Exploratory Data Analysis - Data Preparation

6604062636348

นายธีรภัทร์ โคตะมาร

```
[8]  # Install and Import ydata-profiling (Auto EDA tools)
     !pip install ydata_profiling
     from ydata_profiling import ProfileReport
```

Show hidden output

> ติดตั้ง library "ydata_profiling" ใน Google colab

> นำเข้า class "ProfileReport" จาก library "ydata_profiling"

```
[6]  import numpy as np
     import tensorflow as tf
     from tensorflow import keras
     import pandas as pd
     from matplotlib import pyplot as plt
     %matplotlib inline
```

> นำเข้า library "Numpy" และตั้งชื่อเป็น np ใช้จัดการตัวเลขและคณิตศาสตร์ขั้นสูง

> นำเข้า library "Tensorflow" และตั้งชื่อเป็น tf ใช้ Machine Learning หรือ Deep Learning

> นำเข้า Module "keras" จาก "Tensorflow" เพื่อใช้งานและสร้าง Model "Neural Network"

> นำเข้า library "Pandas" และตั้งชื่อเป็น pd เพื่อใช้จัดการและวิเคราะห์ข้อมูลในรูปแบบตาราง (Data Flow)

> นำเข้า Module "pyplot" จาก library "Matplotlib" และตั้งชื่อย่อเป็น plt เพื่อสร้างกราฟและแสดงผลข้อมูลเชิงภาพ

// สำคัญเฉพาะใน Jupyter / Colab เพื่อให้กราฟที่สร้างด้วย "Matplotlib" แสดงผลใน notebook ทันที

```
[10] from google.colab import files
     uploaded = files.upload() # download CSV file and upload to your colab
```

Show hidden output

> นำเข้าฟังก์ชัน file จาก module google.colab เปิดเครื่องมือใช้จัดการไฟล์ใน google.colab

1. เรียกใช้ฟังก์ชัน upload() เพื่อเปิดไฟล์จากเครื่องเราไปยัง google colab

2. ไฟล์ที่อัปโหลดมา จะถูกเก็บในตัวแปร uploaded ในรูปแบบ dictionary โดยชื่อไฟล์เป็น คีย์ และเนื้อหาของไฟล์เป็นค่า

```
[11] df = pd. read_csv("/content/xxx_dataset.csv")
```

1. ใช้ฟังก์ชัน "read_csv" จาก library pandas เพื่อโหลดไฟล์ CSV ที่อยู่ใน path /content/xxx_dataset.csv

2 ไฟล์ CSV จะถูกแปลงเป็น DataFrame (ตารางข้อมูล) และถูกเก็บในตัวแปร df ซึ่งใช้กันทั่วไปสำหรับวิเคราะห์และจัดการข้อมูลใน Pythom

```
[12] # Generate the Profiling Report
     profile = ProfileReport(
         df, title = "xxx_dataset" , html={"style": {"full_width": True}}, sort=None
     )
```

> ทำหน้าที่ให้แบบรายงานผลลัพธ์แสดงผลแบบเต็มความกว้างของหน้าจอในรูปแบบ HTML

> ปิดการจัดเรียง column ในรายงาน (ใช้ลำดับ column เดิมใน DataFrame)

> ตั้งชื่อหัวข้อรายงานเป็น "xxx_dataset"

> ระบุ DataFrame (df) ที่ต้องการวิเคราะห์

1. สร้าง "รายงานการวิเคราะห์ข้อมูล (Data Profiling)" โดยใช้ "ProfileReport" จาก library "ydata_profiling"

2. รายงานประกอบด้วยข้อมูล สถิติ, การแจกแจงข้อมูล, และ การตรวจสอบค่าที่ขาดหาย (missing values)

```
[13] # The HTML report in an iframe
     profile.to_notebook_iframe()
```

> แสดงรายงานวิเคราะห์ข้อมูล profile ในรูปแบบ iframe ในรูปแบบ Jupyter Notebook หรือ Google Colab

```
[14] df.shape
(400, 11)  // ค่าที่คืนเป็น tuple ในรูปแบบ (จำนวนแถว, จำนวนคอลัมน์)
```

> เป็นคำสั่งใช้สำหรวจสอบขนาดของ DataFrame df

```
[15] df.columns
Index(['gender', 'age', 'hypertension', 'heart_disease', 'Marriage',
       'work_type', 'Living_type', 'avg_glucose', 'bmi', 'smoking_status',
       'illness'],
      dtype='object')  // ค่าที่คืนเป็น index object ที่แสดงชื่อคอลัมน์ในรูปแบบลิสต์ (list-like)
```

> ใช้สำหรับ ดูแบบชื่อ คอลัมน์ทั้งหมดใน DataFrame df

```
[16] df.isnull().any()
```

> ใช้ตรวจสอบว่ามี ค่าที่หายไป (missing values) ในแต่ละ column ของ DataFrame df หรือไม่

	0
gender	False
age	False
hypertension	False
heart_disease	False
Marriage	False
work_type	False
Living_type	False
avg_glucose	False
bmi	False
smoking_status	False
illness	False

dtype: bool

> คืนค่าเป็น Series โดยแต่ละค่าจะระบุว่า True หรือ False

- True มีค่า missing ใน column นั้น
- False ไม่มีค่า missing ใน column นั้น

```
[17] df.nunique()
```

> ใช้เพื่อ นับจำนวน ค่าที่ไม่ซ้ำ (unique value) ในแต่ละ column ของ DataFrame df

	0
gender	2
age	79
hypertension	2
heart_disease	2
Marriage	2
work_type	4
Living_type	2
avg_glucose	393
bmi	199
smoking_status	4
illness	2

dtype: int64

> คืนค่าที่เป็น series ที่แสดงจำนวน ค่าที่ไม่ซ้ำ ในแต่ละ คอลัมน์

```
[18] df.describe(include = 'all')
```

	gender	age	hypertension	heart_disease	Marriage	work_type	Living_type	avg_glucose	bmi	smoking_status	illness
count	400	400.00000	400.000000	400.000000	400	400	400	400.000000	400.000000	400	400.000000
unique	2	NaN	NaN	NaN	2	4	2	NaN	NaN	4	NaN
top	Female	NaN	NaN	NaN	Yes	Private	Urban	NaN	NaN	never smoked	NaN
freq	214	NaN	NaN	NaN	306	231	201	NaN	NaN	164	NaN
mean	NaN	55.26780	0.180000	0.132500	NaN	NaN	NaN	119.391950	29.481750	NaN	0.500000
std	NaN	22.51279	0.384669	0.339458	NaN	NaN	NaN	54.377459	6.488354	NaN	0.500626
min	NaN	0.80000	0.000000	0.000000	NaN	NaN	NaN	56.070000	15.600000	NaN	0.000000
25%	NaN	44.00000	0.000000	0.000000	NaN	NaN	NaN	80.460000	25.575000	NaN	0.000000
50%	NaN	59.00000	0.000000	0.000000	NaN	NaN	NaN	97.665000	28.600000	NaN	0.500000
75%	NaN	74.25000	0.000000	0.000000	NaN	NaN	NaN	144.345000	33.025000	NaN	1.000000
max	NaN	82.00000	1.000000	1.000000	NaN	NaN	NaN	271.740000	48.900000	NaN	1.000000

1. ใช้เพื่อ ให้ข้อมูลสถิติเบื้องต้น สำหรับทุก คอลัมน์ใน Data Frame df

2. ตำสั่งนี้จะ สร้างตาราง สถิติพื้นฐาน เช่น ค่าเฉลี่ย , ค่า Max , ค่า Min , ส่วนเบี่ยงเบนมาตรฐาน สำหรับ คอลัมน์ ที่เป็น ตัวเลข

3. การใช้ include = 'all' จะรวมทั้ง column ที่เป็น ตัวเลข และไม่ใช่ตัวเลข (เช่น column ที่เป็นข้อความหรือหมวดหมู่)

และ สร้างตาราง สถิติ ขึ้นมา เช่น จำนวน สถิติที่ไม่ขาด , จำนวนสถิติที่ไม่เป็น NULL , ตรวจดูข้อมูล สถิติที่พบบ่อยที่สุด

```
[94] df['gender'] = df['gender'] .map({'Female' :1, 'Male' :0})
     df['Marriage'] = df['Marriage'].map({'Yes': 1, 'No': 0})
     df['work_type'] = df['work_type'].map({'Private': 0, 'Self-employed': 1, 'Govt_job' :2, 'children' :3})
     df['Living_type'] = df['Living_type'].map({'Urban': 1, 'Rural': 0})
     df['smoking_status'] = df['smoking_status'].map({'formerly smoked' :0, 'never smoked' :1, 'smoke' :2,'Unknown' :3})
```

line 1 • เปลี่ยน ค่า gender โดยแปลง ค่า Female เป็น 1 และ Male เป็น 0

• ใช้ฟังก์ชัน map() เพื่อทำการแมป ค่าตาม dictionary ที่ระบุ

line 2 • เปลี่ยน ค่าของ Marriage โดยแปลงค่า Yes เป็น 1 และ No เป็น 0

• ใช้ map() เพื่อแปลงข้อมูลสถานะการแต่งงาน

line 3 • เปลี่ยน ค่าของ column work_type โดยแปลงแต่ละประเภทงานเป็นค่าตัวเลข เช่น Private เป็น 0, Self-employed เป็น 1 เป็นต้น

line 4 • เปลี่ยน ค่าของ column Living_type โดยแปลง Urban เป็น 1 และ Rural เป็น 0

• ใช้เพื่อแปลงประเภทที่อยู่อาศัย

line 5 • เปลี่ยน ค่าของ column smoking_status โดยแปลงค่า formerly smoked เป็น 0, never smoked เป็น 1, smoked เป็น 2, และ Unknown เป็น 3

```
[95] df
```

	gender	age	hypertension	heart_disease	Marriage	work_type	Living_type	avg_glucose	bmi	smoking_status	illness
0	0	67.0	0	1	1	0	1	228.69	36.6	0	1
1	0	80.0	0	1	1	0	0	105.92	32.5	1	1
2	1	49.0	0	0	1	0	1	171.23	34.4	2	1
3	1	79.0	1	0	1	1	0	174.12	24.0	1	1
4	0	81.0	0	0	1	0	1	186.21	29.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...
395	1	54.0	0	1	1	0	1	140.28	37.1	0	0
396	0	67.0	0	0	1	2	0	244.28	29.4	0	0
397	0	53.0	0	0	1	0	1	124.16	31.7	1	0
398	0	47.0	0	0	1	0	0	93.55	31.4	1	0
399	0	44.0	0	0	1	0	1	99.34	33.1	1	0

→ Column ใหม่ที่ถูกแปลงเป็นตัวเลข และพร้อมใช้งานในกระบวนการวิเคราะห์หรือการสร้าง Model Machine Learning ได้อย่างมีประสิทธิภาพ

```
[23] y = df ['illness']
     x = df.drop(columns=['illness'])
```

> กำหนดตัวแปร y เป็น column illness ของ DataFrame df

- กำหนดตัวแปร x เป็น DataFrame ที่ไม่รวม column illness มาจาก df
- ต่างกัน drop (columns = ['illness']) จะทำการลบ column illness ออกจาก df

```
[97] x.head()
```

> แสดงข้อมูล 5 แถวแรก DataFrame x ซึ่งเป็นข้อมูลที่ใช้เป็น features สำหรับการฝึกโมเดล

	gender	age	hypertension	heart_disease	Marriage	work_type	Living_type	avg_glucose	bmi	smoking_status
0	0	67.0	0	1	1	0	1	228.69	36.6	0
1	0	80.0	0	1	1	0	0	105.92	32.5	1
2	1	49.0	0	0	1	0	1	171.23	34.4	2
3	1	79.0	1	0	1	1	0	174.12	24.0	1
4	0	81.0	0	0	1	0	1	186.21	29.0	0

```
[99] y.value_counts()
```

> นับจำนวนของแต่ละค่าในตัวแปร y ซึ่งตอนนี้ y คือ column illness ดังนั้นจึงนับว่าในคอลัมน์นี้มีค่า 1 และ 0 อย่างละกี่ตัว (สำคัญมากสำหรับ target variable ที่ต้องการทำนาย)

illness	count
1	200
0	200

dtype: int64

```python
from sklearn. preprocessing import MinMaxScaler # line 1

scale=MinMaxScaler() # line 2
x_scaled=pd.DataFrame(scale.fit_transform(x),columns=x.columns) # line 3
x_scaled # line 4
```

	gender	age	hypertension	heart_disease	Marriage	work_type	Living_type	avg_glucose	bmi	smoking_status
0	0.0	0.815271	0.0	1.0	1.0	0.000000	1.0	0.800389	0.630631	0.000000
1	0.0	0.975369	0.0	1.0	1.0	0.000000	0.0	0.231140	0.507508	0.333333
2	1.0	0.593596	0.0	0.0	1.0	0.000000	1.0	0.533964	0.564565	0.666667
3	1.0	0.963054	1.0	0.0	1.0	0.333333	0.0	0.547364	0.252252	0.333333
4	0.0	0.987685	0.0	0.0	1.0	0.000000	1.0	0.603422	0.402402	0.000000
...	...	...	...	...	...	...	...	...	...	...
395	1.0	0.655172	0.0	1.0	1.0	0.000000	1.0	0.390458	0.645646	0.000000
396	0.0	0.815271	0.0	0.0	1.0	0.666667	0.0	0.872676	0.414414	0.000000
397	0.0	0.642857	0.0	0.0	1.0	0.000000	1.0	0.315714	0.483483	0.333333
398	0.0	0.568966	0.0	0.0	1.0	0.000000	0.0	0.173784	0.474474	0.333333
399	0.0	0.532020	0.0	0.0	1.0	0.000000	1.0	0.200631	0.525526	0.333333

400 rows × 10 columns

line 1 นำเข้า MinMax Scaler จาก sklearn . preprocessing ซึ่งใช้ในการสเกลข้อมูลให้มีค่าระหว่างช่วงที่กำหนด (โดยปกติจะเป็น 0 ถึง 1) เพื่อให้ข้อมูลเหมาะสมกับการใช้งาน model Machine Learning เช่น การคำนวณระยะทาง อัลกอริทึมที่มีค่ามากจะมีอิทธิพลเกินไป

line 2 สร้างตัวแปร scale เป็น object ของ MinMax Scaler ซึ่งจะใช้ในการสเกลข้อมูล

line 3 • ใช้ fit_transform (x) เพื่อคำนวณค่าขั้นต่ำและสูงสุดของข้อมูลใน DataFrame x และเปลี่ยนข้อมูลใน x ให้มีค่าอยู่ในช่วง 0 ถึง 1 โดยใช้ MaxMin Scaler

• หลังจากนั้น แปลงผลลัพธ์ ให้เป็น DataFrame ใหม่ x_scaled โดยใช้ชื่อ columns เหมือนกับ x.columns

line 4 • ค่าของข้อมูลใน x_scaled จะถูกสเกลให้อยู่ในช่วง [0,1] ซึ่งช่วยลดปัญหาของฟีเจอร์ที่มีค่าต่างกันมาก

• ข้อมูลนี้ก็จะถูกนำไปใช้ใน model Machine Learning เพื่อให้ model สามารถเรียนรู้ข้อมูลได้ดีขึ้น

```
[103] from sklearn.model_selection import train_test_split // line 1
      x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.30, random_state=42) // line 2
```

line 1 นำเข้าข้อมูล train_test_split จาก sklearn.model_selection ซึ่งเป็นฟังก์ชันใช้แบ่งข้อมูลออกเป็นชุดฝึก (training set) และชุดทดสอบ (test set) สำหรับการฝึกและทดสอบ model Machine Learning

line 2
- train_test_split() จะสุ่มแบ่งข้อมูล x_scaled และ y โดยใช้ สัดส่วน ที่กำหนดใน test_size (ในกรณีนี้ คือ 30% สำหรับชุดทดสอบ)
- ชุดข้อมูล ที่ได้จาก train_test_split() จะถูกแบ่งเป็น ตัวนี้
  - x_train ข้อมูลฟีเจอร์ สำหรับใช้ฝึก
  - x_test ข้อมูลฟีเจอร์ สำหรับชุดทดสอบ
  - y_train ค่าที่ต้องการทำนาย (target variable หรือ label) สำหรับชุดฝึก
  - y_test ค่าที่ต้องการทำนาย สำหรับชุดทดสอบ
- test_size = 0.30 กำหนดขนาดของชุดทดสอบให้เป็น 30% ของข้อมูลทั้งหมด (ส่วนที่เหลือ จะเป็นชุดฝึก 70%)
- random_state = 42 กำหนดค่า random_state เพื่อให้การแบ่งข้อมูลเป็นแบบรูปแบบเดียวกัน ทุกครั้งที่ run code (เหมือนการตั้งค่า seed เพื่อความมั่นคงในการแบ่งข้อมูล)

Example

สมมุติว่าเรามีข้อมูลฟีเจอร์ x_scaled และข้อมูล target y

- เมื่อ run code ข้อมูลจะถูกแบ่งเป็น 2 ชุด
  1. 70% ของข้อมูล จะใช้ในการ ฝึกโมเดล (train)
  2. 30% ของข้อมูล จะใช้ในการ ทดสอบโมเดล (test)

data-prep.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

RAM Disk

Gemini

```
# Install and Import ydata-profiling (Auto EDA tools)
!pip install ydata_profiling
from ydata_profiling import ProfileReport
```

```
Requirement already satisfied: ydata_profiling in /usr/local/lib/python3.10/dist-packages (4.12.1)
Requirement already satisfied: scipy<1.14,>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (1.13.1)
Requirement already satisfied: pandas!=1.4.0,<3,>1.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (2.2.2)
Requirement already satisfied: matplotlib<3.10,>=3.5 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (3.9.4)
Requirement already satisfied: pydantic>=2 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (2.10.4)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (6.0.2)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (3.1.5)
Requirement already satisfied: visions<0.7.7,>=0.7.5 in /usr/local/lib/python3.10/dist-packages (from visions[type_image_path]<0.7.7,>=0.7.5->ydata_profiling) (0.7.6)
Requirement already satisfied: numpy<2.2,>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (1.26.4)
Requirement already satisfied: htmlmin==0.1.12 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (0.1.12)
Requirement already satisfied: phik<0.13,>=0.11.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (0.12.4)
Requirement already satisfied: requests<3,>=2.24.0 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (2.32.3)
Requirement already satisfied: tqdm<5,>=4.48.2 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (4.67.1)
Requirement already satisfied: seaborn<0.14,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (0.13.2)
Requirement already satisfied: multimethod<2,>=1.4 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (1.12)
Requirement already satisfied: statsmodels<1,>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (0.14.4)
Requirement already satisfied: typeguard<5,>=3 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (4.4.1)
Requirement already satisfied: imagehash==4.3.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (4.3.1)
Requirement already satisfied: wordcloud>=1.9.3 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (1.9.4)
Requirement already satisfied: dacite>=1.8 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (1.8.1)
Requirement already satisfied: numba<1,>=0.56.0 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (0.60.0)
Requirement already satisfied: PyWavelets in /usr/local/lib/python3.10/dist-packages (from imagehash==4.3.1->ydata_profiling) (1.8.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from imagehash==4.3.1->ydata_profiling) (11.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2<3.2,>=2.11.1->ydata_profiling) (3.0.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.10,>=3.5->ydata_profiling) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.10,>=3.5->ydata_profiling) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.10,>=3.5->ydata_profiling) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.10,>=3.5->ydata_profiling) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.10,>=3.5->ydata_profiling) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.10,>=3.5->ydata_profiling) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.10,>=3.5->ydata_profiling) (2.8.2)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba<1,>=0.56.0->ydata_profiling) (0.43.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=1.4.0,<3,>1.1->ydata_profiling) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas!=1.4.0,<3,>1.1->ydata_profiling) (2024.2)
Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.10/dist-packages (from phik<0.13,>=0.11.1->ydata_profiling) (1.4.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2->ydata_profiling) (0.7.0)
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2->ydata_profiling) (2.27.2)
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2->ydata_profiling) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata_profiling) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata_profiling) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata_profiling) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata_profiling) (2024.12.14)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels<1,>=0.13.2->ydata_profiling) (1.0.1)
Requirement already satisfied: attrs>=19.3.0 in /usr/local/lib/python3.10/dist-packages (from visions<0.7.7,>=0.7.5->visions[type_image_path]<0.7.7,>=0.7.5->ydata_profiling) (24.3.0)
Requirement already satisfied: networkx>=2.4 in /usr/local/lib/python3.10/dist-packages (from visions<0.7.7,>=0.7.5->visions[type_image_path]<0.7.7,>=0.7.5->ydata_profiling) (3.4.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib<3.10,>=3.5->ydata_profiling) (1.17.0)
```

```
[84] import numpy as np
     import tensorflow as tf
     from tensorflow import keras
     import pandas as pd
     from matplotlib import pyplot as plt
     %matplotlib inline
```

```
[85] from google.colab import files
     uploaded = files.upload() # download CSV file and upload to your colab
```

Choose Files xxx_dataset.csv

- **xxx_dataset.csv**(text/csv) - 23316 bytes, last modified: 1/9/2025 - 100% done

Saving xxx_dataset.csv to xxx_dataset.csv

```
[86] df = pd. read_csv("/content/xxx_dataset.csv")
```

```
[87] # Generate the Profiling Report
     profile = ProfileReport(
         df, title = "xxx_dataset" , html={"style": {"full_width": True}}, sort=None
     )
```

```
[88] # The HTML report in an iframe
     profile.to_notebook_iframe()
```

Summarize dataset: 100% 29/29 [00:03<00:00, 4.04it/s, Completed]

Generate report structure: 100% 1/1 [00:03<00:00, 3.91s/it]

Render HTML: 100% 1/1 [00:00<00:00, 1.38it/s]

xxx_dataset

Overview Variables Interactions Correlations Missing values Sample

Overview

Brought to you by YData

Overview Alerts 5 Reproduction

Dataset statistics

Number of variables	11
Number of observations	400
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	34.5 KiB
Average record size in memory	88.3 B

Variable types

Categorical	7
Numeric	3
Boolean	1

Variables

Select Columns

```
[89] df.shape
```

```
(400, 11)
```

```
[90] df.columns
```

```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'Marriage',
       'work_type', 'Living_type', 'avg_glucose', 'bmi', 'smoking_status',
       'illness'],
      dtype='object')
```

```
[91] df.isnull().any()
```

	0
gender	False
age	False
hypertension	False
heart_disease	False
Marriage	False
work_type	False
Living_type	False
avg_glucose	False
bmi	False
smoking_status	False
illness	False

dtype: bool

```
[92] df.nunique()
```

	0
gender	2
age	79
hypertension	2
heart_disease	2
Marriage	2
work_type	4
Living_type	2
avg_glucose	393
bmi	199
smoking_status	4
illness	2

dtype: int64

```
[93] df.describe(include = 'all')
```

	gender	age	hypertension	heart_disease	Marriage	work_type	Living_type	avg_glucose	bmi	smoking_status	illness
count	400	400.00000	400.000000	400.000000	400	400	400	400.000000	400.000000	400	400.000000
unique	2	NaN	NaN	NaN	2	4	2	NaN	NaN	4	NaN
top	Female	NaN	NaN	NaN	Yes	Private	Urban	NaN	NaN	never smoked	NaN
freq	214	NaN	NaN	NaN	306	231	201	NaN	NaN	164	NaN
mean	NaN	55.26780	0.180000	0.132500	NaN	NaN	NaN	119.391950	29.481750	NaN	0.500000
std	NaN	22.51279	0.384669	0.339458	NaN	NaN	NaN	54.377459	6.488354	NaN	0.500626
min	NaN	0.80000	0.000000	0.000000	NaN	NaN	NaN	56.070000	15.600000	NaN	0.000000
25%	NaN	44.00000	0.000000	0.000000	NaN	NaN	NaN	80.460000	25.575000	NaN	0.000000
50%	NaN	59.00000	0.000000	0.000000	NaN	NaN	NaN	97.665000	28.600000	NaN	0.500000
75%	NaN	74.25000	0.000000	0.000000	NaN	NaN	NaN	144.345000	33.025000	NaN	1.000000
max	NaN	82.00000	1.000000	1.000000	NaN	NaN	NaN	271.740000	48.900000	NaN	1.000000

```
[94] df[ 'gender'] = df[ 'gender'] .map({'Female' :1, 'Male' :0})
     df['Marriage'] = df['Marriage'].map({'Yes': 1, 'No': 0})
     df['work_type'] = df['work_type'].map({'Private': 0, 'Self-employed': 1, 'Govt_job' :2, 'children' :3})
     df['Living_type'] = df['Living_type'].map({'Urban': 1, 'Rural': 0})
     df['smoking_status'] = df['smoking_status'].map({'formerly smoked' :0, 'never smoked' :1, 'smokes' :2,'Unknown' :3})
```

```
[95] df
```

	gender	age	hypertension	heart_disease	Marriage	work_type	Living_type	avg_glucose	bmi	smoking_status	illness
0	0	67.0	0	1	1	0	1	228.69	36.6	0	1
1	0	80.0	0	1	1	0	0	105.92	32.5	1	1
2	1	49.0	0	0	1	0	1	171.23	34.4	2	1
3	1	79.0	1	0	1	1	0	174.12	24.0	1	1
4	0	81.0	0	0	1	0	1	186.21	29.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...

395	1	54.0	0	1	1	0	1	140.28	37.1	0	0
396	0	67.0	0	0	1	2	0	244.28	29.4	0	0
397	0	53.0	0	0	1	0	1	124.16	31.7	1	0
398	0	47.0	0	0	1	0	0	93.55	31.4	1	0
399	0	44.0	0	0	1	0	1	99.34	33.1	1	0

400 rows × 11 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
[96] y = df ['illness']
     x = df.drop(columns=['illness'])
```

```
[97] x.head()
```

	gender	age	hypertension	heart_disease	Marriage	work_type	Living_type	avg_glucose	bmi	smoking_status
0	0	67.0	0	1	1	0	1	228.69	36.6	0
1	0	80.0	0	1	1	0	0	105.92	32.5	1
2	1	49.0	0	0	1	0	1	171.23	34.4	2
3	1	79.0	1	0	1	1	0	174.12	24.0	1
4	0	81.0	0	0	1	0	1	186.21	29.0	0

Next steps: Generate code with x | View recommended plots | New interactive sheet

```
[99] y.value_counts()
```

illness	count
1	200
0	200

dtype: int64

```
from sklearn. preprocessing import MinMaxScaler

scale=MinMaxScaler()
x_scaled=pd.DataFrame(scale.fit_transform(x),columns=x.columns)
x_scaled
```

	gender	age	hypertension	heart_disease	Marriage	work_type	Living_type	avg_glucose	bmi	smoking_status
0	0.0	0.815271	0.0	1.0	1.0	0.000000	1.0	0.800389	0.630631	0.000000
1	0.0	0.975369	0.0	1.0	1.0	0.000000	0.0	0.231140	0.507508	0.333333
2	1.0	0.593596	0.0	0.0	1.0	0.000000	1.0	0.533964	0.564565	0.666667
3	1.0	0.963054	1.0	0.0	1.0	0.333333	0.0	0.547364	0.252252	0.333333
4	0.0	0.987685	0.0	0.0	1.0	0.000000	1.0	0.603422	0.402402	0.000000
...	...	...	...	...	...	...	...	...	...	...
395	1.0	0.655172	0.0	1.0	1.0	0.000000	1.0	0.390458	0.645646	0.000000
396	0.0	0.815271	0.0	0.0	1.0	0.666667	0.0	0.872676	0.414414	0.000000
397	0.0	0.642857	0.0	0.0	1.0	0.000000	1.0	0.315714	0.483483	0.333333
398	0.0	0.568966	0.0	0.0	1.0	0.000000	0.0	0.173784	0.474474	0.333333
399	0.0	0.532020	0.0	0.0	1.0	0.000000	1.0	0.200631	0.525526	0.333333

400 rows × 10 columns

Next steps: Generate code with x_scaled | View recommended plots | New interactive sheet

```
[103] from sklearn.model_selection import train_test_split
      x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.30, random_state=42)
```