目录

一、 问题重述

1.1 问题背景

随着科技快速进步，工业自动化领域的智能控制系统不断完善。现在的自动化生产线能够独立完成从物料输送到元件抓取，再到产品安装和质量检验等任务，这不仅显著提高了制造效率和产品质量，还有效地降低了生产成本。

在此背景下，就需要对生产线产生的数据进行深入分析，这样可以使业管理层和工程技术人员能及时了解生产线的运行状况和潜在问题，提高生产线的效率，减少产品的次品率。

1.2 问题重述

附件 1 包含了某生产企业两条生产线全年加工处理过程中各个工序的生产线数据，附件 2 包含了需要提交的模板文件。

本文基于这些上述数据，通过统计分析及优化建模，从各种角度完成以下任务：

1. 对两条生产线全年加工处理过程中各个工序的生产线数据进行整理与统计
2. 对两条生产线运行情况进行可视化分析
3. 应用数据分析方法分析影响产品合格率和生产线产量的可能因素。

二、 思路分析

根据题目中所给的三个任务，本文概括出的解题思路与答大体步骤如图1所示：

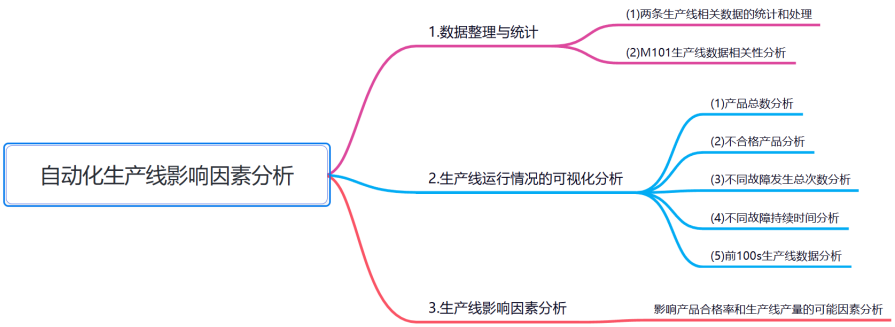


图 **1**　解题流程的思维导图

基于上述思维导图，可知本文将题目所给任务总体以下三步，首先对两条生产线的相关数据进行统计和整理，并且根据得到的 M101 生产线数据，对其每天推出的电路板数量、抓取的元件数量与抓取的故障次数做相关性分析，即针对任务一，将任务分解为数据的统计与处理和相关性分析两部分；其次，基于任务 1 整理与统计的数据，对产品总数、不合格产品、各故障类别的占比、全年故障发生持续时长和生产线 M101 的 4 个

工序的数据进行可视化分析，即任务二；最后基于任务 1 和任务 2 得到的数据，分析影响产品合格率和生产线产量的可能因素，即任务三。

三、 任务的求解

3.1 任务一的求解

3.1.1 数据的整理与统计

(一) 任务 1.1 的求解

首先通过对附件 1 两条生产线的数据进行查看可以发现：

- 表中数据量较大，以至于表格软件无法显示全所有数据。
- 表中的数据项包含我们需要统计的产品总数、合格产品数、不合格产品数，产品总数与表格中数据项检测累计数的含义相同。
- 表中数据随着时间地增加，当天的加工处理过程中各个工序的数据以及产品总数、合格产品数、不合格产品数逐渐递增。

基于上述附件表中的特征，统计两条生产线每天的产品总数（包含不合格产品）、合格产品数、不合格产品数，只需要获取表中当天机器开机时长的值最大时对应的的检测累计数、合格产品累计数和不合格产品累计数即可；对于合格率，只需要用合格产品累计数除以检测累计数即可求出，详细结果见 result1_1.xlsx。对于全年的产品总数、合格产品数、不合格产品数，只需要将每天的数据对应相加，合格率只需要用合格产品数除以产品总数即可得到，各生产线全年产品总数、合格产品数、不合格产品数与合格率如表1所示，统计所用到的 python 部分关键代码具体如下。

生产线	产品总数（件）	合格产品数（件）	不合格产品数（件）	合格率（%）
M101	1183016	1180910	2106	99.82%
M102	1186838	1184461	2377	99.80%

表 **1**　各生产线全年产品总数、合格产品数、不合格产品数与合格率表

```
df_M101['新日期'] = df_M101['月份'].astype(str) + '-' + df_M101['日期'].astype(str)
daily_stats_M101 = df_M101.groupby('新日期').agg(
总产品数=('检测累计数', 'max'),
合格产品数=('合格产品累计数', 'max'),
不合格产品数=('不合格产品累计数', 'max')
).reset_index()

daily_stats_M101['合格率'] = daily_stats_M101['合格产品数'] / daily_stats_M101['总产品数']

# 计算M101全年的产品总数、合格产品数、不合格产品数与合格率
```

```
annual_stats_M101 = daily_stats_M101.agg({
    '总产品数': 'sum',
    '合格产品数': 'sum',
    '不合格产品数': 'sum'
}).to_frame().T
annual_stats_M101['合格率'] = (annual_stats_M101['合格产品数'] /
    annual_stats_M101['总产品数'])

# 计算M102全年的产品总数、合格产品数、不合格产品数与合格率
annual_stats_M102 = daily_stats_M102.agg({
    '总产品数': 'sum',
    '合格产品数': 'sum',
    '不合格产品数': 'sum'
}).to_frame().T
annual_stats_M102['合格率'] = (annual_stats_M102['合格产品数'] /
    annual_stats_M102['总产品数'])
```

(二) 任务 1.2 的求解

(1) 数据预处理

由数据字段说明.xlsx 可知，只有当故障类别不为空时才表示有故障；而且表格数据量庞大，所以我们需要对文件 M101.csv、M102.csv 做预处理，我们只保留” 故障类别” 数据项不会空的数据项，便于之后的统计工作。处理完的表格详情见附件。

(2) 统计故障相关信息

现在只需要对预处理之后的表格进行统计。对于如何判断为同一次故障，当连续的两个记录的时间只相差 1，且故障类别相同，就认为这是同一次故障。基于上述判断，持续时间只需要用对应故障的结束时间减去开始时间即可求得，两条生产线每次故障的相关信息详情见 result1_2.xlsx。由于 result1_2.xlsx 是按照月份、日期和开始时间升序排列，所以统计各生产线每种故障一年中第 25 次发生的相关信息，只需要选取键值为相应的故障类别，然后求取第 25 行数据即为每种故障一年中第 25 次发生的相关信息。各生产线每种故障一年中第 25 次发生的相关信息表如表2所示，统计所用到的 python 部分关键代码具体如下。

```
    # 用于遍历
    start_idx = 0
    while start_idx < len(df_2_faults):
    end_idx = start_idx # 开始一次识别
    while (end_idx + 1 < len(df_2_faults) and
    df_2_faults.iloc[end_idx + 1]['故障类别'] == df_2_faults.iloc[start_idx]['故障类别'] and
    df_2_faults.iloc[end_idx + 1]['时间'] - df_2_faults.iloc[end_idx]['时间'] == 1):
    end_idx += 1 # 说明同一条记录
    # 当上面循环结束
    start_time = df_2_faults.iloc[start_idx]['时间']
    end_time = df_2_faults.iloc[end_idx]['时间']
```

```
        duration = end_time - start_time
        fault_periods_M102.append({
            '月份': df_2_faults.iloc[start_idx]['月份'],
            '日期': df_2_faults.iloc[start_idx]['日期'],
            '故障类别': df_2_faults.iloc[start_idx]['故障类别'],
            '开始时间': start_time,
            '结束时间': end_time,
            '持续时间': duration
        })
        start_idx = end_idx + 1
    # 转换结果为DataFrame
    df_fault_periods_M102 = pd.DataFrame(fault_periods_M102)
    # 按照月份、日期和开始时间升序排列
    df_fault_periods_M102 = df_fault_periods_M102.sort_values(by=['月份', '日期',
        '开始时间'])

    # M101 每种故障第25次出现的月份、日期、开始时间、持续时长
    M101_fault_25th_time = df_fault_periods.groupby('故障类别').nth(24).reset_index()
    # 调整列的顺序为 ‘故障类别’ ‘月份’ ‘日期’ ‘开始时间’ ‘持续时间’，按照类别排列
    M101_fault_25th_time = M101_fault_25th_time[['故障类别', '月份', '日期', '开始时间',
        '持续时间']]
    # 按照类别为A1, A2, A3, A4
    M101_fault_25th_time = M101_fault_25th_time.sort_values(by='故障类别')
```

生产线	故障类别	月份	日期	开始时间	持续时长 (秒)
M101	A1	2	14	28553	246
M101	A2	1	4	19819	671
M101	A3	1	30	3717	864
M101	A4	1	13	22814	886
M102	A1	3	2	25982	580
M102	A2	1	5	7842	383
M102	A3	1	14	28122	624
M102	A4	2	12	12307	736

表 **2**　各生产线每种故障一年中第 **25** 次发生的相关信息表

(三) 任务 1.3 的求解

由于已经得到任务 1.2 预处理之后的表格和 result1_2.xlsx 的结果，可以再次基础上继续进行统计。对于各类故障每天发生的总次数，可以在预处理之后的表格上选取日期和故障类别相同，计算总次数；对于各类故障每天发生的平均持续时长，可以通过 result1_2.xlsx 中的对应日期和故障类别的持续时长求平均得到。详细数据见 result1_3.xlsx。

两条生产线各类故障发生的总次数、平均持续时长、故障发生频率可以对 result1_3.xlsx 进行统计得到。M101 故障发生的总次数、平均持续时长、故障发生频率和汇总如表3所示；M102 故障发生的总次数、平均持续时长、故障发生频率和汇总如表4所示；统计所用到的 python 部分关键代码具体如下。

```
# 和M102没任何关系，只是借用一下统计好的每日日期，M102里面的每日日期刚好覆盖365天
df_temp = daily_stats_M102[['月份', '日期']].copy()
# 故障类别取值
all_fault_types = ['A1', 'A2', 'A3', 'A4']
# 使用pd.DataFrame的reindex方法扩展行数
df_temp = df_temp.reindex(df_temp.index.repeat(len(all_fault_types)))
# 添加故障类别列
df_temp['故障类别'] = all_fault_types * (len(df_temp) // len(all_fault_types))
df_temp.reset_index(drop=True, inplace=True)
# 到这里设置好了每日日期和故障类别
# 下面从M102故障记录中统计每日故障次数和平均持续时间， 然后进行表的合并
# 统计每天每种故障的次数以及平均持续时间
fault_stats_M102 = df_fault_periods_M102.groupby(['月份', '日期', '故障类别']).agg(
总次数=('故障类别', 'count'),
平均持续时长=('持续时间', 'mean')
).reset_index()
# 将统计结果合并到df_temp中
df_temp = df_temp.merge(fault_stats_M102, on=['月份', '日期', '故障类别'], how='left'
# 填充缺失值为0
df_temp['总次数'] = df_temp['总次数'].fillna(0).astype(int)
df_temp['平均持续时长'] = df_temp['平均持续时长'].fillna(0).astype(float).round(2)
# 对于平均持续时长为0的，填充Null而不是0
df_temp.loc[df_temp['平均持续时长'] == 0, '平均持续时长'] = np.nan
# 拼接M101_daily_fault和M102_daily_fault
combined_daily_fault = pd.concat([M101_daily_fault, M102_daily_fault],
    ignore_index=True)

# 统计各类故障的总次数和平均持续时长
fault_summary_M101 = df_fault_periods.groupby('故障类别').agg(
总次数=('故障类别', 'count'),
平均持续时长=('持续时间', 'mean')
).reset_index()
# 计算发生频率（次/天）
total_days = 365
fault_summary_M101['发生频率'] = fault_summary_M101['总次数'] / total_days
# 增加汇总行
total_faults = df_fault_periods['故障类别'].count()
average_duration = df_fault_periods['持续时间'].mean()
frequency = total_faults / total_days

summary_row = pd.DataFrame([{
```

```
        '故障类别': '汇总',
        '总次数': total_faults,
        '平均持续时长': average_duration,
        '发生频率': frequency
    }])
    fault_summary_M101 = pd.concat([fault_summary_M101, summary_row], ignore_index=True)
```

生产线 M101	故障类别 A1	故障类别 A2	故障类别 A3	故障类别 A4	汇总
总次数	240	2335	319	826	3720
平均持续时长（秒/次）	473.03	540.31	727.61	737.83	595.89
发生频率（次/天）	0.66	6.40	0.87	2.26	10.19

表 **3**　**M101** 故障发生的总次数、平均持续时长、故障发生频率表

生产线 M102	故障类别 A1	故障类别 A2	故障类别 A3	故障类别 A4	汇总
总次数	217	2263	318	852	3650
平均持续时长（秒/次）	467.86	540.52	726.33	744.40	599.98
发生频率（次/天）	0.59	6.20	0.87	2.33	10.00

表 **4**　**M102** 故障发生的总次数、平均持续时长、故障发生频率表

(四) 任务 1.4 的求解

通过观察附件中数据我们可以知道：两条生产线的开机时长都为 28800 秒，也就是 8 小时，所以我们只需要求出故障停机的时长就可以求得有效工作时长。两条生产线每天的故障停机时长可以对 result1_2.xlsx 相同日期的持续时长相加可以得到，然后再用 8 小时相减就可以得到每天的有效工作时长。求出的详细结果详情见 result1_4.xlsx。

对于每条生产线的日平均有效时长，我们可以对 result1_4.xlsx 取平均值得到。得到的结果如表5所示。统计所用到的 python 部分关键代码具体如下。

```
    # 计算每天的故障持续时间总和
    daily_fault_duration = df_fault_periods.groupby(['月份',
        '日期'])['持续时间'].sum().reset_index()
    # 一天总时间
    total_time_per_day = 28799
    # 计算每天的有效工作时长（小时）
    daily_fault_duration['M101（小时）'] =((total_time_per_day -
        daily_fault_duration['持续时间']) / 3600).round(2)
    # 只保留需要的列
    daily_fault_duration = daily_fault_duration[['月份', '日期', 'M101（小时）']]
    # 合并M101和M102每天有效工作时长表
    combined_daily_duration = pd.merge(daily_fault_duration, daily_fault_duration_M102,
        on=['月份', '日期'])
```

```
# 求M101 日平均有效工作时长 （小时/天）
M101_avg_daily_working_hours = daily_fault_duration['M101（小时）'].mean()
print(f"M101 日平均有效工作时长: {M101_avg_daily_working_hours:.2f} 小时/天")
# 求M102 日平均有效工作时长 （小时/天）
M102_avg_daily_working_hours = daily_fault_duration_M102['M102（小时）'].mean()
print(f"M102 日平均有效工作时长: {M102_avg_daily_working_hours:.2f} 小时/天")
# 创建包含M101和M102日平均有效工作时长的DataFrame
task_1_3_2_table_8 = pd.DataFrame({
   '生产线': ['M101', 'M102'],
   '日平均有效工作时长（小时/天）': [M101_avg_daily_working_hours,
       M102_avg_daily_working_hours]
})
```

生产线	日平均有效工作时长（小时/天）
M101	6.31
M102	6.33

表 **5**　各生产线日平均有效工作时长表

3.1.2 生产线 M101 数据的相关性分析

(一) 数据预处理

通过观察 M101.csv 中的数据，我们可以得到当抓取状态为-1 时，故障类别总是 A2，对数据进行详细查找，尝试查找故障类别为 A2，且抓取状态不为-1 的记录，得到的结果为空。因此可以判断，原数据中，故障类别为 A2 对应的就是抓取故障。关键代码如下：

```
# 输出df_M101中，故障记录为A2，且抓取状态不为-1的
df_M101_A2_not_minus_1 = df_M101[(df_M101['故障类别'] == 'A2') & (df_M101['抓取状态'] !=
    -1)]
print(df_M101_A2_not_minus_1)
```

为了更方便得到每天推出的电路板数量、抓取的元件数量与抓取的故障次数做相关性，我们首先对附件 M101.csv 进行统计，获得生产线 M101 每天推出的电路板数量、抓取的元件数量与 A2 故障次数。详细的数据见 1.5 数据.csv，其中的部分数据如图2所示

(二) 相关性分析

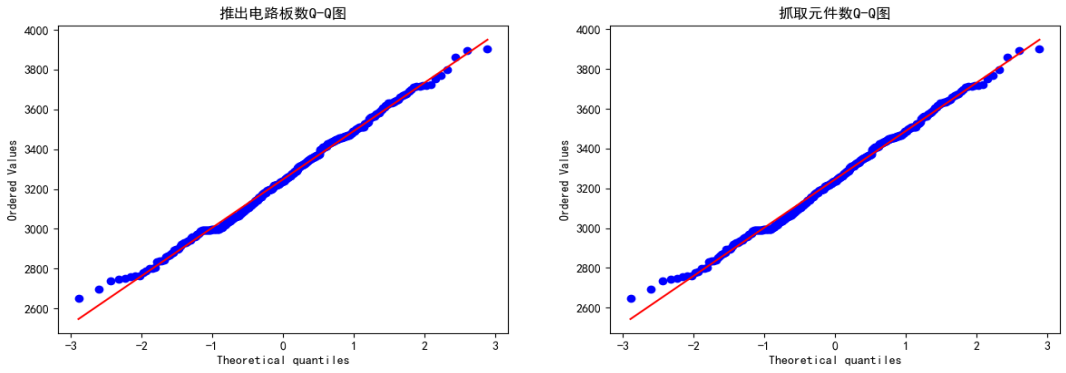
对生产线 M101 每天推出的电路板数量、抓取的元件数量与抓取的故障次数做相关性分析，首先对数据的正态性进行检验，以确保选择合适的相关性度量方法。在相关性分析中，若数据符合正态分布，可采用皮尔逊相关系数；若数据不符合正态分布，可采用斯皮尔曼相关系数。经过数据预处理，得到进行相关性分析的数据有 365 条记录，对

月份	日期	推出电路板	抓取元件数	故障A2次数
1	1	3469	3465	7
1	2	3259	3255	9
1	3	3461	3457	4
1	4	3474	3470	7
1	5	3451	3447	4
1	6	3194	3190	6
1	7	3674	3670	4
1	8	3344	3340	4
1	9	3107	3103	6
1	10	3353	3350	5
1	11	3411	3407	7
1	12	3591	3588	4
1	13	3217	3213	4
1	14	3529	3526	4
1	15	3278	3274	7
1	16	3905	3901	3
1	17	3538	3534	5
1	18	3078	3075	7
1	19	3325	3322	4
1	20	3631	3627	4
1	21	3798	3794	3
1	22	3639	3635	5
1	23	3132	3128	9
1	24	3575	3571	7

图 **2**　数据 **1.5.csv** 的部分数据

应每一天的统计数据。对于推出的电路板数量、抓取的元件数量，Q-Q 图来检验数据的正态性。对于抓取的故障次数，使用 Shapiro-Wilk 检验来检验数据的正态性。

Q-Q 图可以直观显示样本分布与理论正态分布的吻合程度。如果数据点大致落在 45 度对角线附近，则表示该数据较好地服从正态分布。该方法为我们提供了辅助的视觉验证，便于更直观地判断正态性。使用 scipy.stats 工具和 pyplot 绘制推出的电路板数量、抓取的元件数量的 Q-Q 图如图3所示。



(a) 推出的电路板数量 **Q-Q** 图　　**(b)** 抓取的元件数量 **Q-Q** 图

图 **3**　推出的电路板数量和抓取的元件数量的 **Q-Q** 图

经过检验，变量符合正态分布，因此可以使用皮尔逊相关系数进行相关性分析。计算三组数据的皮尔逊相关系数矩阵，得到结果如表6所示

推出电路板数与抓取元件数的皮尔逊相关系数: 0.9999980110790238,p 值: 0.0 推出电路板数与故障 A2 次数的皮尔逊相关系数: -0.6162929407758282, p 值: $1.4870798869361526e^{-39}$

	推出电路板数	抓取元件数	故障 A2 次数
推出电路板数	1.000000	0.999998	-0.616293
抓取元件数	0.999998	1.000000	0.616500
故障 A2 次数	-0.616293	-0.616500	1.000000

表 **6**　皮尔逊相关系数矩阵

抓取元件数与故障 A2 次数的皮尔逊相关系数: -0.6164997436659938, p 值: $1.3796877227059268e^{-39}$。

由皮尔逊相关系数矩阵可知：

1. 推出电路板数与抓取元件数之间的皮尔逊相关系数极接近 1，且 p 值为 0，说明两者之间存在极强的线性相关性。推出电路板数与故障 A2 次数之间的皮尔逊相关系数为-0.616，中等程度的负相关关系。且 p 值几乎为 0，远小于 0.05，说明该负相关关系在统计上显著。
2. 抓取元件数与故障 A2 次数的皮尔逊相关系数为-0.6165，也显示出与上述类似的负相关关系，且相关性强度和显著性水平与推出电路板数和故障 A2 次数的关系相似。这一结果表明，元件抓取量越大，故障 A2 的发生频率越低。p 值几乎为 0，说明该负相关关系在统计上显著。
3. 推出电路板数和抓取元件数在生产过程中几乎是完全正相关的，且他们都与故障 A2 次数呈负相关关系。每个电路板都需要相应数量的元件抓取动作，因此推板和抓取元件数高度一致，说明生产线的元件供给和产出之间在该数据期间保持稳定，这也是理想的状态。
4. 当生产线上发生频繁故障时，生产会出现停滞，影响到推板和抓取的连续性，导致产量下降。而当故障次数较低时，生产线保持稳定运转，推动产量提升。这种关系符合实际的生产逻辑，揭示了设备状态对生产效率的影响。符合实际的生产逻辑，揭示了设备状态对生产效率的影响。

3.2 任务二的求解

3.2.1 各月产品总数的可视化分析

各月的产品总数（包含不合格产品）情况如图4所示。

从图中可以看出，两条生产线在不同月份的产品数量均存在波动，但 M101 生产线的产量波动性较大，而 M102 生产线的产量波动性相对较小。这意味着 M102 生产线在控制生产稳定性方面做得更好；M101 和 M102 生产线都在 1 月以及 8 月-12 月这几个月份明显要比其他月份的产量有不不同程度的上升，并在 12 月和 1 月的产量都达到一年中的最高值，分别超过了 10.5w, 总产量超过了 21w，我们可以推出此产品的生产存在一定的季节性，秋冬季节生产较多，春夏季节生产较少。

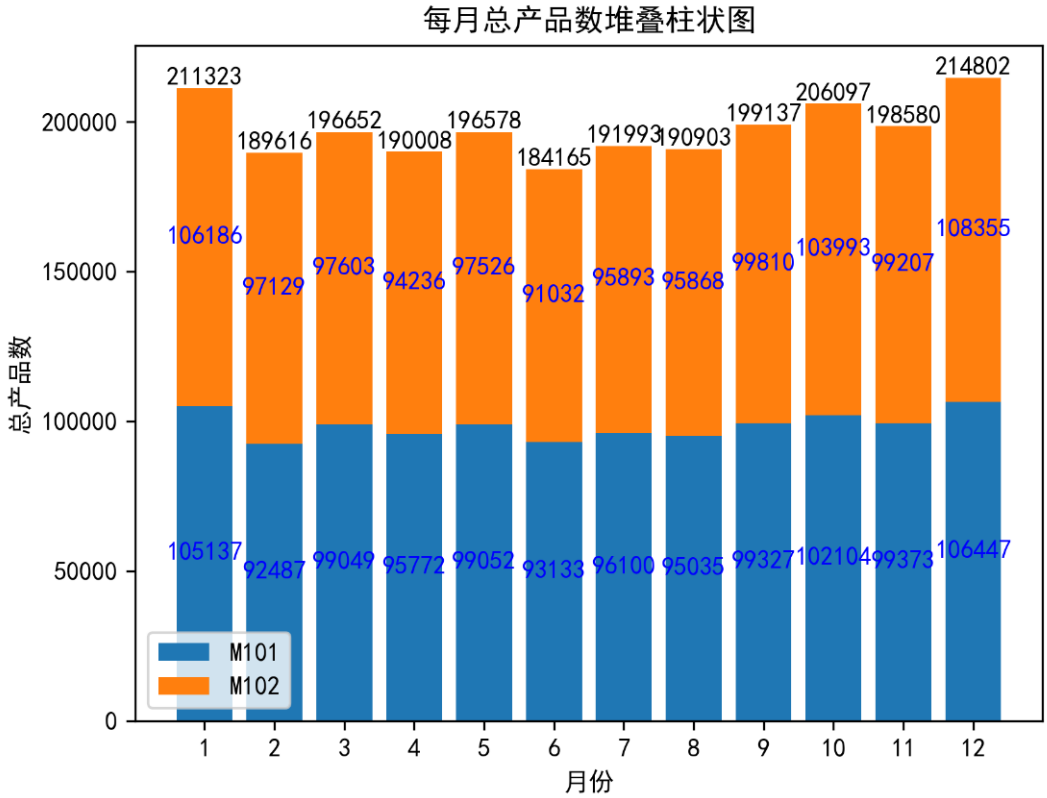


图 **4**　各月的产品总数图

3.2.2 不合格产品数和不合格率的可视化分析

两条生产线每天的不合格产品数和不合格率的双 Y 轴折线图如图5和图6所示。

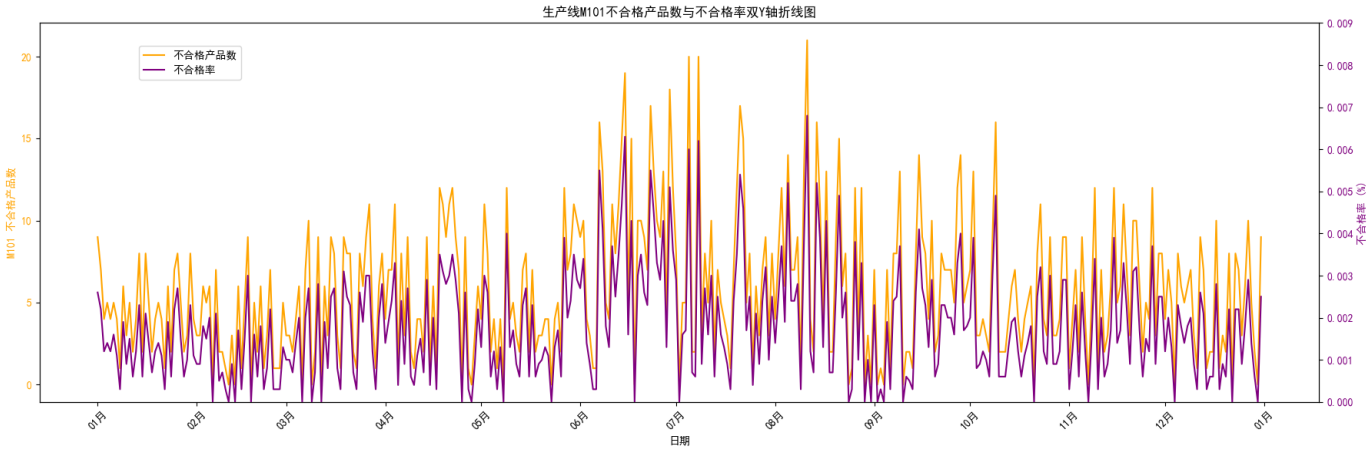


图 **5**　**M101** 生产线每天的不合格产品数和不合格率图

从图中我们可以看出，总体而言 M101 和 M102 两条数据线都存在明显的季节性特征：春夏天即 6-9 月份的不合格率和不合格产品数相较于其他年份来说最多，冬天即 11 月-12 月，1 月-2 月的不合格率和不合格产品数相较于其他年份来说最少；M101 和 M102 生产线的波动幅度和增加趋势有所不同，M101 生产线从 6 月开始才开始有不合

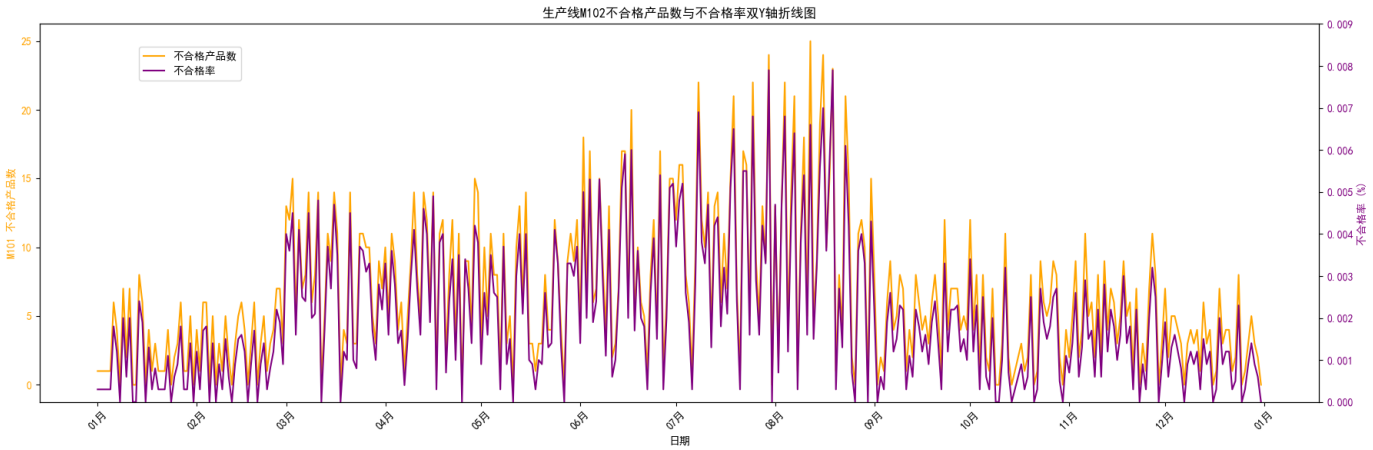


图 6　**M102** 生产线每天的不合格产品数和不合格率图

格率和不合格产品数的突然上升，而 M102 生产线从 3 月就开始有不合格率和不合格产品数的突然上升，并在 7-9 月到达峰值。

3.2.3 各故障类别的占比的可视化的分析

两条生产线各故障类别的占比双层环形图如图7所示。

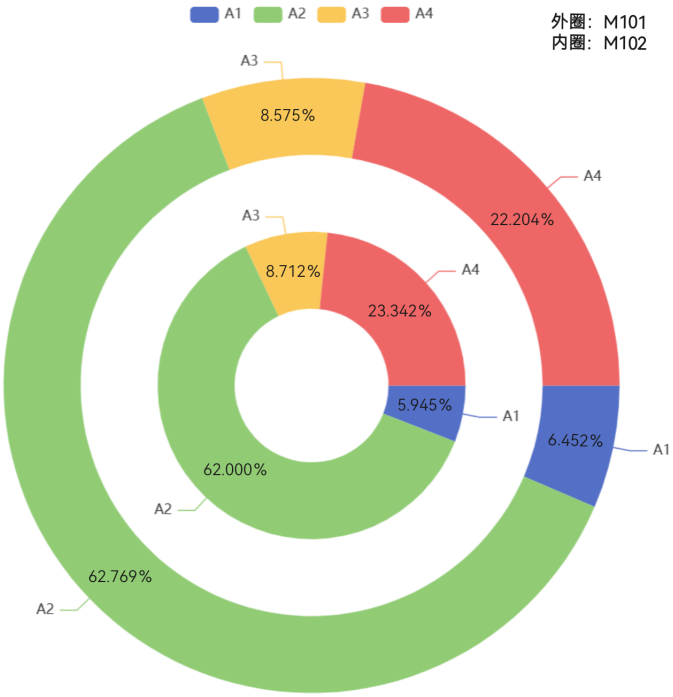


图 7　两条生产线各故障类别的占比双层环形图

从图中可以看出，M101 生产线上 A1，A2，A3，A4 的占比分别为：6.45%,62.77%，8.58%,22.20%,M102 生产线上 A1，A2，A3，A4 的占比分别为：5.94%，62.00%，8.71%，23，34%；两条生产线上，各个故障发生次数的占比大致相等；类别 A2 的故障，即抓取

故障发生的次数最多，略高于 60%；故障类别为 A1 和 A3，即推出故障和安装故障的发生次数最少，分别约为 6% 和 8% 左右。检测故障占比约 22%。所以可以得出结论：在生产线生产的过程中，抓取故障发生次数最多。

3.2.4 全年故障发生持续时长的可视化分析

两条生产线全年故障发生持续时长的叠加直方图分别如图8和图9所示

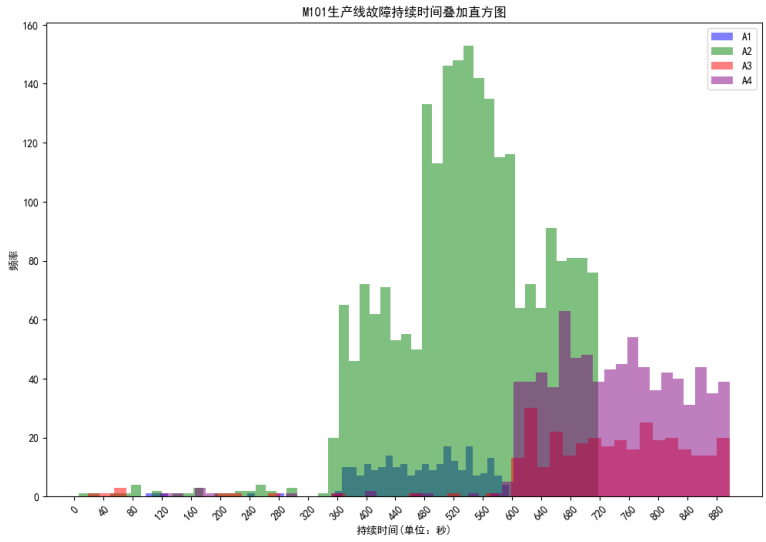


图 **8**　**M101** 生产线全年故障发生持续时长图

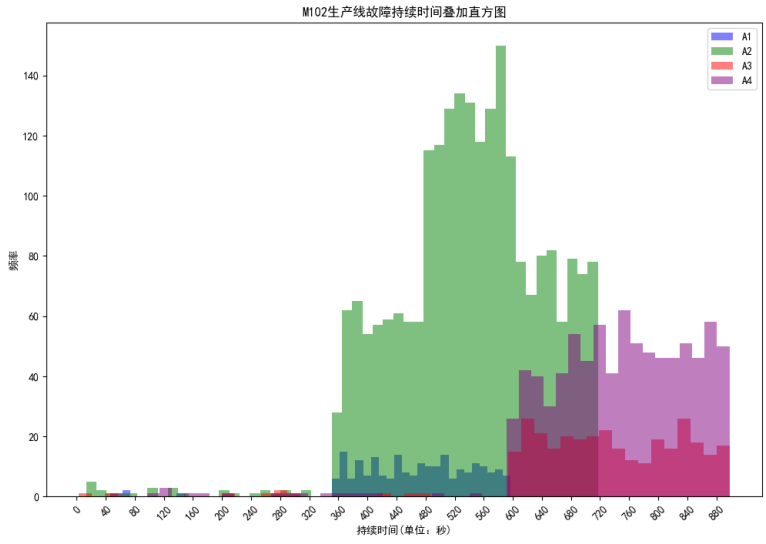


图 **9**　**M102** 生产线全年故障发生持续时长图

从图中我们可以看出，两条生产线的故障持续时间叠加直方图形状大致相同。从两条生产线各种故障持续时间的叠加直方图可以看出，A2 类型即抓取故障，在图中占有最多的面积，是故障中最主要的，与各种故障发生次数占比环状图是一致的。

持续时间直方图可以直观反映不同类型故障发生的时间分布，可以直观看出，推出故障（A1）所需的时间持续时间最短，其次是抓取故障。而安装故障和检测故障的持续时间都比较长，能够持续 600-900 秒。

同时，持续时间叠加直方图可以看出发生次数占比图无法表达的信息。尽管推出故障和安装故障的发生次数占比上相差不大，但是安装故障的持续时间长，在生产中造成的负面影响远大于推出故障。

3.2.5 生产线 M101 四个工序数据的可视化分析

生产线 M101 包含 4 个工序的甘特图如图10所示。

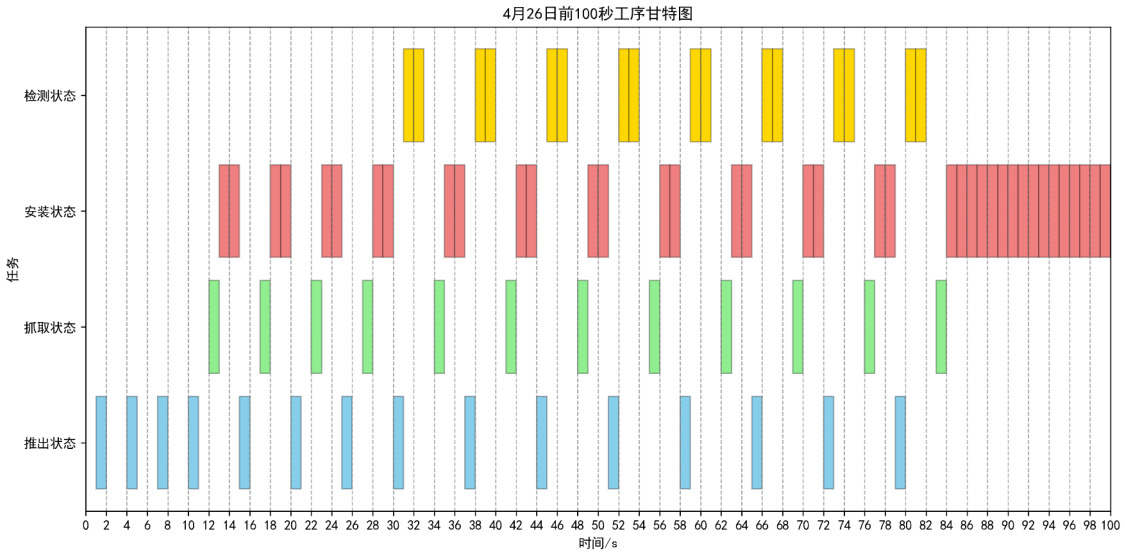


图 **10**　生产线 **M101** 包含 **4** 个工序的甘特图

由图中可以看出，生产线工序由电路板推出-抓取-安装-检测的顺序组成，单次推出或抓取工作需要 1 秒，单次安装或检测工作需要 2 秒。起始时每隔 3 秒推出一块电路板，速度较快，工作稳定后每道工序的运行周期都是 7 秒。在 84 秒发生 A1 故障（推出故障）时，抓取和检测环节停止工作，安装环节保持工作状态，直至推出环节恢复。

由上述可视化分析得出的结果可以得到如下的结论：生产线 M101 与 M102 季节性生产明显，秋冬高产，春夏次品增多。M102 稳定性优于 M101。故障中，抓取故障频发，安装故障虽少但影响大。优化方向应聚焦于减少抓取故障及提升故障恢复效率。

3.3 任务三的求解

基于任务二可视化分析得到的结论，可以得到两条生产线所表现出的特征相似度较大，所以我们以 M101 生产线为例来通过以下角度分析影响产品合格率和生产线产量的可能因素。

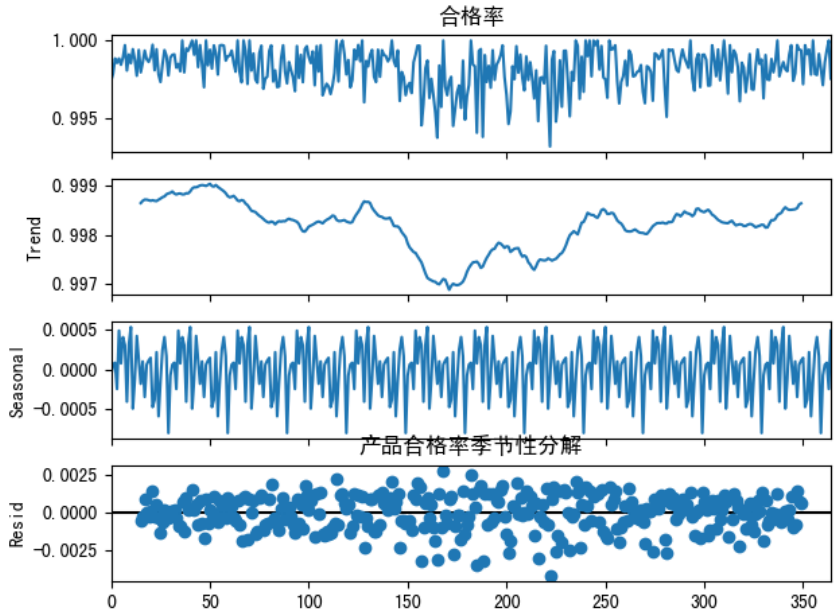


图 **11**　产品合格率及季节性分解图

图11为 M101 生产线一年内产品合格率随时间的变化。合格率在 0.993 到 1.000 之间波动，整体趋势较为稳定，但存在一定的波动性。可以看出再一年中间（约 6，7 月份）时，合格率偏低。且合格率在某些时间段内有明显的下降和上升。

图11中的趋势图显示了合格率的长期变化趋势。从图中可以看出，趋势线在中间部分有一个明显的下降，然后逐渐回升，表明合格率在这段时间内经历了一个下降后回升的过程。与合格率随时间变化图是一致的，说明一年中间时候确实会在合格率上下降。季节性：显示了合格率的周期性波动。从图中可以看出，季节性波动较为明显，表明合格率存在一定的周期性变化。

图11中的残差图显示了去除趋势和季节性后的随机波动。残差图显示了合格率的随机波动情况。残差无明显模式、呈白噪声分布，说明剩余的随机波动是很合理的，分解模型较好地拟合了数据。

从图12可以看出，总产品数随时间的变化总体在 3000 到 3500 之间波动。有与产品合格率类似的变化。在年初和年末时，总产量略高与年中。一年生产的总产品数的趋势线，在 7、8 约时到低谷，到年末又回升，与全年的变化也是一致的。从季节性图可以看出，产量的变化有明显的季节性特征。残差分布图几乎是均匀分布在中心轴两侧，残差分布良好，说明分解模型已充分解释了数据的趋势和季节性。这与此前统计的不合格率随时间变化相符合，一年的中间时候，不合格率达到高峰值。

通过以上时间序列分析可以说明，生产线生产产品的产量与产品合格率受时间的影响，有明显的季节性特征。在一年中段的时候，即第二季度末、第三季度初期，产量和合格率都到达低谷。这可能与夏季气温高，影响了生产线的某些元器件，使得生产线更容易出故障等。也有可能与该生产线设施的检修期恰好是 7 月份左右有关系。

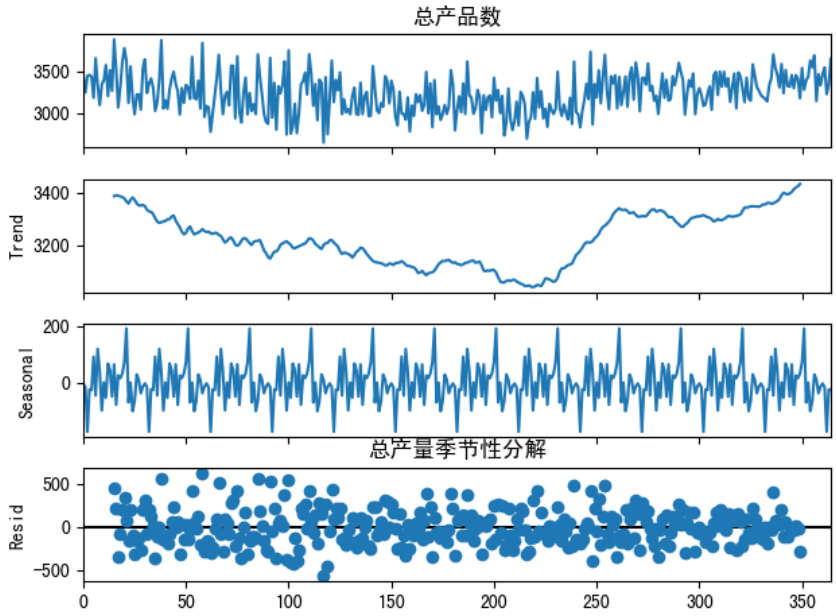


图 **12**　总产量数及季节性分解图