華中科技大學

大数据推荐系统实验报告

院（系、所） 电子信息与通信学院

班　　级 提高 2101 班

专　　业 电子信息工程（提高班）

姓　　名 杨筠松

学　　号 U202115980

指 导 老 师 陈建文

2024 年 6 月 7 日

一、实验环境

软件环境：**wsl2** (Windows Subsystem for Linux)

Linux 版本：Ubuntu 22.04.4LTS

```
(base) ➜ ~ cat /proc/version
Linux version 5.15.146.1-microsoft-standard-WSL2 (root@65c757a075e2
) (gcc (GCC) 11.2.0, GNU ld (GNU Binutils) 2.37) #1 SMP Thu Jan 11
04:09:03 UTC 2024
(base) ➜ ~ uname -a
Linux Hale 5.15.146.1-microsoft-standard-WSL2 #1 SMP Thu Jan 11 04:
09:03 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
(base) ➜ ~ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.4 LTS
Release:        22.04
Codename:       jammy
```

Python 版本：python3.7.16

```
(py37) ➜ code git:(main) X python --version
Python 3.7.16
```

二、实验内容

本实验基于阿里巴巴集团主办的天池大数据竞赛中的一项赛题，即天猫推荐算法大挑战。该赛题要求参赛者根据用户在天猫三个月内的行为日志，预测他们在未来一个月内对品牌商品的购买行为。这是一个典型的推荐系统问题，涉及到大数据处理、用户行为分析和预测模型构建。

实验的主要目标是使学生掌握推荐系统的基本工作流程，包括数据加载、特征工程、模型训练和推荐效果评估。通过实践，学生将学习如何实现和优化推荐算法，提高推荐系统的准确率和召回率。

实验方法和步骤

1. **数据准备和加载：**
    - 使用的数据集包含约 100,000 条天猫用户的行为记录，涉及点击、购买、收藏和加入购物车等行为。
    - 数据加载部分涉及从 CSV 文件中读取数据，并对数据进行初步处理。

2. **推荐方法：**
    - **基于规则的推荐**：此方法直接根据用户的行为日志生成推荐，如推荐用户最近一个月内互动次数较多的品牌。
    - **基于逻辑回归的推荐**：使用逻辑回归模型基于用户的历史行为预测其对品牌的购买概率。
3. **特征工程：**
    - 开发和选择影响用户购买决策的特征，如用户对品牌的点击次数、购买次数、收藏次数、加入购物车次数以及最近的互动日期等。
4. **模型训练与推荐实施：**
    - 使用训练数据集构建模型，并应用模型于测试集生成购买推荐。
    - 推荐结果的格式为每个用户推荐的品牌列表。
5. **推荐效果评估：**
    - 使用准确率、召回率和 F1 得分来评估推荐效果。

三、实验过程说明及问题

实验过程

1. **数据加载与预处理：**
    - 从 data.csv 文件中加载用户行为数据，该数据包括用户 ID、品牌 ID、行为类型、月份和日期。
    - 清洗数据，包括处理缺失值、异常值和加密字段。
2. **特征提取：**
    - 根据用户行为日志，提取用户对品牌的交互特征，如点击次数、购买次数、收藏次数和加入购物车次数。
    - 引入时间衰减因子以赋予近期行为更高的权重。
3. **模型训练与验证：**
    - 使用逻辑回归模型，基于提取的特征和前两个月的数据训练模型。
    - 将模型应用于后两个月的数据以生成推荐结果。
4. **推荐效果评估：**
    - 评估推荐结果，计算准确率、召回率和 F1 得分。
    - 分析模型表现并识别改进领域。

遇到的问题

- 数据质量问题：数据集中存在缺失值和异常值，需要进行适当的数据清洗。
- 特征工程复杂性：难以确定哪些特征对推荐效果影响显著。
- 模型过拟合：模型在训练集上表现良好，但在未见过的数据上表现不佳。

四、解决方案

已解决的问题：

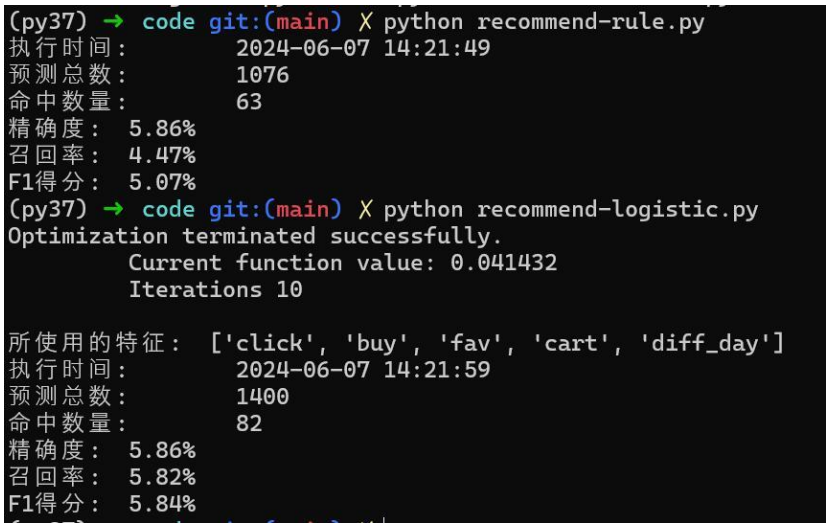
1. **数据预处理：**
    - 实施了数据清洗步骤，包括删除或填充缺失值和处理异常值。
    - 对字段进行了解密和格式转换，以便于分析。
2. **特征工程：**
    - 通过试验和错误方法，识别出对模型影响最大的特征。
    - 引入了时间衰减因子，以增加模型对近期行为的敏感度。

未解决的问题：

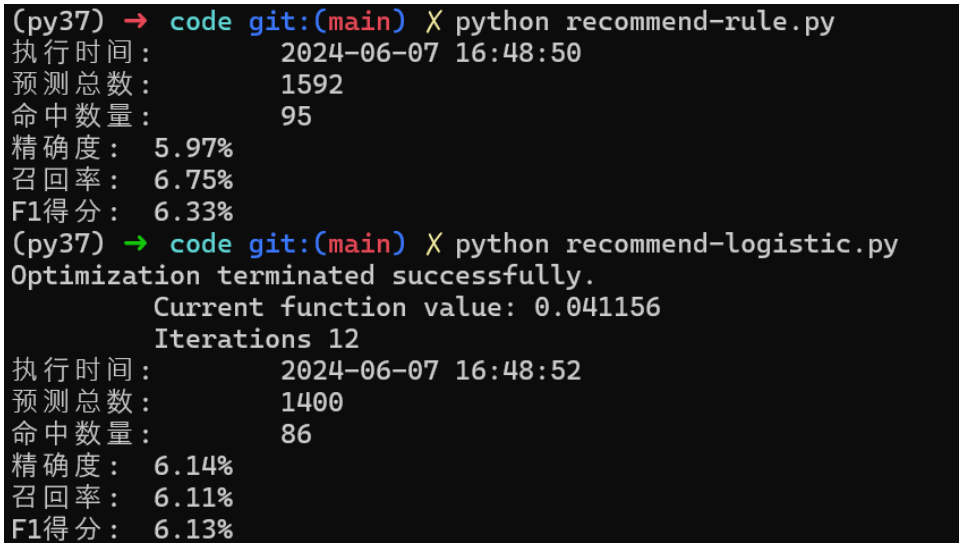
- 特征选择仍有改进空间，可能存在更有效的特征组合未被探索。
- 模型在极端数据情况下的稳定性和鲁棒性有待提高。

五、实验结果

未进行任何操作之前的结果：



进行了特征选择和规则推荐后的结果



六、程序源代码

```
#
coding = utf - 8# 特征选取与计算

from copy
import copy
from tmall
import *
import math

# 引入时间衰减因子函数
def time_decay_factor(days_since_last_action, decay_rate = 0.95):
    return decay_rate ** days_since_last_action

# 特征计算# 基于逻辑回归的推荐需要划分训练集与预测集# 将前 2 个月的交互划分为训练集 train# 将后 2 个月的交互划分为# 通过参数 classify 进行判断， 不处理与当前类型不符合的数据。
def generateFeature(classify, data):
    F = {}

item = {
    'click': 0,
    #点击次数 'buy': 0,
    #购买次数 'fav': 0,
    #加入收藏夹次数 'cart': 0,
    #加入购物车次数 'diff_day': 1000,
    #相差天数初始化 'total_activity': 0,
    #活跃度 'time_decay_click': 0,
    #点击时间衰减特征 'time_decay_buy': 0,
    #购买时间衰减特征 'time_decay_fav': 0,
    #收藏时间衰减特征 'time_decay_cart': 0,
    #车时间衰减特征
}

feature_name = ['click', 'buy', 'fav', 'cart', 'diff_day',
'total_activity',
                    'time_decay_click', 'time_decay_buy',
'time_decay_fav', 'time_decay_cart']

for uid, bid, action_type, month, day in data:
    if classify != getClassify(month, day):
    continue
```

```
F.setdefault(uid, {})
F[uid].setdefault(bid, copy(item))

e = F[uid][bid]

diff_day = getDiffDayByClass(classify, (month, day))
if diff_day < e['diff_day']:
    e['diff_day'] = diff_day

# 基础特征计算并引入时间衰减
if action_type == 0:
    e['click'] += 1
e['time_decay_click'] += time_decay_factor(diff_day)
elif action_type == 1:
    e['buy'] += 1
e['time_decay_buy'] += time_decay_factor(diff_day)
elif action_type == 2:
    e['fav'] += 1
e['time_decay_fav'] += time_decay_factor(diff_day)
elif action_type == 3:
    e['cart'] += 1
e['time_decay_cart'] += time_decay_factor(diff_day)

# 综合得分计算， 考虑时间衰减
for uid, bid_list in F.items():
    for bid, e in bid_list.items():
    e['total_activity'] = (e['fav'] * e['time_decay_fav'] + e['cart'] *
e['time_decay_cart'])

return F, feature_name



#
coding = utf - 8# 自定义的推荐规则

from copy
import copy
from tmall
import getDiffDay
```

```
# 推荐规则# 函数可分为两部分# 1. 计算用户特征# 2. 根据规则进行筛选## 参数
data: 数组， 数组元素为(user_id, brand_id, action_type, month, day)# 返回
值 R: 数组， 数组元素为(
    user_id, brand_id)
def getRecommendByRule(data): #设置推荐阈值， 这个值可能需要根据实验结果进行
调整
threshold = 0.5
F = {}#
存储用户特征
R = []# 存储推荐结果

# 所有要进行统计的特征， 在这里进行声明并赋予初始值
item = {
    'click': 0,
    #点击次数 'buy': 0,
    #购买次数 'fav': 0,
    #加入收藏夹次数 'cart': 0,
    #加入购物车次数 'diff_day': 1000,
    #因为是要推测下一个月的购买情况# 显然在最近一段时间有交互的， 购买可能性越
大# 因此将最后一次交互的相差天数也作为一个特征# 如我们推测 7 月 15 - 8 月 15 这
一个月的购买情况， 用户在 7 月 8 号跟 7 月 12 号均有交互记录# 则 diff_day 为 3（ 取
最近的 7 月 12， 计算跟 7 月 15 的相差天数） 'total_activity': 0,
    #用户活跃度， 总活动次数
}


# 1. 计算用户特征
for uid, bid, action_type, month, day in data: #初始化
F.setdefault(uid, {})
F[uid].setdefault(bid, copy(item))

# 新建一个引用， 简化代码
e = F[uid][bid]

# 基础特征计算
if action_type == 0:
    e['click'] += 1
e['total_activity'] += 0.4# 用户对该商品的活跃度
elif action_type == 1:
    e['buy'] += 1
e['total_activity'] += 3.5
elif action_type == 2:
    e['fav'] += 1
e['total_activity'] += 1.5
```

```
elif action_type == 3:
    e['cart'] += 1
e['total_activity'] += 2.5

# 时间特征
diff_day = getDiffDay((month, day), (7, 15))
if diff_day < e['diff_day']:
    e['diff_day'] = diff_day


# 2. 根据特征进行筛选
for uid, bid_list in F.items():
    for bid, e in bid_list.items(): #综合得分计算， 可以根据需要调整各项的权
score = (e['click'] * 0.4 + e['buy'] * 3.5 + e['fav'] * 1.5 + e['cart'] * 2.5) * (1 / (e['diff_day'] + 1))# 综合得分高于某个阈值， 并且 diff_day 小于 30， 并且用户活跃度大于 1 .5
if score > threshold and e['diff_day'] < 30 and e['total_activity'] > 1.5:
    R.append((uid, bid))

return R
```