**【P1】-【基于 PySpark 的分布式机器学习在葡萄牙银行定期存款认购预测】**

**实验结果与问题报告**

**成员：林悦 李育茵 许杉杉 李佳一**

**一、项目背景与目标**

**项目背景：**在竞争激烈的金融市场环境下，葡萄牙银行期望借助先进的技术手段精准预测客户对定期存款的认购行为，进而优化营销策略，提升营销效率，实现资源的合理配置。

**目标：**本项目旨在运用 PySpark 的分布式计算能力与机器学习算法构建预测系统，完成数据预处理、模型训练与优化、多种功能模块开发以及前端展示等工作，为银行提供全面的决策支持，辅助其制定更具针对性的营销策略并进行有效的风险管控。

**二、实验环境与工具**

1. **硬件环境**：
   * 处理器：4节点分布式服务器集群，每节点配备 1 核 CPU
   * 内存：每节点 1GB 内存，4 节点共 4GB
   * 存储：集群存储容量 30G
2. **软件环境**：
   * 操作系统：Linux Centos7
   * 编程语言：Python
   * 大数据处理平台：Hadoop、Spark
   * 依赖库：PySpark、NumPy、Matplotlib 等
3. **项目内容与步骤**

**测试数据集来源：**从葡萄牙银行电话营销活动记录中获取数据，形成包含客户基本信息、经济指标、营销活动相关信息以及客户认购定期存款标签的数据集，总量达 4 万条记录。

**（一）模型训练**

**1.数据预处理：**

1.1使用 PySpark 对数据进行清洗：

* + - * 重命名包含特殊字符的列（将点号替换为下划线）
      * 将分类变量中的 "unknown" 值替换为缺失值（None）
      * 对数值型特征使用均值填充缺失值
      * 对分类型特征使用众数填充缺失值
      * 过滤年龄异常值（保留 18-100 岁之间的数据）

1.2开展特征工程：

* + - * 对分类特征进行编码（StringIndexer + OneHotEncoder）
      * 对数值特征进行标准化处理（StandardScaler）
      * 移除不适合用于预测的 duration 列
      * 使用 VectorAssembler 合并所有特征
      * 对目标变量进行编码（将 'yes/no' 转换为 0/1）
      * 构建自动化预处理 Pipeline 并保存模型
      * 将数据按 7:1.5:1.5 比例划分为训练集、验证集和测试集
      * 将处理后的数据保存为 Parquet 格式以便后续建模使用

**2. 模型构建与训练**

小组成员分工训练四个模型，在经过对比后，采用随机森林分类器进行模型训练，通过设置超参数网格和交叉验证调整超参数，优化模型性能。

**（二）功能模块开发**

* ****客户分层与统计****：基于模型预测概率对客户分层，并统计各分层客户数量。
* ****风险预警****：依据设定规则评估客户分层结果数据的风险等级。
* ****特征重要性分析****：提取并分析模型特征重要性，绘制条形图。
* ****营销活动模拟****：根据客户分层结果模拟不同营销活动策略，对比关键指标
* ****客户认购预测****：通过用户输入客户信息进行认购预测，输出预测结果（是 / 否）、客户潜力、风险预警和营销建议。

**四、实验结果**

1.**数据预处理结果**：

* 缺失值填充后，各数值列缺失率为 0%；分类变量中 “unknown” 替换率达 100%；“age” 列异常值过滤后在 18 - 100 范围内。分类特征编码成功，数值特征标准化后均值接近 0、标准差接近 1。训练集、验证集和测试集记录数量分别为 28915 条、6091 条和 6177 条，符合预期比例。
* 数据质量显著提高，为后续分析奠定了基础。

**2.模型训练结果:**

* 完成模型训练（含超参数调优）耗时约 15 分钟。最佳 numTrees 为 100，最佳 maxDepth 为 10。验证集 AUC 为 0.79，准确率为 0.91，精确率为 0.90，召回率为 0.91，F1 分数为 0.89；测试集 AUC 为 0.83，准确率为 0.89，精确率为 0.87，召回率为 0.89，F1 分数为 0.86。
* 模型训练稳定，超参数调优有效，在验证集和测试集上预测效果良好。

**3. 功能模块实现结果:**

* ****客户分层与统计****：高潜力客户占比 1%，中潜力客户占比 3%，低潜力客户占比 96%。分层逻辑有效，利于银行识别不同潜力客户群体，支撑精准营销。
* ****风险预警****：清晰划分各风险等级客户占比，风险分布可视化图表直观展示风险状况，能有效识别风险客户，辅助银行风险管理。
* ****特征重要性分析****：明确 “poutcome\_encorded\_failure” 等特征对客户认购决策影响较大，条形图直观呈现差异，助力银行理解客户行为，优化营销特征选择。
* ****营销活动模拟****：模拟结果显示，针对高潜力客户营销转化率达 1.0000；高潜力和中潜力客户营销转化率为 1.0000；随机抽样 30% 客户营销转化率仅 0.0416。为银行制定营销策略提供数据支持，优化资源分配。
* ****客户认购预测****：能准确处理用户输入信息，输出预测结果、客户潜力、风险预警和营销建议。

**五、遇到的问题与解决方案**

1. **数据集数据正负样本分布不均问题**：

* ****问题描述****：原始数据集中，客户购买定期存款的正样本和负样本数量差异较大，这种不平衡会使模型在训练过程中倾向于多数类，导致对少数类的预测能力不足，影响模型整体的泛化性能和预测准确性。例如，在训练随机森林模型时，模型可能会更多地学习到未购买客户的特征模式，而对购买客户的特征模式学习不够充分，从而在实际预测购买客户时出现较多误判。
* ****解决方案****：采用 SMOTE算法对训练数据中的少数类进行过采样。在预处理过程中，将训练数据转换为适合 SMOTE 处理的格式，通过 SMOTE 算法生成新的少数类样本，增加购买客户样本数量，使正负样本比例更加均衡。同时，调整模型评估指标，如采用 F1 分数、AUC 等更适合不均衡数据的指标进行模型评估。

1. **模型训练效率问题**

* ****问题描述****：项目初期使用 2 个节点的 Spark 集群进行模型训练，随着数据规模扩大和模型复杂度增加，训练时间显著变长。主要原因包括数据在节点间分布不均衡，导致部分节点负载过高成为性能瓶颈，节点数量少使得 Executor 数量不足，无法充分利用 CPU 核心资源，并行计算能力受限。
* ****解决方案****：将 Spark 集群节点从 2 个增加到 4 个，提升整体计算能力。设置合适的并行任务数和 Shuffle 分区数，提升计算效率 根据集群规模动态调整 Shuffle 分区数，减少数据在节点间的传输量。

**3. Pipeline 模型缺失导致预测失败问题**

* ****问题描述****：项目前期开发预测功能时，未意识到保存 Pipeline 模型的重要性，仅聚焦于随机森林模型训练。实际预测新客户数据时，发现缺少 Pipeline 模型致使数据预处理环节无法正常进行，因为手动预处理无法精准还原训练数据时的完整预处理流程，导致数据特征与模型期望不匹配，最终预测功能无法正常运作。
* ****解决方案****：在预处理阶段添加保存 Pipeline 模型的步骤，确保路径正确。修改预测功能，优先加载保存的 Pipeline 模型进行预处理。完善异常处理，记录详细错误信息。更新文档，说明 Pipeline 模型的重要性及操作流程。

****4. 用户界面交互性问题****

* ****问题描述****：在 Web 应用程序的用户界面中，如index.html、predict\_form.html和prediction\_result.html等页面，存在交互性不足的问题。用户在操作过程中，反馈机制不够完善，操作流程不够便捷，导致用户体验不佳。例如，在填写预测表单时，用户输入错误数据后，没有及时的错误提示信息，用户需要提交表单后才能发现问题，增加了操作成本。
* ****解决方案****：在前端页面中增加实时验证功能。利用 JavaScript 等前端技术，在用户输入数据时，实时检查数据的格式和范围是否符合要求。如果输入错误，立即弹出提示框告知用户错误原因和正确的输入格式。优化页面布局，根据用户操作习惯，合理安排表单字段和按钮的位置，提高操作的便捷性。同时，增加操作引导和提示信息，对于复杂的操作流程，提供详细的步骤说明，帮助用户更好地使用系统。

**六、总结与计划**

* **总结：**

本项目依托 PySpark 搭建分布式机器学习系统，专注于葡萄牙银行定期存款认购预测任务，并取得了阶段性成果。在数据处理方面，有效的预处理步骤显著提升了数据质量，为后续的模型训练和预测奠定了坚实基础。基于随机森林模型的预测和分析流程，在客户行为预测上达到了较高的准确率。

系统具备多个核心功能模块，涵盖客户行为预测、客户分层、风险预警以及营销策略模拟等。通过收集客户的个人信息、财务状况、联系历史等多维度数据，系统能够精准分析客户特征，为不同潜力和风险级别的客户量身定制营销策略建议，为银行的营销活动和风险管理提供了有力支持。同时，分布式架构在数据处理和模型训练过程中展现出高效性，充分发挥了并行计算的优势。

然而，目前项目代码存在明显的结构问题。所有功能实现的代码都集中在 app.py 文件中，随着系统功能的持续拓展，该文件变得日益庞大和复杂，严重影响了代码的可读性和可维护性。这不仅增加了开发人员理解和修改代码的难度，也对项目的长期发展构成了潜在威胁。

* **计划:**

为了优化代码结构，提高开发和维护效率，我们计划对现有代码进行拆分重构。具体将按照功能模块对代码进行分离，将不同功能的代码逻辑封装到独立的 Python 文件中，从而实现代码的模块化管理。初步规划对以下 5 个核心功能模块进行分离操作：

1.客户分层模块：把客户分层相关的代码封装成 customer\_segmentation.py 文件。此模块将根据客户的预测结果，对客户进行分层，并提供分层统计和可视化功能。

2.风险预警模块：将风险预警相关的代码封装成risk\_warning.py 文件。该模块将定义风险预警规则，标记风险客户，并进行风险统计和可视化。

3.特征重要性模块：将特征重要性分析相关的代码封装到 characteristic\_influence.py 文件中。该模块将加载已训练的模型，获取特征重要性数组，从数据的元数据中提取特征名称，对特征重要性进行排序并输出前 15 个最重要的特征及其重要性得分，同时绘制特征重要性条形图。

4.营销策略模拟模块：把营销策略模拟相关的代码封装到 marketing\_simulation.py 文件中。此模块将模拟不同的营销策略，计算每个策略的总人数、预计转化人数和转化率，并进行可视化展示。

5.客户行为预测模块：提取客户行为预测相关的代码，封装到 customer\_prediction.py 文件中。该模块将负责处理客户数据的输入、模型预测以及结果输出等功能。

完成代码拆分后，app.py 文件将主要承担调用其他 Python 文件中功能函数的职责。通过这种方式，代码结构将更加清晰，各个模块的职责也将更加明确，便于开发人员进行代码的维护和扩展。同时，模块化的设计也有助于提高代码的复用性和可测试性，为项目的长期发展提供有力保障。