同濟大學

TONGJI UNIVERSITY

设计报告

课题名称 基于随机森林的二分类设计

副标题 模式识别——银行客户定期存款意向分析

学院（系） 电子与信息工程学院

专业 自动化

学号 1452373

学生姓名 陈志良

日期 2017 年 11 月 16 日- 2017 年 12 月 7 日

目录

一、设计概述

1.1 设计时间

2017 年 11 月 16 日 ——2017 年 12 月 7 日。

1.2 问题概述

利用提供的银行客户信息对客户是否会定期存款进行二分类预测。首先使用训练数据集对分类器进行训练，然后用训练好的分类器在测试集上进行二分类预测，预测结果提交给系统进行打分。系统采用 MeanF1 对预测结果进行评估并排名。

二、设计内容

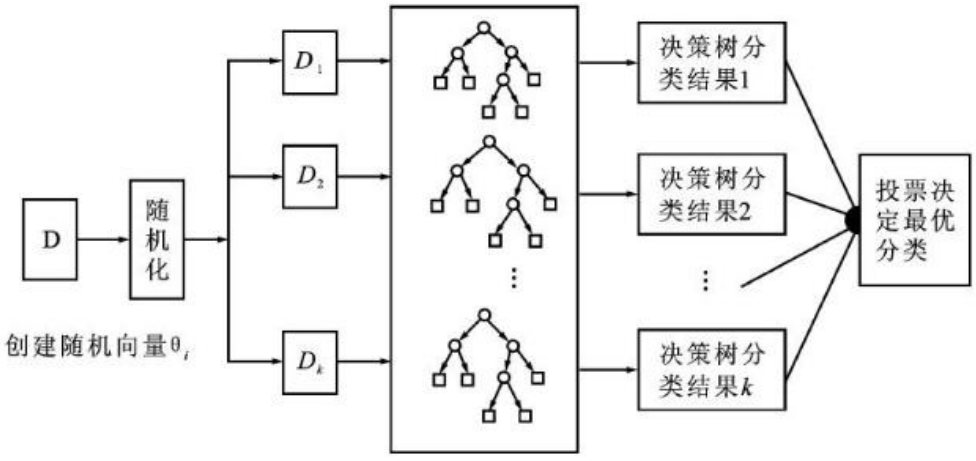
2.1 相关背景知识

2.1.1 随机森林

在机器学习领域，基于决策树的方法曾被广泛使用，一是由于决策树本身简单、快速，决策树模型易于理解，通过决策树的工作过程可以直观理解问题的求解过程；二是由于决策树能够对广泛的问题给出准确的解。

随机森林是一种包含多棵决策树的分类器,其中每棵决策树的构造和分类测试均相互独立。训练过程中,每棵决策树对原始的训练数据集进行采样替换,构造新的训练数据集(bootstrap 方法); 决策树中每个决策节点上的分裂测试均从一个随机测试集合中产生:根据某种量化评价标准,例如信息熵等, 从随机测试集合中选择一个最佳测试作为决策点的分裂测试。随机森林中的每棵决策树均不需要进行剪枝。

随机森林分类器（RandomForestClassifier）的基本思想:首先,利用 bootstrap 抽样从原始训练集抽取 k 个样本,且每个样本的样本容量都与原始训练集一样;其次,对 k 个样本分别建立 k 个决策树模型,得到 k 种分类结果; 最后,根据 k 种分类结果对每个记录进行投票表决决定其最终分类。其结构图如下图所示：



2.1.2 RandomForestClassifier( )函数

随机森林模型的建立主要依赖于 sklearn 库中的 RandomForestClassifier 函数。该函数的参数如下表 1 所示：

表 1 RandomForestClassifier（）函数参数

序号	参数	说明
1	class_weight	类别的权值
2	n_estimators	子模型（即子数）的数量 int：个数，默认值 10
3	criterion	判断节点是否继续分裂采用的计算方法 可取：entropy、gini
4	max_features	节点分裂时参与判断的最大特征数 可取：int、float、auto、sqrt、log2、None
5	max_depth	树的最大深度，可取：int、None
6	min_samples_split	分裂所需的最小样本数 可取：int。默认值 2
7	min_samples_leaf	叶节点最小样本数 可取：int。默认值 1
8	min_weight_fractioin_leaf	叶节点最小样本权重总值 可取：float。默认值 0
9	max_leaf_nodes	最大叶节点数，可取：int、None
10	bootstrap	是否 bootstrap 对样本抽样 可取：False、True
11	random_state	随机器对象

2.2 算法原理分析

2.2.1 思路分析

算法实现的思路大致分为以下四步：

- 第一步，数据预处理阶段，包括导入原始数据，对数据进行编码数字化以及将数据集划分为训练集和测试集等；
- 第二步，模型训练阶段，主要根据划分的训练集进行随机森林模型的构建；
- 第三步，模型测试阶段，主要用训练好的模型对测试集进行测试，将测试结果和真实标记统计比较，求出分类查准率，召回率，mean-F1，精度等相关指标，并调节参数使其在测试集上得到以上指标的最大优化；
- 第四步，给出结果阶段，在第三步调好参数的基础上，用模型对未标记的测试集作出预测，并将预测结果写入 csv 文件。

2.2.2 程序实现分析

1.数据预处理

通过 pandas 库里的 read_csv 函数读取原始数据集，保存至 Data 变量里。由于数据集里"job","marital","education"等属性值是字符串类型，所以还需要对这些字符串型数据进行编码，化为数字型才能进行模型的训练，编码主要是用 col.codes 实现，如下图所示：

Data=pandas.read_csv("MT_Train.csv")

Data.head()用于显示对数据集编码以后的数据，默认显示前 5 行。

print(Data['y'].value_counts())的功能是统计数据集中结果”y”列中的”yes”和”no”的个数。

```
#------------------------------数据预处理--------------------------------#
#读入数据集
Data=pandas.read_csv("MT_Train.csv")

#把文本型变量变成数字型
for name in ["job","marital","education","default","housing","loan","contact",
             "month","day_of_week","poutcome"]:
    col=pandas.Categorical(Data[name])
    Data[name]=col.codes                                    #对字符串型数据进行编码
Data.head()                                                 #显示对数据集编码以后的数据
#统计数据集中结果"y"列中的"yes""和"no"的个数
print(Data['y'].value_counts())
```

数据预处理还包括将数据集划分为训练集和测试集，这里设置为数据集的 80%为训练集，20%为测试集，如下图所示：

```
#重排数据
Data=Data.reindex(numpy.random.permutation(Data.index))
#设置训练样本占原始数据集的80%
train_max_row=math.floor(Data.shape[0]*0.8)
train=Data.iloc[:int(train_max_row)]
test=Data.iloc[int(train_max_row):]
```

2.模型训练

在对数据集进行了读取、编码和划分训练集和测试集等操作后，接下来就可以在训练集上对随机森林模型进行训练了。模型的训练主要基于从 sklearn 库中导入的 RandomForestClassifier 函数。

训练时不必用到数据集中所有属性，可以抛弃一些对分类结果影响不大的属性，自主选择模型需要的属性。

```
#-----------------------随机森林模型训练-----------------------------#
from sklearn.ensemble import RandomForestClassifier
#选择x变量属性
columns=["age","job","marital","education","default","housing","loan","contact",
         "month","day_of_week","duration","campaign","pdays","previous","poutcome",
         "emp.var.rate","cons.price.idx","cons.conf.idx","euribor3m","nr.employed"]
#建立190棵树
clf=RandomForestClassifier(n_estimators=190,random_state=1,
min_samples_leaf=2)
#进行数据拟合，训练模型
clf.fit(train[columns],train["y"])
#对测试集进行预测
predictions=clf.predict(test[columns])
#输出测试集预测结果
print(predictions)
```

3.模型测试

在训练好了模型后，为了提升泛化性能，我们需要在测试集上进行测试。首先基于模型得到测试结果 predictions，并将测试结果与样本真实结果 data1 比较，进而由基本概念算出查准率 P，召回率 R，mean-F1，精度 accuracy 等相关指标，才能一步步调试模型参数使其分类效果最优。

$$P=\frac{TP}{TP+FP} \qquad R=\frac{TP}{TP+FN} \qquad F1=\frac{2*P*R}{P+R}$$

$$Accuracy=\frac{样本分类正确的个数}{样本总个数}=\frac{correct}{len(data1)}$$

通过不断测试，根据以上指标调节 RandomForestClassifier 函数参数，我最终选择的模型参数为：决策树棵树为 190，添加随机器对象，叶节点最小样本数为 2，其余参数均为默认值。

```
#------------------------计算相关指标-------------------------------#
TP=0                                    #初始化真正例个数为0
T_predict=0                             #初始化预测结果为正例个数为0
T_real=0                                #初始化样本真实结果为正例个数为0
correct=0                               #初始化样本分类正确个数为0
for i in range(1,len(data1)):
    if(predictions[i]=='yes'):
        T_predict+=1                                        #统计预测结果为正例个数
    if(data1.iloc[i,0]=='yes'):
        T_real+=1                                           #统计样本真实结果为正例个数
    if((data1.iloc[i,0]=='yes')&(predictions[i]=='yes')):
        TP+=1                                               #统计真正例个数
    if(data1.iloc[i,0]==predictions[i]):
        correct+=1                                          #统计样本分类正确个数
#计算并输出查准率P, 召回率R, mean-F1, 精度accuracy
P=(float) (TP)/T_predict
R=(float) (TP)/T_real
F1=(float) (2*P*R/(P+R))
precision=(float) (correct)/len(data1)
print(P)
print(R)
print(F1)
print(precision)
```

4.给出结果

通过上一步对模型参数的调节，模型泛化性能得到了优化。最后一步就是用模型对未标记的测试集作出预测，将预测结果写入最后的 csv 文件中。

首先这里同样需要读入未标记的测试集数据，将里面的字符串型数据进行同样的编码，化为数字型，然后选择与上面第 2 步中对应的属性。如果没有这一步，程序会报错，即模型的维数与数据集的数据维数不一致，导致无法作出预测。

```
#-------------------------对未标记的测试集作出预测-------------------------#
#读入测试集
data_test=pandas.read_csv('MT_Test.csv')
#把文本型变量变成数字型
for name in ["job","marital","education","default","housing","loan","contact",
             "month","day_of_week","poutcome"]:
    col=pandas.Categorical.from_array(data_test[name])
    data_test[name]=col.codes                               #对字符串型数据进行编码
data_test.head()                                            #显示对测试集编码以后的数据
columns=["age","job","marital","education","default","housing","loan","contact",
         "month","day_of_week","duration","campaign","pdays","previous","poutcome",
         "emp.var.rate","cons.price.idx","cons.conf.idx","euribor3m","nr.employed"]
result=clf.predict(data_test[columns])                      #对测试集进行预测
print(result)
```

程序中 result 是对测试集作出的预测结果，是一个字典型数据，在写入 csv 文件之前，我对 result 数据进行了一些变形，使其格式正确，首先通过 reshape 函数将行向量转化为纵向量 result1，然后通过 pandas 库里的 DataFrame 函数将 result1 化成 DataFrame 类型的数据 data2，最后就可以用 data2.to_csv 将数据直接写入 csv 文件中，并规定好列名为”SampleId”和”y”，文件名保存为”FinalResult.csv”。

此外，还可以利用 FinalData['y'].value_counts()来计算预测结果”中的”yes”和”no”的个数。

```
106 df = read_csv('result.csv')
107 df.columns = ['SampleId', 'y']                    #写入列名
108 df.to_csv('.result.csv')
109
110 with open(".result.csv","r") as source:
111     rdr= csv.reader( source )
112     with open("FinalResult.csv","w") as result:
113         wtr= csv.writer( result )
114         for r in rdr:
115             del r[0]
116             wtr.writerow(r)
117
118 #将预测结果写入最终csv文件
119 FinalData=pandas.read_csv("FinalResult.csv")
120 #检查结果”y”列中的”yes””和”no”的个数
121 FinalData['y'].value_counts()
```

2.2.3 结果分析

经过上面的几个步骤，程序最终运行结果为：

指标	查准率 P	召回率 R	Mean-F1	精度 accuracy
结果	0.6979	0.7128	0.7053	0.893

如下图所示：

```
no     8690
yes    1888
Name: y, dtype: int64
['no' 'no' 'no' ..., 'no' 'no' 'yes']
0.6979166666666666
0.7127659574468085
0.7052631578947368
0.8936672967863895
```

通过将最后结果上传到网上，测试系统给出的评分为 0.90087。最终效果尚可，不足之处在于由于函数参数个数太多，若全部用上则费事费力，因此我在调参的过程中放弃了一些参数只选择其默认值，这也是手动调参带来的弊端。由于时间限制，无法对于模型的进一步优化，可以从以下几个方面着手：

- 数据预处理部分。针对缺失数据，选择更好的处理方式；选择模型需要的属性个数，剔除对分类结果影响很小的属性，以提高模型性能；

- RandomForestClassifie（）考虑更多参数，通过程序实现求解最优的参数值；
- 优化原始数据集中训练集与测试集的比例，这里只是大致取了 0.8，可以进一步优化；
- 可以采取交叉验证方法提升性能。

三、设计总结

在前后将近 20 天的时间里，我独立完成了本次期中考试设计，通过看书以及网上查找相关资料，我学习了许多以前从来没有接触过的知识。最主要的是学习了随机森林的相关知识，了解了它的原理和大致实现流程，然后以数据集进行了模型的训练与调参，我相信这些理论的学习会对我以后的学习和生活带来很大的帮助。

在实践方面，我也花费了不少时间，尤其是对于 python 的使用，我在写程序和调试的过程中，加深了对 python 的理解，锻炼了使用 python 编程能力，当然也包括学习 jupyter notebook、anaconda 和 spyder 等工具。

当然，本次设计的完成由于时间有限，再加上我又是初次接触 python 机器学习，因此还存在很多的不足之处。比如由于 python 编程能力有限，我对于里面数据类型理解不够透彻，导致在求解各项指标时遇到了很大的困难，最后通过将结果化为一维列向量才得以解决。

在设计过程中，我体会到了作为科研人员必须要有踏实负责、不怕困难的生活态度，每当遇到困难进行不下去的时候，我也有过灰心丧气，但最终还是静下心来理清思路，一步一个脚印，因此成功的路上没有捷径，只有脚踏实地才能成功。

总之，这次的期中考试设计让我受益匪浅。考试虽然结束了，但我学习的过程远未结束，我将以更加崭新的姿态和积极的生活态度去挑战生活中的问题，不断扬长避短，提高自我！

参考文献

[1] scikit learn 官网说明文档（http://scikit-learn.org/stable/supervised_learning.html#supervised-learning）

[2] Random Forest 和 Gradient Tree Boosting 如何调参 - CSDN 博客 .(http://blog.csdn.net/bryan__/article/details/52090392)

[3] Random Forest（sklearn 参数详解) – CSDN 博客.（http://blog.csdn.net/u012102306/article/details/52228516）

[4] python 机器学习及实践——从零开始通往 kaggle 竞赛之路。2016.10 范淼，李超.清华大学出版社