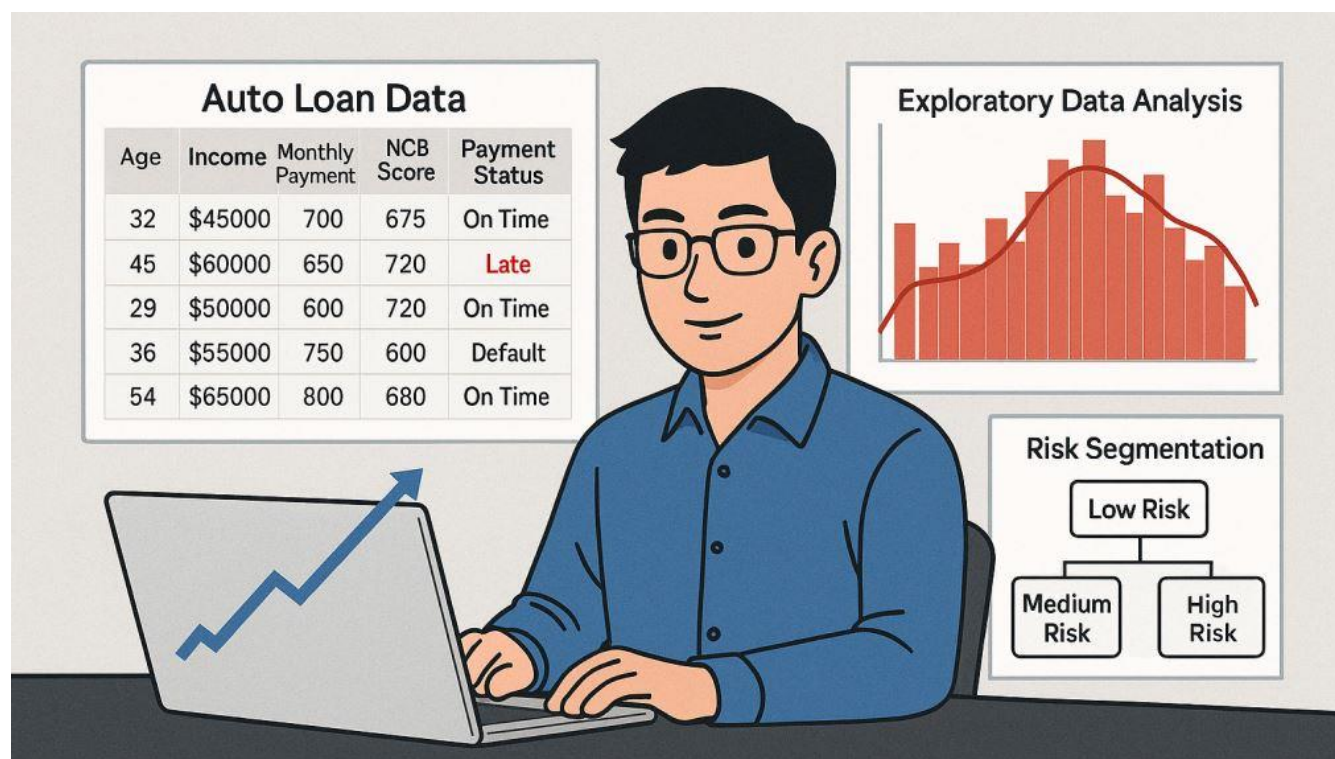


Introduction

สวัสดีครับ ผมชื่อ ภาคิน เลิศวิจิตรวัฒน์ ปัจจุบันทำงานในสาย Internal Audit โดยมีหน้าที่หลักในการตรวจสอบความถูกต้องของกระบวนการภายในองค์กร ทั้งในด้านการเงิน การดำเนินงาน และการบริหารความเสี่ยง แต่ในยุคที่ข้อมูลมีอยู่รอบตัว การตรวจสอบแบบเดิม ๆ อาจไม่เพียงพออีกต่อไป ผมเองมีความหลงใหลในการใช้ Data Analytics มาช่วยในงาน Audit มากขึ้นเรื่อย ๆ เพราะมันสามารถช่วยเราวิเคราะห์พฤติกรรม ตรวจจับความเสี่ยง และคาดการณ์แนวโน้มที่ผิดปกติได้อย่างรวดเร็ว แม่นยำ และลึกซึ้ง

วันนี้ผมจึงอยากแชร์ Case Study ง่าย ๆ ที่ผมนำ Data Analytics มาประยุกต์ใช้กับ ข้อมูลจำลองเกี่ยวกับสินเชื่อเช่าซื้อรถยนต์ (ข้อมูลชุดนี้เป็นข้อมูลที่สร้างขึ้นมาเองเพื่อใช้ในการเรียนรู้และนำเสนอเท่านั้น) เพื่อให้เห็นภาพว่า หากเรามีข้อมูลในลักษณะนี้จริง ๆ จะสามารถนำไปใช้วิเคราะห์ความเสี่ยงของลูกค้าได้อย่างไรตั้งแต่ขั้นตอนการเตรียมข้อมูล, วิเคราะห์เบื้องต้น, ไปจนถึงการจัดกลุ่มความเสี่ยง หวังว่าตัวอย่างนี้จะช่วยให้เห็นภาพชัดเจนขึ้นว่า “Data Analytics สามารถเพิ่มพลังให้งาน Audit ได้มากแค่ไหน”



คำอธิบายคอลัมน์สำคัญใน Data

ชื่อคอลัมน์	ความหมาย
Age	อายุของลูกค้า
Income	รายได้ต่อเดือน
Monthly Payment	ค่างวดที่ลูกค้าต้องผ่อนต่อเดือน
NCB Score	คะแนนเครดิตของลูกค้า (ช่วง 300-850)
Missed Payments	จำนวนงวดที่เคยขาดส่ง
Payment Status	สถานะการชำระ เช่น On Time, Late, Default
Start Date / End Date	วันที่เริ่มและสิ้นสุดสัญญาเช่าซื้อ

ใช้ตรวจสอบอะไรได้บ้าง

ด้าน Audit	ใช้ข้อมูลอะไร	ตัวอย่างคำถามที่ตอบได้
Compliance	NCB, Ratio	มีใครที่ได้อนุมัติทั้งที่ NCB ต่ำเกินเกณฑ์ไหม
Risk Analysis	Payment Status, Ratio	กลุ่มเสี่ยงสูงมีแนวโน้มผิดนัดมากแค่ไหน
Smart Sampling	Missed Payments	จะเลือกตรวจลูกค้ากลุ่มไหนก่อนดี
Behavior Review	History/Score	ลูกค้าที่ไม่เคยขาดส่งแต่มี NCB ต่ำ คือใคร

ข้อมูลชุดนี้ถูกออกแบบให้สะท้อนสถานการณ์จริงในธุรกิจสินเชื่อ และเหมาะมากในการนำมาใช้ฝึกวิเคราะห์
ด้าน Internal Audit เพื่อช่วย:

- ตรวจสอบนโยบายสินเชื่อ
- ประเมินพฤติกรรมลูกค้า
- ลดความเสี่ยงขององค์กร

ขั้นตอนการทำ Data Analytics ที่สำคัญ

1. การเตรียมข้อมูล (Data Cleaning)
2. การวิเคราะห์ข้อมูลเบื้องต้น (EDA)
3. สรุปผลการวิเคราะห์ (Result Summary)

เครื่องมือที่ใช้ในการวิเคราะห์ (Tools Used)

- Google Colab
- ภาษาที่ใช้ Python

ขั้นตอนที่ 1: การเตรียมข้อมูล (Data Cleaning)

1.1 วิธีอัปโหลดไฟล์จากเครื่องคุณ





พิมพ์ Code ดังนี้ โดย Code จะสร้างปุ่ม Upload File ที่เราต้องการใช้งาน และสามารถเลือกไฟล์ที่อยู่ในเครื่องคอมพิวเตอร์ของเราได้

```
1 from google.colab import files
2 uploaded = files.upload()
```

เลือกไฟล์ ไม่ได้เลือกไฟล์ใด

1.2 ผมได้เตรียมข้อมูลตัวอย่างสำหรับการทดลองใช้งานชื่อว่า “ข้อมูลจำลองเกี่ยวกับสินเชื่อเช่าซื้อรถยนต์”

สามารถ Download ได้จาก https://github.com/pakin77/Hire_purchase ชื่อไฟล์ “Vehicle_Hire_Purchase_Data.csv”

 pakin77 Add files via upload		32b9d36 · 24 minutes ago	 2 Commits
 README.md	Initial commit	27 minutes ago	
 Vehicle_Hire_Purchase_Data.csv	Add files via upload	24 minutes ago	

1.3 เมื่อเรา Upload Data (ข้อมูล) เข้ามาแล้ว และข้อมูลของเรายังอยู่ในรูปแบบของ CSV file ซึ่งยังไม่พร้อมใช้งาน ให้เราทำการจัดให้อยู่ในรูปแบบ Data Frame โดยเขียน Code ดังนี้

1.3.1 ต้องทำการ import pandas library ที่ใช้สำหรับการจัดการข้อมูลในรูปแบบตาราง

```
1 import pandas as pd
```

1.3.2 เมื่อเรา import pandas library เรียบร้อยแล้ว ก็มาจัดข้อมูลให้อยู่ในโครงสร้างข้อมูล

```
1 df = pd.read_csv("Vehicle_Hire_Purchase_Data.csv")
```

1.4 เมื่อเราจัดข้อมูลให้อยู่ในโครงสร้างข้อมูลเรียบร้อยแล้ว ทีนี้เราก็มามาทำการตรวจสอบข้อมูล โดยใช้คำสั่ง

df.head() # แสดง 5 แถวแรกของข้อมูล

df.shape # ดูว่ามีกี่แถว กี่คอลัมน์

df.columns # ดูชื่อคอลัมน์ทั้งหมด

df.dtypes # ดูชนิดข้อมูลของแต่ละคอลัมน์

df.isnull().sum() # ตรวจสอบจำนวนค่าว่าง (Missing Values) ในแต่ละคอลัมน์

1.5 เมื่อเราทำการตรวจสอบข้อมูลเรียบร้อยแล้ว เราจะพบว่า types ของข้อมูลวันที่เป็น “object” ซึ่งเป็นชนิดข้อมูลใน pandas library (เช่น ชื่อ, วันที่แบบ string) จะถูกระบุว่าเป็น “object”

```
Start Date      object
```

เราต้องทำการแปลงวันที่ให้อยู่ในรูปแบบ Date Time ที่สามารถใช้งานได้ โดยพิมพ์คำสั่งดังนี้

```
1 df['Start Date'] = pd.to_datetime(df['Start Date'])
2 df['End Date'] = pd.to_datetime(df['End Date'])
```

1.6 เมื่อเราแปลงวันที่ให้เป็นข้อมูลที่พร้อมใช้งานแล้ว ในขั้นต่อไปเราต้องลบข้อมูลที่ไม่มีความสำคัญ ดังนี้
เงื่อนไขในการลบ

- อายุ < 18 หรือ > 80
- รายได้ ≤ 0
- NCB Score นอกช่วง 300–850

1.6.1 ลบข้อมูลอายุ < 18 หรือ > 80 เพราะในโลกความเป็นจริง คนที่อายุต่ำกว่า 18 ยังไม่สามารถทำสัญญาสินเชื่อได้ตามกฎหมาย และคนอายุเกิน 80 ก็มักไม่ใช่กลุ่มเป้าหมายในการเข้าซื้อรถยนต์

```
1 df = df[(df['Age'] >= 18) & (df['Age'] <= 80)]
```

1.6.2 ลบข้อมูลรายได้ ≤ 0 เพราะรายได้ 0 หมายถึงไม่มีเงินเดือนเลย ซึ่งไม่มีทางจะผ่านอนุมัติสินเชื่อ บางครั้งอาจเป็นเพราะข้อมูลผิดพลาด เช่น พิมพ์ผิดหรือลืมใส่

```
1 df = df[df['Income'] > 0]
```

1.6.3 ลบข้อมูล NCB Score นอกช่วง 300–850 คะแนนเครดิต (NCB) จะอยู่ในช่วง 300–850 ตามมาตรฐานของเครดิตบูโรไทย ถ้าเกินช่วงนี้ แสดงว่าอาจเป็นข้อมูลผิด เช่น พิมพ์เกิน 1 หลัก หรือลืมตรวจสอบก่อนบันทึก

```
1 df = df[(df['NCB Score'] >= 300) & (df['NCB Score'] <= 850)]
```

1.7 เมื่อเราลบข้อมูลที่ไม่มีความสำคัญเรียบร้อยแล้ว เราสามารถบันทึกและดาวน์โหลดไฟล์ CSV หลังจากทำ Data Cleaning เสร็จแล้วออกมาใช้งานต่อ เช่น นำไปวิเคราะห์ต่อ หรือใช้ใน Power BI หรือส่งให้ทีมอื่นๆ โดยพิมพ์คำสั่งดังนี้

```
1 df.to_csv("cleaned_data.csv", index=False)  
2 files.download("cleaned_data.csv")
```

ขั้นตอนที่ 2: การวิเคราะห์ข้อมูลเบื้องต้น (EDA)

EDA (Exploratory Data Analysis) คือการทำความเข้าใจข้อมูลก่อนวิเคราะห์จริง โดยใช้ทั้งการดูสถิติเบื้องต้น และการวาดกราฟเพื่อมองเห็น “รูปแบบ” หรือ “ความผิดปกติ” ที่ซ่อนอยู่ในข้อมูล

2.1 การ import library

2.1.1 โหลดไลบรารี pandas สำหรับจัดการข้อมูลในรูปแบบตาราง (DataFrame)

2.2.2 โหลด matplotlib.pyplot สำหรับสร้างกราฟ เช่น เส้น, แท่ง, จุด, ฮิสโตแกรม ฯลฯ

2.3.3 โหลดไลบรารี seaborn สำหรับสร้างกราฟวิเคราะห์ข้อมูล

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
```

2.2 เริ่มวิเคราะห์ข้อมูล (EDA) โดยเริ่มดูที่การกระจายของคะแนนเครดิต (NCB Score) เพราะคะแนนเครดิต (NCB Score) เป็นตัวแปรที่สำคัญดังนี้

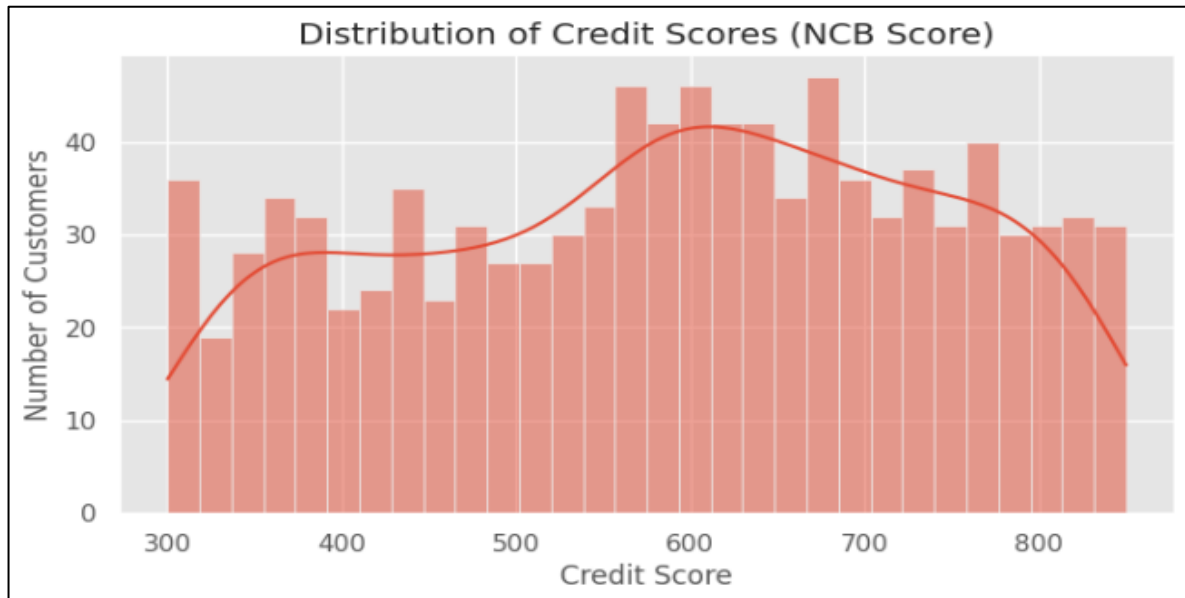
- สะท้อนความน่าเชื่อถือทางการเงินของลูกค้า
- ใช้เป็นปัจจัยหลักในการพิจารณาปล่อยสินเชื่อ
- มีความสัมพันธ์กับ “ความเสี่ยงในการผิดนัดชำระ” สูงมาก

```
1 plt.figure(figsize=(8, 4))
2 sns.histplot(df['NCB Score'], bins=30, kde=True)
3 plt.title('Distribution of Credit Scores (NCB Score)')
4 plt.xlabel('Credit Score')
5 plt.ylabel('Number of Customers')
6 plt.show()
```

คำอธิบายเพิ่มเติม:

- bins=30 แบ่งช่วงคะแนนเครดิตเป็น 30 ช่วง
- kde=True วาดเส้นโค้งความหนาแน่น (ช่วยให้เห็นรูปแบบมากขึ้น)

ตัวอย่างกราฟที่ได้:



วิธีอ่านกราฟ Histogram "การแจกแจงคะแนนเครดิต (NCB Score)"

- แกน X (แนวนอน): คะแนนเครดิต (NCB Score)
 - ตัวเลขที่อยู่ด้านล่างของกราฟ เช่น 300, 400, ... ไปถึง 850 คือ ช่วงคะแนนเครดิตของลูกค้า
- แกน Y (แนวตั้ง): จำนวนลูกค้า
 - ตัวเลขด้านซ้าย เช่น 10, 20, 30, 40, ... คือ "จำนวนลูกค้า"
- แท่งกราฟ (Bar สีแดง): แสดงจำนวนลูกค้าในแต่ละช่วงคะแนนเครดิต ตัวอย่างเช่น:
 - แท่งที่อยู่ที่ 600 สูงที่สุด = แปลว่ามีลูกค้ามากที่สุดที่มี NCB Score ประมาณ 600
- เส้นโค้งแดง (เส้น KDE): คือเส้นแนวโน้มการกระจายของข้อมูล
 - เส้นนี้ช่วยให้เราเห็นว่า คะแนนเครดิตกระจายอย่างไรบ้าง

Insight จากกราฟนี้

- กลุ่มลูกค้าที่มี NCB 600–700 ควรเป็นกลุ่มเป้าหมายหลักในการปล่อยสินเชื่อ
- กลุ่ม NCB < 500 ควรมีมาตรการป้องกันความเสี่ยง เช่น: เพิ่มเงินค้ำประกัน, ลดจำนวนวงเงิน เป็นต้น
- กลุ่มที่มี NCB > 750 มีศักยภาพสูง แต่อาจต้องมีโปรโมชั่นเพื่อดึงดูด (เพราะมีไม่มาก)

2.3 วิเคราะห์ข้อมูล NCB Score กับการขาดส่งค่างวด การวิเคราะห์ในส่วนนี้จะช่วยให้เราสามารถเข้าใจในพฤติกรรมลูกค้าได้มากขึ้น ดังนี้

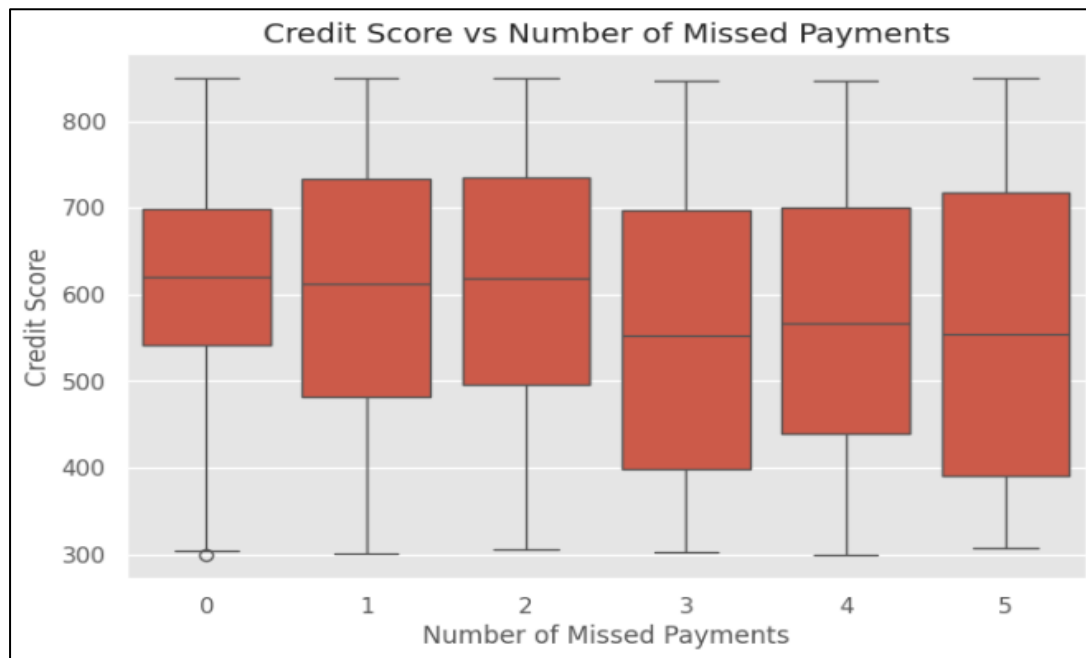
- ตรวจสอบว่าพฤติกรรมลูกค้า "สะท้อน" ในเครดิตจริงหรือไม่
- ช่วยตั้ง "เกณฑ์ความเสี่ยง" สำหรับอนุมัติสินเชื่อ
- ช่วยให้ฝ่าย Audit เลือกกลุ่มลูกค้าที่ควรตรวจสอบเชิงลึก
- วางพื้นฐานสำหรับการจัดกลุ่มความเสี่ยง (Risk Segmentation)

```
1 plt.figure(figsize=(8, 5))
2 sns.boxplot(x='Missed Payments', y='NCB Score', data=df)
3 plt.title('Credit Score vs Number of Missed Payments')
4 plt.xlabel('Number of Missed Payments')
5 plt.ylabel('Credit Score')
6 plt.show()
```

คำอธิบายเพิ่มเติม:

1. plt.figure(figsize=(8, 5))
 - เป็นคำสั่งของ matplotlib ใช้สำหรับสร้าง ขนาดของกราฟ ความกว้าง 8 นิ้ว, ความสูง 5 นิ้ว
2. sns.boxplot(x='Missed Payments', y='NCB Score', data=df)
ใช้ seaborn สร้าง **boxplot** (กราฟกล่อง)
 - x='Missed Payments' → แบ่งกลุ่มตามจำนวนงวดที่ขาดส่ง (แนวนอน)
 - y='NCB Score' → แสดงค่าคะแนนเครดิต (แนวตั้ง)
 - data=df → ใช้ DataFrame ชื่อ df เป็นแหล่งข้อมูล
3. plt.title('Credit Score vs Number of Missed Payments')
 - ตั้งชื่อกราฟ (แสดงบนหัวกราฟ)
4. plt.xlabel('Number of Missed Payments') > ตั้งชื่อ แกน X
5. plt.ylabel('Credit Score') > ตั้งชื่อ แกน Y
6. plt.show() > แสดงผลกราฟบนหน้าจอ

ตัวอย่างกราฟที่ได้:



วิธีอ่านกราฟ Histogram "การแจกแจงคะแนนเครดิต (NCB Score)"

- แกน X (แนวนอน): จำนวนงวดที่ขาดส่ง
 - หมายเลข 0, 1, 2, 3, 4, 5 = ขาดส่งกี่งวด
- แกน Y (แนวตั้ง): คะแนนเครดิต (NCB Score)
 - ช่วงคะแนนเครดิตของลูกค้า เช่น 300, 400, ... ไปถึง 850

Insight จากกราฟนี้

- กลุ่ม Missed Payments = 0 มี ค่ากลาง (Median) ของ Credit Score อยู่สูงที่สุด (~640-660) การกระจาย (กล่อง) แคบกว่ากลุ่มอื่น → คะแนนนิ่งและอยู่ในเกณฑ์ดี
- กลุ่ม 1-2 งวด มี ค่ากลาง (Median) ลดลง (~600-630) การกระจายกว้างขึ้น → มีทั้งคนที่ยังมีเครดิตดีและเริ่มตกต่ำ
- กลุ่ม 3-5 งวด มี ค่ากลาง (Median) ลดลงชัด (~540-580) กล่องกระจายกว้าง → กลุ่มนี้มีความเสี่ยงหลากหลาย

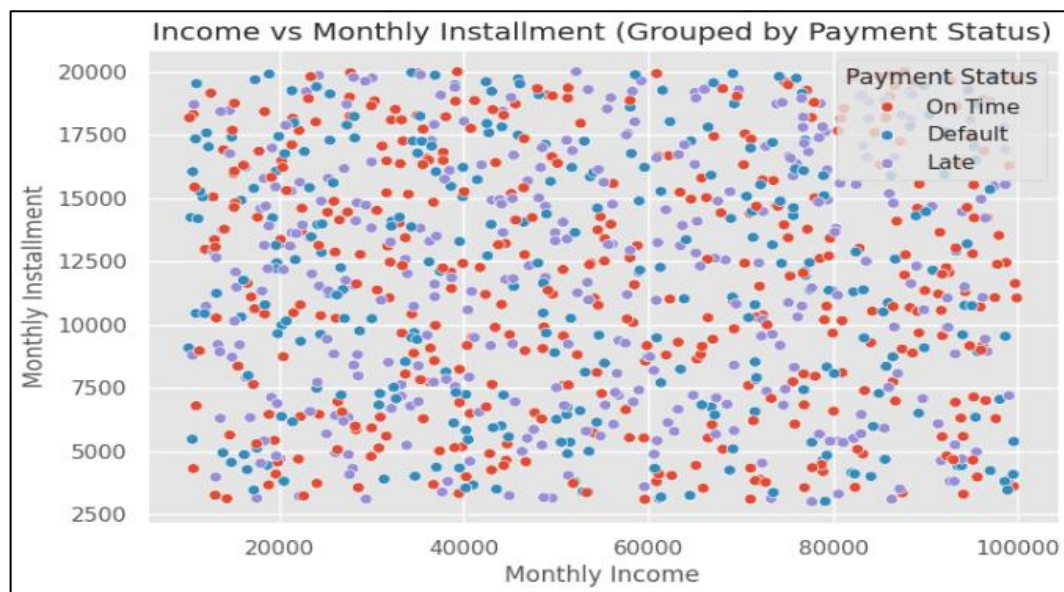
2.4 วิเคราะห์ข้อมูล รายได้ vs ค่างวดรายเดือน โดยแยกตามสถานะการชำระเงิน (Payment Status) เป็นหนึ่งในมุมที่สำคัญมากในการประเมิน “ความสามารถในการผ่อนชำระ” ของลูกค้า ซึ่งมีผลโดยตรงต่อความเสี่ยงด้านเครดิตและการอนุมัติสินเชื่อ

```
1 plt.figure(figsize=(8, 5))
2 sns.scatterplot(data=df, x='Income', y='Monthly Payment', hue='Payment Status')
3 plt.title('Income vs Monthly Installment (Grouped by Payment Status)')
4 plt.xlabel('Monthly Income')
5 plt.ylabel('Monthly Installment')
6 plt.show()
```

คำอธิบายเพิ่มเติม:

1. plt.figure(figsize=(8, 5))
 - เป็นคำสั่งของ matplotlib ใช้สำหรับสร้าง ขนาดของกราฟ ความกว้าง 8 นิ้ว, ความสูง 5 นิ้ว
2. sns.scatterplot(data=df, x='Income', y='Monthly Payment', hue='Payment Status')
 - ใช้ seaborn สร้าง กราฟ scatter plot (กราฟจุด)
 - data=df ใช้ DataFrame ชื่อ df เป็นแหล่งข้อมูล
 - x='Income' ค่าบนแกน X คือรายได้ต่อเดือน
 - y='Monthly Payment' ค่าบนแกน Y คือค่างวดที่ต้องจ่าย
 - hue='Payment Status' แยกสีของจุดตามสถานะการชำระ เช่น On Time / Late / Default
3. plt.title('Income vs Monthly Installment (Grouped by Payment Status)')
 - ตั้งชื่อกราฟด้านบน
4. plt.xlabel('Monthly Income')
 - ตั้งชื่อแกน X = Monthly Income
5. plt.ylabel('Monthly Installment')
 - ตั้งชื่อแกน Y = Monthly Installment
6. plt.show() > แสดงผลกราฟที่เราสร้างทั้งหมด

ตัวอย่างกราฟที่ได้:



วิธีอ่านกราฟ Scatterplot "รายได้ vs ค่างวด (ดูความสามารถในการผ่อน)"

1. แกน X (แนวนอน): รายได้ต่อเดือน
2. แกน Y (แนวตั้ง): ค่างวดต่อเดือน
3. จุดแต่ละจุด คือลูกค้าหนึ่งคน
4. สีของจุด บอกสถานะชำระเงิน (Payment Status) → On Time, Late, Default

Insight จากกราฟนี้

1. กลุ่มสีแดง (On Time – จ่ายตรงเวลา) กระจายครอบคลุม เกือบทุกระดับรายได้ ค่างวด
 - มีทั้งคนรายได้น้อย (10,000–20,000) และคนที่ผ่อนแพง (10,000–15,000)
2. กลุ่มสีม่วง (Late – จ่ายล่าช้า) พบมากในรายได้ระดับกลาง (30,000–60,000)
 - กระจายกว้างในกลุ่มที่ผ่อนประมาณ 8,000–13,000
3. กลุ่มสีน้ำเงิน (Default – ผิดนัด) ชัดเจนมากกว่า พบมากในกลุ่มรายได้ต่ำ (<40,000) และ ผ่อนหนัก (>9,000)
 - จุดสีน้ำเงินจะกระจุกตรงมุมล่างซ้ายของกราฟ (รายได้น้อย + ผ่อนเยอะ)

2.5 สร้างตัวชี้วัดภาระทางการเงิน “Installment-to-Income Ratio” เป็นเครื่องมือสำคัญในการวิเคราะห์ความสามารถของลูกค้าว่าผ่อนไหวไหม, เสี่ยงไหม, และใช้วางแผนอนุมัติสินเชื่อหรือจัดกลุ่มความเสี่ยงเนื่องจากในข้อมูลตัวอย่างไม่มีคอลัมน์ที่ชื่อ “Installment-to-Income Ratio” ดังนั้นเราจึงต้องทำการเพิ่มคอลัมน์ขึ้นมาโดยใช้ Library pandas:

2.5.1 โดยการพิมพ์คำสั่งดังนี้

```
1 df['Installment_to_Income_Ratio'] = df['Monthly Payment'] / df['Income']
```

2.5.2 เมื่อเราสร้างคอลัมน์เรียบร้อยแล้วก็ทำการตรวจสอบข้อมูล โดยการพิมพ์คำสั่งดังนี้

```
1 df[['Income', 'Monthly Payment', 'Installment_to_Income_Ratio']].head(10)
```

ตัวอย่างข้อมูล:

	Income	Monthly Payment	Installment_to_Income_Ratio
0	34951	18269	0.522703
1	45400	11511	0.253546
2	47573	5868	0.123347
3	66979	9429	0.140775
4	50042	6265	0.125195
5	91412	11214	0.122675
6	28297	14106	0.498498
7	34578	8876	0.256695
8	44701	5290	0.118342
9	75160	10881	0.144771

คำอธิบายข้อมูลแต่ละคอลัมน์:

- Income รายได้ต่อเดือนของลูกค้า (หน่วย: บาท)
- Monthly Payment ค่างวดที่ลูกค้าต้องผ่อนต่อเดือน
- Installment_to_Income_Ratio อัตราส่วนระหว่างค่างวดกับรายได้ (ภาระผ่อนชำระ)

Ratio: ช่วงของข้อมูล < 0.30 ภาระต่ำ (ปลอดภัย), 0.30 – 0.50 ภาระเริ่มสูง (เฝ้าระวัง)

และ > 0.50 ภาระเกินตัว (ความเสี่ยงสูง)

2.6 สร้างตัวแปรใหม่ชื่อว่า (Risk Level) เพื่อใช้จัดกลุ่มความเสี่ยงของลูกค้าโดยอิงจาก 2 ปัจจัยหลัก ได้แก่:

- Installment-to-Income Ratio (ภาระการผ่อนเทียบกับรายได้)
- NCB Score (คะแนนเครดิต)

```
1 def assign_risk(row):
2     ratio = row['Installment_to_Income_Ratio']
3     ncb = row['NCB Score']
4
5     if ratio > 0.6 or ncb < 550:
6         return 'High Risk'
7     elif ratio > 0.4 or ncb < 600:
8         return 'Medium Risk'
9     else:
10        return 'Low Risk'
11
12 df['Risk_Level'] = df.apply(assign_risk, axis=1)
```

คำอธิบายเพิ่มเติม:

1. def assign_risk(row):
 - สร้างฟังก์ชัน assign_risk เพื่อประเมินความเสี่ยงของลูกค้าแต่ละราย
2. ratio = row['Installment_to_Income_Ratio']:
 - ดึงค่า Ratio ที่เรากำหนดไว้มาก่อนหน้านี้
3. ncb = row['NCB Score']:
 - ดึงคะแนนเครดิตของลูกค้า
4. df['Risk_Level'] = df.apply(assign_risk, axis=1):
 - ใส่ผลลัพธ์ไว้ในคอลัมน์ใหม่ชื่อ Risk_Level

เกณฑ์ในการจัดระดับความเสี่ยง:

- ratio > 0.6 หรือ ncb < 550 'High Risk'
- ratio > 0.4 หรือ ncb < 600 'Medium Risk'
- ไม่เข้าเงื่อนไขใดเลย 'Low Risk'

2.7 การจัดกลุ่มความเสี่ยงของลูกค้า (Risk Segmentation) เพื่อแบ่งกลุ่มลูกค้าออกเป็นระดับความเสี่ยงตามพฤติกรรมทางการเงิน และคุณลักษณะสำคัญ (เช่น NCB Score, ความสามารถในการผ่อน)

```
1 risk_summary = df['Risk_Level'].value_counts().to_frame(name='Count')
2 risk_summary['Percentage'] = (df['Risk_Level'].value_counts(normalize=True) * 100).round(2)
3 risk_summary
```

คำอธิบายเพิ่มเติม:

1. `df['Risk_Level'].value_counts()`:
 - นับจำนวนลูกค้าในแต่ละระดับความเสี่ยง (High, Medium, Low)
2. `.to_frame(name='Count')`:
 - แปลงผลลัพธ์เป็น DataFrame และตั้งชื่อคอลัมน์ว่า Count
3. `.value_counts(normalize=True)`:
 - คำนวณ สัดส่วน (เปอร์เซ็นต์) ของแต่ละกลุ่ม โดย `normalize=True` หมายถึงให้หารด้วยจำนวนรวมทั้งหมด
4. `* 100).round(2)`:
 - คูณด้วย 100 เพื่อแปลงเป็นเปอร์เซ็นต์ และปัดเศษให้เหลือ 2 ตำแหน่ง

ตัวอย่างข้อมูล:

	Count	Percentage
Risk_Level		
High Risk	467	46.7
Low Risk	367	36.7
Medium Risk	166	16.6

การจัด Risk Segmentation ช่วยให้:

- เข้าใจภาพรวมของพอร์ตลูกค้า
- วางแผนควบคุมความเสี่ยงอย่างแม่นยำ

การประยุกต์ใช้ข้อมูลในงานตรวจสอบภายใน

จากการวิเคราะห์ข้อมูลในกรณีศึกษาจะเห็นได้ว่า Data Analytics ไม่ได้เป็นเพียงเครื่องมือด้านเทคนิคเพียงอย่างเดียว แต่สามารถนำมาใช้ในงาน Internal Audit ได้อย่างมีประสิทธิภาพ โดยเฉพาะในสายงานที่เกี่ยวข้องกับการอนุมัติสินเชื่อ หรือ การบริหารความเสี่ยงทางการเงิน ข้อมูลเหล่านี้สามารถช่วยทีม Audit ได้ในหลายด้าน เช่น:

1. ตรวจสอบความถูกต้องตามนโยบาย (Compliance Check)

- ใช้ข้อมูลย้อนหลังเพื่อตรวจสอบว่า กระบวนการอนุมัติสินเชื่อ เป็นไปตามเกณฑ์ที่บริษัทกำหนดหรือไม่ เช่น NCB Score ต้องมากกว่า 600 หรือผ่อนไม่เกิน 50% ของรายได้

2. วิเคราะห์พฤติกรรมและจัดลำดับความเสี่ยง (Risk Prioritization)

- นำข้อมูล เช่น Installment-to-Income Ratio และ NCB Score มาวิเคราะห์เพื่อจัดกลุ่มความเสี่ยงของลูกค้า ทำให้สามารถเลือกกลุ่มที่ต้อง "ตรวจสอบเชิงลึก" ได้แม่นยำยิ่งขึ้น

3. ค้นหาแนวโน้มผิดปกติ (Anomaly Detection)

- ตรวจเจอลูกค้าที่พฤติกรรมไม่สอดคล้องกับสถานะเครดิต
- เจอความเสี่ยงที่อาจ “ซ่อนอยู่” เช่น รายได้ไม่สัมพันธ์กับค่างวด หรือ NCB ต่ำผิดปกติ

4. เพิ่มประสิทธิภาพการตรวจสอบ (Smart Sampling)

- เลือกตัวอย่างเพื่อตรวจสอบแบบมีเหตุผล (Risk-based Audit) แทนการสุ่มแบบเดิม
- ตรวจเฉพาะกลุ่มเสี่ยงจริง = ประหยัดเวลา + ได้ผลลัพธ์แม่นยำ

