# เอกสารประกอบการบรรยาย

# Data Science with Python

บริษัท แลนด์ แอนด์ เฮ้าส์ จำกัด (มหาชน) | Land & Houses

โดย ผู้ช่วยศาสตราจารย์ ดร.วสิศ ลิ้มประเสริฐ

# บทที่ 3: การวิเคราะห์และสร้างตัวแปร (Analysis & Feature Engineering)

**Main Story (บทที่ 3):** "ตอนนี้เรามี 'Master Table' (df\_merged) ที่สะอาดและรวมข้อมูลทุกอย่างแล้ว แต่ข้อมูลดิบๆ เช่น FDPDUE01 (27 วัน, 98 วัน) วิเคราะห์ได้ยาก ในบทนี้ เราจะ 'สร้างตัวแปรใหม่' (Feature Engineering) เช่น 'DPD Bucket', 'Stage Name' และ 'Loan Age' เพื่อทำให้ข้อมูล 'มีประโยชน์' และง่ายต่อการสรุปผล"

## 3.1 การสร้างตัวแปรใหม่ (Feature Engineering) (3.5 ชม.)

### 3.1.1 การจับคู่และแปลงข้อมูล (Mapping)

* **Demo:** .map()
* **Main Story Link:** "แปลง STAGE\_CIF ที่เป็นเลข (1, 2, 3) ให้เป็นข้อความที่อ่านง่าย"
* **Workshop 3.1 (Mapping):**  
  stage\_map = {  
   1: '1. Performing',   
   2: '2. Under-performing',   
   3: '3. NPL'  
  }  
  df\_merged['Stage\_Name'] = df\_merged['STAGE\_CIF'].map(stage\_map)  
  df\_merged['Stage\_Name'].fillna('0. Unknown', inplace=True) # เผื่อค่าว่าง  
  print(df\_merged[['STAGE\_CIF', 'Stage\_Name']].head())

### 3.1.2 การสร้างตัวแปรจากเงื่อนไข (Conditional)

* **Demo:** np.where() (เทียบเท่า IF ใน Excel)
* **Workshop 3.2 (Conditional):**  
  import numpy as np  
  # สร้าง Flag 'Is\_Overdue'  
  df\_merged['Is\_Overdue'] = np.where(df\_merged['FDPDUE01'] > 0, True, False)  
  print(df\_merged[['FDPDUE01', 'Is\_Overdue']].head())

### 3.1.3 การจัดกลุ่มตัวเลข (Binning)

* **Demo:** pd.cut() (เทียบเท่า IF ซ้อนกันหลายชั้น หรือ VLOOKUP แบบ True)
* **Main Story Link:** "นี่คือหัวใจของรายงาน! การจัดกลุ่ม DPD เป็น Bucket"
* **Workshop 3.3 (Binning):**  
  bins = [-1, 0, 30, 60, 90, float('inf')]  
  labels = [  
   '0. No DPD',   
   '1. 1-30 Days',   
   '2. 31-60 Days',   
   '3. 61-90 Days',   
   '4. 90+ Days'  
  ]  
  # เราต้องมั่นใจว่า FDPDUE01 ไม่มีค่า NaN (ซึ่งเราทำแล้วในบทที่ 2)  
  df\_merged['DPD\_Bucket'] = pd.cut(df\_merged['FDPDUE01'], bins=bins, labels=labels, right=True)  
  print(df\_merged[['FDPDUE01', 'DPD\_Bucket']].head())

### 3.1.4 การใช้ .apply() กับฟังก์ชัน

* **Demo:** การใช้ .apply(lambda x: ...) หรือ .apply(my\_function)
* **Main Story Link:** "นำฟังก์ชัน classify\_stage ที่เราสร้างในบทที่ 1 มาใช้กับข้อมูลจริงทั้งคอลัมน์"
* **Workshop 3.4 (Apply):**  
  # (ทบทวน) สร้างฟังก์ชัน  
  def classify\_stage(dpd\_days):  
   if dpd\_days > 90:  
   return "3. NPL"  
   elif dpd\_days > 30:  
   return "2. Under-performing"  
   else:  
   return "1. Performing"  
    
  # ใช้ .apply กับฟังก์ชันที่สร้าง  
  df\_merged['Stage\_from\_DPD'] = df\_merged['FDPDUE01'].apply(classify\_stage)  
    
  # (ตัวอย่างขั้นสูง 1) .apply(lambda)  
  # สร้าง Flag สินเชื่อวงเงินสูง  
  df\_merged['High\_Value\_Flag'] = df\_merged['FPRINCAM'].apply(  
   lambda x: 'High Value' if x > 5000000 else 'Standard'  
  )  
    
  # (ตัวอย่างขั้นสูง 2) .apply(axis=1) (ใช้ข้อมูลหลายคอลัมน์)  
  # คำนวณอัตราส่วนหนี้ต่อวงเงิน (สมมติมี FFLGBWFW คือวงเงิน)  
  def calculate\_ratio(row):  
   if row['FFLGBWFW'] > 0:  
   return row['FPRINCAM'] / row['FFLGBWFW']  
   else:  
   return np.nan  
    
  # df\_merged['Debt\_to\_Limit\_Ratio'] = df\_merged.apply(calculate\_ratio, axis=1)

### 3.1.5 การสร้างตัวแปรจากวันที่ (Date Feature Engineering)

* **Demo:** การใช้ .dt accessor (เช่น .dt.year, .dt.month)
* **Main Story Link:** "ข้อมูล 'วันที่' มีค่ามหาศาล เรามาคำนวณ 'อายุสินเชื่อ' และ 'เวลาที่เหลือ' กัน"
* **Workshop 3.5 (Date Features):**  
  # (ทบทวน) เราแปลงคอลัมน์วันที่ในบทที่ 2 แล้ว  
  # df\_merged['FRPDATE'] = pd.to\_datetime(df\_tran['FRPDATE'], format='%Y%m%d')  
  # df\_merged['FORDATE'] = pd.to\_datetime(df\_tran['FORDATE'], format='%Y%m%d', errors='coerce')  
  # df\_merged['FMATDATE'] = pd.to\_datetime(df\_tran['FMATDATE'], format='%Y%m%d', errors='coerce')  
    
  # 1. สร้าง "อายุสินเชื่อ" (Loan Age in Days)  
  # (ต้องระวัง TimeDelta object, .dt.days เพื่อแปลงเป็นตัวเลข)  
  df\_merged['Loan\_Age\_Days'] = (df\_merged['FRPDATE'] - df\_merged['FORDATE']).dt.days  
    
  # 2. สร้าง "เวลาที่เหลือ" (Remaining Tenor in Days)  
  df\_merged['Remaining\_Tenor\_Days'] = (df\_merged['FMATDATE'] - df\_merged['FRPDATE']).dt.days  
    
  # 3. ดึง "เดือน" "ไตรมาส" "ปี" ที่ปล่อยกู้  
  df\_merged['Loan\_Orig\_Year'] = df\_merged['FORDATE'].dt.year  
  df\_merged['Loan\_Orig\_Quarter'] = df\_merged['FORDATE'].dt.quarter  
    
  print(df\_merged[['FRPDATE', 'FORDATE', 'Loan\_Age\_Days', 'Loan\_Orig\_Year']].head())

## 3.2 การวิเคราะห์และสรุปผล (Aggregation & Analysis) (2.5 ชม.)

### 3.2.1 การจัดกลุ่มข้อมูล (Grouping)

* **Demo:** .groupby()
* **Main Story Link:** "สรุปยอดหนี้รวม (SUM) และยอดหนี้เฉลี่ย (MEAN) โดยแบ่งตาม 'Stage\_Name' และ 'DPD\_Bucket' ที่เราเพิ่งสร้าง"
* **Workshop 3.6 (GroupBy):**  
  # สรุปยอดหนี้รวมตาม Stage  
  report\_by\_stage = df\_merged.groupby('Stage\_Name')['FPRINCAM'].agg(['sum', 'mean', 'count'])  
  print(report\_by\_stage)  
    
  # สรุปยอดหนี้รวมตาม DPD Bucket  
  report\_by\_dpd\_bucket = df\_merged.groupby('DPD\_Bucket')['FPRINCAM'].agg(['sum', 'count'])  
  print(report\_by\_dpd\_bucket)

### 3.2.2 การ Grouping ขั้นสูง (Advanced Grouping)

* **Demo:** Group by หลายคอลัมน์ และ .agg() ด้วย Dictionary
* **Workshop 3.7 (Advanced GroupBy):**  
  # 1. Group by 2 ระดับ (Stage และ Product Type)  
  report\_multi\_level = df\_merged.groupby(['Stage\_Name', 'FPRODTY'])['FPRINCAM'].agg(['sum', 'count'])  
  print(report\_multi\_level)  
    
  # 2. ใช้ .agg() กับ Dictionary  
  # สรุป FPRINCAM (ยอดหนี้) เป็น 'sum'  
  # สรุป FDPDUE01 (วันค้าง) เป็น 'mean'  
  # สรุป Loan\_Age\_Days (อายุสินเชื่อ) เป็น 'mean'  
  summary\_agg = df\_merged.groupby('Stage\_Name').agg(  
   Total\_Principal=('FPRINCAM', 'sum'),  
   Avg\_DPD=('FDPDUE01', 'mean'),  
   Avg\_Loan\_Age=('Loan\_Age\_Days', 'mean'),  
   Num\_Accounts=('FACCNO', 'count')  
  )  
  print(summary\_agg)

### 3.2.3 การสรุปผลด้วย Pivot Table

* **Demo:** pd.pivot\_table()
* **Main Story Link:** "สร้าง 'ตารางสรุป' ที่ผู้บริหารต้องการดู"
* **Workshop 3.8 (Pivot Table):**  
  # สร้างตารางสรุป "ยอดหนี้รวม"  
  # แถว (index) = DPD\_Bucket  
  # คอลัมน์ (columns) = FPRODTY (ประเภท Product)  
  pivot\_report = df\_merged.pivot\_table(  
   index='DPD\_Bucket',   
   columns='FPRODTY',   
   values='FPRINCAM',   
   aggfunc='sum',   
   fill\_value=0,  
   margins=True # เพิ่ม Total  
  )  
  print(pivot\_report)

### 3.2.4 การเรียงลำดับและส่งออก (Sorting & Export)

* **Demo:** sort\_values(), to\_excel()
* **Workshop 3.9 (Sorting & Export):**  
  # 1. เรียงลำดับตาราง summary\_agg (จาก Workshop 3.7)  
  # เรียงตาม "ยอดหนี้รวม" (Total\_Principal) จากมากไปน้อย  
  summary\_agg\_sorted = summary\_agg.sort\_values(by='Total\_Principal', ascending=False)  
  print(summary\_agg\_sorted)  
    
  # 2. ส่งออกรายงาน (จาก Workshop 3.8)  
  pivot\_report.to\_excel("Monthly\_NPL\_Report.xlsx", sheet\_name="Pivot by DPD")  
    
  # 3. (ขั้นสูง) ส่งออกหลายตารางในไฟล์เดียว  
  with pd.ExcelWriter('Full\_NPL\_Report.xlsx') as writer:  
   pivot\_report.to\_excel(writer, sheet\_name='Pivot\_by\_DPD')  
   summary\_agg\_sorted.to\_excel(writer, sheet\_name='Summary\_by\_Stage')

## 3.3 Mini-Game: ทบทวนความรู้ (Review Game)

**ภารกิจ:** จับคู่ "สิ่งที่เราอยากทำ" (ข้อ 1-8) กับ "โค้ด Pandas ที่ถูกต้อง" (ข้อ A-H)

**สิ่งที่เราอยากทำ (Tasks):**

1. สร้างตารางสรุปผลแบบไขว้ (แถว=DPD, คอลัมน์=Product) เหมือนใน Excel
2. แปลงตัวเลข STAGE\_CIF (1, 2, 3) เป็นข้อความ ('Performing', 'NPL')
3. สร้าง Flag Is\_Overdue (True/False) โดยใช้เงื่อนไข FDPDUE01 > 0
4. สรุปยอด FPRINCAM รวม โดยจัดกลุ่มตาม Stage\_Name
5. ซอย FDPDUE01 (ตัวเลข) ออกเป็น "ช่วง" (เช่น '1-30 Days', '31-60 Days')
6. คำนวณ "อายุสินเชื่อ" (เป็นวัน) จาก FRPDATE และ FORDATE
7. เรียงลำดับตาราง summary\_agg ตามคอลัมน์ Total\_Principal จากมากไปน้อย
8. สรุปผลหลายแบบ (Sum ของ FPRINCAM และ Mean ของ FDPDUE01) โดยจัดกลุ่มตาม Stage\_Name

โค้ด Pandas (Code Snippets):

A. pd.cut(df\_merged['FDPDUE01'], bins=bins, labels=labels)

B. df\_merged.groupby('Stage\_Name')['FPRINCAM'].sum()

C. summary\_agg.sort\_values(by='Total\_Principal', ascending=False)

D. df\_merged['Stage\_Name'] = df\_merged['STAGE\_CIF'].map(stage\_map)

E. df\_merged.pivot\_table(index='DPD\_Bucket', columns='FPRODTY', ...)

F. np.where(df\_merged['FDPDUE01'] > 0, True, False)

G. (df\_merged['FRPDATE'] - df\_merged['FORDATE']).dt.days

H. df\_merged.groupby('Stage\_Name').agg(Total\_Principal=('FPRINCAM', 'sum'), Avg\_DPD=('FDPDUE01', 'mean'))

*(เฉลย: 1=E, 2=D, 3=F, 4=B, 5=A, 6=G, 7=C, 8=H)*