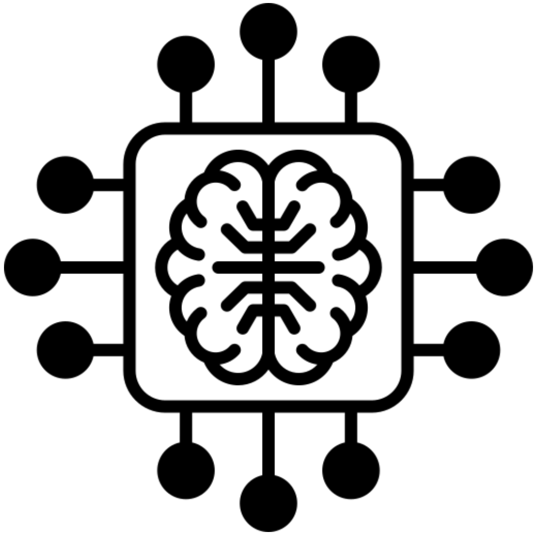
SBS4115 Fundamentals of AI & Data Analytics



AI in Action I

Lecturer: Ir Dr Kelvin K. W. Siu
email: kelvinsiu@thei.edu.hk

高科院
Thei

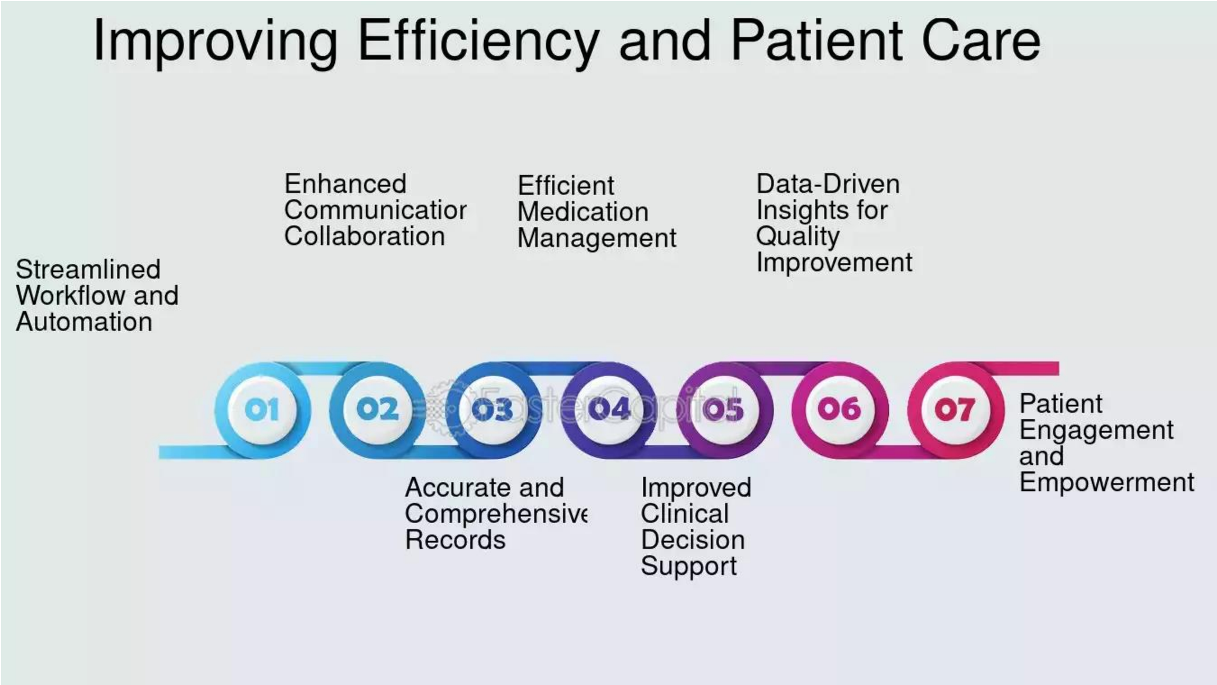
Department of Construction,
Environment and Engineering

Intended Learning Outcomes

- By the end of this lecture, you will be able to…
  - Applications of AI in healthcare industry
  - Introduce Python data analysis (Pandas)
  - Apply descriptive statistics and arithmetic in Pandas
  - Read data for analytics in Pandas
  - Prepare and clean data in Pandas.
  - Introduce computer vision

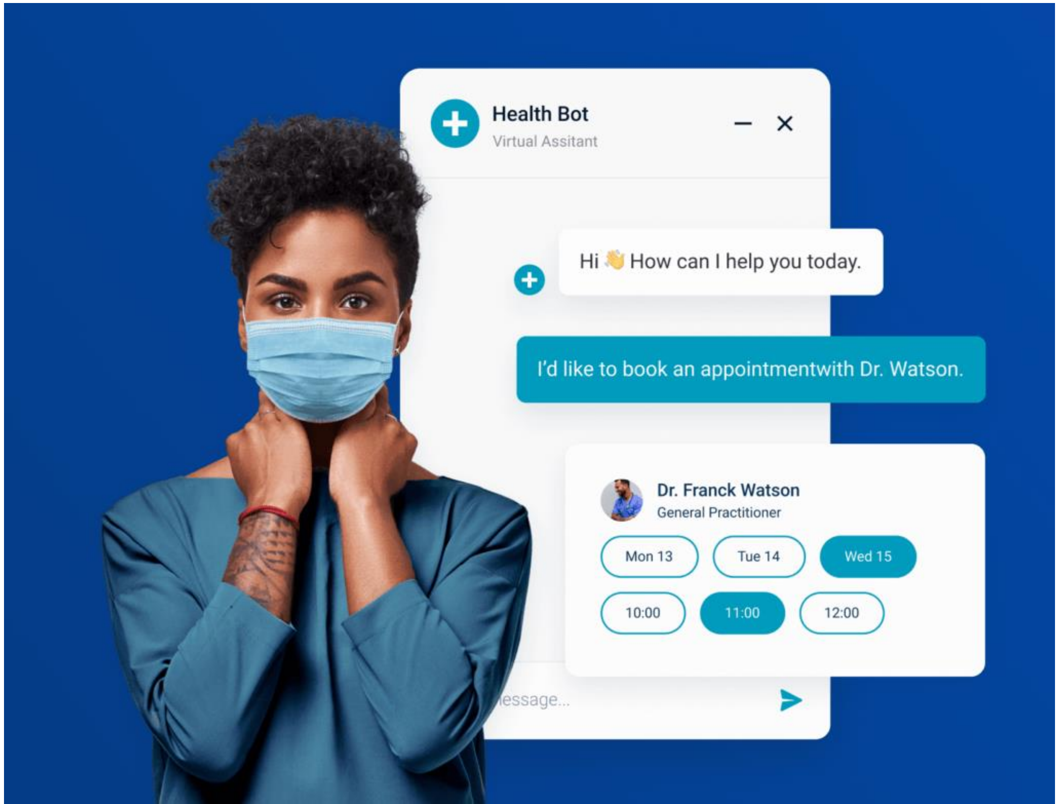
Application of AI - Healthcare

- Enhancing patient care, diagnostics, and administrative operations
- Enhancing operational efficiency and patient experiences


Improving Efficiency and Patient Care
Streamlined Workflow and Automation
Enhanced Communicatior Collaboration
Efficient Medication Management
Data-Driven Insights for Quality Improvement
01
02
03
04
05
06
07
FasterCapital
Patient Engagement and Empowerment
Accurate and Comprehensive Records
Improved Clinical Decision Support

AI in Front Office Automation

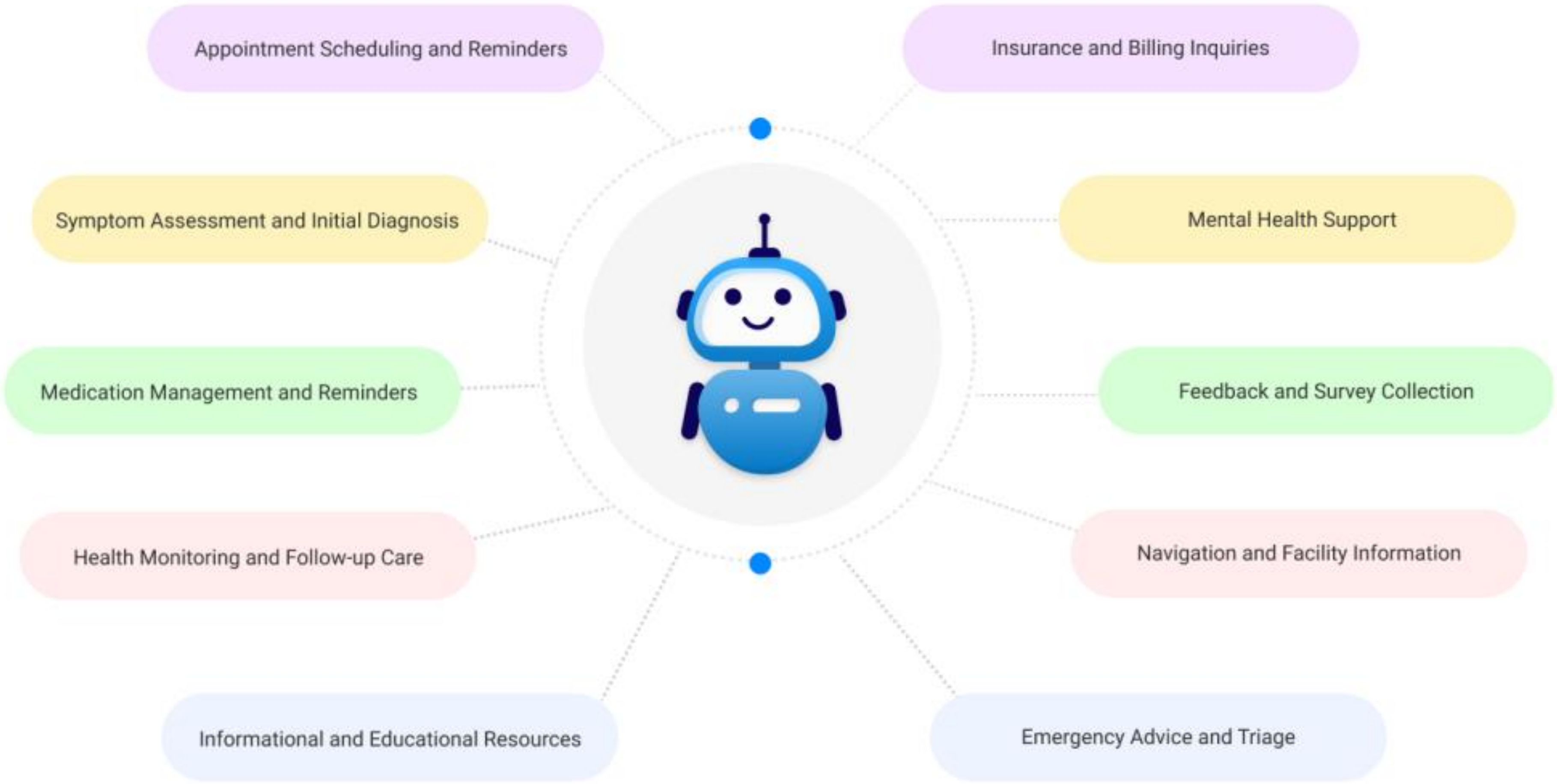
- AI-powered chatbots & virtual assistants
- Manage patient inquiries, schedule appointments, update EHRs
- Reduces errors and enhances patient interaction



Hi 👋 How can I help you today.

I'd like to book an appointmentwith Dr. Watson.

What are Chatbots Used for in Healthcare?



Appointment Scheduling and Reminders

Insurance and Billing Inquiries

Symptom Assessment and Initial Diagnosis

Mental Health Support

Medication Management and Reminders

Feedback and Survey Collection

Health Monitoring and Follow-up Care

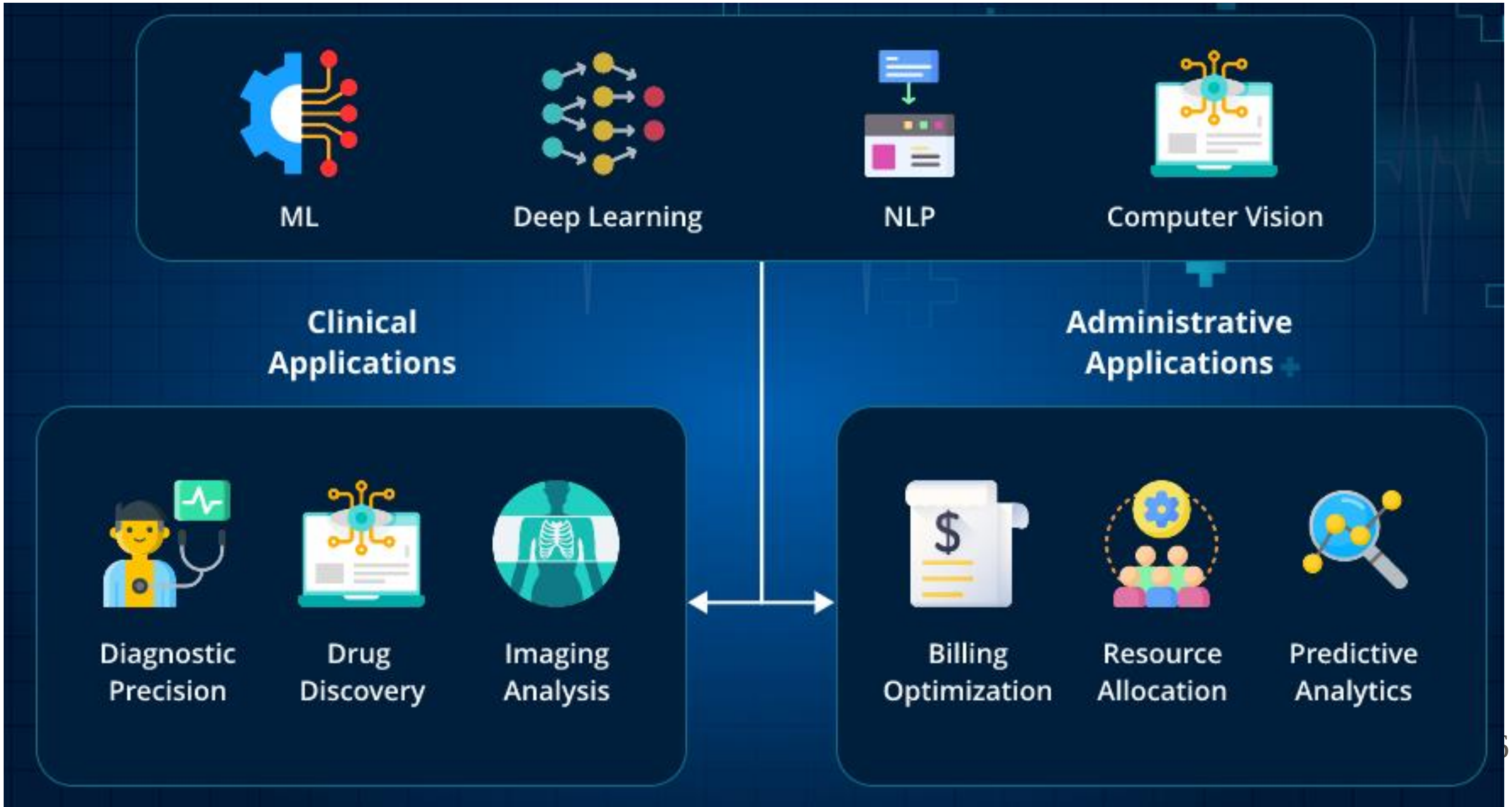
Navigation and Facility Information

Informational and Educational Resources

Emergency Advice and Triage

AI in Back Office Automation

- Automates medical coding, billing, and supply chain management
- Reduces coding errors, speeds up revenue cycles



AI in Predictive Analytics

- Forecasts patient volumes, optimizes resources, reduces unplanned admissions
- Example: Mount Sinai Health Systems predictive model for highrisk patients
- Early intervention improves outcomes

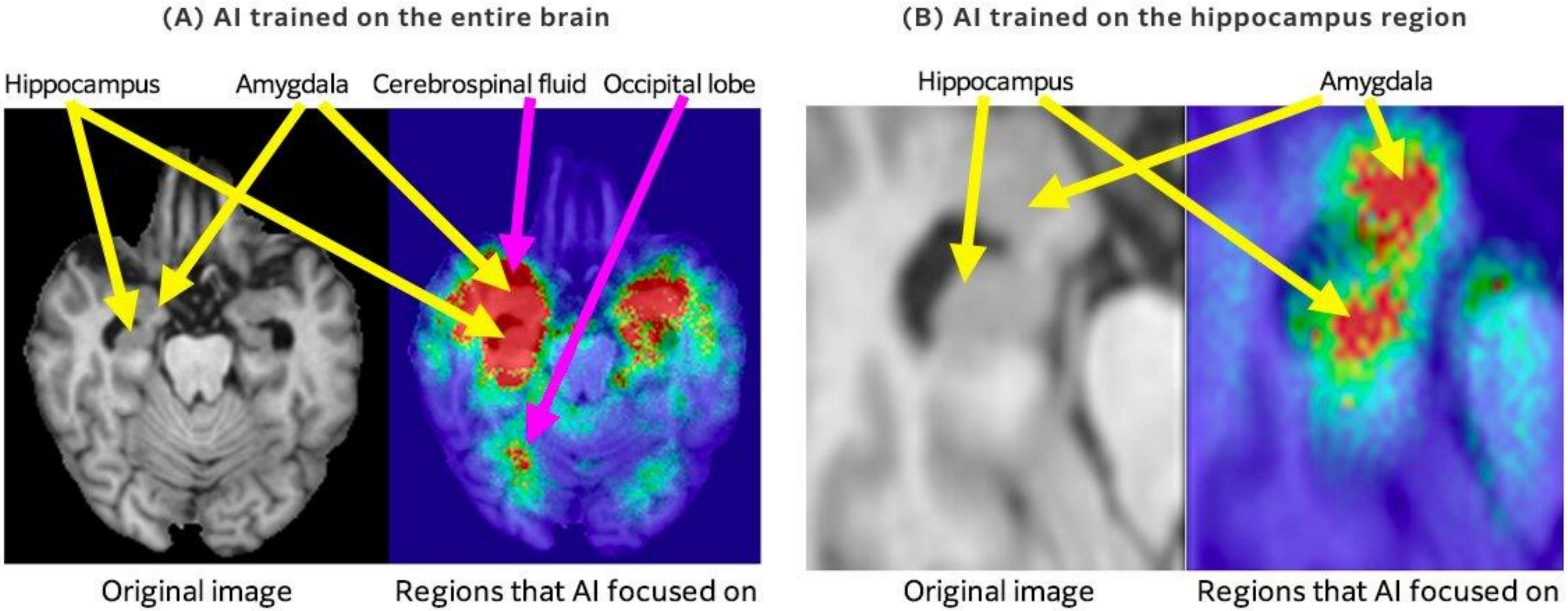
AI in Chronic Disease Management

- Virtual nurse assistants like Sensely guide patients with chronic conditions
- Continuous monitoring reduces hospital visits, improves adherence

AI's Role in Predictive Care

- Predicts disease progression and surgical complications
- Example: Sheba Medical Center’s AI for colorectal cancer surgery risks

[Figure 1] Detailed atrophy patterns that AI focused on predicting the progression to AD (the three-dimensional MRI images)

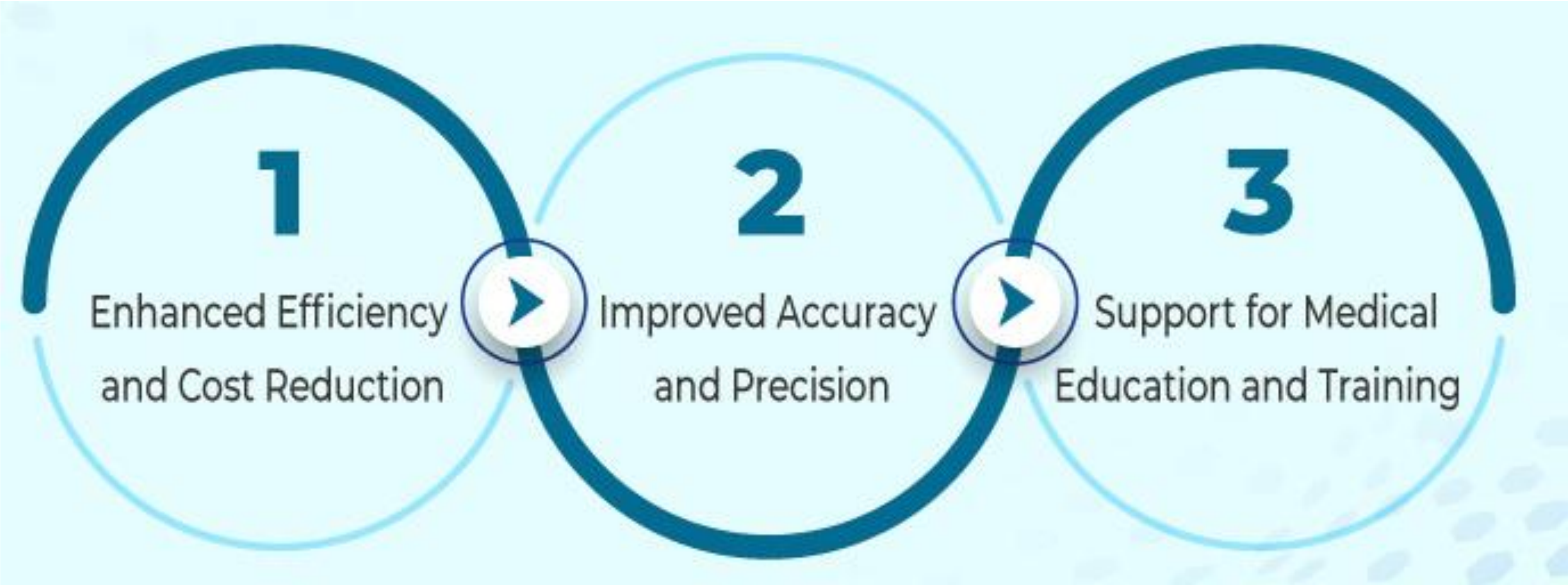


(A) AI trained on the entire brain

(B) AI trained on the hippocampus region

Future Directions for AI in Healthcare

- Enhances personalized care, operational efficiency, population health management
- Continued collaboration needed for ethical, effective AI integration


1
Enhanced Efficiency and Cost Reduction
2
Improved Accuracy and Precision
3
Support for Medical Education and Training

Descriptive Statistics and Arithmetic in Pandas

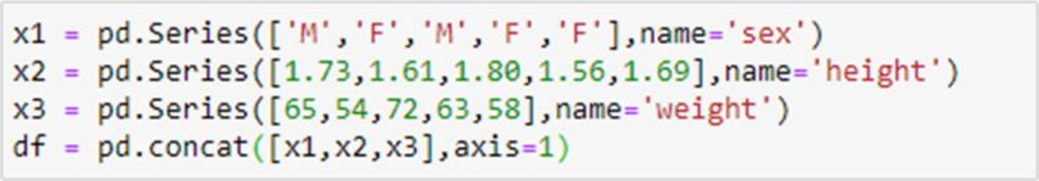
- **Series** is only a one-dimensional array-type of data structure, which is **only suitable for storing data set of a single variable**.
- However, in real-life the data file might contain multiple variables.

- For example, the health record of a class of students might contain their gender (string), height (float) and weight (float).
- Each of these three variables can stored as a Series, and we can combine these Series together into a **two-dimensional tabular form** called **DataFrame**.
- Each of the Series is regarded as one column in the DataFrame.
- In order to distinguish, we can also give names to these columns.

Descriptive Statistics and Arithmetic in Pandas

- In the example below, we first **define the three Series with values and names**.
- Then we **combine the Series together into a DataFrame** using the `concat` method.
- The syntax is:

```
df_name = pd.concat([Series1, Series2, ...], axis = 1)
```



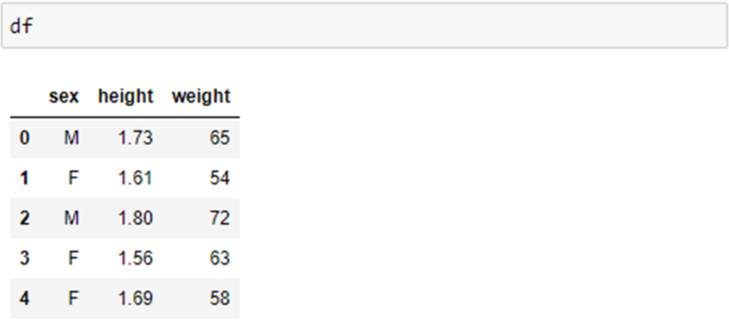
```
x1 = pd.Series(['M','F','M','F','F'],name='sex')
x2 = pd.Series([1.73,1.61,1.80,1.56,1.69],name='height')
x3 = pd.Series([65,54,72,63,58],name='weight')
df = pd.concat([x1,x2,x3],axis=1)
```

Descriptive Statistics and Arithmetic in Pandas

```
import pandas as pd

x1 = pd.Series(['M', 'F', 'M', 'F', 'F'], name='sex')
x2 = pd.Series([1.73, 1.61, 1.80, 1.56, 1.69], name='height')
x3 = pd.Series([65, 54, 72, 63, 58], name='weight')

df = pd.concat([x1, x2, x3], axis=1)
print(df)
```

Descriptive Statistics and Arithmetic in Pandas

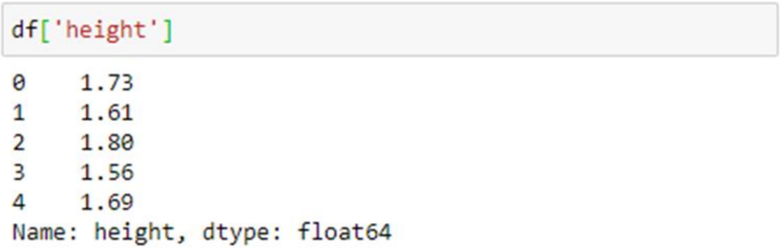
- The **DataFrame** is displayed in tabular form tidily as shown below.



```
df
```

	sex	height	weight
0	M	1.73	65
1	F	1.61	54
2	M	1.80	72
3	F	1.56	63
4	F	1.69	58

- Since DataFrame is a **two-dimensional data structure**, we can call out either a row, a column or a single entry from a DataFrame.
- To call a single column, directly use the column name:

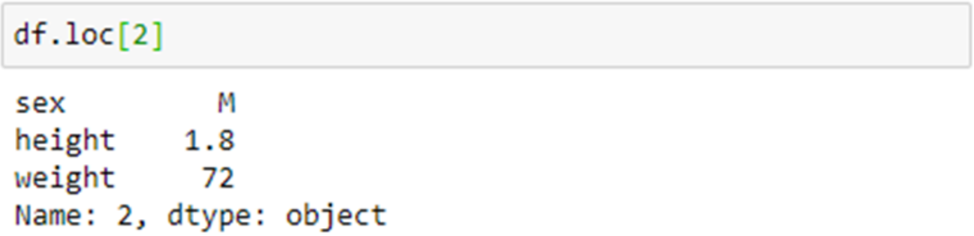


```
df['height']
```

```
0    1.73
1    1.61
2    1.80
3    1.56
4    1.69
Name: height, dtype: float64
```

Descriptive Statistics and Arithmetic in Pandas

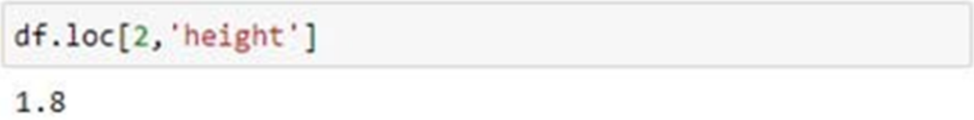
- To call a single row, apply the `loc` method and use the index of the row:



```
df.loc[2]
```

```
sex          M
height     1.8
weight      72
Name: 2, dtype: object
```

- To call a single entry, we can apply the `loc` method and include both row index and column name of the target entry.



```
df.loc[2,'height']
```

```
1.8
```

Descriptive Statistics and Arithmetic in Pandas

- For a DataFrame, we might want to **add new columns** based on some arithmetic of the existing columns.
- Recall that since a column of a DataFrame is a Series, it also supports **vectorized computation**.
- If we make arithmetic operations between two columns, the result is a Series with the same dimension.

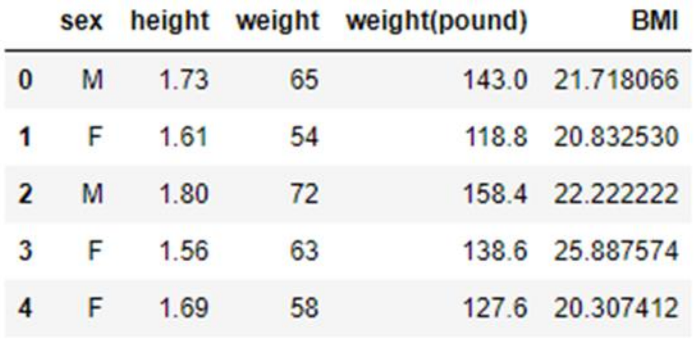
- We can create a new column in a DataFrame with such result.
- The syntax is:

```
DataFrame_name["new_col_name"] = Series_name
```

Descriptive Statistics and Arithmetic in Pandas

- For example, we would like to create a new column of weight in pounds, which equals to weight in kilograms multiplied by 2.2.
- We would like to create another column of BMI (body mass index) which is the weight in kg over square of height in m.
- Refer to the coding below:

```
df["weight(pound)"] = df["weight"] * 2.2
df["BMI"] = df["weight"] / df["height"] ** 2

print(df)
```



	sex	height	weight	weight(pound)	BMI
0	M	1.73	65	143.0	21.718066
1	F	1.61	54	118.8	20.832530
2	M	1.80	72	158.4	22.222222
3	F	1.56	63	138.6	25.887574
4	F	1.69	58	127.6	20.307412

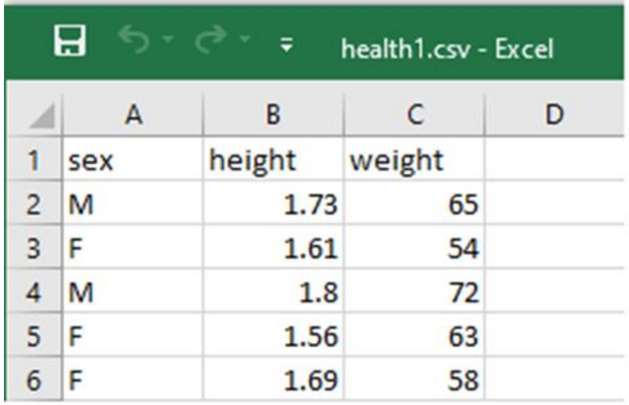
Data Loading

- In the previous section, the data no matter in Series or DataFrame form are input one-by-one on our own.
- In reality, this would be impossible due to the volume of big data.
- **Pandas provides various methods to read data from various file formats or sources into a DataFrame for analytics.**

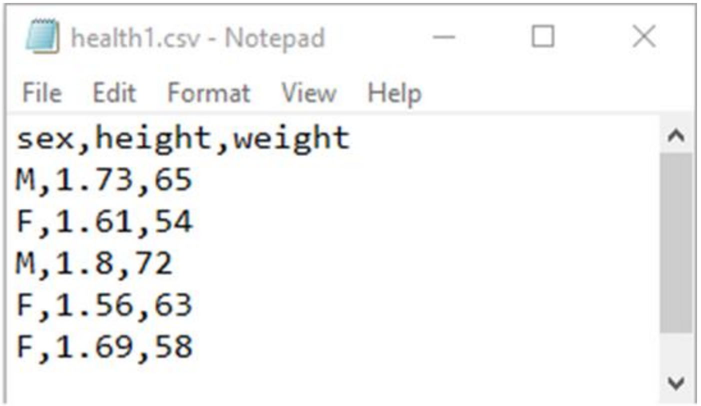
- One common type of data file is **comma-separated values (csv)** file.
- It is a delimited text file that uses a comma to separate values.
- Each line is regarded as a data record.
- However, the first line is usually used as column titles, indicating the meaning of the values stored in this column.

Data Loading

- When opened on Excel, the values are automatically arranged by rows and columns without showing the commas.
- When opened on Notepad, each record is stored on a line with its values separated by commas.



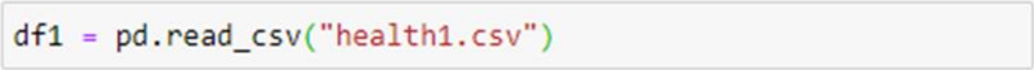
	A	B	C	D
1	sex	height	weight	
2	M	1.73	65	
3	F	1.61	54	
4	M	1.8	72	
5	F	1.56	63	
6	F	1.69	58	



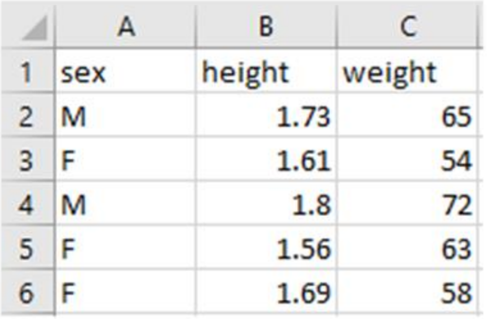
```
sex,height,weight
M,1.73,65
F,1.61,54
M,1.8,72
F,1.56,63
F,1.69,58
```

Data Loading

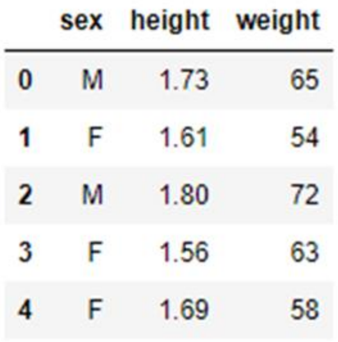
- As an example, if we read data from an excel file "health1.csv" which already preserves the first row as the header, providing information of each column.
- The header row will be converted to the column names of the DataFrame and not regarded as data values.
- The syntax is: `df_name = pd.read_csv("file_path")`



```
df1 = pd.read_csv("health1.csv")
```



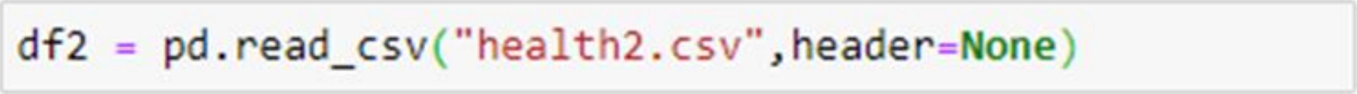
	A	B	C
1	sex	height	weight
2	M	1.73	65
3	F	1.61	54
4	M	1.8	72
5	F	1.56	63
6	F	1.69	58



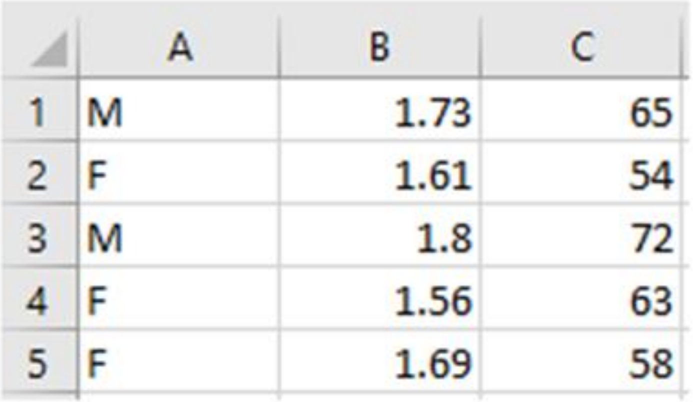
	sex	height	weight
0	M	1.73	65
1	F	1.61	54
2	M	1.80	72
3	F	1.56	63
4	F	1.69	58

Data Loading

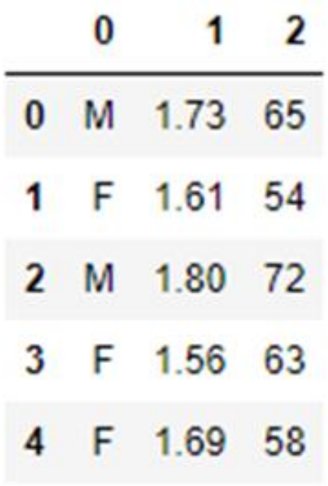
- If the data file does **not contain a header** as in "health2.csv", we need to specify by putting `header=None`.
- The column names in the DataFrame created will be by default 0, 1, 2,...



```
df2 = pd.read_csv("health2.csv",header=None)
```



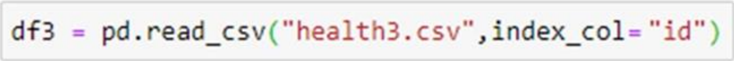
	A	B	C
1	M	1.73	65
2	F	1.61	54
3	M	1.8	72
4	F	1.56	63
5	F	1.69	58



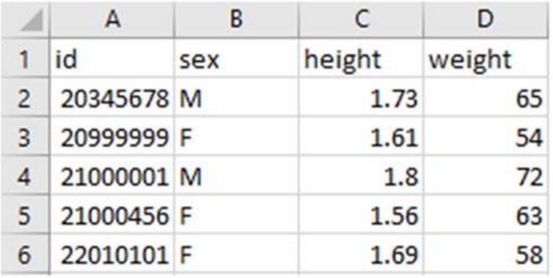
	0	1	2
0	M	1.73	65
1	F	1.61	54
2	M	1.80	72
3	F	1.56	63
4	F	1.69	58

Data Loading

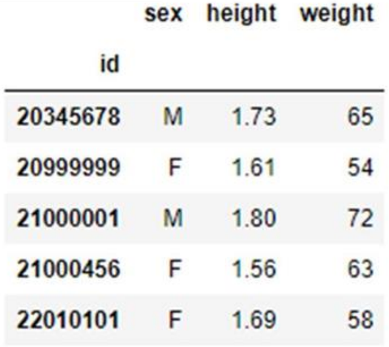
- By default, the DataFrame created from reading a data file will be **assigned with index** 0, 1, 2,... and so on.
- In some data file, one of the column might contain the index of this set of data.
- If you wish to **set a particular column from the data file to be the index column**, put the parameter `index_col` within the brackets of `read_csv`.
- Example: in "health3.csv", a column called "id" contains the student identity no.
- We can set it to be the index column as this value can uniquely distinguish different rows (students).



```
df3 = pd.read_csv("health3.csv",index_col="id")
```



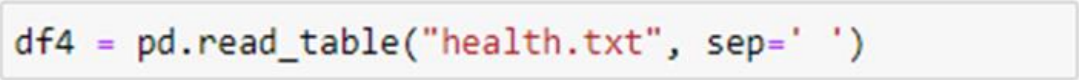
	A	B	C	D
1	id	sex	height	weight
2	20345678	M	1.73	65
3	20999999	F	1.61	54
4	21000001	M	1.8	72
5	21000456	F	1.56	63
6	22010101	F	1.69	58



	sex	height	weight
id			
20345678	M	1.73	65
20999999	F	1.61	54
21000001	M	1.80	72
21000456	F	1.56	63
22010101	F	1.69	58

Data Loading

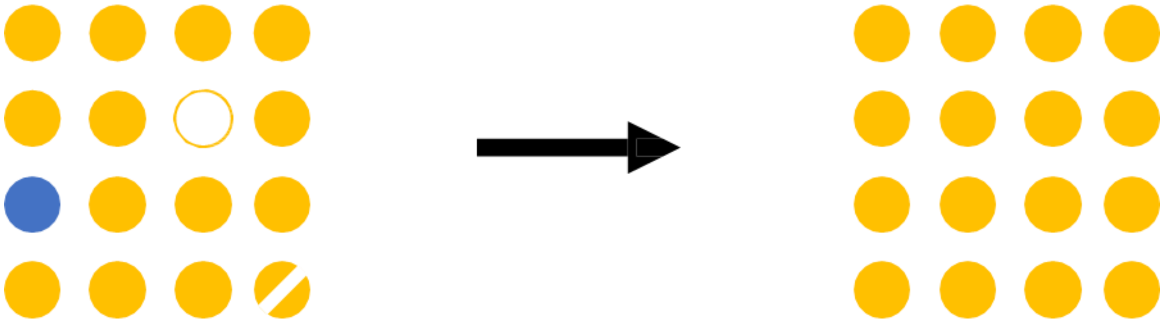
- In case the data file is **not a csv file**, we can still read it using `read_table` method.
- But we need to **specify the delimiting character**, which separate between values.
- In the example below, the file "health.txt" use **whitespace** as the delimiting character.
- We can read it by:



```
df4 = pd.read_table("health.txt", sep=' ')
```

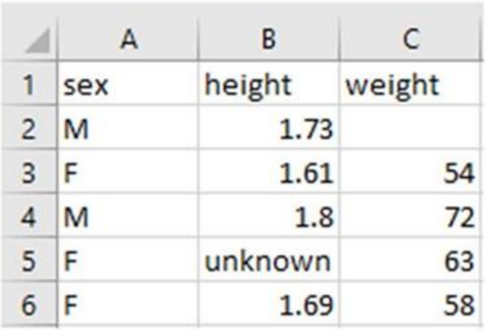
Data Preparation and Data Cleaning

- We have learnt data loading and simple data analytics.
- But in reality, you will find **a gap between these two steps**.
- Due to the process of data collection, the original data file might contain problematic entries and hence not ready for carrying out data analytics.
- To fill in this gap, we need a process called **data preparation**.
- In data preparation, the most important process is **data cleaning**.
- It is a process to fix or remove incorrect, corrupted or missing data.

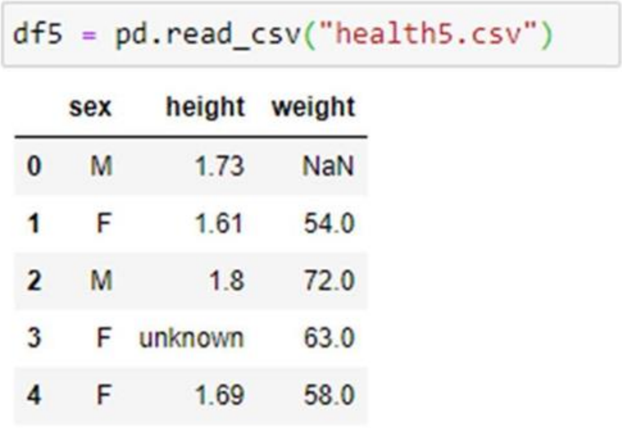


Data Preparation and Data Cleaning

- In the example below, two entries in the csv file are replaced by a word **"unknown" and an empty cell**.
- When this file is read as a DataFrame, they are not regarded as numerical values.
- The "unknown" is read as a string and not valid for statistical measures such as `mean()` or `sum()`.
- An error message will occur.



	A	B	C
1	sex	height	weight
2	M	1.73	
3	F	1.61	54
4	M	1.8	72
5	F	unknown	63
6	F	1.69	58



```
df5 = pd.read_csv("health5.csv")
```

```
   sex   height  weight
0    M     1.73     NaN
1    F     1.61    54.0
2    M      1.8    72.0
3    F  unknown    63.0
4    F     1.69    58.0
```

Data Preparation and Data Cleaning

```
import pandas as pd
df5 = pd.read_csv(r'C:\Users\User user\Desktop\health5.csv')
print(df5)
```

Try

```
df5['weight'].mean()
df5['height'].mean()
```

Data Preparation and Data Cleaning

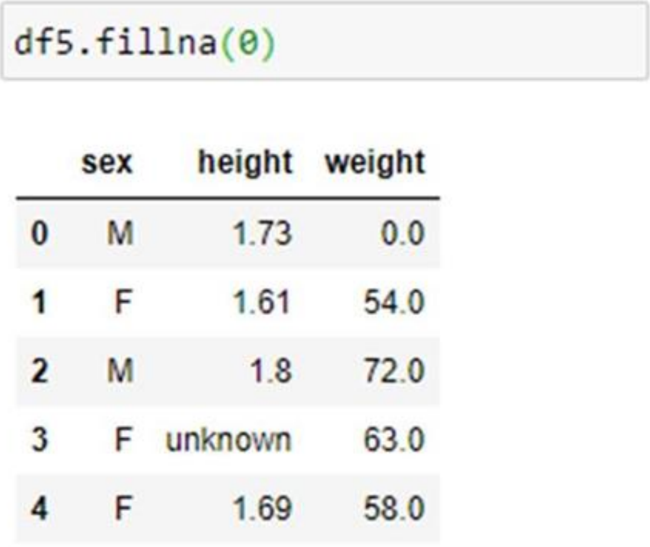
- On the other hand, the empty cell is read as NaN which means Not-a-Number.
- The statistical measures can still be evaluated but this entry will be ignored.
- Oppositely, if you want to empty the value in a cell, you can enter `None`.

- To remedy these problematic entries, we might not want to edit it one-by-one (as there might be thousands of such in a set of big-data!).

- Instead, we can use some existing methods in Pandas.
- Applying `fillna(0)` to a Series, DataFrame or a particular column of a DataFrame replaces all the NaN by 0.
- This 0 can be changed to other values.

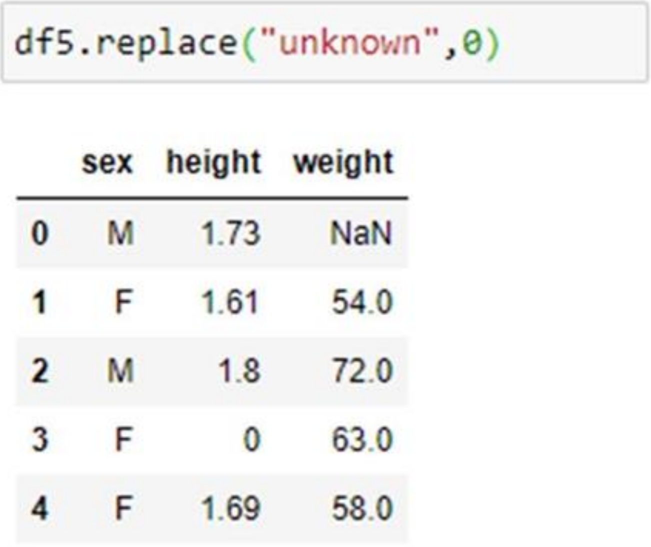
Data Preparation and Data Cleaning

- As a more general way, the `replace()` method allows you to **replace any old value** in the Series/DataFrame to a new value.
- The old and new values have to be specified inside the brackets separated by comma.



```
df5.fillna(0)
```

	sex	height	weight
0	M	1.73	0.0
1	F	1.61	54.0
2	M	1.8	72.0
3	F	unknown	63.0
4	F	1.69	58.0

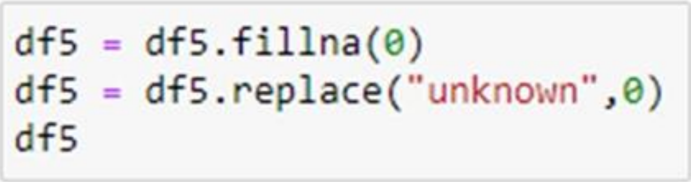


```
df5.replace("unknown",0)
```

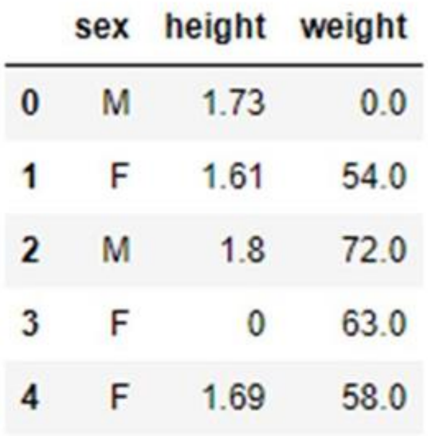
	sex	height	weight
0	M	1.73	NaN
1	F	1.61	54.0
2	M	1.8	72.0
3	F	0	63.0
4	F	1.69	58.0

Data Preparation and Data Cleaning

- Notice that for either these two methods, the result is another object (Series or DataFrame) without changing the original one.
- To update the original object, assign it to the new object.



```
df5 = df5.fillna(0)
df5 = df5.replace("unknown",0)
df5
```

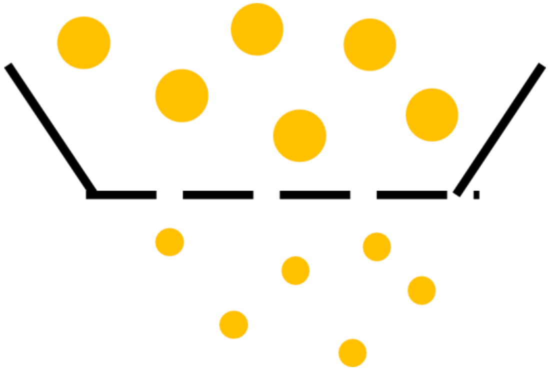


	sex	height	weight
0	M	1.73	0.0
1	F	1.61	54.0
2	M	1.8	72.0
3	F	0	63.0
4	F	1.69	58.0

Data Preparation and Data Cleaning

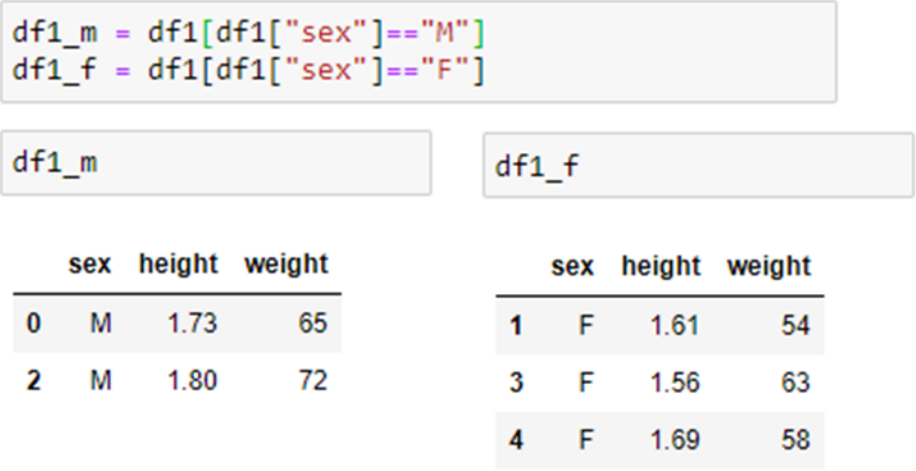
- In data preparation, another useful technique is **data filtering** which refers to selecting desirable samples from the dataset under some certain criteria.
- In a pandas DataFrame, such criteria can be based on the values of its columns.
- The syntax is as follows:

```
new_df = old_df[old_df["col_name"](relation)(number)]
```



Data Preparation and Data Cleaning

- For example, let’s consider the original DataFrame `df1` containing the health data of 5 students in the previous session.
- Suppose we would like to create two new DataFrames by separating `df` into the two gender groups.
- We can check if the value in `df["sex"]` is equal to "`M`" or "`F`". Notice that **for equality we use double equal signs** ==.



```
df1_m = df1[df1["sex"]=="M"]
df1_f = df1[df1["sex"]=="F"]
```

```
df1_m
```

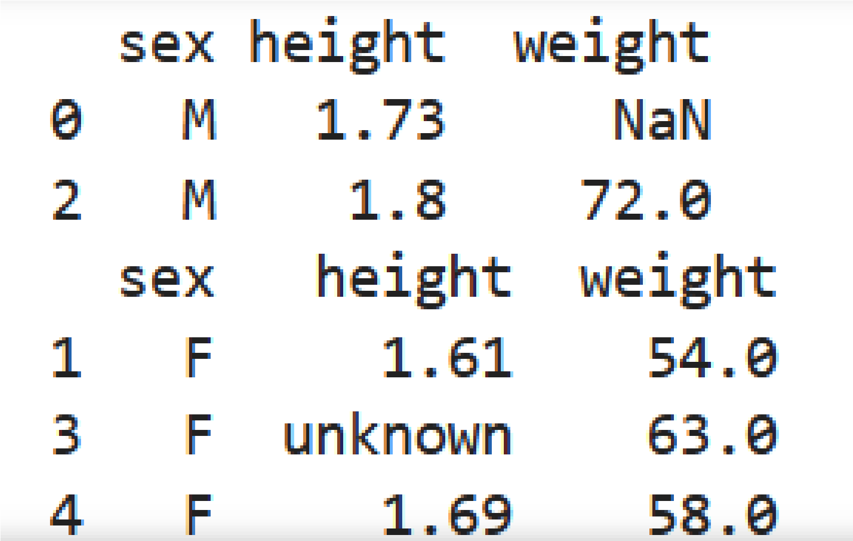
	sex	height	weight
0	M	1.73	65
2	M	1.80	72

```
df1_f
```

	sex	height	weight
1	F	1.61	54
3	F	1.56	63
4	F	1.69	58

Data Preparation and Data Cleaning

```
df5_m = df5[df5["sex"]=="M"]
df5_f = df5[df5["sex"]=="F"]
print(df5_m)
print(df5_f)
```



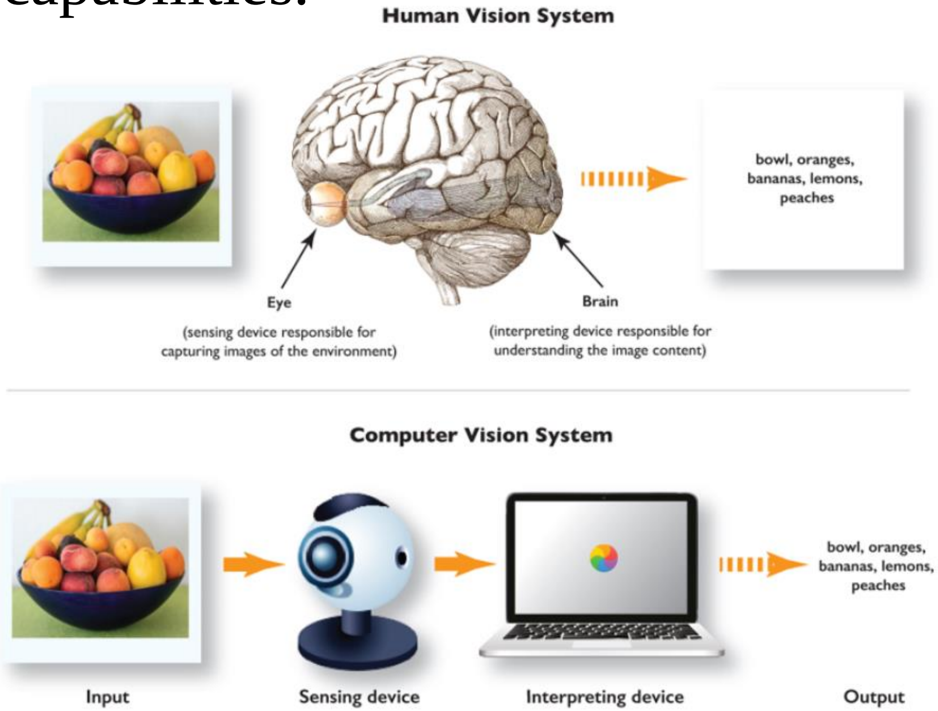
```
  sex height  weight
0   M   1.73     NaN
2   M    1.8    72.0
  sex    height  weight
1   F      1.61    54.0
3   F   unknown    63.0
4   F      1.69    58.0
```

Introduction to Computer Vision

- Definition: A subfield of AI that extracts meaningful information from images, videos, and other visual inputs.
- Goal: Train machines to observe, interpret, and make decisions from visual data using machine learning and deep learning algorithms.

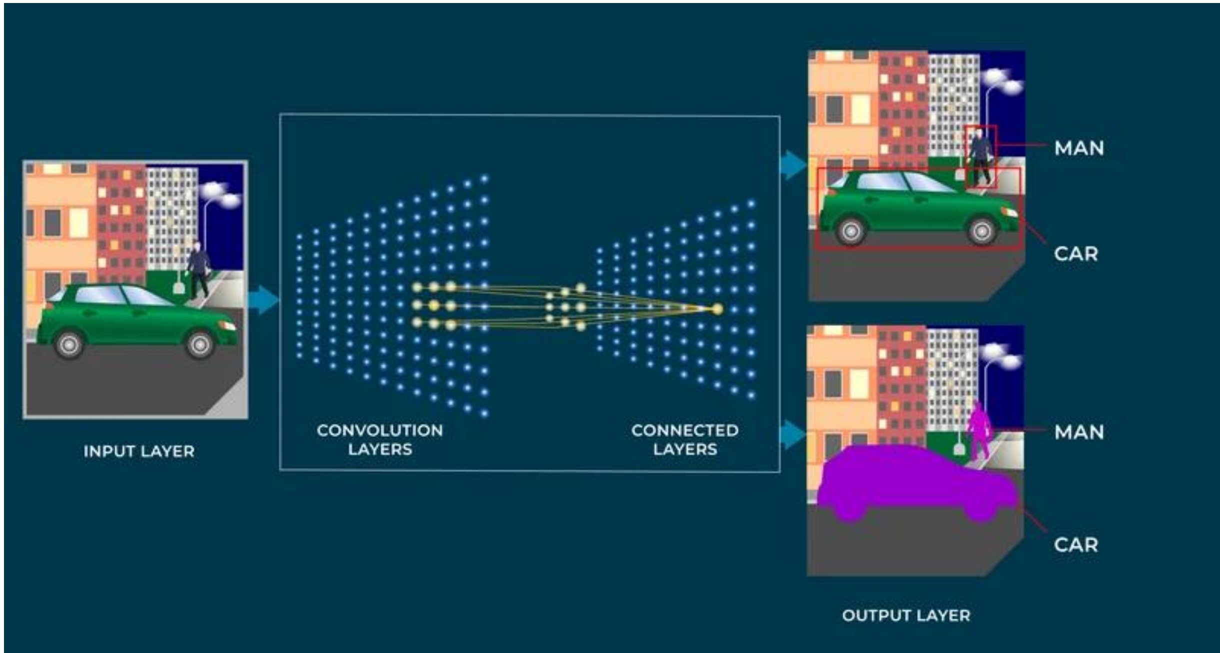
How Computer Vision Works?

- Computer vision functions similarly to human vision but uses cameras, sensors, and algorithms instead of biological mechanisms.
- Machines can analyze thousands of images per minute, often surpassing human capabilities.


Human Vision System
bowl, oranges, bananas, lemons, peaches
Eye
(sensing device responsible for capturing images of the environment)
Brain
(interpreting device responsible for understanding the image content)
Computer Vision System
bowl, oranges, bananas, lemons, peaches
Input
Sensing device
Interpreting device
Output

Key Technologies in Computer Vision

- Deep Learning: Uses artificial neural networks to mimic the human brain's learning process.
- Convolutional Neural Networks (CNNs): Break images into pixels and use convolutions to identify and label content.
- Recurrent Neural Networks (RNNs): Analyze video data by detecting patterns across multiple frames.

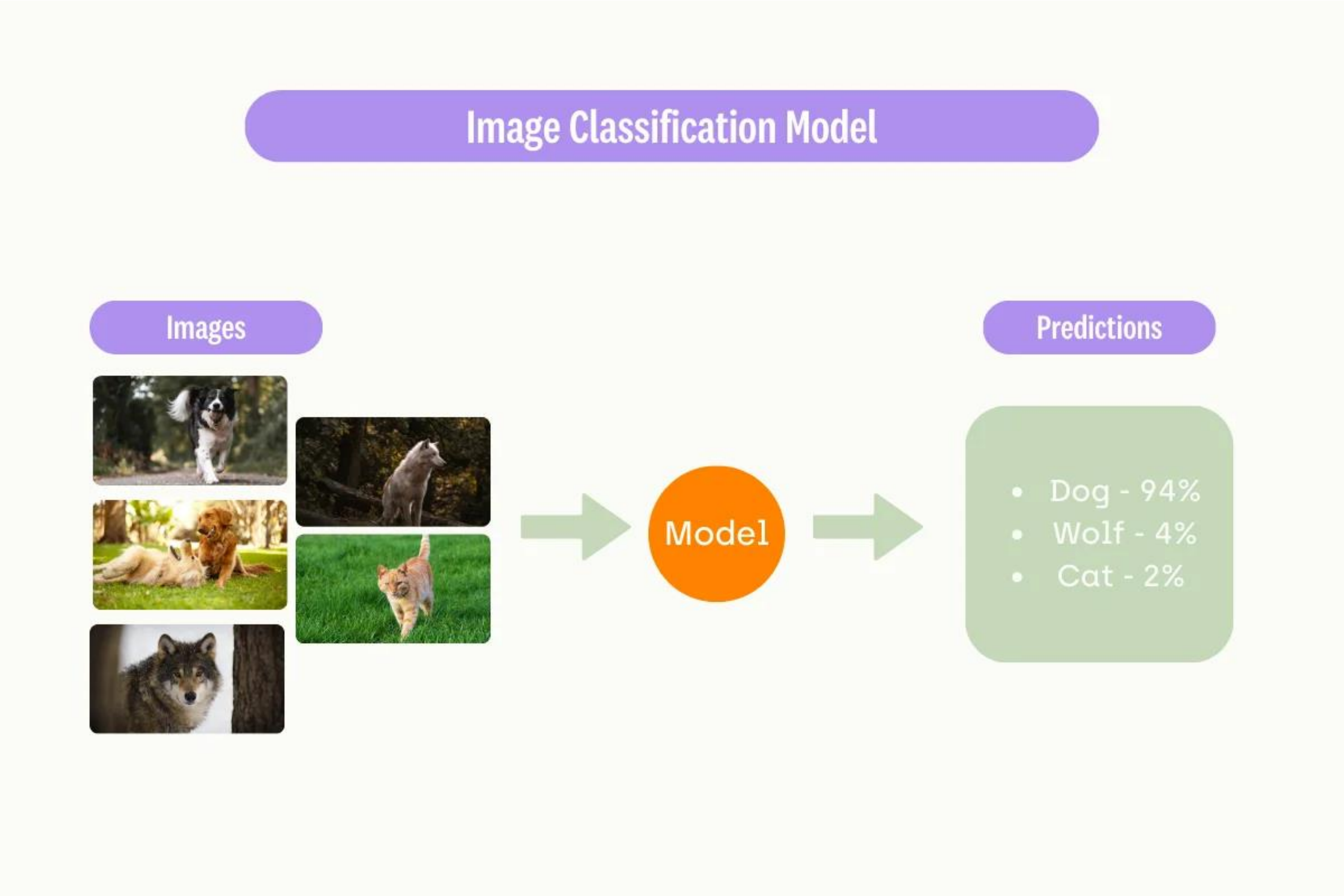


Applications of Computer Vision

- Autonomous Vehicles: Identifying road signs, pedestrians, and other cars in real-time.
- Healthcare: Analyzing medical images for tumor detection, X-ray analysis, and more.
- Agriculture: Monitoring crops, predicting yields, and detecting soil conditions.
- Security: Analyzing live footage to detect unauthorized access or safety issues.

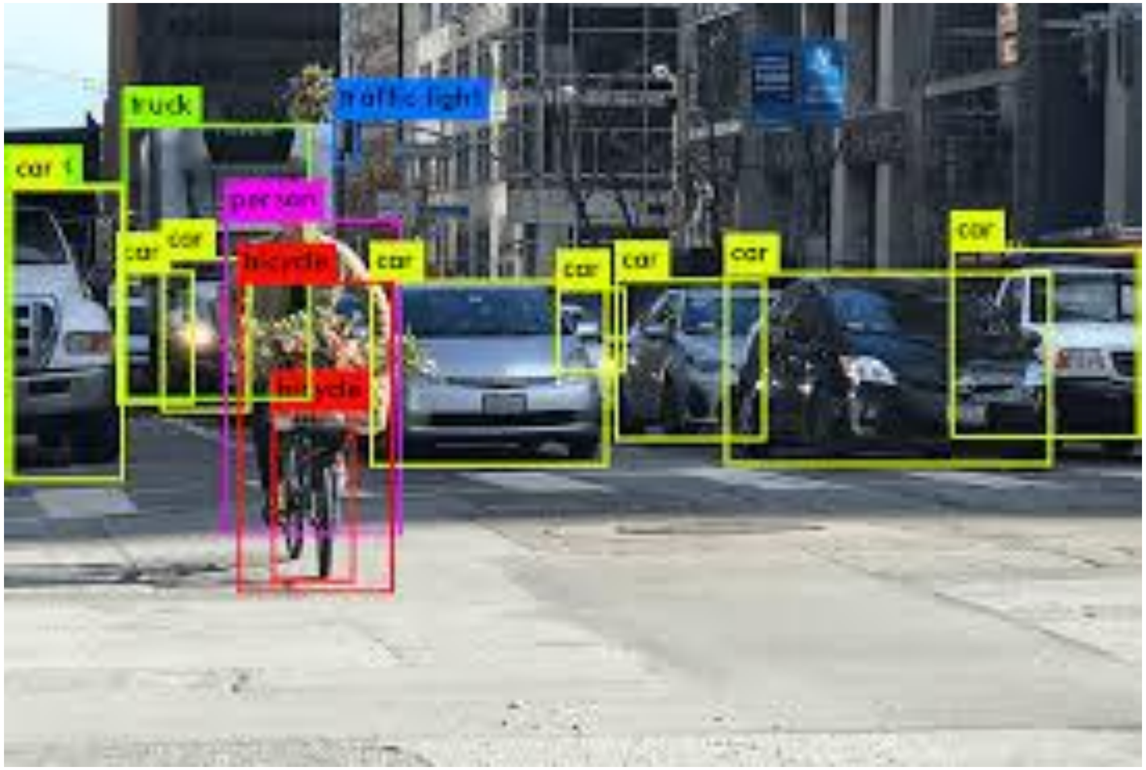
Common Tasks in Computer Vision

- Image Classification: Categorizing images into specific classes.



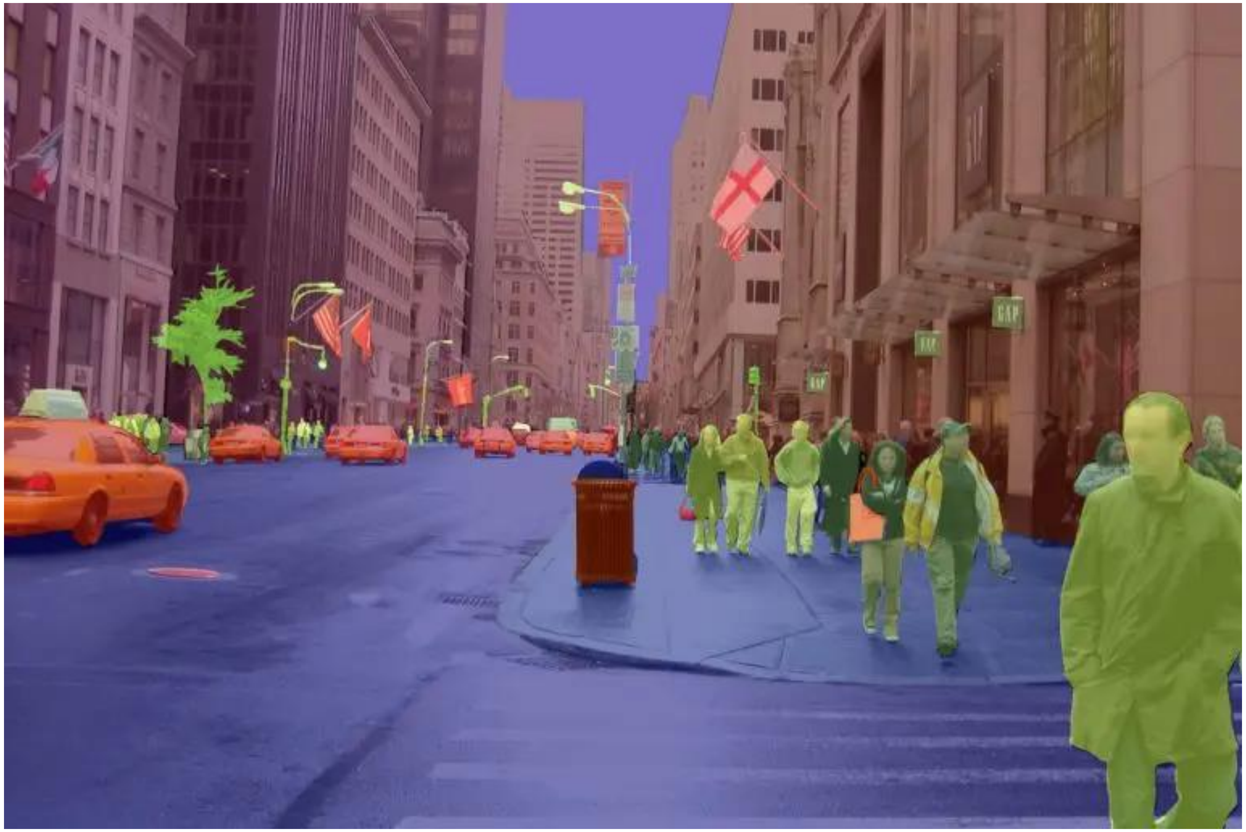
Common Tasks in Computer Vision

- Object Detection: Identifying and locating objects within an image.



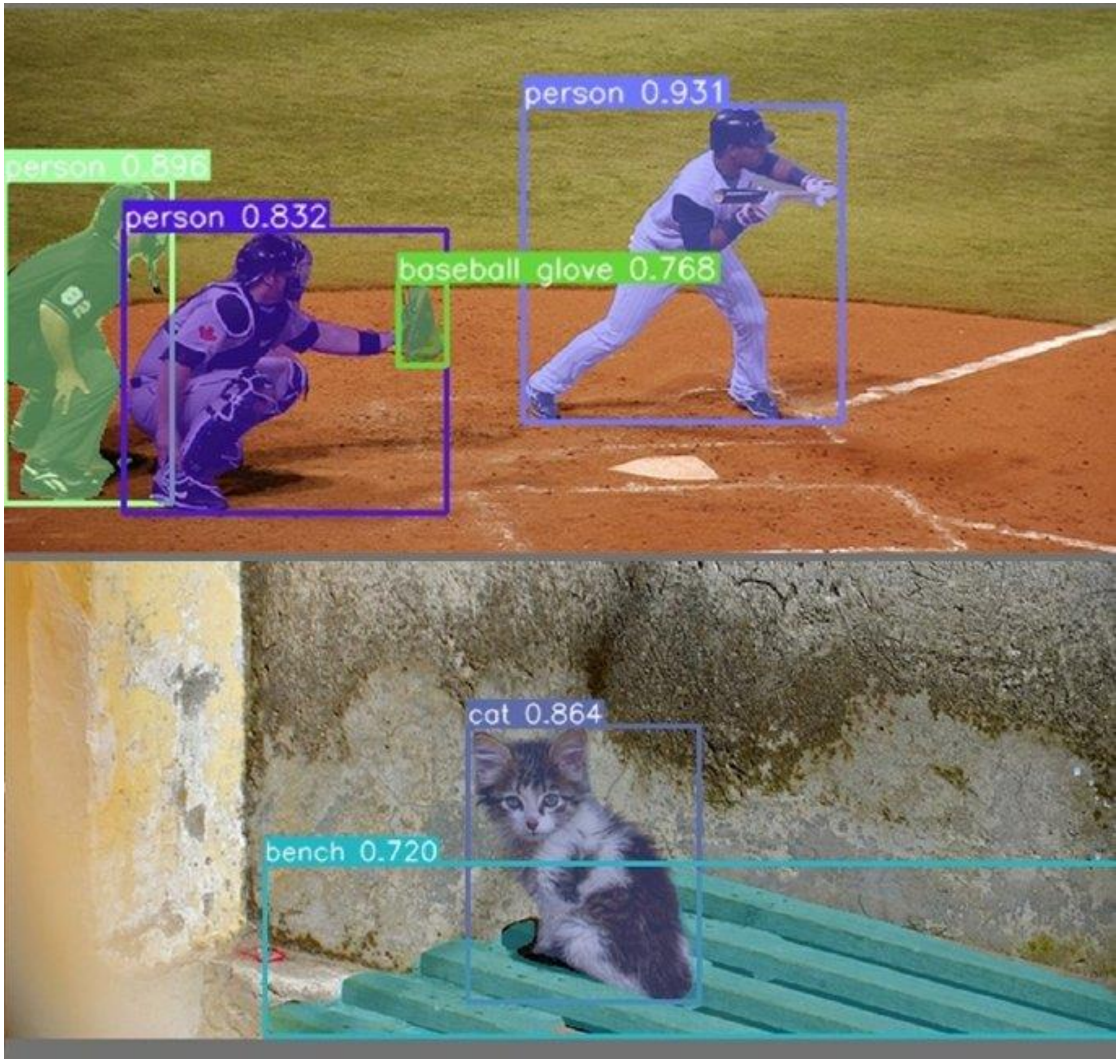
Common Tasks in Computer Vision

- Segmentation: Dividing images into regions to differentiate multiple objects within a frame.



YOLOv7-mask algorithm

- You Only Look Once (YOLO) is a state-of-the-art, real-time object detection algorithm introduced in 2015
- YOLOv7-mask algorithm for instance segmentation.
- YOLOv7 is one of the best-performing real-time algorithms.



Introduction to OpenCV

- OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library.
- OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.
- OpenCV makes it easy for businesses to utilize and modify the code.

Install OpenCV

```
!pip install opencv-python
```

Install it first, and then check the version:

```
import cv2

print(cv2.__version__)
```

Install OpenCV

Sometimes, Jupyter runs in a different environment than where you installed opencv-python. Make sure you run !pip show opencv-python to check if it's installed in the same environment as Jupyter.

Also, try restarting your Jupyter kernel if you haven't already. Go to Kernel > Restart and give it a whirl.

Install OpenCV

If you're still hitting a wall, you can try installing the package directly within your Jupyter cell:

```
!pip install opencv-python
import cv2
print(cv2.__version__)
```

Doing this ensures that the package is installed in the environment where Jupyter is running.

Open a picture

```
import cv2

img = cv2.imread(r'C:\Users\User user\Desktop\koala.jpg')
cv2.imshow("Koala", img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Open a picture

```
import cv2
import imutils
import matplotlib.pyplot as plt

img = cv2.imread(r'C:\Users\User user\Desktop\koala.jpg')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

Resize a picture

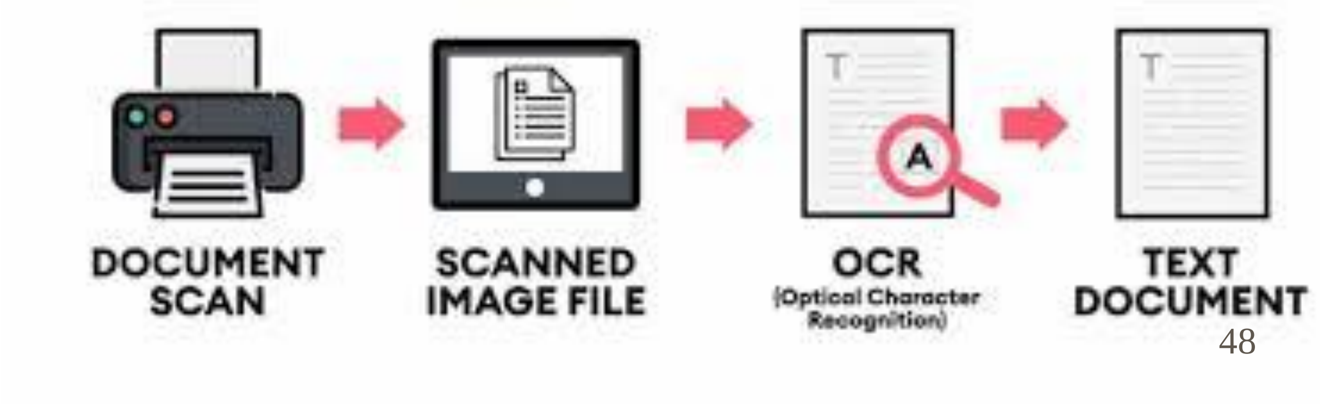
```
resized_img = imutils.resize(img, width=50)
plt.imshow(cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB))
plt.show()
```

Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is the process that converts an image of text into a machine-readable text format.

For example, if you scan a form or a receipt, your computer saves the scan as an image file. You cannot use a text editor to edit, search, or count the words in the image file.

However, you can use OCR to convert the image into a text document with its contents stored as text data.



Optical Character Recognition (OCR)

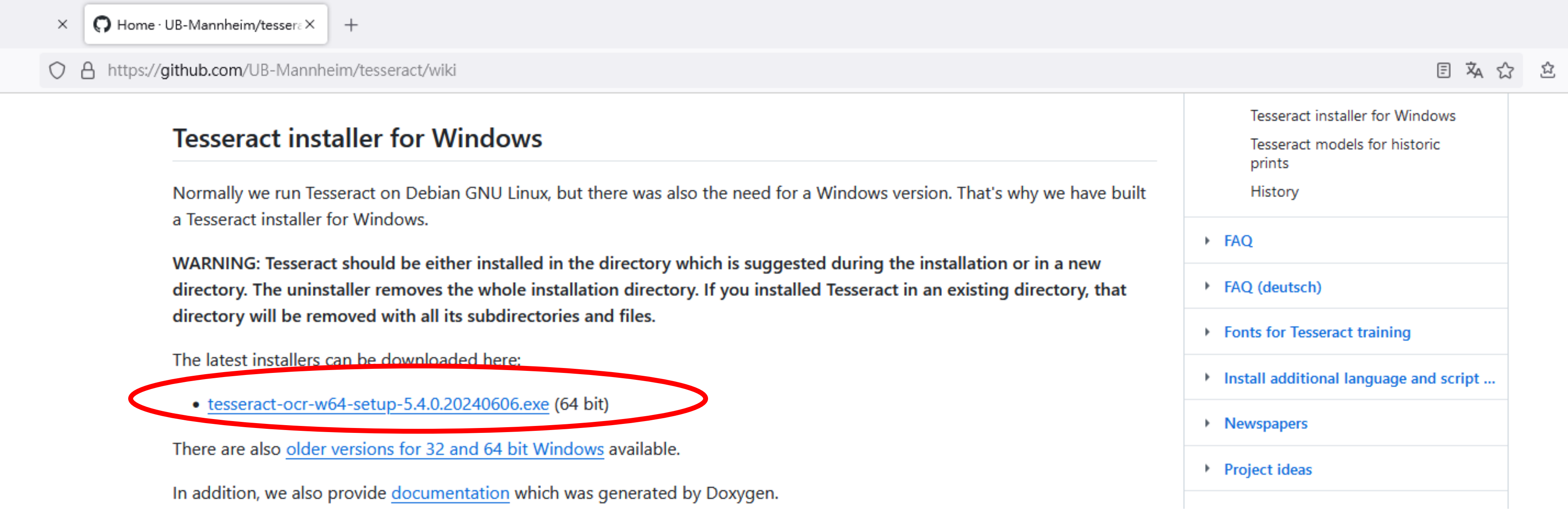
Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images. To use OCR, install pytesseract first:

```
!pip install pytesseract==0.3.8
```

Optical Character Recognition (OCR)

Download the Tesseract-OCR software from:

https://github.com/UB-Mannheim/tesseract/wiki



Home · UB-Mannheim/tesser
https://github.com/UB-Mannheim/tesseract/wiki

Tesseract installer for Windows

Normally we run Tesseract on Debian GNU Linux, but there was also the need for a Windows version. That's why we have built a Tesseract installer for Windows.

WARNING: Tesseract should be either installed in the directory which is suggested during the installation or in a new directory. The uninstaller removes the whole installation directory. If you installed Tesseract in an existing directory, that directory will be removed with all its subdirectories and files.

The latest installers can be downloaded here:

- tesseract-ocr-w64-setup-5.4.0.20240606.exe (64 bit)

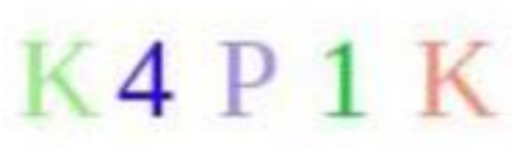
There are also older versions for 32 and 64 bit Windows available.

In addition, we also provide documentation which was generated by Doxygen.

Tesseract installer for Windows
Tesseract models for historic prints
History
FAQ
FAQ (deutsch)
Fonts for Tesseract training
Install additional language and script ...
Newspapers
Project ideas

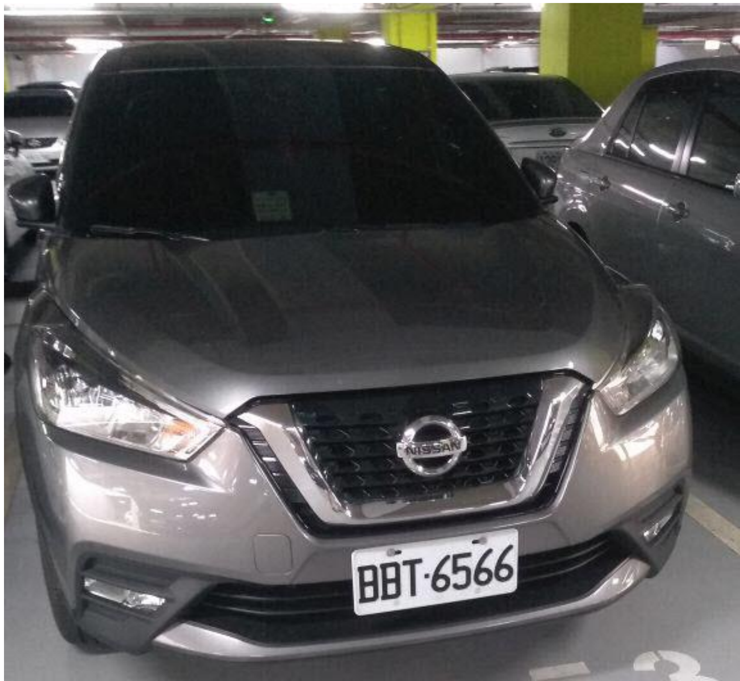
Optical Character Recognition (OCR)

```
from PIL import Image
import pytesseract

pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

img = Image.open(r'C:\Users\User user\Desktop\number.jpg')

text = pytesseract.image_to_string(img, lang="eng")
print(text.strip())
```



Reading number plate of a car

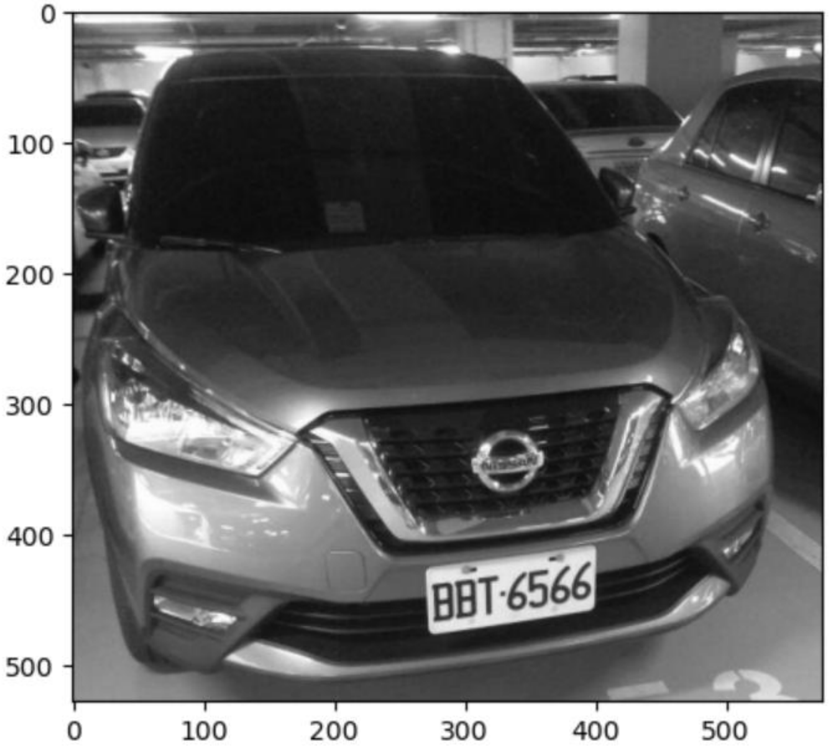
```
import cv2
import pytesseract
import imutils
import matplotlib.pyplot as plt

pytesseract.pytesseract.tesseract_cmd=r"C:\Program Files\Tesseract-OCR\tesseract.exe"
path = r"C:\Users\User user\Desktop\car.jpg".strip("\u202A")
img = cv2.imread(path)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```



Reading number plate of a car

Change it to gray scale:

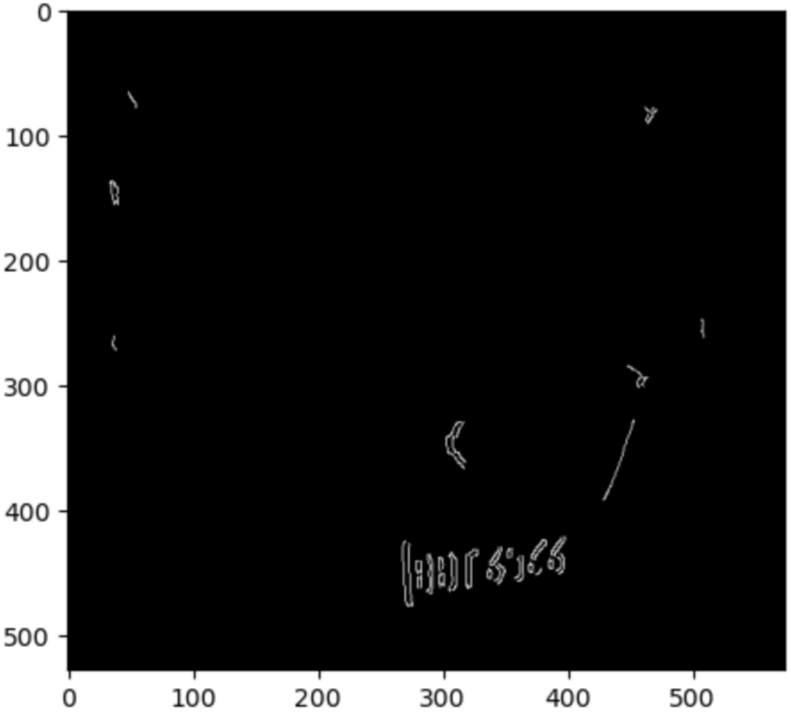
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(cv2.cvtColor(gray, cv2.COLOR_BGR2RGB))
plt.show()
```



Reading number plate of a car

Use different functions to do edge detection:

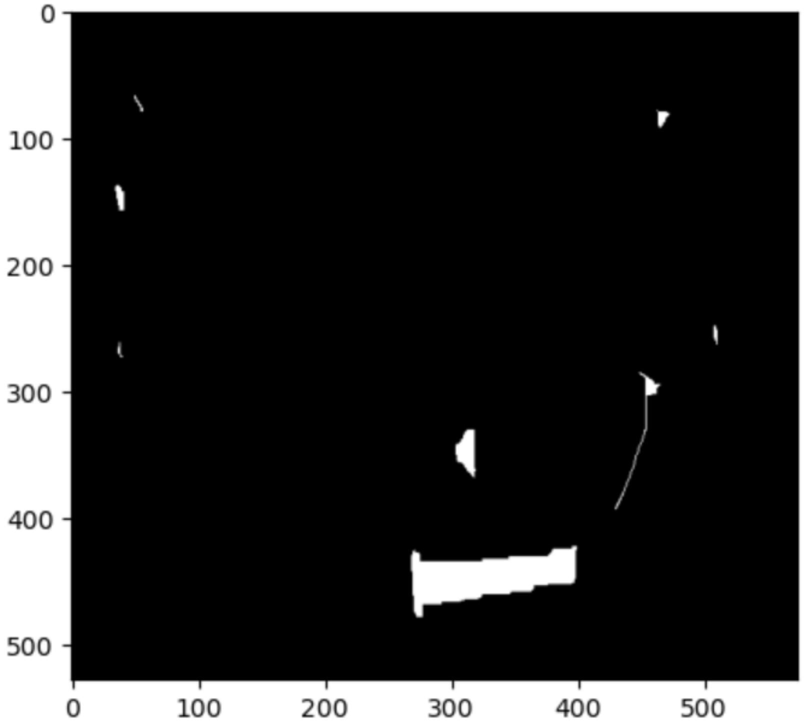
```
gblur = cv2.GaussianBlur(gray,(5,5),10)

sobel = cv2.Sobel(gblur, cv2.CV_8U, 1, 0, ksize=1)

canny = cv2.Canny(sobel, 250, 100)

plt.imshow(cv2.cvtColor(canny, cv2.COLOR_BGR2RGB))
plt.show()
```



Reading number plate of a car

Locate the segment where the plate is located:

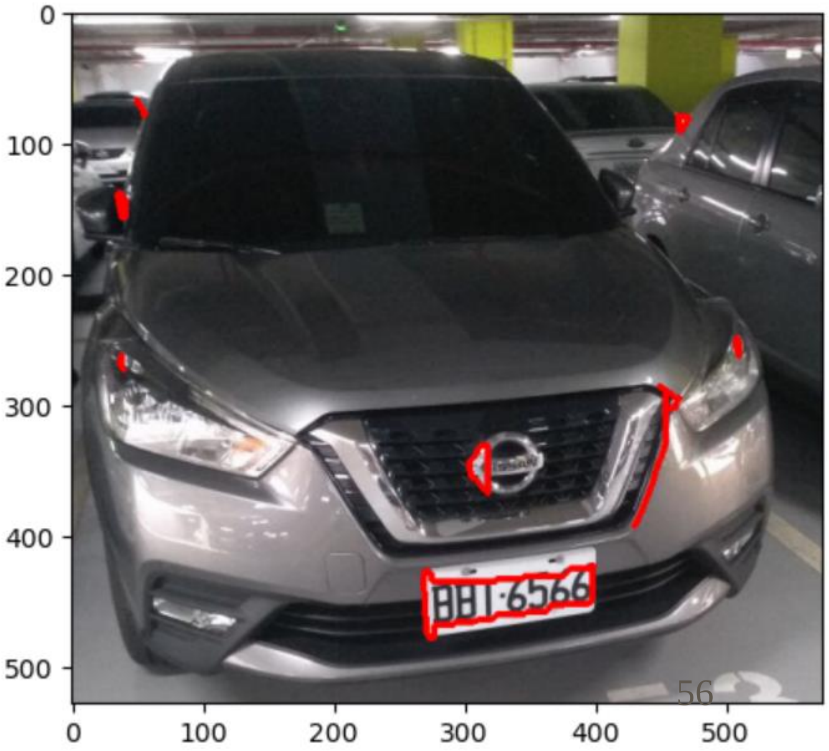
```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(40,40))
morph = cv2.morphologyEx(canny, cv2.MORPH_CLOSE, kernel)

plt.imshow(cv2.cvtColor(morph, cv2.COLOR_BGR2RGB))
plt.show()
```



Reading number plate of a car

Show the contours:

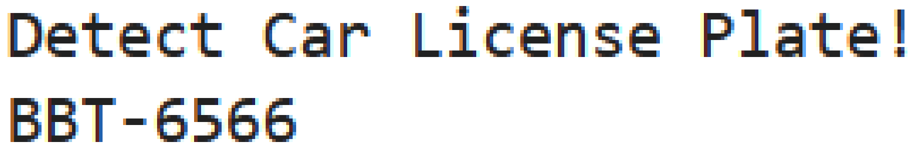
```
contours, hierarchy = cv2.findContours(morph, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
img2 = img.copy()
cv2.drawContours(img2,contours, -1, (0,0,255), 3)

plt.imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))
plt.show()
```



Reading number plate of a car

Read the number plate:

```
result = None
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    if w > 2 * h:
            print("Detect Car License Plate!")
            result = img[y:y+h,x:x+w]
            cv2.imshow("Plate", result)
            text = pytesseract.image_to_string(result, lang="eng")
            if text:
                    print(text.strip())
                    break
cv2.waitKey(0)
cv2.destroyAllWindows()
```



```
Detect Car License Plate!
BBT-6566
```

Checklist

- Can you:
  1. Describe AI’s application in healthcare?
  2. Introduce Python data analysis (Pandas)?
  3. Apply descriptive statistics and arithmetic in Pandas?
  4. Read data for analytics in Pandas?
  5. Prepare and clean data in Pandas?
  6. Describe what is computer vision?