

FitRoute AI

自行車運動智能助手

- 整合人工智能、大數據分析與客製化訓練方案的完整解決方案 -

六人小組分工表

組員	角色定位	實作任務內容
陳奎佑	 企劃與資料邏輯設計	設計使用者輸入項目與流程的定義推薦邏輯、規劃運算規則表
許心怡	 UI/UX 規劃與設計	使用 Figma 畫 wireframe、頁面CSS設計、流程圖設計
王碩鋒	 前端工程	<u>開發網頁地圖定位與使用者活動記錄系統，整合進度統計柱狀圖表</u>
蕭任珽	 外部 API 整合與資料處理	Google Cloud API整合，實現地圖視覺化路線規劃與即時資料展示。
李可非	 AI 模型與推薦邏輯	從資料建立迴歸模型，預測時間與配速表，大語言模型調整、量化與本地化實作
陳泰宇	 成果呈現與成長紀錄設計	分配工作及審核結果, 整合前後端資料及微調coding參數

專案動機與構想

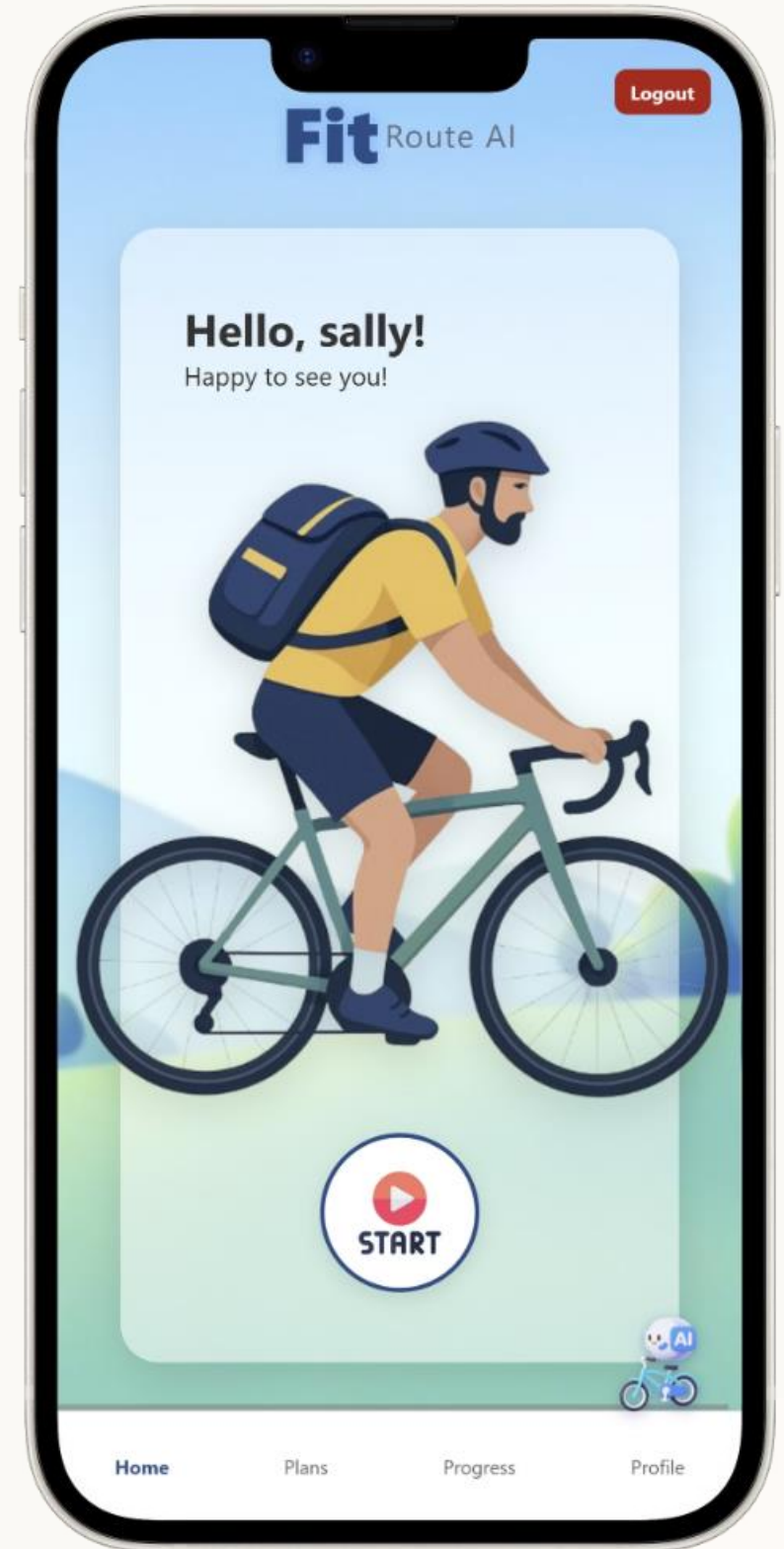
自行車運動越來越受歡迎，但每個人的體能、時間與環境都不同，初學者與銀髮族常因準備不足，導致受傷、效率低落，甚至喪失運動動力。但，如果有一套「量身打造、即時調整」的智能運動規劃工具呢？



專案動機與構想

自行車運動越來越受歡迎，但每個人的體能、時間與環境都不同，初學者與銀髮族常因準備不足，導致受傷、效率低落，甚至喪失運動動力。但，如果有一套「量身打造、即時調整」的智能運動規劃工具呢？

💡 **FitRoute AI** 結合 AI技術與多源資料，打造個人化的自行車運動助手，幫助使用者更聰明、更安全、更有效完成每一次訓練。



(UI/UX) 設計導向

1

用戶入門

簡化用戶入門流程，讓新用戶輕鬆建立個人檔案並設定目標。

2

運動規劃

設計一個直觀的運動規劃流程，提供即時推薦。

3

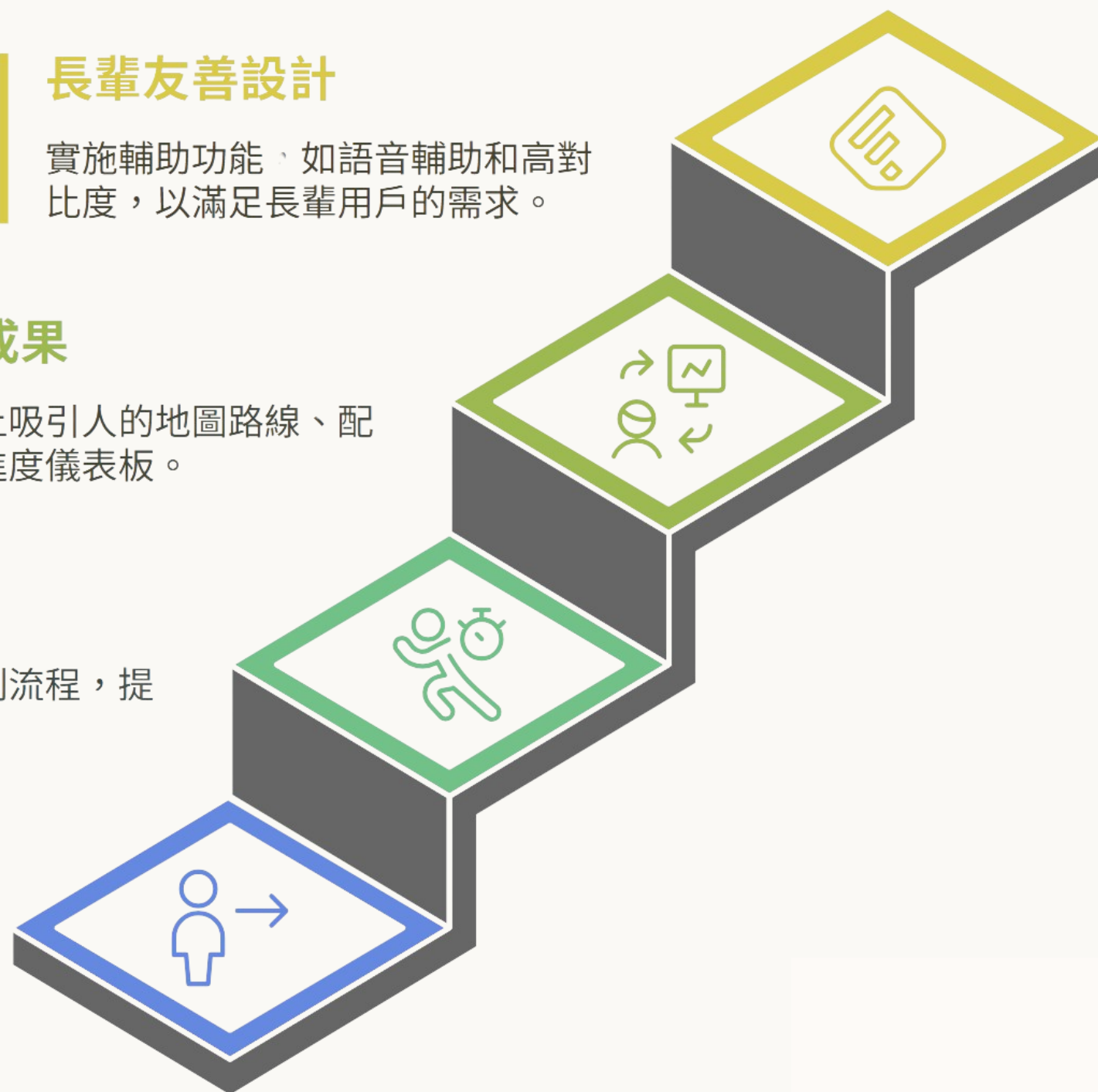
視覺化成果

呈現視覺上吸引人的地圖路線、配速建議和進度儀表板。

4

長輩友善設計

實施輔助功能，如語音輔助和高對比度，以滿足長輩用戶的需求。



核心功能模組



智慧推薦模組 (AI邏輯)



AI 行前簡報- 智慧分析與推薦

 **Road section information**

Distance : 14.7 公里

Duration : 58 分鐘

Start Elevation : 3.1 m , End Elevation : 117.1 m

Estimate Slope : 0.78%

 **Weather**

Temperature: 27.5°C

狀況: 晴朗

Moisture: 73%

(模擬資料)

 **Air quality**

AQI : 74 (Good air quality)

Pollutants : pm10

Ride Tips

 Recommended speed : 15.2 km/h

 Estimated calories burned : 514 kcal

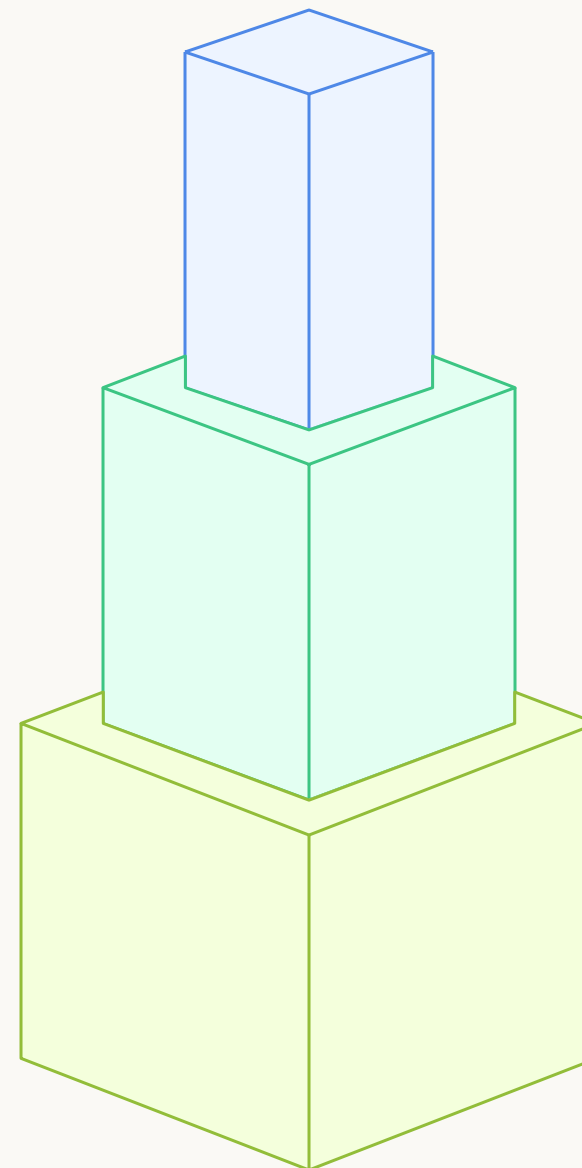
 Recommended water intake : 2939 ml

 **Riding tips :** 嘿，狀況不錯！心情、能量、水分都滿點，但疲勞也有一點點，別讓它影響你！目標是耐力，14.6km的挑戰，天氣超棒！建議你今天輕鬆騎起，保持穩定踩踏，感受陽光和風！注意補充水分，並適時調整呼吸，享受騎行的樂趣！👉

Reset Filters

Return to Home

Start Workout!



功能整合

地圖、路線、天氣、空氣
品質資訊

騎乘建議

整合天氣、地圖、海拔與地形 API
智慧AI給予行前簡報與貼心叮嚀

設計目標

情境引導 (個人狀態選擇)、
使用者有參與感



選手計畫-Plans訓練功能

 週期訓練模式

 客製化模擬模式

 路線配速分析

📅 週期訓練模式

Training Plans

Logout

Week

Custom

Pacing

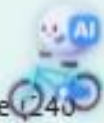
Match Date (YYYY-MM-DD) : 2025/07/10

Target Hours : 20

Training Days : ☐ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat ☒ Sun

Training Plan

Cycle Phase

<div>Tue 2025-06-10 VO2 Max (60 min) 60min</div>	<div>Wed 2025-06-11 Endurance (210 min) 210min</div>	<div>Thu 2025-06-12 Recovery Ride (40 min) 40min</div>
<div>Fri 2025-06-13 Threshold (80 min) + Endurance (240 min)</div>	<div>Sat 2025-06-14 Endurance (210 min)</div>	<div>Sun 2025-06-15 Tempo (110 min) + Endurance (240 min) </div>

根據比賽日期與可訓練時數，
自動生成週期性訓練計畫。






- 自動安排休息日與不同訓練類型
- 智能控制總訓練負荷
- 依週期調整訓練強度


AI 客製化模擬模式


Training Plans Logout


Week Custom Pacing


Custom Plan Setup


 Weight (kg)	 Avg Power (W)
<input type="text" value="60"/>	<input type="text" value="180"/>
 Distance (m)	 Avg Grade (%)
<input type="text" value="5000"/>	<input type="text" value="2.5"/>
 Season	
<input type="text" value="Spring"/>	


 Simulate

 Predicted Time: 00:17:09

 Power-to-Weight Ratio: 3.00 W/kg

 Route Difficulty: 4.94

 Climb Classification: Cat 4



輸入個人參數與路線資料，
預測完賽表現與難度評估。

- 精準計算功率重量比(W/kg)
- 賽事強度分析
- 路線難度評級系統

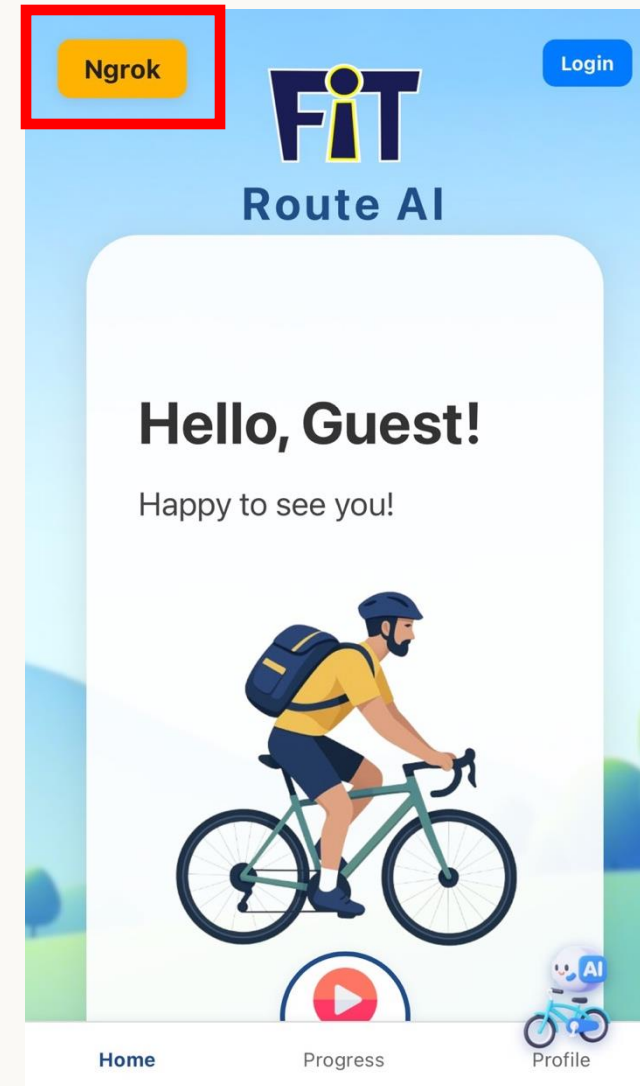
AI 路線配速分析



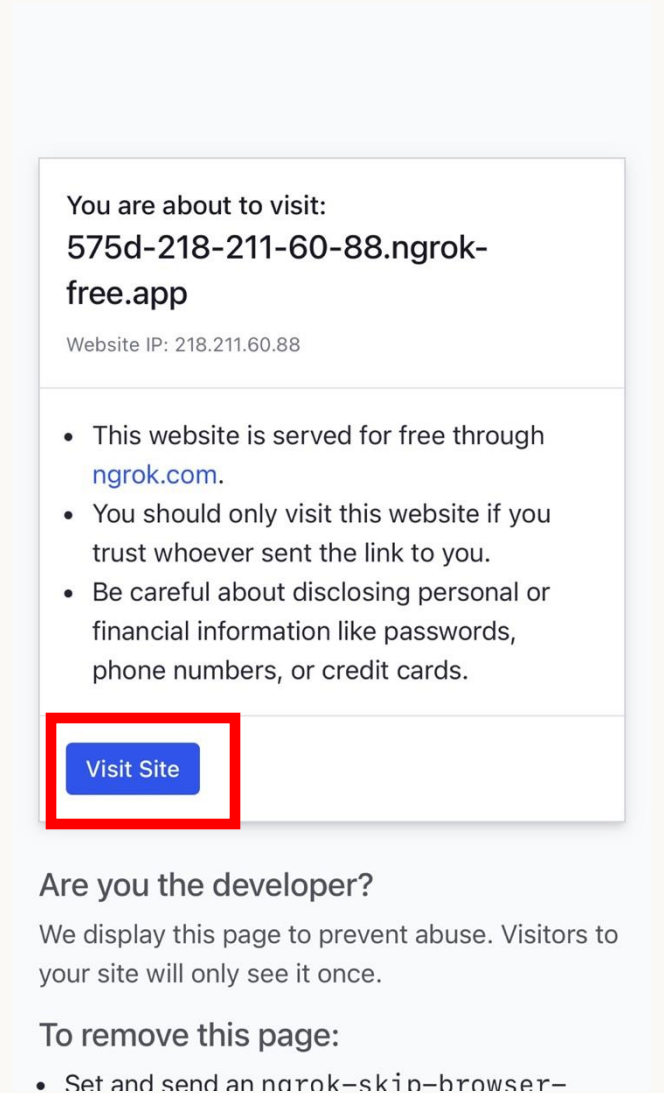
選擇實際路線，
生成功率與距離配速圖表。

- 整合地形與阻力模型
- 分段策略建議
- 比賽與訓練雙重應用

FitRouteAI DEMO



1. 點開Ngrok



2. 選擇 Visit site

時間預測模型報告

本報告介紹一款基於多因子分析的時間預測模型。





功能目標簡介

輸入條件

W/kg、體重、路長、坡度

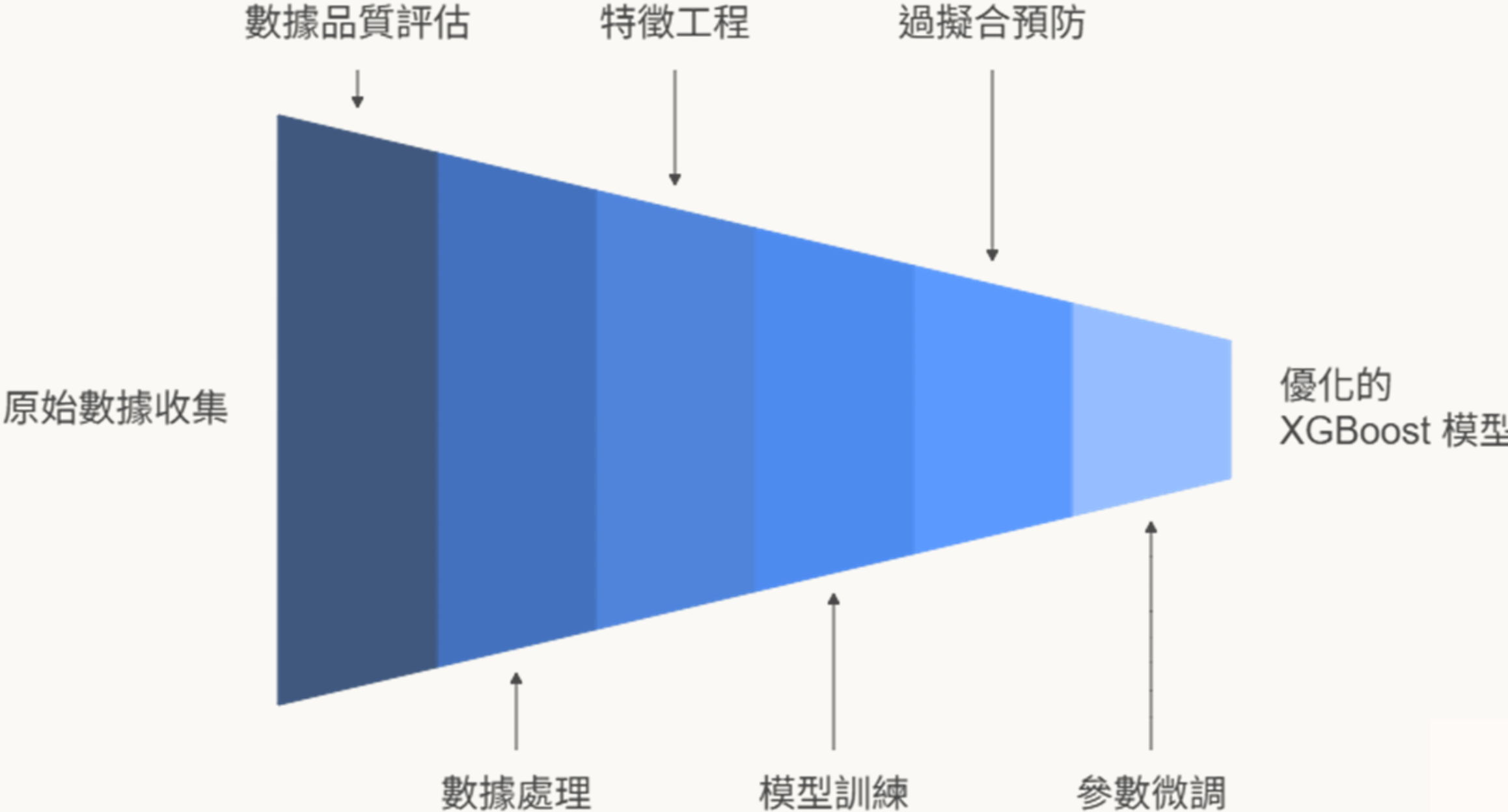
預測目標

完賽時間

解決問題

克服傳統估算粗略且忽略多因子

XGBoost 模型優化流程



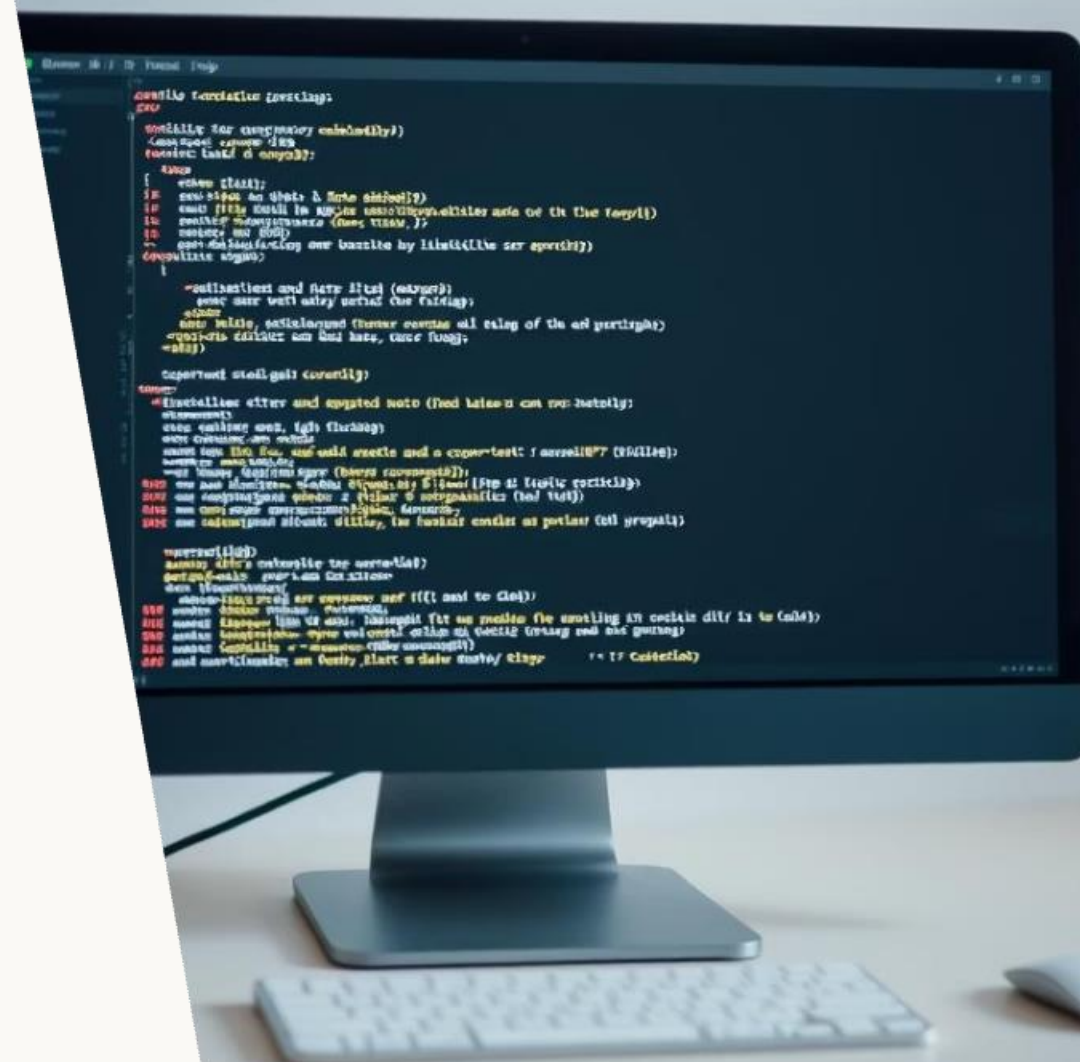
資料來源與爬蟲設計

爬蟲工具

使用 Playwright 自動抓取 Strava Segment 排行榜
(Strava 是一個國際專業運動軟體)

問題與解法

- 路段名稱與ID不對應 → SerpAPI
- 無法直接取得體重 → 以體重級距平均



數據量與品質

數據量

超過15000筆

篩選條件

- 使用**功率計**，含功率資料
- 資料完整無缺漏

A	B	C	D	E	F	G	H	I	J	K	
rank	name	weight_group	estimated_time	time	date	activity_url	speed	avg_power	avg_hr	w_per_kg	
蝦皮搜尋	蝦皮搜尋🔍	WiNS	55 to 64 kg	59.5	15:56	##### https://www.strava.com/segment_efforts/30288514	36.9 km/h	304 W	Power Meter	168 bpm	5.11
2	Yi Chang Wu	55 to 64 kg	59.5	16:04	3-Apr-21	https://www.strava.com/segment_efforts/28132626	36.6 km/h	317 W	Power Meter	165 bpm	5.33
3	YUMING SU	55 to 64 kg	59.5	16:28	#####	https://www.strava.com/segment_efforts/79580590	35.7 km/h	263 W	Power Meter	164 bpm	4.42
4	CHIEN-CHOU CHE	55 to 64 kg	59.5	16:35	#####	https://www.strava.com/segment_efforts/27345926	35.5 km/h	301 W	Power Meter	181 bpm	5.06
4	Kuan hsien Li	55 to 64 kg	59.5	16:35	#####	https://www.strava.com/segment_efforts/27345936	35.5 km/h	213 W	Power Meter	-	3.58
6	Sergio Tu	55 to 64 kg	59.5	16:39	27-Jul-19	https://www.strava.com/segment_efforts/64436569	35.3 km/h	307 W	Power Meter	-	5.16
7	苗栗 遑鯉氨	55 to 64 kg	59.5	16:41	#####	https://www.strava.com/segment_efforts/79583038	35.2 km/h	304 W	Power Meter	-	5.11
8	玄曄 陳	55 to 64 kg	59.5	16:53	1-Aug-20	https://www.strava.com/segment_efforts/27244824	34.8 km/h	275 W	Power Meter	177 bpm	4.62
9	北體 管理員	55 to 64 kg	59.5	16:55	#####	https://www.strava.com/segment_efforts/26864016	34.8 km/h	285 W	Power Meter	-	4.79
9	維庸 曾	55 to 64 kg	59.5	16:55	#####	https://www.strava.com/segment_efforts/29426192	34.8 km/h	304 W	Power Meter	177 bpm	5.11
11	Joseph Lin	55 to 64 kg	59.5	17:03	3-May-25	https://www.strava.com/segment_efforts/33536021	34.5 km/h	279 W	Power Meter	178 bpm	4.69
12	Real Andy	55 to 64 kg	59.5	17:06	#####	https://www.strava.com/segment_efforts/79568383	34.4 km/h	294 W	Power Meter	-	4.94
12	子峯 (Adam chou)	55 to 64 kg	59.5	17:06	#####	https://www.strava.com/segment_efforts/29426127	34.4 km/h	286 W	Power Meter	175 bpm	4.81
14	Ray Hsu	55 to 64 kg	59.5	17:20	9-Apr-16	https://www.strava.com/segment_efforts/13014872	33.9 km/h	286 W	Power Meter	-	4.81
14	KaiWei Chiang	55 to 64 kg	59.5	17:20	3-May-25	https://www.strava.com/segment_efforts/33536037	33.9 km/h	275 W	Power Meter	157 bpm	4.62
16	小弱狗 永和	55 to 64 kg	59.5	17:22	#####	https://www.strava.com/segment_efforts/26965588	33.9 km/h	278 W	Power Meter	162 bpm	4.67

資料處理與特徵工程

資料清洗

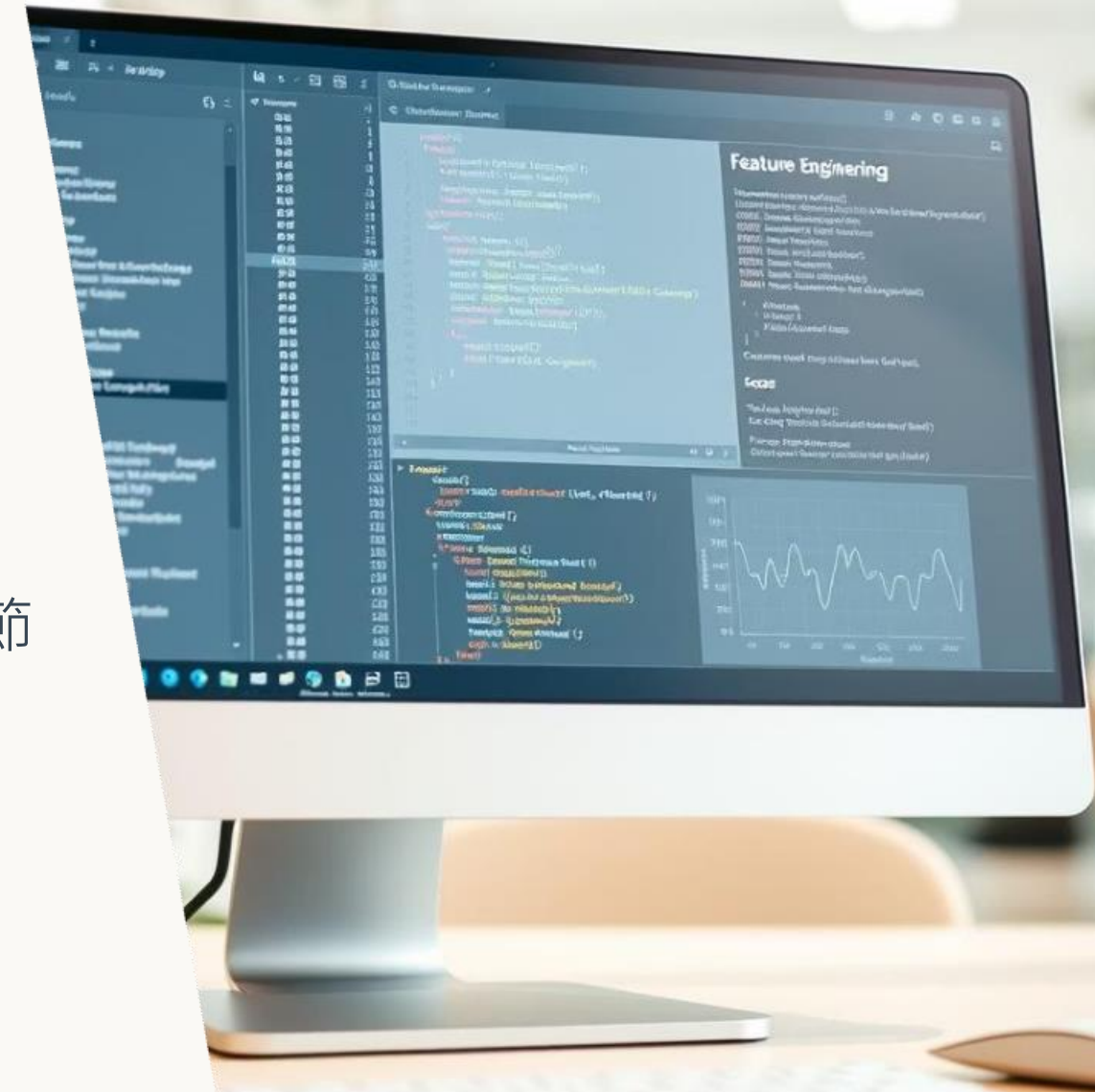
移除缺值與異常值

建立欄位

- 推力比 (W/kg)
- 總距離、平均坡度、季節

初步分析

時間與推力比呈明顯負相關



什麼是 XGBoost ?



為什麼 XGBoost 表現比較好？

- 非線性能力強，跑步/騎車時間等變數之間通常是非線性的，XGBoost 可以捕捉這些複雜關係，而 Linear Regression 無法。
- 自動特徵交互學習XGBoost 可以學習特徵間的交互作用（例如 power 和 weight 同時影響時間）。
- 內建正則化與剪枝減少過擬合風險，使模型更泛化。



Linear Regression Performance:

R^2 : 0.828, MAE: 239.702 sec, RMSE: 342.714 sec

XGBoost Performance:

R^2 : 0.969, MAE: 64.705 sec, RMSE: 146.440 sec

驗證與防止過擬合機制

1. 使用 `train_test_split` 切分資料目的：分出獨立測試集，確認模型泛化能力。
2. 有進行**資料清理**（含時間與功率欄位）處理內容包含：
 - 時間欄位轉為秒數（如 `time_to_seconds()`）
 - 平均功率欄位 `avg_power` 篩選有功率計者並轉為數字，避免訓練錯誤或偏差。
 - dummy 欄位如 `season_Winter` 填補為 0。

驗證與防止過擬合機制

3. 使用合理的超參數設定以下參數可有效控制模型複雜度、避免過擬合：

```
xgb_model = XGBRegressor(  
    n_estimators=100,          # 控制樹的數量，避免過多學習  
    learning_rate=0.1,        # 小學習率讓模型穩健收斂  
    max_depth=5,              # 限制樹的深度，避免學太細  
    random_state=42,          # 固定隨機性  
    objective='reg:squarederror' # 使用適合回歸任務的 loss function  
)
```

✓ 參數調整與交叉驗證

```
1  from sklearn.model_selection import GridSearchCV
2
3  param_grid = {
4      'n_estimators': [50, 100],
5      'max_depth': [3, 4, 5],
6      'learning_rate': [0.05, 0.1, 0.2]
7  }
8
9  grid = GridSearchCV(XGBRegressor(objective='reg:squarederror'),
10                      param_grid,
11                      cv=5,
12                      scoring='neg_mean_absolute_error',
13                      verbose=1)
14
15  grid.fit(X_train, y_train)
16
17  print("最佳參數: ", grid.best_params_)
18  print("最佳 MAE: ", -grid.best_score_)
19
```



```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
最佳參數: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
最佳 MAE: 64.68128419340776
```

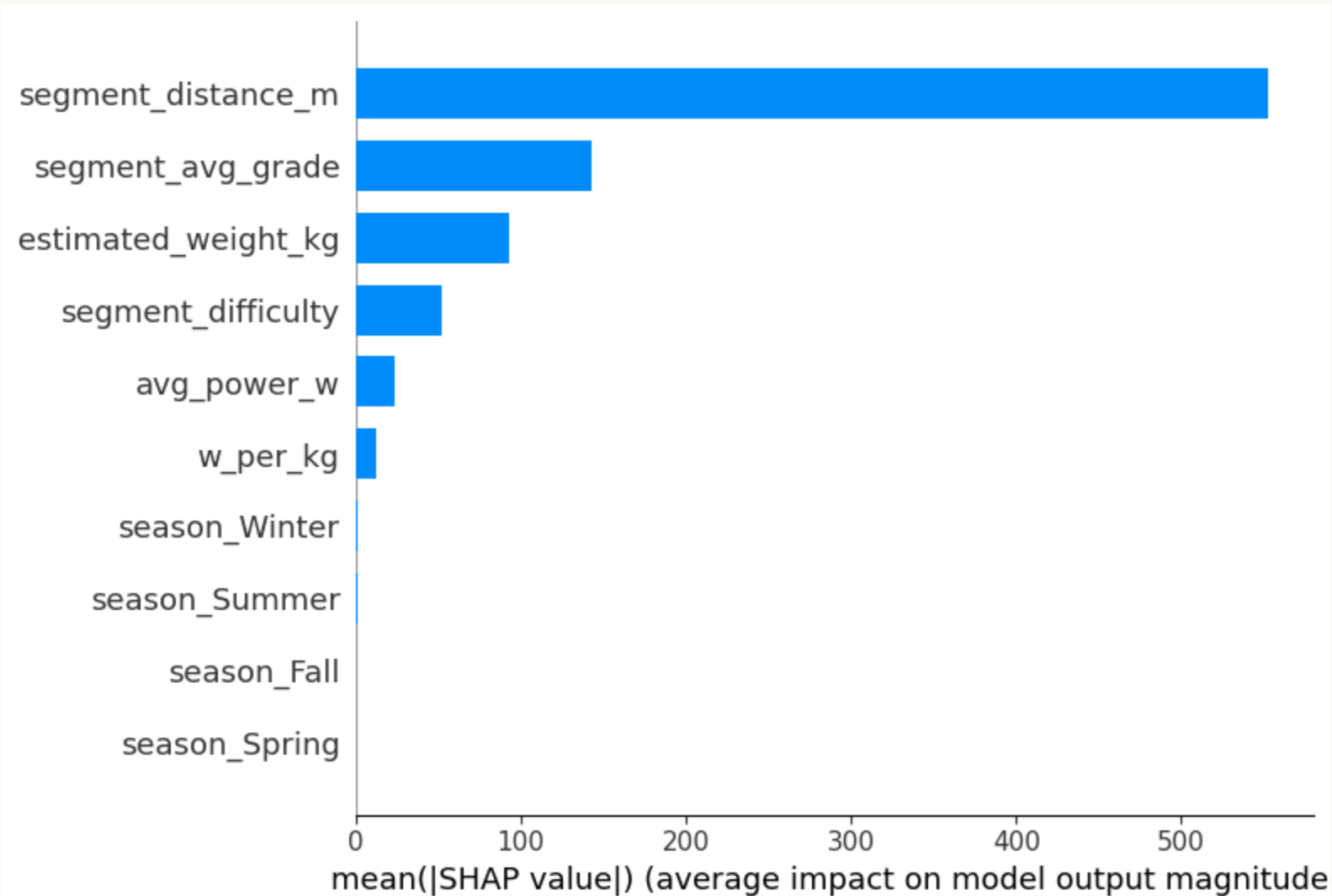
模型評估與解釋

評估指標

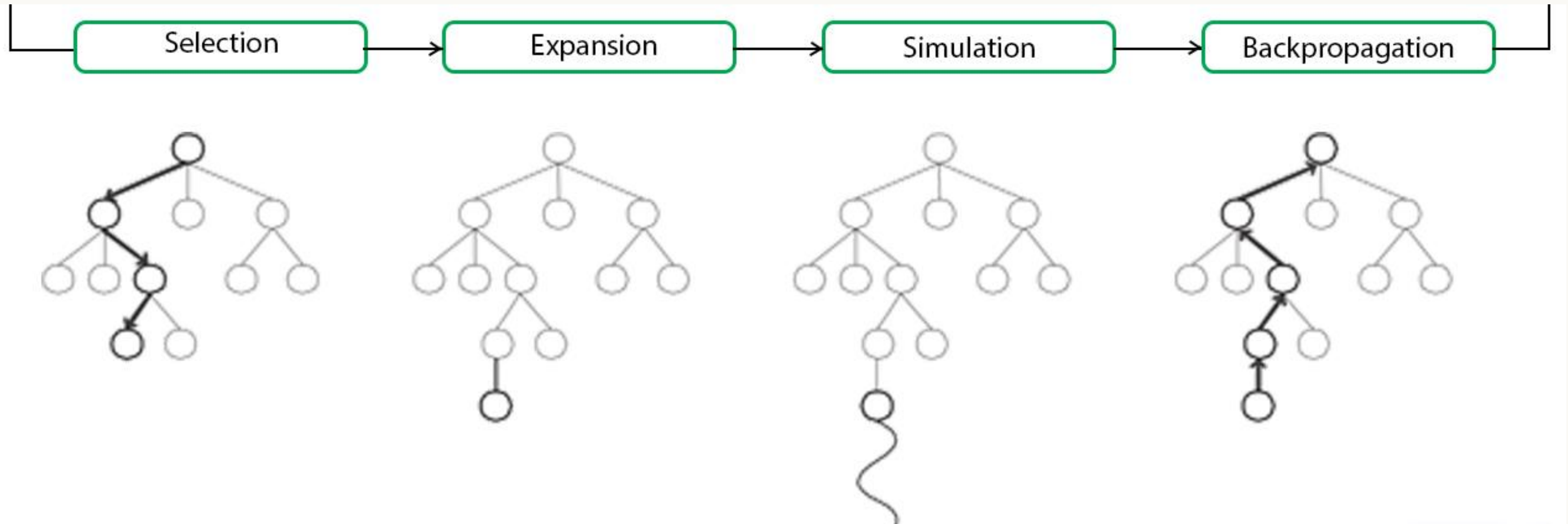
$R^2 \approx 0.969$ ，高準確度

SHAP解釋

- 距離影響最大
- 坡度次高
- 最後才是推力比

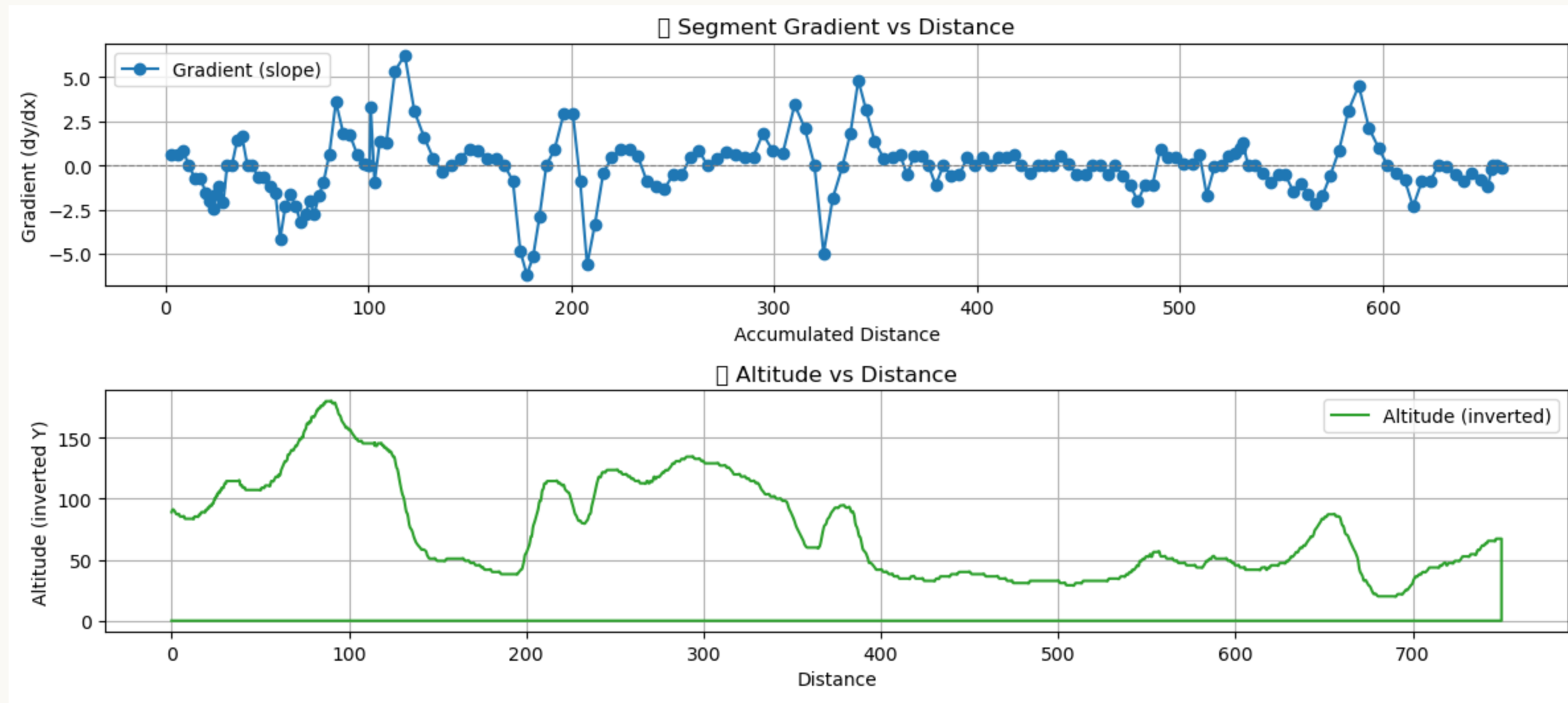


Monte Carlo Tree Search



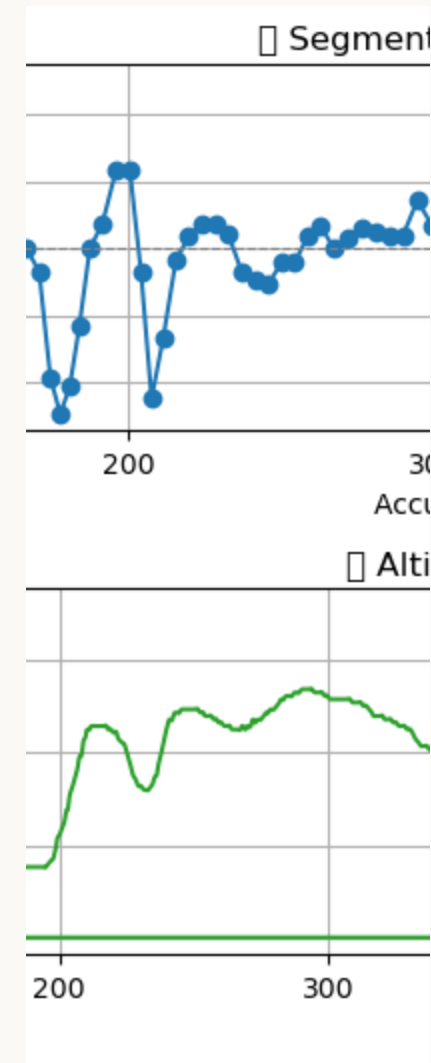
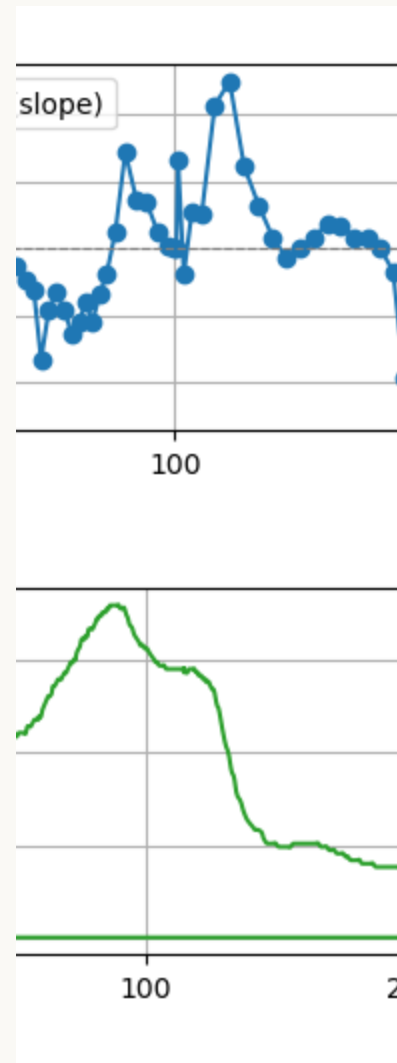
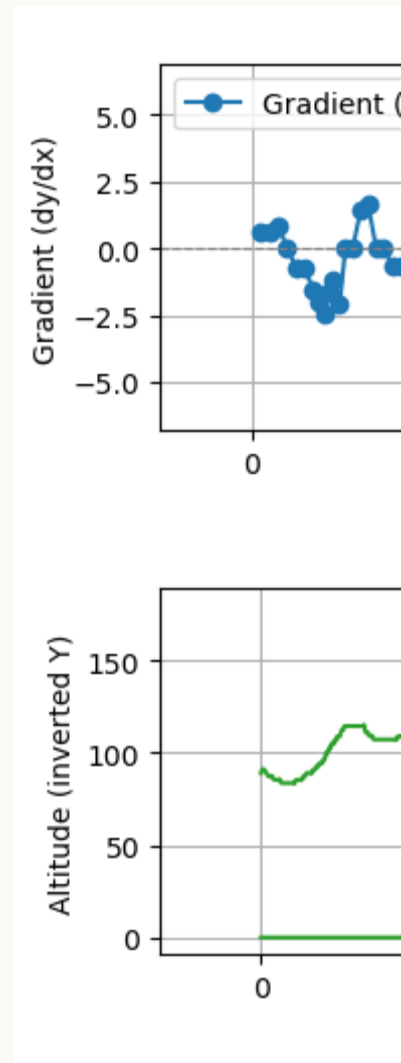
XGBoost 模型應用

將路段圖拆分，針對每一個路段假設功率套用模型計算時間，最後得到總時間



XGBoost 模型應用

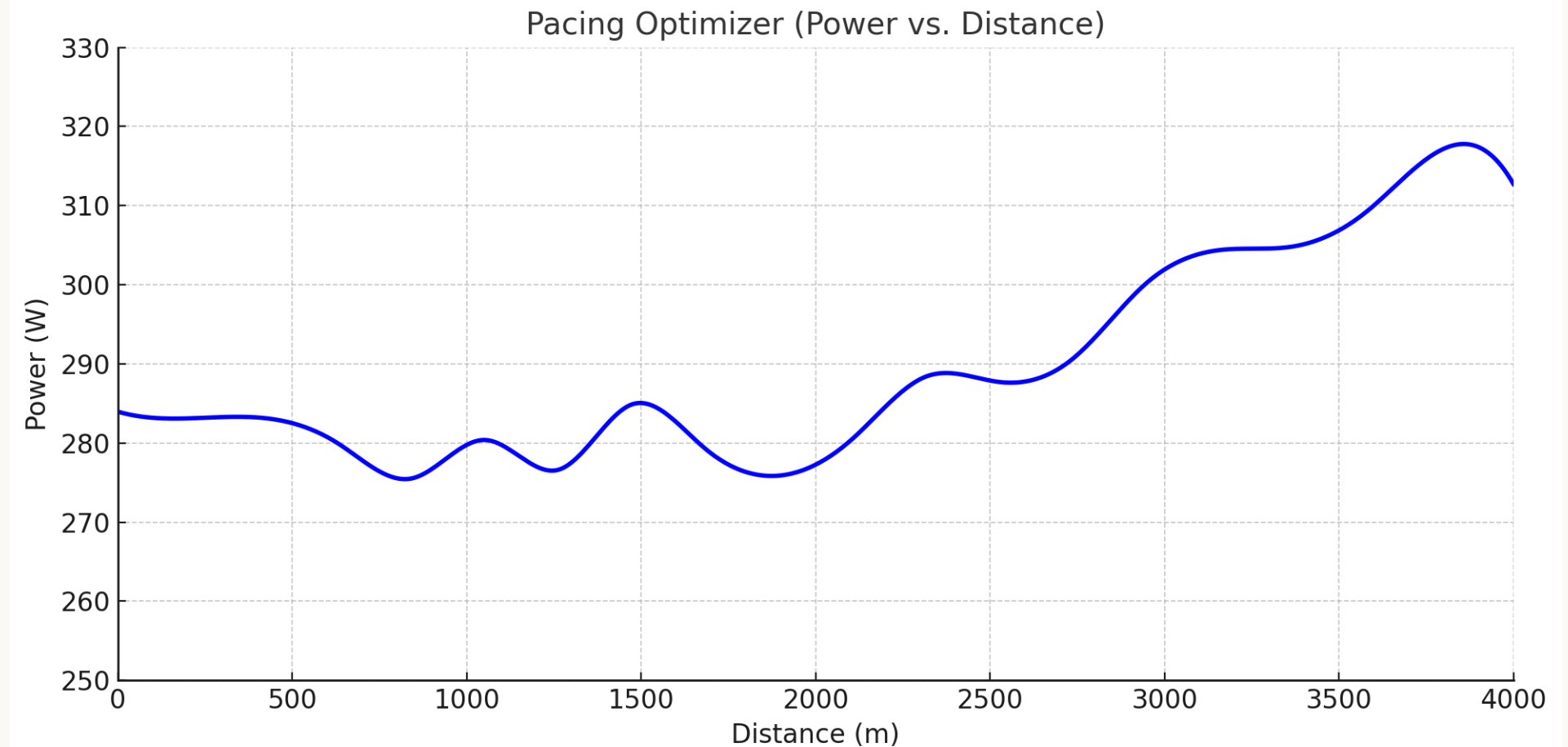
將路段圖拆分，針對每一個路段假設功率套用模型計算時間，最後得到總時間



XGBoost 模型+MCTS=pacing optimizer

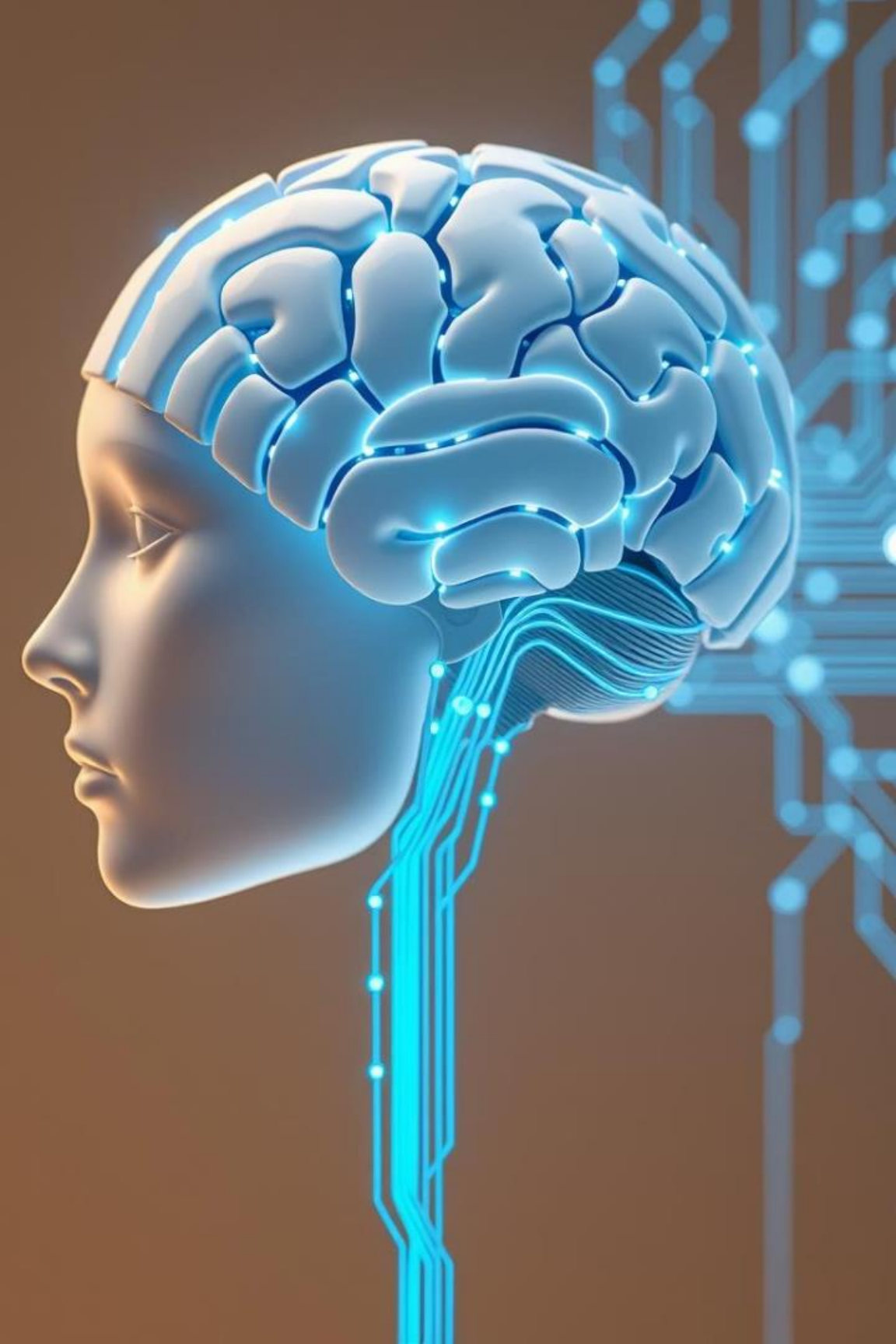
🏃 Best power plan with durations:

2.9 m: 850 W
6.0 m: 450 W
8.6 m: 250 W
11.4 m: 300 W
14.3 m: 1150 W
17.2 m: 500 W
19.8 m: 900 W
21.6 m: 550 W
23.2 m: 1000 W
24.5 m: 1200 W
26.3 m: 1050 W
28.4 m: 550 W
30.2 m: 1100 W
32.6 m: 450 W
35.2 m: 750 W
37.6 m: 900 W
40.2 m: 450 W



智能客服（智慧教練） 模型開發歷程與技術大綱

本簡報介紹智能克服模型的功能、技術挑戰
與最終解決方案。





功能目標與技術特點

功能定位

系統以自行車專家角色回覆訓練、補給、器材建議。

技術亮點

本地語言模型，純CPU環境，回應時間低於3秒。

階段	模型與技術架構	優點	困難與瓶頸
初期模型嘗試	LLaMA 2 7B + 微調 (1000筆 Guanaco 資料)	<ul style="list-style-type: none">- 熟悉模型- 可本地微調流程測試	<ul style="list-style-type: none">- 僅1000筆資料，生成效果模糊- 訓練需0.5小時，資源限制
RAG 架構實作	Mistral 7B + RAG Q4_K_M.gguf + llama.cpp + FAISS	<ul style="list-style-type: none">- 支援知識檢索- 有初步語意提升效果	<ul style="list-style-type: none">- 生成超過2分鐘- CPU限制明顯- Docker效能損耗- 模型與向量庫需每次重新載入
最終解法	Gemma 3 4B + Streaming Q4_K_M.gguf + llama.cpp	<ul style="list-style-type: none">- 輕量化模型，語言邏輯仍佳- 支援 Streaming- 可保留上下文語境	<ul style="list-style-type: none">- 仍需模型壓縮與記憶體優化

階段	模型與技術架構	優點	困難與瓶頸
初期模型嘗試	LLaMA 2 7B + 微調 (1000筆 Guanaco 資料)	<ul style="list-style-type: none">- 熟悉模型- 可本地微調流程測試	<ul style="list-style-type: none">- 僅1000筆資料，生成效果模糊- 訓練需0.5小時，資源限制
RAG 架構實作	Mistral 7B + RAG Q4_K_M.gguf + llama.cpp + FAISS	<ul style="list-style-type: none">- 支援知識檢索- 有初步語意提升效果	<ul style="list-style-type: none">- 生成超過2分鐘- CPU限制明顯- Docker效能損耗- 模型與向量庫需每次重新載入
最終解法	Gemma 3 4B + Streaming Q4_K_M.gguf + llama.cpp	<ul style="list-style-type: none">- 輕量化模型，語言邏輯仍佳- 支援 Streaming- 可保留上下文語境	<ul style="list-style-type: none">- 仍需模型壓縮與記憶體優化

```

# 載入模型 (建議使用較短 context, 加快速度)
llm = Llama(
    model_path=model_path,
    n_gpu_layers=0,      # 若 GPU 有啟用 CUDA, 加速用
    n_threads=8,        # 視 CPU 調整
    n_ctx=1024,         # 調整 context 長度 (速度影響大)
    n_batch=32,         # 建議 batch size, 視記憶體調整
    use_mlock=True,     # 鎖住模型至記憶體, 避免 swaps
    verbose=True
)

# 初始化聊天歷史 (開頭設置 system 指令)
chat_history = [{"role": "system", "content": "你是一位自行車專家。"}]

def chat_with_model_streaming(user_message):
    # 將使用者訊息加入聊天歷史
    chat_history.append({"role": "user", "content": user_message})

    # Streaming 輸出回應 (會逐步產生內容)
    response_text = ""
    for output in llm.create_chat_completion(messages=chat_history, stream=True):
        delta = output["choices"][0]["delta"]
        content = delta.get("content", "")
        print(content, end="", flush=True)
        response_text += content

    print() # 換行用
    chat_history.append({"role": "assistant", "content": response_text})
    return response_text

# 測試對話
user_input = "如何選擇適合的自行車?"
assistant_reply = chat_with_model_streaming(user_input)

```

```
# 初始化聊天歷史（開頭設置 system 指令）
chat_history = [{"role": "system", "content": "你是一位自行車專家。"}]

def chat_with_model_streaming(user_message):
    # 將使用者訊息加入聊天歷史
    chat_history.append({"role": "user", "content": user_message})
```




後續擴展與應用潛力

- 引入更多個人化生理數據（如心率變異、疲勞指標等）
- 整合即時路況與天氣數據，動態調整配速策略
- 開發更智能的訓練顧問，支持更多騎士需求場景
- 擴展至其他運動項目，打造跨領域運動AI輔助系統
- 系統優化兼容手機與微型裝置，提供隨時隨地適合戶外騎行和賽事現場的智能配速與訓練建議。

Q&A