分类号：TP311.5　　单位代码：10335

密　级：无　　学　　号：Z15030801

浙江大学

硕士学位论文

中文论文题目：基于 R 语言的分布式机器学习平台设计与实现

英文论文题目：Design and Implementation of a Distributed Machine Learning Platform Based on R

申请人姓名：顾全

指导教师：汤斯亮　副教授

合作导师：

专业学位类别：工程硕士

专业学位领域：软件工程

所在学院：软件学院

论文提交日期　2018 年 1 月 1 日

基于R语言的分布式机器学习平台设计与实现

顾全

浙江大学

基于 R 语言的分布式机器学习平台设计与实现

论文作者签名:_____

指导教师签名:_____

论文评阅人 1：_____

评阅人 2：_____

评阅人 3：_____

评阅人 4：_____

评阅人 5：_____

答辩委员会主席：_____

委员 1：_____

委员 2：_____

委员 3：_____

委员 4：_____

委员 5：_____

答辩日期：_____

Design and Implementation of a Distributed Machine Learning Platform Based on R

Author's signature:_____

Supervisor's signature:_____

Thesis reviewer 1：_____

Thesis reviewer 2：_____

Thesis reviewer 3：_____

Thesis reviewer 4：_____

Thesis reviewer 5：_____

Chair：_____

(Committee of oral defence)

Committeeman 1：_____

Committeeman 2：_____

Committeeman 3：_____

Committeeman 4：_____

Committeeman 5：_____

Date of oral defence：_____

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **浙江大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 **浙江大学** 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 **浙江大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

摘要

随着机器学习的盛行，越来越多的公司开发了自己的机器学习云平台，它们大多能提供一个友好的界面供数据挖掘人员较为便捷地部署机器学习的整套流程。本文从实际项目的需求出发，利用数据科学领域比较流行的 R 语言，设计并实现了一个多算法的分布式的机器学习平台 SEML。

首先，本文研究了当前国内外比较知名的大型机器学习云平台，对它们的优缺点进行了比较，并取长补短。随后，本文介绍了 SEML 开发当中所使用的 R、Hive 和 Spark 的相关编程语言或技术，并着重对 SEML 中所使用到的各类机器学习算法进行了分析，这些算法包括 SVM、Random Forest、GBDT、XGBoost、Apriori、Eclat 等，本文从算法原理、数学推导以及伪代码实现方面对它们进行了详细说明。

之后，本文分析了机器学习的一般流程，这包括数据源获取、数据预处理、特征工程、训练和评估、预测推断五个步骤。然后本文从机器学习的一般流程出发，结合机器学习在具体业务场景中的定制化需求，设计了 SEML 机器学习平台的整体架构：以 Hive 为数据底层，通过 Sparklyr 连接 R 和 Hive，将应用层划分为标准化模块和定制化模块，前者又细分为数据导入、模型训练和预测推断三块，后者又细分为差异性特征和共性特征两块，并使用 Shiny 提供最上层的 Web 界面。

最后，本文利用 R 语言对 SEML 进行了编程实现，其中完成的工作包括：（1）利用 Spark 的 R 接口 Sparklyr 连接到 Hive 数据库，编程实现数据导入模块；（2）利用 CARET 集成 R 中的各类机器学习算法，编程实现模型训练以及预测推断模块，包括自动化调参、交叉验证、模型评估及可视化、测试数据的导入及预测推断等功能；（3）利用 Eclat 算法，用代码实现定制化模块的功能；（4）最后利用 R 的 Web 开发框架 Shiny，打通前后台，完整地开发出具备高交互性可视化界面的分布式机器学习平台。实践证明，SEML 的开发，大大缩短了机器学习项目实施的时间，有效地满足了实际项目需求。

关键词： R 语言，Spark，机器学习，平台研发

Abstract

With the prevalence of machine learning, more and more companies have developed their own machine learning cloud platforms, most of which provide a friendly interface for data mining staff to deploy an entire machine learning processes more easily. Based on the actual project requirements, this paper has designed and implemented a distributed machine learning platform named SEML by using R，which is one of the most popular language in data science.

First of all, this paper has studied the current well-known large-scale machine learning cloud platform, compares their strengths and weaknesses. Subsequently, this article has introduced related programming languages or technologies including R, Hive and Spark, and has analyzed all kinds of machine learning algorithms used in SEML, including SVM, Random Forest, GBDT , XGBoost, Apriori, Eclat, etc. In this paper, they are described in detail in terms of algorithm principle, mathematical derivation and implementation of pseudocode.

After that, this paper analyzes the general process of machine learning, including five steps of data source acquisition, data preprocessing, feature engineering, training and inference. Then, following the general process and the customization needs in the specific business, this paper has designed the overall structure of the SEML machine learning platform, dividing the distributed machine learning platform into two parts: the standardization module and the customized module. The former is subdivided into data import, model training and inference, and the latter is subdivided into recognizing different features and finding common features.

Finally, this paper has programmed SEML by using R language. The work we completed includes: (1) Connect to the Hive database by using Sparklyr and fulfil the data import module; (2) Integrate various types of machine learning algorithm by using CARET, then fulfil the training module and inference module, including auto-tuning, cross-validation, model evaluation and visualization, test data import with prediction inference; (3) using Eclat algorithm, Function; (4) Finally, complete the distributed

machine learning platform by providing a user interface via the R Web development framework Shiny. Practice has proved that the development of this machine learning platform has effectively shortened the implementation time of machine learning projects and satisfied the project requirements.

Key Words： R , Spark, Machine Learning, Platform Research

目录

图目录

表目录

第1章 绪论

1.1 研究背景及意义

为了让数据分析人员和机器学习工程师能够快速的针对特定机器学习任务进行模型开发，很多大公司都开发了自己的机器学习云平台，其目的是为了加速整个研发过程、降低人工成本，比如微软的 AMLS，亚马逊的 Amazon Machine Learning 以及国内的如阿里 PAI 和百度 BML。这些平台往往提供了一整套机器学习模块，并且能胜任海量数据的分布式存储和大规模的机器学习建模。另外，这些平台不仅可作为企业的自有工具，还能作为服务或产品供客户使用。鉴于此，本文认为机器学习平台的意义在于，一方面可以让工程师加快对机器学习算法的应用和部署，另一方面可以较为便捷地为不同行业的客户提供机器学习解决方案。

1.2 分布式机器学习研究现状

当前，在并行化机器学习算法的研究方面常见的并行手段主要有以下四种:

（1）多核并行。在多核上实现并行的算法有: K 邻近、决策树、贝叶斯网络[1]、SVM[2]、协同过滤[3]、神经网络[4]。算法多核并行的优点是核与核之间的数据通信快，缺点是单机性能依赖高。

（2）基于 MapReduce 并行。典型的如 Hadoop 的官方分布式机器学习库 Mahout，支持聚类、分类等常见机器学习算法[5]。除了官方以外，基于 Hadoop 实现的分布式算法还包括: 贝叶斯算法[6]、协同过滤[7]。利用 MapReduce 对算法进行多机并行的开创，但缺点是 MapReduce 进程的启动和停止耗费了大量时间。

（3）利用 Spark 并行。同 Hadoop 的 Mahout 一样，Spark 也有官方机器学习库 MLlib[8]，比起 Mahout，MLlib 提供了丰富且完整的算法。由于 Spark 是用迭代计算来设计的，而机器学习的特点就是利用梯度下降等迭代算法对参数进行优化，因此它可以开发大规模机器学习算法的分布式实现。

（4）基于 Tensorflow 进行 GPU 并行。Tensorflow 是谷歌大脑开发的深度学习框架[9]，也是目前最为先进的框架之一，内置 CNN、RNN、LSTM 等多种深度学习算法。由于 Tensorflow 可以利用 GPU 的大量 CUDA 单元做并行计算，因而速

度十分之快。

1.3 大型机器学习平台介绍

1.3.1 微软 AMLS

Azure Machine Learning Studio（AMLS）是微软在 Azure 云计算服务的基础上开发的机器学习平台。它是一个十分强大的整体服务框架，它提供了模块拖拽的工具来构建数据挖掘的 DAG 图，可以调用绝大多数机器学习算法，可以保存和部署实验及模型，可以共享工作空间，还能将模型作为 Web Service 开发自定义应用程序。此外，本文认为，AMSL 区别其他机器学习平台的特色是，它能支持 R 语言和 Python 的各种包和函数库，并支持 R、Python 以及 SQL 的自定义代码。

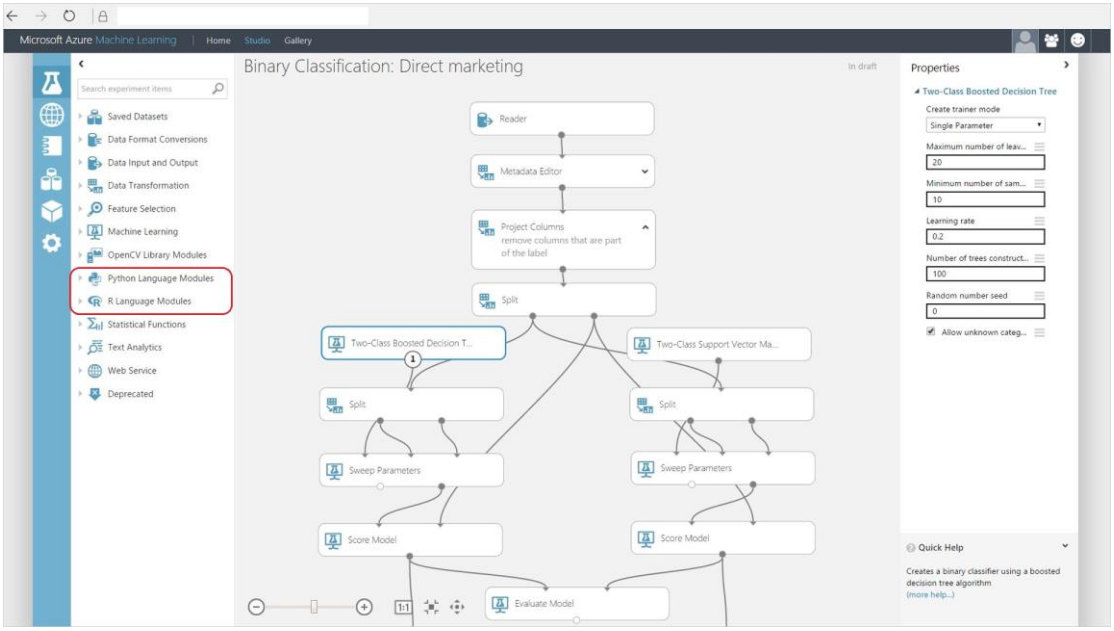

图 1.1 微软 AMLS 界面

1.3.2 亚马逊 AML

Amazon Machine Learning（AML） 是基于亚马逊云的一项机器学习服务，它主要通过用户在 AWS 管理控制台的选择和点击操作来构建机器学习模型。虽

然没有更灵活的拖拽式操作，但整体界面风格清晰，也有不错的用户体验。AML 的主要特点在于它提供了模型评估的多种方式和可视化模型结果的工具，且它在数据量上有很高的扩展性，号称可以对十亿级数据进行实时计算和预测。

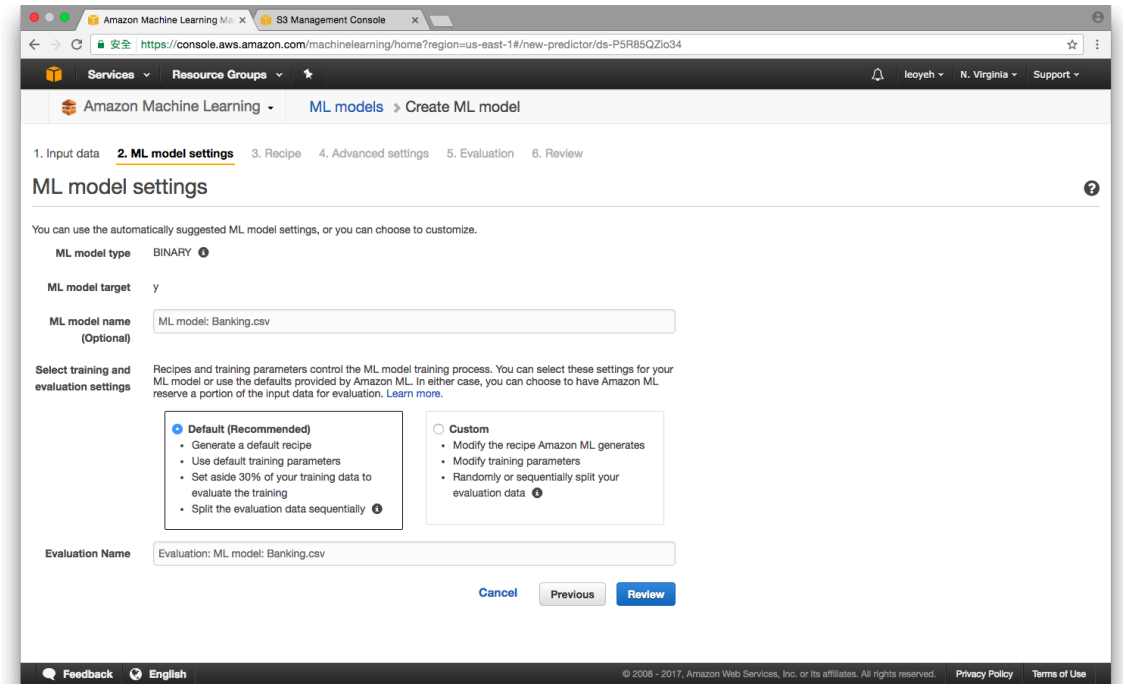

图 1.2 Amazon Machine Learning 界面

1.3.3 阿里 PAI

PAI 全称是 Platform of Artificial Intelligence，由阿里巴巴公司开发，底层号称拥有大量的 CPU 和 GPU 集群，能提供非常高效的算力。PAI 同样是一个支持组件拖拽的平台，它预先对各种算法组件进行了封装，用户只需要将这些组件拖到合适的位置，并以箭头的形式来连接这些组件，即可进行机器学习建模，一定程度上实现了人工智能触手可及。PAI 的主要特点与功能包括：容易上手、算法十分丰富、友好的模型评估界面、同时支持在线预测和离线调度、有多种流行深度学习框架（Tensorflow、Caffe、MXNet）的接口。

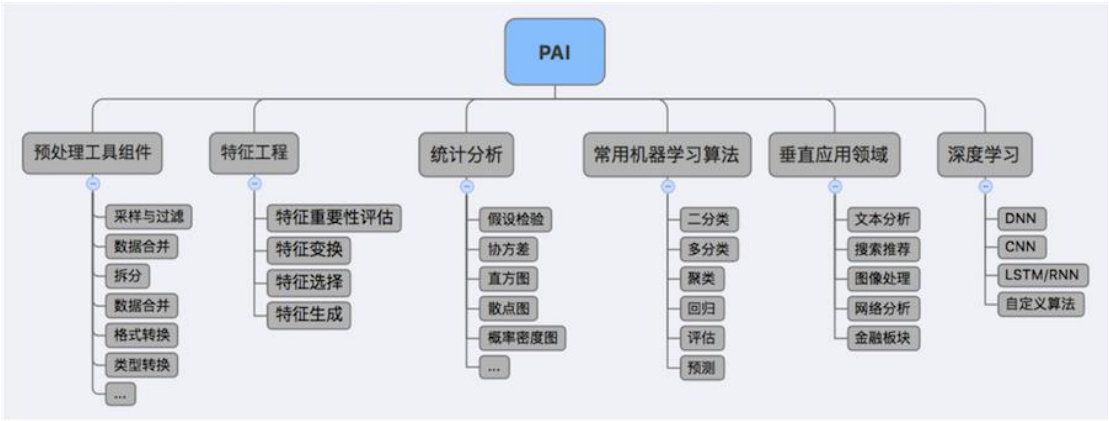

图 1.3 阿里 PAI 所支持的算法

图 1.4 阿里 PAI 模型评估界面

1.3.4 百度 BML

BML 的全称是 Baidu Machine Learning，取名十分朴素，由百度公司开发，是另一个支持拖拽的机器学习平台。但相比 AMLS 和 PAI，BML 目前所支持的算法并不是很多，且 BML 构建实验耗时较长，后台算法可能还需要一定程度的优化。BML 的特点有：操作简单、支持连接百度用户画像等。

图 1.5 百度机器学习 BML 界面

1.3.5 各大云平台的对比及总结

经过对各大机器学习平台的介绍，大致了解了每个平台的特点及优势。本文从易用性、交互性以及算法支持程度等方面对这些平台的特点进行了整理和对比，如表 1.1 所示。总体上来说，微软的 AMLS 和阿里 PAI 平台所支持的功能更加全面，对用户也更加友好。

表 1.1 各大机器学习平台对比

	实验建模方式	界面可交互程度	R/Python 扩展	是否支持 SQL	多算法支持度	模型评估方式
微软机器学习 AMLS	组件拖拽	高	支持	是	高	多样
Amazon Machine Learning	选择和点击	低	无	否	低	基本
阿里云机器学习 PAI	组件拖拽	高	无	是	高	较丰富
百度机器学习 BML	组件拖拽	高	无	否	中	基本

1.4 本章小结

本章指出了分布式机器学习平台的研究背景，逐个分析了国内外各大机器学习云平台的特性，并进行了比较。

第2章 相关技术

2.1 机器学习概述

自从谷歌研发的 AlphaGo 在 2016 年 3 月的人机围棋大战中战胜了世界知名棋手李世乭之后，人工智能的概念越来越火热跟流行。之后 AlphaGo 又升级成 Master、AlphaGo Zero 乃至后来的通用棋类 AI “Alpha Zero”，中间还以 3:0 完胜了围棋世界冠军柯洁，可见人工智能发展十分迅猛。除了人工智能，还有另外两个重要概念：机器学习（Machine Learning）和深度学习（Deep Learning）。AI、ML 和 DL 三者是逐层包含的关系。机器学习是人工智能当前发展最重要的一部分，深度学习的全称是“深度神经网络机器学习”，它属于机器学习，并且现在已经是机器学习领域的明星。

机器学习最早的定义是：它是让计算机不用显式的编程就能进行学习的一门学科[10]，属于计算机科学范畴之一。几十年来，机器学习已经发展称为一门多领域交叉学科，涉及 Calculus, Probability theory, Statistics, Linear algebra, Convex optimization theory，它通过设计算法让计算机可以从数据中寻找规律，自动推导出比编程专家的作品还要好的程序，并将程序应用到未知数据进行预测。形象地讲，机器学习就是把计算机当做“小孩”一样“喂”给它数据，然后这个“小孩”会对这些数据进行“消化吸收”，理解数据中隐藏的知识和规律，最后利用所获得的知识去做一系列任务，比如对图片进行分类、对房价进行预测等等。

对机器学习最为常见的分类是有监督学习（Supervised Learning）和无监督学习（Unsupervised Learning）两大块，区分的依据就是“喂”给机器的数据是否包含因变量，通常用“y”表示，有“y”那么就是有监督，这里的 y 变量规定了任意一条记录的“标签”（label）是什么，比如应该取哪个具体的值或属于哪个类别。有监督学习又包括分类模型和回归模型，区分的依据是这个 y 变量取值是连续的还是离散的。而无监督学习则包括聚类、降维、关联规则、AutoEncoder 等模型。

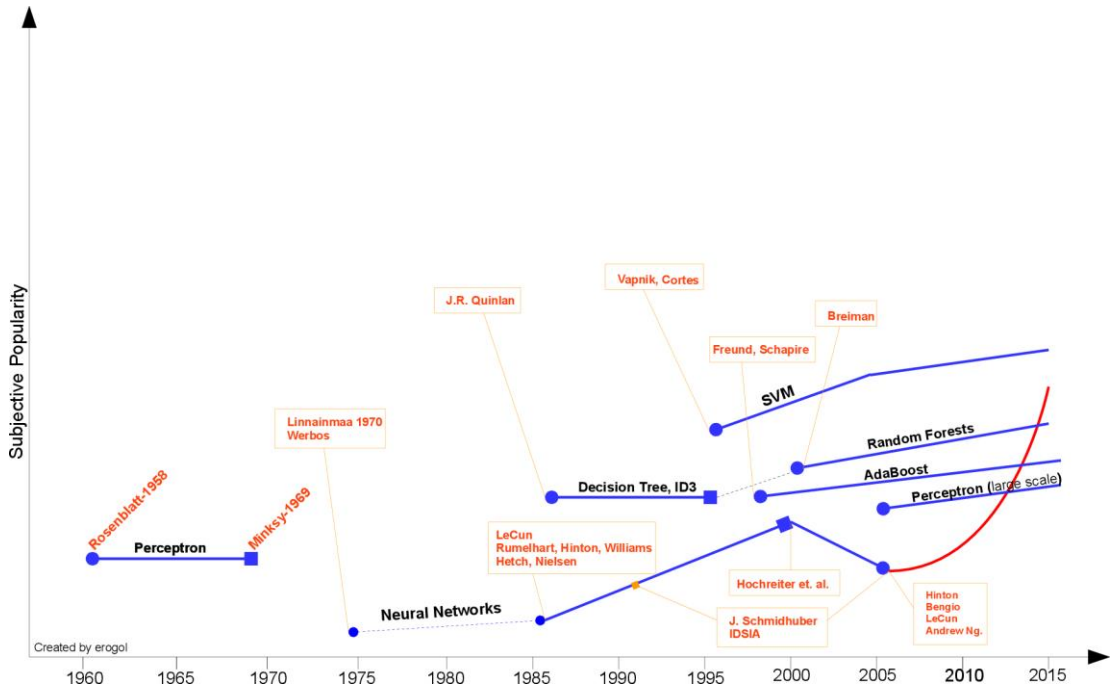

图 2.1 机器学习发展时间线

图 2.1 展示了机器学习中各类算法发展的趋势，横轴为时间线，纵轴为算法流行的程度。其中 Neural Networks 为代表的神经网络模型（也就是深度学习），直到 2005 年后才开始飞速发展，最近几年尤其炙手可热。

2.2 R 语言

2.2.1 R 语言简介

R 语言是一门用于数据科学领域的编程语言，由两位“R 姓”的先生在 1993 年发明，故取名为“R”，它的前辈是贝尔实验室开发的 S 语言，在 R 中大量借用了 S 语言的方法。R 语言自由、免费、开源，现在已经成为统计计算、可视化、数据挖掘和机器学习等领域最为流行的编程语言之一。R 最大的特点是它有上万个扩展包（Packages），每个包都能够实现在某一特定领域和学科中的功能，这些 Package 有些是 R 语言的核心团队开发的，但更多的是由开源社区众多分析师和程序员编写并发布，其中最著名和高产的开发者比如 Hadley Wickham、Dirk Eddelbuettel、谢益辉等人。

R 是一种解释型语言，并提供支持常见的结构，如条件执行（if else）和循环

（for，while，repeat）等。R 还支持通过使用 Vector、Matrix、Array 等数据类型进行广泛的数值计算[11]。在 R 中最重要的数据类型称为数据框（DataFrame）——除了数值计算外，R 支持通过十分丰富的操作对数据框进行结构化数据处理。数据框是表格形式的数据结构，它的每列都是特定类型的向量（例如 character 或 numeric）。DataFrame 为数据过滤和汇总提供了一个简单的语法，再配合 R 包 dplyr 或 data.table 可以进一步简化了复杂数据操作任务的执行，这包括一些常见的关系数据库操作，如选择、投影、聚合、联结等。鉴于 DataFrame 在用户中的普及，它也逐渐被其他语言所采用，比如 Python 的 Pandas 库。

2.2.2 R VS Python

关于 R 和 Python 的对比一直是数据科学领域被人津津乐道的话题。图 2.2 展示了一项调查[12]，是 2017 年世界最知名的数据挖掘和机器学习竞赛网站 Kaggle 对其用户使用的编程语言调查的结果。可以看到，在 7955 个用户中，Python 使用率排名第一（76.3%），R 为第二位（59.2%）。

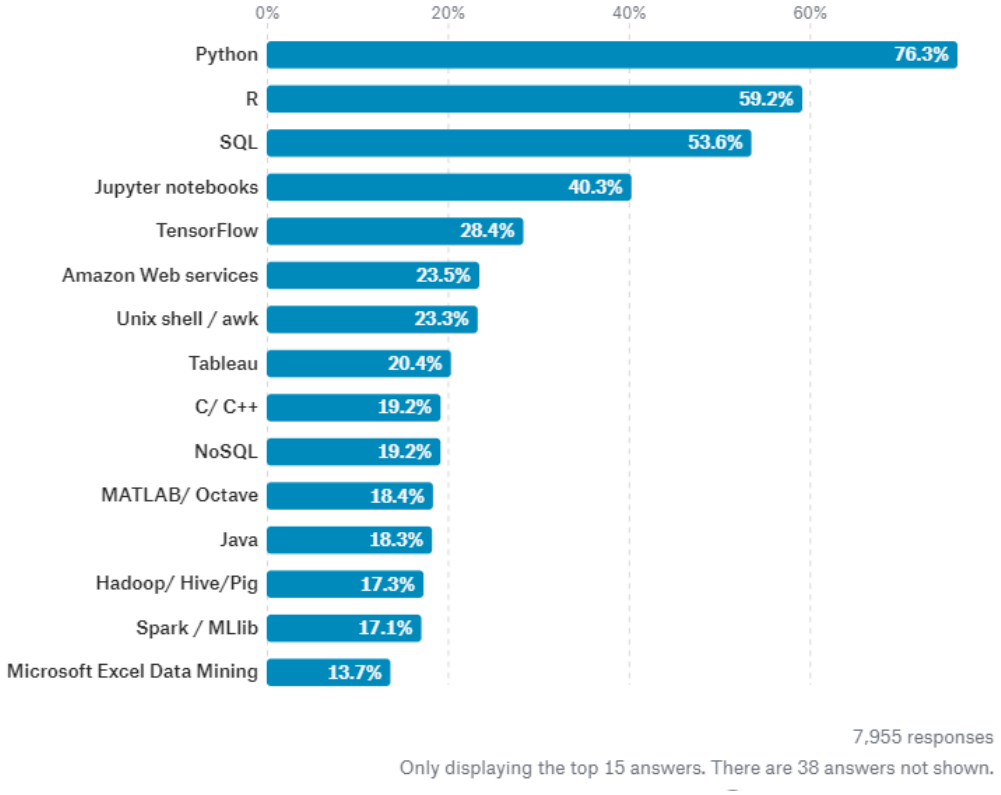

图 2.2 2017 年 Kaggle 用户使用的编程语言排行

图 2.3 则展示了另一项调查结果[13],它是数据挖掘专业网站 KDnugget 在 2015 至 2017 年对数据科学社区的 2900 名受访者关于他们对编程语言使用偏好的调查。可以看到 2015、2016 两年 R 语言还是排行第一，直到 2017 年被 Python 打败掉至第二（Python: 52.6% VS R: 52.1%）。

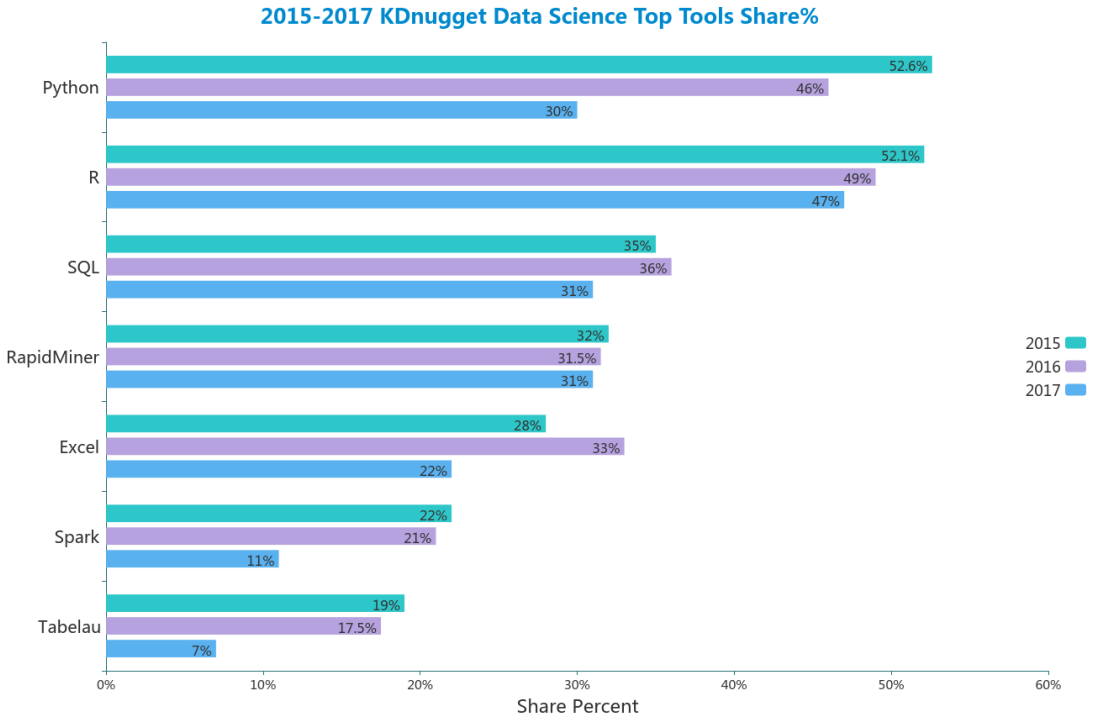

图 2.3 2015-2017 年 KDnugget 对编程语言的使用情况调查

事实上，R 在数据科学中面临的最大竞争对手就是 Python。近年来 Python 崛起，笔者认为这很大程度上得益于谷歌对深度学习框架 Tensorflow 的开发，但随着 Tensorflow 也有了 R 的接口，两者在这一领域又逐渐打成平手。总体而言，R 和 Python 可以说是不分伯仲。最重要的是，不论是 R 还是 Python 的使用者，他们之中很大一部分人会同时使用这两种语言。

2.2.3 R 语言与机器学习

R 在各大领域都有无数热心的贡献者为 R 编写扩展包，当然也包括机器学习。在机器学习领域，每种机器学习算法都会有一个甚至多个 R 包提供了相应的接口或函数。比如经典的决策树算法有 C50、rpart、tree 等，广义线性模型有 LogicReg 、glmnet 等，支持向量机有 e1701、penalizedSVM 等、梯度提升模型有 fastAdaboost、

gbm、bst 等，以及近年来声名鹊起的 XGBoost 也有对应的 R 包 xgboost，至于深度学习领域，R 也有 MXnet、H2O、Tensorflow、kerasR 等众多的包对应到这些深度学习框架。

2.3 分布式机器学习框架

2.3.1 分布式数据仓库 Hive

大数据时代的到来，各种信息呈指数级增长，这对大数据的存储和计算提出了非常高的挑战和要求，传统的关系型数据库难以对这种“4V”特性的大型数据集进行高效的处理。在这种困境下，Apache Hadoop 应时而生。

Hive 正是一个基于 Hadoop 的数据仓库基础设施工具。说起 Hive 的诞生，它最初是由 Facebook 开发的，后来转移到 Apache 软件基金会名下并开源，Apache 之后对其进行了二次开发。Hive 依赖于 Hadoop 分布式文件系统（HDFS），它位于 Hadoop 生态圈的顶部，利用 MapReduce 的分布式计算性能对大规模数据进行操作。Hive 使得工程师通过编写 SQL 代码就能对大型数据集进行统计、分析和处理，使得查询和分析变得十分简单。得益于 Hadoop 的流行，现在 Hive 已经成为许多大数据公司必备的数据库存储系统之一。

需要注意，尽管 Hive 也使用 SQL，但 Hive 不是关系型数据库，它也并非为在线事务处理（OLTP）或实时查询而设计。总而言之，Hive 的特点包括：存储 schema、底层为 HDFS、为 OLAP 多维分析设计、支持 Hive SQL 且查询较快、较为成熟、容易扩展等。

2.3.2 分布式计算框架 Spark

（1）Spark 概述

MapReduce 的并行计算框架为 Hadoop 带来了大量的行业粉丝或使用人群，因为 MapReduce 使得大数据计算方案具有高度的伸缩性及灵活性，极其符合行业需求，并为企业在对大数据的处理上降低了巨大成本。但即便如此，MapReduce 在查询之间的等待时间和运行程序的等待时间方面依然略显不足。

继 MapReduce 之后，又出现一种“闪电般”的集群计算技术，它的名字叫 Spark。Spark 由 Apache 软件基金会推出，之所以开发它，正是为了弥补 MapReduce 在计算进程上效率的不足。Spark 的基本思想也是并行计算，这与 MapReduce 类似，

即 Spark 会将数据切分成片，将片断发送到不同的计算机节点上进行处理，然后将结果发回并处理聚合以得到最终结果。更具体地说，基本的计算范式是：在多个节点上分配一个大的数据集，逐行映射函数，用 Key 分组数据，然后执行集合操作。

需要指出，Spark 并不是 Hadoop 的“修订版”或“增强版”，而且实际上在条件允许的情况下，Spark 并不依赖于 Hadoop，因为它拥有自己的集群管理。Hadoop 只是应用 Spark 的其中一种方式。Spark 与 Hadoop 之间也不是替代关系，充其量说 Spark 是 MapReduce 一种升级，另外笔者认为，Hadoop 生态圈的完善让 Hadoop 平台在大数据领域依然占据着统治地位。Spark 的之所以快速，主要原因是其基于 RDD（Resilient Distributed Datasets）的内存式集群计算，可以提高应用程序的处理和响应速度，官方宣称 Spark 相对于 MapReduce 有 10 到 100 倍的速度提升。

Spark 广受欢迎的另一个原因是它有多种编程语言接口，包括 Scala、Java、Python 以及 R 语言，因此无论平台研发工程师还是数据科学领域的工程师都会去使用它。Spark 有多个组件，功能丰富多样，最重要的是它能通过 Spark MLlib 来支持分布式的机器学习计算。

（2）Spark 的机器学习库 MLlib

由于 Spark 在迭代计算方面的强大优势，因此非常适合大型机器学习应用程序的开发。MLlib 正是由此而生，其全称为 Spark Machine Learning Library。最初的 MLlib 版本是由 11 位贡献者在加州大学伯克利分校开发的，提供了比较有限的一些机器学习方法。自最初发行以来，MLlib 在贡献者方面经历了巨大的增长。不到两年时间，MLlib 拥有来自 50 多个组织的 140 多名贡献者。开源社区的优势促使了 MLlib 开发出更多的功能。

MLlib 同时利用数据并行和模型并行两种手段实现了大规模的分布式机器学习，它涵盖了绝大多数有监督和无监督学习算法的快速和可扩展的实现。MLlib 用 Scala 编写，在每个节点上使用本地（基于 C++的）线性代码库，此外，MLlib 也提供了 Javaa 和 Python 的 API，同时也支持 R 隐式的调用它。MLlib 与 Spark 的紧密集成带来了许多好处。首先，由于 Spark 是用迭代计算来设计的，因此它可以开发大规模机器学习算法的高效实现。Spark 低级组件的改进常常能转化为 MLlib 中的性能优势，而不会直接改变库本身。其次，Spark 的活跃开源社区已经导致了 MLlib 的快速增长和使用。第三，MLlib 是构建在 Spark 上的几个高级库

之一，作为 Spark 丰富生态系统的一部分，MLlib 为开发人员提供了大量工具来简化机器学习流程的开发。

2.4 本章小结

本章介绍了关于分布式机器学习平台的相关技术框架，包括机器学习的基本概念、R 语言以及 Spark 的相关技术。

第3章 机器学习算法分析

算法作为该机器学习平台开发中最为核心的部分，本章将对平台中应用到的各类有监督和无监督机器学习算法进行介绍，并对各类算法的优劣进行分析。对于机器学习中耳熟能详的经典算法（如支持向量机和随机森林），本章将进行简单的概述，而对于近年来愈来愈火热的梯度提升模型（如 GBDT 和 XGBoost），本章还会从原理、数学推导以及伪代码实现方面对算法进行详细说明。

3.1 有监督学习

在有监督学习中，本文列举了支持向量机（SVM）、随机森林（Random Forest）、梯度提升树（GBDT）和极限梯度提升树（XGBoost）模型，不管在数据挖掘竞赛还是工程实践当中，它们都是应用最为广泛、且模型性能都最稳健的模型。其中 SVM 属于单模型，而后三者都是集成模型（Ensemble Model）。Ensemble 意为“融合”，该方法的主要思想类似于“三个臭皮匠，顶一个诸葛亮”的想法，即整合多个较弱学习器对某一问题的预测结果，来得到一个较为准确的最终结果。这种思想对于单个学习器的准确度要求并不高，比如在二分类问题的表现上准确率大于 0.5 即可，但对不同学习器之间的差异性会有一定要求。在 Ensemble 体系中有各种各样的方式，但 Bagging 和 Boosting 最具代表性、使用最为广泛。在本文列举的模型中，Random Forest 采用的是 Bagging 的方式，而 GBDT 和 XGBoost 都是 Boosting 模型。

Bagging 翻译为“装袋”，该方法是基于一种投票或平均的思想。比如分类问题，不同的学习器对不用的样本可能会产生相异的分类结果，那么最终结果的判定按照少数服从多数来决定。即使用多个分类器对所有预测的类别投票，哪个类别票数最高，样本就属于哪个类别。Bagging 方法默认将每个学习器“等而视之”。考虑到由于每个学习器的性能和准确率不同，也可以手动依照学习器的准确率来对它们进行加权后再融合。比如回归问题，那么在最终预测时，可以先对各个学习器按照准确程度分配权重，然后计算加权平均值。

Boosting 翻译为“提升”，它则是在 Bagging 方法的基础上，充分发挥资源调配作用，将训练的重点放在上一轮中模型的预测误差上，即进行一次迭代后，它会将模型预测错误的部分重新分离来训练新的学习器。在 Boosting 的各种算法

中比较经典的叫做 Adaboost。Adaboost 是 Boosting 的一种实现，它在迭代过程中同时考虑了样本权重和学习器的权重[14]。以分类问题为例，Adaboost 按分类对错来对不同的分类器分配不同的权重（weight），错误率越高的模型权重越低；同时设法对每一轮中预测错误的样本增大权重，从而让“错分的样本权重越来越大，使它们更被重视”。也就是说，在每一轮中，训练集中每个样本被抽样的概率都会按照上一轮的分类对错被重新分配，然后在本轮中按照这个概率分布重新抽取 N 个样本出来（可重复抽样），再对这 N 个新的样本训练一轮。这样处理的好处是可以逐渐对错分样本进行“重点关照”，因为越容易被分错的样本，其下次被抽中的概率会逐步增大。这种 Bad Cases 会让模型分辨它们时更加“谨慎”，从而更好的改善模型。这种思想就好比许多学生在备考时会准备一个“错题集”，将每次在模拟考中答错的题记录在错题集中，然后反复的去研究这些错题以加深印象（增大权重），尤其是那些在多次考试中都答错的题（权重变得更加大），这样便会在下一次的考试中取得更好的成绩。

Adaboost 算法在理论和实践当中都被证明表现不错，但 GBDT（或 XGBoost）在对“提升”的理解上更进了一步。GBDT 和 Adaboost 的不同点在于，GBDT 并没有通过前一轮迭代中弱学习器的误差率来更新训练集每个样本的权重，它采取的方式更为直接——将前一轮预测的误差直接拿来作为新的预测目标。GBDT 通过使用了前向分布算法（Forward Stagewise Algorithm）来进行迭代，其核心在于，迭代的方向与损失函数的负梯度方向一致，且弱学习器限定了只能使用 CART 决策树。

3.1.1 支持向量机

支持向量机（Support Vectors Machine，SVM）的取名比较有特点，原因是这个模型的理论推导中用到了被称为“支持向量”的样本点，除了分类问题，SVM 对回归也适用。回归版本的 SVM 又叫做支持向量机回归（SVR）。支持向量机算法早在 50 多年前就被提出，后来经过逐代改进[15]，已经十分成熟。从前文中的“机器学习时间线”图 2.1 中可以看到，在深度学习（2012）成熟之前，SVM 一直是热点，几乎在机器学习领域占据了半壁江山。

标准支持向量机是一个二分类模型，也就是说对于每个给定的输入，支持向量机会判定该样本到底会落入到两个类别中的哪一类。我们知道，训练集中的每一个样本可以映射为高维空间中的一个点，支持向量机便通过寻找一个最为合适的分类面来设法将两类样本点有效的区分开来，让两类样本点尽可能的分别分布

在分类面的两侧。之所以把该分类面称作“支持向量机”，原因是两类样本中与分类面距离最小的点叫做“支持向量”。由于这些点的连线与分类面是平行的，看上去分类面好像被它们“支撑”起来了，故称为支持向量机。从图 3.1 可以看到，两类样本点之间可以有多个分类面（由于图是二维的，所以也叫分割线），那么哪个分类面是最优的呢？支持向量机认为具有最大化分类间隔的分类面最好，原因是这样的分类面具有最好的泛化能力，对新的样本点能够进行更稳健的划分。关于支持向量机的推导本文不详细展开，但其基本目标可以转化为如下带约束的优化问题:

$$\begin{aligned} &\underset{w}{\text{minimize}} \quad \frac{\|w\|^2}{2} \\ &\text{subject to} \quad y_i(\vec{w}\cdot\vec{x}_i+b)\geq 1, \quad i=1,2,...,N \end{aligned} \tag{3.1}$$

式（3.1）中（xi,yi）是样本点，w 和 b 是分类面参数。经过一定推导可知，分类间隔大小与 1/|w|成正比，因此目标函数规定为最小化|w|的平方就等价于最大化分类间隔。在样本点线性可分的情况下，上述优化问题可通过拉格朗日乘子法、KKT 条件和对偶优化理论来求解[16]。

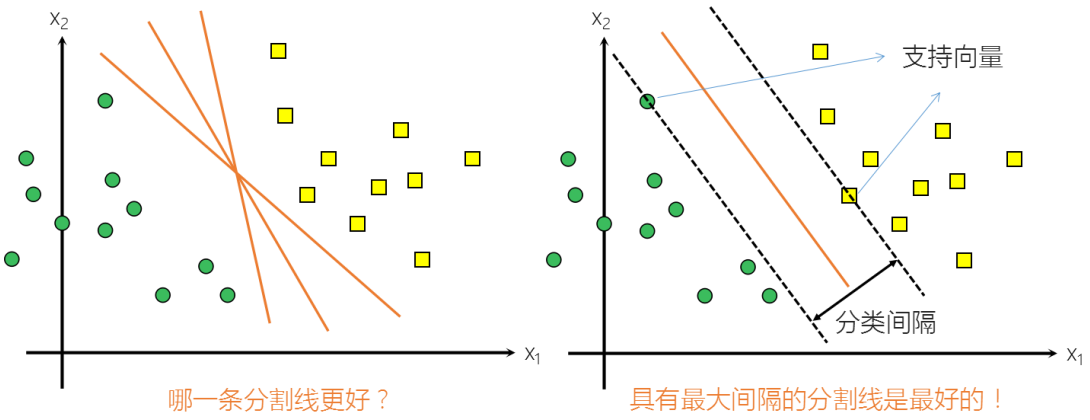

图 3.1 最大化分类间隔的支持向量机

对于无法通过线性准则对训练数据进行划分的情况（包括非线性情形），支持向量机则应用软边缘（Soft Margin）和核技术（Kernel Trick）来解决。软边缘是指允许分类器来一定程度上“越界”，对应到算法中，也就是在受限条件中引入松弛变量（Slack Variable），然后再在目标函数中对松弛变量进行惩罚。通俗点讲就是“允许你预测错，但不能预测的太错”，以增强模型的包容能力和泛化能力，这种思想有点类似于正则化理论。而核技术则是找到一个高维空间，通过

特定的转换（通常是非线性的）将低维样本投影到高维空间，让它们在新的空间域中能够线性可分。核技术的关键点不仅仅在于找到合适的从低维到高维的映射关系，还包括降低样本点在高维空间中两两之间距离计算的复杂度，也就是所谓的避免“维度灾难”。后者要通过核函数（Kernel Function）在低维空间中计算距离去实现。数学理论证明，核函数的充分条件只需满足半正定，故常见的核函数有 Gaussian Kernel（又称作 Radial Basis Kernel）、Polynomial Kernel,、Sigmoid Kernel 以及这些函数的组合等等。

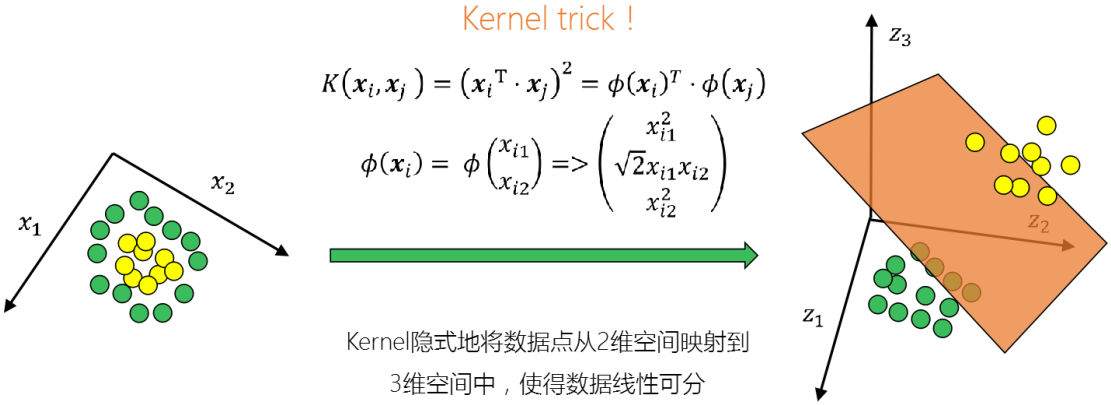

图 3.2 SVM 线性不可分情况下使用核函数

在二分类支持向量机提出后，解决多分类的支持向量机也逐步发展出来[17]。多分类 SVM 的思想容易理解，实际上就是通过某种方法把一个多分类问题转化为多个二分类问题，比如使用一类对一类方法（One Vs One）或一类对其余（One Vs Rest）的方法。“One Vs One” 是指在任意两类样本之间设计一个二分类 SVM，假设训练集中一共有 K 个类别，那么就需要设计 K(K-1)/2 个 SVM。最终的分类是通过最多投票策略来完成的，因为每个 SVM 都会对某个样本的两种“候选”类别选出一类投票，最后可计算出每个样本在每个类别上的总票数，取最多票数的类即可。这种方法的优点是预测的结果相对更准确，但缺点是需要训练的 SVM 数量太多，时间复杂度较高。“One Vs Rest” 则是指每次将分类器将其中一类标记为一类，其余的类别归并为剩下的一类，于是 K 个类别就会训练出 K 个对应的二分类 SVM。然后对于新样本使用这 K 个分类器去预测，最终选择预测概率值最大的那个类别作为分类结果。这种方法的好处是时间复杂度低，缺点是一对多的训练可能会导致正负样本在数量上产生类失衡，而影响模型的准确率。

除了支持向量机分类，还衍生出了支持向量机回归算法，其原理基本与分类

大同小异，主要是在约束问题的受限条件上做了修改:

$$\begin{aligned} &\underset{w}{\text{minimize}} \quad \frac{\|w\|^2}{2} \\ &\text{subject to} \quad \left|\vec{w}\cdot\vec{x}_i + b - y_i\right| \le \varepsilon, \quad i = 1,2,...,N \end{aligned} \tag{3.2}$$

支持向量机作为深度学习之前通用的机器学习方法,其特点是：（1）具有比较完备的理论基础：SVM 使用了包括统计学习、最优化理论、核技术等一整套数学理论。它利用统计学习方法将机器学习中的问题转化为最优化问题，然后尽可能使用时间复杂度最低的算法求解。（2）不需要调节过多的参数，使用起来方便，便于实际工作者掌握。正是由于这两个特点，在成熟的深度学习理论问世之前，SVM 在图像、视频、声音、文本等多个大数据领域大行其道，且关于 SVM 的理论的深入、延伸和扩展一直是机器学习领域的研究热点。

3.1.2 随机森林

随机森林是在 20 多年前被 Tin Kam Ho 提出的算法，当时名为“Random Decision Forests”[18]。之后在 2001 年和 2004 年，Leo Breiman 和 Adele Cutler 分别扩展了该算法，并将“Random Forests”在美国注册成为了他们的商标。

随机森林的思想易于理解，简单点说是决策树算法加上 Bagging 的集成，所谓“森林”就是形象的指很多棵决策树。每棵树的生成依赖的算法是一样的，不同的是所依赖的数据，因为数据通过随机在原始数据中进行有放回的重采样获得，故称之为“随机”。假设一共抽取了 K 次，那么会形成 K 棵决策树。依据 Bagging 的原则，如果是分类，那么新数据的分类结果按这 K 棵分类树基于投票原则输出的类别众数而定，如果是回归，则是将所有决策树的平均值拿来预测。随机森林本质上是对决策树算法进行了改良，因为单棵决策树容易陷入两难困境：如果树的深度或叶子节点树不足则欠拟合；如果无限增大树的深度则会造成过拟合，故单棵决策树的预测能力十分有限。而随机森林集“众人之力”将多棵决策树进行组合，这些决策树两两之间有着一定的差异性，因此预测出的结果更加稳健，且可以通过控制每棵树的深度避免过拟合。

随机森林的随机性不仅仅体现在对样本的随机抽取，还包括在树节点分裂时对特征的随机选择，用以增强不同决策树之间的差异性。单棵树的分类（预测）能力可能很低——当然，至少要超过二分之一，但在随机产生大量的决策树后，往往能起到“众人拾柴火焰高”的效果。

通常认为，随机森林的优点包括:

1) 对于很多数据，它都可以保持较高的准确率。
2) 在输入特征数量很多的时候，依然快速有效。
3) 它可以很好的在不太影响模型准确率的同时处理缺失特征。
4) 它对异常值的鲁棒性很强。
5) 它可以一定程度上对类别不均匀的样本进行平衡。
6) 它可以在模型训练完成后输出特征的重要性评估。
7) 学习过程是比较快速的。

3.1.3 GBDT

（1）算法概述

GBDT（Gradient Boosting Decision Tree），翻译为“梯度提升决策树”，它最早是 1999 年由 Friedman 提出[19]。其别名有很多，比如 GBRT（Gradient Boosting Regression Tree）、GBM（Gradient Boosting Machine）或 MART（Multiple Additive Regression Tree)，。谈及 GBDT 时有时专指 GBDT，有时指 GBDT 家族，即所有基于决策树的提升算法，这还包括下文中的 XGBoost。不同于 Adaboost，GBDT 中的的 Boosting 核心思想是“累加模型”（Additive Model），也就是最终模型是由多个基学习器的结果累加起来的。

正是由于 GBDT 是将累加结果作为最终结论，所以 GBDT 中每棵树的预测响应并不是目标变量本身，而是目标在每一轮的增量或残差。预测残差的好处在于，当算法将所有决策树的结果相加，就会更逼近真实值，因为如果不够逼近，还可以继续增加决策树。举一个直白的例子，假如某个人的月收入是 5000 元，我们要去预测它。最开始我们用 4000 元去估计，发现差了 1000 元（第一轮的损失），然后第二轮我们用 1500 元去拟合差掉的这 1000 元，又发现多预测了 500 块钱（第二轮损失变成-500 元），于是第三轮我们用-600 元拟合剩下的差距，也就是在上一轮的基础上减去 600 元，那么差距就只有 100 元了。如果我们认为只差这 100 元时已经是预测的比较准确了，迭代就可以停止；如果想继续提高精度，则每一轮都会让残差变得更小。直到满足收敛条件，比如相邻两轮之间的残差变化很小时，迭代便会终止。

综上所述，在 GBDT 的迭代中，假设第 t 轮损失为 $L(y,y^{(t)})$，第 t 轮训练的 CART 树为 $f_t(x)$，则每一轮都会以上一轮预测出的残差为基准去训练 $f_t(x)$，让这一轮的 $L(y,y^{(t)}) = L(y,y^{(t-1)}+f_t(x))$ 最小化。也就是说，本轮迭代的目标是找到一棵决策树，

将其加在前面的预测结果之中，使得加上后的损失尽可能地变得更小。

（2）数学推导

记 $y^{(t)}$为第 t 轮迭代的预测值，对每一轮迭代的预测值，Boosting 累加模型基本框架如下:

$$\begin{aligned}
&\hat{y}_i^{(1)} = f_1(x_i)\\
&\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)\\
&\hat{y}_i^{(3)} = f_1(x_i) + f_2(x_i) + f_3(x_i) = \hat{y}_i^{(2)} + f_3(x_i)\\
&\cdots\\
&\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{3.3}$$

其中，xi 为输入的第 i 个样本，$f_t(x)$为第 t 个学习器，也就是第 t 棵决策树。若定义损失函数为 $L(y,y^{(t)})$,则 GBDT 通过最小化 $L(y,y^{(t)})$求解最优模型:

$$\hat{y}^{(t)} \doteq \underset{\hat{y}^{(t)}}{argmin} \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(t)}) \tag{3.4}$$

根据 Boosting 的累加公式和泰勒一阶展开:

$$\begin{aligned}
\sum_{i=1}^{N} L(y_i, \hat{y}_i^{(t)}) &= \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))\\
&\approx \sum_{i=1}^{N} \left[L(y_i, \hat{y}_i^{(t-1)}) + L'(y_i, \hat{y}_i^{(t-1)}) \cdot \Delta\hat{y}_i^{(t-1)} \right]\\
&= \sum_{i=1}^{N} \left[L(y_i, \hat{y}_i^{(t-1)}) + \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \cdot f_t(x_i) \right]
\end{aligned} \tag{3.5}$$

由于模型要求 $L(y,y^{(t)})$随着迭代轮数增加逐渐减小，因此 $L(y,y^{(t)})$需要小于 $L(y,y^{(t-1)})$，而 $f_t(x)$即是我们要训练的第 t 棵决策树，故可取 $f_t(x)$的目标函数为:

$$\hat{f}_t(x) \doteq -\alpha \cdot \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \tag{3.6}$$

其中 α 通常是一个比较小的正数，在模型中称之为步长或学习率。从推导公式中可以看到，GBDT 是在损失函数 $L(y,y^{(t)})$的负梯度方向对新的一棵 CART 决策树进行训练，然后再把训练的结果累加到模型当中。

（3）伪码实现

记决策树的参数为 w，即 $f_t(x)=f_t(x;w)$

输入：样本 (x_i,y_i)，损失函数 $L(y_i,\hat{y}_i^{(t)})$

执行：

1. 初始化： $f_0(x)\leftarrow 0$
2. *For t = 1 to T do:*

2.1 计算目标响应： $\hat{f}_t=-\dfrac{\partial L(y_i,\hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$

2.2 学习第 t 棵树的参数： $w_t^* \doteq \underset{w}{argmin}\sum_{i=1}^{N}(\hat{f}_t - f_t(x;w))^2$

2.3 寻找最优步长（可选）： $\alpha_t^* \doteq \underset{\alpha}{argmin}\sum_{i=1}^{N} L(y_i,\hat{y}_i^{(t-1)}+\alpha\cdot f_t(x_i;w^*))$

2.4 生成第 t 棵树并更新模型：

$$f_t(x)\doteq \alpha^*\cdot f_t(x;w^*)$$

$$\hat{y}^{(t)}=\hat{y}^{(t\text{-}1)}+f_t(x)$$

输出： $\hat{y}^{(T)}$

3.1.4 XGBoost

（1）算法概述

XGBoost 的全称是 eXtreme Gradient Boosting，意为“极限梯度提升”，是最近几年被华盛顿大学（University of Washington）的计算机系博士生陈天奇开发的算法，后来被陈作为了深度机器学习社区（DMLC）的一个项目。从取名可以看出 XGBoost 也是基于梯度的 Boosting 模型，因此它也是 Gradient Boosting Machine 家族的一员，而命名为“极限”是因为它的确对 GBDT 从速度和性能上都做了改进。原生版本的 XGBoost 是由 C++ 编写的，用户利用 Java、Python、Julia 和 R 等编程语言调用相应接口也能完整地使用它。

笔者认为，同样是梯度提升，但相比于 GBDT，XGBoost 最大的不同点在于对泰勒展开公式在损失函数上的近似不同，前者为一阶近似，后者则是二阶。众所周知，在最优化理论中有两个最为常见的算法，分别是梯度下降法（Gradient Descent）和牛顿迭代法（Newton-Raphson Method）。如果将 GBDT 称作“梯度

下降版的提升树模型”，那么 XGBoost 则可称为“牛顿迭代法的提升树模型”。XGBoost 的第二个特点是它的损失函数里面已经包含了正则化项，包括决策树的叶子节点数量和各个叶节点的权重。加入正则化之后，模型会对决策树的复杂程度进行惩罚，防止了过拟合[20]。此外，XGBoost 另一个引以为傲的特点是它能做并行计算，比如能利用 CPU 的多个核，在最近版本的 XGBoost 中，甚至支持了“直方图节点分裂算法”的 GPU 并行[21]，因而 XGBoost 相比 GBDT 或 RF 速度相当之快。

自从 XGBoost 诞生以来，在各大数据挖掘竞赛上都有了它的身影。最先让它受到关注的是 Kaggle Higgs Boson Machine Learning 竞赛，XGBoost 在千余队伍中拔得头筹[22]。随着它在 Kaggle 社区中逐渐声名鹊起，在诸如 Kaggle、阿里云天池、京东金融、DataCastle 等知名机器学习竞赛中，各个队伍借助 XGBoost 往往能在积分榜上独占鳌头。

（2）数学推导

XGBoost 同 GBDT 一样使用了 Boosting 累加框架，因此其第 t 轮的预测值同样为:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{3.7}$$

这里的 $f_t(x)$同样定义为 CART 决策树:

$$f_t(x) \doteq w_{q(x)} \tag{3.8}$$

其中 q(x)表示样本 x 所属的叶子节点，w 是叶子节点的分数（leaf score），一棵决策树可以定义为各个叶节点的编号以及对应的分数，所以 $w_{q(x)}$表示决策树对样本的预测值。使用泰勒二阶展开公式对 XGBoost 第 t 轮的损失函数 $L^{(t)}$进行近似可得:

$$\begin{aligned}
\tilde{L}^{(t)} &= \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \\
&\approx \sum_{i=1}^{N} \left[L(y_i, \hat{y}_i^{(t-1)}) + L'(y_i, \hat{y}_i^{(t-1)}) \bullet \Delta\hat{y}_i^{(t-1)} + \frac{1}{2!} \cdot L''(y_i, \hat{y}_i^{(t-1)}) \bullet (\Delta\hat{y}_i^{(t-1)})^2 \right] + \Omega(f_t) \\
&= \sum_{i=1}^{N} \left[L(y_i, \hat{y}_i^{(t-1)}) + g_i \cdot f_t(x_i) + \frac{1}{2} h_i \cdot f_t^2(x_i) \right] + \Omega(f_t)
\end{aligned} \tag{3.9}$$

该公式中，g_i 和 h_i 分别是 $L^{(t-1)}$的一阶梯度和二阶梯度:

$$g_i = \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}},\quad h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}_i^{(t-1)})^2} \tag{3.10}$$

同 GBDT，由于在上一轮迭代完成后，$L^{(t-1)}$已经计算出来为常量，故最小化损失函数可去掉常数项。再假设惩罚项Ω取叶子节点数目 T（防止树过大增长）和 L2 正则，并将 $f_t(x)$写为树的形式及叶子节点形式（定义 I_j 为叶子节点 j 涵盖的所有样本），则损失可写为:

$$\begin{aligned}
\tilde{L}^{(t)} &= \sum_{i=1}^{N}\left[g_i \cdot f_t(x_i) + \frac{1}{2} h_i \cdot f_t^2(x_i) \right] + \Omega(f_t) \\
&= \sum_{i=1}^{N}\left[g_i \cdot w_{q(x_i)} + \frac{1}{2} h_i \cdot w_{q(x_i)}^2 \right] + \gamma T + \lambda \cdot \frac{1}{2}\sum_{j=1}^{T} w_j^2 \quad \text{(Tree Pattern)} \\
&= \sum_{j=1}^{T}\left[(\sum_{i\in I_j} g_i)\cdot w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)\cdot w_j^2 \right] + \gamma T \quad \text{(Leaf Pattern)} \\
&= \sum_{j=1}^{T}\left[G_j \cdot w_j + \frac{1}{2}(H_j + \lambda)\cdot w_j^2 \right] + \gamma T
\end{aligned} \tag{3.11}$$

上式中由于是 L2 正则故 λ 与 H 并列，如果是 L1 正则化则 λ 会与 G 并列。上式可以看作是关于 w_j 的一元二次函数，故要求 $L^{(t)}$最小，可取 w_j 为:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{3.12}$$

代入到 $L^{(t)}$中，可得:

$$\tilde{L}^{(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \tag{3.13}$$

式（3.13）表示了在训练第 t 棵决策树时的损失。由于第 t 棵树的训练是从根节点逐渐进行节点分裂而来，每次进行节点分裂时，由于叶节点数目和叶节点的样本都发生了改变，因此 G、H 和 T 会变化。考虑到节点的分裂始终要朝着 $L^{(t)}$最小化的方向进行，因此 XGBoost 取节点分裂前后 $L^{(t)}$减少的量作为节点是否分裂的指标:

$$\begin{aligned}
Gain &= \tilde{L}_{before-split}^{(t)} - \tilde{L}_{after-split}^{(t)} \\
&= \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma
\end{aligned} \tag{3.14}$$

由于 γ 是人为设定的常数，因此 Gain 值取决于于中括号内的项。每次在增加叶节点时，Gain 的值越大，$L^{(t)}$减小得越多。所以当对一个叶节点分裂时，XGBoost

会计算所有候选特征对应的 Gain，然后选取 Gain 最大的进行分裂即可。

（3）伪码实现

下述伪代码展示的是 XGBoost 训练第 t 棵树寻找最优节点时的贪心算法（Exact Greedy Algorithm）。尽管该算法最易理解和实现，但事实上 XGBoost 在进行节点分裂时采取了多种方式进行优化加速，比如分位数分裂、稀疏值处理、并行化等等。

输入：当前节点的样本集合 I，m 个特征

执行：

1. 初始化：$Gain \leftarrow 0$
   $G \leftarrow \sum_{i \in I} g_i,\ H \leftarrow \sum_{i \in I} h_i$

2. 外层循环遍历 m 个特征：

   For k = 1 to m do:

   2.1 左节点初始化：$G \leftarrow 0,\ H \leftarrow 0$

   2.2 内层循环寻找该特征最佳分割点：

   For j in sorted(I, by x_k):

   $G_L = G_L + g_j,\ H_L = H_L + h_j$

   $G_R = G - G_L,\ H_R = H - H_L$

   $score = \max(score,\ \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda})$

   循环结束记录 *max(score)* 及 j 的值

   循环结束记录 k 的值

输出：*max(score)* 下的最佳分割特征和分割点

3.2 无监督学习

本节主要对机器学习平台中使用到的无监督学习算法进行分析，由于当前该机器学习平台主要的应用场景是频繁项集和关联规则问题，因此本节介绍的相关

算法是 Apriori 和 Eclat。事实上，无监督学习还包括聚类、降维、AutoEncoder 等。

不论是 Apriori 还是 Eclat 算法，它们的主要应用场景都是“购物篮分析问题”，其最著名的一个例子是“啤酒与尿布”的典故。这是发生在美国某知名连锁超市的真实案例，故事是这样的：一家超市的老板无意中发现，当他把啤酒和尿布这两种商品放到同一片区的货架上时，这两种本来完全无关的商品的销量竟然都相比原来单独摆放大幅增加了。原来，发生这种有趣的现象的原因在于：受当时社会风气的影响，美国的妇女更多是全职太太，往往会整天在家照看 Children，这让他们不太有时间去逛超市。于是购买尿布这样的“重担”便肩负在了男人身上，而啤酒又是男人经常逛超市的必买物品之一，因此他们经常会把这两种商品同时放到车篮之中。啤酒和尿布作为“共现”商品组合的发现，为超市带来了不少的销售收入。类似这种寻找不同物品之间共现联系的问题称为购物篮分析，又叫做关联分析问题。

3.2.1 Apriori

Apriori 算法最早由 Agrawal 和 Srikant 两位博士在 1994 年提出[23]。它是最为经典的关联规则挖掘算法。关联规则的挖掘依赖于频繁项集的寻找，所谓频繁项集，是指经常在一条购物记录中一起出现的商品的集合，这里的“经常性”一般使用“出现次数/总交易次数”来度量，术语叫做支持度（Support）。由于频繁项集若通过层层遍历、暴力搜索（Brute-force）的方式去寻找，会产生大量不必要的计算开销而导致低效率。Apriori 算法就是为解决寻找频繁项集而带来的复杂度问题而生。之所以取名为 Apriori，是由于其核心思想是先验准则——如果某个项集是频繁的，那么由于子集支持度必然不小于原集支持度，故该项集所有子集肯定也是频繁的。当然，实践当中这个准则是反过来使用的[24]。它的等价逆否命题是：如果某项集不频繁，那么所有包含它的项集都不可能是频繁的。这样在知道了一个项集是非频繁的之后，它所对应的超集就可以直接舍去，这在极大程度减少了需要遍历的项集数目，时间和空间复杂度大为降低，可以更加高效的计算频繁项集。

在利用 Apriori 算法时，只需输入购物篮数据集和最低支持度阈值（0 到 1 之间的值）即可，Apriori 会逐轮增加项集中项的数量来做遍历寻找。它的算法步骤如下：

1） 在数据库中进行初次遍历，生成候选 1 项集和频繁 1 项集[25]。

2） 从 2 项集开始递归，由频繁 m-1 项集生成频繁 m 项集。

a） 生成候选：通过两两合并 m-1 项生成候选 m 项集。

b） 过滤：对所有候选 m 项集进行遍历，对每个候选项观察其所有 m-1 项的子集，但凡发现有某个子集不频繁，舍去该候选。

c） 计算支持度：对过滤后的候选 m 项集，逐一计算支持度，舍去支持度低于阈值的候选，剩下的即为频繁 m 项集。

3） 直到当前 m 项集中数目只有 1 个时递归结束。

3.2.2 Eclat

Eclat 全称是“Equivalence Class Transformation”，它是 Zaki 等人于 1997 年提出来的算法[26]。其特点在于利用的是商品（item）对购物篮（basket）而非购物篮对商品的“倒排”的方式进行频繁项集挖掘，正如 Eclat 其名，这种倒排的方式是对原始数据库的一种等价转换。如果购物篮对商品称为“水平”的数据库，那么这种商品对购物篮的倒排数据结构则是一种“垂直”的数据库。

和 Apriori 算法一致，Eclat 算法在寻找频繁项集时也是从少到多逐步增加，但得益于这种垂直数据库，Eclat 算法更加简洁直观。由于商品（item）在哪些购物篮（basket）中出现过这一事实已知，它只需要对频繁的商品逐步的进行取并集的操作并删除不满足支持度要求的组合，就可以把所有频繁项集找到。笔者在实践中发现，Eclat 算法比 Apriori 计算更快，因此本文的机器学习平台的研发使用的便是 Eclat。

3.3 本章小结

本章对该机器学习平台将要使用到的各种有监督和无监督的机器学习算法进行了研究，涉及算法概述、原理推导以及伪代码实现等方面。

第4章 需求分析与架构设计

4.1 需求分析

4.1.1 项目背景与问题提出

（1）某公司的 SuperEye 智能运营系统

本文的机器学习平台的研发以所在公司的 SuperEye 智能运营系统为背景。SuperEye 企业智能运营系统是针对企业的所研发的智能运营系统，帮助企业管理与利用数据资产，实现企业全员参与决策的工具。SuperEye 的目标用户为企业用户，致力于帮助企业实现从传统商业智能（BI）到运营智能（OI）的转变，解决传统企业不会用、不能用数据的痛点，为企业提供数据管理工具，进行内部数据的整合与分析，支持快速的可定制化的智能运营系统开发。SuperEye 支持秒级数据响应、页面支持个性化定制、即时的需求响应、下钻式的交互分析等功能，由下至上分为数据层（Data Hub）、工具层（Studio）和应用层（APP），其整体架构如图 4.1 所示。其中机器学习平台属于 SuperEye Studio 层，它承担着支撑对企业数据的智能分析及预测的功能。由于机器学习平台是 SuperEye 系统的重要组成部分之一，因此该机器学习平台被命名为 SEML，为 SuperEye Machine Learning 的简称。

（2）某证券公司两融业务的客户分析需求

本文所述两融业务是指浙江省杭州市某证券公司的融资融券部门的业务。该融资融券业务部门的需求是希望通过一套智能运营解决方案，协助两融部门的管理人员实时监控公司的两融业务的运营数据、管理两融客户。具体的功能需求包括客户检索、客户群管理、客户画像、客户群对比、单客群分析等。其中，针对客户群对比和单客群分析的功能，该证券公司希望通过使用机器学习的手段能自动对客户数据进行分析，而减少人工操作、降低人工成本。

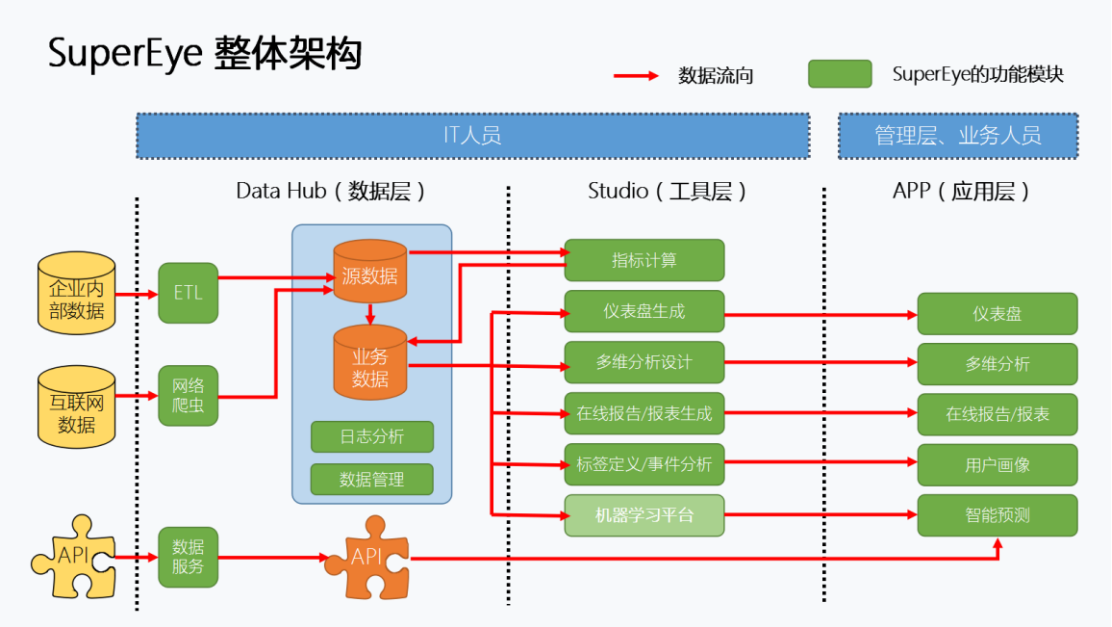

图 4.1 某公司 SuperEye 系统的整体架构

4.1.2 机器学习平台 SEML 的研发目标

不管是从完善 SuperEye 的系统架构和产品功能方面考虑，还是从解决某证券公司的业务需求来考虑，开发一个机器学习平台，进而帮助企业实现运营智能（OI）都是必要的。此外，前文已经介绍，为了让数据分析人员和机器学习工程师能够快速的针对特定机器学习任务进行模型开发，很多大公司也都开发了自己的云机器学习平台，其目的是为了加速整个创新过程，提高工作效率，比如微软的 AMLS，亚马逊的 Amazon Machine Learning 以及国内的如阿里 PAI 和百度 BML。这些平台大都具有处理超大规模数据的能力和分布式的存储能力，同时整个流程支持超大规模的机器学习建模。

本文认为，开发一个基于分布式数据存储的机器学习平台（SEML）应有如下作用。第一，SEML 需要对 SuperEye 智能运营系统中的工具层和应用层进行有力的补充；第二，SEML 可以帮助工程师快速入门和理解机器学习，并通过界面上简单的选择和点击操作，轻易地完成数据挖掘的整套流程；第三，机器学习平台应涵盖从原始数据到上层应用的一条龙服务，利用该平台的标准化模块和针对客户方的定制化模块，用户可以快速迭代即时响应客户方的需求。

基于上述需求，SEML 需要实现的目标是：

a. 依据人机交互的原则，开发出一个初级机器学习工程师也能够便捷使用，交互体验较好的机器学习平台，该平台可以完整实现机器学习的基本流程，并提供分布式数据库的数据源接口。
b. 提供常用的多种机器学习算法接口，拥有进一步开发的潜力。
c. 结合某证券公司的具体业务，有针对性的开发出定制化的功能。

4.2 机器学习的一般流程

机器学习平台的设计应当遵循机器学习的一般流程，因为在机器学习领域，数据的流向通常是有一定的方向性的，尤其是有监督学习问题。尽管由于机器学习算法的原理不同、适用场景不同，机器学习的流程会有细微差异，但通常来说，一次完整的机器学习过程应当包括五个步骤：数据源获取、数据预处理、特征工程、模型训练和评估、预测推断。如图 4.2 所示。

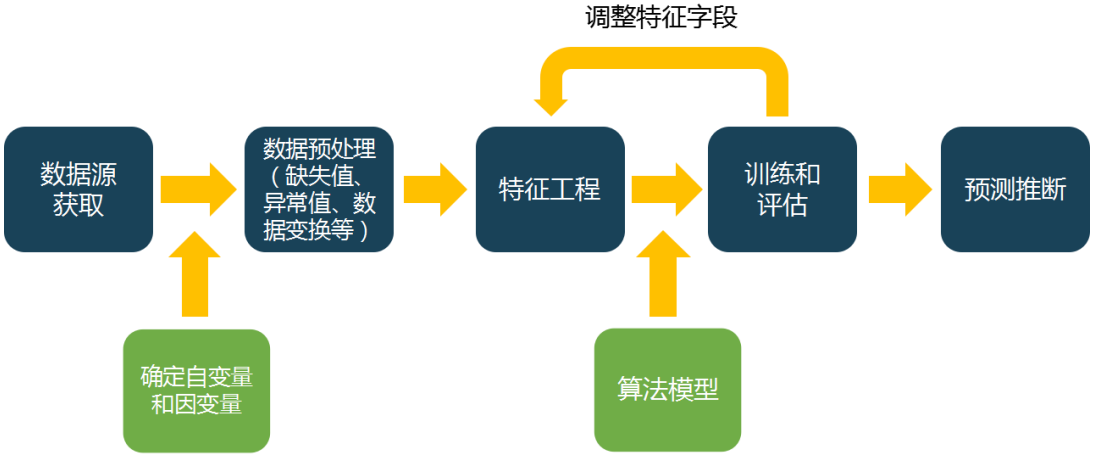

图 4.2 机器学习的一般流程

（1）数据源获取

数据源获取是指通过各种方式和渠道对于相关的原始数据进行收集和整理。在案例学习或许多机器学习竞赛当中，原始数据都是直接提供并准备好的，但是实际问题需要自己获得原始数据，比如通过访问数据库获得合适的数据源。在获取原始数据后，如果是有监督学习问题，需要人工定义好自变量（解释变量）和因变量（响应变量），并结合因变量的特性来判断该问题属于分类问题（Classification）还是回归问题（Regression）。

“Data determines the upper limit of the machine learning results, and the algorithm is just as close to the limit as possible”，在机器学习中，数据是根本，其

重要性就像“米”对“巧妇”一样。数据只有在隐含一定的分布特征的条件下，才能让算法挖掘出其中蕴含的价值。例如分类问题，数据在不同类别（由因变量决定）的样本数量上不能有太大的差别，否则不仅会让模型难以训练，在模型评估上也存在很多问题，也就是通常所说的要尽可能避免“类失衡”。此外，越是复杂、参数越是多样、效果相对越好的机器学习算法，它们对数据总量和维度数量会有一定的下限要求，而机器的性能和存储又决定了数据的总量不能超过一定上限，所以在数据总量、维度数量上要进行取舍权衡。

（2）数据预处理

原始数据就像一堆原材料。数据预处理是指选择合适的方法对这些“原材料”进行清洗、加工和修正，来去掉数据中的异常和“脏数据”、将数据变换为适合分析的形式等，例如缺失值插补、剔除异常值、对数变换、归一化、降维转换等。一般来说，原始数据多少会包含一些噪声和冗余[27]。而模型的效果又与数据的质量息息相关，这个问题对于机器学习算法构成了极大的挑战。因而在正式调用机器学习算法之前，选择合适的方法对原始数据进行预处理是必要的步骤。

（3）特征工程

特征工程可以说是机器学习中最关键的一环。特征工程手段多种多样，又分为特征构建、特征提取、特征筛选等工作。在有监督学习当中，分类和回归的本质都是要让算法去辨别出不同的数据，而这是十分依赖于数据的特征的。通常情况下，特征越多越好，但特征数量过多会导致计算效率降低还可能导致过拟合。因此，工程师要设法设计出有区分度的特征，这就需要工程师能够在面对特定问题时，能够对问题背后的业务逻辑有着比较深的理解。好的特征往往能够凸显数据的本质，充分发挥算法效力。在机器学习竞赛当中，排名靠前的选手一般都是因为在特征工程方面下了很大的功夫。

（4）训练和评估

模型训练是指选择适当的机器学习算法去拟合数据的分布特征，最终在自变量 X 和因变量 y 之间建立起一个良好的映射关系。模型评估则是指通过某种预先设计好的度量指标来衡量不同模型的优良程度。

模型训练中一个比较重要的手段就是参数调优（Parameter Tuning）。参数调优可以是暴力式的，比如 Grid Search，但暴力式的调参其时间复杂度过高，一种

更好的方式是基于对算法的理解去调参，然后基于暴力搜索进行微调，而这就需要工程师对理解各种算法的原理、清楚算法中每个超参数的意义。模型评估就是要设计可靠的指标，来评价机器学习模型到底多大程度上拟合了数据中 y 跟 X 之间的关系。欠拟合（Under-fitting）是指模型解释力不足，没有很好反映这种 y-X 的关系，而过拟合（Over-fitting）则是指虽然这个模型很好的解释了训练数据，但碰到新的数据时，泛化能力不足，在指标上的表现反而更低了。通常情况下，在做了基本的预处理和特征工程，并使用了一个比较流行的机器学习算法进行训练后，数据不会过于欠拟合，工程师遇到的更多问题可能会是过拟合现象，避免过拟合可以使用正则化方法来降低模型复杂度，或者可能要重新去设计特征。因此“特征工程-训练-评估”会是一个反复尝试的过程。模型评估最为常用的方法是交叉验证(Cross Validation，CV)，因为 CV 使用的是真实数据，这样更加有说服力。

（5）预测推断

预测推断是机器学习的最后一步。当我们成功训练并部署好模型，就可以让模型对新的未标注数据进行预测，这个过程又称为“推断”（Inference）。推断过程和训练过程是相对应的，训练过程本质上是利用训练数据寻找到 y 和 X 之间的映射关系 f，而推断过程就是当有新的 X_new 进入到模型当中时，利用已建立好的 f，便可以推测 X_new 对应的 y_new 取值。当然，这里的 X_new 必须要跟 X 一样经过了完全相同的数据处理及特征工程工作。

4.3 机器学习平台 SEML 架构设计

机器学习平台的总体架构应该大致和机器学习的一般流程一致，理论上讲，一个完善的机器学习平台应该涵盖了流程的五个步骤。但考虑到以下几个方面：第一，完善的机器学习平台的研发需要耗费庞大的时间成本和人力资源，尤其是数据预处理和特征工程的方法繁多且杂乱，难以形成通用流程并系统化；第二，由于特征工程主要需要的是深刻的业务理解并且十分依赖工程师的创造力，即便平台涵盖了特征工程几乎所有的方法，在面对新的机器学习问题时，工程师仍需要将这些方法；第三，本文所研发的机器学习平台其目的在于降低初级工程师花在机器学习建模上的成本，而非完全地替代机器学习工程师的工作。基于上述几点，本文所开发的机器学习平台，在难以规范化和系统化的数据处理及特征工程

的模块上进行了取舍。

由于机器学习算法种类的多样性和复杂性，以及具体业务场景中对于不同算法的需求和使用频率的不同，本文依据算法的特性以及业务需求将 SEML 设计成标准化模块和定制化模块。

（1）基于有监督学习的标准化模块

SEML 的标准化模块是基于在真实业务场景中最为常见、也最容易被模型化的有监督学习（Supervised Learning）中的问题，它依赖的机器学习算法包括 SVM、Random Forest、GBDT 以及 XGBoost。标准化模块依据机器学习的一般步骤而设计，在本文中又细分为三大功能：数据导入模块、模型训练模块、预测推断模块。

（2）基于业务场景的定制化模块

SEML 的定制化模块则主要是基于特定业务需求而设计，通常它会用到一些无监督学习（Unsupervised Learning）算法，比如聚类、降维、推荐或关联规则算法，在本文中，基于某证券公司的业务需求（如寻找到一批客户它们所具备的共同的特点），所定义的是频繁项集和关联规则挖掘问题，其所依赖的算法包括 Apriori 和 Eclat 等。在该模块中，又分为客户差异性特征识别与客户共性特征发现两个部分。

（3）模块独立性

需要指出，SEML 的标准化模块和定制化模块之间基本是相互独立的，它们在数据使用、操作逻辑和上是相对封闭的一个完整子集，其目的是为了在设计上做到数据交互的无关性，这样在更换定制化模块时，标准化模块系统不会受到任何影响。

如图 4.3 所示，这种两块式的设计好处在于可以让机器学习平台根据不同需求进行变化和更新，使其具备更好的灵活性和可扩展性。

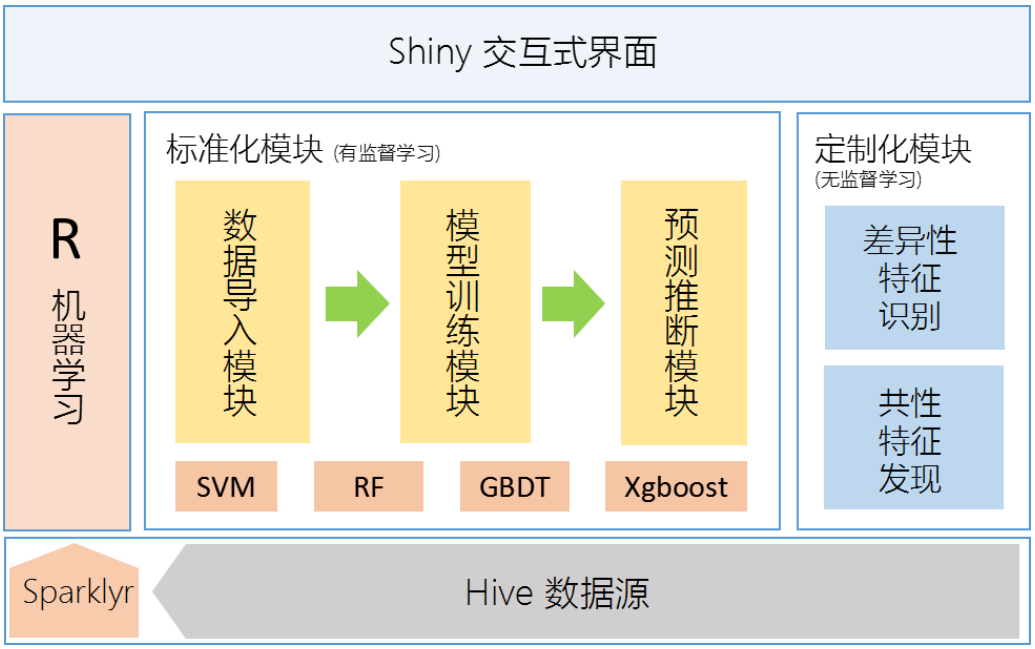

图 4.3 分布式机器学习平台 SEML 的整体架构

4.4 本章小结

本章主要对实际项目需求进行了分析，并依照机器学习的一般流程对本文的分布式机器学习平台的整体架构进行了设计。

第5章 平台设计与实现

5.1 SEML 的基本开发框架

5.1.1 SEML 的算法集成框架

由于 SEML 平台将会用到各种各样的机器学习算法，而这些算法本身彼此之间并无关联，作为一个整体的平台，SEML 需要设法将它们集成起来。

前文已经提到，R 中很多的机器学习包（库），每个机器学习包涵盖了不同的算法，有不同的函数和功能。为了让用户能够更高效的完成机器学习任务，省去在使用不同的算法做机器学习时大量记忆 API 或函数的功夫，R 的开发者设计了“CARET”。CARET 的全称是“Classification and REgression Training”，包如其名，它是专为解决分类和回归问题（有监督学习）而生、试图简化创建机器学习模型过程的工具包。虽然 CARET 也做机器学习，但与众不同的是，CARET 本身“并不生产水，它只是水的搬运工”。CARET 并不开发原生的机器学习算法，而是将大量独立的机器学习包集成起来，提供了让 R 用户能够方便的综合应用各种算法的功能。CARET 主要有以下功能[28]：数据切分、数据预处理、递归特征筛选、模型调参和重采样、特征重要性估计等。

SEML 平台的整体算法框架正是基于 CARET 来实现。通过 CARET 中各种机器学习算法的接口，SEML 能够在代码中很便捷的的调用这些算法，并在各种函数的编写上极大的利用了代码复用的功能。

5.1.2 SEML 的 Web 开发框架

由于笔者对于 SEML 的开发不仅仅涉及数据库和后端处理逻辑，也包括前端的界面交互和展示，因此笔者将要在 SEML 的研发中应用到的 R 的 Web 开发技术。在 R 的 Web 开发框架中，比较有代表性的是 Shiny 和 Fiery，前者专为解决界面交互式的问题而设计，比较偏应用；后者主要围绕的服务器生命周期而设计，相对偏底层。笔者在比较了两种框架的编程逻辑后之后，选择了 Shiny 作为 SEML 的 Web 开发框架。

同万千 R 包一样，Shiny 也是一个开源的 R 包。如同它的名字一样，Shiny 优

雅并且强大[29]。它为 R 提供了一种轻量级的 Web 开发框架，可以帮助 R 用户来快速构建交互式的 Web 应用程序，而无需利用太多 HTML、CSS 和 JavaScript 的知识，可以说它极大解放了作为编程语言的 R 的生产力。一个 Shiny 的 App 主要由两个 R 脚本构成：用户界面脚本（ui.R）和后台服务脚本（server.R）。它的基本流程如图 5.1 所示，ui.R 控制 Web 应用的界面布局和外观（还可以内嵌 CSS 代码）并接收用户在界面上的输入，然后将用户输入传输到后端 server.R，之后 server.R 控制数据在后台的处理逻辑，对数据进行一系列的转换，最后再传输到 ui.R 通过一定的方式进行展现。

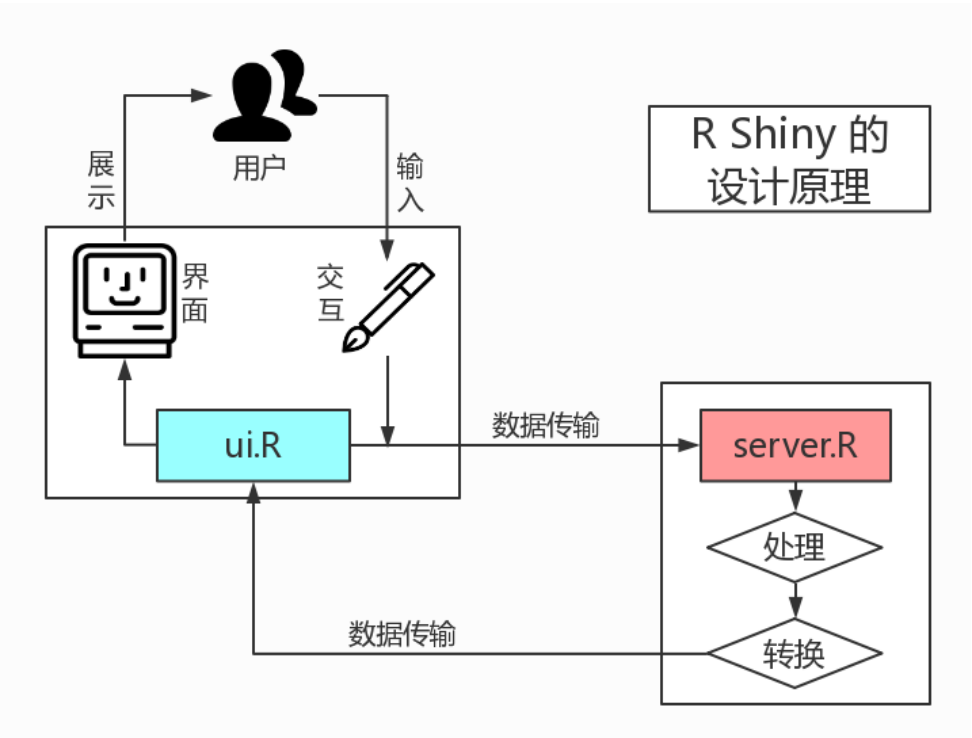

图 5.1 R Shiny 的设计原理

5.1.3 SEML 的分布式数据接口

SEML 作为一个分布式的机器学习平台，首先要解决的问题是如何获取分布式的数据源，即如何将 Hive 的库表导入到 SEML 的平台当中。在 R 中通过 JDBC 的方式连接到 Hive 的方式有三种，分别是 SparkR、Sparklyr 和 RHive。考虑到 RHive 是通过 MapReduce 的模式来执行计算任务，而 SparkR、Sparklyr 可以让 R 利用 Spark 基于 RDD 去做计算，数据通信更加高效，故 SEML 在分布式数据接

口上选择了它们。依据 Hadoop 及 Spark 在不同环境下的部署模式，后两者可任选其一。

（1）SparkR

SparkR 是 Apache 开发的 Spark 官方 R 语言接口。使用 SparkR 十分简单，只需要配置好相应环境变量，并在 R 中写入 “library(SparkR)” 即可，SparkR 对 R 来说就像一个 Package 一样。在 Spark 2.2.0 中，SparkR 还提供了叫做 Spark DataFrame 的数据类型供 R 分析及处理，这类似一个大型的、分布式存储的 R 数据框，语法也与 R 的 DataFrame 类似，支持诸如 select、filter、aggregate 等常见操作。最新版本的 SparkR 还支持 R 去调用 MLlib。

SparkR 的架构原理如图 5.2，同 Spark 的基本框架一样，R 进程分布在两个主要组件 Driver 和 Worker 上。在 Driver 上实现了 R 到 JVM 桥接通信，Worker 则允许 R 程序将作业提交给 Spark 集群并进行分发，在每个执行节点（Spark Executor）上运行 job。

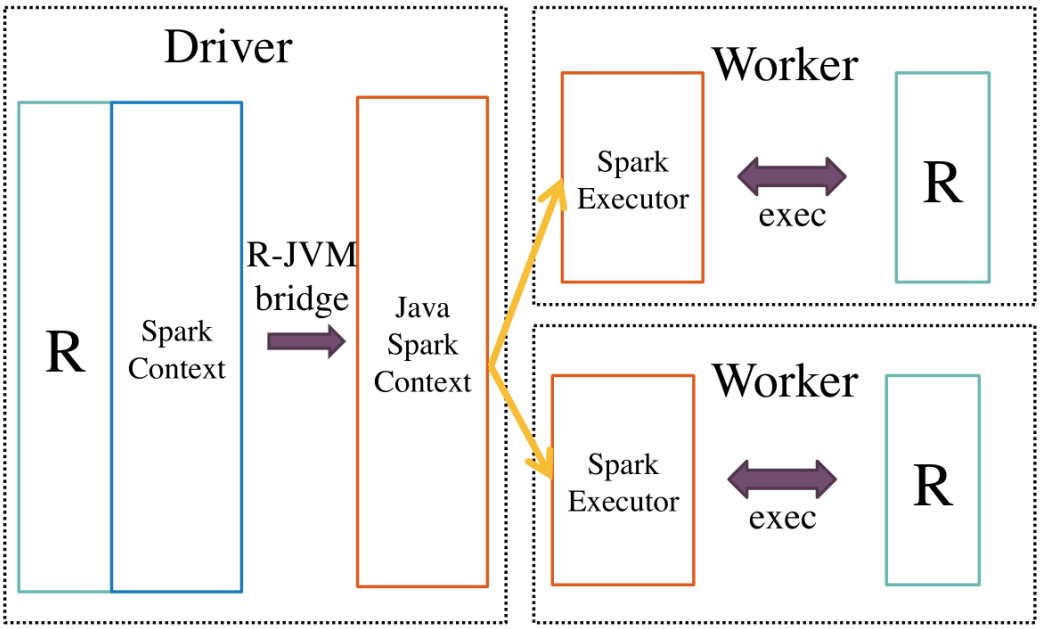

图 5.2 SparkR 架构原理

（2）Sparklyr

Sparklyr 是 Apache Spark 的另一个 R 接口，由 RStudio 公司开发。同 SparkR

一样，Sparklyr 也是一个 R 包，之所以后缀名为“lyr”，原因是 R 中有一个著名的对 DataFrame 分析和处理的包叫“dplyr”。而 Sparklyr 的语法几乎与 dplyr 完全一致——这也降低了 R 用户使用 Sparklyr 的门坎，唯一不同的是 Sparklyr 处理的是 Spark 版的 DataFrame。Sparklyr 支持连接本地和远程的 Spark 群集，并为 MLlib 也提供了算法接口[30]。由于 dplyr 强大的数据处理和分析能力，Sparklyr 也继承了这些本领，帮助 R 在大型数据集上进行复杂的分布式计算提供了多种有效的手段。

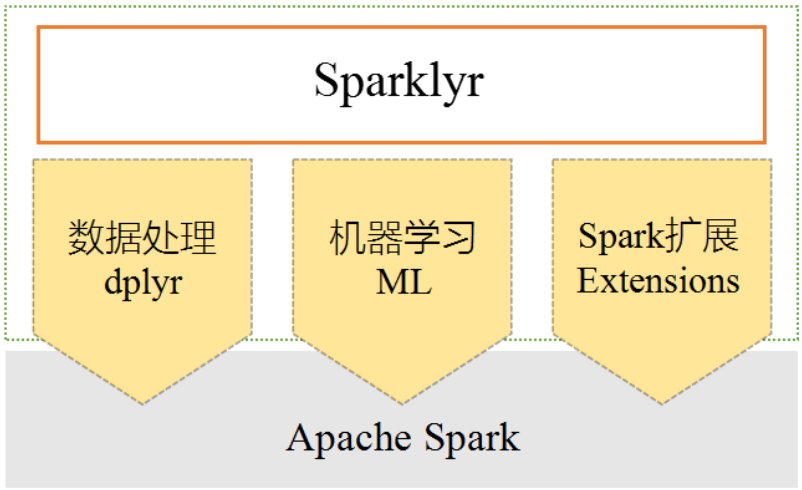

图 5.3 Sparklyr 的功能

5.2 SEML 数据导入模块

SEML 是基于 R 语言开发，而底层数据源在 Hive 中，Hive 是基于 Hadoop 的分布式文件系统的数据仓库，R 是无法直接访问到 HDFS 底层文件的。因此数据导入模块作为该平台的第一个功能，它需要实现让 R 语言通过一定的 Interface(接口)能够访问到 Hive 中的数据。由于 Hadoop 的原生语言是 Java，故 R 访问 Hive 通常是通过 JDBC(Java DataBase Connectivity)的方式进行连接，前文提到，这种途径有三个，分别是 RHive、SparkR 和 Sparklyr。一方面，从数据读取效率的角度，后两者通过 spark 的方式进行数据交互，在海量数据的场景之下反应会更加迅速。另一方面，对比后两者，SparkR 是 Apache Spark 提供的官方 R 接口，Sparklyr 则是 RStudio 公司开发的接口，两者都能提供比较成熟的有关数据处理和机器学习方面的函数或方法，并通过 Spark 进行高效率的分布式数据计算，故两者任取

其一皆可。但需要注意，如果 Spark 采取的是 YARN 的模式进行部署而非 Standalone 模式（独立模式）或 High Alailability(高可用)模式，SparkR 暂时是不能支持的。因此，由于本文开发的机器学习平台也是在 YARN 环境下，因此选择了 Sparklyr 作为 Spark 和 R 之间的接口。

5.2.1 数据源获取

数据源获取功能是让用户能够访问到所有 Hive 的库表结构。并对任一存在的物理表进行数据查看。从设计的角度，用户选择数据库和选择数据表的操作应该满足先后的顺序，因为用户事前并不知道 Hive 中有哪些数据库，以及每个数据库中存在哪些数据表。所以，界面上应该存在两个单选框：第一个单选框让用户能够选择任一个已存在的数据库，当用户完成第一个选择操作，比如选择数据库 A 之后，Sparklyr 会在后台访问库 A 的元数据信息并立即提供给前端，并在第二个单选框中同时刷新 A 中的所有物理表，供用户选择。

此外，用户可能希望在选择一张 Hive 表之后，能够观察该表的数据或样例数据（Sample Data），因此界面上应该有一个按钮提供对数据的查看。当用户点击该按钮时，会以行列表格的形式展现出这些数据真实是什么样子。

5.2.2 特征选择与转换

前面已提到，有监督学习的任务通常是寻找自变量 X(x1,x2,x3...,xn)和因变量 y 之间的映射关系，如分类和回归。“变量”或“字段”（Variables or Columns），在机器学习中又称为“特征”（Features）。在平台界面上，当用户完成数据源获取的操作后，需要确定哪一个变量是 y，并对 X 中包含的变量进行选择。对前端而言，用户要完成选择 y 的操作，需要有一个单选框；要完成选择 X 的操作，则需要一个多选框。

此外，对某一特征字段，用户可能希望在选择该字段后观察到其分布，故平台需要对该字段计算其分位数并绘制分布图进行可视化。考虑到某一特征的分布很可能不是正态分布，比如是严重的左偏或者右偏，平台需要提供对数转换（Log Transformation）的功能。

5.2.3 模块逻辑设计

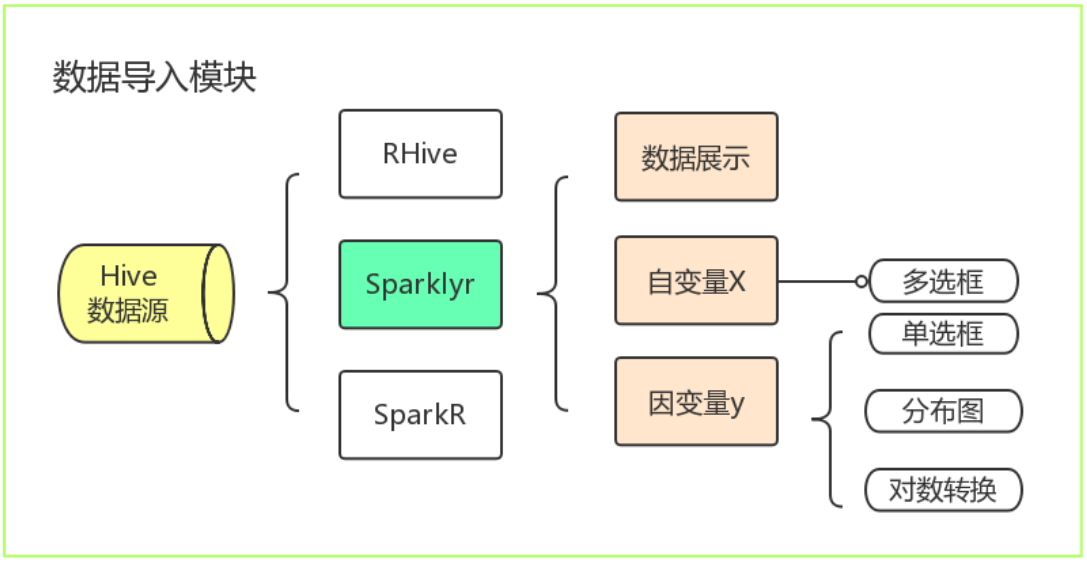

图 5.4 数据导入模块逻辑设计

5.2.4 关键代码实现

```
...
## Global
# 加载依赖包
library(Sparklyr)
library(rJava)
library(DBI)
# 通过Sparklyr访问Hive数据库
sc <- spark_connect(master = "yarn-client", app_name = "machine_learning",
                    version = "1.6.0", hadoop_version = "2.6.0",
                    spark_home = "/opt/cloudera/parcels/CDH-5.10.0-1.cdh5.
# 查询Hive数据库
databases <- dbGetQuery(sc, "show databases")$result

## UI端
# 数据查看
actionButton('btn_viewData',label = 'View Data',icon=icon('table'))
bsModal('data',title = 'Dataset',trigger = 'btn_viewData',
        size = 'large', DT::dataTableOutput('rawdata')
)
# 特征观察与对数变换
conditionalPanel(condition = "output.YtypePrint.substr(6) == 'numeric' |
                 output.YtypePrint.substr(6) == 'integer'",
                 checkboxInput('chk_logY',label = 'log transform'))
plotOutput('Yplot', height=260)
tableOutput('Ystats')
...

## Server端
# 数据表选择框即时更新
observe({
  updatePickerInput(session = session, inputId = "HiveTable", choices = ta
})
# 因变量yvar的选择框即时更新
observe({
  tables <- dbGetQuery(sc, paste0("show tables in ", input$HiveDatabase))
  updateSelectizeInput(session, 'yvar', choices=names(rawdata()),
                       selected = tail(names(rawdata()),1))
})
...
```

图 5.5 数据导入模块关键代码

5.2.5 模块界面展示

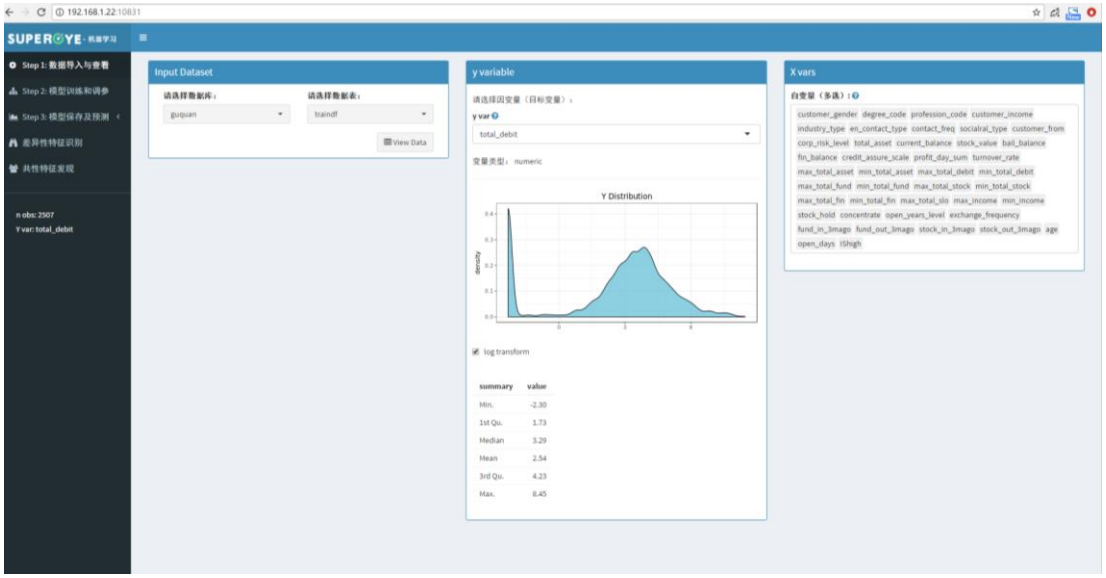

图 5.6 数据导入模块界面 1

说明：图 5.6 左边的第一个框是对 Hive 数据库和数据表进行的选择。第二个框对 y 变量进行了选择（选完可以更换），并展示了对数转换之后的 y 的分布情况及其四分位数。第三个框是用户对自变量 X(x1,x2,x3...,xn)的选择。

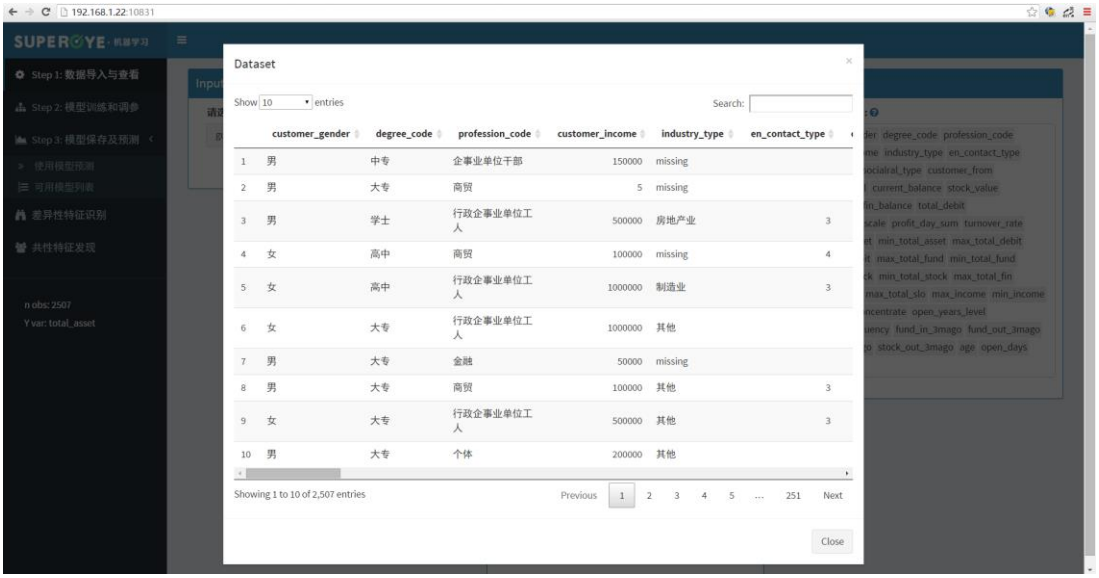

图 5.7 数据导入模块界面 2

说明：图 5.7 展示了用户在选择完 Hive 库表后，后台会通过 Sparklyr 获取数据并传输到前端。当用户点击“View Data”按钮，平台就会以表格形式对数据进行展示。

5.3 SEML 模型训练模块

在完成数据导入、特征选择之后，需要进行机器学习算法建模，该过程又称为“训练”（Training）。“训练”是机器学习的术语，它指的是模型对数据的拟合过程，即模型会随着迭代次数逐渐增加一步步修改自身的参数直至收敛，以“适应”数据的某种分布特征，最终在自变量 X 和因变量 y 之间建立起一个良好的映射关系。而喂给模型训练的数据称之为“训练数据”或“训练集”。前文的数据导入模块中获取的数据即是训练数据。

机器学习算法有简单和复杂之分，对于简单的算法，例如 Logistic Regression 等线性模型，模型训练的最终结果是一个 y 和 X 之间的多项式，其结果是可视的；但对于复杂的算法，例如 Random Forest 等集成树模型、基于最大化分类间隔的 SVM、或者是神经网络算法等，它们通常是一个“黑箱模型”，用户感知到的仅仅是“黑箱”的输入和输出，而无法接触到算法的内在逻辑，“黑箱”通常也是不易解释的。本文认为，SEML 的使用者并非专业的机器学习工程师或算法建模师，且平台设计的目的正是为了降低用户理解及使用机器学习的门坎，因此 SEML 对用户而言应该是一个“黑箱”，用户无需知道平台的底层细节，只需要能够运用一定的算法对数据进行建模即可。当然，为了评估模型的有效性，并对不同的算法或模型进行对比，平台需要进行一定的可视化结果，来让用户真正感觉到模型的效果和价值。

5.3.1 算法选择和参数调优

建模的第一步，是选择合适的机器学习算法。由于用户事前并不知道哪个算法对训练数据将会拟合的更好，因此在界面上需要有一个多选框，提供平台目前支持的针对该特定问题（例如分类或回归）的可用算法，它们包括 svmLinear、Random Forest、Gradient Boosting、XGBoost Linear 和 XGBoost Tree。用户可以挑选想用的算法进行训练，如若不进行挑选，平台会默认选择所有的算法。

机器学习算法复杂多样，且每一个算法都有不同的多种超参数选择，例如

Random Forest 需要指定 tree 的数量、单棵 tree 所使用的特征数量和深度等，SVM 要设定惩罚项的系数，XGBoost 还要设定学习率、损失函数和评价函数等等。XGBoost 专门有一本篇幅很长的参数指南[31]。由于参数空间的数量随着参数的增多呈几何倍增长，如果要遍历各种算法的各种参数，然后选择最优算法及参数，实践中几乎不可能实现，再者用户可能并不会真正理解每种参数的含义。鉴于此，出于简单实用和计算效率的考虑，SEML 采取的策略是对每种算法只调节对模型性能影响较大的那些参数而固定其他相对不重要的参数，如 XGBoost Tree 只调节 tree 的数量、深度、特征采样比例这三个参数，并对每个可调参数进行离散化，如 tree 的数量仅在 10、50、100 之间选择，深度仅在 3、4、5 之间进行选择等。然后基于 Grid Search 对离散化后的参数空间进行遍历搜索，选择相对较优的参数。

5.3.2 基于交叉验证的模型选择

交叉验证(Cross Validation)简称 CV，它是指在模型训练时留出一部分的真实数据不去扔给模型，而把它们用作验证模型是否准确。这样的好处就是利用了真实数据来评价模型性能使评价结果更为可靠——改善模型的交叉验证得分往往就能提高模型的泛化能力。常见的交叉验证方法包括 Hold-Out Method、K 折交叉验证以及留一交叉验证[32]，其中 K 折交叉验证（K-fold CV）对数据的使用率最高，也是当前应用最为广泛的方法。

SEML 选择的正是 K 折交叉验证，出于简单易用考虑，平台界面上提供了 3 折、5 折或 10 折供用户选择。当用户导入了训练数据、选择了想用的算法和交叉验证的折数，就可以点击“Train Models”按钮执行建模过程。

5.3.3 模型评估及统计量

在多个模型训练之后，需要对不同算法产生的模型或同一算法由于超参数不同产生的模型依据一定的评价指标进行评估和排序。对于回归问题，该平台使用的是决定性系数（R-square）和均方误差（RMSE），而对于分类问题，则使用的是准确率（Accuracy）和 Kappa 统计量作为指标。这些评价指标的含义如下。

R-square 又叫“决定性系数”，指的是回归平方和对总误差平方和的比值，它被解释为在回归线附近的点占所有样本点的比例，能反映回归线对数据的拟合程度，其取值范围在 0 到 1 之间。R-square 的计算公式如下。可以看到，当 R-square

趋近于 1，此时分子趋近于 0，也就是 y 和 y-hat 十分接近，说明回归方程拟合的越好；当 R-square 趋近于 0，此时分子比分母等于 1，意味着模型效果等价于直接拿数据的均值去测，说明回归方程拟合的较差。

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{5.1}$$

RMSE 是残差平方和的算数平方根，简称为均方根误差，其值越趋近于 0 表明回归模型拟合的越好。RMSE 计算公式为:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2} \tag{5.2}$$

Accuracy 是分类问题中最常见也是最直接的指标，翻译为“准确率”或“正确率”，表示在所有的样本中，被模型正确分类的样本的占比，也就是混淆矩阵（Confusion Matrix）中主对角线单元格中所有值的总和所占的比值。Accuracy 的计算公公式如下，其中 TN、FN、TP、FP 都是混淆矩阵四个格子中的值:

$$Accuracy = \frac{TP + TN}{TN + FN + TP + FP} \tag{5.3}$$

Kappa 统计量相对而言比较难以理解，它是用于计算分类精度和评价一致性的指标[33]。Kappa 值的公式如下:

$$Kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e} \tag{5.4}$$

其中，Po 为准确率，即上面的 Accuracy，反映了实际一致率；Pe 为正确预测的各分类数值的期望，表明理论一致率。Kappa 设计出来的意义在于剔除随机因素 Pe 造成的预测结果和真实值一致的影响。举个例子，假设 100 条记录中有 40 个 1（正样本）和 60 个 0（负样本），某个模型预测的结果为 50 个 1 和 50 个 0，则 Pe 为 0.4*0.5+0.6*0.5=0.5，也就是随机地分别在真实值和预测值中各挑出一个进行比较，两者类别一致的概率。Kappa 的取值范围是正负 1 之间。通常情况下认为:

Kappa = 1: 说明两组结果完全一致;

Kappa = -1: 说明两组结果完全不一致;

Kappa = 0: 说明两组结果是偶然造成的;

Kappa < 0: 说明两组结果一致性不如偶然状况，还不如随机预测;

Kappa > 0: 说明模型有意义，且值越大模型越有价值;

Kappa ⩾ 0.75: 说明预测结果与真实结果已经相当的一致;

Kappa < 0.4: 说明两组结果虽一致，但一致性不够好。

最后，为了让用户感知到模型的效果，直观的观测到最优模型，平台需要对模型对比的结果以图表的形式可视化的展现出来。平台一共设计了三种可视化的形式。第一是以表格形式展现基于交叉验证的模型评价和统计量信息，以分类问题为例，这些信息包括：模型效果排名、模型名称（包含参数信息）、Accuracy、Kappa、Accuracy 方差、Kappa 方差和模型所属种类，如图 5.8 所示。其中 Accuracy 和 Kappa 存在方差是因为同一模型在 K 折交叉验证下实际上被训练了 K 次，因此所有模型都会存在 K 个 Accuracy 和 Kappa 值。第二是用误差棒图对表格信息做了可视化，依照排名对各个模型从上到下依次对 Accuracy 和 Kappa 绘制了误差棒。第三是利用颜色深浅不同的混淆矩阵对五类算法中各自的最优模型进行展示。

Show 10 entries

rank	modelName	Accuracy	Kappa	AccuracySD	KappaSD	Model
1	xgbLine-10-0.01-0-1	0.9067	0.7130	0.0207	0.0681	xgbLine
2	xgbLine-10-0.01-1-0	0.9059	0.7115	0.0197	0.0622	xgbLine
3	xgbLine-10-0.01-1-1	0.9047	0.7099	0.0193	0.0617	xgbLine
4	gbm-0.01-5-10-200	0.9047	0.7021	0.0119	0.0318	gbm
5	xgbLine-5-0.01-0-1	0.9043	0.7077	0.0186	0.0563	xgbLine
6	xgbLine-10-0.01-0-0	0.9043	0.7062	0.0216	0.0676	xgbLine
7	xgbLine-5-0.01-0-0	0.9035	0.7057	0.0186	0.0601	xgbLine
8	xgbLine-5-0.01-1-0	0.9015	0.7009	0.0175	0.0550	xgbLine
9	xgbLine-5-0.01-1-1	0.9007	0.7002	0.0181	0.0561	xgbLine
10	xgbTree-0.01-4-0-1-1-1-5	0.8999	0.6940	0.0035	0.0102	xgbTree

Showing 1 to 10 of 25 entries Previous 1 2 3 Next

图 5.8 模型评估统计量展示

5.3.4 模块逻辑设计

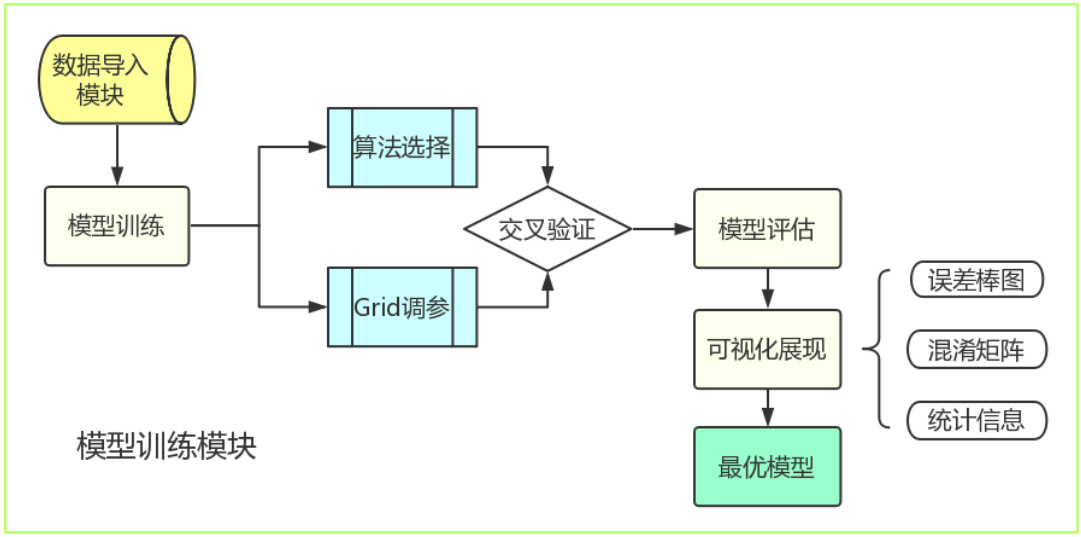

图 5.9 模型训练模块逻辑设计

5.3.5 关键代码实现

```
...
## Global
# 参数空间
tuneParams <- list(
  'svmLinear'=data.frame(C=c(0.01,0.1,1)),
  'rf' = expand.grid(mtry = c(3,4,5)),
  'gbm' = expand.grid(n.trees = c(50,100,200),
                      n.minobsinnode = 10,
                      interaction.depth = c(3,4,5),
                      shrinkage = c(0.01,0.1)),
  'xgbLine' = expand.grid(nrounds=c(5,10), eta=c(0.01,0.1),
                          lambda=c(0,1), alpha=c(0,1)),
  'xgbTree' = expand.grid(nrounds=c(10,50,100), max_depth=c(3,4,5),
                          eta=c(0.01,0.1), gamma=0,
                          colsample_bytree=c(0.5,1),
                          min_child_weight=1, subsample=1)
)

## UI端
# 算法多选框
selectInput('slt_algo',label = '机器学习算法：'%>%label.help('lbl_algo'),
            choices = reg.mdls, selected = reg.mdls, multiple=T)
# 交叉验证选择器
radioButtons('rdo_CVtype',label = '交叉验证：'%>%label.help('lbl_CV'),
             choices = c('3-fold'=3,'5-fold'=5,'10-fold'=10),inline = T)

## Server端
# 模型信息生成函数
getRes <- function(i){
  Model <- names(fits)[i]
  res <- fits[[i]]$results
  df <- res[(ncol(res)-3):ncol(res)]
  apply(res,1,function(r) paste(r[1:(ncol(res)-4)],collapse = '-')) %>%
    paste(Model,.,sep='-') -> modelName
  cbind.data.frame(modelName,df,Model=Model[[1]],stringsAsFactors =F)
}
# 循环调用生成展示信息
df <- plyr::ldply(1:length(fits), getRes)
...
```

图 5.10 模型训练模块关键代码

5.3.6 模块界面展示

图 5.11 模型训练模块界面 1

说明：图 5.11 展示了当用户在模型控制框中选择完算法、调参策略以及交叉验证折数，点击“Train Models”后的结果。左下的模型总结展示了最优模型及其评价指标结果，以及总共训练的模型数量。右边的可视化图展示了所有模型在交叉验证的情况下的 Accuracy 和 Kappa 的排名情况，分越高模型排在越上面。

图 5.12 模型训练模块界面 2

说明：在图 5.12 中，左边的可视化图展现了五种算法中各自的最优模型的预测值（对训练数据自身进行预测）与真实值的对比。以混淆矩阵的形式展现。每个混淆矩阵都有四个格子，格子颜色越深表示其中的值越大。主对角线上的两个格子表示预测准确了，副对角线表示预测错误。

5.4 SEML 预测推断模块

依照机器学习的一般流程，在模型完成训练之后，需要使用已训练好的模型对新的无标注数据进行预测，这些新的数据称之为“测试数据”或“测试集”，而这个预测的过程又称为推断（inference）。对于分类问题，模型往往会对测试集中的每条记录（row）都预测出其在各个类别下的概率分布，概率值总和为 1，且通常取该记录的预测类别为概率值最大的那个类。对于回归问题，模型则会对每条记录（row）给出一个预测的连续值。

5.4.1 测试数据导入

与数据导入模块一致，测试数据同样要通过 Sparklyr 从 Hive 中获取，由于测试数据和训练数据理论上应该位于同一 Hive 数据库中，因此用户只需选择对应的数据表即可。

5.4.2 模型保存与选择

前文已经提到，用户可能会选择多个算法，每个算法中都有一种最优的参数选择，因此可能会产生多个模型。因此，从界面设计的角度，在模型选择框中应当记录每个算法的最优参数，用户可以自行选择任一模型（即算法和参数的组合），如若不选，平台会默认选择在所有算法和参数中的最优模型。

此外，平台应当支持模型保存和加载的功能。一方面，用户在训练好一个模型之后，可能会希望将该模型保存至服务器上，以便将来随时调用。另一方面，用户在加载一个已经存在的模型之时，必须详细了解该模型的信息，包括模型名称、模型所属算法的类别、模型创建的时间、模型的应用场景（如二分类、多分类还是回归）、模型的准确率、模型使用的数据（自变量 X 包含哪些特征、因变量 y 是哪个字段）等。因此平台必须要有一个“可用模型列表”以表格形式来展现模型的信息，用户在详细了解某个模型信息之后，才可以有针对性的调用它。

5.4.3 预测结果展示

用户在导入了测试数据并选择了某个模型进行预测之后，预测的结果（y_pred）应该要呈现出来，同样需要呈现的还包括测试数据的 X 中的各个字段，如果测试集是用来验证模型是否真的准确，那么还要呈现“测试集”中真实的 y 与预测值进行对比。因此界面上应该有个表格，用来展示测试集及其预测情况。从界面设计的角度，如果用户希望将预测的结果复制到剪切板或导出成 Excel/CSV，平台应该提供相应的按钮。

5.4.4 模块逻辑设计

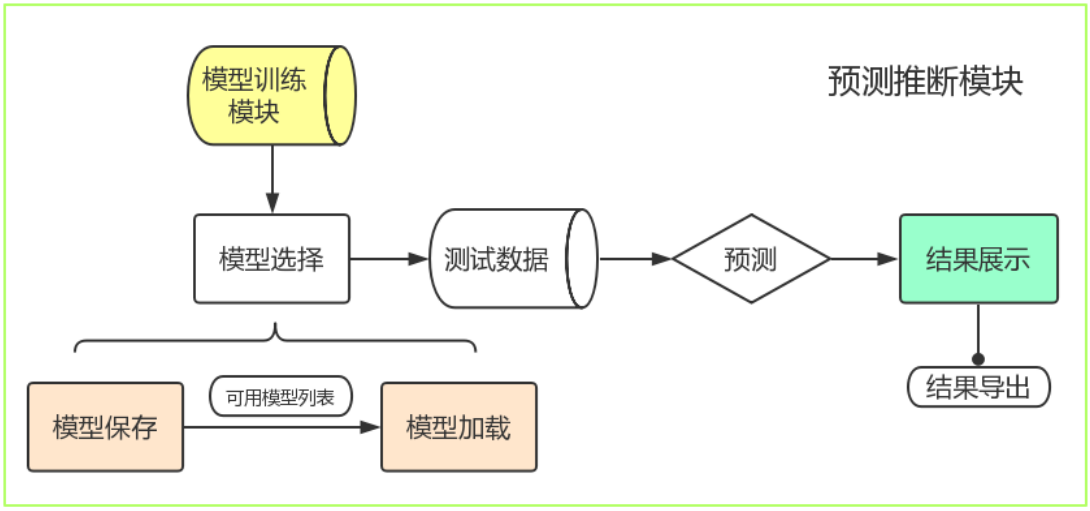

图 5.13 预测推断模块逻辑设计

5.4.5 关键代码实现

```
...
## UI端
# 模型保存按钮
shinySaveButton('modelSave', '保存当前模型', '保存当前选择：',
                filetype=list(data='Rdata'), buttonType='success')
# 模型加载按钮
shinyFilesButton('modelSelect', '试试其他模型？',
                 '加载其他模型后当前模型将失效',
                 multiple=FALSE, buttonType='info')
...

## Server端
observe({
    # 模型保存
    currentModels <- rbind(currentModels, thisModel)
    write.csv(currentModels, './currentModels.csv', row.names = F)
    sendSweetAlert(
      messageId = "modelOut", title = "模型保存成功！", type = "success"
    )
    ...
    # 模型加载
    Finalalgor <- readRDS(path2)
    sendSweetAlert(messageId = "modelIn", title = "模型加载成功！",
                   text = "当前模型已被替换", type = "warning")
    Finalalgor <- CVtune[[input$slt_Finalalgo]]
    topList = topModels()
    algorithm = names(topList)[topList==input$slt_Finalalgo]
    ...
    # 预测推断
    preds = factor(preds)
    levels(preds) = levels(factor(rawdata()[[input$yvar]]))
    testRes[['Preds']][noMiss] = as.character(preds)
    probs = predict.train(Finalalgo, dataTest(), type = 'prob')
    probs = round(probs, 4)
    names(probs) = paste0('Prob_', levels(preds))
    for(pCol in names(probs)){
      testRes[[pCol]] = NA
      testRes[[pCol]][noMiss] = probs[[pCol]]
    }
})
...
```

图 5.14 预测推断模块关键代码

5.4.6 模块界面展示

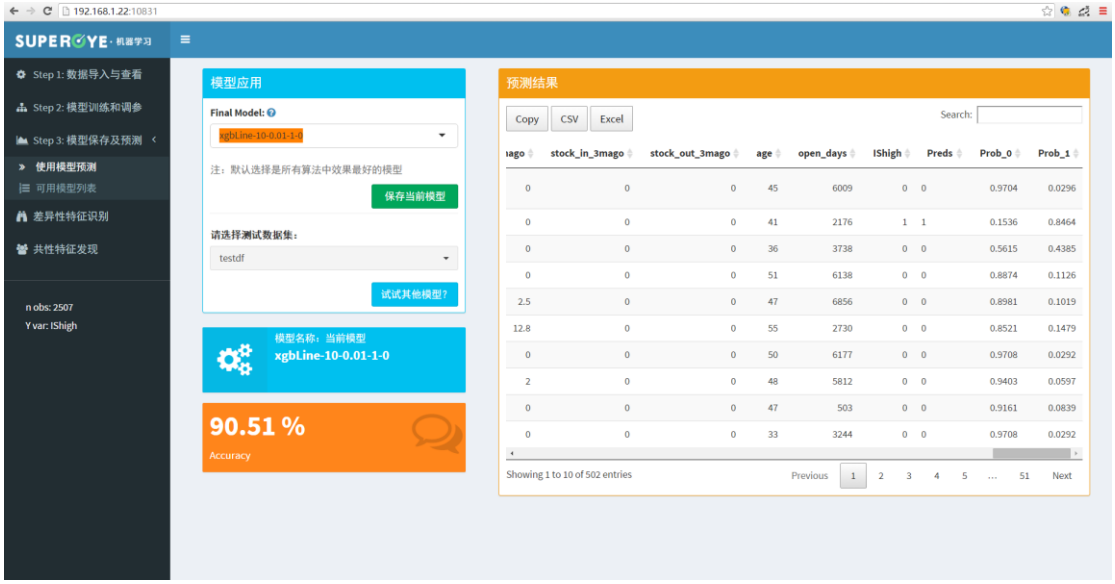

图 5.15 预测推断模块界面 1

说明：图 5.15 展示了用户选择 XGBoost Linear 最优参数模型和 Hive 中的测试集后，数据的预测情况。左下的两个信息框分布展示了模型的具体信息和准确率，高达 90%。右边的表格展示了测试集内容及其预测值，其中“IShigh”为测试集的真实 y 值（表示一个人是否为高收益客户），“Preds”字段为模型的预测 y 值，两字段几乎一致证明模型准确率较高。模型还给出了每条记录预测为 0 和预测为 1 的概率值“Prob_0”和“Prob_1”。

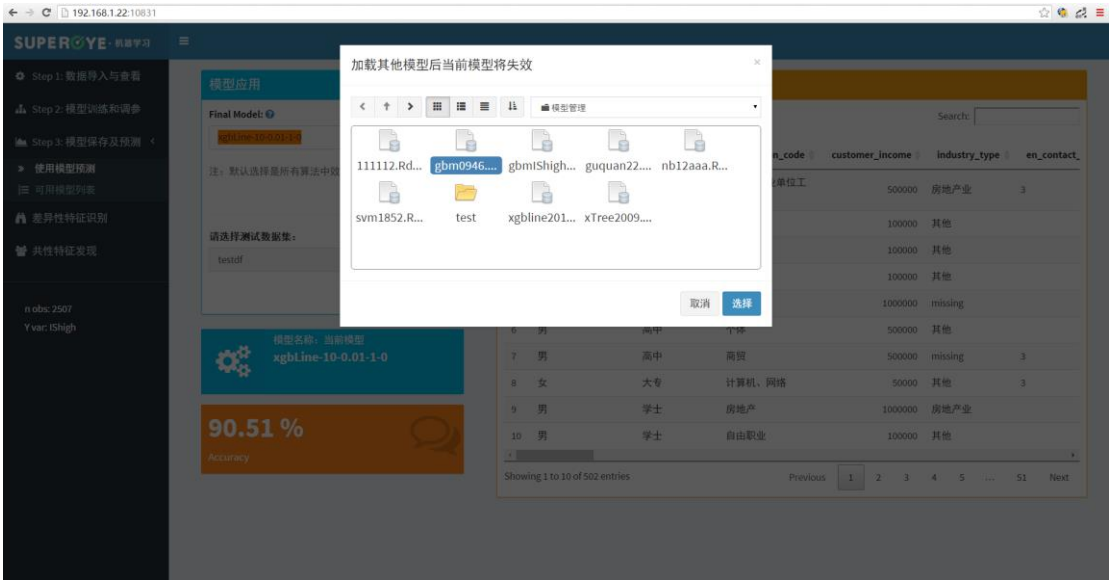

图 5.16 预测推断模块界面 2

说明：图 5.16 展示了当用户点击“试试其他模型”按钮时，会弹出一个下拉框，展现在服务器上存在的所有可用模型。用户如果选中其中的某个模型并点击“选择”，将会把当前的模型替换掉，并以它进行预测。

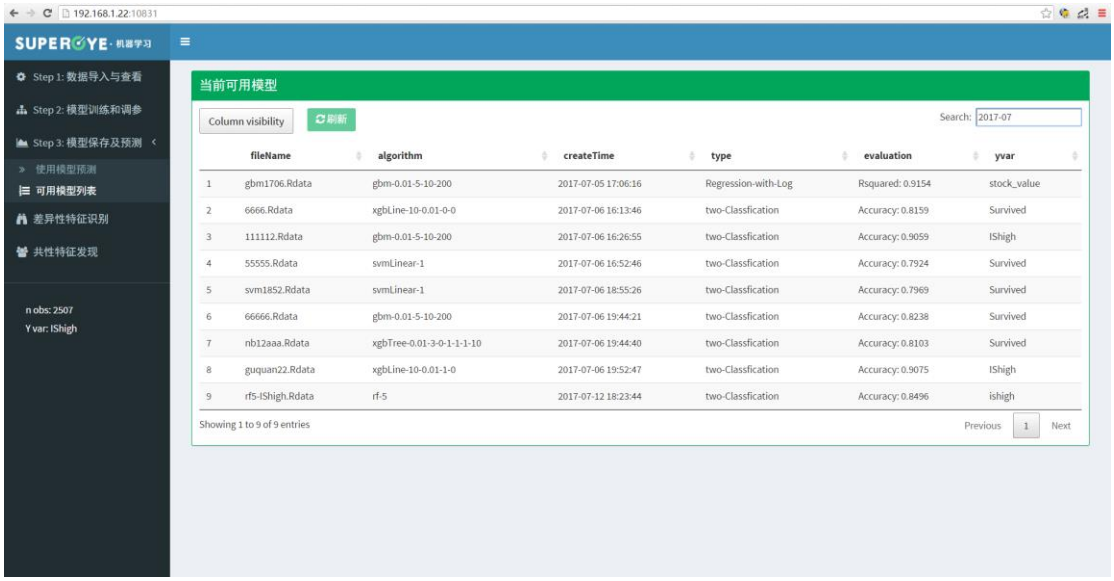

图 5.17 预测推断模块界面 3

说明：图 5.17 展现了服务器上可用模型的详细信息。在搜索框中输入“2017-07”表示筛选了 2017 年 7 月份保存在服务器上的模型。

5.5 SEML 定制化模块

前文已经提到，定制化模块针对的是某一行业或业务的定制化需求，由于前三个模块已经实现了有监督学习的基本流程，因此 SEML 在设计定制化模块时，解决的大多是无监督学习的需求。下文将以某证券公司的客户特征挖掘为案例，来对定制化模块进行说明。

5.5.1 案例背景

某证券公司融资融券业务部门想知道他们的客户到底有哪些特征。场景包括：1，某业务人员通过一定的业务规则筛选出了两批客户，他想知道两批客户的差异性特征在哪里，比如对高收益客户和低收益客户，业务人员想影响他们收益能力产生差异的主要指标是哪些；2，某业务人员通过长期以来与客户的线下沟通，手中握有一批客户群，他想知道这批客户都有哪些共性的特征。

5.5.2 差异性特征识别

基于上述案例的场景 1，SEML 设计了差异性特征识别的功能。用户通过界面上的单选框，选择两个自定义的客群 ID 列表（对底层而言即两张 Hive 表），平台会返回这影响这两批客户差异的主要特征是什么，然后按特征差异大小从高到底返回一个 Top 10 的排序，同时，如果用户点击了任意一个特征，想要知道这个特征在两批客群中的分布情况，平台还会绘制出动态可交互的分布图。其背后的原理是，平台会把这两个客群作为一个二分类问题，通过调用 Random Forest 算法建模，然后输出模型的特征重要性（Feature Importance），由于特征重要性反映了该特征对两类人群的区分力度，故可依据重要性大小来寻找差异特征。

5.5.3 共性特征挖掘

基于上述案例的场景 2，SEML 设计了共性特征挖掘的功能。用户只需要在“我的客群”输入框中选择自己想要观测的客群列表（底层同样是一张 Hive 表），平台便会返回这批客群的共性特征组合列表，并对共性特征利用网络图和平行坐标图进行可视化，让用户能够直观的看到客群的特征分布状况。其背后的原理是，对客群的所有特征进行离散化，然后转换成“购物篮分析”的问题——每一个客户的特征取值组合即是一个购物篮，然后利用 Eclat 算法进行频繁项集（Frequent ItemSets）挖掘，所谓“频繁”即是经常在一起出现的特征，实际上便是共性特

征。返回的频繁项集再按照支持度从大到小进行排序，排在越前面的表示该特征组合在所有的客户中出现的次数越多，意味着该共性越明显。

5.5.4 模块逻辑设计

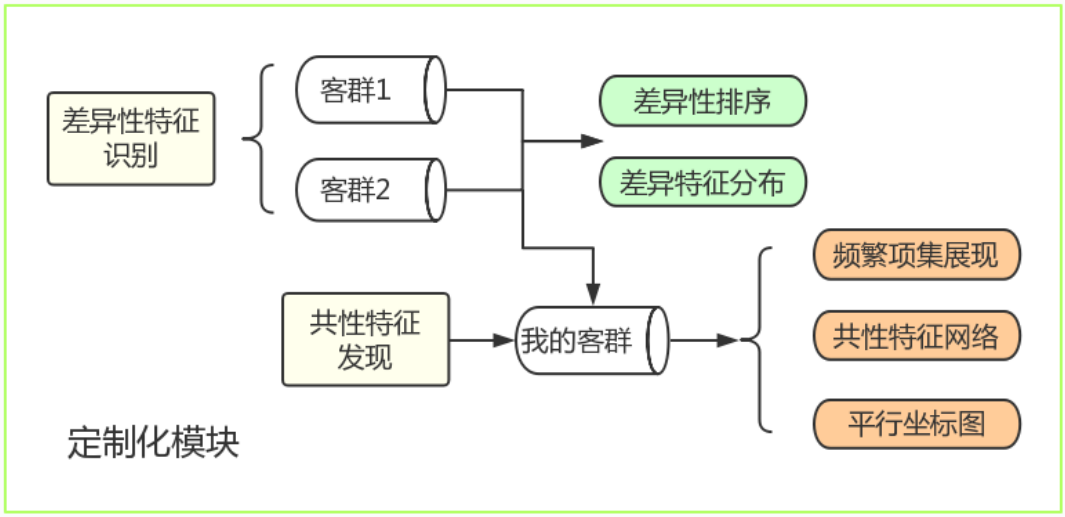

图 5.18 定制化模块逻辑设计

5.5.5 关键代码实现

```
...

## UI端
# 利用Highchart可视化差异性分布
highchartOutput('diffChart', height = "600px")
# 共性特征网络图和平行坐标图
column(width = 4, plotOutput('graph')),
column(width = 8, parcoordsOutput('pcoords', width = "100%", height = "45

## Server端
# 特征重要性函数
feature_importance <- function(X, y, n_trees=50, random_state=2017, max_d
  set.seed(random_state)
  clf = randomForest(X, y, ntree=n_trees, maxnodes=2^max_depth)
  imp_df = as.data.frame(importance(clf))
  imp_df$feature = row.names(imp_df)
  imp_df = imp_df[order(-imp_df[,1]),]
  return(imp_df)
}
# 购物篮生成函数
generate_bskt <- function(df_dscrtz, colsEN, colsCN){
  allcols = colnames(df_dscrtz)
  df_cols = allcols[allcols != 'customer_no']
  for(col in df_cols){
    colCN = colsCN[which(colsEN==col)]
    df_dscrtz[[col]] = factor(df_dscrtz[[col]], exclude = NULL)
    setnames(df_dscrtz, old = col, new = colCN)
  }
  if('customer_no' %in% allcols){
    bskt = as(df_dscrtz[, -c('customer_no')], 'transactions')
    bskt@itemsetInfo$transactionID = df_dscrtz$customer_no
  } else {
    bskt = as(df_dscrtz, 'transactions')
  }
  return(bskt)
}
# 频繁项集挖掘
sets = Eclat(bskt, parameter = list(support = 0.7, minlen = 6),
             control = list(verbose = F))
...
```

图 5.19 定制化模块关键代码

5.5.6 模块界面展示

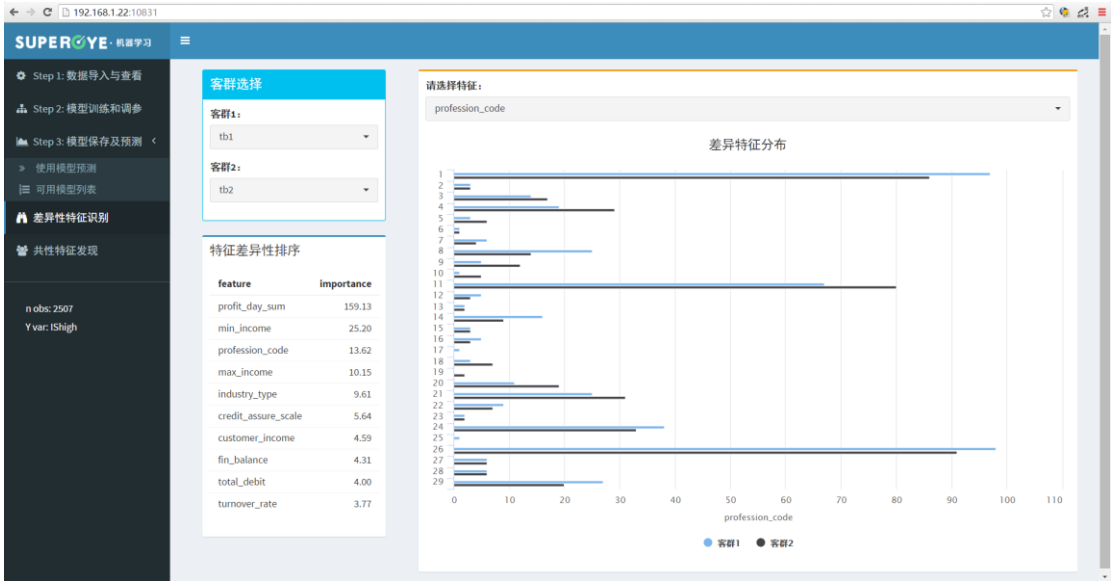

图 5.20 定制化模块界面 1

说明：图 5.20 展示了用户在选择两个客群后的特征差异性结果。右图展示了两个客群职业（profession_code）分布的差异对比图，用户还可以选择任一特征进行观测。

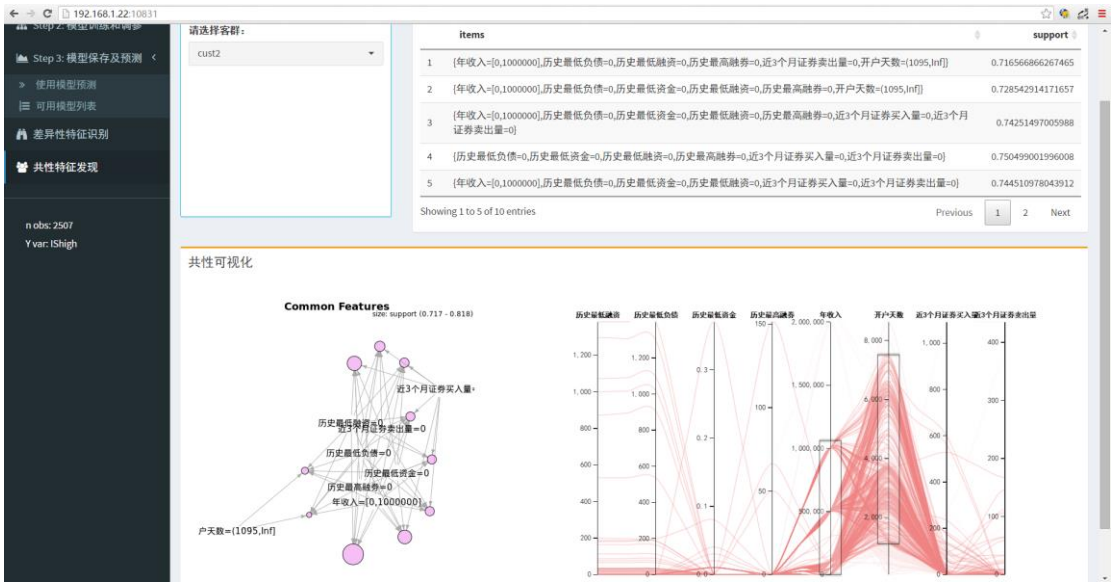

图 5.21 定制化模块界面 2

说明：图 5.21 展示了用户在选择自己想要观测的客群后，计算出的频繁项集结果。右上方的表格第一行表示有 71.6%（即第二列的支持度）的客户都具备“{年收入=[0,1000000],历史最低负债=0,历史最低融资=0,历史最高融券=0,近 3 个月证券卖出量=0,开户天数=(1095,Inf]}”的特征（由于是样例数据所以可能会发生数据失常）。左下以网络图的形式对共性特征进行可视化，圈的大小反映了支持度的值。右下的图是对所有客户绘制了平行坐标图（Parallel Coordinates），可以发现绝大部分的人群都是年收入百万以内、且开户天数在 1000days-8000days 之间。

5.5 本章小结

本章是全文最核心部分，主要对 SEML 的基本开发框架和每一个细分子模块进行了逻辑设计，以及使用 R 语言完成了代码实现，并对每个模块最终的界面进行了展示。

第6章 总结与展望

6.1 工作总结

本文的工作来源于两个实际项目需求：完善某公司的 SuperEye 的系统架构以及为某证券公司两融业务部门的客户数据进行挖掘。本文所研发的分布式机器学习平台正是应用到了这两个业务场景之中，并有效解决了项目需求。

首先，本文分析了机器学习的一般流程——从数据源获取到预测推断一共包括五个步骤。然后依据这个通用流程以及具体业务的定制化需求，本文设计了机器学习平台的整体架构，即将平台分为基于有监督学习的标准化模块和基于业务场景的定制化模块。之后，本文利用 R 语言对 SEML 进行了编程实现，其中用到的技术和框架包括 Sparklyr、CARET、Shiny 等。

本文在设计和实现平台的过程中，主要完成的工作列举如下：

1) 先将 SEML 粗分成标准化模块和定制化模块，然后前者又细分为数据导入、模型训练和预测推断三块，后者又细分为差异性特征和共性特征两块。对每一细分的模块，为 SEML 设计一个页面。
2) 对平台将要使用到的机器学习算法逐一进行分析，理解算法的原理、进行数学推导并完成伪代码。这些算法包括 SVM、RF、GBDT、XGBoost、Eclat 等。
3) 利用 Spark 的 R 接口 Sparklyr 连接到 Hive 数据库，用代码实现数据导入模块。
4) 利用 CARET 集成 R 中的上述算法库，代码实现模型训练以及预测推断模块，包括自动化调参、交叉验证、模型评估可视化、测试数据的导入及预测等功能。
5) 利用 Eclat 算法，用代码实现定制化模块的功能。
6) 最后利用 R 的 Web 开发框架 Shiny，打通前后台，真正开发出具有可视化界面的分布式机器学习平台。

6.2 下一步计划

由于笔者能力和精力的限制，本文所开发出的分布式机器学习平台必然会存在一定的不足。相比于本文在第一章中介绍的国内外大型机器学习台，本文的平台在许多功能上都未能支持。笔者认为 SEML 可以在以下几个方向作出改进:

1) 从数据的分布式扩展为算法的分布式。SEML 的分布式性能体现在数据源的获取上（Hive 之中），但进行机器学习算法建模时，依然是在单机上进行。故后期可以考虑通过 Spark 的 R 接口来使用 MLlib 进行算法的分布式训练。
2) 从点击式的操作到拖拽式的操作。微软 AMLS、阿里 PAI 以及百度 BML 平台的亮点在于可以通过拖拽模块来“画出”DAG 图，进而完成机器学习。为了更好的交互性，这将是 SEML 的发展方向。
3) 支持内嵌 R 和 SQL 的代码。SEML 希望可以和微软 AMLS 一样，除了模块的拖拽与调用外，还支持用户在平台上书写 R 或 SQL 的代码进行数据挖掘，这个功能将会让它具备更好的灵活性。

参考文献

[1] 刘闯.基于多核计算的分类数据挖掘算法研究[D].硕士学位论文,南京:南京航空航天大学, 2011.

[2] Woodsend, K.& Gondzio, J. Hybrid MPI/OpenMP parallel linear support vector machine training[J].Journal of Machine Learning Research, 2009, 10: 1937-1953.

[3] Narang, A., Gupta, R.& Joshi, A. *et al*. Highly scalable parallel collaborative filtering algorithm[C]. Proceedings of the 2010 International Conference on High Performance Computing, Dona Paula, Dec 19-22, 2010. Piscataway, NJ, USA: IEEE, 2010: 1-10.

[4] Jin Lei, Wang Zhaokang, Gu Rong, *et al*. Training large scale deep neural networks on the Intel Xeon Phi many-core coprocessor[C]. Proceedings of the 2014 IEEE 28th International Parallel & Distributed Processing Symposium Workshops (ParLearning), Phoenix, USA, IEEE, 2014: 1622-1630.

[5] The Apache Software Foundation. Apache Mahout: scalable machine learning and data mining[EB/OL]. http://mahout.apache.org, 2017.07.01.

[6] Wei Jie, Shi Hongbo, Ji Suqin. Distributed Naive Bayes text classification using Hadoop[J]. Computer Systems and Applications, 2012, 21(2): 210-213.

[7] 卫洁,石洪波,冀素琴. 基于 Hadoop 的分布式朴素贝叶斯文本分类[J]. 计算机系统应用, 2012, 21(2): 210-213.

[8]The Apache Foundation. Machine Learning Library (MLlib) Guide[EB/OL]. https://spark.apache.org/docs/latest, 2017.07.01.

[9] Dean Jeff, Monga Rajat, *et al*. TensorFlow: Large-scale machine learning on heterogeneous systems[EB/OL]. https://tensorflow.org. Google Research. 2015.11.09

[10] Samuel Arthur[J]. Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 1959, 3 (3).

[11] Brett Lantz. Machine Learning with R[M]. Birmingham:Packt Publishing, 2013, 10:56-60.

[12] Kaggle. 2017 The State of Data Science & Machine Learning[EB/OL]. https://

www.kaggle.com/surveys/2017

[13] Gregory Piatetsky. New Leader, Trends, and Surprises in Analytics, Data Science, Machine Learning Software Poll[EB/OL]. https://www.kdnuggets.com/2017/05/ poll-analytics-data-science-machine-learning-software-leaders.html, 2017.05.

[14] 任亚洲.高维数据上的聚类方法研究[D].博士学位论文,华南理工大学,2014.

[15] Boser, Vapnik. A training algorithm for optimal margin classifiers[J]. Proceedings of the fifth annual workshop on Computational learning theory, doi:10.1145/130385.130401. ISBN 089791497X, 1992:144-145.

[16] 林琳.智能机器人视觉中的人脸识别研究[N].长沙航空职业技术学院学报,2015,15(3):64-69.

[17] 姚文琳.汉语依存句法分析方法的研究与实现[D]博士学位论文,中国海洋大学,2009.

[18] Ho Tin Kam, Random Decision Forests[J]. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 1995.08: 278-282.

[19] Friedman J. H. Greedy Function Approximation: A Gradient Boosting Machine[J]. The Annals of Statistics, 1999.02.

[20] Chen Tianqi, Carlos Guestrin. XGBoost: A Scalable Tree Boosting System[J]. CoRR. abs/1603.02754. arXiv:1603.02754, Freely accessible, 2016.05.09.

[21] DMLC. XGBoost GPU Support[EB/OL]. https://xgboost.readthedocs.io/en/latest/gpu/index.html, 2017.

[22] 何通. Xgboost: 速度快效果好的 boosting 模型[EB/OL]. https://cosx.org/2015/03/xgboost, 2015.

[23] Rakesh Agrawal, Ramakrishnan Srikant. Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, 1994.09: 487-499

[24] 王威,陈梅.基于位集合的 Apriori 算法的改进[J].计算机技术与发展,2011,21(12):70-72.

[25] 赵伟毅.基于数据挖掘技术的图书产品信息服务平台设计与实现[D]硕士学位论文,北方工业大学,2009.

[26] Mohammed J. Z., Srinivasan P.& Ogihara, *et al*. New Algorithms for Fast

Discovery of Association Rules[J]. KDD, 1997.

[27] 梁伟.校园网用户行为分析系统研究与实现[D].硕士学位论文,北京交通大学,2009.

[28] Max Kuhn. The caret Package[EB/OL]. http://topepo.github.io/caret/index.html, 2017.09.04.

[29] 谢佳标. 利用 shiny 包快速搭建可视化原型系统[EB/OL]. https://cosx.org/2016/06/use-shiny-fleetly-set-up-visual-prototype-system, 2015.

[30] Javier Luraschi. sparklyr: R interface for Apache Spark[EB/OL]. http://spark.rstudio.com/, 2017.

[31] Aarshay Jain. Complete Guide to Parameter Tuning in XGBoost [EB/OL]. https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/, 2016.03.01

[32] 黄玉萍.基于 LSSVM 的弓网系统建模研究[D].硕士学位论文,华东交通大学,2015.

[33] Smeeton N.C. Early History of the Kappa Statistic[J]. Biometrics, 1985(41): 795.

作者简历

教育经历:

2010.08 至 2014.06，浙江大学，本科，公共管理 - 行政管理系

2015.08 至 2018.03，浙江大学，硕士，软件工程 - 金融数据分析技术

工作经历:

2016.05 至 2017.04，北京网思科平科技有限公司，任大数据分析工程师

2017.04 至今，杭州量知数据科技有限公司，任数据挖掘工程师

攻读学位期间发表的论文和完成的工作简历:

1，微博计算项目：负责数据处理、可视化、前端展示

2，精准营销 DMP 项目：负责文本挖掘、机器学习建模

3，绍兴公安大数据项目：负责机器学习、可视化

4，机器学习平台开发：全权负责

5，财通证券两融潜在客户挖掘：负责机器学习建模

6，财通证券客户画像构建：负责指标/标签设计及代码实现

致谢

时光荏苒，我的硕士生涯已接进尾声。这几年的时光既漫长又短暂，其中充满了酸甜苦辣，更有收获和成长。几年来，感谢陪我一起度过美好时光的每位尊敬的老师、亲爱的同学和同事，正是你们的帮助，我才能克服困难，正是你们的指导，我才能解决疑惑，直到学业的顺利完成。

本人的学位论文是在我的导师汤斯亮副教授的殷切关怀和耐心指导下进行并完成的，衷心感谢我的导师对我的淳淳教诲和悉心关怀。从课题的选择、项目的实施，直至论文的最终完成，汤教授都始终给予我耐心的指导和支持，我取得的每一点成绩都凝聚着导师的汗水和心血。导师开阔的视野、严谨的治学态度、精益求精的工作作风，深深地感染和激励着我，在此谨向汤教授致以衷心的感谢和崇高的敬意。

感谢所有关心帮助过我的人，祝你们一切顺利，幸福美满！

顾全

于浙江大学软件学院

2018 年 1 月