

**HW3:****Source Reference**

This project builds upon patterns and datasets related to the Spam Email problem from Chapter 3 of the Packt repository below. We used it to expand the preprocessing steps and add richer visualization work (step outputs, metrics, and CLI/Streamlit views).

<https://github.com/PacktPublishing/Hands-On-Artificial-Intelligence-for-Cybersecurity.git>

**Using openspec and AI coding CLI to finish this project**

Requirements :

1. need a github  
<https://github.com/huanchen1107/2025ML-spamEmail>
2. need a Demo site  
<https://2025spamemail.streamlit.app/>

-----

**[my GitHub websit]**

[https://github.com/candice-wu/Security\\_HW\\_03\\_Classification\\_Spam-Email](https://github.com/candice-wu/Security_HW_03_Classification_Spam-Email)

**[my Streamlit Demosite]**

<https://hw03-classification-spam-email.streamlit.app/>

**[執行模式，採二階段開發]**

Phase 1：請 Gemini 根據 openspec 專案架構幫我產生初版

Phase 2：請 Gemini 升級我的專案

註 1：1st 是 openspec 搭配 GitHub Copilot，但 GitHub Copilot 不太聰明，執行過程不太像老師上課所教，及 AI 超元域的網路教學，放棄使用。

註 2：openspec 內建預設的 AI coding CLI 雖沒有 Gemini CLI，但我預選 GitHub Copilot 後，嘗試在 Termina 呼叫 Gemini，意外地發現 Gemini 也能依照 openspec 架構生成相關文件（如同老師上課所教，及 AI 超元域的網路教學），介紹如下：

## [Phase 1：請 Gemini 根據 openspec 專案架構幫我產生初版]

### 1. 啟動 openspec 與 Gemini CLI 並請求產生專案提案的企劃案文件

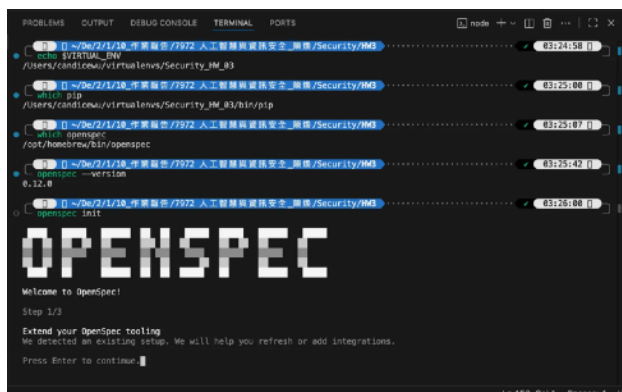


圖 1: openspec init

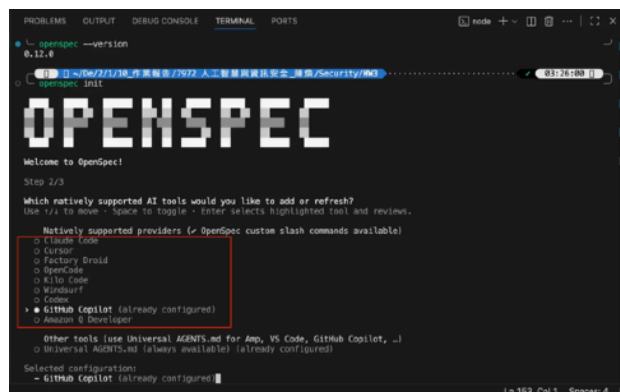


圖 2: Select GitHub Copilot

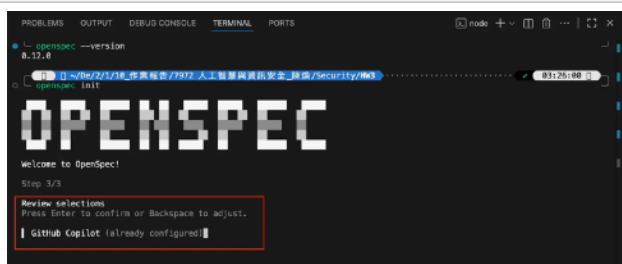


圖 3: Double confirm

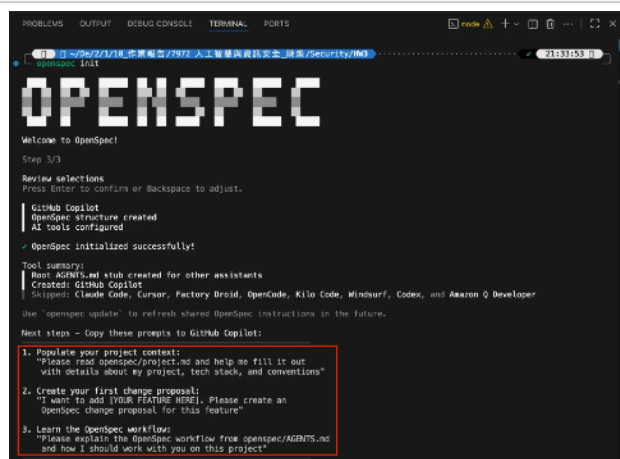


圖 4: Openspec 要求轉貼三個指令到 AI coding CLI

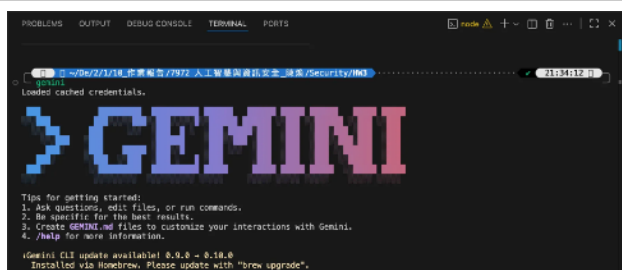


圖 5: 啟動 (呼叫) Gemini CLI

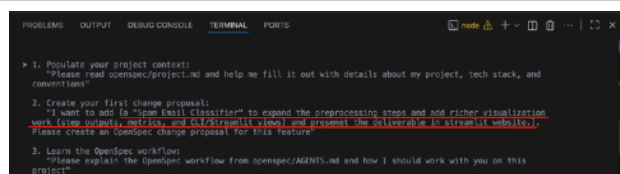


圖 6: 將 openspec 第二個指令的中括號 [YOUR FEATURE HERE] 改成我的專案內容 [a "Spam Email Classifier" to expand the preprocessing steps and add richer visualization work (step outputs, metrics, and CLI/Streamlit views) and present the deliverable in streamlit website.]。

如此，Gemini 更清楚我的專案需求，並依 openspec 架構自動產生專案提案文件：proposal.md、design.md、tasks.md、specs/spam-classifier

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ I've reviewed openspec/project.md. To help you fill it out, please provide the following details for your project:

• Purpose and Goals: What is the main objective of this project?
• Tech Stack: What are the primary technologies, languages, and frameworks you are using?
• Code Style: Do you have specific code style preferences, formatting rules, or naming conventions?
• Architecture Patterns: What architectural decisions or patterns are you following?
• Testing Strategy: How do you approach testing (e.g., unit, integration, end-to-end)?
• Git Workflow: What branching strategy and commit conventions do you use?
• Domain Context: Is there any domain-specific knowledge crucial for understanding this project?
• Important Constraints: Are there any significant technical, business, or regulatory constraints?
• External Dependencies: What key external services, APIs, or systems does your project rely on?
```

圖 7: Gemini CLI 列出 9 項問題要我澄清並回覆，以便它能更清楚掌握我的專案要求

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> • Purpose and Goals:
1. 一個全新的專案，用 Python 開發 spam_email 內容類別是否為廣告垃圾郵件且透過 Streamlit.app 進行網頁互動
2. datasets 來源是 https://github.com/PacktPublishing/Hands-On-Artificial-Intelligence-for-Cybersecurity/blob/master/Chapter03/datasets/spam_email_no_header.csv
• Tech Stack:
Python 程式語言開發一個具有 Classification 機器學習的演算法，that means expand the preprocessing steps and add richer visualization work (step outputs, metrics, and CLI/Streamlit views).
• Code Style:
1. specific code style preferences: Python and related packages applied in this project.
2. formatting rules: Simple and easy to understand
3. 主程式的命名為 5114050013_hw3.py
• Architecture Patterns:
利用 CRISP-DM 架構來設計如何完成這項專案，並用 Markdown 記法產生一個 README.md 檔，並利用 CRISP-DM 架構來設計如何完成這項專案，並用 Markdown 記法產生一個 README.md 檔
• Testing Strategy:
1. 先單元測試，再整合測試
2. end-to-end testing 需要再另外提出請求
• Git Workflow:
1. 基於分支開發，要提交新文件與程式碼上傳到指定於 GitHub (Security_HM_03_Classification_Spam-Email)
2. 遵循 https://github.com/candice-w/Security_HM_03_Classification_Spam-Email.git
3. 遵循嚴格先提交 requirements.txt 再提交 gitignore 檔案
4. 在 .gitignore 中指定 *.pyc 和 *.pyo 檔案
5. 在 README.md 內容中描述如何 CRISP-DM 架構來設計如何完成這項專案，並上方要標出 Streamlit demo website 連結並提供專案成果
• Domain Context:
符合一般大數據理解的範疇，且能一窺機器學習操作 Streamlit 網頁互動的各項功能
• Important Constraints:
None
• External Dependencies:
1. GitHub
2. Streamlit
```

圖 8: 我回覆給 Gemini CLI 的內容 (專案規範要求)

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ Great, thank you for providing these details! I will now populate openspec/project.md with the information you've given me.

Here's the first replacement for the "Purpose" section:

Edit openspec/project.md: # Project Context... ==> # Project Context...
1 # Project Context
2
3 # Purpose
4 - [Describe your project's purpose and goals]
5 + 1. 一個全新的專案，用 Python 開發 spam_email 內容類別是否為廣告垃圾郵件且透過 Streamlit.app
6
7 + 2. datasets
8 來源是 https://github.com/PacktPublishing/Hands-On-Artificial-Intelligence-for-Cybersecurity/blob/master/Chapter03/datasets/spam_email_no_header.csv
9
10 # Tech Stack
11 - [List your primary technologies]
12
13 # The "Purpose" section is updated. Now, I'll update the "Tech Stack" section.

Edit openspec/project.md: # Project Context... ==> # Project Context...
1 # Project Context
2
3 # Purpose
4 - [Describe your project's purpose and goals]
5 + 1. 一個全新的專案，用 Python 開發 spam_email 內容類別是否為廣告垃圾郵件且透過 Streamlit.app
6
7 + 2. datasets
8 來源是 https://github.com/PacktPublishing/Hands-On-Artificial-Intelligence-for-Cybersecurity/blob/master/Chapter03/datasets/spam_email_no_header.csv
9
10 # Tech Stack
11 - [List your primary technologies]
12 + Python 程式語言開發一個具有 Classification 機器學習的演算法，that means expand the preprocessing steps and add richer visualization work (step outputs, metrics, and CLI/Streamlit views).
13
14 # Project Conventions
15
16 # The "Tech Stack" section is updated. Now, I'll update the "Code Style" section.
```

圖 9: Gemini CLI 接著會開始讀取分析並自動生成各資料夾或文件以符合專案需求，過程中會逐一再次詢問你所回覆的 9 項答覆 (專案規範要求) 的項次 1、2

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ "Tech Stack" is done. Moving on to "Code Style."

Edit openspec/project.md: # Project Context... ==> # Project Context...
10 # Project Conventions
11
12 # Code Style
13 - [Describe your code style preferences, formatting rules, and naming conventions]
14 + 1. specific code style preferences: Python and related packages applied in this project.
15 + 2. formatting rules: Simple and easy to understand
16 + 3. 主程式的命名為 5114050013_hw3.py
17
18 # Architecture Patterns
19 - [Document your architectural decisions and patterns]
20
21 # Code Style" updated. Now, "Architecture Patterns."

Edit openspec/project.md: # Project Context... ==> # Project Context...
15 # 3. 主程式的命名為 5114050013_hw3.py
16
17 # Architecture Patterns
18 - [Document your architectural decisions and patterns]
19 + 利用 CRISP-DM 架構來設計如何完成這項專案，並用 Markdown 記法產生一個 README.md 檔，並利用 CRISP-DM 架構來設計如何完成這項專案，並用 Markdown 記法產生一個 README.md 檔
20
21 # Testing Strategy
22 - [Explain your testing approach and requirements]
23
24 # Architecture Patterns" updated. Now, "Testing Strategy."

Edit openspec/project.md: # Project Context... ==> # Project Context...
18 利用 CRISP-DM 架構來設計如何完成這項專案，並用 Markdown 記法產生一個 README.md 檔，並利用 CRISP-DM 架構來設計如何完成這項專案，並用 Markdown 記法產生一個 README.md 檔
19
20 # Testing Strategy
21 - [Explain your testing approach and requirements]
22 + 1. 先單元測試，再整合測試
23 + 2. end-to-end testing 需要再另外提出請求
24
25 # Git Workflow
26 - [Describe your branching strategy and commit conventions]
27
```

圖 10: Gemini CLI 逐一要我確認我回覆的 9 項答覆 (專案規範要求) 的項次 3~5

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ "Testing Strategy" is updated. Now, "Git Workflow."

Edit openspec/project.md: # Project Context... ==> # Project Context...
22 2. end-to-end testing 需要再另外提出請求
23
24 # Git Workflow
25 - [Describe your branching strategy and commit conventions]
26 + 1. 基於分支開發，要提交新文件與程式碼上傳到指定於 GitHub (Security_HM_03_Classification_Spam-Email)
27
28 + 2. 遵循 https://github.com/candice-w/Security_HM_03_Classification_Spam-Email.git
29
30 + 3. 遵循嚴格先提交 requirements.txt 再提交 gitignore 檔案
31
32 + 4. 在 .gitignore 中指定 *.pyc 和 *.pyo 檔案
33
34 + 5. README.md 內容中描述如何 CRISP-DM 架構來設計如何完成這項專案，並上方要標出 Streamlit demo website 連結並提供專案成果
35
36 # Domain Context
37 - [Add domain-specific knowledge that AI assistants need to understand]
38
39 # Git Workflow" updated. Now, "Domain Context."

Edit openspec/project.md: # Project Context... ==> # Project Context...
29 5. README.md 內容是描述如何 CRISP-DM 架構來設計如何完成這項專案，並上方要標出 Streamlit demo website 連結並提供專案成果
30
31 # Domain Context
32 - [Add domain-specific knowledge that AI assistants need to understand]
33 + 符合一般大數據理解的範疇，且能一窺機器學習操作 Streamlit 網頁互動的各項功能
34
35 # Important Constraints
36 - [List any technical, business, or regulatory constraints]
37
38 # Domain Context" updated. Now, "Important Constraints."

Edit openspec/project.md: # Project Context... ==> # Project Context...
32 符合一般大數據理解的範疇，且能一窺機器學習操作 Streamlit 網頁互動的各項功能
33
34 # Important Constraints
35 - [List any technical, business, or regulatory constraints]
36
37 # External Dependencies
38 - [Document any external services, APIs, or systems]
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ "Domain Context" is updated. Now, "External Dependencies."

Edit openspec/project.md: # Project Context... ==> # Project Context...
35 None
36
37 # External Dependencies
38 - [Document key external services, APIs, or systems]
39 + 1. GitHub
40 + 2. Streamlit

+ openspec/project.md is fully updated. Task 1 is complete.

Moving to Task 2: I'll now read openspec/AGENTS.md to explain the OpenSpec workflow.

ReadFile openspec/AGENTS.md

+ The OpenSpec workflow is a structured, three-stage process for managing project changes through spec-driven development.

Stage 1: Creating Changes (Proposals)
This stage is for proposing significant changes like new features, breaking changes, or architectural shifts. It involves:
1. Understanding Context: Reviewing openspec/project.md, existing spec, and pending changes.
2. Choosing a "change-id": A unique, verb-led ID (e.g., add-user-auth).
3. Scaffolded: Creating a changes/change-id/ directory with:
  - proposal.md: Explains why, what, and about.
  - tasks.md: An implementation checklist.
  - design.md (optional): For complex changes.
  - specs/: "Delta spec" detailing ADDED, MODIFIED, REMOVED, or RENAMED requirements with scenarios.
4. Validation: Running openspec validate change-id --strict.
5. Approval: Waiting for your approval before implementation.

Stage 2: Implementing Changes
Once approved, implementation proceeds by:
1. Reusing proposal, design, and tasks.
2. Implementing tasks sequentially and marking them complete.

Stage 3: Archiving Changes
After deployment, the change is archived by moving its directory to changes/archive/ and updating main spec/ if necessary.

How we'll work together:
+ I will always refer to openspec/project.md and existing specs for context.
+ For significant changes, I will create an OpenSpec proposal for your review and approval.
+ I will implement changes only after your approval, tracking progress via tasks.md.
+ I will ask clarifying questions for ambiguous requests and use openspec validate for correctness.

Now, I'm ready to create your first change proposal (Task 2).
```

## WK03 Homework

圖 11: Gemini CLI 逐一要我確認我回覆的 9 項答覆 (專案規範要求) 的項次 6~8

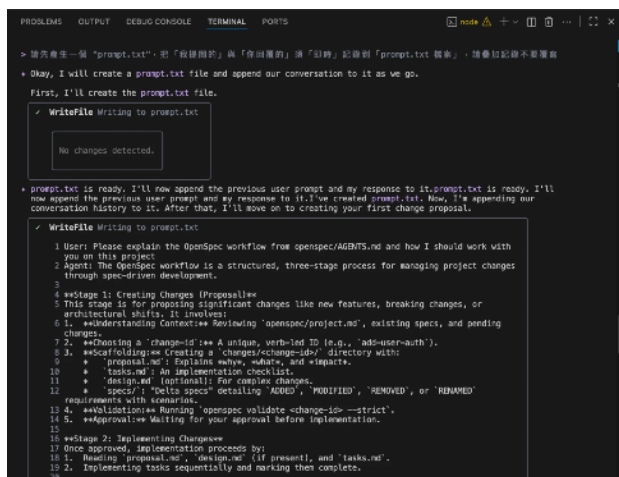


圖 13: Gemini CLI 接下來要進行 Task 2 (create your first change proposal)，但我先請它幫我先做對話紀錄並寫進 prompts.txt

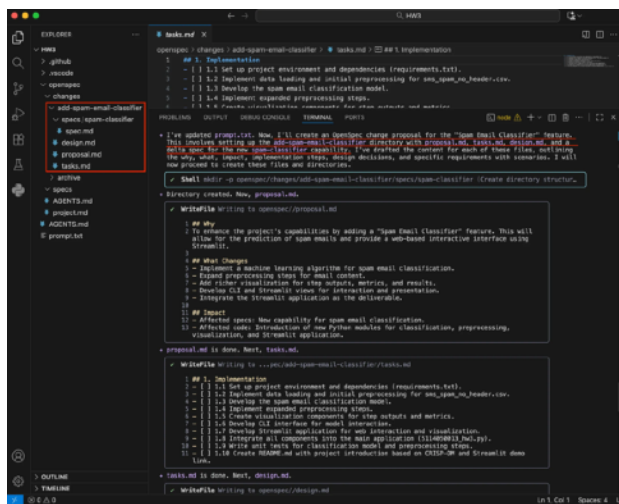


圖 15: Gemini CLI 完成對話記錄檔，自動回到 Task 2 依 openspec 架構去自動產生 OpenSpec change proposal，自動產生 [add-spam-email-classifier](#) 資料夾，底下有 [proposal.md](#), [tasks.md](#), [design.md](#), and a delta spec，即自動地生成專案提案

圖 12: Gemini CLI 逐一要我確認我回覆的 9 項答覆 (專案規範要求) 的項次 9。接著，它表示 Task 1 已完成，並主動先跳到 Task 3 解釋 OpenSpec workflow，最後表示說要開始進行 Task 2

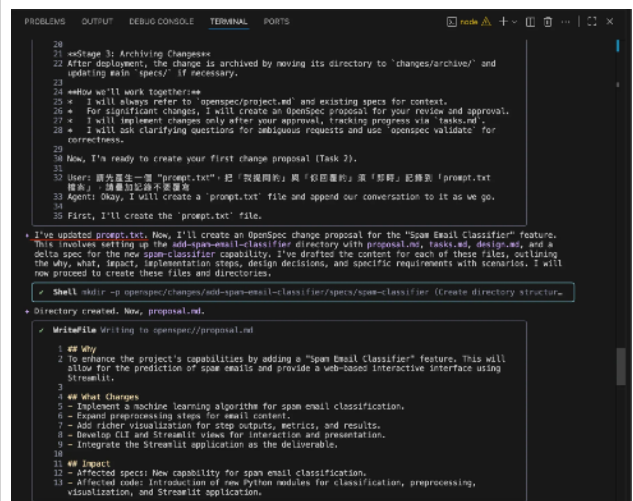


圖 14: Gemini CLI 幫忙生成對話記錄檔，也很聰明地把前面的對話自動回補到 prompts.txt。

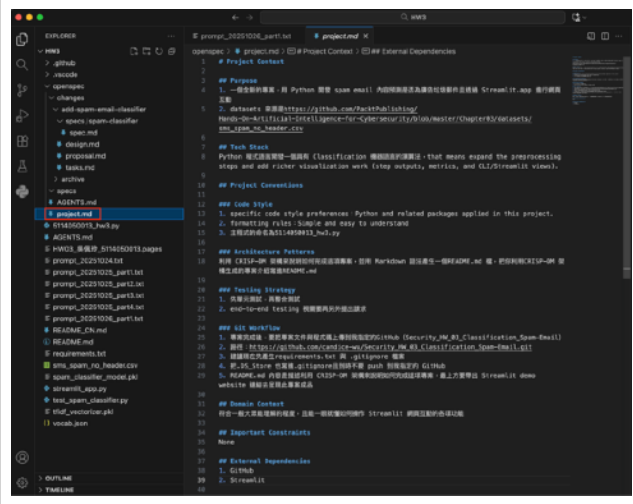


圖 16: 之前我回覆的9項專案規範要求，Gemini CLI 則產生一份 project.md 去記錄

註：可透過以下指令請 Gemini CLI 確認專案內容

指令	用途	常見參數 / 備註
<code>openspec list</code>	列出目前所有「變更 (changes)」資料夾和內容狀態 (例如：哪些 change 尚未審核、哪些已完成)	
<code>openspec show &lt;資料夾名稱&gt;</code>	定義一個新的變更 (例如 “add-spam-classifier”) 後用來顯示詳細內容：Proposal、Tasks、Spec design 等	
<code>openspec validate &lt;資料夾名稱&gt;</code>	驗證指定變更的格式與結構是否正確 (例如是否缺少 <code>proposal.md</code> 、 <code>tasks.md</code> 、 <code>specs</code> 目錄等)	
<code>openspec archive [--yes]</code>	-y]	封存 (archive) 已完成的變更並更新主規格 (Specs) 為最新狀態。

2. Gemini CLI 開始執行自動化程式開發

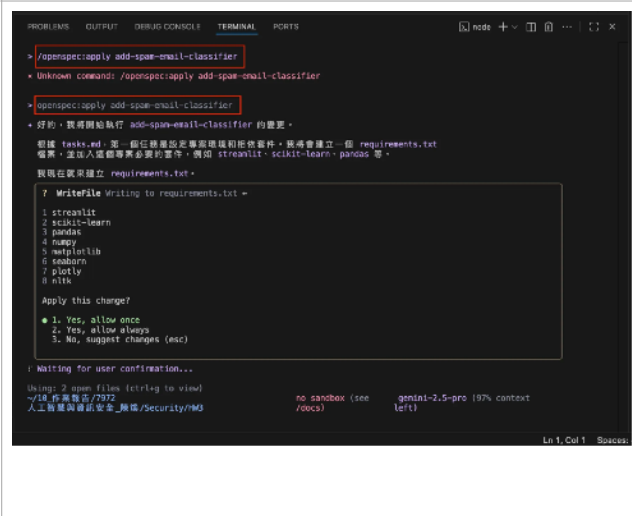


圖 1: 下開發指令：  
`/openspec:apply add-spam-email-classifier`  
或  
`openspec:apply add-spam-email-classifier`  
註：加 slash 若失敗就不要加，再重下指令就會執行

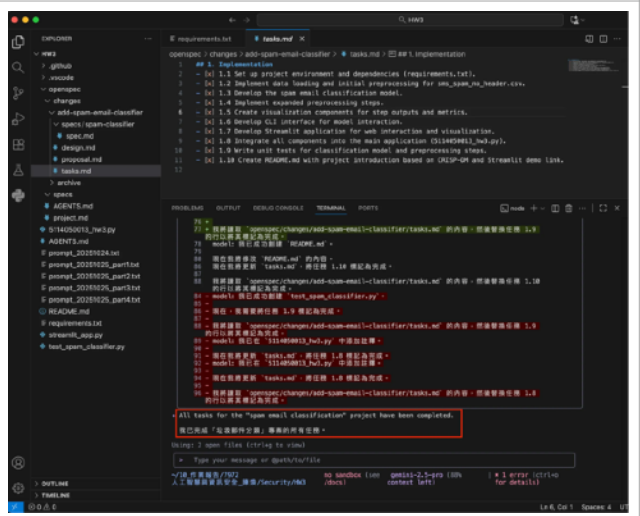


圖 2: Gemini CLI 開發過程中會依照 `task.md` 清單逐步開發導入，完成的項目會打 X，最終完成十項



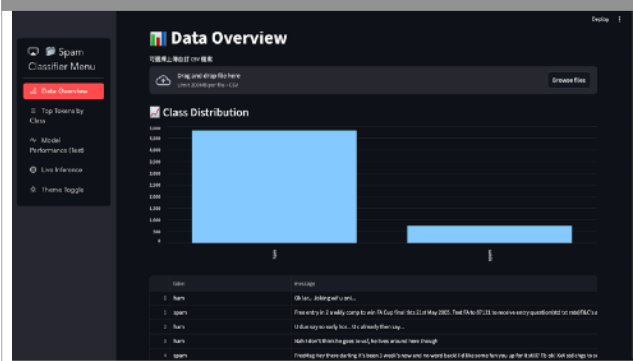


圖 3: Streamlit - Data Overview

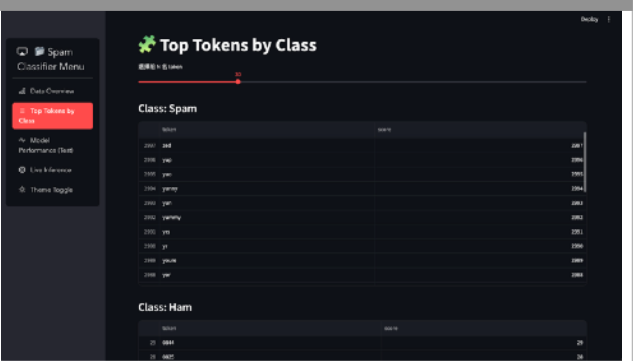


圖 4: Streamlit - Top Tokens by Class

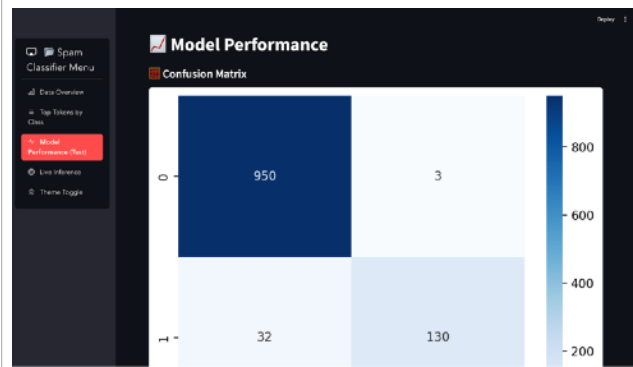


圖 5: Streamlit - Model Performance (Test)



圖 6: Streamlit (Live Inference)

## [Phase 2：請 Gemini 升級我的專案]

註：圖1~5同 Phase 1

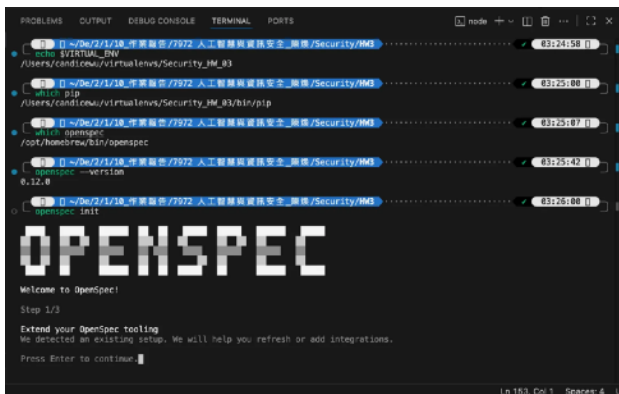


圖 1: openspec init

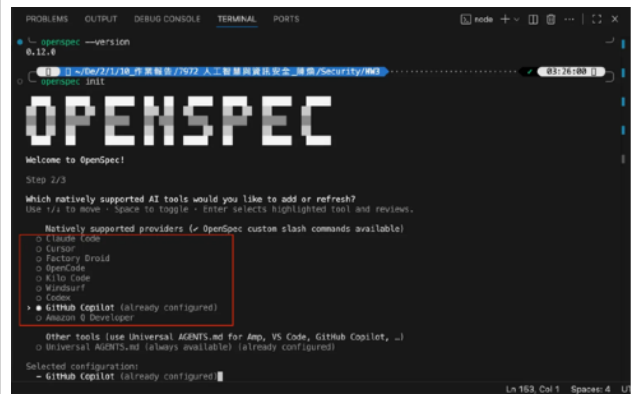


圖 2: Select GitHub Copilot

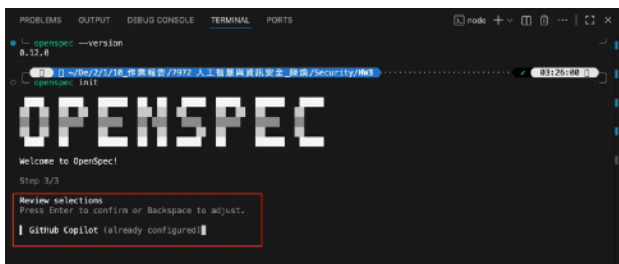


圖 3: Double confirm

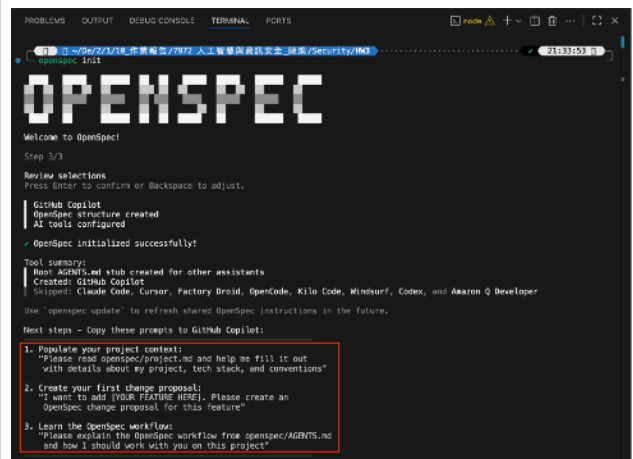


圖 4: Openspec 要求轉貼三個指令到 AI coding CLI

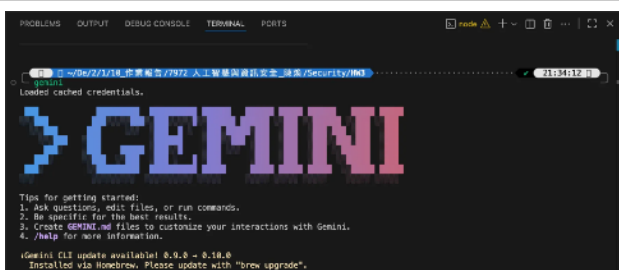


圖 5: 啟動 (呼叫) Gemini CLI



圖 6: 此時不用再轉貼 Openspec 要求的三指令，直接請 Gemini CLI 去讀取現有專案內容，請它幫忙做調整並提出建議

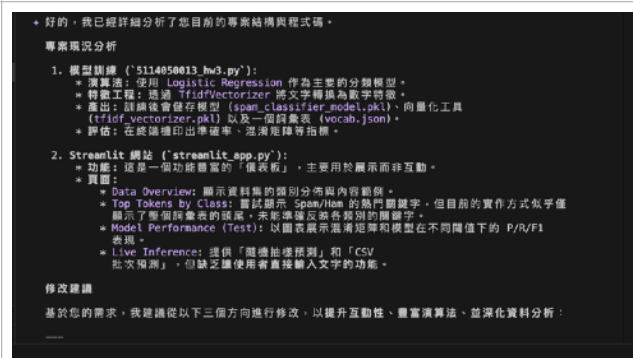


圖 7: Gemini CLI 讀取現有專案結構與程式碼，分析出專案現況



圖 8: Gemini CLI 給出修改建議



圖 9: Gemini CLI 問我修改想法，我回「全部採納」並更新相對應的專案文件。

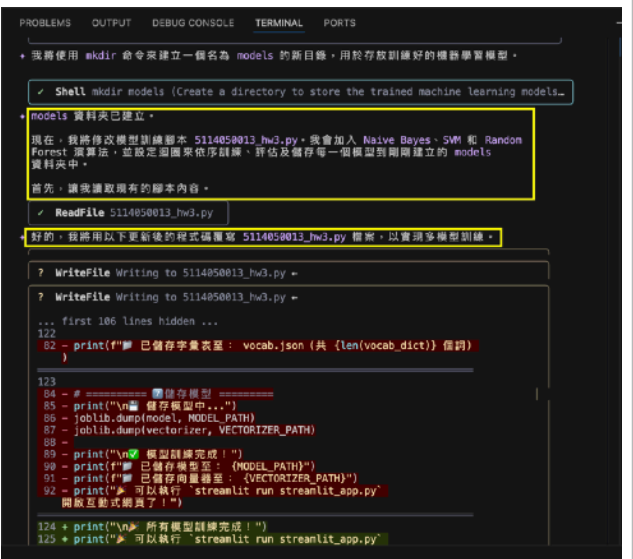


圖 10: Gemini CLI 讀取分析現有專案功能後，給予建議，並使用 mkdir 命令來建立一個名為 models 的新目錄，用於存放訓練好的機器學習模型。



```
prompt_20251026_part1.txt tasks.md X
openspec > changes > add-spam-email-classifier > tasks.md > ## 1. Initial implementation
1 ## 1. Initial Implementation
2 - [x] 1.1 Set up project environment and dependencies (requirements.txt).
3 - [x] 1.2 Implement data loading and initial preprocessing for sms_span_no_header.csv.
4 - [x] 1.3 Develop the spam email classification model.
5 - [x] 1.4 Implement expanded preprocessing steps.
6 - [x] 1.5 Create visualization components for step outputs and metrics.
7 - [x] 1.6 Develop CLI interface for model interaction.
8 - [x] 1.7 Develop Streamlit application for web interaction and visualization.
9 - [x] 1.8 Integrate all components into the main application (5114050013_hw2.py).
10 - [x] 1.9 Write unit tests for classification model and preprocessing steps.
11 - [x] 1.10 Create README.md with project introduction based on CRISP-DM and Streamlit demo Link.
12
13 ## 2. Project Upgrade: Interactive Multi-Model Platform
14 - [x] 2.1 Refactor training script ('5114050013_hw3.py') to support multi-model training (Logistic Regression, Naive Bayes, SVM, Random Forest).
15 - [x] 2.2 Create 'models/' directory to store serialized model files.
16 - [x] 2.3 Completely redesign 'streamlit_app.py' into a multi-page application.
17 - [x] 2.4 Implement "Live Prediction" page with model selection and probability display.
18 - [x] 2.5 Implement "Batch Prediction" page with CSV upload and download functionality.
19 - [x] 2.6 Implement "Model Comparison" page with performance metrics table and F1-score visualization.
20 - [x] 2.7 Implement "Exploratory Data Analysis (EDA)" page with class distribution, message length analysis, and word clouds.
21 - [x] 2.8 Ensure all new library dependencies ('wordcloud') are in 'requirements.txt'.
22 - [x] 2.9 Update 'README.md' to reflect all new features, architecture, and usage instructions.
23 - [x] 2.10 Update 'README_CN.md' with the translated content of the new README.
24 - [x] 2.11 Update 'openspec/proposal.md' to outline the new project goals and scope.
25 - [x] 2.12 Update 'openspec/design.md' with the detailed architecture of the multi-model system and Streamlit app.
26 - [x] 2.13 Update this 'tasks.md' file to log the completion of the upgrade tasks.
```

圖 11: Gemini CLI 針對升級專案功能，不會直接修改原 tasks，而是另外新增，可以清楚掌握前後版更的內容性

```
prompt_20251026_part1.txt X
201 說明執行
202
203 專案現在已準備就緒，您可以透過以下步驟啟動新的應用程式：
204
205 1. (如未安裝) 啟動您的虛擬環境
206 2. 啟動 Streamlit 應用程式
207 3. 透過瀏覽器訪問您的應用
208
209 architecture of the multi-model system and Streamlit app.
210 2.6 - [x] 2.13 Update this 'tasks.md' file to log the completion of the upgrade tasks.
211
212 tasks.md 文件中已更新。
213
214 至此，您已完成包含要求的所有修改，專案升級已完整完成。
215
216 工作總結
217 1. 構建專案架構 ('5114050013_hw3.py')：
218   - 包含多個子目錄，用於存放不同的分類模型 (Logistic Regression, Naive Bayes, SVM, Random Forest)。
219   - 所有模型檔案都將存儲在升級後的 models/ 資料夾中。
220 2. Streamlit 應用程式 ('streamlit_app.py')：
221   - 以多頁面設計，成為一個功能強大的多模型系統。
222   - 包含多個頁面，可選擇不同的模型，並顯示預測結果、CSV 檔案上傳、模型性能比較表、以及包含文字雲的探索性數據分析 (EDA)。
223 3. 專案文件：
224   - README.md 和 README_CN.md 已全面更新，以反映專案的升級功能與架構。
225   - openspec/ 目錄內的 proposal.md, design.md, 和 tasks.md 文件均已同步更新，記錄了此次升級的計劃、設計和完成狀態。
226
227 執行說明
228 專案現在已準備就緒，您可以透過以下步驟啟動新的應用程式：
229
230 1. (如未安裝) 啟動您的虛擬環境
231 2. 啟動 Streamlit 應用程式
232 3. 透過瀏覽器訪問您的應用
233
234 2. 啟動 Streamlit 應用程式
235 1. 透過 Streamlit 應用程式
236 1. 透過 Streamlit 應用程式
237
238 感謝您的耐心，希望對這次的專案升級感到滿意！
239
240 Usage: 1. open file (clicking to view) | 1. openspec.md file
241 2. Type your message or @path/to/file
242
243 /7972 No sandbox issue gpt4o-1.5-gpt-4o (10k context left)
244 人工智慧與資訊安全_博安/Security/HW2 /docs/
```

圖 12: Gemini CLI 已升級功能並更新此專案程式碼、專案文件、對話記錄



圖 13: Streamlit - Live Text Prediction



圖 14: Streamlit - Batch Prediction with CSV

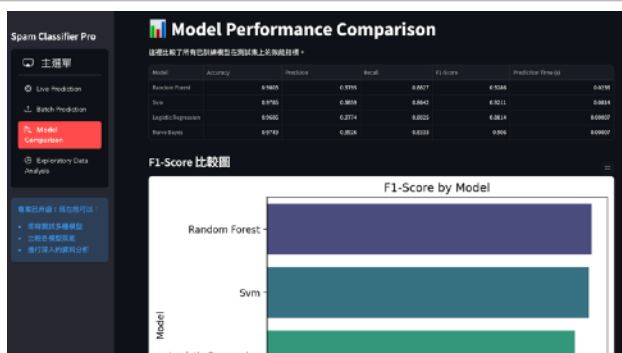


圖 15: Streamlit - Model Performance Comparison

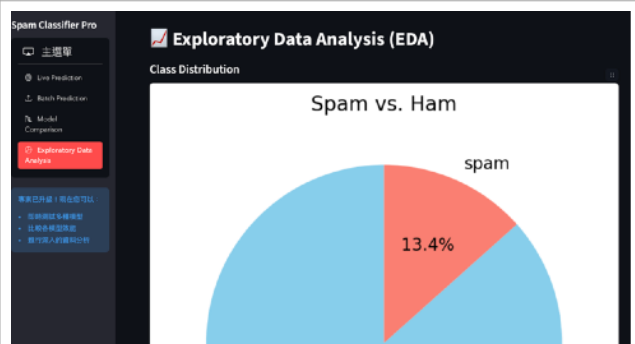


圖 16: Streamlit - Exploratory Data Analysis (EDA) - Class Distribution

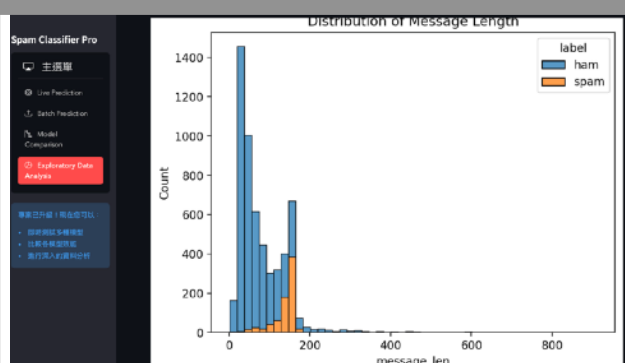


圖 17: Streamlit - Exploratory Data Analysis (EDA) - Message Length Distribution

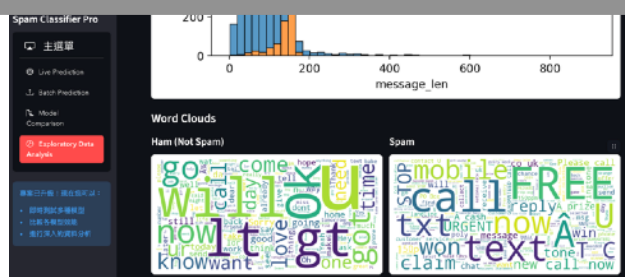


圖 18: Streamlit - Exploratory Data Analysis (EDA) - Word Clouds [Ham (Not Spam) | Spam]

持續請 Gemini CLI 依新需求調整程式，最終結果如下：

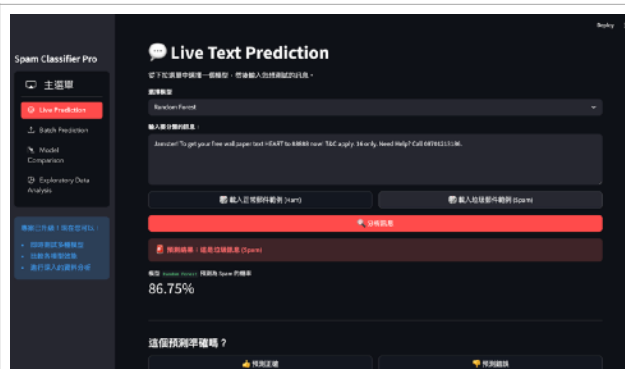


圖 1: Live Text Prediction - 訊息可透過按鍵隨機產生，也能手動輸入



圖 2: 預測正確會有動態氣球慶祝及上方會有 pop-up window (太棒了！感謝您的正面回饋。)

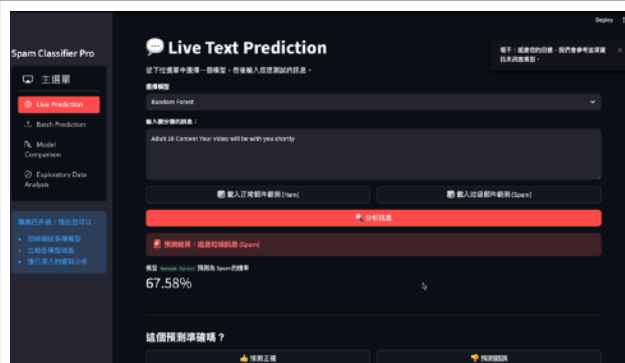


圖 3: 預測失敗則右上角會有 pop-up window (喔不！感謝您的回饋，我們會參考這項資訊來改進模型。)

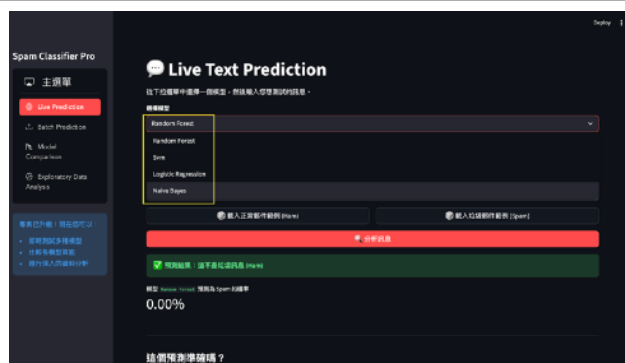


圖 4: Live Text Prediction - 提供不同模型 Random Forest, SVM, Logistic Regression, Naive Bayes

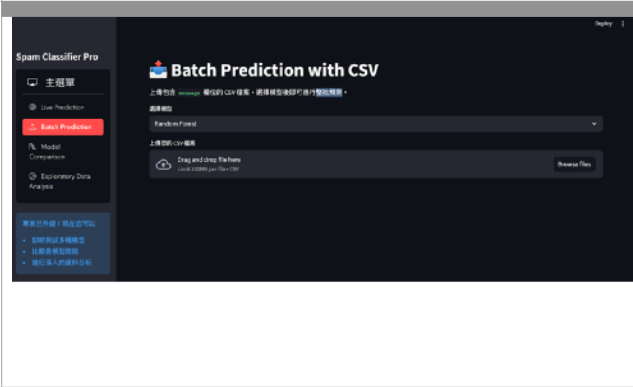


圖 5: Batch Prediction with CSV 提供 csv 上傳功能，更能作整批預測

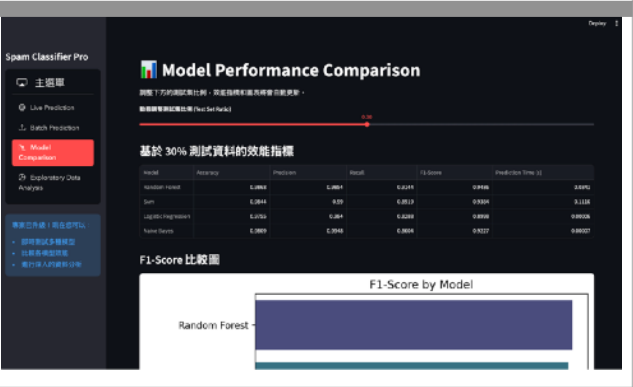


圖 6: Model Performance Comparison 模型比較  
[主要頁面]  
(1) 有動態滑桿  
(2) 下方分析指標與圖表會即時變更  
  
[基於 20% 測試資料的效能指標]  
(1) 指標表可下載  
(2) 可放大畫面  
(3) 關鍵字搜尋  
  
[F1-Score 比較圖]  
(1) 依動態滑桿即時變更  
(2) 可放大畫面  
  
[混淆矩陣 (Confusion Matrix) 詳細分析]  
(1) 依動態滑桿即時變更  
(2) 可放大畫面  
(3) 可任意選擇不同模型以查看其混淆矩陣

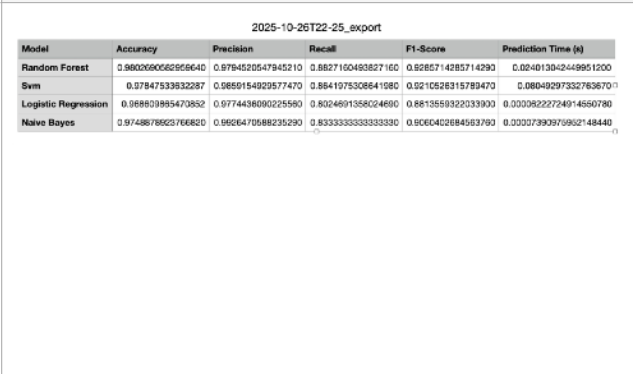


圖 6: Model Performance Comparison 模型比較  
[基於 20% 測試資料的效能指標]  
下載的指標表

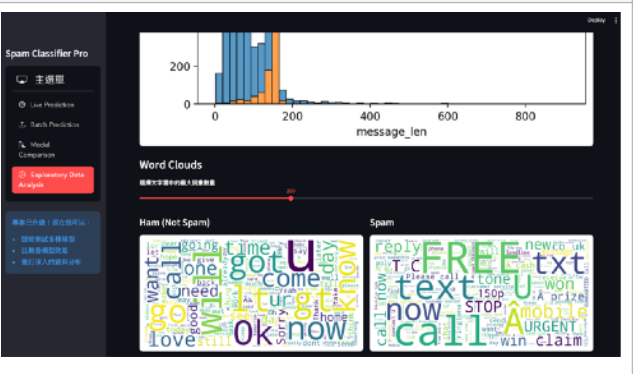
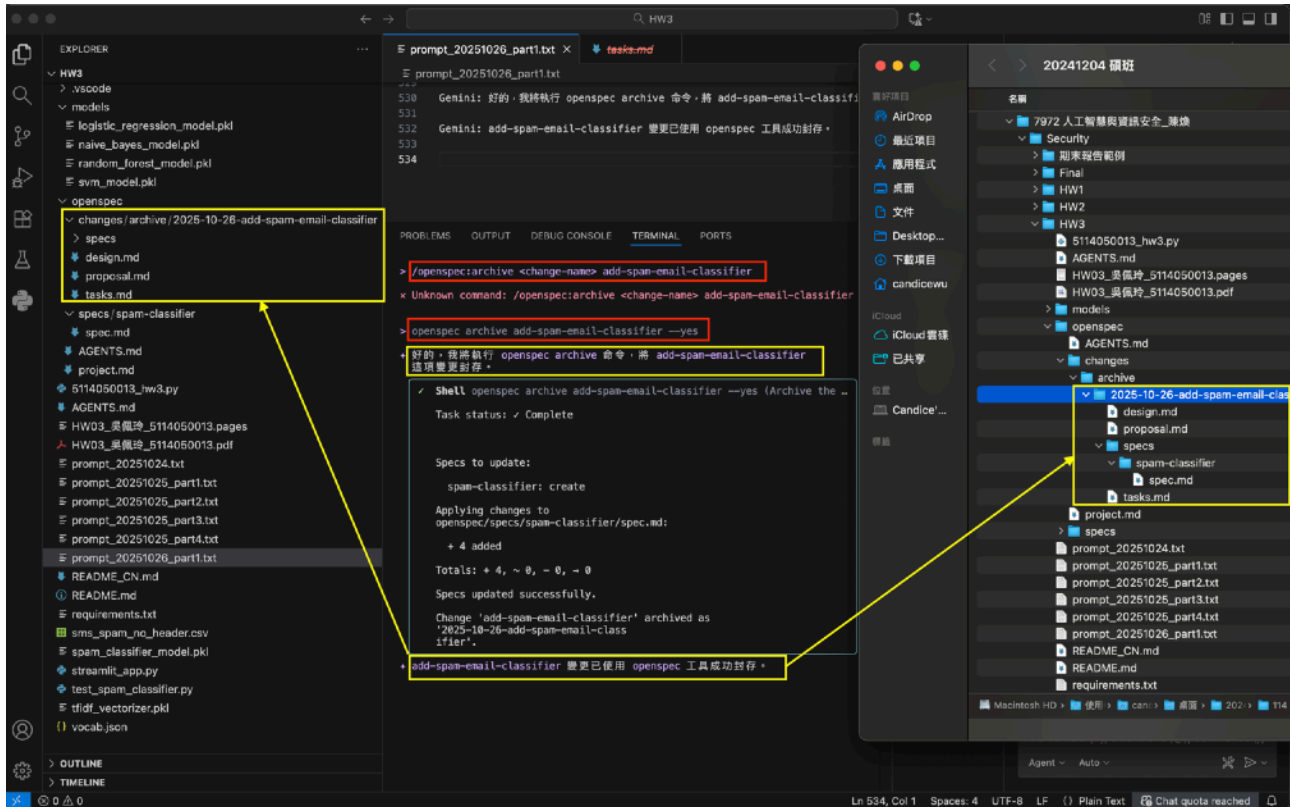


圖 8: Exploratory Data Analysis (EDA)  
[Class Distribution]  
圓餅圖  
  
[Message Length Distribution]  
直方圖  
  
[Word Clouds]  
(1) 依動態滑桿即時變更  
(2) 有詞雲圖

### 3. 歸檔

指令：`openspec archive add-spam-email-classifier --yes`

執行成功，則會新增一個「2025-10-26-add-spam-email-classifier 資料夾」且將 4 份專案文件放在它底下。



### 4. Git push 到 remote GitHub

[https://github.com/candice-wu/Security\\_HW\\_03\\_Classification\\_Spam-Email/tree/main?tab=readme-ov-file](https://github.com/candice-wu/Security_HW_03_Classification_Spam-Email/tree/main?tab=readme-ov-file)

### 5. Deploy to Streamlit Cloud

5.1. Push 專案資料夾 to GitHub

5.2. 至 <https://share.streamlit.io>，點擊“Create app”

5.3. Repository：下拉選擇 [candice-wu/Security\\_HW\\_03\\_Classification\\_Spam-Email](#)

5.4. Branch：Main

5.5. Main file path：5114050013\_hw3.py

5.6. App URL (optional)：預設可以維持，或改掉並另外命名  
如：[hw03-classification-spam-email](#)

5.7. 點擊“Deploy”即完成部署

candice-wu ▾ My apps My profile Explore Discuss ↗ Create app

**candice-wu's apps**

security\_hw\_01\_linear\_regression · master · main.py

**Get started from a template** View all templates →

GDP dashboard  
[View demo](#)

Chatbot  
[View demo](#)

Support tickets  
[View demo](#)

Blank app  
[View demo](#)

candice-wu ▾ My apps My profile Explore Discuss ↗

[← Back](#)

**What would you like to do?**

**Deploy a public app from GitHub**  
My code is ready on a GitHub repo, and it is totally awesome.  
[Deploy now](#)

**Deploy a public app from a template**  
I want to see what kind of amazing concoctions you have for me.  
[Check out templates](#)

**Deploy a private app in Snowflake**  
I want unlimited enterprise-grade apps, with the security of Snowflake.  
[Start trial →](#)

[← Back](#)

**Deploy an app**

Repository ⓘ [Paste GitHub URL](#)

candice-wu/Security\_HW\_03\_Classification\_Spam-Email

Branch

main

Main file path

5114050013\_hw3.py

App URL (optional)

hw03-classification-spam-email .streamlit.app

Domain is available

[Advanced settings](#)

[Deploy](#)