第一章 业务分析

1.1.1 智能医疗系统中的业务数据处理流程设计

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1. 数据采集
# 从本地文件中读取数据  2分
data = pd.read_csv("vehicle_traffic_data.csv")
print("数据采集完成，已加载到DataFrame中")

# 打印数据的前5条记录 2分
print(data.head())
```

```
数据采集完成，已加载到DataFrame中
   VehicleID DriverName  Age Gender  Speed  TravelDistance   TravelTime  \
0          1   Driver_1   62   Male     17             242   854.117647
1          2   Driver_2   65   Male    128             438   205.312500
2          3   Driver_3   18   Male    167             792   284.550898
3          4   Driver_4   21   Male     38             999  1577.368421
4          5   Driver_5   21   Male    193             364   113.160622

  TrafficEvent
0       Normal
1     Accident
2    Breakdown
3  Traffic Jam
4    Breakdown
```

```
4    Breakdown
```

[4]:
```
# 2. 数据清洗与预处理
# 处理缺失值（删除）  2分
data = data.dropna()

# 数据类型转换
data["Age"] = data["Age"].astype(int)       #Age数据类型转换为int 1分
data["Speed"] = data["Speed"].astype(float)     #Speed数据类型转换为float 1分
data["TravelDistance"] = data["TravelDistance"].astype(float)     #TravelDistance数据类型转换为float 1分
data["TravelTime"] = data["TravelTime"].astype(float)     #TravelTime数据类型转换为float 1分

# 处理异常值  2分
data = data[(data["Age"].between(18, 70))  &
            (data["Speed"].between(0, 200)) &
            (data["TravelDistance"].between(1, 1000)) &
            (data["TravelTime"].between(1, 1440))]

# 保存清洗后的数据  1分
data.to_csv('cleaned_vehicle_traffic_data.csv', index=False)
print("数据清洗完成，已保存为 'cleaned_vehicle_traffic_data.csv'")
```

```
数据清洗完成，已保存为 'cleaned_vehicle_traffic_data.csv'
```

```
# 3. 数据合理性审核
# 审核字段合理性 1分
unreasonable_data = data[~((data["Age"].between(18, 70)) &
                           (data["Speed"].between(0, 200)) &
                           (data["TravelDistance"].between(1, 1000)) &
                           (data["TravelTime"].between(1, 1440)))]
print("不合理的数据:\n", unreasonable_data)

# 4. 数据统计
# 统计每种交通事件的发生次数  2分
traffic_event_counts = data['TrafficEvent'].value_counts()
print("每种交通事件的发生次数:\n", traffic_event_counts)

# 统计不同性别的平均车速、行驶距离和行驶时间  2分
gender_stats = data.groupby('Gender')[['Speed', 'TravelDistance', 'TravelTime']].mean()
print("不同性别的平均车速、行驶距离和行驶时间:\n", gender_stats)

# 统计不同年龄段的驾驶员数  5分
age_bins = [18, 26, 36, 46, 56, 66, np.inf]
age_labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '65+']
data['AgeGroup'] = pd.cut(data['Age'],bins=age_bins,labels=age_labels, right=False)
age_group_counts = data['AgeGroup'].value_counts()
print("不同年龄段的驾驶员数:\n", age_group_counts)
```

```
不合理的数据:
 Empty DataFrame
Columns: [VehicleID, DriverName, Age, Gender, Speed, TravelDistance, TravelTime, Traffic
```

1.1.2 智能农业系统中的业务数据采集和处理流程设计

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 读取数据集 2分
data = pd.read_csv("sensor_data.csv")
```

[7]:
```
# 1. 传感器数据统计
# 对传感器类型进行分组，并计算每个组的数据数量和平均值 3分
sensor_stats = data.groupby("SensorType")['Value'].agg(["count","mean"])
# 输出结果
print("传感器数据数量和平均值:")
print(sensor_stats)
```

```
传感器数据数量和平均值:
             count       mean
SensorType
Humidity      2065  48.978302
Light         1950  49.439011
SoilMoisture  1951  49.325119
SoilPH        2029  49.865460
Temperature   2005  49.847153
```

[11]:
```
# 2. 按位置统计温度和湿度数据
# 筛选出温度和湿度数据，然后按位置和传感器类型分组，计算每个组的平均值 2分
location_stats = data[data['SensorType'].isin(['Temperature', 'Humidity'])].groupby(['Location','SensorType'])['Value'].mean().unstack()
# 输出结果
print("每个位置的温度和湿度数据平均值:")
print(location_stats)
```

```
每个位置的温度和湿度数据平均值:
SensorType   Humidity  Temperature
Location
Field1      49.036008    50.482617
Field2      47.379411    49.874282
Field3      49.583266    48.761804
Field4      49.955498    50.241608
```

[27]:
```
# 3. 数据清洗和异常值处理
# 标记异常值 3分
data['is_abnormal'] = np.where(
    ((data['SensorType'] == 'Temperature') & ((data['Value'] < -10) | (data['Value'] > 50))) |
    ((data['SensorType'] == 'Humidity') & ((data['Value'] < 0) | (data['Value'] > 100))),
    True, False
)
# 输出异常值数量 2分
print("异常值数量:", data['is_abnormal'].sum())
# 填补缺失值
# 使用前向填充和后向填充的方法填补缺失值 4分
data['Value'].fillna(method='ffill', inplace=True)
data['Value'].fillna(method='bfill', inplace=True)
# 保存清洗后的数据
# 删除用于标记异常值的列，并将清洗后的数据保存到新的CSV文件中 4分
cleaned_data = data.drop(columns=['is_abnormal'])
cleaned_data.to_csv('1.csv', index=True)
print("数据清洗完成，已保存为 'cleaned_sensor_data.csv'")
```

```
异常值数量: 1011
数据清洗完成，已保存为 'cleaned_sensor_data.csv'
```

1.1.3 金融机构信用评估系统中的业务数据审核流程设计

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 读取数据集
data = pd.read_csv('credit_data.csv')
```

[3]:
```
# 1. 数据完整性审核
missing_values = data.isnull().sum()    #数据缺失值统计 2分 # NULL True False
duplicate_values = data.duplicated().sum()     #数据重复值统计 2分
# 输出结果
print("缺失值统计:")
print(missing_values)
print("重复值统计:")
print(duplicate_values)
```

```
缺失值统计:
CustomerID            0
Name                  0
Age                   1
Income                1
LoanAmount            0
LoanTerm              0
CreditScore           0
Default               0
TransactionHistory    0
dtype: int64
重复值统计:
```

[9]:
```
# 2. 数据合理性审核
data['is_age_valid'] = data['Age'].between(18, 70)              #Age数据的合理性审核 2分
data['is_income_valid'] = data['Income'] > 2000                 #Income数据的合理性审核 2分
data['is_loan_amount_valid'] = data['LoanAmount'] < (data['Income']  * 5)      #LoanAmount数据的合理性审核 2分
data['is_credit_score_valid'] = data['CreditScore'].between(300, 850)   #CreditScore数据的合理性审核 2分
# 合理性检查结果
validity_checks = data[['is_age_valid', 'is_income_valid', 'is_loan_amount_valid', 'is_credit_score_valid']].all(axis=1)
data['is_valid'] = validity_checks
# 输出结果
print("数据合理性检查:")
print(data[['is_age_valid', 'is_income_valid', 'is_loan_amount_valid', 'is_credit_score_valid', 'is_valid']].describe())
```

```
数据合理性检查:
       is_age_valid is_income_valid is_loan_amount_valid  \
count          1000            1000                 1000
unique            2               2                    2
top            True            True                 True
freq            999             999                  796

       is_credit_score_valid is_valid
count                   1000     1000
unique                     1        2
top                     True     True
freq                    1000      795
```

:
```
# 3. 数据清洗和异常值处理
# 标记不合理数据
invalid_rows = data[~data['is_valid']]
# 删除不合理数据行
cleaned_data = data[data['is_valid']]
# 删除标记列
cleaned_data = cleaned_data.drop(columns=['is_age_valid', 'is_income_valid', 'is_loan_amount_valid', 'is_credit_score_valid', 'is_valid'])
# 保存清洗后的数据
cleaned_data.to_csv("cleaned_credit_data.csv", index=False)
print("数据清洗完成，已保存为 'cleaned_credit_data.csv'")
```

```
数据清洗完成，已保存为 'cleaned_credit_data.csv'
```

1.1.4 电商平台用户行为分析系统的数据采集与处理流程设计

[11]:
```
import pandas
import numpy as np
import matplotlib.pyplot as plt

# 1. 数据采集
# 从本地文件中读取数据  2分
data =  pandas.read_csv("user_behavior_data.csv")
print("数据采集完成，已加载到DataFrame中")

# 打印数据的前5条记录  2分
print(data.head())
```

```
数据采集完成，已加载到DataFrame中
   UserID UserName  Age  Gender    Location   LastLogin  PurchaseAmount  \
0       1   User_1   62  Female  Location_1  2023-06-10             118
1       2   User_2   65  Female  Location_2  2023-08-14             466
2       3   User_3   18    Male  Location_3  2023-02-17             869
3       4   User_4   21  Female  Location_4  2023-03-14             486
4       5   User_5   21    Male  Location_5  2023-07-26             753

  PurchaseCategory  ReviewScore LoginFrequency
0         Clothing            3        Monthly
1      Electronics            4         Weekly
2    Home & Garden            3         Weekly
3            Books            2         Weekly
4    Home & Garden            1        Monthly
```

[13]:
```
# 2. 数据清洗与预处理
# 处理缺失值（删除）  2分
data = data.dropna()

# 数据类型转换
data['Age'] = data['Age'].astype(int)   # Age数据类型转换为int 2分 '60'
data['PurchaseAmount'] = data['PurchaseAmount'].astype(float) # PurchaseAmount数据类型转换为float  2分
data['ReviewScore'] = data['ReviewScore'].astype(int)   # ReviewScore数据类型转换为int 2分

# 处理异常值  2分
data = data[(data['Age'].between(18, 70)) &
            (data['PurchaseAmount'] > 0) &
            (data['ReviewScore'].between(1, 5))]

# 数据标准化
data['PurchaseAmount'] = (data['PurchaseAmount'] - data['PurchaseAmount'].mean()) / data['PurchaseAmount'].std()  # PurchaseAmount数据标准化 2分
data['ReviewScore'] = (data['ReviewScore'] - data['ReviewScore'].mean()) / data['ReviewScore'].std()        # ReviewScore数据标准化 2分

# 保存清洗后的数据  1分
data.to_csv('cleaned_user_behavior_data.csv', index=False)
print("数据清洗完成，已保存为 'cleaned_user_behavior_data.csv'")
```

```
数据清洗完成，已保存为 'cleaned_user_behavior_data.csv'
```

[15]:
```
# 3. 数据统计
```

[19]:
```python
# 3. 数据统计
# 统计每个购买类别的用户数 2分
purchase_category_counts = data['PurchaseCategory'].value_counts()
print("每个购买类别的用户数:\n", purchase_category_counts)

# 统计不同性别的平均购买金额 2分
gender_purchase_amount_mean = data.groupby("Gender")['PurchaseAmount'].mean()
print("不同性别的平均购买金额:\n", gender_purchase_amount_mean)

# 统计不同年龄段的用户数 2分
bins = [18, 26, 36, 46, 56, 66, np.inf]
labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '65+']
data['AgeGroup'] = pandas.cut(data["Age"],bins=bins,labels=labels, right=False)
age_group_counts = data['AgeGroup'].value_counts().sort_index()
print("不同年龄段的用户数:\n", age_group_counts)
```

```
每个购买类别的用户数:
 PurchaseCategory
Clothing         214
Electronics      213
Home & Garden    203
Food             197
Books            173
Name: count, dtype: int64
不同性别的平均购买金额:
 Gender
Female   -0.001367
```

1.1.5 智能交通系统的数据采集、处理和审核流程设计

```
]: import pandas as pd
   import numpy as np
   import matplotlib.pyplot as plt

   # 1. 数据采集
   # 从本地文件中读取数据  2分
   data = pd.read_csv("vehicle_traffic_data.csv")
   print("数据采集完成，已加载到DataFrame中")

   # 打印数据的前5条记录 2分
   print(data.head())
```

```
数据采集完成，已加载到DataFrame中
   VehicleID DriverName  Age Gender  Speed  TravelDistance   TravelTime  \
0          1   Driver_1   62   Male     17             242   854.117647
1          2   Driver_2   65   Male    128             438   205.312500
2          3   Driver_3   18   Male    167             792   284.550898
3          4   Driver_4   21   Male     38             999  1577.368421
4          5   Driver_5   21   Male    193             364   113.160622

  TrafficEvent
0       Normal
1     Accident
2    Breakdown
3  Traffic Jam
4    Breakdown
```

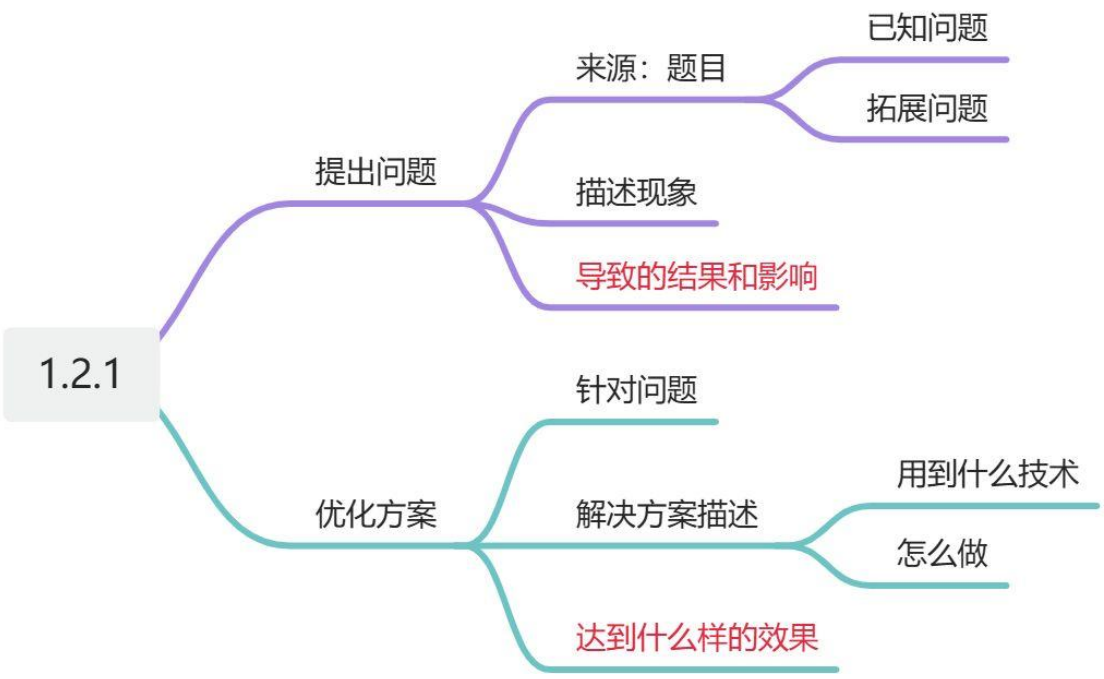
```
|: # 2. 数据清洗与预处理dat
   # 处理缺失值（删除）  2分
   data = data.dropna()

   # 数据类型转换
   data["Age"] = data["Age"].astype(int)       #Age数据类型转换为int 1分
   data["Speed"] = data["Speed"].astype(float)     #Speed数据类型转换为float 1分
   data["TravelDistance"] = data["TravelDistance"].astype(float)     #TravelDistance数据类型转换为float 1分
   data["TravelTime"] = data["TravelTime"].astype(float)     #TravelTime数据类型转换为float 1分

   # 处理异常值  2分
   data = data[(data["Age"].between(18, 70))  &
               (data["Speed"].between(0, 200)) &
               (data["TravelDistance"].between(1, 1000)) &
               (data["TravelTime"].between(1, 1440))]

   # 保存清洗后的数据  1分
   data.to_csv('cleaned_vehicle_traffic_data.csv', index=False)
   print("数据清洗完成，已保存为 'cleaned_vehicle_traffic_data.csv'")
```

```
数据清洗完成，已保存为 'cleaned_vehicle_traffic_data.csv'
```

[9]:
```
# 3. 数据合理性审核
# 审核字段合理性 1分
unreasonable_data = data[~((data["Age"].between(18, 70)) &
                           (data["Speed"].between(0, 200)) &
                           (data["TravelDistance"].between(1, 1000)) &
                           (data["TravelTime"].between(1, 1440)))]



print("不合理的数据:\n", unreasonable_data)

# 4. 数据统计
# 统计每种交通事件的发生次数  2分
traffic_event_counts = data["TrafficEvent"].value_counts()
print("每种交通事件的发生次数:\n", traffic_event_counts)

# 统计不同性别的平均车速、行驶距离和行驶时间  2分
gender_stats = data.groupby("Gender")[['Speed', 'TravelDistance', 'TravelTime']].mean()
print("不同性别的平均车速、行驶距离和行驶时间:\n", gender_stats)

# 统计不同年龄段的驾驶员数  5分
age_bins = [18, 26, 36, 46, 56, 66, np.inf]
age_labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '65+']
data['AgeGroup'] = pd.cut(data["Age"],bins=age_bins,labels=age_labels, right=False)
age_group_counts = data['AgeGroup'].value_counts()
print("不同年龄段的驾驶员数:\n", age_group_counts)
```

```
不合理的数据:
 Empty DataFrame
Columns: [VehicleID, DriverName, Age, Gender, Speed, TravelDistance, TravelTime, TrafficEvent]
Index: []
每种交通事件的发生次数:
 TrafficEvent
Accident       229
```

1.2.1 顾客评价情感识别业务模块效果优化



请勿修改答题卷，在指定单元格内填写答案

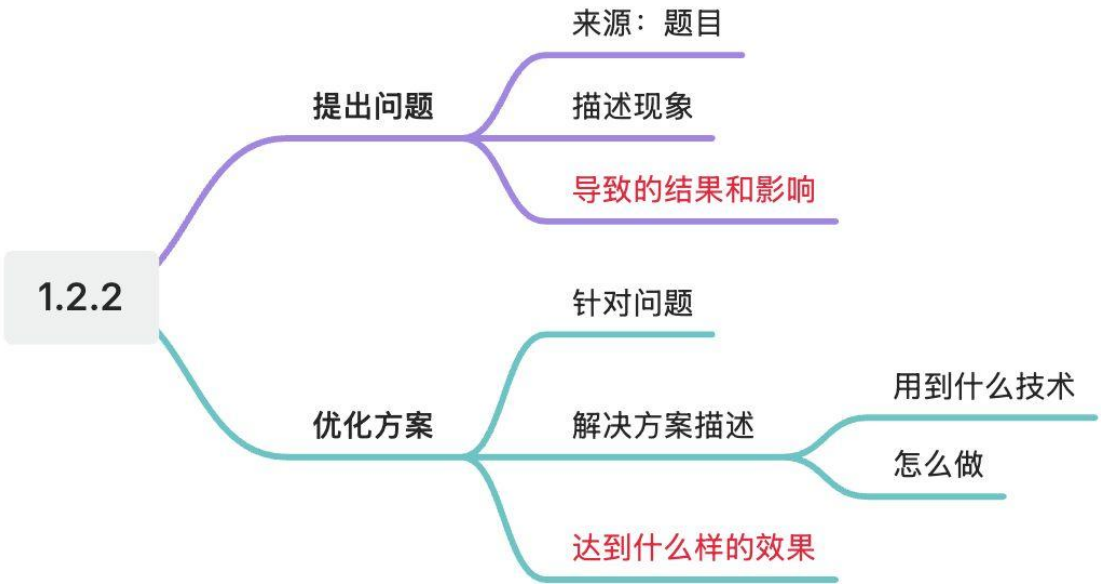
1. 2. 1-1

1、情感识别准确性不高
用户指出情感识别结果和用户实际情况出现偏差，导致对顾客评价的分析不够准确，影响后续的决策。
2、响应速度慢
用户在使用的过程中，等待时间过长，需要很久才能获取识别结果，导致用户失去耐心，降低了使用效率。
3、用户界面不友好
用户界面设计过于负责，不够直观，导致用户的学习成本和使用难度上升，影响用户使用。
4、缺乏定制化服务
无法满足特定用户群体的需求，无法根据实际情况进行调整，缺乏个性化设置。

1. 2. 1-2

1、提高识别准确率
技术升级，引入先进的情感识别算法和技术，提高识别的准确率和效率，为下一步的优化奠定技术基础。
2、加快响应速度
升级网络设备，对现有的业务流程进行优化，减少不必要的环节和步骤，减少响应时间，提高响应速度。
3、用户界面改进
对页面进行重新设计，设计更为直观友好的用户界面，提升页面的美观度和易用性。
4、增加个性化功能
调研更多的客户群体，累计更多的需求数据，建设完善的数据资产，为算法模型的训练提供数据支持。

1.2.2 老年人健康监测与管理服务业务模块效果优化



请勿修改答题卷，在指定单元格内填写答案

1. 2. 2-1

一、数据准确性不高
表现：心率监测数据波动大、误差高，无法准确反映老年人健康状况。
影响：可能导致医生和护理人员误判健康状态，出现误诊或治疗延误。
技术原因：传感器质量差、设备未校准、数据传输不稳定等因素共同造成采集数据失真。

二、异常预警响应慢

表现：心率异常发生后，系统反应滞后，预警机制不及时。
影响：容易错过干预的“黄金时间”，危及老年人生命安全。
技术原因：现有算法识别能力不足、系统流程响应链过长。

三、用户操作复杂

表现：设备操作步骤多、界面设计不友好，老年人使用困难，容易出错。
影响：降低老年人对心率监测系统的使用意愿和频率，最终影响长期健康监测的效果。
流程原因：交互设计未充分考虑老年人认知与行为特征，系统引导不清晰。

1. 2. 2-2

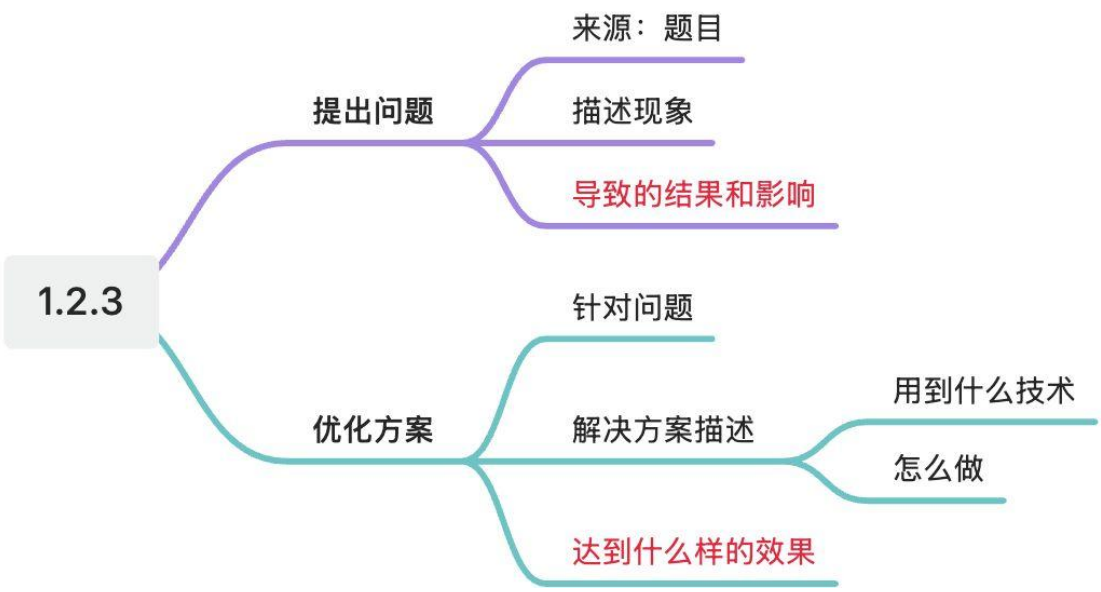
一、优化方向
通过使用高精度传感器和智能算法提升数据准确性，引入实时预警机制加快响应速度，优化操作界面简化老年人使用流程，同时增强设备兼容性以支持多种监测设备。

二、实施步骤
采集高质量心率数据，预处理异常值，训练心率识别模型并部署，优化界面设计与预警机制，确保系统稳定运行并便于老人操作。

三、团队协作
技术团队负责设备与算法，产品团队优化界面交互，医疗团队设定指标，用户服务团队提供指导与支持。

四、预期效果
心率数据更准确，异常能快速识别并提示，老人操作更便捷，系统整体使用效率和安全性显著提升。

1.2.3 智慧金融服务业务模块效果优化



请勿修改答题卷，在指定单元格内填写答案

1. 2. 3-1

在智慧金融服务平台中，用户反馈最强烈的问题主要集中在以下五个方面：

1. 数据准确性问题

平台数据存在误差，直接影响用户对其金融健康状况的判断，进而导致投资或决策失误。用户对金融服务平台的核心诉求就是“可信”，一旦基础数据不准，整个系统的可信度就会崩塌。

2. 异常预警响应慢

异常监测和预警无法及时触发，使用户错过关键节点，无法及时采取止损或风险规避措施，带来潜在资产损失，降低了平台的风险防护能力。

3. 系统稳定性差

系统在实际运行中容易出现卡顿、崩溃，严重影响用户对平台的连续使用，甚至可能导致重要交易中断或数据丢失，极大降低使用信心。

4. 功能不完善

部分核心功能缺失，如财务健康评分维度不全、智能诊断建议匮乏，无法覆盖多样化的金融需求，限制了平台的适用场景，用户获得感不足。

5. 操作复杂度高

当前系统界面设计繁琐、逻辑不清晰，用户需要投入大量时间精力才能完成常规操作，体验差，对中老年用户尤为不友好。

1.2.3-2

优化目标

提升数据质量、加快异常预警响应、简化操作流程、增强系统稳定性和功能完备性，从而全面提升用户体验与服务质量。

关键实施步骤

1. 数据清洗与标注

全面清洗历史与现有金融数据，剔除冗余与错误记录。

引入高质量人工标注流程，保障关键数据维度的准确性与一致性。

2. 算法模型优化

引入轻量级异常检测算法（如 Isolation Forest、LSTM）提升识别率与实时性。

优化数据处理流，降低误报与漏报率。

3. 系统集成与测试

将新算法与模块部署至平台原有架构中，开展稳定性、兼容性和回归测试，保障平台核心功能不受影响。

4. 用户反馈收集与分析

设置主动反馈入口，持续收集用户在使用过程中的建议与问题。

利用 NLP 工具对用户反馈进行情感分析、需求挖掘，指导后续版本迭代。

5. 持续迭代与改进

以用户反馈和运营监测数据为依据，定期优化算法阈值、功能交互和系统响应策略。

引入灰度发布策略，逐步部署与验证改进效果。

预期优化效果

- 数据准确率提高，减少判断误差与决策风险。

- 预警响应速度提升，帮助用户更早应对金融风险。
- 系统界面更友好，操作流程更顺畅，用户上手更轻松。
- 运行稳定性增强，服务不中断，信任度提升。
- 功能覆盖更全面，满足多样化的个性化金融需求。

1.2.4 智能卖点生成系统业务模块效果优化

请勿修改答题卷，在指定单元格内填写答案

1.2.4-1

根据您提供的图片，用户反馈中存在的主要问题如下：
1. 卖点生成不准确：
- 问题描述：系统生成的卖点与实际产品不符，不能准确满足用户需求，影响用户对产品的判断。
- 影响：用户对系统的信任度降低，认为系统无法提供有价值的信息，进而影响其购买行为。这种不匹配直接影响了用户体验，降低了系统的可信度。

2. 缺乏个性化定制：
- 问题描述：系统无法根据不同用户的需求进行个性化卖点生成，无法满足用户的多样化需求，导致部分用户流失。
- 影响：导致用户流失或不再使用该系统，影响了用户粘性和系统的实际价值。

3. 生成速度慢：
- 问题描述：卖点生成过程时间较长，影响用户的使用效率。
- 影响：长时间的等待不仅降低了用户的使用效率，也可能使用户失去耐心，减少其使用系统的频率，影响用户满意度。

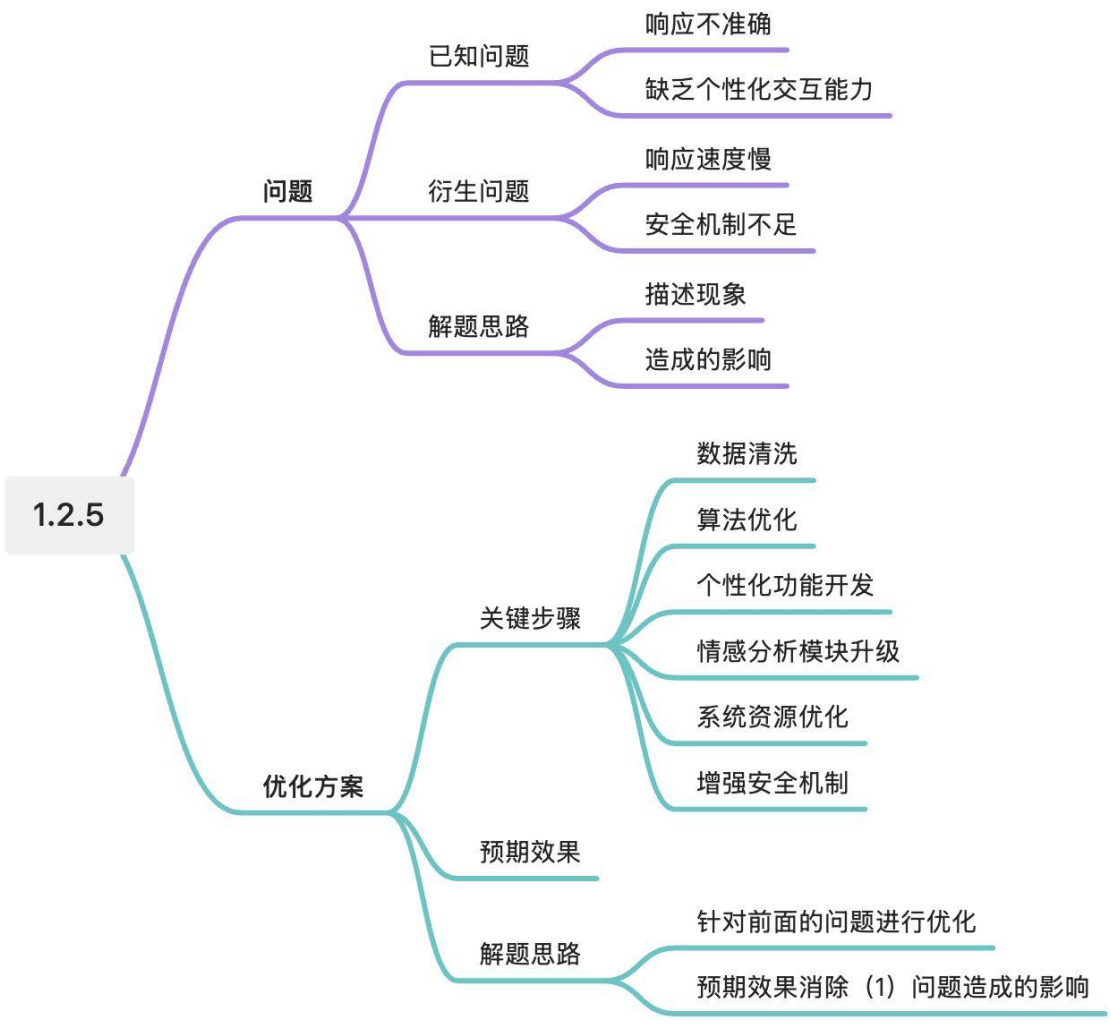
4. 操作复杂，缺乏流畅性：
- 问题描述：系统的操作流程复杂，用户需要花费更多时间和精力来完成操作。
- 影响：过于复杂的操作流程会让用户感到困扰，降低了系统的易用性，增加了用户的学习成本，影响整体用户体验。

1.2.4-2

针对上述问题，以下是优化方案的关键实施步骤和期望效果：
1. 提高卖点生成准确性：
步骤：通过优化算法，增强系统的语义理解能力，并增加更多真实产品数据的训练，使系统能够更好地理解产品特性与用户需求之间的关系。
期望效果：生成的卖点与产品特性和用户需求更加匹配，用户能够获得更准确的产品卖点，从而提升系统的可靠性和用户满意度。
2. 实现个性化卖点推荐：

步骤：根据用户的购买历史、偏好设置、行为数据等，进行个性化模型训练，提供针对不同用户群体的定制化卖点推荐。
期望效果：提高系统的个性化程度，满足用户的多样化需求，增强用户粘性，减少流失。
3. 加快卖点生成速度：
步骤：优化系统的计算流程，减少不必要的中间步骤，利用更高效的算法和硬件加速技术来提高处理速度。
期望效果：缩短卖点生成的时间，提高用户的使用效率，提升整体用户体验。
4. 简化操作流程：
步骤：对用户界面和操作流程进行简化设计，减少冗余步骤，提升系统的易用性。可以通过用户测试收集反馈，进一步优化操作流程。
期望效果：使用户能够更加轻松、高效地使用系统，减少操作难度，提升整体的用户满意度。

1.2.5 腾讯云智能数智人系统业务模块效果优化



请勿修改答题卷，在指定单元格内填写答案

1.2.5-1

1、响应不准确
训练数据存在噪声和错误，导致模型回答不准确，影响用户体验。
缺乏个性化交互能力
当前模型无法根据用户的个性需求进行定制，导致响应不够个性化和有吸引力。
2、情感识别不准确
情感分析模块不够精准，无法准确理解用户的情感状态，导致系统反馈缺乏共情。
3、响应速度慢
在高并发情况下，系统资源分配不均，导致响应时间过长。
安全机制不足
系统安全机制有漏洞，数据保护不够完善，用户隐私存在风险。

1.2.5-2

关键实施步骤：
1. 数据清洗
收集更准确、高质量的数据，去除噪声和错误。
2. 算法优化
引入先进的深度学习技术，提升模型处理复杂问题和个性化需求的能力。
3. 个性化功能开发
基于用户历史行为，进行个性化推荐和定制化服务。
4. 情感分析模块升级
引入更精准的情感识别算法，提升用户互动体验。
5. 系统资源优化
合理分配系统资源，提升响应速度和效率。
6. 增强安全机制
强化数据加密和访问控制，保护用户隐私。
预期效果：
- 提升响应准确性，增强用户信任。
- 提供个性化体验，提高用户满意度。
- 加强情感互动，提升用户体验。
- 加快响应速度，改善交互流畅性。
- 提升安全性，保护用户隐私。