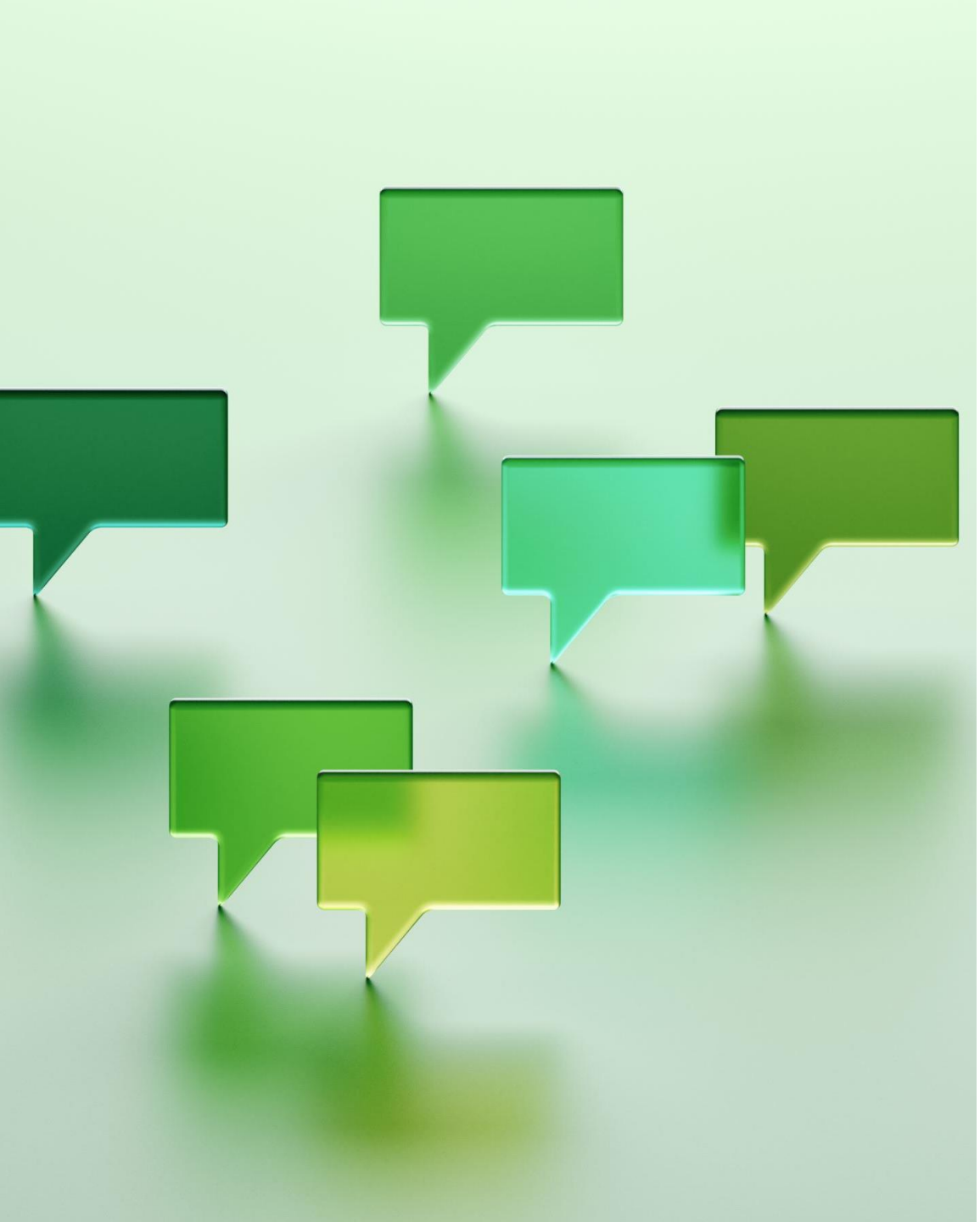


Machine Learning and Data Classification

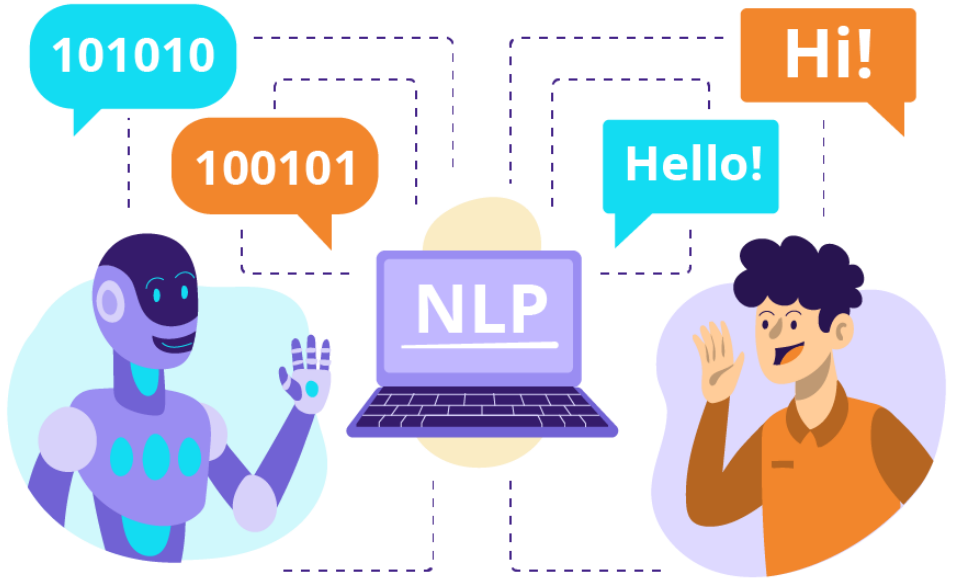
Natural Language Processing: NLP (01418363)

Asst. Prof. Dr Jirawan Charoensuk

Adapt materials from Asst. Prof. Dr. Aurawan Imsombut slide

Agenda

- Machine Learning
- Pandas and data cleaning
- Data Classification
  - Decision Tree
  - Naïve Bays
  - Random Forest



https://www.linkedin.com/pulse/natural-language-processing-bridging-gap-between-ocman-nazir-briet/

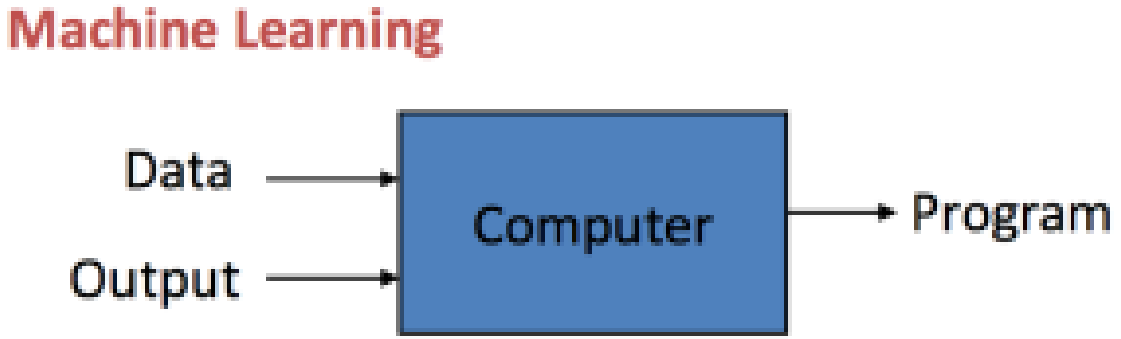
Machine Learning

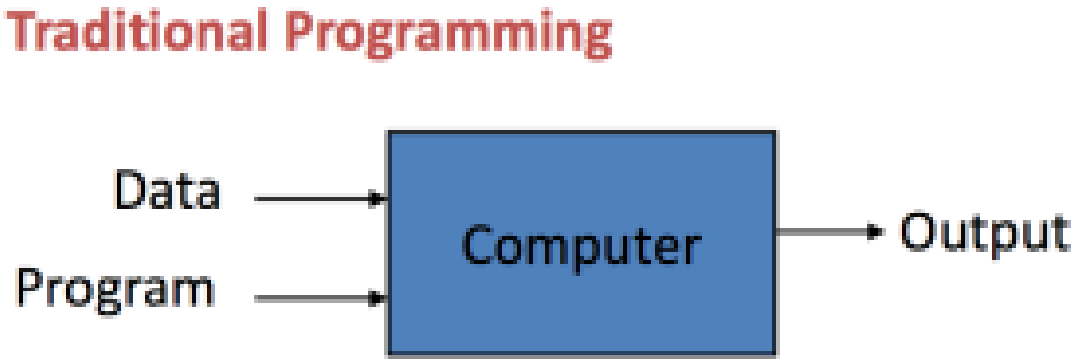
- การเรียนรู้ของเครื่อง (Machine learning - ML) เป็นศาสตร์หนึ่งของปัญญาประดิษฐ์ (AI)
- เป็นการศึกษาอัลกอริทึมของคอมพิวเตอร์ที่มีการพัฒนาการเรียนรู้ของเครื่อง
- อัลกอริทึมจะสร้างแบบจำลองทางคณิตศาสตร์จากข้อมูลตัวอย่าง (เรียกว่า ข้อมูลสอน) เพื่อที่จะคาดการณ์หรือตัดสินใจได้อย่างชัดเจน

Machine Learning

- Machine Learning คือ การทำให้ระบบคอมพิวเตอร์เรียนรู้ได้ด้วยตนเอง โดยใช้ข้อมูลสอน

- แตกต่างกับการเขียนโปรแกรมทั่วไป เพราะโปรแกรมทั่วไปจะใส่ ข้อมูล (Data) และ Program เข้าไปเพื่อให้ได้ Output

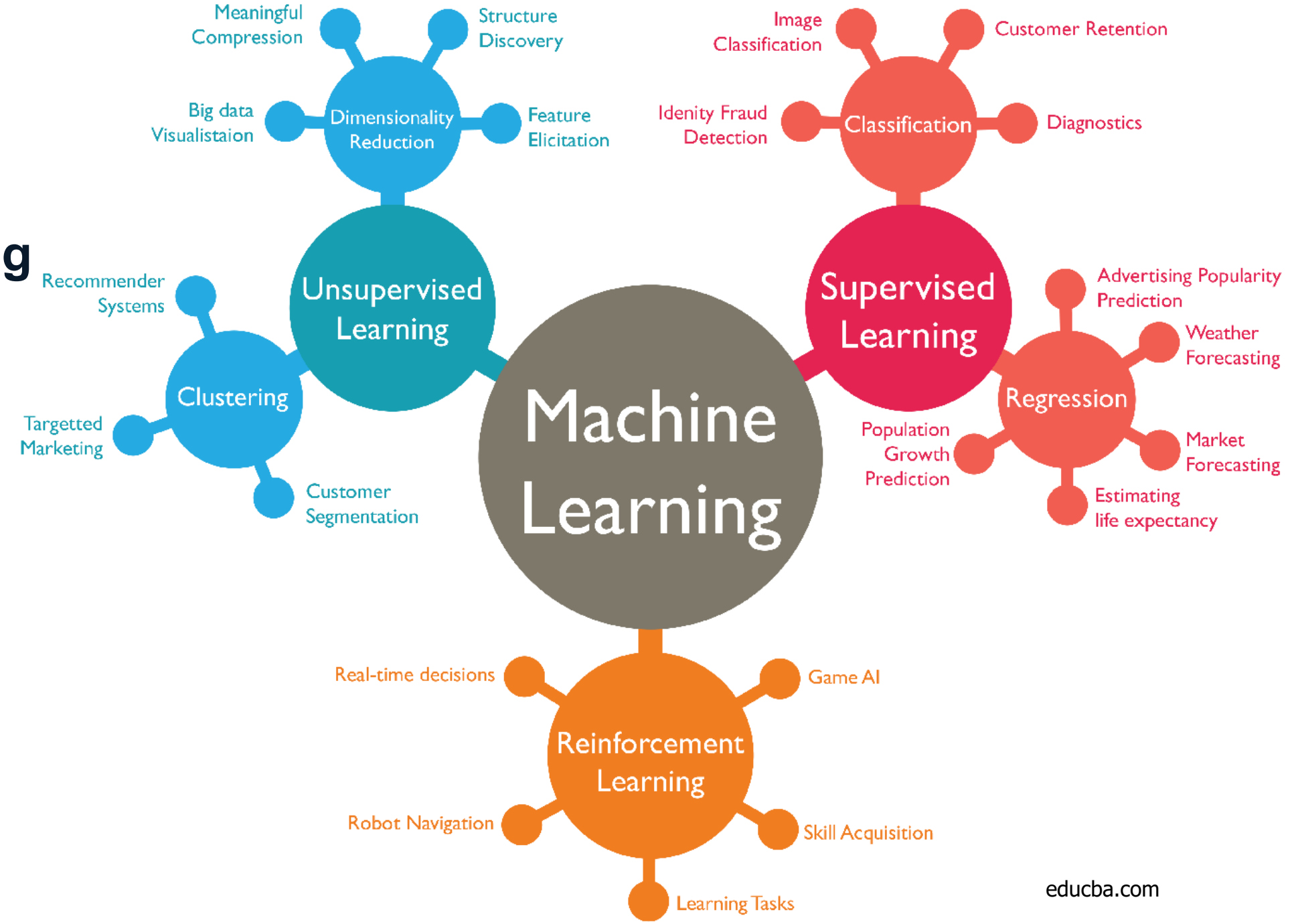




https://medium.com/investic/machine-learning-คืออะไร-fa8bf6663c07

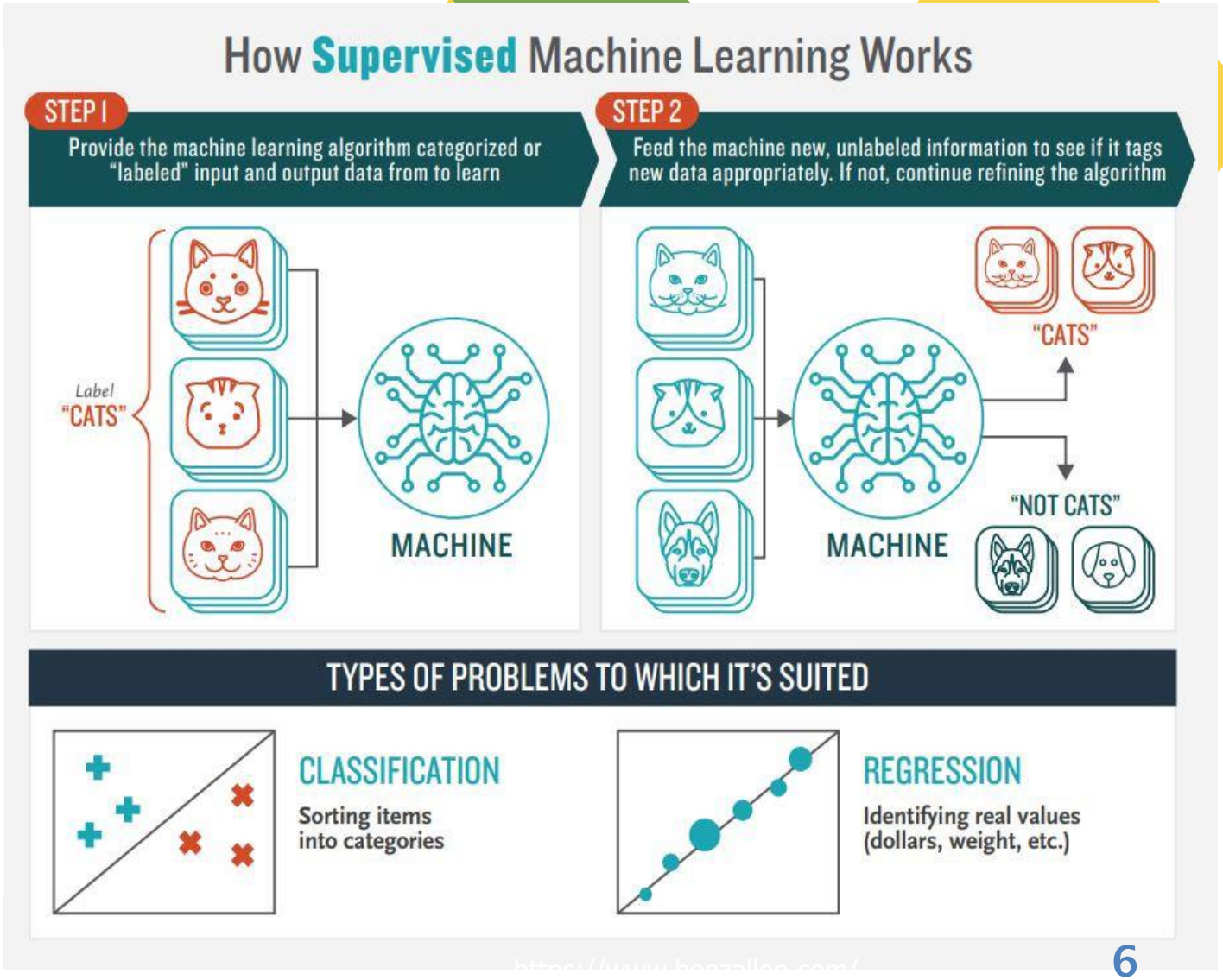
Types of

Machine Learning



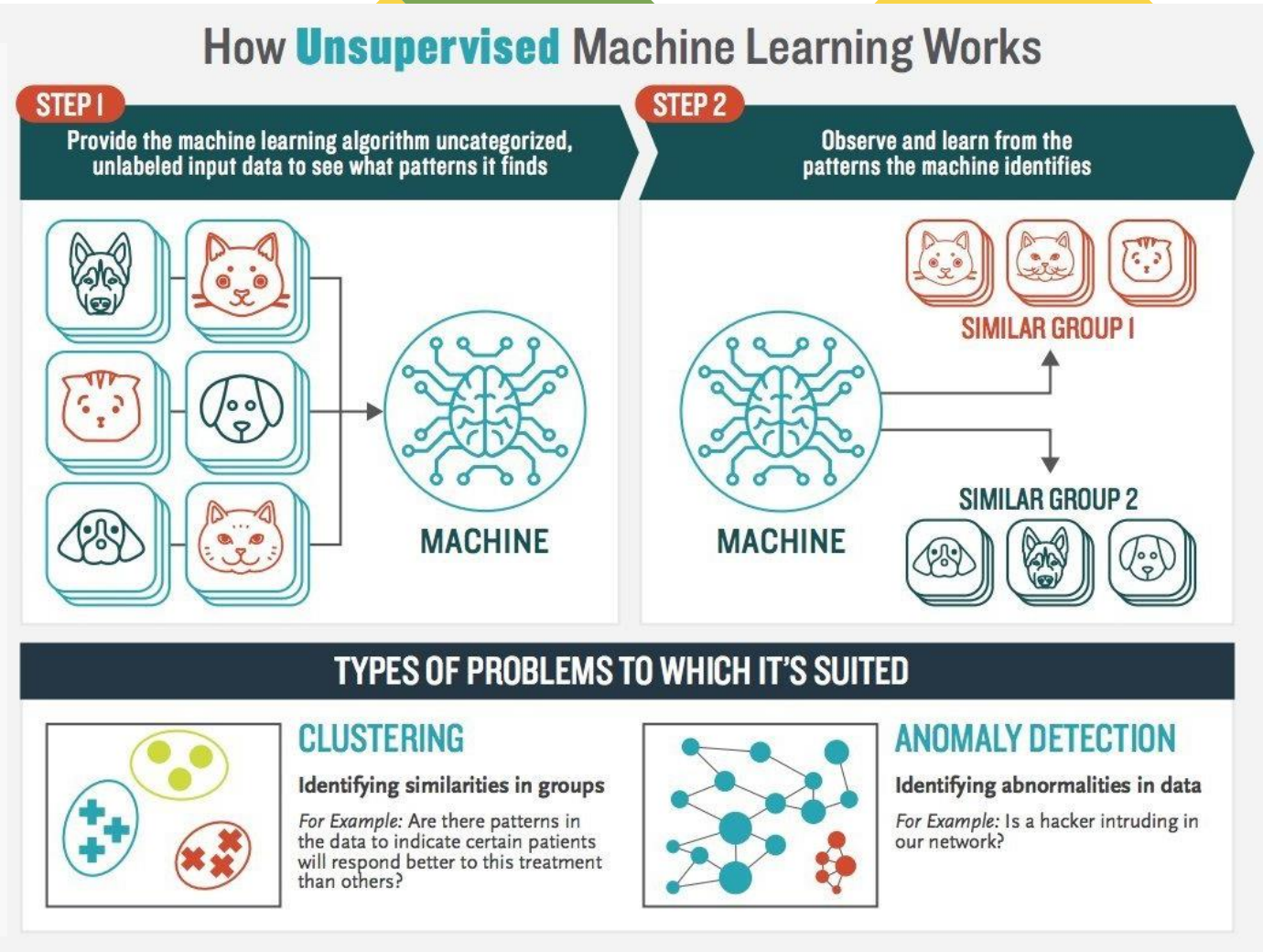
Types of Machine Learning

- **Supervised Learning**
  คือ การเรียนรู้โดยมีข้อมูลที่กำกับคำตอบมาสอน เช่น
- การจำแนก (Classification) ภาพแมว และไม่ใช่แมว
- การพยากรณ์ยอดขายจากงบโฆษณา (Regression)



Types of Machine Learning

- **Unsupervised Learning**
  เป็นการเรียนรู้แบบไม่มีผู้สอน (ไม่มีคำตอบให้)
  คอมพิวเตอร์เรียนรู้ความแตกต่างของข้อมูลแล้วแยกกลุ่มด้วยตัวเอง
- การแบ่งกลุ่มลูกค้า (Clustering)
- การตรวจจับข้อมูลที่ผิดปกติ (Anomaly Detection)

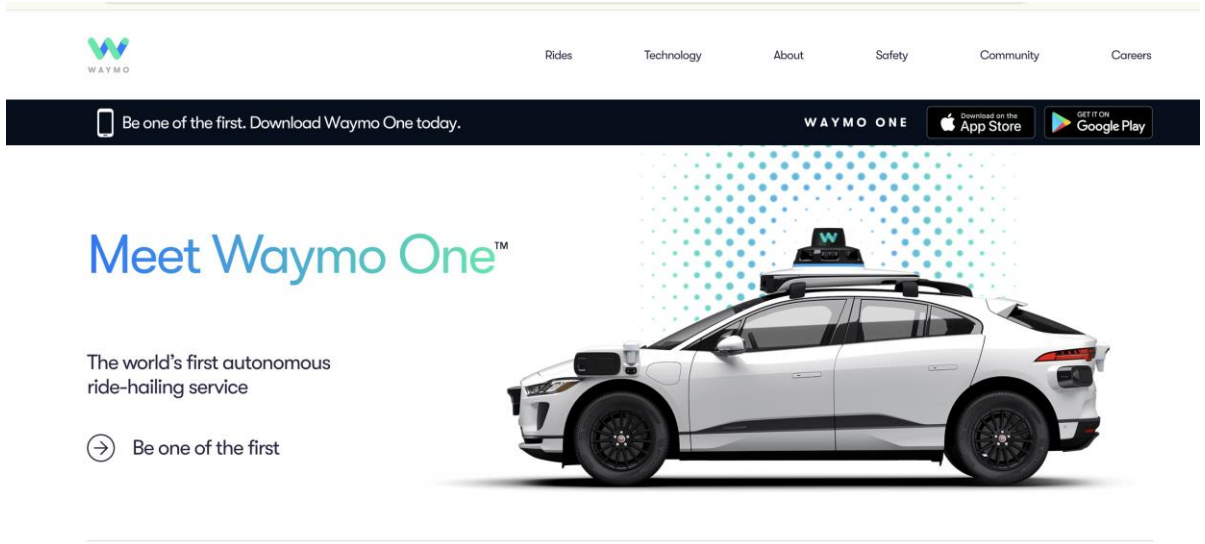


Types of Machine Learning

- **Reinforcement Learning** เป็นการเรียนรู้แบบลองผิดลองถูก และเรียนรู้ผ่านการให้รางวัล

เช่น ถ้าน้องหมาหิว อยากจะได้อาหาร สิ่งที่น้องหมาทำคือ ดูว่ามีใครอยู่บ้าง แล้วก็จะไปร้องขออาหาร และยืน 2 เท้า กับเจ้าของ ก็จะได้อาหารมา หลังจากได้อาหารมาน้องหมาก็จะเรียนรู้ว่าทำอย่างไรจึงจะได้อาหารที่ต้องการ

https://waymo.com/



Machine Learning Pipeline

1. Task Formulation
2. Data Preparation
3. Feature Engineering
4. Model Training
5. Evaluation
6. Deploy

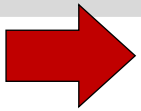
Machine Learning Pipeline

ขั้นตอน	คำอธิบาย	ตัวอย่างข้อมูล
Task Formulation	กำหนดเป้าหมายและประเภทปัญหา	จำแนกประเภทอีเมลเป็น Spam หรือ Non-Spam.
Data Preparation	เตรียมและทำความสะอาดข้อมูล	แบ่งข้อมูลเป็น Train (70%), Validation (15%), Test (15%).
Feature Engineering	สร้างและเลือกฟีเจอร์	ใช้ TF-IDF ของคำในข้อความ.
Model Training	ฝึกโมเดลโดยใช้ข้อมูลที่แบ่งไว้	โมเดล Naive Bayes กับข้อมูล Train.
Evaluation	ประเมินผลโดยใช้ตัวชี้วัดต่างๆ	Accuracy: 92%, Precision: 90%, Recall: 85%.
Deploy	นำโมเดลไปใช้งานจริง	สร้าง API วิเคราะห์อีเมลและตรวจสอบทุกเดือน.

Data Classification

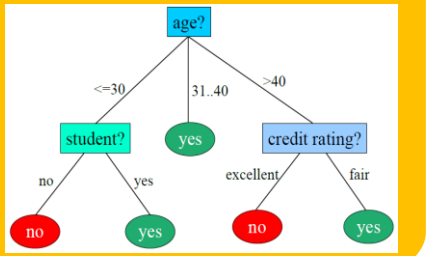
Training data

ID	age	income	student	credit_rating	buys_computer
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31…40	high	no	fair	yes
	>40	medium	no	fair	yes
	>40	low	yes	fair	yes

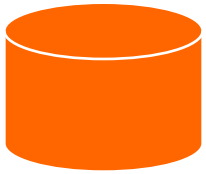


Classification Algorithms

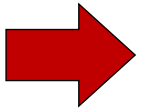
Classification model



Testing data



ID	age	income	student	credit_rating	buys_computer
1	31…40	medium	no	excellent	yes
2	31…40	high	yes	fair	yes
3	>40	medium	no	excellent	no



ID	buys_computer	predict
1	yes	yes
2	yes	yes
3	no	no

Performance

- ตัววัดประสิทธิภาพของโมเดล classification
  - **Confusion matrix**
  - **Precision and recall**
  - **F-measure**
  - **Accuracy**

Performance

- **Confusion matrix**

		Actual (buys_computer)	
		yes	**no**
Predicted	**yes**	True-Positive	False-Positive
	no	False-Negative	True-Negative

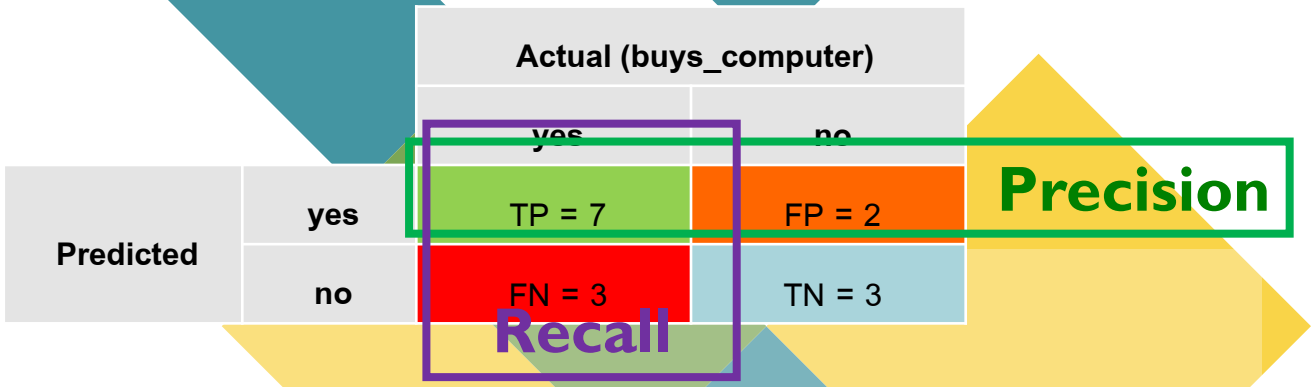
- True Positive (TP) คือ จำนวนข้อมูลที่จำแนกถูกว่าเป็นคลาสที่สนใจ
- True Negative (TN) คือ จำนวนข้อมูลที่จำแนกถูกว่าเป็นคลาสที่ไม่ได้สนใจ
- False Positive (FP) คือ จำนวนข้อมูลที่จำแนกผิดมาเป็นคลาสที่สนใจ
- False Negative (FN) คือ จำนวนข้อมูลที่จำแนกผิดมาเป็นคลาสที่ไม่ได้สนใจ

ID	buys_computer	predict
1	yes	yes
2	yes	no
3	yes	no
4	no	no
5	no	no
6	yes	yes
7	yes	yes
8	yes	yes
9	yes	yes
10	no	yes
11	yes	yes
12	yes	yes
13	no	no
14	no	yes
15	yes	no

		Actual (buys_computer)	
		yes	**no**
Predicted	**yes**	7	2
	no	3	3

Performance

- **Precision and recall**



- **Precision** คำนวณจากจำนวนที่ทำนายถูก หารด้วยข้อมูลที่ทำนายว่าเป็นคลาสที่พิจารณาทั้งหมด

- $\text{Precision} = \frac{\text{True Positive}}{\text{True Positive+False Positive}}$

$$\text{Precision}_{\text{(class yes)}} = \frac{7}{7+2} \times 100 = 77.78\%$$

- **Recall** คำนวณจากจำนวนข้อมูลที่ทำนายถูก หารด้วยจำนวนข้อมูลที่เป็นคลาสที่พิจารณาทั้งหมด

- $\text{Recall} = \frac{\text{True Positive}}{\text{True Positive+False Negative}}$

$$\text{Recall}_{\text{(class yes)}} = \frac{7}{7+3} \times 100 = 70.00\%$$

Performance

		Actual (buys_computer)	
		yes	no
Predicted	yes	7	2
	no	3	3

- **F-measure and Accuracy**
- **F-measure เป็นการวัดค่า Precision และ Recall พร้อมกันของโมเดล**

- $F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$

$$\text{F-measure}_{\text{(class yes)}} = \frac{2 * 77.78 * 70.00}{77.78 + 70.00} \times 100$$

$$= 73.68\%$$

- **Accuracy เป็นการวัดความถูกต้องของโมเดล โดยพิจารณารวมทุกคลาส**

- $Accuracy = \frac{True\ Positive + True\ Negative}{all}$

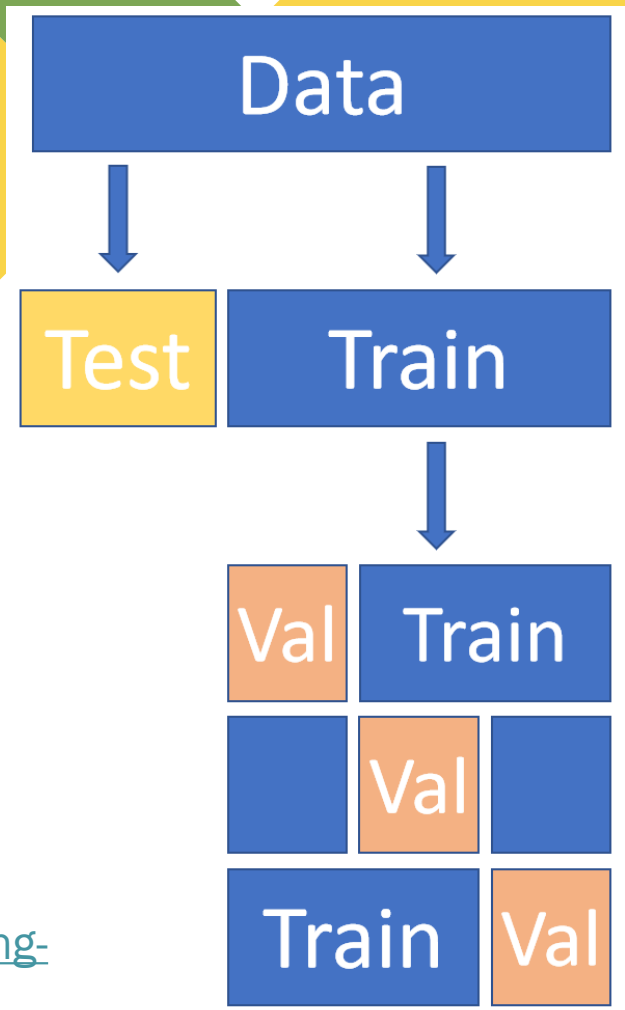
$$\text{Accuracy} = \frac{7+3}{7+2+3+3} \times 100 = 66.67\%$$

Validation

ก่อนการ Split ข้อมูลควร Shuffle หรือสับไพ่ ให้ข้อมูลทุกชุดมีการกระจายของข้อมูล หรือ Distribution ใกล้เคียงกัน ไม่เอนเอียง (Data Skew)

- การแบ่งข้อมูลเพื่อใช้ในการวัดประสิทธิภาพของโมเดล

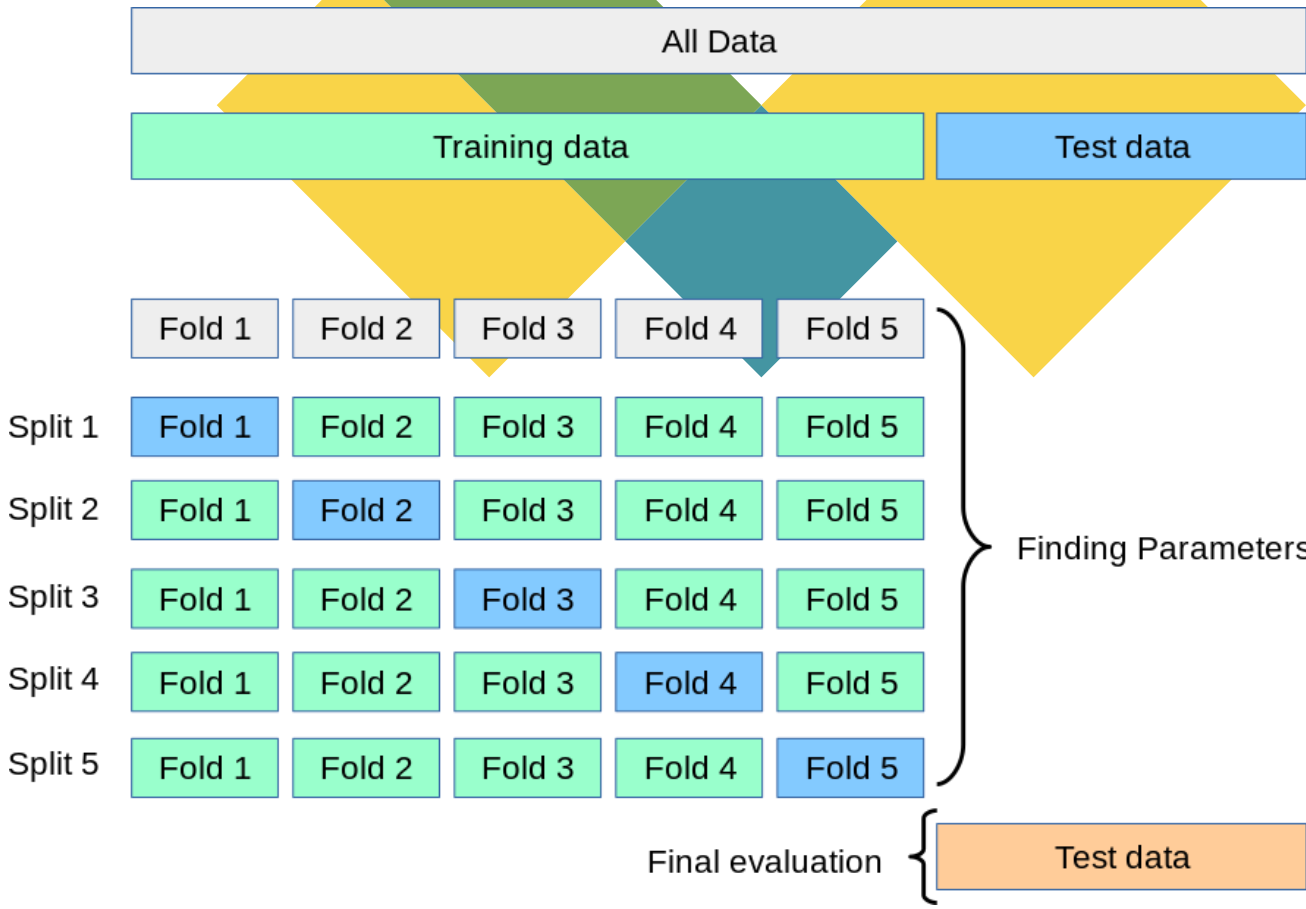
1. **Train set** เป็นชุดข้อมูลที่ใช้สำหรับเรียนรู้ เพื่อปรับพารามิเตอร์ให้เหมาะสม
2. **Validation set** คือ ชุดข้อมูลที่ใช้ประเมิน model ระหว่างการ train เพื่อปรับให้ model ทำนายได้ดีขึ้น
3. **Test set** คือ ชุดข้อมูลที่ใช้สำหรับทดสอบ model ก่อนนำไปใช้งานจริง



https://www.bualabs.com/archives/532/what-is-training-set-why-train-test-split-training-set-validation-set-test-set/#:~:text=Validation%20Set%20ใช้สำหรับทดสอบ,ไม่เคยเห็นมาก่อน

K-Fold Cross-Validation

- **K-Fold Cross-Validation** เป็นเทคนิคหนึ่งที่ใช้ในการประเมินประสิทธิภาพของโมเดล **Machine Learning** โดยการแบ่งข้อมูลเป็นหลายส่วน (**folds**) และฝึกโมเดลซ้ำ ๆ เพื่อให้มั่นใจว่าผลลัพธ์ที่ได้ไม่ขึ้นอยู่กับการแบ่งข้อมูลเพียงครั้งเดียว ซึ่งช่วยลดโอกาสเกิด **overfitting หรือ underfitting** และเพิ่มความน่าเชื่อถือของผลการประเมิน



https://scikit-learn.org/1.5/modules/cross_validation.html

กระบวนการของ K-Fold Cross-Validation

1. แบ่งข้อมูลเป็น K ส่วนที่เท่าๆ กัน (folds):

- ข้อมูลทั้งหมดถูกแบ่งออกเป็น K ชุดย่อย (folds) ที่มีขนาดใกล้เคียงกัน.

2. วนรอบการฝึกและทดสอบโมเดล KKK ครั้ง:

- ในแต่ละรอบ:

  1. ใช้ K−1 folds เป็นชุดข้อมูลสำหรับการฝึก (training set).
  2. ใช้ 1 fold ที่เหลือเป็นชุดข้อมูลสำหรับการทดสอบ (validation set).

3. คำนวณค่าเฉลี่ยของผลลัพธ์:

- หลังจากฝึกและทดสอบครบ K รอบ ให้คำนวณค่าเฉลี่ยของตัวชี้วัด (metrics) เช่น Accuracy, Precision, หรือ Mean Squared Error (MSE).

ภาพรวมกระบวนการ (ตัวอย่าง K = 5)

รอบที่	Training Set	Validation Set
1	Fold 2,3,4,5	Fold 1
2	Fold 1,3,4,5	Fold 2
3	Fold 1,2,4,5	Fold 3
4	Fold 1,2,3,5	Fold 4
5	Fold 1,2,3,4	Fold 5

Pandas and data cleaning

What is Pandas?

- Pandas เป็นไลบรารีภาษา Python ที่ใช้ในการวิเคราะห์ข้อมูลในลักษณะที่เป็นตาราง
- มีฟังก์ชันการวิเคราะห์ ทำความสะอาด ตรวจสอบ และจัดการข้อมูล
- ใช้ dataframe เป็นโครงสร้างพื้นฐานในการจัดการข้อมูลแบบตาราง คือมี แถว และคอลัมน์

- ติดตั้ง Pandas

```
import pandas
```

Pandas DataFrames

- DataFrame เป็นตารางข้อมูลแบบ 2 มิติ ประกอบด้วยแถว และคอลัมน์

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)

#refer to the row index:
print(df.loc[0])

#use a list of indexes:
print(df.loc[[0, 1]])
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

```
calories    420
duration     50
Name: 0, dtype: int64
```

```
   calories  duration
0       420        50
1       380        40
```

Pandas: Merge

- เชื่อมข้อมูลของ 2 DataFrame

Parameter	Value	Description
how	'left' 'right' 'outer' 'inner' 'cross'	Optional. Default 'inner'. Specifies how to merge
left_on	String List	Optional. Specifies in what level to do the merging on the DataFrame to the right
right_on	String List	Optional. Specifies in what level to do the merging on the DataFrame to the right
sort	True False	Optional. Default False. Specifies whether to sort the DataFrame by the join key or not

```
import pandas as pd

data1 = {
  "name": ["Sally", "Mary", "John"],
  "age": [50, 40, 30]
}

data2 = {
  "name": ["Sally", "Peter", "Micky"],
  "weight": [61, 44, 22]
}

df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)

newdf = df1.merge(df2, how='inner', left_on='name', right_on='name')

print(newdf)
```

```
    name  age  weight
0  Sally   50      61
```

Pandas DataFrames

- โหลดข้อมูลจากไฟล์มาสร้าง DataFrame

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df)

#print all data:
print(df.to_string())
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.4
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
```

Pandas - Viewing the Data and Information

- แสดงข้อมูลบางส่วน และแสดงรายละเอียดสรุปของข้อมูล

```
import pandas as pd

df = pd.read_csv('data.csv')

#print head 5 rows (default):
print(df.head())

#print tail 10 rows:
print(df.tail(10))

#print random 5 rows:
print(df.sample(5))

print(df.info())
```

head

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0

sample

	Duration	Pulse	Maxpulse	Calories
21	45	100	119	282.0
85	30	151	170	300.1
114	60	108	131	367.6
23	45	105	132	246.0
98	30	109	131	188.2

tail

	Duration	Pulse	Maxpulse	Calories
159	30	80	120	240.9
160	30	85	120	250.4
161	45	90	130	260.4
162	45	95	130	270.0
163	45	100	140	280.9
164	60	105	140	290.8
165	60	110	145	300.4
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

info

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  169 non-null    int64
 1   Pulse     169 non-null    int64
 2   Maxpulse  169 non-null    int64
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None
```

Cleaning Data

- การทำ cleaning data ด้วย pandas
  - Empty cell (missing data)
  - Data in wrong format
  - Wrong data
  - Duplicate

Cleaning Data

- Data set for the study

- Empty cells (row 18, 22, and 28)
- wrong format ("Date" in row 26)
- wrong data ("Duration" in row 7)
- duplicates (row 11 and 12)

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	2020/12/26	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

Pandas - Cleaning Empty Cells (missing data)

- isnull(): ตรวจสอบว่ามีข้อมูล missing data หรือไม่

```
import pandas as pd

df = pd.read_csv('data.csv’)

# Detect missing values
df.isnull()

# count missing values
df.isnull().sum()

# Detect missing values of col. Date
df["Date"].isnull()

# show the data row of missing values
df[df["Date"].isnull()]
```

	Duration	Date	Pulse	Maxpulse	Calories
22	45	NaN	100	119	282.0

	Duration	Date	Pulse	Maxpulse	Calories
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
5	False	False	False	False	False
6	False	False	False	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False
10	False	False	False	False	False
11	False	False	False	False	False
12	False	False	False	False	False
13	False	False	False	False	False
14	False	False	False	False	False
15	False	False	False	False	False
16	False	False	False	False	False
17	False	False	False	False	False
18	False	False	False	False	True
19	False	False	False	False	False
20	False	False	False	False	False
21	False	False	False	False	False
22	False	True	False	False	False
23	False	False	False	False	False
24	False	False	False	False	False
25	False	False	False	False	False
26	False	False	False	False	False
27	False	False	False	False	False
28	False	False	False	False	True
29	False	False	False	False	False

```
Duration     0
Date         1
Pulse        0
Maxpulse     0
Calories     2
dtype: int64
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22     True
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
```

Pandas - Cleaning Empty Cells

- **dropna(): ลบแถวที่มีค่า NULL value ออก**

```
import pandas as pd

df = pd.read_csv('data.csv’)

#the dropna() method returns a new DataFrame
new_df = df.dropna()
print(new_df.to_string())

#change the original DataFrame
# use the inplace = True argument:
df.dropna(inplace = True)
print(df.to_string())
```

ลบข้อมูลแถวไหนไป ?

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5

Pandas - Cleaning Empty Cells

- ลบคอลัมน์ที่มีค่า missing data

```
import pandas as pd

df = pd.read_csv('data.csv')

# delete missing data column
df.dropna(axis='columns', inplace = True)
print(df.to_string())

# delete missing data column if the column
# has non-missing data less than 31 row
df.dropna(axis=1, thresh=31, inplace = True)
print(df.to_string())
```

	Duration	Pulse	Maxpulse
0	60	110	130
1	60	117	145
2	60	103	135
3	45	109	175
4	45	117	148
5	60	102	127
6	60	110	136
7	450	104	134
8	30	109	133
9	60	98	124
10	60	103	147
11	60	100	120
12	60	100	120
13	60	106	128
14	60	104	132
15	60	98	123
16	60	98	120
17	60	100	120
18	45	90	112
19	60	103	123
20	45	97	125
21	60	108	131
22	45	100	119
23	60	130	101
24	45	105	132
25	60	102	126
26	60	100	120
27	60	92	118
28	60	103	132
29	60	100	132
30	60	102	129
31	60	92	115

	Duration	Date	Pulse	Maxpulse
0	60	'2020/12/01'	110	130
1	60	'2020/12/02'	117	145
2	60	'2020/12/03'	103	135
3	45	'2020/12/04'	109	175
4	45	'2020/12/05'	117	148
5	60	'2020/12/06'	102	127
6	60	'2020/12/07'	110	136
7	450	'2020/12/08'	104	134
8	30	'2020/12/09'	109	133
9	60	'2020/12/10'	98	124
10	60	'2020/12/11'	103	147
11	60	'2020/12/12'	100	120
12	60	'2020/12/12'	100	120
13	60	'2020/12/13'	106	128
14	60	'2020/12/14'	104	132
15	60	'2020/12/15'	98	123
16	60	'2020/12/16'	98	120
17	60	'2020/12/17'	100	120
18	45	'2020/12/18'	90	112
19	60	'2020/12/19'	103	123
20	45	'2020/12/20'	97	125
21	60	'2020/12/21'	108	131
22	45	NaN	100	119
23	60	'2020/12/23'	130	101
24	45	'2020/12/24'	105	132
25	60	'2020/12/25'	102	126
26	60	20201226	100	120
27	60	'2020/12/27'	92	118
28	60	'2020/12/28'	103	132
29	60	'2020/12/29'	100	132
30	60	'2020/12/30'	102	129
31	60	'2020/12/31'	92	115

Pandas - Cleaning Empty Cells

- fillna(): แทนที่ค่า NULL value ทั้งหมดด้วยค่าที่กำหนด

```
import pandas as pd

df = pd.read_csv('data.csv')

df.fillna(130, inplace = True)
```

16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	130.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	130	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	2020/12/26	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	130.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

- ตัวอย่างการแทนที่ค่า NULL value เฉพาะคอลัมน์ที่ต้องการด้วยค่าที่กำหนด

```
import pandas as pd

df = pd.read_csv('data.csv')

df["Calories"].fillna(130, inplace = True)
```

16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	130.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	2020/12/26	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	130.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

Pandas - Cleaning Empty Cells

- ตัวอย่างการแทนที่ค่า NULL value เฉพาะคอลัมน์ที่ต้องการด้วยค่าต่าง ๆ
  - ค่าเฉลี่ย (mean)
  - ค่ากลาง (median)
  - ค่าฐานนิยม (mode)

```
import pandas as pd

df = pd.read_csv('data.csv')

x = df["Calories"].mean()
# x = df["Calories"].median()
# x = df["Calories"].mode()[0]

df["Calories"].fillna(x, inplace = True)
```

16	60	'2020/12/16'	98	120	215.20
17	60	'2020/12/17'	100	120	300.00
18	45	'2020/12/18'	90	112	304.68
19	60	'2020/12/19'	103	123	323.00
20	45	'2020/12/20'	97	125	243.00
21	60	'2020/12/21'	108	131	364.20
22	45	NaN	100	119	282.00
23	60	'2020/12/23'	130	101	300.00
24	45	'2020/12/24'	105	132	246.00
25	60	'2020/12/25'	102	126	334.50
26	60	2020/12/26	100	120	250.00
27	60	'2020/12/27'	92	118	241.00
28	60	'2020/12/28'	103	132	304.68
29	60	'2020/12/29'	100	132	280.00
30	60	'2020/12/30'	102	129	380.30
31	60	'2020/12/31'	92	115	243.00

Pandas - Cleaning Data of Wrong Format

- to_datetime(): แปลงข้อมูลให้อยู่ในรูปแบบ วันเวลา

```
import pandas as pd

df = pd.read_csv('data.csv')

df['Date'] = pd.to_datetime(df['Date'])

print(df.to_string())
```

- ลบแถวข้อมูลที่ไม่ใช่รูปแบบ Date ที่ถูกต้อง

```
df.dropna(subset=['Date'], inplace = True)
```

ค่า **NaT** ย่อมาจาก "Not a Time" และใช้แทนค่าที่ขาดหาย **(missing value)** ใน **ข้อมูลประเภทเวลา (datetime)** เช่นเดียวกับที่ **NaN** ใช้แทนค่าขาดหายในข้อมูลประเภทตัวเลขหรือข้อความ

16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaT	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	'2020/12/26'	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

NAT (Not a Time) : non-parseable dates

Pandas - Fixing Wrong Data

- **loc[ ] : แทนที่ค่าที่ต้องการ ทีละตำแหน่ง**
  - ตัวอย่างการแทนค่าในแถวที่ 7 คอลัมน์ Duration ด้วยค่า 45

```
df.loc[7, 'Duration'] = 45
```

  - ตัวอย่างการวนลูป เพื่อแทนที่ค่าที่ตรงตามเงื่อนไขที่กำหนด ให้เป็นค่าอื่นที่ต้องการ

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.loc[x, "Duration"] = 120
```

  - ตัวอย่างการวนลูป เพื่อลบแถวที่มี ค่าที่ตรงตามเงื่อนไขที่กำหนด

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.drop(x, inplace = True)
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	45	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	120	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3

Pandas - Removing Duplicates

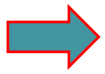
- duplicated() : ตรวจสอบว่ามีข้อมูลซ้ำกับแถวอื่นหรือไม่

```
import pandas as pd

df = pd.read_csv('data.csv’)

print(df.duplicated())
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12     True
13    False
14    False
15    False
```

- drop_duplicates() : ลบแถวที่มีข้อมูลซ้ำ

```
df.drop_duplicates(inplace = True)
```



	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0

Pandas - Data Correlations

- **corr() : แสดงค่าความสัมพันธ์ระหว่างคอลัมน์ต่าง ๆ**

```
df.corr()
```

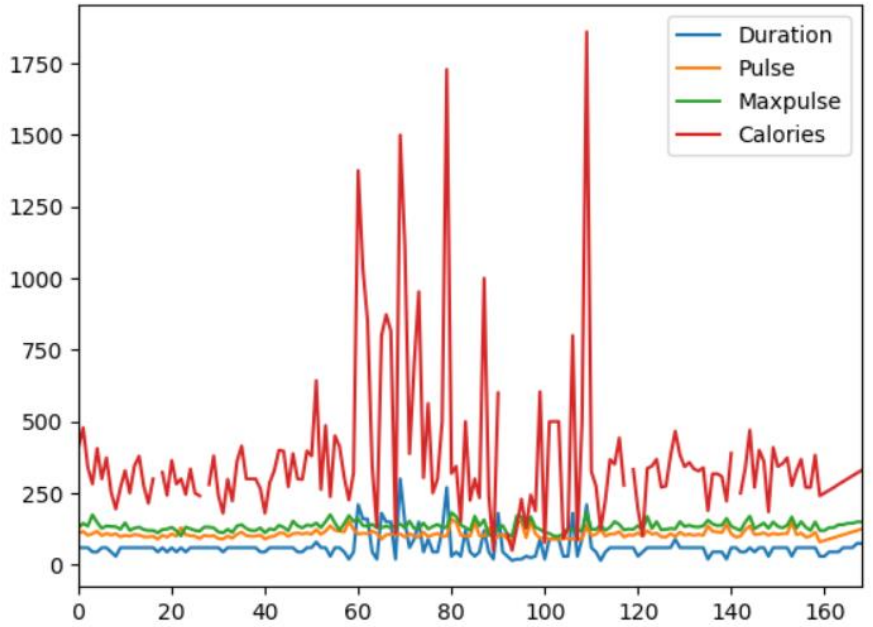
	Duration	Pulse	Maxpulse	Calories
Duration	1.000000	-0.155408	0.009403	0.922721
Pulse	-0.155408	1.000000	0.786535	0.025120
Maxpulse	0.009403	0.786535	1.000000	0.203814
Calories	0.922721	0.025120	0.203814	1.000000

- **Perfect Correlation:**
  - "Duration" and "Duration" got the number 1.000000
- **Good Correlation:**
  - "Duration" and "Calories" got a 0.922721 correlation, which is a very good correlation
  - we can predict that the longer you work out, the more calories you burn
- **Bad Correlation:**
  - "Duration" and "Maxpulse" got a 0.009403 correlation, which is a very bad correlation
  - we can not predict the max pulse by just looking at the duration of the work out, and vice versa.

Pandas - Plotting

- plot() : ฟังก์ชันในการสร้าง diagrams
- pyplot เป็นโมดูลย่อยใน Matplotlib library ใช้ในการแสดงภาพกราฟออกทางหน้าจอ

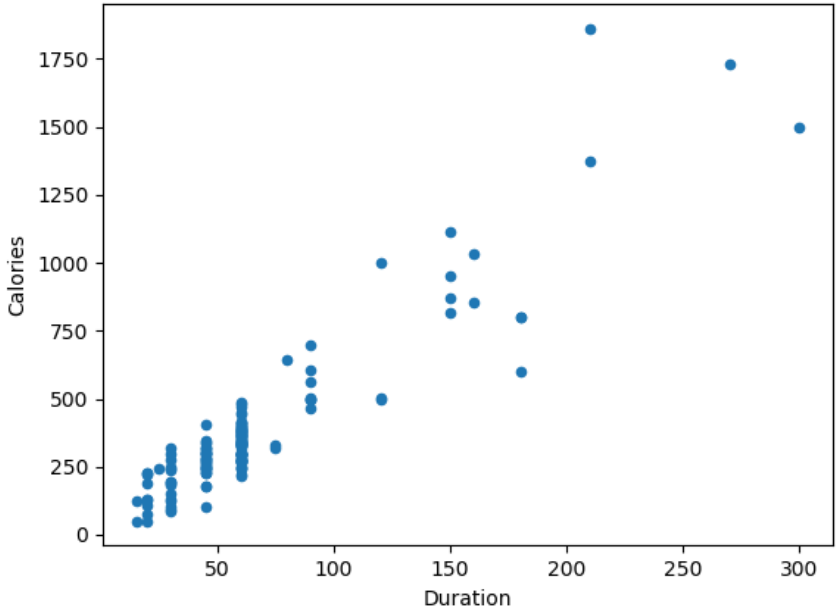
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

df.plot()

plt.show()
```



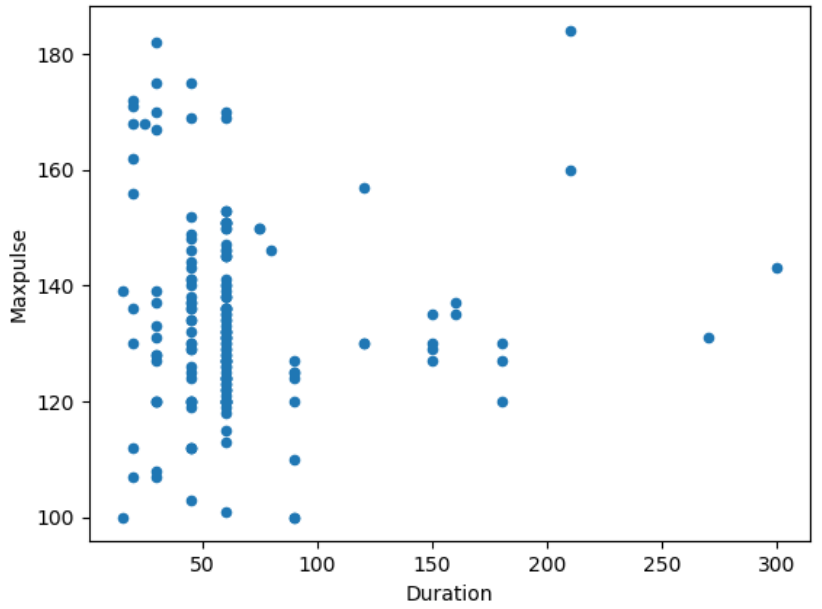
Pandas - Plotting

- กราฟ scatter

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')

plt.show()
```



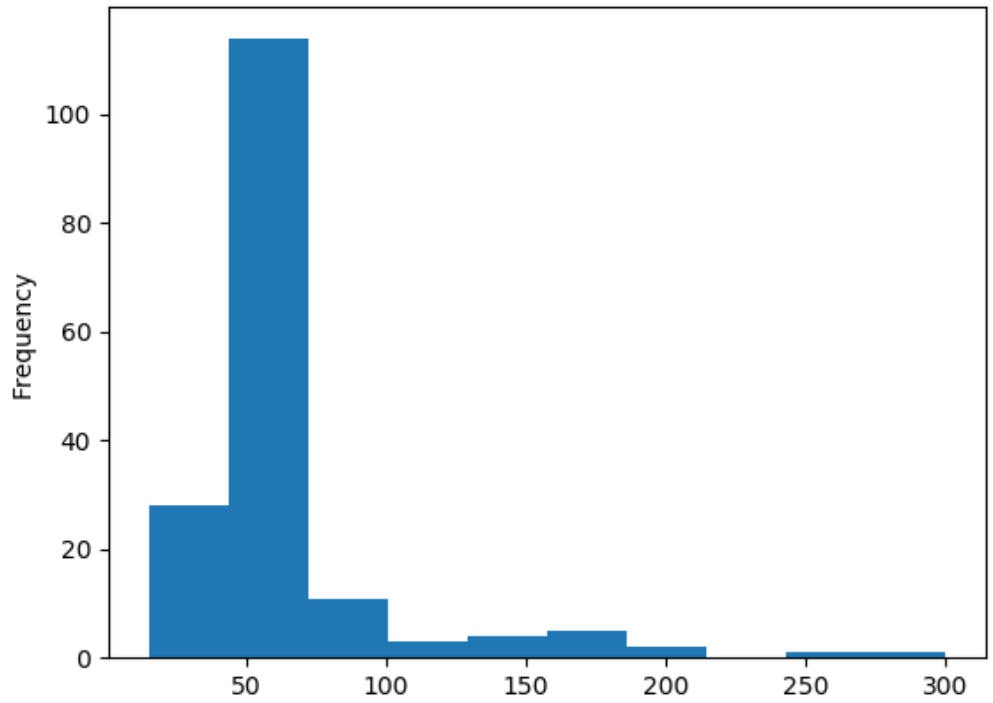

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

df.plot(kind = 'scatter', x = 'Duration', y = 'Maxpulse')

plt.show()
```



Pandas - Plotting

- กราฟ histogram

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv’)

# use the "Duration" column to create the histogram
df["Duration"].plot(kind = 'hist')

plt.show()
```



Plot types in Matplotlib:

https://matplotlib.org/stable/plot_types/index.html

Pandas Cheat Sheet

https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

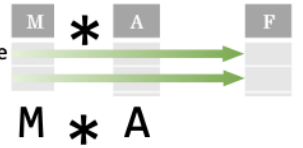
Pandas API Reference Pandas User Guide

Tidy Data – A foundation for wrangling in pandas

Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements pandas's vectorized operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.



Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4, 5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```
Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```
Specify values for each row.

		a	b	c
N	v			
D	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d', 1), ('d', 2),
         ('e', 2)], names=['n', 'v']))
```
Create DataFrame with a MultiIndex

Method Chaining

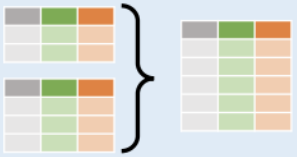
Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable':'var',
                'value':'val'})
        .query('val >= 200')
    )
```

Reshaping Data – Change layout, sorting, reindexing, renaming

`pd.melt(df)`
Gather columns into rows.

`df.pivot(columns='var', values='val')`
Spread rows into columns.



`pd.concat([df1,df2])`
Append rows of DataFrames

`pd.concat([df1,df2], axis=1)`
Append columns of DataFrames

`df.sort_values('mpg')`
Order rows by values of a column (low to high).

`df.sort_values('mpg', ascending=False)`
Order rows by values of a column (high to low).

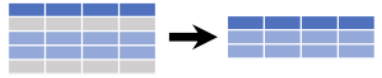
`df.rename(columns = {'y':'year'})`
Rename the columns of a DataFrame

`df.sort_index()`
Sort the index of a DataFrame

`df.reset_index()`
Reset index of DataFrame to row numbers, moving index to columns.

`df.drop(columns=['Length', 'Height'])`
Drop columns from DataFrame

Subset Observations - rows



`df[df.Length > 7]`
Extract rows that meet logical criteria.

`df.drop_duplicates()`
Remove duplicate rows (only considers columns).

`df.sample(frac=0.5)`
Randomly select fraction of rows.

`df.sample(n=10)` Randomly select n rows.

`df.nlargest(n, 'value')`
Select and order top n entries.

`df.nsmallest(n, 'value')`
Select and order bottom n entries.

`df.head(n)`
Select first n rows.

`df.tail(n)`
Select last n rows.

Subset Variables - columns

`df[['width', 'length', 'species']]`
Select multiple columns with specific names.

`df['width']` *or* `df.width`
Select single column with specific name.

`df.filter(regex='regex')`
Select columns whose name matches regular expression *regex*.

Using query

query() allows Boolean expressions for filtering rows.

```
df.query('Length > 7')
df.query('Length > 7 and Width < 8')
df.query('Name.str.startswith("abc")',
         engine="python")
```

Subsets - rows and columns

Use `df.loc[]` and `df.iloc[]` to select only rows, only columns or both.
Use `df.at[]` and `df.iat[]` to access a single value by row and column.
First index selects rows, second index columns.

`df.iloc[10:20]`
Select rows 10-20.

`df.iloc[:, [1, 2, 5]]`
Select columns in positions 1, 2 and 5 (first column is 0).

`df.loc[:, 'x2':'x4']`
Select all columns between x2 and x4 (inclusive).

`df.loc[df['a'] > 10, ['a', 'c']]`
Select rows meeting logical condition, and only the specific columns .

`df.iat[1, 2]` Access single value by index

`df.at[4, 'A']` Access single value by label

Logic in Python (and pandas)

<	Less than	!=	Not equal to
>	Greater than	`df.column.isin(values)`	Group membership
==	Equals	`pd.isnull(obj)`	Is NaN
<=	Less than or equals	`pd.notnull(obj)`	Is not NaN
>=	Greater than or equals	`&,|,~,^,df.any(),df.all()`	Logical and, or, not, xor, any, all

regex (Regular Expressions) Examples

`'\.'`	Matches strings containing a period '.'
`'Length$'`	Matches strings ending with word 'Length'
`'^Sepal'`	Matches strings beginning with the word 'Sepal'
`'^x[1-5]$'`	Matches strings beginning with 'x' and ending with 1,2,3,4,5
`'^(?!Species$).*'`	Matches strings except the string 'Species'

Summarize Data

`df['w'].value_counts()`
Count number of rows with each unique value of variable

`len(df)`
# of rows in DataFrame.

`df.shape`
Tuple of # of rows, # of columns in DataFrame.

`df['w'].nunique()`
# of distinct values in a column.

`df.describe()`
Basic descriptive and statistics for each column (or GroupBy).

pandas provides a large set of summary functions that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

`sum()`
Sum values of each object.

`count()`
Count non-NA/null values of each object.

`median()`
Median value of each object.

`quantile([0.25,0.75])`
Quantiles of each object.

`apply(function)`
Apply function to each object.

`min()`
Minimum value in each object.

`max()`
Maximum value in each object.

`mean()`
Mean value of each object.

`var()`
Variance of each object.

`std()`
Standard deviation of each object.

Handling Missing Data

`df.dropna()`
Drop rows with any column having NA/null data.

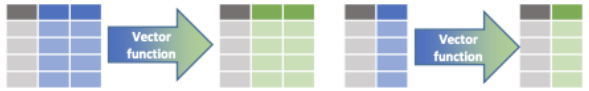
`df.fillna(value)`
Replace all NA/null data with value.

Make New Columns

`df.assign(Area=lambda df: df.Length*df.Height)`
Compute and append one or more new columns.

`df['Volume'] = df.Length*df.Height*df.Depth`
Add single column.

`pd.qcut(df.col, n, labels=False)`
Bin column into n buckets.



pandas provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

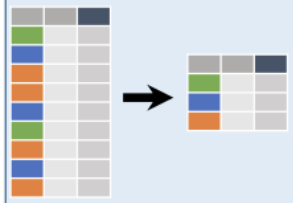
`max(axis=1)`
Element-wise max.

`clip(lower=-10,upper=10)`
Trim values at input thresholds

`min(axis=1)`
Element-wise min.

`abs()`
Absolute value.

Group Data



`df.groupby(by="col")`
Return a GroupBy object, grouped by values in column named "col".

`df.groupby(level="ind")`
Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

`size()`
Size of each group.

`agg(function)`
Aggregate group using function.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

`shift(1)`
Copy with values shifted by 1.

`rank(method='dense')`
Ranks with no gaps.

`rank(method='min')`
Ranks. Ties get min rank.

`rank(pct=True)`
Ranks rescaled to interval [0, 1].

`rank(method='first')`
Ranks. Ties go to first value.

`shift(-1)`
Copy with values lagged by 1.

`cumsum()`
Cumulative sum.

`cummax()`
Cumulative max.

`cummin()`
Cumulative min.

`cumprod()`
Cumulative product.

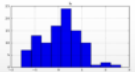
Windows

`df.expanding()`
Return an Expanding object allowing summary functions to be applied cumulatively.

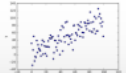
`df.rolling(n)`
Return a Rolling object allowing summary functions to be

Plotting

`df.plot.hist()`
Histogram for each column



`df.plot.scatter(x='w',y='h')`
Scatter chart using pairs of points



Combine Data Sets

adf

x1	x2
A	1
B	2
C	3

+ bdf

x1	x3
A	T
B	F
D	T

=

Standard Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

`pd.merge(adf, bdf, how='left', on='x1')`
Join matching rows from bdf to adf.

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

`pd.merge(adf, bdf, how='right', on='x1')`
Join matching rows from adf to bdf.

x1	x2	x3
A	1	T
B	2	F

`pd.merge(adf, bdf, how='inner', on='x1')`
Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

`pd.merge(adf, bdf, how='outer', on='x1')`
Join data. Retain all values, all rows.

Filtering Joins

x1	x2
A	1
B	2

`adf[adf.x1.isin(bdf.x1)]`
All rows in adf that have a match in bdf.

x1	x2
C	3

`adf[~adf.x1.isin(bdf.x1)]`
All rows in adf that do not have a match in bdf.

ydf

x1	x2
A	1
B	2
C	3

+ zdf

x1	x2
B	2
C	3
D	4

=

Set-like Operations

x1	x2
B	2
C	3

`pd.merge(ydf, zdf)`
Rows that appear in both ydf and zdf (Intersection).

x1	x2
A	1
B	2
C	3
D	4

`pd.merge(ydf, zdf, how='outer')`
Rows that appear in either or both ydf and zdf (Union).

`pd.merge(ydf, zdf, how='outer', indicator=True).query('_merge == "left_only"').drop(columns=['_merge'])`
Rows that appear in ydf but not zdf (Setdiff).

x1	x2
A	1

Exercise

- ให้นิสิต cleaning ข้อมูลไฟล์ Airbnb_NewYork_2019.csv
- https://colab.research.google.com/drive/17ur6PCZks02tNlajqSPiYs2nFTb8F0yS?usp=sharing

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	id	name	host_id	host_name	neighbourho	neighbourho	latitude	longitude	room_type	price	minimum_ni	number_of_	last_review	reviews_per_	calculated_h	availability_365
2	2539	Clean & quiet apt	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	19/10/2018	0.21	6	365
3	2595	Skylit Midtown Ca	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/	225	1	45	21/5/2019	0.38	2	355
4	3647	THE VILLAGE OF H	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.9419	Private room	150	3	0			1	365
5	3831	Cozy Entire Floor c	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/	89	1	270	5/7/2019	4.64	1	194
6	5022	Entire Apt: Spacio	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/	80	10	9	19/11/2018	0.1	1	0
7	5099	Large Cozy 1 BR A	7322	Chris	Manhattan	Murray Hill	40.74767	-73.975	Entire home/	200	3	74	22/6/2019	0.59	1	129
8	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-Stuy	40.68688	-73.95596	Private room	60	45	49	5/10/2017	0.4	1	0
9	5178	Large Furnished R	8967	Shunichi	Manhattan	Hell's Kitche	40.76489	-73.98493	Private room	79	2	430	24/6/2019	3.47	1	2200
10	5203	Cozy Clean Guest I	7490	MaryEllen	Manhattan	Upper West Side		-73.96723	Private room		2	118	11/10/2019	0.99	1	0
11	5238	Cute & Cozy Lowe	7549	Ben	Manhattan	Chinatown		-73.99037	Entire home/	150	1	160	9/6/2019	1.33	4	188
12	5295	Beautiful 1br on U	7702	Lena	Manhattan	Upper West Side		-73.96545	Entire home/	135	5	53	22/6/2019	0.43	1	6
13	5441	Central Manhattar	7989	Kate	Manhattan	Hell's Kitchen		-73.98867	Private room	85	2	188	23/6/2019	1.5	1	39
14	5803	Lovely Room 1, Ga	9744	Laurie	Brooklyn	South Slope		-73.98779	Private room	89	4	167	24/6/2019	1.34	3	314
15	6021	Wonderful Guest I	11528	Claudio	Manhattan	Upper West	40.79826	-73.96113	Private room		2	113	5/7/2019	0.91	1	3333
16	6090	West Village Nest	11975	Alina	Manhattan	West Village	40.7353	-74.00525	Entire home/apt		90	27	31/10/2018	0.22	1	0
17	6848	Only 2 stops to Ma	15991	Allen & Irina	Brooklyn	Williamsburg	40.70837	-73.95352	Entire home/	140	2	148	29/6/2019	1.2	1	46
18	7097	Perfect for Your Pa	17571	Jane	Brooklyn	Fort Greene	40.69169		Entire home/	215	2	198	28/6/2019	1.72	1	321
19	7322	Chelsea Perfect	18946	Doti	Manhattan	Chelsea	40.74192		Private room	140	1	260	1/7/2019	2.12	1	12
20	7726	Hip Historic Brown	20950	Adam And Cl	Brooklyn	Crown Heigh	40.67592		Entire home/	99	3	53	22/6/2019	4.44	1	21
21	7750	Huge 2 BR Upper I	17985	Sing	Manhattan	East Harlem	40.79685		Entire home/	190	7	0			2	249
22	7801	Sweet and Spacio	21207	Chaya	Brooklyn	Williamsburg	40.71842		Entire home/	299	3	9	28/12/2011	0.07	1	0
23	7801	Sweet and Spacio	21207	Chaya	Brooklyn	Williamsburg	40.71842		Entire home/	299	3	9	28/12/2011	0.07	1	0
24	8024	CBG CtyBGd Helps	22486	Lisel	Brooklyn	Park Slope	40.68069		Private room	130	2	130	1/7/2019	1.09	6	347
25	8025	CBG Helps Haiti R	22486	Lisel	Brooklyn	Park Slope	40.67989	-73.97798	Private room	80	1	39	1/1/2019	0.37	6	364
26	8110	CBG Helps Haiti Ru	22486	Lisel	Brooklyn	Park Slope	40.68001	-73.97865	Private room	110	2	71	2/7/2019	0.61	6	304
27	8490	MAISON DES SIRE	25183	Nathalie	Brooklyn	Bedford-Stuy	40.68371	-73.94028	Entire home/	120	2	88	19/6/2019	0.73	2	233
28	8505	Sunny Bedroom Ac	25326	Gregory	Brooklyn	Windsor Ter	40.65599	-73.97519	Private room	60	1	19	23/6/2019	1.37	2	85
29	8700	Magnifique Suite a	26394	Claude & Sop	Manhattan	Inwood	40.86754	-73.92639	Private room	80	4	0			1	0
30	9357	Midtown Pied-a-te	30193	Tommi	Manhattan	Hell's Kitche	40.76715	-73.98533	Entire home/	150	10	58	13/8/2017	0.49	1	75

Data classification

Classification

- การจำแนกกลุ่มข้อมูล เป็นการนำข้อมูลที่มีในอดีตมาสอนระบบเพื่อให้เรียนรู้รูปแบบที่เกิดขึ้นในข้อมูลแล้วสร้างเป็นสมการหรือโมเดล (model) ขึ้นเพื่อหาคำตอบให้กับข้อมูลใหม่
- ข้อมูลในอดีตที่นำมาใช้ในการเรียนรู้นี้จะเรียกว่า ข้อมูลสอนระบบ (Training data)
- โมเดลที่ได้ไปใช้ในการจำแนกข้อมูลชุดใหม่ ที่เรียกว่า ข้อมูลการทดสอบ (Testing Data) เพื่อประเมินประสิทธิภาพของโมเดล

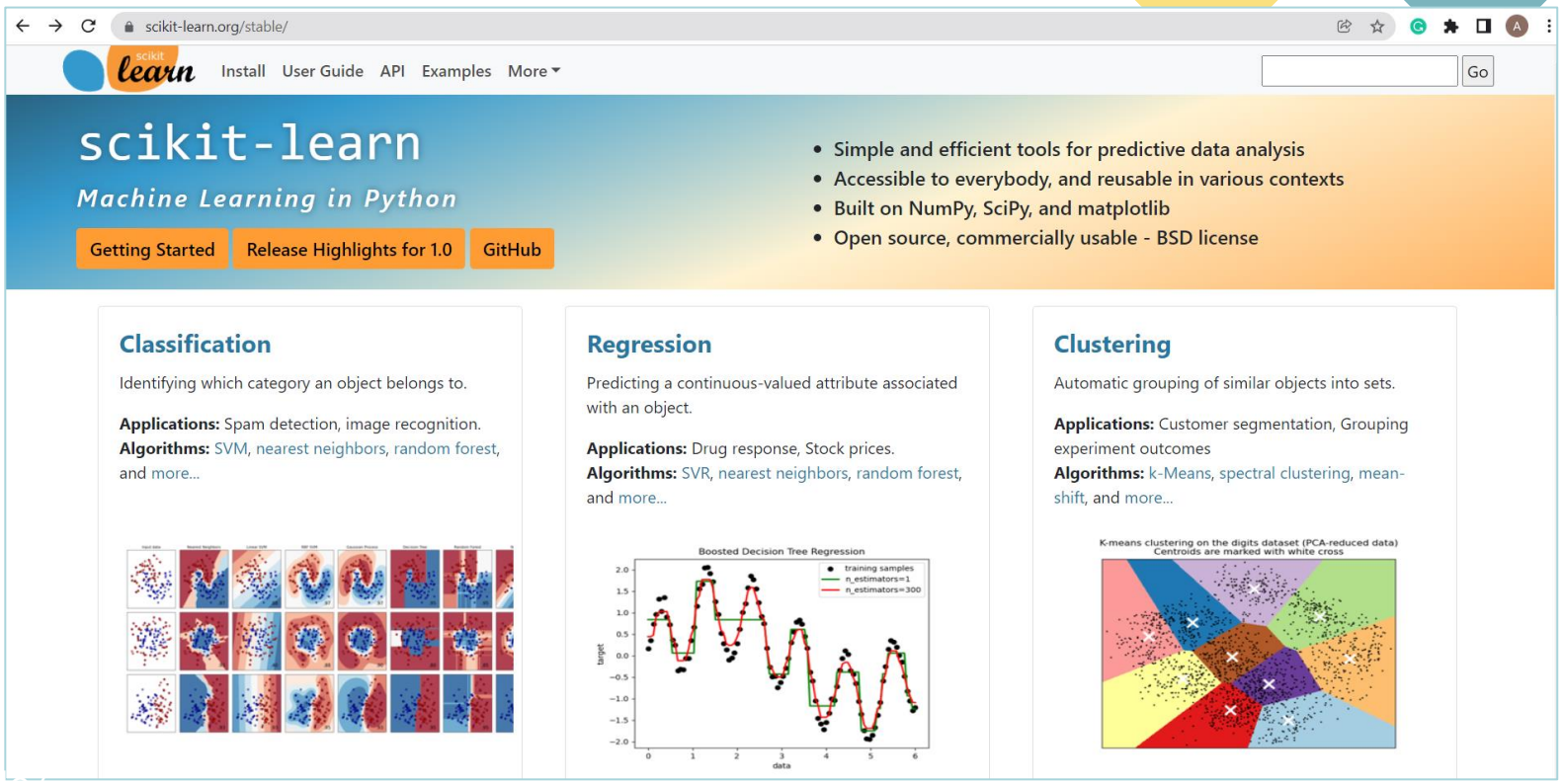
Classification Techniques

- Decision Tree
- Naïve Bays
- Random Forest
- K-Nearest Neighbor
- Ensemble Classification
- Support Vector Machine
- Neural Network
- Deep Learning

Scikit-learn

https://scikit-learn.org/stable

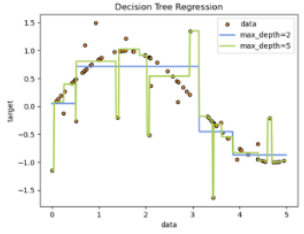
- Scikit-learn is a free software machine learning library for the Python programming language.



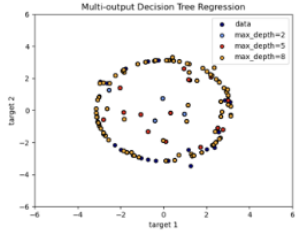
Decision Tree

Decision Tree

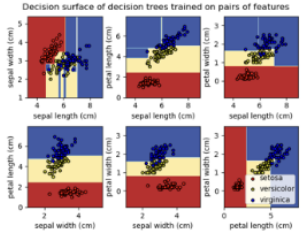
Examples concerning the `sklearn.tree` module.



Decision Tree Regression

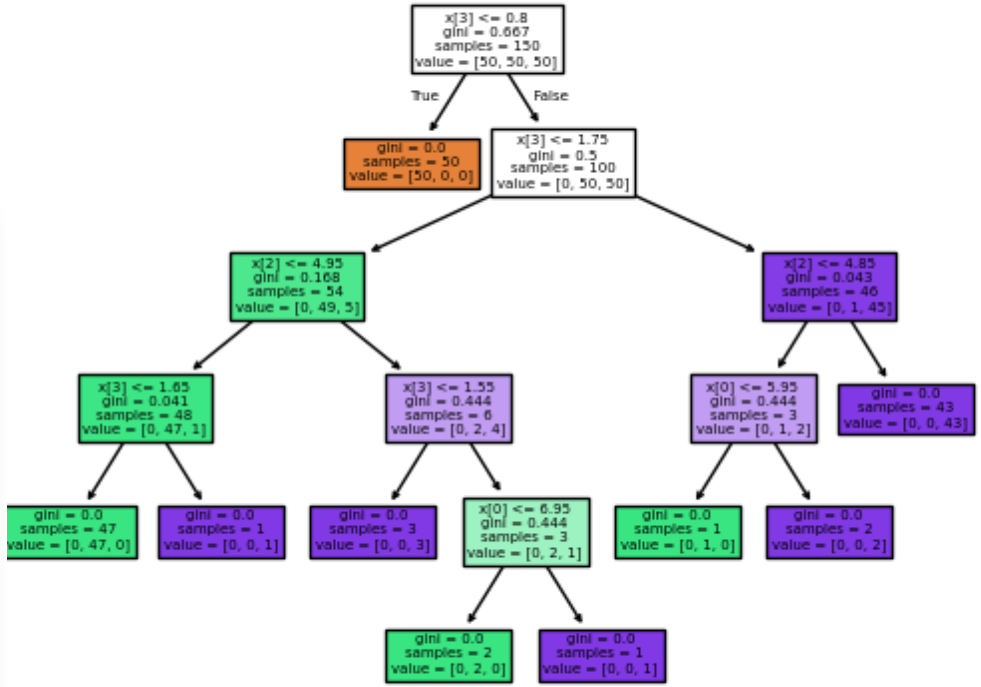


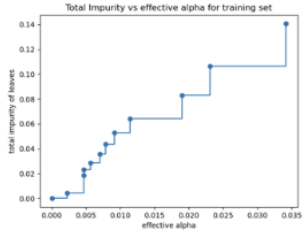
Multi-output Decision Tree Regression



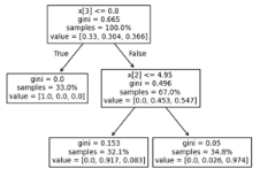
Plot the decision surface of decision trees trained on the iris dataset

Decision tree trained on all the iris features





Post pruning decision trees with cost complexity pruning



Understanding the decision tree structure

https://scikit-learn.org/1.5/auto_examples/tree/index.html

https://scikit-learn.org/1.5/modules/tree.html

Decision Tree – read the dataset

- import the modules, and read the dataset with pandas:

```
import pandas
from sklearn import tree
import pydotplus
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import matplotlib.image as pltimg

df = pandas.read_csv("shows.csv")

print(df)
```

A dataset: the person who goes to a comedy show or not.

	Age	Experience	Rank	Nationality	Go
0	36	10	9	UK	NO
1	42	12	4	USA	NO
2	23	4	6	N	NO
3	52	4	4	USA	NO
4	43	21	8	USA	YES
5	44	14	5	UK	NO
6	66	3	7	N	YES
7	35	14	9	UK	YES
8	52	13	7	N	YES
9	35	5	9	N	YES
10	24	3	5	USA	NO
11	18	3	7	UK	YES
12	45	9	9	UK	YES

Decision Tree – convert the values

- convert the non numerical columns into numerical values.

```
import pandas
from sklearn import tree
import pydotplus
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import matplotlib.image as pltimg

df = pandas.read_csv("shows.csv")

d = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go'] = df['Go'].map(d)

print(df)
```

	Age	Experience	Rank	Nationality	Go
0	36	10	9	0	0
1	42	12	4	1	0
2	23	4	6	2	0
3	52	4	4	1	0
4	43	21	8	1	1
5	44	14	5	0	0
6	66	3	7	2	1
7	35	14	9	0	1
8	52	13	7	2	1
9	35	5	9	2	1
10	24	3	5	1	0
11	18	3	7	0	1
12	45	9	9	0	1

Decision Tree – set the feature and target column

- separate the feature columns from the target column

```
…

df = pandas.read_csv("shows.csv")

d = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go'] = df['Go'].map(d)

features = ['Age', 'Experience', 'Rank', 'Nationality']
X = df[features]
y = df['Go']

print(X)
print(y)
```

```
    Age  Experience  Rank  Nationality
0    36          10     9            0
1    42          12     4            1
2    23           4     6            2
3    52           4     4            1
4    43          21     8            1
5    44          14     5            0
6    66           3     7            2
7    35          14     9            0
8    52          13     7            2
9    35           5     9            2
10   24           3     5            1
11   18           3     7            0
12   45           9     9            0

0     0
1     0
2     0
3     0
4     1
5     0
6     1
7     1
8     1
9     1
10    0
11    1
12    1
Name: Go, dtype: int64
```

Decision Tree – Create a Decision Tree

- Create a Decision Tree, save it as an image, and show the image

```
...
features = ['Age', 'Experience', 'Rank', 'Nationality']
X = df[features]
y = df['Go']

dtree = DecisionTreeClassifier()
dtree = dtree.fit(X, y)

class_names = ['NO', 'YES']
data = tree.export_graphviz(dtree, out_file=None,
       feature_names=features, class_names=class_names)

graph = pydotplus.graph_from_dot_data(data)
graph.write_png('mydecisiontree.png')

img = pltimg.imread('mydecisiontree.png')
imgplot = plt.imshow(img)
plt.show()
```

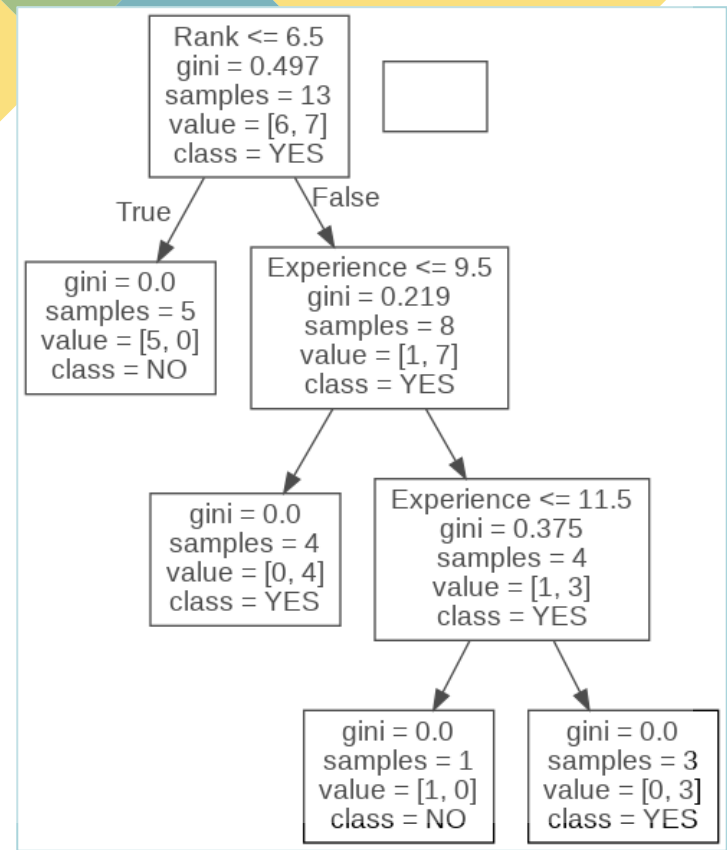
กำหนดให้ใช้อัลกอริทึม Decision tree ในการเรียนรู้

เรียนรู้จากข้อมูลค่าฟีเจอร์ X และค่าเฉลย y ของตัวอย่างต่างๆ

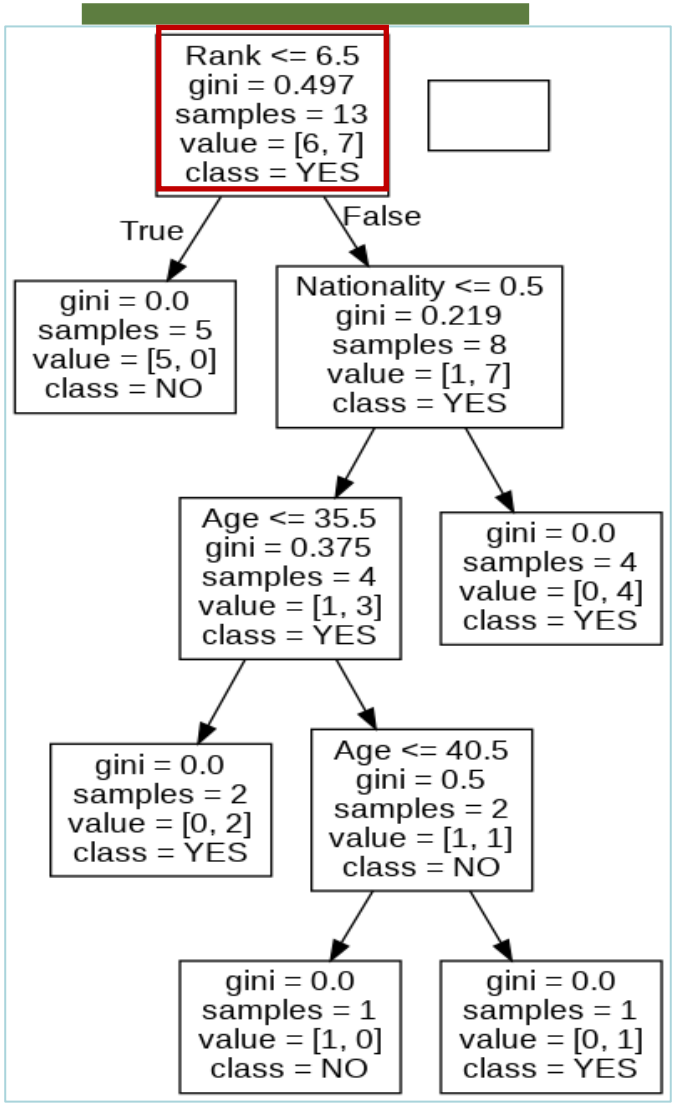
graphviz จะอ่านข้อมูล model แล้วนำมาสร้างเป็นข้อมูลไดอะแกรม หรือกราฟ

pydotplus จะอ่านข้อมูลไดอะแกรมจาก Graphviz แล้วแสดงเป็นแผนภาพต้นไม้แบบกราฟิก โดยบันทึกเก็บไว้เป็นไฟล์ภาพ .png

pltimg แสดงรูปภาพจากไฟล์ .png



Decision Tree – Result Explained



- **Rank <= 6.5** means that every comedian with a rank of 6.5 or lower will follow the True arrow (to the left), and the rest will follow the False arrow (to the right).
- **gini = 0.497** refers to the quality of the split, and is always a number between 0.0 and 0.5, where 0.0 would mean all of the samples got the same result, and 0.5 would mean that the split is done exactly in the middle.
- **samples = 13** means that there are 13 comedians left at this point in the decision, which is all of them since this is the first step.
- **value = [6, 7]** means that of these 13 comedians, 6 will get a "NO", and 7 will get a "GO".

Decision Tree – Display Tree

- Display tree with export_text library

```
…

features = ['Age', 'Experience', 'Rank', 'Nationality']
X = df[features]
y = df['Go']

dtree = DecisionTreeClassifier()
dtree = dtree.fit(X, y)
```

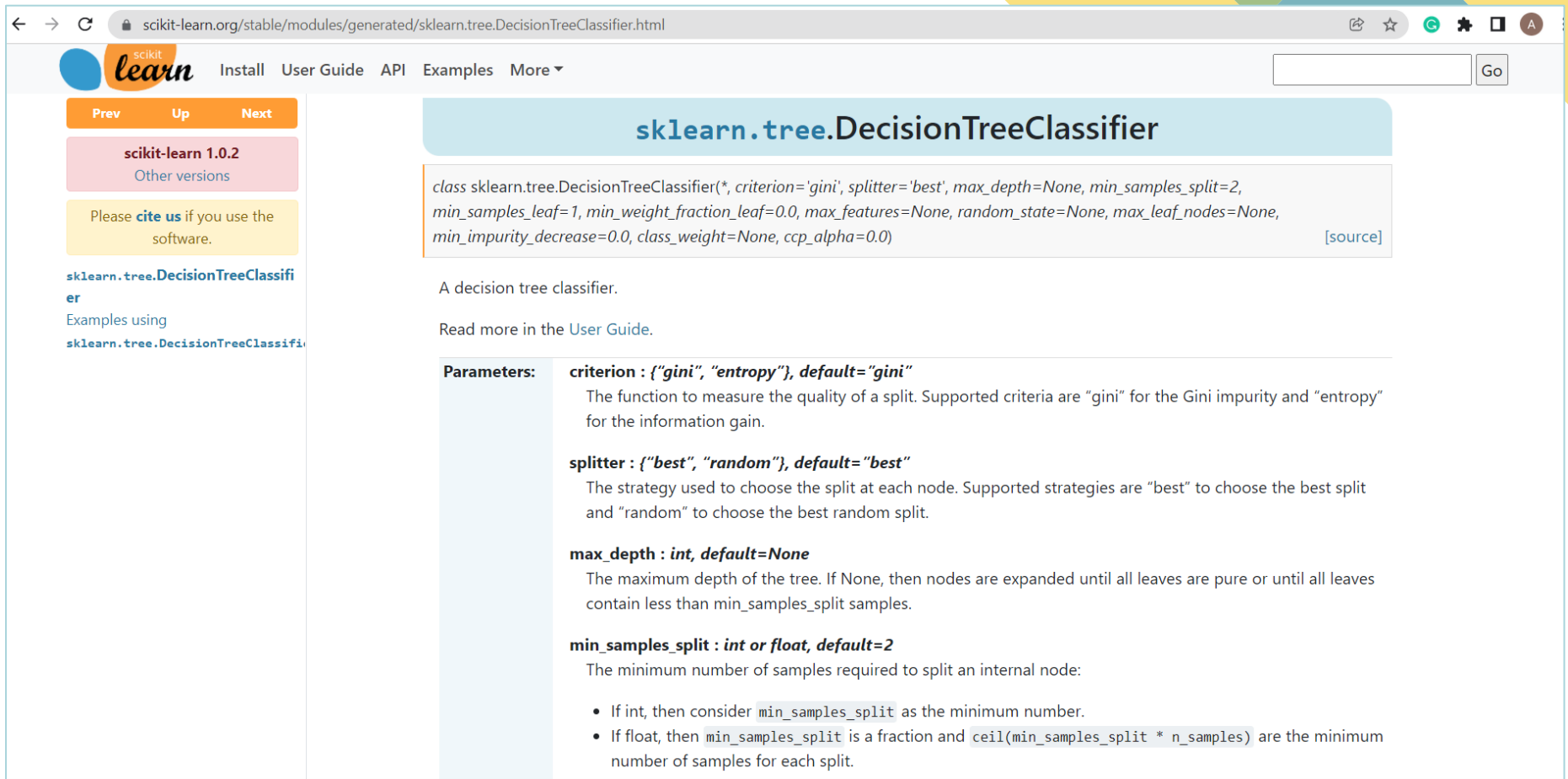
```
from sklearn.tree import export_text
r = export_text(dtree, feature_names=features)
print(r)
```

```
|--- Rank <= 6.50
|   |--- class: 0
|--- Rank >  6.50
|   |--- Nationality <= 0.50
|   |   |--- Age <= 35.50
|   |   |   |--- class: 1
|   |   |--- Age >  35.50
|   |   |   |--- Experience <= 9.50
|   |   |   |   |--- class: 1
|   |   |   |--- Experience >  9.50
|   |   |   |   |--- class: 0
|   |--- Nationality >  0.50
|   |   |--- class: 1
```

Decision Tree

- **sklearn.tree.DecisionTreeClassifier**

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html



DecisionTreeClassifier

class sklearn.tree.DecisionTreeClassifier(*, *criterion*='gini', *splitter*='best', *max_depth*=None, *min_samples_split*=2, *min_samples_leaf*=1, *min_weight_fraction_leaf*=0.0, *max_features*=None, *random_state*=None, *max_leaf_nodes*=None, *min_impurity_decrease*=0.0, *class_weight*=None, *ccp_alpha*=0.0, *monotonic_cst*=None)

Decision Tree

- sklearn.tree.DecisionTreeClassifier
- Method

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

apply(X[, check_input])	Return the index of the leaf that each sample is predicted as.
cost_complexity_pruning_path(X, y[, ...])	Compute the pruning path during Minimal Cost-Complexity Pruning.
decision_path(X[, check_input])	Return the decision path in the tree.
fit(X, y[, sample_weight, check_input, ...])	Build a decision tree classifier from the training set (X, y).
get_depth()	Return the depth of the decision tree.
get_n_leaves()	Return the number of leaves of the decision tree.
get_params([deep])	Get parameters for this estimator.
predict(X[, check_input])	Predict class or regression value for X.
predict_log_proba(X)	Predict class log-probabilities of the input samples X.
predict_proba(X[, check_input])	Predict class probabilities of the input samples X.
score(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_params(**params)	Set the parameters of this estimator.

Decision Tree – Predict Values

```
…

features = ['Age', 'Experience', 'Rank', 'Nationality']
X = df[features]
y = df['Go']

dtree = DecisionTreeClassifier()
dtree = dtree.fit(X, y)

print(dtree.predict([[40, 10, 7, 1]]))
print(dtree.predict([[40, 10, 6, 1]]))

print("[1] means 'GO'")
print("[0] means 'NO'")
```

```
[1]
[0]
[1] means 'GO'
[0] means 'NO'
```

Decision Tree – Model Evaluation

```
…
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
…
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=11)

dtree = DecisionTreeClassifier()
dtree = dtree.fit(X_train, y_train)

y_predict = dtree.predict(X_test)
print(X_test)
print(y_predict)

acc = accuracy_score(y_test, y_predict)
print("accuracy score: ", acc)
cm = confusion_matrix(y_test, y_predict)
print(cm)
print(classification_report(y_test, y_predict, target_names=['NO', 'YES']))
```

```
   Age  Experience  Rank  Nationality
6   66           3     7            2
3   52           4     4            1
8   52          13     7            2
```
X_test

```
[1 0 1]
```
y_predict

```
accuracy score:  1.0
```
accuracy

```
[[1 0]
 [0 2]]
```
confusion matrix

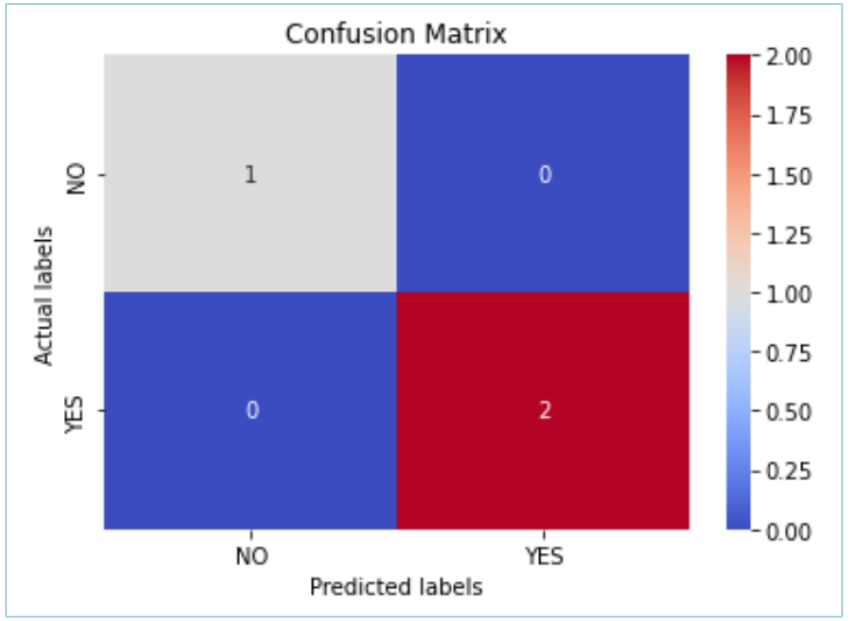
classification report

	precision	recall	f1-score	support
NO	1.00	1.00	1.00	1
YES	1.00	1.00	1.00	2
accuracy			1.00	3
macro avg	1.00	1.00	1.00	3
weighted avg	1.00	1.00	1.00	3

Decision Tree – Seaborn Heat Map

https://seaborn.pydata.org/generated/seaborn.heatmap.html

```
…
import matplotlib.pyplot as plt
import seaborn as sns
…
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=11)
dtree = DecisionTreeClassifier()
dtree = dtree.fit(X_train, y_train)
y_predict = dtree.predict(X_test)
cm = confusion_matrix(y_test, y_predict)

ax = plt.subplot()
sns.heatmap(cm, annot=True, ax=ax, cmap="coolwarm")

ax.set_title("Confusion Matrix")
ax.set_xlabel("Predicted labels")
ax.set_ylabel("Actual labels")
ax.xaxis.set_ticklabels(['NO', 'YES'])
ax.yaxis.set_ticklabels(['NO', 'YES'])
plt.show()
```



Decision Tree – k-fold cross validation

```
import pandas
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

df = pandas.read_csv("shows.csv")

d = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go'] = df['Go'].map(d)

features = ['Age', 'Experience', 'Rank', 'Nationality']
X = df[features]
y = df['Go']

dtree = DecisionTreeClassifier()
cvs = cross_val_score(dtree, X, y, cv=5)
print('cross val scores', cvs.round(3))
print('mean = ', cvs.mean().round(4) * 100)
```

https://seaborn.pydata.org/generated/seaborn.heatmap.html

```
cross val scores [0.667 0.667 1.    1.    0.5  ]
mean =  76.67
```

Naïve Bays

Naïve Bays (scikit-learn)

อัลกอริธึม	ประเภทข้อมูล/ลักษณะงาน	ตัวอย่างการใช้งาน
BernoulliNB	Binary (0/1), การมีอยู่หรือไม่มีอยู่ของฟีเจอร์	Spam Detection, Sentiment Analysis
CategoricalNB	ข้อมูลหมวดหมู่ (Categorical)	Customer Segmentation, Market Analysis
ComplementNB	ข้อมูลที่ไม่สมดุล (Imbalanced Data)	Fraud Detection, Rare Event Prediction
GaussianNB	Continuous Data (ตัวเลขต่อเนื่อง)	Medical Diagnosis, Behavioral Analysis
MultinomialNB	ข้อมูลที่แสดงจำนวน/ความถี่	Document Classification, Text Categorization

https://scikit-learn.org/stable/api/sklearn.naive_bayes.html

Naïve Bays

***class* sklearn.naive_bayes.GaussianNB(*, *priors*=*None*, *var_smoothing*=*1e-09*)**

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB

Naïve Bays - Cross Validation

```
import seaborn as sns
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB

df = sns.load_dataset('iris’)
df.head()

X = df.drop('species', axis=1)
y = df.species

model = GaussianNB()
cvs = cross_val_score(model, X, y, cv=10)
print('cross val scores', cvs.round(3))
print('mean = ', cvs.mean().round(4) * 100)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
cross val scores [0.933 0.933 1.    0.933 0.933 0.933 0.867 1.    1.    1.   ]
mean (%) = 95.333
```

https://seaborn.pydata.org/generated/seaborn.load_dataset.html

Scikit-learn datasets https://scikit-learn.org/stable/datasets.html

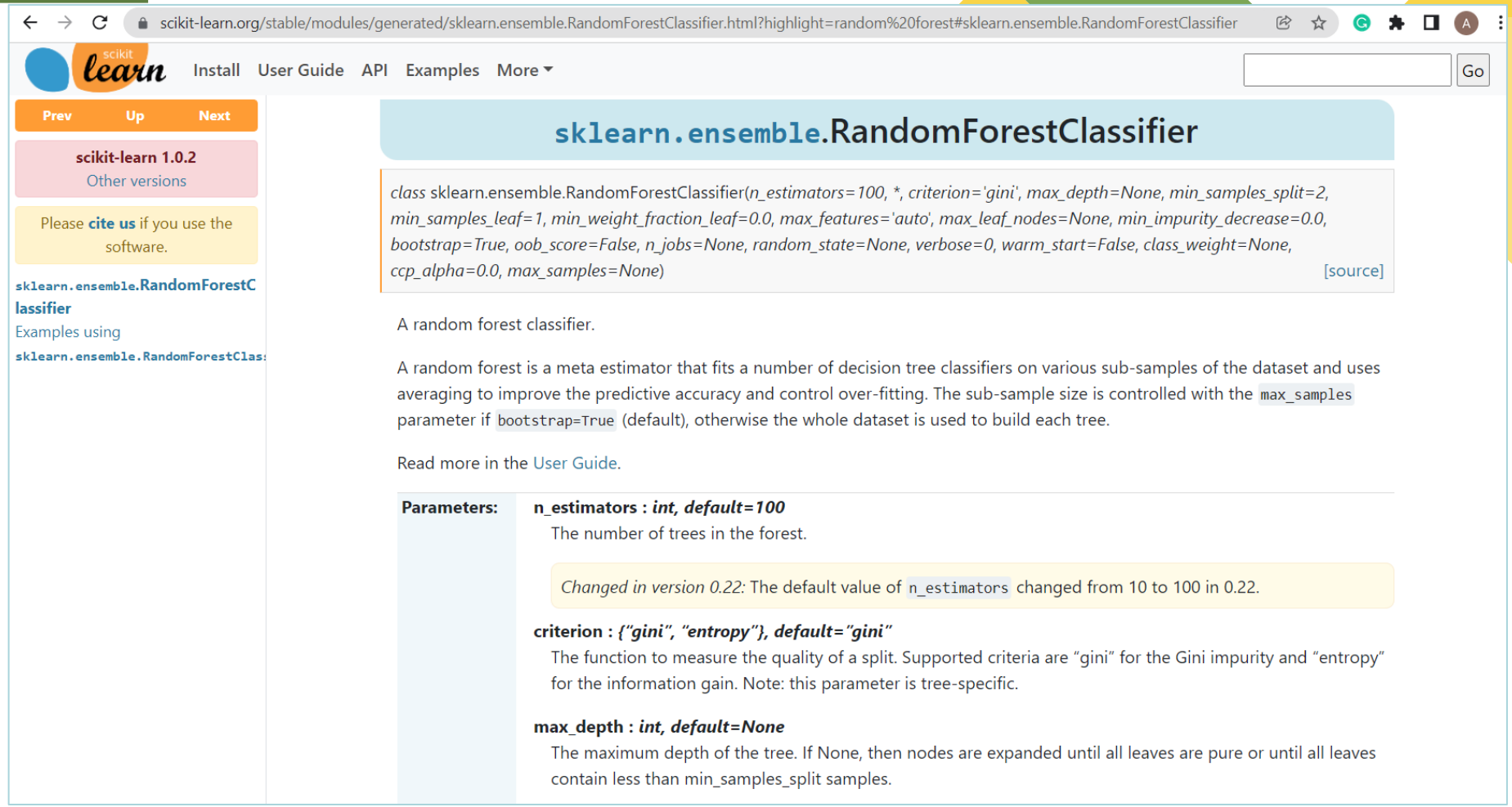
Naïve Bays - Train/Test Split

```
Accuracy Score:  0.9555555555555556
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        17
  versicolor       0.93      0.93      0.93        14
   virginica       0.93      0.93      0.93        14

    accuracy                           0.96        45
   macro avg       0.95      0.95      0.95        45
weighted avg       0.96      0.96      0.96        45

[[17  0  0]
 [ 0 13  1]
 [ 0  1 13]]
```

```
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB

df = sns.load_dataset('iris')
X = df.drop('species', axis=1)
y = df.species

X_train, X_test,y_train,y_test = train_test_split(X, y, test_size=0.3)
model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print('Accuracy Score: ', accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

Random Forest

Random Forest



https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%20forest#sklearn.ensemble.RandomForestClassifier

RandomForestClassifier

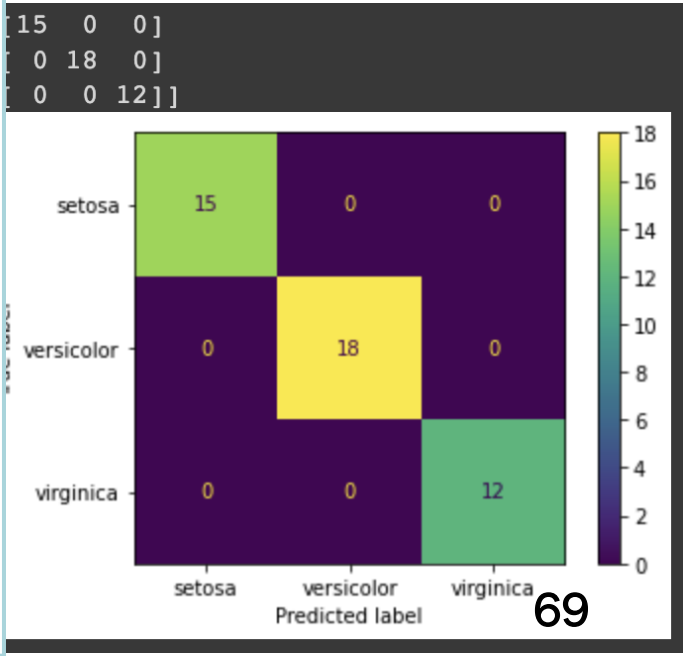
class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None, monotonic_cst=None*)

Random Forest - Cross Validation

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import classification_report

df = sns.load_dataset('iris')
X = df.drop('species', axis=1)
y = df.species

X_train, X_test,y_train,y_test = train_test_split(X, y, test_size=0.3)
model = RandomForestClassifier(n_estimators=80) #The number of trees
model.fit(X_train, y_train)

class_names = df.species.unique()
class_names
y_predict = model.predict(X_test)
print('Score -> {:.4f}' . format(model.score(X_test, y_test)))
print(classification_report(y_test, y_predict, target_names=class_names))
confusion_matrix = confusion_matrix(y_test, y_predict)
print(confusion_matrix)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix,
                display_labels = class_names)
cm_display.plot()
plt.show()
```

```
Score -> 1.0000
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       1.00      1.00      1.00        18
   virginica       1.00      1.00      1.00        12

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```
[[15  0  0]
 [ 0 18  0]
 [ 0  0 12]]
```



Exercise

- เขียนโปรแกรมเพื่อจำแนกผู้ป่วยเบาหวาน โดยประยุกต์เทคนิค Classification ของ Python มา 2 เทคนิค และเปรียบเทียบผลลัพธ์ที่ได้

Row No.	PatientID	HasDiabetes	NumberOfPr...	Glucose	BloodPress...	SkinThickne...	Insulin	BMI	DiabetesPe...	Age
1	1	1	6	148	72	35	0	33.600	0.627	50
2	2	0	1	85	66	29	0	26.600	0.351	31
3	3	1	8	183	64	0	0	23.300	0.672	32
4	4	0	1	89	66	23	94	28.100	0.167	21
5	5	1	0	137	40	35	168	43.100	2.288	33
6	6	0	5	116	74	0	0	25.600	0.201	30
7	7	1	3	78	50	32	88	31	0.248	26
8	8	0	10	115	0	0	0	35.300	0.134	29
9	9	1	2	197	70	45	543	30.500	0.158	53
10	10	1	8	125	96	0	0	0	0.232	54
11	11	0	4	110	92	0	0	37.600	0.191	30
12	12	1	10	168	74	0	0	38	0.537	34
13	13	0	10	139	80	0	0	27.100	1.441	57
14	14	1	1	189	60	23	846	30.100	0.398	59
15	15	1	5	166	72	19	175	25.800	0.587	51
16	16	1	7	100	0	0	0	30	0.484	32
17	17	1	0	118	84	47	230	45.800	0.551	31
18	18	1	7	107	74	0	0	29.600	0.254	31

ExampleSet (768 examples, 2 special attributes, 8 regular attributes)

- มีข้อมูลจำนวน 768 examples
- 8 regular attributes
  - NumberOfPregnant
  - Glucose
  - BloodPressure
  - SkinThickness
  - Insulin
  - BMI
  - DiabetesPedigree
  - Age
- 2 special attribute (Label)
  - ID: PatientID รหัสผู้ป่วย
  - Label: HasDiabetes เป็นหรือไม่เป็นเบาหวาน

https://www.kaggle.com/uciml/pima-indians-diabetes-database

Reference

- https://www.w3schools.com/

- **Code Program in the slide**
  - Pandas and Data Cleaning

https://colab.research.google.com/drive/1tQ_qeQu20arPSJ_6Eox-n5NcAwXCjXbI?usp=sharing

  - Decision Tree Classification

https://colab.research.google.com/drive/1YyInIS4YsStC-1nc7P1eSafhCe9kIvaA?usp=sharing

  - Classification Techniques

https://colab.research.google.com/drive/1IkyUBFUDb96SMtStI5hAjwLOQM2YosvA?usp=sharing

Q&A:

This Photo CC BY-SA