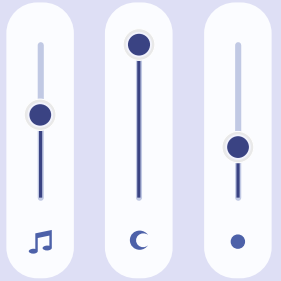
數據分析與預測
--應用與實踐



Download

簡報製作：王柏甯
案例協作：姜柏仰、陳祈恩、范庭郡

大綱

01

資料前處理

原始資料 → 特徵表示

XXX2資料集

標籤:

invoiceNo : 交易代碼
channel : 交易管道
customer : 買家
product : 商品
category : 商品種類
price : 商品定價
datetime : 交易時間
quantity : 購買數量
amount : 交易金額
category2 : 商品種類的子種類
cost : 成本

invoiceNo	channel	customer	product	category	price	datetime	quantity	amount	category2	cost
N1	s1	c1	p1	kind1	1980	2015/1/7 20:07	1	1692	sub1	931.39
N2	s1	c2	p2	kind1	1400	2015/1/18 19:56	1	1197	sub2	793.36
N3	s1	c3	p3	kind1	1600	2015/1/14 21:41	1	1368	sub1	846.72
N4	s1	c4	p3	kind1	1600	2015/1/7 19:12	1	1360	sub1	846.72
N5	s1	c5	p3	kind1	1600	2015/1/31 12:40	1	1368	sub1	846.72
N3	s1	c3	p2	kind1	1400	2015/1/14 21:43	1	1197	sub1	740.88
N6	s1	c6	p3	kind1	1600	2015/1/29 20:10	1	1216	sub1	846.72
N7	s1	c7	p3	kind1	1600	2015/1/17 14:26	1	1360	sub1	846.72
N8	s1	c7	p4	kind1	3250	2015/1/28 20:41	1	2776	sub1	1719.9
N9	s1	c8	p5	kind1	400	2015/1/17 20:06	1	342	sub3	188.16
N10	s1	c9	p6	kind1	2200	2015/1/13 18:26	1	1881	sub1	1164.24
N11	s1	c9	p7	kind1	1080	2015/1/29 21:56	1	540	sub1	508.03

XXX2資料集

總資料筆數:84008

invoiceNo :
共40632種交易代碼
【N1~N40632 】

customer :
共7774位不同的顧客
【c1~c7774】

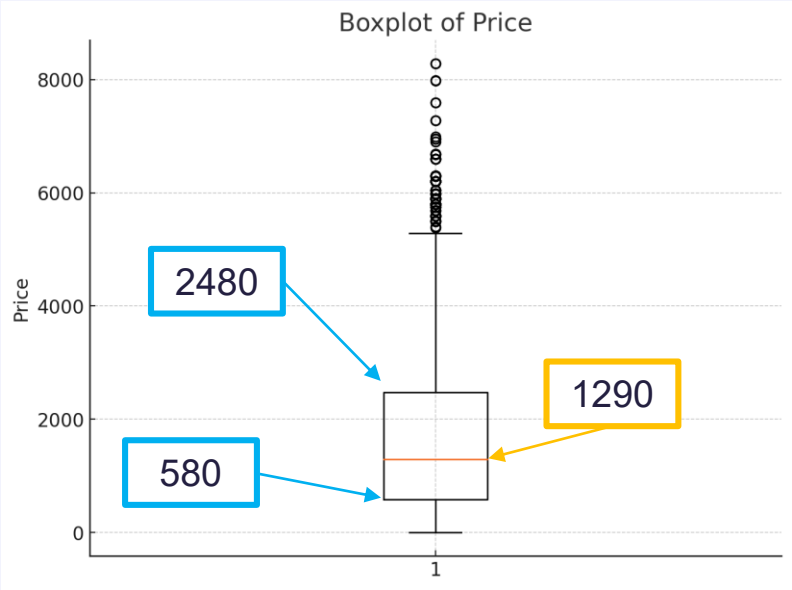
product :
共9013種不同的商品
【c1~c9013】

channel :
共5種交易管道
s1 : 20841 筆
s2 : 39882 筆
s3 : 10444筆
s4 : 7554 筆
s5 : 5287 筆

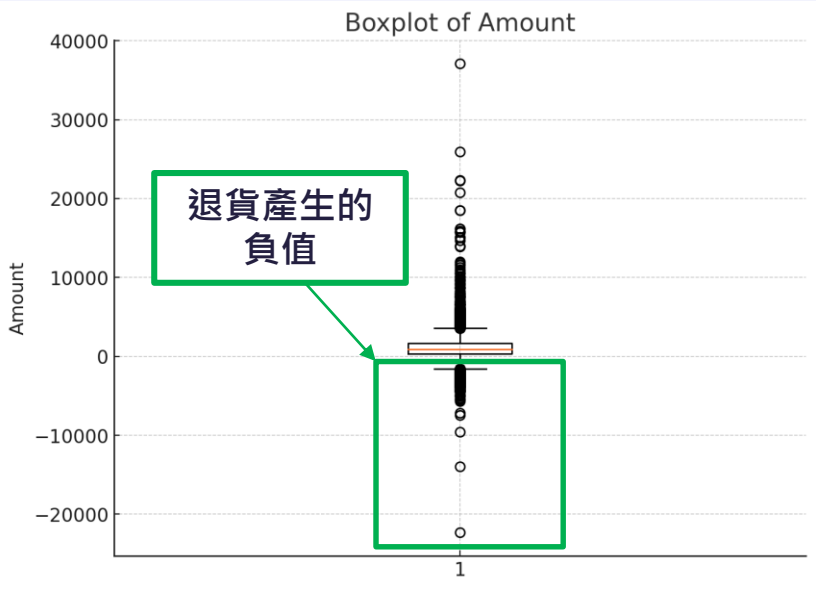
datetime :
從2015年1月~2017年12月
時間包含【年/月/日 時/分】

category :
共63種不同的商品種類
【kind1~kind63】

category 2:
共7種不同的商品子分類
【sub1~sub7】
並非每個種類都有7種子分類



Price的四分位距圖：
【0~8280】



Amount的四分位距圖：
【-22,320~37100】

XXX1資料集

總資料筆數:33605

invoiceNo :
共22818種交易代碼

customer :
共5638位不同的顧客

product :
共6987種不同的商品

channel :
共5種交易管道
s1 : 8383筆
s2 : 16084筆
s3 : 2072筆
s4 : 4128 筆
s5 : 2938 筆

datetime :
從2015年1月~2017年12月
時間包含【年/月/日 時/分】

category :
共61種不同的商品種類
【kind1~kind63】

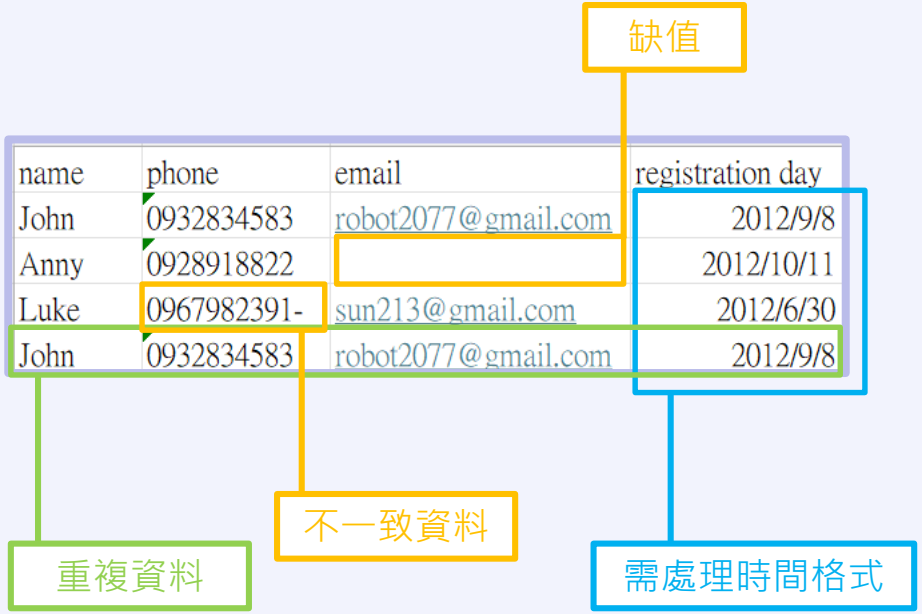
category 2:
共7種不同的商品子分類
【sub1~sub7】
並非每個種類都有7種子分類

資料清洗與缺值處理

資料清洗

識別和修正或移除不完整、不準確或不一致的資料，讓其適合用於分析、模型訓練和決策。

常見任務包含:
處理缺值
處理極端值
處理重複資料
處理不一致資料
標準化和正規化
轉換資料型態



轉換日期格式與檢查重複行數量示範

```
import pandas as pd

file_path = 'C:/data/XXX2.csv’
data = pd.read_csv(file_path)

# 轉換日期時間格式
print("Before:",data['datetime'].dtype)
data['datetime'] = pd.to_datetime(data['datetime'])
print("After:",data['datetime'].dtype)

# 計算重複行的數量
num_duplicates = data.duplicated().sum()
# 顯示重複行的數量
print(f"Number of duplicate rows: {num_duplicates}")
```

資料清洗與缺值處理

缺值處理

大多數模型並沒有能力處理缺失值，需專門對缺失值處理，
來提升預測品質。

缺失值原因:

- 非隨機缺失:
  如：問卷調查時，薪水資訊較為敏感，有些人會選擇不填寫。
- 隨機缺失:
  如：在資料傳送到後端或者資料庫時，因設計問題造成資料遺失。

缺值處理方式:

- **刪除**：刪除有缺失的欄位(column)，或刪除該筆資料(row)。
- **補值**：對缺值進行合理的補值。如:統計數值補值、近似資料補值。

product	price	cost
p6579	8280	4057.2
p6942	8280	4057.2
p3890	6680	29458.8
p4069	2850	NA
p4070	2850	NA

刪除
有缺值資料

product	price	cost
p6579	8280	4057.2
p6942	8280	4057.2
p3890	6680	29458.8

填入
平均值

product	price	cost
p6579	8280	4057.2
p6942	8280	4057.2
p3890	6680	29458.8
p4069	2850	760.4357
p4070	2850	760.4357

缺值處理示範

```
#檢查有多少缺值
print( data.isnull().sum() )
```

```
#刪除有缺值的資料筆
data.dropna()
```

```
#填補缺值，填入平均值(中位數則換成median)
data['cost’]. fillna(data['cost’].mean(), inplace=True)
```

資料串接與特徵轉換

資料串接

將多個資料集或資料框合併成一個。

pd.merge:

根據一個或多個鍵（key）將兩 DataFrame 合併。

on--指定用來匹配的row，合併不同來源、
　　但有**共同欄位**的資料。
how--指定合併方式，
　　如: inner**(默認)**、outer、left、right

pd.concat:

將多個DataFrame沿著特定的軸（row或column）進行串接。

axis=0 **(默認)**--用於垂直串接（沿著row），
　　DataFrame將向下擴展
axis=1-- 用於水平串接（沿著列column），
　　DataFrame將並排擺放。

資料串接示範

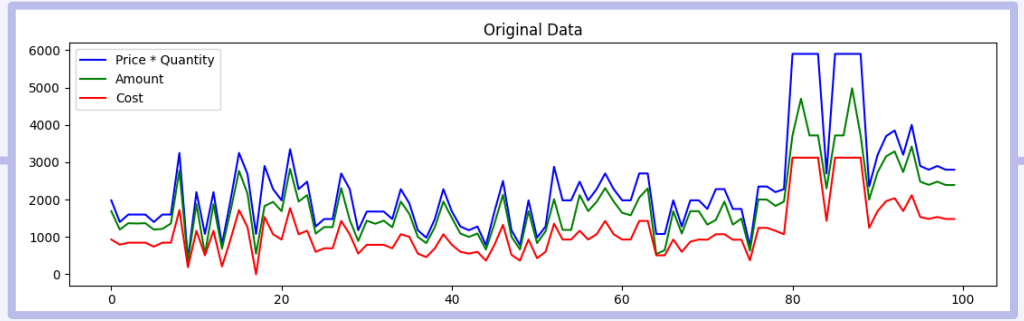
```
# 假設我們有兩個數據部分：銷售數據和產品信息
# 從原始數據集中提取銷售數據部分
sales_data = data[['invoiceNo', 'customer', 'product', 'amount']]

# 從原始數據集中提取產品信息部分
product_data = data[['product', 'category', 'cost']]

# 根據 'product' 欄位進行橫向串接
merged_data = pd.merge(sales_data, product_data, on='product', how='left’)


print(merged_data.head())
```

資料串接與特徵轉換

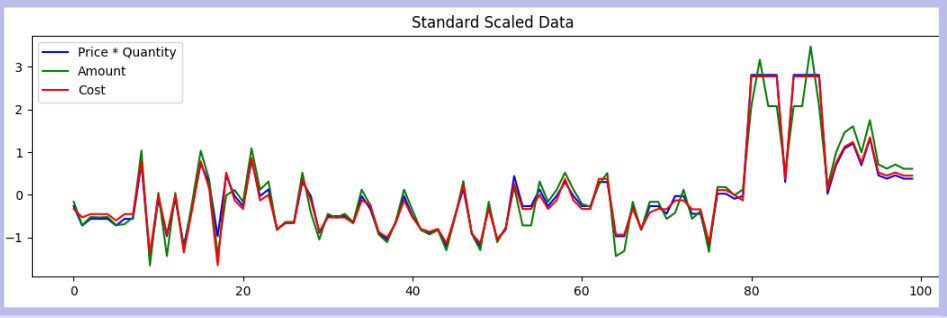
標準化:
StandardScaler()
將數據縮放到具有零均值和單位標準差的分佈。

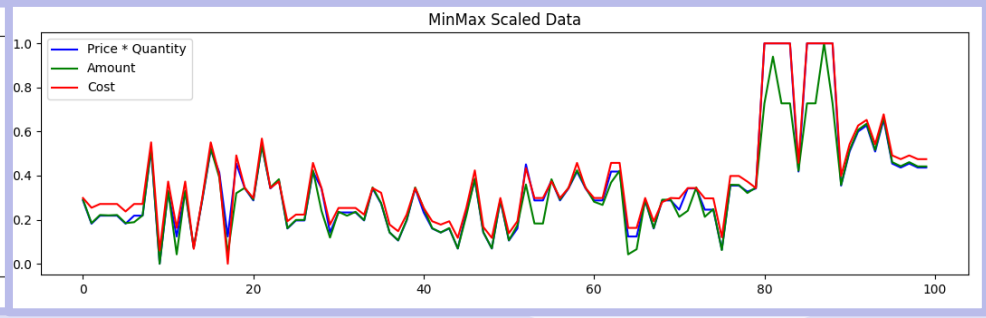
正規化:
MinMaxScaler()
將數據縮放到特定範圍（通常是 0 到 1）。

標準化



正規化





標準化與正規化示範

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# 標準化與正規化數值欄位
numerical_cols = ['price', 'quantity', 'amount', 'cost’]

# 標準化：資料具有零均值和單位方差
scaler = StandardScaler()
data[numerical_cols] = scaler.fit_transform(data[numerical_cols])

# 正規化：將資料縮放到 [0, 1] 範圍內
min_max_scaler = MinMaxScaler()
data[numerical_cols] = min_max_scaler.fit_transform(data[numerical_cols])
```

特徵轉換:日期時間特徵提取

轉換日期格式:

pd.to_datetime :

將日期和時間資料轉換轉換為，可以用於時間序列分析的 datetimeIndex 或 datetime64[ns] (默認)資料類型。

- 可將字串轉換為日期時間
  如 : date= '2023-08-17'
- 可處理不同格式的日期
  如 : dates = ['2023-08-17', '17/08/2023', '08-17-2023']
- 可將 Unix 時間戳記轉換為日期時間:
  如 : timestamps = [1628870400, 1628956800]

日期時間特徵提取:

dt.year ---------- 年份
dt.month -------- 月份
dt.day ---------- 天
dt.weekday ----- 星期幾
dt.hour ---------- 小時
dt.minute ------- 分鐘
dt.second ------- 秒

日期時間特徵提取示範

```
# 轉換日期時間格式
data['datetime'] = pd.to_datetime(data['datetime'])
data = data.sort_values(by='datetime')

# 提取日期時間特徵
data['year'] = data['datetime'].dt.year
data['month'] = data['datetime'].dt.month
data['day'] = data['datetime'].dt.day
data['weekday'] = data['datetime'].dt.weekday
data['hour'] = data['datetime'].dt.hour
```

平衡學習與資料增強

平衡學習

處理不平衡數據集的方法。當數據集中不同類別之間的數據量差異很大時，模型可能會傾向於對數量較多的類別進行優先預測，從而導致對少數類別的預測不準確。

常見方法包含:
重抽樣方法（Resampling Methods ）
-----過抽樣（Oversampling ）
-----欠抽樣（Undersampling ）
加權方法（Weighting Method ）

重抽樣方法

過抽樣（Oversampling ）：增加少數類別的樣本數量，以平衡類別分佈。
欠抽樣（Undersampling ）：減少多數類別的樣本數量，以平衡類別分佈。

資料增強

影音：旋轉、縮放、平移、剪裁、改變亮度/音量、模糊、混合、加噪...
文本：隨機編輯、重點摘錄、增加描述、字母互換、混合、加噪...

過抽樣示範

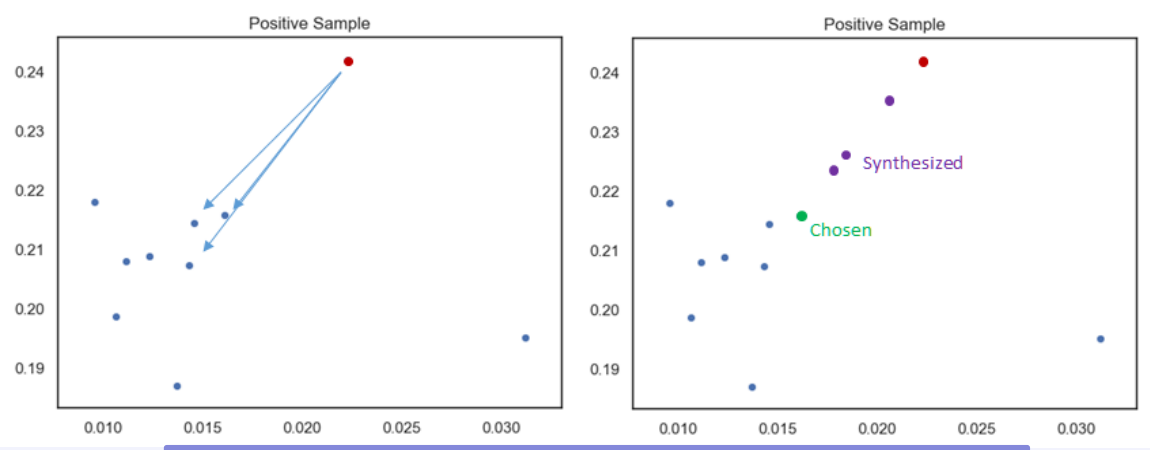
```
from imblearn.over_sampling import RandomOverSampler
X = data.drop(columns=['channel'])  # 特徵集
y = data['channel']  # 目標變量

# 創建 RandomOverSampler 對象
ros = RandomOverSampler(random_state=42)

# 對數據進行過採樣
X_resampled, y_resampled = ros.fit_resample(X, y)

# 檢查平衡後的類別分佈
balanced_class_counts = pd.Series(y_resampled).value_counts()
print("Balanced class distribution:")
print(balanced_class_counts)
```

類別平衡：合成少數類過採樣技術SMOTE

SMOTE：
在少數樣本位置近的地方，
人工合成一些樣本。

1. 設定採樣倍率 N，設定每個樣本需要生成幾個合成樣本。
2. 設定一個近鄰值 K，針對該樣本找出 K 個最近鄰樣本並從中隨機選一個。
3. 最後以該公式創造樣本:

$$x_{new} = x_{chosen} + (x_{nearest} - x_{chosen}) * \delta \; ; \; \delta[0,1]$$



三個近鄰點中隨機選取一個，
並透過公式去合成 3 個樣本點。

x_{chosen}：少數類別中的一個原始樣本。
$x_{nearest}$：與x_{chosen}最近少數類別樣本之一。
δ：一個隨機生成的係數，範圍在0到1之間，決定了合成樣本在 x_{chosen} 和 $x_{nearest}$之間的位置。

Download:

類別平衡：非數值資料SMOTE

SMOTE-NC

適用包含連續型(Continuous)及名稱型(Nominal)屬性的資料，前者用SMOTE進行線性插值，後者是字串型別無法計算，相同名稱予以保留，不同名稱則隨機擇一或選擇少類別比較常見者。

資料增強

運用同義詞置換、隨機編輯、重點摘錄、混合...等文本資料增強技術。

向量化 + SMOTE

透過文本模型(如Word2Vect或BERT)將字串屬性都轉換為字嵌入向量(word embedding)，向量可以進行數值計算，之後就可以直接運用SMOTE進行線性插值。

KNN多數決補缺值或生成對抗網路(GAN)

前者是利用字串屬性的相近程度(proximity)量測可以找到最近鄰居，取近鄰的多數決合成樣本。

02

模型面面觀

訓練模型 → 測試評估

模型選用與評估指標

模型選用:
不同類型的數據、特性和任務目標，有其合適的模型與實驗設計方法。在有些情況下，使用最簡單的模型是最好的選擇，有助於避免過擬合。

問題類型:
分類問題：將數據點分配到特定的類別中。
如，垃圾郵件分類
回歸問題：目標是預測連續的數值。
如，房價預測

問題複雜性：
簡單問題可能使用線性回歸、決策樹等效果更好。
複雜問題可能需要用到深度學習模型。

評估指標:
模型的評估指標用來衡量模型在特定任務中的性能和表現。合適的評估指標能驗證模型的有效性。

分類問題常見評估指標:
準確率 (Accuracy)
精確率 (Precision)
召回率 (Recall)
F1-score

回歸問題常見評估指標:
均方誤差(MSE)
均方根誤差(RMSE)
決定係數(R-squared)

回歸任務常見評估指標

◆ 決定係數R squared (R^2):
計算預測值與實際值的偏離比例。

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

SS_{res} : $\sum_{i=1}^{n}(y_i - f_i)^2$
SS_{tot} : $\sum_{i=1}^{n}(y_i - \bar{y})^2$

n : 樣本的總數, y_i : 第 i 個實際值,
$\bar{y}$: 所有 y_i 的平均值, f_i : 第 i 個預測值

R^2 範圍: [0~1]

◆ MSE (Mean Squared Error) :
計算的是預測值與實際值之間誤差的平方的平均值。

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

n : 樣本的總數, y_i : 第 i 個實際值, $\hat{y}_i$: 第 i 個預測值
MSE範圍: 0~無限大

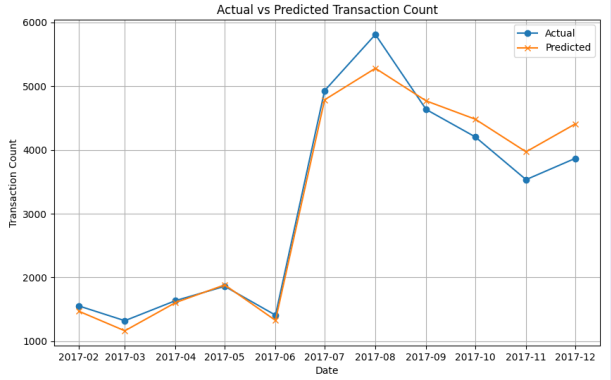
◆ RMSE (Root Mean Squared Error) :
計算的是預測值與實際值之間誤差的平方的平均值。

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

n : 樣本的總數, y_i : 第 i 個實際值, $\hat{y}_i$: 第 i 個預測值
MSE範圍: 0~無限大

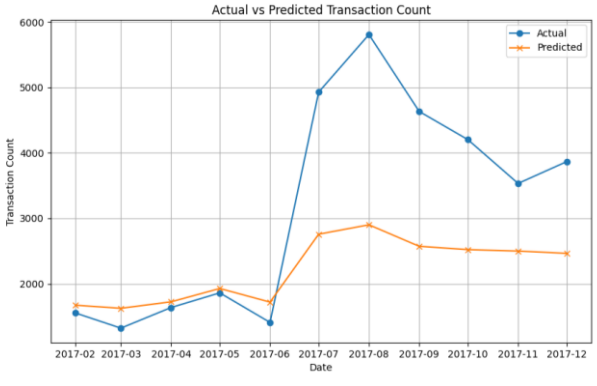
保留了MSE放大大誤差的特性，但通過取平方根，使得誤差度量與原始數據的尺度一致。

LinearRegression 交易量預測



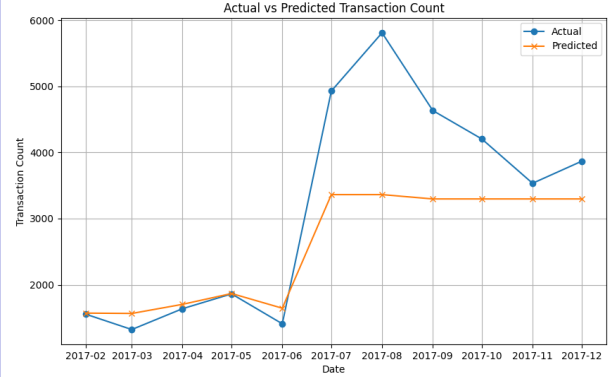
MSE : 83427.395
RMSE : 288.838
R squared: 0.9661

RandomForest Regressor 交易量預測



MSE : 2134780.776
RMSE : 1461.089
R squared: 0.1324

XGBoost 交易量預測



MSE : 1049422.4
RMSE : 1024.413
R squared: 0.5735

不同複雜度的模型，有其適配的任務。

在複雜的任務上，使用太簡單的模型，可能無法捕捉數據中的模式。

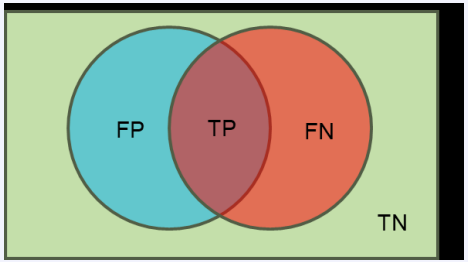
在簡單的任務上，使用太複雜的模型，可能因太注意數據中的噪聲。

分類任務常見評估指標

實際正樣本
其餘空間:實際負樣本
預測正樣本
其餘空間:預測負樣本

◆ 混淆矩陣(Confusion Matrix)

正樣本(Positive)和負樣本(Negative)，與預測正確(True)和錯誤(False)四種情況所形成的矩陣



$$Accuracy = (TP + TN) / (TP + FP + FN + TN)$$

True Positive(TP)：實際是正樣本且預測為正樣本 False Positive(FP)：實際是負樣本但誤判為正樣本 False Negative(FN)：實際是正樣本但誤判為負樣本 True Negative(TN)：實際是負樣本且預測為負樣本

- Recall(召回率) = TP/(TP+FN):所有實際正樣本當中，能夠預測多少正樣本的比例
- Precision(準確率) = TP/(TP+FP):所有預測為正樣本中，有多少為實際正樣本

分類任務常見評估指標

◆ 多分類混淆矩陣(Confusion Matrix)

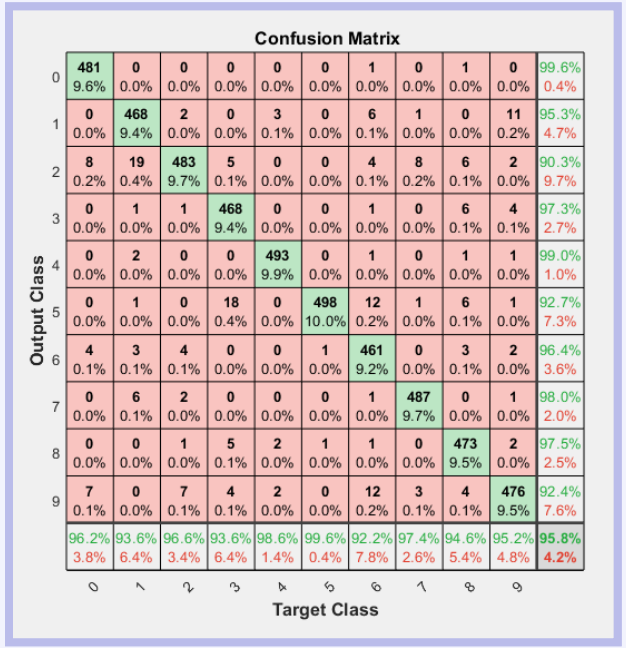
橫軸: 真實標籤
縱軸: 預測標籤

- 對角線(綠色): 模型正確預測的數量。

如:第一行第一列，表示模型正確預測了 481 個實際類別為 0 的資料筆。

- 非對角線(紅色):錯誤分類的數量。

如:第一行第二列， 表示模型將實際類別為 1 的樣本錯誤地分類為 0 的數量為 0。



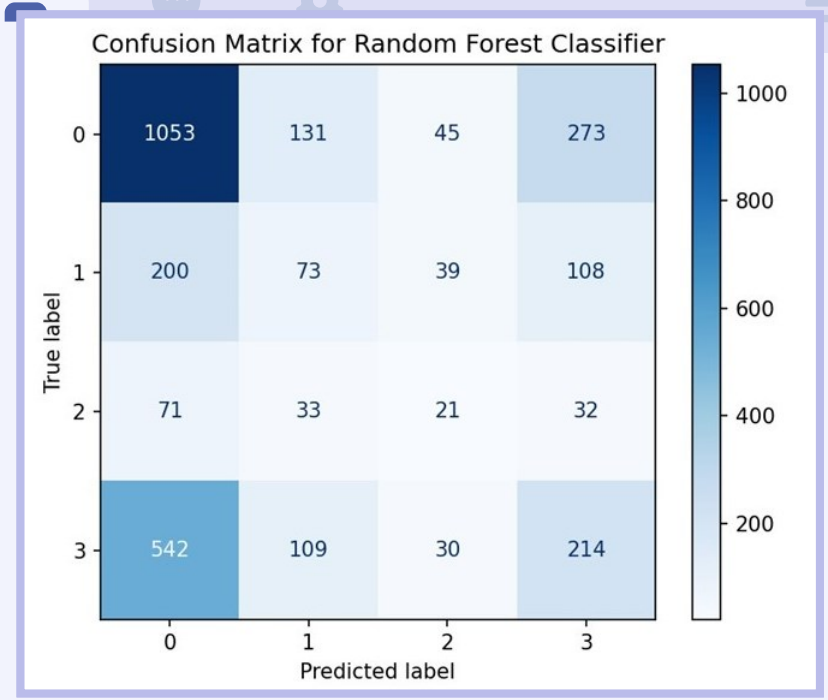
預測目標: 四種消費力顧客(由k-means分群)

類別	0	1	2	3
數量	7816	4205	2082	766

判斷元素: 每月消費次數與每月消費金額
模型: 隨機森林

Accuracy: 0.457

```
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.70      0.63      1502
           1       0.21      0.17      0.19       420
           2       0.16      0.13      0.14       157
           3       0.34      0.24      0.28       895

    accuracy                           0.46      2974
   macro avg       0.32      0.31      0.31      2974
weighted avg       0.43      0.46      0.43      2974
```



除第0群以外，其他群的準確率都不足40%甚至不足20%。
單個月的消費次數與總金額，不足以判斷顧客的最終消費力。

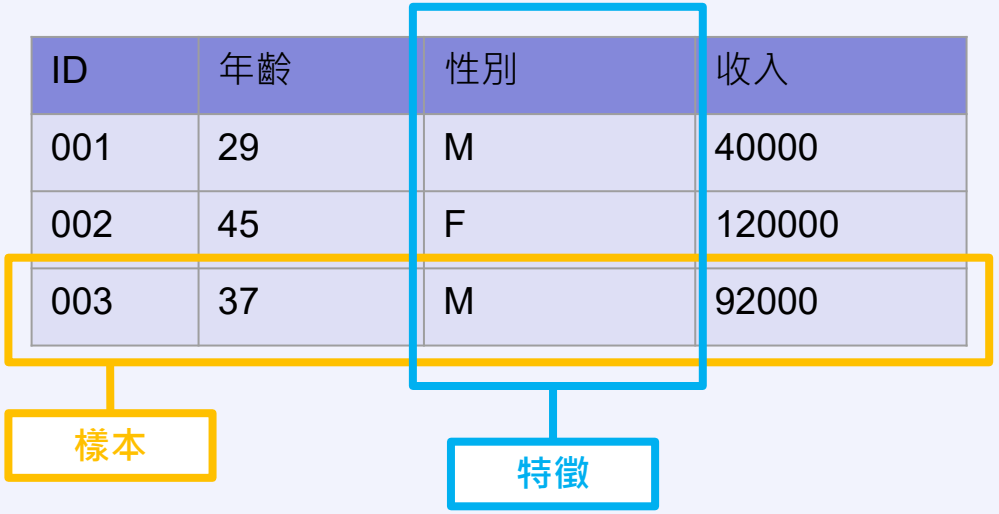
Download:

表格資料與集成學習

表格資料（Tabular Data）：
表格資料通常以行列結構呈現，像是電子表格或數據庫中的數據表格。

每一行(row)代表一個**樣本或記錄**，
每一列(column)代表一個**特徵或屬性**。

ID	年齡	性別	收入
001	29	M	40000
002	45	F	120000
003	37	M	92000



表格資料與集成學習

集成學習（Ensemble Learning）：
結合多個基學習器（Base Learners）的預測結果來提升整體模型性能的技術。

基學習器（Base Learner）：
構建集成模型（Ensemble Model）的基本單元，即每個單獨訓練的機器學習模型。
基學習器可以是任何機器學習算法，
例如決策樹、線性回歸、神經網絡等。

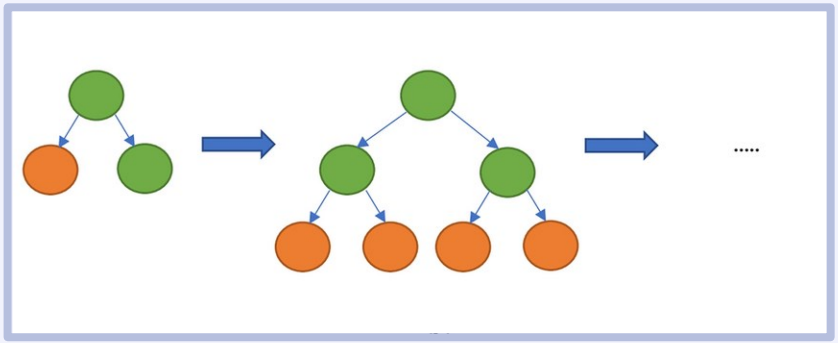
集成學習類型:

裝袋法（Bagging, Bootstrap Aggregating）：
通過對訓練集多次隨機重採樣，生成多個訓練子集，對每個子集訓練一個基學習器，最終對這些基學習器的預測結果進行**平均（回歸）**或**投票（分類）**。
如：隨機森林（Random Forest）

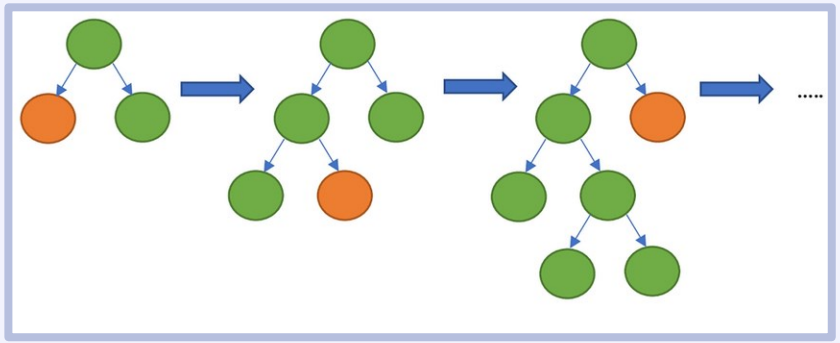
提升法（Boosting）：
逐步訓練基學習器，每個基學習器都試圖修正前一個的錯誤。最終的模型是這些基學習器的**加權和**。
如： XGBoost、LightGBM

表格資料與集成學習

XGBoost:
基於水平樹增長（Level-wise Growth）：
構建樹時每一層都會平衡地分裂，使樹更加平衡，深度也更淺。

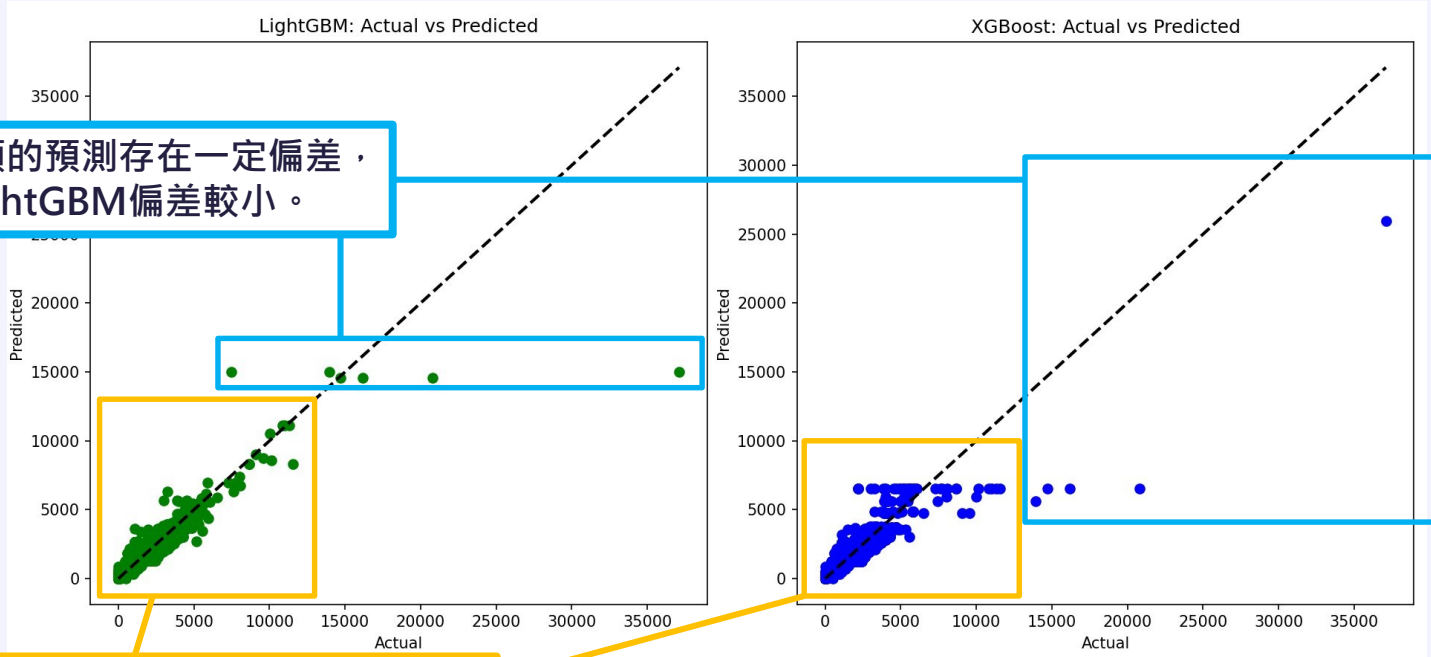


LightGBM:
基於葉子增長（Leaf-wise Growth）：
每次選擇損失最小的葉子分裂，可以更快地減少損失，但也可能導致樹變得更不平衡，從而更容易過擬合。



表格資料與集成學習

從銷售管道、商品定價與購買數量預測售價



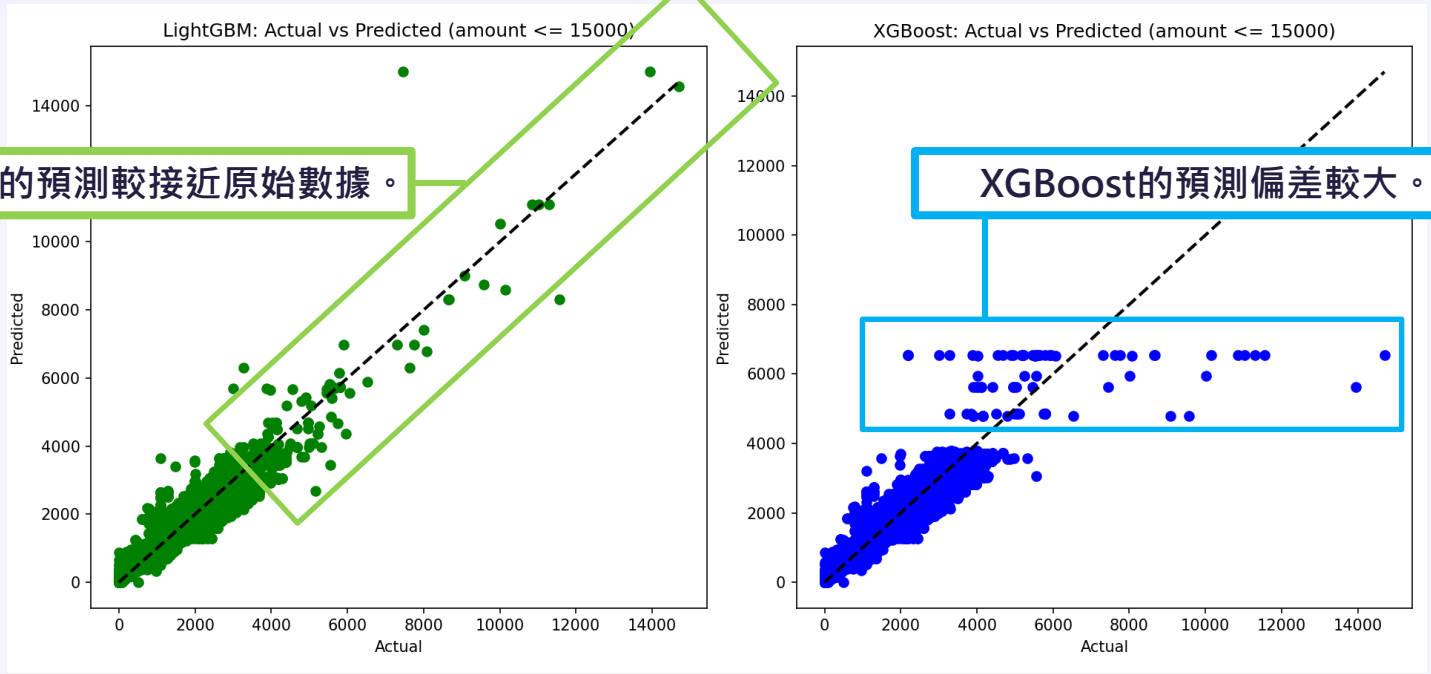
對於高金額的預測存在一定偏差，
不過LightGBM偏差較小。

大部分的預測點聚集在虛線附近，
在低金額區間LightGBM的預測較準確。

Download:

表格資料與集成學習

從銷售管道、商品定價與購買數量預測售價 【原售價小於15000】



LightGBM的預測較接近原始數據。

XGBoost的預測偏差較大。

序列資料與深度學習

序列資料（Sequential Data）：
序列資料是一種數據點按時間或順序排列的數據類型，每個數據點依賴於前後的數據點。

時間序列資料:

時間: [2024-08-01, 2024-08-02, 2024-08-03]
價格: [100, 105, 110]

文本資料:

句子: "今天天氣很好"

序列資料與深度學習

欄位編碼

One-Hot Encoding：

新增N個Columns，每個Columns用0與1表，示原本的Feature是否為該類別。

	s1	s2	…	s5
N19	1	0	…	0
N1194	0	1	…	0
……	…	…	…	…
N22211	0	0	…	1

欄位編碼

LabelEncoder：

把所有類別，轉換成0~N-1的數值。

	channel
N19	1
N1194	2
……	…
N22211	5

欄位編碼

Frequency Encoding ：

每個類別出現的數量，當成其數值。

	channel
N19	20841
N1194	39882
……	…
N22211	7554

欄位編碼示範

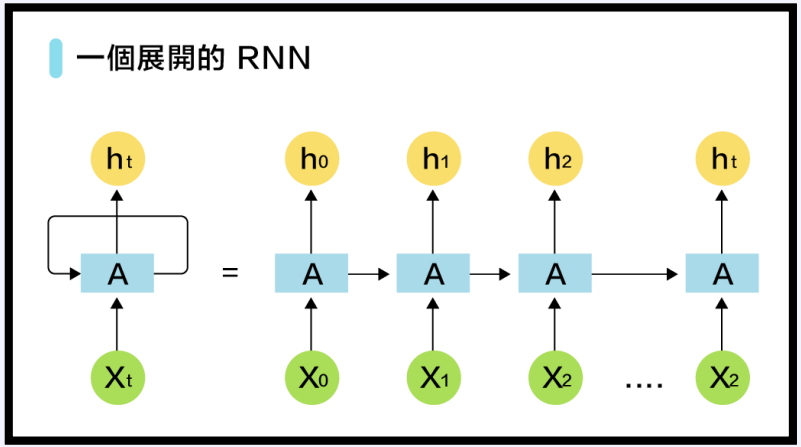
```
# 使用One-Hot Encoding
data = pd.get_dummies(data, columns=['channel’])
from sklearn.preprocessing import LabelEncoder

# 使用label_encoder
label_encoder = LabelEncoder()
data['category_encoded'] = label_encoder.fit_transform(data['category’])

# 使用Frequency Encoding
customer_freq = data['customer'].value_counts().to_dict()
data['customer_freq'] = data['customer'].map(customer_freq)
```

序列資料與深度學習

深度學習（deep learning）：
利用多層神經網絡模擬人腦的學習過程，不需手動設計的特徵，而是能夠自動學習原始數據的複雜特徵和模式。

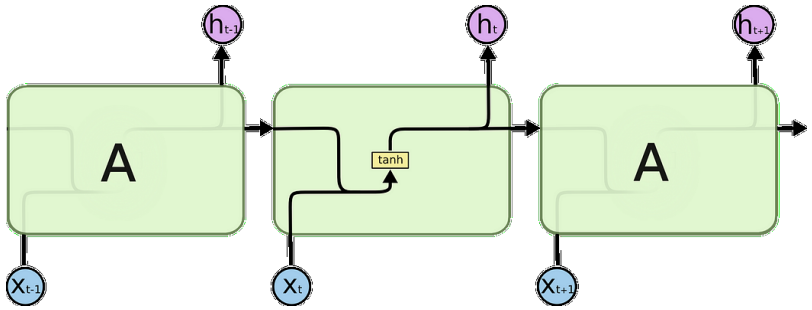
RNN:
專門用於處理序列數據的神經網路結構。RNNs 能夠保留和利用前一時刻的信息，這使其在處理時間序列數據和自然語言處理（NLP）任務時具有顯著的優勢。

單純的RNN因無法處理隨著遞歸，權重指數級爆炸或梯度消失問題，且因建模能力有限，難以捕捉長期時間關聯；而LSTM可以很好解決這個問題。

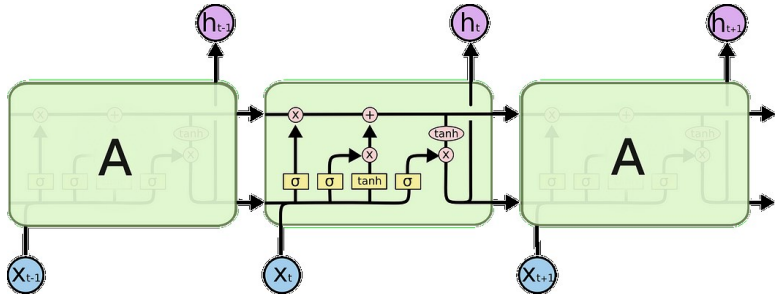


序列資料與深度學習

LSTM（Long Short-Term Memory）：
一種RNN，
適合於處理和預測時間序列中間隔和延遲非常長的重要事件。
作為非線性模型，LSTM可作為複雜的非線性單元用於構造更大型深度神經網路。



標準的 RNN 架構，包含一個激勵函數的方程式的神經層。(如: 雙曲函數 tanh)

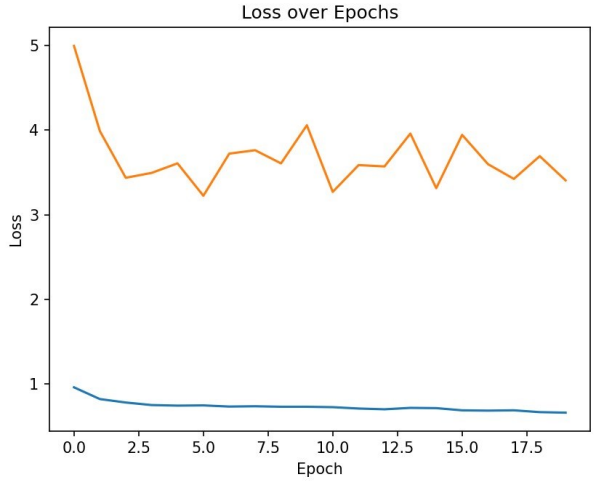


LSTMs 的內部單元由四層不同函數組成，彼此間會相互影響。

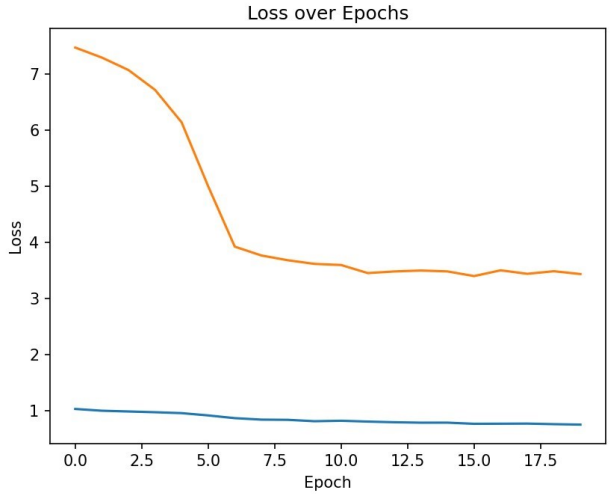
學習率影響

過高的學習率，可能導致訓練過程中的損失不穩定，模型無法穩定收斂。
過少的學習率，可能導致訓練過程的損失下降太慢，模型需要很久才能收斂。

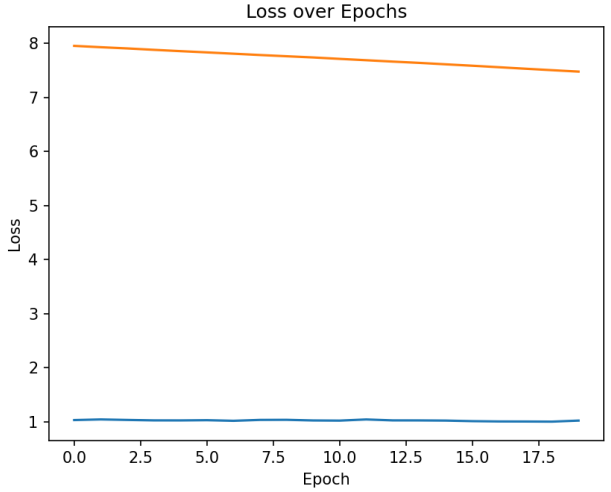
Lr = 0.001



Lr = 0.0001



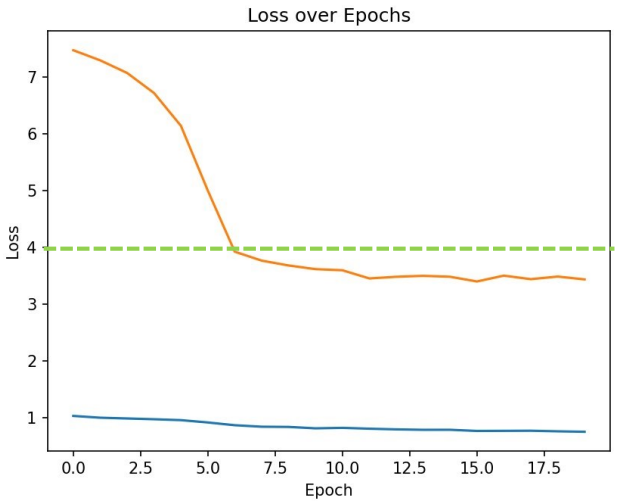
Lr = 0.00001



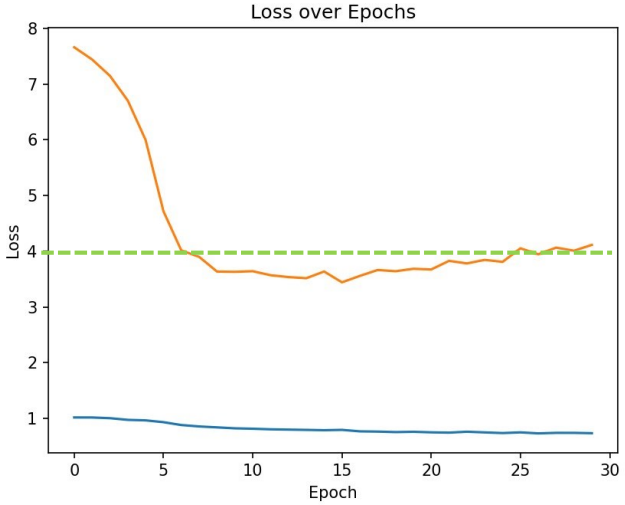
訓練次數影響

過多的訓練次數，可能會導致過擬合，甚至使驗證集的損失提高。
過少的訓練次數，可能會導致欠擬合，損失還未下降到底就結束訓練。

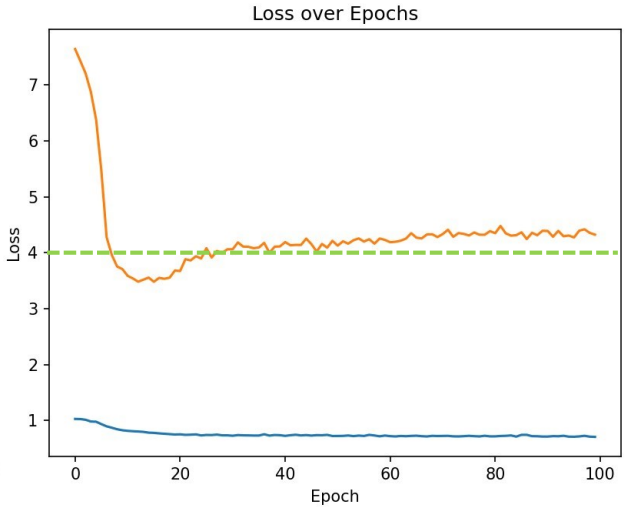
訓練20次



訓練30次



訓練100次



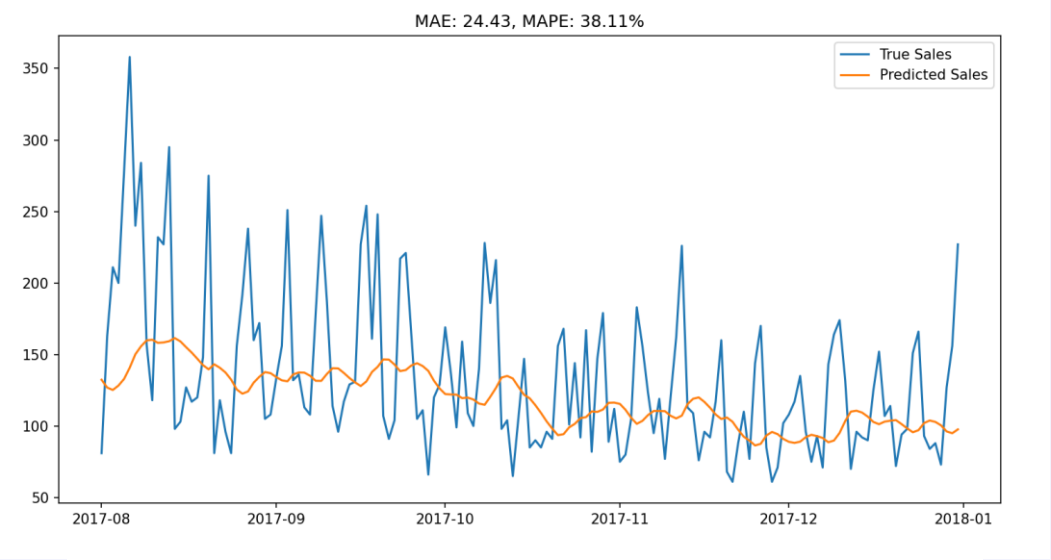
◆ MAE (Mean Absolute Error) :
預測值與真實值之間
絕對誤差的平均值。

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

◆ MAPE (Mean Absolute Percentage Error) :
預測值與真實值之間
絕對百分比誤差的平均值。

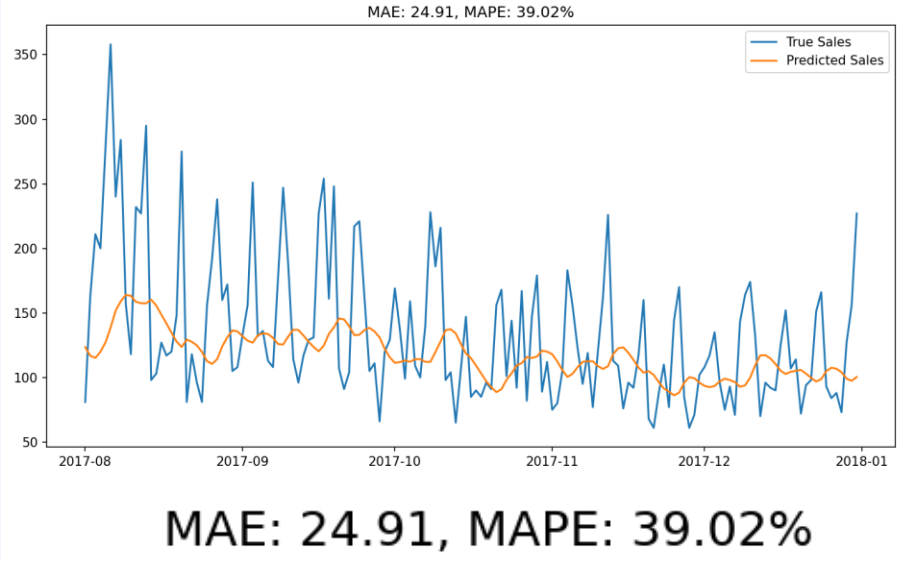
$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

銷售量預測
用九十天的銷售量預測下一天。



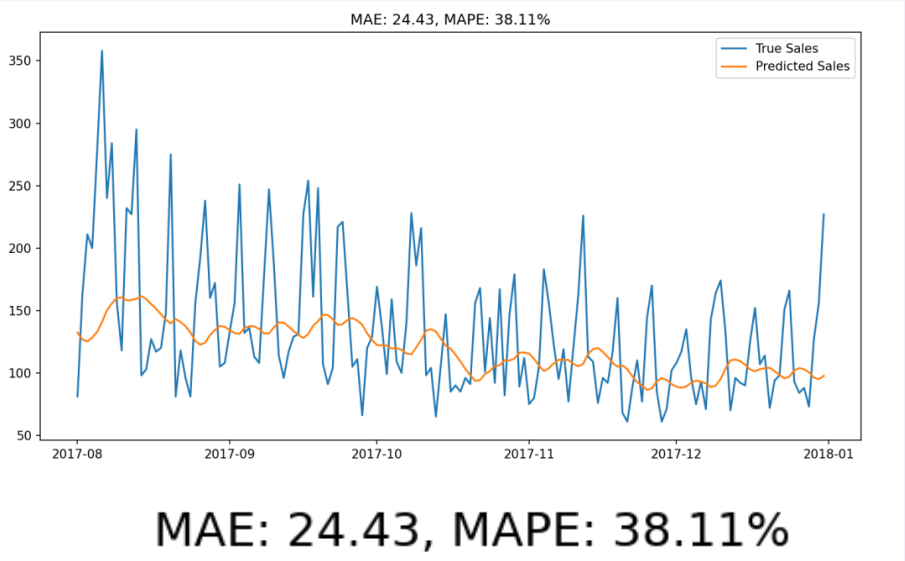
MAE: 24.43, MAPE: 38.11%

用三十天的銷售量預測下一天。



MAE: 24.91, MAPE: 39.02%

用九十天的銷售量預測下一天。



MAE: 24.43, MAPE: 38.11%

Download:

03

解題交流會

分群分類 → 交叉驗證

解題思路

訓練集有13種不同的介電材料電阻值，
每種電阻值處在4種電壓設定 (a、b、c、d)條件下，
重複十次實驗並記錄4000次電流變化。

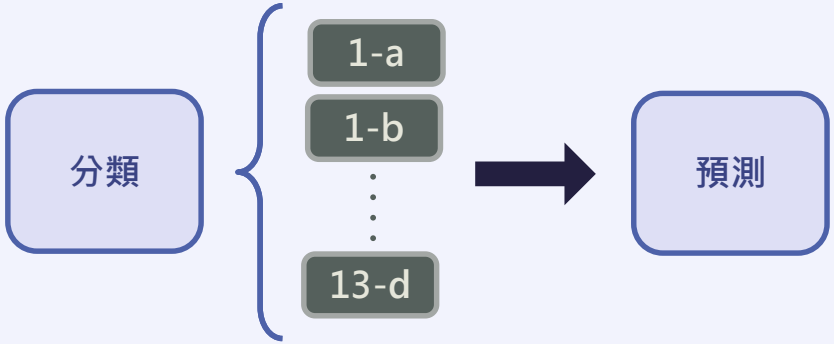
測試集將提供某種電阻值在a、b、c、d其中一種電壓設定的條件下，前50筆的電流數據，並預測後3950筆。

■ 面臨問題：

已知資訊較少(電阻值未知、電壓未知)。
同材料、同環境下初值變化大。
可用預測數據只佔原有數據的1/80。

■ 目前解題思路：

先將已知數據進行分群或分類，再預測後續數據。



分類方法設計

方法(一)

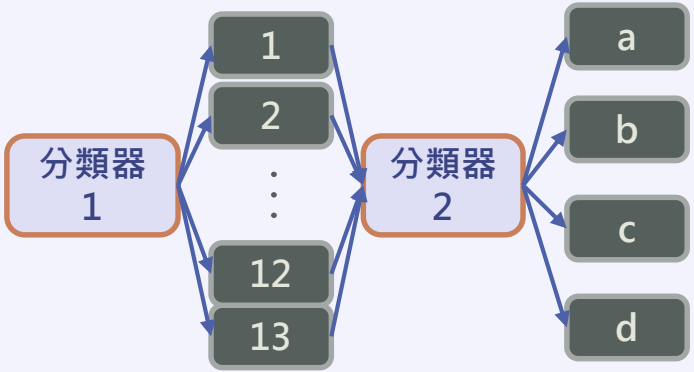
1.先訓練分類器1分類13種材質
2.在訓練分類器2分類4種電壓設定

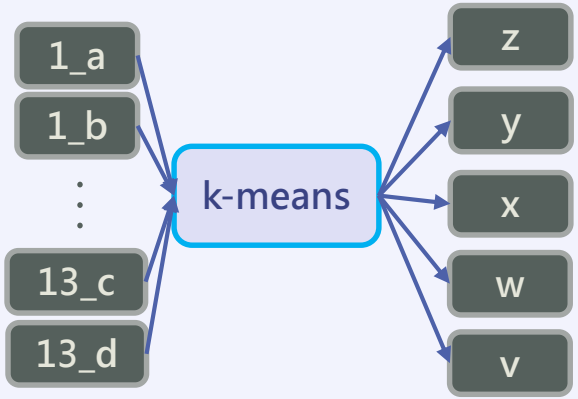
方法(二)

1.將同材質、同電壓分成同一類
2.將同材質的十次實驗視為一組資料

方法(三)

1.利用k-means重新區分類別
2.用分類器分類新類別

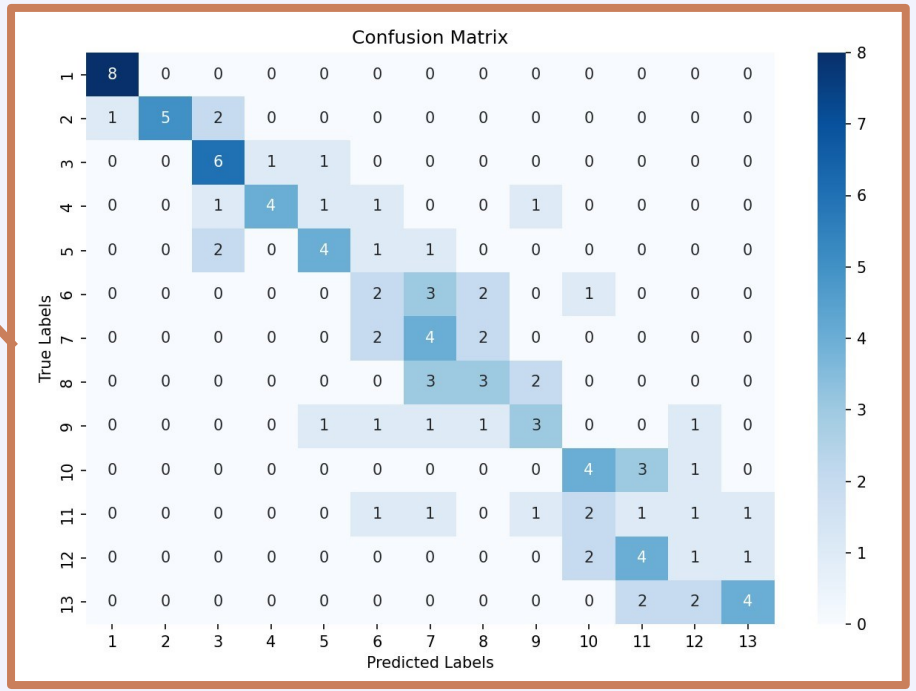




分類實作_方法(一)

任務:
訓練隨機森林分辨不同材質

準確率:0.47 【過低】

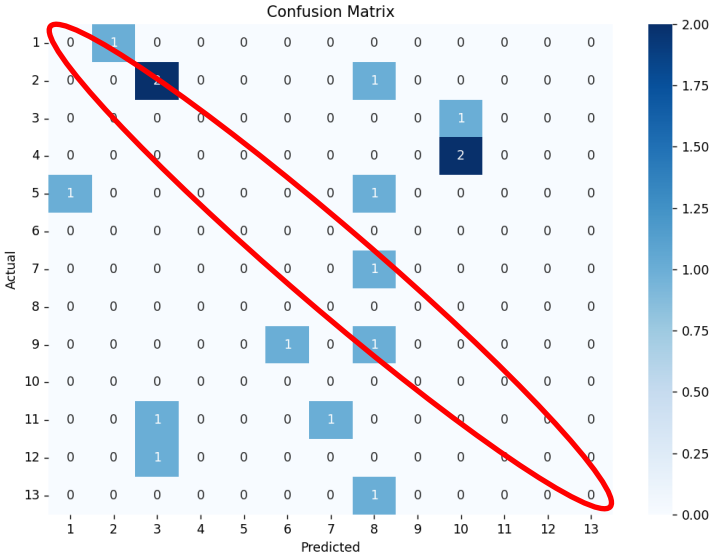


遭遇問題:
隨機森林無法找到同材質的特性，
嘗試不同特徵，準確率也難以上升

分類實作_方法(二)

任務:
將同材質的十次實驗視為一組資料，
共52組資料。
再訓練隨機森林分類。

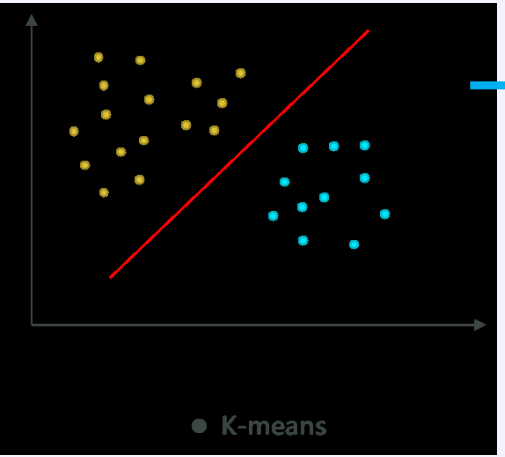
準確率: 0

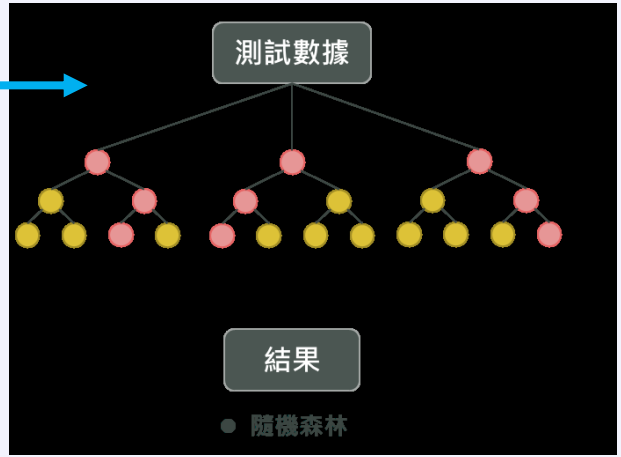


遭遇問題:
每一類可訓練數據太少(訓練+驗證共4筆)
沒有足夠多的數據讓模型發現模式。

分類實作_方法(三)

任務:
由K-means根據數據特性決定新分類，
再讓隨機森林預測。





分類六群情況下:
1 : 242筆【集中在環境a與b】
2 : 20筆
3 : 81筆
4 : 7筆
5 : 28筆
6 : 142筆

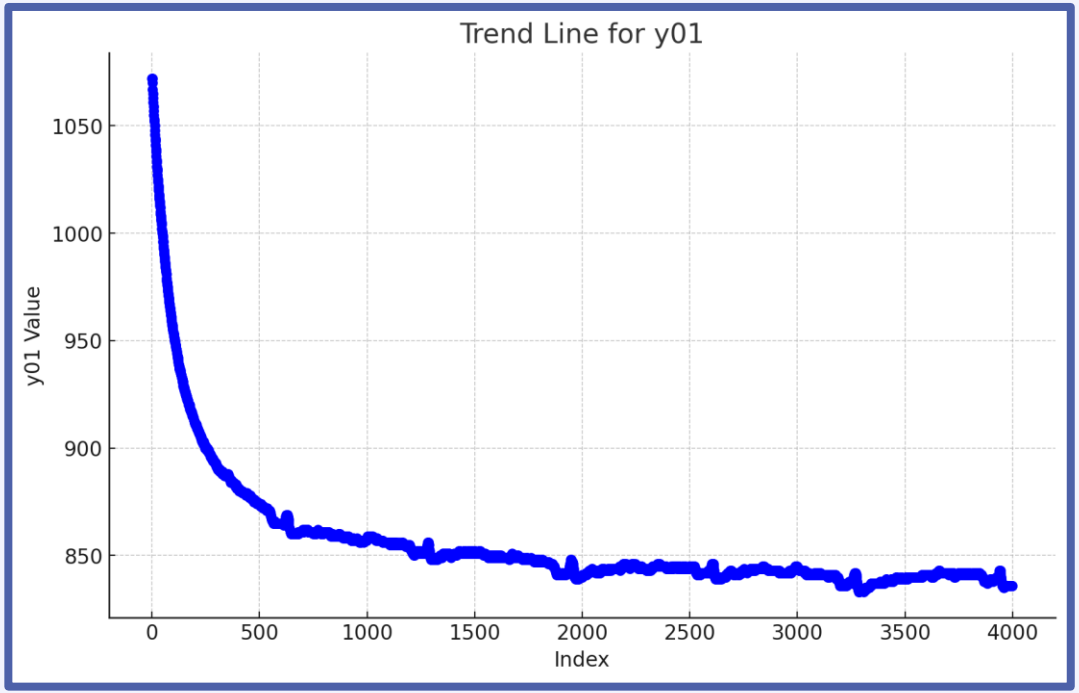
遭遇問題:
K-means無法有效區分多群，
且在部份環境下(如:a)，
幾乎所有材質被視為同一類。

預測方法設計

方法:使用指數函數預測趨勢

根據已有訓練資料，調整函數中的係數

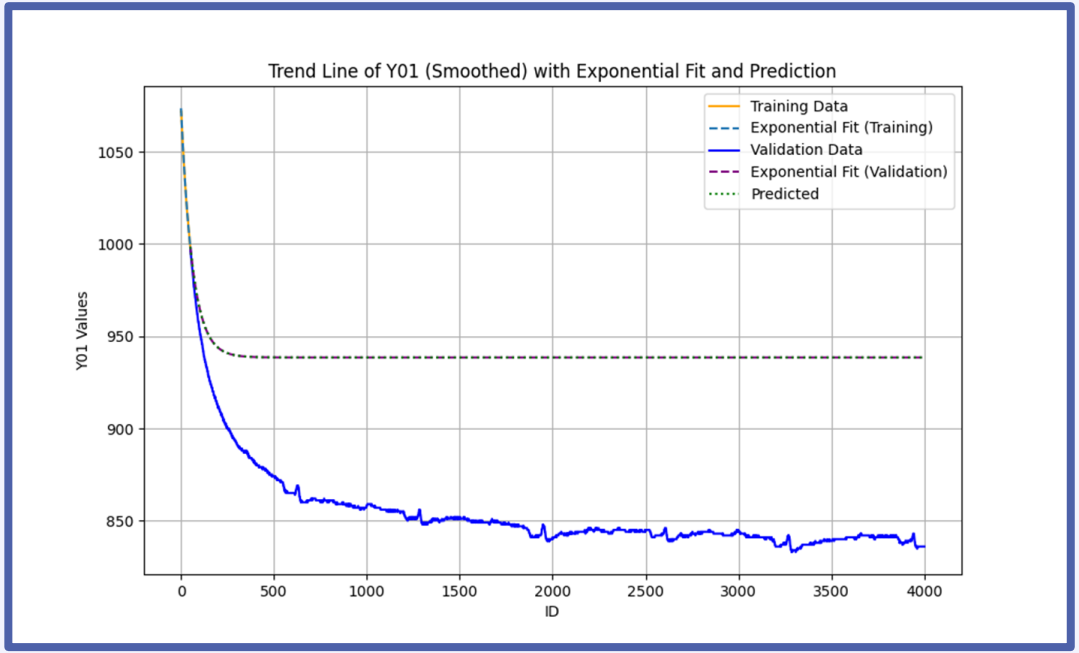
$$y = ae^{-kx} \Rightarrow \begin{cases} a = ? \\ k = ? \end{cases}$$



預測方法實做

遭遇問題

函數會提早收斂，由於前50筆還沒有達拐點，因此難以預測每一次拐點的位置



結語

特徵表示
訓練測試
交叉驗證