朴素贝叶斯分类器实现

1　数据集选择

皮马印第安人糖尿病数据库

文件diabetes.csv包含9个字段，768行，每一行代表在几个医学预测变量和一个目标变量情况下，是否患有糖尿病的结果，具体信息如下：

序号	字段名	数据类型	字段描述
1	Pregnanci	Integer	怀孕次数
2	Glucose	Integer	口服葡萄糖耐量试验中血浆葡萄糖浓度为2小时
3	BloodPressure	Integer	舒张压（mm Hg）
4	SkinThickness	Integer	肱三头肌皮褶厚度（mm）
5	Insulin	Integer	2小时血清胰岛素（μU/ ml）
6	BMI	Integer	体重指数（体重kg /（身高m）^ 2）
7	DiabetesPedigreeFunction	Integer	糖尿病谱系功能
8	Age	Integer	年龄（岁）
9	Outcome	Integer	类变量（0或1）

2　编程实现

代码过程分6部分：

1. 按类别划分样本（映射成字典）
2. 提取属性特征（对一个类的所有样本，计算每个属性的均值和方差）
3. 按类别提取属性特征（存在字典）；计算高斯概率密度函数，计算样本的某一属性x的概率，归属于某个类的似然
4. 对一个预测样本，计算它属于每个类的概率（仍存在字典）
5. 预测样本（找到最大概率值的类）
6. 计算准确率、精确率、召回率、F1 值

2.1　按类别划分样本

```
# separated = {0: [[att1, att2, ... att8, 0], ...],
#              1: [[att1, att2, ... att8, 1], ...]}
def separate_by_class(dataset):
    separated = {}
    for i in range(len(dataset)):
        vector = dataset[i]
        if (vector[-1] not in separated):
            separated[vector[-1]] = []
        separated[vector[-1]].append(vector)
    return separated
```

2.2　提取属性特征

```
# 对一个类的所有样本,计算每个属性的均值和方差,
# summaries = [(att1_mean,att1_stdev), ...]
def mean(numbers):
    return sum(numbers) / float(len(numbers))


def stdev(numbers):
    avg = mean(numbers)
    variance = sum([pow(x - avg, 2) for x in numbers]) /
float(len(numbers) - 1)
    return math.sqrt(variance)


def summarize(dataset):
    summaries = [(mean(attribute), stdev(attribute)) for
attribute in zip(*dataset)]
    del summaries[-1]
    return summaries
```

2.3 按类别提取属性特征

```
# summaries = {0:[(att1_mean,att1_stdev), ...],
#              1:[(att1_mean,att1_stdev), ...]}
def summarize_by_class(dataset):
    separated = separate_by_class(dataset)
    summaries = {}
    keyList = list(separated.keys())
    for classValue in keyList:
        summaries[classValue] = summarize(separated[classValue])
    return summaries


# 计算高斯概率密度函数. 计算样本的某一属性x的概率,归属于某个类的似然
def calculate_probability(x, mean, stdev):
    exponent = math.exp(-(math.pow(x - mean, 2) / (2 *
math.pow(stdev, 2))))
    return (1 / (math.sqrt(2 * math.pi) * stdev)) * exponent

```

2.4 对一个预测样本，计算它属于每个类的概率

```
def calculate_class_probabilities(summaries, inputVector):
    probabilities = {}
    keyList = list(summaries.keys())
    for classValue in keyList:
        probabilities[classValue] = 1
        for i in range(len(summaries[classValue])):  # 属性个数
            mean, stdev = summaries[classValue][i]  # 训练得到的第i
个属性的提取特征
            x = inputVector[i]  # 测试样本的第i个属性x
            probabilities[classValue] *= calculate_probability(x,
mean, stdev)
    return probabilities
```

2.5 预测样本

```python
# 单个数据样本的预测. 找到最大的概率值,返回关联的类
def predict(summaries, inputVector):
    probabilities = calculate_class_probabilities(summaries,
inputVector)
    bestLabel, bestProb = None, -1
    keyList = list(probabilities.keys())
    for classValue in keyList:
        if bestLabel is None or probabilities[classValue] >
bestProb:
            bestProb = probabilities[classValue]
            bestLabel = classValue
    return bestLabel


# 多个数据样本的预测
def get_predictions(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions
```

2.6 计算准确率、精确率、召回率、F1 值

```python
# 计算精度
def get_accuracy(testSet, predictions):
    correct = 0
    for x in range(len(testSet)):
        if testSet[x][-1] == predictions[x]:
            correct += 1
    return (correct / float(len(testSet)))


# 计算精确率
def get_precision(testSet, predictions):
    truePositive = 0
    positive = 0
    for x in range(len(testSet)):
        if predictions[x] == 1:
            positive += 1
            if testSet[x][-1] == predictions[x]:
                truePositive += 1
    return (truePositive / positive)


# 计算召回率
def get_recall(testSet, predictions):
    truePositive = 0
    positive = 0
    for x in range(len(testSet)):
        if testSet[x][-1] == 1:
            positive += 1
            if testSet[x][-1] == predictions[x]:
                truePositive += 1
    return (truePositive / positive)
```

2.7 数据集的划分与合并

借助train_test_split()函数对数据集进行划分，将数据随机划分成2/3训练集和1/3测试集，目的是保证性能对比结果是基于相同的训练集和测试集得出。

```
# 读取数据
    filename = 'pima-indians-diabetes.csv'
    dataset = pd.read_csv(filename, header=None)
    dataset = dataset.sample(frac=1.0)
    y = dataset[8]
    X = dataset[[0, 1, 2, 3, 4, 5, 6, 7]]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 1 / 2, random_state=1, stratify=y)
    dataset = np.array(dataset)
    train = pd.concat([X_train, y_train], axis=1)
    test = pd.concat([X_test, y_test], axis=1)
    train = np.array(train)
    test = np.array(test)

    # 随机划分数据:1/2训练和1/2测试
    trainSize = int(len(train))
    randomIdx = [i for i in range(len(train))]
    trainSet = []
    testSet = []
    trainSet.extend(train[idx, :] for idx in randomIdx[:trainSize])
    testSet.extend(test[idx, :] for idx in randomIdx[:trainSize])
```

3 性能对比

为便于后续T检验，选择进行5次2折交叉验证，并计算十次结果的各项平均值作为性能对比。

自实现分类器各次准确率、精确率、召回率、F1 值如下（最后一行为平均值）：

准确率	精确率	召回率	F1值
0.739583	0.603659	0.738806	0.664430
0.760417	0.661538	0.641791	0.651515
0.721354	0.584906	0.694030	0.634812
0.75	0.633803	0.671642	0.652174
0.721354	0.579882	0.731343	0.646865
0.763021	0.674796	0.619403	0.645914
0.75	0.625	0.708955	0.664336
0.755208	0.636986	0.694030	0.664286
0.78125	0.686567	0.686567	0.686567
0.703125	0.556818	0.731343	0.632258
0.744531	0.624396	0.691791	0.654316

Scikit-Learn中朴素贝叶斯算法分类器平均准确率、精确率、召回率、F1 值如图（最后一行为平均值）：

准确率	精确率	召回值	F1值
0.752604	0.646617	0.641791	0.644195
0.757813	0.691589	0.552239	0.614108
0.736980	0.625954	0.611940	0.618868
0.757813	0.675214	0.589552	0.629482
0.75	0.635714	0.664179	0.649635
0.742188	0.666667	0.522388	0.585774
0.752604	0.658537	0.604478	0.630350
0.757813	0.664	0.619403	0.640927
0.773438	0.715596	0.582090	0.641975
0.695313	0.556291	0.626866	0.689473
0.747657	0.653618	0.611493	0.634479

分析如下：

1. 整体的准确率上，两者相差不大，均超过七成，即总的预测结果准确率相近。
2. 在本数据集中，精确率的含义为“被正确预测患病的人(TP)”占所有"被预测患病的人(TP+FP)"的比例。Scikit-Learn中朴素贝叶斯算法分类器的精确率要稍高于自实现分类器，说明Scikit-Learn中朴素贝叶斯算法分类器所预测的患病病例的正确率较高。
3. 在本数据集中，召回率的含义为“被预正确测患病的人(TP)”占所有"实际患病的人(TP+FN)"的比例。自实现分类器的召回率要显著高于Scikit-Learn中朴素贝叶斯算法分类器，说明自实现分类器能够预测出更多的患病病例。考虑到本数据集的实际意义，在预测患病的现实情况中，召回率高比精确率高更加重要。
4. F1值是精确率和召回率的综合，F1值越高，说明分类模型越稳健。总的来说，两种分类器的F1值相近，自实现分类器的F1值略高。

4 T检验

5x2交叉验证法：

5x2交叉验证是做5次2折交叉验证，在每次2折交叉验证之前随机将数据打乱，使得5次交叉验证中的数据划分不重复。对两个分类器A和B，第i次2折交叉验证将产生两对测试错误率，我们对它们分别求差，得到第1折上的差值 Δ_i^1 和第2折上的差值 Δ_i^2 。为缓解测试错误率的非独立性，我们仅计算第1次2折交叉验证的两个结果的平均值 μ= 0.5($\Delta_i^1 + \Delta_i^2$)，但对每次2折实验的结果都计算出其方差 $\sigma_i^2 = \left(\Delta_i^1 - \frac{\Delta_i^1 + \Delta_i^2}{2}\right)^2 + \left(\Delta_i^2 - \frac{\Delta_i^1 + \Delta_i^2}{2}\right)^2$ 变量 $\tau_t = \frac{\mu}{\sqrt{0.2\sum_{i=1}^{5}\sigma_i^2}}$ 服从自由度为5的t分布，其双边检验的临界值 $t_{\alpha/2,5}$ 当α= 0.05 时为 2.5706， α= 0.1 时为 2.0150。

准确率A	准确率B	Δ_i^1	准确率A	准确率B	Δ_i^2
0.739583	0.752604	0.013021	0.760417	0.757813	0.002604
0.721354	0.736980	0.015626	0.75	0.757813	0.007813
0.721354	0.75	0.028646	0.763021	0.742188	0.020833
0.75	0.752604	0.002604	0.755208	0.757813	0.002605
0.78125	0.773438	0.007812	0.703125	0.695313	0.007812

$\mu = 0.5(\ \Delta_i^1 + \Delta_i^2\ ) = 0.007813$

$$\tau_t = \frac{\mu}{\sqrt{0.2\sum_{i=1}^{5}\sigma_i^2}} = 1.627$$

$\alpha = 0.1$的情况下，由T检验相关知识，我们认为两种分类器性能差异并不显著。