云南大学数学与统计学院
《数据挖掘与决策支持实验》上机实践报告

课程名称： 数据挖掘与决策支持实验	**年级：** 2015 级	**上机实践成绩：**
指导教师： 彭程	**姓名：** 刘鹏	**专业：** 信息与计算科学
上机实践名称： 对 iris 数据进行贝叶斯分类	**学号：** 20151910042	**上机实践日期：** 2018-07-05
上机实践编号： 04	**组号：**	

一、实验目的

学习使用 R 语言进行变量选择。

二、实验内容

对 iris 数据进行贝叶斯分类。

三、实验平台

Windows 10 Pro 1803；

Microsoft© Visual Studio 2017 Enterprise。

Version 1.1.442 – © 2009-2018 RStudio, Inc.

四、算法设计

贝叶斯方法是一种分类法。数据并不是总体或待建模系统的唯一可用的信息资源。贝叶斯方法提供了一套将这些外部信息融入数据分析过程的原理方法。这个过程先给出待分析数据集的概率分布。因为这个分布在给出时没有考虑任何数据，所以称为先验分布（prior distribution）。新的数据集将先验分布修正后得到后验分布（posterior distribution）。进行这个修正的基本工具就是贝叶斯定理。

设X是一个类标号未知的数据样本，H为某种假定：数据样本X属于某特定的类C。要求确定$P(H|X)$，即给定了观测数据样本X，假定H的概率。其实这就是贝叶斯公式的一种具体形式。$P(H|X)$表示给定数据集X之后，我们对假设H成立的后验概率。后验概率$P(H|X)$比先验概率$P(H)$含有更多的信息。

贝叶斯定理可以表示如下：

$$\mathrm{P(H|X)} = \frac{\mathrm{P(X|H)} \cdot \mathrm{P(H)}}{\mathrm{P(X)}}$$

五、程序代码

5.1 程序描述

5.2 程序代码

```
# Load libraries
import pandas
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
from sklearn import model_selection  # 模型比较和选择包
from sklearn.naive_bayes import GaussianNB


class Bayes_Test():
    # 读取样本 数据集
    def load_dataset(self):
        url = 'Iris.csv'
        names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
        dataset = pandas.read_csv(url, names=names)
        return dataset

    # 提取样本特征集和类别集 划分训练/测试集
    def split_out_dataset(self, dataset):
        array = dataset.values  # 将数据库转换成数组形式
        X = array[:, 0:4].astype(float)  # 取特征数值列
        Y = array[:, 4]  # 取类别列
        validation_size = 0.20  # 验证集规模
        seed = 7
        # 分割数据集 测试/验证
        X_train, X_validation, Y_train, Y_validation = \
            model_selection.train_test_split(X, Y, test_size=validation_size, \
            random_state=seed)
        return X_train, X_validation, Y_train, Y_validation

    """第一步：划分样本集"""

    # 提取样本 特征
    def split_out_attributes(self, X, Y):
        # 提取 每个类别的不同特征
        # c1 第一类的特征数组
        c1_1 = c1_2 = c1_3 = c1_4 = []
        c2_1 = c2_2 = c2_3 = c2_4 = []
```

```
        c3_1 = c3_2 = c3_3 = c3_4 = []
        for i in range(len(Y)):
            if (Y[i] == 'Iris-setosa'):
                c1_1.append(X[i, 0])
                c1_2.append(X[i, 1])
                c1_3.append(X[i, 2])
                c1_4.append(X[i, 3])
            elif (Y[i] == 'Iris-versicolor'):
                # c2 第二类的特征数组
                c2_1.append(X[i, 0])
                c2_2.append(X[i, 1])
                c2_3.append(X[i, 2])
                c2_4.append(X[i, 3])
            elif (Y[i] == 'Iris-virginica'):
                # c3 第三类的特征数组
                c3_1.append(X[i, 0])
                c3_2.append(X[i, 1])
                c3_3.append(X[i, 2])
                c3_4.append(X[i, 3])
            else:
                pass

        return [c1_1, c1_2, c1_3, c1_4,
                c2_1, c2_2, c2_3, c2_4,
                c3_1, c3_2, c3_3, c3_4]

    """因为符合多变量正态分布，所以需要(μ，∑)两个样本参数"""
    """第二步：计算样本期望 μ 和样本方差 s"""

    # 计算样本期望
    def cal_mean(self, attributes):
        c1_1, c1_2, c1_3, c1_4, c2_1, c2_2, c2_3, c2_4, c3_1, c3_2, c3_3, c3_4 = attributes

        # 第一类的期望值 μ
        e_c1_1 = np.mean(c1_1)
        e_c1_2 = np.mean(c1_2)
        e_c1_3 = np.mean(c1_3)
        e_c1_4 = np.mean(c1_4)
        # 第二类的期望值 μ
        e_c2_1 = np.mean(c2_1)
        e_c2_2 = np.mean(c2_2)
        e_c2_3 = np.mean(c2_3)
        e_c2_4 = np.mean(c2_4)
```

```
        # 第三类的期望值 μ
        e_c3_1 = np.mean(c3_1)
        e_c3_2 = np.mean(c3_2)
        e_c3_3 = np.mean(c3_3)
        e_c3_4 = np.mean(c3_4)

        return [e_c1_1, e_c1_2, e_c1_3, e_c1_4,
                e_c2_1, e_c2_2, e_c2_3, e_c2_4,
                e_c3_1, e_c3_2, e_c3_3, e_c3_4]

    # 计算样本方差
    def cal_var(self, attributes):
        c1_1, c1_2, c1_3, c1_4, c2_1, c2_2, c2_3, c2_4, c3_1, c3_2, c3_3, c3_4 = attributes

        # 第一类的方差 var
        var_c1_1 = np.var(c1_1)
        var_c1_2 = np.var(c1_2)
        var_c1_3 = np.var(c1_3)
        var_c1_4 = np.var(c1_4)
        # 第二类的方差 s
        var_c2_1 = np.var(c2_1)
        var_c2_2 = np.var(c2_2)
        var_c2_3 = np.var(c2_3)
        var_c2_4 = np.var(c2_4)
        # 第三类的方差 s
        var_c3_1 = np.var(c3_1)
        var_c3_2 = np.var(c3_2)
        var_c3_3 = np.var(c3_3)
        var_c3_4 = np.var(c3_4)

        return [var_c1_1, var_c1_2, var_c1_3, var_c1_4,
                var_c2_1, var_c2_2, var_c2_3, var_c2_4,
                var_c3_1, var_c3_2, var_c3_3, var_c3_4]

    # 计算先验概率 P(Y=ck)
    def cal_prior_probability(self, Y):
        a = b = c = 0

        for i in Y:
            if (i == 'Iris-setosa'):
                a += 1
            elif (i == 'Iris-versicolor'):
                b += 1
```

```
125             elif (i == 'Iris-virginica'):
126                 c += 1
127             else:
128                 pass
129 
130         pa = a / len(Y)
131         pb = b / len(Y)
132         pc = c / len(Y)
133         return pa, pb, pc
134 
135     # 计算后验概率 P(Y=ck|X)=P(X|Y=ck)*P(Y=ck)/∑
136     def cal_posteriori_probability(self, X, Y, p, means, vars):
137         pa, pb, pc = p
138         e_c1_1, e_c1_2, e_c1_3, e_c1_4, e_c2_1, e_c2_2, e_c2_3, \
139         e_c2_4, e_c3_1, e_c3_2, e_c3_3, e_c3_4 = means
140         var_c1_1, var_c1_2, var_c1_3, var_c1_4, var_c2_1, var_c2_2, \
141         var_c2_3, var_c2_4, var_c3_1, var_c3_2, var_c3_3, var_c3_4 = vars
142 
143         print('p:', p)
144         print('means:', means)
145         print('vars:', vars)
146 
147         # 分解四维输入向量 X=[X1，X2，X3，X4]为 4 个一维正态分布函数
148         X1 = X[:, 0]
149         X2 = X[:, 1]
150         X3 = X[:, 2]
151         X4 = X[:, 3]
152 
153         # 分类正确数/分类错误数=>计算正确率
154         true_test = 0
155         false_test = 0
156 
157         # 遍历训练整个输入空间，计算后验概率并判决
158         for i in range(len(X1)):
159             # 计算后验概率=P(X|Y=C1)P(Y=C1)
160             P_1 = stats.norm.pdf(X1[i], e_c1_1, var_c1_1) * stats.norm.pdf(X2[i], \
161             e_c1_2, var_c1_2) * stats.norm.pdf(
162                 X3[i], e_c1_3,
163                 var_c1_3) * stats.norm.pdf(
164                 X4[i], e_c1_4, var_c1_4) * pa
165             # 计算后验概率=P(X|Y=C2)P(Y=C2)
166             P_2 = stats.norm.pdf(X1[i], e_c2_1, var_c2_1) * stats.norm.pdf(X2[i], \
167             e_c2_2, var_c2_2) * stats.norm.pdf(
```

```
168                 X3[i], e_c2_3,
169                 var_c2_3) * stats.norm.pdf(
170                 X4[i], e_c2_4, var_c2_4) * pb
171             # 计算后验概率=P(X|Y=C3)P(Y=C3)
172             P_3 = stats.norm.pdf(X1[i], e_c3_1, var_c3_1) * stats.norm.pdf(X2[i], \
173             e_c3_2, var_c3_2) * stats.norm.pdf(
174                 X3[i], e_c3_3,
175                 var_c3_3) * stats.norm.pdf(
176                 X4[i], e_c3_4, var_c3_4) * pc
177
178             # 计算判别函数，选取概率最大的类
179             max_P = max(P_1, P_2, P_3)
180             # 输出分类结果，并检测正确率
181             if (max_P == P_1):
182                 if (Y[i] == 'Iris-setosa'):
183                     print('分为第一类，正确')
184                     true_test += 1
185                 else:
186                     print('分为第一类，错误')
187                     false_test += 1
188             elif (max_P == P_2):
189                 if (Y[i] == 'Iris-versicolor'):
190                     print('分为第二类，正确')
191                     true_test += 1
192                 else:
193                     print('分为第二类，错误')
194                     false_test += 1
195             elif (max_P == P_3):
196                 if (Y[i] == 'Iris-virginica'):
197                     print('分为第三类，正确')
198                     true_test += 1
199                 else:
200                     print('分为第三类，错误')
201                     false_test += 1
202             else:
203                 print('未分类')
204                 false_test += 1
205         # 打印分类正确率
206         print('训练正确率为:', (true_test / (true_test + false_test)))
207
208     # 模板方法对照
209     def cal_dataset(self, X_train, Y_train):
210         # Test options and evaluation metric
```

```
211         seed = 7
212         scoring = 'accuracy'
213         # Check Algorithms
214         model = GaussianNB()
215         name = 'bayes classifier'
216         # 建立 K 折交叉验证 10 倍
217         kfold = model_selection.KFold(n_splits=10, random_state=seed)
218         # cross_val_score() 对数据集进行指定次数的交叉验证并为每次验证效果评测
219         cv_results = \
220             model_selection.cross_val_score(model, X_train, Y_train, \
221             cv=kfold, scoring=scoring)
222         results = cv_results
223         msg = "%s: %f (%f)" % (name + '精度', cv_results.mean(), cv_results.std())
224         print(msg)
225 
226         # Show Algorithms
227         dataresult = pandas.DataFrame(results)
228         dataresult.plot(title='Bayes accuracy analysis', kind='density', \
229         subplots=True, layout=(1, 1), sharex=False,
230                         sharey=False)
231         dataresult.hist()
232         plt.show()
233 
234 
235 bayes = Bayes_Test()
236 dataset = bayes.load_dataset()
237 # 划分训练集 测试集
238 X_train, X_validation, Y_train, Y_validation = bayes.split_out_dataset(dataset)
239 print('得到的 X_train', X_train)
240 print('得到的 Y_train', Y_train)
241 
242 # 分割属性--训练集
243 attributes = bayes.split_out_attributes(X_train, Y_train)
244 print('得到的训练集', attributes)
245 
246 # 计算期望--训练集
247 means = bayes.cal_mean(attributes)
248 print('得到的 means', means)
249 
250 # 计算方差--训练集
251 vars = bayes.cal_var(attributes)
252 print('得到的 vars', vars)
253 
```

```
254 # 计算先验概率--训练集
255 prior_p = bayes.cal_prior_probability(Y_train)
256 # 验证分类准确性--测试集
257 bayes.cal_posteriori_probability(X_train, Y_train, prior_p, means, vars)
258
259 # 模板方法--性能对比
260 bayes.cal_dataset(X_validation, Y_validation)
```

程序代码 1

六、运行结果

运行结果 1 （经过了反相处理）

代码分析

七、实验体会

八、参考文献

https://zhuanlan.zhihu.com/p/29431399
http://www.cnblogs.com/leoo2sk/archive/2010/09/17/naive-bayesian-classifier.html
http://www.ruanyifeng.com/blog/2013/12/naive_bayes_classifier.html