队伍编号	MCB2202370
赛道	B

关于北京移动用户体验影响因素的研究

——2022 年 MathorCup 高校数学建模挑战赛大数据竞赛

摘 要

客户满意度是客户对移动运营商产品服务的满意程度，反映了客户期望与实际感知的产品服务之间的差异。特别是在信息透明、产品同质化的今天，客户满意度的表现成为各大运营商市场运营状况的重要体现。本研究拟通过分析影响用户满意度的各种因素，为决策提供依据，从而实现更早、更全面提升用户满意度。

面对大量的数据，首先需要转换文件格式提高数据读取效率，并且将重要的且难以读取的汉字信息转换成计算机便于接受的数字符号。经过数据的预处理后尝试决策树、BP 神经网络等算法，探索出适合本题求解的模型——决策树。从语音业务用户满意度和上网业务用户满意度两个方面构建模型，进行数据的预测，并验证预测数据的准确性和合理性，得出结论：语音业务用户满意度受网络覆盖与信号强度、语音通话清晰度和语音通话稳定性影响最大；手机上网整体满意度受网络覆盖与信号强度、手机上网速度和手机上网稳定性影响最大。

关键词：决策树；随机森林；相关性检验；BP 神经网络

目　　录

1 前言

1.1 简介

北京移动通信有限责任公司是中国移动（香港）有限公司全资间接拥有的外商独资企业，主要经营移动电话通信业务（包括语音、数据、多媒体等），IP 电话及互联网接入服务；具有移动通信、IP 电话和互联网网络设计、投资和建设资格。基站总数超过 220 万个，客户总数超过 8 亿户，是全球网络规模、客户规模最大的移动通信运营平台，秉持着“创无限通信世界，做信息社会栋梁”的企业使命。客户满意度是客户对运营商产品服务的满意程度，反映了客户期望与实际感知的产品服务之间的差异。本研究拟通过分析影响用户满意度的各种因素，为决策提供依据，从而实现更早、更全面提升用户满意度。

1.2 问题重述

通过题目所给若干影响因素及相关数据，分析各因素对语音业务用户满意度打分和对上网业务用户满意度打分的影响程度并建立相关模型，从而通过所有影响因素对应的数据预测用户的满意度打分。

2　语音业务用户满意度数据构建与分析

2.1　模型的建立与求解

2.1.1　文件读取

由于附件的数据不算小，xlsx 格式的数据读取较慢，所以我们在解题之前先将 xlsx 格式的文件转化为 csv 格式并放置在 data 文件夹下，以便后续快速读取数据，如图 2.1 所示：

```
df1 = pd.read_csv('./data/附件1语音业务用户满意度数据.csv')
df2 = pd.read_csv('./data/附件2上网业务用户满意度数据.csv')
```

图 2.1　文件读取

2.1.2　数据处理

原数据给出 5433 个数据，经过观察发现，有一些数据的产生带有明显的随意打分的情况，具有强烈的人为因素，因此这些数据并不能对正确结论的产生起促进作用，只会带来负面的影响，所以应当首先排除无用数据。

由实际生活和题目所给数据可知，网络覆盖与信号强度打分、语音通话清晰度打分、语音通话稳定性打分最能决定语音业务用户满意度，网络覆盖与信号强度打分、手机上网速度打分、手机上网稳定性打分最能决定手机上网整体满意度。

运用下列相关性检验公式 2.1 进行验证。

$$r=\frac{\sum_{i=1}^{n}\left(X_i-\overline{X}\right)\left(Y_i-\overline{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i-\overline{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i-\overline{Y}\right)^2}} \tag{2.1}$$

其中：r为相关系数，$|r|\leqslant 1$；X_i为第i个自变量；Y_i为第i个因变量；$\overline{X}$为i个自变量的平均值；$\overline{Y}$为i个因变量的平均值。

检验结果如表 2.1 所示，所有数据均在 0.900 左右，由表 2.2 可知属于高度相关或极高相关，此时可以认为语音通话整体满意度受网络覆盖与信号强度、语音通话清晰度、语音通话稳定性影响很大，可以通过这三者数据和整体满意度数据的平衡性进行数据的初步筛选。

表 2.1　四个打分之间的相关系数

	语音通话整体满意度	网络覆盖与信号强度	语音通话清晰度	语音通话稳定性
语音通话整体满意度	1.000	0.895	0.908	0.886
网络覆盖与信号强度	0.895	1.000	0.900	0.895
语音通话清晰度	0.908	0.897	1.000	0.902
语音通话稳定性	0.886	0.895	0.902	1.000

表 2.2　$|r|$ 的取值与相关程度

$|r|$ 的取值范围	$|r|$ 的意义
0.00--0.19	极低相关
0.20--0.39	低度相关
0.40--0.69	中度相关
0.70--0.89	高度相关
0.90--1.00	极高相关

2.1.3　模型形式设定

对于表格中难以直接处理的汉字，通常需要转换成计算机可以直接读取的形式。

（1）某情况下是否出现网络问题

这里统一将出现问题的情况视为 1，未出现问题的情况视为 0，例如：

用户id	语音通话整体满意度	网络覆盖与信号强度	语音通话清晰度	语音通话稳定性	是否遇到过网络问题	居民小区	办公室	高校	商业街	地铁	农村	高铁	其他，请注明
1	10	6	6	6	1	-1	2	-1	-1	-1	-1	-1	-1

图 2.2　原始数据

改为：

用户id	语音通话整体满意度	网络覆盖与信号强度	语音通话清晰度	语音通话稳定性	是否遇到过网络问题	居民小区	办公室	高校	商业街	地铁	农村	高铁	其他，请注明
1	10	6	6	6	1	0	1	0	0	0	0	0	0

图 2.3　更改数据

（2）4/5G 用户

表 2.3　原始数据对应的更新数据

原始数据	更新数据
2G	0
4G	1

5G	2

（3）语音方式

表 2.4　原始数据对应的更新数据

原始数据	更新数据
GSM	0
CSFB	1
VOLTE	2
EPSFB	3
VONR	4

（4）客户星级标识

表 2.5　原始数据对应的更新数据

原始数据	更新数据
未评级	1
准星	2
一星	3
二星	4
三星	5
白金卡	6
钻石卡	7
银卡	8
金卡	9

（5）性别

表 2.6　原始数据对应的更新数据

原始数据	更新数据
女	0
男	1
性别不详	1

（6）是/否

表 2.7　原始数据对应的更新数据

原始数据	更新数据
是	1
否	0

2.1.4 建立模型

（1）决策树[1]

决策树有三种经典算法：ID3、C4.5、CART。

①ID3（信息增益）

信息熵可以度量信息的潜在价值和有效程度，也是度量样本集合纯度最常用的一种指标。通常对于信息熵的计算如下：

$$H(x)=-\sum_{i=1}^{k}P_i\ln P_i \tag{2.2}$$

其中：$H(x)$为信息熵，P_i为出现第i种情况的概率，这里的底数设定为e。

由公式可见，当数据量一致时，系统越有序或信息越确定，熵值越低；系统越混乱或者信息越不确定，熵值越高。那么，想要得出准确的信息，就必须要合理有效地降低信息熵，信息熵降低的大小称为信息增益，信息增益计算公式如下：

$$gain(Y,X)=H(Y)-H(Y|X) \tag{2.3}$$

其中：$gain(Y,X)$为信息增益，$H(Y)$为没有X条件约束的信息熵，$H(Y|X)$为在X条件约束下的信息熵，其值通过公式 2.4 计算。

$$H(Y|X)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log\frac{|D_{ik}|}{|D_i|} \tag{2.4}$$

决策树的搭建就是通过贪心算法选择局部最优解，尽量按照信息增益大的取值作为划分叶子结点的依据，同时也是一种递归的算法，不断分裂结点降低信息熵。决策树的结点在以下两种情况下停止分裂：一是属性已经全部计算完毕，二是训练的数据已经属于同一类别，没有必要继续进行划分。

那么在应用决策树解决问题时，会涉及到样本很多不同的属性，所以最优划分属性对于决策树而言就很重要了，信息增益的计算当然也就必不可少。

但如果只用一棵决策树，没法得到有效的以及希望得到的结果，所以我们通常采用随机森林的方法构建大量的决策树进行操作，利用多棵树的力量进行决策。

利用决策树分析出的结果如表 2.8 所示，前三项仍然为语音通话清晰度、网络覆盖与信号强度、语音通话稳定性，经过三元权重分析展现出的高准确率，同样也可证明数据剔除的正确性。

表 2.8　各项属性的权重

属性	训练结果
语音通话清晰度	0.2467
网络覆盖与信号强度	0.2349

语音通话稳定性	0.1835
当月 ARPU	0.0447
mos 质差次数	0.0324
是否遇到过网络问题	0.0246
脱网次数	0.0209
居民小区	0.0198
通话中有杂音、听不清、断断续续	0.0166
重定向次数	0.0164
通话过程中突然中断	0.0162
未接通掉话次数	0.0162
重定向驻留时长	0.0149
通话过程中一方听不见	0.0145
有信号无法拨通	0.0143
手机没有信号	0.0133
办公室	0.0127
地铁	0.0100
高铁	0.0075
农村	0.0074
商业街	0.0073
前第 3 个月欠费金额	0.0040
串线	0.0038
其他，请注明	0.0037
当月欠费金额	0.0033
高校	0.0027
其他，请注明.1	0.0026
ARPU	0.0020
前 3 月 ARPU	0.0020
是否 4G 网络用户（本地剔除物联网）	0.0011

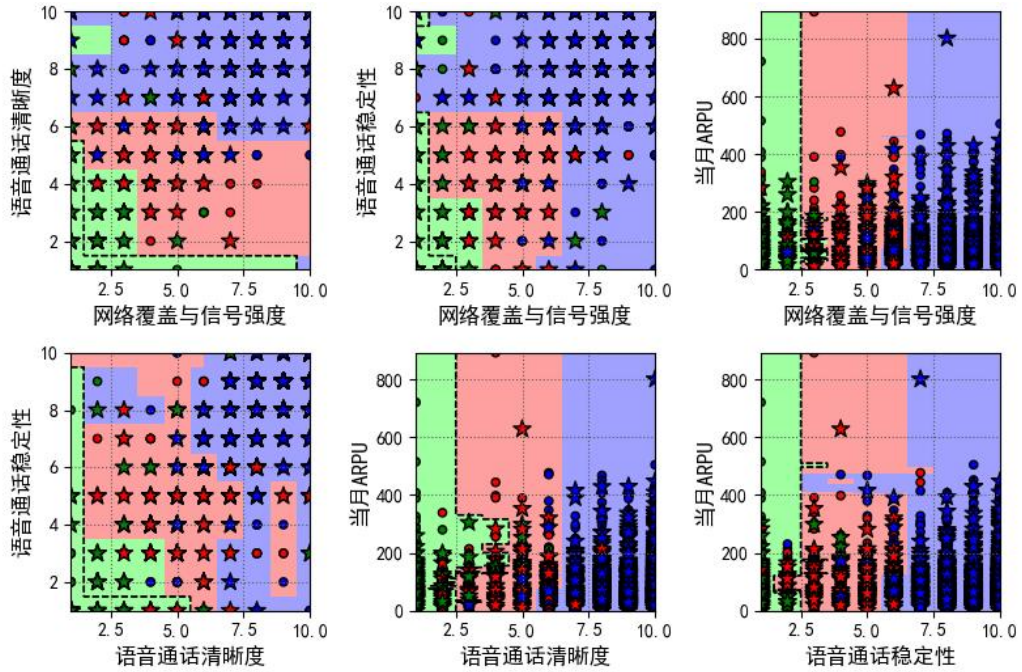

图 2.5　两特征组合的分类结果

②CART（Gini 系数[2]）

Taylor 公式有

$$f(x)=f(x_0)+\frac{f^{'}(x_0)}{1!}(x-x_0)+\frac{f^{''}(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x) \tag{2.5}$$

其中：$R(x)$为余项

当$x_0=0$时，得到Maclaurin公式

$$f(x)=f(0)+\frac{f'(0)}{1!}x+\frac{f''(0)}{2!}x^2+\ldots+\frac{f^{(n)}(0)}{n!}x^n+R_n(x) \tag{2.6}$$

其中：$R(x)$为余项

令$f(x)=\ln x$，取一阶麦克劳林展开式，有

$$\ln x=1+x+o(x) \tag{2.7}$$

所以当$0<x<1$时，可以认为

$$\ln x\approx 1+x \tag{2.8}$$

代入信息熵的计算公式，有

$$Gini(p)=\sum_{k=1}^{K}p_k(1-p_k)=1-\sum_{k=1}^{K}p_k^2 \tag{2.9}$$

同样也可以用来计算，此处省略。

③C4.5（信息增益率）

信息增益率的计算公式为：

$$g_r(D,A)=\frac{g(D,A)}{H(A)} \tag{2.10}$$

此处省略具体计算过程。

（2）BP 神经网络[3] [4] [5]

BP 神经网络是目前比较常用的人工神经网络，其结构包括输入层、输出层以及若干个隐含层。其典型 3 层 BP 神经网络拓扑结果如图 2.6 所示

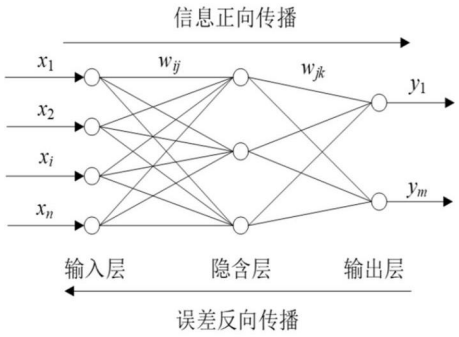

图 2.6　BP 神经网络图解

BP 神经网络为多层前反馈结构，同层神经元节点无信息交互，不同层之间根据连接权重传递信息。BP 神经网络的误差反向传播和多层设计使得其能够更准确的反映输入和输出之间的映射关系，以便完成复杂任务。

BP 神经网络的传递函数分为线性和非线性，非线性传递函数一般使用 Sigmoid 函数（该函数及其导数均为连续函数，是进行神经网络计算的首选激活函数），根据输出值范围不同，Sigmoid 函数可分为 Log-Sigmoid 和 Tan-Sigmoid 函数。函数表达式如式 2.11 所示

$$\begin{aligned} f(x)&=\frac{1}{1+e^{-x}}, f(x)\in[0,1] \\ f(x)&=\frac{1-e^{-x}}{1+e^{-x}}, f(x)\in[-1,1] \end{aligned} \tag{2.11}$$

通常，实现 BP 神经网络的步骤有：

①读取数据

②设置训练数据和预测数据

③训练样本数据归一化

④构建 BP 神经网络

⑤网络参数配置(训练次数，学习速率，训练目标最小误差.等)

⑥BP 神经网络训练

⑦测试样本归一化

⑧BP 神经网络预测

⑨预测结果反归一化与误差计算

⑩验证集的真实值与预测值误差比较

那么，根据以上步骤，我们可以初步分析得出以下图像：

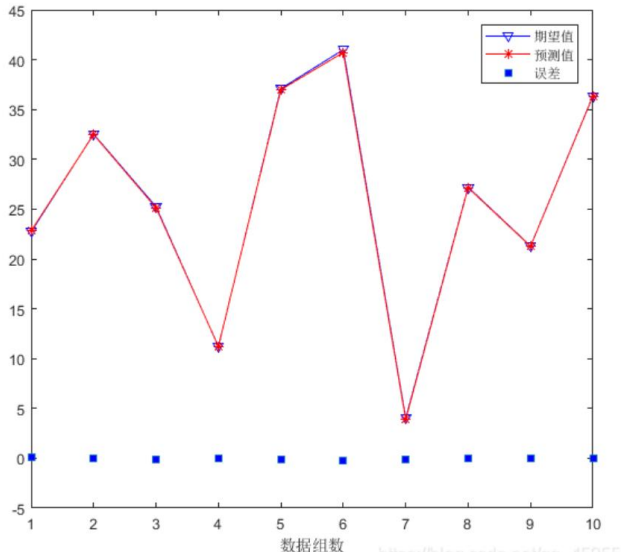

图 2.7　预测值和真实值、误差的分析图像

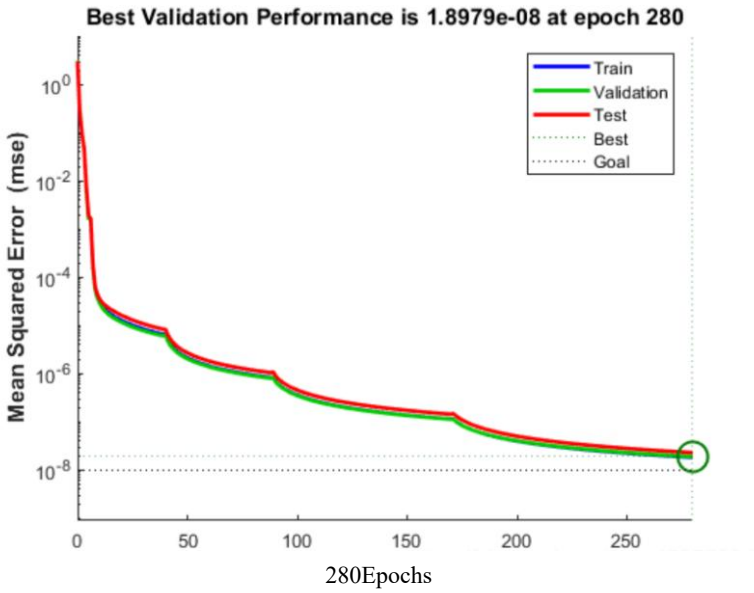

图 2.8　训练集、验证集、测试集和总体的均方误差随训练次数的变化图像

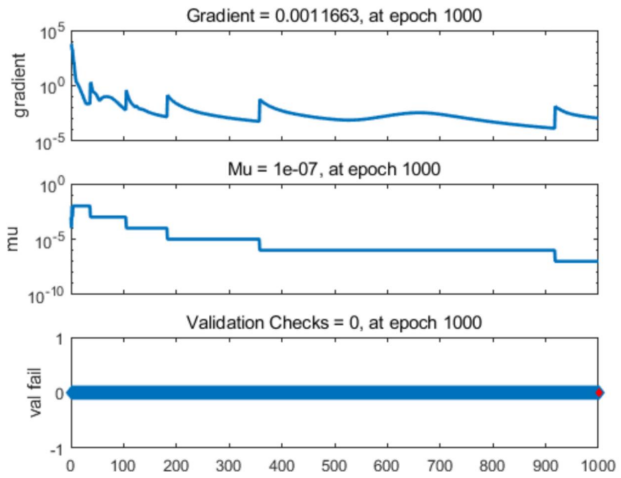

图 2.9　BP 神经网络各阶段的训练图像

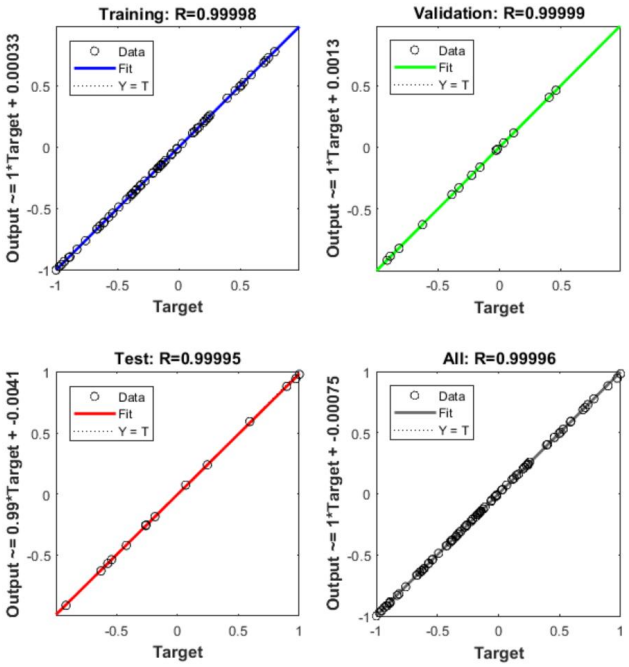

图 2.10　各个样本集和总体的相关性分析图像

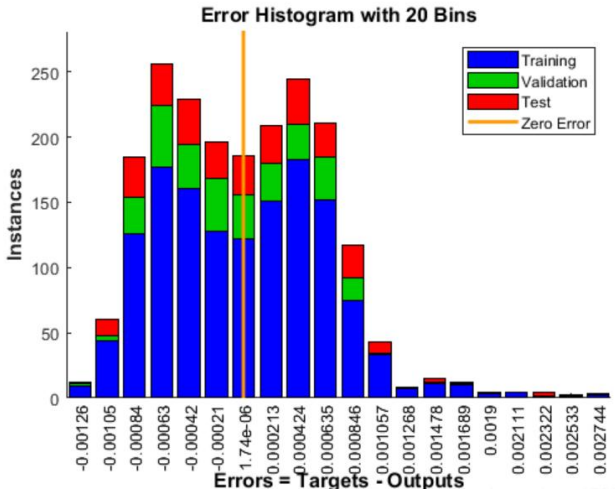

图 2.11　训练集、验证集和测试集的误差分布直方图像

针对此题，有结果图 2.12，图 2.13，出现了严重的过拟合情况，此方法在本情境中不如决策树准确率高。

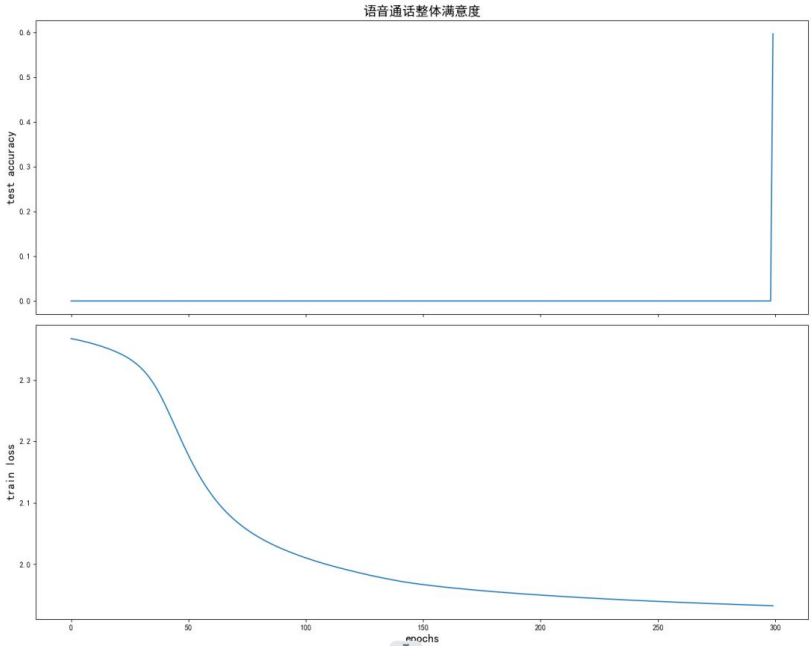

图 2.12 训练集和测试集

图 2.13 训练集和测试集准确率

2.2 利用模型进行预测

利用决策树进行预测，准确率如图 2.14,2.15 所示：

图 2.14 决策树准确率最高

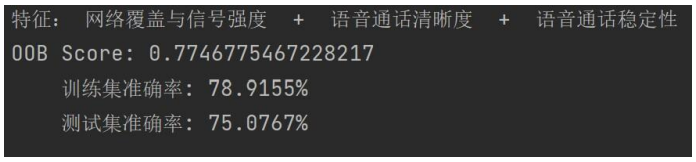

图 2.15 三个属性分析

2.3 结论

语音业务用户满意度受网络覆盖与信号强度、语音通话清晰度和语音通话稳定性影响最大，可以说这三者占据主导，为后续预测数据提供便利，其他属性均占比很小但各有其作用，总体为数据的计算提供有力帮助。

3　上网业务用户满意度数据构建与分析

3.1　模型的建立与求解

3.1.1　文件读取

```
df3 = pd.read_csv('./data/附件3语音业务用户满意度预测数据.csv')
df4 = pd.read_csv('./data/附件4上网业务用户满意度预测数据.csv')
```

图 3.1　读取文件

3.1.2　数据处理

依据 2.1.2 中陈述的原因，同样需对表中 7020 组数据利用相关性进行筛选。

表 3.1　四个打分之间的相关系数

	手机上网整体满意度	网络覆盖与信号强度	手机上网速度	手机上网稳定性
手机上网整体满意度	1.000	0.901	0.911	0.891
网络覆盖与信号强度	0.901	1.000	0.910	0.899
手机上网速度	0.911	0.910	1.000	0.909
手机上网稳定性	0.891	0.899	0.909	1.000

表 3.2　$|r|$ 的取值与相关程度

$|r|$ 的取值范围	$|r|$ 的意义
0.00--0.19	极低相关
0.20--0.39	低度相关
0.40--0.69	中度相关
0.70--0.89	高度相关
0.90--1.00	极高相关

由此可见，所有数据均在 0.900 左右，属于高度相关或极高相关，此时可以认为手机上网整体满意度受网络覆盖与信号强度、手机上网速度、手机上网稳定性影响很大，可以通过这三者数据和整体满意度数据的平衡性进行数据的初步筛选。

3.1.3　模型形式设定

（1）某情景是否出现网络问题

这里统一将出现问题的情况视为 1，未出现问题的情况视为 0，如下图所示：

用户	手机上网整体满意度	网络覆盖与信号强度	手机上网速度	手机上网稳定性	居民小区	办公室	高校	商业街	地铁	农村	高铁	其他，请注明
用户1	8	7	7	7	-1	-1	-1	-1	-1	-1	-1	-1

图 3.2　原始数据

用户	手机上网整体满意度	网络覆盖与信号强度	手机上网速度	手机上网稳定性	居民小区	办公室	高校	商业街	地铁	农村	高铁	其他，请注明
用户1	8	7	7	7	0	0	0	0	0	0	0	0

图 3.3　更改数据

网络信号差/没有信号	显示有信号上不了网	上网过程中网络时断时续或时快时	手机上网速度慢	其他，请注明
-1	-1	-1	-1	-1

图 3.4　原始数据

网络信号差/没有信号	显示有信号上不了网	上网过程中网络时断时续或时快时	手机上网速度慢	其他，请注明
0	0	0	0	0

图 3.5　更改数据

看视频卡顿	打游戏延时大	打开网页或APP图片慢	下载速度慢	手机支付较慢	其他，请注明
-1	-1	-1	-1	-1	-1

图 3.6　原始数据

看视频卡顿	打游戏延时大	打开网页或APP图片慢	下载速度慢	手机支付较慢	其他，请注明
0	0	0	0	0	0

图 3.7　更改数据

爱奇艺	优酷	腾讯视频	芒果TV	搜狐视频	抖音	快手	火山	咪咕视频	其他，请注明	APP小类视频备注	全部都卡顿
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1		-1

图 3.8　原始数据

爱奇艺	优酷	腾讯视频	芒果TV	搜狐视频	抖音	快手	火山	咪咕视频	其他，请注明	APP小类视频备注	全部都卡顿
0	0	0	0	0	0	0	0	0	0		0

图 3.9　更改数据

和平精英	王者荣耀	穿越火线	梦幻西游	龙之谷	梦幻诛仙	欢乐斗地主	部落冲突	炉石传说	阴阳师	其他，请注明	APP小类游戏备注	全部游戏都卡顿
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1		-1

图 3.10　原始数据

和平精英	王者荣耀	穿越火线	梦幻西游	龙之谷	梦幻诛仙	欢乐斗地主	部落冲突	炉石传说	阴阳师	其他，请注明	APP小类游戏备注	全部游戏都卡顿
0	0	0	0	0	0	0	0	0	0	0		0

图 3.11　更改数据

微信	手机QQ	淘宝	京东	百度	今日头条	新浪微博	拼多多	其他，请注明	APP小类上网备注	全部网页或APP都慢
-1	-1	-1	-1	-1	-1	-1	-1	-1		-1

图 3.12　原始数据

微信	手机QQ	淘宝	京东	百度	今日头条	新浪微博	拼多多	其他，请注明	APP小类上网备注	全部网页或APP都慢
0	0	0	0	0	0	0	0	0		0

图 3.13　更改数据

（2）4/5G

表 3.3　原始数据对应的更新数据

原始数据	更新数据
2G	0
4G	1
5G	2

（3）是/否

表 3.4　原始数据对应的更新数据

原始数据	更新数据
是	1
否	0

（4）客户星级标识

表 3.5　原始数据对应的更新数据

原始数据	更新数据
未评级	1
准星	2

一星	3
二星	4
三星	5
白金卡	6
钻石卡	7
银卡	8
金卡	9

（5）性别

表 3.6　原始数据对应的更新数据

原始数据	更新数据
女	0
男	1
性别不详	1

3.1.4　建立模型

利用决策树分析出的结果如下：

表 3.7　　各项属性的权重

属性	训练结果
手机上网整体满意度	0.2660
网络覆盖与信号强度	0.2565
手机上网速度	0.1725
其他，请注明	0.0263
脱网次数	0.0203
网络信号差/没有信号	0.0199
手机上网稳定性	0.0192
显示有信号上不了网	0.0191
居民小区	0.0177
上网过程中网络时断时续或时快时慢	0.0133
商业街	0.0121
高校	0.0105
打游戏延时大	0.0100
农村	0.0095
地铁	0.0090
下载速度慢	0.0078
其他，请注明.1	0.0077

打开网页或 APP 图片慢	0.0075
全部游戏都卡顿	0.0060
高铁	0.0059
看视频卡顿	0.0058
全部网页或 APP 都慢	0.0050
芒果 TV	0.0049
办公室	0.0047
手机 QQ	0.0045
京东	0.0044
淘宝	0.0041
其他，请注明.3	0.0039
百度	0.0039
其他，请注明.2	0.0037
优酷	0.0035
手机上网速度慢	0.0034
今日头条	0.0033
和平精英	0.0031
新浪微博，拼多多	0.0029
微信	0.0028
爱奇艺	0.0026
抖音	0.0025
其他，请注明.4	0.0023
全部都卡顿	0.0021
手机支付较慢	0.0015
火山	0.0015
咪咕视频	0.0013
腾讯视频	0.0012
快手	0.0009
梦幻诛仙	0.0008
阴阳师	0.0008
其他，请注明.5	0.0008
王者荣耀	0.0005
部落冲突	0.0003
穿越火线	0.00014
欢乐斗地主	0.00012
炉石传说	0.00008

龙之谷	0.00006

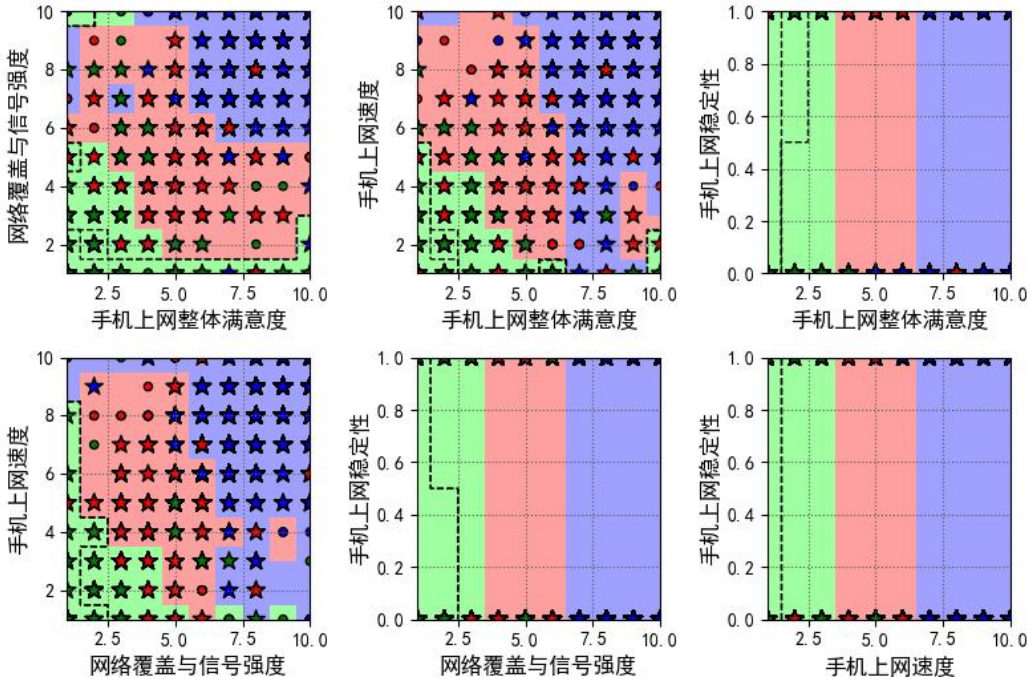

图 3.14　两特征组合的分类结果

3.2　利用模型进行预测

利用决策树进行预测，准确率如图 3.15、图 3.16、图 3.17 所示：

```
E:\anaconda\python.exe C:/Users/Lenovo/PycharmProjects/bigdata/test2.py
特征：  手机上网整体满意度  +  网络覆盖与信号强度  +  手机上网速度  +  手机上网稳定性
OOB Score: 0.6967846967846968
    训练集准确率: 70.4518%
    测试集准确率: 68.9459%
```

图 3.15　四个属性分析

```
特征：  手机上网整体满意度  +  网络覆盖与信号强度  +  手机上网速度
OOB Score: 0.6864061864061864
    训练集准确率: 70.0244%
    测试集准确率: 68.5185%
```

图 3.16 三个属性分析

```
特征：  手机上网整体满意度  +  网络覆盖与信号强度
OOB Score: 0.6923076923076923
    训练集准确率: 70.1465%
    测试集准确率: 68.9459%
```

图 3.17　最明显的二元组

3.3 结论

手机上网整体满意度受网络覆盖与信号强度、手机上网速度和手机上网稳定性影响最大，可以说这三者占据主导，为后续预测数据提供便利，其他属性均占比很小但各有其作用，总体为数据的计算提供有力帮助。

参 考 文 献

[1] 乔麟婷，决策树算法研究，课程教育研究，48，224-225，2018。

[2] 上交大和清华大学，人工智能/AI/神经网络/Python 基础，https://b23.tv/1iWArBF，2023.01.05。

[3] 许扬，基于 BP 神经网络和多因素权重分析的气热除冰温度影响因素研究，热力发电，（12）,131--140,2022。

[4] 郁美霞，基于 BP 神经网络的精密时基源校准预测模型，中国测试，2023。

[5] CJ-leaf，BP 神经网络预测 matlab 代码讲解与实现步骤，https://blog.csdn.net/qq_57971471/article/details/121766454，2013.01.08。

附录 A　语音数据初步分析

表 A.1　原始 sheet1 的各项满意度对整体的影响数据分析

网络覆盖与信号强度	数据数目	平均数	标准差	最小值	四分之一位数	二分之一位数	四分之三位数	最大值
	5433	8.34	2.41	1	8	9	10	10
语音通话清晰度	数据数目	平均数	标准差	最小值	四分之一位数	二分之一位数	四分之三位数	最大值
	5433	8.63	2.2	1	8	10	10	10
语音通话稳定性	数据数目	平均数	标准差	最小值	四分之一位数	二分之一位数	四分之三位数	最大值
	5433	8.42	2.37	1	8	10	10	10

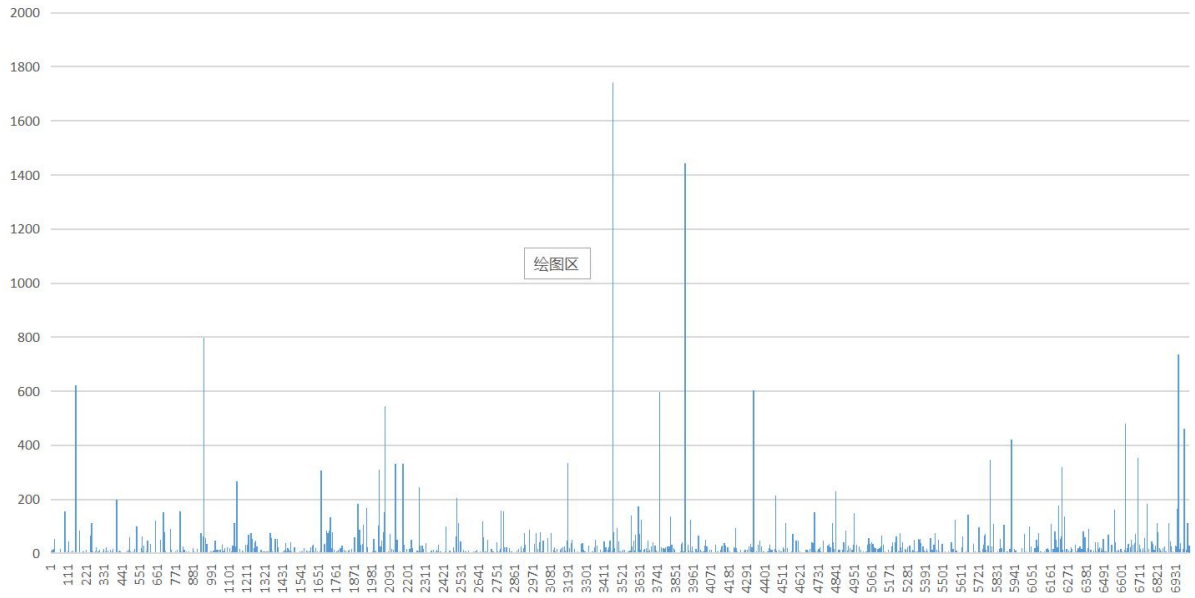

图 A.1　脱网次数数据分析

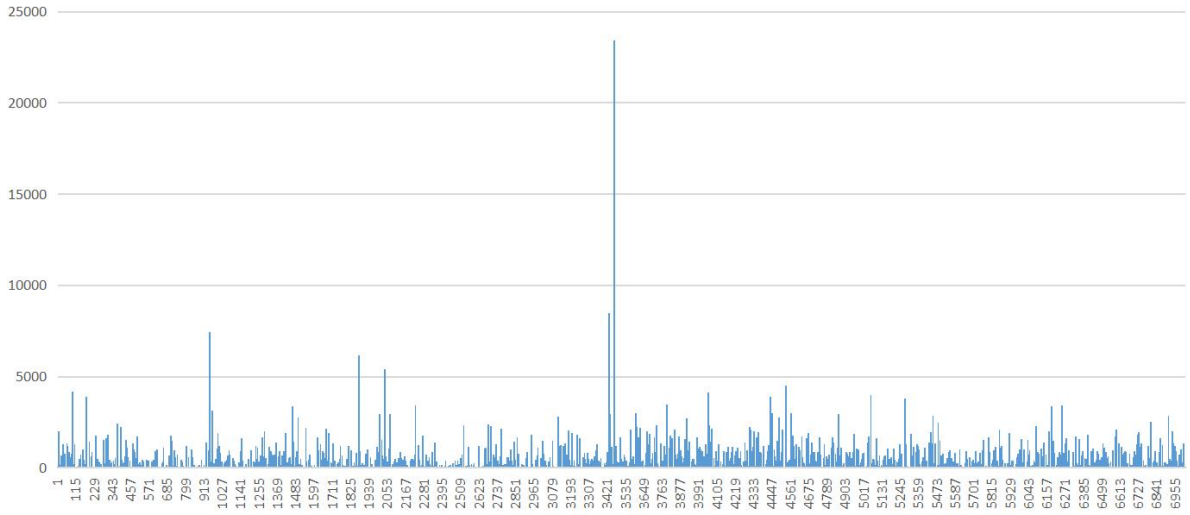

图 A.2　重定向次数数据分析

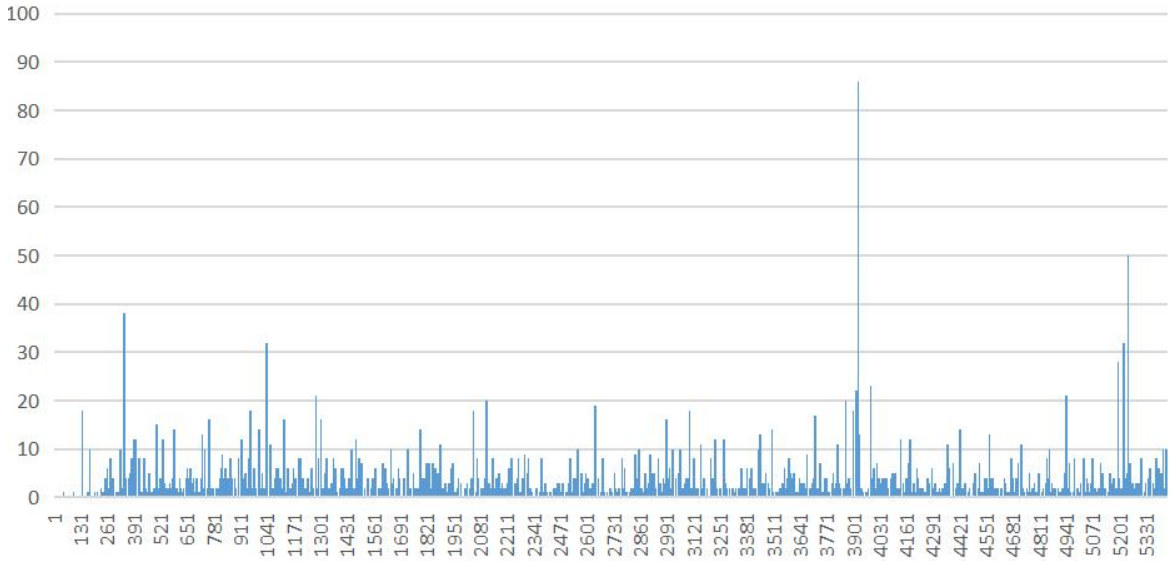

图 A.3　未接通掉话次数数据分析

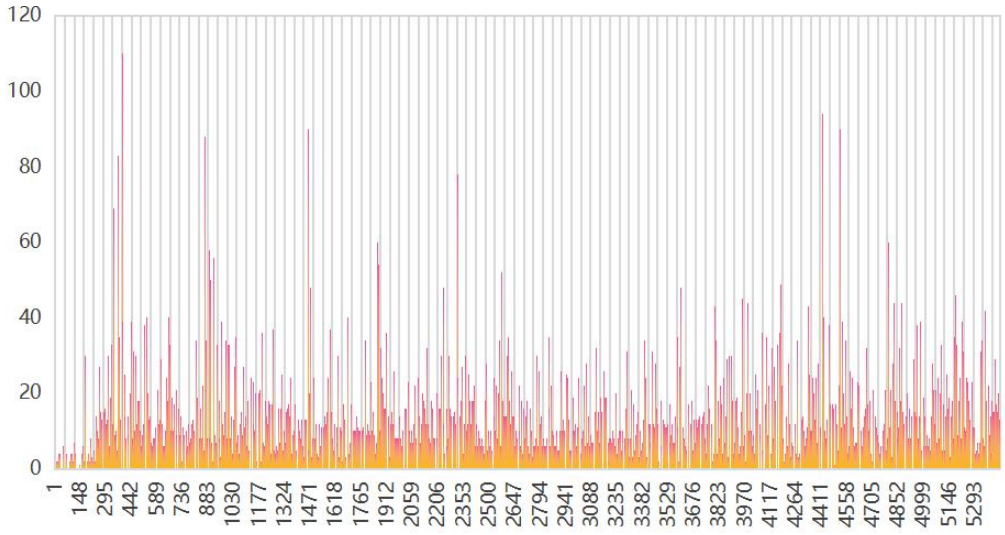

图 A.4　mos 质差次数数据分析

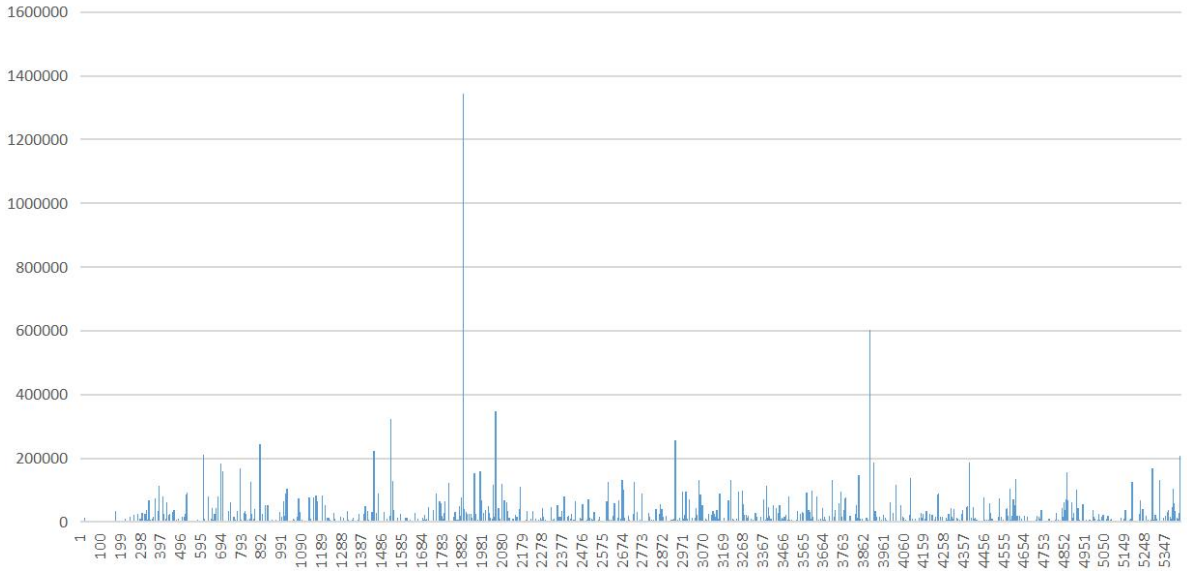

图 A.5　重定向驻留时长数据分析

附录 B　上网数据初步分析

表 B.1　原始 sheet2 的各项满意度对整体的影响数据分析

表二的各项满意度对整体的应影像数据分析								
网络覆盖与信号强度	数据数目	平均数	标准差	最小值	四分之一位数	二分之一位数	四分之三位数	最大值
	7020	7.62	2.69	1	6	8	10	10
手机上网速度	数据数目	平均数	标准差	最小值	四分之一位数	二分之一位数	四分之三位数	最大值
	7020	7.63	2.67	1	6	8	10	10
手机上网稳定性	数据数目	平均数	标准差	最小值	四分之一位数	二分之一位数	四分之三位数	最大值
	7020	7.55	2.74	1	6	8	10	10

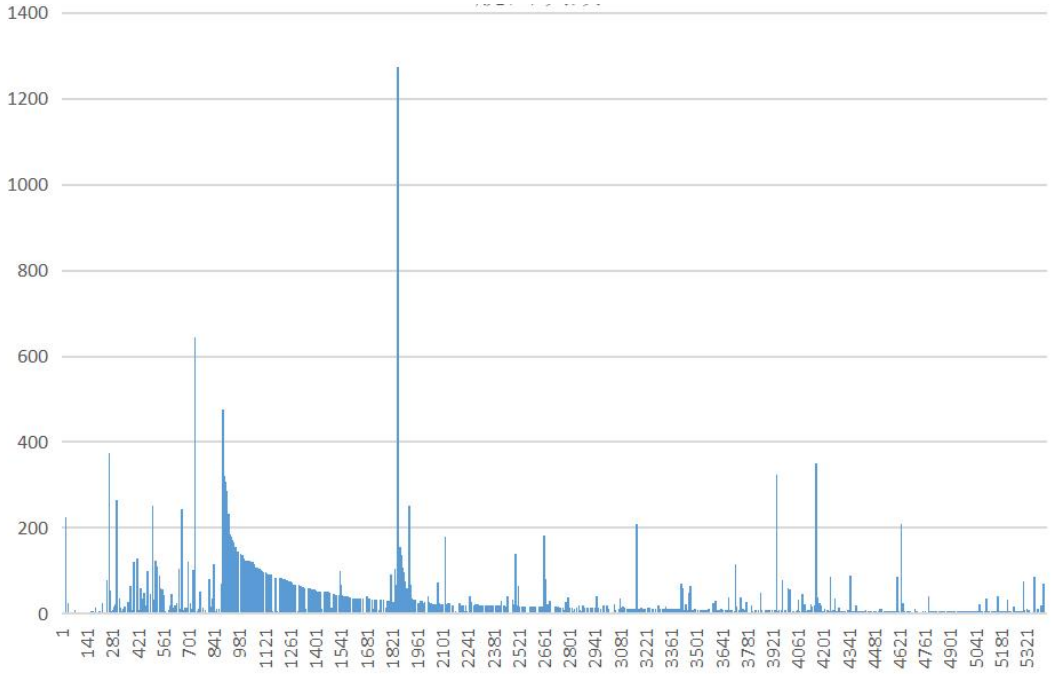

图 B.1 脱网次数数据分析

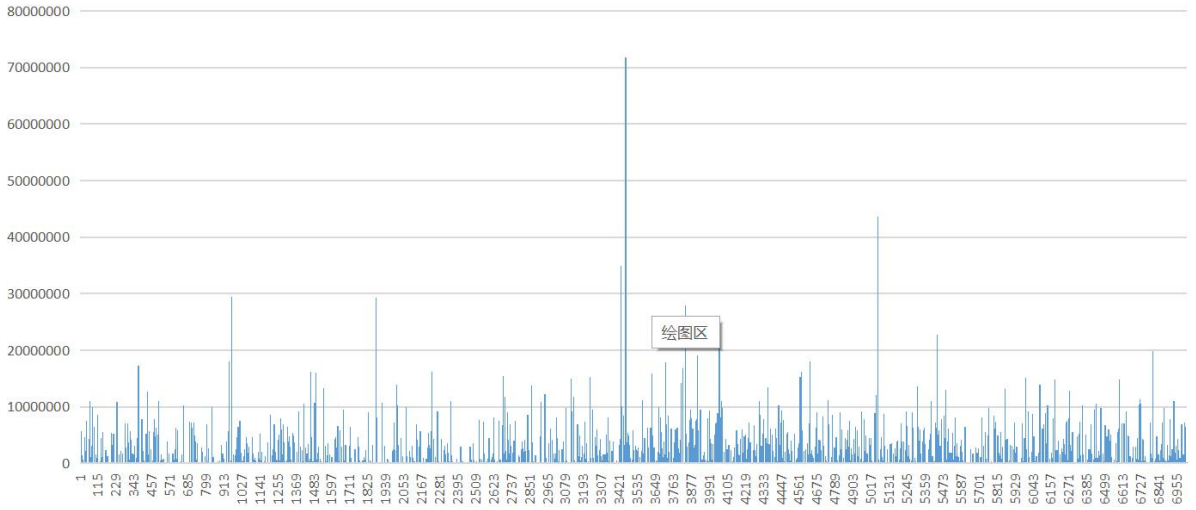

图 B.2　2G 驻留时长数据分析

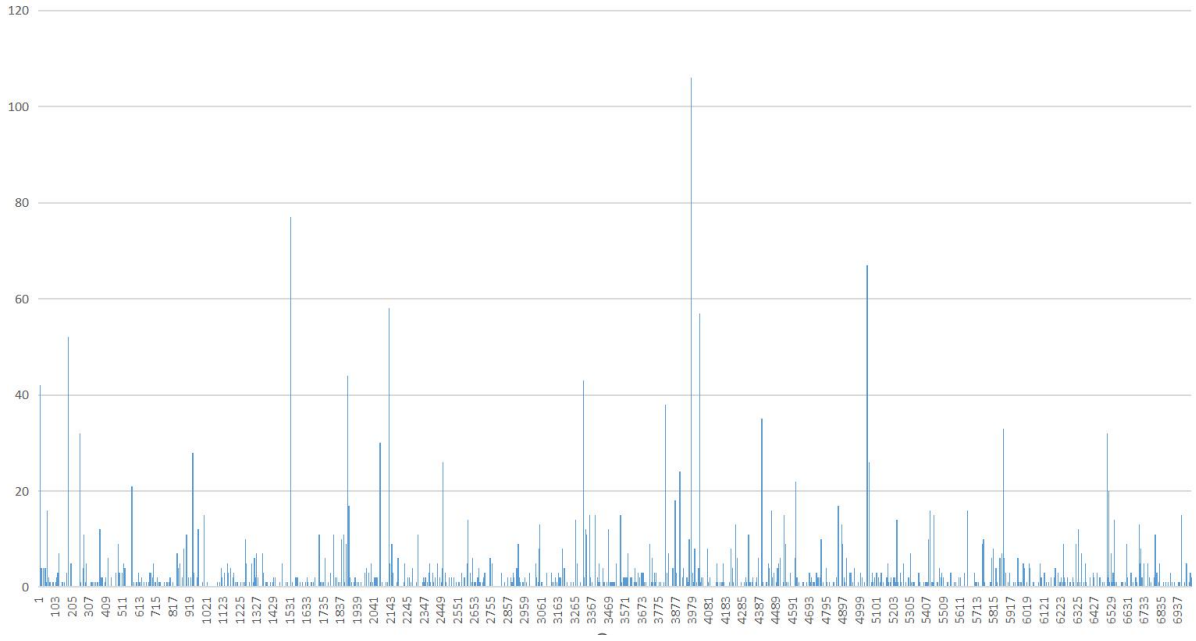

图 B.3　上网质差次数数据分析

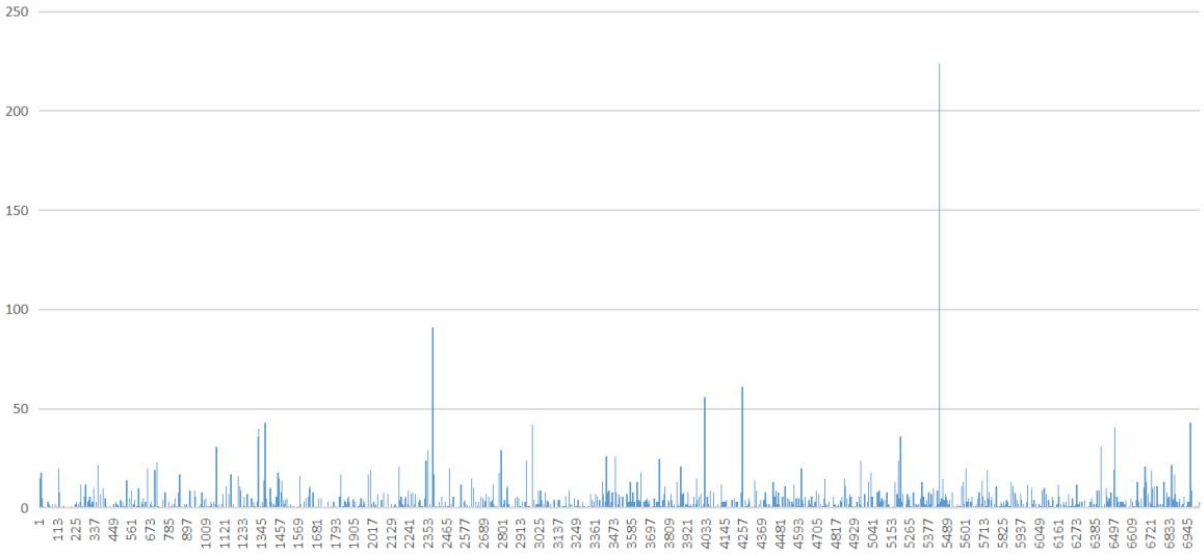

图 B.4　微信质差次数数据分析

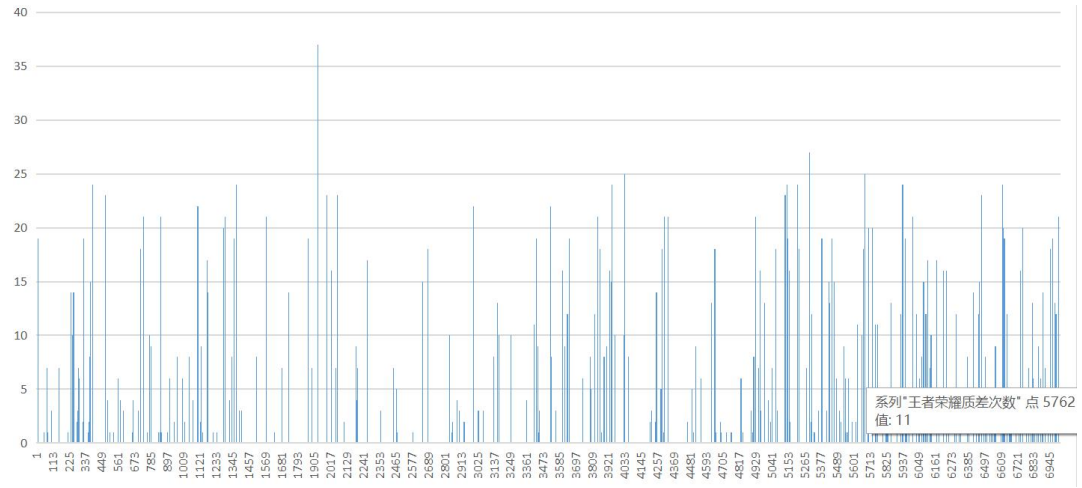

图 B.5　王者荣耀质差次数数据分析

附录 C　源代码

环境：Pycharm

1、数据预处理源代码

```
import pandas as pd
# 读取数据
df1 = pd.read_csv('./data/附件 1 语音业务用户满意度数据.csv')
df2 = pd.read_csv('./data/附件 2 上网业务用户满意度数据.csv')
df3 = pd.read_csv('./data/附件 3 语音业务用户满意度预测数据.csv')
df4 = pd.read_csv('./data/附件 4 上网业务用户满意度预测数据.csv')
# 各个文件的指标名称
c1, c2, c3, c4 = list(df1.columns), list(df2.columns), list(df3.columns), list(df4.columns)
# [markdown]
# 整体满意度与三个方面（网络覆盖与信号强度、语音通话清晰度、语音通话稳定性）之间的关系
'''
可以看出，三个方面的评分与整体满意度是高度相关的（当然，这是显然的结果）
所以我们下面可以仅考虑整体满意度的评分与其他影响因素之间的关系即可。
'''
df1.iloc[:, 1:5].corr()
df2.iloc[:, 1:5].corr()
df3.iloc[:,1:5].corr()
df4.iloc[:,1:5].corr()
# [markdown]
# 数据编码
'''
由于原始数据中含有较多的非数值型数据（顺序编码）
为了便于理解，我们将改变数据的原始编码方式（原始表述方式是用 1/2，或者-1/N 来表示“是否”，
我们习惯上用 0/1 表示）
手机型号在一定程度上反映的是用户个人的信息，不考虑
'''
# 附件一
temp1 = df1.copy()
del temp1['用户 id'], temp1['用户描述'], temp1['用户描述.1'], temp1['终端品牌'], temp1['终端品牌类型']

d4 = {1: 1, 2: 0}   # 是否遇到过网络问题  0 1
d5_12 = {-1: 0, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 98: 1}
d13_19 = {-1: 0, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 98: 1}
d245g = {'2G': 0, '4G': 1, '5G': 2}
d_yuyin = {'GSM': 0, 'CSFB': 1, 'VOLTE': 2, 'VoLTE': 2, 'EPSFB': 3, 'VONR': 4}
d_yesno = {0: 0, '是': 1, '否': 0}
d_level = {'未评级': 1, '准星': 2, '一星': 3, '二星': 4, '三星': 5, '白金卡': 6, '钻石卡': 7, '银卡': 8, '金卡': 9}
temp1.iloc[:, 4] = temp1.iloc[:, 4].map(d4)
for i in range(5, 13):
    temp1.iloc[:, i] = temp1.iloc[:, i].map(d5_12)
for i in range(13, 20):
    temp1.iloc[:, i] = temp1.iloc[:, i].map(d13_19)
temp1['重定向次数'] = temp1['重定向次数'].fillna(0)
temp1['重定向驻留时长'] = temp1['重定向驻留时长'].fillna(0)
```

```
temp1['4\\5G 用户'] = temp1['4\\5G 用户'].map(d245g)
temp1['语音方式'] = temp1['语音方式'].map(d_yuyin)
temp1['外省流量占比'] = temp1['外省流量占比'].fillna(0)
for i in ['是否关怀用户', '是否去过营业厅', '是否 4G 网络客户（本地剔除物联网）', '是否 5G 网络客户
', '是否实名登记用户']:
    temp1[i] = temp1[i].fillna(0).map(d_yesno)
temp1['客户星级标识'] = temp1['客户星级标识'].fillna(0).map(d_level)
temp1 = temp1.dropna()  # 只有五行有缺失，直接删除
# temp1.to_excel('f1.xlsx',index=None)
temp2 = df2.copy()
del temp2['用户'], temp2['场景备注数据'], temp2['现象备注数据'], temp2['APP 大类备注'], temp2['APP
小类视频备注'], temp2['APP 小类游戏备注']
del temp2['APP 小类上网备注'], temp2['终端类型'], temp2['操作系统'], temp2['终端制式'], temp2['终端
品牌'], temp2['终端品牌类型']
del temp2['当月高频通信分公司'], temp2['码号资源-激活时间'], temp2['码号资源-发卡时间'], temp2['
客户星级标识'], temp2['畅享套餐名称']
temp2 = temp2.fillna(0)
d4_55 = {-1: 0, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 1, 10: 1, 98: 1, 99: 1}  # 第 4-11 列对应字典
d_yesno = {0: 0, '是': 1, '否': 0}
d_sex = {'女': 0, '性别不详': 1, '男': 1}
for i in range(4, 56):
    temp2.iloc[:, i] = temp2.iloc[:, i].map(d4_55)
for i in [62, 63, 66, 67, 97, 99, 100, 101]:
    temp2.iloc[:, i] = temp2.iloc[:, i].map(d_yesno)
temp2['性别'] = temp2['性别'].map(d_sex)
# temp2.to_excel('f2.xlsx',index=None)
# [markdown]
#  相关矩阵
x1 = temp1.corr().iloc[:, [0]].apply(abs).sort_values('语音通话整体满意度', ascending=0)
x2 = temp2.corr().iloc[:, [0]].apply(abs).sort_values('手机上网整体满意度', ascending=0)
# x1.to_excel('x1_top.xlsx')
# x2.to_excel('x2_top.xlsx')
# [markdown]
#  根据 1 相关矩阵筛选变量
#  根据各个影响因素对于整体满意度的相关系数，我们将删去下面这些变量，并将筛选后的文件保
存到 f1_.xlsx 和 f2_.xlsx
print(temp1.shape, temp2.shape)
dd1 = ['当月 MOU', '语音通话-时长（分钟）', '前 3 月 MOU', '省际漫游-时长（分钟）', '外省流量占比
',
       '外省语音占比', 'GPRS-国内漫游-流量（KB）', 'GPRS 总流量（KB）', '套外流量（MB）', '
是否实名登记用户',
       '是否关怀用户', '客户星级标识', '是否去过营业厅', '语音方式', '是否 5G 网络客户', '资费投诉',
'套外流量费（元）', '4\\5G 用户', '家宽投诉']
for i in dd1:
    del temp1[i]
dd2 = ['年龄', '2G 驻留时长', '重定向次数', '王者荣耀使用天数', '搜狐视频', '王者荣耀质差次数', '是否
全月漫游用户',
       '梦幻西游', '王者荣耀 APP 使用流量', '游戏类 APP 使用天数', '当月 GPRS 资源使用量（GB）
', '大众点评使用流量',
```

```
        '王者荣耀使用次数', '是否校园套餐用户', '校园卡校园合约捆绑用户', '抖音使用流量（MB）',
'主套餐档位',
        '高单价超套客户（集团）', '视频类应用流量', '本年累计消费（元）', '是否不限量套餐到达用
户', '饿了么使用流量',
        '近 3 个月平均消费（元）', '近 3 个月平均消费（剔除通信账户支付）', '套外流量费（元）', '
网易系 APP 流量',
        '音乐类应用流量', '微信质差次数', '天猫使用流量', '阿里系 APP 流量', '小视频系 APP 流量', '
上网质差次数',
        '通信类应用流量', '游戏类 APP 使用流量', '游戏类应用流量', '校园卡无校园合约用户', '畅享
套餐档位',
        '快手使用流量', '优酷视频使用流量', '蜻蜓 FMAPP 使用流量', '高频高额超套客户（集团）',
        '游戏类 APP 使用次数', '今日头条使用流量', '美团外卖使用流量', '滴滴出行使用流量', '网页
类应用流量',
        '腾讯系 APP 流量', '邮箱类应用流量', '当月 MOU', '套外流量（MB）', '腾讯视频使用流量', '
是否 5G 网络客户', '性别']
for i in dd2:
    del temp2[i]
print(temp1.shape, temp2.shape)
# 3
temp3 = df3.copy()
del temp3['用户 id'], temp3['用户描述'], temp3['用户描述.1'], temp3['终端品牌'], temp3['终端品牌类型']
d4 = {1: 1, 2: 0}   # 是否遇到过网络问题  0 1
d5_12 = {-1: 0, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 98: 1}
d13_19 = {-1: 0, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 98: 1}
d245g = {'2G': 0, '4G': 1, '5G': 2}
d_yesno = {0: 0, '是': 1, '否': 0}
d_level = {'未评级': 1, '准星': 2, '一星': 3, '二星': 4, '三星': 5, '白金卡': 6, '钻石卡': 7, '银卡': 8, '金卡': 9}
temp3.iloc[:, 0] = temp3.iloc[:, 0].map(d4)
for i in range(1, 9):
    temp3.iloc[:, i] = temp3.iloc[:, i].map(d5_12)
for i in range(9, 16):
    temp3.iloc[:, i] = temp3.iloc[:, i].map(d13_19)
temp3['4\\5G 用户'] = temp3['4\\5G 用户'].map(d245g)
temp3['外省流量占比'] = temp3['外省流量占比'].fillna(0)
temp3['外省语音占比'] = temp3['外省语音占比'].fillna(0)
for i in ['是否关怀用户', '是否 4G 网络客户（本地剔除物联网）', '是否 5G 网络客户']:
    temp3[i] = temp3[i].fillna(0).map(d_yesno)
temp3['客户星级标识'] = temp3['客户星级标识'].fillna(0).map(d_level)
# 4
temp4 = df4.copy()
del temp4['用户 id']
del temp4['终端品牌'], temp4['终端品牌类型']
del temp4['客户星级标识']
temp4 = temp4.fillna(0)
d4_55 = {-1: 0, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 1, 10: 1, 98: 1, 99: 1}   # 第 4-11 列对应字典
d_yesno = {0: 0, '是': 1, '否': 0}
d_sex = {'女': 0, '性别不详': 1, '男': 1}
for i in range(1, 61):
    temp4.iloc[:, i] = temp4.iloc[:, i].map(d4_55)
```

```
for i in [0, 64, 67, 69, 81, 82]:
    temp4.iloc[:, i] = temp4.iloc[:, i].map(d_yesno)
temp4['性别'] = temp4['性别'].map(d_sex)
# x1.to_excel('x1_top.xlsx')
# x2.to_excel('x2_top.xlsx')
# [markdown]
# 根据 1 相关矩阵筛选变量
# 根据各个影响因素对于整体满意度的相关系数，我们将删去下面这些变量，并将筛选后的文件保存到 f1_.xlsx 和 f2_.xlsx
print(temp3.shape, temp4.shape)
dd1 = ['当月 MOU', '语音通话-时长（分钟）', '前 3 月 MOU', '省际漫游-时长（分钟）', '外省流量占比','外省语音占比', 'GPRS-国内漫游-流量（KB）', 'GPRS 总流量（KB）', '套外流量（MB）',
'是否关怀用户', '客户星级标识', '是否 5G 网络客户', '套外流量费（元）', '4\\5G 用户']
for i in dd1:
    del temp3[i]
dd2 = ['搜狐视频', '梦幻西游', '是否不限量套餐到达用户', '套外流量费（元）', '微信质差次数', '上网质差次数',
       '当月 MOU', '套外流量（MB）',  '是否 5G 网络客户', '性别']
for i in dd2:
    del temp4[i]
print(temp3.shape, temp4.shape)
#  保留测试集中有的属性
for i in temp3:
    if i not in temp1.columns:
        del temp3[i]
for i in temp4:
    if i not in temp2.columns:
        del temp4[i]
temp3.to_csv('pre1.csv',index=None)
temp4.to_csv('pre2.csv',index=None)
for i in temp1.columns:
    if i not in temp3.columns:
        del temp1[i]
for i in temp2.columns:
    if i not in temp4.columns:
        del temp2[i]
temp1.to_csv('train1_.csv',index=None)
temp2.to_csv('train2_.csv',index=None)
print(temp1.shape, temp3.shape)
print(temp2.shape, temp4.shape)
```

2、语音分析源代码

2.1 前导分析(展示数据各项指标)

```
import pandas as pd
import numpy as np
df=pd.read_excel("附件 1 语音业务用户满意度数据.xlsx")
df_load=pd.DataFrame({
    "网络覆盖与信号强度":df["网络覆盖与信号强度"].values,
    '语音通话清晰度':df["语音通话清晰度"].values,
    '语音通话稳定性':df["语音通话稳定性"].values,
})
```

```
print(df_load.describe())
#     count：数量统计，此列共有多少有效值
#     unique：不同的值有多少个
#     std：标准差
#     min：最小值
#     25%：四分之一分位数
#     50%：二分之一分位数
#     75%：四分之三分位数
#     max：最大值
#     mean：均值
```

2.2 计算权重

```
x=np.array(df_load.网络覆盖与信号强度.std()/df_load.网络覆盖与信号强度.mean())
x=np.append(x,df_load.语音通话清晰度.std()/df_load.语音通话清晰度.mean())
x=np.append(x,df_load.语音通话稳定性.std()/df_load.语音通话稳定性.mean())
for i in range(0,3):
      print(x[i]/x.sum())
      print(x)
```

2.3 决策树代码

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from itertools import combinations
if __name__ == "__main__":
    mpl.rcParams['font.sans-serif'] = ['SimHei']
    mpl.rcParams['axes.unicode_minus'] = False
    iris_feature = '网络覆盖与信号强度','语音通话清晰度','语音通话稳定性','是否遇到过网络问题','居民小区','办公室','高校','商业街','地铁','农村','高铁','其他，请注明','手机没有信号','有信号无法拨通','通话过程中突然中断','通话中有杂音、听不清、断断续续','串线','通话过程中一方听不见','其他，请注明.1','脱网次数','mos 质差次数','未接通掉话次数','重定向次数','重定向驻留时长','ARPU（家庭宽带）','是否 4G 网络客户（本地剔除物联网）','当月 ARPU','前 3 月 ARPU','当月欠费金额','前第 3 个月欠费金额'
    path = 'f1_.csv'   # 数据文件路径
    data = pd.read_csv(path, header=None)
    x_prime = data[list(range(1,31))]
    y = pd.Categorical(data[0]).codes
    x_prime_train, x_prime_test, y_train, y_test = train_test_split(x_prime, y, test_size=0.3, random_state=0)
    pairs = [c for c in combinations(range(1,4), 2)]
    feature_pairs = []
    num = len(pairs)
    for i in range(num):
        feature_pairs.append(list(pairs[i]))
    plt.figure(figsize=(8, 6), facecolor='#FFFFFF')
    for i, pair in enumerate(feature_pairs):
```

```
            # 准备数据
            x_train = x_prime_train[pair]
            x_test = x_prime_test[pair]
            # 决策树学习
            model = RandomForestClassifier(n_estimators=100, criterion='entropy', max_depth=10,
oob_score=True)
            model.fit(x_train, y_train)
            # 画图
            N, M = 500, 500  # 横纵各采样多少个值
            x1_min, x2_min = x_train.min()
            x1_max, x2_max = x_train.max()
            t1 = np.linspace(x1_min, x1_max, N)
            t2 = np.linspace(x2_min, x2_max, M)
            x1, x2 = np.meshgrid(t1, t2)  # 生成网格采样点
            x_show = np.stack((x1.flat, x2.flat), axis=1)  # 测试点
            # 训练集上的预测结果
            y_train_pred = model.predict(x_train)
            acc_train = accuracy_score(y_train, y_train_pred)
            y_test_pred = model.predict(x_test)
            acc_test = accuracy_score(y_test, y_test_pred)
            print('特征：', iris_feature[pair[0]], ' + ', iris_feature[pair[1]])
            print('OOB Score:', model.oob_score_)
            print('\t 训练集准确率: %.4f%%' % (100*acc_train))
            print('\t 测试集准确率: %.4f%%\n' % (100*acc_test))
            cm_light = mpl.colors.ListedColormap(['#A0FFA0', '#FFA0A0', '#A0A0FF'])
            cm_dark = mpl.colors.ListedColormap(['g', 'r', 'b'])
            y_hat = model.predict(x_show)
            y_hat = y_hat.reshape(x1.shape)
            plt.subplot(2, 3, i+1)
            plt.contour(x1, x2, y_hat, colors='k', levels=[0, 1], antialiased=True, linestyles='--', linewidths=1)
            plt.pcolormesh(x1, x2, y_hat, cmap=cm_light)  # 预测值
            plt.scatter(x_train[pair[0]], x_train[pair[1]], c=y_train, s=20, edgecolors='k', cmap=cm_dark,
label='训练集')
            plt.scatter(x_test[pair[0]], x_test[pair[1]], c=y_test, s=100, marker='*', edgecolors='k',
cmap=cm_dark, label='测试集')
            plt.xlabel(iris_feature[pair[0]], fontsize=12)
            plt.ylabel(iris_feature[pair[1]], fontsize=12)
            # plt.legend(loc='upper right', fancybox=True, framealpha=0.3)
            plt.xlim(x1_min, x1_max)
            plt.ylim(x2_min, x2_max)
            plt.grid(b=True, ls=':', color='#606060')
        plt.suptitle('随机森林对数据两特征组合的分类结果', fontsize=15)
        plt.tight_layout(1, rect=(0, 0, 1, 0.95))       # (left, bottom, right, top)
        plt.show()
```

2.4 决策树源代码（改进）

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from itertools import combinations
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
iris_feature = '网络覆盖与信号强度','语音通话清晰度','语音通话稳定性','当月 ARPU'
path = 'f1_.csv'  # 数据文件路径
data = pd.read_csv(path, header=None)
x_prime = data[[1, 2, 3, 27]]
x_prime.columns = [0, 1, 2, 3]
y = pd.Categorical(data[0]).codes
x_prime_train, x_prime_test, y_train, y_test = train_test_split(x_prime, y, test_size=0.3, random_state=0)
feature_pairs = [[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]
plt.figure(figsize=(8, 6), facecolor='#FFFFFF')
for i, pair in enumerate(feature_pairs):
    # 准备数据
    x_train = x_prime_train[pair]
    x_test = x_prime_test[pair]
    # 决策树学习
    model = RandomForestClassifier(n_estimators=100, criterion='entropy', max_depth=5,
oob_score=True)
    model.fit(x_train, y_train)
    # 画图
    N, M = 500, 500  # 横纵各采样多少个值
    x1_min, x2_min = x_train.min()
    x1_max, x2_max = x_train.max()
    t1 = np.linspace(x1_min, x1_max, N)
    t2 = np.linspace(x2_min, x2_max, M)
    x1, x2 = np.meshgrid(t1, t2)  # 生成网格采样点
    x_show = np.stack((x1.flat, x2.flat), axis=1)  # 测试点
    # 训练集上的预测结果
    y_train_pred = model.predict(x_train)
    acc_train = accuracy_score(y_train, y_train_pred)
    y_test_pred = model.predict(x_test)
    acc_test = accuracy_score(y_test, y_test_pred)
    print('特征：', iris_feature[pair[0]], ' + ', iris_feature[pair[1]])
    print('OOB Score:', model.oob_score_)
    print('\t 训练集准确率: %.4f%%' % (100 * acc_train))
    print('\t 测试集准确率: %.4f%%\n' % (100 * acc_test))
    cm_light = mpl.colors.ListedColormap(['#A0FFA0', '#FFA0A0', '#A0A0FF'])
    cm_dark = mpl.colors.ListedColormap(['g', 'r', 'b'])
    y_hat = model.predict(x_show)
    y_hat = y_hat.reshape(x1.shape)
    plt.subplot(2, 3, i + 1)
    plt.contour(x1, x2, y_hat, colors='k', levels=[0, 1], antialiased=True, linestyles='--', linewidths=1)
    plt.pcolormesh(x1, x2, y_hat, cmap=cm_light)  # 预测值
    plt.scatter(x_train[pair[0]], x_train[pair[1]], c=y_train, s=20, edgecolors='k', cmap=cm_dark, label='训
练集')
    plt.scatter(x_test[pair[0]], x_test[pair[1]], c=y_test, s=100, marker='*', edgecolors='k', cmap=cm_dark,
                label='测试集')
```

```
        plt.xlabel(iris_feature[pair[0]], fontsize=12)
        plt.ylabel(iris_feature[pair[1]], fontsize=12)
        # plt.legend(loc='upper right', fancybox=True, framealpha=0.3)
        plt.xlim(x1_min, x1_max)
        plt.ylim(x2_min, x2_max)
        plt.grid(b=True, ls=':', color='#606060')
plt.suptitle('两特征组合的分类结果', fontsize=15)
plt.tight_layout(rect=(0, 0, 1, 0.95))    # (left, bottom, right, top)
plt.show()
```

2.5 三元决策树分析（说明二元的合理性）

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from itertools import combinations
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
iris_feature = '网络覆盖与信号强度','语音通话清晰度','语音通话稳定性','当月 ARPU'
path = 'f1_.csv'    # 数据文件路径
data = pd.read_csv(path, header=None)
x_prime = data[[1, 2, 3, 27]]
x_prime.columns = [0, 1, 2, 3]
y = pd.Categorical(data[0]).codes
x_prime_train, x_prime_test, y_train, y_test = train_test_split(x_prime, y, test_size=0.3, random_state=0)
feature_pairs = [[0, 1, 2]]
plt.figure(figsize=(8, 6), facecolor='#FFFFFF')
for i, pair in enumerate(feature_pairs):
    # 准备数据
    x_train = x_prime_train[pair]
    x_test = x_prime_test[pair]
    # 决策树学习
    model = RandomForestClassifier(n_estimators=100, criterion='entropy', max_depth=5,
oob_score=True)
    model.fit(x_train, y_train)
    y_train_pred = model.predict(x_train)
    acc_train = accuracy_score(y_train, y_train_pred)
    y_test_pred = model.predict(x_test)
    acc_test = accuracy_score(y_test, y_test_pred)
    print('特征：', iris_feature[pair[0]], ' + ', iris_feature[pair[1]])
    print('OOB Score:', model.oob_score_)
    print('\t 训练集准确率: %.4f%%' % (100 * acc_train))
    print('\t 测试集准确率: %.4f%%\n' % (100 * acc_test))
```

2.6 神经网络（过拟合过于严重）

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
from sklearn.model_selection import train_test_split   # 引入训练集、测试集划分函数
import torch
```

```
import torch.nn.functional as Fun
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
# 0. 超参数设置
lr = 0.00002
epochs = 300
n_feature = 22
n_hidden = 300
n_output = 10
# 1. 数据准备
train1 = pd.read_csv('train1_.csv', header=None)
# train2 = pd.read_csv('train2_.csv', header=None)
pre1 = pd.read_csv('pre1.csv', header=None)
# pre2 = pd.read_csv('pre2.csv', header=None)
y1 = pd.read_csv('f1_.csv', header=None)
# y2 = pd.read_csv('f2_.csv', header=None)
x_prime = train1
y = pd.Categorical(y1[1]).codes
x_p_train, x_p_test, y_train, y_test = train_test_split(x_prime, y, test_size=0.2, random_state=22)
x_train = np.array(x_p_train)
x_pre = np.array(pre1)
x_test = np.array(x_p_test)
x_train = torch.FloatTensor(x_train)
y_train = torch.LongTensor(y_train)
x_test = torch.FloatTensor(x_test)
y_test = torch.LongTensor(y_test)
x_pre = torch.FloatTensor(x_pre)
# 2. 定义 BP 神经网络
class bpnnModel(torch.nn.Module):
    def __init__(self, n_feature, n_hidden, n_output):
        super(bpnnModel, self).__init__()
        self.hidden = torch.nn.Linear(n_feature, n_hidden)  # 定义隐藏层网络
        self.out = torch.nn.Linear(n_hidden, n_output)  # 定义输出层网络
    def forward(self, x):
        x = Fun.relu(self.hidden(x))  # 隐藏层的激活函数,采用 relu,也可以采用 sigmod,tanh
        out = Fun.softmax(self.out(x), dim=1)  # 输出层 softmax 激活函数
        return out
# 3. 定义优化器损失函数
net = bpnnModel(n_feature=n_feature, n_hidden=n_hidden, n_output=n_output)
optimizer = torch.optim.Adam(net.parameters(), lr=lr)  # 优化器选用随机梯度下降方式
loss_func = torch.nn.CrossEntropyLoss()  # 对于多分类一般采用的交叉熵损失函数
# 4. 训练数据
loss_steps = np.zeros(epochs)
accuracy_steps = np.zeros(epochs)
for epoch in range(epochs):
    y_pred = net(x_train)  # 前向过程
    loss = loss_func(y_pred, y_train)  # 输出与 label 对比
    optimizer.zero_grad()  # 梯度清零
    loss.backward()  # 反向传播
    optimizer.step()  # 使用梯度优化器
    loss_steps[epoch] = loss.item()  # 保存 loss
```

```
with torch.no_grad():
    y_pred = net(x_test)
    y0 = net(x_pre)
    y = torch.argmax(y0, dim=1)
    correct = (torch.argmax(y_pred, dim=1) == y_test).type(torch.FloatTensor)
    accuracy_steps[epoch] = correct.mean()
print("预测准确率", accuracy_steps[-1])
# 5 绘制损失函数和精度
fig_name = '网络覆盖与信号强度(语音)'
fontsize = 15
fig, (ax1, ax2) = plt.subplots(2, figsize=(15, 12), sharex=True)
ax1.plot(accuracy_steps)
ax1.set_ylabel("test accuracy", fontsize=fontsize)
ax1.set_title(fig_name, fontsize='xx-large')
ax2.plot(loss_steps)
ax2.set_ylabel("train loss", fontsize=fontsize)
ax2.set_xlabel("epochs", fontsize=fontsize)
plt.tight_layout()
plt.savefig(fig_name + '.png')
plt.show()
for j in range(len(y)):
    y[j] = y[j] + 1
data = pd.DataFrame(y)
data.to_csv('PD 网络覆盖与信号强度(语音).csv', index=None)
```

3、上网分析源代码

3.1 决策树源代码

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from itertools import combinations
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
iris_feature = '手机上网整体满意度', '网络覆盖与信号强度', '手机上网速度', '手机上网稳定性'
path = 'f2_.csv'  # 数据文件路径
data = pd.read_csv(path, header=None)
x_prime = data[[1, 2, 3, 4]]
x_prime.columns = [0, 1, 2, 3]
y = pd.Categorical(data[0]).codes
x_prime_train, x_prime_test, y_train, y_test = train_test_split(x_prime, y, test_size=0.3, random_state=0)
feature_pairs = [[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]
plt.figure(figsize=(8, 6), facecolor='#FFFFFF')
for i, pair in enumerate(feature_pairs):
    # 准备数据
    x_train = x_prime_train[pair]
    x_test = x_prime_test[pair]
```

```
    # 决策树学习
    model = RandomForestClassifier(n_estimators=100, criterion='entropy', max_depth=5,
oob_score=True)
    model.fit(x_train, y_train)
    # 画图
    N, M = 500, 500  # 横纵各采样多少个值
    x1_min, x2_min = x_train.min()
    x1_max, x2_max = x_train.max()
    t1 = np.linspace(x1_min, x1_max, N)
    t2 = np.linspace(x2_min, x2_max, M)
    x1, x2 = np.meshgrid(t1, t2)  # 生成网格采样点
    x_show = np.stack((x1.flat, x2.flat), axis=1)  # 测试点
    # 训练集上的预测结果
    y_train_pred = model.predict(x_train)
    acc_train = accuracy_score(y_train, y_train_pred)
    y_test_pred = model.predict(x_test)
    acc_test = accuracy_score(y_test, y_test_pred)
    print('特征：', iris_feature[pair[0]], ' + ', iris_feature[pair[1]])
    print('OOB Score:', model.oob_score_)
    print('\t 训练集准确率: %.4f%%' % (100 * acc_train))
    print('\t 测试集准确率: %.4f%%\n' % (100 * acc_test))
    cm_light = mpl.colors.ListedColormap(['#A0FFA0', '#FFA0A0', '#A0A0FF'])
    cm_dark = mpl.colors.ListedColormap(['g', 'r', 'b'])
    y_hat = model.predict(x_show)
    y_hat = y_hat.reshape(x1.shape)
    plt.subplot(2, 3, i + 1)
    plt.contour(x1, x2, y_hat, colors='k', levels=[0, 1], antialiased=True, linestyles='--', linewidths=1)
    plt.pcolormesh(x1, x2, y_hat, cmap=cm_light)  # 预测值
    plt.scatter(x_train[pair[0]], x_train[pair[1]], c=y_train, s=20, edgecolors='k', cmap=cm_dark, label='训
练集')
    plt.scatter(x_test[pair[0]], x_test[pair[1]], c=y_test, s=100, marker='*', edgecolors='k', cmap=cm_dark,
                label='测试集')
    plt.xlabel(iris_feature[pair[0]], fontsize=12)
    plt.ylabel(iris_feature[pair[1]], fontsize=12)
    # plt.legend(loc='upper right', fancybox=True, framealpha=0.3)
    plt.xlim(x1_min, x1_max)
    plt.ylim(x2_min, x2_max)
    plt.grid(b=True, ls=':', color='#606060')
plt.suptitle('两特征组合的分类结果', fontsize=15)
plt.tight_layout(rect=(0, 0, 1, 0.95))  # (left, bottom, right, top)
plt.show()
```

3.2 三元决策树分析（说明二元的合理性）

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

```
from itertools import combinations
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
iris_feature = '手机上网整体满意度', '网络覆盖与信号强度', '手机上网速度', '手机上网稳定性'
path = 'f2_.csv'  # 数据文件路径
data = pd.read_csv(path, header=None)
x_prime = data[[1, 2, 3, 4]]
x_prime.columns = [0, 1, 2, 3]
y = pd.Categorical(data[0]).codes
x_prime_train, x_prime_test, y_train, y_test = train_test_split(x_prime, y, test_size=0.3, random_state=0)
feature_pairs = [[0, 1, 2]]
plt.figure(figsize=(8, 6), facecolor='#FFFFFF')
for i, pair in enumerate(feature_pairs):
    # 准备数据
    x_train = x_prime_train[pair]
    x_test = x_prime_test[pair]
    # 决策树学习
    model = RandomForestClassifier(n_estimators=100, criterion='entropy', max_depth=5,
oob_score=True)
    model.fit(x_train, y_train)
    # # 画图
    # N, M = 500, 500  # 横纵各采样多少个值
    # x1_min, x2_min = x_train.min()
    # x1_max, x2_max = x_train.max()
    # t1 = np.linspace(x1_min, x1_max, N)
    # t2 = np.linspace(x2_min, x2_max, M)
    # x1, x2 = np.meshgrid(t1, t2)  # 生成网格采样点
    # x_show = np.stack((x1.flat, x2.flat), axis=1)  # 测试点
    # 训练集上的预测结果
    y_train_pred = model.predict(x_train)
    acc_train = accuracy_score(y_train, y_train_pred)
    y_test_pred = model.predict(x_test)
    acc_test = accuracy_score(y_test, y_test_pred)
    print('特征：', iris_feature[pair[0]], ' + ', iris_feature[pair[1]], ' + ', iris_feature[pair[2]])
    print('OOB Score:', model.oob_score_)
    print('\t 训练集准确率: %.4f%%' % (100 * acc_train))
    print('\t 测试集准确率: %.4f%%\n' % (100 * acc_test))
#
#     cm_light = mpl.colors.ListedColormap(['#A0FFA0', '#FFA0A0', '#A0A0FF'])
#     cm_dark = mpl.colors.ListedColormap(['g', 'r', 'b'])
#     y_hat = model.predict(x_show)
#     y_hat = y_hat.reshape(x1.shape)
#     plt.subplot(2, 3, i + 1)
#     plt.contour(x1, x2, y_hat, colors='k', levels=[0, 1], antialiased=True, linestyles='--', linewidths=1)
#     plt.pcolormesh(x1, x2, y_hat, cmap=cm_light)  # 预测值
#     plt.scatter(x_train[pair[0]], x_train[pair[1]], c=y_train, s=20, edgecolors='k', cmap=cm_dark, label='
训练集')
#             plt.scatter(x_test[pair[0]], x_test[pair[1]], c=y_test, s=100, marker='*', edgecolors='k',
cmap=cm_dark,
#                         label='测试集')
#     plt.xlabel(iris_feature[pair[0]], fontsize=12)
```

```
#         plt.ylabel(iris_feature[pair[1]], fontsize=12)
#         # plt.legend(loc='upper right', fancybox=True, framealpha=0.3)
#         plt.xlim(x1_min, x1_max)
#         plt.ylim(x2_min, x2_max)
#         plt.grid(b=True, ls=':', color='#606060')
# plt.suptitle('两特征组合的分类结果', fontsize=15)
# plt.tight_layout(rect=(0, 0, 1, 0.95))    # (left, bottom, right, top)
# plt.show()
```

环境：Matlab

神经网络模型源代码

```
%% 此程序为 matlab 编程实现的 BP 神经网络
% 清空环境变量
clear
close all
clc
%%第一步 读取数据
input=randi([1 20],200,2);   %载入输入数据
output=input(:,1)+input(:,2);   %载入输出数据
%% 第二步 设置训练数据和预测数据
input_train = input(1:190,:)';
output_train =output(1:190,:)';
input_test = input(191:200,:)';
output_test =output(191:200,:)';
%节点个数
inputnum=2; % 输入层节点数量
hiddennum=5;% 隐含层节点数量
outputnum=1; % 输出层节点数量
%% 第三本 训练样本数据归一化
[inputn,inputps]=mapminmax(input_train);%归一化到[-1,1]之间，inputps 用来作下一次同样的归一化
[outputn,outputps]=mapminmax(output_train);
%% 第四步 构建 BP 神经网络
net=newff(inputn,outputn,hiddennum,{'tansig','purelin'},'trainlm');% 建立模型，传递函数使用 purelin，
采用梯度下降法训练
W1= net. iw{1, 1};%输入层到中间层的权值
B1 = net.b{1};%中间各层神经元阈值
W2 = net.lw{2,1};%中间层到输出层的权值
B2 = net. b{2};%输出层各神经元阈值
%% 第五步 网络参数配置（ 训练次数，学习速率，训练目标最小误差等）
net.trainParam.epochs=1000;          % 训练次数，这里设置为 1000 次
```

```
net.trainParam.lr=0.01;                          % 学习速率，这里设置为 0.01
net.trainParam.goal=0.00001;                     % 训练目标最小误差，这里设置为 0.00001
%% 第六步 BP 神经网络训练
net=train(net,inputn,outputn);%开始训练，其中 inputn,outputn 分别为输入输出样本
%% 第七步 测试样本归一化
inputn_test=mapminmax('apply',input_test,inputps);% 对样本数据进行归一化
%% 第八步 BP 神经网络预测
an=sim(net,inputn_test); %用训练好的模型进行仿真
%% 第九步 预测结果反归一化与误差计算
test_simu=mapminmax('reverse',an,outputps); %把仿真得到的数据还原为原始的数量级
error=test_simu-output_test;          %预测值和真实值的误差
%%第十步 真实值与预测值误差比较
figure('units','normalized','position',[0.119 0.2 0.38 0.5])
plot(output_test,'bo-')
hold on
plot(test_simu,'r*-')
hold on
plot(error,'square','MarkerFaceColor','b')
legend('期望值','预测值','误差')
xlabel('数据组数')
ylabel('样本值')
title('BP 神经网络测试集的预测值与实际值对比图')
[c,l]=size(output_test);
MAE1=sum(abs(error))/l;
MSE1=error*error'/l;
RMSE1=MSE1^(1/2);
disp(['-----------------------误差计算--------------------------'])
disp(['隐含层节点数为',num2str(hiddennum),'时的误差结果如下：'])
disp(['平均绝对误差 MAE 为：',num2str(MAE1)])
disp(['均方误差 MSE 为：              ',num2str(MSE1)])
disp(['均方根误差 RMSE 为：     ',num2str(RMSE1)])
% 附
eval(['web ', char([104 116 116 112 115 58 47 47 98 108 111 103 46 99 115 100 110 46
    110 101 116 47 113 113 95 53 55 57 55 49 52 55 49 47 97 114 116 105 99
    108 101 47 100 101 116 97 105 108 115 47 49 50 49 55 54 55 48 48 52 32
    45 98 114 111 119 115 101 114])])
eval(['web                                                                          ',
char([104,116,116,112,115,58,47,47,109,98,100,46,112,117,98,47,111,47,98,114,101,97,100,47,109,98,10
0,45,89,90,109,84,109,112,116,118,32,45,98,114,111,119,115,101,114])])
```

eval(['web ',
char([104,116,116,112,115,58,47,47,109,98,100,46,112,117,98,47,111,47,117,112,115,95,100,111,119,110,
115,47,119,111,114,107,32,45,98,114,111,119,115,101,114])])