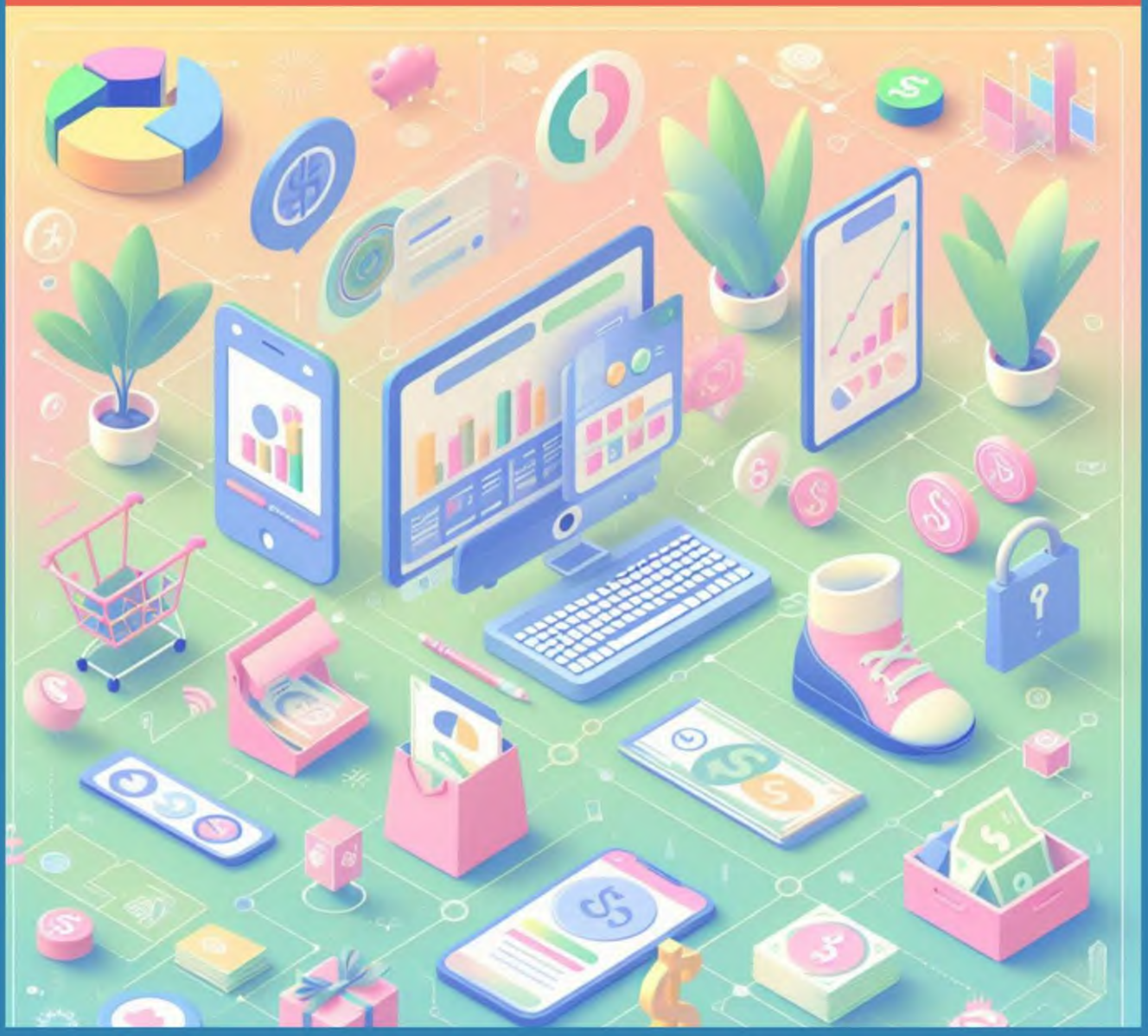
Data Science
วิเคราะห์การตลาด
ด้วย Python



มีโค้ดตัวอย่าง พร้อมใช้งาน

ผศ.ประกายรัตน์ วิเศษสงวน และอมรวิทย์ วิเศษสงวน

Data Science วิเคราะห์การตลาด ด้วย Python

ปรับปรุงล่าสุด: มกราคม 2567
ราคา: 350 บาท

ข้อมูลทางบรรณานุกรมของสำนักหอสมุดแห่งชาติ

ประกายรัตน์ วิเศษสงวน.
Data Science วิเคราะห์การตลาด ด้วย Python.-- กรุงเทพฯ : ม.ป.พ., 2567.
505 หน้า.

1. การตลาด -- การวิเคราะห์. 2. ไพธอน (ภาษาคอมพิวเตอร์).
I. อมรวิทย์ วิเศษสงวน, ผู้แต่งร่วม. II. ชื่อเรื่อง.

658.83
ISBN 978-616-608-216-6

ติดต่อผู้เขียน: อมรวิทย์ วิเศษสงวน จัดทำโดย: ประกายรัตน์ วิเศษสงวน
อีเมล: amornwit.bookwork@gmail.com
เว็บไซต์: https://github.com/prakayrat/MarketingAnalyticsWithPython หรือ
https://bit.ly/MarketingAnalyticsPython หรือสแกน QR code นี้
(ดาวน์โหลด code, data และอื่นๆ ได้ที่นี่)

เกี่ยวกับหนังสือ

คู่มือเชิงปฏิบัติ สำหรับการจัดการทางการตลาด โดยใช้การวิเคราะห์ข้อมูล ด้วย Python หนังสือจะช่วยให้ผู้อ่านสามารถใช้ประโยชน์จากชุดข้อมูล ฝึกใช้กับโครงการวิเคราะห์ตลาดในโลกแห่งความเป็นจริง

ผู้อ่านจะได้เรียนรู้วิธีคิดเหมือน data scientist สร้างทักษะในการแก้ปัญหา ค้นพบวิธีการดูข้อมูลในรูปแบบใหม่ๆ เพื่อนำเสนอข้อมูลเชิงลึกในทางธุรกิจ (business insights) และการใช้ข้อมูลอย่างชาญฉลาดในการตัดสินใจ (intelligent data-driven decisions)

นอกจากนี้ ผู้อ่านจะได้เรียนรู้วิธีการ clean ข้อมูล การสำรวจข้อมูล (explore data) และการแสดงผลข้อมูลด้วยภาพ (visualize data) การใช้อัลกอริทึมแมชชีนเลิร์นนิ่ง (machine learning algorithms) เพื่อสร้างโมเดล ช่วยในการคาดการณ์ การจำแนกประเภท การจัดกลุ่ม และการตัดสินใจ โดยใช้เครื่องมือของ Python

วิเคราะห์ข้อมูลทางการตลาด เช่น เพื่อติดตามกิจกรรมการตลาดต่างๆ เพื่อให้เข้าใจว่าอะไรที่ได้ผล หรือเพื่อปรับปรุงกลยุทธ์การตลาดให้เหมาะสมมากขึ้น หรืออื่นๆ และยังสามารถใช้เครื่องมือและทักษะต่างๆ เพื่อการทำงานได้อย่างมีประสิทธิภาพ

ในหนังสือได้จัดเตรียมโค้ด Python ที่ใช้งานจริงและผ่านการทดสอบอย่างดี ช่วยให้ผู้อ่านสามารถคัดลอกและวางโค้ด พร้อมปรับใช้ตามที่ผู้อ่านต้องการใช้ในโลกแห่งความจริง

นอกจากนี้หนังสือยังมีตัวอย่าง และกิจกรรม ที่ให้ฝึกปฏิบัติ การวิเคราะห์ทางการตลาดและวิธีการสร้างผลกระทบที่วัดผลได้ต่อธุรกิจทั้งขนาดใหญ่ กลาง และเล็ก

หลังจากเรียนรู้เล่มนี้จบ ผู้อ่านจะได้รับความรู้ ทักษะ และความเชื่อมั่น ในการดำเนินการด้านวิทยาการข้อมูล และเทคนิคของแมชชีนเลิร์นนิ่ง เข้าใจข้อมูลทางการตลาด และเพิ่มประสิทธิภาพการตัดสินใจ

เนื้อหาโดยสังเขป

คำนำ

หนังสือเล่มนี้ จะช่วยให้ผู้อ่านเริ่มต้นการเดินทางสู่การเป็นผู้เชี่ยวชาญด้านการวิเคราะห์การตลาด ด้วย Python ผู้อ่านจะได้ทำงานกับชุดข้อมูลที่เกี่ยวข้องและสร้างทักษะการปฏิบัติของผู้อ่าน โดยการฝึกการจัดการกับตัวอย่าง และกิจกรรม ที่จำลองโครงการวิเคราะห์ตลาดในโลกแห่งความเป็นจริง

ผู้อ่านจะได้เรียนรู้การเตรียมข้อมูล การใช้อัลกอริทึมแมชชีนเลิร์นนิ่ง ในการสร้างโมเดลเพื่อทำการจำแนก และการคาดการณ์แล้ว ยังได้เรียนรู้การใช้งานโปรแกรม Python ในการวิเคราะห์ทางการตลาด เช่น การขาย การโฆษณา คาดการณ์รายได้ การจัดการกับการเลิกใช้งานของลูกค้า และใช้การแบ่งกลุ่มลูกค้า รวมทั้งทำความเข้าใจพฤติกรรมของลูกค้า

หนังสือเล่มนี้เหมาะกับผู้ที่ต้องการเรียนรู้วิธีใช้ Python สำหรับการวิเคราะห์การตลาดที่ทันสมัย ไม่ว่าผู้อ่านจะเป็นนักพัฒนาซอฟต์แวร์ที่ต้องการเข้าสู่ตลาด หรือนักวิเคราะห์การตลาดที่ต้องการเรียนรู้เครื่องมือและเทคนิคที่ซับซ้อนมากขึ้น หรือต้องการเรียนรู้เพิ่มเติมทางด้านนี้ หนังสือเล่มนี้จะพาผู้อ่านไปสู่เส้นทางนั้น

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

บทที่ 1

Anaconda สำหรับวิเคราะห์ข้อมูล

สาระสำคัญในบทนี้

- Anaconda และ Python
  - การติดตั้ง Anaconda
  - Python คืออะไร
  - Environment
  - ติดตั้ง JupyterLab
- Conda
- เตรียมข้อมูล

Anaconda และ Python

Anaconda เป็นบริษัทที่ก่อตั้งขึ้นในปี 2012 โดย Peter Wang และ Travis Oliphant เพื่อเป็นเครื่องมือในการวิเคราะห์ด้านวิทยาการข้อมูล รองรับการใช้งานได้หลายภาษาคอมพิวเตอร์ เช่น Python, R โดยเฉพาะ Python เป็นภาษาโปรแกรมที่ได้รับความนิยมมากที่สุดในปัจจุบัน และมีผู้ใช้งานกว่า 40 ล้านคน บริษัทนี้เป็นผู้นำด้าน Open Source และเป็นตัวกลางของการพัฒนา AI โดยให้เครื่องมือด้าน Data Science, MLOps และ Data & Model Management เพื่อเพิ่มศักยภาพให้กับลูกค้าและชุมชนของ Anaconda ด้วยความสามารถด้าน AI เพื่อขับเคลื่อนโครงการไปข้างหน้า

Anaconda ให้ใช้งานได้ฟรี สำหรับบุคคลและองค์กรที่มีพนักงานน้อยกว่า 200 คน ที่สิ่งสำคัญคือ มีเครื่องมือวิทยาการข้อมูลระดับองค์กรสำหรับทุกคน

Open-source software (OSS) ได้กลายเป็นรากฐานของกลุ่มเทคโนโลยีในหลากหลายอุตสาหกรรม ในช่วงทศวรรษที่ผ่านมา Anaconda ได้ทุ่มเงินหลายสิบล้านดอลลาร์ให้กับนวัตกรรมโอเพ่นซอร์ส และการบำรุงรักษาโครงการในรูปแบบของ เวลาของพนักงาน การบริจาคโดยตรง การสนับสนุนกิจกรรม และอื่นๆ อีกมากมาย เพราะเชื่อว่า OSS เป็นกุญแจสำคัญในการปลดล็อกนวัตกรรมแห่งอนาคต และจะสนับสนุนความเชื่อนี้ต่อไป

นอกจาก Anaconda แล้ว ยังมี PyCharm, Jupyter Notebook, ibi WebFOCUS, Spyder, MATLAB, Microsoft Visual Studio Code, IBM Watson Studio, DataRobot Path to AI Success, KNIME Analytics Platform, RapidMiner, Saturn Cloud, Spotfire Data Science, Dataiku DSS และอื่นๆ

เครื่องมือต่างๆ ของ Anaconda ที่ช่วยให้ผู้ใช้งาน สามารถวิเคราะห์ข้อมูลได้อย่างมีประสิทธิภาพ ประกอบไปด้วย

- **Anaconda Distribution** เป็นเครื่องมือที่ใช้ในการจัดการและติดตั้ง packages ต่างๆ ที่เกี่ยวข้องกับการวิเคราะห์ข้อมูล โดยเครื่องมือนี้มี packages ที่จำเป็นต้องใช้ในการวิเคราะห์ข้อมูลอยู่แล้ว และสามารถเพิ่ม packages อื่นๆ ได้ตามความต้องการ
- **Anaconda Navigator** เป็นเครื่องมือที่ใช้ในการจัดการและเรียกใช้เครื่องมือต่างๆ ที่เกี่ยวข้องกับการวิเคราะห์ข้อมูล โดยผู้ใช้งานสามารถเรียกใช้เครื่องมือต่างๆ ได้ผ่านทาง GUI ที่ใช้งานง่าย

- **Jupyter Notebook** เป็นเครื่องมือที่ใช้ในการเขียนโปรแกรมและวิเคราะห์ข้อมูล โดยเครื่องมือนี้สามารถเขียนโปรแกรม Python ได้ และสามารถแสดงผลข้อมูลในรูปแบบต่างๆ ได้
- **Conda** เป็นเครื่องมือที่ใช้ในการจัดการและติดตั้ง package ต่างๆ ที่เกี่ยวข้องกับการวิเคราะห์ข้อมูล โดยเครื่องมือนี้สามารถติดตั้ง package ที่ต้องการได้ง่ายๆ และสามารถจัดการ package ที่ติดตั้งไว้แล้วได้

การติดตั้ง Anaconda

ก่อนอื่น ผู้อ่านต้องตัดสินใจว่าจะเลือก Anaconda Distribution หรือ Miniconda โดย **Anaconda Distribution** มีพร้อมทุกอย่างที่ผู้อ่านต้องการ ได้แก่ conda, command line package manager, Anaconda Navigator, desktop graphical user interface สำหรับเปิดแอปพลิเคชัน และจัดการกับ packages, environments และ channels

รวมทั้ง Python เวอร์ชันล่าสุดที่สนับสนุนโดย Anaconda Distribution และ packages ที่คัดสรรมากกว่า 350 ตัว สำหรับงานด้านวิทยาการข้อมูล (data science) และแมชชีนเลิร์นนิ่ง (machine learning)

สำหรับ **Miniconda** คล้ายคอมพิวเตอร์ที่สร้างจาก scratch ที่ผู้อ่านสามารถเลือกปรับตามที่ต้องการได้ Miniconda จะคล้ายกับ Anaconda Distribution รวมทั้ง Python เวอร์ชันล่าสุด แต่มีขนาดเล็กกว่า 10 เท่าของ Anaconda Distribution installer

ถ้าผู้อ่านเลือก Anaconda Distribution ให้ไปที่ https://www.anaconda.com/ และเลือกระบบปฏิบัติการตามที่ใช้งาน หรือเลือก OS ตามโลโก้  จากนั้นดำเนินการติดตั้งตามปกติ

ในหนังสือเล่มนี้ จะใช้ Windows

<u>หมายเหตุ</u> รายละเอียดขั้นตอนการติดตั้ง สามารถดูได้ที่ https://docs.anaconda.com/anaconda/install/

Python คืออะไร

Python เป็นภาษาสคริปต์ที่เรียบง่าย แต่ทรงพลัง และได้รับความนิยมเป็นอย่างมาก เนื่องจาก มีระบบโต้ตอบกับผู้อ่านในรูปแบบที่ง่าย และใช้งานง่าย ใช้ไวยากรณ์ที่สั้นและชัดเจน (clean and concise) สามารถดำเนินการที่ซับซ้อนด้วยคำเพียงไม่กี่คำ โครงสร้างภาษาที่ง่ายต่อการจดจำ การออกแบบมีขนาดกะทัดรัด มีไลบรารีที่มีประสิทธิภาพมากของฟังก์ชันที่มีประโยชน์ และมาพร้อมกับ packages ขนาดใหญ่

Python เป็น General purpose programming language หรือภาษาโปรแกรมเอนกประสงค์ เป็น Object-oriented programming language หรือภาษาการเขียนโปรแกรมเชิงวัตถุ และเป็น High-level, interpreted language หรือภาษาระดับสูง

ผู้อ่านสามารถใช้ Python สร้างเว็บไซต์ พัฒนาเกม ใช้งานด้าน computer vision เช่น ตรวจจับใบหน้าหรือสี ใช้งานกับแมชชีนเลิร์นนิ่ง และกระบวนการอัตโนมัติ (automate processes) ใช้งานหุ่นยนต์ เก็บข้อมูลจากเว็บไซต์ (web scraping) วิเคราะห์และจัดการข้อมูลด้วย Python libraries เช่น Pandas, NumPy และอื่นๆ อีกมากมาย

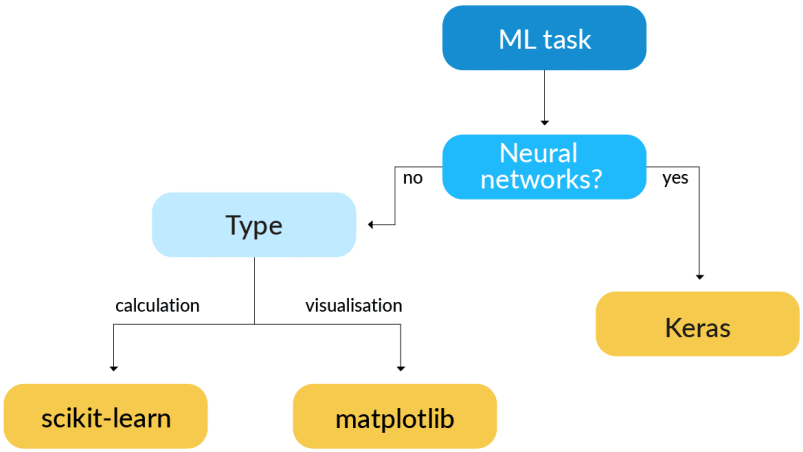
มีผู้ใช้ Python มากมาย เช่น

รูปที่ 1.1 ตัวอย่างบริษัทชั้นนำที่ใช้ Python

(ที่มา: https://www.logicfinder.net/2019/03/16/page/2/)

Python มีเครื่องมือการวิเคราะห์ (data analysis tools) มากมายให้เลือกใช้ เช่น Pandas, NumPy, SciPy รวมทั้งมีเครื่องมือในการใช้สำหรับ read และ write ข้อมูล การจัดการกับตัวเลข การแก้ไข (interpolation), Linear algebra และอื่นๆ

Python มีเครื่องมือแมชชีนเลิร์นนิ่ง (machine learning tools) เช่น TensorFlow, PyTorch, Keras, Scikit-learn ใช้กับ classification, regression, cluster, predictive analytics, deep learning models และอื่นๆ



รูปที่ 1.2 machine learning tools

(ที่มา: https://litslink.com/blog/python-for-machine-learning)

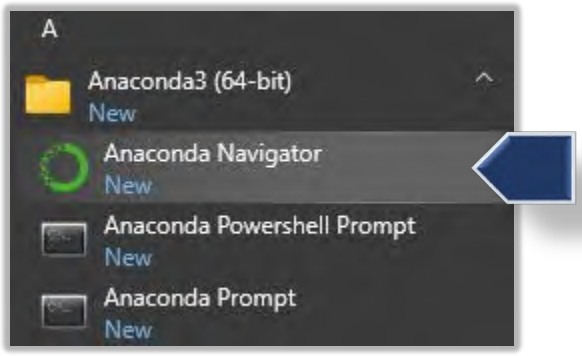
และ Python มีเครื่องมือแสดงภาพข้อมูล (data visualization tools) เช่น Matplotlib, bokeh, plotly, streamlit แสดงภาพด้วยข้อมูล (visualizations), dashboard และ Widgets และอื่นๆ

รูปที่ 1.3 data visualization tools

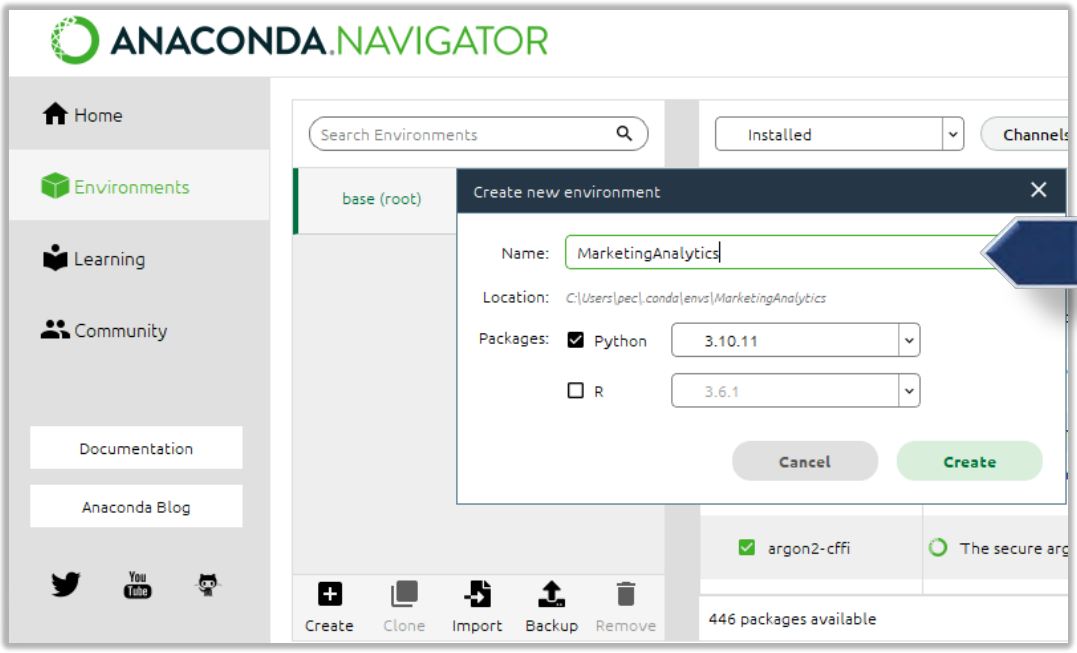
(ที่มา: https://www.shanelynn.ie/plotting-with-python-and-pandas-libraries-for-data-visualisation/)

Environments

หลังจากติดตั้งเสร็จแล้ว ให้เลือก Anaconda Navigator



รูปที่ 1.4 Anaconda Navigator

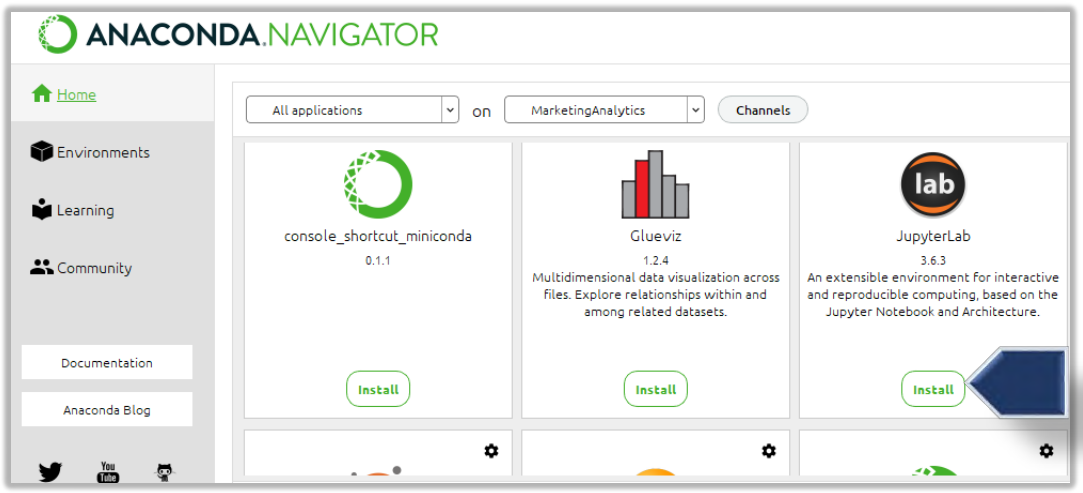


รูปที่ 1.5 Environments

เลือก **Environments** และคลิกปุ่ม **Create** (ด้านล่าง-ซ้าย) จากนั้นพิมพ์ชื่อสภาพแวดล้อมใหม่ เลือก Packages: Python และคลิกปุ่ม **Create** (ด้านขวา)

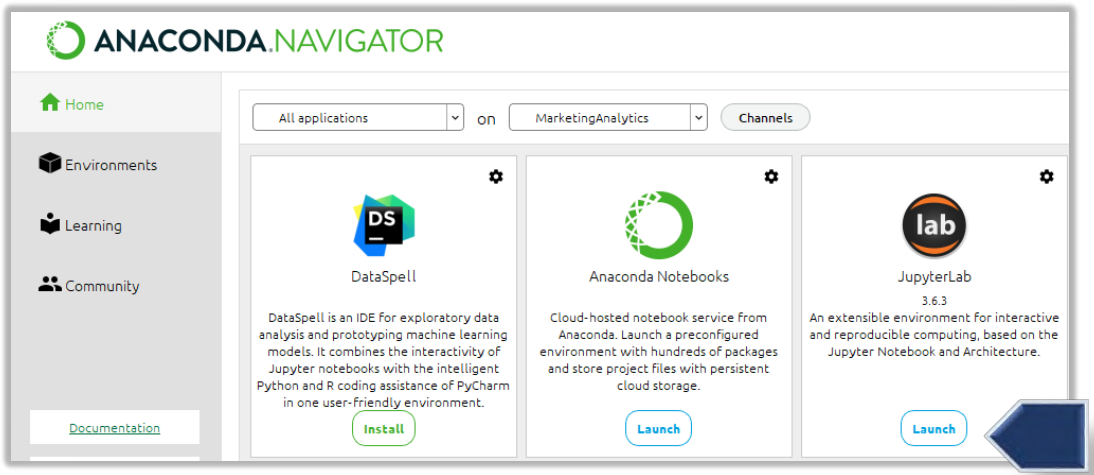
ติดตั้ง JupyterLab

เลือก **Home** (สังเกตว่าเรายังอยู่ในสภาพแวดล้อมใหม่ ที่สร้างจากขั้นตอนก่อนหน้านี้) จากนั้นติดตั้ง JupyterLab คลิกปุ่ม **Install**



รูปที่ 1.6 ติดตั้ง JupyterLab

การเรียกใช้งาน JupyterLab ให้คลิกปุ่ม **Launch**



รูปที่ 1.7 เรียกใช้งาน JupyterLab

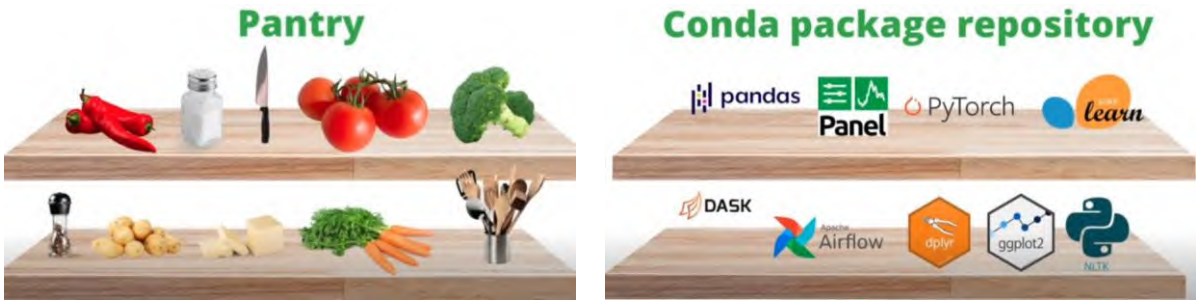
Conda

conda เป็นเครื่องมือสำหรับการจัดการ package และสภาพแวดล้อม สำหรับ Python และภาษาโปรแกรมอื่นๆ โดยเฉพาะอย่างยิ่งสำหรับวิทยาการข้อมูล วิทยาการคำนวณ และการเรียนรู้เชิงลึก โดยที่ผู้ใช้งานไม่จำเป็นต้องมีความรู้ด้านระบบปฏิบัติการหรือการจัดการ dependencies ในโปรแกรมของตนเอง

conda เป็นซอฟต์แวร์ที่มีคุณภาพและได้รับการพัฒนาอย่างต่อเนื่องโดยทีมงานที่มีความเชี่ยวชาญ (โดยทีมงานของ Anaconda ซึ่งเป็นบริษัทเอกชนที่เป็นผู้นำในการพัฒนา Python package) และเป็นซอฟต์แวร์โอเพ่นซอร์ส ซึ่งสามารถเข้าถึง source code และแก้ไขได้ตามต้องการ ทำให้สามารถใช้งานได้อย่างอิสระและเป็นกลาง อีกทั้งยังเป็นเครื่องมือที่มีความยืดหยุ่นและสามารถใช้งานได้กับหลายระบบปฏิบัติการ รวมถึง Linux, macOS, และ Windows

รองรับภาษาโปรแกรมมิ่งหลายตัวใน conda ได้แก่ Python, R, Ruby, Lua, Scala, Java, JavaScript, C/C++, FORTRAN

เพื่อให้ผู้อ่านเข้าใจ conda ได้ง่ายขึ้น เราจะเปรียบเทียบ conda กับ ตู้กับข้าว ผู้อ่านจะประกอบอาหาร ผู้อ่านจะต้องเตรียมวัตถุดิบ เครื่องปรุง อุปกรณ์ต่างๆ จากนั้นผู้อ่านประกอบอาหารขึ้นมา ทั้งนี้อาหารที่ได้ยังขึ้นอยู่กับเครื่องใช้และส่วนผสมต่างๆ เหล่านั้น



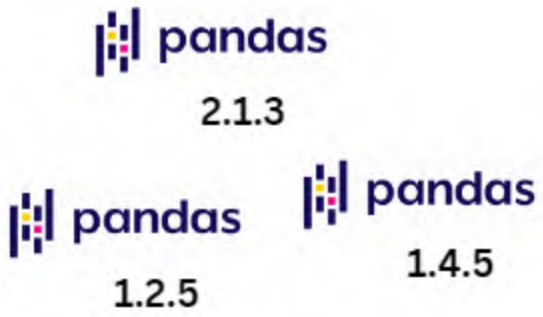
รูปที่ 1.8 เปรียบเทียบ Conda กับตู้กับข้าว

(ที่มา: https://www.anacoda.com)

เช่นเดียวกับ โลกของ conda แต่ละ package อาจขึ้นอยู่กับ package อื่นๆ เช่น เมื่อผู้อ่านติดตั้ง pandas จะมีการติดตั้ง package อื่นๆ ด้วยโดยอัตโนมัติที่ pandas ขึ้นอยู่ เช่น NumPy

บางทีผู้อ่านอาจใช้ชุดอาหาร (meal kit) เพื่อทำมันฝรั่งบด ซึ่งมาพร้อมกับสูตรและเครื่องปรุงที่จำเป็นทั้งหมด ผู้อ่านสามารถทำอาหารซ้ำได้ จากสูตรที่คนอื่นสร้างขึ้น หรืออาจใช้มันฝรั่งประเภทอื่น เช่น มันม่วง ทดแทนได้

คล้ายกับโลกของ conda สภาพแวดล้อม (environments) ทำหน้าที่เหมือนชุดอาหาร (meal kits) ผู้อ่านสามารถสร้างสภาพแวดล้อมที่ทำซ้ำได้ จากแฟ้ม environment.yml ที่ผู้อ่านหรือบุคคลอื่นสร้างขึ้น และ pin versions ของ packages ในสภาพแวดล้อมของผู้อ่าน



รูปที่ 1.9 เวอร์ชัน pandas

แนะนำให้ผู้อ่านสร้างสภาพแวดล้อมของแต่ละโครงการแยกกัน เช่น สภาพแวดล้อมสำหรับ ML/AI project สภาพแวดล้อมสำหรับ ETL และ dashboarding project และสภาพแวดล้อมอื่นๆ สำหรับ R project เป็นต้น ทั้งนี้ conda สามารถจัดการสภาพแวดล้อมได้ง่าย

รูปที่ 1.10 environment ของแต่ละโครงการ

รูปที่ 1.11 environment management

Conda Workflow

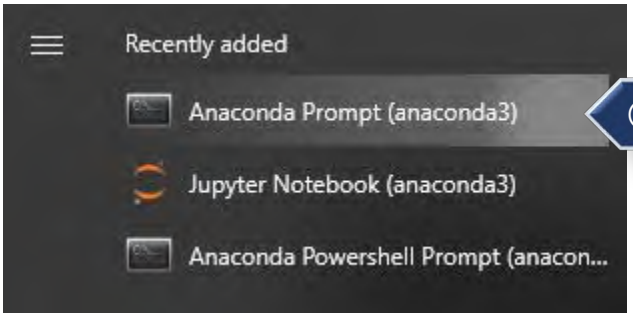
การสร้างสภาพแวดล้อมใหม่ใน conda เป็นการสร้างพื้นที่ที่แยกจากสภาพแวดล้อม (environment) หลักที่มีอยู่เดิม ซึ่งสามารถใช้งาน packages ต่างๆ ได้โดยไม่มีผลต่อสภาพแวดล้อมหลัก นอกจากนี้ยังช่วยให้ผู้ใช้สามารถจัดการการติดตั้ง package และการใช้งานได้อย่างเหมาะสมกับงานที่ต้องการทำ

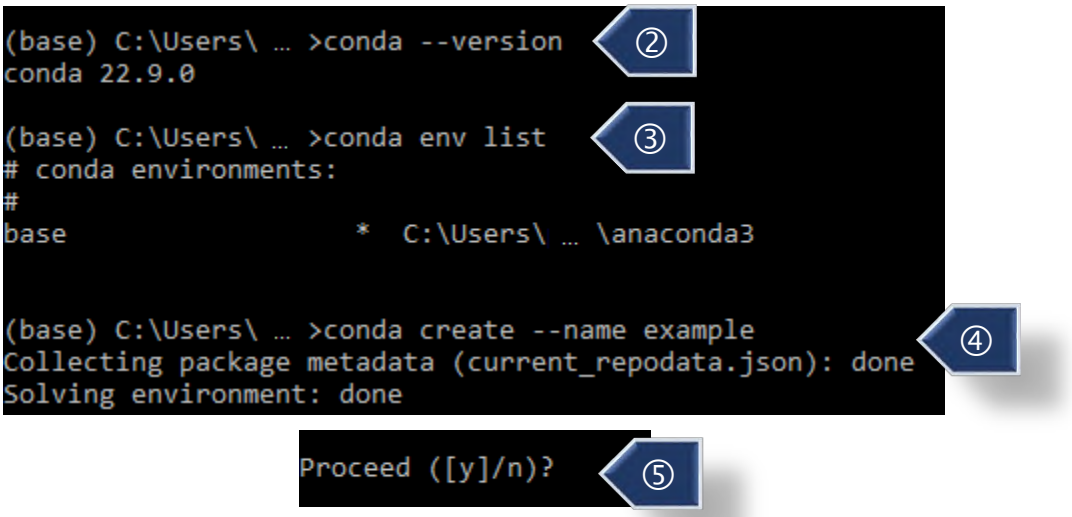
นอกจากนี้ยังช่วยให้ผู้ใช้งานสามารถทดลองและทดสอบ package ใหม่ๆ ได้โดยไม่ต้องกังวลเกี่ยวกับการแก้ไขปัญหาที่เกิดขึ้นกับ packages ที่มีอยู่แล้ว และยังช่วยลดความเสี่ยงที่อาจเกิดขึ้นเมื่อต้องใช้ package ที่มีการอัปเกรดเวอร์ชันใหม่ๆ โดยไม่ทำให้สภาพแวดล้อมหลักเกิดผลกระทบใดๆ

ขั้นตอนการทำงาน มีดังนี้

1. สร้างสภาพแวดล้อม (create)
2. เปิดใช้งานสภาพแวดล้อม (activate)
3. ติดตั้ง package
4. ใช้งาน JupyterLab
5. ปิดการใช้งานสภาพแวดล้อม (deactivate)

1. สร้างสภาพแวดล้อม





① เข้าใช้งาน Anaconda Prompt

② ตรวจเช็คเวอร์ชัน ใช้คำสั่ง conda ---version

③ เรียกดูสภาพแวดล้อมที่มีอยู่ ใช้คำสั่ง conda env list หรือ conda info ---v

<u>หมายเหตุ</u> แนะนำผู้อ่านควรสร้างสภาพแวดล้อมใหม่สำหรับงานของผู้อ่าน ไม่ควรทำงานใน base

④ สร้างสภาพแวดล้อมใหม่ ชื่อ example ใช้คำสั่ง conda create –name example

⑤ ในระหว่างการสร้างสภาพแวดล้อมใหม่ จะมีการถามว่าจะดำเนินการต่อหรือไม่?

ให้ Enter หรือพิมพ์ y

<u>หมายเหตุ</u> ในการสร้างสภาพแวดล้อมใหม่ ผู้อ่านสามารถระบุเวอร์ชันได้ เช่น

```
conda create --name example python=3.9
```

2. เปิดใช้งานสภาพแวดล้อม (activate)

ผู้อ่านจะตั้งชื่อสภาพแวดล้อมใหม่ของผู้อ่าน เป็นชื่ออะไรก็ได้ เช่น MarketingAnalytics, example หรือ ProjectY

```
(base) C:\Users\ … >conda env list
# conda environments:
#
base                  *  C:\Users\ … \anaconda3
example                  C:\Users\ … \anaconda3\envs\example

(base) C:\Users\ … >conda activate example

(example) C:\Users\ … >conda list
# packages in environment at C:\Users\ …\anaconda3\envs\example:
#
# Name                    Version                   Build  Channel
```

⑥ ⑦ ⑧

⑥ เรียกดูสภาพแวดล้อมที่มีอยู่อีกครั้ง จะเห็นว่า มี example เพิ่มเข้ามา

⑦ เปิดใช้งานสภาพแวดล้อมใหม่ ใช้คำสั่ง conda activate example

ให้สังเกต วงเล็บด้านหน้า จะเปลี่ยนจาก (base) เป็น (example)

⑧ เรียกดู package ที่มีอยู่ ใช้คำสั่ง conda list

3. ติดตั้ง package

```
C:\Users\ … >conda install -n example jupyterlab
```

⑨

⑨ ติดตั้ง package ใช้คำสั่ง conda install -n example jupyterlab

```
(example) C:\Users\ … >conda list
# packages in environment at C:\Users\ …\anaconda3\envs\example:
#
# Name                    Version                   Build  Channel
anyio                     3.5.0           py310haa95532_0
argon2-cffi               21.3.0             pyhd3eb1b0_0
argon2-cffi-bindings      21.2.0          py310h2bbff1b_0
asttokens                 2.0.5              pyhd3eb1b0_0
attrs                     22.1.0          py310haa95532_0

...

jupyter_client            7.4.9           py310haa95532_0
jupyter_core              5.2.0           py310haa95532_0
jupyter_server            1.23.4          py310haa95532_0
jupyterlab                3.5.3           py310haa95532_0
jupyterlab_pygments       0.1.2                    py_0
jupyterlab_server         2.19.0          py310haa95532_0
libffi                    3.4.2                hd77b12b_6
libiconv                  1.16                 h2bbff1b_2
```

⑩

⑩ เรียกดู package ที่มีอยู่ อีกครั้ง ใช้คำสั่ง conda list

เลื่อนลงมาจะเห็น jupyterlab พร้อมเวอร์ชัน และ Build name

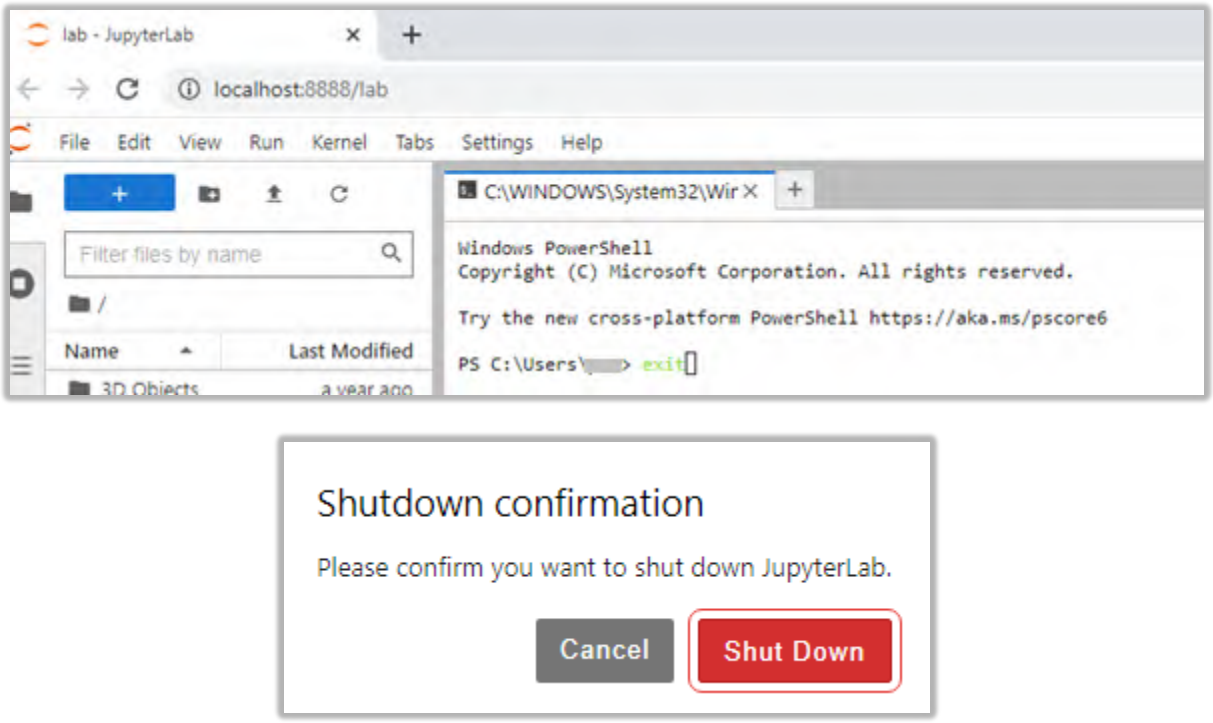
หมายเหตุ เพิ่ม -c ในคำสั่งเพื่อระบุ channel name เช่น conda install -c conda-forge condastats

4. ใช้งาน JupyterLab

```
(example) C:\Users\ … >jupyter-lab
```

พิมพ์คำสั่ง jupyter-lab

เมื่อจะออกจาก jupyterLab ให้เลือก File>Shut Down



5. ปิดการใช้งานสภาพแวดล้อม (deactivate)

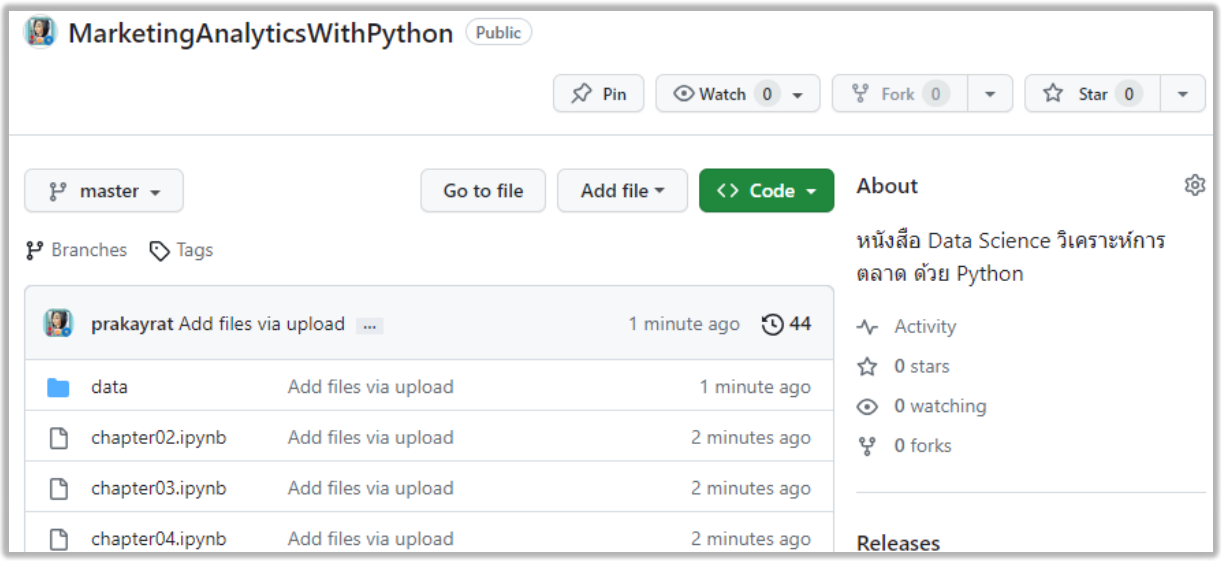
```
(example) C:\Users\ … >conda deactivate

(base) C:\Users\ … >
```

พิมพ์คำสั่ง conda deactivate

สำหรับหนังสือเล่มนี้ จะสร้างสภาพแวดล้อมใหม่ ตั้งชื่อ **MarketingAnalytics**

เตรียมข้อมูล

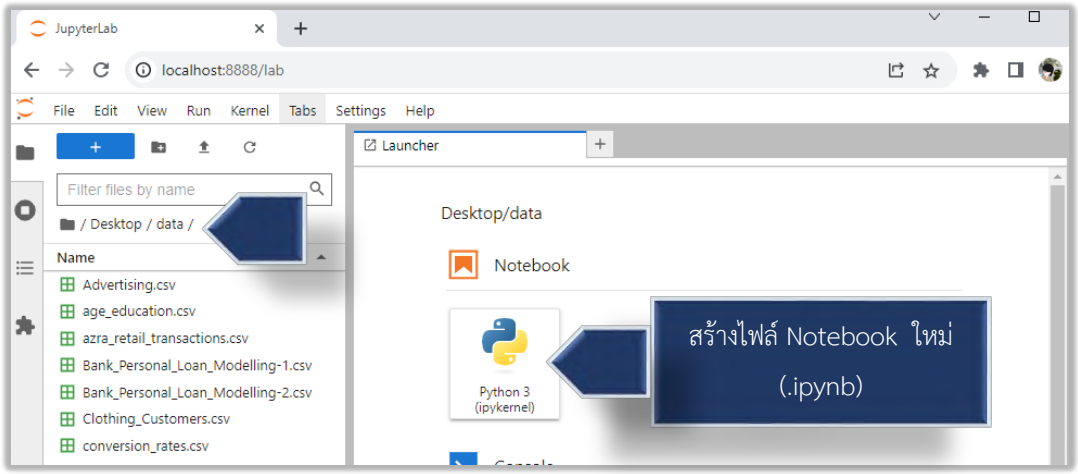
ดาวน์โหลดไฟล์ต่างๆ ได้ที่ https://github.com/prakayrat/MarketingAnalyticsWithPython



รูปที่ 1.12 ดาวน์โหลดไฟล์ข้อมูล

นำไฟล์ต่างๆ ไว้ในเครื่องคอมพิวเตอร์ของผู้อ่าน ในที่นี้ผู้เขียนไว้ที่ Desktop จากนั้นสร้างโฟลเดอร์ data

เรียกใช้งาน JupyterLab ด้านซ้ายของหน้าจอให้ไปที่ Desktop/data/ จากนั้นด้านขวาให้เลือกสร้างไฟล์ใหม่



รูปที่ 1.13 เริ่มใช้งาน JupyterLab

ตอนนี้ ผู้อ่านพร้อมสำหรับเรียนรู้ในหัวข้อถัดไปแล้ว!


DATA SCIENCE
with Python

บทที่ 2

การเตรียมข้อมูลก่อนการวิเคราะห์

สาระสำคัญในบทนี้

- ทักษะที่จำเป็นในการประมวลผลและการทำความสะอาดข้อมูล (clean data) เพื่อให้พร้อมสำหรับการวิเคราะห์ข้อมูลอย่างมีประสิทธิภาพ
- การใช้ไลบรารี pandas ใน Python สำหรับอ่านและนำเข้าข้อมูลจากไฟล์รูปแบบต่างๆ
- เรียนรู้วิธีการ slicing, aggregation และ filtering บน DataFrames
- เรียนรู้ทักษะวิธีการ join DataFrames การจัดการกับค่าสูญหาย และการรวมข้อมูลจากแหล่งข้อมูลต่างๆ

Data Science for Marketing หรือวิทยาการข้อมูลสำหรับการตลาด เป็นสาขาที่ผสมผสานการวิเคราะห์สถิติ แมชชีนเลิร์นนิ่งหรือการเรียนรู้ของเครื่อง และวิทยาการคอมพิวเตอร์ เพื่อสกัดความรู้จากข้อมูลที่มีโครงสร้างและไม่มีโครงสร้าง ช่วยให้ธุรกิจตัดสินใจ และสร้างความได้เปรียบในการตลาด

นักวิทยาการข้อมูล (data scientist) สามารถช่วยการตลาดได้โดยวิเคราะห์ข้อมูลลูกค้า ตรวจจับรูปแบบ สร้างโมเดลทำนายเพื่อคาดการณ์พฤติกรรมของผู้บริโภค และใช้อัลกอริทึมแมชชีนเลิร์นนิ่งเพื่อปรับแต่งแคมเปญการตลาดให้มีผลสูงสุด

นอกจากนี้ นักวิทยาการข้อมูล สามารถทำงานร่วมกับทีมการตลาดในการพัฒนาผลิตภัณฑ์และบริการใหม่ ปรับปรุงประสบการณ์ของลูกค้า และปรับแต่งกลยุทธ์การตั้งราคา

ดังนั้น การใช้ Data Science หรือวิทยาการข้อมูล ในการตลาด จึงช่วยให้ธุรกิจสามารถประสบความสำเร็จในการตลาด และตัดสินใจที่มีข้อมูลเป็นพื้นฐานได้

ทุกวันนี้ คำแนะนำ (recommendations) มีอิทธิพลต่อการตัดสินใจเลือกซื้อสินค้า หรือเลือกใช้บริการของลูกค้า ไม่ว่าจะเป็นสินค้าหรือบริการที่เกี่ยวกับ lifestyle รถยนต์ IT อาหารการกิน สุขภาพ บ้าน เกม ความสวยความงาม คำแนะนำดังกล่าว เป็นไปได้ด้วยเทคนิคต่างๆ รวมถึงเทคนิควิทยาการข้อมูล (data science) ที่ใช้ประโยชน์จากข้อมูล ในการสร้างแบบจำลองที่ซับซ้อน (complex models) การทำงานที่ซับซ้อน และการได้รับข้อมูลเชิงลึก (insights) ของลูกค้าที่มีค่า ด้วยความแม่นยำที่ยอดเยี่ยม

แม้ว่าจะใช้หลักการด้านวิทยาการข้อมูล ในการวิเคราะห์การตลาด จะเป็นกลยุทธ์ที่ได้รับการพิสูจน์แล้วว่าคุ้มค่า และมีประสิทธิภาพ แต่หลายๆ บริษัทยังคงใช้เทคนิคเหล่านี้อย่างไม่เต็มประสิทธิภาพ มีช่องว่าง ระหว่างความเป็นไปได้กับการใช้งานจริงของเทคนิคเหล่านี้

หนังสือเล่มนี้จะช่วยเสริมทักษะในการเชื่อมช่องว่างนั้น โดยครอบคลุมเทคนิคที่เป็นประโยชน์มากมาย ทั้งในแง่ของกลยุทธ์ และการตัดสินใจ (strategies and decision-making) ทางการตลาด และผู้อ่านจะสามารถสร้างและจัดการโซลูชันการวิเคราะห์ทางการตลาด (marketing analytics solution) โดยใช้ Python การแบ่งกลุ่มลูกค้าจากข้อมูลที่หาได้ คาดการณ์มูลค่าตลอดการที่ยังเป็นลูกค้าอยู่ (lifetime value) และจำลองพฤติกรรมการตัดสินใจโดยใช้เทคนิคด้านวิทยาการข้อมูล

ในบทนี้จะพูดถึงการเตรียมข้อมูลก่อนการวิเคราะห์ ซึ่งจะเกี่ยวกับกระบวนการตรวจสอบ สะสาง แก้ไข หรือจัดการกับข้อมูลให้อยู่ในสภาพที่พร้อมใช้งานที่สุด

ข้อมูลอาจมาจากแหล่งภายนอก (external sources) ที่ไม่สามารถนำมาใช้ได้โดยตรง จำเป็นต้องได้รับการวิเคราะห์ จัดโครงสร้าง และกรองข้อมูล (analyzed, structured, and filtered) ก่อนที่จะนำไปใช้ต่อไป

นอกจากนี้ผู้อ่านจะได้เรียนรู้การจัดการ (manipulate) ข้อมูลแถวและคอลัมน์ และแปลงข้อมูล (transformation) เพื่อให้แน่ใจว่าผู้อ่านมีข้อมูลที่ถูกต้อง ด้วยแอตทริบิวต์ที่ถูกต้อง (the right data with the right attributes) นี่เป็นทักษะที่จำเป็นสำหรับนักวิเคราะห์ข้อมูล (data analyst) เพราะเอาต์พุตขึ้นอยู่กับการวิเคราะห์ ถ้าข้อมูลไม่ถูกต้อง ก็ไม่ต่างจากขยะ ดังคำกล่าวที่ว่า **"garbage in, garbage out"** ฉะนั้นก่อนที่ผู้อ่านจะเริ่มทำงานกับข้อมูล ผู้อ่านต้องเข้าใจธรรมชาติของข้อมูลก่อนที่จะใช้งาน

Data models

ในการสร้างโซลูชันการวิเคราะห์ สิ่งแรกที่ต้องทำคือการสร้างโมเดลข้อมูล (data model) โมเดลข้อมูลเป็นภาพรวมของแหล่งข้อมูล (data sources) ที่ผู้อ่านจะใช้ดูความสัมพันธ์กับข้อมูลจากแหล่งต่างๆ ที่จะถูกดึงมาและอยู่ในรูปแบบใด เช่น ไฟล์ Excel ฐานข้อมูล หรือ JSON จากแหล่งอินเทอร์เน็ต

หมายเหตุ โมเดลข้อมูล (data model) จะพัฒนาไปตามแหล่งข้อมูลและกระบวนการ (data sources and processes) ที่เปลี่ยนแปลง

โมเดลข้อมูล สามารถประกอบด้วยข้อมูล 3 ประเภท ได้แก่

Structured data

ข้อมูลที่มีโครงสร้าง เป็นข้อมูลที่ง่ายที่สุดในการจัดการข้อมูล (manage information) ข้อมูลจะถูกจัดในรูปแบบตาราง มีค่าที่ถูกต้องสอดคล้องกับแอตทริบิวต์ที่ถูกต้อง มีคอลัมน์ที่ไม่ซ้ำกัน (unique column) เรียกว่า "index" เข้าถึงได้ง่ายและรวดเร็ว เช่น คอลัมน์ employee_id

เราจึงสามารถใช้งาน SQL queries ในการเข้าถึงแถวและคอลัมน์ได้อย่างง่ายดาย นอกจากนี้ถ้าไม่มีแถวว่าง ไม่มีข้อมูลสูญหาย คอลัมน์ไม่ซ้ำกัน จะทำให้ชุดข้อมูลนี้ทำงานได้ง่ายยิ่งขึ้น

และนี่คือสิ่งที่ข้อมูลแบบมีโครงสร้างมีการใช้งานอย่างแพร่หลายมากที่สุด เพราะง่ายต่อการวิเคราะห์ อีกทั้งมีการจัดเก็บไว้ในรูปแบบตารางมาตรฐาน (standardized tabular format) จึงทำให้การเพิ่ม การลบ และการอัปเดต (adding, deleting, and updating) รายการ ทำได้ง่ายและโปรแกรมได้ และด้วยข้อมูลที่มีโครงสร้าง ผู้อ่านอาจไม่ต้องออกแรงมากในระหว่างขั้นตอนการเตรียมข้อมูลและทำความสะอาดข้อมูล (preparation and cleaning stage)

ข้อมูลที่จัดเก็บไว้ในฐานข้อมูลเชิงสัมพันธ์ เช่น MySQL, Amazon Redshift และอื่นๆ คือตัวอย่างของข้อมูลที่มีโครงสร้าง

	Column1	Q1	Q2	Q3	Q4
2	Digital Advertising	84345	67222	83629	3445
3	Personalised Advertising	23232	3334	84202	1113
4	Video Advertising	23224	1455	64742	2244
5	Mobile Advertising	34234	34432	35454	2224

รูปที่ 2.1 ตัวอย่าง Structure data

Semi-structured data

การเก็บข้อมูลเป็นตารางแบบลำดับชั้น (tabular hierarchy) มีการจัดกลุ่มองค์ประกอบและความสัมพันธ์ระหว่างกัน เช่น เมตาดาต้าของเพลง จะมีข้อมูลหน้าปก ศิลปิน ความยาวของเพลง และอาจมีเนื้อร้อง ผู้อ่านสามารถค้นหาชื่อศิลปินหรือชื่อเพลงตามที่ต้องการได้ ข้อมูลดังกล่าวไม่มีลำดับชั้นแบบตายตัว (fixed hierarchy mapping) ในการแมปคอลัมน์ที่ไม่ซ้ำกันกับแถวในรูปแบบที่คาดไว้ แต่ผู้อ่านสามารถค้นหาข้อมูลที่ต้องการได้

ตัวอย่างเช่น ไฟล์ JSON เป็นไฟล์ที่สามารถเข้าใจได้ง่าย เช่น ข้อมูลของ Focus Su

```
[
    {
        "id": "E101"
        "name": "Focus Su"
        "email": "focusSu@example.com"
        "age": "12"
        "address": "39, Intramara Rd"
        "province": "BKK"
        "Country": "Thailand"
    }
]
```

Unstructured data

ข้อมูลที่ไม่มีโครงสร้างอาจไม่เป็นตาราง หรือถ้าเป็นตารางจำนวนแอตทริบิวต์หรือคอลัมน์เป็นไปตามอำเภอใจ (completely arbitrary) ข้อมูลเดียวกันแต่อาจแสดงในรูปแบบที่แตกต่างกัน หรือแอตทริบิวต์อาจไม่ตรงกัน

ตัวอย่างเช่น การรีวิวผลิตภัณฑ์ต่างๆ ที่จัดเก็บในแถวของ Excel sheet หรือข้อความทวีตในโปรไฟล์ Twitter การค้นหาเราจะค้นหาผ่าน คีย์เวิร์คเฉพาะ (specific keywords) ในข้อมูลนั้นๆ แต่เราไม่สามารถจัดเก็บในฐานข้อมูลเชิงสัมพันธ์ (relational database) ได้ และไม่สามารถสร้างลำดับชั้นที่ชัดเจนระหว่างองค์ประกอบหรือแถวต่างๆ

ข้อมูลที่ไม่มีโครงสร้างสามารถจัดเก็บเป็น text files, images หรือ audio clips

โดยทั่วไป ข้อมูลการตลาด (marketing data) จะประกอบด้วยข้อมูล 3 ประเภทข้างต้น และมาจากหลากหลายแหล่ง มีขนาดและความหมายต่างกัน เช่น ค่าของฟิลด์ (field) อาจมีความยาวต่างกัน บางฟิลด์อาจไม่ตรงกับฟิลด์อื่น ชื่อฟิลด์อาจแตกต่างกัน บางแถวอาจมีค่าสูญหาย เป็นต้น

ซึ่งผู้อ่านจะได้เรียนรู้วิธีจัดการกับปัญหาดังกล่าวกับข้อมูลของผู้อ่านได้อย่างมีประสิทธิภาพ โดยใช้ Python

แผนภาพต่อไปนี้แสดงให้เห็นว่า โมเดลข้อมูลสำหรับการวิเคราะห์ทางการตลาดมีลักษณะอย่างไร โมเดลข้อมูล ประกอบด้วยข้อมูลทุกประเภท

รูปที่ 2.2 ข้อมูลสำหรับการวิเคราะห์ทางการตลาด

เมื่อโมเดลข้อมูลมีความซับซ้อน ความน่าจะเป็นของการมีข้อมูลที่ไม่ดีจะเพิ่มขึ้น เช่น การวิเคราะห์ทางการตลาดที่ใช้ข้อมูลประชากรศาสตร์ของลูกค้า นักวิเคราะห์อาจอ่านอายุของลูกค้าเป็น text แทนที่จะเป็น integer การทำเช่นนี้จะทำให้ไม่สามารถวิเคราะห์อายุลูกค้าได้ นั่นคือไม่สามารถใช้ฟังก์ชันหรือ aggregation functions ได้ เช่น ไม่สามารถหาค่าเฉลี่ยของลูกค้าได้ แต่ทั้งนี้เราสามารถแก้ไขได้โดยด้วยการตรวจสอบคุณภาพของข้อมูล เพื่อให้แน่ใจว่าข้อมูลเป็นประเภทที่เหมาะสมและถูกต้องหรือไม่?

Python เป็นภาษาโปรแกรมทั่วไปอเนกประสงค์ (all-purpose general programming language) ที่รวมเข้ากับเกือบทุกแพลตฟอร์ม และช่วยในการผลิตและวิเคราะห์ข้อมูลเป็นไปโดยอัตโนมัติ

นอกจากที่ Python จะทำความเข้าใจรูปแบบ (patterns) และให้โครงสร้างข้อมูลพื้นฐานแล้ว Python ยังสามารถทำให้โมเดลข้อมูลยอมรับค่าที่ถูกต้องสำหรับแอตทริบิวต์ วิธีการวิเคราะห์การตลาดส่วนใหญ่ในปัจจุบันจะจัดโครงสร้างข้อมูลประเภทต่างๆ โดยส่งผ่านสคริปต์ เช่น

รูปที่ 2.3 Python สำหรับการวิเคราะห์ทางการตลาด

อย่างไรก็ตาม ข้อมูลยังไม่ได้อยู่ในรูปแบบที่ดีที่สุดเท่าที่จะเป็นไปได้ในการวิเคราะห์ ถ้าผู้อ่านสามารถจัดโครงสร้างข้อมูลได้อย่างสมบูรณ์ (นั่นคือ จัดเรียงข้อมูลในรูปแบบตาราง ด้วยค่าที่ถูกต้องชี้ไปยังแอตทริบิวต์ที่ถูกต้องโดยไม่มีการซ้อนกัน (no nesting)) จะดูได้ง่ายว่าจุดข้อมูลแต่ละจุดเปรียบเทียบกับจุดอื่นๆ ผู้อ่านสามารถเข้าใจข้อมูลได้อย่างง่ายดาย กล่าวคือ ดูว่าค่าส่วนใหญ่อยู่ในช่วงใด ระบุค่าผิดปกติที่ชัดเจน และอื่นๆ เพียงแค่เลื่อนดูข้อมูล

แม้ว่าจะมีเครื่องมือ (tools) มากมายที่สามารถใช้ในการแปลงข้อมูล (convert data) จากรูปแบบที่ไม่มีโครงสร้าง/กึ่งโครงสร้าง (unstructured/semi-structured format) เป็นรูปแบบที่มีโครงสร้าง (structured format) เช่น Spark, STATA และ SAS

แต่ทั้งนี้ยังมีเครื่องมืออีกชิ้นหนึ่ง ที่ใช้กันอย่างแพร่หลายสำหรับวิทยาการข้อมูล และสามารถรวมเข้ากับเฟรมเวิร์ค (framework) ใดก็ได้ มีฟังก์ชันการทำงาน (functionalities) มากมาย ต้นทุนต่ำ และใช้งานง่าย ในที่นี้ก็คือ pandas

Pandas

pandas software library ที่เขียนด้วย Python และเป็น basic building block สำหรับการจัดการและวิเคราะห์ข้อมูล (data manipulation and analysis) นำเสนอคอลเลกชันของโครงสร้างข้อมูลและเครื่องมือการวิเคราะห์ที่มีประสิทธิภาพสูง และใช้งานง่าย (easy-to-use) ซึ่งเป็นประโยชน์อย่างยิ่งสำหรับนักวิเคราะห์การตลาดและนักวิทยาการข้อมูล (marketing analysts and data scientists)

<u>หมายเหตุ</u> ผู้อ่านสามารถใช้คำสั่ง `pip install pandas` โดยเรียกใช้คำสั่งนี้ใน terminal app หรือ command prompt เพื่อติดตั้งได้ (สำหรับ macOS หรือ Linux ผู้อ่านใช้คำสั่ง `pip3` แทน `pip`)

แผนภาพต่อไปนี้แสดงให้เห็นว่าข้อมูลประเภทต่างๆ ถูกแปลงเป็นรูปแบบที่มีโครงสร้างได้อย่างไร โดยใช้ pandas

รูปที่ 2.4 Pandas สำหรับการวิเคราะห์ทางการตลาด

การทำงานกับ pandas ผู้อ่านจะต้องจัดการกับอ็อบเจกต์สองประเภทหลัก (primary object types) ได้แก่ DataFrames และ Series

1) **DataFrame** โครงสร้างตารางพื้นฐานที่เก็บข้อมูลในแถวและคอลัมน์ (คล้ายกับ spreadsheet) การวิเคราะห์ข้อมูล ผู้อ่านสามารถใช้ฟังก์ชัน ดำเนินการกับ DataFrames ได้โดยตรง

2) **Series** เป็นคอลัมน์เดียว (a single column) ของ DataFrame ถ้ารวม series ตั้งแต่ 2 series ขึ้นไป จะเป็น DataFrame การเข้าถึงข้อมูล ทำได้โดยผ่านดัชนี (index)

Series

	school
0	2
1	4
2	3
3	1

+

Series

	office
0	4
1	2
2	1
3	0

=

DataFrame

	school	office
0	2	4
1	4	2
2	3	1
3	1	0

รูปที่ 2.5 Series และ DataFrame

ข้อมูลแต่ละคอลัมน์ (school, office) มีโครงสร้างเป็น series และข้อมูลทั้งตาราง (รวมดัชนี ในคอลัมน์แรก) มีโครงสร้างเป็น DataFrames

ตัวอย่างคำสั่ง Python ในการสร้าง DataFrame จาก สอง Series ข้างต้น เช่น

python

```
import pandas as pd
df = pd.DataFrame({'school':pd.Series([2,4,3,1]),
                   'office':pd.Series([4,2,1,0])})
df
```

	school	office
0	2	4
1	4	2
2	3	1
3	1	0

รูปที่ 2.6 คำสั่งสร้าง DataFrame

ผู้อ่านสามารถดู source code ตัวอย่าง ที่อยู่บนเว็บไซต์ดาวน์โหลดของหนังสือ เทียบกับ หมายเลขรูปในหนังสือได้ (https://github.com/prakayrat/MarketingAnalyticsWithPython)

Importing และ exporting data

ทุกทีมในกลุ่มการตลาด สามารถมีแหล่งข้อมูลในการใช้งานของตนได้ ทีมที่จัดการกับข้อมูลลูกค้าจำนวนมาก เช่น ข้อมูลพื้นฐานของลูกค้าหรือข้อมูลการซื้อสินค้า เราอาจต้องใช้ฐานข้อมูล เช่น MySQL หรือ Oracle

ในขณะที่บางทีมอาจจะจัดการกับข้อมูลที่เป็น ข้อความจำนวนมาก เช่น JSON, CSV หรือ XML เนื่องจากมีการใช้แหล่งข้อมูลหลายแห่ง เราจึงมีไฟล์ที่หลากหลาย ในกรณีนี้ อาจต้องใช้ pandas library มาช่วย เนื่องจากมี API (Application Program Interfaces) ที่สามารถใช้อ่านข้อมูลประเภทต่างๆ ลงใน pandas DataFrame

API ที่ใช้บ่อยที่สุด เช่น

แหล่งข้อมูล	Read function	Write function	Format type
CSV	read_csv	to_csv	text
JSON	read_json	to_json	text
HTML	read_html	to_html	text
XML	read_xml	to_xml	text
Excel	read_excel	to_excel	binary
Stata	read_stata	to_stata	binary
SAS	read_sas	to_sas	binary
Clipboard	read_clipboard	to_clipboard	text
Python Pickle File Format	read_pickle	to_pickle	binary

รูปแบบที่ใช้ทั่วไปในการอ่านข้อมูลจากไฟล์ CSV (read a CSV file)

```
import pandas as pd
```

แล้วจึง import ไฟล์ข้อมูล เช่น

```
df = pd.read_csv("sales.csv")
```

หรือระบุเส้นทางที่เก็บไฟล์ เช่น

```
df = pd.read_csv(r'C:\Users\xxx\Documents\sales.csv')
```

หมายเหตุ การเพิ่ม r ก่อนระบุเส้นทางที่เก็บไฟล์ เพื่อดูแลอักขระพิเศษ (special characters) ใดๆ ในเส้นทาง

พารามิเตอร์เพิ่มเติม (additional parameters) ที่ผู้อ่านสามารถส่งไปยังฟังก์ชัน read (ทั้งนี้ ค่าเริ่มต้นของ index ใน DataFrame เริ่มต้นด้วย 0)

คำสั่ง	
skiprows = k	ข้าม k แถวแรก
nrows = k	วิเคราะห์เฉพาะ k แถวแรก
names = [col1, col2 ..]	รายชื่อคอลัมน์ใน DataFrames ที่จะใช้ในการวิเคราะห์
header = k	ชื่อหัวคอลัมน์อยู่แถวที่ k โดยที่ k สามารถเป็น None ได้ (คือไม่มีชื่อคอลัมน์)
index_col = col	กำหนด col เป็นดัชนีของ DataFrame ที่กำลังใช้งาน index_col สามารถเป็นรายชื่อคอลัมน์ได้ (หรือ MultiIndex) หรือเป็น None MultiIndex DataFrame ใช้มากกว่าหนึ่งคอลัมน์ของ DataFrame เป็น index
usecols = [I1, I2 ..]	กำหนดคอลัมน์ที่ต้องการให้ read เช่น [0, 1, 2] หรือระบุชื่อคอลัมน์ ['name', ' views']

ตัวอย่าง สมมติว่า ผู้อ่านต้องการ import ไฟล์ CSV ไปยัง DataFrame ชื่อ df และกำหนดเงื่อนไขดังนี้

- แถวแรกของไฟล์เป็น header
- import เพียง 100 แถวแรก
- import เฉพาะ 3 คอลัมน์แรก

โค้ดที่สอดคล้องกับเงื่อนไข จะเป็นดังนี้

```
df = pd.read_csv("sales.csv", header=1, nrows=100, usecols=[0, 1, 2])
หรือ
df= pd.read_csv('sales.csv', header=1, nrows=100, usecols=[ 'foo', 'bar', 'baz']
```

หมายเหตุ สามารถดูรายละเอียดเกี่ยวกับเอกสาร pandas ได้จาก: https://pandas.pydata.org/pandas-docs/stable/

หลังจาก import ข้อมูลแล้ว ขั้นตอนถัดไปจะเป็น การตรวจสอบว่านำเข้าอย่างถูกต้องหรือไม่ เราจะมาทำความเข้าใจวิธีการทำในส่วนต่อไปนี้

Viewing และ Inspecting data

การตรวจสอบ (inspect) ข้อมูลหลังจาก read เข้ามาใน Python แล้วว่าแอตทริบิวต์ถูกต้องหรือไม่? และค่าถูกต้องหรือไม่? ผู้อ่านสามารถใช้ built-in pandas functions ได้หลายตัว

ที่นิยมใช้กันมากที่สุด คือ ใช้คำสั่ง head() คำสั่งนี้จะแสดงห้าแถวแรกของ DataFrame

```
df.head()
```

ในทำนองเดียวกัน ผู้อ่านสามารถใช้คำสั่ง tail() จะเป็นการแสดงห้าแถวสุดท้ายของ DataFrame

```
df.tail()
```

นอกจากนี้ผู้อ่านยังสามารถระบุจำนวนแถวได้ เช่น head(11) จะเป็นการแสดง 11 แถวแรก ยังมีคำสั่งอื่นๆ ในการตรวจสอบข้อมูล เช่น

คำสั่ง	
df.head(n)	ส่งคืน n แถวแรกของ DataFrame
df.tail(n)	ส่งคืน n แถวสุดท้ายของ DataFrame
df.shape	ส่งคืนมิติของ DataFrame นั่นคือแสดงจำนวนแถวและจำนวนคอลัมน์
df.dtypes	ส่งคืนประเภทข้อมูลของแต่ละคอลัมน์ใน pandas DataFrame เช่น float, object, int64 เป็นต้น
df.info()	สรุป DataFrame และให้แสดง size, type of values และจำนวนนับ (count) ของ non-null values

ตัวอย่าง 2.1 โหลดข้อมูลที่จัดเก็บไว้ในไฟล์ JSON

ทีมเทคโนโลยีในบริษัทของผู้อ่าน ได้ทำการทดสอบแอปชอปปิ้ง จากผู้ใช้บางรายที่อาสาทดสอบเว็บไซต์และถูกขอให้ส่งรายละเอียดผ่านแบบฟอร์มออนไลน์ โดยแบบฟอร์มนี้จะบันทึกรายละเอียดบางอย่างที่เป็นประโยชน์ (เช่น อายุ รายได้ และอื่นๆ) พร้อมกับรายละเอียดบางอย่างที่ไม่เป็นประโยชน์ (เช่น สีตา)

จากนั้นทีมเทคโนโลยีได้ทดสอบโมดูลหน้าโปรไฟล์ (profile page module) ใหม่ โดยบันทึกรายละเอียดเพิ่มเติมบางส่วน ข้อมูลทั้งหมดนี้จัดเก็บไว้ในไฟล์ JSON ชื่อ **user_info.json** ซึ่งทีมเทคโนโลยีส่งให้ผู้อ่านตรวจสอบ

เป้าหมาย คือนำเข้าไฟล์ JSON นี้ และตอบคำถามต่อไปนี้

1) โหลดข้อมูลถูกต้องหรือไม่?
2) มีค่าใดหายไปจากคอลัมน์หรือไม่?
3) ชนิดข้อมูลของแต่ละคอลัมน์คืออะไร?
4) ข้อมูลมีกี่แถวและกี่คอลัมน์?

คำสั่ง ใน python

```python
import pandas as pd

df = pd.read_json('data/user_info.json')
df.head(1)
df.info()
df.shape
```

รูปที่ 2.7 คำสั่งตัวอย่างที่ 2.1

อธิบาย

บรรทัดที่ 1, 3	เรียกใช้ไลบารี่ pandas จากนั้น read_json()
บรรทัดที่ 4	เราสามารถตรวจเช็คว่าโหลดข้อมูลถูกต้องหรือไม่ จากการใช้คำสั่ง head() โดยการตรวจสอบ เช่น พิจารณาข้อมูลในแถวและคอลัมน์ ชื่อคอลัมน์กับข้อมูลสอดคล้องกันหรือไม่ อย่างที่เห็นจากภาพหน้าจอ เรา**ตอบคำถามข้อที่ 1)** ว่า **ถูกต้อง**

```
df.head(1)
```

	_id	index	guid	isActive	balance	pic
0	5c0a28d7a647437fd3d3a6aa	0	832f1af4-fb18-4ba8-b032-f4cbf343dbe9	False	$3,806.93	http://placehold.it/3

1 rows × 22 columns

รูปที่ 2.8 คำสั่ง head()

บรรทัดที่ 5	สำหรับการตอบคำถามข้อ 2) – 4) เราใช้คำสั่ง info() จากภาพหน้าจอ (ในหน้าถัดไป) จะเห็นว่าทุกคอลัมน์ non-null ทั้งหมด ดังนั้น **ตอบคำถามข้อที่ 2)** ว่า **ไม่มีค่าสูญหาย** ส่วนชนิดข้อมูล เรา**ตอบคำถามข้อที่ 3)** ว่า

- คอลัมน์ isActive เป็น Boolean
- คอลัมน์ index, age เป็น integers
- คอลัมน์ latitude, longitude เป็น floats
- ที่เหลือ เป็น objects

และ**ตอบคำถามข้อที่ 4)** ว่า **ข้อมูลมี 6 แถวและ 22 คอลัมน์**

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   _id            6 non-null      object
 1   index          6 non-null      int64
 2   guid           6 non-null      object
 3   isActive       6 non-null      bool
 4   balance        6 non-null      object
 5   picture        6 non-null      object
 6   age            6 non-null      int64
 7   eyeColor       6 non-null      object
 8   name           6 non-null      object
 9   gender         6 non-null      object
 10  company        6 non-null      object
 11  email          6 non-null      object
 12  phone          6 non-null      object
 13  address        6 non-null      object
 14  about          6 non-null      object
 15  registered     6 non-null      object
 16  latitude       6 non-null      float64
 17  longitude      6 non-null      float64
 18  tags           6 non-null      object
 19  friends        6 non-null      object
 20  greeting       6 non-null      object
 21  favoriteFruit  6 non-null      object
dtypes: bool(1), float64(2), int64(2), object(17)
memory usage: 1.1+ KB
```

รูปที่ 2.9 คำสั่ง info()

บรรทัดที่ 6	คำสั่ง shape ใช้ตรวจสอบจำนวนแถวและคอลัมน์ได้อีกด้วย

```
df.shape
```

```
(6, 22)
```

รูปที่ 2.10 คำสั่ง shape

ตัวอย่าง 2.2 โหลดข้อมูลจากหลายแหล่งข้อมูล

ถ้าบริษัทใช้ Facebook ในการทำแคมเปญการตลาด และทำการบันทึกข้อมูล view และ links จำนวน 100 รายการ ลงในไฟล์ **data.csv** และข้อมูลการซื้อขายของลูกค้าในร้านค้าในช่วงไม่กี่ปีที่ผ่านมา ลงในไฟล์ **sales.csv**

เป้าหมาย เพื่อ read ไฟล์เข้ามาใน pandas DataFrames และตรวจสอบดังนี้

1) ชุดข้อมูลใด มีค่าว่าง (null) หรือค่าสูญหาย (missing values) หรือไม่
2) ข้อมูลถูกจัดเก็บในคอลัมน์ที่เหมาะสมหรือไม่? ชื่อคอลัมน์เหมาะสมหรือไม่? รวมทั้งประเภทข้อมูลถูกต้องหรือไม่?

คำสั่ง ใน python

python

```python
import pandas as pd
data = pd.read_csv("data/data.csv")
data.head()
data.tail()
data.info()
sales = pd.read_csv("data/sales.csv")
sales.head(2)
sales.info()
```

รูปที่ 2.11 คำสั่งตัวอย่างที่ 2.2

อธิบาย

บรรทัดที่ 1-2	เรียกใช้ไลบารี่ pandas จากนั้น read_csv() ไฟล์แรกคือ data.csv สร้าง DataFrame ชื่อ data
บรรทัดที่ 3	ทำการตรวจสอบข้อมูลโดยใช้คำสั่ง head() ดังภาพด้านซ้าย พบว่าข้อมูลบรรทัดแรกเป็นชื่อคอลัมน์ ดังนั้นเราต้องปรับคำสั่ง read_csv() โดยเพิ่ม header=1 ดังภาพด้านขวา-ล่าง

```
data = pd.read_csv("data/data.csv")
data.head()
```

Campaign Data	
views	likes
90006	402
101141	389
97297	403
117182	397

```
data = pd.read_csv("data/data.csv", header=1)
data.head()
```

	views	likes
0	90006	402
1	101141	389
2	97297	403
3	117182	397
4	99637	404

รูปที่ 2.12 เพิ่ม header

บรรทัดที่ 4	คำสั่ง `tail()` เป็นการตรวจสอบข้อมูล 5 แถวสุดท้าย

```
data.tail()
```

	views	likes
95	100464	391
96	96382	387
97	94971	389
98	103070	410
99	110387	411

รูปที่ 2.13 คำสั่ง tail()

บรรทัดที่ 5	คำสั่ง info() เป็นการตรวจสอบข้อมูลสูญหาย จากภาพหน้าจอบอกว่า DataFrame มี 2 คอลัมน์ 100 แถว และไม่มีค่าสูญหาย

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   views   100 non-null    int64
 1   likes   100 non-null    int64
dtypes: int64(2)
memory usage: 1.7 KB
```

รูปที่ 2.14 คำสั่ง info()

บรรทัดที่ 6-7	ถัดมา read_csv() ไฟล์ที่สองคือ sales.csv สร้าง DataFrame ชื่อ sales จากนั้นตรวจสอบข้อมูลโดยใช้คำสั่ง head() พบว่าคอลัมน์ Year มีข้อมูลเป็นตัวเลขปีที่ถูกต้อง แต่คอลัมน์ line ข้อมูลดูไม่สมเหตุสมผล แต่ถ้านำสองคอลัมน์ Product, line มาต่อกัน จะดูเข้าใจมากขึ้น ดังเช่นข้อมูลที่แสดงในสองแถวแรก คือ Camping Equipment เช่นเดียวกับ คอลัมน์อื่นๆ เช่น Product.1, type, Order, method มีปัญหาเหมือนกัน

```
sales = pd.read_csv("data/sales.csv")
sales.head(2)
```

	Year	Product	line	Product.1	type	Product.2	Order	method	type.1	Retailer
0	2004	Camping	Equipment	Cooking	Gear	TrailChef	Water	Bag	Telephone	United
1	2004	Camping	Equipment	Cooking	Gear	TrailChef	Water	Bag	Telephone	Canada

รูปที่ 2.15 ข้อมูลที่ดูไม่สมเหตุสมผล

บรรทัดที่ 8	คำสั่ง info() ตรวจสอบค่า null values และชนิดข้อมูลของแต่ละคอลัมน์ จากภาพหน้าจอ คอลัมน์ country มีค่าสูญหาย

```
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Year       100 non-null    int64
 1   Product    100 non-null    object
 2   line       100 non-null    object
 3   Product.1  100 non-null    object
 4   type       100 non-null    object
 5   Product.2  100 non-null    object
 6   Order      100 non-null    object
 7   method     100 non-null    object
 8   type.1     100 non-null    object
 9   Retailer   100 non-null    object
 10  country    9 non-null      object
 11  Revenue    100 non-null    float64
dtypes: float64(1), int64(1), object(10)
memory usage: 9.5+ KB
```

รูปที่ 2.16 country มีค่าสูญหาย

โดยสรุป ผู้อ่านจะพบว่า ข้อมูลแคมเปญการตลาด (data.csv) นั้นดูดี ไม่มีค่าสูญหาย ในขณะที่ข้อมูลการขาย (sales.csv) มีค่าสูญหายเล็กน้อย และชื่อคอลัมน์ไม่ถูกต้อง

สิ่งที่ผู้อ่านเรียนรู้ถึงตอนนี้ ผู้อ่านยังไม่สามารถกำหนดมาตรฐานของข้อมูลได้ ผู้อ่านจะต้องรู้จักโครงสร้างภายใน pandas objects และการจัดเก็บใน pandas ก่อน

DataFrame และ Series

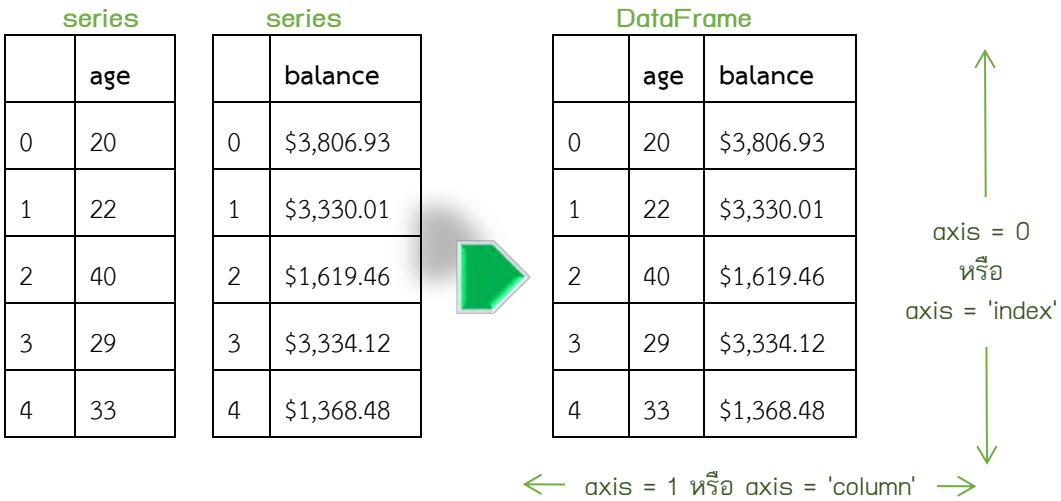
การจัดเก็บข้อมูลในรูป pandas series เป็นการจัดทำ indexed NumPy array เริ่มจากดัชนีที่ 0 การเข้าถึงค่าต่างๆ จะต้องใช้ดัชนี

ดัชนี จะเริ่มต้นที่ค่า 0 และค่าจะเพิ่มขึ้นไปเรื่อยๆ ตามความยาวของ series เช่น

Index	0	1	2	3	4	5	...
array	20	22	40	29	33	23	...

รูปที่ 2.17 ตัวอย่าง pandas series

pandas DataFrame เป็น dictionary ที่มีชื่อคอลัมน์ ประกอบด้วย คีย์และค่าต่างๆ ของ pandas series เข้าไว้ด้วยกัน เช่น



	age
0	20
1	22
2	40
3	29
4	33

	balance
0	$3,806.93
1	$3,330.01
2	$1,619.46
3	$3,334.12
4	$1,368.48

	age	balance
0	20	$3,806.93
1	22	$3,330.01
2	40	$1,619.46
3	29	$3,334.12
4	33	$1,368.48

รูปที่ 2.18 ตัวอย่าง pandas DataFrame

pandas อนุญาตให้ใช้การดำเนินการทั้งแถวและคอลัมน์ของ DataFrame โดยระบุ axis = 0 อ้างถึงแถว และ axis = 1 อ้างถึงคอลัมน์ เช่น จากตัวอย่าง 2.1 ถ้าผู้อ่านต้องการหาผลรวมทุกแถวในคอลัมน์ age ผู้อ่านสามารถใช้คำสั่ง

```python
import pandas as pd
df = pd.read_json("data/user_info.json")
df['age'].sum(axis=0)
```

```
167
```

รูปที่ 2.19 ผลรวมของ age

Data manipulation

เราสามารถจัดการข้อมูล (data manipulation) กับ DataFrames หรือที่เรียกว่า **wrangling tasks** นั่นคือ การสร้าง (creating) การเลือก (selecting หรือ slicing) เป็นส่วนๆ การกรอง (filtering) การรวม (joining) และอื่นๆ มีรายละเอียดดังนี้

1) Selecting และ filtering

2) Creating DataFrames

3) Adding และ removing แอตทริบิวต์และการสังเกต

4) รวม DataFrames

5) ค่าสูญหาย

1) Selecting และ filtering

ตัวอย่างการเลือกข้อมูลบางส่วนใน DataFrame โดยใช้ pandas เช่น จากตัวอย่าง 2.1 ให้ค้นหา index ที่ 0, 3 พร้อมแสดงคอลัมน์ age, name, eyeColor เราจะใช้คำสั่ง loc ดังนี้

```
df.loc[[0,3], ['age', 'name', 'eyeColor']]
```

	age	name	eyeColor
0	20	Grace Berry	green
3	29	Caldwell Patterson	blue

รูปที่ 2.20 แสดง index ที่ 0, 3

นอกจากนี้ การเข้าถึงข้อมูลมากกว่าหนึ่งเซลล์ หรือชุดย่อยของแถวและบางคอลัมน์ของ DataFrame หรือเปลี่ยนลำดับการแสดงบางคอลัมน์ใน DataFrame ผู้อ่านสามารถใช้ไวยากรณ์ที่แสดงในตารางต่อไปนี้

Operation	Syntax	Result
Select a column	df[col]	series
Select multiple columns	df[[col1, col2 …]]	DataFrame
Select a row by label	df.loc[label]	series
Select a row by integer location	df.iloc[loc]	series
Slice rows	df[[start_idx: end_idx]]	DataFrame
Select multiple rows by Boolean vector	df[bool_vec]	DataFrame

2) Creating DataFrames

ในการสร้าง DataFrame สามารถทำได้สองวิธี

- สร้าง DataFrame ใหม่
- Duplicating หรือ slicing จาก DataFrame ที่มีอยู่

2.1) สร้าง DataFrame ใหม่

การสร้าง new DataFrame เพียงใช้ฟังก์ชัน DataFrame ฟังก์ชันนี้จะแปลง Python object ไปเป็น pandas DataFrame โดยทั่วไปฟังก์ชันนี้จะรวบรวมข้อมูลประเภท dict หรือ list เช่น

```python
import pandas as pd
#list
df1 = pd.DataFrame({'Currency': pd.Series(['USD','EUR','JPY']),
                    'Value': pd.Series([35.19, 38.32, 0.24])})
df1
```

	Currency	Value
0	USD	35.19
1	EUR	38.32
2	JPY	0.24

รูปที่ 2.21 ตัวอย่างการสร้าง DataFrame จากข้อมูล list

```python
#dict
df2 = pd.DataFrame.from_dict({'Currency': ['USD','EUR','JPY'],
                              'Value': [35.19, 38.32, 0.24]})
df2
```

	Currency	Value
0	USD	35.19
1	EUR	38.32
2	JPY	0.24

รูปที่ 2.22 ตัวอย่างการสร้าง DataFrame จากข้อมูล dict

2.2) Duplicating หรือ slicing จาก DataFrame ที่มีอยู่

การสร้าง DataFrame ขึ้นมาใหม่ โดยคัดลอกข้อมูลจาก DataFrame ที่มีอยู่ ทำได้เหมือนกับการคัดลอกวัตถุ เช่น obj1 = obj2 หรืออีกวิธีเรียกใช้ฟังก์ชัน deepcopy ที่อนุญาตให้ผู้ใช้วนซ้ำผ่านวัตถุที่ถูกอ้างอิง แล้วนำไปสร้างวัตถุใหม่ เช่น

```python
# Duplicating หรือ slicing
df1_1 = df1.copy
df1_1
```

```
<bound method NDFrame.copy of   Currency  Value
0       USD  35.19
1       EUR  38.32
2       JPY   0.24>
```

หรือ

```
df1_1 = df1.copy(deep = True)
df1_1
```

	Currency	Value
0	USD	35.19
1	EUR	38.32
2	JPY	0.24

หรือ

```
df1_2 = df1
df1_2
```

	Currency	Value
0	USD	35.19
1	EUR	38.32
2	JPY	0.24

รูปที่ 2.23 สร้าง DataFrame จากการคัดลอก DataFrame เดิม

3) Adding และ removing แอตทริบิวต์และการสังเกต

การเพิ่ม และการลบ แถว (หรือการสังเกต) และคอลัมน์ (หรือแอตทริบิวต์) มีคำสั่งที่สามารถเลือกใช้ได้ดังนี้

คำสั่ง	
df['col'] = s	เพิ่มคอลัมน์ใหม่ (ชื่อ col) เข้าไปใน dataFrame, s เป็นข้อมูลแบบ series
df.assign(c1 = s1, c2 = s2 ..)	เพิ่มคอลัมน์ c1, c2 ... ด้วย series s1, s2 ... ใน df
df.append(df2)	เพิ่มค่าจาก df2 ไปต่อท้าย df ทุกที่ที่คอลัมน์ df2 ตรงกับ df
df.drop(labels, axis)	ลบ แถว หรือ คอลัมน์ ตามป้ายกำกับ (labels) และสอดคล้องกับ axis หรือระบุ ดัชนี หรือ ชื่อคอลัมน์โดยตรงก็ได้
df.dropna(axis, how)	ลบแถวที่มีข้อมูลสูญหาย (แต่ถ้าระบุ axis = 1 จะหมายถึงลบคอลัมน์ที่มีข้อมูลสูญหาย)

คำสั่ง	
df.drop_duplicates(keep)	ลบแถวที่มีค่าซ้ำกันใน DataFrame และเก็บแถวแรกไว้ที่ซ้ำไว้ (keep = 'first') หรือ เก็บแถวสุดท้ายที่ซ้ำไว้ (keep = 'last') หรือ no occurrence (keep = False)
df.concat([df1, df2 ..])	สร้าง df1, df2 .. รวมกันตามลำดับ โดยรวมข้อมูลที่มีชื่อคอลัมน์เหมือนกัน โดยอัตโนมัติ

ตัวอย่างเช่น DataFrame ในตัวอย่างก่อนหน้านี้ เราจะลบคอลัมน์ 'Currency' ออก คำสั่งที่ใช้คือ

python

```
df1_2 = df1_2.drop(['Currency'], axis=1)
df1_2
```

	Value
0	35.19
1	38.32
2	0.24

รูปที่ 2.24 ลบคอลัมน์ออกจาก DataFrame

4) รวม DataFrames

ถ้าทีมผลิตภัณฑ์ส่งรายละเอียดเกี่ยวกับราคาผลิตภัณฑ์ของบริษัทที่ได้รับความนิยม ข้อมูลถูกจัดเก็บไว้ในรูป DataFrame ชื่อว่า **df_products** มีรายละเอียดคือ

	Year	Price	Version
0	2017	199	v1
1	2018	199	v1
2	2019	199	v1
3	2020	299	v2
4	2021	299	v2
5	2022	349	v2
6	2023	349	v3

ทีมการเงินได้ส่งรายละเอียดเกี่ยวกับรายได้ของผลิตภัณฑ์เหล่านี้ให้กับผู้อ่าน ข้อมูลถูกจัดเก็บไว้ในรูป DataFrame ชื่อว่า **df_revenue** มีรายละเอียดคือ

	Year	Revenue
0	2018	9473
1	2019	8422
2	2020	9987
3	2021	7994
4	2022	9530
5	2023	9444

ทั้งสองตารางมีคอลัมน์ 'Year' เหมือนกัน แต่ทีมการเงินไม่มีข้อมูลปี 2017 เราสามารถใช้ฟังก์ชันรวมสองตารางนี้ได้ คือ pd.merge ดังนี้

python

```
import pandas as pd
df_products = pd.DataFrame({
    'Year':pd.Series([2017, 2018, 2019, 2020, 2021, 2022, 2023]),
    'Price':pd.Series([199, 199, 199, 299, 299, 349, 349]),
    'Version':pd.Series(['v1', 'v1', 'v1', 'v2', 'v2', 'v2', 'v3'])})
df_products
```

	Year	Price	Version
0	2017	199	v1
1	2018	199	v1
2	2019	199	v1
3	2020	299	v2
4	2021	299	v2
5	2022	349	v2
6	2023	349	v3

รูปที่ 2.25 รวม DataFrame

```python
df_revenue = pd.DataFrame({
    'Year':pd.Series([2018, 2019, 2020, 2021, 2022, 2023]),
    'Revenue':pd.Series([9473, 8422, 9987, 7994, 9530, 9444])})
df_revenue
```

	Year	Revenue
0	2018	9473
1	2019	8422
2	2020	9987
3	2021	7994
4	2022	9530
5	2023	9444

```python
df = pd.merge(df_products, df_revenue)
df
```

	Year	Price	Version	Revenue
0	2018	199	v1	9473
1	2019	199	v1	8422
2	2020	299	v2	9987
3	2021	299	v2	7994
4	2022	349	v2	9530
5	2023	349	v3	9444

รูปที่ 2.25 (ต่อ) รวม DataFrame

หลังจากรวมสองชุดข้อมูลนี้แล้ว จะเห็นว่าฟังก์ชันจะรวมเฉพาะ Year ที่ทั้งสองชุดข้อมูลนี้มีเหมือนกันเท่านั้น ดังนั้นปี 2017 จะหายไป