SBS4115 Fundamentals of AI & Data Analytics

AI Technologies and Data Analytics

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

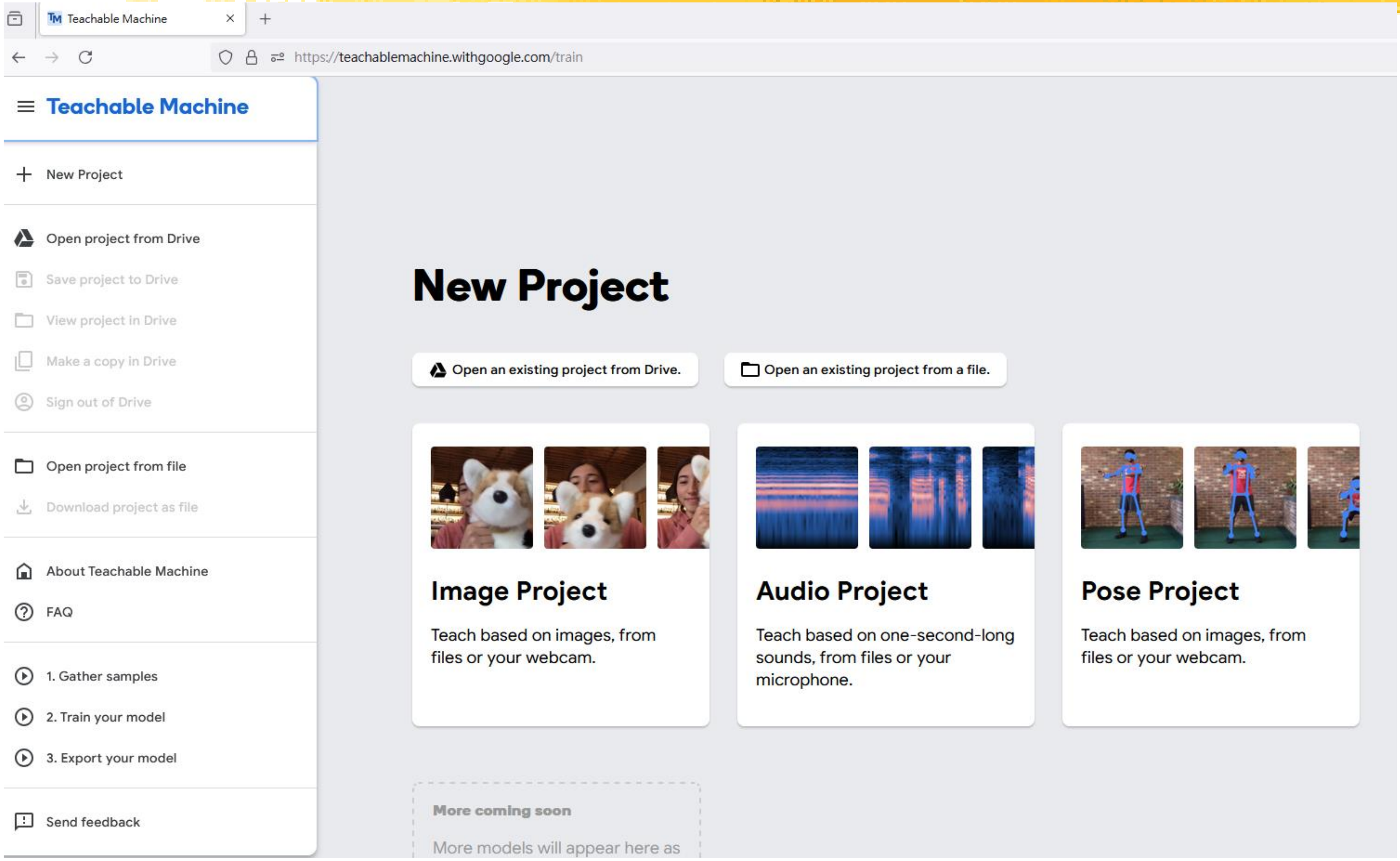
高科院
Thei

Department of Construction, Environment and Engineering

Intended Learning Outcomes

- By the end of this lecture, you will be able to…
  - AI model training with Teachable Machine
  - Discuss data visualization
  - Display a frequency plot in Pandas
  - Plot a histogram or a density plot using data binning
  - Use a line graph
  - Make a scatter plot.

Teachable Machine



Teachable Machine

https://teachablemachine.withgoogle.com/train

Teachable Machine

New Project

Open project from Drive

Save project to Drive

View project in Drive

Make a copy in Drive

Sign out of Drive

Open project from file

Download project as file

About Teachable Machine

FAQ

1. Gather samples

2. Train your model

3. Export your model

Send feedback

New Project

Open an existing project from Drive.

Open an existing project from a file.

Image Project

Teach based on images, from files or your webcam.

Audio Project

Teach based on one-second-long sounds, from files or your microphone.

Pose Project

Teach based on images, from files or your webcam.

More coming soon

More models will appear here as

Teachable Machine

- Developed by Google
- No coding required
- Applications: Image, Sound, and Pose recognition

Teachable Machine

- Visit Teachable Machine (https://teachablemachine.withgoogle.com/)
- Overview of the interface
- Types of projects: Image, Audio, Pose

Gather data

Step 1: Open Google Chrome

- Make sure you are using Google Chrome on your computer.

Step 2: Open Chrome Web Store

- In the Chrome browser, navigate to the Chrome Web Store by entering this URL in the address bar: https://chrome.google.com/webstore.
- This is where you can find all the extensions available for Chrome.

Gather data

Step 3: Search for an Image Downloader Extension

- In the Chrome Web Store, there is a search bar at the top left.
- Type "Image Downloader" or "Batch Image Downloader" into the search bar and press Enter.
- Some recommended extensions for downloading images in bulk include:
  - Image Downloader
  - Fatkun Batch Download Image

Step 4: Install the Extension

- In the list of results, find the extension you want to use (e.g., Image Downloader or Fatkun Batch Download Image).
- Click on the extension name to view details about it.
- On the extension’s page, click the Add to Chrome button in the top right corner.

Gather data

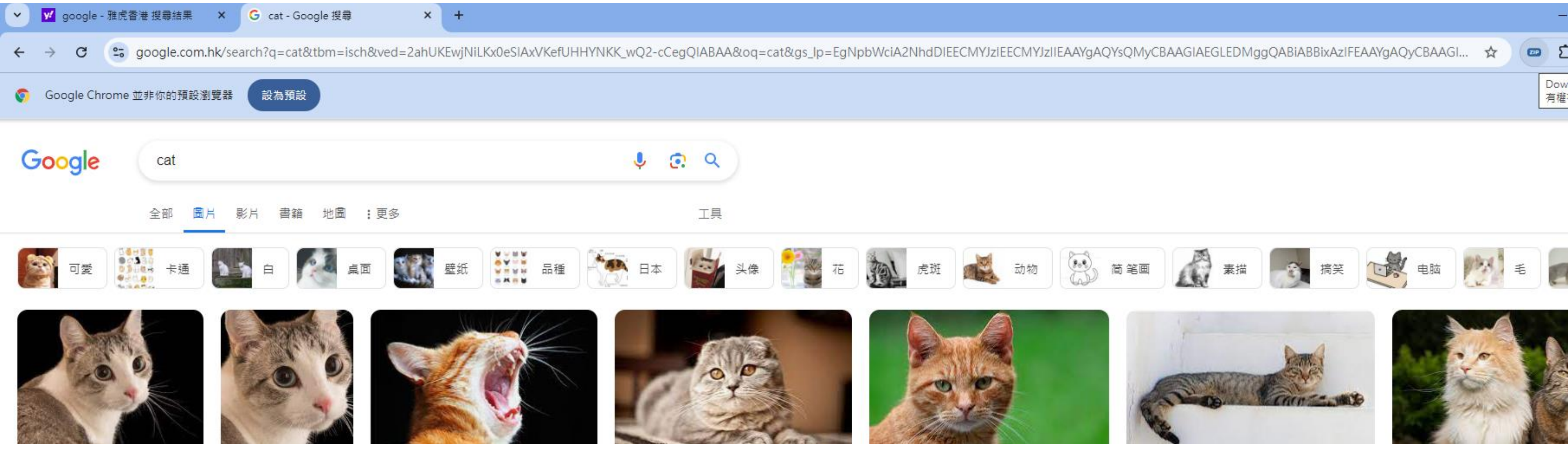
Step 5: Confirm Installation

- A pop-up will appear asking you to confirm the installation of the extension.
- Click Add Extension to proceed.
- Chrome will now download and install the extension. Once installed, you'll see the extension's icon in the top-right corner of your browser (next to the address bar).

Gather data

Step 6: Use the Extension to Download Images in Bulk

- Go to Google Images or any webpage where you want to download images.
  For example, search for “cats" in Google Images.
- Click the extension’s icon in the top-right corner of Chrome (next to the address bar).
- The extension will scan the page and list all the images it has found.
- Select the images you want to download, or use the Select All option if you want to download all the images.
- Click the Download or similar button to save all the selected images to your computer.


google - 雅虎香港 搜尋結果
cat - Google 搜尋
google.com.hk/search?q=cat&tbm=isch&ved=2ahUKEwjNiLKx0eSIAxVKefUHHYNKK_wQ2-cCegQIABAA&oq=cat&gs_lp=EgNpbWciA2NhdDIEECMYJzIEECMYJzIIEAAYgAQYsQMyCBAAGIAEGLEDMggQABiABBixAzIFEAAYgAQyCBAAGI...
Google Chrome 並非你的預設瀏覽器
設為預設
Google
cat
全部
圖片
影片
書籍
地圖
更多
工具
可愛
卡通
白
桌面
壁紙
品種
日本
头像
花
虎斑
动物
简 笔画
素描
搞笑
电脑
毛

Teachable Machine

Train the AI model

- Click “Train Model” button
- Overview of training process
- Adjusting parameters (optional)

Teachable Machine

Test and Evaluate

- Test the model with new examples
- Evaluate performance
- Make adjustments if necessary

Introduction to Data Visualization

- **Data visualization** refers to the representation of data through use of graphics.
- Making informative visualization is an important task in data analysis, no matter as a part of the exploratory process or as a way of generating ideas for models.

Introduction to pandas series

A **Pandas Series** is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). Here are some key aspects of its structure:

Data: The actual data stored in the Series. This can be a list, NumPy array, dictionary, or scalar value.

Index: Labels for the data, which can be integers, strings, dates, etc. If not specified, Pandas will create a default integer index starting from 0.

dtype: The data type of the Series elements, which can be specified or inferred from the data.

Name: An optional name for the Series.

Copy: A boolean indicating whether to copy the input data.

Introduction to pandas series

Try this!

```
import pandas as pd

# Creating a Series from a list
data = [1, 2, 3, 4]
series = pd.Series(data)
print(series)
```

Introduction to pandas series

Try this!

```
import pandas as pd

# Creating a Series with a custom index
data = [1, 2, 3, 4]
index = ['a', 'b', 'c', 'd']
series = pd.Series(data, index=index)
print(series)
```

Introduction to pandas series

Try this!

```
# Accessing by label
print(series['a'])  # Output: 1

# Accessing by integer position
print(series[0])  # Output: 1
```

Differences between pandas series and DataFrame

Both Series and DataFrame are essential data structures, but they serve different purposes:

Pandas Series

One-dimensional: A Series is essentially a one-dimensional array-like object that can hold data of any type (integers, strings, floats, etc.).

Indexed: Each element in a Series has a unique identifier called an index, which can be used to access or manipulate the data.

Single Column: Think of a Series as a single column in a table. It can be created from lists, arrays, dictionaries, and more.

Differences between pandas series and DataFrame

Pandas DataFrame

Two-dimensional: A DataFrame is a two-dimensional, tabular data structure with labeled axes (rows and columns).

Multiple Columns: It can contain multiple Series, making it similar to a table in a database or an Excel spreadsheet.

Flexible Indexing: DataFrames allow for more complex data manipulation and analysis, as they can handle multiple columns of data, each potentially of different types.

Differences between pandas series and DataFrame

```
import pandas as pd

# Creating a Series
data_series = pd.Series([1, 2, 3, 4, 5])

# Setting the name of the Series
data_series.name = 'MyColumnName'

print("Series:\n", data_series)
```

Differences between pandas series and DataFrame

```
# Creating a DataFrame
data_frame = pd.DataFrame({
    'Column1': [1, 2, 3, 4, 5],
    'Column2': ['A', 'B', 'C', 'D', 'E']
})
print("\nDataFrame:\n", data_frame)
```

Storing pandas series in .csv

```
data_series.to_csv(r'C:\Users\User user\Desktop\my_series.csv')
```

```
data_frame.to_csv(r'C:\Users\User user\Desktop\data_frame.csv')
```

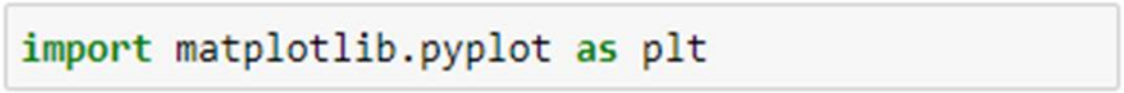
Introduction to Data Visualization

- Python has many **add-on libraries** for making static or dynamic visualizations.
- In this chapter, we will introduce a library called **matplotlib** which stands for mathematics-plot-library and the techniques of making various graphs for presenting statistical data and also the result of artificial intelligence.

matplotlib

Introduction to Data Visualization

- In fact, matplotlib is a desktop plotting package designed for **creating publication quality plots**.
- There are many modules under it including **pyplot**.
- We usually import it by:



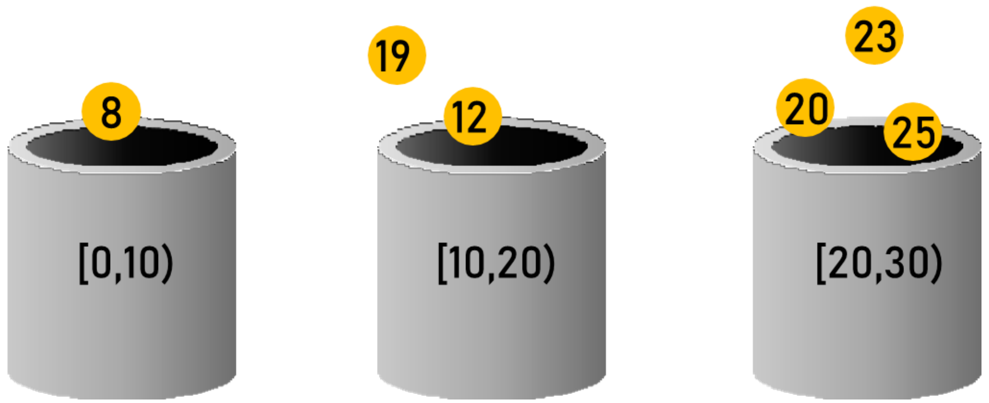
```
import matplotlib.pyplot as plt
```

*in case you fail to import the library, you may install it first by executing the following code in a cell:

```
conda install matplotlib
```

Data Pinning

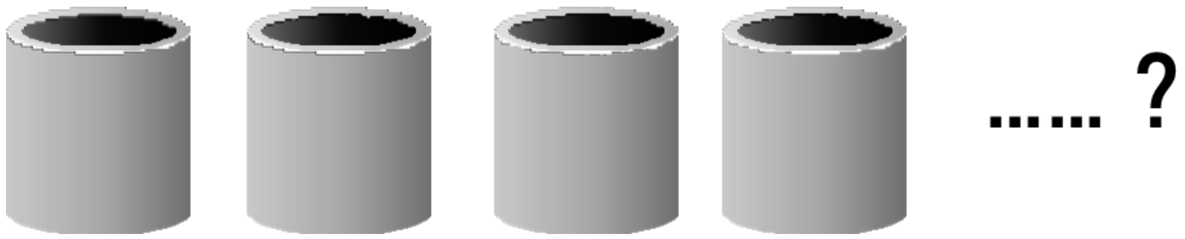
- It would be unrealistic to make a frequency plot of each unique mark.
- Instead, we might **group similar marks together**.
- In statistics, **data binning** is a way to group numbers of more-or-less continuous values into a smaller number of "bins".
- After binning, we might **plot a histogram, or a density plot** based on the frequency of binned values.



Data Pinning

- In pyplot, when we plot a histogram of an array of values from a continuous variable, it will be **automatically binned**.
- We can also **specify the binning criteria** by the number of even width intervals by the following syntax:

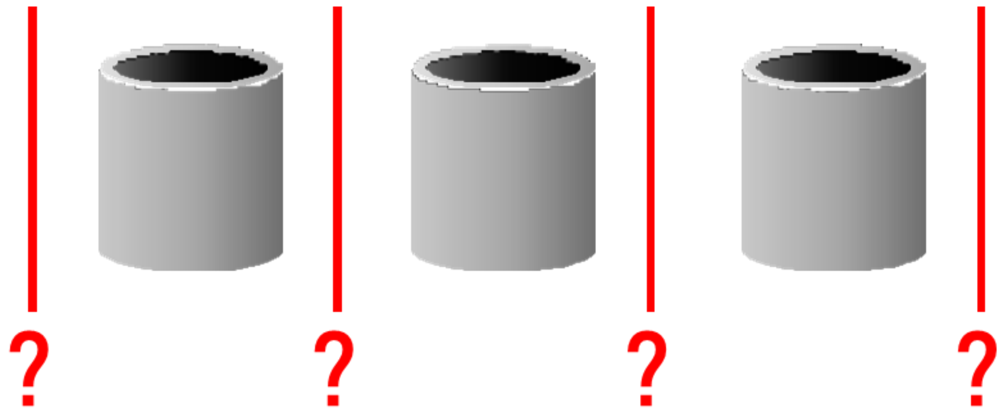
```
plt.hist(data_set, bins=number_of_bins)
```



Data Pinning

or by the end points between intervals by the following syntax:

```
plt.hist(data_set, bins=[point_0,point_1,...,point_n])
```



Frequency Plot

- Consider a set of one-dimensional data.
- The values can be either numerical values (e.g. marks of students) or non-numerical values (e.g. letter grades of students).
- **Frequency plot is based on the number of occurrence of each unique value.**

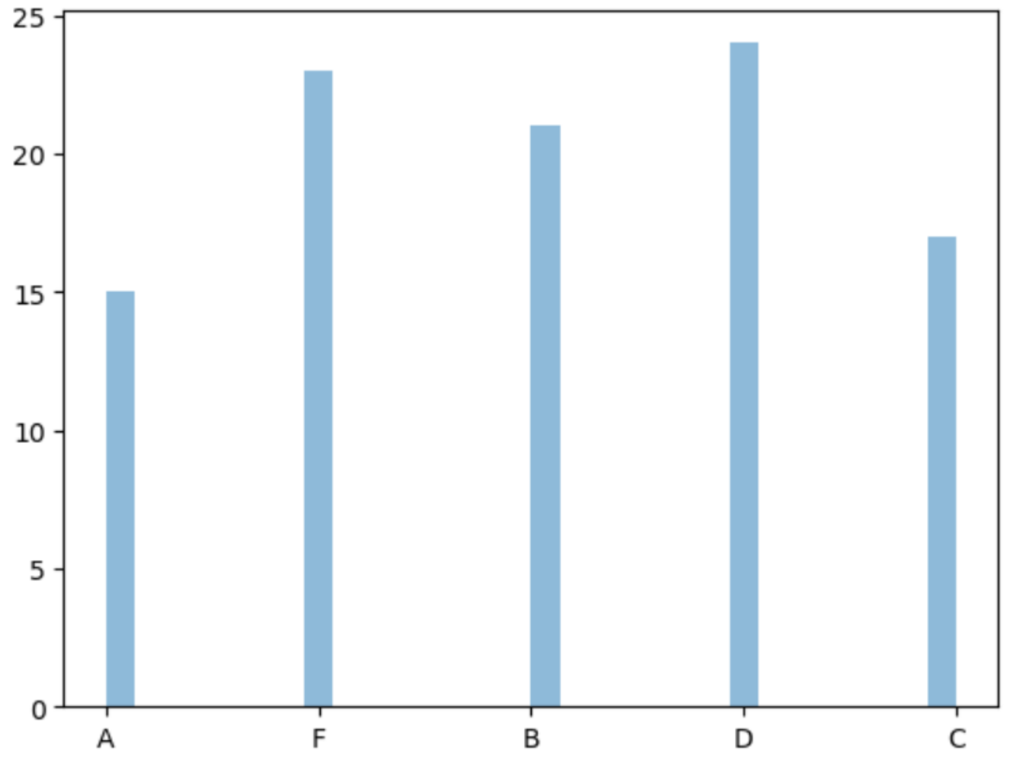
- Let’s use an example to illustrate different plots.
- The file "student_grades.csv" contains the marks and letter grades of 100 students.
- We can first read the file as a single DataFrame, then extract the two columns as two Series.

Histogram Plot

```
import pandas as pd

# Read the file into a DataFrame
df = pd.read_csv(r'C:\Users\User user\Desktop\student_grades.csv')

# Extract the two columns as Series
StudentMarks = df['Marks']
StudentGrade = df['Grade']

# Display the Series
print(StudentMarks)
print(StudentGrade)
```

Histogram Plot

- Based on the grades, we can directly **plot a histogram** using `hist()` to show the frequency of students obtaining each grades.
- The `show()` method displays the plot.



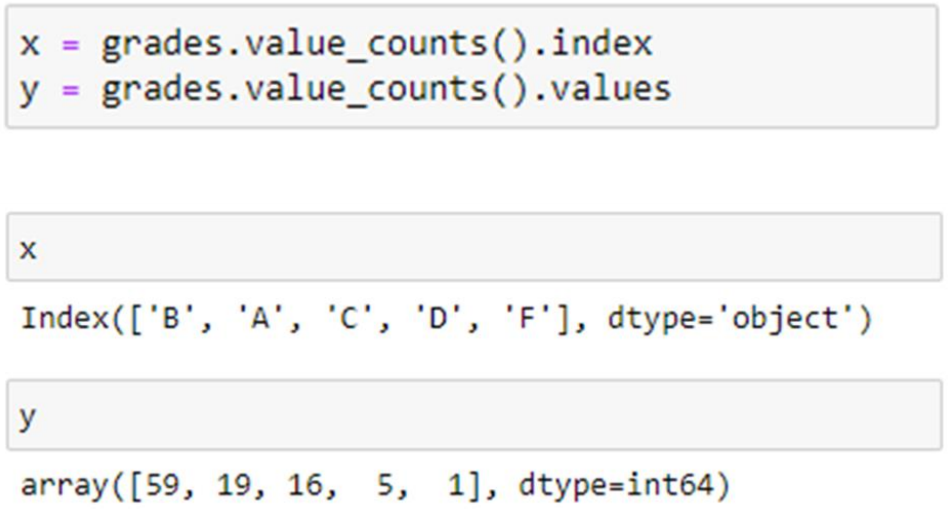
Histogram Plot

```
import matplotlib.pyplot as plt

# Plotting the histograms
plt.hist(StudentGrade, bins=30, alpha=0.5, label='Grade')

# Adding labels and title
plt.xlabel('Grade')
plt.ylabel('Frequency')
plt.title('Histogram of Student Grades')

# Display the plot
plt.show()
```

Frequency Count

- `value_counts()` method in Pandas gives a frequency table of a set of data by counting the frequency of each unique value.
- The result is a Series with the index being each unique value and the values being the frequency of each unique value.

- We might first store it as **two arrays**:



```
x = grades.value_counts().index
y = grades.value_counts().values
```

```
x
```

```
Index(['B', 'A', 'C', 'D', 'F'], dtype='object')
```

```
y
```

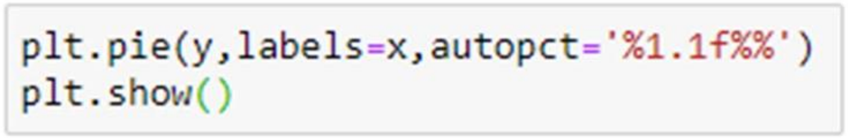
```
array([59, 19, 16,  5,  1], dtype=int64)
```

Frequency Count

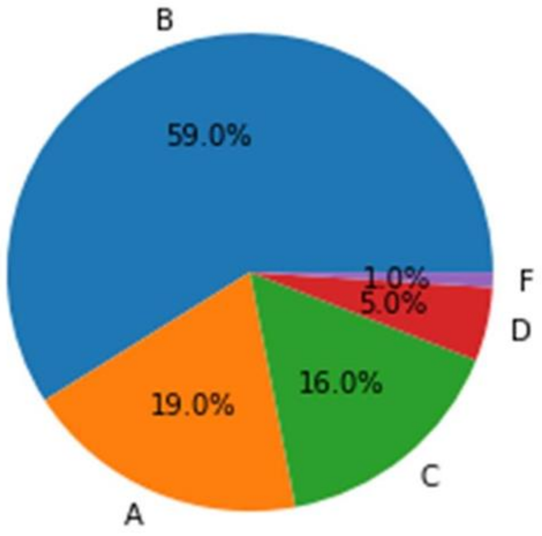
```
import pandas as pd

# Read the CSV file
df = pd.read_csv(r'C:\Users\User user\Desktop\student_grades.csv')

# Perform value counts on the 'grades' column
counts = df['Grade'].value_counts()

# Get unique values and their counts
x = counts.index
y = counts.values
print(x)
print(y)
```

Pie chart

- For pie chart, only an array of value is necessary.
- We might also put the labelling and auto-percentage optionally.



```
plt.pie(y,labels=x,autopct='%1.1f%%')
plt.show()
```



Pie chart

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

# Read the file into a DataFrame
df = pd.read_csv(r'C:\Users\User user\Desktop\student_grades.csv')

# Extract the 'Grade' column as a Series
StudentGrade = df['Grade']
```

Pie chart

```
# Count the occurrences of each grade
grade_counts = StudentGrade.value_counts()

# Plot the pie chart
plt.figure(figsize=(8, 8))
plt.pie(grade_counts, labels=grade_counts.index, autopct='%1.1f%%')
plt.title('Distribution of Student Grades')
plt.axis('equal')

# Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

Line Graph

- Some set of data is dependent to a continuous variable, e.g. stock price is dependent on the time variable.
- To present such data, we may use a **line graph**, which is good for **showing the trend and local extreme values**.

Line Graph

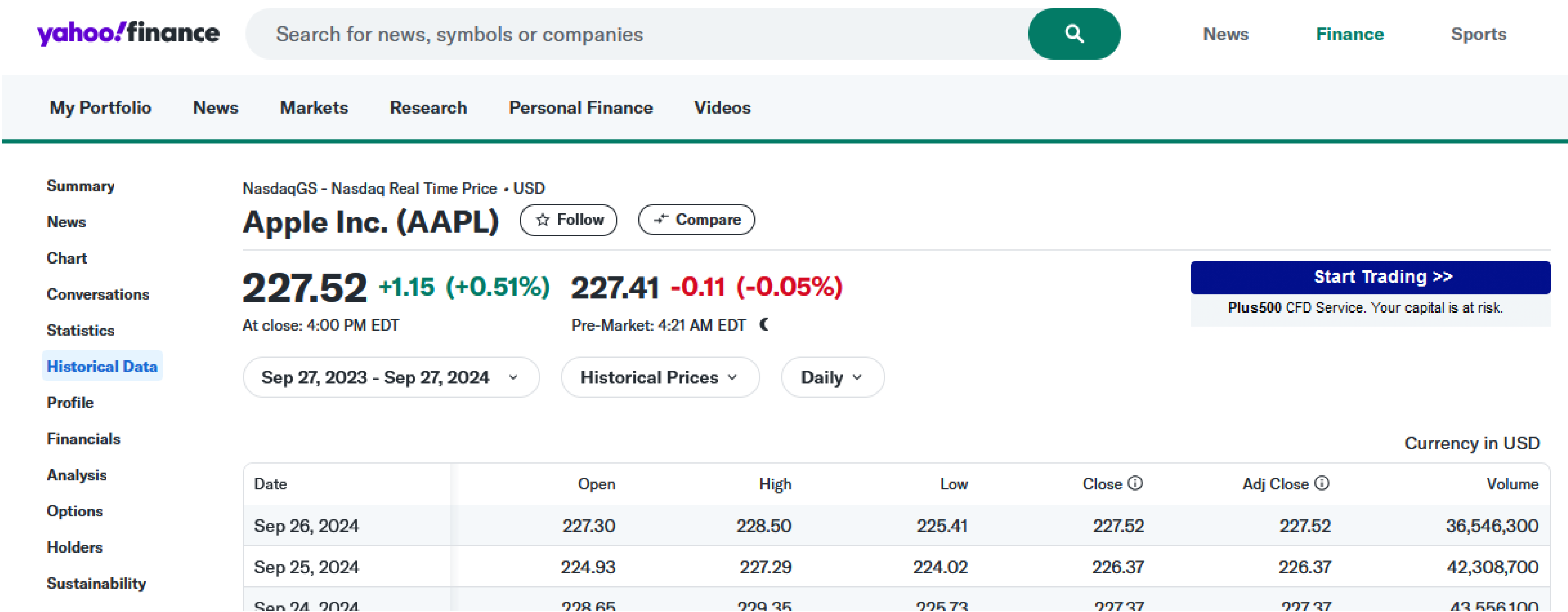
- In pyplot, we can directly use `plot()` to make a line graph of a Series or array.
- Moreover, **the colour and style of the line can be adjusted** by the commands inside the brackets.

command	colour
`'k'`	black
`'g'`	green
`'r'`	red
`'b'`	blue
`'y'`	yellow

command	style
`'--'`	dashed line
`':'`	dotted line
`'*'`	points with stars
`'o'`	points with circle
`'+'`	points with plus sign

Line Graph

- As an example, we will try to study the stock price of two companies, Apple (AAPL) and Microsoft (MSFT).
- The files "AAPL.csv" and "MSFT.csv" contains the historical data in 5 years.
- The column of adjusted close price is stored as a Series.



NasdaqGS - Nasdaq Real Time Price • USD

Apple Inc. (AAPL)

227.52 +1.15 (+0.51%) 227.41 -0.11 (-0.05%)

At close: 4:00 PM EDT Pre-Market: 4:21 AM EDT

Plus500 CFD Service. Your capital is at risk.

Sep 27, 2023 - Sep 27, 2024

Currency in USD

Date	Open	High	Low	Close	Adj Close	Volume
Sep 26, 2024	227.30	228.50	225.41	227.52	227.52	36,546,300
Sep 25, 2024	224.93	227.29	224.02	226.37	226.37	42,308,700
Sep 24, 2024	228.65	229.35	225.73	227.37	227.37	43,556,100

Line Graph

- You can download the csv files using Python code:

```
# Install yahoo finance to Jupyter Notebook
!pip install yfinance

import yfinance as yf

# Download historical data for AAPL
apple_price = yf.download('AAPL', start='2020-01-01', end='2024-09-27')

# Save to CSV
apple_price.to_csv(r'C:\Users\User user\Desktop\AAPL.csv')
```

Line Graph

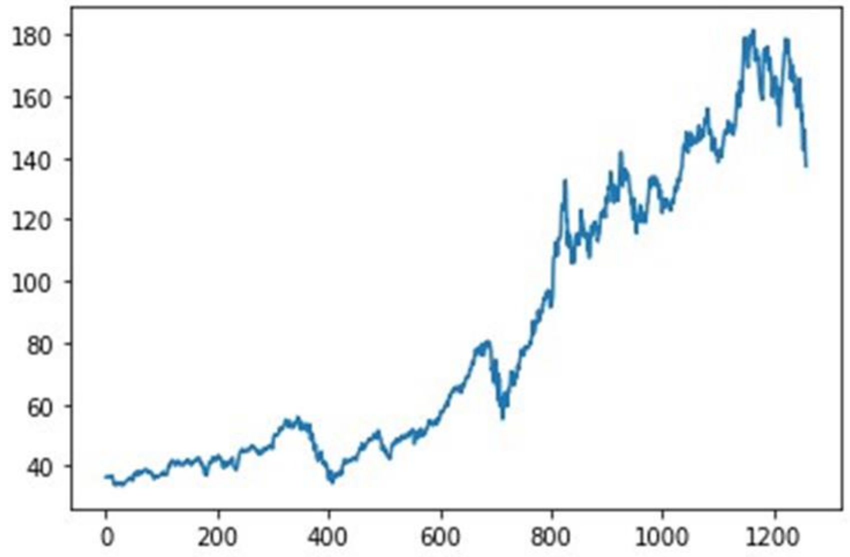
- Repeat the process for Microsoft’s stock prices:

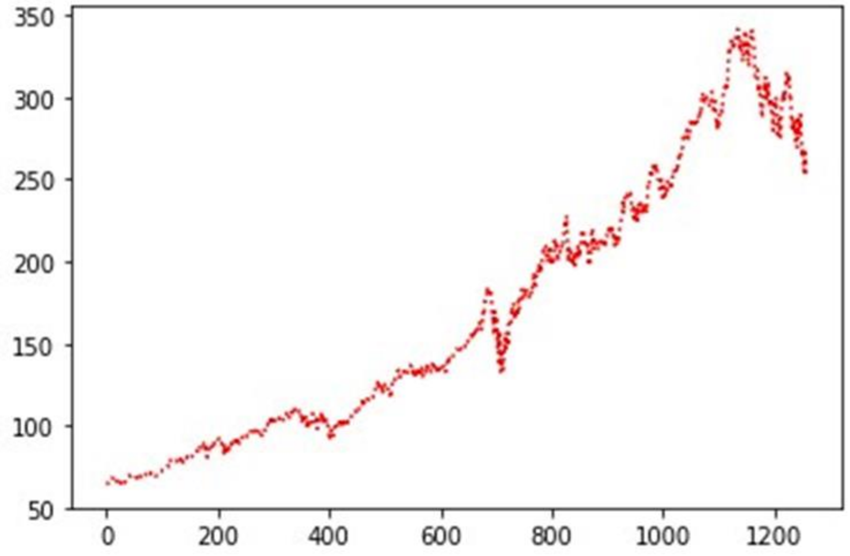
```
# Download historical data for MSFT
msft_price = yf.download('MSFT', start='2020-01-01', end='2024-09-27')

# Save to CSV
msft_price.to_csv(r'C:\Users\User user\Desktop\MSFT.csv')
```

Line Graph

```
plt.plot(apple_price)
plt.show()
```

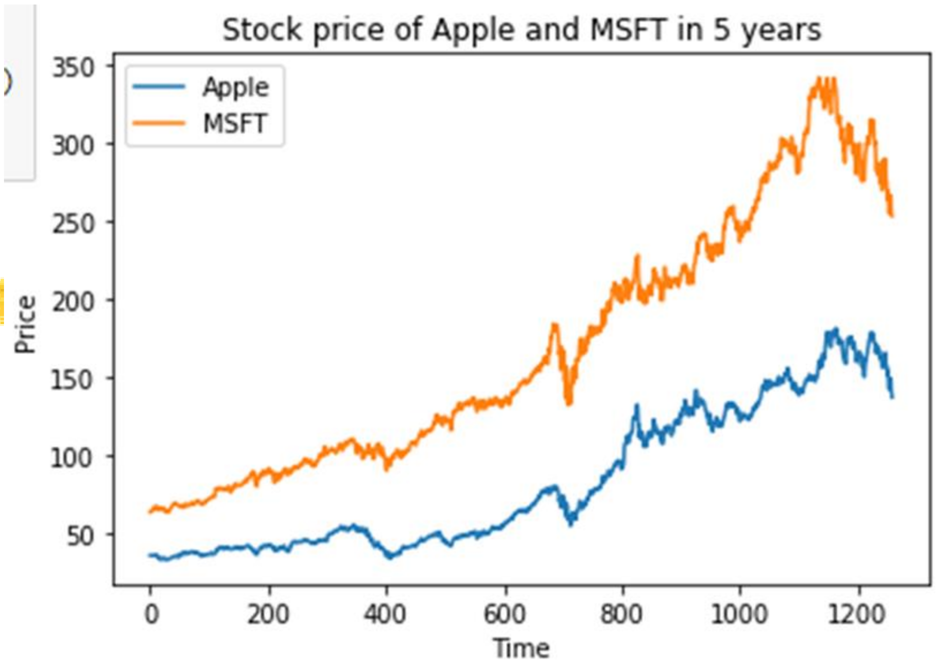
```
plt.plot(msft_price,'r:')
plt.show()
```





- However, this is hard for comparison.

Line Graph



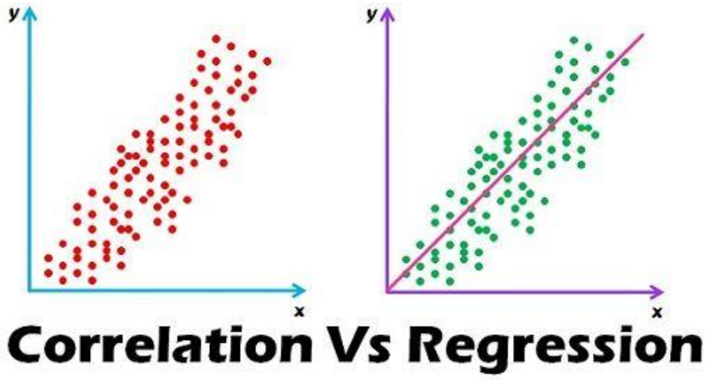
- In matplotlib, we can **plot several lines on the same figure**.
- To show information of the figure and distinguish the lines, we can also **add title, labels and legend**.
- This applies not only on line graph but also on the graphs we have introduced before.
- Upon executing the `show()` command, all these graphs and information before will be displayed on the same figure.

Line Graph

```
appl_price_series = apple_price["Adj Close"]

msft_price_series = msft_price["Adj Close"]

plt.plot(appl_price_series)
plt.plot(msft_price_series)
plt.xlabel("Time")
plt.ylabel("Price")
plt.title("Stock price of Apple and MSFT in 4 years")
plt.legend(["Apple", "MSFT"])
plt.show()
```

Scatter Plot

- A set of data might consist of **more than one variables**.
- An example is the health data contains two variables, height and weight.
- For such data, **the purpose of visualization is to show the relationship between the two variables**.
- This can be illustrated by a **scatter plot**.
- More details will be discussed in a later chapter related to correlation and regression.



Scatter Plot

- For example, we might want to study the relation between height and weight.
- After reading the csv file "health.csv" into a DataFrame, extract the columns representing the two variables (height and weight) into two Series.
- For each data index, a point with x-coordinate being its height and y-coordinate being its weight is plot on the figure.
- As a result, a scatter plot should contain as many points as the number of rows of the original DataFrame.

Scatter Plot

```
df1 = pd.read_csv(r'C:\Users\User user\Desktop\health.csv’)

x = df1["Height(m)"]
y = df1["Weight(kg)"]

plt.scatter(x,y)
plt.title("Health data of students")
plt.xlabel("Height(m)")
plt.ylabel("Weight(kg)")
plt.show()
```

Checklist

- Can you:
  1. Discuss Teachable Machine?
  2. Discuss data visualization?
  3. Display a frequency plot in Pandas?
  4. Plot a histogram or a density plot using data binning?
  5. Use a line graph?
  6. Make a scatter plot?