****

**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**

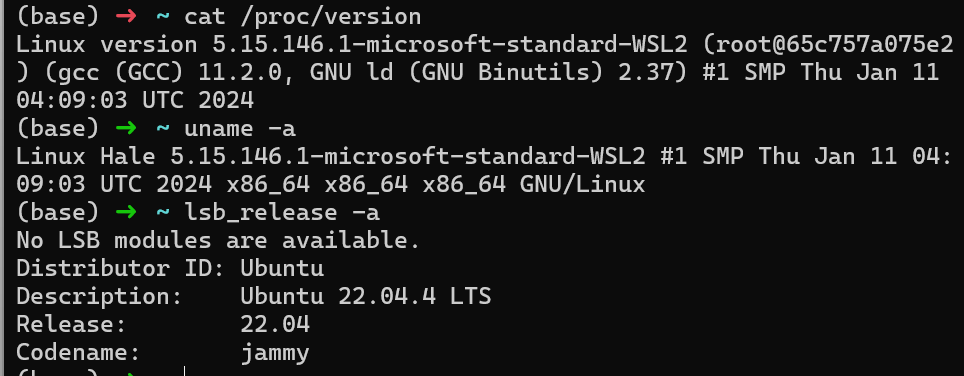
**指 导 老 师 陈建文**

**2024年 6 月 7 日**

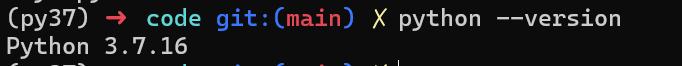
一、实验环境

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

Linux版本：Ubuntu 22.04.4LTS



Python版本：python3.7.16



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：

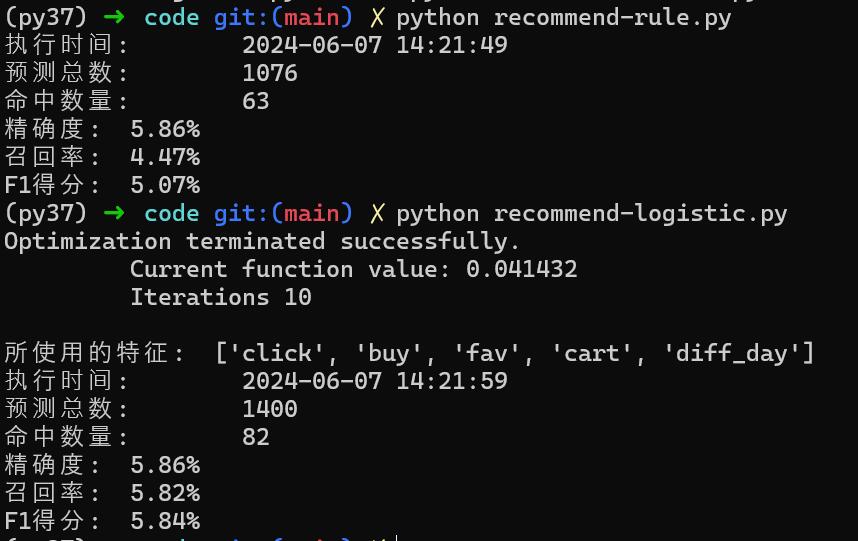
1. **数据预处理**：
   * 实施了数据清洗步骤，包括删除或填充缺失值和处理异常值。
   * 对字段进行了解密和格式转换，以便于分析。
2. **特征工程**：
   * 通过试验和错误方法，识别出对模型影响最大的特征。
   * 引入了时间衰减因子，以增加模型对近期行为的敏感度。

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

文本

描述已自动生成

六、程序源代码

**User\_feature.py**

1. *# coding=utf-8*
2. *# 特征选取与计算*
3. from copy import copy
4. from tmall import \*
5. *# 特征计算*
6. *# 基于逻辑回归的推荐需要划分训练集与预测集*
7. *# 将前2个月的交互划分为训练集 train*
8. *# 将后2个月的交互划分为*
9. *# 通过参数classify进行判断，不处理与当前类型不符合的数据。*
10. def generateFeature(classify, data):
11. F = {}
12. user\_brand\_interaction = {}
13. item = {
14. 'click': 0,  *# 点击次数*
15. 'buy': 0,  *# 购买次数*
16. 'fav': 0,  *# 加入收藏夹次数*
17. 'cart': 0,  *# 加入购物车次数*
18. 'diff\_day': 1000,  *# 相差天数初始化*
19. 'click\_decay': 0,  *# 点击时间衰减特征*
20. 'buy\_decay': 0,  *# 购买时间衰减特征*
21. 'fav\_decay': 0,  *# 收藏时间衰减特征*
22. 'cart\_decay': 0,  *# 加入购物车时间衰减特征*
23. 'interaction\_frequency': 0,  *# 用户与品牌的交互频率*
24. 'historical\_activity': 0,  *# 用户历史活跃度*
25. }
27. feature\_names = ['click', 'buy', 'fav', 'cart', 'diff\_day', 'click\_decay',
28. 'buy\_decay', 'fav\_decay', 'cart\_decay', 'interaction\_frequency', 'historical\_activity']
29. for uid, bid, action\_type, month, day in data:
30. if classify != getClassify(month, day):
31. continue
32. F.setdefault(uid, {})
33. F[uid].setdefault(bid, copy(item))
34. user\_brand\_interaction.setdefault(uid, set()).add(bid)
35. e = F[uid][bid]
36. diff\_day = getDiffDayByClass(classify, (month, day))
37. if diff\_day < e['diff\_day']:
38. e['diff\_day'] = diff\_day
39. *# 基础特征计算并引入时间衰减*
40. delay = (1.0 / (1 + diff\_day))
41. if action\_type == 0:
42. e['click'] += 1
43. e['click\_decay'] += delay
44. elif action\_type == 1:
45. e['buy'] += 1
46. e['buy\_decay'] += delay
47. elif action\_type == 2:
48. e['fav'] += 1
49. e['fav\_decay'] += delay
50. elif action\_type == 3:
51. e['cart'] += 1
52. e['cart\_decay'] += delay
53. *# 计算额外的特征*
54. for uid, bid\_list in F.items():
55. for bid, e in bid\_list.items():
56. e['interaction\_frequency'] = len(user\_brand\_interaction[uid])
57. e['historical\_activity'] = e['click'] + e['buy'] + e['fav'] + e['cart']
58. return F, feature\_names

**User\_rule.py**

1. *# coding=utf-8*
2. *# 自定义的推荐规则*
3. from copy import copy
4. from tmall import getDiffDay
5. *# 推荐规则*
6. *# 函数可分为两部分*
7. *# 1. 计算用户特征*
8. *# 2. 根据规则进行筛选*
9. *#*
10. *# 参数 data: 数组，数组元素为 (user\_id, brand\_id, action\_type, month, day)*
11. *# 返回值 R : 数组，数组元素为 (user\_id, brand\_id)*
12. def getRecommendByRule(data):
13. F = {}  *# 存储用户特征*
14. R = []  *# 存储推荐结果*
15. *# 所有要进行统计的特征，在这里进行声明并赋予初始值*
16. item = {
17. 'click': 0,  *# 点击次数*
18. 'buy': 0,  *# 购买次数*
19. 'fav': 0,  *# 加入收藏夹次数*
20. 'cart': 0,  *# 加入购物车次数*
21. 'diff\_day': 1000, *# 因为是要推测下一个月的购买情况*
22. *# 显然在最近一段时间有交互的，购买可能性越大*
23. *# 因此将最后一次交互的相差天数也作为一个特征*
24. *# 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*
25. *# 则diff\_day为3（取最近的7月12，计算跟7月15的相差天数）*
26. 'score': 0,  *# 用户质量进行打分*
27. 'buy\_last': 1000,  *# 最后一次购买距离现在的天数*
28. }
30. weight = {
31. 'click': 1.0,
32. 'buy': 9.0,
33. 'fav': 3.5,
34. 'cart': 4.0,
35. }
36. factor = 0.95
37. *# 1. 计算用户特征*
38. for uid, bid, action\_type, month, day in data:
39. *# 初始化*
40. F.setdefault(uid, {})
41. F[uid].setdefault(bid, copy(item))
42. *# 新建一个引用，简化代码*
43. e = F[uid][bid]
44. *# 基础特征计算*
45. if action\_type == 0:
46. e['click'] += 1
47. elif action\_type == 1:
48. e['buy'] += 1
49. elif action\_type == 2:
50. e['fav'] += 1
51. elif action\_type == 3:
52. e['cart'] += 1
53. *# 时间特征*
54. diff\_day = getDiffDay((month, day), (7, 15))
55. if diff\_day < e['diff\_day']:
56. e['diff\_day'] = diff\_day
57. if action\_type == 1:
58. e['buy\_last'] = min(e['buy\_last'], diff\_day)
59. *# 2. 根据特征进行筛选*
60. for uid, bid\_list in F.items():
61. for bid, e in bid\_list.items():
63. *# 给用户质量进行打分*
64. score = e['click'] \* weight['click'] + \
65. e['buy'] \* weight['buy'] + \
66. e['fav'] \* weight['fav'] + \
67. e['cart'] \* weight['cart']
68. *# 加入时间延迟因子*
69. time\_score = score \* factor \*\* e['diff\_day']
71. if time\_score > 6.5  and score > 9.0:
72. R.append((uid, bid))
73. return R