

# Spam email預測專案分析



我要做一個專案

資料集是[https://github.com/PacktPublishing/Hands-On-Artificial-Intelligence-for-Cybersecurity/blob/master/Chapter03/datasets/sms\\_spam\\_no\\_header.csv](https://github.com/PacktPublishing/Hands-On-Artificial-Intelligence-for-Cybersecurity/blob/master/Chapter03/datasets/sms_spam_no_header.csv)

要做spam email 內容預測 是否是廣告郵件

要使用streamlit.app進行網頁互動

請你根據 CRISP-DM進行分析 並且告知我如何完成我的專案

非常好 🍌

你要做的是一個 **Spam Email 分類預測系統**（垃圾郵件辨識），並且要用 **Streamlit** 建立互動網頁。我們可以依據資料科學的標準流程 **CRISP-DM** 六大步驟，完整規劃與實作這個專案。

## 🚀 一、CRISP-DM 六步驟說明與對應到你的專案

CRISP-DM 步驟	說明	在你的專案中要做的事
1 業務理解 (Business Understanding)	明確定義問題與目標	預測一封 SMS 或 Email 是否為廣告（spam）郵件。
2 資料理解 (Data Understanding)	了解資料集的結構與特性	讀取 <code>sms_spam_no_header.csv</code> ，檢視資料筆數、欄位、類別比例、文字長度分布等。
3 資料準備 (Data Preparation)	清理資料與特徵工程	對文字進行清理（移除符號、轉小寫）、Tokenize、TF-IDF 向量化。
4 建立模型 (Modeling)	選擇與訓練模型	使用如 Logistic Regression、Naive Bayes 或 SVM 建立分類模型。
5 評估模型 (Evaluation)	驗證模型表現	使用 accuracy、precision、recall、F1-score、混淆矩陣。
6 部署 (Deployment)	應用於實際環境	使用 <b>Streamlit</b> 建立互動網頁，讓使用者輸入文字，模型即時預測是否為 spam。

## 🧠 二、資料說明

GitHub 上的資料集：

`sms_spam_no_header.csv`

- 沒有標頭，第一欄是「標籤（spam/ham）」，第二欄是「簡訊內容」
- 範例：

```
arduino
```

```
spam,"Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005"
ham,"Nah I don't think he goes to usf, he lives around here though"
```

## 三、完整開發步驟

### 1 資料理解與載入

```
python

import pandas as pd

# 讀取資料
url = "https://raw.githubusercontent.com/PacktPublishing/Hands-On-Artificial-Intelligence-for-Cybersecurity/master/Chapter03/datasets/sms_spam_no_header.csv"
df = pd.read_csv(url, names=["label", "message"])

# 查看基本資訊
print(df.head())
print(df['label'].value_counts())
```

### 2 資料前處理

- 將文字轉為小寫
- 移除標點符號
- 使用 TF-IDF 轉換文字為向量

```
python

import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

# 清理文字
df['message'] = df['message'].str.lower().str.translate(str.maketrans('', '', string.punctuation))

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(df['message'], df['label'], test_size=0.2, random_state=42)

# 向量化
vectorizer = TfidfVectorizer(stop_words='english')
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

### 3 模型訓練

用 **Naive Bayes**（常見於文字分類）：

```
python

from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

model = MultinomialNB()
model.fit(X_train_tfidf, y_train)

y_pred = model.predict(X_test_tfidf)
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

## 4 建立 Streamlit App (部署階段)

建立檔案：`app.py`

```
python

import streamlit as st
import joblib
import string
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# 載入訓練好的模型與向量器
model = joblib.load("spam_model.pkl")
vectorizer = joblib.load("vectorizer.pkl")

st.title("📧 Spam Email 分類系統")
st.write("輸入郵件內容，我會幫你預測是否為廣告郵件（Spam）")

# 使用者輸入文字
user_input = st.text_area("請輸入郵件內容：")

if st.button("開始預測"):
    if user_input.strip() == "":
        st.warning("請輸入郵件內容！")
    else:
        # 前處理
        text = user_input.lower().translate(str.maketrans('', '', string.punctuation))
        vect_text = vectorizer.transform([text])
        prediction = model.predict(vect_text)[0]

        if prediction == "spam":
            st.error("🚫 這封郵件是【Spam 廣告郵件】！")
        else:
            st.success("✅ 這封郵件是【正常郵件】。")
```

## 5 儲存模型（訓練後執行一次）

```
python

import joblib

joblib.dump(model, "spam_model.pkl")
joblib.dump(vectorizer, "vectorizer.pkl")
```

## 6 執行 Streamlit 應用

在終端機執行：

```
bash

streamlit run app.py
```

開啟後網址預設為：

<http://localhost:8501>

## 四、評估與強化方向

項目	可改進方式
文字前處理	可加入 Stemming / Lemmatization
模型優化	比較 Logistic Regression、SVM、Random Forest
資料平衡	若 spam/ham 比例懸殊，可用 SMOTE 或 class_weight
模型保存	使用 <code>joblib</code> 保存模型以便部署
UI 改進	在 Streamlit 加入統計圖與模型準確率展示

## 五、最終專案結構建議

```
bash

spam_detector/
├── data/
│   └── sms_spam_no_header.csv
├── models/
│   ├── spam_model.pkl
│   └── vectorizer.pkl
├── app.py                # Streamlit 主程式
├── train_model.py        # 模型訓練程式
└── requirements.txt      # 所需套件
```

## ✓ 主要套件需求

```
text

streamlit
scikit-learn
pandas
joblib
```

如果你願意，我可以幫你：

- 1. 🧩 生成完整可執行的 `train_model.py` + `app.py` 兩個檔案
- 2. 🧠 加上 CRISP-DM 的分析報告文字描述（可放進簡報或論文）

要我現在幫你一起生成這兩個程式檔嗎？