HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

Business Process Automation with VBA and Python

Mr. Eddie Chow / 22 March 2025

HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

Table Of Contents

Intended Learning Outcomes

1. explain the basic concepts and principles of web scraping, including the ethical considerations and legal implications involved in data extraction from websites.
2. demonstrate the ability to utilize Python libraries such as Beautiful Soup and Scrapy to efficiently extract, parse, and clean data from various web sources.
3. learn to visualize the scraped data using libraries like Matplotlib or Seaborn, enabling them to create informative graphs and charts that effectively communicate insights derived from the data.
4. apply data analysis techniques to interpret the visualized data, drawing conclusions that can inform decision-making processes in real-world scenarios, such as market analysis or competitive intelligence.

Agenda

1. Introduction to Data collection and Cleaning
2. ETL Data Processing
3. Machine Learning Framework
4. Introduction to Data Visualization
5. Data Visualization Tools and Technologies
6. Drawing Insights from Visualized Data
7. Best practices for presenting data visualizations

Data Collection - Reading API

Study the Weather API data dict Documentation in
https://www.hko.gov.hk/en/weatherAPI/doc/files/HKO_Open_Data_API_Documentation.pdf

1. Weather Information API

Dataset

- 9-day Weather Forecast
- Current Weather Report
- Local Weather Forecast
- Weather Warning Information
- Weather Warning Summary
- Special Weather Tips

API URL

https://data.weather.gov.hk/weatherAPI/opendata/weather.php

Please include valid parameters in API request. For valid parameters, please refer to Request table in this section.

HTTP Request Method

GET

Return Type

JSON

Request Example

https://data.weather.gov.hk/weatherAPI/opendata/weather.php?dataType=flw&lang=en

Parameter

1. **dataType**
2. **lang**

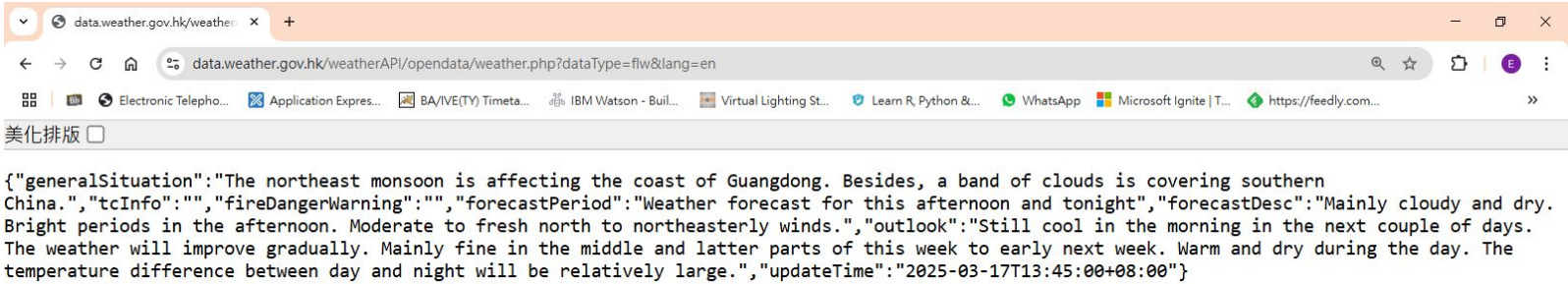
Request

Parameter	Accepted values	Description
dataType	flw	flw: Local Weather Forecast
	fnd	fnd: 9-day Weather Forecast
	rhrread	rhrread: Current Weather Report
	warnsum	warnsum: Weather Warning Summary
	warningInfo	warningInfo: Weather Warning Information
	swt	swt: Special Weather Tips
lang	en	en: English
	tc	tc: Traditional Chinese
	sc	sc: Simplified Chinese
		Default language: en

4

Data Collection - Reading API

When browser load https://data.weather.gov.hk/weatherAPI/opendata/weather.php?dataType=flw&lang=en



generalSituation	tcInfo	fireDangerWarning	forecastPeriod	forecastDesc	outlook	updateTime
":"The northeast monsoon is affecting the coast of Guangdong. Besides, a band of clouds is covering southern China			Weather forecast for this afternoon and tonight	Mainly cloudy and dry. Bright periods in the afternoon. Moderate to fresh north to northeasterly winds	Still cool in the morning in the next couple of days. The weather will improve gradually. Mainly fine in the middle and latter parts of this week to early next week. Warm and dry during the day. The temperature difference between day and night will be relatively large.	2025-03-17T13:45:00+08:00

Data Collection - Reading API

Case Study - Read Weather Forecasting from HKO API

Step 1. Import the required libraries:

```
#import necessary library
import requests
import pandas as pd
```

Step 2: Load the weather forecasting data from 'https://data.weather.gov.hk/weatherAPI/opendata/'

```
# Initialize Parameters for the API
datatype = 'fnd'
lang = 'en'
apiurl = f'https://data.weather.gov.hk/weatherAPI/opendata/weather.php?dataType={datatype}&lang={lang}'

# Fetching the JSON data
response = requests.get(apiurl)
response.raise_for_status()  # Check for request errors

# Converting JSON to a DataFrame
json_data = response.json()

# Convert JSON data to DataFrame
weatherforcast_df = pd.DataFrame(json_data['weatherForecast'])

# Display the DataFrame
print(weatherforcast_df.head())
```

```
  forecastDate       week                                forecastWind  \
0     20250115  Wednesday                            North force 4 to 5.
1     20250116   Thursday      North to northeast force 4, force 5 at first.
2     20250117     Friday             Northeast force 4, east force 5 later.
3     20250118   Saturday  East to northeast force 4 to 5, occasionally f...
4     20250119     Sunday                  East to northeast force 2 to 3.

                                forecastWeather  \
0  Mainly cloudy with a few light rain patches at...
1  Fine. Rather cool in the morning. Very dry dur...
2  Fine. Rather cool in the morning. Very dry dur...
3                          Dry with sunny periods.
4                            Mainly fine and dry.

             forecastMaxtemp             forecastMintemp  \
0  {'value': 20, 'unit': 'C'}  {'value': 15, 'unit': 'C'}
1  {'value': 19, 'unit': 'C'}  {'value': 13, 'unit': 'C'}
2  {'value': 19, 'unit': 'C'}  {'value': 13, 'unit': 'C'}
3  {'value': 20, 'unit': 'C'}  {'value': 15, 'unit': 'C'}
4  {'value': 21, 'unit': 'C'}  {'value': 16, 'unit': 'C'}

                    forecastMaxrh                    forecastMinrh  \
0  {'value': 80, 'unit': 'percent'}  {'value': 40, 'unit': 'percent'}
1  {'value': 65, 'unit': 'percent'}  {'value': 35, 'unit': 'percent'}
2  {'value': 65, 'unit': 'percent'}  {'value': 30, 'unit': 'percent'}
3  {'value': 70, 'unit': 'percent'}  {'value': 40, 'unit': 'percent'}
4  {'value': 70, 'unit': 'percent'}  {'value': 40, 'unit': 'percent'}

   ForecastIcon  PSR
0            51  Low
1            92  Low
2            50  Low
3            51  Low
4            50  Low
```

Data Collection - Reading API

#Continue from previous page

Step 3. Saving Weather Forecasting DataFrame to a CSV

filecsv_file = 'weatherForecast.csv' # Desired output CSV file
nameweatherforcast_df.to_csv(csv_file, index=False)

print(f"Data has been successfully saved to {csv_file}")

A	B	C	D	E	F	G	H	I	J
forecastDate	week	forecastWind	forecastWeather	forecastMaxtemp	forecastMintemp	forecastMaxrh	forecastMinrh	ForecastIcon	PSR
20250115	Wednesday	North force 4 to 5.	Mainly cloudy with a fe	{'value': 20, 'unit':	{'value': 15, 'unit':	{'value': 80, 'un	{'value': 40, 'un	51	Low
20250116	Thursday	North to northeast force	Fine. Rather cool in the r	{'value': 19, 'unit':	{'value': 13, 'unit':	{'value': 65, 'un	{'value': 35, 'un	92	Low
20250117	Friday	Northeast force 4, east	Fine. Rather cool in the r	{'value': 19, 'unit':	{'value': 13, 'unit':	{'value': 65, 'un	{'value': 30, 'un	50	Low
20250118	Saturday	East to northeast force	Dry with sunny periods.	{'value': 20, 'unit':	{'value': 15, 'unit':	{'value': 70, 'un	{'value': 40, 'un	51	Low
20250119	Sunday	East to northeast force	Mainly fine and dry.	{'value': 21, 'unit':	{'value': 16, 'unit':	{'value': 70, 'un	{'value': 40, 'un	50	Low
20250120	Monday	East to northeast force	Fine and dry.	{'value': 22, 'unit':	{'value': 16, 'unit':	{'value': 70, 'un	{'value': 40, 'un	50	Low
20250121	Tuesday	East to northeast force	Fine and dry.	{'value': 22, 'unit':	{'value': 17, 'unit':	{'value': 70, 'un	{'value': 40, 'un	50	Low
20250122	Wednesday	East force 4, occasiona	Sunny periods.	{'value': 21, 'unit':	{'value': 17, 'unit':	{'value': 75, 'un	{'value': 50, 'un	51	Low
20250123	Thursday	East force 4, occasiona	Sunny intervals.	{'value': 21, 'unit':	{'value': 17, 'unit':	{'value': 75, 'un	{'value': 50, 'un	52	Low

System Architecture

HK Weather Forecast

Data from API

weatherForecast.csv

Key: Date

HK Building Record

Data from Web Wrapping

hkbuildinginfo.csv

Key: District

HK School Net Number

Data from Manual Input

Primary Sch_Net Numbers By District.csv

Key: District

HK School Net Number CSV File

Data from Manual Input

District and POA School Net Look-up Table

https://www.edb.gov.hk/attachment/en/edu-system/primary-secondary/spa-systems/primary-1-admission/school-lists/District_Net_LookUpTable.pdf

區域 District	小一學校網編號 POA School Net Number
中西區 Central & Western District	11
灣仔 Wan Chai	12
北角/筲箕灣/柴灣 North Point/ Shau Kei Wan/ Chai Wan	14, 16
薄扶林/香港仔/黃竹坑/赤柱 Pok Fu Lam/ Aberdeen/ Wong Chuk Hang/ Stanley	18
油麻地/尖沙咀/旺角 Yau Ma Tei/ Tsim Sha Tsui/ Mong Kok	31, 32
何文田/紅磡/九龍城 Ho Man Tin/ Hung Hom/ Kowloon City	34, 35, 41
深水埗 Sham Shui Po	40
黃大仙 Wong Tai Sin	43, 45
九龍灣/牛頭角/觀塘/油塘 Kowloon Bay/ Ngau Tau Kok/ Kwun Tong/ Yau Tong	46, 48
荃灣 Tsuen Wan	62
葵涌/荔景/大窩口/青衣 Kwai Chung/ Lai King/ Tai Wo Hau/ Tsing Yi	64, 65, 66
屯門 Tuen Mun	70, 71
天水圍/元朗 Tin Shui Wai/ Yuen Long	72, 73, 74
上水/粉嶺/北區 Sheung Shui/ Fanling/ North District	80, 81, 83
大埔 Tai Po	84
沙田 Shatin	88, 89, 91
西貢/將軍澳 Sai Kung/ Tseung Kwan O	95
離島/東涌 Islands/ Tung Chung	96, 97, 98, 99

Combining several CSV File

This Tasks is to combining Building Information and School Net Numbers

Step 1. Import the required libraries:

#Loading orginal CSV file containing property data:

```
hkbuildingdata = pd.read_csv('hkbuildinginfo.csv', encoding='ISO-8859-1')
hkbuildingdata.head(5)
```

Step 2: #Clean up the data

```
# Check for missing values
print("\nMissing values in each column:")
print(hkbuildingdata.isnull().sum())

#Handle missing values by Dropping rows with missing values
# For columns with a high percentage of missing values, consider dropping or filling them
# Here we will drop 'block' and 'state' columns due to excessive missing values
hkbuildingdata.drop(columns=['withpre', 'rootid', 'fatherid', 'catid', 'source', 'block', 'state', 'date_dm'],
        inplace=True, errors='ignore')

# Assuming 'floor' is the column causing the error
hkbuildingdata['floor'] = hkbuildingdata['floor'].astype(str)  # Convert to string type

# Now you can safely use the .str accessor
hkbuildingdata['floor'] = hkbuildingdata['floor'].str.replace(r'[^\d]', '', regex=True)  # Remove
        non-numeric characters

# Convert back to numeric if needed
hkbuildingdata['floor'] = pd.to_numeric(hkbuildingdata['floor'], errors='coerce')  # Convert to numeric,
        coercing errors to NaN
```

```
Missing values in each column:
date                  0
withpre           23564
id                    0
rootid                0
fatherid              0
catid                 0
catname               0
catfathername         0
url_father            0
url_cat               0
source            11556
contract              0
memo                  0
price                 0
price_value           0
holddate              0
winloss_flag          0
winloss               0
act_area              0
area                  0
arearaw               4
sq_price              0
sq_price_value        0
sq_actprice           0
sq_actprice_value     0
month                 0
day                   0
date_dm               0
date_y                0
block             32308
state             77003
floor                32
room                258
addr                  0
Unnamed: 34       80000
dtype: int64
```

Combining several CSV File

#Continue from previous page

```
# Continue with your existing code for filling missing values and other operations
hkbuildingdata['floor'] = hkbuildingdata['floor'].fillna(hkbuildingdata['floor'].median())  # Fill with median for numeric data

# Step 4: Fill missing values for 'floor' and 'room'
hkbuildingdata['floor'] = hkbuildingdata['floor'].fillna(hkbuildingdata['floor'].median())  # Fill with median for numeric data
hkbuildingdata['room'] = hkbuildingdata['room'].fillna(hkbuildingdata['room'].mode()[0])   # Fill with mode for categorical data

# Step 5: Fill missing values for 'arearaw'
hkbuildingdata['arearaw'] = hkbuildingdata['arearaw'].fillna(hkbuildingdata['arearaw'].mean())

# Step 6: Rename columns to more meaningful names
hkbuildingdata.rename(columns={'floor': 'Floor', 'arearaw': 'House Area', 'sq_price_value': 'Price Per Square', 'catname': 'Building
        Name', 'catfathername': 'District'}, inplace=True)

# Step 7: Display the cleaned DataFrame and check for remaining missing values
print("\nCleaned DataFrame:")
print(hkbuildingdata.head())

print("\nRemaining Missing Values:")
print(hkbuildingdata.isnull().sum())
```

```
Cleaned DataFrame:
         date      id       Building Name       District  \
0  2020-11-27  683333     Bel Air Heights   Diamond Hill
1  2020-11-27  683332       Fa Yuen Plaza       Mong Kok
2  2020-11-27  683331      Caldecott Hill  Yau Yat Tsuen
3  2020-11-27  683330    Pang Ching Court   Wong Tai Sin
4  2020-11-27  683329  Metro Harbour View   Tai Kok Tsui

                                  url_father  \
0   https://data.28hse.com/en/kl/diamond-hill
1       https://data.28hse.com/en/kl/mong-kok
2  https://data.28hse.com/en/kl/yau-yat-tsuen
3   https://data.28hse.com/en/kl/wong-tai-sin
4   https://data.28hse.com/en/kl/tai-kok-tsui

                                        url_cat   contract  \
0  https://data.28hse.com/en/kl/diamond-hill/2058...  Agreement
1  https://data.28hse.com/en/kl/mong-kok/2712_fa-...  Agreement
2  https://data.28hse.com/en/kl/yau-yat-tsuen/309...  Agreement
3  https://data.28hse.com/en/kl/wong-tai-sin/5035...  Agreement
4  https://data.28hse.com/en/kl/tai-kok-tsui/2564...  Agreement

            memo   price  price_value  ... Price Per Square  sq_actprice  \
0  20112702480015    $10M     10000000  ...         12106.54       $15221
1  20112702470044  $4.28M      4280000  ...         12334.29       $20000
2  20112702440018    $11M     11000000  ...          8580.34       $11100
3  20112702410017   $4.9M      4900000  ...          8404.80           --
4  20112702400032  $7.15M      7150000  ...         12016.81       $17354

   sq_actprice_value month day  date_y Floor  room                addr  \
0           15220.70   Nov  27    2020   2.0     E  BLOCK 2 2#/F Room E
1           20000.00   Nov  27    2020   1.0     C          1#/F Room C
2           11099.90   Nov  27    2020   7.0     E   TOWER 1 7/F Room E
3               0.00   Nov  27    2020   1.0    22         1#/F Room 22
4           17354.37   Nov  27    2020   1.0     B  BLOCK 8 1#/F Room B
```

Combining Building Information and School Net Numbers

#Combine Original Data with School District Data

```
schoolnetnumber_df = pd.read_csv("Primary Sch_Net Numbers By District.csv")

# Normalize the 'District' column in both DataFrames to lowercase
hkbuildingdata['District'] = hkbuildingdata['District'].str.lower()
schoolnetnumber_df['District'] = schoolnetnumber_df['District Areas'].str.lower()

# Merge schooldistrict_df with districtcode_df on the normalized 'District' column
merged_df = hkbuildingdata.merge(schoolnetnumber_df[['District', 'Sch_Net_No']], on='District', how='left')

# Create the 'District_Code' column based on the logic provided
hkbuildingdata['School_Netnumber'] = merged_df['Sch_Net_No'].fillna(0).astype(int)
hkbuildingdata
```

[27]:

	date	id	Building Name	District	url_father	url_cat	contract	memo	price	price_va
0	2020-11-27	683333	Bel Air Heights	diamond hill	https://data.28hse.com/en/kl/diamond-hill	https://data.28hse.com/en/kl/diamond-hill/2058...	Agreement	20112702480015	$10M	10000(
1	2020-11-27	683332	Fa Yuen Plaza	mong kok	https://data.28hse.com/en/kl/mong-kok	https://data.28hse.com/en/kl/mong-kok/2712_fa-...	Agreement	20112702470044	$4.28M	4280(
2	2020-11-27	683331	Caldecott Hill	yau yat tsuen	https://data.28hse.com/en/kl/yau-yat-tsuen	https://data.28hse.com/en/kl/yau-yat-tsuen/309...	Agreement	20112702440018	$11M	11000(
3	2020-11-27	683330	Pang Ching Court	wong tai sin	https://data.28hse.com/en/kl/wong-tai-sin	https://data.28hse.com/en/kl/wong-tai-sin/5035...	Agreement	20112702410017	$4.9M	4900(
4	2020-11-27	683329	Metro Harbour View	tai kok tsui	https://data.28hse.com/en/kl/tai-kok-tsui	https://data.28hse.com/en/kl/tai-kok-tsui/2564...	Agreement	20112702400032	$7.15M	7150(

5 rows × 29 columns

Exploratory Data Analysis(ETL) and Data Visualization

Exploratory Data Analysis and Visualization

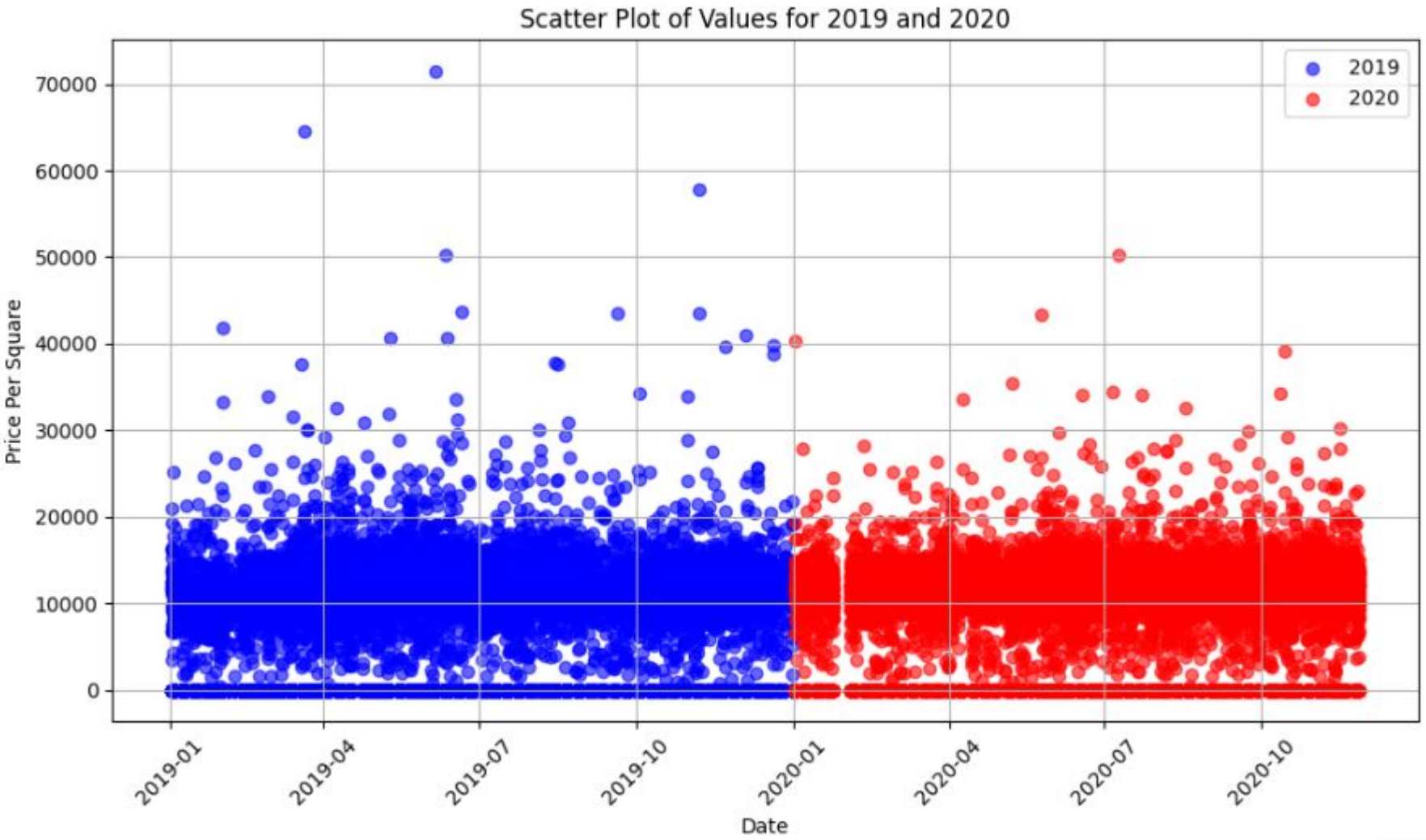
```
#filter the data with period "2020-1-1" to "2020-12-31"
# Create a full date column from 'month', 'day', and 'date_y'
hkbuildingdata['full_date'] = pd.to_datetime(hkbuildingdata['month'] + ' ' + hkbuildingdata['day'].astype(str) + ' ' +
      hkbuildingdata['date_y'].astype(str))

# Filter the DataFrame for the date range "2020-01-01" to "2020-12-31"
start_date = '2020-01-01'
end_date = '2020-12-31'
covidfiltered_data = hkbuildingdata[(hkbuildingdata['full_date'] >= start_date) & (hkbuildingdata['full_date'] <= end_date)]

# Display the first few rows of the filtered DataFrame
print(covidfiltered_data.head(5))

# Filter the DataFrame for the date range "2019-
start_date = '2019-01-01'
end_date = '2019-12-31'
filtered_data = hkbuildingdata[(hkbuildingdata['fu

# Display the first few rows of the filtered DataFr
print(filtered_data.head(5))
```

```
         date      id       Building Name       District  \
0  2020-11-27  683333     Bel Air Heights   diamond hill
1  2020-11-27  683332       Fa Yuen Plaza       mong kok
2  2020-11-27  683331      Caldecott Hill  yau yat tsuen
3  2020-11-27  683330    Pang Ching Court   wong tai sin
4  2020-11-27  683329  Metro Harbour View   tai kok tsui

                                 url_father  \
0   https://data.28hse.com/en/kl/diamond-hill
1       https://data.28hse.com/en/kl/mong-kok
2  https://data.28hse.com/en/kl/yau-yat-tsuen
3   https://data.28hse.com/en/kl/wong-tai-sin
4   https://data.28hse.com/en/kl/tai-kok-tsui

                                          url_cat   contract  \
0  https://data.28hse.com/en/kl/diamond-hill/2058...  Agreement
1  https://data.28hse.com/en/kl/mong-kok/2712_fa-...  Agreement
2  https://data.28hse.com/en/kl/yau-yat-tsuen/309...  Agreement
3  https://data.28hse.com/en/kl/wong-tai-sin/5035...  Agreement
4  https://data.28hse.com/en/kl/tai-kok-tsui/2564...  Agreement
```

Exploratory Data Analysis(ETL) and Visualization

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a scatter plot
plt.figure(figsize=(10, 6))

# Plot for 2019
plt.scatter(filtered_data['full_date'], filtered_data['Price Per Square'], color='blue', label='2019', alpha=0.6)

# Plot for 2020
plt.scatter(covidfiltered_data['full_date'], covidfiltered_data['Price Per Square'], color='red', label='2020', alpha=0.6)

# Adding titles and labels
plt.title('Scatter Plot of Values for 2019 and 2020')
plt.xlabel('Date')
plt.ylabel('Price Per Square')
plt.xticks(rotation=45)
plt.legend()
plt.grid()

# Show the plot
plt.tight_layout()
plt.show()
```

Exploratory Data Analysis(ETL) and Visualization



Practice 4 – File Processing with DATA.GOV.HK data

- Read the CSV file
- Loop through the data and do some counting

資料一線通 DATA.GOV.HK

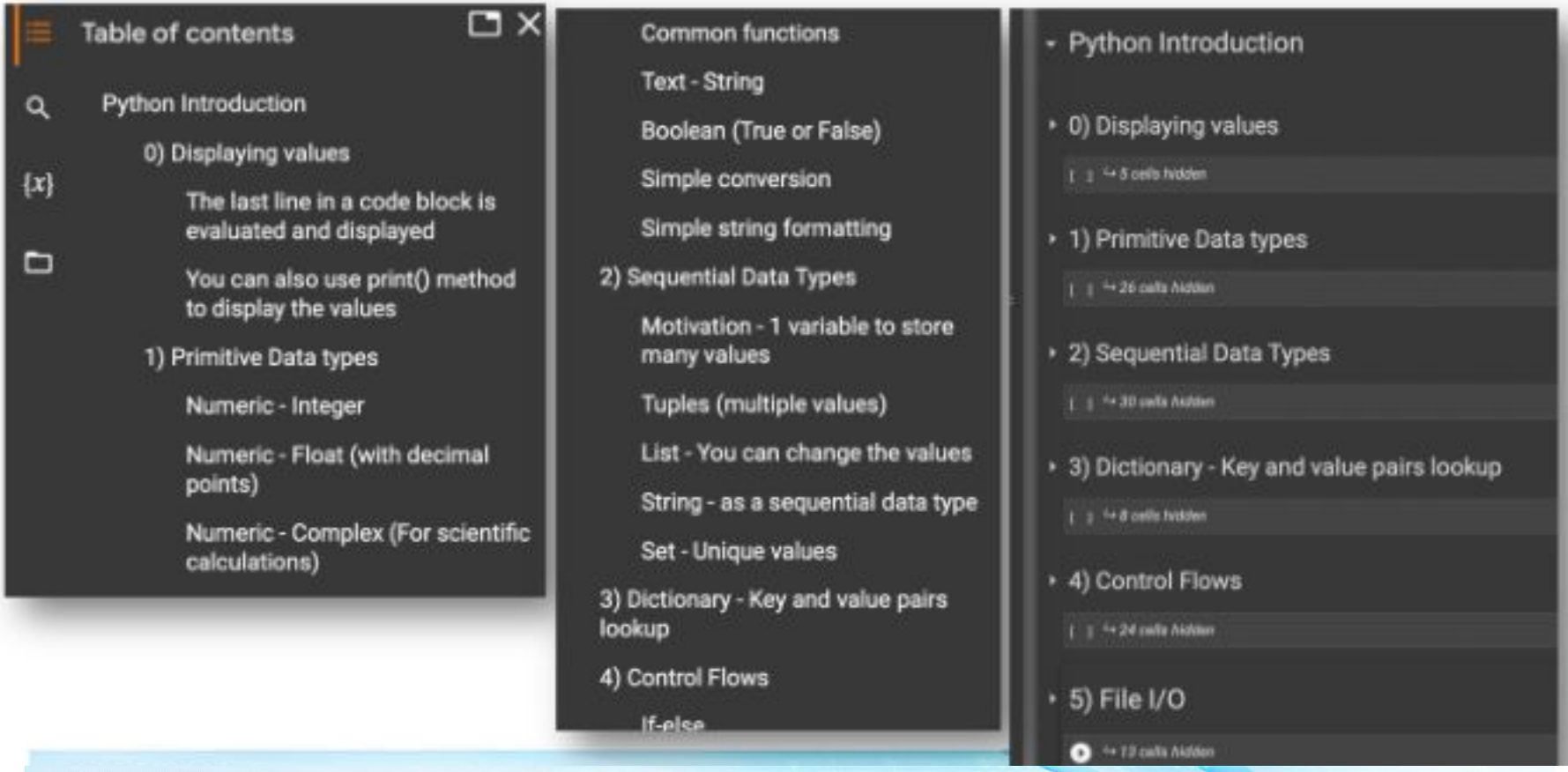
	A	B	C	D	E
1	Account	Expenditure Group	Subhead	Item	Actual($)
2	RECURRENT	PERSONNAL EMOLUMENTS	SALARIES	SALARIES	151,760,978
3	RECURRENT	PERSONNAL EMOLUMENTS	ALLOWANCES	OVERTIME	507,492
4	RECURRENT	PERSONNAL EMOLUMENTS	ALLOWANCES	ACTING ALLOWANCES	5,636,585
5	RECURRENT	PERSONNEL RELATED EXPENSES	MANDATORY PROVIDENT FUND CONTRIBUTION	CONTRIBUTION FOR RECRUITS UNDER THE NEW TERMS BEFORE CONFIRMATION	265,855
6	RECURRENT	PERSONNEL RELATED EXPENSES	CIVIL SERVICE PROVIDENT FUND CONTRIBUTION	MANDATORY CONTRIBUTION UNDER THE MANDATORY PROVIDENT FUND SCHEMES ORDINANCE	1,468,649
7	RECURRENT	PERSONNEL RELATED EXPENSES	CIVIL SERVICE PROVIDENT FUND CONTRIBUTION	GOVERNMENT VOLUNTARY CONTRIBUTION	9,534,616
8	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	CLOTHES AND UNIFORMS	21,940
9	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	CLEANING MATERIALS	107,532
10	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	OFFICE STATIONERY AND MATERIALS	672,661
11	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	GENERAL PUBLICATIONS, PERIODICALS AND JOURNALS	123,429
12	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	OFFICE FURNITURE AND EQUIPMENT(INCLUDING RENTAL)	445,709
13	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	PAPERS	53,221
14	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	INFORMATION AND COMMUNICATIONS TECHNOLOGY	3,299,560
15	RECURRENT	DEPARTMENAL EXPENSES	LIGHT AND POWER	ELECTRICITY	422,305

- **Starting notebook:**

  https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice4_file_IO_gov_hk_202409.ipynb

Notebook Summary

https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice2_word_count_202409.ipynb

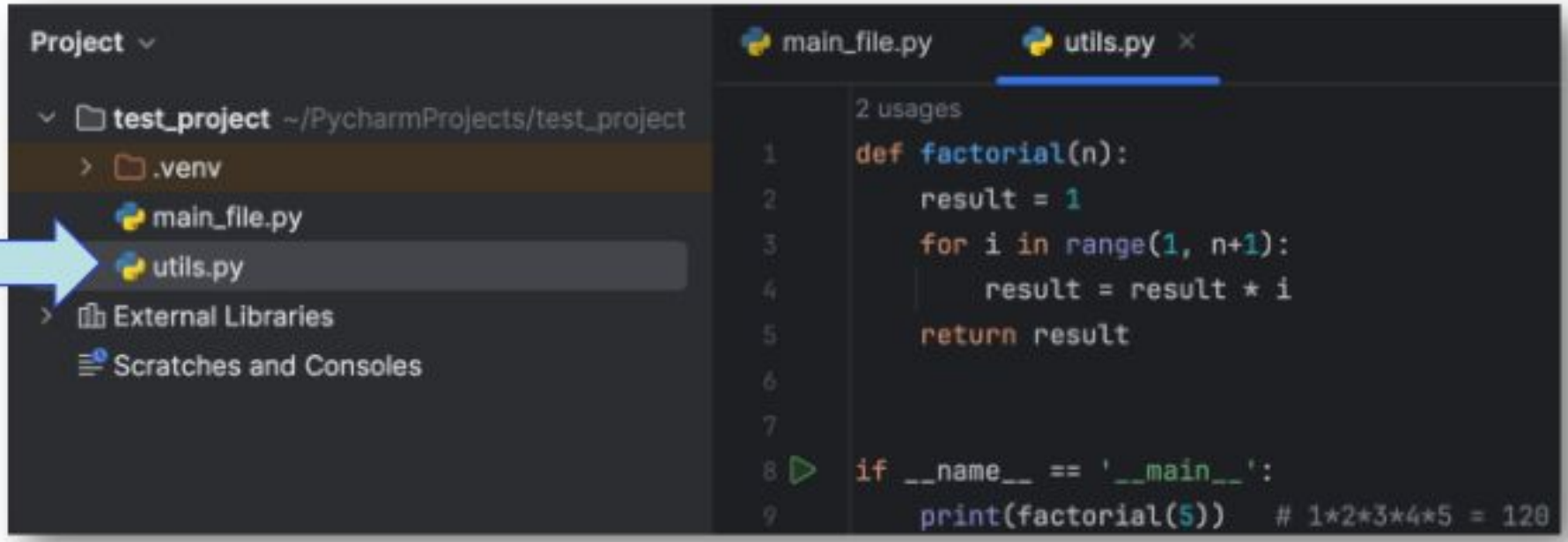


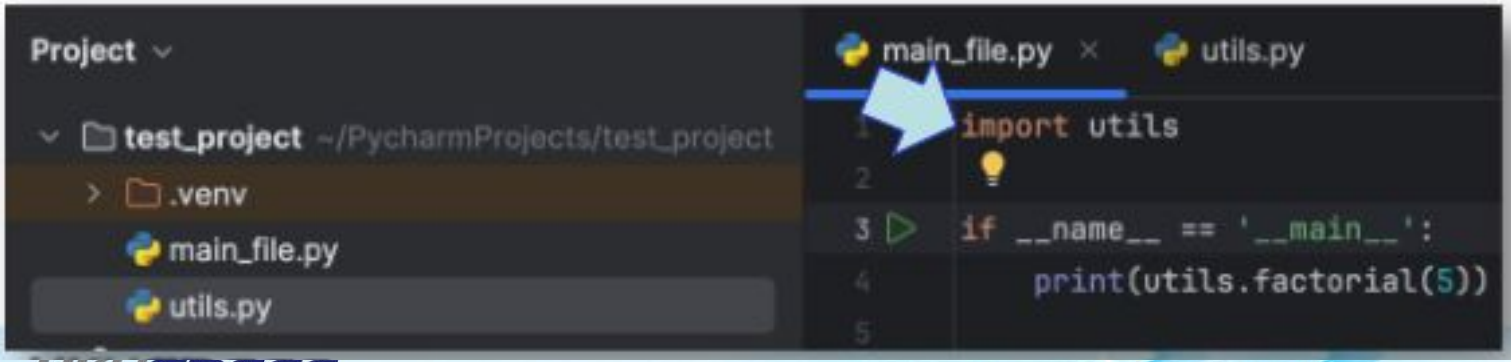
Practice 4 - Review

- It takes so much effort to read CSV files
  - Q: But this should be a common task?
  - Any instant noodles for me to “add hot water & eat”?

Business Process Automation with Python

Importing Modules





Python Architecture (Python runtime + Packages)

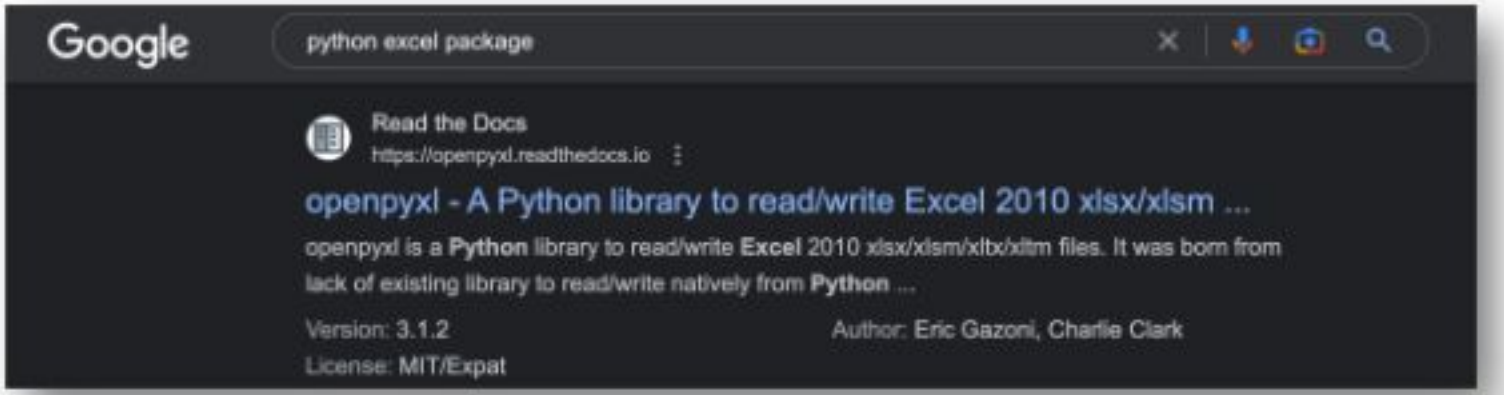
- Packages
  - Built-in / published codes which can be imported to use
  - e.g., **datetime** module ➔ **date** class ➔ **strftime** function

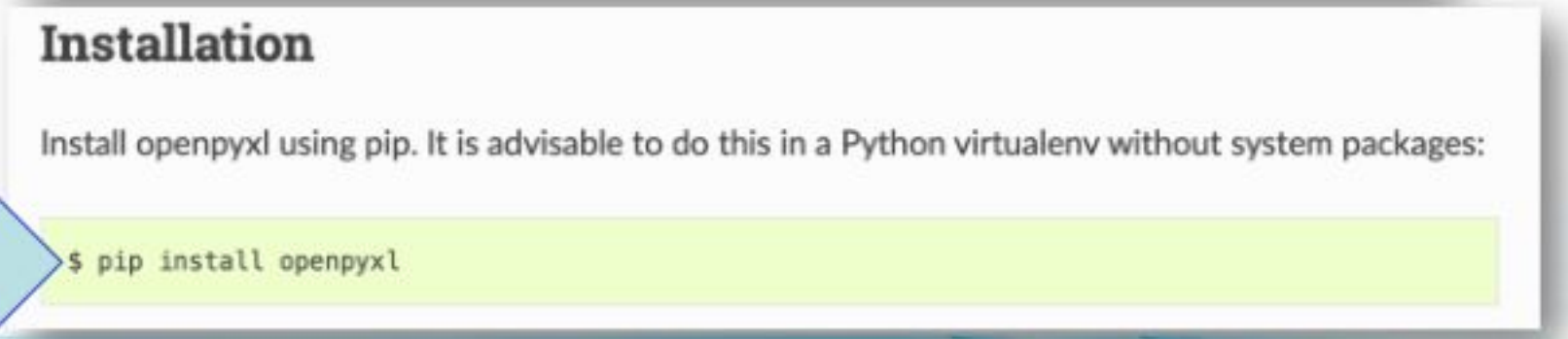
```
import datetime

dt = datetime.date(2024, 12, 25)
print(dt.strftime('%Y-%m-%d'))
print(dt.strftime('%d %B %Y (%A)'))
```

```
2024-12-25
25 December 2024 (Wednesday)
```

Packages - Installation

- Default installer – pip (Python 3.4 or later)
  - Search for “Python” + (something you need) + “Package”
  - Run “pip install xxx” in command prompt / terminal





Packages - Installation

- In Jupyter Notebooks
  - Add "!" ➔ Install system-wide
  - Add "%" ➔ Install within notebook environment

```
!pip install pandas
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (1.5.3)
Requirement already satisfied: numpy>=1.20.3 in /usr/local/lib/python3.9/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
%pip install pandas
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (1.5.3)
Requirement already satisfied: numpy>=1.20.3 in /usr/local/lib/python3.9/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

Exploratory data analysis (EDA)

Exploring your data is a crucial step in data analysis. It involves:

Organising the data set

Plotting aspects of the data set

Maybe producing some numerical summaries; central tendency and spread, etc.

"Exploratory data analysis can never be the whole story, but nothing else can serve as the foundation stone."
- John Tukey.

Exploratory Data Analysis (EDA)

Why?

- EDA encompasses the "*explore* data" part of the data science process
- EDA is crucial but often overlooked:
  - If your data is bad, your results will be bad
  - Conversely, understanding your data well can help you create smart, appropriate models

Exploratory Data Analysis (EDA)

What?

1. Store data in data structure(s) that will be convenient for exploring/processing
   (Memory is fast. Storage is slow)
2. Clean/format the data so that:
   - Each row represents a single object/observation/entry
   - Each column represents an attribute/property/feature of that entry
   - Values are numeric whenever possible
   - Columns contain atomic properties that cannot be further decomposed*

* Unlike food waste, which can be composted.
  Please consider composting food scraps.

Exploratory Data Analysis (EDA)

What? (continued)

3. Explore **global** properties: use histograms, scatter plots, and aggregation functions to summarize the data
4. Explore **group** properties: group like-items together to compare subsets of the data (are the comparison results reasonable/expected?)

This process transforms your data into a format which is easier to work with, gives you a basic overview of the data's properties, and likely generates several questions for you to follow-up in subsequent analysis.

EDA: without Pandas

Say we have a small dataset of the top 50 most-streamed Spotify songs, globally, for 2019.

EDA: without Pandas

Say we have a small dataset of the top 50 most-streamed Spotify songs, globally, for 2019.

NOTE: The following music data are used purely for illustrative, educational purposes. The data, including song titles, may include explicit language. Harvard, including myself and the rest of the CS109 staff, does not endorse any of the entailed contents or the songs themselves, and we apologize if it is offensive to anyone in anyway.

EDA: without Pandas

top50.csv

Each row represents a distinct song. The columns are:

- **ID:** a unique ID (i.e., 1-50)
- **TrackName:** Name of the Track
- **ArtistName:** Name of the Artist
- **Genre:** the genre of the track
- **BeatsPerMinute:** The tempo of the song.
- **Energy:** The energy of a song - the higher the value, the more energetic.
- **Danceability**: The higher the value, the easier it is to dance to this song.
- **Loudness**: The higher the value, the louder the song.
- **Liveness**: The higher the value, the more likely the song is a live recording.
- **Valence**: The higher the value, the more positive mood for the song.
- **Length**: The duration of the song (in seconds).
- **Acousticness**: The higher the value, the more acoustic the song is.
- **Speechiness**: The higher the value, the more spoken words the song contains.
- **Popularity**: The higher the value, the more popular the song is.

EDA: without Pandas

top50.csv

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

-
-
-

Q1: What are some ways we can store this file into data structure(s) using regular Python (not the Pandas library).

EDA: without Pandas

top50.c

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

Possible Solution #1: A 2D array (i.e., matrix)

Weaknesses:

- What are the row and column names? Need separate lists for them – clumsy.
- Lists are O(N). We'd need 2 dictionaries just for column names

```
data = [][]
col_name
-> index
index ->
col_name
```

EDA: without Pandas

top50.csv

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

-
-
-

Possible Solution #2: A list of dictionaries

list

Item 1	=	{“Track.Name”: “Senorita”, “Artist.Name”: “Shawn Mendes”, “Genre”: “Canadian pop”, …}
Item 2	=	{“Track.Name”: “China”, “Artist.Name”: “Anuel AA”, “Genre”: “reggaetón flow”, … }
Item 3	=	{“Track.Name”: “Ariana Grande”, “Artist.Name”: “boyfriend”, “Genre”: “dance pop”, … }

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

```
f = open("../data/top50.csv", encoding = "ISO-8859-1")
column_names = f.readline().strip().split(",")[1:] # puts names in a list
cleaned_column_names = [name[1:-1] for name in column_names]
cleaned_column_names.insert(0, "ID")

dataset = []

# iterates through each line of the .csv file
for line in f:
    attributes = line.strip().split(",")

    # constructs a new dictionary for each line
    dataset.append(dict(zip(cleaned_column_names, attributes)))
```

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q2: Write code to print all songs (Artist and Track name) that are longer than 4 minutes (240 seconds):

```
for song in dataset:
    if int(song["Length."]) > 240:
        print(song["Artist.Name"], "-", song["Track.Name"], "is", song["Length."],"seconds long")
```

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q3: Write code to print the most popular song (artist and track) – if ties, show all ties.

```
max_score = -1
most_populars = set()
for song in dataset:
    if int(song["Popularity"]) > max_score:
        most_populars = set([str(song["Artist.Name"] + "-" + song["Track.Name"])])
        max_score = int(song["Popularity"])
    elif int(song["Popularity"]) == max_score:
        most_populars.add(str(song["Artist.Name"] + "-" + song["Track.Name"]))
print(most_populars)
```

EDA: list of dictionaries

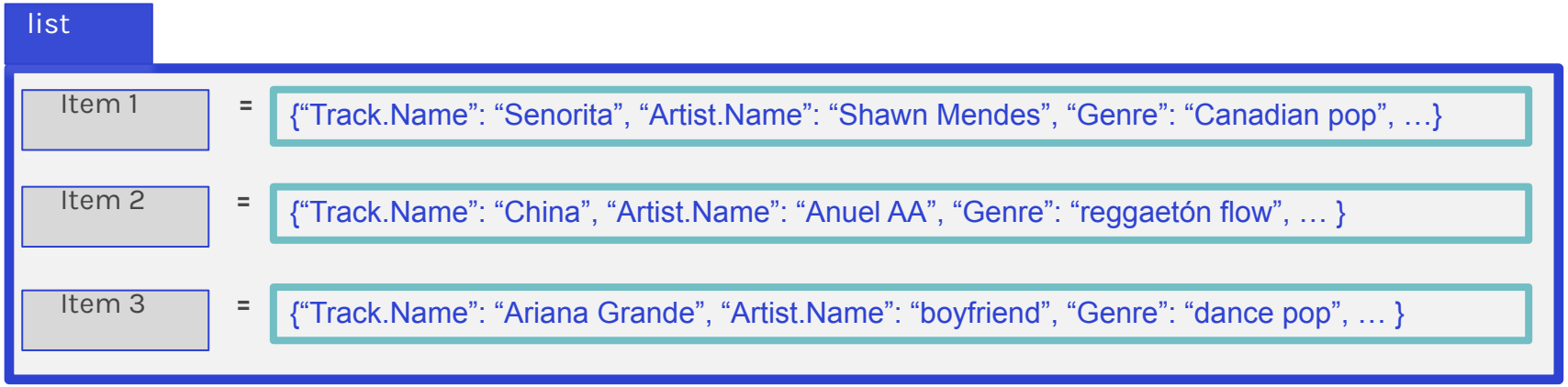
Possible Solution #2: A list of dictionaries

Q4: Write code to print the songs (and their attributes), if we sorted by their popularity (highest scoring ones first).

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q4: Write code to print the songs (and their attributes), if we sorted by their popularity (highest scoring ones first).

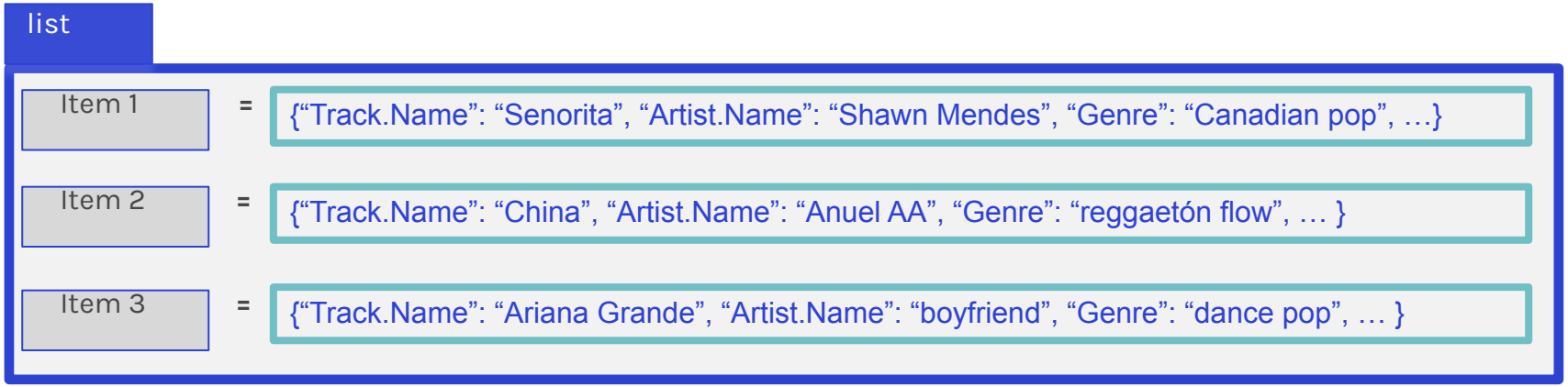


Cumbersome to move dictionaries around in a list. Problematic even if we don’t move the dictionaries.

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q5: How could you check for null/empty entries? This is only 50 entries. Imagine if we had 500,000.



EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q6: Imagine we had another table* below (i.e., .csv file). How could we combine its data with our already-existing *dataset*?

spotify_aux.

	TrackName	ArtistName	ExplicitLanguage	
1	Senorita	Shawn Menc	TRUE	
2	China	Anuel AA	FALSE	
3	boyfriend (w	Ariana Granc	TRUE	
4	Beautiful Pec	Ed Sheeran	FALSE	

* 3rd column is made-up by me. Random values. Pretend they're accurate.

EDA: with Pandas!

Kung Fu Panda is property of DreamWorks and Paramount Pictures

Lecture Outline

- Exploratory Data Analysis (EDA):
  - Without Pandas (part 1) – These slides
  - With Pandas (part 2) – Mostly Jupyter Notebook
- Data concerns (part 3) – These slides
- Web Scraping with Beautiful Soup (part 4) – Mix

EDA: with Pandas

What / Why?

- Pandas is an *open-source* Python library (anyone can contribute)
- Allows for high-performance, easy-to-use data structures and data analysis
- Unlike NumPy library which provides multi-dimensional arrays, Pandas provides 2D table object called **DataFrame**
  (akin to a spreadsheet with column names and row labels).
- Used by *a lot* of people

EDA: with Pandas

How

- import **pandas** library (convenient to rename it)
- Use **read_csv()** function

```
import pandas as pd
dataframe = pd.read_csv("yourfile.csv")
```

Common Panda functions

High-level viewing:

- head() – first N observations
- tail() – last N observations
- columns() – names of the columns
- describe() – statistics of the quantitative data
- dtypes() – the data types of the columns

Common Panda functions

Accessing/processing:

- df[“column_name”]
- Df.column_name
- .max(), .min(), .idxmax(), .idxmin()
- <dataframe> <conditional statement>
- .loc[] – label-based accessing
- .iloc[] – index-based accessing
- .sort_values()
- .isnull(), .notnull()

Common Panda functions

Grouping/Splitting/Aggregating:

- groupby(), .get_groups()
- .merge()
- .concat()
- .aggegate()
- .append()

Lecture Outline

- Exploratory Data Analysis (EDA):
  - Without Pandas (part 1) – These slides
  - With Pandas (part 2) – Mostly Jupyter Notebook
- Data concerns (part 3) – These slides
- Web Scraping with Beautiful Soup (part 4) – Mix

Data Concerns

When determining if a dataset is sound to use, it can be useful to think about these four questions:

- Did it come from a trustworthy, authoritative source?
- Is the data a complete sample?
- Does the data seem correct?
- **(optional)** Is the data stored efficiently or does it have redundancies?

Data Concerns: the format

- Often times, there may not exist a single dataset that contains all of the information we are interested in.
- May need to merge existing datasets
- Important to do so in a sound and efficient format

Data Concerns: the format

For example, say we have two datasets:

Dataset 1

Top 200 most-frequent streams per day (for June 2019)

SpotifySongID, # of Streams, Date

200

2789179, 42003, 06-01

3819390, 89103, 06-01

200

4492014, 52923, 06-02

8593013, 189145, 06-02

6,000 x 3

Dataset 2

Top 50 most streamed in 2019, so far

SpotifySongID, Artist, Track, [10 acoustic features]

50

2789179, Billie Eilish, bad guy, 3.2, 5.9, ...

3901829, Outkast, Elevators, 9.3, 5,1, ...

50 x 13

Data Concerns: the format

For example, say we have two datasets:

Dataset 1

Top 200 most-frequent streams per day (for June 2019)

SpotifySongID, # of Streams, Date

200

2789179, 42003, 06-01

▪

▪

3819390, 89103, 06-01

200

4492014, 52923, 06-02

▪

▪

8593013, 189145, 06-02

6,000 x 3

Dataset 2

Top 50 most streamed in 2019, so far

SpotifySongID, Artist, Track, [10 acoustic features]

50

2789179, Billie Eilish, bad guy, 3.2, 5.9, …

▪

▪

3901829, Outkast, Elevators, 9.3, 5,1, …

50 x 13

We are interested in determining if songs with high danceability are more popular during the weekends of June than weekdays in June. What should our merged table look like? Concerns?

Data Concerns: the format

This is wasteful, as it has 10 acoustic features, artist, and track repeated many times for each unique song.

Datasets Merged (poorly)

SpotifySongID, # of Streams, Date,Artist, Track, [10 acoustic features]

200

2789179, 42003, 06-01 Billie Eilish, bad guy, 3.2, 5.9, …

3819390, 89103, 06-01 Outkast, Elevators, 9.3, 5,1, …

200

4492014, 52923, 06-02

8593013, 189145, 06-02

6,000 x 15 □ 90,000 cells

Data Concerns: the format

Some rows may have null values for # of Streams (if the song wasn't popular in June)

Datasets Merged (better)

SpotifySongID, Artist, Track, [10 acoustic features], 06-01 Streams, 06-02 Streams

50

2789179, Billie Eilish, bad guy, 3.2, 5.9, ..., 42003, 42831, 43919

⋮

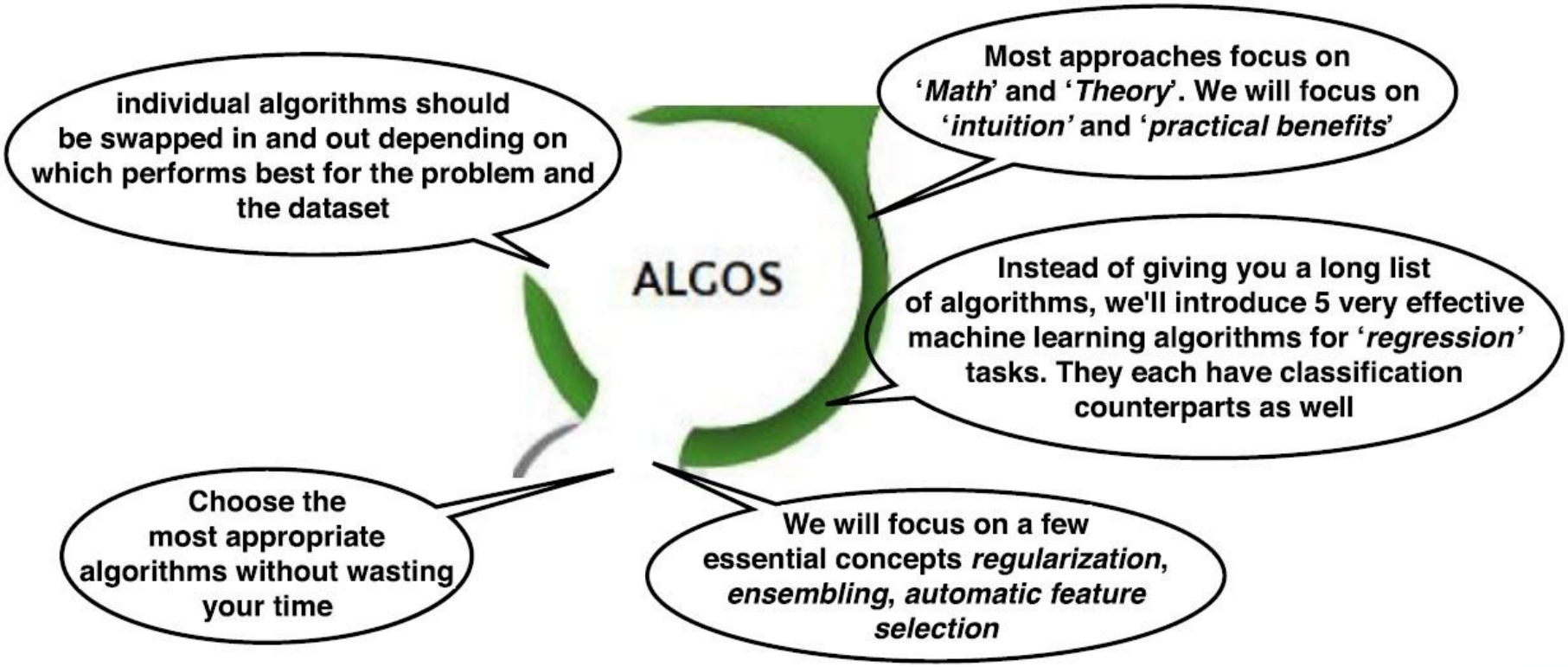
3901829, Outkast, Elevators, 9.3, 5,1, ... 29109, 27193, 25982

50 x 70 □ 3,500 cells

Data Concerns: the format

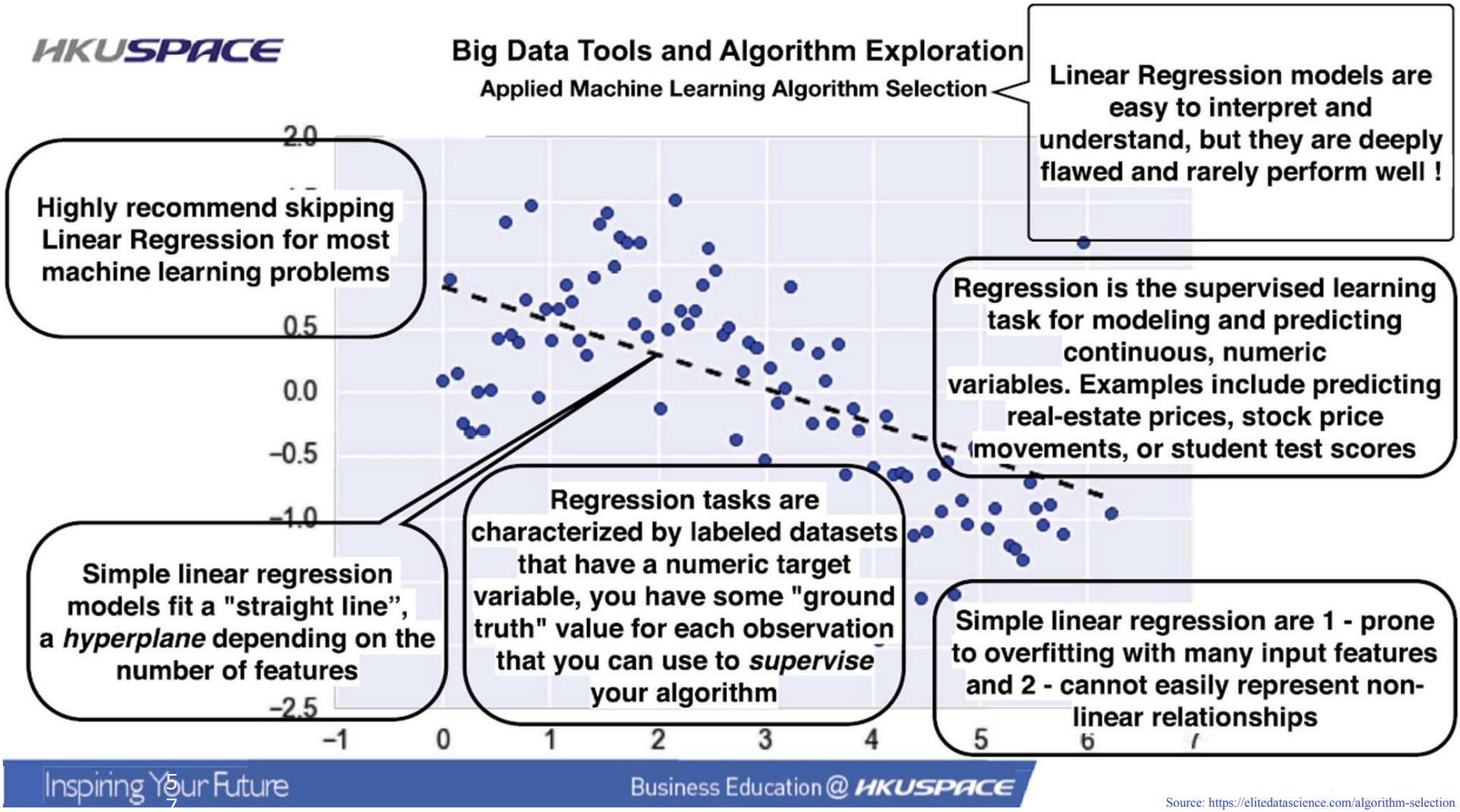
- Is the data correctly constructed (or are values wrong)?
- Is there redundant data in our merged table?
- Missing values?

Big Data Tools and Algorithm Exploration
Applied Machine Learning Algorithm Selection Overview



Source: https://elitedatascience.com/algorithm-selection

Big Data Tools and Algorithm Exploration
Applied Machine Learning Algorithm Selection



Source: https://elitedatascience.com/algorithm-selection

Big Data Tools and Algorithm Exploration
Applied Machine Learning Algorithm Selection

Random forests train a large number of "strong" decision trees and combine their predictions through bagging

Each tree is only trained on a random subset of observations (a process called resampling)

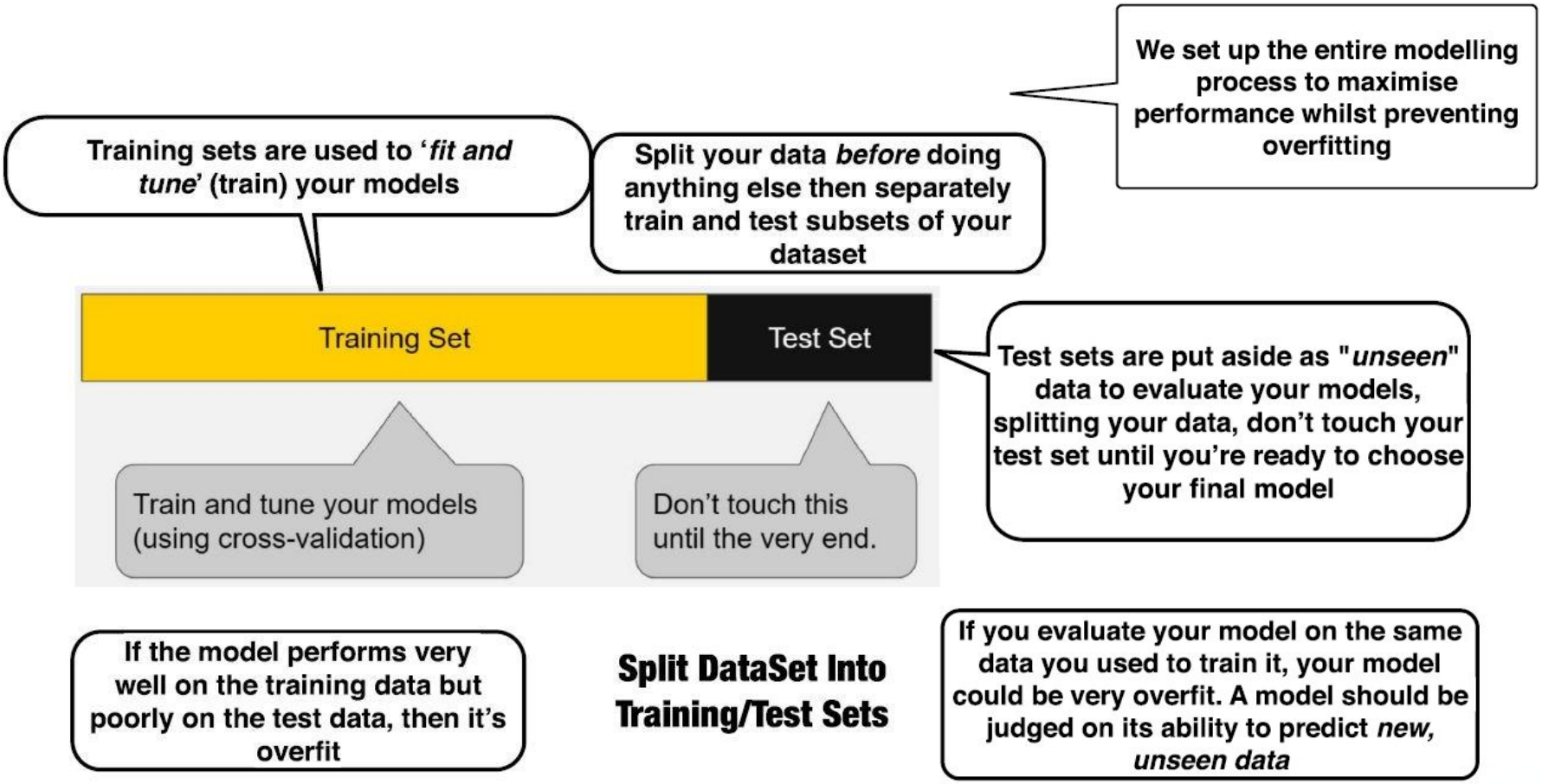
Random Forests

Each tree is only allowed to choose from a random subset of features to split on (leading to feature selection)

random forests tend to perform very well right out of the box often beat many other models that take up to weeks to develop, and are the perfect "*swiss-army-knife*" algorithm that almost always gets good results

Source: https://elitedatascience.com/algorithm-selection

Big Data Tools and Algorithm Exploration
Model Training

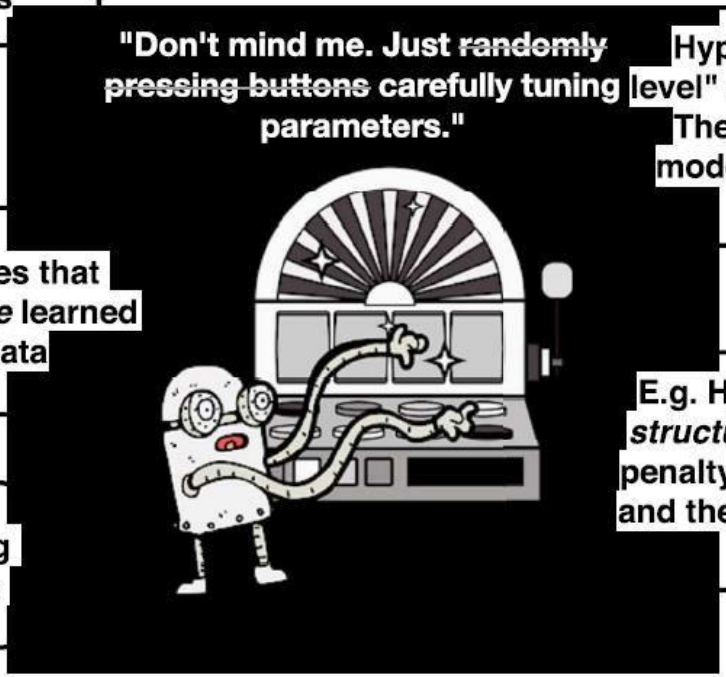


Source: https://elitedatascience.com/model-training

Big Data Tools and Algorithm Exploration
Model Training

Hyperparameters: When we talk of tuning models, we specifically mean tuning hyperparameters

Examples of Model parameters include regression coefficients, decision tree split locations



Hyperparameters express "higher-level" *structural settings* for algorithms. They are decided before fitting the model because they can't be learned from the data

Model parameters are attributes that define *individual models, and are* learned directly from the training data

E.g. Hyperparameters "higher-level" *structural settings* are strength of the penalty used in regularized regression and the number of trees to include in a random forest

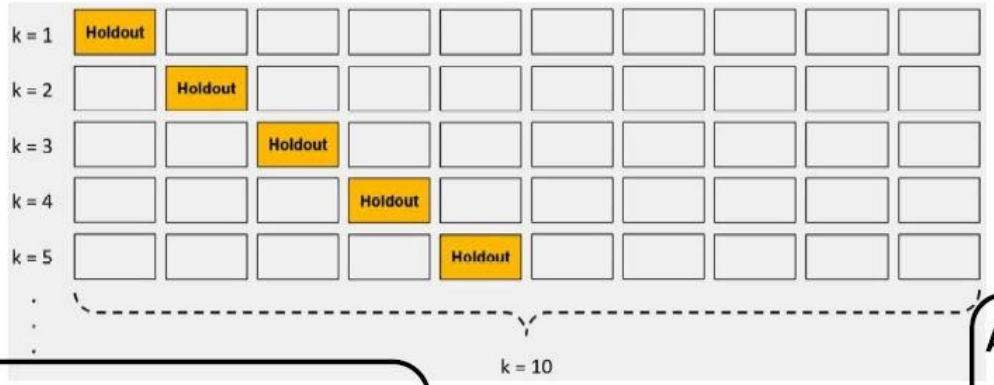
There are two types of parameters in machine learning algorithms: Model Parameters and hyperparameters

Source: https://elitedatascience.com/model-training

Big Data Tools and Algorithm Exploration
Model Training

Cross-validation is a tuning method for getting a reliable estimate of model performance using only training data

1- Split your data into 10 equal parts, or "folds", 2- Train your model on 9 folds (e.g. the first 9 folds)



3 - Evaluate it on the 1 remaining "hold-out" fold, Perform steps (2) and (3) 10 times, each time holding out a different fold

10-fold cross-validation, breaks your training data into 10 equal parts (or *folds*), essentially creating 10 miniature train/test splits

Average the performance across all 10 hold-out folds, average performance across the 10 hold-out folds is your final performance estimate, also called your *cross-validated* score

Source: https://elitedatascience.com/model-training

Big Data Tools and Algorithm Exploration
Model Training

We've split our dataset into training and test sets, and we've learned about hyperparameters and cross-validation, we're ready fit and tune our models

perform the entire cross-validation loop detailed above on each set of hyperparameter values using a high-level pseudo-code

```
For each algorithm (i.e. regularized regression, random forest, etc.):
  For each set of hyperparameter values to try:
    Perform cross-validation using the training set.
    Calculate cross-validated score.
```

At the end of this process, you will have a cross-validated score for each set of hyperparameter values... for each algorithm

Elastic-Net		
Penalty Ratio	Penalty Strength	CV-Score
75/25	0.01	0.63
75/25	0.05	0.64
75/25	0.10	**0.67**
50/50	0.01	0.62
50/50	0.05	0.63
50/50	0.10	0.66

Source: https://elitedatascience.com/model-training

Big Data Tools and Algorithm Exploration
Model Training

By now, you'll have 1 "best" model *for each algorithm* that has been tuned through cross-validation. Most importantly, you've only used the training data so far

Keep the set of hyperparameter values with best cross-validated score. Re-train the algorithm on the entire training set (without cross-validation)

For classification tasks, we recommend Area Under ROC Curve (AUROC). *Higher values are better*

Because you've saved your test set as a truly unseen dataset, you can now use it get a reliable estimate of each models' performance.There are a variety of performance metrics you could choose from

For regression tasks, we recommend Mean Squared Error (MSE) or Mean Absolute Error (MAE). *(Lower values are better)*

like the Hunger Games... each algorithm sends its own "representatives" (i.e. model trained on the best set of hyperparameter values) to the final selection

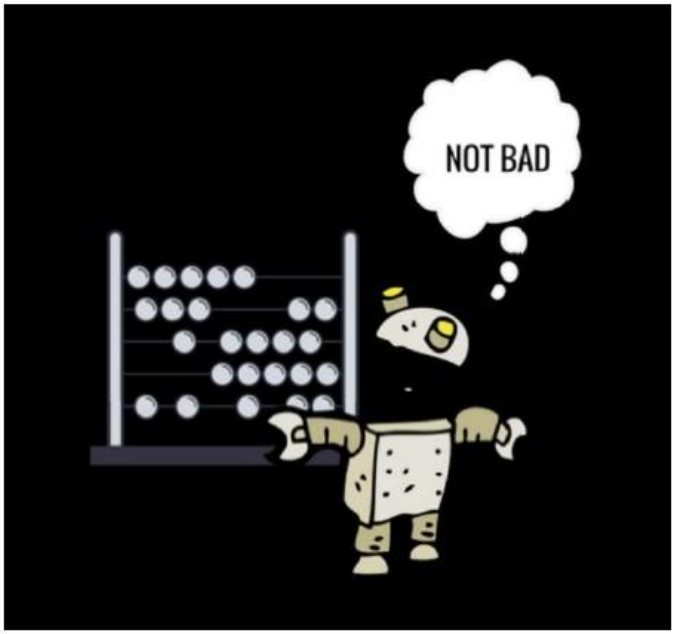
Source: https://elitedatascience.com/model-training

Big Data Tools and Algorithm Exploration
Model Training Conclusion

Pick the winning model by asking: Which model had the best performance on the test set? (performance)

Calculate performance metrics using those predictions and the "ground truth" target variable from the test set

For each of your models, make predictions on your test set



Pick the winning model by asking: Does it perform well across various performance metrics? (robustness)

Pick the winning model by asking: Did it also have (one of) the best cross-validated scores from the training set? (consistency)

Pick the winning model by asking: Does it solve the original business problem? (win condition)

Source: https://elitedatascience.com/model-training

Conduct Random Forest Model training

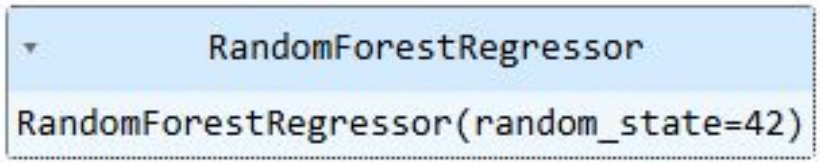
#This code is going to train an random forest model to find top important features

```
#Step 1 : Import necessary library
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import plot_tree
import joblib  # Import joblib for saving the model

#Step 2 : import data for machine learning model training
data = pd.read_csv('combined_building_withschool.csv')

# #Step 3 : Machine Learning Model Implementation
# Encode categorical variables
label_encoder = LabelEncoder()
data['District'] = label_encoder.fit_transform(data['District'])
data['School_Netnumber'] = label_encoder.fit_transform(data['School_Netnumber'])

# Define features and target variable
X = data[['District', 'Floor', 'School_Netnumber']]  # Features
y = data['Price Per Square']  # Target variable

# Check the distribution of the target variable
print(y.value_counts())
print(X)
```

```
Price Per Square
0.00        30508
10000.00      217
8333.33       109
8000.00        81
12500.00       70
             ...
8554.05          1
13764.04         1
10443.86         1
15421.30         1
4672.90          1
Name: count, Length: 31331, dtype: int64
       District  Floor  School_Netnumber
0             1    2.0                 8
1            16    1.0                 2
2            31    7.0                 5
3            28    1.0                 7
4            24    1.0                 2
...         ...    ...               ...
79995         6    2.0                 9
79996         9    9.0                 6
79997         3    4.0                 4
79998         6    1.0                 9
79999        26    3.0                 1

[80000 rows x 3 columns]
```

Conduct Random Forest Model training

```
# Step 3: Split the data into training and testing sets with stratification
# Ensure the target variable is continuous
print(f"Target variable type: {y.dtype}")

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Train the Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Step 6: Display a decision tree from the Random Forest with reduced size
plt.figure(figsize=(6, 6))  # Reduced by 10%
plot_tree(rf_model.estimators_[5], feature_names=X.columns.tolist(), filled=True)
plt.title("Decision Tree from Random Forest")
plt.show()
```



Conduct Random Forest Model training

```
# Display feature importances
importances = rf_model.feature_importances_
feature_names = X.columns.tolist()
feature_importances = pd.DataFrame({'Feature': feature_names, 'Importance':
    importances})

# Sort by importance and display the top 5 features
top_features = feature_importances.sort_values(by='Importance',
    ascending=False).head(5)
print("Top 5 Features and their Importance Scores:")
print(top_features)

# Step 5: Save the trained model to a file
model_filename = 'housing_valuation_model.joblib'  # Specify the filename
joblib.dump(rf_model, model_filename)  # Save the model
print(f'Model saved to {model_filename}')
```


```
Top 5 Features and their Importance Scores:
            Feature  Importance
0          District    0.728459
2  School_Netnumber    0.199190
1             Floor    0.072351
```

Financial Forecasting with Machine Learning using Python (Numpy, Pandas, Matplotlib and Scikit-learn)

Open Google Colab

Step 1. Import the required libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing  import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from keras.models  import Sequential
from keras.layers import Dense, LSTM
import requests
```

Step 2: Load the finance data from https://www.alphavantage.co:

```
api_key = 'demo‘
symbol = 'IBM‘
url =
      f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY_ADJUSTED&symbol={symbol}&outputsize=f
      ull&apikey={api_key}‘
response = requests.get(url)
data = response.json()
df = pd.DataFrame(data['Time Series (Daily)']).transpose()
df.index = pd.to_datetime(df.index)
df = df.sort_index()
```

Financial Forecasting with Machine Learning using Python (Numpy, Pandas, Matplotlib and Scikit-learn)

Step 3 – Preprocess the data:

```
# Extract the closing prices
y = df['4. close'].values.astype(float)
# Normalize the closing prices
scaler = MinMaxScaler(feature_range=(0, 1))
y = scaler.fit_transform(y.reshape(-1, 1))
# Create the feature matrix
X = []
for i in range(60, len(df)):
      X.append(y[i-60:i, 0])
X = np.array(X)# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y[60:], test_size=0.2, shuffle=False)
```

Step 4 – Define the model:

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

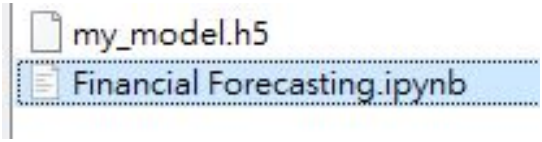
Financial Forecasting with Machine Learning using Python (Numpy, Pandas, Matplotlib and Scikit-learn)

Step 5 – Train the model with 100 steps:

```
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_val, y_val))
```

Step 6 – Save the model in h5 format:

```
model.save('my_model.h5')  # Save the model in HDF5 format
```

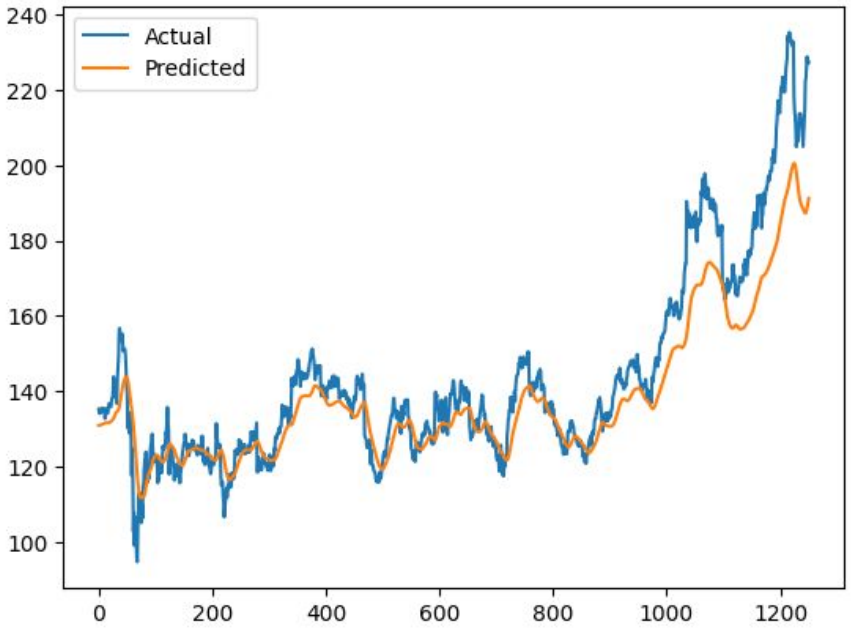


Step 7 – Evaluate the model:

```
# Evaluate the model on the validation set
y_pred = model.predict(X_val)
rmse = np.sqrt(mean_squared_error(y_val, y_pred))
print('Root Mean Squared Error:', rmse)
```

Step 8 – Visualize the results:

```
# Visualize the results
y_pred = scaler.inverse_transform(y_pred)
y_val = scaler.inverse_transform(y_val)
plt.plot(y_val, label='Actual')
plt.plot(y_pred, label='Predicted')
plt.legend()
plt.show()
```



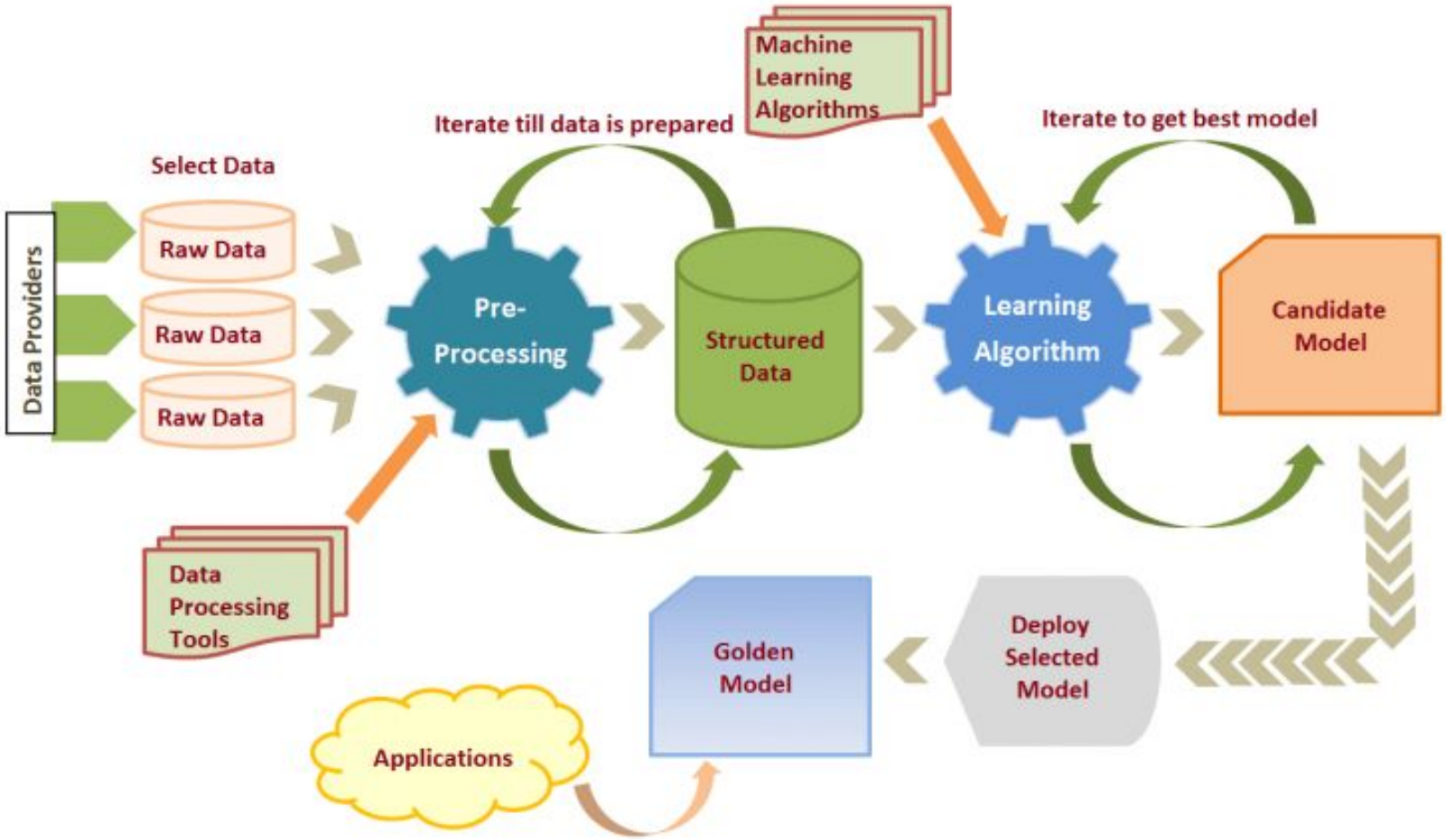
Financial Forecasting with Machine Learning using Python (Numpy, Pandas, Matplotlib and Scikit-learn)

Step 9 – Make predictions:

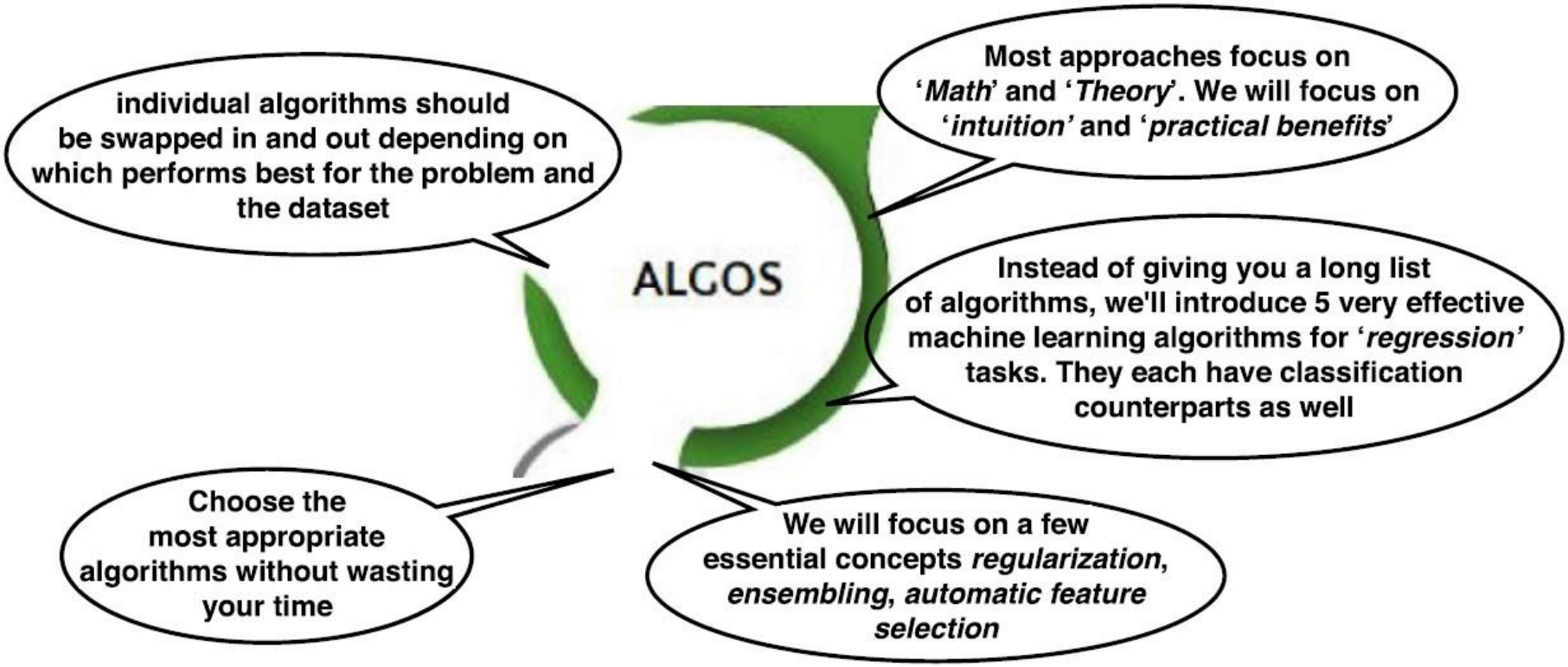
```
last_60_days = y[-60:]
last_60_days_scaled = scaler.transform(last_60_days.reshape(-1, 1))
X_test = []
X_test.append(last_60_days_scaled)
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
y_pred = model.predict(X_test)
y_pred = scaler.inverse_transform(y_pred)
print('Predicted price:', y_pred[0][0])
```

```
1/1 [==============================] - 1s 675ms/step
Predicted price: 40.51613
```

Applied Machine Learning Pipeline Overview

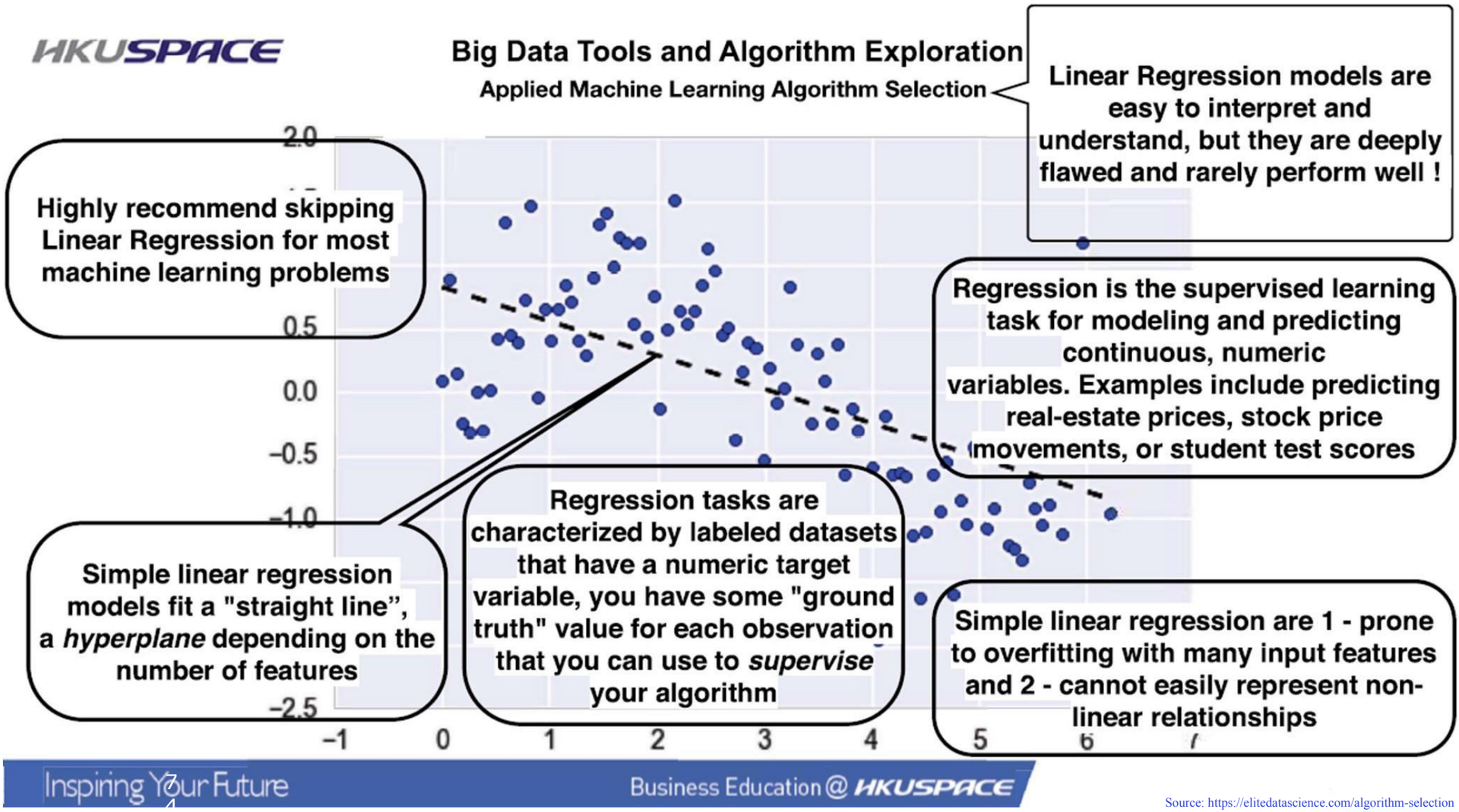


Applied Machine Learning Algorithm Selection Overview



Source: https://elitedatascience.com/algorithm-selection

Applied Machine Learning Algorithm Selection



Source: https://elitedatascience.com/algorithm-selection

Applied Machine Learning Algorithm Selection

Random forests train a large number of "strong" decision trees and combine their predictions through bagging

Each tree is only trained on a random subset of observations (a process called resampling)

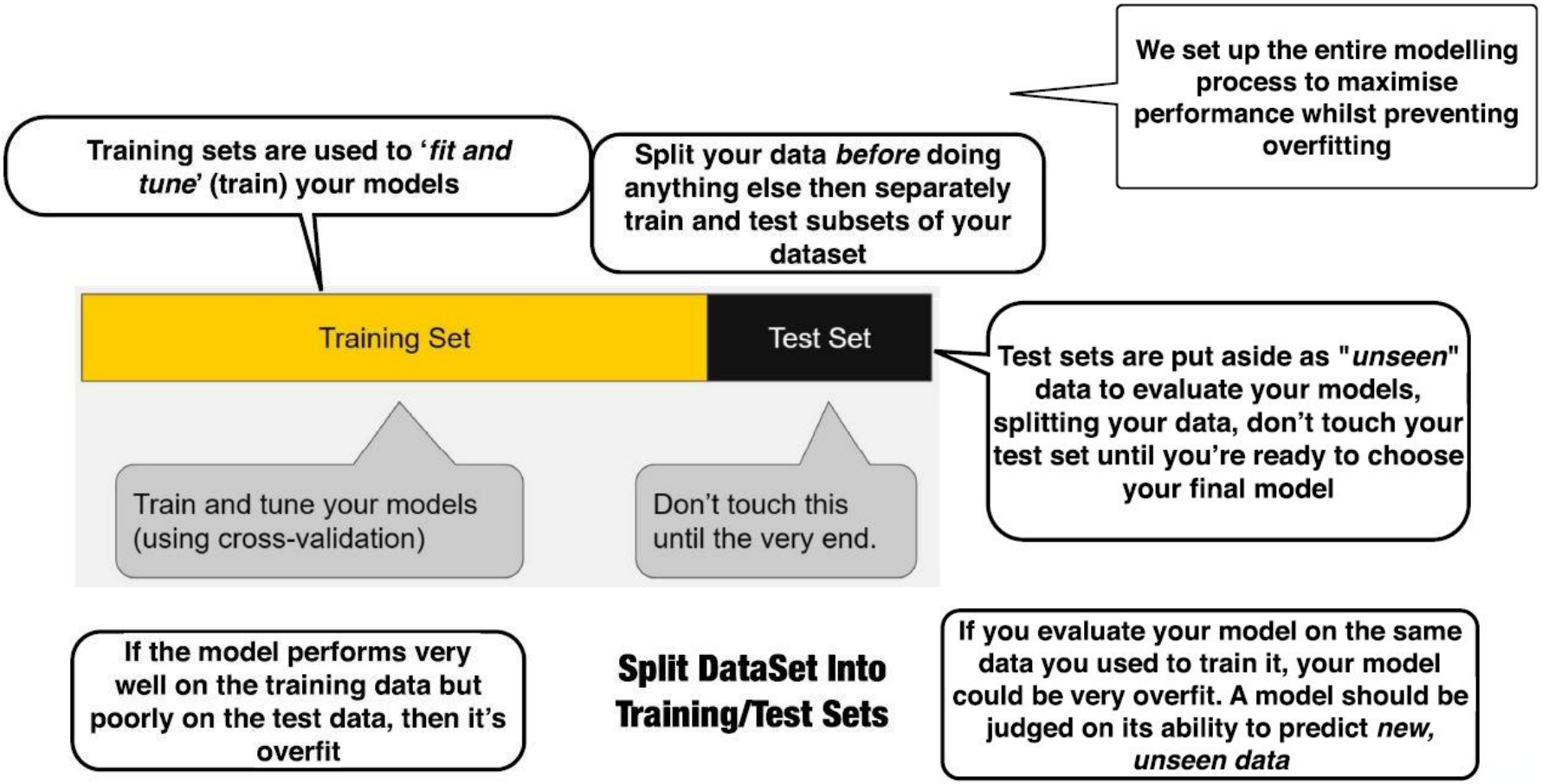
Random Forests

Each tree is only allowed to choose from a random subset of features to split on (leading to feature selection)

random forests tend to perform very well right out of the box often beat many other models that take up to weeks to develop, and are the perfect "*swiss-army-knife*" algorithm that almost always gets good results

Source: https://elitedatascience.com/algorithm-selection

Machine Learning Model Training

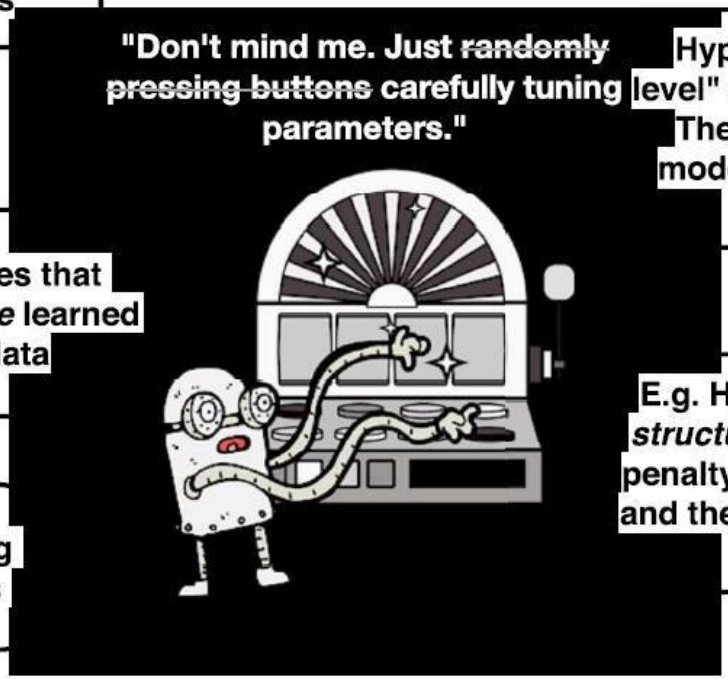


Source: https://elitedatascience.com/model-training

Machine Learning Model Training

Examples of Model parameters include regression coefficients, decision tree split locations

Hyperparameters: When we talk of tuning models, we specifically mean tuning hyperparameters



Hyperparameters express "higher-level" *structural settings* for algorithms. They are decided before fitting the model because they can't be learned from the data

Model parameters are attributes that define *individual models, and are* learned directly from the training data

E.g. Hyperparameters "higher-level" *structural settings* are strength of the penalty used in regularized regression and the number of trees to include in a random forest

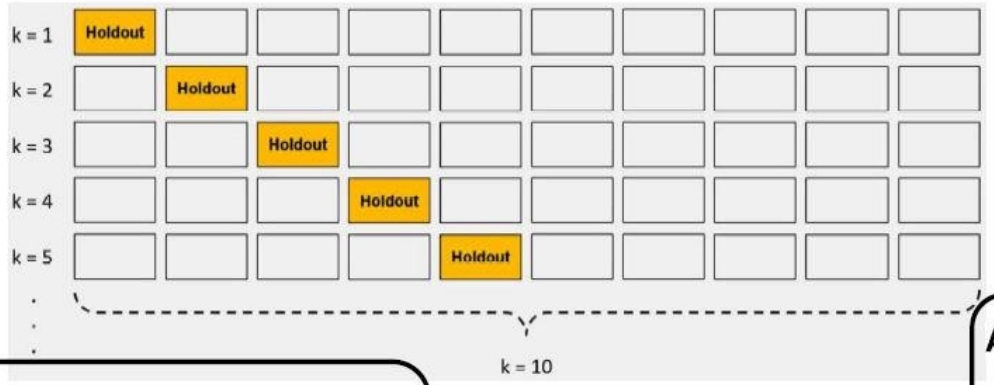
There are two types of parameters in machine learning algorithms: Model Parameters and hyperparameters

Source: https://elitedatascience.com/model-training

Machine Learning Model Training

Cross-validation is a tuning method for getting a reliable estimate of model performance using only training data

1- Split your data into 10 equal parts, or "folds", 2- Train your model on 9 folds (e.g. the first 9 folds)



3 - Evaluate it on the 1 remaining "hold-out" fold, Perform steps (2) and (3) 10 times, each time holding out a different fold

10-fold cross-validation, breaks your training data into 10 equal parts (or *folds*), essentially creating 10 miniature train/test splits

Average the performance across all 10 hold-out folds, average performance across the 10 hold-out folds is your final performance estimate, also called your *cross-validated* score

Source: https://elitedatascience.com/model-training

Machine Learning Model Training

We've split our dataset into training and test sets, and we've learned about hyperparameters and cross-validation, we're ready fit and tune our models

perform the entire cross-validation loop detailed above on each set of hyperparameter values using a high-level pseudo-code

```
For each algorithm (i.e. regularized regression, random forest, etc.):
  For each set of hyperparameter values to try:
    Perform cross-validation using the training set.
    Calculate cross-validated score.
```

At the end of this process, you will have a cross-validated score for each set of hyperparameter values... for each algorithm

Elastic-Net		
Penalty Ratio	Penalty Strength	CV-Score
75/25	0.01	0.63
75/25	0.05	0.64
75/25	0.10	**0.67**
50/50	0.01	0.62
50/50	0.05	0.63
50/50	0.10	0.66

Source: https://elitedatascience.com/model-training

Machine Learning Model Training

By now, you'll have 1 "best" model *for each algorithm* that has been tuned through cross-validation. Most importantly, you've only used the training data so far

Keep the set of hyperparameter values with best cross-validated score. Re-train the algorithm on the entire training set (without cross-validation)

For classification tasks, we recommend Area Under ROC Curve (AUROC). *Higher values are better*

Because you've saved your test set as a truly unseen dataset, you can now use it get a reliable estimate of each models' performance.There are a variety of performance metrics you could choose from

For regression tasks, we recommend Mean Squared Error (MSE) or Mean Absolute Error (MAE). *(Lower values are better)*

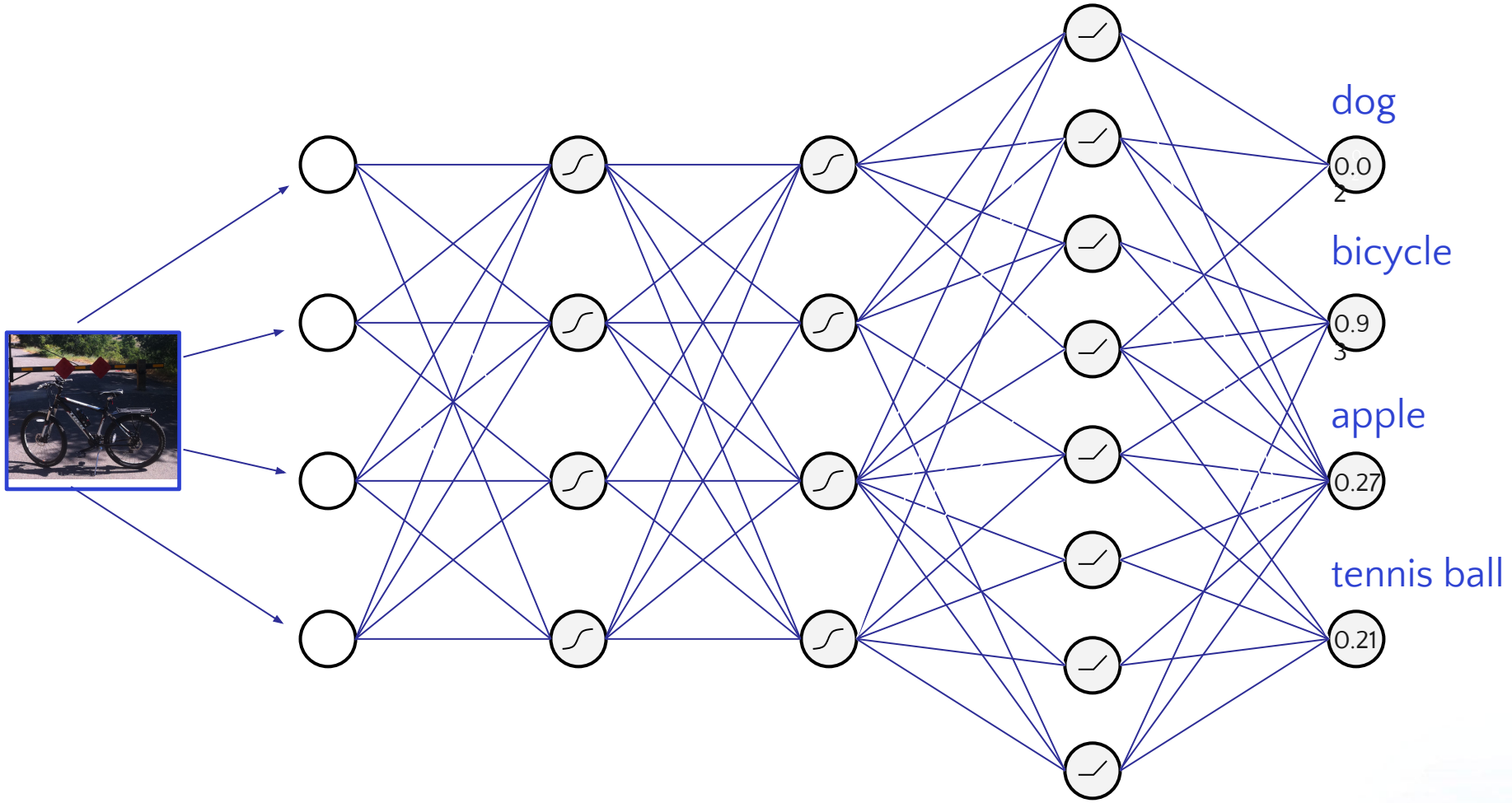
like the Hunger Games... each algorithm sends its own "representatives" (i.e. model trained on the best set of hyperparameter values) to the final selection

Source: https://elitedatascience.com/model-training

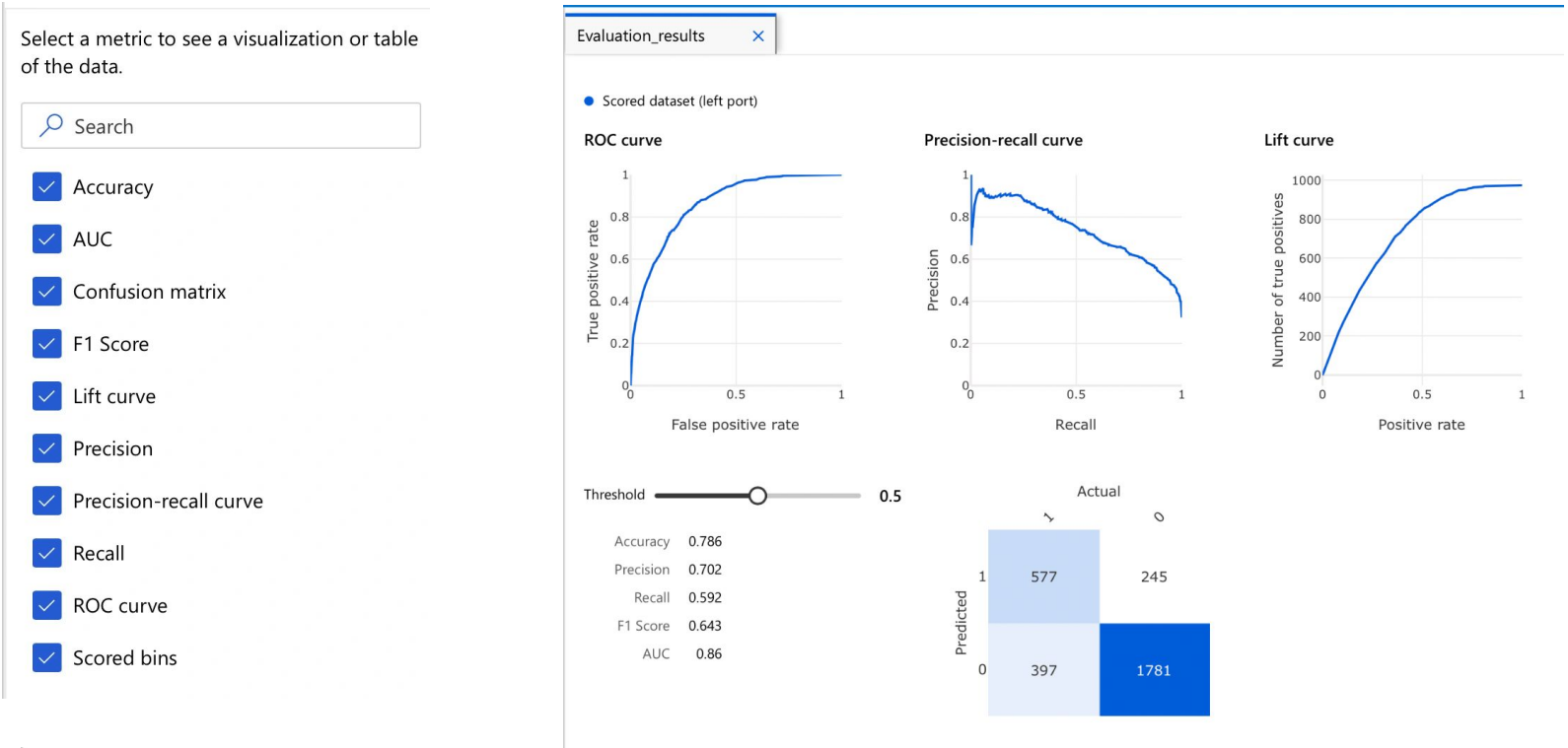
Modelling Techniques

NLP	Machine Learning	Deep Learning
Word2vec	Linear Regression	Deep Neural Network - DNN
Fasttext	Neural Network - NN	Convolutional Neural Network - CNN
Bert	SVM	Region CNN - R-CNN
elmo	Xgboost	Long-short term memory - LSTM
BoW	Random forest	Recurrent neural networks - RNN
Flair	Artificial Neural Network - ANN	Autoregressive integrated moving average - arima

Trained Convolutional NN



Machine Learning Model Training Visulisation



Score bin ↓	Positive exam...	Negative exam...	Fraction above thres...	Accura...	F1 Score	Precisi...	Recall	Negative precis...	Negative recall	Cumulative AUC
(0.900,1.000]	95	11	0.035	0.703	0.176	0.896	0.098	0.696	0.995	0.000
(0.800,0.900]	149	21	0.092	0.746	0.390	0.884	0.251	0.732	0.984	0.002

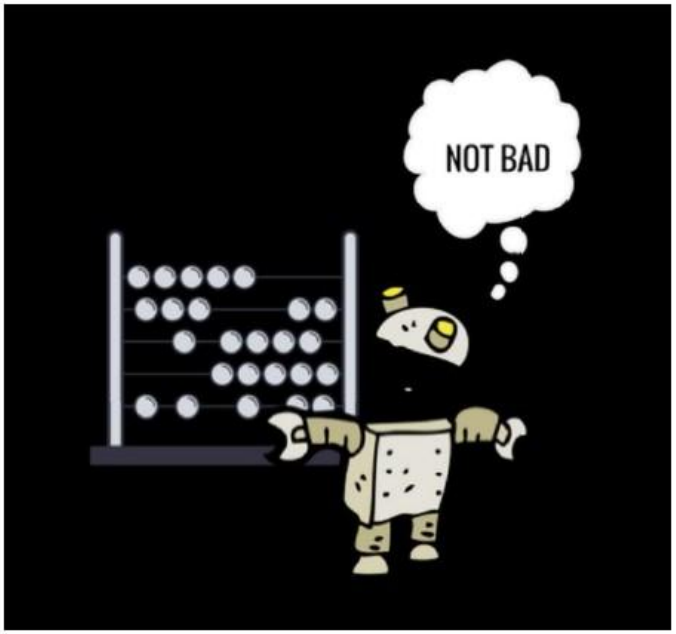
https://www.analyticsvidhya.com/blog/2021/09/a-comprehensive-guide-on-using-azure-machine-learning/

Machine Learning Model Training Conclusion

Pick the winning model by asking: Which model had the best performance on the test set? (performance)

Calculate performance metrics using those predictions and the "ground truth" target variable from the test set

For each of your models, make predictions on your test set



Pick the winning model by asking: Does it perform well across various performance metrics? (robustness)

Pick the winning model by asking: Did it also have (one of) the best cross-validated scores from the training set? (consistency)

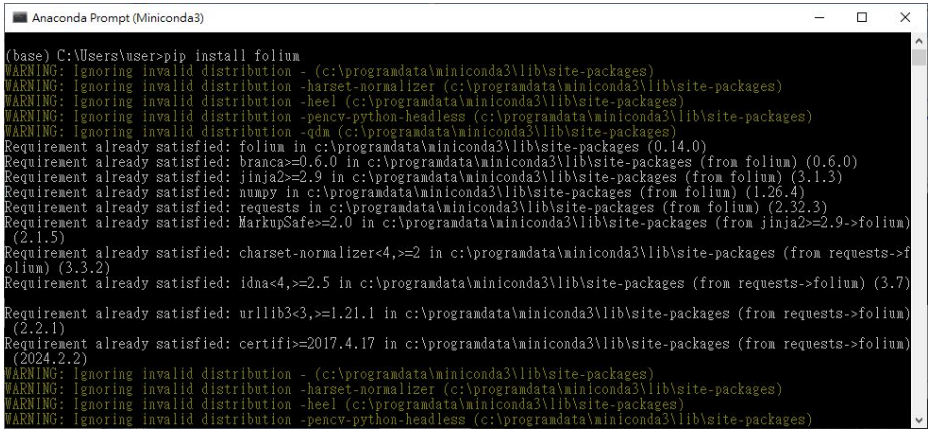
Pick the winning model by asking: Does it solve the original business problem? (win condition)

Source: https://elitedatascience.com/model-training

Obtaining Data – Web Scraping

1. Open tools “Anaconda Prompt” which will direct you to the command prompt.

2. Install python package requests by entering “pip install folium”

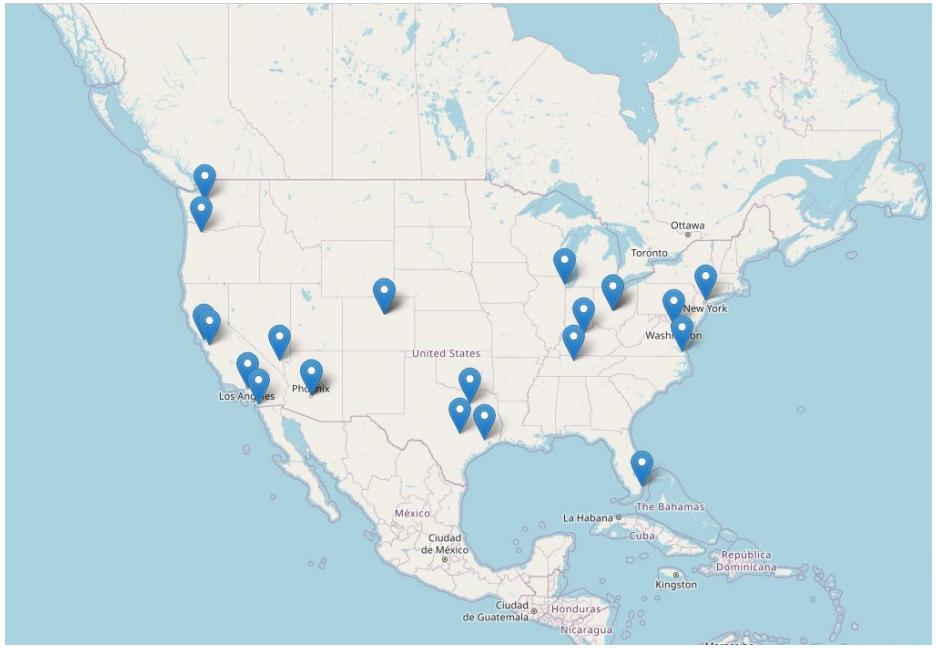


Data Visualizations

1. describe the contemporary trends of process automation and explain the opportunities and challenges of business process automation;
2. outline key steps in process automation project management and illustrate the significance of each step;
3. apply computational tools to implement process automation;
4. discuss the development of process automation and practical cases for business.

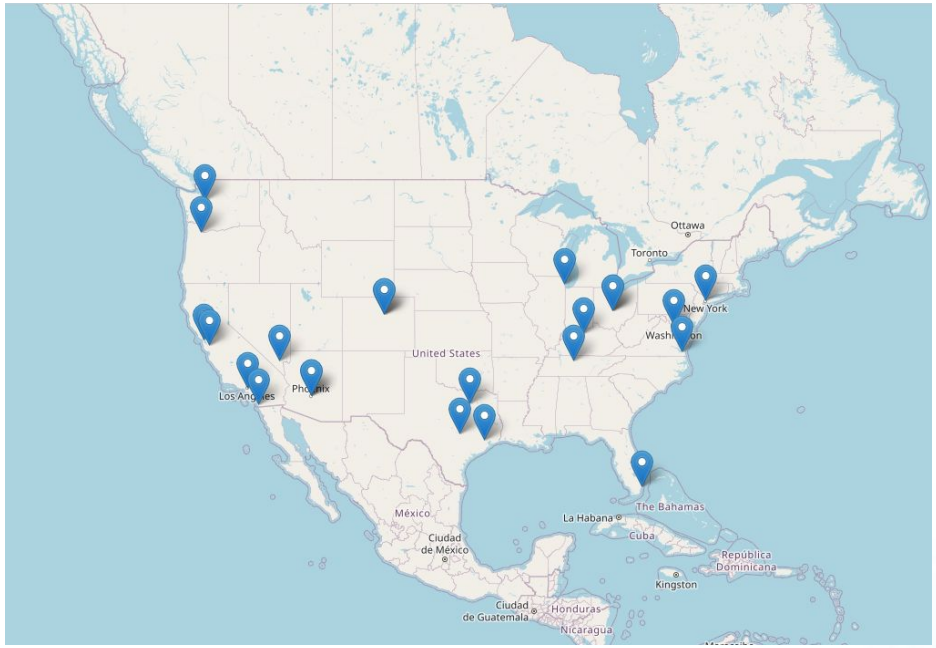
Python Google Map Tutorial

```
import folium

# List of 23 coordinates (latitude, longitude)
coordinates = [
    (37.7749, -122.4194),  # San Francisco, CA
    (34.0522, -118.2437),  # Los Angeles, CA
    (40.7128, -74.0060),   # New York, NY
    (41.8781, -87.6298),    # Chicago, IL
    (29.7604, -95.3698),    # Houston, TX
    (33.4484, -112.0740),   # Phoenix, AZ
    (39.7392, -104.9903),   # Denver, CO
    (47.6062, -122.3321),   # Seattle, WA
    (38.9072, -77.0369),    # Washington, D.C.
    (39.9612, -82.9988),    # Columbus, OH
    (30.2672, -97.7431),    # Austin, TX
    (32.7767, -96.7970),    # Dallas, TX
    (37.3382, -121.8863),   # San Jose, CA
    (39.7392, -104.9903),   # Denver, CO
    (36.1699, -115.1398),    # Las Vegas, NV
    (25.7617, -80.1918),    # Miami, FL
    (38.2527, -85.7585),    # Louisville, KY
    (39.9612, -82.9988),    # Columbus, OH
    (36.8508, -76.2859),    # Norfolk, VA
    (33.4484, -112.0740),   # Phoenix, AZ
    (45.5155, -122.6793),   # Portland, OR
    (36.1627, -86.7816),    # Nashville, TN
    (32.7157, -117.1611),   # San Diego, CA
]
```



Python Google Map Tutorial

```
# Create a map centered at the first coordinate
my_map = folium.Map(location=coordinates[0],
    zoom_start=4)

# Add markers for each coordinate
for lat, lon in coordinates:
    folium.Marker(location=(lat,
    lon)).add_to(my_map)

# Save the map to an HTML file
my_map.save("my_map.html")

# If you're using Jupyter Notebook, you can
    display the map inline
my_map
```



Exercise 1

Starting notebook:

https://github.com/innoviai/ipa_courses/blob/main/Lecture%203/python_data_visulisation_start.ipynb

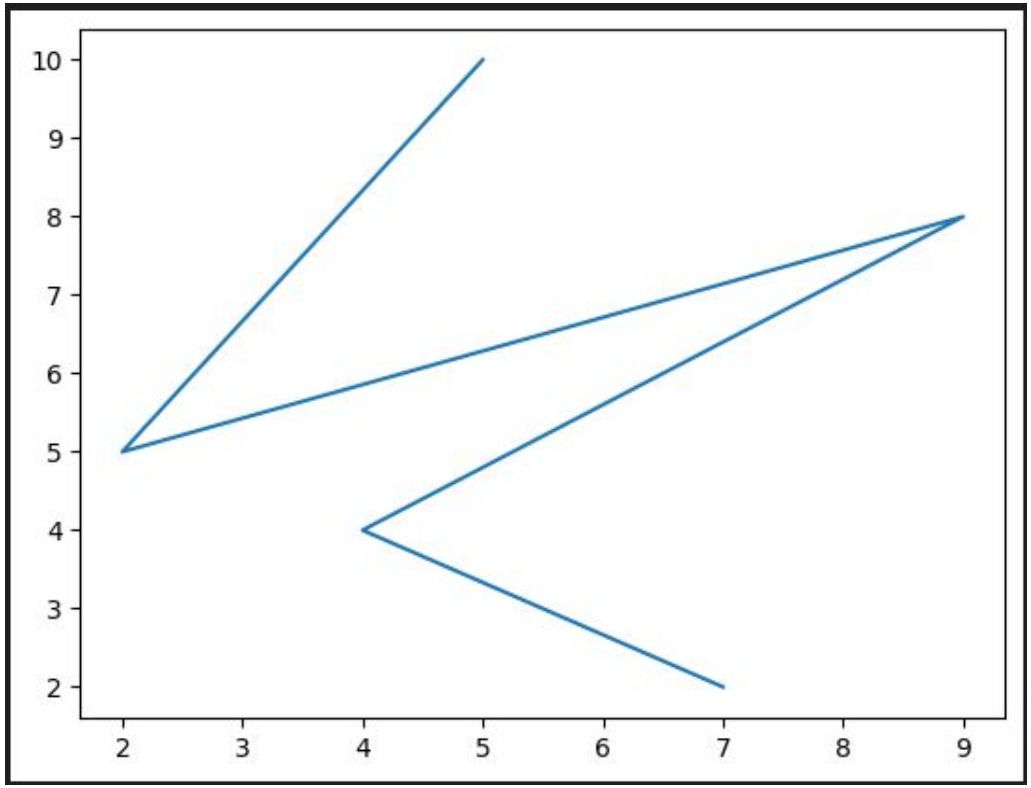
Step

1. *Open "Anaconda Prompt" and type "python"*
2. *Import library* "matplotlib" by entering code "import matplotlib"

Exercise 1

Step 2

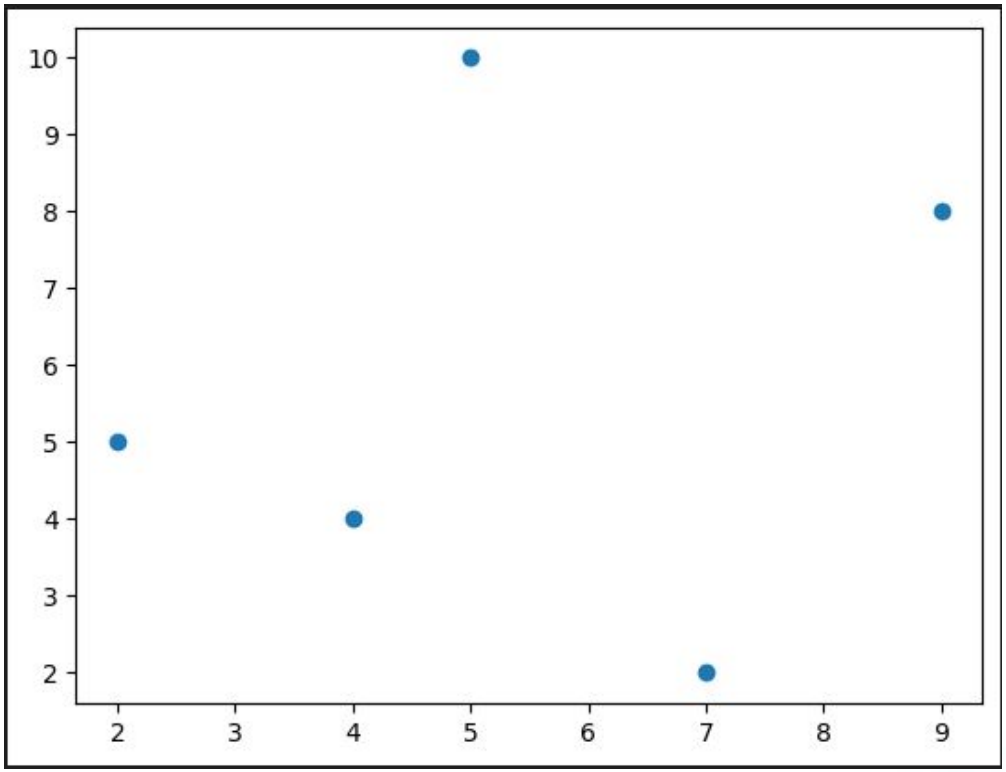
```
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot
plt.plot(x, y)

# function to show the plot
plt.show()
```



Exercise 1

Step 4

```
#scatter
# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot scatter
plt.scatter(x, y)

# function to show the plot
plt.show()
```



HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

THANK YOU