



# Introduction

---

Jiaze Li

February 11, 2026

The University of Hong Kong

# Overview

Installation

IDE

Python

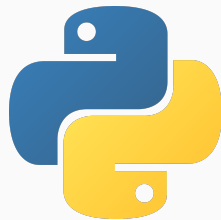
pandas

sklearn

# Installation

---

- In this course, we will use **Python** for textual analysis.
- Why?
  - Python has *rich libraries* for textual analysis.
  - Python has *clear syntax*.
  - *AI agents* are good at Python.



- In this course, we will use **Python** for textual analysis.
- Why?
  - Python has **rich libraries** for textual analysis.
  - Python has **clear syntax**.
  - **AI agents** are good at Python.



- In this course, we will use **Python** for textual analysis.
- Why?
  - Python has **rich libraries** for textual analysis.
  - Python has **clear syntax**.
  - **AI agents** are good at Python.



- In this course, we will use **Python** for textual analysis.
- Why?
  - Python has **rich libraries** for textual analysis.
  - Python has **clear syntax**.
  - **AI agents** are good at Python.



- In this course, we will use **Python** for textual analysis.
- Why?
  - Python has **rich libraries** for textual analysis.
  - Python has **clear syntax**.
  - **AI agents** are good at Python.





- The best way to install Python is **NOT** to install Python.
- The standard way to install Python is through `conda`, an open-source package and environment manager.
  - I recommend `conda-forge`, a community-led channel. See <https://conda-forge.org/>.
- If you need to use advanced modules like `torch`, consider installing Python through `uv`, a fast package manager. See <https://docs.astral.sh/uv/>.

- The best way to install Python is **NOT** to install Python.
- The standard way to install Python is through **conda**, an open-source package and environment manager.
  - I recommend **conda-forge**, a community-led channel. See <https://conda-forge.org/>.
- If you need to use advanced modules like **torch**, consider installing Python through **uv**, a fast package manager. See <https://docs.astral.sh/uv/>.



- The best way to install Python is **NOT** to install Python.
- The standard way to install Python is through **conda**, an open-source package and environment manager.
  - I recommend **conda-forge**, a community-led channel. See <https://conda-forge.org/>.
- If you need to use advanced modules like **torch**, consider installing Python through **uv**, a fast package manager. See <https://docs.astral.sh/uv/>.



# Installation

- The best way to install Python is **NOT** to install Python.
- The standard way to install Python is through **conda**, an open-source package and environment manager.
  - I recommend **conda-forge**, a community-led channel. See <https://conda-forge.org/>.
- If you need to use advanced modules like **torch**, consider installing Python through **uv**, a fast package manager. See <https://docs.astral.sh/uv/>.



- You can check if you have **conda-forge** installed correctly by running **conda --version** in your **Miniforge Prompt** (Windows) or **Terminal** (macOS/Linux).
  - For Windows users, if you want to use **conda** in Terminal, you should run **conda init** in Miniforge Prompt once.
- Your terminal should look like these:
  - (base) C:\Users\Jiaze Li>
  - (base) jiaze@jiaze-legion:~\$

- You can check if you have **conda-forge** installed correctly by running **conda --version** in your **Miniforge Prompt** (Windows) or **Terminal** (macOS/Linux).
  - For Windows users, if you want to use **conda** in Terminal, you should run **conda init** in Miniforge Prompt once.
- Your terminal should look like these:
  - (base) C:\Users\Jiaze Li>
  - (base) jiaze@jiaze-legion:~\$

# Environments

- **conda** is a package and environment manager.
  - Packages are collections of code that provide specific functionality.
    - Examples: `pandas`, `matplotlib`, etc.
  - Environments are isolated spaces where you can install packages without affecting other environments.
- To create a new environment, run `conda create -n <env_name> python=<version>` in your **Terminal**.
  - For example, `conda create -n py312 python=3.12` will create a new environment named `py312` with Python 3.12 installed.
- For many commands (not only `conda`), you may see **Proceed ([y]/n)?**. If you know what you are doing, type `y` and press **Enter**.

# Environments

- `conda` is a package and environment manager.
  - Packages are collections of code that provide specific functionality.
    - Examples: `pandas`, `matplotlib`, etc.
  - Environments are isolated spaces where you can install packages without affecting other environments.
- To create a new environment, run `conda create -n <env_name> python=<version>` in your **Terminal**.
  - For example, `conda create -n py312 python=3.12` will create a new environment named `py312` with Python 3.12 installed.
- For many commands (not only `conda`), you may see `Proceed ([y]/n)?`. If you know what you are doing, type `y` and press `Enter`.



# Environments

- **conda** is a package and environment manager.
  - Packages are collections of code that provide specific functionality.
    - Examples: **pandas**, **matplotlib**, etc.
  - Environments are isolated spaces where you can install packages without affecting other environments.
- To create a new environment, run `conda create -n <env_name> python=<version>` in your **Terminal**.
  - For example, `conda create -n py312 python=3.12` will create a new environment named `py312` with Python 3.12 installed.
- For many commands (not only **conda**), you may see **Proceed ([y]/n)?**. If you know what you are doing, type **y** and press **Enter**.

# Environments

- `conda` is a package and environment manager.
  - Packages are collections of code that provide specific functionality.
    - Examples: `pandas`, `matplotlib`, etc.
  - Environments are isolated spaces where you can install packages without affecting other environments.
- To create a new environment, run `conda create -n <env_name> python=<version>` in your **Terminal**.
  - For example, `conda create -n py312 python=3.12` will create a new environment named `py312` with Python 3.12 installed.
- For many commands (not only `conda`), you may see `Proceed ([y]/n)?`. If you know what you are doing, type `y` and press `Enter`.

# Environments

- **conda** is a package and environment manager.
  - Packages are collections of code that provide specific functionality.
    - Examples: **pandas**, **matplotlib**, etc.
  - Environments are isolated spaces where you can install packages without affecting other environments.
- To create a new environment, run **conda create -n <env\_name> python=<version>** in your **Terminal**.
  - For example, **conda create -n py312 python=3.12** will create a new environment named **py312** with Python 3.12 installed.
- For many commands (not only **conda**), you may see **Proceed ([y]/n)?**. If you know what you are doing, type **y** and press **Enter**.

## Environments (Cont.)

- To list all your environments, run `conda env list`.
  - **base** is the default environment that comes with **conda**. You should avoid installing packages in the **base** environment to prevent conflicts.
- To activate the environment, run `conda activate <env_name>`.
  - Every time you run commands in the terminal, you should make sure you are in the correct environment.
  - Once you activate the environment, your terminal should look like these:
    - (py312) C:\Users\Jiaze Li>
    - (py312) jiaze@jiaze-legion:~\$
- For more commands, run `conda --help` or check the official documentation at <https://docs.conda.io/>.

## Environments (Cont.)

- To list all your environments, run `conda env list`.
  - **base** is the default environment that comes with **conda**. You should avoid installing packages in the **base** environment to prevent conflicts.
- To activate the environment, run `conda activate <env_name>`.
  - Every time you run commands in the terminal, you should make sure you are in the correct environment.
  - Once you activate the environment, your terminal should look like these:
    - (py312) C:\Users\Jiaze Li>
    - (py312) jiaze@jiaze-legion:~\$
- For more commands, run `conda --help` or check the official documentation at <https://docs.conda.io/>.

## Environments (Cont.)

- To list all your environments, run `conda env list`.
  - **base** is the default environment that comes with **conda**. You should avoid installing packages in the **base** environment to prevent conflicts.
- To activate the environment, run `conda activate <env_name>`.
  - Every time you run commands in the terminal, you should make sure you are in the correct environment.
  - Once you activate the environment, your terminal should look like these:
    - (py312) C:\Users\Jiaze Li>
    - (py312) jiaze@jiaze-legion:~\$
- For more commands, run `conda --help` or check the official documentation at <https://docs.conda.io/>.

# Packages

- To install packages in the activated environment, run `conda install <package_name_1> <package_name_2> ....`
  - For example, `conda install jupyterlab pandas` will install the `jupyterlab` and `pandas` packages in the currently activated environment.
- Not all packages are available on `conda-forge`. You should check the official documentation of the package for installation instructions.
  - For example, `torch` no longer supports installation through `conda` since version 2.6.0.
- Recommended packages for general use:
  - `jupyterlab`: Add support for `.ipynb` files, which are interactive notebooks that allow you to run code and see the output in the same document.
  - `pandas`: A powerful library for data manipulation and analysis

# Packages

- To install packages in the activated environment, run `conda install <package_name_1> <package_name_2> ....`
  - For example, `conda install jupyterlab pandas` will install the `jupyterlab` and `pandas` packages in the currently activated environment.
- Not all packages are available on **conda-forge**. You should check the official documentation of the package for installation instructions.
  - For example, **torch** no longer supports installation through **conda** since version 2.6.0.
- Recommended packages for general use:
  - `jupyterlab`: Add support for `.ipynb` files, which are interactive notebooks that allow you to run code and see the output in the same document.
  - `pandas`: A powerful library for data manipulation and analysis



# Packages

- To install packages in the activated environment, run `conda install <package_name_1> <package_name_2> ....`
  - For example, `conda install jupyterlab pandas` will install the `jupyterlab` and `pandas` packages in the currently activated environment.
- Not all packages are available on **conda-forge**. You should check the official documentation of the package for installation instructions.
  - For example, **torch** no longer supports installation through **conda** since version 2.6.0.
- Recommended packages for general use:
  - **jupyterlab**: Add support for `.ipynb` files, which are interactive notebooks that allow you to run code and see the output in the same document.
  - **pandas**: A powerful library for data manipulation and analysis

IDE

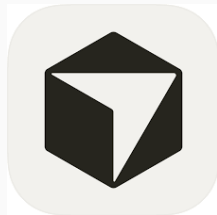
---

- In principle, you can write any code in any text editor like Notepad, but an Integrated Development Environment (IDE) can provide useful features like syntax highlighting, code completion, debugging, etc.
- I recommend using **Visual Studio Code**, an open-source code editor developed by Microsoft. See <https://code.visualstudio.com/>.
- You may also consider **Cursor**, a code editor based on VS Code that integrates AI features. See <https://www.cursor.com/>.
  - Caveat: Although Cursor is forked from VS Code, some extensions like Data Wrangler may not work properly in Cursor.

- In principle, you can write any code in any text editor like Notepad, but an Integrated Development Environment (IDE) can provide useful features like syntax highlighting, code completion, debugging, etc.
- I recommend using **Visual Studio Code**, an open-source code editor developed by Microsoft. See <https://code.visualstudio.com/>.
- You may also consider **Cursor**, a code editor based on VS Code that integrates AI features. See <https://www.cursor.com/>.
  - Caveat: Although Cursor is forked from VS Code, some extensions like Data Wrangler may not work properly in Cursor.



- In principle, you can write any code in any text editor like Notepad, but an Integrated Development Environment (IDE) can provide useful features like syntax highlighting, code completion, debugging, etc.
- I recommend using **Visual Studio Code**, an open-source code editor developed by Microsoft. See <https://code.visualstudio.com/>.
- You may also consider **Cursor**, a code editor based on VS Code that integrates AI features. See <https://www.cursor.com/>.
  - Caveat: Although Cursor is forked from VS Code, some extensions like Data Wrangler may not work properly in Cursor.



# Extensions

- Extensions for Python development:
  - **Python**: Provides rich support for Python development. See <https://marketplace.visualstudio.com/items?itemName=ms-python.python>.
  - **Jupyter**: Provides support for Jupyter notebooks. See <https://marketplace.visualstudio.com/items?itemName=ms-toolsai.jupyter>.
- Recommended extensions for Python development:
  - **GitHub Copilot Chat**: Provides AI-powered code suggestions and explanations. See <https://marketplace.visualstudio.com/items?itemName=GitHub.copilot-chat>.
  - **Data Wrangler**: Provides a visual interface for data manipulation and analysis. See <https://marketplace.visualstudio.com/items?itemName=ms-toolsai.datawrangler>.

# Extensions

- Extensions for Python development:
  - **Python**: Provides rich support for Python development. See <https://marketplace.visualstudio.com/items?itemName=ms-python.python>.
  - **Jupyter**: Provides support for Jupyter notebooks. See <https://marketplace.visualstudio.com/items?itemName=ms-toolsai.jupyter>.
- Recommended extensions for Python development:
  - **GitHub Copilot Chat**: Provides AI-powered code suggestions and explanations. See <https://marketplace.visualstudio.com/items?itemName=GitHub.copilot-chat>.
  - **Data Wrangler**: Provides a visual interface for data manipulation and analysis. See <https://marketplace.visualstudio.com/items?itemName=ms-toolsai.datawrangler>.

File Edit Selection View Go Run Terminal Help

test.ipynb M

test.ipynb > import pandas as pd

Generate + Code + Markdown Run All Restart

hku-econ6087 [Python 3.14.2]

```
from datasets import load_dataset
dataset = load_dataset("ag_news")
```

[1] 5.8s Python

```
for split in dataset.keys():
    dataset[split].to_csv(f"{split}.csv")
```

[2] 1.4s Python

Creating CSV from Arrow format: 100% 120/120 [00:01<00:00, 98.66ba/s]

Creating CSV from Arrow format: 100% 8/8 [00:00<00:00, 68.26ba/s]

Generate + Code + Markdown

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

# Load the datasets
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")

print("Train shape:", train_df.shape)
print("Test shape:", test_df.shape)
print(train_df.head())
```

[3] 23s Python

Train shape: (120000, 2)  
Test shape: (7600, 2)

	text	label
0	Wall St. Bears Claw Back Into the Black (Reute...	2
1	Carlyle Looks Toward Commercial Aerospace (Reu...	2
2	Oil and Economy Cloud Stocks' Outlook (Reuters...	2
3	Iraq Halts Oil Exports from Main Southern Pipe...	2
4	Oil prices soar to all-time record, posing new...	2

PROBLEMS 10 OUTPUT TERMINAL JUPYTER GITLENS PORTS 0 DEBUG CONSOLE

```
● jiaze@jiaze-dell:~/hku-econ6087$ source /home/jiaze/hku-econ6087/.venv/bin/activate
○ (hku-econ6087) jiaze@jiaze-dell:~/hku-econ6087$
```

train\_df [DW]

Export as file Refresh data Report an issue 120000 rows x 2 columns Go to column Viewing

text # label

Missing: 0 (0%) Missing: 0 (0%)

Distinct: 120000 (100%) Distinct: 4 (<1%)

120000 Distinct values

Min 0 Max 3

0	Wall St. Bears Claw Back Into the Bla	2
1	Carlyle Looks Toward Commercial Ae	2
2	Oil and Economy Cloud Stocks' Outl	2
3	Iraq Halts Oil Exports from Main Sou	2
4	Oil prices soar to all-time record, poi	2
5	Stocks End Up, But Near Year Lows ()	2
6	Money Funds Fell in Latest Week (AF	2
7	Fed minutes show dissent over inflat	2
8	Safety Net (Forbes.com) Forbes.com	2
9	Wall St. Bears Claw Back Into the Bla	2
10	Oil and Economy Cloud Stocks' Outl	2
11	No Need for OPEC to Pump More-Ir	2
12	Non-OPEC Nations Should Up Outp	2
13	Google IPO Auction Off to Rocky Sta	2
14	Dollar falls broadly on Record Trade	2
15	Rescuing an Old Saver If you think y	2
16	Kids Rule for Back-to-School The pu	2
17	In a Down Market, Head Toward Val	2
18	US trade deficit swells in June The U	2
19	Shell 'could be target for Total' Oil gi	2
20	Google IPO faces Playboy slip-up Th	2
21	Eurozone economy keeps growing C	2
22	Expansion slows in Japan Economic i	2
23	Rand falls on shock SA rate cut Inter	2
24	Car prices down across the board Th	2
25	South Korea lowers interest rates So	2
26	Google auction begins on Friday An	2

bash - hku-econ6087 + - - - -

You, 57 seconds ago Spaces 4 LF {} Cell 3 of 12



# Python

---

# Running Your First Code

- Create a new file and save it with the `.ipynb` extension (Jupyter Notebook).
- At the upper-right corner of the VS Code editor, click "Select Kernel" and choose the conda environment we created.
- Type the following in a code cell and run it.

```
print("Hello, World!")
```

# Running Your First Code

- Create a new file and save it with the `.ipynb` extension (Jupyter Notebook).
- At the upper-right corner of the VS Code editor, click "Select Kernel" and choose the `conda` environment we created.
- Type the following in a code cell and run it.

```
print("Hello, World!")
```

# Running Your First Code

- Create a new file and save it with the `.ipynb` extension (Jupyter Notebook).
- At the upper-right corner of the VS Code editor, click "Select Kernel" and choose the `conda` environment we created.
- Type the following in a code cell and run it.

```
print("Hello, World!")
```

# Function

- The `print()` function is technically defined as follows:

```
def print(  
    *values: object,  
    sep: str | None = " ",  
    end: str | None = "\n",  
    file: None = None,  
    flush: bool = False  
) -> None: ...
```

- You can move your mouse cursor over the function name to see its definition.

# Function

- The `print()` function is technically defined as follows:

```
def print(  
    *values: object,  
    sep: str | None = " ",  
    end: str | None = "\n",  
    file: None = None,  
    flush: bool = False  
) -> None: ...
```

- You can move your mouse cursor over the function name to see its definition.

# Type

- Every value is an object, and every object has a **type**.
- You can check the type of an object using the `type()` function.

```
type("Hello, World!")    # str
type(42)                  # int
type(3.14)                # float
type(True)               # bool
type([1, 2, 3])           # list
type((1, 2, 3))           # tuple
type({"a": 1, "b": 2})    # dict
```

- This is useful when you see `TypeError: ...` in your code, which means you are using a value of the wrong type.

# Type

- Every value is an object, and every object has a **type**.
- You can check the type of an object using the **type()** function.

```
type("Hello, World!")    # str
type(42)                  # int
type(3.14)                # float
type(True)               # bool
type([1, 2, 3])           # list
type((1, 2, 3))           # tuple
type({"a": 1, "b": 2})    # dict
```

- This is useful when you see **TypeError: ...** in your code, which means you are using a value of the wrong type.



# Type

- Every value is an object, and every object has a **type**.
- You can check the type of an object using the **type()** function.

```
type("Hello, World!")    # str
type(42)                  # int
type(3.14)                # float
type(True)               # bool
type([1, 2, 3])           # list
type((1, 2, 3))           # tuple
type({"a": 1, "b": 2})    # dict
```

- This is useful when you see **TypeError: ...** in your code, which means you are using a value of the wrong type.

pandas

---

- **pandas** is a powerful package for data manipulation and analysis.
- It introduces two classes that can hold any type.
  - **Series**, a one-dimensional labeled array (Column)
  - **DataFrame**, a two-dimensional labeled data structure (Table)
- pandas is nothing but a combination of functions written by the contributors.

- `pandas` is a powerful package for data manipulation and analysis.
- It introduces two classes that can hold any type.
  - `Series`, a one-dimensional labeled array (Column)
  - `DataFrame`, a two-dimensional labeled data structure (Table)
- `pandas` is nothing but a combination of functions written by the contributors.

- `pandas` is a powerful package for data manipulation and analysis.
- It introduces two classes that can hold any type.
  - `Series`, a one-dimensional labeled array (Column)
  - `DataFrame`, a two-dimensional labeled data structure (Table)
- `pandas` is nothing but a combination of functions written by the contributors.

pandas-dev / pandas

Code

Issues 3.4k

Pull requests 207

Agents

Actions

Projects

Security

Insights

Files

main

Go to file

github

LICENSES

asv\_bench

ci

doc

pandas

config

libs

testing

api

arrays

compat

core

errors

io

clipboard

excel

formats

json

parsers

sas

\_io.py

\_util.py

api.py

clipboard.py

common.py

feather\_format.py

html.py

iceberg.py

orc.py

parquet.py

pickle.py

pytables.py

spss.py

sql.py

stata.py

xml.py

parquet / pandas / io / parquet.py

schyberg: DOC: Replace @doc decorator with inline docstrings in pandas/io/parq... 249k43d · 2 months ago History

Code

Blame

488 lines (599 loc) · 23.9 KB

Raw

```
1  """parquet compat"""
2
3  from __future__ import annotations
4
5  import io
6  import json
7  import os
8  from typing import (
9      TYPE_CHECKING,
10     Any,
11     Literal,
12 )
13
14 from warnings import (
15     catch_warnings,
16     filterwarnings,
17 )
18
19 from pandas._libs import lib
20 from pandas.compat.optional import import_optional_dependency
21 from pandas.errors import (
22     AbstractMethodError,
23     PandasWarning,
24 )
25 from pandas.util._decorators import set_module
26 from pandas.util._validators import check_dtype_backend
27
28 from pandas import (
29     DataFrame,
30     get_option,
31 )
32
33 from pandas.io._util import arrow_table_to_pandas
34 from pandas.io.common import (
35     _handleiles,
36     get_pandas,
37     is_fsspec_url,
38     is_url,
39     stringify_path,
40 )
41
42 if TYPE_CHECKING:
43     from pandas._typing import (
44         DtypeBackend,
45         FioPaths,
46         ParquetCompressionOptions,
47         ReadBuffer,
48         StorageOptions,
49         WriteBuffer,
50     )
51
52 def get_engine(engine: str) -> BaseDml:
53     """Return our implementation"""
54     if engine == "auto":
55         engine = get_option("io.parquet.engine")
56     if engine == "auto":
57         # try engines in this order
```

15/26

## pandas (Cont.)

- To use any package, you need to import it first.

```
import pandas as pd
```

- This line imports the pandas package and names it `pd` for short.
- You can now use the functions in pandas by `pd.function_name()`.
  - For example, `pd.DataFrame()` is a function that returns a new DataFrame object.

## pandas (Cont.)

- To use any package, you need to import it first.

```
import pandas as pd
```

- This line imports the **pandas** package and names it **pd** for short.
- You can now use the functions in pandas by `pd.function_name()`.
  - For example, `pd.DataFrame()` is a function that returns a new DataFrame object.



## pandas (Cont.)

- To use any package, you need to import it first.

```
import pandas as pd
```

- This line imports the **pandas** package and names it **pd** for short.
- You can now use the functions in **pandas** by **pd.function\_name()**.
  - For example, **pd.DataFrame()** is a function that returns a new DataFrame object.

## pandas (Cont.)

- An object also has **attributes** (values) and **methods** (functions).
- You can check them using the `dir()` function.

```
df = pd.DataFrame({  
    'A': [1, 2, 3],  
    'B': [4, 5, 6]  
})  
dir(df)
```

- To access an attribute or method, use the dot notation on the object.

```
df.columns    # attribute  
df.head()     # method
```

## pandas (Cont.)

- An object also has **attributes** (values) and **methods** (functions).
- You can check them using the `dir()` function.

```
df = pd.DataFrame({  
    'A': [1, 2, 3],  
    'B': [4, 5, 6]  
})  
dir(df)
```

- To access an attribute or method, use the dot notation on the object.

```
df.columns    # attribute  
df.head()     # method
```

## pandas (Cont.)

- An object also has **attributes** (values) and **methods** (functions).
- You can check them using the **dir()** function.

```
df = pd.DataFrame({  
    'A': [1, 2, 3],  
    'B': [4, 5, 6]  
})  
dir(df)
```

- To access an attribute or method, use the dot notation on the object.

```
df.columns    # attribute  
df.head()     # method
```

## pandas (Cont.)

- Many packages including **pandas** have a gigantic number of functions. You don't need to know all of them.
- For all packages, you can learn via the following channels:
  - Official documentation (<https://pandas.pydata.org/docs/> for pandas)
  - AI agents
    - You need to be extremely careful when using AI agents to learn packages. The core reason is that many AI agents are trained on the **previous** version of a package, which can be very different from the current version.
    - For example, AI agents used to suggest `pd.DataFrame.append()` for adding a new row, but this method has become deprecated since version 1.4.0.
    - **pandas** just released version 3.0.0 on January 21, 2026, which has many breaking changes including introducing `pd.col()` syntax.
  - Online courses and tutorials

## pandas (Cont.)

- Many packages including **pandas** have a gigantic number of functions. You don't need to know all of them.
- For all packages, you can learn via the following channels:
  - Official documentation (<https://pandas.pydata.org/docs/> for **pandas**)
  - AI agents
    - You need to be extremely careful when using AI agents to learn packages. The core reason is that many AI agents are trained on the **previous** version of a package, which can be very different from the current version.
    - For example, AI agents used to suggest `pd.DataFrame.append()` for adding a new row, but this method has become deprecated since version 1.4.0.
    - **pandas** just released version 3.0.0 on January 21, 2026, which has many breaking changes including introducing `pd.col()` syntax.
  - Online courses and tutorials

sklearn

---

- **sklearn** is a powerful package for machine learning.
- To install **sklearn**, run `conda install scikit-learn`.
- It provides many functions for data preprocessing, model training, evaluation, etc.
- For the first assignment, we will use the `CountVectorizer` (BoW), `TfidfVectorizer` (TF-IDF), and `KNeighborsClassifier` (KNN) functions in **sklearn**.



- `sklearn` is a powerful package for machine learning.
- To install `sklearn`, run `conda install scikit-learn`.
- It provides many functions for data preprocessing, model training, evaluation, etc.
- For the first assignment, we will use the `CountVectorizer` (BoW), `TfidfVectorizer` (TF-IDF), and `KNeighborsClassifier` (KNN) functions in `sklearn`.

- `sklearn` is a powerful package for machine learning.
- To install `sklearn`, run `conda install scikit-learn`.
- It provides many functions for data preprocessing, model training, evaluation, etc.
- For the first assignment, we will use the `CountVectorizer` (BoW), `TfidfVectorizer` (TF-IDF), and `KNeighborsClassifier` (KNN) functions in `sklearn`.

- `sklearn` is a powerful package for machine learning.
- To install `sklearn`, run `conda install scikit-learn`.
- It provides many functions for data preprocessing, model training, evaluation, etc.
- For the first assignment, we will use the `CountVectorizer` (BoW), `TfidfVectorizer` (TF-IDF), and `KNeighborsClassifier` (KNN) functions in `sklearn`.

# CountVectorizer

- Whenever you see a new function, you should first check the [official documentation](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html).  
`https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`
- Some packages also have a more detailed [user guide](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html).  
`https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`
- `CountVectorizer` is a class with the following parameters, attributes, and methods.
  - `from sklearn.feature_extraction.text import CountVectorizer`
  - Parameters: `tokenizer`, `stop_words`, `token_pattern`
  - Methods: `fit`, `transform`, `fit_transform`

# CountVectorizer

- Whenever you see a new function, you should first check the [official documentation](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html).  
`https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`
- Some packages also have a more detailed [user guide](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html).  
`https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`
- `CountVectorizer` is a class with the following parameters, attributes, and methods.
  - `from sklearn.feature_extraction.text import CountVectorizer`
  - Parameters: `tokenizer`, `stop_words`, `token_pattern`
  - Methods: `fit`, `transform`, `fit_transform`

# CountVectorizer

- Whenever you see a new function, you should first check the [official documentation](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html).  
`https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`
- Some packages also have a more detailed [user guide](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html).  
`https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`
- **CountVectorizer** is a class with the following parameters, attributes, and methods.
  - `from sklearn.feature_extraction.text import CountVectorizer`
  - Parameters: `tokenizer`, `stop_words`, `token_pattern`
  - Methods: `fit`, `transform`, `fit_transform`

# tokenizer

- The **tokenizer** parameter is a function that takes a string as input and returns a list of tokens (words).
- If you don't specify a **tokenizer**, **CountVectorizer** will use a default tokenizer that splits the string into words based on whitespace and punctuation.
  - For example, the string "Hello, World!" will be tokenized into ["Hello", "World"].
- You can also specify a custom **tokenizer** if you want to use a different tokenization method.
  - For example, you can use **jieba** for Chinese tokenization. Then, the string "你好，世界！" will be tokenized into ["你好", " ", " ", "世界", "!", " "].

# tokenizer

- The **tokenizer** parameter is a function that takes a string as input and returns a list of tokens (words).
- If you don't specify a **tokenizer**, **CountVectorizer** will use a default tokenizer that splits the string into words based on whitespace and punctuation.
  - For example, the string "Hello, World!" will be tokenized into ["Hello", "World"].
- You can also specify a custom **tokenizer** if you want to use a different tokenization method.
  - For example, you can use **jieba** for Chinese tokenization. Then, the string "你好，世界！" will be tokenized into ["你好", " ", " ", "世界", "!", " "].



# tokenizer

- The **tokenizer** parameter is a function that takes a string as input and returns a list of tokens (words).
- If you don't specify a **tokenizer**, **CountVectorizer** will use a default tokenizer that splits the string into words based on whitespace and punctuation.
  - For example, the string "Hello, World!" will be tokenized into ["Hello", "World"].
- You can also specify a custom **tokenizer** if you want to use a different tokenization method.
  - For example, you can use **jieba** for Chinese tokenization. Then, the string "你好，世界！" will be tokenized into ["你好", " ", " ", "世界", "!", " "].

## stop\_words

- The **stop\_words** parameter is a list of words that will be ignored during tokenization.
- If you set `stop_words="english"`, `CountVectorizer` will use a built-in list of common English stop words (e.g., "the", "is", "in", etc.).
- You can also specify your own **list** of stop words if you want to ignore different words.
  - For example, you can use a custom list of stop words for Chinese, including "综上所述", "总的来看", "总的来说". You may refer to <https://github.com/goto456/stopwords/tree/master>.

## stop\_words

- The `stop_words` parameter is a list of words that will be ignored during tokenization.
- If you set `stop_words="english"`, `CountVectorizer` will use a built-in list of common English stop words (e.g., "the", "is", "in", etc.).
- You can also specify your own **list** of stop words if you want to ignore different words.
  - For example, you can use a custom list of stop words for Chinese, including "综上所述", "总的来看", "总的来说". You may refer to <https://github.com/goto456/stopwords/tree/master>.

## stop\_words

- The `stop_words` parameter is a list of words that will be ignored during tokenization.
- If you set `stop_words="english"`, `CountVectorizer` will use a built-in list of common English stop words (e.g., "the", "is", "in", etc.).
- You can also specify your own **list** of stop words if you want to ignore different words.
  - For example, you can use a custom list of stop words for Chinese, including "综上所述", "总的来看", "总的来说". You may refer to <https://github.com/goto456/stopwords/tree/master>.

## token\_pattern

- The `token_pattern` parameter is a regular expression that defines the pattern for tokenization.
- The default value is `r'(?u)\b\w\w+\b'`, which means that tokens must be at least 2 characters long and consist of word characters (letters, digits, or underscores).
- You can change the regular expression to include different types of tokens.
  - For example, if you want to only include alphabetic tokens, you can set `token_pattern=r'[a-zA-Z]+'`. Then, the string "Hello, World! 123" will be tokenized into ["Hello", "World"].
  - If you want to include Chinese characters, you can set `token_pattern=r'[\u4e00-\u9fff]+'`. Then, the string "你好，世界！" will be tokenized into ["你好", "世界"].

## token\_pattern

- The `token_pattern` parameter is a regular expression that defines the pattern for tokenization.
- The default value is `r'(?u)\b\w\w+\b'`, which means that tokens must be at least 2 characters long and consist of word characters (letters, digits, or underscores).
- You can change the regular expression to include different types of tokens.
  - For example, if you want to only include alphabetic tokens, you can set `token_pattern=r'[a-zA-Z]+'`. Then, the string "Hello, World! 123" will be tokenized into ["Hello", "World"].
  - If you want to include Chinese characters, you can set `token_pattern=r'[\u4e00-\u9fff]+'`. Then, the string "你好，世界！" will be tokenized into ["你好", "世界"].

## token\_pattern

- The `token_pattern` parameter is a regular expression that defines the pattern for tokenization.
- The default value is `r'(?u)\b\w\w+\b'`, which means that tokens must be at least 2 characters long and consist of word characters (letters, digits, or underscores).
- You can change the regular expression to include different types of tokens.
  - For example, if you want to only include alphabetic tokens, you can set `token_pattern=r'[a-zA-Z]+'`. Then, the string "Hello, World! 123" will be tokenized into ["Hello", "World"].
  - If you want to include Chinese characters, you can set `token_pattern=r'[\u4e00-\u9fff]+'`. Then, the string "你好，世界！" will be tokenized into ["你好", "世界"].

## `fit`, `transform`, `fit_transform`

- The `fit` method learns the vocabulary from the input data.
- The `transform` method transforms the input data into a document-term matrix based on the learned vocabulary.
- The `fit_transform` method is a combination of `fit` and `transform`. It learns the vocabulary and transforms the input data in one step.
- Since we train the model on the training data and apply the model to the test data, we should use `fit_transform` on the training data and `transform` on the test data to avoid data leakage.



## `fit`, `transform`, `fit_transform`

- The `fit` method learns the vocabulary from the input data.
- The `transform` method transforms the input data into a document-term matrix based on the learned vocabulary.
- The `fit_transform` method is a combination of `fit` and `transform`. It learns the vocabulary and transforms the input data in one step.
- Since we train the model on the training data and apply the model to the test data, we should use `fit_transform` on the training data and `transform` on the test data to avoid data leakage.

## `fit`, `transform`, `fit_transform`

- The `fit` method learns the vocabulary from the input data.
- The `transform` method transforms the input data into a document-term matrix based on the learned vocabulary.
- The `fit_transform` method is a combination of `fit` and `transform`. It learns the vocabulary and transforms the input data in one step.
- Since we train the model on the training data and apply the model to the test data, we should use `fit_transform` on the training data and `transform` on the test data to avoid data leakage.

## `fit`, `transform`, `fit_transform`

- The `fit` method learns the vocabulary from the input data.
- The `transform` method transforms the input data into a document-term matrix based on the learned vocabulary.
- The `fit_transform` method is a combination of `fit` and `transform`. It learns the vocabulary and transforms the input data in one step.
- Since we train the model on the training data and apply the model to the test data, we should use `fit_transform` on the training data and `transform` on the test data to avoid data leakage.

## Further Reading

- For more details on `TfidfVectorizer`, please refer to [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_extraction.text.TfidfVectorizer.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) and [https://scikit-learn.org/stable/modules/feature\\_extraction.html#tfidf](https://scikit-learn.org/stable/modules/feature_extraction.html#tfidf).
- For more details on `KNeighborsClassifier`, please refer <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html> and <https://scikit-learn.org/stable/modules/neighbors.html#classification>.

## Further Reading

- For more details on `TfidfVectorizer`, please refer to [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_extraction.text.TfidfVectorizer.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) and [https://scikit-learn.org/stable/modules/feature\\_extraction.html#tfidf](https://scikit-learn.org/stable/modules/feature_extraction.html#tfidf).
- For more details on `KNeighborsClassifier`, please refer <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html> and <https://scikit-learn.org/stable/modules/neighbors.html#classification>.

- Some of the students have already submitted the first assignment with the help of AI agents.
- My test on Github Copilot using the following prompt with Claude Opus 4.6:
  - “I am working in a uv project and have already run `uv add datasets pandas scikit-learn`. Based on the first question in `#file:assignment_1.tex`, generate Python code for a Jupyter Notebook. The code must load the `ag_news` dataset from Hugging Face using the `datasets` library. Provide the solution logic for the question using `pandas` for data manipulation.”

- Some of the students have already submitted the first assignment with the help of AI agents.
- My test on Github Copilot using the following prompt with Claude Opus 4.6:
  - “I am working in a uv project and have already run `uv add datasets pandas scikit-learn`. Based on the first question in `#file:assignment_1.tex`, generate Python code for a Jupyter Notebook. The code must load the `ag_news` dataset from Hugging Face using the `datasets` library. Provide the solution logic for the question using `pandas` for data manipulation.”