**面向领域实践1个人实验报告**

**北京邮电大学计算机学院**

**2024-2025 学年第 1 学期项目总结报告**

**课程名称： 面向特色化领域的实践1-大数据软件**

**项目名称： MarketPulse交易数据分析**

**项目完成人：**

**姓名： 徐楚欣阳 学号： 2022211451**

**姓名： 谢又新 学号： 2022211464**

**姓名： 周裕顺 学号： 2022211466**

**姓名： 高志强 学号： 2022211453**

**姓名： 华为 学号： 2022211457**

**指导教师： 孙鹏飞 杨金翠**

**日期：2024 年12月25日**

**一. 项目背景和目的：**

**1.1项目背景：**

**1.2项目目的：**

**二.项目开发环境：**

**三.项目数据及来源：**

**3.1 项目数据：**

**3.2 数据来源：**

**四．项目内容：**

**4.1问题分析：**

**4.1.1 欺诈交易识别模块**

欺诈交易会对用户体验和商家运营造成重大影响。

准确检测欺诈交易是商家风控的核心需求。

本项目旨在构建高效的欺诈预测模型，以满足**准确率、召回率**等关键指标的要求。

**4.1.2 商家数据分析**

通过大规模数据分析识别商家的销售模式和消费者行为。数据量庞大且复杂，涉及多个变量（如交易时间、支付方式、商家类型等），挑战在于如何高效处理这些数据并从中提取有价值的洞察。传统分析方法在处理这些数据时存在性能瓶颈，而基于Spark的分布式计算框架能够有效应对大数据分析中的挑战，提供实时、高效的数据处理能力。

**4.1.5 流失用户预测模块**

用户流失预测是用户行为分析中的关键任务之一，其目标是提前发现有流失倾向的用户并采取针对性的运营策略，减少企业损失。该项目聚焦于以下问题：

* **问题定义**：通过用户的历史交易数据预测用户是否会在未来一段时间内流失。
* **现状挑战**：

1. 数据量大：1400万条交易记录，覆盖2000名用户。
2. 数据稀疏：某些用户在较长时间内交易频率低，可能导致数据缺失问题。
3. 实时性需求：需要将预测结果存储到数据库，为后续业务系统提供支持。

* **目标**：构建一个基于 LSTM 的用户流失预测模型，并设计完整的数据预处理、模型训练、预测与存储的系统化解决方案。

**4.2设计方案：**

**4.2.1 欺诈交易识别模块**

**模块目标**

* 构建一个高效的欺诈预测模型，能够准确检测欺诈交易，同时满足精准率、召回率等关键指标的要求。
* 提升特征工程和模型性能，通过针对性的特征编码与优化实现业务价值最大化。

**特征工程策略**

1. **独热编码**

* 将离散特征（如商家 ID、用户 ID、MCC）转换为独热向量，增强特征可解释性。

1. **目标编码**

* 计算每个商家 ID、用户 ID、MCC 对应的欺诈率作为新的特征。
* 避免过拟合，采用交叉验证分组计算目标编码。

1. **频率编码**

* 统计商家 ID、用户 ID、MCC 在数据集中出现的频率作为特征。

1. **神经网络嵌入**

* 使用神经网络对离散特征（如商家 ID、用户 ID）进行嵌入编码，降低维度的同时保留业务相关性。

1. **时间特征**

* 提取交易发生时间的特征（如小时、工作日/非工作日），分析时间模式与欺诈的相关性。

1. **金额分箱**

* 对交易金额进行分箱处理，捕捉金额区间与欺诈行为的关联。

**模型选择**

* 基于随机森林模型：适合处理高维稀疏数据，具有良好的可解释性。
* 深度学习模型（如 DNN）：结合嵌入层与多层感知器，适合高维特征。
* 集成学习：通过模型融合提升整体性能。

**4.2.5 流失用户预测**

**整体架构**

本项目分为四个主要模块：数据处理、模型构建与训练、预测与存储、前后端集成。

1. **数据处理模块**：

* 数据清洗与格式化。
* 按时间序列生成输入特征（每日交易数量）及标签（是否流失）。

1. **模型构建与训练模块**：

* 使用 PyTorch 实现基于 LSTM 的预测模型。
* 按序列时间长度设计输入特征。

1. **预测与存储模块**：

* 使用训练好的模型对数据集进行批量预测。
* 将预测结果（流失概率）写入 MySQL 数据库。

1. **前后端集成模块**：

* 设计前端界面显示预测结果。
* 后端提供 API 接口支持实时查询与展示。

**核心算法**

* **长短期记忆网络（LSTM）**：
* LSTM 适合处理时间序列数据，能够捕捉交易行为中的长期依赖关系，准确预测用户流失倾向。
* 模型输出二分类结果（流失或非流失），同时保存用户流失概率。

**4.3相关知识和算法：**

**4.3.1 欺诈交易识别模块**

**特征编码方法**

1. **独热编码（One-Hot Encoding）**

* 将离散特征转换为二进制矩阵表示。
* 优点：简单直接，易于理解。
* 缺点：高基数特征会导致维度膨胀——本项目有15000个商家、1000个用户，直接使用独热编码维度约为16000维，矩阵过大内存不够无法运行。

1. **目标编码（Target Encoding）**

|  |  |
| --- | --- |
| * 核心公式： |  |

这种方法通过结合全局均值和特定特征的目标均值，使编码结果更鲁棒且易于泛化。

* 其中，

yi 是目标变量值（例如欺诈标签：0 或 1）。

N 是特定特征值的样本数量。

Prior 是目标变量的全局平均值。

α 是平滑参数，用于平衡全局均值与特定特征均值的权重，避免过拟合。

* 优点：利用目标变量的信息增强特征表达。
* 缺点：容易过拟合，需结合交叉验证计算。

1. **频率编码（Frequency Encoding）**

|  |  |
| --- | --- |
| * 核心公式： |  |

* Xi 表示类别 i 的频率编码值。
* count(i) 表示类别 i 在数据集中的出现次数。
* N 表示数据集中的总样本数。

**优点：**

* 简单高效，适用于大规模离散特征。

**缺点：**

* **可能引入偏差**：频率编码只考虑了类别出现的频率，而没有捕捉到类别与目标变量之间的关系，这可能导致模型学习不到类别的潜在特征，特别是在类别频率分布不均的情况下。
* **无法处理类别新颖性**：频率编码对于未在训练数据中出现的新类别无法处理，会导致无法为这些类别生成有效的编码。
* **可能导致过拟合**：如果某些类别频繁出现，频率编码可能会过度强调这些类别的影响，进而增加过拟合的风险，尤其是在训练集和测试集分布不一致时。

1. **神经网络嵌入（Embedding）**

* 使用嵌入层将高维稀疏特征映射到低维连续空间。

**优点：**

* **捕获特征之间的潜在关系**：嵌入层可以学习到类别之间的关系，并将相似的类别映射到相近的向量。这有助于模型理解类别之间的潜在结构和关联性。
* **适用于大规模数据**：对于高基数的特征，嵌入层提供了一种有效的编码方式，避免了独热编码可能带来的维度爆炸问题。
* **更高效的特征表示**：嵌入层通过将类别映射为低维向量，减少了计算资源的消耗，尤其是在大规模数据集上，能够更有效地处理特征。
* **可迁移性**：在不同任务之间，嵌入学习到的表示可以被迁移到新的任务中，进行迁移学习，从而加速新任务的学习过程。

**缺点：**

* **训练需求大**：嵌入层需要通过神经网络训练来学习合适的嵌入向量，这对于数据量较小或者没有足够训练样本的情况可能表现不佳。
* **需要更多计算资源**：与传统的编码方法相比，神经网络嵌入需要较多的计算资源，尤其是在处理大量类别和高维特征时。
* **可能过拟合**：如果嵌入层的维度过高，或者数据本身的噪声较大，模型容易出现过拟合，尤其是当类别数量过多而数据量不足时。
* **黑箱性质**：与一些传统编码方法相比，嵌入层的学习过程较为复杂，不容易解释每个嵌入向量背后的含义，使得模型的可解释性较低。

**机器学习模型**

1. **随机森林**

* 特点：支持稀疏特征，具有高效计算能力和良好的可解释性。
* 应用：处理高维数据和特征编码后的结果。
* 方法：使用神经网络降维处理后的特征列再进行随机森林模型处理。

1. **深度神经网络（DNN）**

* 特点：适合高维嵌入特征，支持复杂非线性关系。
* 应用：结合嵌入编码实现端到端欺诈检测。

1. **模型融合**

* 方法：分步使用。
* 目标：使用神经网络降维处理后的特征列再进行随机森林模型处理——综合多个模型的优势，提升性能。

**4.3.5 流失用户预测模块**

**数据处理相关知识**

* **时间序列数据特征提取**：为每个用户生成每天的交易记录（input\_a）及未来 30 天交易总数，构造输入特征和标签。
* **归一化处理**：交易记录数量具有较大波动，需对数据进行 Min-Max 归一化以适配 LSTM 模型。

**模型相关算法**

* **LSTM 模型结构**：
* 输入层：每日交易记录。
* 隐藏层：LSTM 层提取时间序列特征。
* 全连接层：预测未来是否流失（输出为流失概率）。
* 激活函数：使用 Sigmoid 函数输出流失概率。
* 损失函数：二分类交叉熵。
* **优化方法**：Adam 优化器，学习率设置为 0.001。

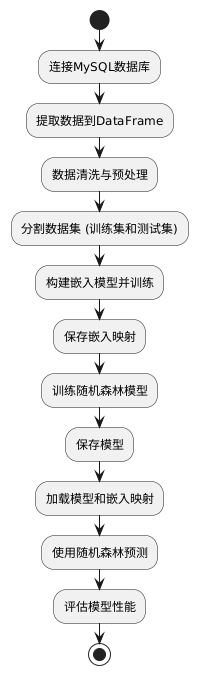
**数据库和前后端知识**

* 数据存储：使用 MySQL 存储预测结果，结构包括日期、用户 ID 和流失概率。
* 前端开发：利用 vue框架 构建可视化界面，展示预测结果。
* 后端开发：利用 springboot框架 提供 API，实现数据库访问与查询。

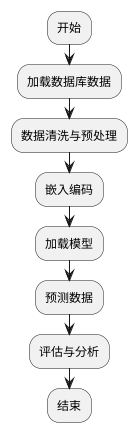
**4.4流程图和设计图:**

**4.4.1 欺诈交易识别模块**

训练模型：

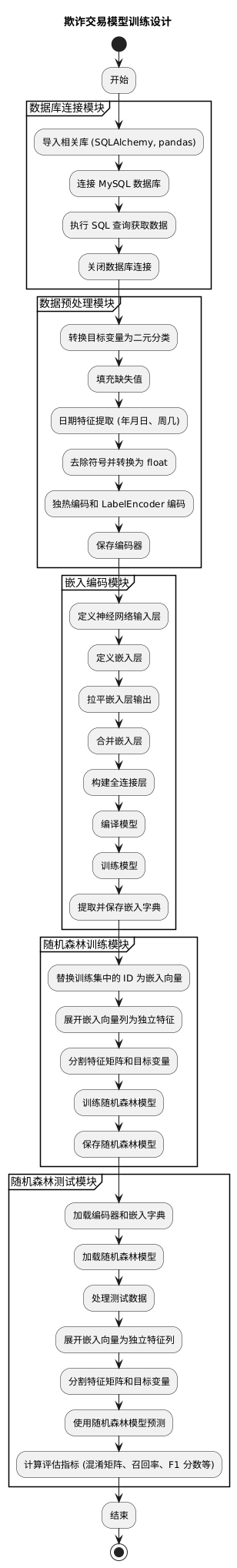


对于训练好的模型进行测试：

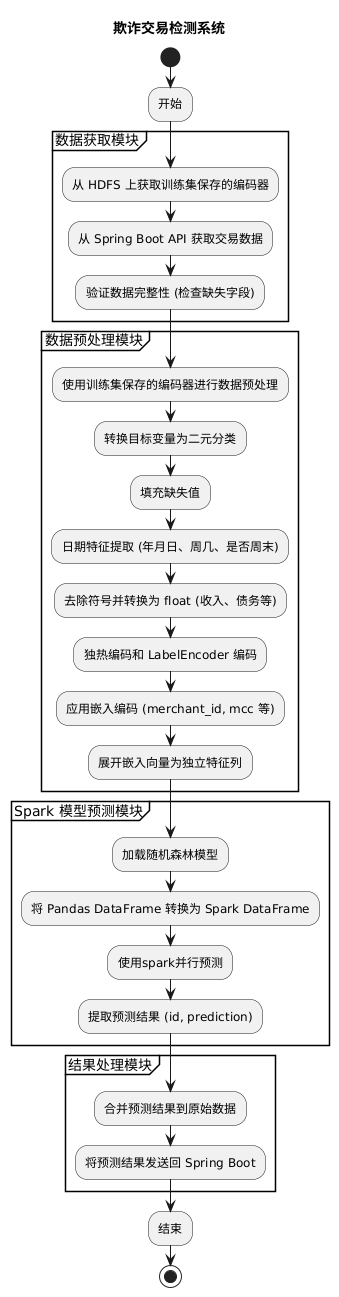


**详细设计：**

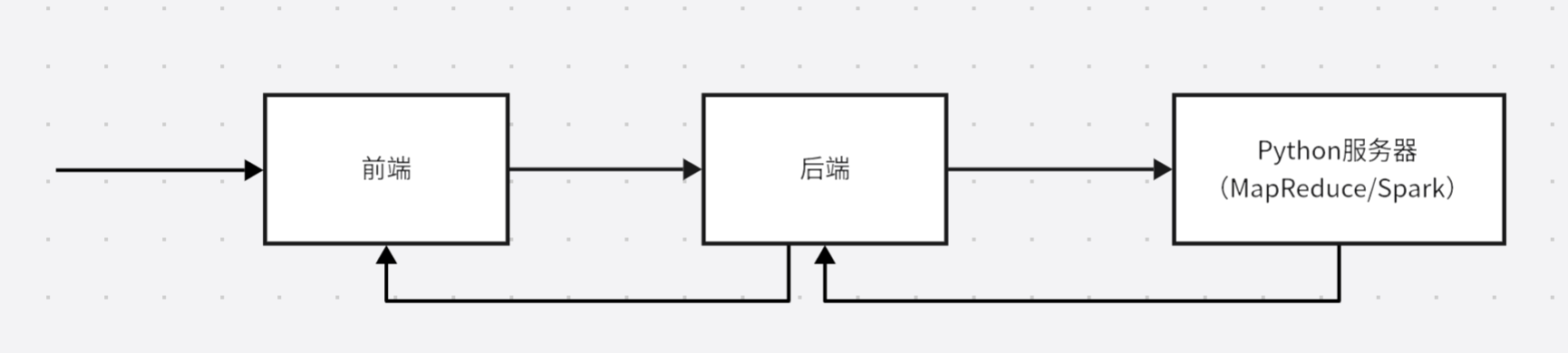
训练部分



**部署、测试部分：**

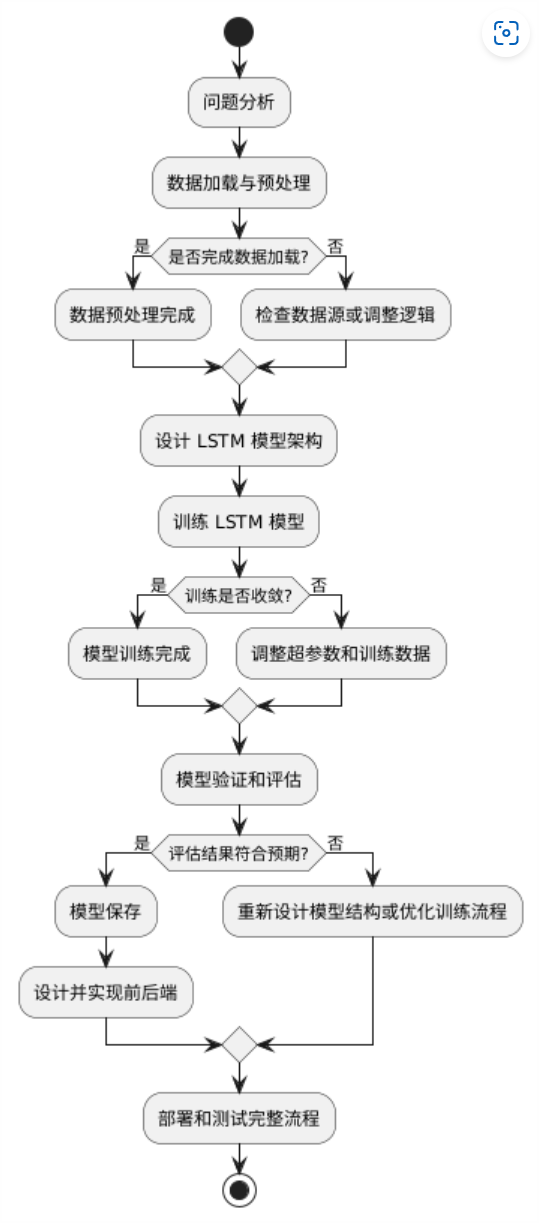


**4.4.2 商家推荐模块**

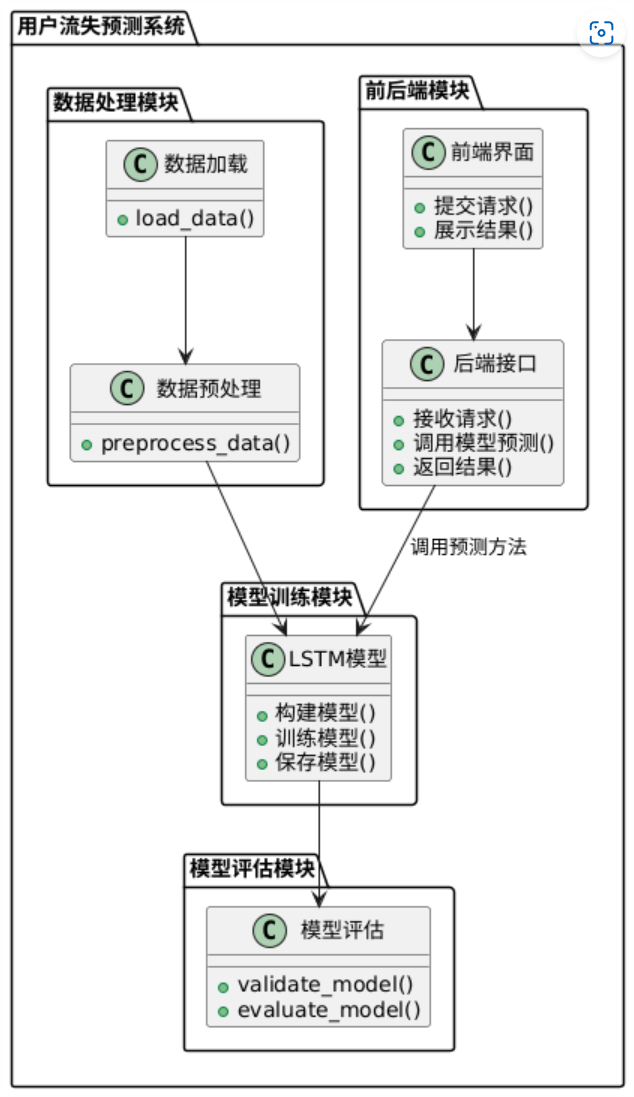


**4.5 MapReduce/Spark设计及分析：**

**流程图**

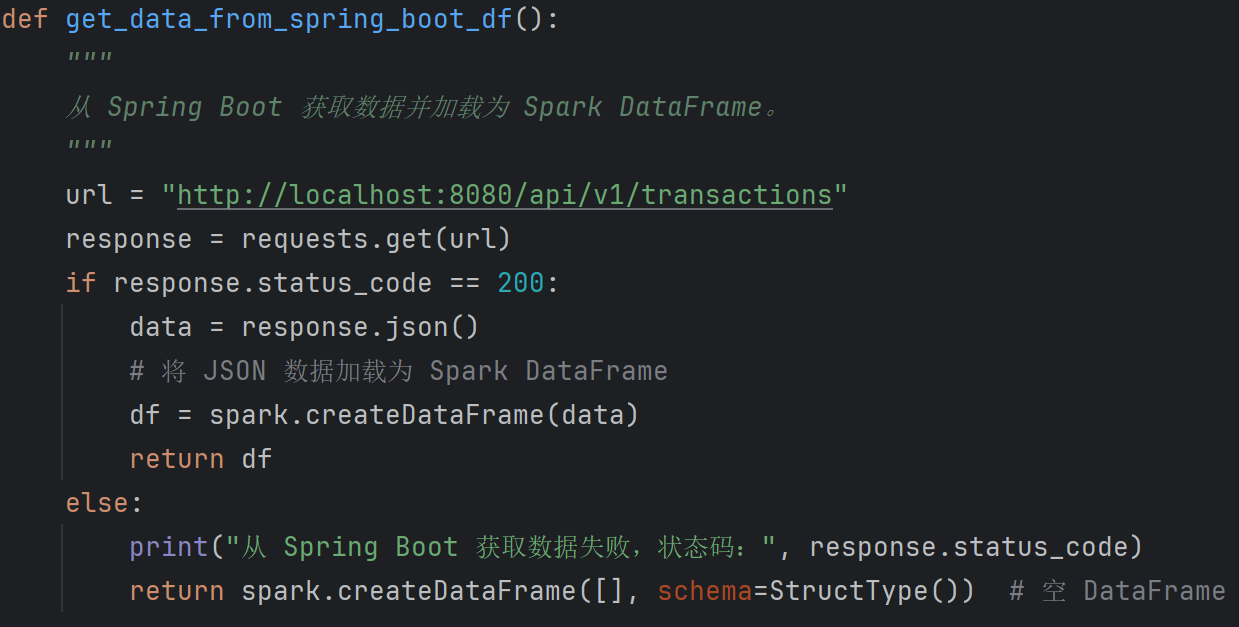


**设计图**



**4.5.1 欺诈识别模块**

1. **从 Spring Boot 获取数据**



1. **createDataFrame**

* 直接将从 API 获取的数据转换为 Spark DataFrame，方便进行分布式操作。

1. **空 DataFrame**

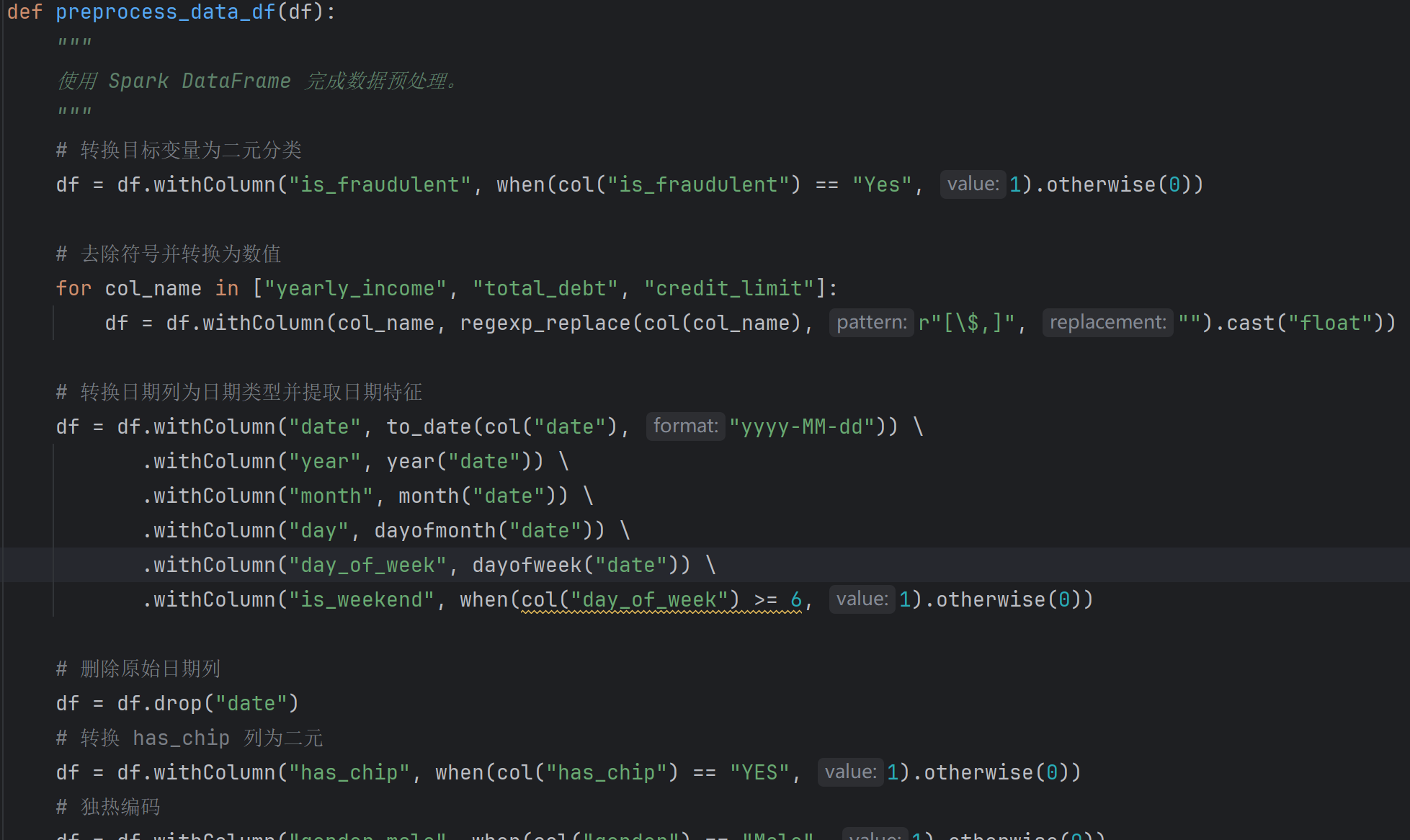
* 当无数据时，创建一个空 DataFrame 防止后续流程出错。

2. **验证数据格式 (validate\_data)**



* **df.columns**  
  获取 DataFrame 的所有列名。
* **集合操作**  
  使用 set 对比实际列和预期列，快速识别缺失字段。

3.  **数据预处理 (load\_and\_preprocess\_data)**



1. **withColumn**

* 新增或修改列。
* 灵活支持复杂逻辑，如条件分支 (when) 和正则替换 (regexp\_replace)。

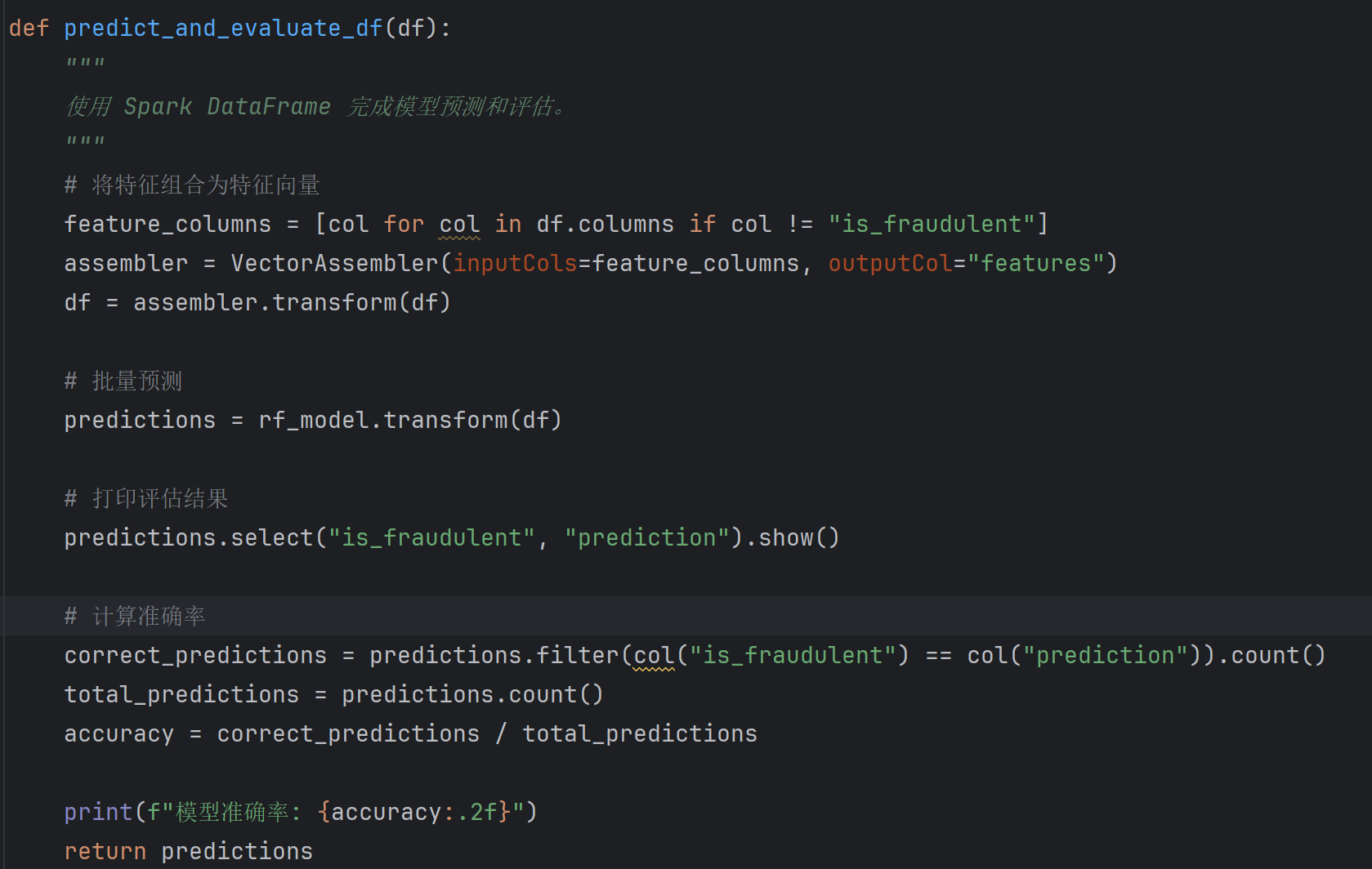
1. **日期处理**

* 使用 to\_date 转换为日期类型。
* 使用 year、month 等提取日期特征。

1. **特征列编码**

* 加载训练时保存的的编码器进行编码。

4. **模型预测和评估**



1. **VectorAssembler**

* 将多个特征列组合为单一的向量列 features，供模型使用。

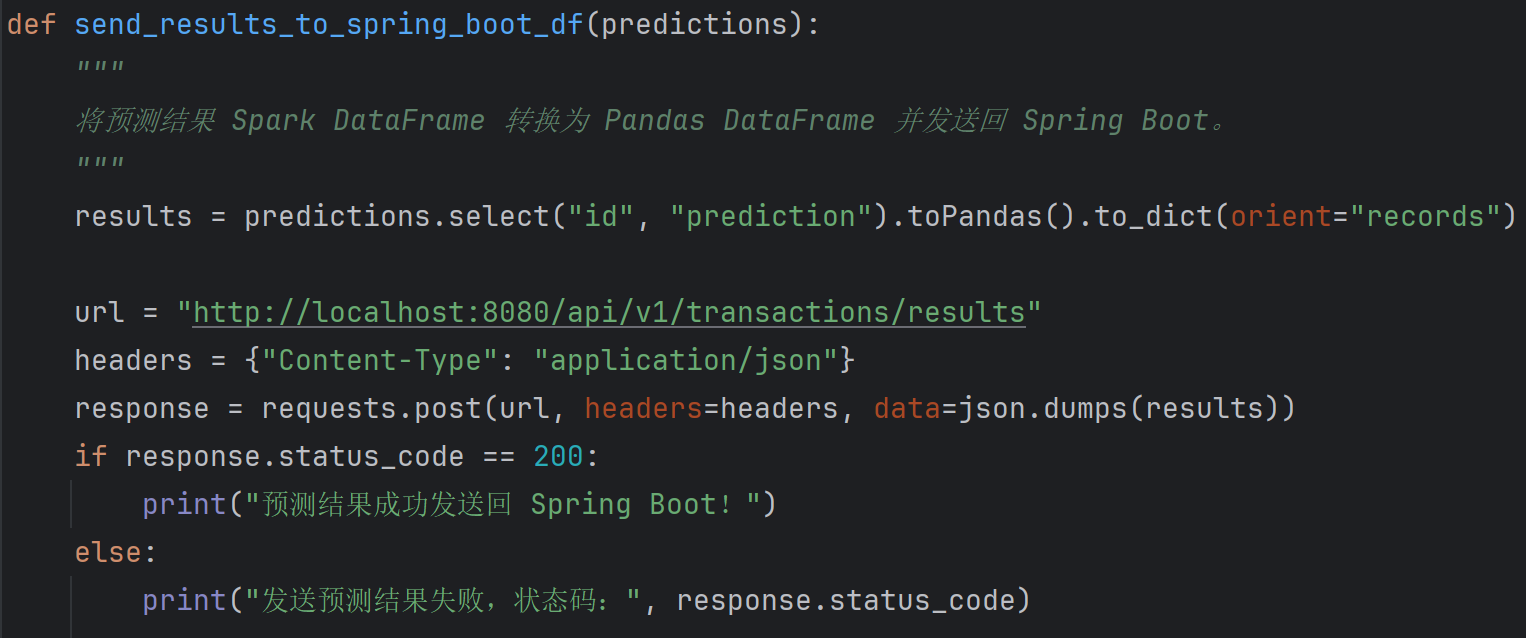
1. **批量预测**

* 使用 rf\_model.transform(df) 对整个 DataFrame 批量预测。

1. **准确率计算**

* 用 filter 筛选正确预测结果并统计。

5. **发送结果到 Spring Boot**



1. **toPandas**

* 将 Spark DataFrame 转为 Pandas DataFrame，方便处理小规模结果集。

1. **字典转换**

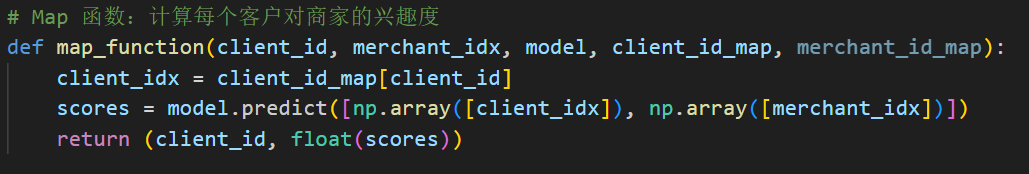
* 用 .to\_dict 将 Pandas DataFrame 转为 API 可用的 JSON 格式。

**4.5.4 商家推荐模块**

从前端接收输入的数据，发送到后端。后端调用python服务器对相关数据进行计算，然后接收从服务器返回的结果再传给前端。其中涉及到分布式计算的部分部署在python服务器上，对其MapReduce/Spark的设计进行分析介绍：

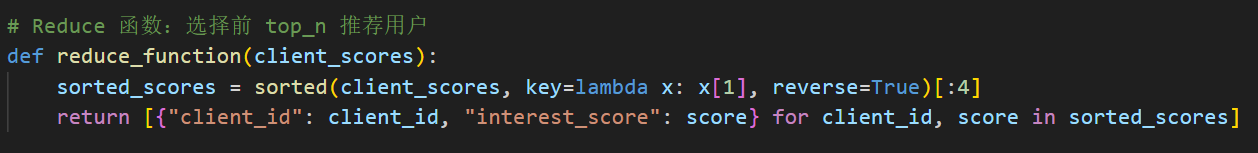
**Map操作**

Map操作的主要任务是并行计算每个用户对特定商家的兴趣度。输入数据是每个用户的ID以及商家的ID。然后使用训练好的模型预测每个用户对该商家的兴趣得分。最后返回每个客户的兴趣得分。



**Reduce操作**

Reduce操作的任务是对Map操作产生的结果进行聚合，并选择出前top\_n个兴趣度最高的用户。这一步骤涉及到排序并选择前几个用户。



**Spark并行计算**

这里的 recommend\_clients 函数负责将 map 和 reduce 操作结合起来，完成分布式计算并生成推荐。整个过程在Spark中并行处理，使用 sc.parallelize 将用户ID分发到多个工作节点；使用 .map() 方法并行执行 map\_function，计算每个用户对商家的兴趣得分；使用 .collect() 方法将所有计算结果收集到本地；使用 reduce\_function 对结果进行排序，筛选出前top\_n个用户。

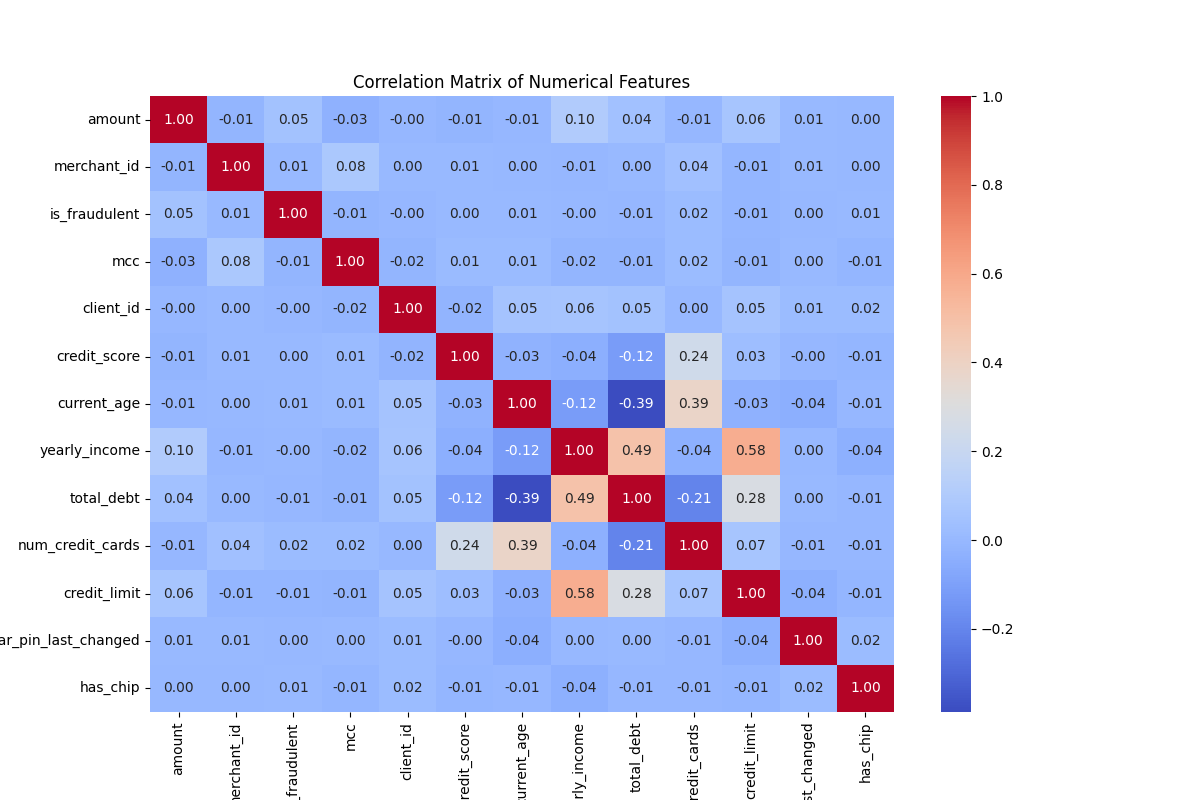


**五.项目过程及结果：**

**5.1欺诈识别模块运行过程及结果**

**实验过程**

1. **数据分析**



数据集包含 800 万条带标签的交易记录。

* 样本最重要的特征包括**商家 ID、用户 ID、MCC**（商家分类代码）、商家地址（邮编）、交易金额等。
* 标签为二分类：0 表示正常交易，1 表示欺诈交易。

**关键发现**

* 商家 ID 与欺诈强相关：某些商家 ID 的欺诈率显著高于平均水平。
* 用户 ID 与欺诈强相关：特定用户 ID 出现欺诈行为的概率较高。
* MCC 与欺诈强相关：特定商家类别的欺诈交易比例较高。

2. **数据预处理**

* 本身为有数学含义的数字特征列无需处理
* 金额特征列将字符串转为数字——去除$符号，删除","分隔符
* 将日期转化为年、月、日、小时、分、星期几这些特征列
* 取值较少的特征列直接使用独热编码（如has\_chip、credit\_card\_type、51个州等）

关键是**如何对这些ID进行编码**

将 ID（如用户 ID 或商家 ID）直接作为特征列输入模型导致以下问题

* 无法捕获业务含义
* 原因：ID 本身是纯标识符，缺乏业务语义。
* 结果：模型无法从中提取出任何有意义的统计规律或业务相关性。

**特征编码方案**

* 独热编码：将每一个ID当作一个0、1特征列
* 目标编码：基于目标变量（欺诈标签）计算特征的统计值。
* 频率编码：统计特征在数据集中出现的频率。
* 神经网络聚类编码：使用嵌入方法对特征进行降维处理。

由于ID数量较多——15000个商户ID、1000个用户ID、800个邮政编码ID、51个州，独热编码在大规模数据集无法使用，因此主要考虑其他三种：

3.  **目标编码实验**

**模型类型：**

* 随机森林。

**数据规模：**

* 100 万条记录。

**训练参数：**

由于欺诈交易极少，需要调整权重

* 100-300 棵树，权重1：10、1：100、1：869（数据库中欺诈与非欺诈比例）。

**最好结果**：300 棵树、权重1：10

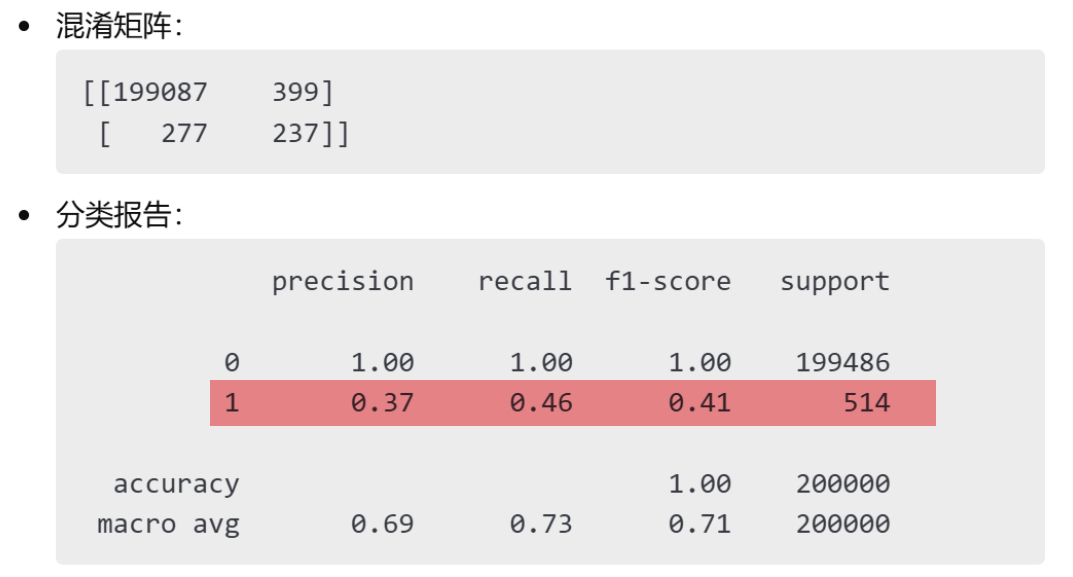
**评估指标：**

* F1-score：0.4122
* 召回率：0.4611
* ROC-AUC：0.9528

少数类（欺诈类）的

**精准度（precision）**——模型识别到的欺诈中确实为欺诈的比例：0.37

**召回率（recall）** ——模型识别到的欺诈占真实欺诈的比例：0.46



4. **频率编码实验**

* **模型类型：**
* 随机森林。

**数据规模：**

* 100 万条记录。

**训练参数：**

* 100-300 棵树，权重1：10、1：100、1：869（数据库中欺诈与非欺诈比例）。

**最好结果**：300 棵树、权重1：100

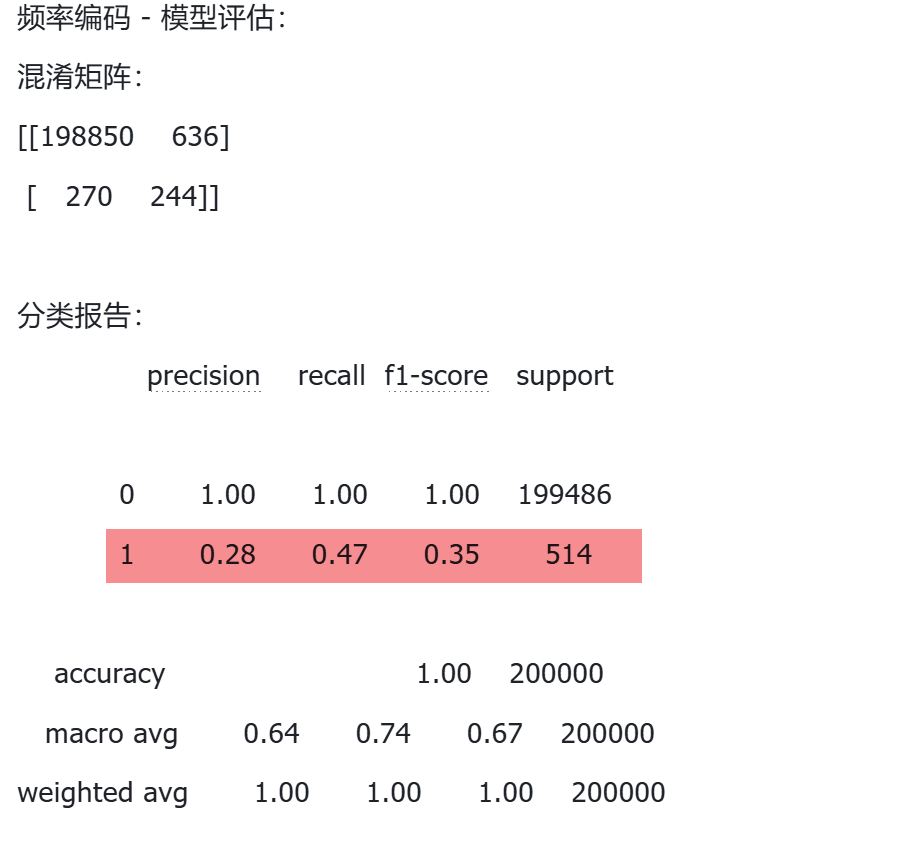
**评估指标：**

* F1-score：0.3444
* 召回率：0.4728
* ROC-AUC：0.9779

少数类（欺诈类）的

**精准度（precision）**——模型识别到的欺诈中确实为欺诈的比例：0.28

**召回率（recall）** ——模型识别到的欺诈占真实欺诈的比例：0.47



可以看到两种编码方式的最优结果的精准度（precision）指标、召回率（recall）指标对于少数类（欺诈类）都均非常低，这对于一个欺诈交易识别软件来说是不可接受的，因此考虑第四种——使用神经网络编码

|  |
| --- |
| 精准度（precision）——模型识别到的欺诈中确实为欺诈的比例  召回率（recall） ——模型识别到的欺诈占真实欺诈的比例 |

5. **神经网络聚类编码实验**

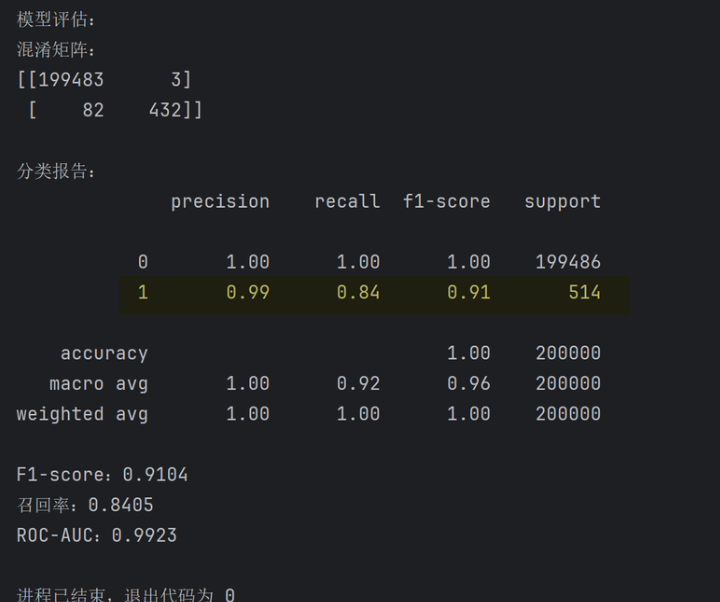
使用神经网络将 ID 映射为嵌入向量

在实际使用神经网络编码还有个问题，由于训练集、测试集数据不一样，测试集会遇到训练集没有见过的ID，此时无法编码，我采取的是以下处理方式：

* **最近邻插值法**：基于特征相似性找到最相似的商家，使用其嵌入值（首选方法）
* **默认嵌入值**：使用嵌入均值作为默认值-1（最近邻插入法失效，次选方法）



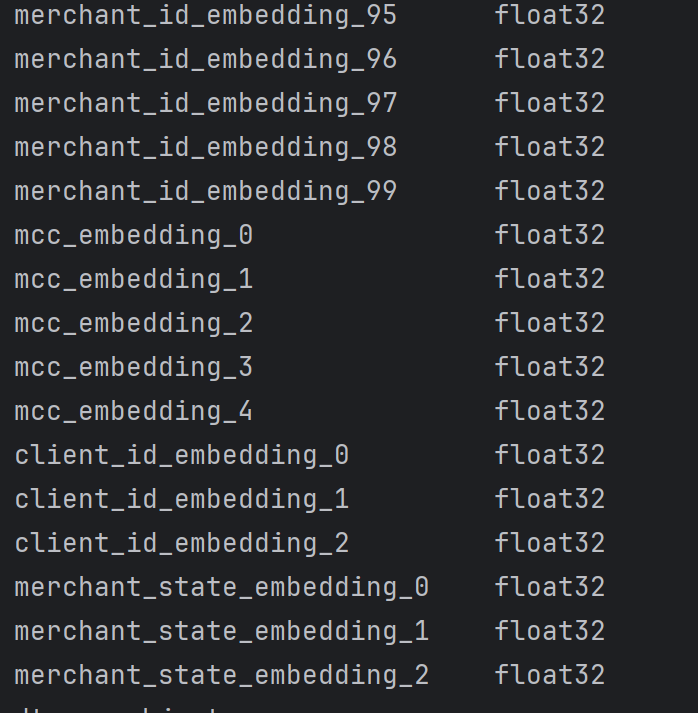
结果：



可以看到使用神经网络嵌入编码的少数类（欺诈类）精准度（precision）指标、召回率（recall）指标非常高，可见是一个很好的方向，开始调整超参数获得更优结果

6. **调整超参数**

* 增大数据量到700万训练集、100万测试集
* 使用**网格法**调整神经网络超参数判断每个特征向量的最佳降维（**聚类**）数
* 硬件加速：启用 NVIDIA CUDA 提升训练速度。
* 获得最优超参数——商家 ID 映射为嵌入向量（100 维）、用户ID降维为3维、mcc码（商品类别码）降为5维、商家所在州降维为3维



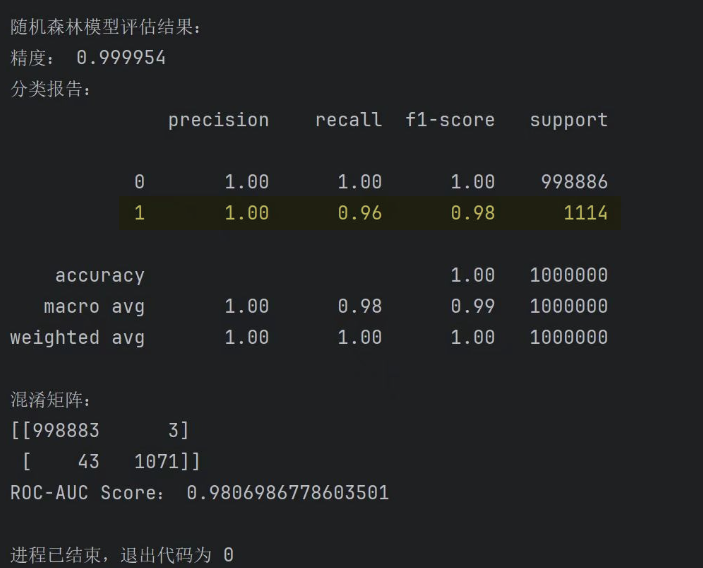
**实验结果**

**最终结果（700万训练集、100万测试集）**：

**精准率100%**：预测到的欺诈全部是真实的欺诈

**召回率96%**：有96%的真实欺诈被模型预测出来

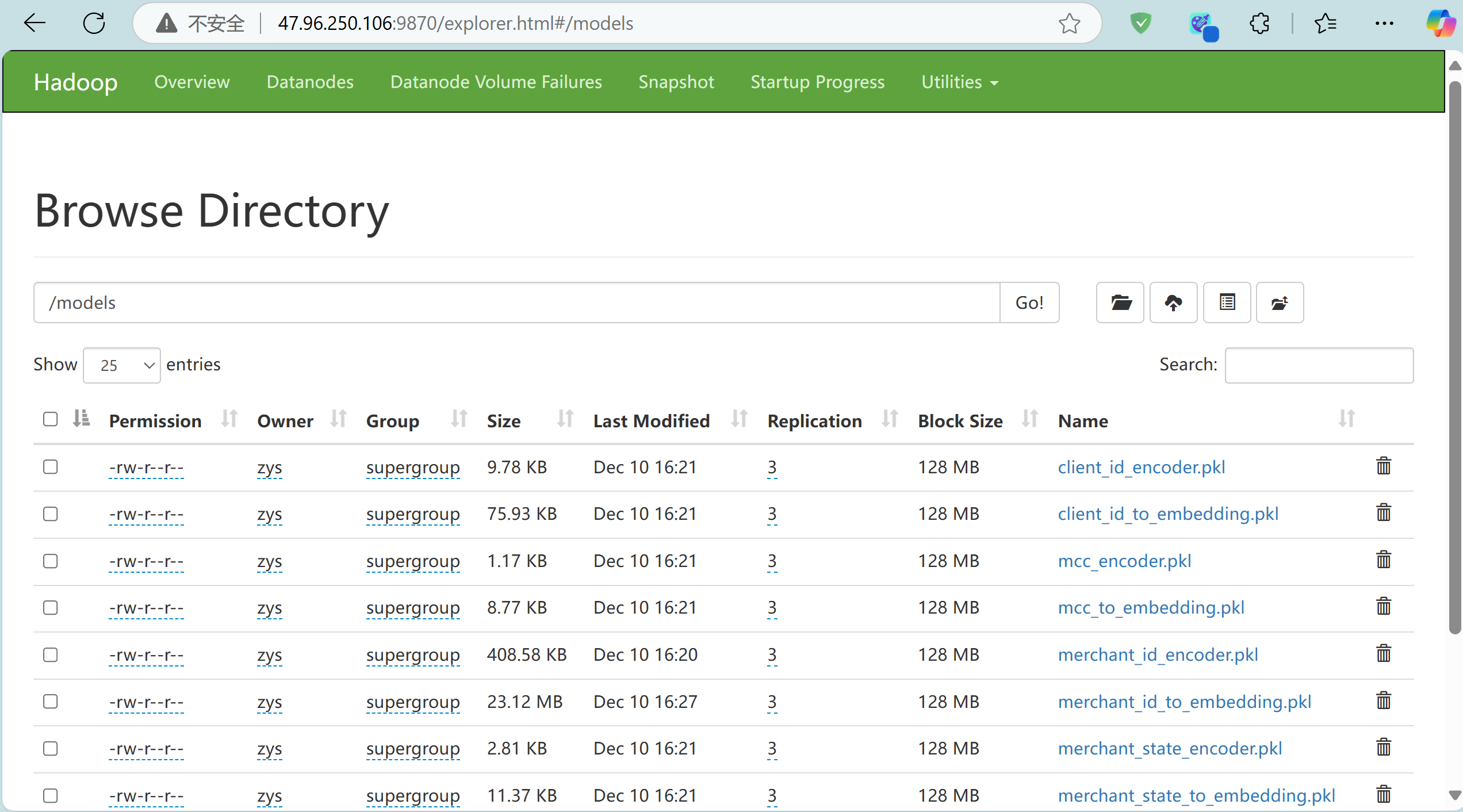
可以看到训练出了一个非常好的模型，此时可以进行生产环境的部署了



7. **模型部署到Spark**

* 训练代码与测试代码分离，模拟真实场景。
* 使用 PySpark 将测试代码嵌入 Spark 环境，提升分布式处理能力。
* 模型与编码文件上传至 HDFS，支持大规模数据预测。
* 实时预测：通过 Spring Boot 提供 API 服务，支持前端和数据库交互。

模型和编码文件存入HDFS



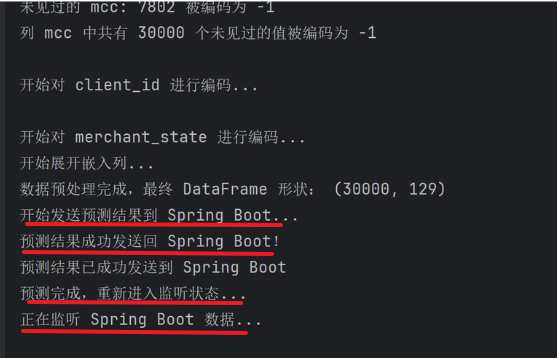
PySpark控制台：



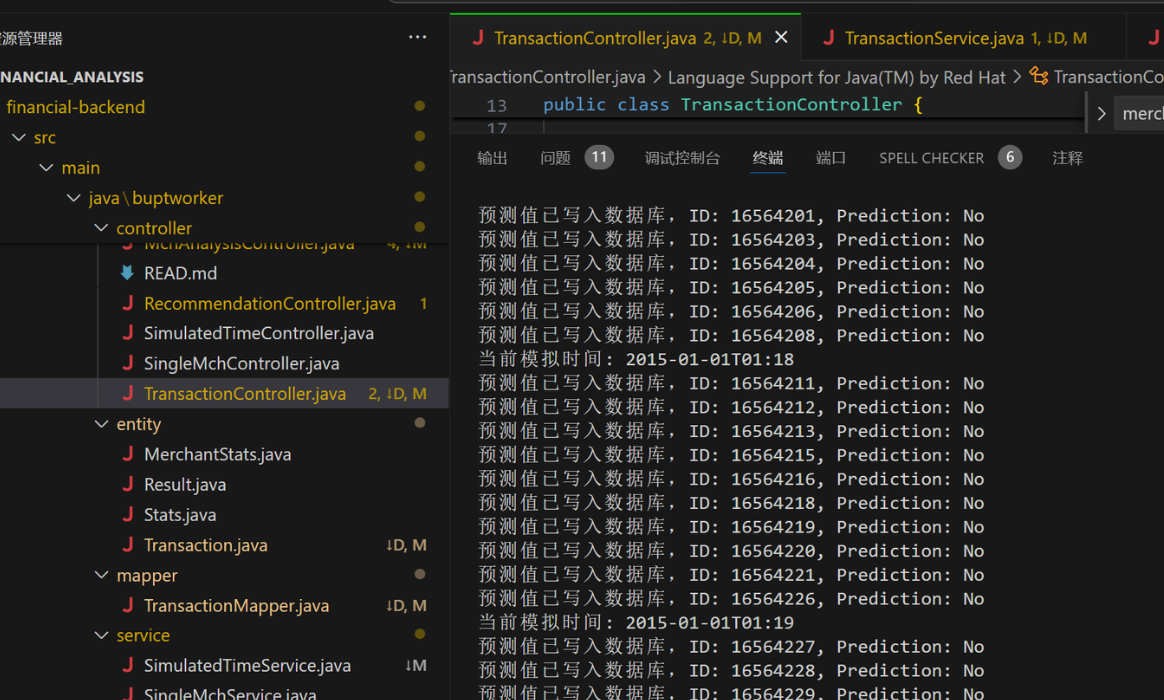
8. **连接后端Spring-boot：**

* 提供标准化接口，便于扩展和维护。
* 数据交互：通过 HTTP 协议连接后端**Spring-boot**，实时获取数据和返回预测结果。

运行在spark上的模型监听spring-boot端口获取数据进行预测并返回结果



后端接收数据，并存入数据库



9. **前后端连接**

* 后端不断向spark上的模型发送数据（模拟真实预测场景），并将预测的统计信息在前端展示
* 接受前端传来的预测请求，传递给模型进行预测，并将结果返回前端

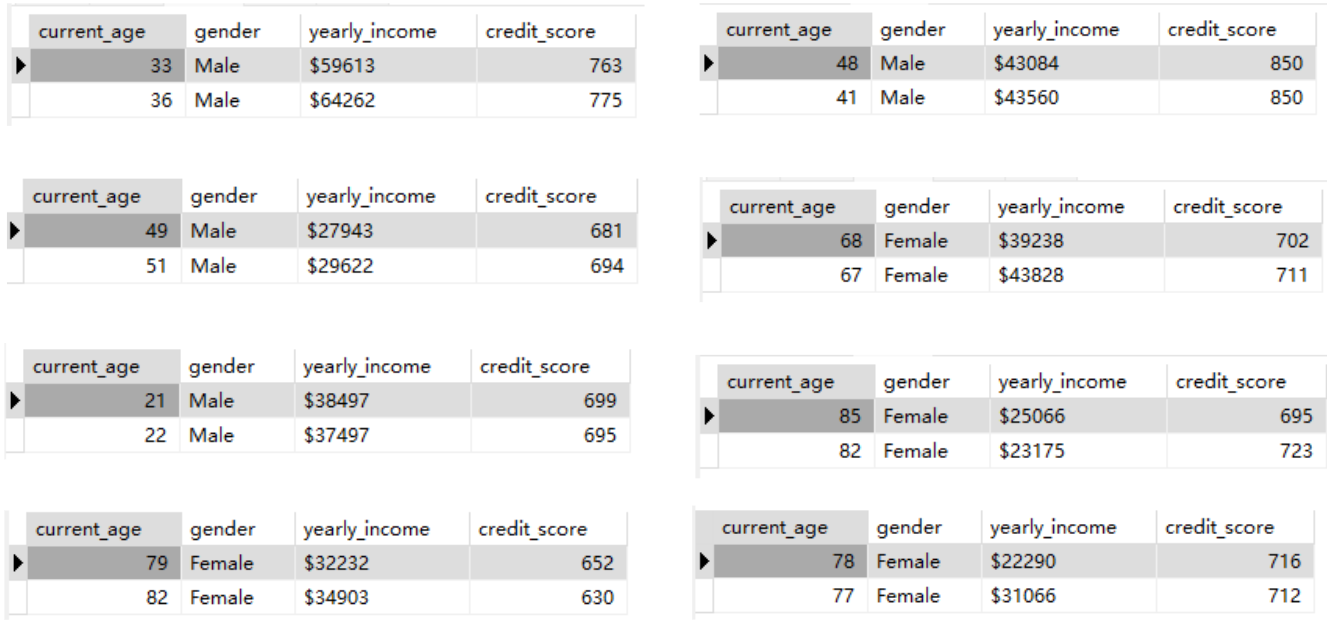




**5.3为用户推荐商家部分的运行过程及结果**

1. 查看用户相似度处理结果。

* 经过算法执行后输出相似度高于0.98的用户。
* 可以看到，两个用户在年龄，性别，年收入和信誉卡积分上都相差极小。



1. 查看商品相似度处理结果。

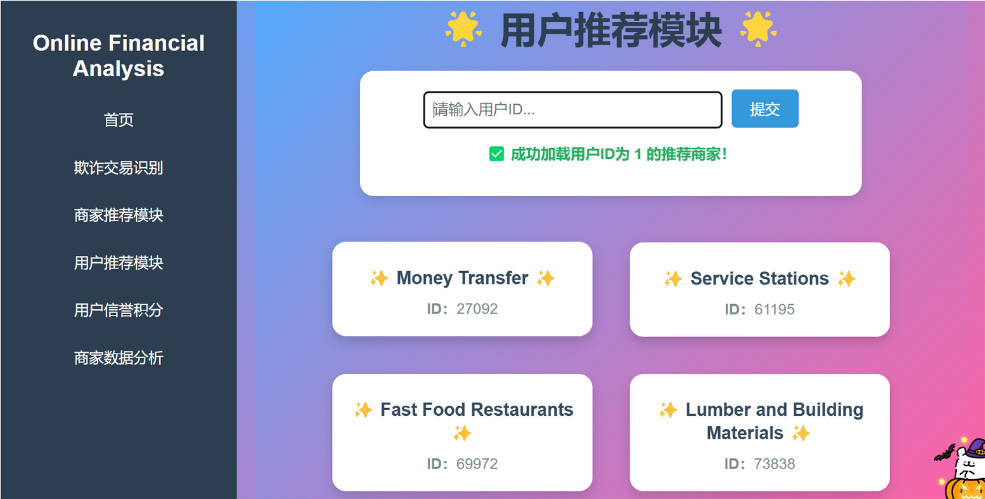
|  |  |
| --- | --- |
|  |  |

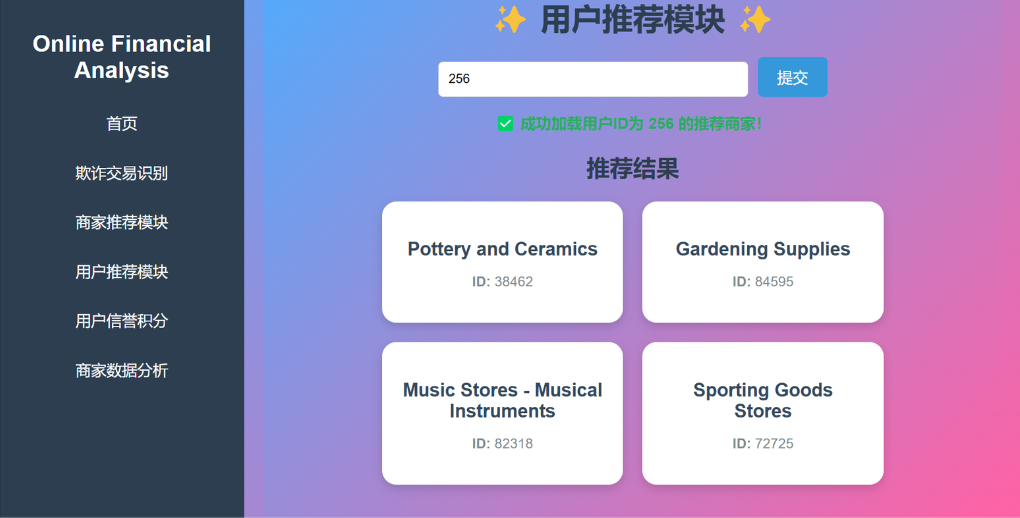
* 为了查看更加方便，我们将结果处理成中文输出。发现商品对都是高关联的物品。在现实中彼此有关系。



1. 查看前端处理结果。

* 输入用户id后，去数据库中检索这个用户。
* 左边两个是基于物品的协同过滤，右边两个是基于用户的协同过滤。





**5.4 商家推荐模块运行过程及结果**

* 打开界面，输入商家ID，显示对应的推荐用户ID和兴趣度。
* 如果输入了不存在的商家ID则给出提示。
* 结果如下：

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

**5.5 流失用户预测模块**

**实验过程：**

1. **定义流失用户**

由于该功能目的是分析流失用户数据并预测潜在的流失用户，但是我们所拥有的数据集中并不存在流失用户标签，而仅有交易记录数据，因此首先需要定义流失用户与潜在流失用户。

**流失用户：** 开始考虑到定义30天内无交易行为的用户为流失用户，但数据集中几乎不存在满足该要求的用户，因此定义在连续30天内交易次数小于某个阈值的用户为流失用户。

**潜在流失用户：**在未来30天内交易数量小于阈值（即满足流失用户要求），则当日该用户被定义为潜在流失用户。

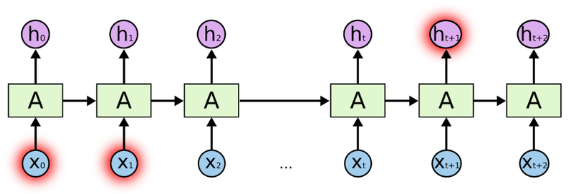
2. **数据选择与划分**

我们选择了2019-1-1至2019-9-30日的数据，其中1-1至8-31日数据作为训练集，9月份数据作为测试集。

3. **模型选择**

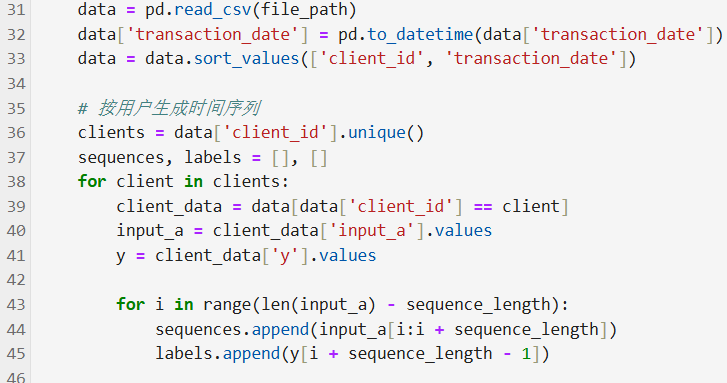
用户的交易行为是一个随时间产生的序列化信息，比较直观的想法是使用循环神经网络进行拟合。

这里选择LSTM模型，以用户当日的交易次数为输入x，用户未来30天的交易总数计算获得标签y（0/1）训练LSTM模型。

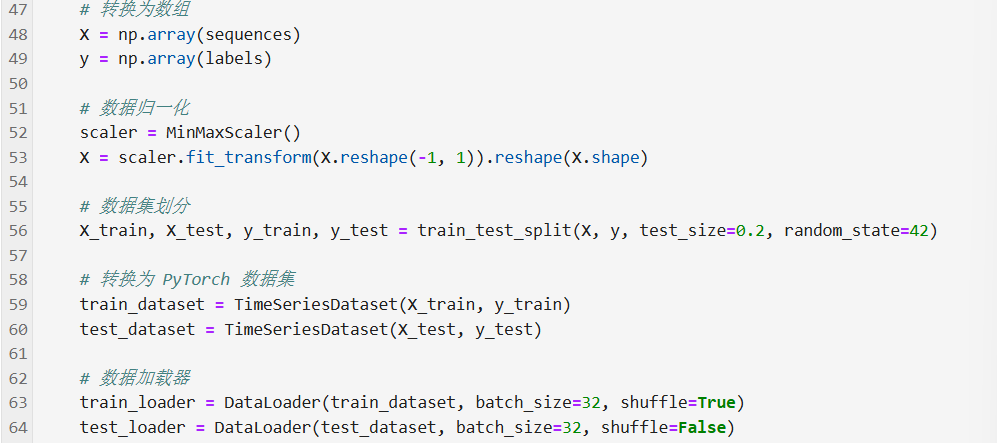


4. **数据预处理**

选择窗口大小为30，将连续30天的交易记录作为输入，最后一天的潜在流失标签作为输出训练模型。

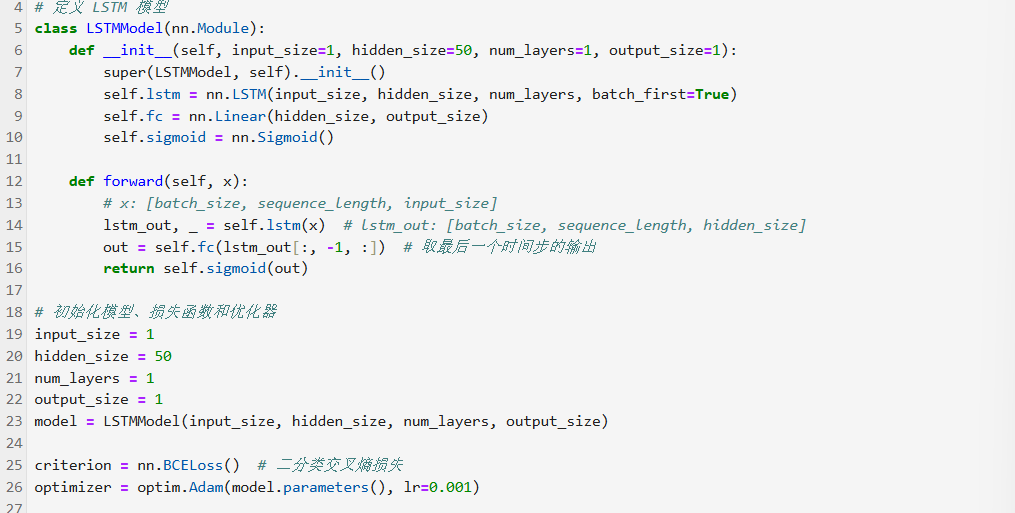


对数据进行归一化等预处理。

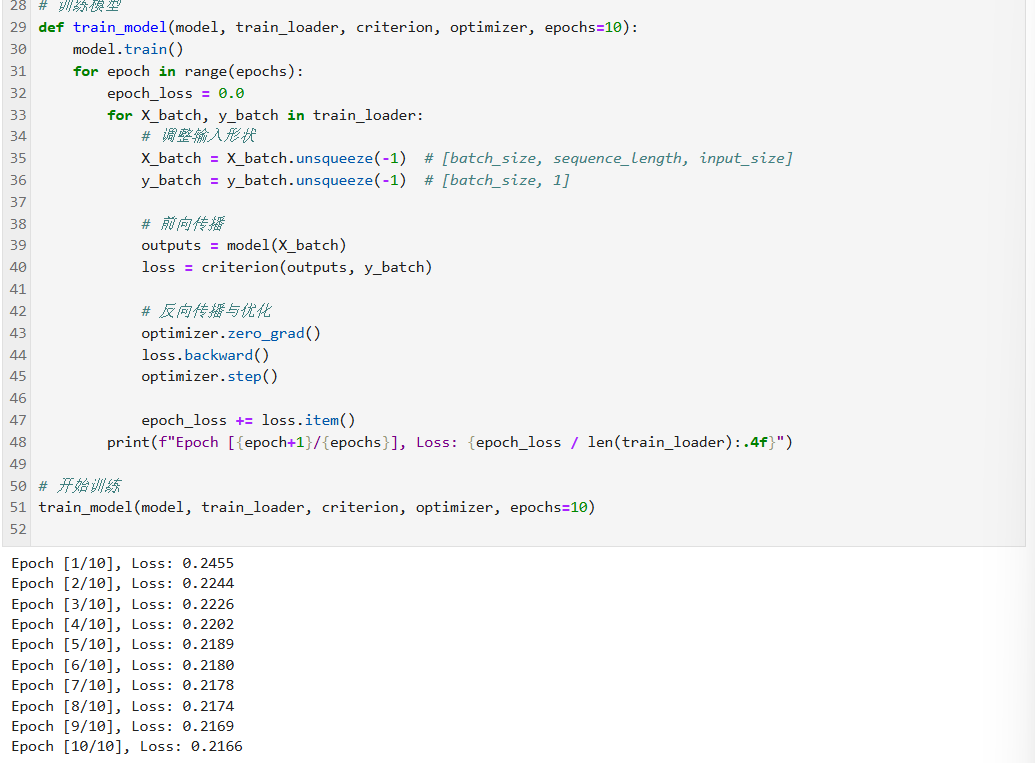


5. **模型定义与训练**

定义模型，这里使用单层的LSTM。



训练模型，epochs=10。



6. **模型评价**

我们在测试数据集（2019-9-1到2019-9-30之间的数据）上对模型进行评价，

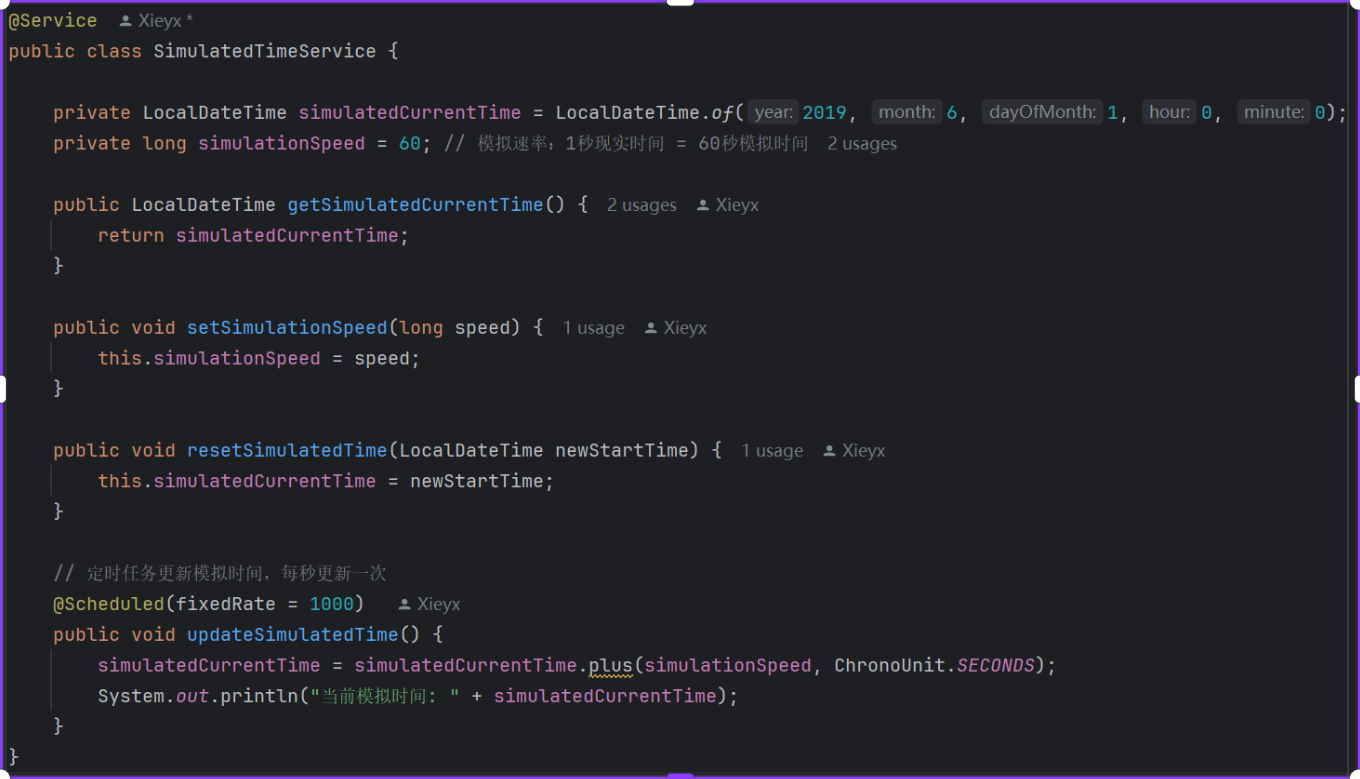


7. **模型部署应用**

由于模型预测依赖于当前时间，而我们所拥有的是一个历史数据集，因此首先需要对当前时间进行模拟。

我们设计了一个时间模拟服务，在后端服务器启动时该服务启动，默认启动时间为2019-6-1，速率1min/s。

使用该服务可以模拟一个伪当前时间，便于业务的使用。



模型对每日数据预测完成后保存到一张predictions表中，前后端工作较为简单，只需要从当日的predictions表中筛选出预测流失概率最高的用户并返回即可。

**实验结果**

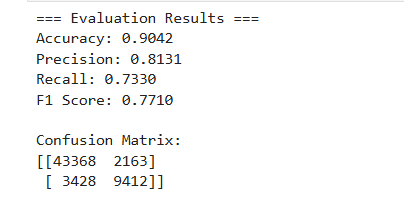
以下为模型测试指标和部署后的应用结果。

|  |  |
| --- | --- |
|  |  |

我还尝试了扩大模型，使用了双层LSTM进行训练，但是最终模型效果差距并不大，因此最终部署使用了较为轻量级的单层LSTM模型。

我认为可能的原因是流失用户并没有特别强的规律性，很可能在同样30天内交易数据分布相似，但未来30天的交易数据分布有很大差别，因此模型难以达到很高的准确率。





**六. 项目人员任务分配比率（给出各人的工作量百分比）：**

|  |  |  |
| --- | --- | --- |
| **成员** | **主要工作** | **工作量占比** |
| 高志强 |  | 20% |
| 周裕顺 |  | 20% |
| 华为 |  | 20% |
| 徐楚欣阳 |  | 20% |
| 谢又新 | LSTM模型实现流失用户分析与预测 | 20% |

**七. 项目心得及体会：**

**7.1 欺诈识别模块**

**实验心得**

1. **数据分析与特征选择**

* **样本分布不均衡**：欺诈交易占比极低（可能不到 1%），对模型构建提出了重大挑战。选择了目标编码、频率编码和神经网络嵌入等方法处理大规模 ID 特征，这种多阶段实验验证了编码方法对模型效果的显著影响。
* **强相关特征挖掘**：发现商家 ID、用户 ID、MCC 等与欺诈行为密切相关，表明特征工程是模型性能的关键，尤其是针对大规模标识符的处理。

1. **编码方式对比**

* **目标编码与频率编码**：传统统计方法可以在一定程度上改善模型效果，但对于复杂的大规模数据集，模型性能（尤其是少数类指标）仍然较低。
* **神经网络嵌入编码**：通过深度学习的嵌入向量，不仅捕获了 ID 的语义信息，还显著提升了模型对少数类的精准率和召回率。

1. **模型效果对比**

* **传统编码方法局限性**：目标编码和频率编码下的 F1-score、精准率和召回率均较低，表明这些方法难以有效利用数据中的高维离散信息。
* **神经网络编码的优势**：实验中，神经网络编码的精准率达到 100%，召回率达到 96%，表明嵌入向量能显著提升模型对欺诈交易的识别能力。

1. **超参数调优与硬件优化**

* 通过网格搜索调整神经网络的嵌入向量维度，最终找到不同 ID 特征的最佳降维数（例如，商家 ID 降维为 100）。同时，利用 GPU 加速显著提升了模型训练效率。

1. **模型部署**

* 将模型部署在 Spark 上并通过 Spring Boot 提供的 API 接口实现实时预测。通过分布式处理和高效的数据流架构，保证了系统在大规模数据下的稳定性和高效性。

**实验总结**

1. **模型性能**

* 实验最终训练出的模型在精准率和召回率上都达到了生产级别的要求（精准率 100%，召回率 96%），成功解决了欺诈交易识别中少数类样本检测的难题。
* 神经网络嵌入编码是整个实验的核心突破点，展现了深度学习方法在高维稀疏数据处理中的强大能力。

1. **编码方式选择**

* 独热编码无法应对大规模 ID 特征，目标编码和频率编码虽然简单，但效果有限。
* 神经网络嵌入编码在实际业务场景中展现了更强的泛化能力和业务适配能力，特别是在解决测试集中未见 ID 的问题时，通过最近邻插值法和默认嵌入值，保证了模型在生产环境的稳定性。

1. **实验方法**

* 数据集从小规模（100 万条）到大规模（800 万条）的逐步验证过程，保证了模型在不同数据量下的稳健性。
* 超参数调优（嵌入维度选择、树的数量和权重设置）进一步优化了模型性能。

1. **技术实现**

* Spark 的分布式计算框架有效解决了大规模数据的处理瓶颈，将训练好的模型嵌入 Spark 上运行，提升了预测效率。
* 使用 Spring Boot 搭建标准化接口，实现了模型与前后端的无缝对接，为实时预测提供了强有力的支持。

1. **生产环境部署**

* 模型和编码文件上传至 HDFS，结合 Spark 分布式预测能力，实现了对实时流数据的快速处理。
* 前后端通过 HTTP 接口交互，前端可以实时查看预测结果和统计信息，增强了系统的可用性和易用性。

**未来改进方向**

1. **数据采样与增强**

* 可以探索更加智能的过采样技术（如 SMOTE），进一步提高模型对少数类样本的召回率。

1. **实时学习能力**

* 引入在线学习或增量学习方法，随着数据的实时更新，动态优化模型性能。

1. **特征工程优化**

* 针对高频 ID（如商家 ID）进一步挖掘其他潜在关联特征（如交易时间模式、地区分布等），提升模型的细粒度识别能力。

1. **更复杂的模型架构**

* 尝试基于 Transformer 或 GNN（图神经网络）的模型架构，进一步挖掘特征之间的深层关联。

1. **监控与报警机制**

* 增强系统的可观测性（如引入模型性能监控、异常检测模块），提高系统在生产环境下的鲁棒性。

通过这次实验不仅学习了完整的欺诈识别模型开发流程，也认识到了深度学习与分布式计算在金融风控中的重要价值。这种基于大规模 ID 数据的嵌入编码方法，可推广到其他需要处理高维稀疏数据的业务场景中。

**7.5 流失用户预测**

在这次大数据实践课程的用户流失预测项目中，我深刻感受到理论知识与实际应用结合的重要性，同时在处理海量数据和模型应用过程中也积累了许多宝贵经验。以下是我的心得体会：

1. **大数据处理能力的锻炼**  
   本项目涉及 1400 万条交易记录，如何高效处理如此庞大的数据成为首要挑战。通过课程中学到的 SQL优化技术和数据分片处理方法，我成功将复杂的数据预处理任务拆解成多阶段、高效的子任务。这种基于分布式和分批处理思想的实践让我对大数据场景下的数据处理能力有了显著提升。
2. **时间序列分析和大数据特征提取**  
   在课程中学习的时间序列处理方法对本项目帮助很大。针对用户每日交易行为，我结合时间序列的规律性特点，设计了流失用户的预测逻辑（未来 30 天交易量小于 60 定义为流失）。这种特征提取方法结合了大数据分析的理论与实际场景需求，成功生成了高质量的输入数据。
3. **LSTM 模型在大数据预测中的应用**  
   该项目实践强调了大数据环境下的深度学习应用。在本项目中，我选择 LSTM 模型处理交易行为的时间依赖性问题。LSTM 能够捕获用户交易频次的变化趋势，从而更精准地预测用户流失风险。这一过程让我更加理解了深度学习在大数据分析中的价值，并提升了模型调优的能力。
4. **大数据工程化的完整流程**  
   课程实践过程中，不仅需要设计模型，还需要完成从数据预处理到模型落地的完整流程。在本项目中，我使用 MySQL 数据库存储预测结果，并结合 Python 脚本完成数据提取、模型加载与预测结果的存储。同时，预测结果通过接口与前端对接，展示了每日用户流失概率。这一从开发到部署的闭环过程，让我更深入理解了大数据应用系统的设计与实现。
5. **资源优化与效率提升的重要性**  
   本次实践让我意识到，在大数据场景下，资源优化和效率提升尤为关键。通过减少冗余计算、优化 SQL 查询、缩小数据加载范围等方法，我将预处理效率提高了 2 倍以上。这种优化思路不仅提升了程序运行速度，也为后续类似任务积累了宝贵经验。
6. **技术落地能力的提升**  
   本项目不仅完成了模型设计与训练，还完成了前后端部署。利用 MySQL 数据库存储预测结果，并通过接口展示，进一步提高了我对深度学习模型工程化落地的理解。
7. **团队协作与问题解决能力的提升**  
   在项目中，我与数据、模型和业务团队紧密协作，快速解决了标签设计、数据处理效率等问题。将业务需求转化为技术实现的过程，让我更加理解技术的实际价值。

通过大数据实践课程的学习，我掌握了大规模数据处理、深度学习模型应用以及系统工程化落地的关键技术。在这次用户流失预测项目中，我不仅验证了课程理论知识的实用性，还提升了自己的问题解决能力和大数据工程能力。这次实验既是理论知识的实践应用，也是我对大数据和机器学习领域更深刻的探索。

**八. 附录（附上项目其他文档，具体内容根据项目要求确定）**