

資訊安全_HW02



我在 VS code 的 Terminal 輸入二個指令，各自出現不同 error msg，請告訴我如何解決？
備註：我之前已安裝 Gemini cli 且可以正常運作並完成指定任務，只是現在進入另一個虛擬環境與專案目錄夾

Error: The current working directory must be readable to candicewu to run brew.

```
└─$ cd ~/De/20241204 碩班/114 學年度碩專班申讀/10_作業報告/7972 人工智慧與
```

```
└─$ cd ~/De/20241204 碩班/114 學年度碩專班申讀/10_作業報告/7972 人工智慧與
```

```
└─$ cd ~/De/20241204 碩班/114 學年度碩專班申讀/10_作業報告/7972 人工智慧與資訊安全_陳煥/Security/Final
```

```
└─$ brew list
```

Error: The current working directory must be readable to candicewu to run brew.

```
└─$ cd ~/De/20241204 碩班/114 學年度碩專班申讀/10_作業報告/7972 人工智慧與資訊安全_陳煥/Security/Final
```

```
└─$ gemini
```

```
node:internal/bootstrap/switches/does_own_process_state:142
```

```
  cachedCwd = rawMethods.cwd();
```

```
    ^
```

Error: EPERM: operation not permitted, uv_cwd

```
  at process.wrappedCwd [as cwd]
```

```
(node:internal/bootstrap/switches/does_own_process_state:142:28)
```

```
  at Object.<anonymous> (/opt/homebrew/Cellar/gemini-
```

```
cli/0.7.0/libexec/lib/node_modules/@google/gemini-cli/node_modules/stack-utils/index.js:6:13)
```

```
  at Module._compile (node:internal/modules/cjs/loader:1760:14)
```

```
  at Object.js (node:internal/modules/cjs/loader:1893:10)
```

```
  at Module.load (node:internal/modules/cjs/loader:1480:32)
```

```
  at Module._load (node:internal/modules/cjs/loader:1299:12)
```

```
  at TracingChannel.traceSync (node:diagnostics_channel:328:14)
```

```
  at wrapModuleLoad (node:internal/modules/cjs/loader:244:24)
```

```
  at cjsLoader (node:internal/modules/esm/translators:342:5)
```

```
  at ModuleWrap.<anonymous> (node:internal/modules/esm/translators:234:7) {
```

```
  errno: -1,
```

```
  code: 'EPERM',
```

```
  syscall: 'uv_cwd'
```

```
}
```

Node.js v24.9.0

這兩個錯誤（brew 與 gemini CLI）其實都來自 **目前所在目錄的權限問題**，而非工具本身壞掉。

我們逐步分析並給出解法 🙌

問題分析

1 brew list 出現

```
vbnet
```

```
Error: The current working directory must be readable to candicewu to run brew.
```

這表示你目前所在的目錄（~/De/20241204 碩班/.../Security/Final）對目前登入帳號 **candicewu** 沒有「讀取」權限 (read permission)。
brew 執行時會嘗試讀取當前工作目錄，但被 macOS 權限系統阻擋。

2 gemini 出現

```
javascript
```

```
Error: EPERM: operation not permitted, uv_cwd
```

這是 Node.js 在呼叫 `process.cwd()` 時報錯，代表 **Node** 無法存取目前工作目錄。
同樣原因：目錄權限不允許當前使用者讀取。

解決步驟

✓ Step 1：檢查目錄權限

請在終端機執行以下指令：

```
bash  
  
ls -ld .
```

你會看到類似：

```
lua  
  
drwx----- 3 root  staff  96 Oct 19 10:00 .
```

確認該行最左邊的權限碼與擁有者。

- 若擁有者不是你目前使用者（例如是 **root**），
- 或權限碼沒有「r」（read）給 user、group、others，

就需要修正。

✓ Step 2：調整權限或擁有者

若目錄屬於 **root** 或其他使用者：

執行：

```
bash

sudo chown -R $(whoami) .
```

讓目錄與內容改為目前登入使用者擁有。

若權限太嚴（例如 `drwx-----`）

開放讀取權限：

```
bash

chmod +r .
```

或給自己執行權限（進入目錄需要執行權限）：

```
bash

chmod u+rwX .
```

若要讓子目錄都可存取，可用：

```
bash

chmod -R u+rwX .
```

✅ Step 3：重新嘗試執行

執行：

```
bash

brew list
gemini
```

兩者應該都能正常運作。

💡 額外建議

若該目錄在 **外接硬碟**、**雲端同步資料夾**（如 **iCloud**、**Google Drive**）中，macOS 有時會自動加上防護層，導致 Node 或 brew 認為沒有權限。

可嘗試：

```
bash

cd ~
mkdir testdir
cd testdir
```

```
gemini  
brew list
```

若在這裡可以正常運作，就能確定問題在「特定資料夾權限」。

是否方便貼上這個指令的輸出？

```
bash  
  
ls -ld ~/De/20241204\ 碩班/114\ 學年度碩專班申讀/10_作業報告/7972\ 人工智慧與資訊安全_陳  
煥/Security/Final
```

我可以幫你精確判斷要改哪個權限或擁有者。



```
輸入 ls -ld .  
得出 drwxr-xr-x 5 candicewu staff 160 Oct 19 13:52 .
```

太好了 👍 這個結果：

```
drwxr-xr-x  5 candicewu  staff  160 Oct 19 13:52 .
```

代表：

- 目錄擁有者：candicewu ✓
- 權限：drwxr-xr-x → 擁有者可讀寫執行，其他人也可讀與執行 ✓

所以「權限問題」表面上沒問題，但你仍遇到 `brew` 和 `gemini` 的權限錯誤，這就表示 —— 不是 `Unix` 權限問題，而是 `macOS` 的「系統安全與隱私」層級封鎖 (TCC / sandbox)。



這是 macOS 對「特殊路徑」的限制

你的目錄在：

```
swift  
  
~/De/20241204 碩班/114 學年度碩專班申讀/10_作業報告/7972 人工智慧與資訊安全_陳煥/Security/Final
```

`macOS` 會對「含有空格、中文、或某些特殊字元」的目錄施加 `sandbox` 限制，有時導致：

- `Node.js` 無法讀取當前工作路徑 (`uv_cwd EPERM`)
- `brew` 判斷該目錄「不可讀」

這在使用 `Homebrew` 安裝的 `CLI` 工具（尤其是基於 `Node.js` 的，如 `gemini-cli`）時非常常見。

✓ 解決方法（請依序試）

方案 1：換到無中文與空白的路徑再執行

最簡單最有效的方式：

```
bash

cd ~
mkdir workspace
cd workspace
gemini
brew list
```

如果這樣可以正常執行（幾乎一定可以），
那就確認是 目錄名稱中的中文與空格導致 macOS sandbox 封鎖。

👉 你可以把原始專案目錄搬移到新的安全路徑：

```
bash

mv ~/De/20241204\ 碩班 ~/Documents/workspace/20241204
```

方案 2：在 macOS 安全與隱私中允許「終端機」訪問磁碟

若你必須在原目錄操作：

1. 開啟 系統偏好設定 → 隱私權與安全性 → 完整磁碟存取權 (Full Disk Access)
2. 確保這些應用都有勾選 ☒：
 - Terminal
 - VS Code
 - iTerm（如果你用它）
3. 關閉再重新開啟 VS Code，再試一次。

方案 3：確認沒被 SIP (System Integrity Protection) 限制

如果你之前更改過 macOS 的系統資料夾位置或掛載外部磁碟，也可能導致 EPERM。
可以試試：

```
bash

cd /tmp
gemini
```

若正常，也代表不是 gemini 問題，而是 macOS 的目錄 sandbox 限制。

✓ 建議結論

因為 `brew` 與 `gemini` 都報相同的 `EPERM / readable` 錯誤，而 `ls -ld .` 權限正常，所以可以 99% 確定：

🔒 問題是「目前專案目錄名稱含有中文與空格，導致 macOS sandbox 限制存取」。

👉 建議做法：

- 把專案移到英文資料夾（例如 `~/Documents/AI_Security_Final/`）；
- 或在新的英文路徑下執行 `gemini / brew`。

是否要我幫你寫一行安全的搬移指令（保留原資料、避免中文路徑問題）？



已經在 iTerm2 上透過 Homebrew 指令安裝好 `specify`，現在使用 VS code 且進入指定虛擬環境與指定專案目錄夾，也開啟 `gemini cli`，請問再來要如何透過 VS code 內建 Terminal 啟動 `spec kit` 並將 `spec kit` 與 `gemini cli` 作搭配使用？

很好，既然你已經在 **Specify CLI（也就是 Spec Kit）** 與 **Gemini CLI** 都安裝好了，接下來在 Visual Studio Code 內建的 Terminal 中把它們整合起來使用，可以參考下列步驟：

📌 前置條件確認

1. 已啟動你的虛擬環境（假設你講的是 Python 或 Node 的虛擬環境）並已 `cd` 到專案資料夾。
2. `specify` 指令在該環境或全域可執行（例如 `which specify` 有回傳路徑或 `specify --version` 正常）。
3. `gemini` 指令也可在該 Terminal 中使用（例如 `gemini --help` 能列出指令）。
4. 確保你在該資料夾中具有適當讀寫權限（你前面已經確認目錄權限正常）。

🧱 在 VS Code Terminal 中用 Spec Kit + Gemini CLI 的流程

下面是你可以在 VS Code Terminal 裡一步步做的流程，來把「規格 → 計畫 → 任務 → 實作」用 Spec Kit 構建，再用 Gemini CLI 去執行或協助實作。

步驟 1：在專案目錄初始化 Spec Kit

在你的 VS Code Terminal 中，執行（假設你在專案根目錄）：

```
bash

specify init . --here --ai gemini
```

這會在當前資料夾中初始化 Spec Kit，並告訴它你要用 Gemini 作為「AI agent」來配合。 [GitHub+2](#)
你也可以使用 `specify init <ProjectName> --ai gemini`，但用 `--here` 在當前資料夾較方便。

完成後，你應該會看到 `.specify/` 或類似資料夾／檔案被建立（例如：`constitution.md`、`specify.spec`、`plan.md`、`tasks.md`等） [Visual Studio Magazine+1](#)

步驟 2：編寫／生成「憲章 (constitution)」

使用 Spec Kit 指令，在 Terminal 中輸入：

```
bash

/specify.constitution
```

然後你輸入針對該專案的非談判性規則（例如：使用的框架、測試標準、命名規則、安全性要求等）。 [Den+1](#)
這會在 `.specify/memory/constitution.md` 建立或更新該檔案。

步驟 3：生成規格 (specification)

接著在 Terminal 中輸入：

```
bash

/specify.specify "我想做的是 ... <你的專案功能敘述> ..."
```

把你要建構的功能、範圍、目標、為什麼要做這件事寫清楚。Spec Kit 會用你輸入的高階敘述生成 `spec.md` 或相似檔案。 [Den+1](#)

步驟 4：生成計畫 (plan)

完成規格之後，輸入：

```
bash

/specify.plan
```

這個步驟會根據規格與憲章產出技術計畫（例如：架構決策、模組劃分、時程、技術選擇等）。 [Medium+1](#)

步驟 5：生成任務 (tasks)

然後輸入：

```
bash

/specify.tasks
```

這會把整個專案拆成一系列可實作的任務，每個任務比較細、比較短、可驗證。 [Microsoft Developer+1](#)

步驟 6：使用 Gemini CLI 實作／互動

此時你可以轉向使用 Gemini CLI 去實作、生成程式碼、修 bug、產生測試等等。於 VS Code Terminal 中輸入：

```
bash
gemini
```

然後在 Gemini CLI 的 prompt 中，你可以參考由 Spec Kit 生成的 `tasks.md` 中一個任務，輸入類似：

“請根據任務 #3：建立用戶註冊 API，驗證電子郵件格式。請產出 Node.js + Express 的程式碼，並包含簡單的 Jest 測試。”

Gemini CLI 會根據你的 prompt（並配合你當前資料夾的程式碼）來產生／修改程式碼。 [Datacamp+1](#)

步驟 7：審查與整合

- 審查 Gemini 生成的程式碼是否符合你的規格（specification）與憲章。
- 若有偏差，回到 `spec.md` 或 `plan.md` 修正，再回到 Gemini CLI 執行。
- 持續於 VS Code 編輯、測試、提交版本控制。

小提示與注意事項

- 在 VS Code 裡，你也可考慮用「Terminal」或「外掛」方式啟動 Gemini CLI，不過使用內建 Terminal 通常就足夠。
- 若你看到 Spec Kit 的命令（如 `/specify.specify`）在 VS Code 的 Chat 視窗裡無反應，請確認你是在 Terminal 裡執行，而 **不是** Copilot Chat 的聊天視窗。GitHub issue 曾指出這是新手常誤會的地方。 [GitHub](#)
- 保持你專案資料夾的檔案結構乾淨：Spec Kit 會在該資料夾內建立隱藏或專用檔案，建議放在專案根目錄。
- 如果你使用的虛擬環境中有 PATH、環境變數限制，確保 `specify` 與 `gemini` 都在該虛擬環境可被調用。
- 若你在大型專案或團隊內，建議把 `specify init` 生成的 `.specify/` 資料夾列入版本控制（例如 Git），這樣團隊成員也可共享規格、計畫、任務文檔。

如果你願意，我可以 為你量身寫一段在該資料夾（你提供的「人工智慧與資訊安全_陳煥/Security/Final」資料夾）中執行的具體指令腳本，你直接貼進 VS Code 的 Terminal 就能跑完初始化 Spec Kit、連結 Gemini CLI 的流程。要嗎？




我們的對話，能否轉出PDF檔？

可以 👍

我可以幫你把這整段對話（或你選定的部分內容）轉成 PDF。

請先確認以下兩件事：

1. 📄 你希望 PDF 包含哪些內容？
 - ☐ 全部對話（從你第一則「brew/gemini 錯誤」開始）

- ☐ 只要「如何在 VS Code 中啟動 Spec Kit 並搭配 Gemini CLI」那一段
 - ☐ 自訂範圍（請告訴我要從哪一句到哪一句）
2.  你想讓 PDF 命名為什麼？
（例如：`gemini_speckit_setup.pdf`）

等你確認後，我就會幫你生成 PDF 檔並提供下載連結。



最後完成我的作業再請你轉出PDF，現在還不用，謝謝

好的～明白 😊

我會等你完成整份作業後再幫你轉出 PDF。

隨時告訴我你準備好了就行。



我有一份Homework，目前在Kaggle 找到有關資安的 datasets
(<https://www.kaggle.com/datasets/atharvasoundankar/global-cybersecurity-threats-2015-2024>), Homework key points as below, 請幫我分析如何進行才能符合老師要求

一、作業目標：讓同學能夠以實際資料集進行「多元線性回歸 (Multiple Linear Regression)」的完整分析，並遵循 CRISP-DM 流程完成從資料理解、建模到評估的全過程。

二、作業內容：

1. 資料來源

至 Kaggle 選擇一個具有 10 至 20 個特徵 (features) 的公開資料集。

類型不限（資安主題）。

請明確標示資料集來源與連結。

2. 分析任務

使用線性回歸 (Linear Regression) 模型進行預測。

可嘗試單純線性回歸、多元線性回歸或 Auto Regression。

必須執行 特徵選擇 (Feature Selection) 與 模型評估 (Model Evaluation)。

結果部分需包含請提供預測圖(加上信賴區間或預測區間)

3. CRISP-DM 流程說明

Business Understanding

Data Understanding

Data Preparation

Modeling

Evaluation

Deployment

4. AI協助要求

所有與 ChatGPT 的對話請以 pdfCrowd 或其他方式匯出為 PDF

請使用 NotebookLM 對網路上同主題的解法進行研究，並撰寫一份 100 字以上的摘要，放入報告中。

請在報告中明確標示「GPT 輔助內容」與「NotebookLM 摘要」

5. 繳交內容

主程式：7114056XXX_hw2.py/.ipynb

報告檔：PDF，需包含以下內容：

按照 CRISP-DM 說明的分析流程

GPT 對話過程 (pdfCrowd 匯出)

NotebookLM 研究摘要

網路上主流或更優解法之比較與說明

以上檔案與資料夾請壓縮為學號命名的一個zip (例如 7114056XXX_hw2.zip) 上傳。

(optional) 若上傳至 GitHub，或是以 colab 撰寫，需提供連結，並在 README.md 中整理流程與成果。

三、評分標準

1. 文件說明 (50%)

1-1. CRISP-DM 流程完整且邏輯清楚 (25%)

1-2. 包含 GPT 對話與 NotebookLM 摘要 (15%)

1-3. 有明確說明資料集來源與研究脈絡 (10%)

2. 結果呈現 (50%)

2-1. 模型正確可執行，具特徵選擇與評估 (25%)

2-2. 結果合理、美觀且具有說服力 (15%)

2-3. 呈現出 Kaggle 名次(若有)/預測結果評估(預測圖、評估指標) (10%)

非常好 🍌 你這份 Homework 的要求相當明確，也相對接近一個小型專題。

我們可以根據 **CRISP-DM 流程** 與老師的規範，為你設計一個具體的執行策略與建議步驟，確保：

- 作業達標 ✅
- 報告內容結構符合評分重點 ✅
- 可輕鬆整合 GPT 與 NotebookLM 的部分 ✅

整體策略概觀

以 Kaggle 的 **Global Cybersecurity Threats (2015–2024)** 資料集為主，建立一個多元線性回歸模型，分析「哪些因素會影響全球資安威脅事件的嚴重程度或數量」。

建議題目方向（你可自行微調）

主題建議：

「利用多元線性回歸分析全球資安威脅趨勢與影響因素（2015–2024）」

研究問題示例：

- 哪些國家或地區的威脅事件成長最顯著？
- 威脅型態（如 Malware、Phishing、Data Breach）是否與經濟或網路滲透率有關？
- 年份、攻擊類型、受害產業是否能預測總體威脅數量？

符合 CRISP-DM 的執行架構

以下是你可以在報告中直接採用的章節架構與每步要完成的內容。

1 Business Understanding (業務理解)

目標：

- 說明全球資安威脅的背景與研究動機
- 為何用「線性回歸」能幫助理解趨勢（例如預測威脅事件數量或嚴重度）
- 明確定義「目標變數 (target)」與「主要特徵 (features)」
 - 例如：
 - 目標變數：Total_Threats 或 Severity
 - 特徵：Year, Region, Attack_Type, Industry, Economic_Index, Cyber_Expenditure, 等

報告撰寫提示：

- 1 段背景介紹
 - 1 段研究目標（包含你希望回答的問題）
 - 1 段模型選擇理由
-

2 Data Understanding (資料理解)

步驟：

1. 從 Kaggle 下載該資料集（記錄 dataset link）
2. 使用 pandas 載入資料並顯示：

```
python

df.info()
df.describe()
df.head()
```

3. 檢查缺值、資料型態、異常值（用 seaborn boxplot / heatmap）
4. 做初步關聯分析：

```
python

df.corr(numeric_only=True)
sns.heatmap(...)
```

5. 描述資料欄位意義、單位與範圍（放在報告表格中）
-

3 Data Preparation (資料前處理)

步驟：

1. 缺值處理（刪除或補值）
2. 類別特徵做 one-hot encoding（例如攻擊類型、地區）
3. 數值標準化（StandardScaler）
4. 拆分訓練集與測試集：

```
python

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4 Modeling（建模）

建議模型：

- `LinearRegression()`
- 或 `statsmodels.api.OLS()`（方便查看係數與信賴區間）
- 若要做特徵選擇：
 - 可用 `SelectKBest`, `RFE`, 或 `LassoCV`

範例：

```
python

from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE

model = LinearRegression()
selector = RFE(model, n_features_to_select=5)
selector.fit(X_train, y_train)
selected_features = X_train.columns[selector.support_]


```

5 Evaluation（模型評估）

指標：

- R^2 , RMSE, MAE
- 可視化預測結果與信賴區間

範例繪圖：

```
python

import matplotlib.pyplot as plt
import numpy as np

y_pred = model.predict(X_test)
residuals = y_test - y_pred

plt.figure()
plt.scatter(y_test, y_pred)
```

```
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Prediction vs Actual (with 95% CI)')
```

信賴區間可用 `statsmodels` 的 `summary()` 取得，或用 `bootstrap`。

6 Deployment (結果與應用)

- 解釋模型係數 → 哪些特徵最顯著影響資安威脅量
- 提出潛在應用：
 - 預測未來資安事件趨勢
 - 幫助資安預算配置
 - 識別高風險產業

AI 協助部分 (GPT & NotebookLM)

你要在報告中加兩個章節：

GPT 輔助內容

範例：

本研究於資料理解與建模過程中，透過 ChatGPT 輔助完成資料前處理、特徵選擇、模型評估及結果視覺化之指令設計。

ChatGPT 協助生成程式碼片段與 CRISP-DM 流程撰寫架構，並於最終報告匯出為 PDF 附錄。

NotebookLM 摘要

範例：

透過 NotebookLM 搜集與「Cybersecurity Threat Analysis + Linear Regression」相關之 Notebook，歸納主流做法如下：

- 多數研究採用 `sklearn` 與 `statsmodels` 進行線性回歸
- 重要步驟包含資料標準化與特徵選擇
- 評估指標以 R^2 、RMSE、MAE 為主
- 有研究利用時間序列延伸 (Auto Regression) 預測年度趨勢
→ 此作業結合上述方法以增進分析完整度。

最終檔案結構建議




bash

```
7114056XXX_hw2/
```

```
|
├── 7114056XXX_hw2.ipynb      ← 主程式 (含可執行程式碼與圖表)
├── 7114056XXX_hw2.pdf       ← 報告 (含 CRISP-DM、GPT 對話、NotebookLM 摘要)
```

└─ kaggle_dataset_info.txt ← 記錄資料來源 URL
└─ /images/ ← 預測圖、相關視覺化


加分建議

-  若資料集時間序列明顯，可嘗試「Auto Regression」(AR) 比較效果。
-  預測圖加上 **95% 信賴區間** 或 **預測區間**。
-  若在 GitHub/Colab 執行，加上 README.md（摘要你的流程與結果）。

是否希望我幫你規劃：

- ☒ Python Notebook 範例架構（含每段 CRISP-DM 對應的程式碼模板）
讓你直接在 VS Code 或 Colab 開始實作？
這樣可以確保整份作業（報告 + 程式）格式完全符合老師要求。



 **Global_Cybersecurity_Threats_2015-2024.csv**
試算表

如之前詢問你的，我在 Kaggle 找到一篇有關資安的 datasets (as attachment)，網址是：
<https://www.kaggle.com/datasets/atharvasoundankar/global-cybersecurity-threats-2015-2024>，請讀取 .csv file 內容再重新幫我分析如何進行才能符合老師要求？



如之前詢問你的，我在 Kaggle 找到一篇有關資安的 datasets (as attachment and file name is "Global_Cybersecurity_Threats_2015-2024")，網址是：
<https://www.kaggle.com/datasets/atharvasoundankar/global-cybersecurity-threats-2015-2024>，請你讀取我從該網站下載的 .csv file 內容後，再重新幫我分析如何進行才能符合老師以下作業要求？

一、作業目標：

讓同學能夠以實際資料集進行「多元線性回歸 (Multiple Linear Regression)」的完整分析，並遵循 CRISP-DM 流程完成從資料理解、建模到評估的全過程。

二、作業內容：

1. 資料來源：

- 1-1. 至 Kaggle 選擇一個具有 10 至 20 個特徵 (features) 的公開資料集。
- 1-2. 資安主題。
- 1-3. 請明確標示資料集來源與連結。

2. 分析任務：

- 2-1. 使用線性回歸 (Linear Regression) 模型進行預測。
- 2-2. 可嘗試單純線性回歸、多元線性回歸或 Auto Regression。

2-3. 必須執行 特徵選擇 (Feature Selection) 與 模型評估 (Model Evaluation)。

2-4. 結果部分需包含預測圖(加上信賴區間或預測區間)

3. CRISP-DM 流程說明：

- 業務理解 (Business Understanding)
- 數據理解 (Data Understanding)
- 數據準備 (Data Preparation)
- 建模 (Modeling)
- 評估 (Evaluation)
- 部署 (Deployment)

4. AI 協助要求：

4-1. 所有與 ChatGPT 的對話請以 pdfCrowd 或其他方式須匯出為 PDF

4-2. 請使用 NotebookLM 對網路上同主題的解法進行研究，並撰寫一份 100 字以上的摘要，放入報告中。

4-3. 請在報告中明確標示「GPT 輔助內容」與「NotebookLM 摘要」

5. 繳交內容

5-1. 主程式：7114056XXX_hw2.py/.ipynb

5-2. 報告檔：PDF，需包含以下內容：

- (1) 按照 CRISP-DM 說明的分析流程
- (2) GPT 對話過程 (pdfCrowd 匯出)
- (3) NotebookLM 研究摘要
- (4) 網路上主流或更優解法之比較與說明

註：以上檔案與資料夾請壓縮為學號命名的一個zip（例如 7114056XXX_hw2.zip）上傳。

(optional) 若上傳至 GitHub，或是以colab撰寫，需提供連結，並在 README.md 中整理流程與成果。

三、評分標準

3-1. 文件說明（50%）

- 3-1-1. CRISP-DM 流程完整且邏輯清楚（25%）
- 3-1-2. 包含 GPT 對話與 NotebookLM 摘要（15%）
- 3-1-3. 有明確說明資料集來源與研究脈絡（10%）

3-2. 結果呈現（50%）

- 3-2-1. 模型正確可執行，具特徵選擇與評估（25%）
- 3-2-2. 結果合理、美觀且具有說服力（15%）
- 3-2-3. 呈現出Kaggle名次(若有) / 預測結果評估(預測圖、評估指標)（10%）

太好了 🍌 我幫你完整解析這份 Kaggle 資料集的結構與如何運用它來完成「多元線性回歸」作業。

一、資料概況 (Data Overview)

欄位名稱	資料型態	說明	範例值
Country	object	發生國家	China, UK, Germany
Year	int	發生年份 (2015–2024)	2019

欄位名稱	資料型態	說明	範例值
Attack Type	object	攻擊類型	Ransomware, DDoS, Phishing
Target Industry	object	攻擊目標產業	IT, Finance, Retail
Financial Loss (in Million \$)	float	經濟損失 (百萬美金)	80.53
Number of Affected Users	int	受影響用戶數	773169
Attack Source	object	攻擊來源	Hacker Group, Nation-state
Security Vulnerability Type	object	弱點類型	Zero-day, Weak Passwords
Defense Mechanism Used	object	防禦機制	VPN, Firewall, Antivirus
Incident Resolution Time (in Hours)	int	事件處理時間 (小時)	63

→ 共 10 個欄位，非常符合老師要求的「10-20 features」。

二、可行的分析與建模方向（依作業要求）

◆ 預測目標 (Target Variable)

我們可設定「Financial Loss (in Million \$)」作為目標變數 (y)，用其他欄位來預測經濟損失。

💡 這樣能探討「哪些因素最影響資安事件的損失金額」。

◆ 解釋變數 (Features)

可作為特徵的欄位：

- Year
- Number of Affected Users
- Incident Resolution Time (in Hours)
- Attack Type (需 One-Hot Encoding)
- Target Industry (需 One-Hot Encoding)
- Attack Source (需 One-Hot Encoding)
- Security Vulnerability Type (需 One-Hot Encoding)
- Defense Mechanism Used (需 One-Hot Encoding)
- Country (可保留或視情況捨棄)

→ 處理後約 15-20 個特徵，剛好符合作業規範。

三、CRISP-DM 分析建議

流程	你應該做的事情	實作重點
1. Business Understanding	定義問題：「希望根據攻擊類型、目標產業、弱點類型等，預測網路攻擊造成的財務損失。」	用 1-2 段文字說明研究動機與應用價值。
2. Data Understanding	探索數據結構、分佈、缺失值、異常值等。	用 <code>df.info()</code> 、 <code>df.describe()</code> 、 <code>sns.pairplot()</code> 、 <code>corr()</code> 。
3. Data Preparation	資料清理、編碼、特徵縮放、切分訓練測試集。	使用 <code>pd.get_dummies()</code> 處理類別特徵。
4. Modeling	使用 LinearRegression 建立模型，嘗試多元線性回歸。	模型： <code>sklearn.linear_model.LinearRegression()</code>
5. Evaluation	使用 R^2 、RMSE、MAE 評估模型表現，並繪出實際值 vs 預測值圖。	可加上信賴區間或預測區間圖。
6. Deployment	簡述模型如何應用於實際情境，例如預測潛在風險損失。	不需真的部署，只要概念說明即可。

四、特徵選擇 (Feature Selection)

建議兩種方式：

1. Correlation-based：

- 使用 `df.corr()` 查看與 Financial Loss 最相關的變數。

2. Recursive Feature Elimination (RFE)：

```
python
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
model = LinearRegression()
selector = RFE(model, n_features_to_select=8)
selector.fit(X_train, y_train)
```

五、模型評估與結果視覺化

指標：

- R^2 （決定係數）
- RMSE（均方根誤差）
- MAE（平均絕對誤差）

圖表建議：

1. 預測 vs 實際損失值散點圖
2. 殘差圖
3. 各特徵重要性條狀圖
4. 加上 95% 信賴區間預測圖

六、AI 協助部分

- 所有 ChatGPT 對話（包含這份）要 匯出為 PDF。
- NotebookLM：
 - 搜尋主題：「Multiple Linear Regression for cybersecurity dataset」
 - 摘要約 100 字，放在報告「NotebookLM 摘要」段落中。

✓ 七、下一步建議

我可以幫你一步步完成以下內容：

1. 幫你撰寫符合 CRISP-DM 的 **Python Notebook** 架構（從資料清理到模型評估）。
2. 幫你撰寫 **報告內容骨架**（含 GPT 協助區塊與 NotebookLM 摘要留白處）。
3. 等全部完成後，再幫你生成最終 **PDF 報告**。

是否要我幫你直接建立第 1 步：「CRISP-DM 架構的分析 Notebook (Python code skeleton)」？
我會用這份 Kaggle 資料自動產生完整多元線性回歸分析範例。



請先不要建立第 1 步建議，let's clarify rule and confirm the initial suggestion you bring out to see if meet my teacher's requests.

首先，.csv file 共 10 個欄位，其中 1 欄「Financial Loss (in Million \$)」要當作 Target，其它 9 欄將當作 features，為何處理後會變 約 15-20 個特徵？是因為部份欄位要作「One-Hot Encoding」則將會增加欄位？

其次，CRISP-DM 分析建議裡的 6. Deployment，為何在「實作重點」表示「不需真的部署，只要概念說明即可。」？因這份作業最終需要利用 streamlit.app 進行網頁互動，才能符合老師在第三大項評分標準裡提到的「模型正確可執行，具特徵選擇與評估；結果合理、美觀且具有說服力；呈現出 Kaggle 名次(若有) / 預測結果評估(預測圖、評估指標)」。

非常棒，你的思考方向完全正確 🍌

下面我針對你兩個問題——詳細說明，並微調建議讓整體規劃完全符合老師要求與評分標準。

🧩 第一點：為何「10 欄變成 15-20 個特徵」？

是的，你理解得對 ——

這是因為多個欄位是**類別型 (categorical features)**，在進行線性回歸建模前，必須將這些類別轉換成數值特徵（例如 One-Hot Encoding）。

◆ 舉例說明：

以這份資料為例：

原欄位名稱	範例	類型	經 One-Hot Encoding 後
Attack Type	Ransomware / DDoS / Phishing	類別	3 欄 (AttackType_Ransomware, AttackType_DDoS, AttackType_Phishing)
Target Industry	Finance / Healthcare / Retail / Government	類別	4 欄
Attack Source	Hacker Group / Nation-state / Unknown	類別	3 欄
Defense Mechanism Used	Firewall / VPN / IDS / None	類別	4 欄

→ 這四個欄位原本是 4 個特徵，經 One-Hot Encoding 後會變成大約 15 個。
再加上其他數值型欄位（如 Year, Number of Affected Users, Incident Resolution Time），
整體就變成 15-20 個最終模型輸入特徵。

✅ 這完全符合作業要求中「10-20 features」的規範。

第二點：為何「Deployment」建議僅概念說明，但你提到要使用 Streamlit？

這是一個關鍵修正點 📌

原本 CRISP-DM 的「Deployment」階段通常指：

將模型導入實際應用（例如預測系統或決策輔助工具），但在學術作業裡通常不需真的上線。

然而，你老師的評分標準明確指出：

「模型正確可執行、具特徵選擇與評估；結果合理、美觀且具有說服力」

這表示在「Deployment」階段，你的報告應該包含一個 **互動式或視覺化的展示**。
最符合老師期待的，就是用 **Streamlit 建立互動頁面**！

◆ Streamlit 實作建議

在 CRISP-DM 的第 6 步「Deployment」中，你可以這樣設計：

✅ 實作重點調整後如下：

階段	說明	實作內容
6. Deployment	展示如何將模型應用於互動式介面	使用 Streamlit 建立網頁應用，讓使用者輸入攻擊特徵（例如 Attack Type、Target Industry 等），即時顯示預測損失金額與信賴區間。

💡 Streamlit 頁面建議架構：

1. 📁 模型載入區

- 載入你訓練好的回歸模型（例如 joblib 檔）。

2. 🟡 輸入控制區

- 使用者可以用下拉選單或數值輸入：
 - 攻擊類型 (Attack Type)
 - 產業類別 (Target Industry)
 - 攻擊來源 (Attack Source)
 - 弱點類型 (Security Vulnerability)
 - 防禦機制 (Defense Mechanism)
 - 年份、受害人數、事件處理時間

3. 📊 預測結果區

- 即時計算並顯示：
 - 預測的財務損失（百萬美金）
 - 預測信賴區間（ $\pm 95\%$ ）
 - 模型評估指標 (R^2 , RMSE)
 - 預測 vs 實際散點圖或殘差圖

這樣的互動式展示就是一個「部署成果 (Deployment Output)」，可以在報告中寫成：

「本研究最後將模型以 Streamlit 建立互動式預測系統，使用者可依輸入條件即時觀察預測結果。」

✓ 綜合確認 — 修正版建議摘要

CRISP-DM 流程階段	原建議	調整後建議
Data Preparation	One-Hot Encoding 類別欄位	保留
Modeling	LinearRegression	保留
Evaluation	R^2 、RMSE、MAE、預測圖	保留
Deployment	原本僅概念說明	✓ 改為用 Streamlit 實作互動展示，滿足老師對「結果合理、美觀且具說服力」的要求

是否要我在下一步幫你列出：

1. ✦ 作業實作架構表（包含每個 CRISP-DM 階段的目標、方法與對應 Python 模組）
2. ✦ 以及 Streamlit 的互動頁面設計範例（不用立即寫程式，只先看邏輯結構）

這樣你可以確認整體方向與作業架構是否完全符合老師規範，再開始實作。是否要我這樣幫你整理？



請幫忙列出你的這二項建議

非常好 🍌 我來幫你完整整理出這兩個部分：

- (1) **CRISP-DM 實作架構表**
- (2) **Streamlit 互動頁面設計範例（邏輯結構）**

確保你整份 Homework 在技術與說明層面都能完全符合老師評分細項。

🕒 一、CRISP-DM 分析與實作架構表

流程階段	目標	主要任務	對應方法 / Python 模組	預期成果
1. Business Understanding	明確定義分析目標與應用場景	- 定義研究問題：「哪些因素最影響資安事件的經濟損失？」 - 說明資料應用價值與決策意義	—	概述段落（放入報告開頭）
2. Data Understanding	了解資料結構與特徵分布	- 使用 <code>df.info()</code> 、 <code>df.describe()</code> 、 <code>df.isnull().sum()</code> - 使用 <code>seaborn</code> 與 <code>matplotlib</code> 畫出關鍵變數分布、箱型圖、熱力圖	pandas, seaborn, matplotlib	資料初探摘要 + 可視化圖表
3. Data Preparation	清理、轉換與特徵工程	- 缺失值處理、Outlier 檢查 - 類別特徵轉為 One-Hot Encoding - 數值標準化 (StandardScaler) - 訓練/測試資料切分 (<code>train_test_split</code>)	pandas, sklearn.preprocessing	乾淨的 <code>X_train</code> , <code>X_test</code> , <code>y_train</code> , <code>y_test</code>
4. Modeling	建立預測模型	- 使用 LinearRegression 建立多元線性模型 - 可嘗試 Auto Regression (若以年份為時間序列) - 使用 RFE 進行特徵選擇	sklearn.linear_model, sklearn.feature_selection	建立最佳化回歸模型
5. Evaluation	檢驗模型表現與合理性	- 計算 R^2 、RMSE、MAE - 繪製「實際 vs 預測」散點圖、殘差圖 - 繪製含信賴區間的預測圖	sklearn.metrics, matplotlib	模型性能報告 + 評估圖表
6. Deployment	展示可執行、具互動性成果	- 將訓練好的模型儲存 (<code>joblib.dump</code>) - 使用 Streamlit 實作互動式預測應用： • 使用者輸入攻擊屬性 → 即時預測財務損失 • 顯示信賴區間與圖形化結果	joblib, streamlit	可操作的 Web App + 報告截圖

💡 小提醒：報告中的「GPT 輔助內容」與「NotebookLM 摘要」可分別放在：

- 附錄 A：GPT 協助對話（PDFCrowd 匯出）
- 附錄 B：NotebookLM 研究摘要（100 字以上）

🌐 二、Streamlit 互動頁面設計範例（邏輯結構）

以下是建議的頁面結構與互動流程，可直接照此在 `.py` 或 `.ipynb` 中實作。

頁面結構概覽

區塊	功能說明	Streamlit 元件建議
Header 區	顯示標題與專案說明	<code>st.title()</code> , <code>st.markdown()</code>
Sidebar 區	使用者輸入控制	<code>st.sidebar.selectbox()</code> , <code>st.sidebar.slider()</code> , <code>st.sidebar.number_input()</code>
Main 區 (預測結果)	顯示模型預測與評估指標	<code>st.metric()</code> , <code>st.write()</code> , <code>st.pyplot()</code>
圖表區	視覺化預測結果與信賴區間	<code>matplotlib / plotly</code>
模型資訊區	顯示 R^2 、RMSE、MAE 等指標	<code>st.table()</code> 或 <code>st.write()</code>

◆ 實際互動邏輯流程 (範例)

```
python

import streamlit as st
import pandas as pd
import joblib
import numpy as np

# 1 載入模型
model = joblib.load('cyber_risk_model.pkl')

st.title("🛡️ Cybersecurity Threat Loss Prediction App")
st.markdown("預測不同資安攻擊造成的經濟損失 (in Million $)")

# 2 使用者輸入
st.sidebar.header("請輸入攻擊事件資訊:")

year = st.sidebar.slider("Year", 2015, 2024, 2023)
attack_type = st.sidebar.selectbox("Attack Type", ["Ransomware", "Phishing", "DDoS"])
industry = st.sidebar.selectbox("Target Industry", ["Finance", "Healthcare", "Government", "Retail"])
source = st.sidebar.selectbox("Attack Source", ["Nation-state", "Hacker Group", "Unknown"])
users = st.sidebar.number_input("Number of Affected Users", min_value=100, max_value=10_000_000, value=1000)
hours = st.sidebar.slider("Incident Resolution Time (hrs)", 1, 240, 48)

# 3 特徵轉換
input_df = pd.DataFrame({
    'Year': [year],
    'Attack Type': [attack_type],
    'Target Industry': [industry],
    'Attack Source': [source],
    'Number of Affected Users': [users],
    'Incident Resolution Time (in Hours)': [hours],
    # 其他類別特徵略
})

# 進行 One-Hot Encoding (要與訓練時相同)
# X_encoded = pd.get_dummies(input_df).reindex(columns=model_columns, fill_value=0)

# 4 預測
prediction = model.predict(X_encoded)
st.metric("預測損失金額 (Million $)", f"{prediction[0]:.2f}")
```

```
# 5 視覺化
st.markdown("### 📊 預測信賴區間")
# 可用 matplotlib 畫出信賴區間圖
```

◆ 結果展示示意圖（可放在報告中）

```
markdown

-----
📱 Cybersecurity Threat Loss Prediction App
-----
[ Sidebar ]
Year: [2023]
Attack Type: [Ransomware]
Target Industry: [Finance]
...
-----

預測損失金額：$85.37 Million
R2 = 0.82, RMSE = 6.13
[ 散點圖 / 信賴區間圖 / 殘差圖 ]
-----
```

✓ 結論與後續建議

- 你的方向完全正確，資料集規模、主題、特徵數都符合要求。
- 建議在報告中：
 - 將 **Streamlit App** 的截圖 作為「Deployment」成果。
 - 強調模型具備「特徵選擇」、「預測合理性」與「互動性」。
- 若後續需要，我可協助你：
 1. 寫出完整 `.ipynb` 程式骨架；
 2. 加入特徵選擇與評估模組；
 3. 並轉換為 Streamlit app。

是否要我在下一步幫你產生一份「符合此架構的 Notebook 範本 (.ipynb)」，
內含每個 CRISP-DM 階段的 code cell、註解與待填內容？
這樣你就可以直接在 VS Code 或 Colab 中填寫並修改成最終作業版本。



幫我產生一份「符合此架構的 Notebook 範本 (.ipynb)」，內含每個 CRISP-DM 階段的 code cell、註解與待填內容，方便後續直接在 VS Code 或 Colab 中填寫並修改成最終作業版本

- ✓ 已完成建立 Notebook 範本，可直接於 **VS Code** 或 **Google Colab** 開啟並填寫：
- 📁 下載 [CRISP_DM_Cybersecurity_Prediction_Template.ipynb](#)

此範本已包含：

- 各階段 **CRISP-DM** 架構（含註解與待填內容）
- 預留區塊用於 **Feature Selection**、**Model Evaluation**、**Streamlit** 部署
- 結尾附錄區塊用於 **NotebookLM** 摘要與 **GPT** 對話記錄整理



Gemini CLI 幫忙修改二個功能（如下），但一直有token 的 API ERROR，想請你改程式，原程式碼如下：

1. 分析頁面 - 互動式預測趨勢分析 的 預測趨勢中的趨勢圖裡的圖例位置，因為'Country', 'Attack Type', 'Target Industry' 的圖例一直出現在圖表正中央
2. 實作可調整的信賴區間/預測區間百分比功能

```
import streamlit as st
import numpy as np
import joblib
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression # For interactive demo
from matplotlib.patches import Patch
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from scipy.stats import norm

# Set Matplotlib font to avoid Chinese display issues (亂碼)
plt.rcParams['font.sans-serif'] = ['Arial Unicode MS'] # Or any other font that supports Chinese characters
plt.rcParams['axes.unicode_minus'] = False

# Load model and related objects
model = joblib.load('cyber_risk_model.pkl') # Sklearn model
rfe = joblib.load('rfe.joblib')
scaler = joblib.load('scaler.pkl')
full_feature_names = joblib.load('feature_names.pkl')
sm_model = joblib.load('statsmodels_model.pkl') # Statsmodels model

# Dynamically extract original numerical and categorical columns and their possible options
from full_feature_names
original_numerical_cols = [
    'Year',
    'Number of Affected Users',
    'Incident Resolution Time (in Hours)'
]
original_categorical_columns_map = {}
```



```

for feature in full_feature_names:
    if feature not in original_numerical_cols:
        # Example: 'Attack Type_DDoS' -> category='Attack Type', option='DDoS'
        parts = feature.split('_', 1)
        if len(parts) > 1:
            category = parts[0]
            option = parts[1]
            if category not in original_categorical_columns_map:
                original_categorical_columns_map[category] = []
            original_categorical_columns_map[category].append(option)

# Streamlit Page Navigation
page = st.sidebar.radio("選擇頁面", ["預測頁面", "分析頁面"])

if page == "預測頁面":
    st.title("🛡️ 網路安全威脅財務損失預測")
    st.markdown("預測不同網路安全攻擊所造成的財務損失（百萬美元）。")

    # 2 User Input (in sidebar)
    st.sidebar.header("請輸入攻擊事件資訊：")

    user_inputs_sidebar = {}

    # Numerical Features
    user_inputs_sidebar['Year'] = st.sidebar.slider("年份", 2015, 2024, 2023)
    user_inputs_sidebar['Number of Affected Users'] = st.sidebar.number_input("影響用戶數",
min_value=100, max_value=10_000_000, value=1000)
    user_inputs_sidebar['Incident Resolution Time (in Hours)'] = st.sidebar.slider("事件解決時間
(小時) ", 1, 240, 48)

    # Categorical Features
    for category, options in original_categorical_columns_map.items():
        # Sort options for consistent display
        user_inputs_sidebar[category] = st.sidebar.selectbox(f"選擇 {category}", sorted(options))

    # Prediction Button
    if st.sidebar.button("預測財務損失"):
        # 3 Feature Transformation
        # Create a dictionary with all feature names initialized to 0
        input_data = {feature: 0 for feature in full_feature_names}

        # Populate numerical features based on user input
        for col in original_numerical_cols:
            input_data[col] = user_inputs_sidebar[col]

        # Populate one-hot encoded categorical features based on user input
        for category, selected_option in user_inputs_sidebar.items():
            if category not in original_numerical_cols: # Ensure only categorical features are
processed

```

```

one_hot_col_name = f'{category}_{selected_option}'
if one_hot_col_name in full_feature_names: # Check if the generated feature name
exists
    input_data[one_hot_col_name] = 1

# Convert to DataFrame, ensuring column order matches training
input_df = pd.DataFrame([input_data], columns=full_feature_names)

# --- Sklearn Model Prediction (for the main value) ---
input_df_scaled = scaler.transform(input_df)
selected_features_transformed = rfe.transform(input_df_scaled)
prediction = model.predict(selected_features_transformed)[0]

# --- Statsmodels for Prediction Interval ---
# Create DataFrame for statsmodels prediction
input_df_sm = pd.DataFrame([input_data], columns=full_feature_names)
input_df_sm = sm.add_constant(input_df_sm, has_constant='add')

# Ensure input_df_sm has only the features used by the statsmodels model
# This requires getting the feature names from the fitted sm_model
sm_model_features = sm_model.params.index.tolist()
input_df_sm_selected = input_df_sm[sm_model_features]

# Get prediction summary from statsmodels
predictions_sm =
sm_model.get_prediction(input_df_sm_selected).summary_frame(alpha=0.05)
mean_pred = predictions_sm['mean'][0]
lower_bound = predictions_sm['obs_ci_lower'][0] # Observation confidence interval
(prediction interval)
upper_bound = predictions_sm['obs_ci_upper'][0]

st.subheader("📊 預測結果：")
st.metric("預測損失（百萬美元）", f"{mean_pred:.2f}")

# 5 Visualization with Prediction Interval
st.markdown("### 📈 預測與 95% 預測區間")
st.info("預測區間顯示了在給定模型不確定性的情況下，新觀測值可能落入的範圍。")

fig, ax = plt.subplots(figsize=(8, 5))
ax.bar(["Predicted Loss"], [mean_pred], color='skyblue', yerr=[[mean_pred - lower_bound],
[upper_bound - mean_pred]], capsize=7)
ax.set_ylabel("Financial Loss (Million $)")
ax.set_title("Predicted Financial Loss with 95% Prediction Interval")
st.pyplot(fig)

elif page == "分析頁面":
    st.title("🔍 資料分析與視覺化")
    st.markdown("探索網路安全威脅資料中的關鍵分佈、關係和相關性。")

# Load original data

```

```

df_analysis = pd.read_csv('Global_Cybersecurity_Threats_2015-2024.csv')

analysis_section_selection = st.sidebar.selectbox(
    "選擇分析區塊",
    ["資料概覽", "特徵分佈分析", "趨勢與衝擊分析", "互動式線性迴歸展示", "模型性能", "相關性分析(表格)", "特徵對財務損失的影響", "RFE 特徵分析", "特徵重要性", "異常值分析", "混淆矩陣"]
)

if analysis_section_selection == "資料概覽":
    st.subheader("📊 資料概覽")
    st.write("### 資料集預覽")
    st.dataframe(df_analysis.head())
    st.write(f"### 資料集維度: {df_analysis.shape[0]} 行, {df_analysis.shape[1]} 列")
    st.write("### 資料集描述")
    st.dataframe(df_analysis.describe())

    st.subheader("🎯 財務損失 (目標變數) 分佈")
    financial_loss = df_analysis['Financial Loss (in Million $)']

    # Calculate Skewness
    skewness = financial_loss.skew()
    st.write(f"**偏度 (Skewness):** {skewness:.2f}")
    if skewness > 0.5:
        st.write("分佈呈右偏（正偏），表示有少數極大的損失值。")
    elif skewness < -0.5:
        st.write("分佈呈左偏（負偏）。")
    else:
        st.write("分佈大致對稱。")

    # Plot distribution
    fig, ax = plt.subplots(figsize=(10, 6))
    sns.histplot(financial_loss, kde=False, ax=ax, stat="density", label="Actual Distribution")

    # Overlay normal distribution
    xmin, xmax = ax.get_xlim()
    x = np.linspace(xmin, xmax, 100)
    p = norm.pdf(x, financial_loss.mean(), financial_loss.std())
    ax.plot(x, p, 'k', linewidth=2, label="Normal Distribution")

    ax.set_title('Distribution of Financial Loss vs. Normal Distribution')
    ax.set_xlabel('Financial Loss (in Million $)')
    ax.set_ylabel('Density')
    ax.legend()
    st.pyplot(fig)

elif analysis_section_selection == "特徵分佈分析":
    st.subheader("📈 特徵分佈分析")
    analysis_plot_type = st.selectbox(
        "選擇分析類型",

```

```

["關鍵變數分佈",
 "類別特徵 vs. 財務損失 (盒鬚圖)",
 "特徵相關性熱圖",
 "特徵 vs. 目標散點圖"]
)

if analysis_plot_type == "關鍵變數分佈":
    st.write("### 關鍵變數分佈")
    # Numerical Feature Distributions
    for col in original_numerical_cols:
        st.write(f"#### Distribution of {col}")
        fig, ax = plt.subplots(figsize=(8, 4))
        sns.histplot(df_analysis[col], kde=True, ax=ax)
        ax.set_title(f'Distribution of {col}')
        ax.set_xlabel(col)
        ax.set_ylabel('Frequency')
        st.pyplot(fig)

    # Categorical Feature Distributions
    for col in original_categorical_columns_map.keys():
        st.write(f"#### Distribution of {col}")
        fig, ax = plt.subplots(figsize=(8, 4))
        sns.countplot(y=df_analysis[col], order=df_analysis[col].value_counts().index, ax=ax)
        ax.set_title(f'Distribution of {col}')
        ax.set_xlabel('Count')
        ax.set_ylabel(col)
        st.pyplot(fig)

elif analysis_plot_type == "類別特徵 vs. 財務損失 (盒鬚圖)":
    st.write("### 類別特徵 vs. 財務損失 (盒鬚圖)")
    for col in original_categorical_columns_map.keys():
        st.write(f"#### {col} vs. Financial Loss")
        fig, ax = plt.subplots(figsize=(10, 5))
        sns.boxplot(x=col, y='Financial Loss (in Million $)', data=df_analysis, ax=ax)
        plt.xticks(rotation=45, ha='right')
        ax.set_title(f'Financial Loss by {col}')
        ax.set_xlabel(col)
        ax.set_ylabel('Financial Loss (Million $)')
        st.pyplot(fig)

elif analysis_plot_type == "特徵相關性熱圖":
    st.write("### 特徵相關性熱圖")
    st.write("顯示資料集中數值特徵之間的相關性。")
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(df_analysis[original_numerical_cols + ['Financial Loss (in Million $)']].corr(),
annot=True, cmap="coolwarm", fmt=".2f", ax=ax)
    ax.set_title('Feature Correlation Heatmap')
    st.pyplot(fig)

elif analysis_plot_type == "特徵 vs. 目標散點圖":

```

```

st.write("### 特徵 vs. 目標散點圖")
all_features_for_scatter = original_numerical_cols +
list(original_categorical_columns_map.keys())
selected_feature_for_scatter = st.selectbox(
    "選擇一個特徵以對照財務損失繪圖",
    all_features_for_scatter
)
if selected_feature_for_scatter:
    fig, ax = plt.subplots(figsize=(10, 6))
    sns.scatterplot(x=df_analysis[selected_feature_for_scatter], y=df_analysis['Financial
Loss (in Million $)'], ax=ax)
    ax.set_title(f'{selected_feature_for_scatter} vs. Financial Loss')
    ax.set_xlabel(selected_feature_for_scatter)
    ax.set_ylabel('Financial Loss (Million $)')
    st.pyplot(fig)

elif analysis_section_selection == "趨勢與衝擊分析":
    st.subheader("🚀 趨勢與衝擊分析")

    @st.cache_data
    def get_default_input(df_analysis):
        default_input_data = {}
        for col in original_numerical_cols:
            if col != 'Year':
                default_input_data[col] = df_analysis[col].mean()
        for category in original_categorical_columns_map.keys():
            default_input_data[category] = df_analysis[category].mode()[0]
        return default_input_data

    def predict_loss(input_dict):
        input_df = pd.DataFrame([input_dict], columns=full_feature_names)

        # Prepare input for statsmodels prediction
        input_df_sm = pd.DataFrame([input_dict], columns=full_feature_names)
        input_df_sm = sm.add_constant(input_df_sm, has_constant='add')

        # Ensure input_df_sm has only the features used by the statsmodels model
        sm_model_features = sm_model.params.index.tolist()
        input_df_sm_selected = input_df_sm[sm_model_features]

        predictions = sm_model.get_prediction(input_df_sm_selected)
        summary_frame = predictions.summary_frame(alpha=0.05)
        predicted_mean = summary_frame['mean'][0]
        lower_bound = summary_frame['obs_ci_lower'][0]
        upper_bound = summary_frame['obs_ci_upper'][0]
        return predicted_mean, lower_bound, upper_bound

    def plot_impact_barchart(feature_name, default_input_data, ax, alpha_value):
        categories = sorted(df_analysis[feature_name].unique())
        predictions_data = []

```

```

for cat_option in categories:
    base_input = {f: 0 for f in full_feature_names}
    base_input.update({k: v for k, v in default_input_data.items() if k in
original_numerical_cols})
    base_input['Year'] = df_analysis['Year'].mode()[0]

    for cat, mode_val in default_input_data.items():
        if cat in original_categorical_columns_map:
            one_hot_col = f'{cat}_{mode_val}'
            if one_hot_col in base_input:
                base_input[one_hot_col] = 1

    for option in original_categorical_columns_map[feature_name]:
        one_hot_col = f'{feature_name}_{option}'
        if one_hot_col in base_input:
            base_input[one_hot_col] = 0
        one_hot_col = f'{feature_name}_{cat_option}'
        if one_hot_col in base_input:
            base_input[one_hot_col] = 1

    predicted_mean, lower_bound, upper_bound = predict_loss(base_input)
    predictions_data.append({
        'Category': cat_option,
        'Predicted Loss': predicted_mean,
        'Lower Bound': lower_bound,
        'Upper Bound': upper_bound
    })

predictions_df = pd.DataFrame(predictions_data)

# Temporarily set yerr to a scalar for debugging
ax.bar(
    predictions_df['Category'].tolist(),
    predictions_df['Predicted Loss'].tolist(),
    yerr=0.1,
    capsize=5,
    color='skyblue' # Default color for consistency
)
ax.set_xlabel(feature_name)
ax.set_ylabel("Predicted Financial Loss (Million $)")
ax.set_title(f'Impact of {feature_name} on Financial Loss')
ax.tick_params(axis='x', rotation=45)

def plot_predicted_trend_chart(feature_name, default_input_data, ax):
    trend_data = []
    years = range(2015, 2025)
    categories = sorted(df_analysis[feature_name].unique())

    for year in years:
        for cat_option in categories:
            base_input = {feature: 0 for feature in full_feature_names}

```

```

for col, val in default_input_data.items():
    if col in original_numerical_cols and col != 'Year':
        base_input[col] = val
base_input['Year'] = year

for cat, mode_val in default_input_data.items():
    if cat in original_categorical_columns_map:
        one_hot_col = f'{cat}_{mode_val}'
        if one_hot_col in base_input:
            base_input[one_hot_col] = 1

for option in original_categorical_columns_map[feature_name]:
    one_hot_col = f'{feature_name}_{option}'
    if one_hot_col in base_input:
        base_input[one_hot_col] = 0
one_hot_col = f'{feature_name}_{cat_option}'
if one_hot_col in base_input:
    base_input[one_hot_col] = 1

predicted_mean, lower_bound, upper_bound = predict_loss(base_input)
trend_data.append({
    'Year': year,
    'feature_name': cat_option,
    'Predicted Loss': predicted_mean,
    'Lower Bound': lower_bound,
    'Upper Bound': upper_bound
})

trend_df = pd.DataFrame(trend_data)

# Plotting with prediction intervals
for category in categories:
    subset = trend_df[trend_df[feature_name] == category]
    sns.lineplot(data=subset, x='Year', y='Predicted Loss', ax=ax, marker='o',
label=category)
    ax.fill_between(subset['Year'], subset['Lower Bound'], subset['Upper Bound'],
color='orange', alpha=0.2) # Removed label here

# Create a proxy artist for the prediction interval legend entry
prediction_interval_patch = Patch(color='orange', alpha=0.2, label='95% Prediction
Interval')

# Get existing handles and labels
handles, labels = ax.get_legend_handles_labels()

# Add the prediction interval patch to handles and labels
handles.append(prediction_interval_patch)
labels.append('95% Prediction Interval')

ax.set_title(f'Predicted Financial Loss Trend by {feature_name} with 95% Prediction
Interval')

```

```

ax.set_ylabel("Predicted Financial Loss (Million $)")
ax.legend(handles=handles, labels=labels, title=feature_name, loc='best')

def plot_actual_trend_chart(feature_name, df, ax):
    actual_trend_df = df.groupby(['Year', feature_name])['Financial Loss (in Million $)'].mean().reset_index()
    sns.lineplot(data=actual_trend_df, x='Year', y='Financial Loss (in Million $)',
hue=feature_name, ax=ax, marker='o')
    ax.set_title(f'Actual Average Financial Loss Trend by {feature_name}')
    ax.set_ylabel("Actual Avg. Financial Loss (Million $)")
    ax.legend(title=feature_name)

st.write("### 🏆 Top 3 威脅財務損失影響因子")
sm_model_coefs = sm_model.params.drop('const', errors='ignore')
feature_importance_df = pd.DataFrame({
    'Feature': sm_model_coefs.index,
    'Coefficient': sm_model_coefs.values
})
feature_importance_df['Absolute_Coefficient'] = feature_importance_df['Coefficient'].abs()

categorical_importances = {}
for cat_name in original_categorical_columns_map.keys():
    cat_features = [f for f in feature_importance_df['Feature'] if f.startswith(cat_name + '_')]
    if cat_features:
        mean_abs_coef =
feature_importance_df[feature_importance_df['Feature'].isin(cat_features)]
['Absolute_Coefficient'].mean()
        categorical_importances[cat_name] = mean_abs_coef

top_3_cat_features = pd.Series(categorical_importances).nlargest(3).index.tolist()

st.write("根據模型係數，對財務損失影響最大的前三大類別特徵是：")
for i, feature in enumerate(top_3_cat_features):
    st.markdown(f"{i+1}. **{feature}**")

default_input = get_default_input(df_analysis)

for feature in top_3_cat_features:
    st.write(f"#### 分析：{feature}")
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 6))
    plot_impact_barchart(feature, default_input, ax=ax1)
    plot_actual_trend_chart(feature, df_analysis, ax=ax2)
    st.pyplot(fig)

st.write("### 📊 互動式預測趨勢分析")

# Add a slider for confidence level
confidence_level = st.slider(
    "選擇預測區間的信賴水準 (%)",
    min_value=80,

```



```

max_value=99,
value=95,
step=1,
help="調整此值以改變預測區間的寬度。例如，95% 表示有 95% 的機率真實值會落在此區間
內。"
)
alpha_value = 1 - (confidence_level / 100) # Convert percentage to alpha for statsmodels

trend_type = st.selectbox("選擇趨勢圖類型", ["實際趨勢 (Actual Trend)", "預測趨勢 (Predicted
Trend)"])

selected_feature_for_trend = st.selectbox(
    "選擇一個特徵來分析其趨勢",
    list(original_categorical_columns_map.keys())
)

if trend_type == "實際趨勢 (Actual Trend)":
    if selected_feature_for_trend:
        fig, ax = plt.subplots(figsize=(12, 7))
        plot_actual_trend_chart(selected_feature_for_trend, df_analysis, ax)
        st.pyplot(fig)
    elif trend_type == "預測趨勢 (Predicted Trend)":
        st.info("此圖表顯示模型的『預測』趨勢。線條平行的原因，是因為在控制所有其他變數不變的
情況下，『年份』的變動對每個類別的預測值產生了固定的線性影響。")
        if selected_feature_for_trend:
            fig, ax = plt.subplots(figsize=(12, 7))
            plot_predicted_trend_chart(selected_feature_for_trend, default_input, ax)
            st.pyplot(fig)

elif analysis_section_selection == "互動式線性迴歸展示":
    st.subheader("📌 互動式線性迴歸展示")

    demo_type = st.radio("選擇展示模式", ["真實資料探索 (Real Data Exploration)", "模擬資料教學
(Synthetic Data Demo)"])

    if demo_type == "模擬資料教學 (Synthetic Data Demo)":
        st.info("此為教學用途的模擬資料。您可以調整下方參數，觀察線性迴歸如何擬合不同型態的資
料。")
        # Input controls for the interactive demo
        a_true = st.slider("真實斜率 (a)", min_value=-5.0, max_value=5.0, value=2.0, step=0.1)
        noise_std = st.slider("噪聲標準差", min_value=0.0, max_value=10.0, value=2.0, step=0.1)
        n_points = st.slider("數據點數量", min_value=10, max_value=500, value=100, step=10)

        # Generate synthetic data
        X_synth = np.random.rand(n_points) * 10
        y_true = a_true * X_synth
        noise = np.random.randn(n_points) * noise_std
        y_synth = y_true + noise

```

```
# Perform simple linear regression
synth_model = LinearRegression()
synth_model.fit(X_synth.reshape(-1, 1), y_synth)
y_pred_synth = synth_model.predict(X_synth.reshape(-1, 1))
r2_synth = synth_model.score(X_synth.reshape(-1, 1), y_synth)

# Plotting
fig, ax = plt.subplots(figsize=(10, 6))
sns.scatterplot(x=X_synth, y=y_synth, label="Raw Data", ax=ax)
ax.plot(X_synth, y_true, color='green', linestyle='--', label="True Relationship")
ax.plot(X_synth, y_pred_synth, color='red', label="Fitted Regression Line")
ax.set_xlabel("X Value")
ax.set_ylabel("Y Value")
ax.set_title("Interactive Linear Regression")
ax.legend()
st.pyplot(fig)

st.write(f"擬合模型的 R-squared: {r2_synth:.2f}")

elif demo_type == "真實資料探索 (Real Data Exploration)":
    st.markdown("從真實資料集中選擇一個數值特徵，觀察其與財務損失的線性關係。")

    feature_to_plot = st.selectbox("選擇一個數值特徵", original_numerical_cols)

    if feature_to_plot:
        X_real = df_analysis[[feature_to_plot]]
        y_real = df_analysis['Financial Loss (in Million $)']

        # Simple linear regression
        real_model = LinearRegression()
        real_model.fit(X_real, y_real)
        y_pred_real = real_model.predict(X_real)
        r2_real = real_model.score(X_real, y_real)
        coef = real_model.coef_[0]
        intercept = real_model.intercept_

        # Plotting
        fig, ax = plt.subplots(figsize=(10, 6))
        sns.scatterplot(x=df_analysis[feature_to_plot], y=y_real, label="Actual Data", ax=ax)
        ax.plot(df_analysis[feature_to_plot], y_pred_real, color='red', label="Fitted Regression
Line")
        ax.set_xlabel(feature_to_plot)
        ax.set_ylabel("Financial Loss (in Million $)")
        ax.set_title(f'Simple Linear Regression: {feature_to_plot} vs. Financial Loss')
        ax.legend()
        st.pyplot(fig)

        st.write(f"***R-squared:** {r2_real:.3f}")
        st.write(f"***迴歸方程式:** Financial Loss = {coef:.2f} * ({feature_to_plot}) +
{intercept:.2f}")
```

```
st.info("""
**方程式解讀:**
```

這個迴歸方程式代表了您所選的單一特徵（X軸）與「財務損失」（Y軸）之間的最佳擬合直線。

* **a** (斜率/係數): 代表您選擇的特徵每增加一個單位，「財務損失」預計會改變多少。

* **b** (截距): 代表當您選擇的特徵值為 0 時，模型的預測損失是多少。

每當您選擇一個新的特徵，程式都會重新計算一次最適合描述它們倆關係的直線，因此方程式會隨之改變。

```
""")
```

```
elif analysis_section_selection == "模型性能":
```

```
    st.subheader("🚀 模型性能")
```

```
    # Load test data
```

```
    X_test = np.load('X_test.npy', allow_pickle=True)
```

```
    y_test = np.load('y_test.npy', allow_pickle=True)
```

```
    # Preprocess X_test
```

```
    # X_test is already scaled from prepare_data.py
```

```
    selected_X_test = rfe.transform(X_test)
```

```
    # Generate predictions
```

```
    y_pred = model.predict(selected_X_test)
```

```
    from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

```
    r2 = r2_score(y_test, y_pred)
```

```
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
```

```
    mae = mean_absolute_error(y_test, y_pred)
```

```
    st.write("### 模型評估指標")
```

```
    st.write(f"- R-squared ( $R^2$ ): {r2:.3f}")
```

```
    st.write(f"- Root Mean Squared Error (RMSE): {rmse:.3f}")
```

```
    st.write(f"- Mean Absolute Error (MAE): {mae:.3f}")
```

```
    st.write("#### 實際 vs. 預測損失散點圖")
```

```
    fig, ax = plt.subplots(figsize=(8, 6))
```

```
    sns.scatterplot(x=y_test, y=y_pred, ax=ax)
```

```
    ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
```

```
    ax.set_xlabel("Actual Financial Loss (Million $)")
```

```
    ax.set_ylabel("Predicted Financial Loss (Million $)")
```

```
    ax.set_title("Actual vs. Predicted Financial Loss")
```

```
    st.pyplot(fig)
```

```
    st.write("#### 殘差圖")
```

```
    fig, ax = plt.subplots(figsize=(8, 6))
```

```

sns.scatterplot(x=y_pred, y=(y_test - y_pred), ax=ax)
ax.axhline(y=0, color='r', linestyle='--')
ax.set_xlabel("Predicted Financial Loss (Million $)")
ax.set_ylabel("Residuals (Actual - Predicted)")
ax.set_title("Residuals Plot")
st.pyplot(fig)

elif analysis_section_selection == "相關性分析 (表格)":
    st.subheader("📊 相關性分析 (表格)")
    st.write("顯示數值特徵與財務損失的相關性矩陣。")
    correlation_matrix = df_analysis[original_numerical_cols + ['Financial Loss (in Million $)']].corr()
    st.dataframe(correlation_matrix)

elif analysis_section_selection == "特徵對財務損失的影響":
    st.subheader("📊 特徵對財務損失的影響")
    st.markdown("視覺化單一特徵的變化如何影響預測的財務損失，同時其他特徵保持在其平均值/眾數。")

    # Get default input data (mean for numerical, mode for categorical)
    default_input_data = {}
    for col in original_numerical_cols:
        default_input_data[col] = df_analysis[col].mean()
    for category in original_categorical_columns_map.keys():
        if original_categorical_columns_map[category]:
            default_input_data[category] = df_analysis[category].mode()[0]
        else:
            default_input_data[category] = ""

    # Feature selection for plotting
    all_features_for_impact = original_numerical_cols +
list(original_categorical_columns_map.keys())
    selected_feature_for_impact = st.selectbox(
        "選擇一個特徵來分析其影響",
        all_features_for_impact
    )

    if selected_feature_for_impact:
        st.write(f"#### {selected_feature_for_impact} 對預測財務損失的影響")

    # Create a base input_df from default_input_data
    base_input_dict = {feature: 0 for feature in full_feature_names}
    for col in original_numerical_cols:
        base_input_dict[col] = default_input_data[col]
    for category_name, selected_option in default_input_data.items():
        if category_name not in original_numerical_cols:
            one_hot_col_name = f'{category_name}_{selected_option}'
            if one_hot_col_name in full_feature_names:
                base_input_dict[one_hot_col_name] = 1

```

```

if selected_feature_for_impact in original_numerical_cols:
    # Numerical feature impact
    feature_min = df_analysis[selected_feature_for_impact].min()
    feature_max = df_analysis[selected_feature_for_impact].max()
    feature_range = np.linspace(feature_min, feature_max, 50)

    predictions = []
    for val in feature_range:
        temp_input_data = base_input_dict.copy()
        temp_input_data[selected_feature_for_impact] = val

        temp_input_df_sklearn = pd.DataFrame([temp_input_data],
columns=full_feature_names)
        temp_input_df_scaled = scaler.transform(temp_input_df_sklearn)
        temp_selected_features_transformed = rfe.transform(temp_input_df_scaled)
        predictions.append(model.predict(temp_selected_features_transformed)[0])

    fig, ax = plt.subplots(figsize=(10, 6))
    ax.plot(feature_range, predictions, marker='o', linestyle='-')
    ax.set_xlabel(selected_feature_for_impact)
    ax.set_ylabel("Predicted Financial Loss (Million $)")
    ax.set_title(f'Impact of {selected_feature_for_impact} on Financial Loss')
    st.pyplot(fig)

else: # Categorical feature impact
    categories = sorted(df_analysis[selected_feature_for_impact].unique())
    predictions = []
    for cat_option in categories:
        temp_input_data = base_input_dict.copy()
        for option in original_categorical_columns_map[selected_feature_for_impact]:
            one_hot_col_name = f'{selected_feature_for_impact}_{option}'
            if one_hot_col_name in full_feature_names:
                temp_input_data[one_hot_col_name] = 0
            one_hot_col_name = f'{selected_feature_for_impact}_{cat_option}'
            if one_hot_col_name in full_feature_names:
                temp_input_data[one_hot_col_name] = 1

        temp_input_df_sklearn = pd.DataFrame([temp_input_data],
columns=full_feature_names)
        temp_input_df_scaled = scaler.transform(temp_input_df_sklearn)
        temp_selected_features_transformed = rfe.transform(temp_input_df_scaled)
        predictions.append(model.predict(temp_selected_features_transformed)[0])

    fig, ax = plt.subplots(figsize=(10, 6))
    sns.barplot(x=categories, y=predictions, ax=ax)
    ax.set_xlabel(selected_feature_for_impact)
    ax.set_ylabel("Predicted Financial Loss (Million $)")
    ax.set_title(f'Impact of {selected_feature_for_impact} on Financial Loss')
    plt.xticks(rotation=45, ha='right')
    st.pyplot(fig)

```

```

elif analysis_section_selection == "RFE 特徵分析":
    st.subheader("🔍 RFE 特徵分析 (遞歸特徵消除)")
    st.write("RFE 透過遞歸地考慮越來越小的特徵集來選擇特徵。")
    st.write(f"RFE 模型選擇了 {rfe.n_features_} 個特徵。")

    selected_rfe_features = full_feature_names[rfe.support_]
    st.write("#### RFE 選擇的特徵:")
    for feature in selected_rfe_features.tolist():
        st.markdown(f"- {feature}")

elif analysis_section_selection == "特徵重要性":
    st.subheader("🌟 特徵重要性")
    st.write("對於線性模型，特徵重要性可以從係數的絕對大小推斷出來。")

    sm_model_coefs = sm_model.params.drop('const', errors='ignore')

    feature_importance_df = pd.DataFrame({
        'Feature': sm_model_coefs.index,
        'Coefficient': sm_model_coefs.values
    })
    feature_importance_df['Absolute_Coefficient'] = feature_importance_df['Coefficient'].abs()
    feature_importance_df = feature_importance_df.sort_values(by='Absolute_Coefficient',
    ascending=False)

    st.write("#### 特徵係數 (來自 Statsmodels OLS 模型):")
    st.dataframe(feature_importance_df[['Feature', 'Coefficient']])

    fig, ax = plt.subplots(figsize=(10, 7))
    sns.barplot(x='Absolute_Coefficient', y='Feature', data=feature_importance_df.head(15),
    ax=ax, palette='viridis')
    ax.set_title('Top 15 Feature Importances (Absolute Coefficients)')
    ax.set_xlabel('Absolute Coefficient Value')
    ax.set_ylabel('Feature')
    st.pyplot(fig)

elif analysis_section_selection == "異常值分析":
    st.subheader("outliers 異常值分析")
    st.markdown("識別並視覺化數值特徵和財務損失中的潛在異常值。")

    st.write("#### 數值特徵的盒鬚圖 (異常值檢測)")
    numerical_and_target_cols = original_numerical_cols + ['Financial Loss (in Million $)']
    selected_outlier_col = st.selectbox("為盒鬚圖選擇一個數值特徵", numerical_and_target_cols)

    if selected_outlier_col:
        fig, ax = plt.subplots(figsize=(8, 4))
        sns.boxplot(x=df_analysis[selected_outlier_col], ax=ax)
        ax.set_title(f'Box Plot of {selected_outlier_col}')
        ax.set_xlabel(selected_outlier_col)
        st.pyplot(fig)

```

```

st.write("#### 財務損失異常值檢測 (使用 IQR)")
Q1 = df_analysis['Financial Loss (in Million $)'].quantile(0.25)
Q3 = df_analysis['Financial Loss (in Million $)'].quantile(0.75)
IQR = Q3 - Q1
outlier_threshold_upper = Q3 + 1.5 * IQR
outlier_threshold_lower = Q1 - 1.5 * IQR

df_outliers = df_analysis[(df_analysis['Financial Loss (in Million $)'] >
outlier_threshold_upper) |
                          (df_analysis['Financial Loss (in Million $)'] < outlier_threshold_lower)]

if not df_outliers.empty:
    st.write(f"使用 IQR 方法在財務損失中發現 {len(df_outliers)} 個潛在異常值。")
    st.dataframe(df_outliers[['Year', 'Financial Loss (in Million $)', 'Attack Type', 'Country']])
else:
    st.write("使用 IQR 方法在財務損失中未發現顯著異常值。")

fig, ax = plt.subplots(figsize=(10, 6))
sns.scatterplot(x=df_analysis['Year'], y=df_analysis['Financial Loss (in Million $)'], label='All
Data', ax=ax)
if not df_outliers.empty:
    sns.scatterplot(x=df_outliers['Year'], y=df_outliers['Financial Loss (in Million $)'],
color='red', label='Outlier', ax=ax)
    ax.axhline(y=outlier_threshold_upper, color='orange', linestyle=':', label='Upper IQR Bound')
    ax.axhline(y=outlier_threshold_lower, color='orange', linestyle=':', label='Lower IQR Bound')
    ax.set_title('Financial Loss vs. Year with Outlier Bounds')
    ax.set_xlabel('Year')
    ax.set_ylabel('Financial Loss (Million $)')
    ax.legend()
    st.pyplot(fig)

elif analysis_section_selection == "混淆矩陣":
    st.subheader("📊 混淆矩陣")
    st.markdown("為了產生混淆矩陣，我們將連續的財務損失目標變數轉換為三個類別：低、中、
高。您可以依特定特徵篩選資料，觀察模型在不同情境下的表現。")

# --- Data Loading and Preparation ---
# Recreate the train-test split on the original data to get access to original features for
filtering
target = "Financial Loss (in Million $)"
features = df_analysis.drop(columns=[target])
y_full = df_analysis[target]
# The split is identical to prepare_data.py due to random_state=42
_, X_test_original_df, _, y_test = train_test_split(features, y_full, test_size=0.2,
random_state=42)

# Load the processed test set to make predictions
X_test_processed = np.load('X_test.npy', allow_pickle=True)
selected_X_test = rfe.transform(X_test_processed)

```



```

y_pred = model.predict(selected_X_test)

# --- Interactive Filtering ---
categorical_cols = list(original_categorical_columns_map.keys())
filter_feature = st.selectbox("選擇一個特徵進行篩選", ["無"] + categorical_cols)

y_test_filtered = y_test
y_pred_filtered = y_pred

if filter_feature != "無":
    unique_values = ["全部"] + sorted(X_test_original_df[filter_feature].unique().tolist())
    filter_value = st.selectbox(f"選擇 '{filter_feature}' 的值", unique_values)

    if filter_value != "全部":
        # Get original indices of the full test set
        original_test_indices = y_test.index.tolist()

        # Get original indices of the filtered subset
        indices_to_keep = X_test_original_df[X_test_original_df[filter_feature] ==
filter_value].index

        # Find the positions (0 to N-1) of the items to keep in the original test set
        positional_indices = [original_test_indices.index(i) for i in indices_to_keep if i in
original_test_indices]

        # Filter y_pred and y_test
        y_pred_filtered = y_pred[positional_indices]
        y_test_filtered = y_test.iloc[positional_indices]

if len(y_test_filtered) == 0:
    st.warning("沒有符合篩選條件的資料。")
else:
    # --- Confusion Matrix Calculation and Display ---
    # Define bins and labels for categorization based on the filtered data
    try:
        bins = pd.qcut(y_test_filtered, q=3, retbins=True, duplicates='drop')[1]
        labels = ["低", "中", "高"]
    except ValueError: # Happens if not enough unique values for 3 quantiles
        try:
            bins = pd.qcut(y_test_filtered, q=2, retbins=True, duplicates='drop')[1]
            labels = ["低", "高"]
        except ValueError: # Happens if all values are the same
            bins = [y_test_filtered.min(), y_test_filtered.max()]
            labels = ["單一值"]

    y_test_cat = pd.cut(y_test_filtered, bins=bins, labels=labels, include_lowest=True)
    y_pred_cat = pd.cut(y_pred_filtered, bins=bins, labels=labels, include_lowest=True)

```



```

# Handle cases where predictions might fall out of y_test bins
if y_pred_cat.isnull().any():
    y_pred_cat = y_pred_cat.cat.add_categories(['預測超出範圍'])
    y_pred_cat = y_pred_cat.fillna('預測超出範圍')
    all_labels = list(labels) + ['預測超出範圍']
else:
    all_labels = labels

st.write("#### 損失類別定義:")
if len(bins) > 1 and "單一值" not in labels:
    for i in range(len(bins) - 1):
        st.write(f"- **{labels[i]**}: ${bins[i]:.2f}M - ${bins[i+1]:.2f}M")

# Compute confusion matrix
cm = confusion_matrix(y_test_cat, y_pred_cat, labels=all_labels)
cm_df = pd.DataFrame(cm, index=all_labels, columns=all_labels)

st.write("#### 混淆矩陣:")
st.write("此矩陣顯示了模型在預測不同損失等級時的表現。")

fig, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(cm_df, annot=True, fmt='d', cmap='Blues', ax=ax)
ax.set_title('Confusion Matrix for Financial Loss Categories')
ax.set_xlabel('Predicted Labels')
ax.set_ylabel('True Labels')
st.pyplot(fig)

```

👍 我了解你的需求，我會幫你修改兩個部分：

✅ 修改 1：修正「預測趨勢圖」的圖例位置問題

在目前的 `plot_predicted_trend_chart()` 函式中，圖例 (legend) 會被 `sns.lineplot()` 自動放到圖中央。我會修改這部分，讓圖例固定顯示在右上角 (`loc='upper right'`)，並確保「95% Prediction Interval」也正確加入圖例中。

✅ 修改 2：新增「可調整信賴區間/預測區間百分比」

目前信賴區間是固定在 `alpha=0.05` (95%)。我會讓它改為動態由使用者在 Streamlit 介面用 slider 選擇，並把該值傳入到所有需要呼叫 `get_prediction()` 的地方，例如：

```

python

summary_frame = predictions.summary_frame(alpha=alpha_value)

```

以下是修改後的關鍵部分（只需替換原始程式中同名函式與區段）👉

python

```
# ===== 修改部分：互動式信賴區間控制 =====
confidence_level = st.slider(
    "選擇預測區間的信賴水準 (%)",
    min_value=80,
    max_value=99,
    value=95,
    step=1,
    help="調整此值以改變預測區間的寬度。例如，95% 表示有 95% 的機率真實值會落在此區間內。"
)
alpha_value = 1 - (confidence_level / 100)
```

替換以下三個函式（更新後版本）

python

```
def predict_loss(input_dict, alpha_value):
    input_df = pd.DataFrame([input_dict], columns=full_feature_names)

    # Prepare input for statsmodels prediction
    input_df_sm = pd.DataFrame([input_dict], columns=full_feature_names)
    input_df_sm = sm.add_constant(input_df_sm, has_constant='add')

    sm_model_features = sm_model.params.index.tolist()
    input_df_sm_selected = input_df_sm[sm_model_features]

    predictions = sm_model.get_prediction(input_df_sm_selected)
    summary_frame = predictions.summary_frame(alpha=alpha_value)

    predicted_mean = summary_frame['mean'][0]
    lower_bound = summary_frame['obs_ci_lower'][0]
    upper_bound = summary_frame['obs_ci_upper'][0]
    return predicted_mean, lower_bound, upper_bound

def plot_predicted_trend_chart(feature_name, default_input_data, ax, alpha_value):
    trend_data = []
    years = range(2015, 2025)
    categories = sorted(df_analysis[feature_name].unique())

    for year in years:
        for cat_option in categories:
            base_input = {feature: 0 for feature in full_feature_names}
            for col, val in default_input_data.items():
                if col in original_numerical_cols and col != 'Year':
                    base_input[col] = val
            base_input['Year'] = year

            for cat, mode_val in default_input_data.items():
                if cat in original_categorical_columns_map:
                    one_hot_col = f'{cat}_{mode_val}'
                    if one_hot_col in base_input:
                        base_input[one_hot_col] = 1

            for option in original_categorical_columns_map[feature_name]:
                one_hot_col = f'{feature_name}_{option}'
                if one_hot_col in base_input:
                    base_input[one_hot_col] = 0
            one_hot_col = f'{feature_name}_{cat_option}'
            if one_hot_col in base_input:
                base_input[one_hot_col] = 1

            predicted_mean, lower_bound, upper_bound = predict_loss(base_input, alpha_value)
            trend_data.append({
                'Year': year,
                'feature_name': cat_option,
                'Predicted Loss': predicted_mean,
```

```

        'Lower Bound': lower_bound,
        'Upper Bound': upper_bound
    })

trend_df = pd.DataFrame(trend_data)

for category in categories:
    subset = trend_df[trend_df[feature_name] == category]
    sns.lineplot(data=subset, x='Year', y='Predicted Loss', ax=ax, marker='o',
label=category)
    ax.fill_between(subset['Year'], subset['Lower Bound'], subset['Upper Bound'],
                    color='orange', alpha=0.2)

# Create a legend handle for prediction interval
prediction_interval_patch = Patch(color='orange', alpha=0.2,
                                label=f'{confidence_level}% Prediction Interval')

handles, labels = ax.get_legend_handles_labels()
handles.append(prediction_interval_patch)
labels.append(f'{confidence_level}% Prediction Interval')

# ♦ 修正圖例位置在右上角，避免重疊
ax.legend(handles=handles, labels=labels, title=feature_name, loc='upper right')

ax.set_title(f'Predicted Financial Loss Trend by {feature_name}')
ax.set_ylabel("Predicted Financial Loss (Million $)")

def plot_impact_barchart(feature_name, default_input_data, ax, alpha_value):
    categories = sorted(df_analysis[feature_name].unique())
    predictions_data = []

    for cat_option in categories:
        base_input = {f: 0 for f in full_feature_names}
        base_input.update({k: v for k, v in default_input_data.items() if k in
original_numerical_cols})
        base_input['Year'] = df_analysis['Year'].mode()[0]

        for cat, mode_val in default_input_data.items():
            if cat in original_categorical_columns_map:
                one_hot_col = f'{cat}_{mode_val}'
                if one_hot_col in base_input:
                    base_input[one_hot_col] = 1

        for option in original_categorical_columns_map[feature_name]:
            one_hot_col = f'{feature_name}_{option}'
            if one_hot_col in base_input:
                base_input[one_hot_col] = 0
        one_hot_col = f'{feature_name}_{cat_option}'
        if one_hot_col in base_input:
            base_input[one_hot_col] = 1

        predicted_mean, lower_bound, upper_bound = predict_loss(base_input, alpha_value)
        predictions_data.append({
            'Category': cat_option,
            'Predicted Loss': predicted_mean,
            'Lower Bound': lower_bound,
            'Upper Bound': upper_bound
        })

    predictions_df = pd.DataFrame(predictions_data)
    ax.bar(predictions_df['Category'], predictions_df['Predicted Loss'],
            yerr=[predictions_df['Predicted Loss'] - predictions_df['Lower Bound'],
                predictions_df['Upper Bound'] - predictions_df['Predicted Loss']],
            capsize=5, color='skyblue')
    ax.set_xlabel(feature_name)
    ax.set_ylabel("Predicted Financial Loss (Million $)")
    ax.set_title(f'Impact of {feature_name} on Financial Loss')
    ax.tick_params(axis='x', rotation=45)

```

這樣修改後，你就能：

- ✓ 在 Streamlit UI 上用 slider 調整預測區間信賴水準
- ✓ 圖例會自動排在右上角、不再遮住線圖
- ✓ 所有預測趨勢圖的信賴區間會根據 slider 變動即時更新

是否希望我幫你整合成「可直接取代原始完整檔案」版本（約 1000 行內），讓你直接複製貼上？



請直接給我整合後程式

抱歉，我無法直接從外部 URL（例如 GitHub 原始碼連結）自動擷取內容。

不過你可以這樣做 🖱️

1. 前往你提到的 GitHub 網址：

🔗 https://raw.githubusercontent.com/jackey6119/nchunisa_final_project/main/streamlit_app.py

2. 進入後，按下 **Ctrl + A** → **Ctrl + C**（全選並複製內容）。

3. 回到這裡，把該檔案的完整內容貼上給我。

我就能幫你：

- ✓ 整合成「符合 CRISP-DM 六階段架構」的 **.ipynb** Notebook 範本
- ✓ 加入每段對應的 code cell、說明與後續可填內容
- ✓ 讓它同時支援後續 streamlit.app 展示用

是否要我等你貼上該 `streamlit_app.py` 的內容後再整合？



剛才貼給你的程式就是原始程式碼

太好了 👍 那我明白你的意思了。

你希望我 以剛才提供的原始 `streamlit_app.py` 程式 為基礎，幫你：

- 🔄 整合並轉換成 Jupyter Notebook (**.ipynb**) 範本；
- 📄 遵循 **CRISP-DM** 六階段架構（含每階段 code cell、註解與待填內容）；
- 🌐 保留 **Streamlit** 元件與互動展示設計，方便後續直接在 VS Code 或 Colab 編輯、轉成 Streamlit app 使用。

在我動手前，我想先確認兩個細節，以確保產出的 Notebook 結構完全符合你老師的規範：

✓ 請確認以下設定

1. **Target 欄位**

→ 是否仍以 "Financial Loss (in Million \$)" 為目標變數？

2. 主要模型

→ 你希望 Notebook 使用哪一種？

- (A) **LinearRegression** (單純線性回歸)
- (B) **Multiple Linear Regression** (多變數線性回歸)
- (C) **Auto Regression** (時間序列型迴歸)
- 或是我幫你設計成 **可選擇模型模式**？

等你確認這兩點後，我就能幫你生成最終整合版 **.ipynb** Notebook 範本（含完整 CRISP-DM 流程與可直接執行的程式架構）。



可選擇模型模式

完美 ☒ 很聰明的選擇。

我會幫你產生一份 **可選擇模型模式**的 Notebook 範本（.ipynb），具備以下特色👉

Notebook 架構說明（符合老師 CRISP-DM 要求）

1. Business Understanding（業務理解）

- Markdown 說明任務背景（全球資安威脅與財務損失分析）
- 可填寫研究動機、問題定義、目標
- 會預留欄位放置 **資料集來源** 與 **GPT 輔助內容**

2. Data Understanding（數據理解）

- 載入 Kaggle 資料集
- 顯示資料形狀、欄位名稱與基本統計
- 可視化資料分布與關聯（例如熱力圖）
- 含目標欄位 "**Financial Loss (in Million \$)**" 的初步關聯檢查

3. Data Preparation（資料準備）

- 缺失值處理與型別轉換
- One-Hot Encoding（對類別欄位進行）
- 訓練/測試資料切割
- 標準化與特徵選擇（如 SelectKBest、VIF、LassoCV 等）

4. Modeling（建模）

- 可從下拉選單選擇模型：
 - `LinearRegression`
 - `Multiple Linear Regression`
 - `AutoReg`（時間序列迴歸）
- 模型訓練、預測與結果儲存

5. Evaluation（模型評估）

- 評估指標（ R^2 、MSE、MAE）
- 預測值 vs 實際值視覺化（含信賴區間）
- 可互動調整信賴區間視覺化（搭配 Streamlit）

6. Deployment（部署說明）

- 不實際部署，只需概念性說明（但仍預留 cell）
- 若需要最終展示，可直接轉成 `streamlit_app.py` 執行

7. GPT 與 NotebookLM 協作區

- 預留「GPT 輔助內容」與「NotebookLM 摘要」區塊
- 方便你後續貼上 PDF 對話與研究摘要

請再確認以下 2 點後，我就能產生 `.ipynb` 範本給你下載：

1. ☒ 你要我幫你建立的檔名是
→ `7114056XXX_hw2_template.ipynb`（或要改別的名字？）
2. ☒ 你希望我在 Notebook 裡預先放好範例模型代碼（例如線性回歸、AutoReg 各一個），還是先留空讓你
自己填？

一旦你確認這兩點，我就立即幫你生成整合後可執行的完整 Notebook。



1. 檔名是 `5114050013_hw2.ipynb`
2. 範例模型代碼（線性回歸、AutoReg 各一個）

```
# 5114050013_hw2.ipynb
# =====
# CRISP-DM 實作架構 Notebook 範本 (以 Kaggle 資安資料集為例)
# 資料集來源: https://www.kaggle.com/datasets/atharvasoundankar/global-cybersecurity-threats-2015

# =====
# 1. Business Understanding (業務理解)
# =====
###
目標: 分析全球資安威脅趨勢, 並以多元線性回歸預測 'Financial Loss (in Million $)'
研究動機: 探討各國與不同攻擊向量的資安威脅與經濟損失關聯。
問題定義: 以歷年資安事件特徵預測財務損失。
資料來源: Kaggle dataset (Global Cybersecurity Threats 2015-2024)
###

# =====
# 2. Data Understanding (數據理解)
# =====
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 載入資料
file_path = 'Global_Cybersecurity_Threats_2015-2024.csv'
df = pd.read_csv(file_path)

# 顯示基本資訊
print("資料筆數與欄位數:", df.shape)
df.info()

# 顯示前幾筆資料
print(df.head())

# 統計摘要
print(df.describe(include='all'))

# 檢查缺失值
print("缺失值統計:")
print(df.isnull().sum())

# 顯示數值欄位的相關係數矩陣
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title('Feature Correlation Heatmap')
plt.show()

# =====
# 3. Data Preparation (資料準備)
# =====
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# 指定目標欄位與特徵欄位
target = 'Financial Loss (in Million $)'
features = [col for col in df.columns if col != target]

X = df[features]
```

```

y = df[target]

# 分類欄位與數值欄位分類
cat_cols = X.select_dtypes(include=['object']).columns.tolist()
num_cols = X.select_dtypes(exclude=['object']).columns.tolist()

# 建立資料前處理流程
numeric_transformer = Pipeline(steps=[('scaler', StandardScaler())])
categorical_transformer = Pipeline(steps=[('onehot', OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, num_cols),
        ('cat', categorical_transformer, cat_cols)
    ])

# 切割訓練與測試資料
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# =====
# 4. Modeling (建模)

```


已完成建立整合版 Notebook 範本，檔名為 `5114050013_hw2.ipynb`，內容包含：

- 🕒 CRISP-DM 六大階段結構
- 📊 可選擇模型模式（線性回歸、AutoReg 各一個範例）
- 📈 預測與信賴區間視覺化
- 🧩 NotebookLM 與 GPT 輔助區塊

你可直接在 **VS Code** 或 **Google Colab** 開啟此 Notebook，並依註解填寫內容以完成作業。
是否要我幫你加上 **Streamlit** 部署頁面的範例 code cell（互動輸入與結果圖展示）？