南昌大学

《数据挖掘》课程期末考查（大作业）

姓　　名：黄健榜

学　　号：6103115003

专业班级：计算机科学与技术 151 班

时　　间：2018.12.21----2019.1.14

2018 年 12 月 21 日

成绩	评阅人	评阅日期

简易系统开发

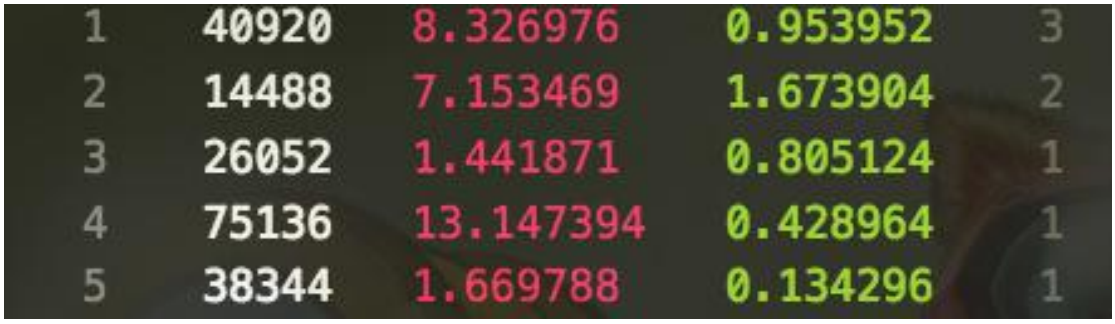
一、目的

（1）掌握数据挖掘基本理论和基本方法，以及会进行简单应用。重点掌握分类预测建模、具类分析和关联分析这三种数据挖掘策略及每种策略的适用范围；
（2）加深理解和领悟常见的数据挖掘算法的程序设计思想与基本实现技术；
（3）培养独立查询资料和解决实际问题的能力，提高自己的编程能力和程序调试能力。

二、内容

（一）需求分析
1、根据收集到的关于每个人一年内的飞行里程、玩视频游戏所消耗时间百分比、每一周消费的冰激淋公斤数的 1000 组数据中，预测一个人对自己的吸引力大小：large dose、small dose、didn’t like
2、根据收集到的关于汽车的一系列数据，预测某辆车在用户心中的接受程度：accept、unaccept、good、very good
3、算法的执行结果要求知道错误率

（二）概要设计
1、使用 python 程序设计语言，在 macOS 平台下进行简易系统的开发，该系统的使用对象数据挖掘相关的专业人员，操作方式是从系统的无图形终端执行。
2、采集分类型的数据，并将数据简单处理，将数据存到文本文件中以便程序读取。
3、建立合适的项目目录结构，编写 KNN 分类算法、ID3 分类算法程序
4、对算法进行训练（如果有需要的话），并根据测试数据集，对算法模型进行测试，返回测试的错误率
（三）、详细设计
对 KNN 分类算法系统：
1、收集数据，并将数据按照如下图整理好，等待程序读入数据



1	40920	8.326976	0.953952	3
2	14488	7.153469	1.673904	2
3	26052	1.441871	0.805124	1
4	75136	13.147394	0.428964	1
5	38344	1.669788	0.134296	1

该数据集为 1000 行，即 1000 组数据第一列为数据集的行数，第二列为每年获得的飞行里程数，第三列为玩视频游戏所耗时间的百分比，第四列为每周消费的冰激淋公斤数，第五列为标签数据，为了方便处理，将三种类型使用 1、2、3 数据来表示。列与列之间是一个制表符的距离。

2、将文本数据转换成数值类型：def file_2_matrix(filename),该函数接受数据文件名称为参数，返回数据列表（前三列数据），以及标签数据列表，即最后一列的数据

3、查看数据集发现每一列数据之间的数值相差太大，将每一列的数据尽心归一化处理：def auto_norm(dataset),该函数接受读入的数据集，并返回处理好之后的数据集以及前三列中每一列的最大值和最小值。

4、编写 KNN 分类算法：def classify0(in_x, dataset, labels, k),该函数接受等待分类的一组数据，in_x,数据集：dataset，标签数据：labels,以及 KNN 算法中用户决定的 k 值。返回分类的结果 large dose、small dose、didn't like 中任意一项

5、测试算法，并将返回算法的错误率：def dating_class_test()

6、预测，根据用户输入的信息，预测属于哪一类：def classify_person()

7、测试数据，由于是开发阶段的测试，所以使用程序生成一小段的数据，

```
def create_dataset():
    group = array([[1.0,1.1], [1.0,1.0], [0,0], [0,0.1]])
    labels = ['A', 'A', 'B', 'B']
    return group, labels
```

返回测试用的数据集以及标签列表，然后再根据数据结构，输入简单的数据，看看是否符合预期，符合，则进行下一步，不符合则调试代码

8、进一步完善程序，代码更整洁，增强代码健壮性

9、编写使用手册，并提交系统

对 ID3 分类算法：

1、收集数据并按照如下图整理好数据，以便程序读入数据

```
1  vhigh,vhigh,2,2,small,low,unacc
2  vhigh,vhigh,2,2,small,med,unacc
3  vhigh,vhigh,2,2,small,high,unacc
4  vhigh,vhigh,2,2,med,low,unacc
5  vhigh,vhigh,2,2,med,med,unacc
6  vhigh,vhigh,2,2,med,high,unacc
```

该数据为 1728 行，没有数据缺失，第一列为数据的行数，第二列到第七列数据的属性分别为 buying, maint, doors, persons, lug_boot, safety 最后一列为标签列，共有 unacc, acc, good, vvgood 四种标签值。列与列之间的数据使用逗号隔开。

2、将文本数据读入到程序中。由于数据太多，如果使用全部的数据来建立决策树，会非常耗时间，所以从 1728 条中随机选取了 174 条完整的数据，并命名为 train_data_car.txt，选 87 条数据作为测试数据，命名为 test_data_car.txt。读取数据的程序独立成一个文件。

3、构建决策树。

①计算给定数据集的熵：def calc_shannon_ent(dataset),该函数接受一个给定的数据集，返回该数据集合中的数据的熵。计算熵的公式为：

$$H = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

②划分数据集：根据给定的特征划分数据集：def split_dataset(dataset, axis, value)该函数接受一个数据集，指定一个特征，以及该特征对应的值，返回划分好的数据集

③选择最好的数据划分方式：根据前面计算熵的算法，划分决策树的算法，计算出最好的信息增益，然后选择最好的特征进行数据集的划分。def choose_best_feature_to_split(dataset)：该函数接受一个数据集，返回选中的最合适的特征。

④递归构建决策树：根据前面三个步骤的子模块递归建立树的结构。def create_tree(dataset, labels):该函数接受给定的数据集，以及数据集对应的标签列表，返回建立好的决策树。

4、绘制决策树：为了更好更直观的观察决策树的样子，使用 python 的 matplotlib 画图模块将决策树绘制出来。该功能独立成一个模块，放置在 plot_tree.py 文件中，绘制决策树的函数为：create_plot(in_tree),该函数接受一棵已经建立好的树，将绘制好的树以图形的形式展现给用户

5、测试：根据测试的数据集，对决策树进行分类，并将结果与原有的标签列表进行对比，返回决策树的错误率 testing_tree()，返回决策树的错误率

6、进一步完善程序，使得代码更加健壮。

7、书写用户使用手册，并提交系统。

（四）调试与算法分析

对 KNN 算法：

1、算法的实现原理：对未知类别属性的数据集中的每个数据点依次执行一下操作

①计算已知类别数据集中的数据点与当前未知点之间的距离

②按照距离递增依次排序

③选取与当前点距离最小的 k 个点

④确定当前 k 个点所在类别的出现频率

⑤返回前 K 个点出现频率最高的类别作为当前点的预测分类

2、计算两个数据点之间的距离公式如下

$$d = \sqrt{(xA_0 - xB_0)^2 + (xA_1 - xB_1)^2}$$

3、算法的主要代码实现

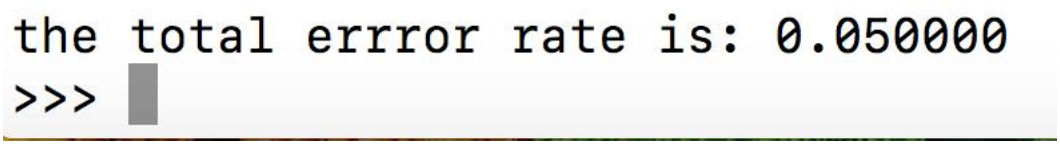
```
def classify0(in_x, dataset, labels, k):
    """knn 算法"""

    #计算距离
    dataset_size = dataset.shape[0]#4
    diff_mat = tile(in_x, (dataset_size, 1)) - dataset
    sq_diff_mat = diff_mat**2
    sq_distance = sq_diff_mat.sum(axis=1)
    distances = sq_distance**0.5
    sorted_dist_indicies = distances.argsort()
    class_count = {}
    #选择距离最小的 k 个点
    for i in range(k):
        vote_i_label = labels[sorted_dist_indicies[i]]
        class_count[vote_i_label] = class_count.get(vote_i_label, 0) + 1
    #排序
```

```
    sorted_class_count = sorted(class_count.items(),
                                key=operator.itemgetter(1), reverse=True)
    #返回第一个数据，即频率最高的那个值
    return sorted_class_count[0][0]
```

可以看出该算法的算法复杂度为常树级，即 O（n）

4、算法调试:根据分类返回的结果与标签列表进行对比，返回错误率，我们可以看到该算法的错误率为 0.05，效果很不错。



对 ID3 分类算法

1、算法的实现原理

①收集数据

②分析数据：检查收集到的数据是否为离散型数据，选择离散型数据

③训练数据：构造合适的树的数据结构

④测试算法的错误率

2、创建树的主要代码

```
def create_tree(dataset, labels):
"""根据数据集和属性建立决策树"""
#特征值完全分完的时候停止递归
    class_list = [example[-1] for example in dataset]
    if class_list.count(class_list[0]) == len(class_list):
        return class_list[0]
#遍历完所有的特征时，返回出现次数最多的结果
    if len(dataset[0]) == 1:
        return majority_cnt(class_list)
    best_feature = choose_best_feature_to_split(dataset)
    best_feature_label = labels[best_feature]
    myTree = {best_feature_label:{}}
    del labels[best_feature]
    feat_values = [example[best_feature] for example in dataset]
    unique_values = set(feat_values)
    for value in unique_values:
        sub_labels = labels[:]
        #递归创建子树
        myTree[best_feature_label][value] = create_tree(split_dataset(
```

```
                dataset, best_feature, value),
                sub_labels
                )
    return myTree
```

3、算法调试

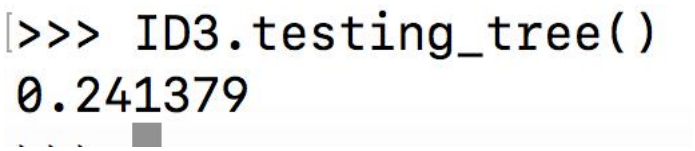
为了方便调试，首先根据目标算法创建了调试用的数据集、

```
def create_dataset():
    dataset = [
        [1, 1, 'yes'],
        [1, 1, 'yes'],
        [1, 0, 'no'],
        [0, 1, 'no'],
        [0, 1, 'no']
    ]
    labels = ['no surface', 'flippers']
    return dataset, labels
my_dat, labels = create_dataset()
my_tree = trees.create_tree(my_dat, labels)
print(my_tree)
```

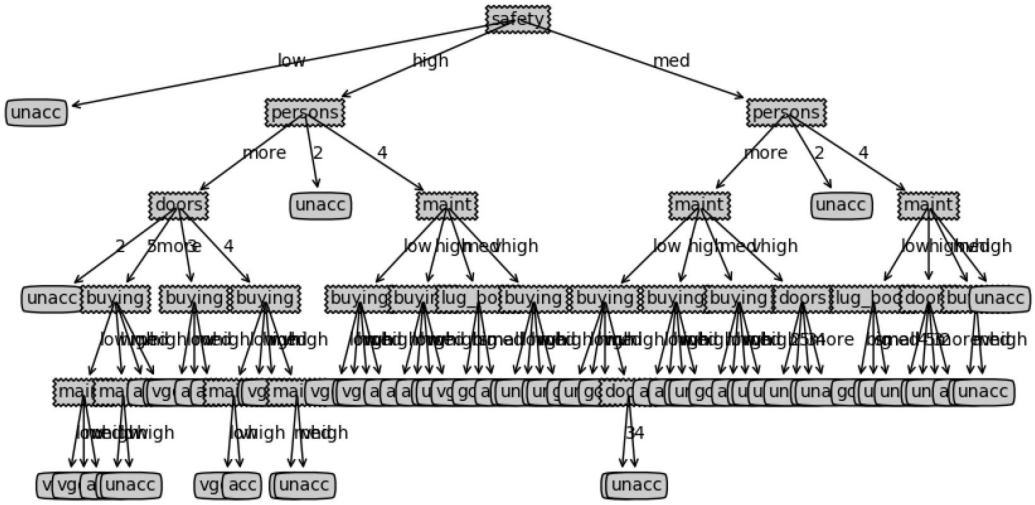
输出树的结构如下

符合预期。

4、根据测试数据对决策树进行测试，返回的结果如下：



描绘出的决策树如下图：



错误率大概 0.24，这个结果有点不理想。因为总的数据集是 1728 条，而我们只选了 174 条数据来构造决策树，所以构造出来的决策树不是很理想。但是从构造出得决策树形状来看，虽然数据集比较小，但是构造出的树还是存在很多的叶子结点。

（五）用户手册

1、简易系统的运行环境条件：

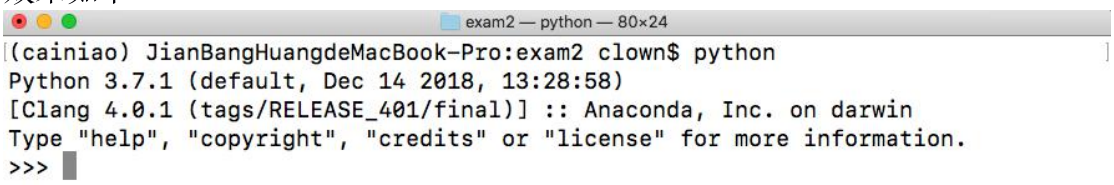
程序语言：python3

操作系统:macOS

软件包：matplotlib

2、进入到 KNN 简易系统中，从后台进入 python 控制台

效果如下：

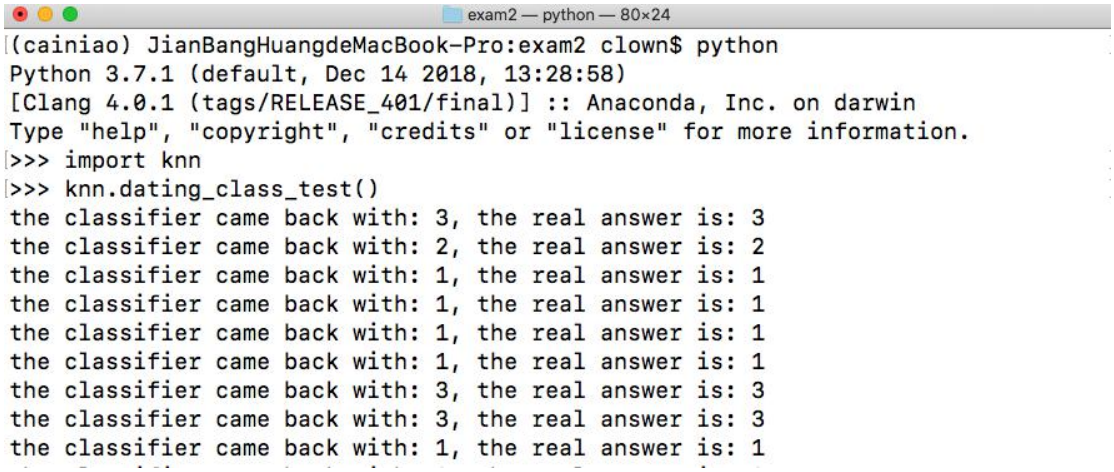


3、导入 knn 模块，分别输入如下命令，对应如下图功能

knn.dating_class_test():返回算法的错误率

knn.classify_person():根据用户输入的信息返回预测的结果

返回算法错误率



返回预测结果

```
>>> knn.classify_person()
percent of time spent playing video games?10
Frequent flier miles earned per year?10000
liters of ice cream consumed per year?0.5
You will probably like this person:  in small doses
>>>
```

4、退出系统：exit()

```
>>> exit()
(cainiao) JianBangHuangdeMacBook-Pro:exam2 clown$
```

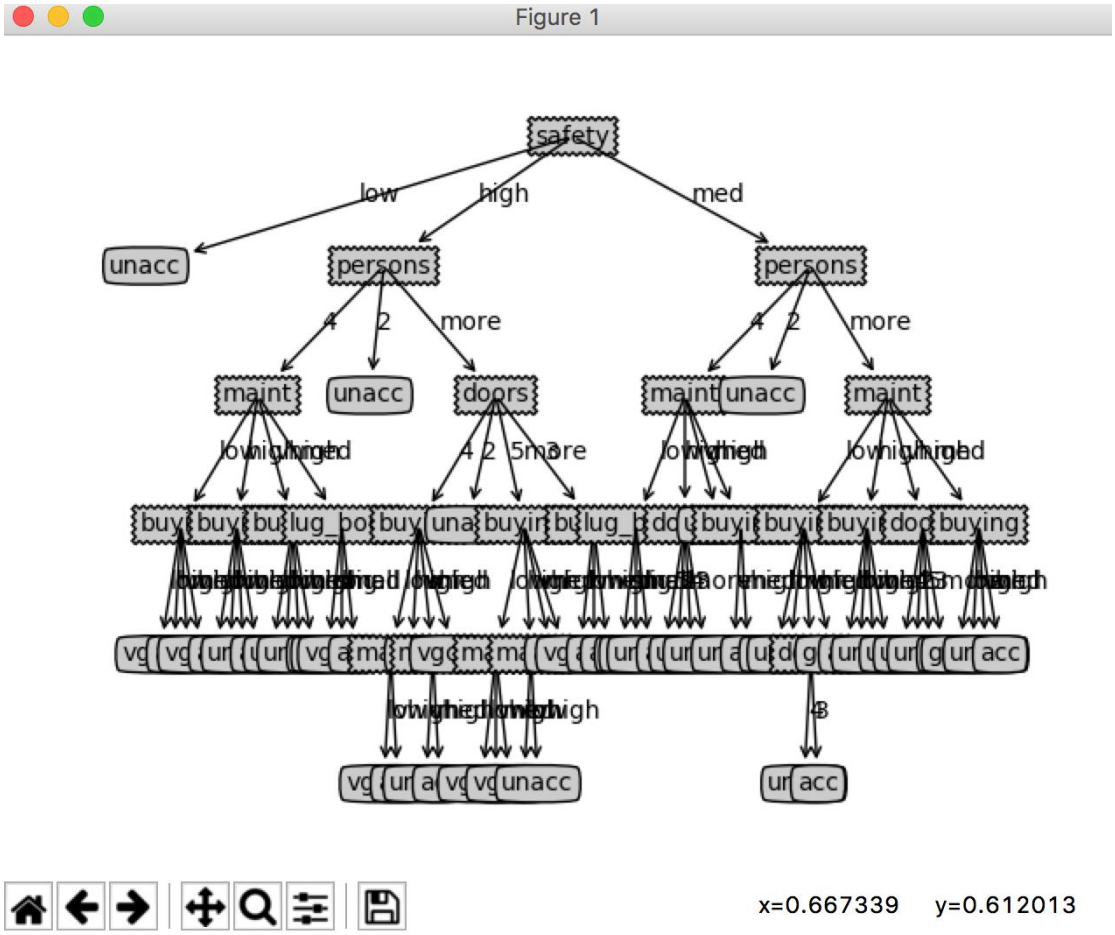
5、进入到 ID3 简易系统中，从后台进入 python 控制台

6、导入 ID3 算法

```
exam — python — 80×24
(cainiao) JianBangHuangdeMacBook-Pro:exam clown$ python
Python 3.7.1 (default, Dec 14 2018, 13:28:58)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import ID3
>>>
```

7、分别输入下列命令，对应的功能如下
ID3.create_store_tree():创建并存储树
ID3.plot_a_tree()：绘制已经创建好的树
ID3.testing_tree()：测试决策树的错误率

ID3.plot_a_tree():



ID3.testing_tree()

```
>>> ID3.testing_tree()
0.241379
>>>
```

四、总结

1、数据挖掘中算法的设计与实现主要分为以下几个步骤
①收集数据：一般是文本文件
②准备数据：使用程序语言解析文本文件，提取我们感兴趣的数据
③分析输入的数据：主要是人工分析以前得到的数据，查看数据的格式是否符合要求，数据是否缺失等等。
③训练算法：将之前准备好的数据格式化输入到算法中，从中抽取知识或者信息。
④测试算法：根据测试用的数据集，将测试数据集输入到算法模型中，返回模型的错误率
⑤使用算法：将机器学习算法转换成为应用程序，执行实际的任务，检验算法模型是否可以在实际工作中应用。
2、KNN 分类算法适用于数值类型的数据进行分类，该算法精度高，对异常值不敏感，没有数据输入假定。但是该算法计算的复杂度高，空间复杂度高
3、决策树分类算法适用于标称行和数值型，但是后者需要将数据进一步处理成离散型。该算法计算复杂度不高，输出结易于理解，对中间值的缺失不敏感，可以处理不相关特征数据。但是该算法可能会产生过度匹配的问题。

4、在编写实现算法的过程中，分模块化，及时测试的做法很重要，不能将程序完全写完了再进行测试，应该一个模块一个模块的测试，只有每个模块都测试通过了才进行下一个模块的开发。

五、参考文献

1、Peter Harrington，机器学习实战（M），北京：人民邮电出版社，2016