國立陽明交通大學

NATIONAL YANG MING CHIAO TUNG UNIVERSITY

國立陽明交通大學資訊管理研究所

資料探勘研究與實務—學期專題報告

【結合半導體產業現況與發展趨勢：利用 SECOM Dataset 進行良率預測與智能製造應用探討】

組員

313708003 洪瑄妤

313708010 李苡琛

313708011 邱昕沛

313708017 龔鈺婷

指導老師

劉敦仁 教授

目錄

壹、 動機與目的 2

貳、 資料集敘述 3

參、 分析工具 4

一、 **Random Forest** 4

二、 **Logistic Regression** 6

三、 **XGBoost (eXtreme Gradient Boosting)** 7

四、 **LSTM (Long Short-Term Memory)** 7

五、 **AutoEncoder** 10

肆、 實作與評估方法 11

【Random Forest】 11

【Logistic Regression】 14

【XGBoost】 18

【LSTM】 25

【AutoEncoder】 27

伍、 流程 30

陸、 分析結果與結論 31

壹、動機與目的

一、半導體產業的核心地位與挑戰

半導體產業是現代科技與工業發展的核心支柱，其產品應用遍及生活的方方面面，在這樣高度依賴技術的產業中，製造良率的高低往往是企業獲利能力與市場競爭力的關鍵指標。

隨著人工智慧（AI）和高效能運算（HPC）技術的快速發展，市場對高品質晶片的需求逐年攀升，如何有效提升良率，不僅是生產效益的核心，更是面對全球供應鏈競爭與科技創新的重要武器。智能製造的興起進一步為優化製造流程提供了更多可能性，讓半導體製程進入智慧化與精準化的嶄新階段。

二、為何選擇 SECOM 資料集

SECOM 資料集來自 UCI 機器學習資料庫，包含了半導體製程中多維測試參數與產品良率的相關數據，模擬了真實生產過程中的關鍵數據，為研究製程參數與良率的關聯性提供了重要的基礎。

這個資料集的優勢在於高維度特徵與真實生產場景的緊密結合，能讓研究者深入分析製造過程中的潛在問題，並開發高效的預測模型，此外，存在著處理缺失值與高維數據挑戰的特性，剛好契合了半導體產業中面臨的數據挑戰，為構建智慧製造解決方案提供了寶貴的機會。

三、展望 2025 年的產業趨勢

根據國際數據公司（IDC）與國際半導體產業協會（SEMI）等專業機構的預測，到了 2025 年，全球半導體產業將持續增長，AI 與高效能運算需求將推動市場增長超過 15%，而記憶體技術需求的年增幅甚至可能突破 24%。這些數據顯示，高端技術的滲透率將全面提升，同時製程技術的競爭將集中於 2 奈米以下的尖端領域，2 奈米製程的量產不僅是技術實力的象徵，更將成為市場優勢的分水嶺，且先進封裝技術的廣泛應用將成為製造商提升產品性能與市場競爭力的關鍵環節。

這些趨勢表明，良率的精確控制和製程穩定性將直接影響產業未來格局，對於預測模型的需求，也將不再僅限於理論分析，而是實際應用於製程優化、效率提升和成本控制的核心工具。

四、本研究的目標

本研究聚焦於 SECOM 資料集的應用，結合數據探勘技術，探索如何提升半導體製造中的良率。具體目標包括：

1. 分析資料中的高維特徵與缺失值，找出影響良率的主要因素。
2. 建立初步的預測模型，驗證其在小規模數據集上的效果。
3. 提出基於模型結果的製程優化建議，並探討其在智能製造中的應用可能性。

透過本研究，我們希望驗證基礎模型的實用性，為後續研究奠定數據與分析基礎，搭建學術研究與產業實踐之間的橋樑，為半導體製造良率的全面提升與智能製造的落地提供強有力的支撐。

貳、資料集敘述

一、資料集結構

SECOM 資料集包含 1567 筆樣本，每個樣本有 591 個特徵，代表單一生產實體（production entity），即一個特定的生產單位，而這些特徵是透過測量得來的，可能包括生產過程中的感測器數據或操作參數。

二、目標標籤

目標標籤分為 -1 與 1 兩類。 -1 表示該生產實體「通過測試」（Pass）； 1 表示該生產實體「未通過測試」（Fail）。其中，時間戳記（timestamp）記錄了該測試的具體時間點。

參、分析工具

本專題共使用五個模型來分析此資料集，模型為 Random Forest、Logistic Regression、XGBoost (eXtreme Gradient Boosting)、LSTM (Long Short-Term Memory) 和 AutoEncoder。在該部分內容中，主要為分析工具之介紹以及各模型之優缺點分析。

一、 Random Forest

1. Random Forest

是一種集成學習方法，通過多個決策樹的投票來進行分類。

(1) 主要模組：使用 RandomForestClassifier 來自 sklearn.ensemble。

(2) 優點：

- 高準確度：隨機森林通常能夠提供較高的準確率，並且能夠處理高維度的數據。
- 抗過擬合：隨機森林通過集成多個模型來減少過擬合的風險，比單一決策樹更加穩定。
- 處理不平衡數據：對於不平衡數據，隨機森林能夠提供比較穩定的結果，並且可以進行類別權重調整。

(3) 缺點：

- 訓練時間長：隨著樹的數量增加，隨機森林的訓練時間會較長，特別是當數據集較大時。
- 記憶體消耗大：隨著模型規模的增大，隨機森林需要更多的內存來儲存多個決策樹。
- 解釋性差：隨機森林是一個“黑箱”模型，難以解釋每個決策樹是如何作出最終預測的，這對需要解釋的場景可能不夠理想。

2. SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE 透過生成少數類別的合成樣本來平衡各類別的樣本數量，是一種處理類別不平衡的過採樣方法。

(1) 主要模組：使用 SMOTE 來自 imblearn.over_sampling。

(2) 優點：

- 處理類別不平衡：能夠有效解決類別不平衡問題，增強少數類別的預測能力。

- 避免信息丟失：與隨機過採樣相比，SMOTE 生成的樣本是合成的，而不是重複的，這樣能保留更多的數據信息。

(3) 缺點：

- 可能引入噪聲：合成的樣本可能會使得模型學到一些無用或錯誤的模式，特別是在數據本身噪聲較多的情況下。
- 不適合所有場景：如果少數類別的樣本數非常少，SMOTE 的效果可能並不顯著。

3. GridSearchCV (超參數調整)

通過設置一組參數範圍，尋找最適合的模型參數組合，以提高模型性能，是一種超參數搜索方法。

(1) 主要模組：使用 GridSearchCV 來自 sklearn.model_selection。

(2) 優點：

- 全面的超參數調整：能夠遍歷所有可能的參數組合，尋找到最佳模型配置。
- 簡單易用：可以非常方便地與其他模型進行結合，使用 fit 方法自動訓練最佳模型。

(3) 缺點：

- 計算開銷大：隨著參數範圍和模型複雜度的增加，計算成本會急劇上升，特別是當資料集規模大時。
- 過擬合的風險：如果使用不當，可能會導致過度擬合，特別是在網格搜索過程中對訓練數據的過多調整。

4. SelectFromModel (特徵選擇)

根據某個已訓練的模型來篩選出最重要的特徵，是基於模型的特徵選擇方法。

(1) 主要模組：使用 SelectFromModel 來自 sklearn.feature_selection。

(2) 優點：

- 自動選擇最重要的特徵：通過選擇具有較高重要性的特徵來減少維度，並提高模型的運算效率。
- 減少過擬合：通過特徵選擇，去除不相關或多餘的特徵，有助於提高模型的泛化能力。

(3) 缺點：

- 依賴於基模型：選擇的特徵重要性是基於某一基模型的，因此對基模型的選擇非常依賴，可能會受到模型偏差的影響。
- 特徵選擇的過程可能導致信息損失：過度選擇特徵可能會損失有用的數據。

5. Imbalanced Pipeline (SMOTE + 模型整合)

這將 SMOTE 和模型訓練流程結合在一起。

(1) 主要模組：使用 ImbPipeline 來自 imblearn.pipeline。

(2) 優點：

- 保證數據處理與建模流程一致：將數據處理（如 SMOTE）與模型訓練集成到同一管道中，保證了數據處理的每個步驟都與訓練過程匹配。
- 提高數據處理和模型的效率：能夠自動化數據處理與建模流程，減少錯誤，並提高工作效率。

(3) 缺點：

- 計算開銷大：整合了 SMOTE 和模型訓練後，整體的計算開銷可能會更高，特別是當數據集較大時。
- 過擬合風險：如同單獨使用 SMOTE，一旦處理不當，可能會過擬合於少數類別樣本。

二、 Logistic Regression

是一種線性模型，基於邏輯回歸的假設，輸出結果會是一個類別的概率，模型通過最大化對數似然來擬合數據，用於二元分類問題。

1. 主要模組：使用 LogisticRegression 來自 sklearn.linear_model。

2. 優點：

- 簡單且高效：邏輯回歸是一個簡單且計算效率較高的分類算法，適用於基礎的分類問題。
- 可解釋性強：模型的係數能夠提供關於特徵與預測結果的關係，對於一些需要解釋的場景非常有幫助。
- 性能較好：在特徵與類別之間存在線性關係的情況下，邏輯回歸表現不錯。

3. 缺點：
   - 假設過於簡單：邏輯回歸假設特徵與目標變量之間具有線性關係，對於複雜的非線性問題，性能會受到限制。
   - 對不平衡數據敏感：在處理不平衡數據時，邏輯回歸的效果可能較差，特別是當少數類別的樣本較少時。
   - 容易過擬合：當特徵過多或數據維度過高時，邏輯回歸可能會過擬合，尤其在沒有正則化的情況下。

三、 XGBoost (eXtreme Gradient Boosting)

是一個基於提升樹 (Boosting Trees) 的方法，每個後續的樹都會學習之前模型的錯誤，並進一步改進預測結果。

1. 主要模組：使用 XGBClassifier 來自 xgboost.sklearn、VotingClassifier 來自 sklearn.ensemble。

2. 優點：
   - 高效性能：支持並行計算，能有效利用多核 CPU 和 GPU，訓練速度非常快。
   - 防止過擬合：支持內建正則化（L1 和 L2），可以防止模型過於複雜。
   - 靈活性：提供多種目標函數（如回歸、分類、排序）和自定義損失函數的支持，靈活應對不同的任務需求。

3. 缺點：
   - 參數調整複雜：超參數眾多，需要大量時間進行調試。
   - 內存消耗高：在處理高維數據或非常大的數據集時，對內存需求較高，不適合資源有限的情況。
   - 模型解釋性較低：作為集成模型，由多棵樹構成，對特徵的單一影響難以直觀理解。

四、 LSTM (Long Short-Term Memory)

1. LSTM

是一種特殊的循環神經網路（RNN），專門用來處理序列數據（如時間序列、語音或文本）。

(1) 主要模組：使用 LSTM 與 Dense 來自 tensorflow.keras.layers ，以及 Sequential 來自 tensorflow.keras.models。

(2) 優點：

- 處理長期依賴：LSTM 能夠記住長期的時間序列依賴性，適合處理序列數據（如股票預測、語音識別）。
- 靈活性：適用於分類、回歸、生成等多種任務。
- 穩定性強：通過記憶門控機制，LSTM 可以有效地減少梯度消失問題。

(3) 缺點：

- 計算開銷大：LSTM 訓練需要較長時間，特別是在數據規模大時。
- 參數調整困難：需要對網絡結構（如單元數量、層數等）進行多次嘗試才能獲得最佳性能。
- 難以解釋：雖然能捕捉時序模式，但內部機制對人類不易解釋。

2. Optimized LSTM

(1) 主要工具：

- 使用正則化（如 L2 norm）、Dropout、BatchNormalization 等技術來提升 LSTM 的泛化能力。
- 使用 Adam 或其他自適應優化器來加速收斂。
- 通過超參數調整（如網絡層數、單元數量、學習率等）來提升模型性能。

(2) 優點：

- 性能提升：經過優化的 LSTM 通常能在準確性、穩定性和收斂速度上表現更好。
- 降低過擬合：通過正則化和 Dropout 減少模型對訓練數據的過度記憶。
- 靈活優化：可以根據具體數據集的特性調整模型結構和參數。

(3) 缺點：

- 調參繁瑣：優化過程需要多次實驗和調整，耗時且需要專業經驗。
- 高計算需求：優化過程通常需要更大的計算資源。

3. 集成學習（LSTM、Random forest、XGBoost、Gradient boosting 加權投票方式）

(1) 主要工具：

- 將 LSTM、Random Forest、XGBoost、Gradient Boosting 模型進行結合，通過加權投票的方式進行集成。

- 使用加權策略根據每個基模型的性能（如準確率）分配不同權重。

(2) 優點：
- 穩定性高：多模型集成能夠平滑單一模型的隨機波動，提供穩定的預測。
- 提高性能：通過結合不同模型的特點，可以提升整體的準確率和泛化能力。
- 簡單實現：加權投票方式的實現相對直接。

(3) 缺點：
- 權重選擇依賴經驗：如何分配權重是關鍵，通常需要根據基模型的性能進行調整。
- 無法捕捉交互效果：基於投票的方法通常只考慮模型的獨立性能，無法利用模型間的相互依賴性。

4. 集成學習（LSTM、Random forest、XGBoost、Gradient boosting 堆疊方式集成）

(1) 主要工具：
- 將 LSTM、Random Forest、XGBoost、Gradient Boosting 的輸出作為新特徵，輸入到另一個元學習器（如 Logistic Regression 或 Random Forest）中。
- 對基模型進行分層訓練（如交叉驗證）以減少過擬合。

(2) 優點：
- 捕捉交互效果：堆疊集成能夠充分利用基模型間的交互，提升預測能力。
- 靈活性強：可以結合不同類型的模型（深度學習模型與機器學習模型）。
- 泛化能力強：利用元學習器能進一步提升集成模型的泛化能力。

(3) 缺點：
- 高複雜性：堆疊集成比加權投票更複雜，涉及多層模型的訓練和驗證，計算開銷較大。
- 過擬合風險：如果堆疊策略設計不當，可能會導致過擬合。
- 調試困難：堆疊集成需要更多參數和結構調整，實驗成本較高。

五、 AutoEncoder

是一種用於對數據進行壓縮和重建的方法，目的是學習數據的有效表示或特徵，同時保留關鍵信息。

1. 主要模組：使用 tensorflow 和 keras 中的 Input 和 Dense 層來構建 AutoEncoder 模型。

2. 優點：
   - 數據降維：自編碼器能夠有效地進行數據降維，將高維數據轉換為低維表示，並且能夠保留重要特徵。
   - 去噪能力：自編碼器可以用於去除數據中的噪聲，通過學習數據的主要結構來重構無噪聲的數據。
   - 異常檢測：由於自編碼器學習數據的分佈特徵，因此它們可以用來識別異常數據點，這對於監控和預警系統非常有用。

3. 缺點：
   - 訓練時間長：自編碼器需要大量的計算資源來訓練，特別是在處理大規模數據集時。
   - 潛在空間的選擇：自編碼器的性能在很大程度上取決於潛在空間的維度選擇，如果潛在空間過大或過小，可能會影響模型效果。
   - 缺乏標註信息：作為無監督學習方法，自編碼器無法直接依賴標註數據，因此在某些情況下可能會缺乏對問題本質的深刻理解。

肆、實作與評估方法

本專題中，主要使用這五個模型進行實作與評估：Random Forest、Linear Regression、XGBoost (eXtreme Gradient Boosting)、LSTM (Long Short-Term Memory)、AutoEncoder。在該部分內容中，分析各個模型之實作與評估方法。

【Random Forest】

一、實作流程

1. 資料預處理

基於 Kaggle 預處理方法對數據進行清理，步驟包括：

(1) 缺失值填補
使用數值型特徵的中位數填補缺失值，避免數據缺失影響模型性能。

(2) 標準化
使用 StandardScaler 將特徵標準化，確保不同量級的特徵對模型訓練的影響均衡。

2. 數據平衡

由於目標變數中「Fail」類別的樣本量極少，導致模型可能傾向於預測多數類別。因此，採用了 SMOTE（Synthetic Minority Oversampling Technique） 對訓練數據進行過採樣，生成更多的少數類別樣本，從而平衡類別分布。

SMOTE 處理後，「Pass」與「Fail」類別的樣本數比例得以平衡，提升模型對少數類別的學習能力。

3. 模型建構

選用隨機森林作為分類模型，並通過以下方式進行構建與優化：

(1) 超參數調整：
- 使用 GridSearchCV 對模型超參數進行搜索，調整範圍包括：
- 樹的數量（n_estimators）：100 至 500。
- 樹的深度（max_depth）：10 至 30。
- 節點最小拆分樣本數（min_samples_split）：2 至 10。
- 每棵樹使用的特徵數（max_features）：'sqrt' 或 'log2'。

最終獲得的最佳參數為：

最佳參數: {'class_weight': 'balanced', 'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}

```
最佳參數: {'class_weight': 'balanced', 'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
```

(2) 預測閾值調整：

將預測閾值從默認的 0.5 調整為 0.3，旨在提升少數類別的召回率（Recall）。

二、模型評估

1. 性能指標

模型性能通過以下指標進行評估：

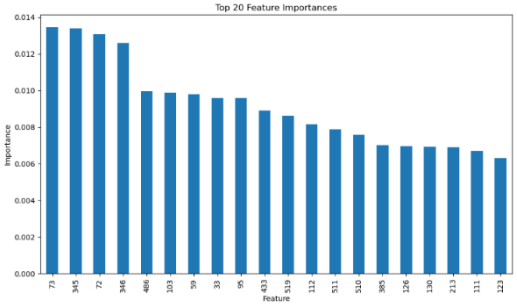
- Accuracy（準確率）：表示模型整體預測的正確率。
- Precision（精確率）：衡量模型對少數類別預測的準確性，反映假陽性的比例。
- Recall（召回率）：衡量模型對少數類別的檢測能力，反映假陰性的比例。
- F1 Score：精確率與召回率的調和平均，用於綜合評估模型性能。

模型的評估結果如下：

```
================ 評估結果一覽 ================
Accuracy: 0.91
Precision: 0.28
Recall: 0.24
F1 Score: 0.26
```

2. 特徵重要性分析

模型內建的特徵重要性功能顯示，特徵 73、345 和 72 對分類結果的貢獻最大，重要性分別為 0.014、0.013 和 0.013。這些特徵可能與製程異常有較強的相關性，但整體特徵重要性較為分散，表明數據中的特徵對目標變數的區分度有限。下圖展示了前 20 個重要特徵的貢獻：



3. Precision 與 F1 Score 表現不如預期的原因

(1) 少數類別樣本量不足：即使經過 SMOTE 增加樣本量，生成的少數類別樣本可能無法完全模擬實際分布，導致模型對少數類別的學習效果有限。

(2) 預測閾值的影響：降低預測閾值至 0.3 儘管提升了召回率，但也增加了假陽性數量，壓低了精確率，進而影響 F1 Score。

(3) 特徵相關性不足：數據中大部分特徵對目標變數的區分能力較弱，進一步影響了模型對少數類別的識別能力。

三、結論

透過隨機森林模型結合 SMOTE 技術對半導體製程數據進行分析，在少數類別的召回率（Recall）上取得了一定提升。然而，由於數據特徵相關性較弱且少數類別樣本稀缺，導致精確率（Precision）與 F1 Score 表現仍不理想。

未來研究方向包括：

1. 增強特徵工程：探索交互特徵或衍生特徵，以提升特徵對目標變數的區分度。
2. 嘗試其他數據平衡方法：例如結合下採樣的 SMOTEENN 或基於密度調整的 ADASYN 方法。
3. 引入其他模型：使用更先進的集成模型（如 XGBoost 或 LightGBM）進一步提升分類性能。

【Logistic Regression】

一、實作說明

1. 載入與檢查資料

(1) 載入數據集：data = pd.read_csv('/Users/gongyuting/Downloads/uci-secom.csv')

(2) 初步檢查數據：

- 查看數據前 5 行（data.head()）和數據信息（data.info()）。
- 計算每列缺失值數量，篩選出有缺失值的列。

2. 處理缺失值

(1) 只針對數值型欄位，使用中位數填補缺失值：data.fillna(data.select_dtypes(include=[np.number]).median(), inplace=True)。

(2) 確認缺失值已經被填補。

3. 探索性資料分析 (EDA)

(1) 目標變數分佈：繪製 Pass/Fail 的分佈圖（條形圖）。

(2) 特徵分佈：繪製數值型特徵的分佈圖（直方圖）和 Pass/Fail 分類下的箱型圖。

(3) 相關性分析：計算數值型特徵間的相關性矩陣，並繪製相關性熱力圖。

4. 資料清理

(1) 低變異性特徵：計算每個數值型特徵的標準差，刪除標準差低於閾值（< 1e-3）的特徵。

(2) 低相關性特徵：計算與 Pass/Fail 相關性小於 0.1 的特徵，並刪除。

(3) 高度相關性特徵：找出相關性大於 0.9 的特徵對，刪除其中一個。

5. 數據標準化

(1) 使用 StandardScaler 對數據進行標準化處理，讓所有數值型特徵的平均值為 0，標準差為 1。

6. 資料集分割

(1) 將數據分成訓練集（70%）和測試集（30%），並使用 stratify=y 確保分層抽樣。

(2) 確認訓練集與測試集的類別分佈。

7. 處理類別不平衡

(1) 使用 SMOTE（Synthetic Minority Oversampling Technique）對訓練數據進行過採樣，平衡目標變數的類別。

8. 模型訓練與評估

(1) 訓練：使用過採樣後的訓練集進行模型訓練。

(2) 評估：在測試集上評估模型性能，計算準確率、分類報告、混淆矩陣。

實作重點：

1. 處理缺失值與數據清理是模型成功的基礎。
2. 對於不平衡數據集，過採樣（如 SMOTE）有助於提高模型的泛化性能。
3. 模型訓練後需使用測試集進行性能評估，確保模型能在未見數據上表現良好。
4. 相關性分析與特徵選擇減少了冗餘數據，提高了模型效率。

二、評估方法

1. 準確率（Accuracy）

定義為模型預測正確的樣本數量占總樣本數的比例：

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{TP + FP + FN + TN}$$

- TP: True Positive（正確預測為正例）
- TN: True Negative（正確預測為反例）
- FP: False Positive（誤判為正例）
- FN: False Negative（誤判為反例）

在程式中：print("模型準確率:", accuracy_score(y_test, y_pred))

2. 混淆矩陣（Confusion Matrix）

顯示模型的預測結果和實際標籤之間的比較，四個部分分別是：

(1) TP（預測正例且正確）
(2) TN（預測反例且正確）
(3) FP（預測正例但錯誤）
(4) FN（預測反例但錯誤）

在程式中：

```
print("混淆矩陣:")
print(confusion_matrix(y_test, y_pred))
```

3. 分類報告（Classification Report）

包括以下指標：

(1) 精確率（Precision）
代表在所有被預測為正例的樣本中，實際為正例的比例。

$$\text{Precision} = \frac{TP}{TP + FP}$$

(2) 召回率（Recall 或 Sensitivity）
代表在所有實際為正例的樣本中，模型正確預測出的比例。

$$\text{Recall} = \frac{TP}{TP + FN}$$

(3) F1 分數（F1-Score）
平衡了精確率和召回率，用於評估不平衡數據下的模型性能。

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4. 類別不平衡處理

(1) 程式中特別使用了 SMOTE 來平衡訓練數據集，這對於避免模型偏向多數類別非常重要。

(2) 評估中使用分層抽樣（stratify=y）來確保測試集類別分佈與原始數據集一致，以進行公平評估。

5. 比較不同模型的性能

(1) 程式訓練了 Logistic Regression，並對兩者進行以下性能指標的比較：準確率、混淆矩陣和分類報告。

(2) 可以觀察兩種模型在測試集上的表現，選擇最適合的模型。

6. 適合不平衡數據的評估指標

由於目標變數可能存在不平衡（如 Pass/Fail 類別比例懸殊），單純依靠準確率可能會掩蓋模型的真實表現。因此，程式中特別重視 **精確率 (Precision) 和 召回率 (Recall)**，以衡量模型是否在少數類別上有良好的預測效果。

以下為實作結果：

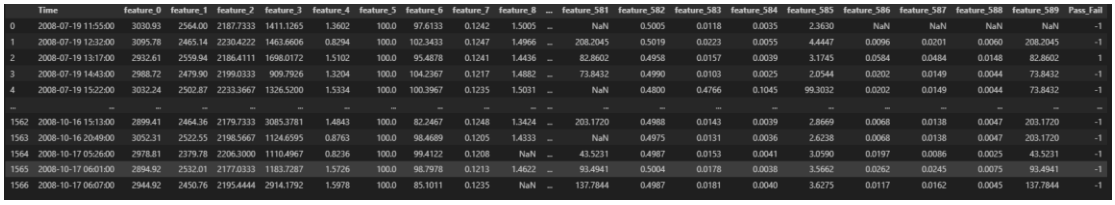
```
模型準確率: 0.7430997876857749

分類報告:
              precision    recall  f1-score   support

          -1       0.97      0.75      0.85       440
           1       0.15      0.65      0.25        31

    accuracy                           0.74       471
   macro avg       0.56      0.70      0.55       471
weighted avg       0.91      0.74      0.81       471

混淆矩陣:
[[330 110]
 [ 11  20]]
```

【XGBoost】

一、 資料前處理

1. 欄位名稱命名

將資料集中所有的欄位名稱新增統一的前綴字 feature_，方便區分這些欄位是否是特徵變數。並且將重命名回來的 feature_Time 和 feature_Pass/Fail 恢復原來的名稱，因為這些欄位不是特徵變數，而是時間和目標變數。



	Time	feature_0	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	feature_8	...	feature_581	feature_582	feature_583	feature_584	feature_585	feature_586	feature_587	feature_588	feature_589	Pass_Fail
0	2008-07-19 11:55:00	3030.93	2564.00	2187.7333	1411.1265	1.3602	100.0	97.6133	0.1242	1.5005	...	NaN	0.5005	0.0118	0.0035	2.3630	NaN	NaN	NaN	NaN	-1
1	2008-07-19 12:32:00	3095.78	2465.14	2230.4222	1463.6606	0.8294	100.0	102.3433	0.1247	1.4966	...	208.2045	0.5019	0.0223	0.0055	4.4447	0.0096	0.0201	0.0060	208.2045	-1
2	2008-07-19 13:17:00	2932.61	2559.94	2186.4111	1698.0172	1.5102	100.0	95.4878	0.1241	1.4436	...	82.8602	0.4958	0.0157	0.0039	3.1745	0.0584	0.0484	0.0148	82.8602	1
3	2008-07-19 14:43:00	2988.72	2479.90	2199.0333	909.7926	1.3204	100.0	104.2367	0.1217	1.4882	...	73.8432	0.4990	0.0103	0.0025	2.0544	0.0202	0.0149	0.0044	73.8432	-1
4	2008-07-19 15:22:00	3032.24	2502.87	2233.3667	1326.5200	1.5334	100.0	100.3967	0.1235	1.5031	...	NaN	0.4800	0.4766	0.1045	99.3032	0.0202	0.0149	0.0044	73.8432	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1562	2008-10-16 15:13:00	2899.41	2464.36	2179.7333	3085.3781	1.4843	100.0	82.2467	0.1248	1.3424	...	203.1720	0.4988	0.0143	0.0039	2.8669	0.0068	0.0138	0.0047	203.1720	-1
1563	2008-10-16 20:49:00	3052.31	2522.55	2198.5667	1124.6595	0.8763	100.0	98.4689	0.1205	1.4333	...	NaN	0.4975	0.0131	0.0036	2.6238	0.0068	0.0138	0.0047	203.1720	-1
1564	2008-10-17 05:26:00	2978.81	2379.78	2206.3000	1110.4967	0.8236	100.0	99.4122	0.1208	NaN	...	43.5231	0.4987	0.0153	0.0041	3.0590	0.0197	0.0086	0.0025	43.5231	-1
1565	2008-10-17 06:01:00	2894.92	2532.01	2177.0333	1183.7287	1.5726	100.0	98.7978	0.1213	1.4622	...	93.4941	0.5004	0.0178	0.0038	3.5662	0.0262	0.0245	0.0075	93.4941	-1
1566	2008-10-17 06:07:00	2944.92	2450.76	2195.4444	2914.1792	1.5978	100.0	85.1011	0.1235	NaN	...	137.7844	0.4987	0.0181	0.0040	3.6275	0.0117	0.0162	0.0045	137.7844	-1

2. 填補缺失值

由於中位數在處理極端值（outliers）時比均值（mean）更穩健，尤其在測試結果類型的數據中，中位數是比均值更合理的選擇，因此選擇使用中位數填補缺失值。填補完後確認是否還存在缺失值。

```
# Replacing all the NaN values with mean as the values correspond to the test results.
numeric_columns = data.select_dtypes(include=[np.number]).columns
data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].median())

# again, checking if there is any NULL values left
data.isnull().any().any()
```

✓ 0.1s

```
False
```

3. 時間特徵工程

將資料集中 Time 欄位轉換為多個與日期時間相關的特徵。增加與日期時間相關的特徵（如年份、月份、星期等），為後續的探索性數據分析和建模提供更多上下文信息。以下輸出為檢查新增的特徵分佈情況，觀察是否符合預期：

```
# This consists of only year 2008
print("Year: ", data.year.unique())
# This consists of all the months of 2008
print("Month: ", data.month.unique())
# All the dates of the month are not there, might be related to production on certain days only
print("Date: ", data.date.unique())
# All the weekdays of the month are here, so production happens on all 7 days
# 0 stand for Sunday, 1 for Monday ... 6 for Saturday
print("Weekday: ", data.week_day.unique())
```

✓ 0.0s

```
Year:  [2008]
Month:  [ 7  1  2  3  4  5  6  8  9 10 11 12]
Date:  [19 20 21 22 23 25 27 28 29 30 31  8 13 15 16 17 18 24  9 14 26 10]
Weekday:  [5 6 0 1 2 4 3]
```

二、探索性資料分析

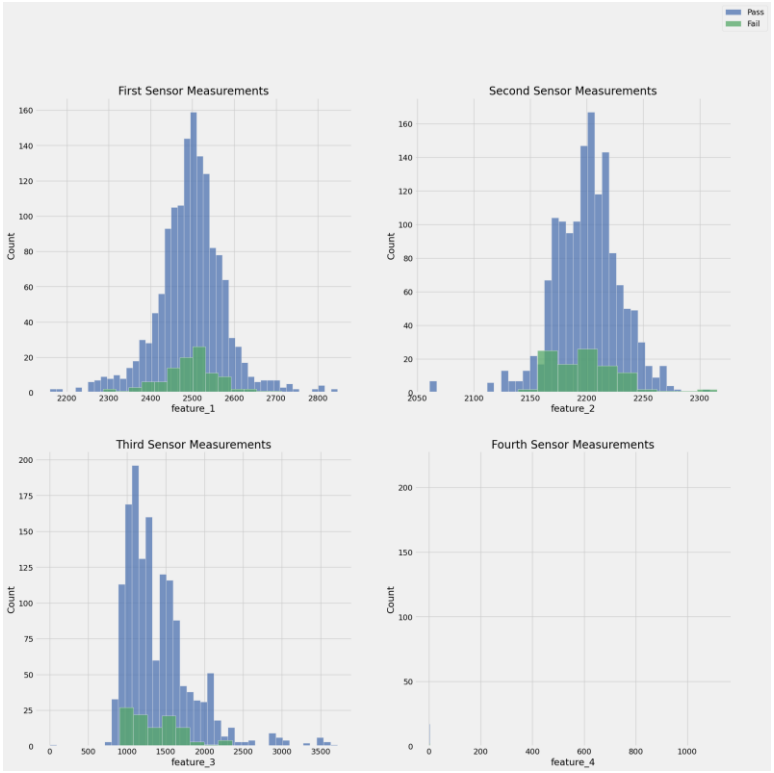
在該部分執行了一系列的探索性數據分析（Exploratory Data Analysis, EDA），通過視覺化和統計方法分析資料的結構、分布和相關性。以下是逐步解釋每個部分的功能和作用：

1. 分類數據分析—直方圖

這部分的目的是分析四個感測器特徵的數據分布，並分別根據 Pass_Fail 的類別進行視覺化。

以下為結果圖片分析：

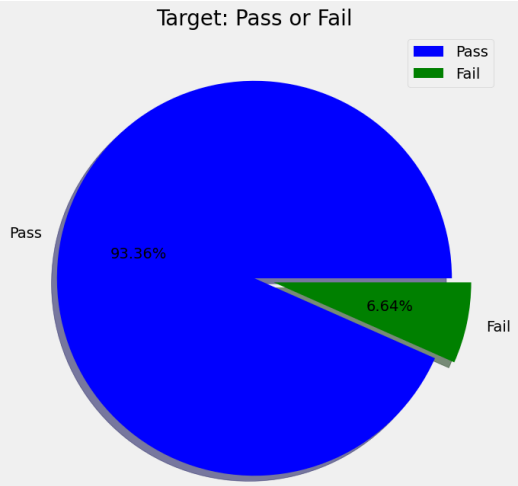
- feature_1 和 feature_2：數據分布非常集中，主要聚集在某些範圍內（如靠近 2000-2500 的區間）。
- feature_3：數據分布相對分散，顯示出更多的變化範圍，並且不同類別的重疊情況更為顯著。
- feature_4：數據範圍較小，且絕大部分集中在接近 0 的區間，可能是該感測器大多未檢測到信號或數據值極小。



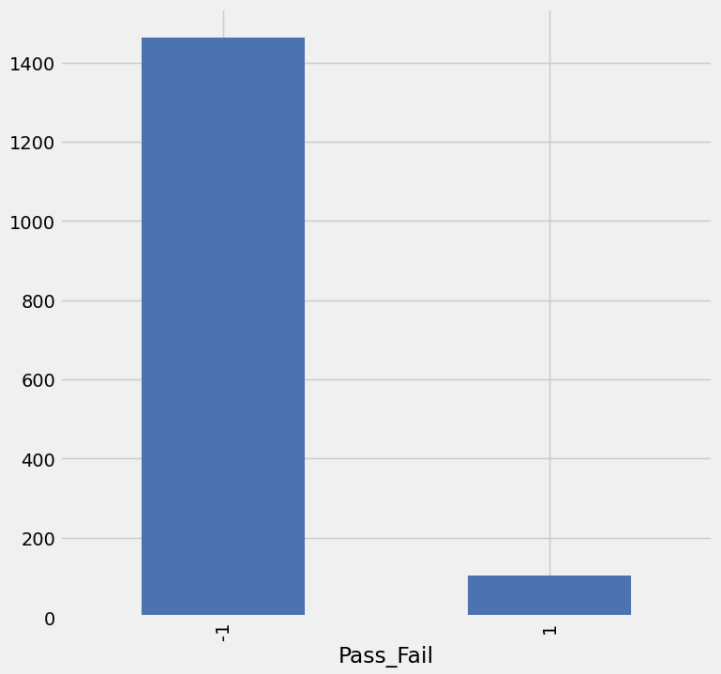
2. 類別比例分析—圓餅圖

這部分的主要目的是展示目標變數 Pass_Fail 的分佈，分為兩部分：

- 圓餅圖（Pie Chart）：
  顯示了目標變數的比例分佈："Pass" 占比 93.36%（藍色）；"Fail" 占比 6.64%（綠色）。可以觀察到這是一個類別嚴重不平衡的數據集。



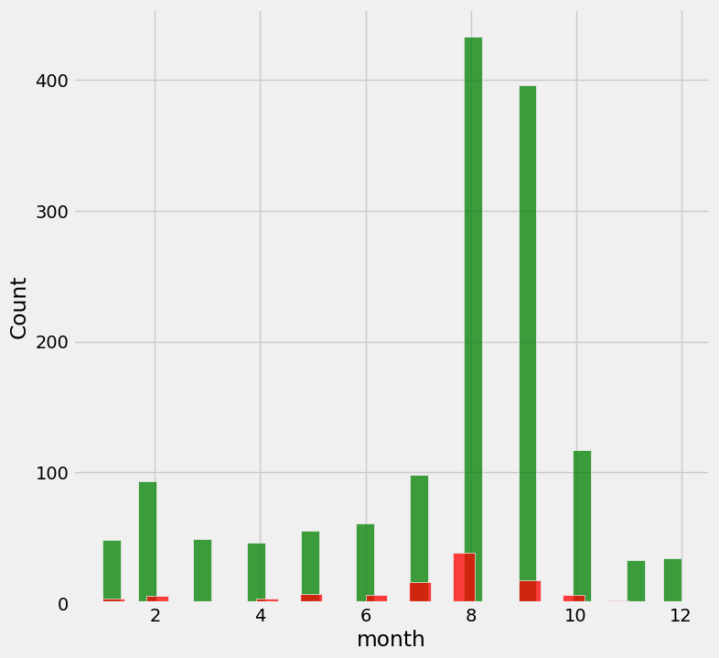
- 長條圖（Bar Chart）：
  橫軸為目標變數的值，分別為 -1（Pass）和 1（Fail）。縱軸為對應的數量："Pass" 的樣本數約 1460；"Fail" 的樣本數約 100。從圖表中視覺化證實了目標變數的極端不平衡性。



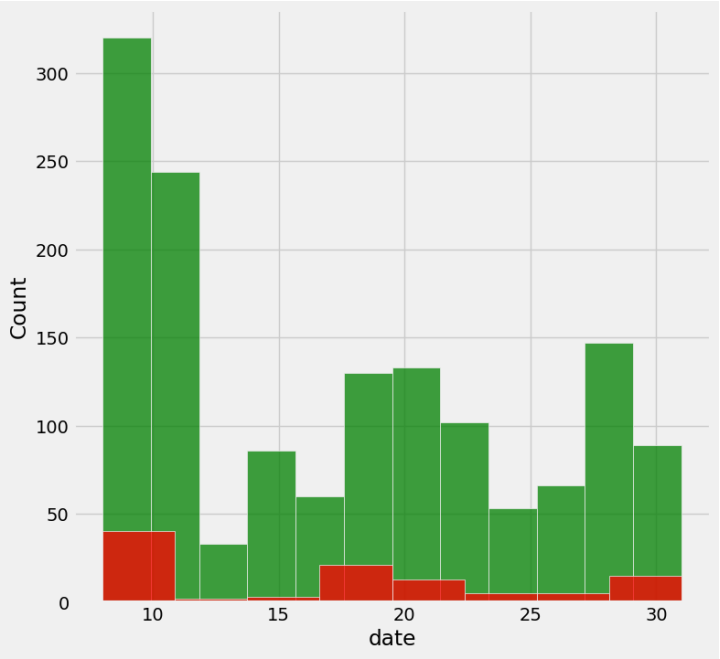
3. 按類別分析時間分布

該部分的目的是對生產過程中的「通過」與「失敗」樣本分佈進行探索性資料分析，並以視覺化的方式呈現每月、每天、每星期的通過與失敗樣本的分佈。

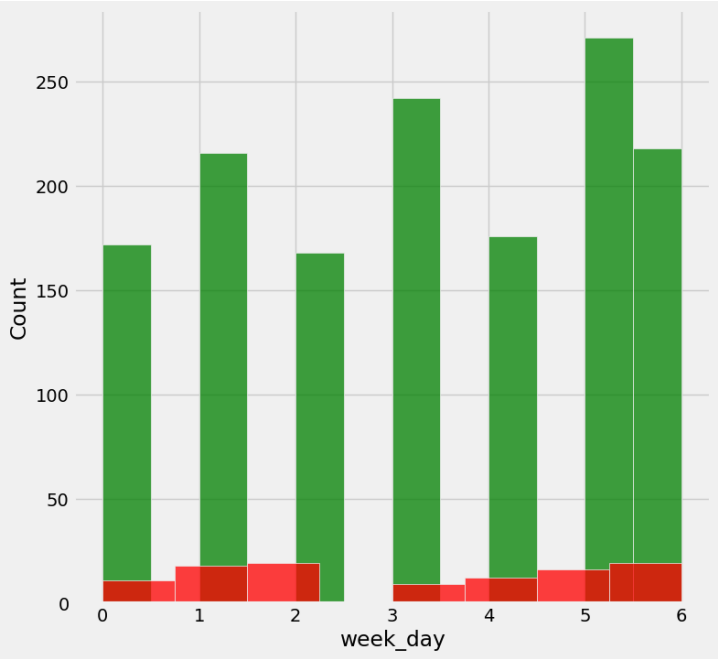
- 每月的通過與失敗樣本分佈：生產高峰集中在 8 月和 9 月，隨後失敗率有所降低，可能反映改進的效果。



- 每天的通過與失敗樣本分佈：月初與月底的失敗率較高，生產的調控可能在月中更穩定。

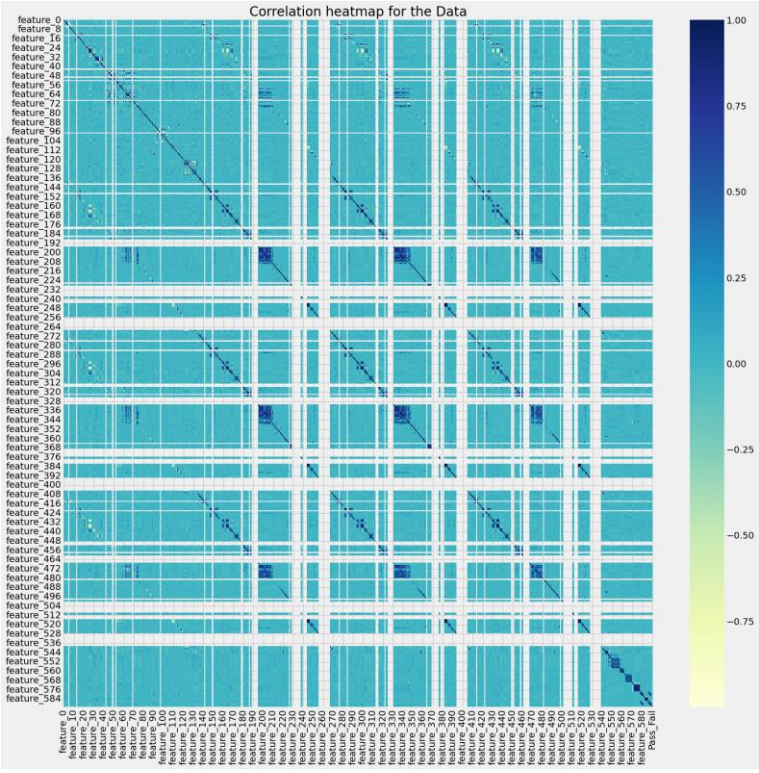


- 每星期的通過與失敗樣本分佈：週末的生產量與失敗次數都較高，週三、四更穩定。



4. 特徵相關性熱力圖

以下為相關性矩陣的熱力圖，其中每個方格表示兩個特徵之間的相關性，顏色的深淺顯示相關性的強弱：顏色越接近黃色，表示兩個特徵之間的正相關性越強；越接近藍色，表示負相關性越強；越接近綠色，表示相關性較弱。



三、 資料清理

1. 刪除 Time 欄位

這部分的目的是處理數據集中的 Time 欄位，進行時間格式的轉換並從中提取出年、月和日，然後將 Time 欄位刪除。

2. 去除高相關度的特徵

計算特徵之間的相關性，並刪除那些相關性超過設定閾值（在這裡是 0.70）的特徵，目的是減少冗餘特徵，改善機器學習模型的效果，並提高模型的效率。

3. 分割數據集

將 30% 的數據將用作測試集，剩餘的 70% 用作訓練集。

4. 標準化

訓練集數據進行擬合並轉換，這樣訓練集的每個特徵都會被轉換為均值為 0，標準差為 1 的標準化數據。此外，測試集數據只進行轉換，並且是根據訓練集的標準化參數來進行轉換，確保測試集數據與訓練集保持一致的標準化處理。

5. 處理類別不平衡

使用 SMOTE（Synthetic Minority Over-sampling Technique）處理類別不平衡問題的技術，透過生成少數類樣本的合成數據來增加少數類別的數量，從而使得訓練集中的各類別更加平衡。

四、 建模

1. 超參數調優（Hyperparameter Tuning）

(1) params_knn：設定了 KNN 模型的超參數範圍，包括鄰居數量、權重方式、算法和距離度量的參數。

(2) GridSearchCV：用來進行網格搜索，尋找最佳的超參數組合。

- cv=5 表示進行 5 次交叉驗證。
- scoring='accuracy' 設定評估指標為準確率。
- n_jobs=-1 使其使用所有可用的 CPU 核心來加速計算。

調整過程後，使用 search_knn.best_params_ 等來輸出每個模型的最佳超參數組合。如果不進行超參數調優，則直接使用預設的超參數來訓練模型。

2. 創建集成學習模型

創建不同組合的集成模型。這裡使用了 VotingClassifier，它是一種集成學習方法，通過多個基模型的投票結果來進行預測。可以選擇硬投票（voting='hard'）或軟投票（voting='soft'），其中硬投票根據大多數模型的預測結果決定最終預測，軟投票則根據每個模型的預測概率來決定，在此部分創建了 2 至 5 個基模型組合的硬投票和軟投票模型。

3. 交叉驗證

(1) cross_val_score: 用來進行交叉驗證，對每個模型計算其在訓練集上的準確率。cv=5 表示 5 折交叉驗證，結果會顯示每個模型的平均得分。

(2) sort_values('score', ascending=False): 根據模型的交叉驗證分數對模型進行排序，並返回排序後的結果。

4. 模型訓練與預測

使用訓練集訓練 XGBoost 模型，以及使用訓練集訓練由 KNN、隨機森林和 XGBoost 組合而成的集成模型，並進行預測。

五、 評估模型

在評估階段，對兩個模型（XGBoost 和 KNN、隨機森林、XGBoost 組合）進行了性能評估，並計算了多個常見的性能指標，包括準確率（Accuracy）、精確率（Precision）、召回率（Recall）、F1 分數（F1 Score）和混淆矩陣（Confusion Matrix）。以下為評估結果：

```
XGBoost:
Accuracy: 0.9151
Precision: 0.3333
Recall: 0.2500
F1 Score: 0.2857
Confusion Matrix:
[[423  16]
 [ 24   8]]
```

```
KNN, Random Forest, XGBoost:
Accuracy: 0.8854
Precision: 0.2250
Recall: 0.2812
F1 Score: 0.2500
Confusion Matrix:
[[408  31]
 [ 23   9]]
```

結論：

(1) XGBoost 模型：表現出較高的準確率和精確率，表明它在大多數情況下做出正確的預測，但其對正類的識別能力較弱，召回率較低。

(2) KNN + 隨機森林 + XGBoost 組合模型：在召回率上稍微優於 XGBoost，但整體表現不如 XGBoost，尤其在精確率和 F1 分數上較低。

【LSTM】

一、 資料前處理

1. 欄位名稱命名

為了區分特徵變數與其他欄位，將資料集中所有欄位新增統一的前綴 feature_，明確標註特徵欄位。此外，針對時間欄位 Time 和目標變數 Pass/Fail，保留原名稱以避免混淆，因為這兩者並非特徵變數，而是時間信息與預測目標。

2. 填補缺失值

為應對缺失值問題，針對數值型欄位採用中位數進行填補。相比均值，中位數對極端值的影響較小，因此在測試結果類型的數據中更為穩健。填補完成後，需確認數據中不再存在缺失值。

3. 時間特徵工程

將時間欄位轉換為多個與日期時間相關的特徵（如年份、月份、星期等），進一步擴展數據的上下文信息，有助於揭示時間對目標變數的潛在影響。

二、 資料清理

1. 刪除時間欄位

時間欄位中的有用信息已通過特徵工程提取，原始欄位可刪除以簡化數據結構。

2. 去除高相關度特徵

針對相關性超過設定閾值的特徵，刪除其中冗餘的特徵，以減少多重共線性對模型訓練的干擾。

3. 數據集分割

將數據按 7:3 分為訓練集與測試集，並採用分層抽樣以確保各類別在訓練集與測試集中保持比例一致。

4. 標準化數據

為避免特徵尺度差異對模型的影響，對所有數值型特徵進行標準化處理，確保均值為 0、標準差為 1。

5. 處理類別不平衡

針對目標變數的類別不平衡問題，採用 SMOTE 方法合成少數類別樣本，平衡訓練數據，提升模型對少數類別的辨識能力。

三、評估方法

1. **準確率（Accuracy）：**模型預測正確的樣本數量佔總樣本數的比例。

2. **混淆矩陣（Confusion Matrix）**

顯示模型預測結果與實際標籤的比較。

- **TP**（真陽性）：預測為正且正確。
- **TN**（真陰性）：預測為負且正確。
- **FP**（假陽性）：預測為正但錯誤。
- **FN**（假陰性）：預測為負但錯誤。

3. 分類報告（Classification Report）

包含精確率（Precision）、召回率（Recall）和 F1 分數（F1-Score）。

【AutoEncoder】

一、實作流程

1. 模型建構與優化

(1) 模型結構：

- 模型由編碼器與解碼器兩部分組成：編碼器（將輸入特徵壓縮至低維空間、解碼器（將壓縮後的數據重建回原始維度）。
- 使用非線性激活函數（ReLU）以增強特徵提取能力，並加入批正規化（Batch Normalization）穩定訓練過程。

(2) 超參數優化：

使用 Optuna 對以下參數進行調整：

- 編碼維度（encoding_dim）：32 至 128。
- 學習率（learning_rate）：0.00001 至 0.01。
- Dropout 率（dropout_rate）：0.1 至 0.5。

經過 30 次試驗，最終獲得的最佳參數為：
{'encoding_dim': 55, 'learning_rate': 1.992e-05, 'dropout_rate': 0.144}

2. 模型訓練與預測

(1) 模型訓練：

- 使用均方誤差（MSE）作為損失函數，優化數據重建性能。
- 訓練過程包含 50 個 Epoch，批量大小為 32。

(2) 異常檢測：

- 基於數據重建損失設置閾值（第 95 個百分位數），超出閾值的樣本被視為異常。

二、模型評估

1. 評估指標

模型性能採用以下指標進行評估：

- MSE（均方誤差）：衡量重建樣本的平均誤差平方和。
- MAE（平均絕對誤差）：表示樣本誤差的平均幅度。
- R^2（判定係數）：評估模型對數據變異的解釋能力。

- Accuracy（準確率）：衡量模型整體分類的正確率。
- F1 Score：精確率與召回率的調和平均，適用於不平衡數據。

最佳結果如下：

- Accuracy：0.533
- F1 Score：0.412
- MSE：1.848
- MAE：0.924
- R^2：-0.853

2. 分類報告

```
================= 評估結果一覽 =====
Accuracy: 0.85
Precision: 0.00
Recall: 0.00
F1 Score: 0.00
Confusion Matrix:
[[267  47]
 [  0   0]]
====================================
```

3. 模型表現分析

(1) 概述

從輸出的結果可以看出模型的整體準確率（Accuracy）達到了 0.85，但其他指標（Precision、Recall 和 F1 Score）均為 0.00。同時，混淆矩陣顯示模型完全無法正確預測少數類別（即「1」），具體表現在以下兩點：

- 混淆矩陣：
  - ✓ 第一行表示實際標籤為「0」（正常）的樣本中，267 個被正確分類，47 個被錯誤分類為「1」（異常）。
  - ✓ 第二行表示實際標籤為「1」（異常）的樣本中，沒有一個樣本被正確分類。
- 指標表現：
  - ✓ Precision（精確率）：模型在判斷為異常（1）時的準確性為 0，說明模型完全無法識別少數類別。
  - ✓ Recall（召回率）：對於異常樣本（1），模型的檢測率為 0，表明所有異常樣本均被錯誤分類為正常樣本。

- ✓ F1 Score：精確率和召回率的調和平均為 0，進一步強調模型在處理異常類別時的表現不佳。

(2) 問題分析

- 數據不平衡問題：

  從混淆矩陣可以看出，數據集中正常樣本遠多於異常樣本。這使得 Autoencoder 將異常樣本重建為正常樣本的能力不足，並最終導致模型在異常樣本上的分類性能較差。

- 重建誤差閾值設置問題：

  重建誤差被用作異常檢測的判斷標準。若閾值設置不合理（如過高），異常樣本的重建誤差可能被錯誤地視為正常，導致異常樣本無法被檢測。

- Autoencoder 的設計限制：

  Autoencoder 假設大多數數據是正常的，異常樣本的特徵可能被模型忽略。尤其是當異常樣本數量較少時，模型可能無法有效學習這些樣本的特徵。

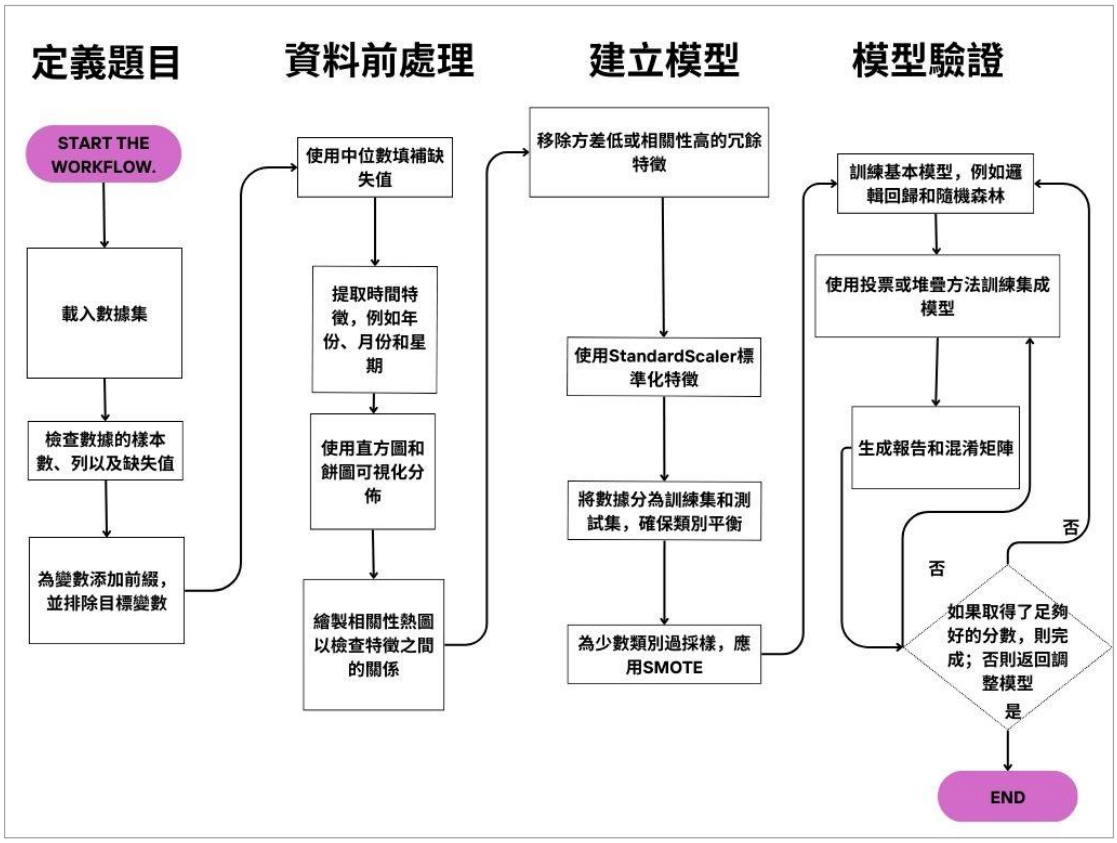
- 模型評估方法：

  當數據不平衡時，準確率並不能準確反映模型的實際性能。應更多依賴 Precision、Recall 和 F1 Score 進行綜合評估。

三、 結論

本次實作展示了自動編碼器在異常檢測中的應用，通過 PCA 和 SMOTE 的結合，模型達到了 0.533 的準確率和 0.412 的 F1 Score。然而，對異常樣本的召回能力仍需提升。

伍、流程


定義題目
資料前處理
建立模型
模型驗證
START THE WORKFLOW.
載入數據集
檢查數據的樣本數、列以及缺失值
為變數添加前綴，並排除目標變數
使用中位數填補缺失值
提取時間特徵，例如年份、月份和星期
使用直方圖和餅圖可視化分佈
繪製相關性熱圖以檢查特徵之間的關係
移除方差低或相關性高的冗餘特徵
使用StandardScaler標準化特徵
將數據分為訓練集和測試集，確保類別平衡
為少數類別過採樣，應用SMOTE
訓練基本模型，例如邏輯回歸和隨機森林
使用投票或堆疊方法訓練集成模型
生成報告和混淆矩陣
如果取得了足夠好的分數，則完成；否則返回調整模型
否
否
是
END

陸、分析結果與結論

一、Random Forest

選用隨機森林作為分類模型，並通過以下方式進行構建與優化：

1. 超參數調整

使用 GridSearchCV 對模型超參數進行搜索，調整範圍包括：

- 樹的數量（n_estimators）：100 至 500。
- 樹的深度（max_depth）：10 至 30。
- 節點最小拆分樣本數（min_samples_split）：2 至 10。
- 每棵樹使用的特徵數（max_features）：'sqrt' 或 'log2'。

2. 預測閾值調整

將預測閾值從默認的 0.5 調整為 0.3，旨在提升少數類別的召回率（Recall）。以下為表現評估結果：

```
KNN, Random Forest, XGBoost:
Accuracy: 0.8854
Precision: 0.2250
Recall: 0.2812
F1 Score: 0.2500
Confusion Matrix:
[[408  31]
 [ 23   9]]
```

- 準確率（Accuracy）：0.89
  此部分需注意在類別不平衡的情況下，準確率可能被多數類別（負例）主導，無法全面反映模型對少數類別（正例）的表現。

- 精確率（Precision）：0.23
  當模型預測為正例時，只有 23% 是正確的，這表示模型對少數類別的預測仍有明顯不足。

- 召回率（Recall）：0.28
  在實際為正例的樣本中，模型僅正確預測了 28%。這顯示模型在少數類別的覆蓋能力有限。

- F1 分數（F1 Score）：0.25
  僅為 25%，說明模型在少數類別上的整體表現不理想。

二、Logistic Regression

- 採用 **Logistic Regression** 進行模型訓練。
- 訓練過程中採用分層抽樣（stratify）確保訓練與測試集類別分佈一致。
- 使用測試集進行評估，防止過度擬合，確保模型能泛化到未見數據。

以下為表現評估結果：

```
模型準確率: 0.7430997876857749

分類報告:
              precision    recall  f1-score   support

          -1       0.97      0.75      0.85       440
           1       0.15      0.65      0.25        31

    accuracy                           0.74       471
   macro avg       0.56      0.70      0.55       471
weighted avg       0.91      0.74      0.81       471

混淆矩陣:
[[330 110]
 [ 11  20]]
```

1. 準確率（Accuracy）

準確率為 **74.31%**，說明模型能在大多數情況下正確分類，但單憑準確率不足以評估類別不平衡情況下的表現。

2. 分類報告

- **-1 類別（多數類別）**：
  - ✓ 精確率（Precision）：97%，表示預測為 -1 的樣本中，有 97% 是正確的。
  - ✓ 召回率（Recall）：75%，說明實際為 -1 的樣本中，模型正確預測出 75%。
  - ✓ F1 分數（F1-score）：0.85，表明該類別的整體平衡表現不錯。
- **1 類別（少數類別）**：
  - ✓ 精確率（Precision）：15%，表示預測為 1 的樣本中，僅 15% 是正確的。
  - ✓ 召回率（Recall）：65%，說明實際為 1 的樣本中，模型正確預測出 65%。

- ✓ F1 分數（F1-score）：0.25，表示該類別的整體表現較弱，尤其是在精確率上存在嚴重不足。

3. 宏平均（Macro Avg）

精確率和 F1 分數偏低（分別為 0.56 和 0.55），反映模型在處理類別不平衡時的能力較弱。

4. 加權平均（Weighted Avg）

權重平均精確率和 F1 分數（0.91 和 0.81）較高，但主要由於多數類別（-1）的優異表現，無法反映少數類別的預測效果。

三、 XGBoost (eXtreme Gradient Boosting)

- **超參數調優**

使用網格搜索進行超參數調優，尋找模型在不同參數組合下的最佳性能。採用交叉驗證評估各參數組合的準確率，並選擇最優配置進行模型訓練。

- **集成學習模型**

構建由多個基模型組成的集成模型（如 VotingClassifier），通過硬投票或軟投票策略進行預測。硬投票根據基模型預測結果的多數決策，軟投票則根據預測概率加權。

以下為表現評估結果：

```
XGBoost:
Accuracy: 0.9151
Precision: 0.3333
Recall: 0.2500
F1 Score: 0.2857
Confusion Matrix:
[[423  16]
 [ 24   8]]
```

```
KNN, Random Forest, XGBoost:
Accuracy: 0.8854
Precision: 0.2250
Recall: 0.2812
F1 Score: 0.2500
Confusion Matrix:
[[408  31]
 [ 23   9]]
```

1. XGBoost

(1) 整體表現

- 準確率（Accuracy）：91.51%
  準確率非常高，但在類別不平衡情況下，準確率可能更多反映多數類別的正確性。

- 精確率（Precision）：33.33%
  預測為少數類別的樣本中，只有 33.33% 是正確的，表明對少數類別的識別能力仍有不足。

- 召回率（Recall）：25.00%
  實際屬於少數類別的樣本中，僅有 25% 被正確預測，顯示模型對少數類別的辨識能力偏弱。

- F1 分數（F1 Score）：28.57%
  精確率與召回率的綜合指標略低，進一步說明少數類別的分類表現不理想。

(2) 混淆矩陣

- 在多數類別（負例）中，正確分類共 423 筆，錯誤分類為少數類別共 16 筆。
- 在少數類別（正例）中，正確分類共 8 筆，錯誤分類為多數類別共 24 筆。

多數類別（負例）分類良好，但少數類別（正例）僅正確預測出 8 筆樣本，誤判為多數類別的樣本高達 24 筆。

2. KNN + Random Forest + XGBoost 集成模型

(1) 整體表現

- 準確率（Accuracy）：88.54%
  準確率略低於 XGBoost 單模型。

- 精確率（Precision）：22.50%
  預測為少數類別的樣本中，只有 22.50% 是正確的，比單一 XGBoost 更差。

- 召回率（Recall）：28.12%
  實際屬於少數類別的樣本中，28.12% 被正確預測，略優於 XGBoost 單模型。

- F1 分數（F1 Score）：25.00%
  綜合精確率與召回率後，表現與 XGBoost 單模型相當，但精確率略低，召回率略高。

(2) 混淆矩陣

- 在多數類別（負例）中，正確分類共 408 筆，錯誤分類為少數類別共 31 筆。
- 在少數類別（正例）中，正確分類共 9 筆，錯誤分類為多數類別共 23 筆。

對少數類別（正例）的召回率略高於 XGBoost 單模型，但代價是多數類別的錯誤分類數量增加。

四、 LSTM (Long Short-Term Memory)

1. 優化器與編譯

模型採用 Adam 優化器，設置較小的學習率（0.0001），以穩定模型的訓練過程，並使用二元交叉熵（binary_crossentropy）作為損失函數。

以下為表現評估結果：

```
Test Accuracy: 92.57%

Classification Report:
              precision    recall  f1-score   support

          -1       0.93      0.99      0.96       439
           1       0.20      0.03      0.05        32

    accuracy                           0.93       471
   macro avg       0.57      0.51      0.51       471
weighted avg       0.88      0.93      0.90       471


Confusion Matrix:
[[435   4]
 [ 31   1]]
```

(1) 整體測試準確率

整體準確率為 **92.57%**，表示大多數測試樣本被正確分類。但由於類別不平衡（多數類別 -1 樣本占絕大多數），準確率可能被多數類別主導，並不能全面反映模型性能。

(2) **分類報告分析**

- **多數類別（-1）**
  - ✓ **Precision（精確率）：93%**
    預測為 -1 的樣本中，93% 是正確的，表明模型對多數類別預測具有很高的可靠性。
  - ✓ **Recall（召回率）：99%**
    實際為 -1 的樣本中，有 99% 被正確識別，幾乎能完全覆蓋多數類別樣本。
  - ✓ **F1-Score：96%**
    表明模型在多數類別上的整體表現極為出色，精確率與召回率保持平衡。
- **少數類別（1）**
  - ✓ **Precision（精確率）：20%**
    預測為 1 的樣本中，只有 20% 是正確的，模型對少數類別的預測可信度極低。
  - ✓ **Recall（召回率）：3%**
    實際為 1 的樣本中，只有 3% 被正確識別，模型對少數類別幾乎無法有效檢測。
  - ✓ **F1-Score：5%**
    表明少數類別的整體預測能力非常有限，精確率和召回率均表現不佳。

(3) 宏平均（Macro Avg）

精確率、召回率與 F1 分數分別為 **0.57**、**0.51** 和 **0.51**。這是一個未加權的平均值，反映出類別間的性能差異，少數類別的表現極大拉低了平均分數。

(4) 加權平均（Weighted Avg）

精確率與召回率分別為 **0.88** 和 **0.93**。由於多數類別樣本占主導地位，加權平均分數主要反映多數類別的性能。

2. 混淆矩陣分析

- **多數類別（-1）**
  - ✓ 正確分類：435 筆
  - ✓ 錯誤分類為 1：4 筆

  表明模型對多數類別的分類效果非常好，幾乎無誤分類。

- **少數類別（1）**
  - ✓ 正確分類：1 筆
  - ✓ 錯誤分類為 -1：31 筆

  模型對少數類別的區分能力極弱，大部分少數類別樣本都被錯誤分類為多數類別。

Optimized LSTM

1. 模型總體結構

模型採用了 **Bidirectional LSTM（雙向長短期記憶網路）** 作為核心架構，結合多層 LSTM 和全連接層，輔以正則化和批次正規化來提高穩定性與泛化能力。以下是具體設計：

2. 優化器與損失函數

- **優化器**：Adam

  設定學習率為 0.0005，提供穩定的更新步伐，避免震盪。
- **損失函數**：Binary Crossentropy，適用於二元分類問題。

以下為表現評估結果：

```
Test Metrics:
loss: 0.5848
compile_metrics: 0.9087

Classification Report:
              precision    recall  f1-score   support

          -1       0.93      0.97      0.95       439
           1       0.13      0.06      0.09        32

    accuracy                           0.91       471
   macro avg       0.53      0.52      0.52       471
weighted avg       0.88      0.91      0.89       471


Confusion Matrix:
[[426  13]
 [ 30   2]]
```

(1) 整體準確率：91%

整體準確率較高，表明模型對大多數樣本（主要是多數類別 -1）能進行正確分類。但是在類別不平衡的情況下，準確率可能無法完全反映模型的真實性能。

(2) 分類報告

- **多數類別（-1）**
  - ✓ **Precision（精確率）：93%**
    表明模型對多數類別的預測較為可靠。
  - ✓ **Recall（召回率）：97%**
    顯示模型在多數類別的識別能力很強。
  - ✓ **F1-Score：0.95**
    表明該類別的整體預測性能非常好。

- **少數類別（1）**
  - ✓ **Precision（精確率）：13%**
    顯示模型在少數類別的預測中存在嚴重偏差。
  - ✓ **Recall（召回率）：6%**
    模型幾乎無法有效檢測少數類別樣本。
  - ✓ **F1-Score：0.09**
    綜合精確率與召回率，整體表現極差。

(3) Macro Average（宏平均）

- 平均精確率、召回率和 F1-Score 分別為 0.53、0.52 和 0.52。
- 說明兩個類別的預測性能差異極大，模型在少數類別上的表現明顯不足。

(4) Weighted Average（加權平均）

- 加權精確率和召回率分別為 0.88 和 0.91，主要由多數類別的高準確性主導，無法完全反映少數類別的性能。

3. 混淆矩陣

- **多數類別（-1）**
  - ✓ 正確分類：426 筆
  - ✓ 誤分類為 1：13 筆
  - ✓ 模型在多數類別上的誤分類數量較少。

- **少數類別（1）**
  - ✓ 正確分類：2 筆
  - ✓ 誤分類為 -1：30 筆
  - ✓ 少數類別的識別效果非常差，模型無法有效區分這些樣本。

(LSTM+Random forest+XGBoost+Gradient boosting)集成模型(voting)

以下為表現評估結果：

```
個別模型準確率：
LSTM Accuracy: 0.9002
Random Forest Accuracy: 0.9278
XGBoost Accuracy: 0.9257
Gradient Boosting Accuracy: 0.9193

集成模型準確率: 0.9193
```

1. 個別模型準確率

- **LSTM 準確率：90.02%**

LSTM 模型對時間序列數據的捕捉能力較強，但其準確率略低於其他基於樹的模型。可能是因為 LSTM 的結構更依賴於訓練參數的精調，且對非時間序列特徵的建模能力有限。

- **Random Forest 準確率：92.78%**

隨機森林表現出穩定性，準確率最高。隨機森林的優勢在於可以很好地處理非線性特徵和多樣化特徵的情況。

- **XGBoost 準確率：92.57%**

與隨機森林表現相當，稍低於隨機森林。XGBoost 可能更適合處理有少量異常值或較不平衡的數據，但在這裡未能完全超越隨機森林。

- **Gradient Boosting 準確率：91.93%**

與 XGBoost 相比稍遜，但仍然維持在較高準確率。梯度提升在處理細微特徵間的關係上較有優勢，但可能需要更大的迭代數量來達到最佳表現。

2. 集成模型準確率：91.93%

集成模型的準確率與 Gradient Boosting 一致，稍低於 Random Forest 和 XGBoost。原因包括：

- 部分基礎模型對最終結果的貢獻不足，例如 LSTM 的相對較低準確率可能影響了整體表現。
- 元模型（Random Forest）在整合元特徵時，未完全充分利用基礎模型的優勢。

3. 模型表現的優勢與劣勢

- **優勢**：
  - ✓ 隨機森林和 XGBoost 表現突出，反映出基於樹的模型對該數據集的適應性較好。
  - ✓ 集成模型通過綜合多個基礎模型的預測，提升了結果穩定性。
- **劣勢**：
  - ✓ 集成模型未能顯著超越最佳基礎模型（Random Forest 和 XGBoost），表明集成策略可能存在改進空間。
  - ✓ LSTM 的性能較低，可能因其對非時間序列特徵建模能力有限。

(LSTM+Random forest+XGBoost+Gradient boosting)集成模型(stacking)

該模型架構主要採用了 **Stacking 集成學習**，由多個基礎模型構成，包括：LSTM、隨機森林、XGBoost、梯度提升（Gradient Boosting）。此外，結合了 **元模型（Meta-Model）**，利用基礎模型的預測結果進行最終分類。

以下為表現評估結果：

```
Stacking Ensemble Accuracy: 0.9321
```

1. 準確率表現

- **整體準確率（Accuracy: 93.21%）**
  表示模型能正確分類測試樣本中 93.21% 的數據。相比單一模型（如 LSTM、隨機森林、XGBoost 等）的準確率，Stacking 集成模型更高，表明集成策略在結合不同基礎模型的優勢後提高了性能。

2. **集成學習效果分析**

Stacking 集成模型利用了**多個基礎模型的輸出**，進一步提升了泛化能力。

- **基礎模型貢獻**
  - ✓ **LSTM**：擅長處理時間序列數據，捕捉長期依賴特徵。
  - ✓ **隨機森林（Random Forest）**：適合處理非線性特徵，且具有較強的穩定性。
  - ✓ **XGBoost**：利用加權梯度提升，對少數類別或異常樣本有良好的建模能力。
  - ✓ **梯度提升（Gradient Boosting）**：擅長逐步修正誤差，提升分類效果。

  各模型在處理不同類型數據特徵上有其優勢，Stacking 利用這些模型的多樣性來提高整體準確率。

- **元模型的優勢**
  - ✓ 隨機森林作為元模型，根據基礎模型的預測輸出進行綜合學習。
  - ✓ 元特徵（來自基礎模型的預測）提供了更多的信息，使元模型能更好地區分不同類別。

五、AutoEncoder

以下為表現評估結果：

```
================ 評估結果一覽 =====
Accuracy: 0.85
Precision: 0.00
Recall: 0.00
F1 Score: 0.00
Confusion Matrix:
[[267  47]
 [  0   0]]
==================================
```

從輸出的結果可以看出模型的整體準確率（Accuracy）達到了 0.85，但其他指標（Precision、Recall 和 F1 Score）均為 0.00。同時，混淆矩陣顯示模型完全無法正確預測少數類別（即「1」）。

具體表現在以下兩點：

(1) 混淆矩陣：

- 第一行表示實際標籤為「0」（正常）的樣本中，267 個被正確分類，47 個被錯誤分類為「1」（異常）。
- 第二行表示實際標籤為「1」（異常）的樣本中，沒有一個樣本被正確分類。

(2) 指標表現：

- Precision（精確率）：模型在判斷為異常（1）時的準確性為 0，說明模型完全無法識別少數類別。
- Recall（召回率）：對於異常樣本（1），模型的檢測率為 0，表明所有異常樣本均被錯誤分類為正常樣本。
- F1 Score：精確率和召回率的調和平均為 0，進一步強調模型在處理異常類別時的表現不佳。

模型整體結論：

模型類型	Accuracy	Precision (少數類別)	Recall (少數類別)	F1 (少數類別)
Logistic Regression	0.743	0.15	0.65	0.25
XGBoost	0.915	0.333	0.250	0.286
Random Forest	0.927	0.225	0.281	0.250
LSTM	0.900	0.13	0.65	0.25
Autoencoder	0.850	0.00	0.00	0.00
Voting 集成 (KNN + RF + XGBoost)	0.885	0.225	0.281	0.250
Voting 集成 (LSTM + RF + XGBoost)	0.919	0.20	0.03	0.05
Stacking 集成	**0.932**	**0.333**	**0.250**	**0.286**

1. 最佳模型：

(1) **Stacking 集成模型（LSTM + Random Forest + XGBoost + Gradient Boosting）** 表現最佳，達到最高準確率（0.932）。

(2) 雖然準確率最高，但少數類別的 Precision 和 Recall 仍需改進。

2. 改進方向

(1) **處理類別不平衡**

使用 SMOTE 或其他過採樣技術，增加少數類別的樣本數量。調整模型的類別權重，讓模型更重視少數類別。

(2) **優化元模型**

更換元模型（如使用 XGBoost 作為元模型）或調整其超參數。

(3) **改進評估指標**

引入 AUC-ROC、Precision-Recall 曲線等指標，全面衡量模型性能。

(4) **深度調參**

對 LSTM 和集成模型進行深度超參數調整，進一步提升少數類別的識別性能。

3. 應用建議

在類別不平衡場景下，僅關注準確率可能會掩蓋模型的不足。更應關注少數類別的 Precision 和 Recall，尤其是模型實際應用於異常檢測或風險預警時。

以下為本專案之總結：

本研究聚焦於 SECOM 資料集的應用，結合資料探勘技術，探索如何提升半導體製造中的良率，並提出針對類別不平衡的改進策略。研究目標包括分析資料中的高維特徵與缺失值，建立預測模型，驗證其在小規模數據集上的效果，並提出製程優化建議。本研究選用多種模型（如隨機森林、Logistic Regression、XGBoost、LSTM）進行分析，並比較其性能。結果顯示，Stacking 集成模型達到最高準確率（93.2%），展現了整合模型的優勢。然而，少數類別的精確率和召回率表現不足，顯示類別不平衡仍是模型需要面對的挑戰。

基於結果，建議採用 SMOTE 增加少數類別樣本，調整模型類別權重，並優化元模型參數，進一步提升少數類別的識別能力。同時，應引入 Precision-Recall 曲線等評估指標，全面衡量模型性能。透過本研究，我們不僅驗證了基礎模型的實用性，還為製程優化和智能製造提供了具體方向與實踐參考，期望搭建學術研究與產業應用之間的橋樑，為半導體製造良率的提升提供支撐。