摘 要

情绪时时刻刻伴随着人们的生活，并且每分每秒都在影响着人们的工作和生活的状态。而随着我国近些年经济快速发展，人们也逐渐意识到情绪对于生活的重要影响，越来越多的人们开始关注自身情绪的状态和变化。

由于人的情绪与人体的各项数据密切相关，安卓设备作为天然的数据收集器条件优越，本文以安卓手机为数据收集的载体，通过收集用户的运动数据、环境数据、设备使用情况等数据，基于机器学习分类器原理构造情绪分类模型，从而实现对用户情绪的识别。文章首先介绍了情绪分析的研究进展，并指出了不同情绪研究的优缺点。然后，对实验用到的平台和主要技术进行了简要的介绍，接着又对机器学习分类器的原理进行了分析。然后，结合分析和讨论详细介绍了本实验的数据收集、数据预处理、特征工程、模型构建过程，其中模型构建过程中基于 K 近邻算法、SVM 算法、朴素贝叶斯算法，通过参数调整构建了三类七种不同的模型。

最后，本文还针对构建出来的模型，进行了准确性测试和性能测试，测试的结果表明，构建出来的情绪分析模型对于情绪的识别较为准确，同时具有良好的性能，达到了实验的预期目的。

关键字： 机器学习 情绪分析 分类算法 安卓

ABSTRACT

Emotions are accompanied by people’s lives all the time, and affects the people's work and life every minute. With the rapid economic development in China in recent years, people are gradually realizing the important influence of emotion on life. More and more people begin to pay attention to the state and changes of their own emotions.

People's emotions are closely related to people's data，and Android devices have advantage in collecting data information. For these reasons, we choose Android devices as carriers to collect data information, including user’s motion data, environmental data, and device usage data. The emotion classification model is constructed to recognize user emotions based on the principles of machine learning classifier. The article first introduced the research progress of emotion analysis and pointed out the advantages and disadvantages of different emotion studies. Secondly, the platform and main technology used in the experiment are briefly introduced. Then the principle of the machine learning classifier is analyzed. Next, the data collection, data preprocessing, feature engineering, and model construction process of this experiment are described in detail in connection with the analysis and discussion. The K-nearest neighbor algorithm, SVM algorithm, and Naive Bayes algorithm are used to build the models, including three types of seven different models.

Finally, this article also carried out the accuracy test and the performance test on the constructed model. The test results show that the constructed emotion analysis model has good accuracy in the recognition of emotions, and has good performance at the same time, achieving the intended purpose of the experiment.

Keywords: Machine learning Emotion analysis Classifier Android

目 录

第一章 绪论

从 2001 年世界上第一部智能手机诞生开始，智能手机的发展速度可谓是日新月异。从以诺基亚为代表的塞班一家独大，到 Android，ios，WindowsOS 三足鼎立，到如今 Android 和 ios 几乎已经占领了智能手机操作系统 95%以上的份额，而其中又以 Android 份额最多。伴随着智能手机数量的爆发式增长，手机的硬件也得到了极大的升级，功能也变得更加丰富。手机已经不仅仅是用来打电话，发信息，而是集通信，娱乐，办公，社交等诸多功能于一体的智能终端设备。随着手机内置的传感器越来越多，手机所能收集到的信息也越来越多，而且因为多数人都有随身携带的习惯，让手机俨然成为一个天然的“人体数据收集站”。而且，近些年机器学习热度越来越高，“情绪分析”作为其中的一个重要方向也吸引了大量研究人员的兴趣。鉴于智能手机的优势和特性，通过分析智能手机的数据来进行情绪分析，正逐渐成为机器学习的一个研究热点。

1.1 研究背景

情绪是人在探索和发现周围环境时，对外界环境刺激做出的心理或生理反应，是一种综合了人的思想，感觉，和行为的状态[1]。常见的情绪有高兴，平静，悲伤，生气等。积极的情绪可以让人精神焕发，干劲十足，消极的情绪可以让人萎靡不振，日渐消沉，人的情绪极大影响了人们的日常生活和工作的效率。因此在这种情况下，对情绪的了解显得极为重要。

人的情绪变化时通常伴随着人体的一系列生理表现，或者行为表现，比如心跳加快，血压升高，脑电波活跃，激素分泌，又比如欢呼雀跃，手舞足蹈，唉声叹气，愁眉苦脸等等。正是因为情绪变化时，人体会产生这些变化，如果可以通过愈来愈发达的技术手段准确的获取人体的这些信息，再加上科学正确的分析，就可以判断出用户此刻的心情。情绪分析和识别是一项多领域交叉的科研问题，目前，多个领域都在对于情绪进行相关的研究，涉及的领域有医学，心理学，神经科学，人工智能等。

智能手机在发展过程中，为了丰富功能，增强用户体验，内置了越来越丰富的传感器，比如光线传感器，温度传感器，磁力传感器，重力传感器，陀螺仪，及速度传感器等等，这为研究人员进行实验提供给了非常有利的条件。

之所以选用安卓手机来收集这些信息，是因为，首先，越来越多的人拥有智能手机,其中绝大部分是安卓手机（下文“手机”，“智能手机”都指代“安卓手机”），安卓手机的大规模使用保障了数据的充足，在之前是不存在这样优越的研究条件的。其次，随着手机越来越智能，功能越来越多，浏览网页，聊天阅读，拍照娱乐，移动支付等等，使用手机的场景越来越多，人们越来越离不开手机，这使得很多人有了随身携带手机的习惯。并且，手机的携带对被测试者的正常生活影响较小，这使得持续获取数据成为了可能。

机器学习经过多年的发展，相关理论已经非常成熟。随着科技水平的飞速发展，计算机的处理能力大幅提升，并且伴随着数据量的暴增，人们进入“大数据”时代，在这样的时代背景下，机器学习具有传统统计难以匹及的优势，因此本文选用了机器学习的方式来进行情绪的识别。

1.2 研究意义

情绪与人们的生活密切相关，人生活中的每时每刻都对应着一种情绪，或开心，或难过，或平静。心理学研究表明，情绪很大程度上影响着人们的工作和生活，积极的情绪可以提高人体的机能，能够形成一种促进人体活动的动力，激励人去努力，提高工作的效率。消极情绪会让人觉得难受，会降低人的活力，比如活动起来动作迟缓，工作效率低下，消极的情绪还会对人的精力和体力造成影响，浑身无力，疲乏易累。因此，时刻了解自己的情绪状态和变化是很有必要的。

而且，随着近年来我国经济飞速发展，人们对于自己身体健康的关注也渐渐地从只关注生理健康到生理健康与心理健康并重，越来越多的人开始关注自己的情绪状态和情绪变化。但现代社会生活节奏较快，多数人常常无暇顾及自己身体信号、行为状态的微小变化，比如行走的速度、活动的幅度、室内室外活动时间等等，但相关研究表明人的这些行为举止的微小变化常常对应着人的情绪变化。传统的情绪识别一般分为基于生理信号和基于非生理信号两种。基于生理信号的情绪识别

一般需要依赖于精密的仪器，对人体的心率、皮肤阻抗等生理信号进行采集并进向相关分析，这种方式的情绪分析往往存在着一定的实施困难，比如仪器对人正常生活影响较大不便于采集真实的数据，又比如，这些仪器一般很容易受到干扰，收集来的信息往往掺杂了大量噪声，提高了分析工作的难度等。而基于非生理信号的情绪分析一般集中在语音语调方面，并且往往容易被人的刻意伪装而欺骗。为了帮助人们随时了解自己的情绪状态，使人能够及时意识到这些变化，并进行有效的自我调节，同时考虑到智能手机已经非常普及并且大多数人已经养成随身携带智能手机的习惯，本文作者通过安卓设备对用户的运动数据、环境数据、手机使用情况等信息进行采集，将数据进行特征提取后，利用已经较为成熟的机器学习分类算法构建模型，实现对用户情绪的识别。通过这种方式可以减小数据收集设备对用户的干扰，并且人的行为活动不容易伪装，更容易获取到真实的数据，而且通过收集多方面的信息更能全面地反映一个人目前的情绪状态。

情绪识别这项技术的应用前景十分广泛，比如听音乐时，如果能够识别出用户此刻的心情，进而推送给用户符合当前心情状态的歌曲，那么用户的使用体验就会得到极大的提升。再比如，在医疗护理过程中，如果可以准确分辨出患者此刻的心情，尤其是具有表达沟通障碍的患者，进而对其提供针对性的护理，无疑对于治疗过程是有极大帮助的。再比如，如果可以准确获知用户使用一款产品时的情绪状态，那么开发人员就可以针对用户使用时的情绪进行更有针对性的优化处理，从而持续提供更好的服务。

1.3 研究现状

目前有关人的情绪状态分析识别的研究主要有两类方向，一类是基于生理信号进行情绪识别，一类是基于非生理信号进行识别[2]。

1.3.1 非生理信号情绪分析

非生理信号的情绪分析研究主要包括基于面部表情的情绪分析，基于语音语调的情绪分析两种。

基于面部表情的情绪分析：不同的表情常常会反映不同的情绪，所以研究人员

可以通过分析人的面部表情以及伴随的面部肌肉动作来判断人的情绪[3]。比如嘴角上扬时，并且眼角出现皱纹，通常可以判断此时的情绪状态为高兴；眼睛瞪大，眉头紧皱，通常可以判断此时情绪状态为愤怒。基于面部表情的情绪识别既可以是基于局部特征进行的情绪分析，也可以是基于整体特征的情绪识别。前者是考虑到人在不同情绪下，人脸五官的形状，大小，以及相对位置会有差异，并以此作为情绪分析的数据特征，进行情绪分析。后者则是从整体出发，考虑不同情绪下整体面部特征的区别，提取特征的范围是整个人脸。目前基于面部表情的情绪识别已经取得一定进展，比如美国的情绪识别公司 Affectiva，Affectiva 通过将大量不同种族、年龄、性别的面部表情图片输入计算机，通过算法观察到所有脸部的纹理与皱纹，以及形状的变化等面部重要特征点变化，从而识别出人当前的情绪，该算法可以准确识别出高兴、难过、惊喜、生气、蔑视、厌恶、恐惧七种情绪。据称 Affectiva 的情感 AI 还将被集成到 Voxpopme 平台，使研究人员更容易获知观众的喜好，深入了解观众行为，减少视频工作的传统挑战。

基于语音语调的情绪分析：不同的语言表达方式通常也意味着不同的情绪状态，比如，人的心情愉悦时，语音会轻快，语调会上扬，心情失落时，语调会变得低沉，语调会放缓。基于语音的情绪识别研究一般多关注于基音的频率、声音携带的能量、语速快慢、表达是否流利等特征。目前，基于语音的情绪识别技术已经实现商用，如 2012 年成立的 Beyond Verbal 就是一家语音情绪识别领域领先的公司，该公司实现了通过算法识别音域变化，从而对用户的情绪进行判断，可以识别出 400 个复杂情绪，甚至能够识别其中的微小差别。

基于非生理信号的情绪分析研究简单易行，操作难度小，但是存在识别不准确和不可靠的问题，毕竟人是种复杂的动物，常常会为了隐藏自己的情绪而伪装自己的声音表情。另一方面，对于一些病患人群可能无法提供非生理信号情绪识别所需的特征，所以对于这一人群也不适用。

1.3.2 基于生理信号情绪分析

另一大类是基于生理信号的情绪分析。这类研究关注的重点通常是人体的心率，血压，脑电波等生理信号[4]。人处在不同的情绪状态时，表现出来的生理信号

也是不同的。研究人员通过收集人在喜、怒、哀、乐等不同情绪下的各项生理指标，比如通过测量人体心脏跳动的频率、呼吸的频率、脑内电波的活跃程度、皮肤外表的阻抗等数据，然后与对应的情绪状态高兴还是悲伤建立对应联系，然后分析数据间的内在关系，从而建立数据模型实现对人情绪的判断。生理信息相比非生理信号而言，不易被伪装，并且通过这种方式，在获得准确信息的前提下，分析结果准确率高。但是该方法也存在着收集困难，信号准确率无法保障，易被干扰的缺点。

1.4 本文研究内容

下文主要研究基于机器学习的安卓移动用户情绪分析系统的设计与实现，即将从安卓设备采集来的数据，通过经典的机器学习算法进行分析，从而判断出使用者现在的心情状态。

下面对后边章节将要讨论的内容进行简要介绍。

第二章主要介绍了用到的技术和工具软件，对情绪分类模型的构建原理进行了简要介绍，主要讲了五种机器学习的分类算法，包括 K 近邻分类算法、SVM 分类算法、朴素贝叶斯分类算法、决策树分类算法、Adaboost 分类算法。

第三章详细介绍了实验过程中用户数据收集以及对收集来的原始数据的预处理操作，以及在构建情绪分类模型过程前进行的特征工程，包括了特征提取、相关性分析、特征选择、主成分分析等。

第四章介绍了情绪分析模型的构建，情绪分析模型主要包括 K 近邻分类模型、SVC 分类模型、朴素贝叶斯分类模型三种。其中，K 近邻分类模型中通过交叉验证选择出了准确率最高的 K 值，SVC 分类模型中通过设置不同的核函数构建了三个不同的分类模型，朴素贝叶斯分类模型中按照对数据分布的假设不同设置了三个模型。

第五章对实验结果进行了准确性测试，输出了每个模型的平均准确率，并对测试集样本的最后十个数据的预测结果进行输出，然后与真实标签进行比对。本章同时对不同分类模型进行了性能测试，比较了处理相同数据集时不同的模型的耗时情况。

第六章对本文工作进行了总结，分析了本文工作的缺点与不足，并对后续工作

的一些待完善与继续深入研究的方向进行了展望。

第二章 开发平台及原理分析

2.1 开发工具简介

本文中实现情绪分析系统使用的开发工具是 Anaconda 和 PyCharm，接下来将一一介绍。

Anaconda 是一个用于科学计算的 Python 发行版，它包括 250 多种流行的数据科学软件包，比如大名鼎鼎的 numpy 和 pandas，以及适用于 Windows，Linux 和 MacOS 的 conda 软件包和虚拟环境管理器。Conda 使安装，运行和升级复杂的数据科学和机器学习环境（如 Scikit-learn，TensorFlow 和 SciPy）变得简单快捷。

PyCharm 是一款开发者常用的 Python 集成开发工具，由 Jetbrains 公司开发完成。PyCharm 除了提供了调试，语法高亮，代码跳转等一些基本功能，还提供了智能代码完成，代码检查，即时错误突出显示和快速修复，以及自动代码重构和丰富的导航功能，而且 PyCharm 内置了多种集成的调试器和测试运行器。PyCharm 还与 IPython Notebook 集成，具有交互式 Python 控制台，并支持 Anaconda 以及多个科学软件包，包括 Matplotlib 和 NumPy 等。实际项目的开发过程中，通过使用 PyCharm IDE 可以极大地提升开发者的开发效率。

2.2 Numpy 和 Pandas

NumPy 是用 Python 进行科学计算的基础软件包，它代表“Numeric Python”，是一个由多维数组对象和用于处理数组的例程集合组成的库。Numpy 前身是 Numeric，最初是由 Jim Hugunin 开发的，他同时开发了另一个包 Numarray，这个包提供了一些额外的功能。2005 年 Travis Oliphant 通过将 Numarray 的功能集成到 Numeric 包中来创建了 NumPy 包。

NumPy 除了提供了矩阵数据类型、矢量处理等，还包含了精密的运算库等许多数值编程工具，因此十分擅长严格的数据处理任务[5]。Numpy 内部解除了 Python 的 PIL(全局解释器锁),运算效率极好,是大量机器学习框架的基础库。除了明显的科学用途外，NumPy 还可以用作通用数据的高效多维容器，可以定义任意数据类

型，在 Python 科学计算中应用十分广泛

Pandas （Python Data Analysis Library）是 Python 的一个数据分析包，最初由 AQR Capital Management 于 2008 年 4 月开发，并于 2009 年底开源出来，目前由专注于 Python 数据包开发的 PyData 开发 team 继续开发和维护，属于 PyData 项目的一部分。Pandas 是一种基于 NumPy 的工具，该工具诞生之初就是为了解决数据分析任务。Pandas 中提供了高效地操作大型数据集所需的工具，其中包含了大量库和一些标准的数据模型[6]。Pandas 提供了大量能使开发者快速便捷地进行数据处理的函数和方法。Pandas 里面还定义了 Series 和 DataFrame 两种数据类型，这使得数据操作变得更方便，并且 Pandas 可以轻松处理数据中的缺失数据，插入和删除数据，数据对齐，标签切片等等。此外，由于 Pandas 提供快速、灵活和富有表现力的数据结构，这使得它还可以处理很多不同类型的数据，比如具有异构类型列的表格数据，有序和无序的时间序列数据，以及任何其他形式的观测/统计数据集。

2.3 scikit-learn

scikit-learn 简称 sklearn，于 2007 年问世，是 Python 最重要的机器学习库之一，常被用于机器学习和数据挖掘等应用中[7]。

sklearn 依赖于 matplotlib、NumPy 和 SciPy，内置了丰富的机器学习算法，有效提高了机器学习的效率。此外，sklearn 内置的大量的标准数据集也为开发者节省了不少获取数据和处理数据的时间。而且 sklearn 文档完善，API 丰富，上手难度小，颇受开发人员的喜爱。

sklearn 库主要包含了分类（Classification），回归（Regression），聚类（Clustering），降维（Dimensionality reduction），模型选择（Model selection），预处理（Preprocessing）六大功能。分类，即识别对象属于哪个类别，包含的算法有 K 最近邻，支持向量机分类，决策树，朴素贝叶斯，随机森林等，常用于垃圾邮件检测，图像识别等方向。回归，最主要是预测与对象相关联的连续属性，多应用于药物反应，股价预测等方面，主要包括线性回归，多项式回归，支持向量回归等算法。聚类就是将对象根据数据间的特征划归为不同分类，常用方法有 K 均值、mean-shift 等。跟分类不

同的是，分类是用带标签的数据训练出模型，然后判断新数据哪种类别，聚类的数据是不带有标签的，是完全根据算法分析数据中之间的相似性来对数据进行自动归类。降维，就是降低样本特征的维度，常用来提高计算效率，或者进行可视化。模型选择即比较、验证、选择参数和模型，目标是通过参数调整提高模型精度。预处理则是对数据进行一些操作，如提取数据特征、归一化、标准化、白化、去均值化、二值化等操作来满足计算需求。

2.4 机器学习分类算法

2.4.1 K 最近邻算法

K 最近邻算法[8] (K-Nearest Neighbors，KNN)工作原理：一组每个数据都带有标签的数据集，被称为样本集。样本集中的数据和其对应分类是已知的。输入样本集后，通过对样本集数据特征进行分析，训练出模型。等再输入不带标签的新数据时，提取出新数据的数据特征，与训练好的数据模型进行比对，从训练集中提取出 K 个与新数据最相似的样本的标签，选出这 K 个数据里面比例最高的标签作为新数据的标签，从而实现分类的目的。通常情况下，K 的取值不大于 20。

K 最近邻的主要过程：

1、计算测试对象到训练集中每个对象的距离。

2、按照距离远近排序，这个距离可以是欧式距离，马氏距离，曼哈顿距离。

3、选取与当前测试对象最近的 K 个训练对象，作为该测试对象的邻居。

4、统计这 K 个邻居的类别的出现频率。

5、K 个邻居里出现频率最高的类别，即为测试对象的类别。

K 值选取较小时，也就意味着使用待分类点周围较少的邻居点进行预测，比如极端值 K=1,那么也就是由待分类点最近的一个点来对该点进行分类，如果一旦该邻居点是噪声点，那么就会对预测结果造成较大误差。K 值选取较大时，可以有效降低噪声的影响，但是容易导致分类界限不明显。K 的具体取值一般与数据情况有关，不合理的 K 值会在一定程度上产生过拟合或者欠拟合的问题。具体的 K 值选择可以通过经验判断或者交叉验证来确定。

K 近邻算法优点在于算法成熟，容易理解，准确率高，对异常值不敏感，既可以处理离散型数据，又可以处理连续型数据。缺点在于，样本分布不均匀时容易误分，数据量比较大时计算复杂度和空间复杂度都比较高，因此一般只应用于样本量比较小且分布比较均匀的情况。

2.4.2 支持向量机算法

支持向量机（Support Vector Machine，SVM），本身是一种二分类模型，处理多分类问题时可以转换为一对多（选择某一类，样本中除这一类的其他所有类为一类，两者之间构建支持向量机）和一对一（任意两类之间构建支持向量机）来处理。支持向量机算法可以处理的数据可以分为三类：线性可分，近似线性可分，线性不可分。支持向量机的分类原理是寻找一个最优的超平面将数据进行分类，使边界最大[9]。根据数据可分的三种情况，支持向量机有以下几种分类方式：

1、如果数据线性可分，那么选择硬间隔方法使边界最大化，通过学习线性分类器来完成这一分类过程。硬间隔可以将所有样本正确分类，也正因为如此，受噪声样本影响很大。

2、如果数据近似线性可分，那么选择软间隔方法使边界最大化，通过学习线性分类器完成这一分类过程。软间隔对应于近似线性可分或线性不可分的数据集，允许一些超平面附近的样本被错误分类，从而提升了泛化性能。因为不是所有情况都需要把点全部分对，有时候样本点中存在一些本来就是错误的数据，也就是噪声，学习过程中如果学习了这些噪声，就会出现过拟合的情况，降低模型预测的准确性。这个过程需要加入惩罚因子 C，使得点被错分的情况更合理。

3、如果数据线性不可分，那么就通过核函数以及软间隔最大化的方式，学习非线性分类器实现这一分类过程。当数据线性不可分的时候，SVM 通过将低维数据向高维空间转化实现线性可分。

对于线性不可分的数据集的任意两个实例：a_i，a_j。选取特定映射 f 之后，使得$\mathrm{f}(a_i)$与$\mathrm{f}(a_j)$在高维空间中线性可分，运用上述（近似）线性可分问题的求解方法，可以发现目标函数和分类决策函数只涉及内积$<\mathrm{f}(a_i),\mathrm{f}(a_j)>$。由于高维空间中的内积计算非常复杂，通过引入核函数$\mathrm{K}(a_i,a_j)=<\mathrm{f}(a_i,a_j)>$，内积问题变成了求函

数值问题，由高维运算变成了低维运算，有效降低了计算复杂度。常用的核函数有：

多项式核函数：多项式核函数应用于将低维数据向高维空间转化，缺点是当多项式阶数比较高时，核函数计算复杂度会非常高，甚至无法计算。

$$\kappa(\mathrm{x}, x_i) = ((x \cdot x_i) + 1)^d \quad \text{式（2-1）}$$

线性核函数：主要用于线性可分的情况，其参数较少，运算复杂度低，通常选择线性核函数作为首先尝试的核函数来观察效果。

$$\kappa(\mathrm{x}, x_i) = \mathrm{x} \cdot x_i \quad \text{式（2-2）}$$

高斯径向基核函数：也是将低维数据转换到高维空间，但本身参数相比多项式核函数要少，所以在大样本和小样本时都有很好的性能。使用最广泛的一种核函数，在不确定用哪种核函数时，推荐使用高斯径向基核函数。

$$\kappa(\mathrm{x}, x_i) = \exp(-\frac{||x-x_i||^2}{\delta^2}) \quad \text{式（2-3）}$$

sigmoid 核函数：

$$\kappa(\mathrm{x}, x_i) = \tanh(\eta < \mathrm{x}, x_i > +\theta) \quad \text{式（2-4）}$$

支持向量机算法的优点在于可以以较低的计算量解决高维问题，泛化错误率低，计算开销不大，结果容易解释，并且错误率较低。缺点在于，对参数调节和函数的选择敏感，对于核函数的高维映射解释力不强，尤其是径向基函数。

2.4.3 朴素贝叶斯算法

朴素贝叶斯算法一种简单而且效果比较不错的弱分类器，其理论基础是概率论中的贝叶斯理论。朴素贝叶斯算法虽然构建简单，分类效果却很优秀，甚至比许多复杂算法还要高效，尤其是在大型数据集，表现更佳。之所以说其朴素，是因为朴素贝叶斯算法是基于各个样本特征相互独立的假设的[10]。举个例子，比如一个男生具有长得高，皮肤白，性格好的特点，可以得出结论该男生受女生喜欢，虽然可能这些特征之间具有一定的关联，或者相互依赖，但在朴素贝叶斯算法看来，这些特征在判断男生是不是受女孩喜欢的问题上，特征之间是相互独立的，并且对事件的影响是相同的，即权重相同。

先验概率：在事件发生前，基于历史事件统计，或者背景常识，或者人的经验判断得出的事件可能发生的概率。比如，天空中阴云密布，历史上天空中出现阴云

密布的情况会下雨的可能性是 70%，那么 70%就是先验概率。

后验概率：在事件发生后，依据事件发生的结果反推该事件是由某因素引起的概率，即执果寻因。举个例子，中午吃了苹果，下午肚子疼，想算一下肚子疼是由吃苹果导致的概率，这就是后验概率。

贝叶斯理论：

事件 X 在事件 Y 发生的条件下的概率，与事件 Y 在事件 X 发生的条件下的概率是不一样的，贝叶斯公式就是用来描述这种关系的。

$$\mathrm{P(X|Y)} = \frac{P(Y|X)P(X)}{P(Y)} \qquad \text{式（2-5）}$$

其中 P(X|Y)是在 Y 发生的情况下 X 发生的概率。X，Y 都是事件，并且 P(Y)不为 0。P(X)是事件 X 发生的先验概率，P(X|Y)是 X 的后验概率。

当式（2-5）表示为式（2-6），$\mathrm{P(y|X)} = \frac{P(X|y)P(y)}{P(X)}$，y 表示类变量，X 是特征向量：

$$\mathrm{X} = (x_1, x_2, x_3, \dots, x_n) \qquad \text{式（2-6）}$$

由朴素贝叶斯的朴素假设，也就是每个特征变量之间相互独立，即有：

$$\mathrm{P(AB)} = \mathrm{P(A)P(B)} \qquad \text{式（2-7）}$$

所以，式（2-7）又可以表示为：

$$\mathrm{P(y|}x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)} \qquad \text{式（2-8）}$$

由于输入数据与分母是常量相关，于是进一步推导：

$$\mathrm{P(y|}x_1, \dots, x_n) \propto P(y)\Pi_{i=1}^{n}P(x_i|y) \qquad \text{式（2-9）}$$

所以，目的是选择出类变量 y 的所有可能值的中使概率最大的那个可能值，可以用公式表示为：

$$\hat{y} = arg\max_{y} P(y)\Pi_{i=1}^{n}P(x_i|y) \qquad \text{式（2-10）}$$

最后，通过$\mathrm{P(y)}$与$\mathrm{P}(x_i|\mathrm{y})$的计算得出结果。

朴素贝叶斯分类器具有算法逻辑简单，实现难度小的优点，而且因为朴素贝叶斯分类器建立在特征相互独立的假设下，只涉及二维存储，所以分类过程中时间复杂度低。朴素贝叶斯分类器在样本特征比较少，并且特征之间相关性小时，具有较为良好的分类效果，而且理论上，相比其他分类方法，朴素贝叶斯算法误差率最低。

但实际应用中，情况往往比较复杂，样本特征比较多的时候往往难以保证特征间相互独立的假设，分类效果也会受到影响。

2.4.4 决策树算法

决策树应用十分广泛，既可以用于回归也可以用于分类。分类问题中，决策树采用树形结构对样本的属性进行分类，既可以处理离散（if-then）的特征空间，也可以处理连续的特征空间（只需将连续空间通过阈值化变为 if-then 形式即可）。决策树由边和结点构成，内结点代表属性和特征，外结点代表类别[11]。边代表判别的规则，即 if-then 规则。决策树主要分为特征选择、生成、剪枝三步。根据特征选择度量方式不同，分为 ID3、C4.5、CART 三种决策树算法，它们对应的度量方式分别是信息增益（Information gain）、增益比率（gain ratio）、基尼指数（Gini index）。

信息增益：

首先是熵的定义，

$$\mathrm{E(D)} = -\sum_{k=1}^{K} p_k \log_2 p_k \qquad \text{式（2-11）}$$

变量的不确定性越大，熵也越大。

信息增益，即信息获取量（例如，通过 a 作为结点来分类获取了多少信息）：

$$\mathrm{Gain(D,a)} = \mathrm{E(D)} - \sum_{v=1}^{v} \frac{|D_v|}{|D|} E(D_v) \qquad \text{式（2-12）}$$

依次比较各个属性的信息增益，选择最大的那个属性作为根结点，然后对于后面的结点，依次重复这个过程，直至最后给定结点的所有样本属于同一类或者没有剩余属性可以进一步划分样本，迭代停止。

增益比率：

$$\mathrm{Gain,aio(D,a)} = \frac{Gain(D,a)}{IV(a)} \qquad \text{式（2-13）}$$

$$\mathrm{IV(a)} = -\sum_{x=1}^{x} \frac{|D_x|}{|D|} \log_2 \frac{|D_x|}{|D|} \qquad \text{式（2-14）}$$

其中$\frac{|D_x|}{|D|}$代表的是 a 属性的 x 种情况中某一种在样本中的比例，即p(x)。增益比率，也就是某属性增加的信息熵与某属性自有信息上的比率。选择增益比率高的那个作为根结点。

基尼指数：

基尼指数（基尼不纯度）= 样本被选中的概率 * 样本被分错的概率

$$\mathrm{Gini(D)} = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} {p_k}^2 \qquad \text{式（2-15）}$$

样本集分为 K 类，p_k表示选中的样本在 k 类别中的比例，此处看作样本是 k 类别的概率，则这个样本被分错的概率是$(1-p_k)$。

属性 a 的基尼系数，

$$\mathrm{Gini(D,a)} = \sum_{v=1}^{v} \frac{|D_v|}{|D|} Gini(D_v) \qquad \text{式（2-16）}$$

选择基尼系数低的属性作为根结点。

决策树的优势在于它的数据形式非常容易理解，而且能够给出数据间的内在关系。除此之外，决策树计算复杂度不高，对中间值的缺失不敏感，可以处理不相关的数据特征。而且，决策树对数据形式要求简单，不必像其他分类方法一样统一数据属性，既可以是数值型，也可以是标称型。但缺点是容易拟合过度，处理连续变量效果不好，类别较多时，错误会增加的比较快等。

2.4.5 Adaboost 算法

Adaboost 算法，即自适应增强算法，是一种迭代算法[12]。通过对同一数据集迭代训练不同的分类器，每次找到一个最优的分类器，然后下一次迭代时增大前一个分类器错误分类样本的权值，减小正确分类样本的权值。最后将得到的多个最优的分类器组合起来就得到一个强分类器。

Adaboost 算法实现步骤：

1、初始化各样本数据的权值。假设有 N 个训练数据，第一次开始迭代时，各个样本被赋予相同的权值

2、对数据集迭代训练弱分类器。如果训练过程中，某个训练数据被准确分类，那么在下一次迭代过程中，降低该训练数据的权值，同时提高被错误分类的训练数据的权值。一次迭代过程完成后，使用权重值更新后的训练数据集进行下一次迭代，构造新的弱分类器。如此迭代下去，完成整个训练过程。

3、集成各个弱分类器构建一个新的强分类器。为了让分类准确率高的弱分类器发挥更大的作用，按照分类过程中各个弱分类器的误差大小情况，为各个弱分类器分配权重。误差率越小的分类器，在构建强分类器的过程中，所占权重越高，否

则，所占权重越小。这样，一个强分类器就构建完成了

Adaboost 是一种简单有效的分类算法，很好地利用了不同弱分类器进行级联，并且在构建过程中充分考虑了不同分类器的权重问题，分类结果精度高。主要缺点有，分类精度可能会受数据不平的影响而下降，时间复杂度高，弱分类器的数目也就是迭代次数不易确定。

第三章 数据处理和特征工程

本实验主要研究内容是，以 Android 设备为信息收集载体，通过获取 Android 用户的运动信息、手机使用情况、用户情绪状态，并以此进行分析，探索发现用户的各项信息与情绪状态的内在联系，并建立模型，实现通过各项数据识别用户情绪。实验步骤主要包括：数据的获取、数据预处理、特征提取、PCA 降维、训练模型与预测、参数调优、结果对比。

3.1 用户数据收集

本实验由 10 名使用 Android 设备的同学参与志愿活动，进行数据的收集。每日的数据收集分为早中晚三部分，在早上 8：00，中午 12：00，下午 6：00，分别进行一次用户情绪录入，并与此同时进行离散采样，每隔一分钟进行一次采样，每次采样时间持续十分钟。志愿者的心情部分主要包含高兴、平静、难过、愤怒四种基本情绪类型，由志愿者凭感受主动录入。主要采样信息为 Android 设备各项传感器数据和设备基本情况信息。其中需要采样的传感器部分包括加速度传感器、方向传感器、陀螺仪传感器、磁场传感器、重力传感器、线性加速度传感器、GPS 传感器、光线传感器、的数据。设备情况信息的采集部分包括手机的网速情况和情景模式。收集到的数据以 txt 格式存储，保存在用户手机内存卡根目录。整个采集过程持续两星期。

收集的各项信息数据格式如下：

（加速度 x，加速度 y，加速度 z，方向 x，方向 y，方向 z，陀螺仪 x，陀螺仪 y，陀螺仪 z，磁场 x，磁场 y，磁场 z，重力 x，重力 y，重力 z，线性加速度 x，线性加速度 y，线性加速度 z，GPS 经度，GPS 纬度，光照，网速，情景模式，情绪状态）

将收集到的原始数据转换成 excel 表格，情况如图 3.2，图 3.3，图 3.4 所示。

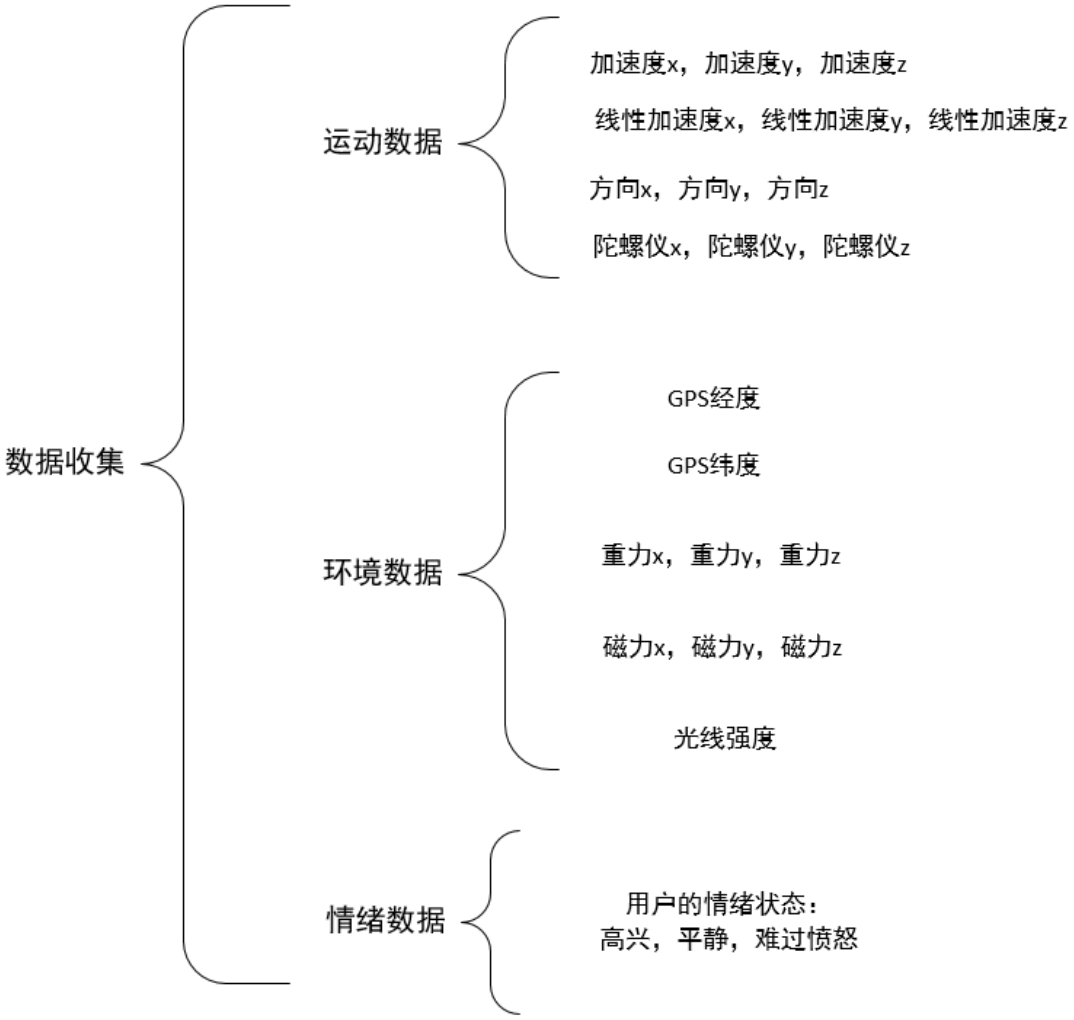
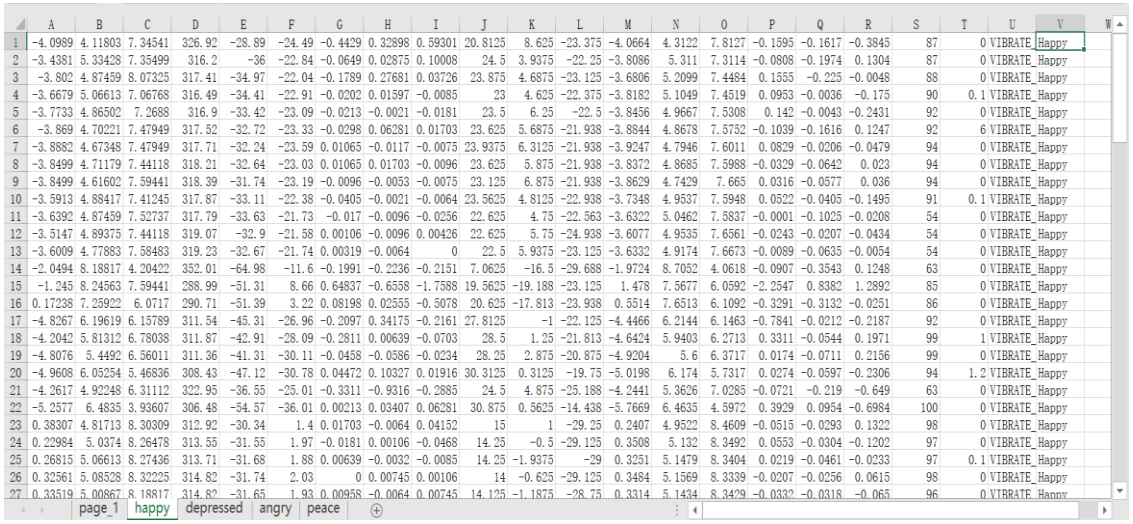


图 3.1 数据收集

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	21	22	23
0	-4.0989	4.11803	7.34541	326.92	-28.89	-24.49	-0.4429	0.32898	0.59301	20.8125	8.625	-23.375	-4.0664	4.3122	7.8127	-0.1595	-0.1617	-0.3845	87	0	VIBRATE_Happ
1	-3.4381	5.33428	7.35499	316.2	-36	-22.84	-0.0649	0.02875	0.10008	24.5	3.9375	-22.25	-3.8086	5.311	7.3114	-0.0808	-0.1974	0.1304	87	0	VIBRATE_Happ
2	-3.802	4.87459	8.07325	317.41	-34.97	-22.04	-0.1789	0.27681	0.03726	23.875	4.6875	-23.125	-3.6806	5.2099	7.4484	0.1555	-0.225	-0.0048	88	0	VIBRATE_Happ
3	-3.6679	5.06613	7.06768	316.49	-34.41	-22.91	-0.0202	0.01597	-0.0085	23	4.625	-22.375	-3.8182	5.1049	7.4519	0.0953	-0.0036	-0.175	90	0.1	VIBRATE_Happ
4	-3.7733	4.86502	7.2688	316.9	-33.42	-23.09	-0.0213	-0.0021	-0.0181	23.5	6.25	-22.5	-3.8456	4.9667	7.5308	0.142	-0.0043	-0.2431	92	0	VIBRATE_Happ
5	-3.869	4.70221	7.47949	317.52	-32.72	-23.33	-0.0298	0.06281	0.01703	23.625	5.6875	-21.938	-3.8844	4.8678	7.5752	-0.1039	-0.1616	0.1247	92	6	VIBRATE_Happ
6	-3.8882	4.67348	7.47949	317.71	-32.24	-23.59	0.01065	-0.0117	-0.0075	23.9375	6.3125	-21.938	-3.9247	4.7946	7.6011	0.0829	-0.0206	-0.0479	94	0	VIBRATE_Happ
7	-3.8499	4.71179	7.44118	318.21	-32.64	-23.03	0.01065	0.01703	-0.0096	23.625	5.875	-21.938	-3.8372	4.8685	7.5988	-0.0329	-0.0642	0.023	94	0	VIBRATE_Happ
8	-3.8499	4.61602	7.59441	318.39	-31.74	-23.19	-0.0096	-0.0053	-0.0075	23.125	6.875	-21.938	-3.8629	4.7429	7.665	0.0316	-0.0577	0.036	94	0	VIBRATE_Happ
9	-3.5913	4.88417	7.41245	317.87	-33.11	-22.38	-0.0405	-0.0021	-0.0064	23.5625	4.8125	-22.938	-3.7348	4.9537	7.5948	0.0522	-0.0405	-0.1495	91	0.1	VIBRATE_Happ
10	-3.6392	4.87459	7.52737	317.79	-33.63	-21.73	-0.017	-0.0096	-0.0256	22.625	4.75	-22.563	-3.6322	5.0462	7.5837	-0.0001	-0.1025	-0.0208	54	0	VIBRATE_Happ
11	-3.5147	4.89375	7.44118	319.07	-32.9	-21.58	0.00106	-0.0096	0.00426	22.625	5.75	-24.938	-3.6077	4.9535	7.6561	-0.0243	-0.0207	-0.0434	54	0	VIBRATE_Happ
12	-3.6009	4.77883	7.58483	319.23	-32.67	-21.74	0.00319	-0.0064	0	22.5	5.9375	-23.125	-3.6332	4.9174	7.6673	-0.0089	-0.0635	-0.0054	54	0	VIBRATE_Happ
13	-2.0494	8.18817	4.20422	352.01	-64.98	-11.6	-0.1991	-0.2236	-0.2151	7.0625	-16.5	-29.688	-1.9724	8.7052	4.0618	-0.0907	-0.3543	0.1248	63	0	VIBRATE_Happ
14	2.78685	6.95276	4.97994	284.14	-59.29	4.46	-0.4887	1.40108	1.21583	18.6875	-24.625	-19.25	0.7637	8.4061	4.9923	1.5596	-1.3337	0.1174	35	1.1	VIBRATE_Happ
16	-1.245	8.24563	7.59441	288.99	-51.31	8.66	0.64837	-0.6558	-1.7588	19.5625	-19.188	-23.125	1.478	7.5677	6.0592	-2.2547	-2.2547	1.2892	85	0	VIBRATE_Happ
17	-0.2298	7.05811	3.85945	291.22	-72.13	-0.51	-1.6534	-1.1307	-0.5185	22.1875	-23.375	-17	-0.0887	9.3332	3.0089	0.8428	-1.3671	-0.394	91	0	VIBRATE_Happ
18	0.17238	7.25922	6.0717	290.71	-51.39	3.22	0.08198	0.02555	-0.5078	20.625	-17.813	-23.938	0.5514	7.6513	6.1092	-0.3291	-0.3132	-0.0251	86	0	VIBRATE_Happ
19	-4.8267	6.19619	6.15789	311.54	-45.31	-26.96	-0.2097	0.34175	-0.2161	27.8125	-1	-22.125	-4.4466	6.2144	6.1463	-0.7841	-0.0212	-0.2187	92	0	VIBRATE_Happ
20	-4.2042	5.81312	6.78038	311.87	-42.91	-28.09	-0.2811	0.00639	-0.0703	28.5	1.25	-21.813	-4.6424	5.9403	6.2713	0.3311	-0.0544	0.1971	99	1	VIBRATE_Happ
21	-4.8076	5.4492	6.56011	311.36	-41.31	-30.11	-0.0458	-0.0586	-0.0234	28.25	2.875	-20.875	-4.9204	5.6	6.3717	0.0174	-0.0711	0.2156	99	0	VIBRATE_Happ
22	-4.9608	6.05254	5.46836	308.43	-47.12	-30.78	0.04472	0.10327	0.01916	30.3125	0.3125	-19.75	-5.0198	6.174	5.7317	0.0274	-0.0597	-0.2306	94	1.2	VIBRATE_Happ
23	-4.2617	4.92248	6.31112	322.95	-36.55	-25.01	-0.3311	-0.9316	-0.2885	24.5	4.875	-25.188	-4.2441	5.3626	7.0285	-0.0721	-0.219	-0.649	63	0	VIBRATE_Happ
24	-5.2577	6.4835	3.93607	306.48	-54.57	-36.01	0.00213	0.03407	0.06281	30.875	0.5625	-14.438	-5.7669	6.4635	4.5972	0.3929	0.0954	-0.6984	100	0	VIBRATE_Happ
25	0.38307	4.81713	8.30309	312.92	-30.34	1.4	0.01703	-0.0064	0.04152	15	1	-29.25	0.2407	4.9522	8.4609	-0.0515	-0.0293	0.1322	98	0	VIBRATE_Happ
26	0.22984	5.0374	8.26478	313.55	-31.55	1.97	-0.0181	0.00106	-0.0468	14.25	-0.5	-29.125	0.3508	5.132	8.3492	0.0553	-0.0304	-0.1202	97	0	VIBRATE_Happ

page_1 happy depressed angry peace

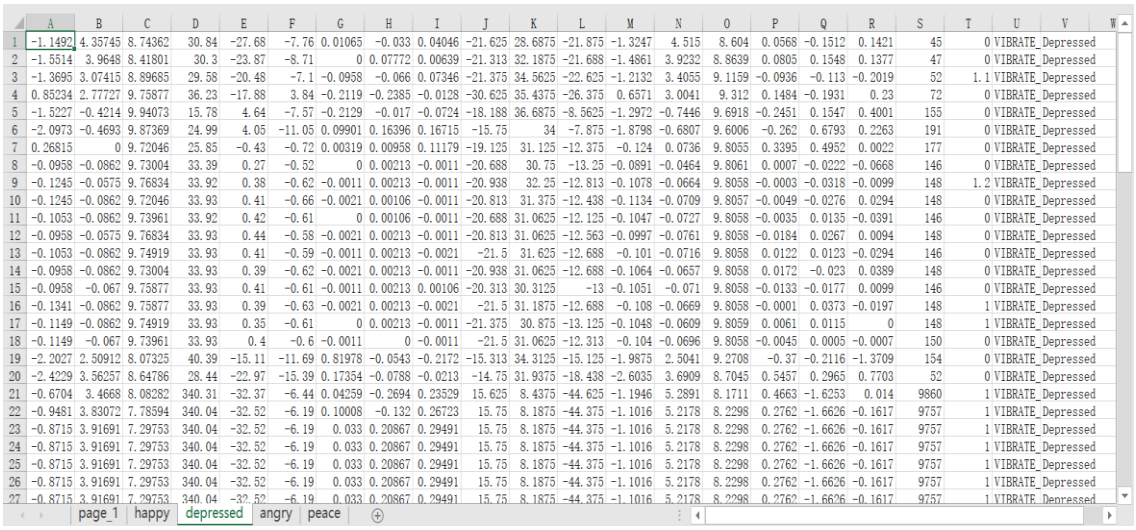
图 3.2 原始数据



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	-4.0989	4.11803	7.34541	326.92	-28.89	-24.49	-0.4429	0.32898	0.59301	20.8125	8.625	-23.375	-4.0664	4.3122	7.8127	-0.1595	-0.1617	-0.3845	87	0	VIBRATE	Happy
2	-3.4381	5.33428	7.35499	316.2	-36	-22.84	-0.0649	0.02875	0.10008	24.5	3.9375	-22.25	-3.8086	5.311	7.3114	-0.0808	-0.1974	0.1304	87	0	VIBRATE	Happy
3	-3.802	4.87459	8.07325	317.41	-34.97	-22.04	-0.1789	0.27681	0.03726	23.875	4.6875	-23.125	-3.6806	5.2099	7.4484	0.1555	-0.225	-0.0048	88	0	VIBRATE	Happy
4	-3.6679	5.06613	7.06768	316.49	-34.41	-22.91	-0.0202	0.01597	-0.0085	23	4.625	-22.375	-3.8182	5.1049	7.4519	0.0953	-0.0036	-0.175	90	0.1	VIBRATE	Happy
5	-3.7733	4.86502	7.2688	316.9	-33.42	-23.09	-0.0213	-0.0021	-0.0181	23.5	6.25	-22.5	-3.8456	4.9667	7.5308	0.142	-0.0043	-0.2431	92	0	VIBRATE	Happy
6	-3.869	4.70221	7.47949	317.52	-32.72	-23.33	-0.0298	0.06281	0.01703	23.625	5.6875	-21.938	-3.8844	4.8678	7.5752	-0.1039	-0.1616	0.1247	92	6	VIBRATE	Happy
7	-3.8882	4.67348	7.47949	317.71	-32.24	-23.59	0.01065	-0.0117	-0.0075	23.9375	6.3125	-21.938	-3.9247	4.7946	7.6011	0.0829	-0.0206	-0.0479	94	0	VIBRATE	Happy
8	-3.8499	4.71179	7.44118	318.21	-32.64	-23.03	0.01065	0.01703	-0.0096	23.625	5.875	-21.938	-3.8372	4.8685	7.5988	-0.0329	-0.0642	0.023	94	0	VIBRATE	Happy
9	-3.8499	4.61602	7.59441	318.39	-31.74	-23.19	-0.0096	-0.0053	-0.0075	23.125	6.875	-21.938	-3.8629	4.7429	7.665	0.0316	-0.0577	0.036	94	0	VIBRATE	Happy
10	-3.5913	4.88417	7.41245	317.87	-33.11	-22.38	-0.0405	-0.0021	-0.0064	23.5625	4.8125	-22.938	-3.7348	4.9537	7.5948	0.0522	-0.0405	-0.1495	91	0.1	VIBRATE	Happy
11	-3.6392	4.87459	7.52737	317.79	-33.63	-21.73	-0.017	-0.0096	-0.0256	22.625	4.75	-22.563	-3.6322	5.0462	7.5837	-0.0001	-0.1025	-0.0208	54	0	VIBRATE	Happy
12	-3.5147	4.89375	7.44118	319.07	-32.9	-21.58	0.00106	-0.0096	0.00426	22.625	5.75	-24.938	-3.6077	4.9535	7.6561	-0.0243	-0.0207	-0.0434	54	0	VIBRATE	Happy
13	-3.6009	4.77883	7.58483	319.23	-32.67	-21.74	0.00319	-0.0064	0	22.5	5.9375	-23.125	-3.6332	4.9174	7.6673	-0.0089	-0.0635	-0.0054	54	0	VIBRATE	Happy
14	-2.0494	8.18817	4.20422	352.01	-64.98	-11.6	-0.1991	-0.2236	-0.2151	7.0625	-16.5	-29.688	-1.9724	8.7052	4.0618	-0.0907	-0.3543	0.1248	63	0	VIBRATE	Happy
15	-1.245	8.24563	7.59441	288.99	-51.31	8.66	0.64837	-0.6558	-1.7588	19.5625	-19.188	-23.125	1.478	7.5677	6.0592	-2.2547	0.8382	1.2892	85	0	VIBRATE	Happy
16	0.17238	7.25922	6.0717	290.71	-51.39	3.22	0.08198	0.02555	-0.5078	20.625	-17.813	-23.938	0.5514	7.6513	6.1092	-0.3291	-0.3132	-0.0251	86	0	VIBRATE	Happy
17	-4.8267	6.19619	6.15789	311.54	-45.31	-26.96	-0.2097	0.34175	-0.2161	27.8125	-1	-22.125	-4.4466	6.2144	6.1463	-0.7841	-0.0212	-0.2187	92	0	VIBRATE	Happy
18	-4.2042	5.81312	6.78038	311.87	-42.91	-28.09	-0.2811	0.00639	-0.0703	28.5	1.25	-21.813	-4.6424	5.9403	6.2713	0.3311	-0.0544	0.1971	99	1	VIBRATE	Happy
19	-4.8076	5.4492	6.56011	311.36	-41.31	-30.11	-0.0458	-0.0586	-0.0234	28.25	2.875	-20.875	-4.9204	5.6	6.3717	0.0174	-0.0711	0.2156	99	0	VIBRATE	Happy
20	-4.9608	6.05254	5.46836	308.43	-47.12	-30.78	0.04472	0.10327	0.01916	30.3125	0.3125	-19.75	-5.0198	6.174	5.7317	0.0274	-0.0597	-0.2306	94	1.2	VIBRATE	Happy
21	-4.2617	4.92248	6.31112	322.95	-36.55	-25.01	-0.3311	-0.9316	-0.2885	24.5	4.875	-25.188	-4.2441	5.3626	7.0285	-0.0721	-0.219	-0.649	63	0	VIBRATE	Happy
22	-5.2577	6.4835	3.93607	306.48	-54.57	-36.01	0.00213	0.03407	0.06281	30.875	0.5625	-14.438	-5.7669	6.4635	4.5972	0.3929	0.0954	-0.6984	100	0	VIBRATE	Happy
23	0.38307	4.81713	8.30309	312.92	-30.34	1.4	0.01703	-0.0064	0.04152	15	1	-29.25	0.2407	4.9522	8.4609	-0.0515	-0.0293	0.1322	98	0	VIBRATE	Happy
24	0.22984	5.0374	8.26478	313.55	-31.55	1.97	-0.0181	0.00106	-0.0468	14.25	-0.5	-29.125	0.3508	5.132	8.3492	0.0553	-0.0304	-0.1202	97	0	VIBRATE	Happy
25	0.26815	5.06613	8.27436	313.71	-31.68	1.88	0.00639	-0.0032	-0.0085	14.25	-1.9375	-29	0.3251	5.1479	8.3404	0.0219	-0.0461	-0.0233	97	0.1	VIBRATE	Happy
26	0.32561	5.08528	8.32225	314.82	-31.74	2.03	0	0.00745	0.00106	14	-0.625	-29.125	0.3484	5.1569	8.3339	-0.0207	-0.0256	0.0615	98	0	VIBRATE	Happy
27	0.33519	5.00867	8.18817	314.82	-31.65	1.93	0.00958	-0.0064	0.00745	14.125	-1.1875	-28.75	0.3314	5.1434	8.3429	-0.0332	-0.0318	-0.065	96	0	VIBRATE	Happy

page_1 | happy | depressed | angry | peace

图 3.3 原始数据中提取“happy”情绪



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	-1.1492	4.35745	8.74362	30.84	-27.68	-7.76	0.01065	-0.033	0.04046	-21.625	28.6875	-21.875	-1.3247	4.515	8.604	0.0568	-0.1512	0.1421	45	0	VIBRATE	Depressed
2	-1.5514	3.9648	8.41801	30.3	-23.87	-8.71	0	0.07772	0.00639	-21.313	32.1875	-21.688	-1.4861	3.9232	8.8639	0.0805	0.1548	0.1377	47	0	VIBRATE	Depressed
3	-1.3695	3.07415	8.89685	29.58	-20.48	-7.1	-0.0958	-0.066	0.07346	-21.375	34.5625	-22.625	-1.2132	3.4055	9.1159	-0.0936	-0.113	-0.2019	52	1.1	VIBRATE	Depressed
4	0.85234	2.77727	9.75877	36.23	-17.88	3.84	-0.2119	-0.2385	-0.0128	-30.625	35.4375	-26.375	0.6571	3.0041	9.312	0.1484	-0.1931	0.23	72	0	VIBRATE	Depressed
5	-1.5227	-0.4214	9.94073	15.78	4.64	-7.57	-0.2129	-0.017	-0.0724	-18.188	36.6875	-8.5625	-1.2972	-0.7446	9.6918	-0.2451	0.1547	0.4001	155	0	VIBRATE	Depressed
6	-2.0973	-0.4693	9.87369	24.99	4.05	-11.05	0.09901	0.16396	0.16715	-15.75	34	-7.875	-1.8798	-0.6807	9.6006	-0.262	0.6793	0.2263	191	0	VIBRATE	Depressed
7	0.26815	0	9.72046	25.85	-0.43	-0.72	0.00319	0.00958	0.11179	-19.125	31.125	-12.375	-0.124	0.0736	9.8055	0.3395	0.4952	0.0022	177	0	VIBRATE	Depressed
8	-0.0958	-0.0862	9.73004	33.39	0.27	-0.52	0	0.00213	-0.0011	-20.688	30.75	-13.25	-0.0891	-0.0464	9.8061	0.0007	-0.0222	-0.0668	146	0	VIBRATE	Depressed
9	-0.1245	-0.0575	9.76834	33.92	0.38	-0.62	-0.0011	0.00213	-0.0011	-20.938	32.25	-12.813	-0.1078	-0.0664	9.8058	-0.0003	-0.0318	-0.0099	148	1.2	VIBRATE	Depressed
10	-0.1245	-0.0862	9.72046	33.93	0.41	-0.66	-0.0021	0.00106	-0.0011	-20.813	31.375	-12.438	-0.1134	-0.0709	9.8057	-0.0049	-0.0276	0.0294	148	0	VIBRATE	Depressed
11	-0.1053	-0.0862	9.73961	33.92	0.42	-0.61	0	0.00106	-0.0011	-20.688	31.0625	-12.125	-0.1047	-0.0727	9.8058	-0.0035	0.0135	-0.0391	146	0	VIBRATE	Depressed
12	-0.0958	-0.0575	9.76834	33.93	0.44	-0.58	-0.0021	0.00213	-0.0011	-20.813	31.0625	-12.563	-0.0997	-0.0761	9.8058	-0.0184	0.0267	0.0094	148	0	VIBRATE	Depressed
13	-0.1053	-0.0862	9.74919	33.93	0.41	-0.59	-0.0011	0.00213	-0.0021	-21.5	31.625	-12.688	-0.101	-0.0716	9.8058	0.0122	0.0123	-0.0294	146	0	VIBRATE	Depressed
14	-0.0958	-0.0862	9.73004	33.93	0.39	-0.62	-0.0021	0.00213	-0.0011	-20.938	31.0625	-12.688	-0.1064	-0.0657	9.8058	0.0172	-0.023	0.0389	148	0	VIBRATE	Depressed
15	-0.0958	-0.067	9.75877	33.93	0.41	-0.61	-0.0011	0.00213	0.00106	-20.313	30.3125	-13	-0.1051	-0.071	9.8058	-0.0133	-0.0177	0.0099	146	0	VIBRATE	Depressed
16	-0.1341	-0.0862	9.75877	33.93	0.39	-0.63	-0.0021	0.00213	-0.0021	-21.5	31.1875	-12.688	-0.108	-0.0669	9.8058	-0.0001	0.0373	-0.0197	148	1	VIBRATE	Depressed
17	-0.1149	-0.0862	9.74919	33.93	0.35	-0.61	0	0.00213	-0.0011	-21.375	30.875	-13.125	-0.1048	-0.0609	9.8059	0.0061	0.0115	0	148	1	VIBRATE	Depressed
18	-0.1149	-0.067	9.73961	33.93	0.4	-0.6	-0.0011	0	-0.0011	-21.5	31.0625	-12.313	-0.104	-0.0696	9.8058	-0.0045	0.0005	-0.0007	150	0	VIBRATE	Depressed
19	-2.2027	2.50912	8.07325	40.39	-15.11	-11.69	0.81978	-0.0543	-0.2172	-15.313	34.3125	-15.125	-1.9875	2.5041	9.2708	-0.37	-0.2116	-1.3709	154	0	VIBRATE	Depressed
20	-2.4229	3.56257	8.64786	28.44	-22.97	-15.39	0.17354	-0.0788	-0.0213	-14.75	31.9375	-18.438	-2.6035	3.6909	8.7045	0.5457	0.2965	0.7703	52	0	VIBRATE	Depressed
21	-0.6704	3.4668	8.08282	340.31	-32.37	-6.44	0.04259	-0.2694	0.23529	15.625	8.4375	-44.625	-1.1946	5.2891	8.1711	0.4663	-1.6253	0.014	9860	1	VIBRATE	Depressed
22	-0.9481	3.83072	7.78594	340.04	-32.52	-6.19	0.10008	-0.132	0.26723	15.75	8.1875	-44.375	-1.1016	5.2178	8.2298	0.2762	-1.6626	-0.1617	9757	1	VIBRATE	Depressed
23	-0.8715	3.91691	7.29753	340.04	-32.52	-6.19	0.033	0.20867	0.29491	15.75	8.1875	-44.375	-1.1016	5.2178	8.2298	0.2762	-1.6626	-0.1617	9757	1	VIBRATE	Depressed
24	-0.8715	3.91691	7.29753	340.04	-32.52	-6.19	0.033	0.20867	0.29491	15.75	8.1875	-44.375	-1.1016	5.2178	8.2298	0.2762	-1.6626	-0.1617	9757	1	VIBRATE	Depressed
25	-0.8715	3.91691	7.29753	340.04	-32.52	-6.19	0.033	0.20867	0.29491	15.75	8.1875	-44.375	-1.1016	5.2178	8.2298	0.2762	-1.6626	-0.1617	9757	1	VIBRATE	Depressed
26	-0.8715	3.91691	7.29753	340.04	-32.52	-6.19	0.033	0.20867	0.29491	15.75	8.1875	-44.375	-1.1016	5.2178	8.2298	0.2762	-1.6626	-0.1617	9757	1	VIBRATE	Depressed
27	-0.8715	3.91691	7.29753	340.04	-32.52	-6.19	0.033	0.20867	0.29491	15.75	8.1875	-44.375	-1.1016	5.2178	8.2298	0.2762	-1.6626	-0.1617	9757	1	VIBRATE	Depressed

page_1 | happy | depressed | angry | peace

图 3.4 原始数据中提取“depressed”情绪

3.2 用户数据预处理

本实验中收集来的大量原始数据，是不能直接被使用的。收集来的原始数据有可能会存在包含大量噪声，数据不完整（比如有的属性可能缺失，或者不确定），数据不一致（比如在不同表中的同一属性名称不一致或者数据矛盾），度量单位不一致等问题[13]。可能产生的原因有，比如收集过程中设备使用方法不正确，用户操作不当，或者设备出现故障和异常，信息收集过程受到干扰和中断，再比如数据收集后数据存储不当，工作人员误操作等等。这就需要通过科学的数据预处理技术对数据进行清洗，剔除坏值，非法值，异常值，填补空值等操作，来消除数据中的噪声，保证数据数据一致性及数据完整性，保证收集来的数据有效和可用。除此之外，

还需要数据进一步进行集成，规约，变换等操作，使得数据满足研究分析的要求。

数据的预处理可以显著提高数据的质量，同时可以有效地提高后续过程中数据分析的效率。Python 中提供了强大的 pandas 和 numpy 库，本文作者使用这两个科学的数据分析库进行了数据的预处理操作。

3.2.1 数据集成

数据集成是把多组源数据融合成一组数据，这多组源数据可能来自多个不同的数据库或者不同的文件，所以集成的过程中要消除数据不一致和数据冗余。数据不一致主要表现为属性名称不一致或者数据矛盾。数据冗余一般是同一属性名称多次出现或者属性间存在线性关系。一般通过相关性分析来消除属性间线性相关。此处主要介绍相关的数据拼接和合并，有关属性间的相关性造成的数据冗余在后面单独会有介绍。

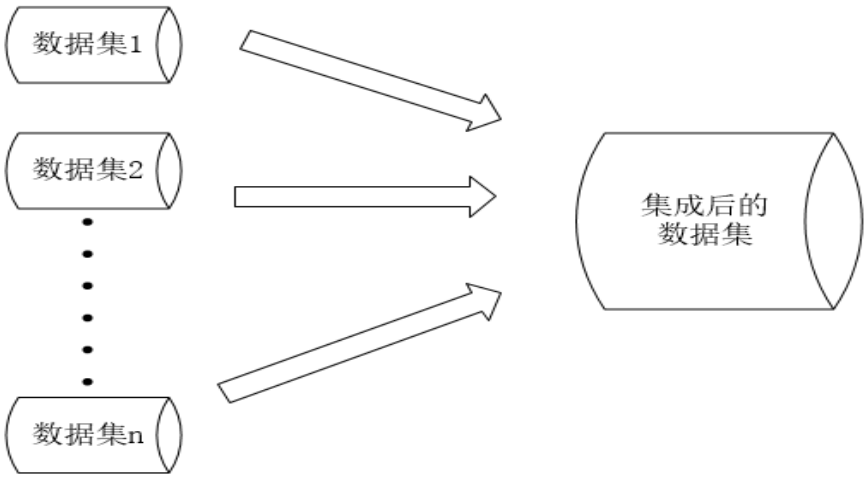


图 3.5 数据集成示意图

本实验由于是从多个用户收集信息，每个用户收集的信息保存在不同的文件中，所以需要将多份数据合并在一起。本文作者首先将多份文件通过 pandas 读文件方法将 txt 文件读入，分别存储为不同的 DataFrame（Pandas 中的一种特殊数据结构，表现形式为二维数组），然后将不同的 DataFrame 拼接在一起。其中需要注意的是，防止数据不一致和数据冗余，不同文件的数据格式要保持一致，同一个属性在不同的文件中是否有不同属性名。因为数据分几次收集，前期收集到一部分数

据后，又进行了数据收集工具的改进，数据格式存在一定的调整，在进行拼接时，本文作者使用 pandas 调整了属性顺序，然后进行拼接操作。

	1	2	3	4	5	6	\
0	-4.098873	4.118027	7.345411	326.91998	-28.890000	-24.490000	
1	-3.438074	5.334281	7.354988	316.19998	-36.000000	-22.840000	
2	-3.801992	4.874595	8.073248	317.41000	-34.969997	-22.039999	
3	-3.667917	5.066131	7.067683	316.49000	-34.410000	-22.910000	
4	-3.773262	4.865018	7.268796	316.90000	-33.420000	-23.090000	
5	-3.869030	4.702212	7.479486	317.52000	-32.719997	-23.330000	
6	-3.888184	4.673482	7.479486	317.71000	-32.239998	-23.590000	
7	-3.849876	4.711789	7.441179	318.21000	-32.640000	-23.029999	
8	-3.849876	4.616021	7.594408	318.38998	-31.740000	-23.189999	
9	-3.591303	4.884171	7.412448	317.87000	-33.110000	-22.380000	
10	-3.639187	4.874595	7.527370	317.78998	-33.630000	-21.730000	
11	-3.514688	4.893748	7.441179	319.07000	-32.899998	-21.580000	
12	-3.600879	4.778827	7.584831	319.22998	-32.670000	-21.740000	
13	-2.049437	8.188170	4.204218	352.00998	-64.979996	-11.599999	
14	2.786851	6.952762	4.979939	284.13998	-59.289997	4.460000	
15	-15.763424	-5.056554	11.185710	34.66000	-13.809999	-38.190000	
16	-1.244985	8.245630	7.594408	288.99000	-51.309998	8.660000	
17	-0.229843	7.058106	3.859453	291.22000	-72.130000	-0.510000	
18	0.172383	7.259220	6.071695	290.71000	-51.390000	3.220000	
19	-4.826711	6.196194	6.157887	311.53998	-45.309998	-26.960000	
20	-4.204218	5.813122	6.780379	311.87000	-42.910000	-28.090000	
21	-4.807557	5.449203	6.560112	311.36000	-41.309998	-30.109999	
22	-4.960786	6.052542	5.468357	308.43000	-47.120000	-30.779999	
23	-4.261679	4.922479	6.311116	322.94998	-36.550000	-25.010000	
24	-5.257667	6.483498	3.936068	306.47998	-54.570000	-36.010000	
25	0.383072	4.817134	8.303091	312.91998	-30.340000	1.400000	
26	0.229843	5.037400	8.264784	313.55000	-31.550000	1.970000	
27	0.268151	5.066131	8.274361	313.71000	-31.679998	1.880000	

	7	8	9	10	...	15	16	\
0	-0.442895	0.328977	0.593011	20.8125	...	7.812700	-0.159500	
1	-0.064944	0.028746	0.100077	24.5000	...	7.311400	-0.080800	
2	-0.178861	0.276809	0.037263	23.8750	...	7.448400	0.155500	
3	-0.020228	0.015970	-0.008517	23.0000	...	7.451900	0.095300	
4	-0.021293	-0.002129	-0.018099	23.5000	...	7.530800	0.142000	
5	-0.029810	0.062814	0.017034	23.6250	...	7.575200	-0.103900	
6	0.010647	-0.011711	-0.007453	23.9375	...	7.601100	0.082900	
7	0.010647	0.017034	-0.009582	23.6250	...	7.598800	-0.032900	
8	-0.009582	-0.005323	-0.007453	23.1250	...	7.665000	0.031600	
9	-0.040457	-0.002129	-0.006388	23.5625	...	7.594800	0.052200	
10	-0.017034	-0.009582	-0.025552	22.6250	...	7.583700	-0.000100	
11	0.001065	-0.009582	0.004259	22.6250	...	7.656100	-0.024300	
12	0.003194	-0.006388	0.000000	22.5000	...	7.667300	-0.008900	
13	-0.199090	-0.223577	-0.215059	7.0625	...	4.061800	-0.090700	
14	-0.488675	1.401080	1.215831	18.6875	...	4.992300	1.559600	
15	0.238482	0.525937	1.238189	6.7500	...	7.631100	-10.925899	
16	0.648372	-0.655825	-1.758803	19.5625	...	6.059200	-2.254700	
17	-1.653403	-1.130659	-0.518485	22.1875	...	3.008900	0.842800	
18	0.081978	0.025552	-0.507838	20.6250	...	6.109200	-0.329100	
19	-0.209736	0.341753	-0.216124	27.8125	...	6.146300	-0.784100	
20	-0.281068	0.006388	-0.070267	28.5000	...	6.271300	0.331100	
21	-0.045780	-0.058556	-0.023422	28.2500	...	6.371700	0.017400	
22	0.044715	0.103271	0.019164	30.3125	...	5.731700	0.027400	
23	-0.331106	-0.931569	-0.288520	24.5000	...	7.028500	-0.072100	
24	0.002129	0.034069	0.062814	30.8750	...	4.597200	0.392900	
25	0.017034	-0.006388	0.041521	15.0000	...	8.460899	-0.051500	
26	-0.018099	0.001065	-0.046845	14.2500	...	8.349200	0.055300	
27	0.006388	-0.003194	-0.008517	14.2500	...	8.340400	0.021900	

	17	18	19	20	21	22	23	24
0	-0.1617	-0.3845	34.124527	108.834654	87.0	0.0	VIBRATE_MODE	Peace
1	-0.1974	0.1304	34.124527	108.834654	87.0	0.0	VIBRATE_MODE	Peace
2	-0.2250	-0.0048	34.124527	108.834654	88.0	0.0	VIBRATE_MODE	Peace
3	-0.0036	-0.1750	34.124527	108.834654	90.0	0.1	VIBRATE_MODE	Peace
4	-0.0043	-0.2431	34.124527	108.834654	92.0	0.0	VIBRATE_MODE	Peace
5	-0.1616	0.1247	34.124527	108.834654	92.0	6.0	VIBRATE_MODE	Peace
6	-0.0206	-0.0479	34.124527	108.834654	94.0	0.0	VIBRATE_MODE	Peace
7	-0.0642	0.0230	34.124527	108.834654	94.0	0.0	VIBRATE_MODE	Peace
8	-0.0577	0.0360	34.124527	108.834654	94.0	0.0	VIBRATE_MODE	Peace
9	-0.0405	-0.1495	34.124527	108.834654	91.0	0.1	VIBRATE_MODE	Peace
10	-0.1025	-0.0208	34.124527	108.834654	54.0	0.0	VIBRATE_MODE	Peace
11	-0.0207	-0.0434	34.124527	108.834654	54.0	0.0	VIBRATE_MODE	Peace
12	-0.0635	-0.0054	34.124527	108.834654	54.0	0.0	VIBRATE_MODE	Peace
13	-0.3543	0.1248	34.124527	108.834654	63.0	0.0	VIBRATE_MODE	Peace
14	-1.3337	0.1174	34.124527	108.834654	35.0	1.1	VIBRATE_MODE	Peace
15	-7.2071	4.0314	34.124527	108.834654	92.0	0.0	VIBRATE_MODE	Peace
16	0.8382	1.2892	34.124527	108.834654	85.0	0.0	VIBRATE_MODE	Peace
17	-1.3671	-0.3940	34.124527	108.834654	91.0	0.0	VIBRATE_MODE	Peace
18	-0.3132	-0.0251	34.124527	108.834654	86.0	0.0	VIBRATE_MODE	Peace
19	-0.0212	-0.2187	34.124527	108.834654	92.0	0.0	VIBRATE_MODE	Peace
20	-0.0544	0.1971	34.124527	108.834654	99.0	1.0	VIBRATE_MODE	Peace
21	-0.0711	0.2156	34.124527	108.834654	99.0	0.0	VIBRATE_MODE	Peace
22	-0.0597	-0.2306	34.124527	108.834654	94.0	1.2	VIBRATE_MODE	Peace
23	-0.2190	-0.6490	34.124527	108.834654	63.0	0.0	VIBRATE_MODE	Peace
24	0.0954	-0.6984	34.124527	108.834654	100.0	0.0	VIBRATE_MODE	Peace
25	-0.0293	0.1322	34.124527	108.834654	98.0	0.0	VIBRATE_MODE	Peace
26	-0.0304	-0.1202	34.124527	108.834654	97.0	0.0	VIBRATE_MODE	Peace
27	-0.0461	-0.0233	34.124527	108.834654	97.0	0.1	VIBRATE_MODE	Peace

图 3.6 拼接后部分数据截图

3.2.2 缺失值处理

通过 numpy 的 isnan 方法（numpy 中的查询空值的方法，返回 DataFrame 的空值和非空值情况）发现，收集来的数据中明显存在一些缺失值，可能造成这一现象原因可能有用户忘记填写、应用被关闭或者后台清理、手机关机或没电等，也有可能是应用出现 bug，未能正常收集数据。

如果有连续多个缺失值，应当采取的操作是将连续空缺的几组数据删除。如果是个别属性不连续的出现缺失，一般的处理方法有中位数替代法，平均值替代法，

频率最高值替代法，默认值替代法，邻近值替代法或者根据需要直接删除属性缺失的行或者列。考虑到人的心情在一段时间内是相对稳定的，所以各项数据也应该是基本稳定的，在本实验中，本文作者采取同组的平均值进行缺失值的替换。

3.3 数据属性观察

通过对收集来的数据进行直观分析，初步判断用户数据中的 GPS 经纬度信息不具有明显变化，造成这一现象的原因可能有用户群体大多是在实验室进行毕业设计的大四同学，一天中待在实验室的时间较长，活动范围有限，基本上存在变化的时刻集中出现在中午就餐时刻，但由于手机内置的 GPS 传感器精度有限，无法明显捕捉到这样小范围的移动。然后，提取出每组数据中的 GPS 经纬度信息，使用 MATLAB 进行绘图，显示出的情况与人为判断基本一致，近乎集中于两三个点。鉴于这种情况，本文作者判断这两个属性对于后面模型构建以及预测分析不具备太大的参考价值，故选择直接删除这两项。

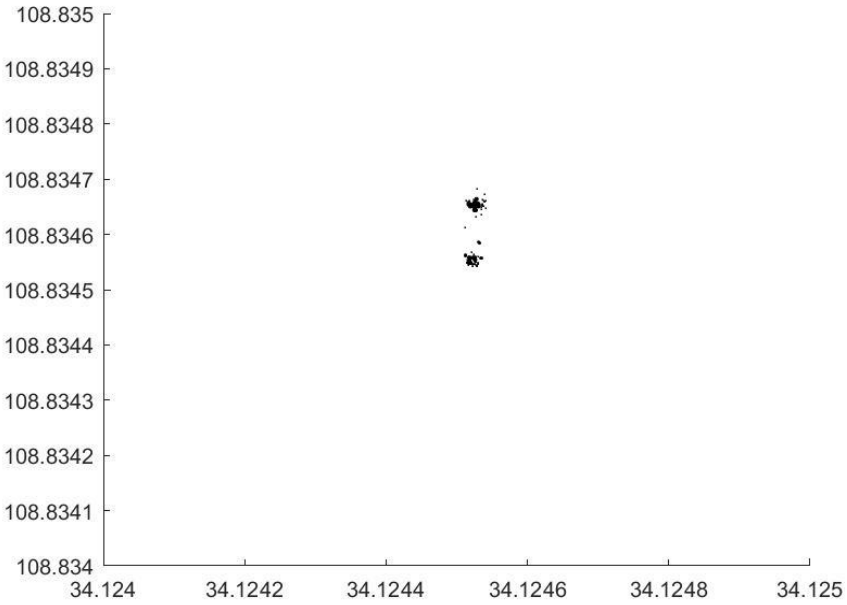


图 3.7 GPS 经纬度数据分布图

3.4 属性的相关性分析

本实验对获得的 20 个变量中，可能出现线性相关的变量之间，进行了相关性

分析，以防止数据线性相关造成的数据冗余。

相关性的计算方法：

属性 A 和属性 B 的相关性计算，

$$r_{A,B} = \frac{\sum(A-\bar{A})(B-\bar{B})}{(n-1)\sigma_A\sigma_B} \quad \text{式（3-1）}$$

$$\sigma_A = \sqrt{\sum(A-\bar{A})^2} \quad \text{式（3-2）}$$

$$\sigma_B = \sqrt{\sum(B-\bar{B})^2} \quad \text{式（3-3）}$$

如果$r_{A,B} > 0$，则有 A 与 B 正相关。

如果$r_{A,B} < 0$，则有 A 与 B 负相关。

如果$r_{A,B} = 0$，则有 A 与 B 相互独立。

如果$|r_{A,B}|$很大，那么就说明 A，B 相关性很强，可以删除其中一个。

3.5 加速度合成

本实验中使用到的加速度类传感器有加速度传感器和线性加速度传感器，每种传感器都有 x，y，z 三个轴，因为本实验中不需要区分具体的方向，所以把两类加速度传感器的 x，y，z 三轴的分量数据（即加速度 x，加速度 y，加速度 z，线性加速度 x，线性加速度 y，线性加速度 z 六个属性），分别进行合成，合成后形成两个新的加速度属性——加速度和线性加速度，特征合成以后的操作都使用新特征进行操作，不再使用旧特征，并将旧属性从数据列表中剔除。将这几项特征合成后，使用合成后的值进行操作，可以有效减少训练模型时的时间复杂度，而且不损失精度。

加速度合成公式：

$$\mathrm{a} = \sqrt{(a_x)^2 + (a_y)^2 + (a_z)^2} \quad \text{式（3-4）}$$

其中，a 代表合成后的加速度，a_x，a_y，a_z分别代表 x，y，z 三轴上的分量加速度。

3.6 数据特征提取

直接收集来的数据特征可能数据关系不够明显，考虑到人的心情是由一段时

间的一个平均状态来表现的，所以对收集来的数据，采用分箱技术进行切片分组，以 10 为单位分组，对每一组数据求出平均值、最大值、最小值、方差作为新的特征。

最大值：一组数据中其他值都小于等于数据中的某个值，这个值就是最大值。

最小值：一组数据中其他值都大于等于数据中的某个值，这个值就是最小值。

均值：即一组数据中的平均数，

$$x_{mean} = \frac{x_1+x_2+\cdots+x_n}{n} \quad \text{式（3-5）}$$

方差：一组数据中每个样本与该组数据平均值的偏离程度，

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i-x_{mean})^2}{n} \quad \text{式（3-6）}$$

表 3.1 传感器数据特征提取表

数据类型	传感器	特征数量	特征总数	
运动数据	方向传感器	12	32	64
	加速度	4		
	线性加速度	4		
	陀螺仪	12		
环境数据	重力传感器	12	28	
	磁力计	12		
	光传感器	4		
手机使用情况	网速	4	4	

3.7 主成分分析

经过前面的数据预处理和特征工程，本文作者已将原数据整理成了500×64的样本集，为了体现整个过程的科学性和严谨性，也为了进一步从中已有的样本中提取对结果影响比较大的有关变量，减小无关变量和噪声的影响，以及降低计算量，本文作者对获得的500×64新样本集进行了主成分分析操作。

通过调用 sklearn.decomposition 模块的 PCA 方法，生成一个 PCA 实例。PCA 降维可以将数据降到指定维数，但是考虑到将数据样本降低越多固然会降低更多的计算量，但是也可能导致预测准确度过低的问题，所以一般降维时多选择指定降维后的最小精度（即保证降维后，新数据集保留的原数据集信息在指定精度以上）

或者设置为“mle”，由 PCA 函数自动确定降低的维数。因为本实验中不知道降低的维数对于精度的影响，所以不容易指定维数，本实验中选择设置为“mle”方式，PCA 函数自动选择最优的降维处理。经过 PCA 降维处理后，得到了新的 60 维样本集。

3.8 数据标准化

收集来的数据因为单位不统一、量纲不统一，是无法直接用来分析的。因为分析时往往不清楚各个属性对于结果的影响，所以一般假设各个属性对于结果的影响是相同的，即权重相同。如果量纲不统一，就可能导致样本之间数量级不统一，有的数据很大，有的数据很小，直接进行分析的话就会导致数值大的属性对结果影响过高，会影响到模型的准确性。

数据标准化主要用来消除量纲的影响，比如把属性按比例缩小，把属性放到一个特定的区间[14]。本文作者收集到的数据包括运动类信息、手机状态类信息、环境信息三大类，其中运动类信息包含了加速度、方向、陀螺仪、重力等传感器数据，手机状态类信息主要包含了网速，环境信息主要包含了光强、磁场强度等数据，经过特征选择和特征提取后，形成了新的 64 维特征向量，即使降维后仍然有 60 维，其中的数据量纲存在巨大差异。在本实验中，本文作者假设每种特征都是与人的情绪状态相关的，并且对情绪状态的影响系数是相同的，为了防止某一特征所占的权重过高，从而过多的影响模型的构建，需要对数据进行标准化操作。在本实验中，本文作者选择了 Z-Sorce 标准化将数据进行了规范化处理。常见的标准化操作还有最大最小规范化，以及对定量特征进行二值化。

1. Z-score 标准化：

根据属性的均值和方差来对属性进行规范化，一般在最大最小化规范化出现异常数据时使用。

$$\vartheta' = \frac{\vartheta - \bar{A}}{\sigma_A} \qquad \text{式（3-7）}$$

其中的$\bar{A}$和σ_A分别为属性 A 的均值和方差。

2. 最小最大规范化：

已知属性区间，将属性的取值范围由 [old_min,old_max] 映射到

[new_min,new_max]

$$\vartheta' = \frac{\vartheta - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A \qquad \text{式（3-8）}$$

该种方法保留了原来数据中存在的关系，但如果将来遇到超过目前属性[old_min,old_max]取值范围的数值，将会引发错误。

3. 对定量特征二值化：

对定量特征进行二值化之前，需要预先设定一个阈值 a，如果比阈值大就赋值为 1，如果比阈值小就赋值为 0。

$$x' = \begin{cases} 0, x \leqq a \\ 1, x > a \end{cases} \qquad \text{式（3-9）}$$

3.9 本章小结

经过前面的数据集成、缺失值处理保证了数据的一致性和完整性，通过属性的相关性分析消除了线性相关的变量，通过加速度合成将重要性不高的属性进行了合并，有效降低了数据的冗余和计算复杂度，特征选择和特征提取减少了无用属性，从原有数据中提取出了最能代表数据特征的属性集合，有效降低了噪声带来的影响，保证了后面模型训练的精度。本实验直接收集来的原始数据涉及 23 个属性，经过这部分的处理，融合成了新的 64 维特征向量，然后经过主成分分析进行进一步降维，降低运算的复杂度，得到了 60 维特征向量。

第四章 构建情绪分析模型

通过前面对数据进行数据预处理操作，本实验得到了一个干净的、有效的样本集，经过特征选择和特征提取操作，获得了最能表现样本特征的特征属性，接下来就需要通过这些特征构建情绪的分析模型。从本质上说，这属于机器学习中有监督学习部分的分类问题，所以我们的目的也就是选择一种合适的分类器，将情绪准确地分类。

机器学习中的分类算法主要有 K 近邻算法、支持向量机算法、朴素贝叶斯算法、决策树算法、随机森林算法、Adaboost 算法[15]。本实验主要选取了 K 近邻算法、支持向量机算法、朴素贝叶斯算法构建情绪分析模型并进行分类。

4.1 K 近邻分类模型

K 近邻分类模型是基于 K 近邻算法构建的。K 近邻算法原理是通过比较待测点与带有标签的样本点的距离，选择 K 个与待测点距离最近的样本点，统计这 K 个点中标签情况，选择比例最高的标签作为待测点的标签。

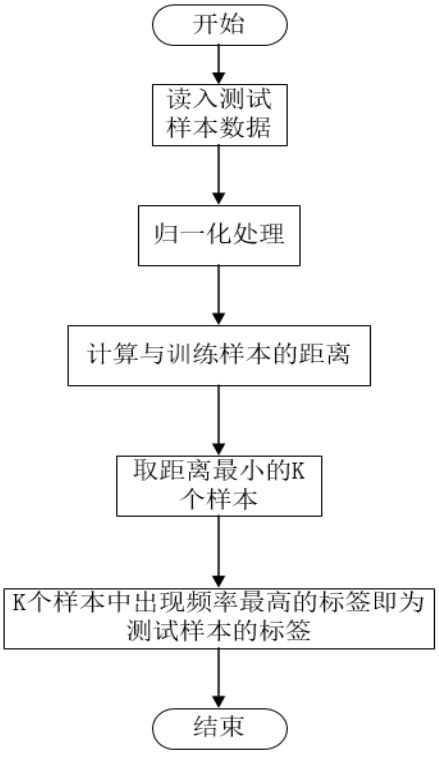


图 4.1 K 近邻分类流程图

K 近邻算法具有理论成熟、简单好用、测试准确率较高、对异常值不敏感等优点。而且考虑到 K 近邻算法要求数据量不能太大，否则会导致计算量过大，也不能数据量太小，这样会容易导致误分，本实验数据量刚好满足这样的要求，基于上述考虑，本实验中选择用 K 近邻分类模型来构建第一个情绪分析模型。

K 近邻算法中最重要的部分是 K 值的选择，不同的 K 值会对 K 近邻模型的分类准确度产生较大影响。通常情况 K 值不应大于 20，而且为避免在通过 K 个标签分类时产生相同比例的标签，K 值一般选择奇数值。具体的 K 值选择可以通过经验判断和交叉验证选择。本实验中，本文作者通过交叉验证选择 K 值。该步骤通过 sklearn.model_selection 模块的 cross_val_score 方法实现，设置 cv 参数为整数，使用 KFold 或 StratifiedKFold 的方法进行数据集打乱。通过设置 K 值的范围，本实验中将该范围设置为了 1~31，然后比较不同 K 值下分类准确率的变化，并通过 matplotlib 模块的 pyplot 方法将不同 K 值下的分类准确率用折线图的方式绘制出来，选择出准确率最高的 K 值。通过交叉验证得到不同 K 值下，K 近邻分类模型准确率变化情况如图 4.2 所示，可以看出 K 值为 16 时，模型的准确率最高，同时泛化能力较强。

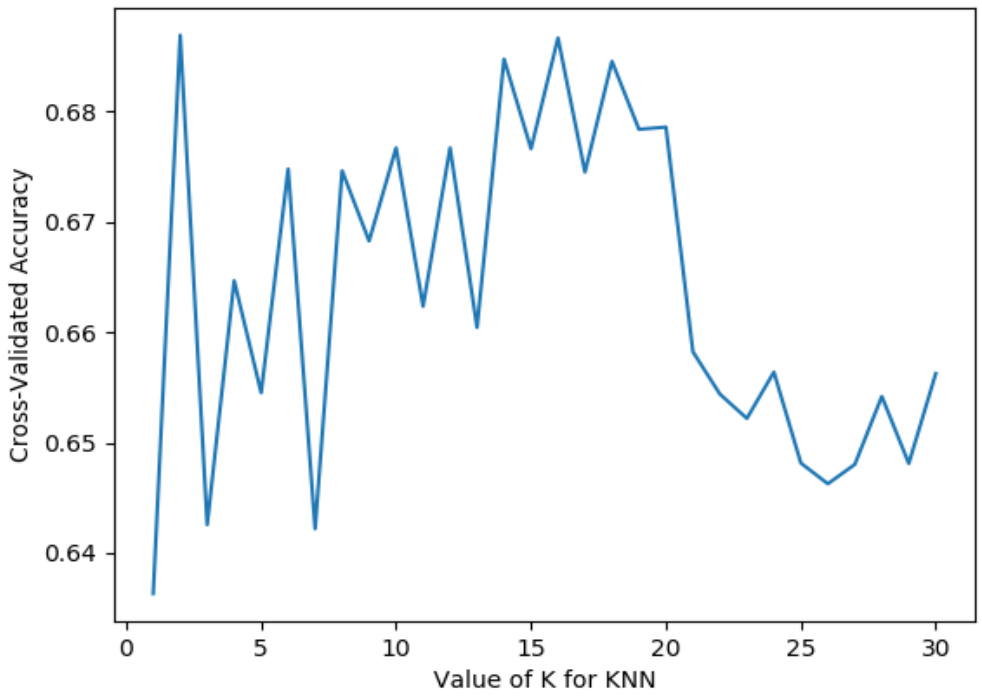


图 4.2 不同 K 值与准确率

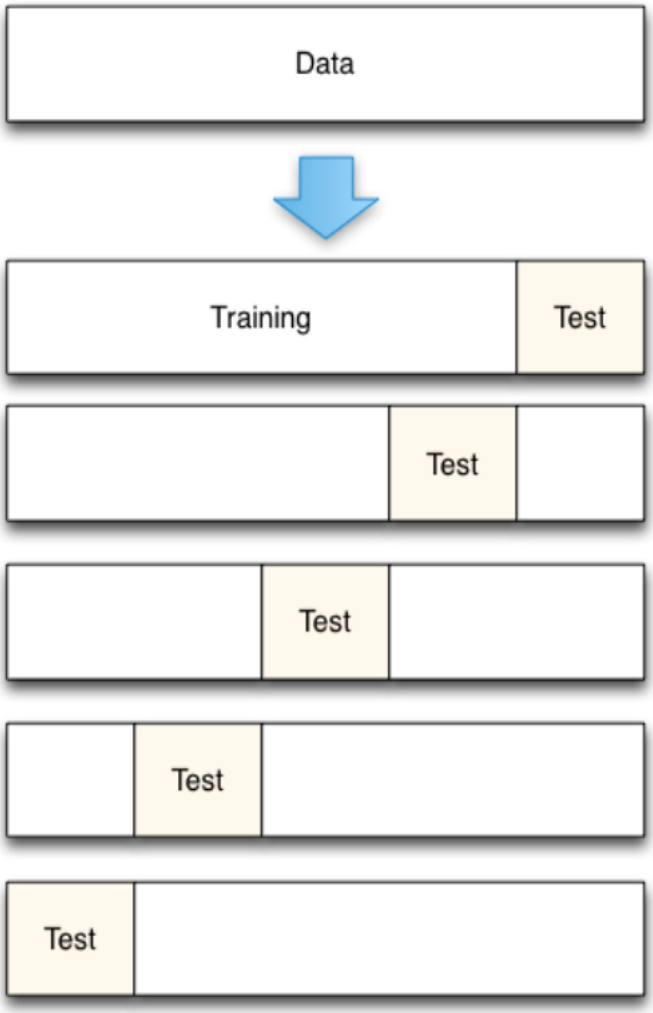


图 4.3 五折交叉验证示意图

然后，通过 sklearn.neighbors 模块的 KNeighborsClassifier 方法生成一个 K 近邻分类器。首先，将 K 值设置为前一步通过交叉验证获得的 K 值。然后设置 K 近邻算法的实现方式。K 近邻算法的实现方式有枚举实现、KD 树实现。枚举实现，即暴力实现，通过挨个搜索待测点距离每个样本点的距离，然后选出 K 个最近邻，这种方式计算量较大，只适合小数据样本。KD 树实现方式没有直接计算待测点距离每个样本点的距离，而是先把数据存储进一个 KD 树，此处的 K 是指 K 个特征，根据构建好的 KD 树模型再进行距离的计算，可以有效减少计算量，提高分类的效率。此处本文作者设置参数 algorithm 为 auto，即由分类器自动选择效率最高的算法。默认情况下，K 个近邻的权重是相同的，但实际情况中可能不是这样的，通常距离待测点越近的样本点的标签越具有参考性，本实验中，作者按照 K 个近邻距离反比为 K 个近邻赋予权重。

经过前面步骤，已经获得了一个 K 近邻分类器，然后传入训练集 X_train 和训练集标签 y_train，即可获得 K 近邻分类模型。

4.2 SVC 分类模型

支持向量机算法的参数只与支持向量有关，数量少，这使得该算法在解决小样本高维度数据集的机器学习问题时具有其他分类器难以媲美的优势，并且支持向量机解决非线性问题有较好效果，泛化能力强，所以本实验中选择支持向量机算法进行了情绪分类模型的构建。

SVC 分类模型是基于支持向量机算法构建的。支持向量机算法理论上仅支持二分类，而情绪具有多种类型，将数据划分为不同情绪类型就是多分类问题。支持向量机解决多分类问题的策略是将多个二分类器组合起来实现一个多分类器，本实验中作者使用了基于支持向量机实现的 SVC（C-Support Vector Classification）来解决情绪的多分类问题。

首先，将收集来的数据按照 80%，20%比例划分为训练集和测试集。

然后通过 sklearn.svm 模块的 SVC 方法生成一个 SVC 分类器。支持向量机解决多分类问题有一对一（one-against-one）和一对多（one-against-rest）两种策略。一对一策略是在任意两类样本之间构建一个支持向量机，本实验中有 4 种情绪类型，4 个类别的样本之间就要构造 6 个支持向量机。然后对未知样本进行情绪分类时，选择得票数高的情绪类型作为未知样本的情绪类型。由于这种方式需要构建的支持向量机较多，影响了模型构建的效率，所以本实验中，作者选择一对多（one-against-rest）的策略构建情绪分析模型，即依次将某个情绪类型的样本归为一类，然后将剩余的类别归为一类，这样 4 个情绪类型的样本就构造出了 4 个 SVM，分类时选择具有最大分类函数值的那类情绪作为待测样本的情绪类型。SVC 解决多分类问题流程图如图 4.4 所示。

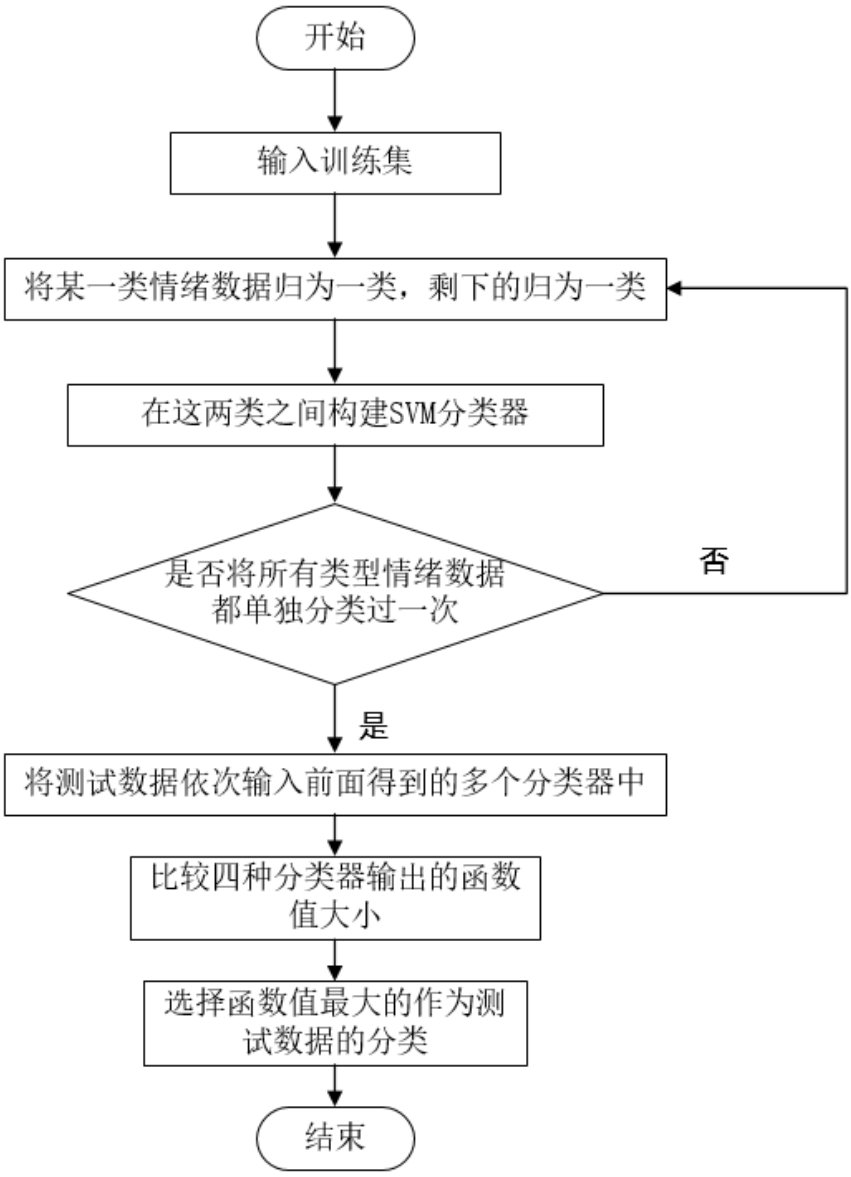


图 4.4 SVC 解决多分类问题流程图

实验中首先选择了线性核函数作为 SVC 分类器的核函数，因为线性核函数简单，计算量小，通常研究过程中常使用这个核函数进行第一次实验。

线性核函数：

$$\kappa(x_i, x_j) = \mathrm{x} \cdot x_i \qquad \text{式（4-1）}$$

因为考虑到情绪相关的数据可能比较复杂，简单的线性核函数可能并不能得到最好的分类效果，实验中还选择了多项式核函数和高斯径向基核函数对情绪数据进行分类。

多项式核函数：

$$\kappa(x, x_i) = ((x \cdot x_i) + 1)^d \quad \text{式（4-2）}$$

高斯核函数：

$$\kappa(x, x_i) = \exp(-\frac{||x-x_i||^2}{\sigma^2}) \quad \text{式（4-3）}$$

最后，将训练集数据及训练集对应的标签传入构造好的支持向量机分类器中，对分类器进行训练，就得到了基于支持向量机的情绪分类模型。

4.3 朴素贝叶斯分类模型

朴素贝叶斯分类模型基于朴素贝叶斯算法，这个模型比较简单。朴素贝叶斯算法基于各个属性对于结果都有影响，并且权重相同的假设。根据$P(x_i|y)$计算方式的不同，朴素贝叶斯有三种实现，高斯朴素贝叶斯、多项式分布朴素贝叶斯、伯努利分布朴素贝叶斯[16]。本实验中选择使用这三种朴素贝叶斯算法进行模型构建。

基于朴素贝叶斯分类模型的主要构建步骤如图 4.5，

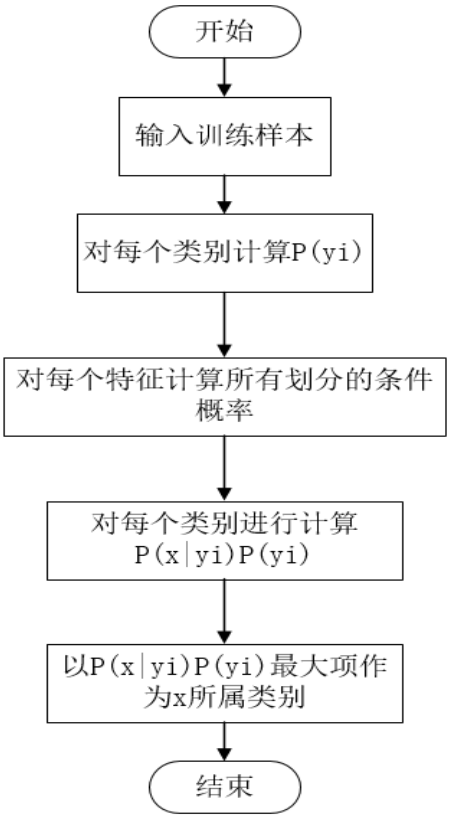


图 4.5 朴素贝叶斯分类模型构建流程图

首先，本文作者将收集来的数据按照 80%，20%比例划分为训练集和测试集。

然后，通过 sklearn.naive_bayes 模块的 GaussianNB、MultinomialNB、BernoulliNB 生成三个朴素贝叶斯分类器，接下来，将已经得到的训练集数据和训练集标签数据分别传入三个分类器，就得到了三种朴素贝叶斯分类模型。

GaussianNB 假设特征的先验概率为正态分布，即如下式：

$$\mathrm{P}(X_j = x_j|\mathrm{Y} = C_k) = \frac{1}{\sqrt{2\sigma_k^2}}\exp(-\frac{(x_j-\mu_k)^2}{2\sigma_k^2}) \quad \text{式（4-4）}$$

其中C_k为 Y 的第 k 类类别。μ_k和σ^2需要从训练集估算得知，μ_k为在样本C_k中，所有X_j的平均值。σ^2为在样本C_k中，所有X_j的方差。

MultinomialNB 假设特征的分布为多项式分布，即如下式：

$$\mathrm{P}(X_j = x_{jm}|\mathrm{Y} = C_k) = \frac{x_{jm}+\lambda}{m_k+n\lambda} \quad \text{式（4-5）}$$

其中$\mathrm{P}(X_j = x_{jm}|\mathrm{Y} = C_k)$是第 k 个类别的第 j 维特征的第 m 个取值条件概率，m_k是训练集中输出为第 k 类的样本个数。λ为一个大于 0 的常数，常常取为 1，即拉普拉斯平滑。也可以取其他值。

BernoulliNB 假设特征的先验概率为二元伯努利分布，即如下式：

$$\mathrm{P}(X_j = x_{jm}|\mathrm{Y} = C_k) = \mathrm{P}(\mathrm{j}|\mathrm{Y} = C_k)x_{jm} + (1 - P(j|Y = C_k)(1 - x_{jm})) \quad \text{式（4-6）}$$

此时 m 只有两种取值，x_{jm}只能取 0 或者 1。

第五章 运行与测试

5.1 准确性测试

准确性测试主要是针对不同情绪分析模型，输入相同的样本，比较输出结果与真实结果。将前面划分好的数据集，取出测试集最后十个数据，如图 5.1 所示。

	avg_1	max_1	min_1	std_1	avg_2	max_2	min_2	\
489	0.040791	0.041028	0.301989	0.029512	0.052409	0.058168	0.137678	
490	0.046696	0.040983	0.307702	0.027960	0.027964	0.050730	0.127941	
491	0.042809	0.040488	0.300799	0.024058	0.036199	0.061225	0.130221	
492	0.040199	0.040278	0.302176	0.028031	0.024074	0.050050	0.133030	
493	0.038759	0.035163	0.305253	0.022721	0.043153	0.055768	0.152432	
494	0.041518	0.037831	0.310797	0.019324	0.040490	0.059750	0.127325	
495	0.039745	0.040313	0.310327	0.020902	0.034529	0.063071	0.158156	
496	0.038850	0.038572	0.303090	0.029277	0.039477	0.050140	0.127511	
497	0.041475	0.036169	0.297914	0.021043	0.036749	0.063872	0.139454	
498	0.042428	0.040752	0.305042	0.029560	0.042795	0.057443	0.157071	

	std_2	avg_3	max_3	...	min_18	std_18	avg_21	\
489	0.047958	0.686735	0.883936	...	0.121219	0.755933	0.600789	
490	0.054892	1.000000	0.937207	...	0.031705	0.883953	0.542016	
491	0.065670	0.642278	0.972083	...	0.014422	0.900753	0.460629	
492	0.044203	0.937402	0.951596	...	0.121870	0.965151	0.437788	
493	0.055586	0.746784	0.934581	...	0.025347	0.989479	0.442695	
494	0.057416	0.624103	0.965247	...	0.091797	0.576974	0.562428	
495	0.051262	0.805069	0.928481	...	0.045501	0.803166	0.586345	
496	0.044737	0.668443	0.835052	...	0.035374	0.848266	0.522606	
497	0.059686	0.470855	0.786579	...	0.243743	0.762524	0.474755	
498	0.048978	0.928961	0.952168	...	0.146689	0.738947	0.431046	

	max_21	min_21	std_21	avg_22	max_22	min_22	std_22
489	0.860296	0.271660	0.406984	0.663983	0.935155	0.126868	0.694855
490	0.959657	0.010608	0.735612	0.643405	0.896879	0.075166	0.756918
491	0.854971	0.049195	0.524415	0.795502	0.949320	0.186211	0.711134
492	0.996007	0.093100	0.519750	0.706557	0.808065	0.121896	0.668763
493	0.883922	0.029131	0.637946	0.769752	0.962402	0.315343	0.747197
494	0.950368	0.059435	0.700364	0.667646	0.944999	0.607877	0.569509
495	0.954472	0.148664	0.622349	0.997669	0.985706	0.074681	0.806584
496	0.934513	0.093099	0.647000	0.603938	0.883546	0.260661	0.736590
497	0.899490	0.016549	0.559173	0.396869	0.781529	0.021396	0.632273
498	0.970259	0.025977	0.671358	0.586924	0.814054	0.241609	0.596285

图 5.1 测试集最后 10 个数据

将取出的 10 个数据分别输入训练好的七个不同的分类模型中，并输出七个分类模型的分类结果，输出结果见表 5.1。

表 5.1 不同分类器对于最后 10 个数据的预测值与真实值

真实标签	KNN	SVC(linear)	SVC(poly)	SVC(rbf)	GaussianNB	MultinomialNB	BernoulliNB
Angry	Angry	Angry	Angry	Angry	Peace	Angry	Happy
Angry	Angry	Angry	Angry	Angry	Angry	Angry	Peace
Depresse	Depressed	Depressed	Depressed	Depresse	Depressed	Depressed	Happy
Angry	Happy	Angry	Angry	Angry	Peace	Angry	Angry
Angry	Angry	Angry	Angry	Angry	Angry	Angry	Depressed
Angry	Happy	Happy	Angry	Angry	Peace	Angry	Happy
Happy	Happy	Depressed	Depressed	Peace	Depressed	Depressed	Happy
Angry	Depressed	Happy	Angry	Angry	Angry	Angry	Happy
Angry	Angry	Angry	Angry	Angry	Angry	Angry	Depressed
Happy	Depressed	Depressed	Depressed	Happy	Depressed	Depressed	Happy

对于训练集最后的 10 个数据，K 近邻、SVC(linear)、SVC(poly)、SVC(rbf)、高斯朴素贝叶斯、多项式朴素贝叶斯、伯努利朴素贝叶斯六个情绪分析模型，分别识别正确了 6 个、6 个、8 个、9 个、5 个、8 个、3 个，与预期效果基本一致。对于整个测试集，不同的模型识别准确率情况如图 5.2 所示。

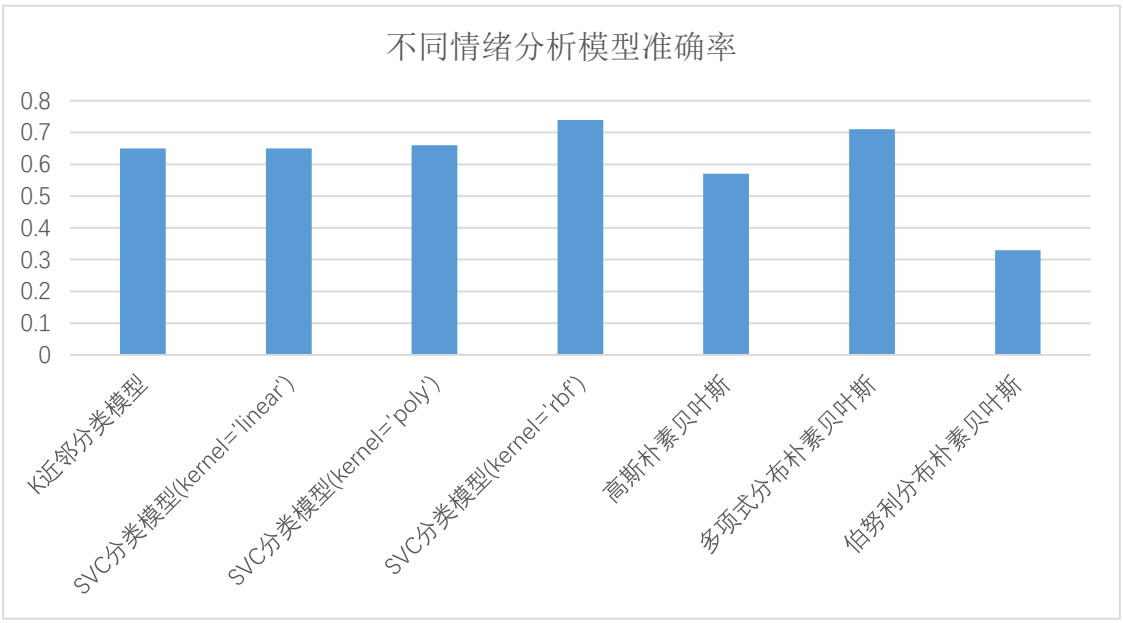


图 5.2 不同模型准确率比较

由图 5.2 可以看出，SVC 分类模型（kernel= ‘rbf’）准确率最高，达到了 74%，接下来依次是多项式分布朴素贝叶斯分类模型，SVC 分类模型（kernel= ‘poly’），K 近邻分类模型和 SVC 分类模型（kernel= ‘linear’），高斯朴素贝叶斯分类模型，

准确率依次为 71%，66%，65%，65%，57%，准确率最低的是伯努利分布朴素贝叶斯分类模型，为 33%。

5.2 性能测试

性能测试部分主要测试的是，对于同一个样本集，采用不同方式构造情绪分类模型并进行识别整个过程的耗时情况。训练集是一个400×60的数据集，测试集大小为100×60。构建不同情绪分类模型的耗时情况如图 5.3 所示。

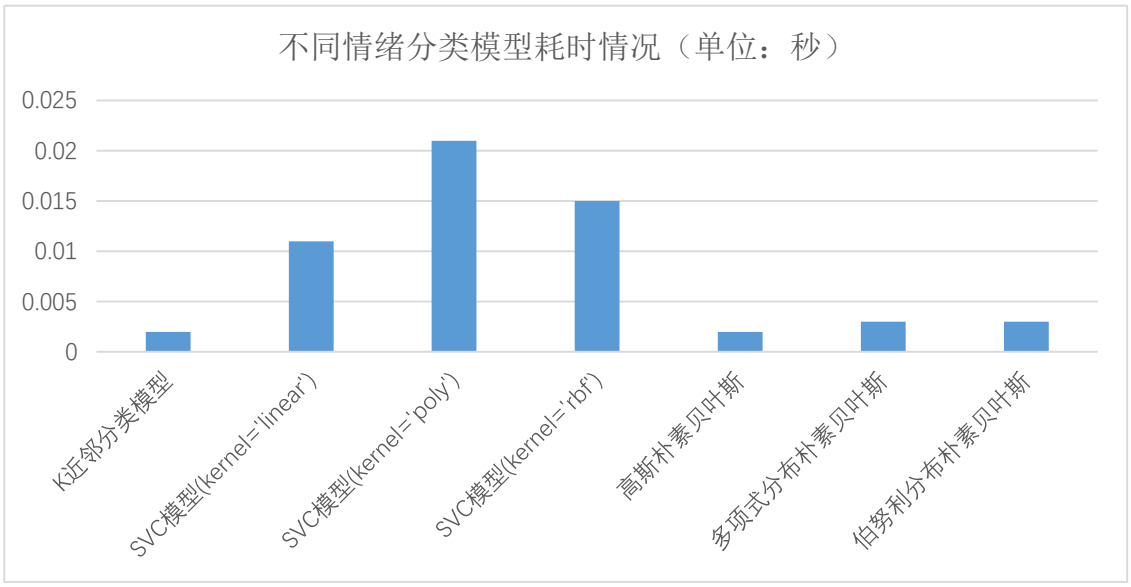


图 5.3 不同情绪分析模型分类性能比较

由图 5.3 可以看出，效率最高的是 K 近邻分类模型和高斯朴素贝叶斯分类模型，出现这种情况的原因是，样本数量比较小，所以 K 近邻分类模型分类效率最高。接下来，效率由高到低依次是多项式分布朴素贝叶斯分类模型和伯努利分布朴素贝叶斯分类模型，SVC 分类模型（kernel=‘linear’），SVC 分类模型（kernel=‘rbf’），效率最低的是 SVC 分类模型（kernel=‘poly’）。

5.3 本章小结

通过对构建出的模型进行了准确性测试和性能测试，结果显示 SVC 分类模型（kernel=‘rbf’）分类准确率最佳，K 近邻分类模型和高斯朴素贝叶斯分类模型分类效率最高，但考虑到样本量比较小，计算时间上的差距不大，都在可接受范围。

所以，综合来说，SVC 分类模型（kernel=‘rbf’）分类效果最佳。

第六章 总结与展望

本文针对智能手机日渐普及以及人们越来越关注自己的情绪的情况，通过收集 Android 用户数据信息，采用机器学习常见分类器如 K 近邻分类器、SVC 分类器、朴素贝叶斯分类器构建了情绪分析模型，实现了对 Android 用户的情绪识别，来帮助人们随时随地了解自己的情绪状态。

本文提出了一种基于机器学习的安卓移动用户情绪分析方法，对于数据的收集过程、处理过程和分析模型的构建都进行了详细的分析和研究，并进行了准确性测试和性能测试。在数据收集过程，针对传感器的选取、收集的数据种类等方面进行了一定的分析。数据收集完成之后，对数据进行了详细的预处理，包括对异常值、缺失值产生原因的分析以及相关的处理，并对不同的原始数据集进行了集成。将数据进行了基本的整理之后，对数据进行了详细的特征提取和特征选择。此过程中，对属性进行了直观观察，进行了相关性的分析，加速度的合成。考虑到一些数据可能表达的数据关系不明显，本文作者又将数据进行了特征提取操作，为使数据满足后面计算需要，又对数据进行了归一化操作。由于新特征集维度比较高，为避免维度灾难，作者又对数据进行了 PCA 降维。然后，利用机器学习算法中三种常见分类器，通过调整参数使之适合已获得的样本集，然后训练出了三种情绪分析模型。最后，对这三种情绪分析模型进行了准确性测试和性能测试，验证的结果表明模型对于情绪的识别效果良好，大多数模型的识别准确率都超过 60%，尤其是采用“rbf”核方法的 SVC 分类模型准确率最高，达到了 74%。性能方面，K 近邻模型和高斯朴素贝叶斯模型处理相同数据集明显耗时更少。综合准确性和性能来看，采用“rbf”核方法的 SVC 分类模型分类效果最好。

取得了一定成果的同时，研究仍然存在着一些不足。首先，数据的收集方面可以进一步改进，本文的研究主要基于运动数据和环境数据以及少量的用户使用数据，后面的研究可以多增加一些用户使用手机情况的数据，如打开某些 APP 的次数，以及使用时长，点亮屏幕的次数，使用 APP 的时间等。其次是，数据的特征提取仍存在改进空间，比如可以提高一些数据的采集频率，模拟成连续数据，然后对连续数据进行特征提取，还可以针对不同的数据进行不同频率的采样，并针对性

的进行特征提取。另外，模型构建时，本文只选用了三种机器学习的分类器进行模型的构建，后续的工作中还可以选用更多的模型进行预测分析，比如随机森林、Adaboost 等。并且，本文只对部分参数进行了调优处理，实际上每种模型可以调节的参数还有很多，还可以在参数调节方面进行改进提升，比如，增加调优的参数个数、改进调优的方法。

本文主要的工作集中在基于机器学习的安卓移动用户情绪分析系统的数据处理与模型构建上，由于本人的水平有限，在某些方面未能进行深入的探讨与研究，难免存在一些不严谨和谬误，敬请各位评委老师指导。

致 谢

时光如梭，转眼间即将离开生活了四年的大学校园，这几年的学习和生活当中，从周围的老师同学、好友亲朋身上得到了很多的帮助，从学习上到生活上，从专业知识到为人处世，在此向所有帮助过我的老师、同学、家人、朋友致以最衷心的感谢。

首先感谢我的指导老师董洛兵老师，感谢董老师在实验过程中为我答疑解惑，耐心细致的帮我解决遇到的困难，指明前进的方向和道路，感谢董老师提供给我的这个学习的平台，为我的实验过程提供了很多便利，接触到了很多前沿的理论和知识，开拓了眼见，增强了自我学习能力。

然后，要感谢所有帮助过我的师兄和师姐们，在百忙之中还对我的毕业设计进行了耐心的指导，并在我毕业设计完成过程中向我提出了不少中肯的意见，使我在完成毕业设计的过程中少走了不少弯路，再次向他们表示感谢。

接着，我要感谢我的父母，一路走来是他们一直在背后支持着我，任劳任怨，默默无闻，失落时的鼓励，沮丧时的宽慰，过往的一幕幕无一不温暖着我的心。我能顺利完成本科学业，我的父母功不可没，真心地感谢他们为我付出的一切，作为子女，我只能更加努力的生活和工作来回报他们，真心祝愿他们身体健康。

最后，感谢所有在我学习和成长道路上帮助过我的人。

参考文献

[1] 孟昭兰. 人类情绪. 上海人民出版社, 1989

[2] 张迪等. 基于生理信号的情绪识别研究进展. 生物医学工程学杂志. 2015,2,13(1)

[3] 傅栩雨,叶健东,王鹏等. 人脸面部表情识别. 计算机与网络,2015,(10):70-71

[4] 赵国朕,宋金晶,葛燕等. 基于生理大数据的情绪识别研究进展.计算机研究与发展,2016,(1):80-92

[5] 姚建盛,李淑梅. Python 在科学计算中的应用. 数字技术与应用,2016(11):76

[6] 齐伟. 跟老齐学 Python——从入门到精通. 北京: 电子工业出版社,2016

[7] Gavin Hackeling. Mastering Machine Learning with scikit-learn. Birmingham: Packt Publishing,2017.7

[8] Peter Harrington. Machine Learning in Action. New York: Manning Publications, 1988.8

[9] Nello Cristianini, John Shawe Taylor. 支持向量机导论. 李国正,王猛,曾华军译. 北京: 电子工业出版社, 2004.3

[10] Peter Harrington. 机器学习实战. 李锐译. 北京: 人民邮电出版社, 2013

[11] Willi Richert, Luis PedroCoelho. 机器学习系统设计. 刘峰译. 北京: 人民邮电出版社,2003.1

[12] Giuseppe Bonaccorso. Machine Learning Algorithms. Birmingham: Packt Publishing,2017.7

[13] Jacqueline Kazil, Katharine Jarmul. Python 数据处理. 张亮,吕家明译. 北京: 人民邮电出版社,2017.6

[14] 沈祥壮. Python 数据分析入门——从数据获取到可视化. 北京: 电子工业出版社,2018.3

[15] Ethem Alpaydin. 机器学习导论. 范明等译. 北京: 机械工业出版社, 2014.4

[16] 米歇尔. 机器学习. 曾华军等译. 北京: 机械工业出版社,2008.3