后用 P(Y|X)=P(X,Y)/P(X)得出。

2）相关统计学知识

在了解朴素贝叶斯的算法之前，我们需要对相关必须的统计学知识做一个回顾。贝叶斯学派很古老，但是从诞生到一百年前一直不是主流。主流是频率学派。频率学派的权威皮尔逊和费歇尔都对贝叶斯学派不屑一顾，但是贝叶斯学派硬是凭借在现代特定领域的出色应用表现为自己赢得了半壁江山。

贝叶斯学派的思想可以概括为先验概率+数据=后验概率。也就是说我们在实际问题中需要得到的后验概率，可以通过先验概率和数据一起综合得到。数据大家好理解，被频率学派攻击的是先验概率，一般来说先验概率就是我们对于数据所在领域的历史经验，但是这个经验常常难以量化或者模型化，于是贝叶斯学派大胆的假设先验分布的模型，比如正态分布，beta 分布等。这个假设一般没有特定的依据，因此一直被频率学派认为很荒谬。虽然难以从严密的数学逻辑里推出贝叶斯学派的逻辑，但是在很多实际应用中，贝叶斯理论很好用，比如垃圾邮件分类，文本分类。

我们先看看条件独立公式，如果 X 和 Y 相互独立，则有：

$P(X,Y)=P(X)P(Y)$

我们接着看看条件概率公式：

$P(Y|X)=P(X,Y)/P(X)$

$P(X|Y)=P(X,Y)/P(Y)$

或者说:

$P(Y|X)=P(X|Y)P(Y)/P(X)$

接着看看全概率公式

$P(X)=\sum kP(X|Y=Yk)P(Yk)$

其中$\sum kP(Yk)=1$

从上面的公式很容易得出贝叶斯公式：

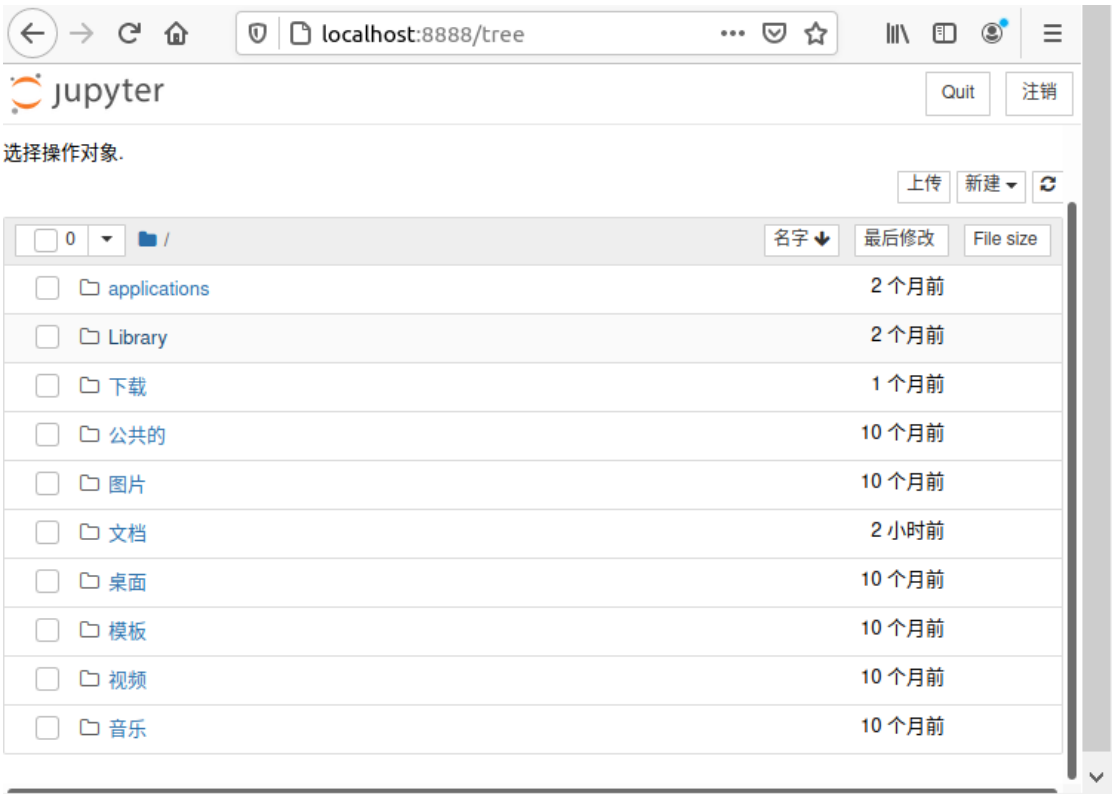
$P(Yk|X)=P(X|Yk)P(Yk)\sum kP(X|Y=Yk)P(Yk)$

3. 实验环境

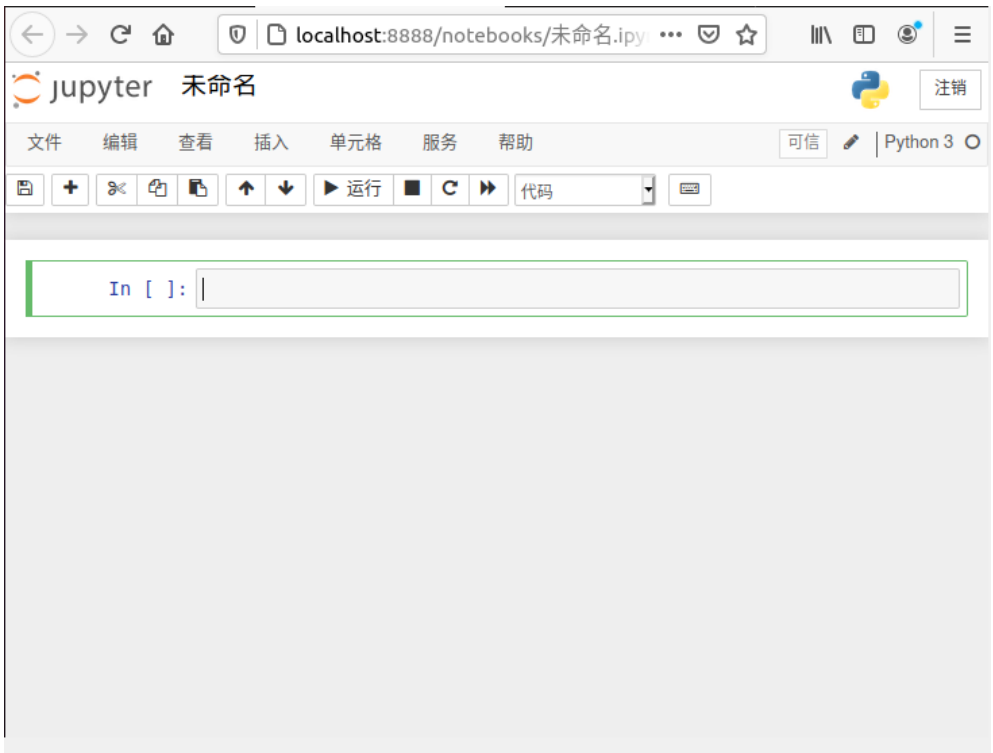
Ubuntu 20.04
Python 3.6
Jupyter notebook

4. 实验步骤

1)打开终端，然后输入 jupyter notebook，出现如下界面



2)选定特定文件夹，新建 ipynb 文件，在未命名出可重命名文件



5. 实操

Step 1:数据预处理

1. 导入库
2. 导入文件
3. 切分词
4. 加载训练集与测试集

```
#导入库
import os
import jieba
import warnings
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

#导入文件
def loadfile(file_dir, label):
    """
    将路径下的所有文件加载
    :参数 file_dir: 保存 txt 文件目录
    :参数 label: 文档标签
    :return: 分词后的文档列表和标签
    """
    file_list = os.listdir(file_dir)
    words_list = []
    labels_list = []
    for file in file_list:
        file_path = file_dir + '/' + file
        words_list.append(cut_words(file_path))
        labels_list.append(label)


    return words_list, labels_list



# 切分词
def cut_words(file_path):
    """
    对文本进行切词
    :参数 file_path: txt 文本路径
    :return: 用空格分词的字符串
    """
    text_with_spaces = ''
    text=open(file_path, 'r', encoding='gb18030').read()
    textcut = jieba.cut(text)
```

```
    for word in textcut:
        text_with_spaces += word + ' '
    return text_with_spaces


train_words_list1, train_labels1 = loadfile('D:/jupyter_notebook/ml_full/5.5bayes/text classification/train/女性', '女性')
train_words_list2, train_labels2 = loadfile('D:/jupyter_notebook/ml_full/5.5bayes/text classification/train/体育', '体育')
train_words_list3, train_labels3 = loadfile('D:/jupyter_notebook/ml_full/5.5bayes/text classification/train/文学', '文学')
train_words_list4, train_labels4 = loadfile('D:/jupyter_notebook/ml_full/5.5bayes/text classification/train/校园', '校园')

train_words_list = train_words_list1 + train_words_list2 + train_words_list3 + train_words_list4
train_labels = train_labels1 + train_labels2 + train_labels3 + train_labels4
                                                    stratify=y,
test_words_list1, test_labels1 = loadfile('text classification/test/女性', '女性')
test_words_list2, test_labels2 = loadfile('text classification/test/体育', '体育')
test_words_list3, test_labels3 = loadfile('text classification/test/文学', '文学')
test_words_list4, test_labels4 = loadfile('text classification/test/校园', '校园')

test_words_list = test_words_list1 + test_words_list2 + test_words_list3 + test_words_list4
test_labels = test_labels1 + test_labels2 + test_labels3 + test_labels4

stop_words = open('text classification/stop/stopword.txt', 'r', encoding='utf-8').read()
stop_words = stop_words.encode('utf-8').decode('utf-8-sig') # 列表头部\ufeff处理
stop_words = stop_words.split('\n') # 根据分隔符分隔
                                                    random_state=0)
```

Step 2:朴素贝叶斯模型

```
# 计算单词权重
```

```
tf = TfidfVectorizer(stop_words=stop_words, max_df=0.5)

train_features = tf.fit_transform(train_words_list)
# 上面 fit 过了，这里 transform
test_features = tf.transform(test_words_list)

# 多项式贝叶斯分类器
clf = MultinomialNB(alpha=0.001).fit(train_features, train_labels)
predicted_labels=clf.predict(test_features)

# 计算准确率
print('准确率为:
', metrics.accuracy_score(test_labels, predicted_labels))
```