```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from IPython.display import display

# 載入資料
file_path = '/Users/hanmingcheng/Documents/python vscode/產學/處理
dataraw_20241031.csv'
data = pd.read_csv(file_path)

# 確保數據格式正確
data['焊錫半成品電感值'] = pd.to_numeric(data['焊錫半成品電感值'],
errors='coerce')
data['點膠重量'] = pd.to_numeric(data['點膠重量'], errors='coerce')
data['塗膠完成品電感值'] = pd.to_numeric(data['塗膠完成品電感值'],
errors='coerce')

# 去除遺失值
data = data.dropna(subset=['焊錫半成品電感值', '點膠重量', '塗膠完成品電感
值'])

# 定義目標變數(Y) 和解釋變數(X)
X = data[['焊錫半成品電感值', '點膠重量']]
Y = data['塗膠完成品電感值']

# 添加常數項(用於回歸截距)
X = sm.add_constant(X)

# 建立回歸模型
model = sm.OLS(Y, X).fit()

# 顯示回歸結果摘要
# print(model.summary())

# 預測值計算
data['預測值'] = model.predict(X)

display(model.summary())
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results

==============================================================================
========
Dep. Variable:               塗膠完成品電感值 R-squared:
0.347
Model:                            OLS   Adj. R-squared:
0.336
Method:                 Least Squares   F-statistic:
```

```
32.36
Date:                Fri, 06 Dec 2024   Prob (F-statistic):
5.30e-12
Time:                        12:19:40   Log-Likelihood:
-89.462
No. Observations:                 125   AIC:
184.9
Df Residuals:                     122   BIC:
193.4
Df Model:                           2

Covariance Type:            nonrobust

======================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
--------
const        -14.0695      4.596     -3.061      0.003     -23.167
-4.972
焊錫半成品電感值   2.3705      0.300      7.904      0.000       1.777
2.964
點膠重量     0.0767      0.089      0.860      0.392      -0.100
0.253
======================================================================
========
Omnibus:                        1.199   Durbin-Watson:
1.791
Prob(Omnibus):                  0.549   Jarque-Bera (JB):
0.754
Skew:                          -0.126   Prob(JB):
0.686
Kurtosis:                       3.285   Cond. No.
1.60e+03
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.6e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
"""

import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import numpy as np
```

```python
from IPython.display import display

# 載入資料
file_path = '/Users/hanmingcheng/Documents/python vscode/產學/處理
dataraw_20241031.csv'
data = pd.read_csv(file_path)

# 確保數據格式正確
data['焊錫半成品電感值'] = pd.to_numeric(data['焊錫半成品電感值'],
errors='coerce')
data['點膠重量'] = pd.to_numeric(data['點膠重量'], errors='coerce')
data['塗膠完成品電感值'] = pd.to_numeric(data['塗膠完成品電感值'],
errors='coerce')

# 去除遺失值
data = data.dropna(subset=['焊錫半成品電感值', '點膠重量', '塗膠完成品電感
值'])

# 對'點膠重量' 取對數
data['點膠重量_log'] = np.log(data['點膠重量'])

# 定義目標變數(Y) 和解釋變數(X)
X = data[['焊錫半成品電感值', '點膠重量_log']]
Y = data['塗膠完成品電感值']

# 添加常數項(用於回歸截距)
X = sm.add_constant(X)

# 建立回歸模型
model = sm.OLS(Y, X).fit()

# 顯示回歸結果摘要
# print(model.summary())

# 預測值計算
data['預測值'] = model.predict(X)

display(model.summary())
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results

==============================================================================
========
Dep. Variable:               塗膠完成品電感值  R-squared:
0.349
Model:                            OLS   Adj. R-squared:
0.338
Method:                 Least Squares   F-statistic:
```

```
32.65
Date:                Fri, 06 Dec 2024   Prob (F-statistic):
4.41e-12
Time:                        12:20:07   Log-Likelihood:
-89.274
No. Observations:                 125   AIC:
184.5
Df Residuals:                     122   BIC:
193.0
Df Model:                           2

Covariance Type:            nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const        -14.0702      4.588     -3.067      0.003     -23.152
-4.988
焊錫半成品電感值   2.3723      0.299      7.933      0.000       1.780
2.964
點膠重量_log       0.1881      0.179      1.053      0.294      -0.165
0.542
==============================================================================
========
Omnibus:                        1.318   Durbin-Watson:
1.789
Prob(Omnibus):                  0.517   Jarque-Bera (JB):
0.857
Skew:                          -0.133   Prob(JB):
0.651
Kurtosis:                       3.306   Cond. No.
1.58e+03
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.58e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
"""
```

----------------------------從這裡開始--------------------------------

決策樹分類模型運作簡要說明

模型概述

- 目標：運用決策樹分割變數之間的關係，根據「焊錫半成品電感值」和「點膠重量」兩個特徵，預測塗膠完成品電感值的狀態。
- 分類結果：
  - 低於 **LCL**：完成品電感值低於下控制界限。
  - 正常範圍：完成品電感值在控制界限內。
  - 高於 **UCL**：完成品電感值高於上控制界限。

決策樹運作邏輯

1. 從第一層節點開始：
   - 根據「焊錫半成品電感值」進行第一次分類。
   - 範例：`焊錫半成品電感值 ≤ 15.495`，成立時向左，否則向右。
2. 第二層節點進一步分類：
   - 使用「焊錫半成品電感值」或「點膠重量」進一步細分。
   - 範例：`點膠重量 ≤ 2.45`，成立時向左，否則向右。
3. 到達第三層節點：
   - 葉節點給出分類結果（例如，「正常範圍」或「高於 UCL」）。
   - 每個節點包含樣本數與分類分佈。

決策路徑示例

範例 1：分類為「正常範圍」

- 條件路徑：
  a. `焊錫半成品電感值 ≤ 15.495`（成立，向左）。
  b. `焊錫半成品電感值 ≤ 15.325`（成立，向左）。
  c. `點膠重量 ≤ 2.45`（成立，向左）。
- 結果：分類為「正常範圍」。

範例 2：分類為「高於 UCL」

- 條件路徑：
  a. `焊錫半成品電感值 > 15.495`（不成立，向右）。
  b. `點膠重量 ≤ 2.85`（成立，向左）。
  c. `點膠重量 ≤ 1.2`（不成立，向右）。

- 結果：分類為「高於 UCL」。

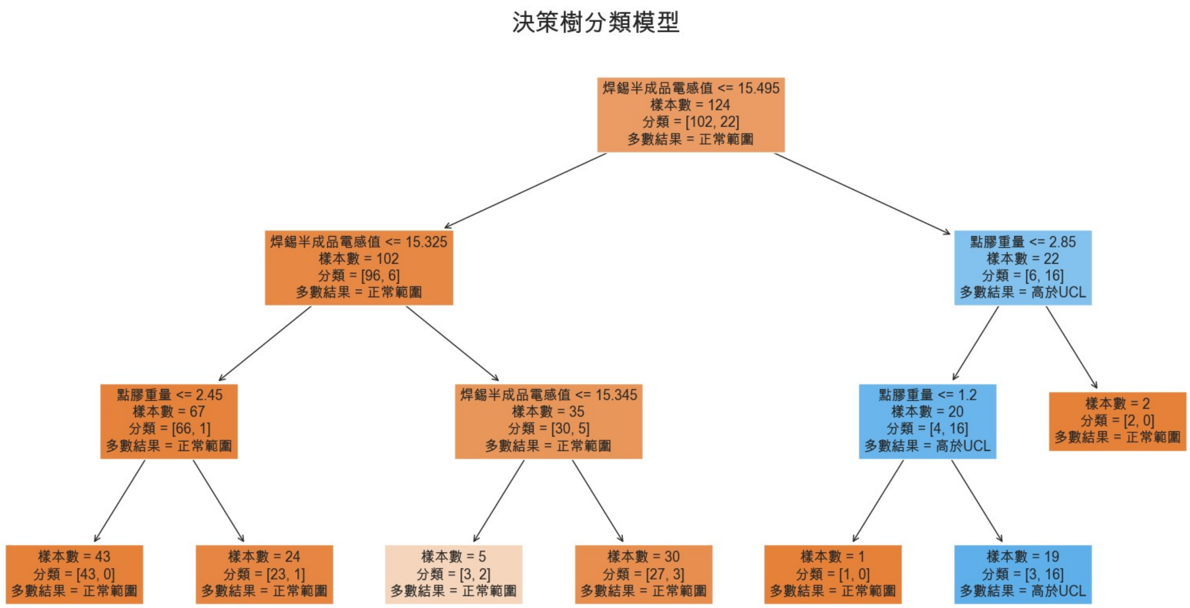
總結

- 向左移動：條件成立。
- 向右移動：條件不成立。
- 節點：最終的分類結果，包含樣本數與類別分佈。

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# 設定Seaborn 的風格
sns.set_style('whitegrid')

# 讀取CSV 資料
file_path = '/Users/hanmingcheng/Documents/python vscode/產
學/dataraw_20241031.csv'
data = pd.read_csv(file_path)

# 檢查並處理缺失值
data = data.dropna(subset=['焊錫半成品電感值', '點膠重量', '塗膠完成品電感
值'])

# 定義控制界限
lcl = 20.95  # 下控制限
ucl = 22.99  # 上控制限

# 將目標變數進行分類
data['y_class'] = pd.cut(
    data['塗膠完成品電感值'],
    bins=[-np.inf, lcl, ucl, np.inf],
    labels=['低於 LCL', '正常範圍', '高於 UCL']
)

# 定義特徵和目標變數
X = data[['焊錫半成品電感值', '點膠重量']]
y_classification = data['y_class']  # 用於分類模型

# 分割資料集
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(
    X, y_classification, test_size=0.2, random_state=42)

# 建立決策樹分類模型，限制最大深度以簡化模型
tree_classifier = DecisionTreeClassifier(random_state=42, max_depth=3)
```

```python
tree_classifier.fit(X_train_c, y_train_c)

# 預測
y_pred_c = tree_classifier.predict(X_test_c)



# 設定中文字體以避免亂碼（針對Mac）
plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']

# 視覺化決策樹分類模型
plt.figure(figsize=(20, 10))
plt.title('決策樹分類模型', fontsize=18)
out = plot_tree(
    tree_classifier,
    feature_names=X.columns,
    class_names=tree_classifier.classes_,
    impurity=False,
    filled=True
)

# 替換圖中文字為中文
for obj in out:
    if hasattr(obj, 'get_text'):
        text = obj.get_text()
        text = text.replace('samples', '樣本數')
        text = text.replace('value', '分類')
        text = text.replace('class', '多數結果')
        obj.set_text(text)
plt.title('決策樹分類模型', fontsize=22)
plt.show()

# 提取決策樹規則
tree_rules = export_text(tree_classifier,
feature_names=list(X.columns), show_weights=True)
print("決策樹規則：\n")
print(tree_rules)

# 顯示特徵重要性
feature_importance = pd.Series(tree_classifier.feature_importances_,
index=X.columns)
print("\n特徵重要性：\n")
print(feature_importance)

# ------------------------------
# 從決策樹中提取規則的函數
def get_rules(tree, feature_names, class_names):
    tree_ = tree.tree_
    feature_name = [
        feature_names[i] if i != _tree.TREE_UNDEFINED else
```

```
"undefined!"
        for i in tree_.feature
    ]

    paths = []
    path = []

    def recurse(node, path):
        if tree_.feature[node] != _tree.TREE_UNDEFINED:
            name = feature_name[node]
            threshold = tree_.threshold[node]
            # 左子節點
            left = tree_.children_left[node]
            path_left = path.copy()
            path_left.append(f"({name} <= {threshold:.2f})")
            recurse(left, path_left)
            # 右子節點
            right = tree_.children_right[node]
            path_right = path.copy()
            path_right.append(f"({name} > {threshold:.2f})")
            recurse(right, path_right)
        else:
            proba = tree_.value[node][0] / tree_.value[node][0].sum()
            class_idx = np.argmax(proba)
            paths.append((path, class_names[class_idx],
proba[class_idx]))

    recurse(0, path)
    return paths

# 獲取並打印決策規則
rules = get_rules(tree_classifier, list(X.columns),
tree_classifier.classes_)

print("\n決策規則：\n")
for path, class_name, proba in rules:
    print("如果：")
    for p in path:
        print(f"  {p}")
    print(f"那麼，類別為：{class_name}，概率為：{proba:.2f}\n")
```




```
決策樹規則：

|--- 焊錫半成品電感值 <= 15.49
|   |--- 焊錫半成品電感值 <= 15.32
|   |   |--- 點膠重量 <= 2.45
|   |   |   |--- weights: [43.00, 0.00] class: 正常範圍
|   |   |--- 點膠重量 >  2.45
|   |   |   |--- weights: [23.00, 1.00] class: 正常範圍
|   |--- 焊錫半成品電感值 >  15.32
|   |   |--- 焊錫半成品電感值 <= 15.35
|   |   |   |--- weights: [3.00, 2.00] class: 正常範圍
|   |   |--- 焊錫半成品電感值 >  15.35
|   |   |   |--- weights: [27.00, 3.00] class: 正常範圍
|--- 焊錫半成品電感值 >  15.49
|   |--- 點膠重量 <= 2.85
|   |   |--- 點膠重量 <= 1.20
|   |   |   |--- weights: [1.00, 0.00] class: 正常範圍
|   |   |--- 點膠重量 >  1.20
|   |   |   |--- weights: [3.00, 16.00] class: 高於 UCL
|   |--- 點膠重量 >  2.85
|   |   |--- weights: [2.00, 0.00] class: 正常範圍


特徵重要性：

焊錫半成品電感值  0.825986
點膠重量    0.174014
dtype: float64
```

```
決策規則：

如果：
  (焊錫半成品電感值 <= 15.49)
  (焊錫半成品電感值 <= 15.32)
  (點膠重量 <= 2.45)
那麼，類別為：正常範圍，概率為：1.00

如果：
  (焊錫半成品電感值 <= 15.49)
  (焊錫半成品電感值 <= 15.32)
  (點膠重量 > 2.45)
那麼，類別為：正常範圍，概率為：0.96

如果：
  (焊錫半成品電感值 <= 15.49)
  (焊錫半成品電感值 > 15.32)
  (焊錫半成品電感值 <= 15.35)
那麼，類別為：正常範圍，概率為：0.60

如果：
  (焊錫半成品電感值 <= 15.49)
  (焊錫半成品電感值 > 15.32)
  (焊錫半成品電感值 > 15.35)
那麼，類別為：正常範圍，概率為：0.90

如果：
  (焊錫半成品電感值 > 15.49)
  (點膠重量 <= 2.85)
  (點膠重量 <= 1.20)
那麼，類別為：正常範圍，概率為：1.00

如果：
  (焊錫半成品電感值 > 15.49)
  (點膠重量 <= 2.85)
  (點膠重量 > 1.20)
那麼，類別為：高於 UCL，概率為：0.84

如果：
  (焊錫半成品電感值 > 15.49)
  (點膠重量 > 2.85)
那麼，類別為：正常範圍，概率為：1.00
```

決策樹視覺化圖表

```python
import pandas as pd
import numpy as np
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
import plotly.graph_objects as go

# 讀取數據
file_path = '/Users/hanmingcheng/Documents/python vscode/產
學/dataraw_20241031.csv'
data = pd.read_csv(file_path)

# 處理缺失值
data = data.dropna(subset=['焊錫半成品電感值', '點膠重量', '塗膠完成品電感
值'])

# 定義控制界限
lcl = 20.95
ucl = 22.99

# 目標變數分類
data['y_class'] = pd.cut(
    data['塗膠完成品電感值'],
    bins=[-np.inf, lcl, ucl, np.inf],
    labels=['低於LCL', '正常範圍', '高於UCL']
)

# 特徵與目標變數
X = data[['焊錫半成品電感值', '點膠重量']]
y = data['y_class']

# 訓練決策樹模型
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
tree_classifier = DecisionTreeClassifier(max_depth=3, random_state=42)
tree_classifier.fit(X_train, y_train)

# 定義節點和流向邏輯
nodes = [
    "根節點 (樣本數: 124)",                          # 0
    "焊錫半成品電感值 ≤ 15.495\n(樣本數: 102)",      # 1
    "焊錫半成品電感值 > 15.495\n(樣本數: 22)",       # 2
    "焊錫半成品電感值 ≤ 15.325\n(樣本數: 67)",       # 3
    "焊錫半成品電感值 > 15.325\n(樣本數: 35)",       # 4
    "點膠重量 ≤ 2.45\n(樣本數: 43)",                 # 5
    "點膠重量 > 2.45\n(樣本數: 24)",                 # 6
    "焊錫電感值 ≤ 15.345\n(樣本數: 5)",              # 7
    "焊錫電感值 > 15.345\n(樣本數: 30)",             # 8
    "點膠重量 ≤ 2.85\n(樣本數: 20)",                 # 9
    "點膠重量 > 2.85\n(樣本數: 2)",                  #10
    "點膠重量 ≤ 1.2\n(樣本數: 1)",                   #11
    "點膠重量 > 1.2\n(樣本數: 19)",                  #12
```

```python
]

# 節點流向(父節點-> 子節點)
sources = [
    0, 0,    # 根節點到第一層
    1, 1,    # 第一層到第二層
    3, 3,    # 第二層左子樹
    4, 4,    # 第二層右子樹
    2, 2,    # 第一層右側
    9, 9     # 節點9 到節點11 和12
]

targets = [
    1, 2,    # 根節點到第一層
    3, 4,    # 第一層到第二層
    5, 6,    # 第二層左子樹
    7, 8,    # 第二層右子樹
    9,10,    # 第一層右側
    11,12    # 節點9 到節點11 和12
]

values = [
    102, 22, # 根節點到第一層
    67, 35,  # 第一層到第二層
    43, 24,  # 第二層左子樹
    5, 30,   # 第二層右子樹
    20, 2,   # 第一層右側
    1, 19    # 節點9 到節點11 和12
]

# 使用Plotly 繪製桑基圖
fig = go.Figure(go.Sankey(
    node=dict(
        pad=25,  # 節點間距
        thickness=20,  # 節點寬度
        line=dict(color="black", width=0.5),  # 節點邊框
        label=nodes,
        color=["#FFDDC1", "#FFABAB", "#FFC3A0", "#D5AAFF", "#85E3FF",
               "#B9FBC0", "#FF9CEE", "#FCF6BD", "#F8C3C3", "#B2F7EF",
               "#A0CED9", "#C6E2E9", "#F5E1FD"]  # 新增顏色
    ),
    link=dict(
        source=sources,  # 父節點索引
        target=targets,  # 子節點索引
        value=values     # 樣本數值
    )
))

# 更新圖表樣式
```

```python
fig.update_layout(
    font=dict(size=18, family="Arial Unicode MS"),
    height=1000, width=1300,
    margin=dict(l=50, r=50, t=30, b=50)
)

# 顯示圖表
fig.show()
```

{"config":{"plotlyServerURL":"https://plot.ly"},"data":[{"link":{"source":[0,0,1,1,3,3,4,4,2,2,9,9],"target":[1,2,3,4,5,6,7,8,9,10,11,12],"value":[102,22,67,35,43,24,5,30,20,2,1,19]},"node":{"color":["#FFDDC1","#FFABAB","#FFC3A0","#D5AAFF","#85E3FF","#B9FBC0","#FF9CEE","#FCF6BD","#F8C3C3","#B2F7EF","#A0CED9","#C6E2E9","#F5E1FD"],"label":["根節點 (樣本數: 124)","焊錫半成品電感值 ≤ 15.495\n(樣本數: 102)","焊錫半成品電感值 > 15.495\n(樣本數: 22)","焊錫半成品電感值 ≤ 15.325\n(樣本數: 67)","焊錫半成品電感值 > 15.325\n(樣本數: 35)","點膠重量 ≤ 2.45\n(樣本數: 43)","點膠重量 > 2.45\n(樣本數: 24)","焊錫電感值 ≤ 15.345\n(樣本數: 5)","焊錫電感值 > 15.345\n(樣本數: 30)","點膠重量 ≤ 2.85\n(樣本數: 20)","點膠重量 > 2.85\n(樣本數: 2)","點膠重量 ≤ 1.2\n(樣本數: 1)","點膠重量 > 1.2\n(樣本數: 19)"],"line":{"color":"black","width":0.5},"pad":25,"thickness":20},"type":"sankey"}],"layout":{"font":{"family":"Arial Unicode MS","size":18},"height":1000,"margin":{"b":50,"l":50,"r":50,"t":30},"template":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpolar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"carpet":[{"aaxis":{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","minorgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","minorgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"choropleth":[{"colorbar":{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],

```
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
```

```
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
```

```
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"width":1
300}}
```


圖表分析：兩個變數之間的關係探討

1. 初步決策樹篩選

- 想法：主要還是想探討焊錫半成品電感值與點膠重量之間如何分割與關係，因此找出兩者的條件做初步比較。
- 第一層：根據焊錫半成品電感值是否小於等於 15.495 分為「低電感」和「高電感」。
- 第二層：根據點膠重量是否小於等於 2.85 分為「低重量」和「高重量」。

2. 圖表概述

- **X** 軸：焊錫半成品電感值。
- **Y** 軸：塗膠完成品電感值。
- 資料點顏色：根據決策樹模型的第一層與第二層條件分類，顯示不同的組別：
  - 紫色：低電感（焊錫電感值 ≤ 15.495）、低重量（點膠重量 ≤ 2.85）。
  - 橙色：低電感（焊錫電感值 ≤ 15.495）、高重量（點膠重量 > 2.85）。
  - 藍色：高電感（焊錫電感值 > 15.495）、低重量（點膠重量 ≤ 2.85）。
  - 綠色：高電感（焊錫電感值 > 15.495）、高重量（點膠重量 > 2.85）。

3. 圖表分析

- 分類邊界：
  - 垂直紅色虛線：焊錫電感值 = 15.495（決策樹第一層分類邊界）。
  - 水平橙色虛線：點膠重量 = 2.85（決策樹第二層分類邊界）。
  - 水平紅色虛線：焊錫電感值 = 15.1 的分界線。
- 分佈觀察：
  - 大多數資料點落在紫色（低電感、低重量）和藍色（高電感、低重量）區域。
  - 橙色（低電感、高重量）和綠色（高電感、高重量）區域的樣本較少，可能表示高重量樣本比例低，但也可以發現高重量的合規率達到 100%。

4. 解讀與發現

- 電感值與完成品品質的關係：
  - 可以發現當焊錫電感值 <= 15.1 時全部的數據都合規。
  - 當焊錫電感值較低時（左側數據點），大多數塗膠完成品電感值也落在合規範圍內，但紫色與黃色點交雜在一起可以再細分割來分析。
  - 當焊錫電感值較高時（右側數據點），部分塗膠完成品電感值接近或超過 UCL。
- 重量對品質的影響：
  - 低重量樣本（點膠重量 ≤ 2.85）分佈較分散，部分超出控制範圍。

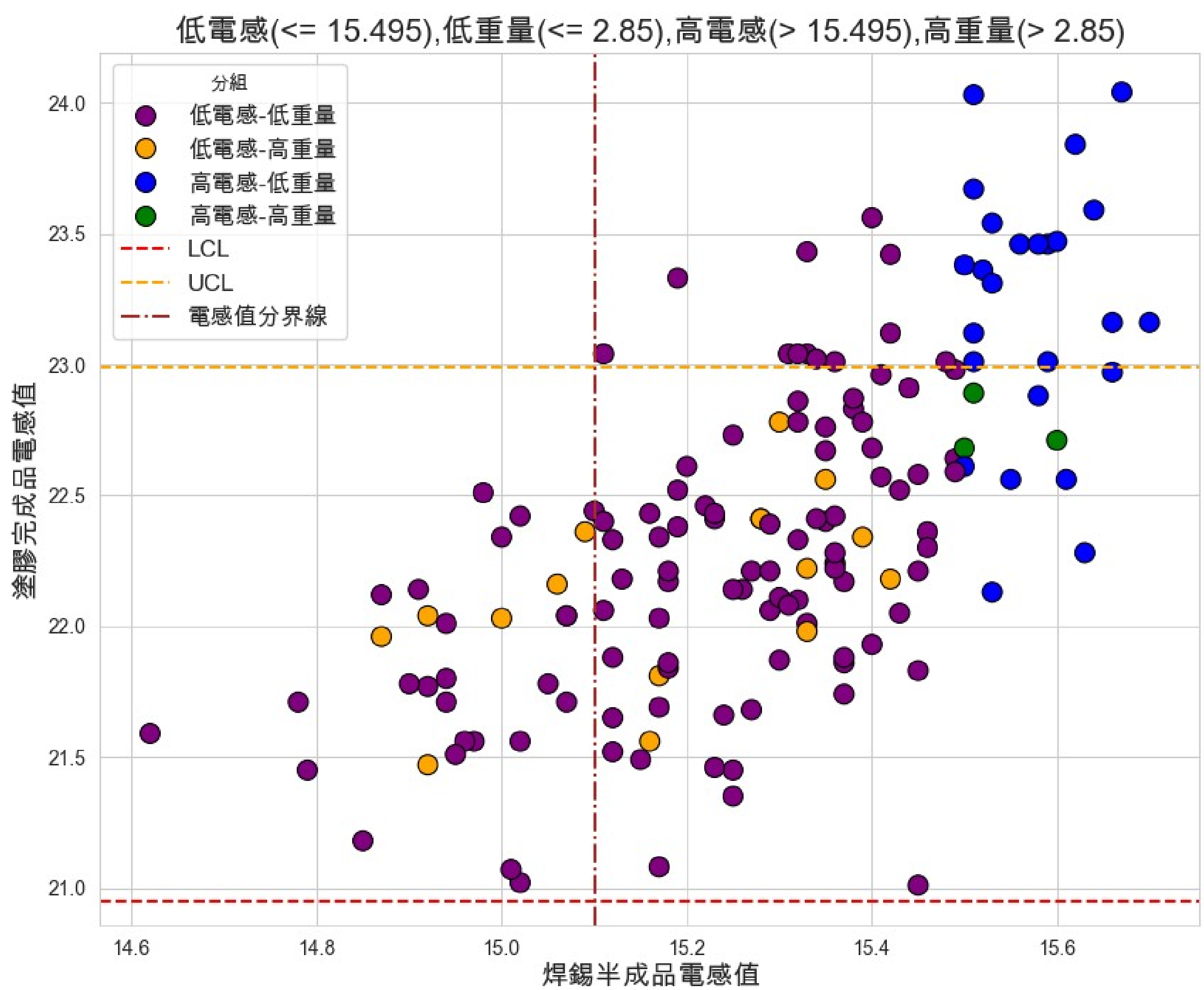
- 高重量樣本（點膠重量 > 2.85）分佈較為集中，塗膠完成品電感值更穩定，可能表示重量對品質有較大影響。

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 更新檔案路徑
file_path = '/Users/hanmingcheng/Documents/python vscode/產
學/dataraw_20241031.csv'  # 更新為您的檔案路徑
data = pd.read_csv(file_path)

# 檢查並處理缺失值
data = data.dropna(subset=['焊錫半成品電感值', '點膠重量', '塗膠完成品電感
值'])

# 定義控制界限
lcl = 20.95  # 下控制限
ucl = 22.99  # 上控制限

# 將目標變數進行分類
data['y_class'] = pd.cut(
    data['塗膠完成品電感值'],
    bins=[-np.inf, lcl, ucl, np.inf],
    labels=['低於 LCL', '正常範圍', '高於 UCL']
)

# 分割資料
low_inductance_data = data[data['焊錫半成品電感值'] <= 15.495]
high_inductance_data = data[data['焊錫半成品電感值'] > 15.495]

# 低電感值組，進一步根據點膠重量分割
low_weight_data = low_inductance_data[low_inductance_data['點膠重量']
<= 2.45]
high_weight_data = low_inductance_data[low_inductance_data['點膠重量']
> 2.45]

# 高電感值組，進一步根據點膠重量分割
normal_weight_data = high_inductance_data[high_inductance_data['點膠重
量'] <= 2.85]
over_weight_data = high_inductance_data[high_inductance_data['點膠重
量'] > 2.85]

# 設定Seaborn 風格和字體
sns.set_style('whitegrid')
plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']  # 解決中文字體顯
示問題

# 定義顏色字典，確保各圖中類別顏色一致
```

```
color_dict = {'低於 LCL': 'red', '正常範圍': 'green', '高於 UCL': 'blue'}

# # 低電感值組：點膠重量<= 2.45
# plt.figure(figsize=(8, 6))
# sns.scatterplot(
#     x=low_weight_data['焊錫半成品電感值'],
#     y=low_weight_data['塗膠完成品電感值'],
#     hue=low_weight_data['y_class'],
#     palette=color_dict,
#     s=100,
#     edgecolor='black'
# )
# plt.axhline(lcl, color='red', linestyle='--', label='LCL')
# plt.axhline(ucl, color='orange', linestyle='--', label='UCL')
# plt.title('低電感值組<= 15.495: 點膠重量<= 2.45', fontsize=14)
# plt.xlabel('焊錫半成品電感值', fontsize=12)
# plt.ylabel('塗膠完成品電感值', fontsize=12)
# plt.legend(title='分類', fontsize=10)
# plt.show()

# # 低電感值組：點膠重量> 2.45
# plt.figure(figsize=(8, 6))
# sns.scatterplot(
#     x=high_weight_data['焊錫半成品電感值'],
#     y=high_weight_data['塗膠完成品電感值'],
#     hue=high_weight_data['y_class'],
#     palette=color_dict,
#     s=100,
#     edgecolor='black'
# )
# plt.axhline(lcl, color='red', linestyle='--', label='LCL')
# plt.axhline(ucl, color='orange', linestyle='--', label='UCL')
# plt.title('低電感值組<= 15.495: 點膠重量> 2.45', fontsize=14)
# plt.xlabel('焊錫半成品電感值', fontsize=12)
# plt.ylabel('塗膠完成品電感值', fontsize=12)
# plt.legend(title='分類', fontsize=10)
# plt.show()

# # 高電感值組：點膠重量<= 2.85
# plt.figure(figsize=(8, 6))
# sns.scatterplot(
#     x=normal_weight_data['焊錫半成品電感值'],
#     y=normal_weight_data['塗膠完成品電感值'],
#     hue=normal_weight_data['y_class'],
#     palette=color_dict,
#     s=100,
#     edgecolor='black'
# )
# plt.axhline(lcl, color='red', linestyle='--', label='LCL')
```

```python
# plt.axhline(ucl, color='orange', linestyle='--', label='UCL')
# plt.title('高電感值組> 15.495: 點膠重量<= 2.85', fontsize=14)
# plt.xlabel('焊錫半成品電感值', fontsize=12)
# plt.ylabel('塗膠完成品電感值', fontsize=12)
# plt.legend(title='分類', fontsize=10)
# plt.show()

# # 高電感值組：點膠重量> 2.85
# plt.figure(figsize=(8, 6))
# sns.scatterplot(
#     x=over_weight_data['焊錫半成品電感值'],
#     y=over_weight_data['塗膠完成品電感值'],
#     hue=over_weight_data['y_class'],
#     palette=color_dict,
#     s=100,
#     edgecolor='black'
# )
# plt.axhline(lcl, color='red', linestyle='--', label='LCL')
# plt.axhline(ucl, color='orange', linestyle='--', label='UCL')
# plt.title('高電感值組> 15.495: 點膠重量> 2.85', fontsize=14)
# plt.xlabel('焊錫半成品電感值', fontsize=12)
# plt.ylabel('塗膠完成品電感值', fontsize=12)
# plt.legend(title='分類', fontsize=10)
# plt.show()

# 根據兩個條件進行多重分組
conditions = [
    (data['焊錫半成品電感值'] <= 15.495) & (data['點膠重量'] <= 2.85),
    (data['焊錫半成品電感值'] <= 15.495) & (data['點膠重量'] > 2.85),
    (data['焊錫半成品電感值'] > 15.495) & (data['點膠重量'] <= 2.85),
    (data['焊錫半成品電感值'] > 15.495) & (data['點膠重量'] > 2.85)
]
choices = ['低電感-低重量', '低電感-高重量', '高電感-低重量', '高電感-高重量']
data['group'] = np.select(conditions, choices, default='未分類')

# 為分組指定更明顯的顏色
group_color_dict = {
    '低電感-低重量': 'purple',
    '低電感-高重量': 'orange',
    '高電感-低重量': 'blue',
    '高電感-高重量': 'green',
    '未分類': 'black'
}

# 繪製分組的散佈圖
plt.figure(figsize=(10, 8))
sns.scatterplot(
    data=data,
    x='焊錫半成品電感值',
```

```
    y='塗膠完成品電感值',
    hue='group',
    palette=group_color_dict,
    s=100,
    edgecolor='black'
)
plt.axhline(lcl, color='red', linestyle='--', label='LCL')
plt.axhline(ucl, color='orange', linestyle='--', label='UCL')
plt.title('低電感(<= 15.495),低重量(<= 2.85),高電感(> 15.495),高重量(>
2.85)', fontsize=16)
plt.axvline(15.1, color='brown', linestyle='-.', label='電感值分界線')
plt.xlabel('焊錫半成品電感值', fontsize=14)
plt.ylabel('塗膠完成品電感值', fontsize=14)
plt.legend(title='分組', fontsize=12)
plt.show()
```



決策樹分割後數據分析

1. 圖表簡介

五張圖根據決策樹的分割條件展示了「焊錫半成品電感值」與「點膠重量」對「塗膠完成品電感值」的影響。每張圖的分組條件由決策樹決定。

---

2. 各圖分析

第一張圖：`焊錫電感值 ≤ 15.325 且 點膠重量 ≤ 2.45`

- 分佈分析：
  - 焊錫電感值與塗膠完成品電感值呈現較為集中的分佈。
  - 點膠重量多數在 1.0 至 2.5 之間，數據點較多。
- 觀察結論：
  - 塗膠完成品電感值大多在合規界限內，品質穩定。

第二張圖：`焊錫電感值 ≤ 15.325 且 點膠重量 > 2.45`

- 分佈分析：
  - 點膠重量高於 2.45 的樣本分佈較分散。
- 觀察結論：
  - 高重量可能導致完成品電感值波動略有增加，但大部分還是都落在合規界線中。

第三張圖：`15.325 < 焊錫電感值 ≤ 15.495`

- 分佈分析：
  - 此分組數據點分散，有部分超出上界線。
  - 但大部分塗膠完成品電感值基本穩定。
- 觀察結論：
  - 可以看出這區間的資料點在上界線附近游移，可能樣本數多一點結果會有所不同。

第四張圖：`焊錫電感值 > 15.495 且 點膠重量 ≤ 2.85`

- 分佈分析：
  - 點膠重量集中在 2.5 左右，焊錫電感值明顯超過 15.495。
  - 大部分完成品電感值超過 UCL。
- 觀察結論：
  - 過多超過上界線，因此範圍內的高電感樣本需要額外關注，特別是低重量樣本。

第五張圖：`焊錫電感值 > 15.495 且 點膠重量 > 2.85`

- 分佈分析：

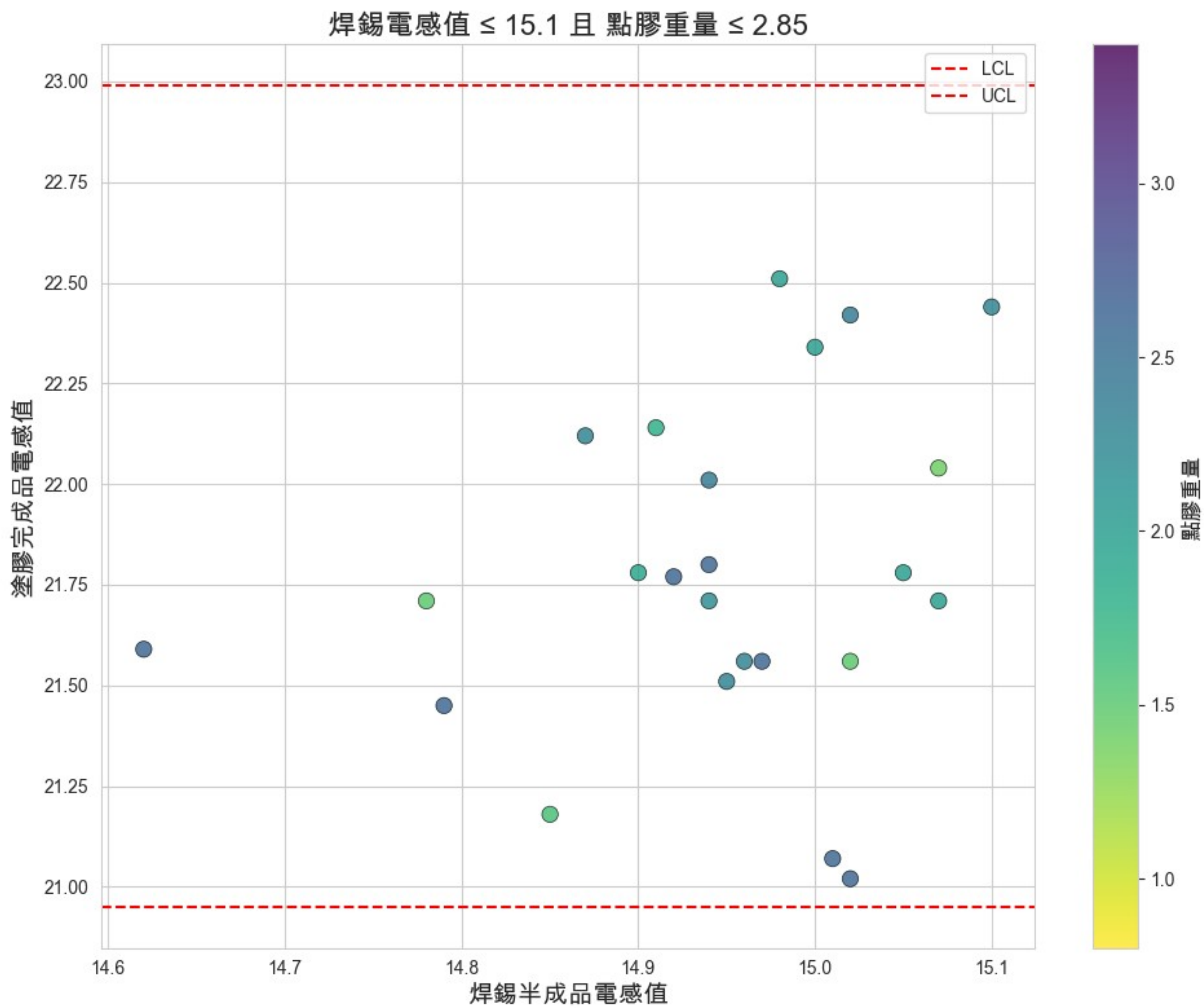
- 數據點極少，幾乎無法觀察明顯趨勢。
- 觀察結論：
    - 此分組較為罕見，可忽略。

第六張圖：`點膠重量 > 2.85`

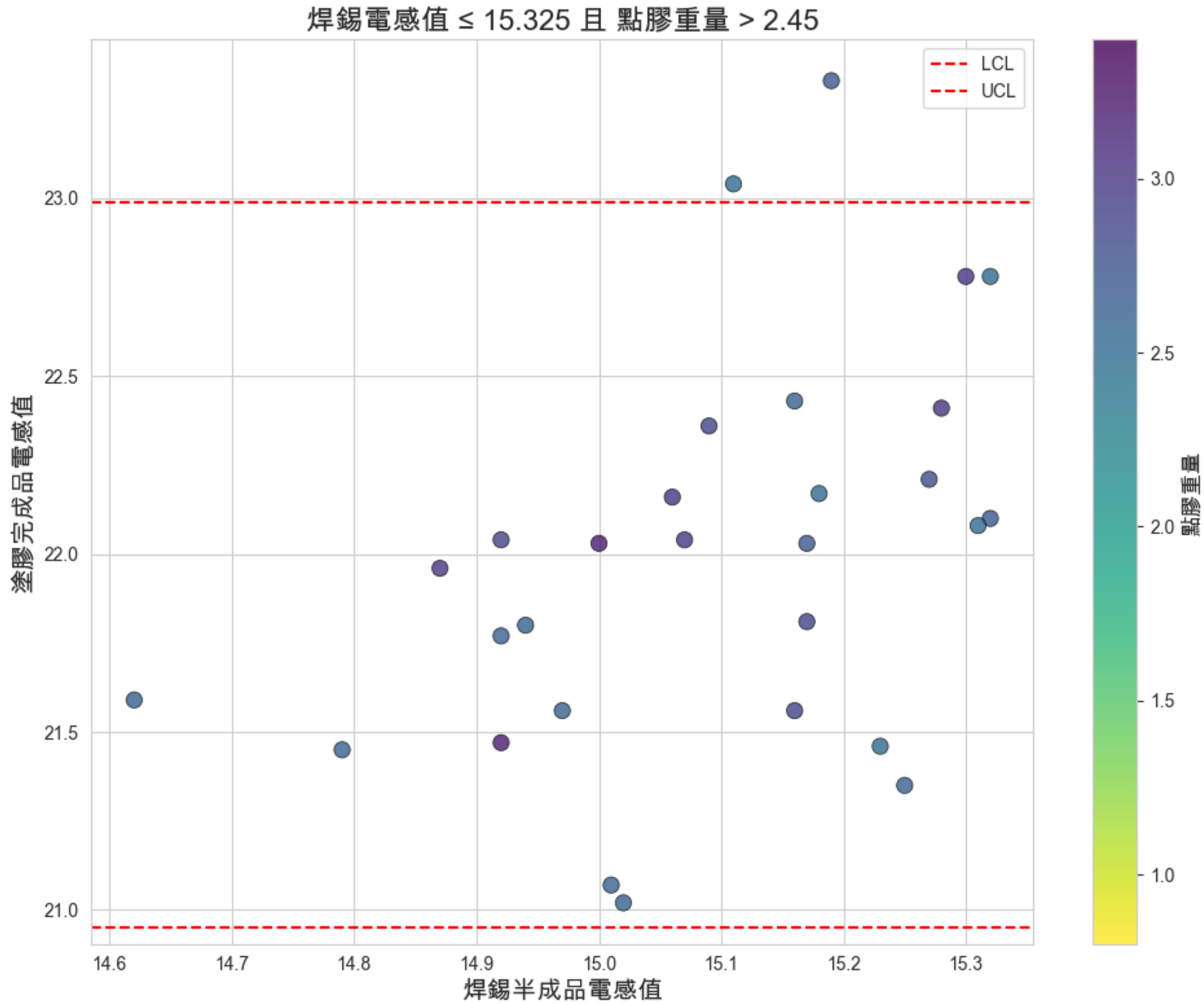
- 分佈分析：
    - 數據點較為分散，無明顯趨勢。
- 觀察結論：
    - 資料分散但可以觀察該切割都進入合規範圍。

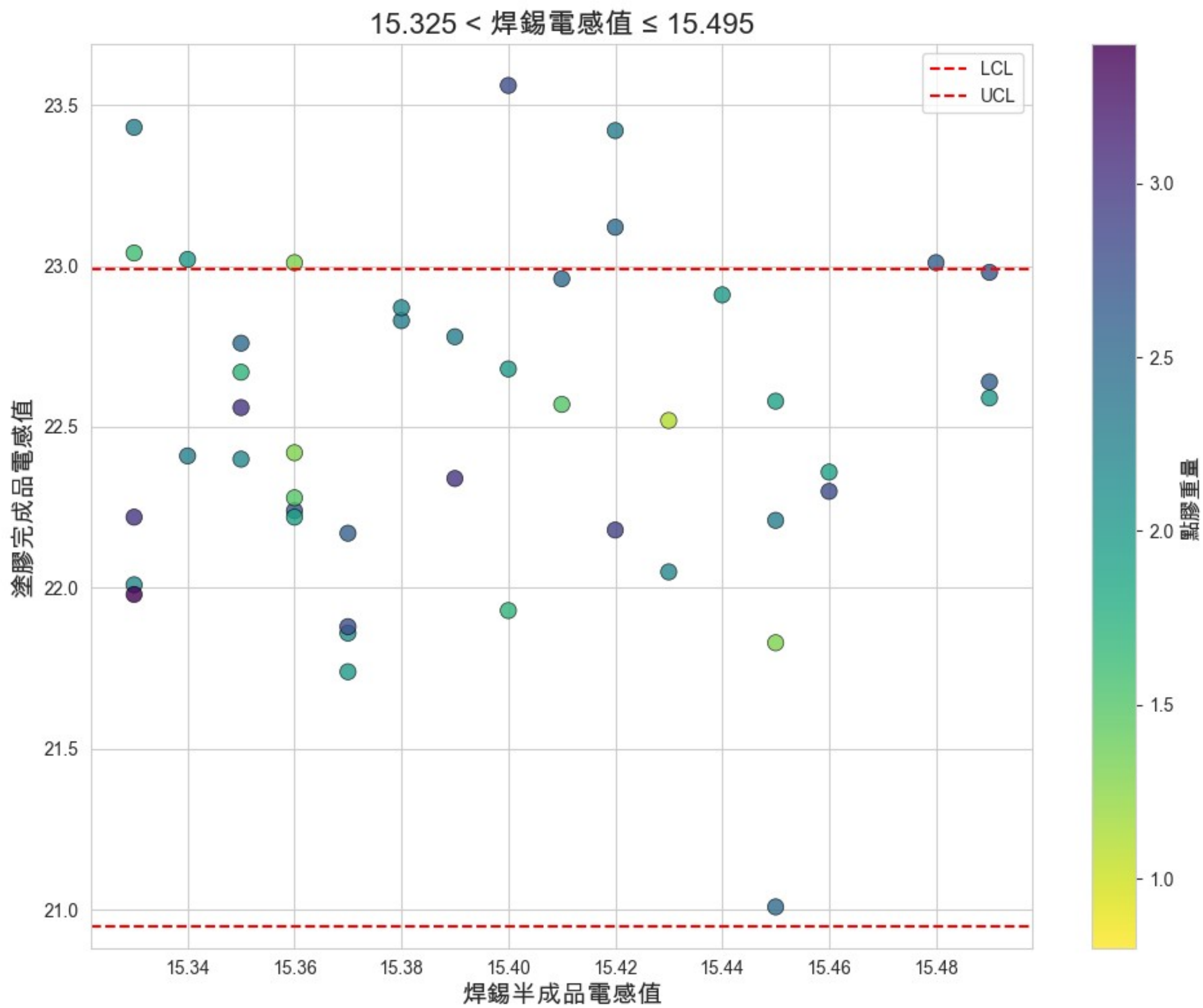
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import cm

# 更新文件路徑
file_path = '/Users/hanmingcheng/Documents/python vscode/產學/dataraw_20241031.csv'

# 讀取數據
data = pd.read_csv(file_path)

# 取得點膠重量的最小與最大值
weight_min = data['點膠重量'].min()
weight_max = data['點膠重量'].max()

# 定義控制界限
ucl = 22.99
lcl = 20.95

# 定義基於決策樹條件的分層
conditions = [
    ("焊錫電感值 ≤ 15.1 且 點膠重量 ≤ 2.85",
     (data['焊錫半成品電感值'] <= 15.1) & (data['點膠重量'] <= 2.85)),
    ("焊錫電感值 ≤ 15.325 且 點膠重量 > 2.45",
     (data['焊錫半成品電感值'] <= 15.325) & (data['點膠重量'] > 2.45)),
    ("15.325 < 焊錫電感值 ≤ 15.495",
     (data['焊錫半成品電感值'] > 15.325) & (data['焊錫半成品電感值'] <= 15.495)),
    ("焊錫電感值 > 15.495 且 點膠重量 ≤ 2.85",
     (data['焊錫半成品電感值'] > 15.495) & (data['點膠重量'] <= 2.85)),
    ("焊錫電感值 > 15.495 且 點膠重量 > 2.85",
     (data['焊錫半成品電感值'] > 15.495) & (data['點膠重量'] > 2.85)),
    ("點膠重量 > 2.78",
     (data['點膠重量'] > 2.78))
]

# 定義顏色映射
cmap = cm.viridis.reversed()
```

```python
# 為每個分層條件繪製散佈圖
for title, condition in conditions:
    subset = data[condition]

    plt.figure(figsize=(10, 8))
    scatter = plt.scatter(
        subset['焊錫半成品電感值'],
        subset['塗膠完成品電感值'],
        c=subset['點膠重量'],    # 使用實際的點膠重量
        cmap=cmap,
        vmin=weight_min,        # 設定顏色映射最小值
        vmax=weight_max,        # 設定顏色映射最大值
        s=80,
        edgecolors='black',
        linewidth=0.5,
        alpha=0.8
    )

    # 添加控制界限
    plt.axhline(y=lcl, color='red', linestyle='--', linewidth=1.5,
label='LCL')
    plt.axhline(y=ucl, color='red', linestyle='--', linewidth=1.5,
label='UCL')

    # 添加標籤和標題
    plt.xlabel('焊錫半成品電感值', fontsize=14)
    plt.ylabel('塗膠完成品電感值', fontsize=14)
    plt.title(f'{title}', fontsize=16)

    # 添加顏色條
    cbar = plt.colorbar(scatter)
    cbar.set_label('點膠重量', fontsize=12)

    # 優化佈局並顯示圖表
    plt.legend(loc='upper right')
    plt.tight_layout()
    plt.show()
```

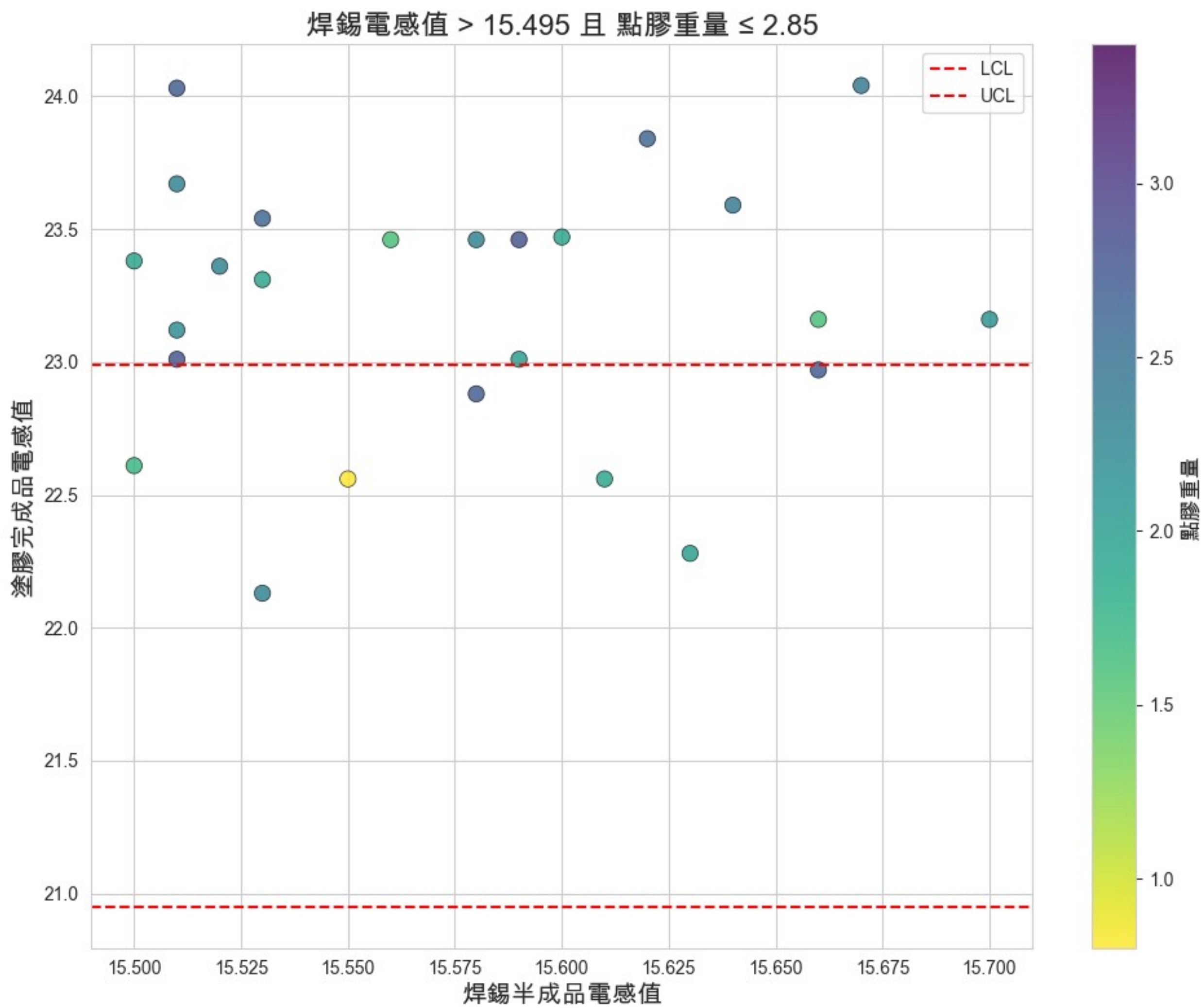


焊錫電感值 ≤ 15.1 且 點膠重量 ≤ 2.85
LCL
UCL
塗膠完成品電感值
焊錫半成品電感值
點膠重量
23.00
22.75
22.50
22.25
22.00
21.75
21.50
21.25
21.00
14.6
14.7
14.8
14.9
15.0
15.1
3.0
2.5
2.0
1.5
1.0

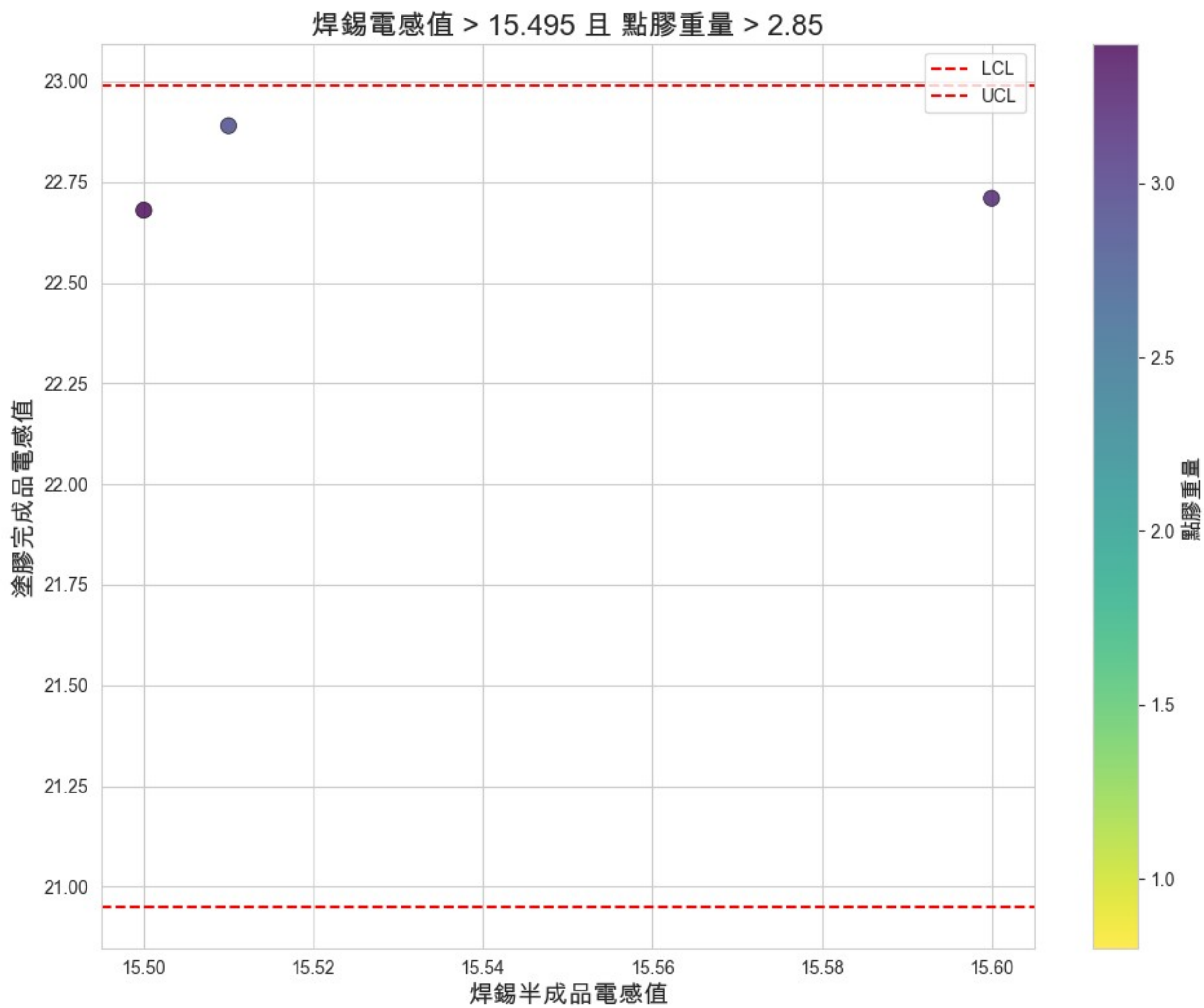

焊錫電感值 ≤ 15.325 且 點膠重量 > 2.45
LCL
UCL
塗膠完成品電感值
焊錫半成品電感值
點膠重量
23.0
22.5
22.0
21.5
21.0
14.6
14.7
14.8
14.9
15.0
15.1
15.2
15.3
3.0
2.5
2.0
1.5
1.0

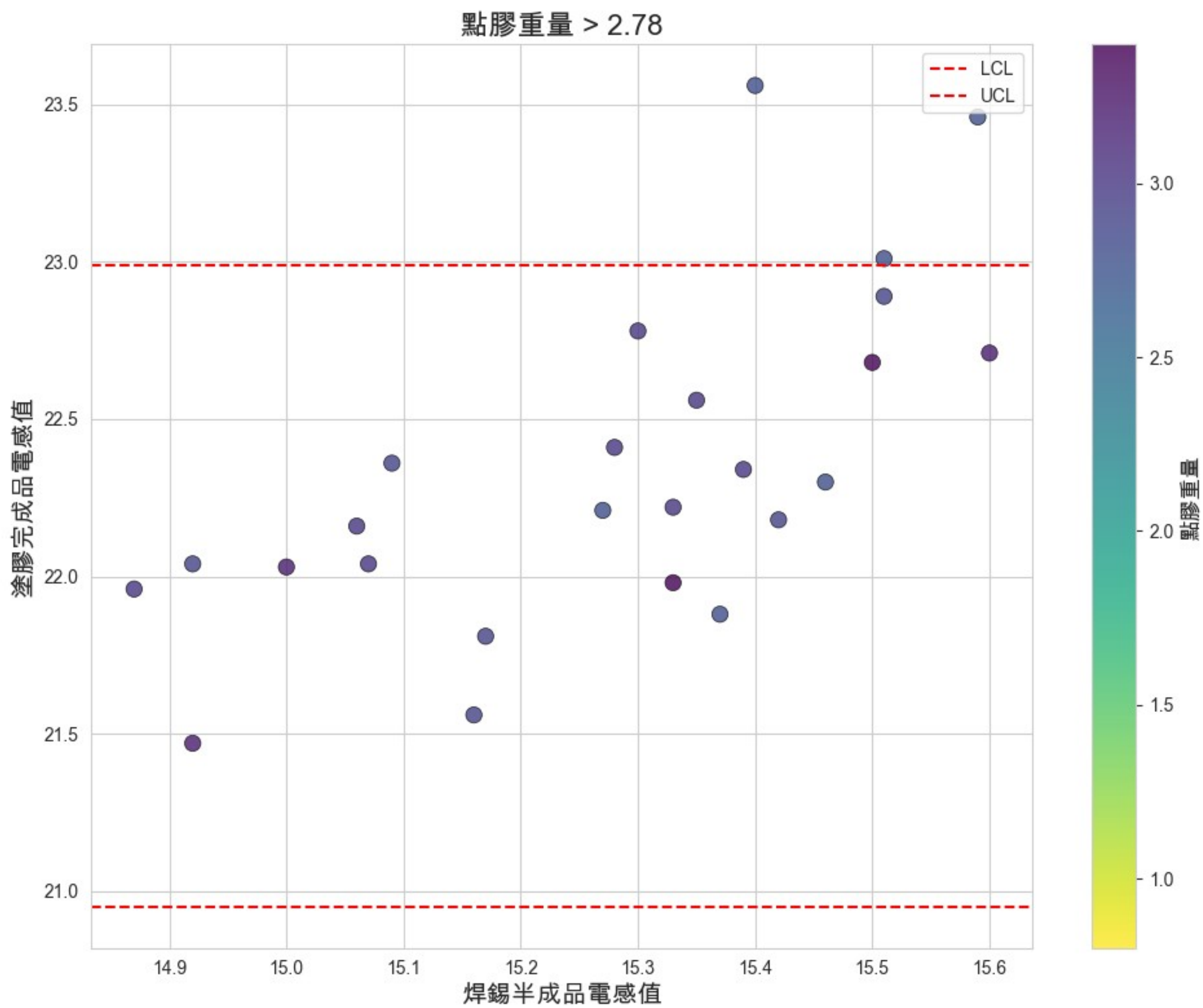


15.325 < 焊錫電感值 ≤ 15.495
LCL
UCL
塗膠完成品電感值
焊錫半成品電感值
點膠重量
23.5
23.0
22.5
22.0
21.5
21.0
15.34
15.36
15.38
15.40
15.42
15.44
15.46
15.48
3.0
2.5
2.0
1.5
1.0



焊錫電感值 > 15.495 且 點膠重量 ≤ 2.85
LCL
UCL
24.0
23.5
23.0
22.5
22.0
21.5
21.0
塗膠完成品電感值
15.500
15.525
15.550
15.575
15.600
15.625
15.650
15.675
15.700
焊錫半成品電感值
3.0
2.5
2.0
1.5
1.0
點膠重量


焊錫電感值 > 15.495 且 點膠重量 > 2.85
LCL
UCL
23.00
22.75
22.50
22.25
22.00
21.75
21.50
21.25
21.00
塗膠完成品電感值
15.50
15.52
15.54
15.56
15.58
15.60
焊錫半成品電感值
3.0
2.5
2.0
1.5
1.0
點膠重量



最後就看標做總整理的部分

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import cm

# 更新文件路徑
file_path = '/Users/hanmingcheng/Documents/python vscode/產
學/dataraw_20241031.csv'

# 讀取數據
data = pd.read_csv(file_path)

# 正規化'點膠重量' 作為顏色映射
data['W_norm'] = (data['點膠重量'] - data['點膠重量'].min()) / (data['點
膠重量'].max() - data['點膠重量'].min())
```

```python
# 定義控制界限
ucl = 22.99
lcl = 20.95

conditions = [
    ("焊錫電感值≤15.1 且點膠重量≤2.85",
     (data['焊錫半成品電感值'] <= 15.1) & (data['點膠重量'] <= 2.5)),
    ("焊錫電感值≤15.325 且點膠重量> 2.45",
     (data['焊錫半成品電感值'] <= 15.325) & (data['點膠重量'] > 2.45)),
    ("15.325 < 焊錫電感值≤15.495",
     (data['焊錫半成品電感值'] > 15.325) & (data['焊錫半成品電感值'] <=
15.495)),
    ("焊錫電感值> 15.495 且點膠重量≤2.85",
     (data['焊錫半成品電感值'] > 15.495) & (data['點膠重量'] <= 2.85)),
    ("焊錫電感值> 15.495 且點膠重量> 2.85",
     (data['焊錫半成品電感值'] > 15.495) & (data['點膠重量'] > 2.85)),
    ("點膠重量> 2.79",
     (data['點膠重量'] > 2.79))
]

# 計算每個分層條件下的良率
results = []

for title, condition in conditions:
    subset = data[condition]

    # 總樣本數
    total_points = len(subset)

    # 點在控制界限內的數量
    in_control_points = subset[(subset['塗膠完成品電感值'] >= lcl) &
(subset['塗膠完成品電感值'] <= ucl)].shape[0]

    # 計算良率
    yield_rate = (in_control_points / total_points) * 100 if
total_points > 0 else 0

    # 儲存結果
    results.append({
        'Condition': title,
        'Total Points': total_points,
        'In Control Points': in_control_points,
        'Yield Rate (%)': yield_rate
    })

# 將結果轉為DataFrame 並顯示
results_df = pd.DataFrame(results)

# 使用print 顯示結果
```

```
print("每個條件下的良率：")
# print(results_df)
# 如果您使用的是Jupyter Notebook，可以使用display
from IPython.display import display
display(results_df)
```

```
每個條件下的良率：

                         Condition  Total Points  In Control Points  \
0    焊錫電感值 ≤ 15.1 且 點膠重量 ≤ 2.85            17                 17
1  焊錫電感值 ≤ 15.325 且 點膠重量 > 2.45            29                 27

2        15.325 < 焊錫電感值 ≤ 15.495            45                 37
3  焊錫電感值 > 15.495 且 點膠重量 ≤ 2.85            25                  7

4  焊錫電感值 > 15.495 且 點膠重量 > 2.85             3                  3

5                      點膠重量 > 2.79            25                 22

   Yield Rate (%)
0      100.000000
1       93.103448
2       82.222222
3       28.000000
4      100.000000
5       88.000000
```

樣本之間變異數檢定，觀察之間差異

```
import pandas as pd
from scipy.stats import ttest_rel

# 讀取資料
file_path = '/Users/hanmingcheng/Documents/python vscode/產學/L&重量-表格22.csv'
df = pd.read_csv(file_path, header=1)

# 設定欄位名稱（根據你給的截圖）
tool_cols = [
    '塗膠完成品\n製工具測試',
    '塗膠完成品\n製工具測試\n重複測試1',
    '塗膠完成品\n製工具測試\n重複測試2'
]
hand_cols = [
    '塗膠完成品\n手工測試1',
    '塗膠完成品\n手工測試2',
    '塗膠完成品\n手工測試3'
]
```

```python
# 將欄位值轉為數值，忽略文字欄位（轉為NaN）
tool_df = df[tool_cols].apply(pd.to_numeric, errors='coerce')
hand_df = df[hand_cols].apply(pd.to_numeric, errors='coerce')

# 計算每列變異數
tool_var = tool_df.var(axis=1, ddof=1)
hand_var = hand_df.var(axis=1, ddof=1)

# 去除含有NaN 的行
valid_data = pd.DataFrame({'tool_var': tool_var, 'hand_var':
hand_var}).dropna()

# 成對樣本t 檢定
t_stat, p_value = ttest_rel(valid_data['tool_var'],
valid_data['hand_var'])

# 輸出結果
print("\n=== 成對樣本 t 檢定結果 ===")
print(f"樣本數：{len(valid_data)}")
print(f"工具測試變異數平均：{valid_data['tool_var'].mean():.6f}")
print(f"手工測試變異數平均：{valid_data['hand_var'].mean():.6f}")
print(f"t 值：{t_stat:.4f}")
print(f"p 值：{p_value:.4f}")
if p_value < 0.05:
    print("結果：工具與手工測試的變異數存在顯著差異。")
else:
    print("結果：工具與手工測試的變異數無顯著差異。")
```

```
=== 成對樣本 t 檢定結果 ===
樣本數：154
工具測試變異數平均：0.012079
手工測試變異數平均：0.000655
t 值：3.2705
p 值：0.0013
結果：工具與手工測試的變異數存在顯著差異。
```